



STEM SEALS

LAND Lectionary



NORTH FLORIDA
COLLEGE

A Student Guide to the LAND Challenge

STEM SEALS LAND Lectionary

A Student Guide to the LAND Challenge

Copyright © 2023 STEM SEALs

All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in reviews and certain other non-commercial uses permitted by copyright law.

To request permission contact at stemseals@gmail.com.

Authors: Dr. Guenter Maresch and Lura L. Murfee

ISBN: 979-8-9874652-2-6

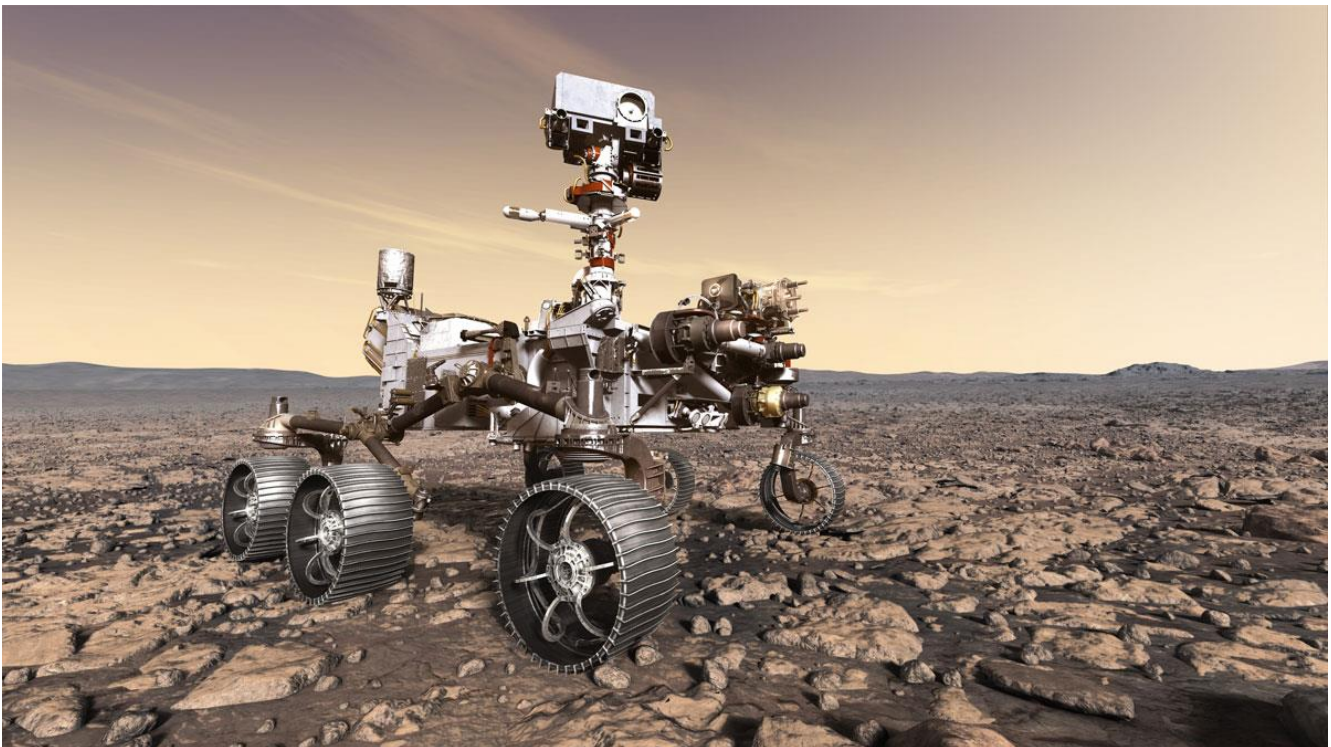
Printed by North Florida College

North Florida College
325 NW Turner Davis Drive
Madison, Florida 32340
United States of America

Synopsis

The STEM SEALs LAND Challenge was developed to familiarize middle school students with the basic concepts of a robot. During the course of a six-day camp, the students build a rover consisting of a micro:bit microcomputer, a motor:bit or motor driver, power supply, DC motors, wheels, actuators (servo motors), and sensors such as ultrasonic sensors.

The students learn about the essential parts of robotic devices, write code for the computer, test and improve their code for certain specific missions. Finally, they compete with their own rovers against each other demonstrating their knowledge and skills and their accomplishments during the preparation and practice phases.



Mars Rover: Perseverance (<https://mars.nasa.gov/mars2020/>)

Table of Contents

Preface – The STEM SEALs Project	8
Module 1: The LAND Challenge	11
1.1 Introduction to Robots	11
1.2 What is the LAND Challenge?	13
1.3 Unpacking the LAND Challenge Kit	14
Module 2: What Does a Microcomputer Do?	21
2.1 The BBC Micro:bit	21
2.2 Let’s Explore the Micro:bit	21
2.3 Accessing and Using the MakeCode Editor	23
2.4 Writing Code	25
2.5 Sharing or Publishing Your Code	31
Module 3: Micro:bit Code	32
3.1 How Does Code Work?	32
3.2 Interruptions in Code	32
3.3 The Essential MakeCode Blocks	40
3.4 Advanced MakeCode Blocks	40
3.5 MakeCode Java Script	41
Module 4: Testing Blood Samples	42
4.1 Why are we testing blood samples?	42
4.2 What are we trying to determine with the blood samples?	44
4.3 What will we use to test blood types?	44
4.4 Blood Type Testing	45
Module 5: Rover Assembly	49
5.1 Motor Test	49
5.2 The Motor Driver	50
5.3 Rover Assembly	52
Module 6: Propulsion	53
6.1 Propulsion Test	54
6.2 Propulsion 2	58
6.3 Understanding the Code	59
6.4 Propulsion 3	61
6.5 Propulsion 4	69

6.6 Propulsion Test (Speed and Distance).....	74
Module 7: Steering with a Remote Control.....	86
7.1 Adding a Remote Control (Propulsion 5 / Rover RC 1).....	86
7.2 Propulsion 6 and Rover RC 2	94
7.3 Rover Remote Control Practice	97
Module 8: Actuators and Sensors	98
8.1 What are Servo Motors?	98
8.2 Cargo Bed Assembly	99
8.3 Cargo Bed Calibration (Cargo Bed Test 1)	100
8.4 Remote-Controlled Cargo Bed (Cargo Bed Test 1 & 2 / Rover RC3 & 4).....	102
8.5 What are Ultrasonic Sensors?.....	114
8.6 Sonar Assembly	115
8.7 Sonar Calibration (Sonar Test 1)	116
8.8 Improving the Code with Functions (Sonar Test 2)	119
8.9 Testing the Sonar (Sonar Test 3 & 4, Rover RC 5).....	125
8.10 Sonar Calibration Test	148
Module 9: Navigation - Autonomous Driving	152
9.1 Writing an Autonomous Program (Navigation Test 1)	152
9.2 Navigation Test 2.....	159
9.3 Navigation Test 3.....	164
9.4 Navigation Test 4.....	176
9.5 Navigation Test 5.....	180
9.6 Remote-Controlled Autonomous Navigation (RC 6 & Nav. Test 6)	184
9.7 Navigation 7 and Rover RC 7.....	207
Module 10: Competition Practice.....	235
Module 11: The Competition	236
Rover Specifics:.....	238
References:	239

Preface – The STEM SEALs Project

The STEM SEALs project began as a three-year program at North Florida College as a research effort for the improvement of learning and teaching in STEM (Science, Technology, Engineering and Mathematics) fields in rural environments. The core motivation was the observation by faculty that the majority but especially students from underprivileged families show a lack of familiarity or even basic exposure to scientific and technological elements, and therefore have major difficulties to adjust to the demands in their secondary educational phase.

The goal was to produce a program exploiting the attractiveness of modern devices such as robots, microcomputers, and unmanned aerial vehicles (UAVs or drones) and involving middle school students in the design of sea, air and land vehicles (hence the “SEALs” term borrowed from the United States Navy SEALs), their construction, testing, improvement, and finally using the vehicles during a competition event. (image) “*SEAL Trident, the special warfare insignia of the U.S. Navy SEALs*”



It was clear from the very beginning that college faculty are not experts in teaching middle school students. The STEM SEALs project involved middle school teachers in all three crucial phases of the program, during the design, review, and activity phase. Groups of teachers formed together with college faculty and held Design Team and Review Team workshops long before the final program for the students’ activities was finalized. Then, during the STEM SEALs Summer Institute, students were led through the activities again, together, by groups of middle school teachers and college faculty.

During all phases a direct link was maintained between students, teachers, and faculty to the Research Team, which acquired data about each event. The Research Team analyzed the data and fed the results back to college faculty for further guidance for improving the next steps.

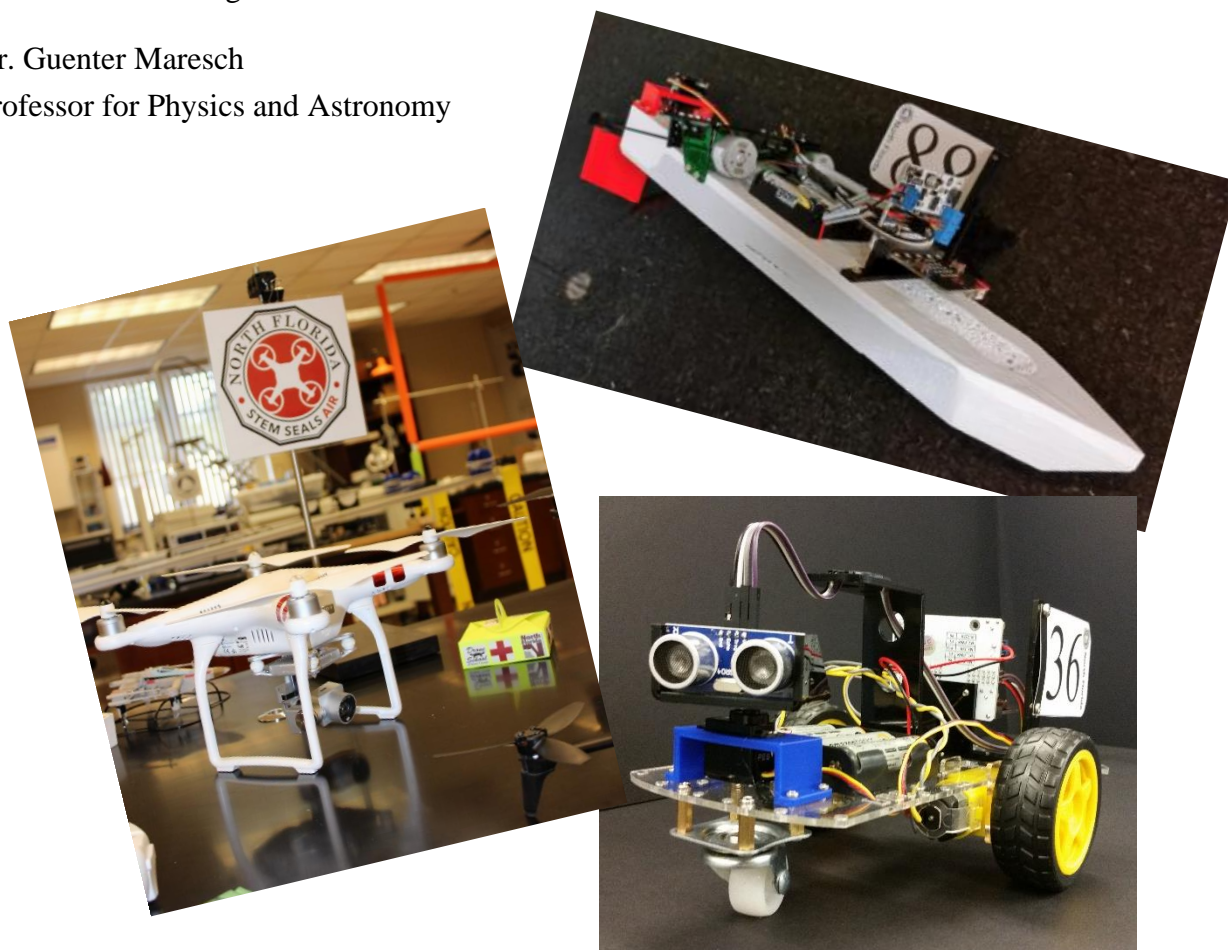
For each of the three disciplines, SEA, AIR and LAND, two documents are provided with instructions. The “Lectionary” is the student guide, the textbook needed by each student, containing modules describing each activity but also providing plans and images for clarification and troubleshooting when needed. The second document, the “Dictionary” is a guide for teachers and / or parents, who might want to know more details for assisting the students, who might need to know alternatives in case an activity develops in an unexpected direction, or who may want to create their own materials and are looking for purchasing, engineering, or manufacturing information.

Although the initial three years of the STEM SEALs project were successful, the program itself is not hewn in stone as described in the materials. Due to time or resource limitations, teachers or parents may want to choose just certain modules from the program. This is certainly possible with the appropriate planning.

Also, the market changes rapidly, new products may be preferable due to better performance or price, or items described in the documents may be discontinued. The core of the STEM SEALs idea is not solely based on the devices used during those first three years. An open-minded STEM teacher or parent can adapt the program to different motors, sensors, computers or drones. Minor modifications likely are easier to accomplish than major changes.

All members of all STEM SEALs teams gratefully acknowledge funding by the National Science Foundation and its Discovery Research PreK-12 (DRK12) program and support by the leadership of North Florida College.

Dr. Guenter Maresch
Professor for Physics and Astronomy



Module 1: The LAND Challenge

1.1 Introduction to Robots

An automatically operating machine that replaces human effort is called a robot. While we often think of robots to resemble humans, then they are called “humanoid robots”, most existing robots look more like mechanical devices and are generally electrically and / or computer controlled.



A famous robot is R2-D2 from the movie series Star Wars (right, source: starwars.wikia.com), a robot with many humanoid features, but at the same time resembling more what we perceive as a machine.





C-3PO, the other famous robot from Star Wars (left, from shortlist.com), demonstrates astonishingly many humanoid elements what seems almost in contradiction to its undoubtedly non-human-type construction from metal.

In some restaurants we find robotic waiters, which serve drinks and food (below, from lonelyplanet.com). In their shape they resemble humans in some ways, while they often have less motional freedom (e.g., wheels instead of feet, fixed arms, and heads).



And the development of driverless cars has been making the news now for decades. They are robots, definitely not humanoid, but they replace human efforts and help us to get from one place to another.

SpaceX's Dragon cargo ship is a robotic vehicle, which performs many functions during flight automatically and with higher precision than any human pilot could accomplish.



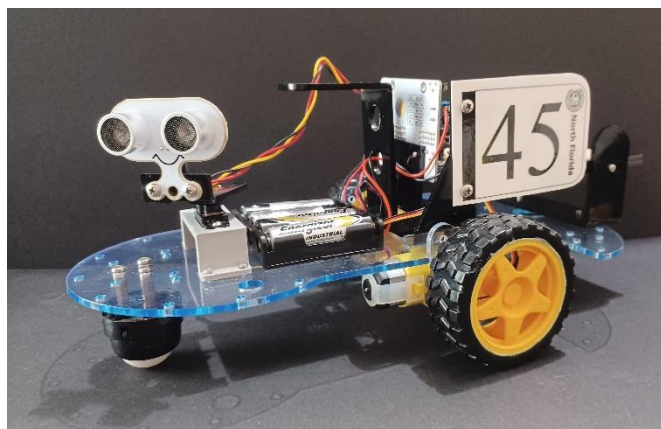
There are numerous other examples for robots in all kinds of shapes and for all kinds of purposes. So, what makes a robot a robot?

Sensors, actuators, and computers are common elements of all robots. We could see **sensors** such as cameras, ultrasonic distance measuring devices, microphones, and light sensitive diodes as man-made devices, which can accomplish a similar function as compared to the senses of biological organisms. We also could perceive actuators, such as electric motors, geared levers, and propellers as man-made devices, which can accomplish what we call motion for either animate or non-animate objects. And finally, the main controlling device of a robot is a computer while all mammals have brains, which have the top-level controlling function.



1.2 What is the LAND Challenge?

The LAND Challenge is one of three programs developed by STEM SEALs to expose middle school students to the concept of robotic devices involving the sea, air, and land. Each discipline guides students through engineering complete robotic devices or their controls. The LAND Challenge involves building a robotic rover capable of being navigated autonomously or by a remote control. Students will build, program, and test the rover through different challenges to ultimately perform a specific purpose, in this case delivering medical supplies between different locations.

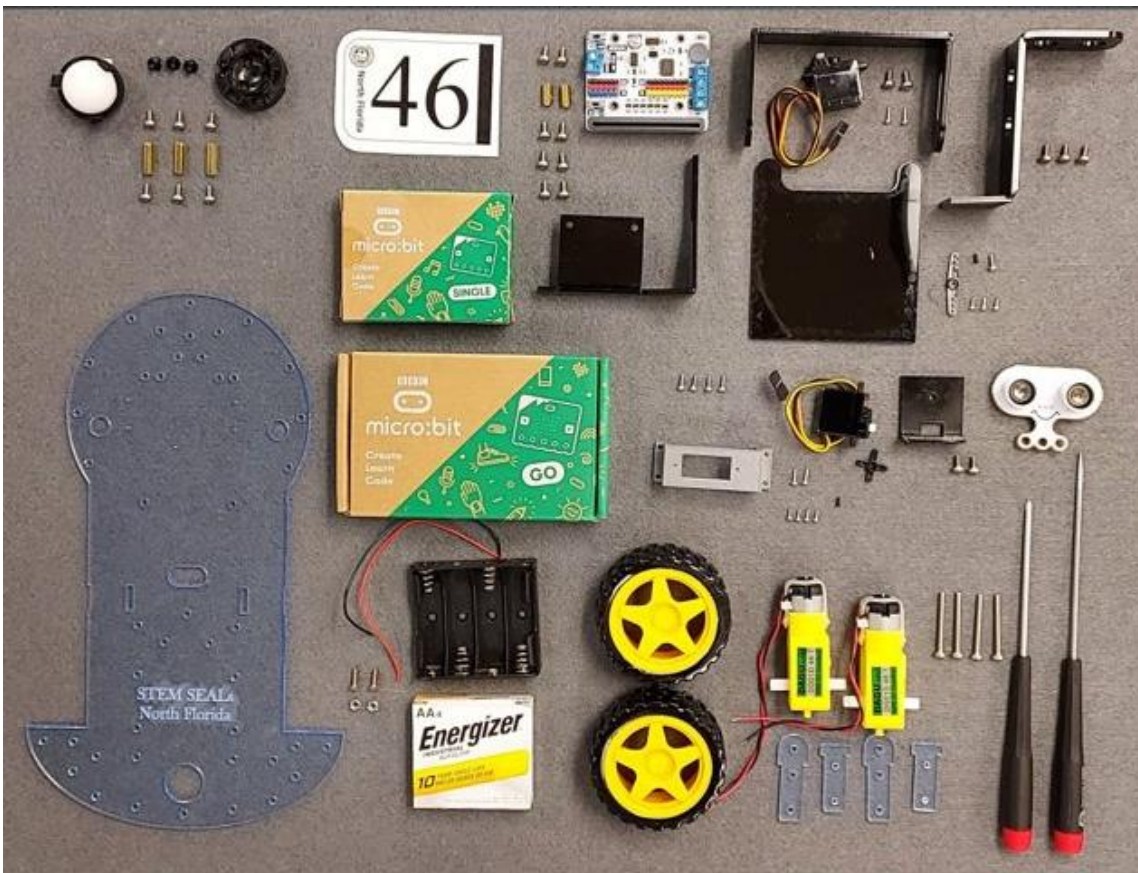


1.3 Unpacking the LAND Challenge Kit

Before starting the LAND Challenge, it is important to make sure you have all the necessary parts. All the parts should be in the plastic container.

Plan Ahead: Use the container to ensure you do not lose any parts.

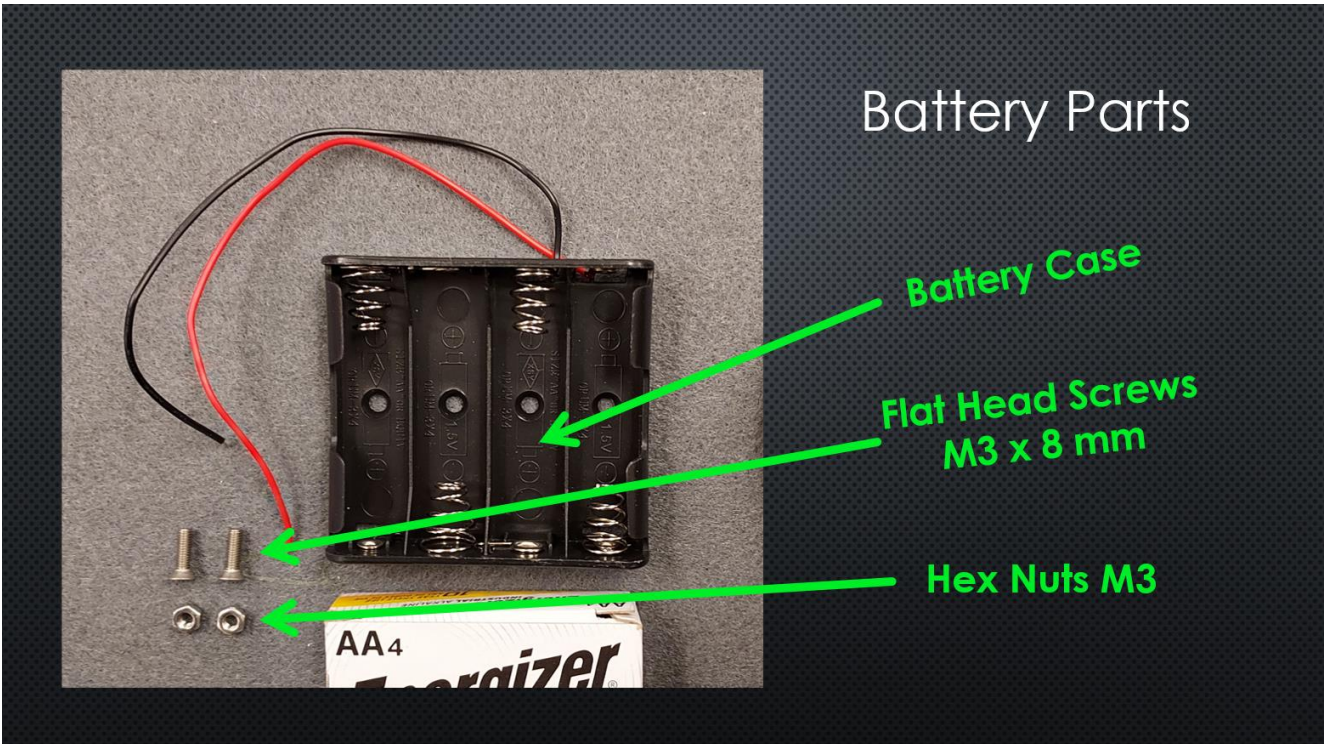
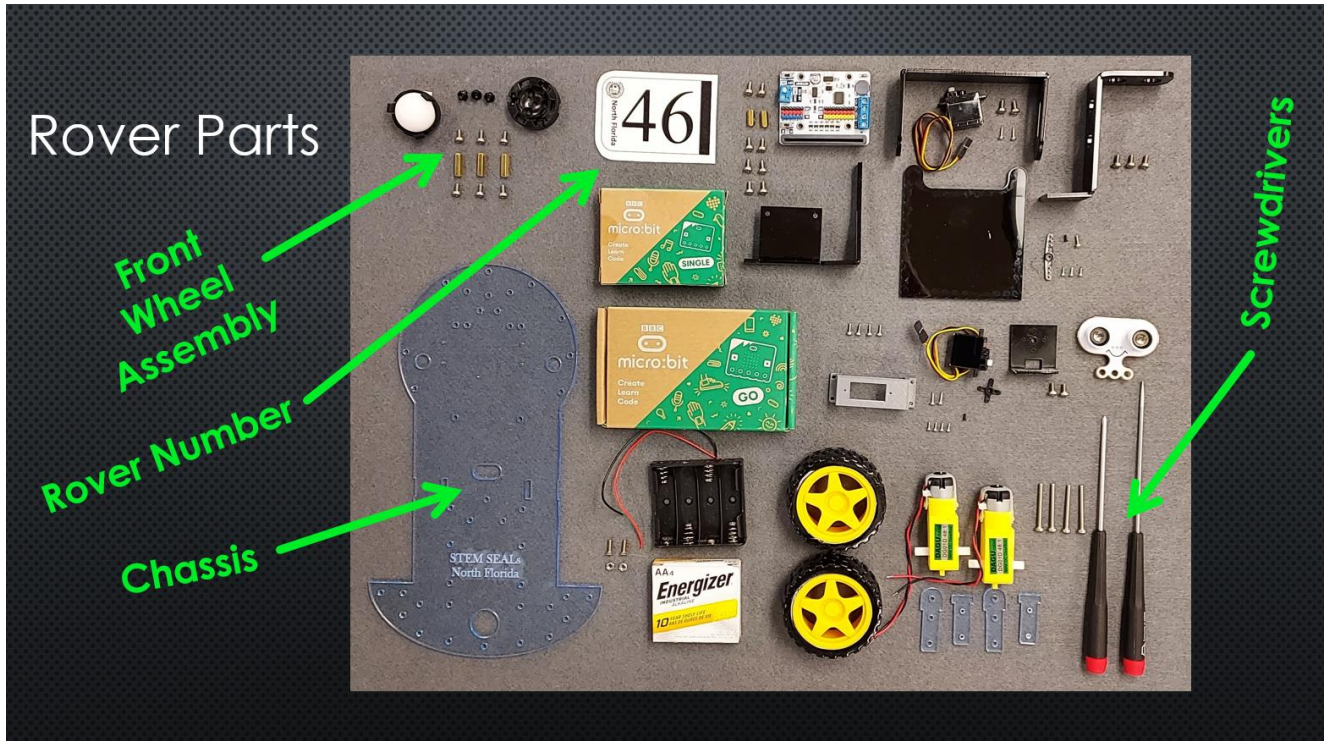
Caution: There are three electronic boards in the kit, two micro:bits and the motor:bit. Electronic boards are touch sensitive. They are stored in antistatic bags to protect the electronic components which are prone to damage caused by electrostatic discharge.

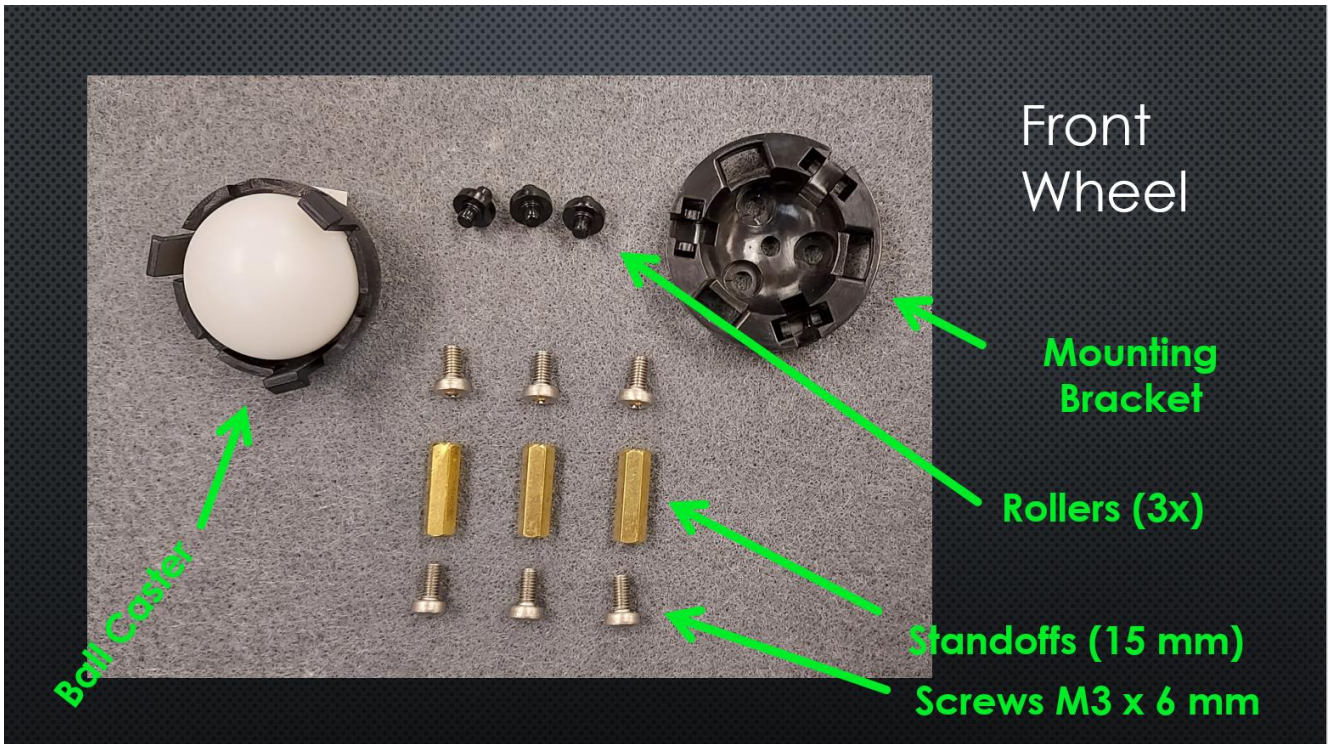
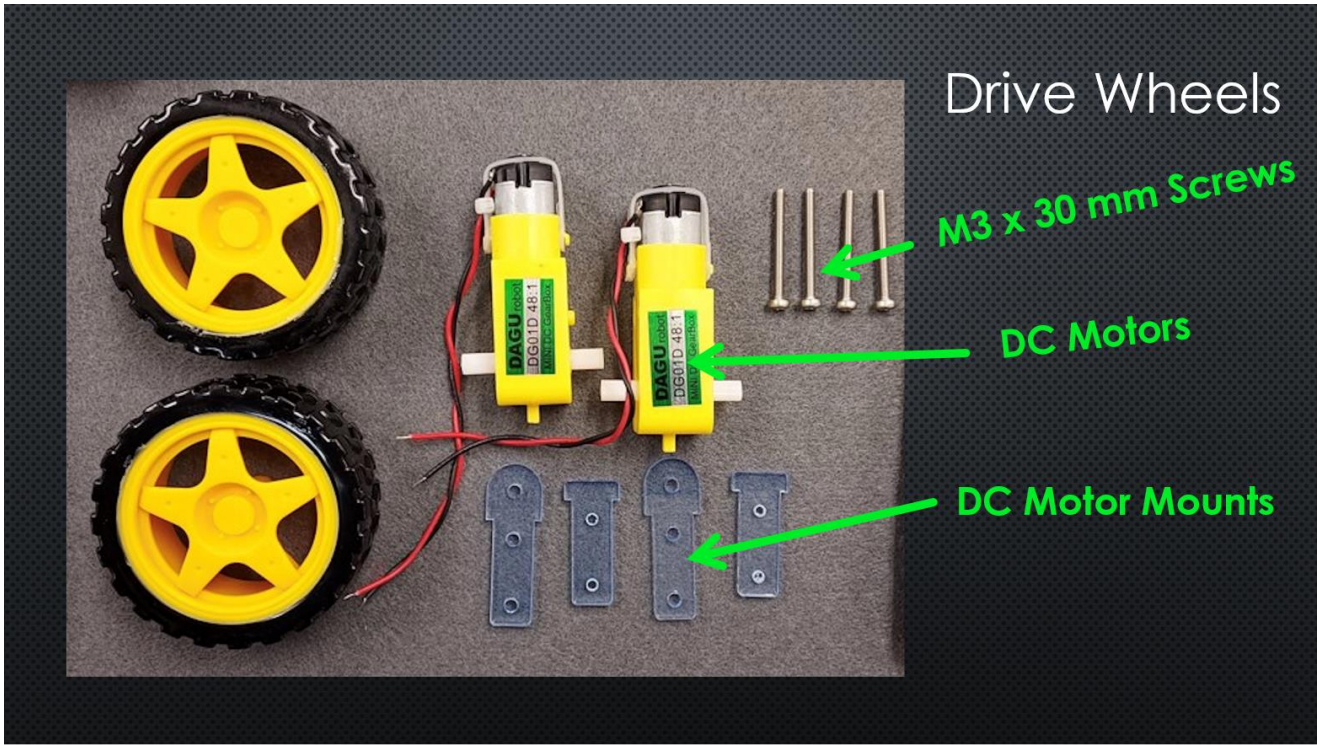


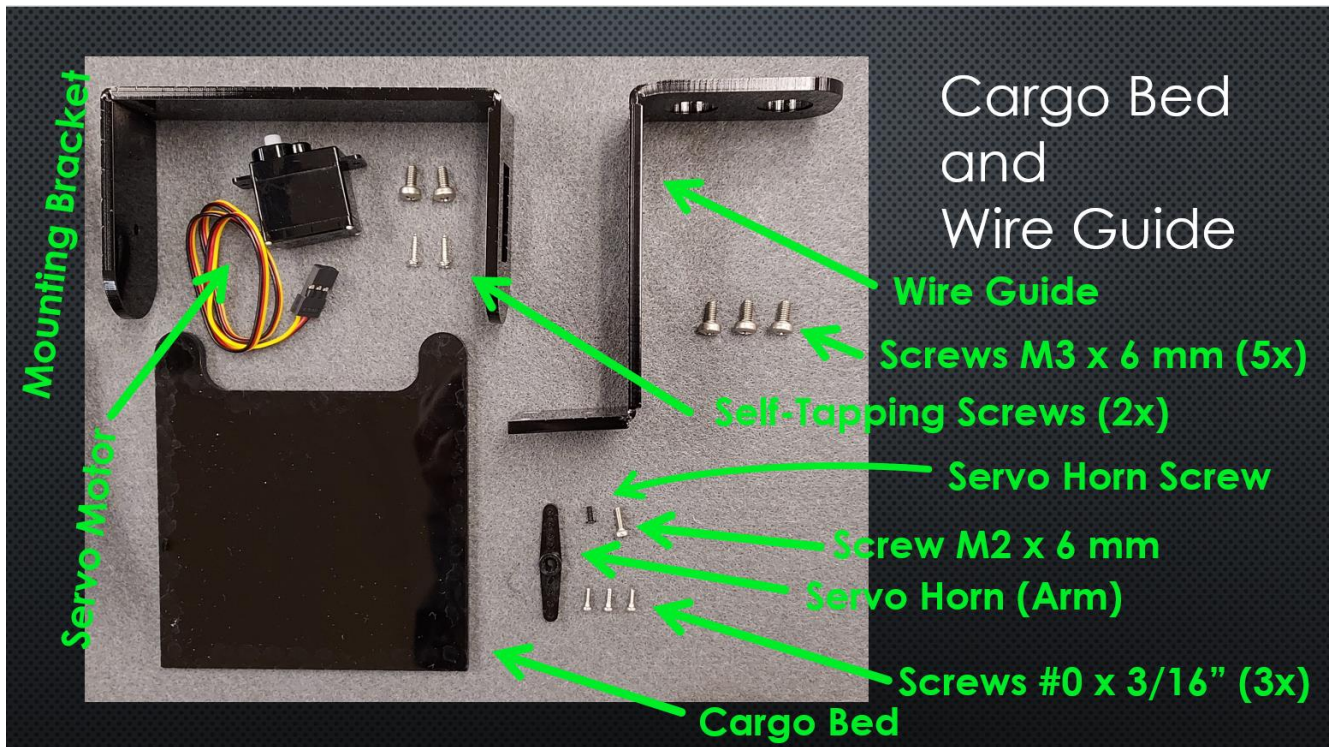
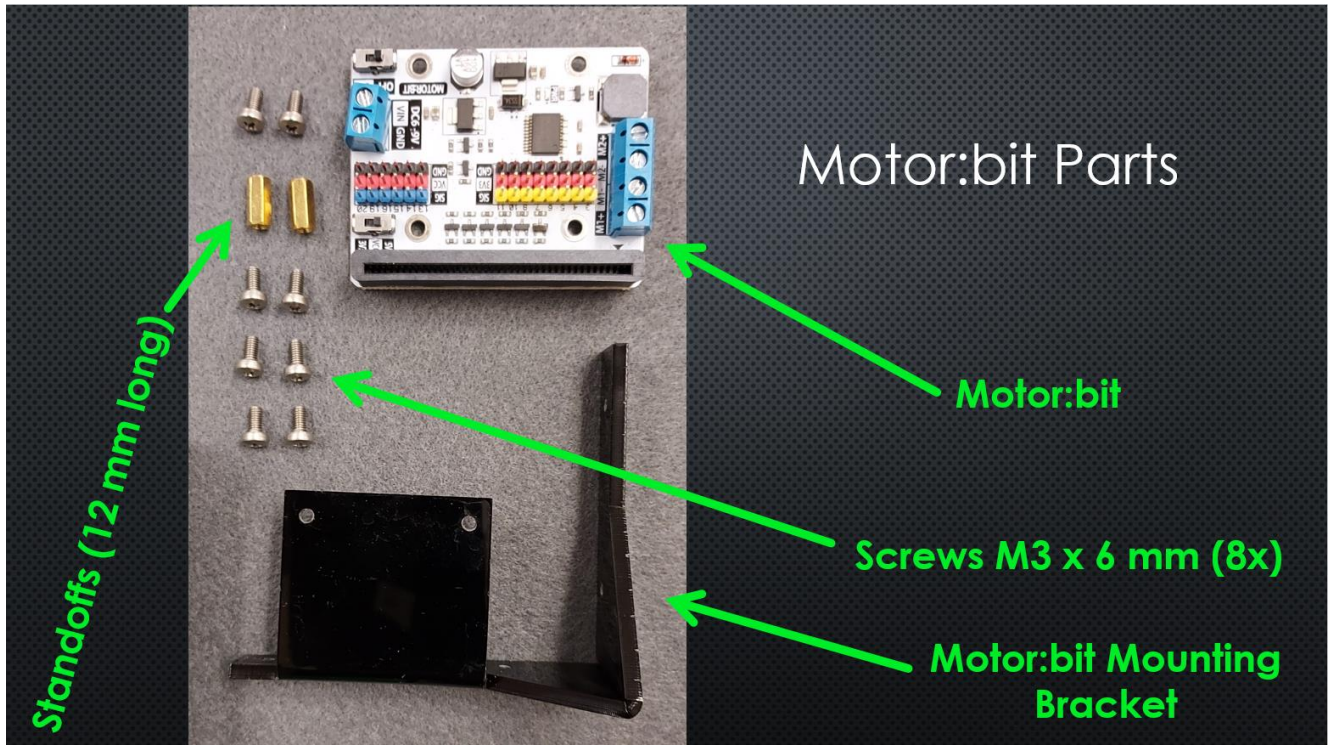
Basic Rover Parts: chassis, micro:bits, batteries, wheels, sonar, cargo bed, and tools.

All parts of the STEM SEALS rover (except the storage container).

Refer to the following detailed and labeled descriptions of the individual parts on pages. There is a check list included in your kit. As you identify each part, check it off on the list. If you are missing a part, let your teacher know.







All Screws

Flat Head Screws

#0

M2

Self-Tapping Screws

M3 is the (metric) thread size



Rover Parts List

No.	Part Name	Description	Packed	Received
1	chassis plate	fluorescent blue, laser cut		
2	DC motors 2x	100 nF ceramic capacitor, yellow and grey wire (one left, one right!)		
3	motor wheels 2x			
4	motor brackets 2x	large (outside), laser cut		
5	motor brackets 2x	small (inside with thread), laser cut & tapped M3		
6	screws M3 x 30 4x	stainless steel, McMaster		
7	ball caster kit	Polulu		
8	standoffs M3 x 15 3x	for front wheel, DigiKey		
9	screws M3 x 6 6x	pan head		
10	4-AA battery case	with leads, Polulu		
11	M3 x 6 flat head 2x	screws, McMaster		
12	M3 hex nuts 2x	McMaster		
13	AA batteries 4x			
14	sonar wire guide	laser cut, tapped M3		
15	M3 x 6 pan head	screws, McMaster		
16	motorbit mount	laser cut, tapped M3		
17	standoffs M3 x 12 2x	DigiKey		
18	M3 x 6 pan head	screws, McMaster		
19	motor:bit	ElecFreaks, China		
20	rover number flag			
21	M3 x 6 pan head	screws, McMaster		
22	servo motor pck 2x	HorizonHobby		
23	sonar servo mount			
24	M2 x 6 screws 4x	to fasten servo mount, McMaster		
25	self-tapping screws 2x	for mounting servo, included with servo motor		
26	sonar servo horn	X-shaped, screwed to sonarbit mount		
27	sonar:bit mount	laser cut		
28	sonar:bit	RobotShop / ElecFreaks, China		
29	M3 x 6 pan head 2x	to mount sonarbit, McMaster		
30	#0 x 3/16 4x	for servo horn, McMaster		
31	cargo bed mount	laser cut, tapped M3		
32	cargo bed	laser cut, drilled M1.4 and 1/8"		
33	self-tapping screws 2x	for mounting servo, included with servo motor		
34	cargo bed servo horn	medium I-shaped, mounted on cargo bed		
35	#0 x 3/16 3x	for servo horn, McMaster		
36	M2 x 6 screw 1x	for cargo bed, McMaster		

37	micro:bit V2 2x	Adafruit, NYC		
38	2-AAA battery case	for RC, included with V2 Go Kits		
39	AAA batteries 2x	included with V2 Go Kits		
40	Phillips screwdriver	#0 75 mm long, McMaster		
41	flat screwdriver	2.5 mm x 100 mm, McMaster		
42	tweezers	plastic, Amazon		
43	USB Type A to micro	cable, 3' long		
44	USB flash drives	8 GB		
45	rulers	12" / 30 cm long, Westcott		
46	protractors	180°, Staedler		
47	blood type test kit	HomeSchoolTools		
48	container with lid	15 qt, 16.25" x 11.25" x 6.75", Walmart		
49	LAND Lectionary			
	Screws Summary			
1	screws M3 x 30 4x	for mounting the DC motors		
2	M3 x 6 21x	pan head		
3	M3 x 8 2x	flat head (battery case)		
4	M3 hex nut 2x	(battery case)		
5	M2 x 6 screw 5x			
6	#0 x 3/16 7x	for servo horns		
7	black #0 1x	for the cog wheel of the servos		
8	self-tapping screws 2x	for mounting servos		

Module 2: What Does a Microcomputer Do?

2.1 The BBC Micro:bit

STEM SEALS devices (rovers and boats) are controlled with a Micro:bit- a microcomputer.

“The BBC micro:bit is a pocket-sized computer that introduces you to how software and hardware work together. It has an LED light display, buttons, sensors and many input/output features that, when programmed, let it interact with you and your world.” (Microbit.org)

Open a browser on your computer. Go to the Micro:bit website: www.microbit.org. Click on the “Get Started” tab at the top of the page. Watch the “Introduction to the Micro:bit” video. To learn more about what the micro:bit can do you may also wish to watch the “Input & Output Devices” and the “Processor” videos on the same page.

2.2 Let’s Explore the Micro:bit

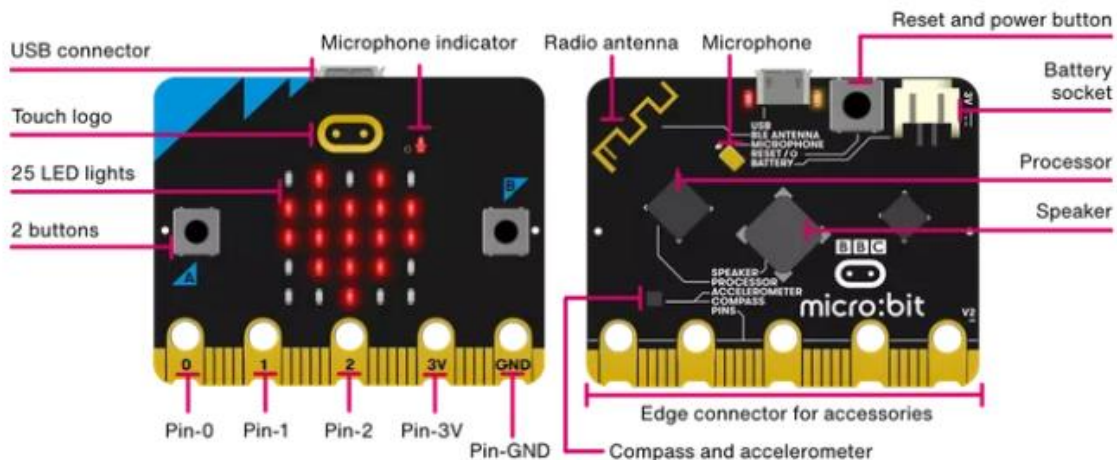
Open the micro:bit. (Go Bundle Kit).

It contains the following items:

- a micro:bit (color may vary)
- a USB cable
- two AAA batteries
- a battery case with a J2 connector



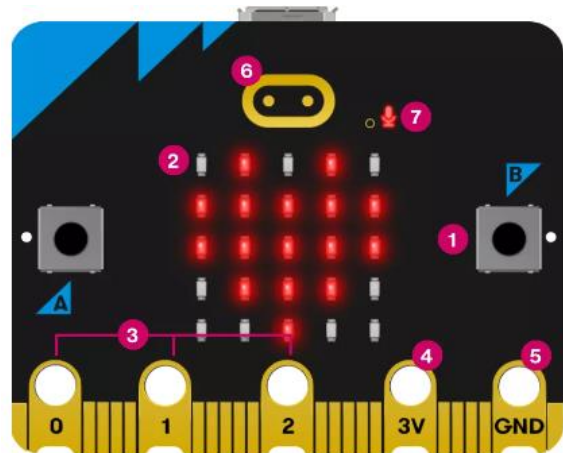
Look closely at the features of the micro:bit.



The front of the micro:bit contains

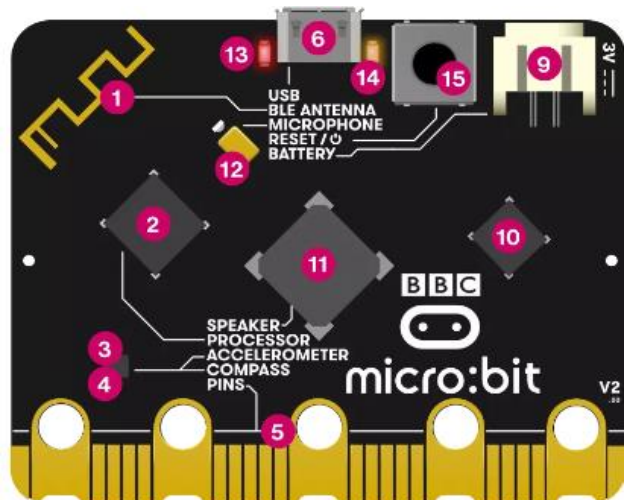
the following features:

1. 2 input buttons (A and B)
2. 25 LEDs in a 5 X 5 array
3. Pins - GPIO
4. Pin - 3-volt power
5. Pin - ground
6. Touch logo
7. Microphone LED



The back of the micro:bit contains these features:

1. Radio and Bluetooth antenna
2. Processor and temperature sensor
3. Compass
4. Accelerometer
5. Pins
6. Micro USB socket
7. Single yellow LED
8. Reset button
9. Battery socket
10. USB interface chip
11. Speaker
12. Microphone
13. Red power LED
14. Yellow USB LED
15. Reset and power button



You can find more about these features online in the Micro:bit User Guide:

<https://www.microbit.org/get-started/user-guide/overview/>



Internet Resources:

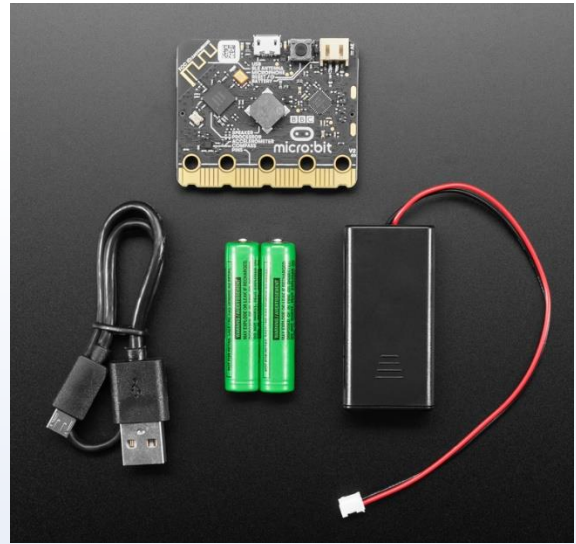
Check out the following to learn more about the micro:bit.

- [Introduction to the Micro:bit](https://youtu.be/u2u7UJSRuko) (video) <https://youtu.be/u2u7UJSRuko>
- [Getting Started with the Micro:bit Part 1: Say Hello](https://youtu.be/kaNtg1HGxbY) (video) <https://youtu.be/kaNtg1HGxbY>
- [Micro:bit Tutorial Series Part 1: Getting Started](https://youtu.be/ZIW_6rxYNBg) (video) https://youtu.be/ZIW_6rxYNBg



Try This: Run the DEMO Program on the Micro:bit

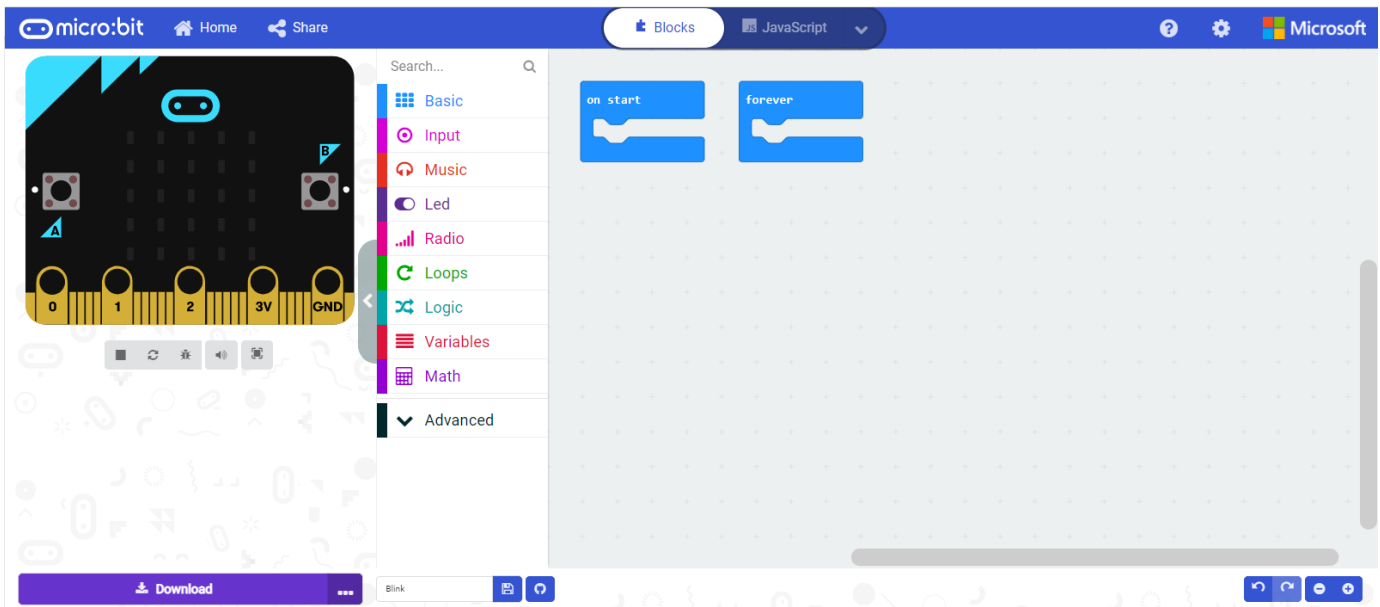
1. Connect the micro:bit from the Go Bundle Kit to a power supply. There are two ways to supply power. Connect to the computer using the USB connector or to the battery case with batteries.
2. Look at the LED display on the front of the micro:bit. This display can show icons, letters, and numbers or any image that can be created with the 25 LED outputs. Can you read what it is displaying?
 - Due to its small size most words and numbers are displayed one character at a time and therefore strings of characters will scroll across the LED display area.
3. Follow the directions by reading the LED display.
 - It will guide you through several activities including pressing buttons A and B, and the logo.
4. Once the demo has finished running, press the reset button on the back of the micro: bit, the demo or code will run again.
5. The micro:bit V2 version has a new feature to allow you to conserve your batteries. Press and hold the rest button on the back. Notice that the red LED indicator will dim and then go off. Release the reset button. Now the micro:bit is in power saving mode- to turn it back on press the reset button again. This will be handy later.



2.3 Accessing and Using the MakeCode Editor

There are multiple ways to create or write code for the micro:bit. STEM SEALs will use the MakeCode editor by Microsoft (www.microbit.org). The MakeCode editor allows the user to use blocks or JavaScript to write code. For coding in the LAND Challenge, we will use the block method of coding. On the next page is a description of what you will find on the MakeCode workspace.

Microsoft MakeCode Editor:



Explore the features of the MakeCode Editor:

- There are three main areas of the MakeCode Editor: the [micro:bit simulator](#), the [toolbox](#), and the [workspace](#).
- The [workspace](#) is where blocks from the toolbox will be dragged and dropped to create code.
- All projects start with an “[on start](#)” and “[forever](#)” block.
- In the middle of the page is a search box and below it is the [toolbox](#), to select tools (blocks) for creating code. It contains blocks such as: “Basic”, “Music”, “Input”, and many more.
- On the left is a [simulated micro:bit](#) that will display the commands that the code calls for such as lighting an LED. Simple coding processes can be viewed here on the simulator.
- In the top tab bar are buttons for Home, Share, tabs to toggle between block or JavaScript code, a help icon, and a settings icon.
- On the bottom of the page are buttons to download the code to the micro:bit, or a save icon to save the code to your computer, undo and redo icons, and zoom buttons.



Try This: Accessing MakeCode

1. On a computer connected to the internet, open a browser (Edge, Firefox, Safari, Chrome, etc.) and go to www.microbit.org.
2. Scroll down the page until you see the Micro:bit tab and click on it.
3. This will open a new page, the home page of the “MakeCode Editor”.
 - There are multiple ways to create or write code for the micro:bit.
 - STEM SEALs will use MakeCode and the block style for code creation.
4. Click on the purple area labeled “New Projects” (+).
5. A popup box will appear (Create a Project) and prompt you to give the project a name.
6. Name the project “Blink Timer” by typing in the space provided.
 - This will open the MakeCode workspace which will look like the image above.
 - Note at the bottom – the title of the project “Blink Timer” appears next to an icon for saving the project which will be used later.
 - Projects are automatically saved in the browser in the MakeCode server as soon as you create or edit them.
 - “Save” downloads a copy of the code to your computer device. This file will be saved as a .hex file.
 - To have your code no matter what computer you are working on, a USB flash drive will be provided. Make it a habit to save on of your coding onto this flash drive.

If you are uncertain how to save programs on a PC, refer to these resources:

<https://microbit.org/get-started/first-steps/set-up/>

2.4 Writing Code



Think About It

Do you know what code is?

Have you ever written any code?

Do you know what a coder or programmer does?

Simply stated: **code** tells a computer what actions to take. When a programmer or you write code, a set of instructions are created that tell the computer what to do or how to behave. Coders or **computer programmers** must think through the sequence of steps needed to obtain a task.

Follow the steps on the next pages to make the LEDs of the micro:bit blink in a timed sequence.

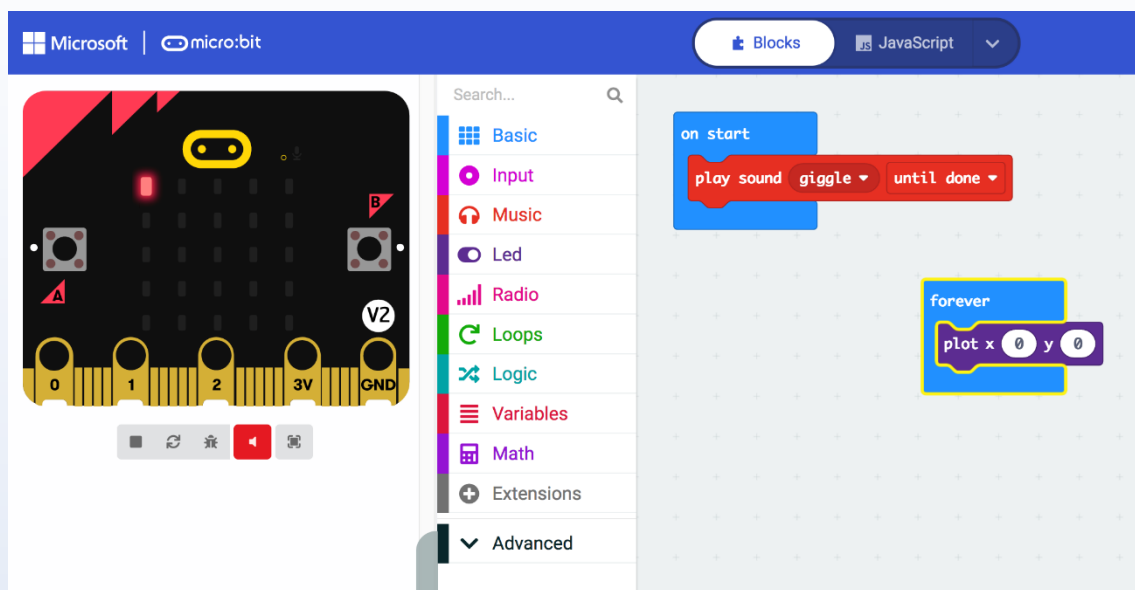




Time to Code: Create Simple Code and Download on the Micro:bit

1. Use the project you named “**Blink Timer**”
 - To locate the “**Blink Timer**” project, go to the home page of MakeCode.
2. Click on the “Music” toolbox and find the “play sound giggle until done” block.
 - Scroll down the list of blocks towards the bottom under the micro:bit (V2) label.
 - Drag and drop the “play sound giggle until done” block and place it in the “on start” block.
3. Click on the “LED” toolbox and drag the “plot x 0 y 0” block into the “forever” block.

Your code in the workspace should look like this:



3. Look at the micro:bit simulator on the left.
 - Do you see one of the LEDs now illuminated or on?
 - **LED** is spelled with capitals letters to represent “light emitting diode”. This is an electric component which produces light when an electric current is flowing through it.
4. Download the code to the computer and then upload it onto the micro:bit.
5. Observe the micro:bit (test the code).
 - You can leave the micro:bit connected to the computer to test this code or connect it to the power supply (batteries/case) provided with the Go Kit.
 - Listen: Does the micro:bit play a sound?
 - Look at the LED display – is there an LED illuminated?

Check out the information on the next page about the LEDs of the micro:bit.

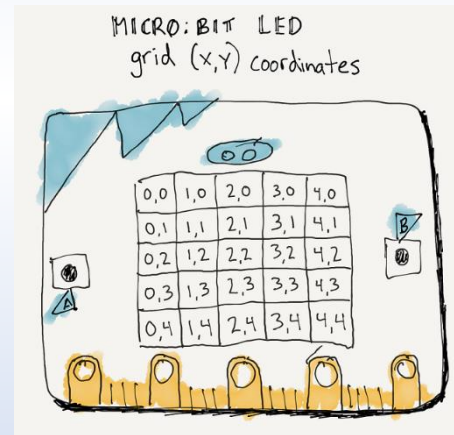


Note: Micro:bit LEDs

The **LED** grid on the micro:bit is different than the one you learned about in math class. The origin (0,0) of the grid is in the top left corner. The values of the y coordinates range from 0 through 4 and increase top to bottom. The values of the x coordinates range from 0 through 4 and increase from left to right just as they do in coordinate grids used in math.

If you want to know more about the micro:bit LEDs, check out these sources:

- <https://makecode.microbit.org/courses/csintro/coordinates/overview>
- <https://youtu.be/eRhlaXqT-0w>



Now that you know a little bit more about the LED features of the micro:bit. Let's make the LED blink on the micro:bit and move to the center.



Time to Code: Making the LED Blink

Let's modify the **Blink Timer** project to make the LED blink.

1. Open up MakeCode, select the "**Blink Timer**" project on the home page. Notice that changes are saved automatically in the program.
2. Change the location of the illuminated LED.
 - Look at the LED grid in the notes above.
 - What are the grid coordinates for the center LED?
 - In the "plot x 0 y 0", change the zeros (0) to twos (2).



Look at the simulator, is the middle LED illuminated now?
Now let's see if we can create a timer to make the LED blink.

3. Make the LED blink.
 - Drag a "pause" block from the "Basic" toolbox and place it after the "plot x y" block.
 - Drag an "unplot x y" block from the "LED" toolbox and place it next. change the zeros (0) to twos (2).
 - Duplicate the "pause" block by right clicking on it and place this new block after the "unplot x y" block.



Now look at the simulator! Is the LED blinking?

You just successfully completed your first coding project. Congratulations!

This simple code is telling the computer to turn on the LED at the center location, pause, then turn the LED off. When we write code, we are giving the computer directions to follow. What if we want to create a blinking LED with a timer to control the blinking? It will take a bit more complicated instruction. Let's explore what variables are and how they can help us write more complex code.

Variables are used in computer programs to store information that can be referenced or manipulated. They provide a way to label data with a descriptive name that can be recalled and used. The name's given variables are relevant to the task and make it easier to understand the code. Variables may be thought of as containers for information.



Internet Resources: Creating Variables in MakeCode

Check out the video to learn more about variables.

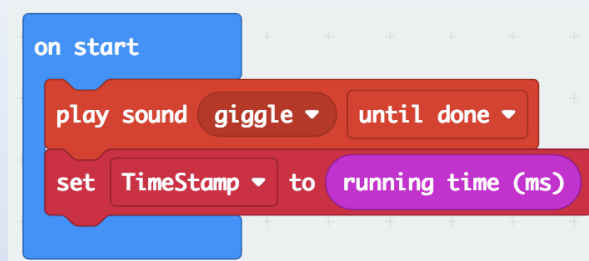
- [Micro: bit – Why variables?](https://youtu.be/HntpSgJIAKE) (video 8:55) <https://youtu.be/HntpSgJIAKE>
- [Create a Variable in MakeCode](https://youtu.be/vD4ywGNsZTA) (video 1:22) <https://youtu.be/vD4ywGNsZTA>



Time to Code: Blink Timer

1. Open up MakeCode, select the “**Blink Timer**” project on the home page.
2. Create a variable named Time Stamp
 - In the "Variables" toolbox, make a variable and name it "TimeStamp"
 - Drag the "set TimeStamp to 0" block to the end of the "on start" block
 - From the "Input" toolbox, in the "... More" list, drag the "running time (ms)" block over the zero in the "set TimeStamp to 0" block: the zero is replaced by the block, i.e. the current running time.

The “on start” block should now look like this:



(Continued on the next page.)



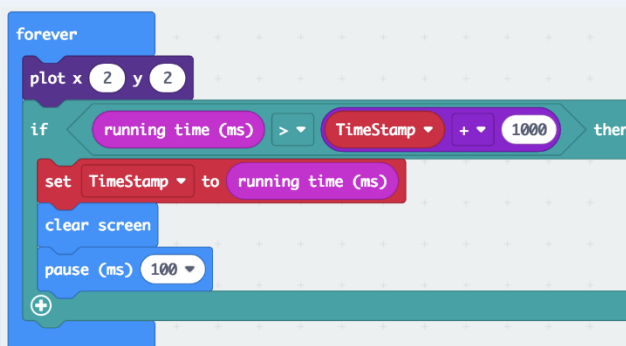
Time to Code: Blink Timer (continued)

Now let's change the "forever" block:

3. Create a logic statement – if true then:

- From the "Logic" toolbox, drag the "if true then" block to the end of the "forever" block.
- From the "Logic" toolbox, drag the " $0 < 0$ " comparison block over the "true" in the "if ..." block.
 - From the "Input" toolbox, drag the "running time (ms)" block over the first zero in the comparison block.
- From the "Math" toolbox, drag the " $0 + 0$ " block (make sure it is a plus sign and not the division sign - they can easily be confused) over the second zero in the comparison block.
 - From the "Variables" toolbox, drag the "TimeStamp" variable over the first zero in the addition block.
 - Enter 1000 instead of the second zero in the addition block.
- Select and copy the "set TimeStamp to running time (ms)" in the "on start" block, drag it into the "if running time $>$ TimeStamp + 1000" block
- From the "Basic" toolbox, drag the "clear screen" block after the "set TimeStamp ... block" in the "if ..." block.
- From the "Basic" toolbox, drag the "pause (ms) 100" block after the "clear screen" block inside the "if ..." block.

The "forever" block should now look like this:



Let's test out this code:

4. Download the code to the PC and then upload it to the micro:bit.

- Listen, does it play a sound?
- Look at the LED display. Do you see the red LED?
- Does the LED blink?
- How quickly does it blink? What is a "ms"?

Your micro:bit can measure time in milliseconds (ms). A thousand milliseconds are equal to one second. Like in your previous "Blink Timer" code, the LED comes on when the "on start" block has finished and the "forever" block says "plot ...". But then, if the current time is greater than the TimeStamp plus one second, the screen is cleared (the LED goes off) for 100 ms (how long is this in seconds?) ... until the LED comes back on at the beginning of the next cycle in the "forever" block.



Time to Code: Blink Timer (Optional Beep)

Let's add a beep to the program:

Start with the Blink Timer code:

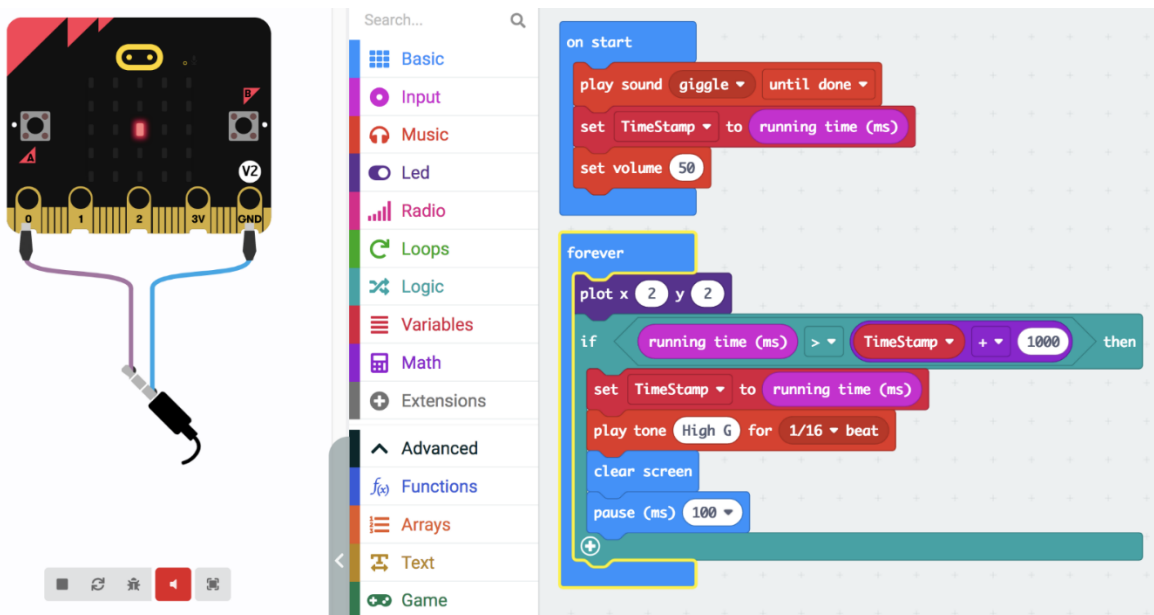
1. In the "forever" block:

- From the "Music" toolbox, drag the "play tone Middle C for 1 beat" block into the "if ..." block and drop after the "set TimeStamp ..." block
 - Change the "Middle C" on the piano keyboard to "High G"
 - Change the "1 beat" in the block to "1/16 beat"

2. In the "on start" block:

- Drag from the "Music" toolbox the "set volume 127" block to the end of the code in the "on start" block
 - Change the number 127 to 50

Your code should look like this:



Download, upload, and test:

- Does it beep?
- Feel your pulse and listen: is your heartbeat slower or faster than the beep from the micro:bit?
- Can you estimate your pulse rate?

Notice: the red speaker icon in the bar below the micro:bit simulator in MakeCode. When you click on it, you will hear the simulator on your computer beeping.

2.5 Sharing or Publishing Your Code

In the MakeCode editor, there is a share image at the top of the page:



Here is a published STEM SEALs link of the code you just completed:

"**Blink Timer**" code: <https://makecode.microbit.org/KFPTALMYDa10>

When you share or publish your code using MakeCode it is not automatically made public. It will generate a unique URL code for your project that you can copy and share with others as you wish.

If you click the share icon in your MakeCode editor, it will have its own unique URL even though it has the same name.



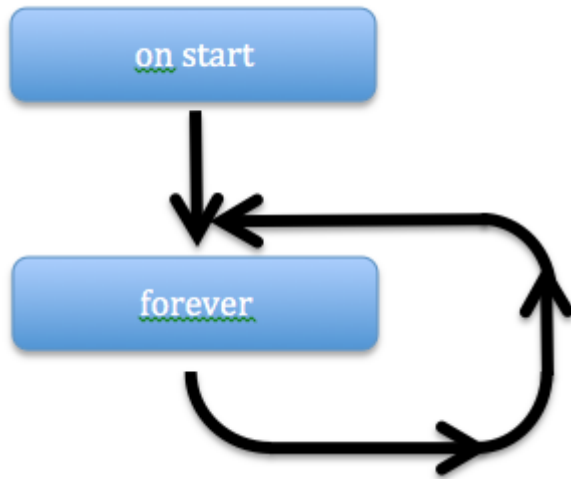
Try This: Share a MakeCode Project

1. Click on the "Share" symbol at the upper left of the MakeCode editor.
2. Click on "Publish Project".
3. Click on "Copy" and....
4. Paste the URL link in a document for safe keeping or send to a friend in an email.
 - If you save this link, you can send it to your friend who then can access your code with this link and work with it.
 - Or if you changed the code and want to get back to it later: Just click on the stored URL and it gets you right back to your original code.
 - It is good practice to save all your projects at each stage of development into a special document. You can save this document on the USB flash drive for safe keeping.
 - You will find links like the above one throughout this document. They allow you to modify complex code without rewriting all of it.

Module 3: Micro:bit Code

3.1 How Does Code Work?

We learned that computer code contains or uses two main parts, which in MakeCode are called the “on start” block and the “forever” block.



All blocks placed in the “on start” block are executed once when the micro:bit is powered up, or when the reset button is pressed.

After completing the commands in the “on start” block, the blocks within the forever block are executed in sequence and then repeated over again and again. Therefore the “forever” block is also called the “forever loop”.

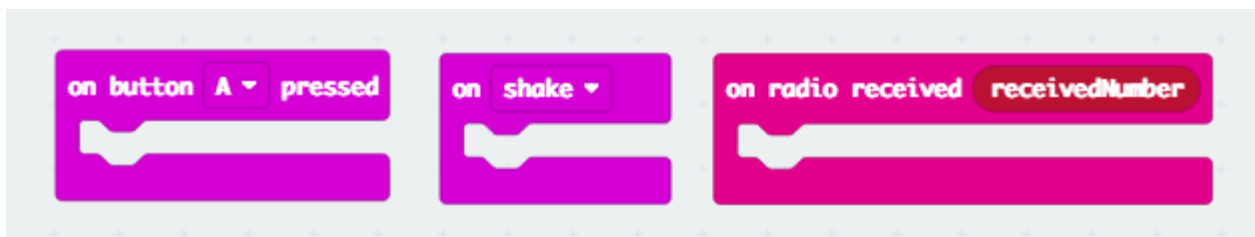
The micro:bit only stops running the “forever” block commands when it is turned off or restarted with the reset button.

Remember code written for the micro:bit computer to process determines what actions the micro:bit takes. It is important to think about how and when you want a task to happen and determine the specific order or sequence in which the action happens. The following sections list some of the blocks available in MakeCode and what they do.

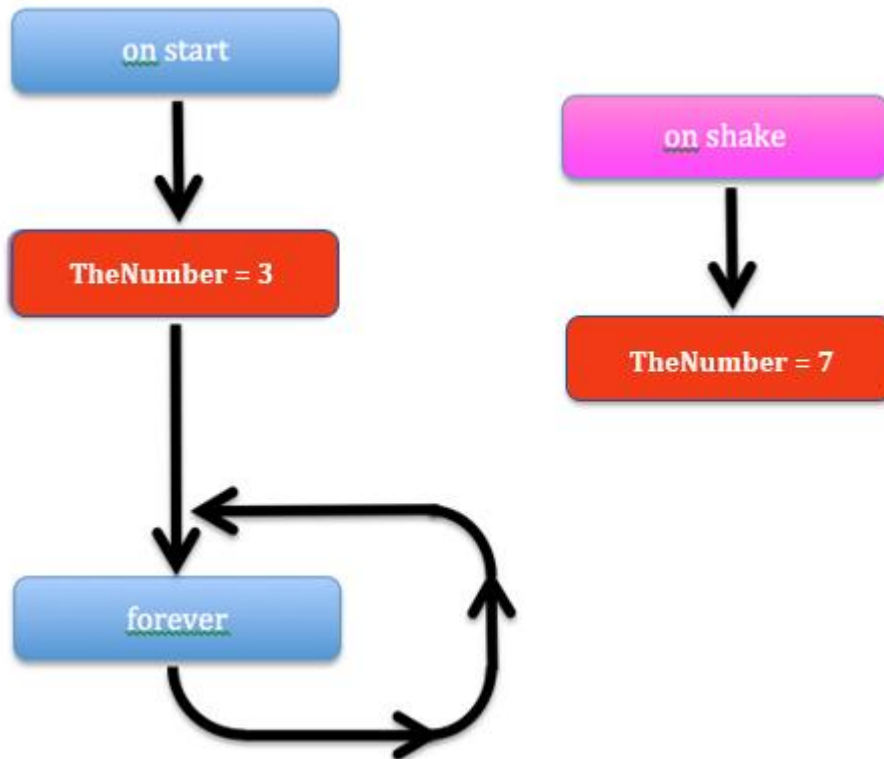
3.2 Interruptions in Code

If all you could do is program the “on start” block and “forever” block, you might find your programs become limiting or boring. Luckily, the computer or micro:bit can also be controlled using [inputs](#). Inputs can change how the code performs during its pass through the forever loop.

One way to make changes is by using external inputs, such as pressing a button on the micro:bit, shaking the micro:bit, or sending a radio signal. When these are used or activated, the task can be altered or changed.

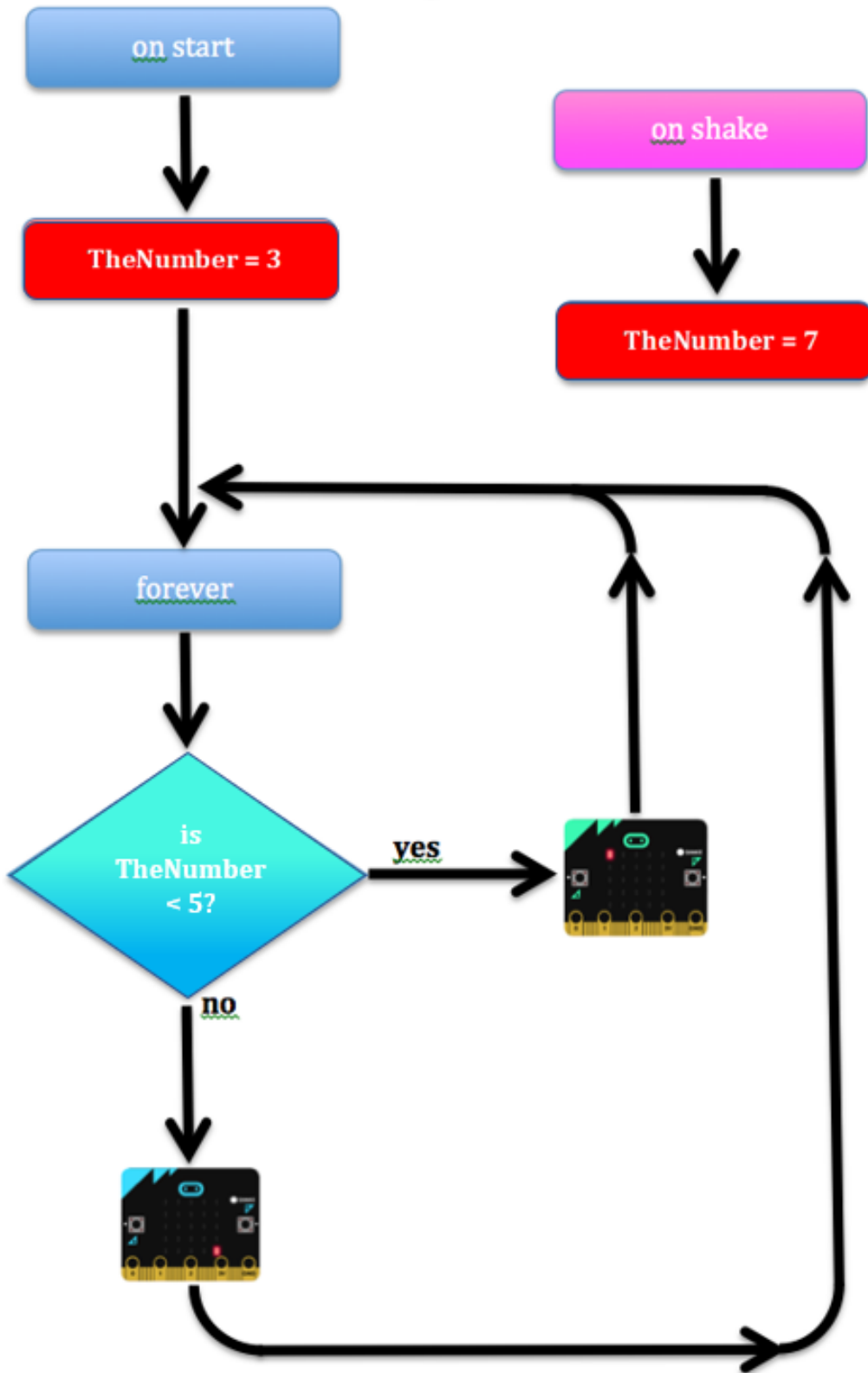


We can place blocks inside these input blocks, which then are executed once each time the event happens such as pressing the button or shaking the micro:bit. After the input commands are completed within the input blocks, the forever loop resumes with whatever it was supposed to do. The empty blocks as shown above do not change anything. The performance of the forever block will only be changed if other blocks or variables are added to the input block with specific commands to do something else.

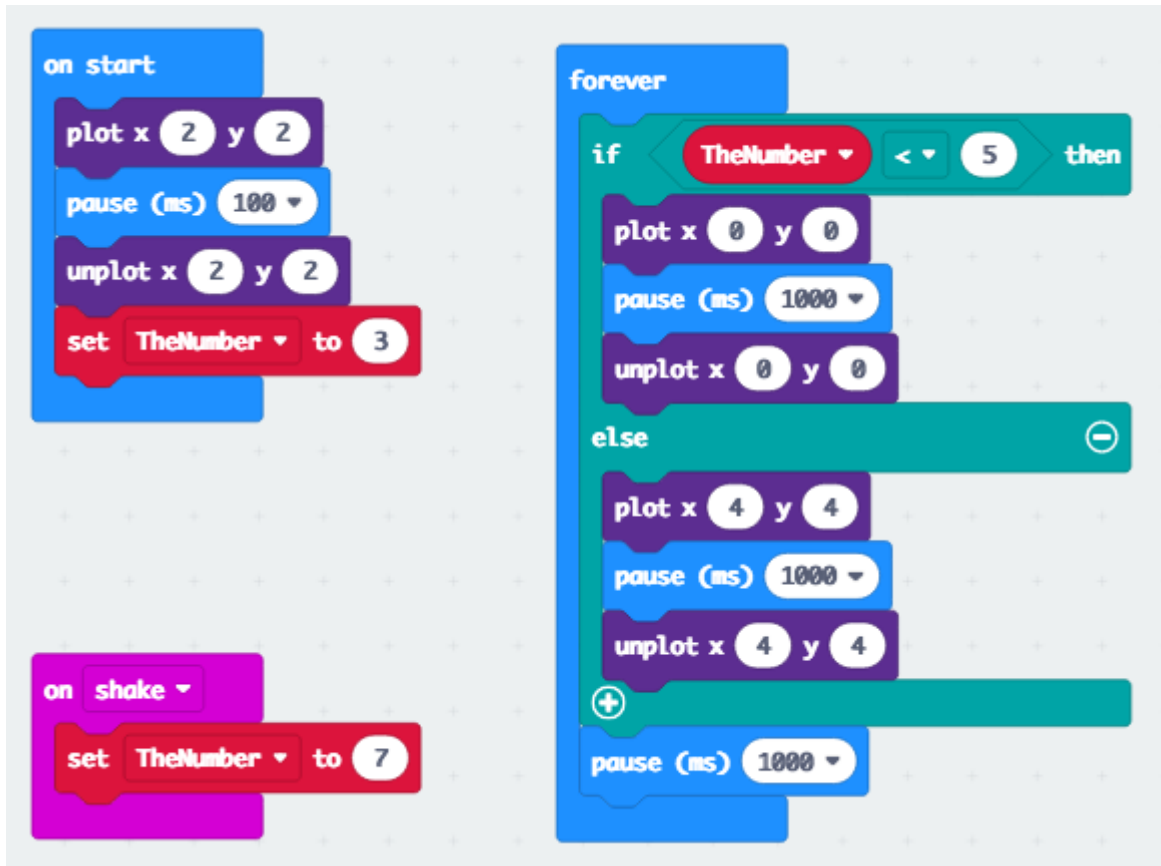


For example, if we set the variable “TheNumber” to 3 in the “on start” block, the forever loop “TheNumber” will always be 3 ... until we shake the micro:bit: Then the “on shake” block will set “TheNumber” to 7 and every time thereafter “TheNumber” will be 7 and the forever loop will “know” this.

The reason we changed the value of the variable was that we wanted the forever loop to do something different. What if we want the number to change based on a set of conditions? We can achieve this by using “if ... then ... else” blocks. These blocks help the micro:bit make decisions! For example: If we want the LED to blink at the top left of the micro:bit if the number is less than 5 and if not or “else” then blink the LED at the lower right. We can think of the sequence or flow of commands to look like the diagram below.



To write this code in MakeCode we do not need all the arrows. The blocks themselves “know” what to do. The flow chart or sequence of events would be represented like the diagram below.



https://makecode.microbit.org/_90UCkgDWteux (Blink 2)



Try This: Adding an External Input (A Shake) (Optional)

1. Open the MakeCode. Create a new project titled “**Blink 2**”.
2. Using the diagram above, see if you can create the same code.
 - Notice that the color of the blocks corresponds or matches the color of the toolbox categories.
3. You can see this new change in the code when you shake the micro:bit simulator.
 - Move your mouse cursor quickly back and forth over the simulated micro:bit – this allows you to shake the micro:bit.
 - Did the blinking LED change locations?



Tip: Using Previous MakeCode Projects

Remember that your projects are always saved on the MakeCode browser. However, if you go back into a project and change the code, this new edited code is now what is saved in the browser. By sharing the URL link to a document, you will always have access to the earlier version.

Another way to save different versions is on the MakeCode projects page. On the Home page, click “view all” projects. Find the project you want to use and click in the “circle”. In the top tab bar, three options will appear: Open, Duplicate, and Delete. If you duplicate the project, a popup box will appear and ask you to name the project. Using this option always allows you to have quick access to all versions of your code in your browser. However, it will only be available on the same computer (device). Therefore, it is **recommended** to do both -save the URL in a document (or a flash drive) and duplicate and rename for extra safety precautions in the computer browser.

In the example above, the forever loop is still processing the same commands repeatedly. By adding an external input (a shake) that changed the value of the variable “TheNumber”, the blinking LED changed location. How can you get the upper left LED to blink again? One solution is to create two different inputs. This time we will use the inputs Button A and Button B on the micro:bit.

Try to edit the code like the diagram below:

```
on start
  plot x 2 y 2
  pause (ms) 100
  unplot x 2 y 2
  set TheNumber to 3

on button A pressed
  set TheNumber to 2

on button B pressed
  set TheNumber to 8

forever
  if TheNumber < 5 then
    plot x 0 y 0
    pause (ms) 1000
    unplot x 0 y 0
  else
    plot x 4 y 4
    pause (ms) 1000
    unplot x 4 y 4
  pause (ms) 1000
```

Now you should be able to control which of the two LEDs is blinking.

Use your mouse to click on the buttons (A and B) on the simulated micro:bit.

Were you able to control which of the two LEDs is blinking?

Can you control more than two LEDs?

Of Course!

Let's use the middle row of LEDs this time.

Try to change your code to look like the code below:

```
on start
  plot x 1 y 2
  pause (ms) 100
  unplot x 1 y 2
  set TheNumber to 0

forever
  plot x TheNumber y 2
  pause (ms) 1000
  unplot x TheNumber y 2
  pause (ms) 1000

on button A pressed
  change TheNumber by -1

on button B pressed
  change TheNumber by 1
```

The image shows a Scratch code editor with a grid background. It contains four main code blocks: 1. An 'on start' block (blue) containing: 'plot x 1 y 2' (purple), 'pause (ms) 100' (light blue), 'unplot x 1 y 2' (purple), and 'set TheNumber to 0' (red). 2. A 'forever' loop block (blue) containing: 'plot x TheNumber y 2' (purple), 'pause (ms) 1000' (light blue), 'unplot x TheNumber y 2' (purple), and 'pause (ms) 1000' (light blue). 3. An 'on button A pressed' block (magenta) containing 'change TheNumber by -1' (red). 4. An 'on button B pressed' block (magenta) containing 'change TheNumber by 1' (red).

Can you control which LED is blinking in the simulator?

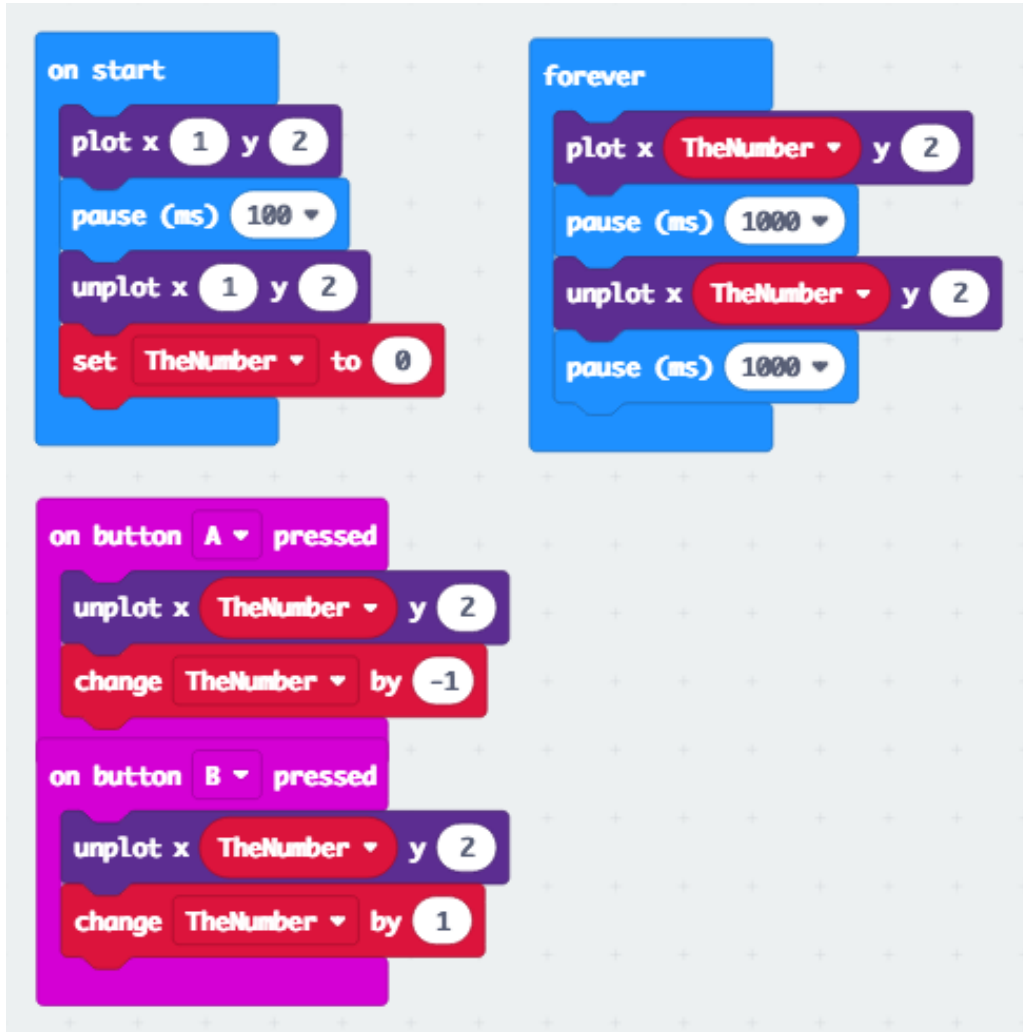
Notice this code is simpler: It does not need the “if ...then...else” block because we use the variable “TheNumber” to control which LED blinks.

We have renamed this version of the code **Blink 5**: <https://makecode.microbit.org/LVU7sbAWq4w5>

Do you see a problem with this?

Since the forever loop is interrupted with input from our buttons, the previous LED can be on or off; and depending on when we push the buttons, the old LED may stay on. This may not be desirable.

We can solve this by turning the old LED off before we change the number: Try this! (optional)



The two additional “unplot ...” blocks take care of unfinished business and clean up the micro:bit’s display avoiding “old” LEDs being displayed.

Do you see another problem we might encounter?

What happens if you push the A or B button more than 5 times?

When we push button A or B too often, the LED may “wander off the display”! Of course, there are no moving LEDs. LEDs at x positions greater than 4 or less than 0 simply do not exist. The five in one row are fixed to the micro:bit board. We only have the impression that the blinking LEDs moves because we shift the target of the “plot ...” block around by changing the value of “TheNumber”.

A possible solution to the problem of the “disappearing” LED would simply be to not let “TheNumber” get too big or too small.

Try this! (optional)



Here we use the “if ... then” block without the “else” part to set the number to a value of an existing LED if the “TheNumber” variable gets out of range.

There are other solutions to this problem. An interesting one would be to let the LED appear in the row above or below when it crosses the borders. Or the micro:bit could simply display a warning instead of going completely dark.

The possibilities are numerous when you combine blocks with many functions!

Now that you understand how interruptions and conditionals can change how the code works, remember the “**Blink Timer**” code, let’s take a look at the code in this link for “**Count Timer**” and note how inputs were used to change the function of the code.

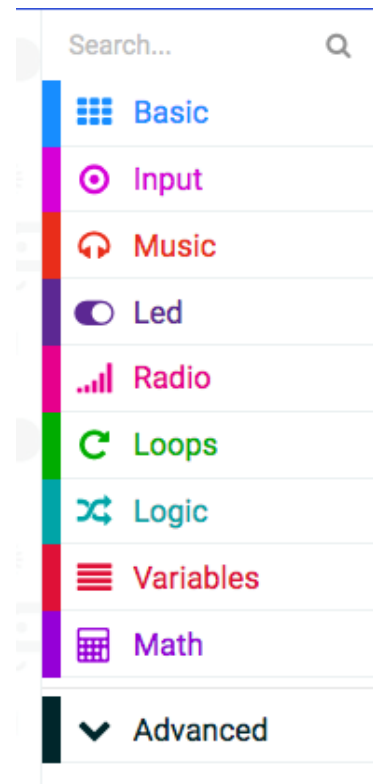
"Count Timer" code: https://makecode.microbit.org/_6XCXtm4ibKyu

Note the use of the “on logo pressed” block. This is a new feature of the V2 version of the micro:bit. Use Button A and B on the micro:bit simulator to change the beating (sound). What does it sound like?

Next, we will explore where the blocks come from and what types of blocks there are on the MakeCode editor.

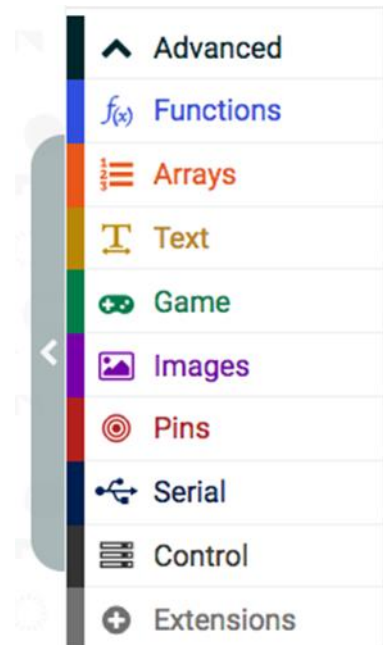
3.3 The Essential MakeCode Blocks

- Essential blocks for coding the micro:bit (and these are very similar for any other computer as well) are shown in the default view of the MakeCode editor underneath the “Search...” textbox:
- Examine each of the essential toolboxes.
- Look at what they contain.
- For example, in “**Input**” you find the buttons, but also a compass and accelerometer.
- “**Led**” are block controlling the LEDs.
- “**Radio**” are the commands how to communicate with other micro:bits.
- “**Loops**” are the tools to handle repetitive tasks.
- “**Logic**” are the components to allow the micro:bit to make decisions.
- “**Variables**” you make your own variables and manage them.
- “**Math**” you can add, subtract, multiply, compute roots and trigonometric functions.



3.4 Advanced MakeCode Blocks

- “**Functions**” allow you to make your own functions. That helps to keep complex code manageable.
- “**Arrays**” blocks operate on arrays of numbers or text.
- “**Text**” blocks allow you to modify text.
- “**Images**” blocks allow to create your own images.
- “**Pins**” blocks will help to operate the boats motors, the rudder and the water sampler.
- “**Serial**” is for communications.
- “**Control**” blocks control the micro:bit’s processing.
- “**Extensions**” give you access to code written by others for specific devices which helps to use those devices easily.



Note: The LAND challenge will be using the V2 version of the micro:bit. You may notice when opening a specific toolbox if you scroll to the end of the listed blocks, you will find blocks that are specific to the V2 micro:bit. While the other blocks will work on the V2 version, the blocks specific to the V2 will not work with the original micro:bit.

3.5 MakeCode Java Script

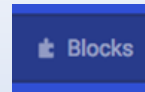
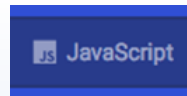
The MakeCode editor you have seen so far is a graphics-oriented editor. The blocks you drag, and drop must be assembled to a meaningful graphical image, which then is translated when you click “Save” into code, which the computer can read.

JavaScript is a text-oriented editor producing code which is equivalent to the code produced by the graphics-oriented editor. We will **not** be using JavaScript for most the STEM SEALs LAND Challenge, but you may utilize it to easily copy some code for a couple of activities.



Try this: MakeCode JavaScript

1. Open the “Blink Timer” project on the MakeCode homepage. This is the code we made earlier.
2. Click on the “JavaScript” tab at the top center.
You now see a text editor specialized to write code.
3. Each line represents a block or is part of a block.
Do you recognize some of the words?
4. The lines are called “statements” instead of “blocks”.
At the end of the statement are often numbers in parentheses separated by commas.
These numbers are called parameters.
5. By clicking on “Blocks” at the top center you will switch back to the graphics-oriented editor.



```
10 <link rel="stylesheet" href="http://localhost/css.css" type="text/css">
11 <script type="text/javascript" src="http://localhost/javascript.js"></script>
12 <script type="text/javascript">
13 (function(){
14   onLoaded: function(request) {
15     if (request.name == "log_error") return;
16     log_trace("Ajax.Request: " + request.name || request.url.slice(4));
17   },
18   onComplete: function(request) {
19     if (request.name == "log_error") return;
20     log_fatal(request.url + " " + request.name);
21   },
22   onException: function(request, e) {
23     if (request.name == "log_error") return;
24     log_fatal(request.url + " " + request.name + " " + e.message);
25   }
26 })();
```


Module 4: Testing Blood Samples

4.1 Why are we testing blood samples?

It is fun to make a robotic rover, but remember they are not only used for educational or entertainment purposes. Robots and automated devices can serve a wide range of tasks and are increasingly becoming common in many workplaces and industries. Among those is the healthcare industry. Robots are used in the medical field to aid in surgeries, streamline supply and delivery, and help in disinfection procedures which enable providers to focus more on the care of their patients.

To give our rover a real-world purpose, you will not only build and program a simple rover that can transport a “blood sample” but you will also act as a lab technician to determine the blood type of a given sample.



Internet Resources: Use of Robotic Rovers in Blood Transport

Check out these resources to learn more:

- Watch a video on the TUG transport system:
https://aethon.com/wp-content/uploads/2018/04/healthcare_box_video.mp4
- Check out how one company’s device is used for automated blood sample transport:
See excerpt on the next page.
<https://www.swisslog-healthcare.com/en-gb/medication-management/transport-automation/blood-transport>

Automated Blood Sample Transport (from Swisslog Healthcare)

From blood bank to the OR, or STAT to the bedside, our innovative solutions automate blood delivery to and from blood banks. Our flexible material-transport technology increases efficiency, reduces risks and loss, and improves patient outcomes. Automated blood sample transport and chain-of-custody solutions provide added safety and trackability, so you get more visibility. The result is an enhanced workflow everyone can rely on. Achieve fast, secure and verified delivery of life-saving blood products with our fully-integrated software solutions.

Automated transport solutions ensure the integrity of blood and specimens – and ensure they get there **quickly and safely**. From the blood bank to the ER, our reliable systems are equipped to meet the special requirements of blood transport.

Get total **visibility into chain-of-custody** through automatic tracking and tracing, delivery verification, and secure-send features.

Pneumatic tube systems and autonomous hospital delivery robots add consistency and predictability, while drastically reducing transport time. So you can focus on other crucial matters, like patient care.

Automated blood transport **reduces wait times**, which is critical to ensuring the integrity of blood products and achieving positive patient outcomes.

Our innovative transport systems allow batching and enable you to **trace materials** to gain the visibility needed to stay on track.

Our blood transport solutions also **minimize errors and increase efficiency** throughout your healthcare facility to give you the time to focus on vital care.



4.2 What are we trying to determine with the blood samples?

The average adult human has about 1.2-1.5 gallons of blood. (American Red Cross)

Imagine someone that you know has been in an accident where they are losing blood due to a deep laceration that has severely damaged some blood vessels. Since blood is essential to life it is important to stop the bleeding as fast as possible by means of direct pressure with compresses and or a tourniquet AND to replace the blood that has been lost.

Can we use blood for any donor to replace the lost blood?

The short answer is no and in fact replacing lost blood in our victim with the wrong type can be fatal.

How do we ensure that our victim has the proper (safe) blood for replacement?

We can do simple tests that will identify our victim's blood type. This information is given to emergency services who will then deliver donor blood of the proper type to ensure a safe transfusion to our victim and stabilize their condition until they can reach a hospital.

To better understand theory behind how blood testing works, watch the video clip:

ABO Blood Types Made Easy! <https://www.youtube.com/watch?v=Amn2EWTY2Lk>

Now that you know about the different blood types how to test for them, we will gain practical experience where each of you do a test with simulated blood to identify what type of blood your victim has.

4.3 What will we use to test blood types?

Don't worry! For determining blood types, you will not be required to provide a sample of blood for testing, nor will we use any real blood samples. We will instead use a realistic simulated blood typing kit from Innovative Science (Simulated ABO Blood Typing Kit).



4.4 Blood Type Testing

In class you will practice blood typing to identify all 4 types of simulated blood samples A, B, AB, and O. You will also have to determine the Rh factor of the blood sample. To conserve samples, you will practice in small groups (4 students) where each group will get an undisclosed sample of each of the four types of blood. Individually, you will get to conduct the complete test on one of the samples, then as a group work to correctly identify all the samples given to your group.

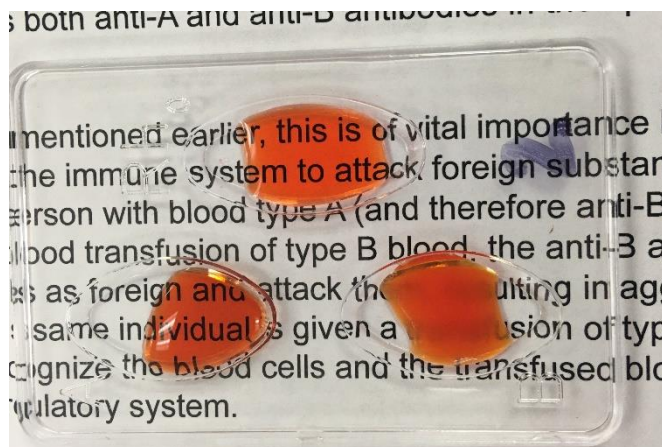
Determining ABO/Rh Blood Type (Innovative Science (Simulated ABO Blood Typing Kit))

Blood type can be easily determined in the lab. To do so, a sample of the blood is placed in a special blood typing tray that contains wells labeled “A”, “B”, and “Rh”. The blood then has anti-serum added to it. The anti-serum has antibodies in it that react against specific antigens on the surface of the red blood cells and cause the sample to agglutinate in the well of the blood typing tray. Like the three labeled wells on the blood typing tray, there are three types of anti-sera: anti-A, anti-B, and anti-Rh. Each is added to the corresponding well on the tray.

Each of the three wells is examined for agglutination and the blood type may be determined depending on which of the wells, if any, show the agglutination reaction (become cloudy). If for example, the blood agglutinates in the presence of anti-A and anti-Rh serum, the blood sample is determined a type A+. If none of the well samples agglutinate, the blood type is O-. See the chart below to help with the analysis of the blood samples:

Blood Type/Antisera Reactions:

Antiserum:	O-	O+	A-	A+	B-	B+	AB-	AB+
Anti-A	(-)	(-)	(+)	(+)	(-)	(-)	(+)	(+)
Anti-B	(-)	(-)	(-)	(-)	(+)	(+)	(+)	(+)
Anti-Rh	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)



In this sample, notice how one of the wells became cloudy and the other two are transparent (you can see through them). You would record a (+) sign for the agglutinated or cloudy sample well and a (-) sign for the other two wells.



Scientific Lab: Blood Type Lab

Time Required: 1 hour

Materials:

- 4 blood typing trays (per group/1 per student)
- 12 toothpicks (per group/4 per student)
- permanent marker (per group)
- pencil (per student)
- a printed paper (to provide contrast to read results)



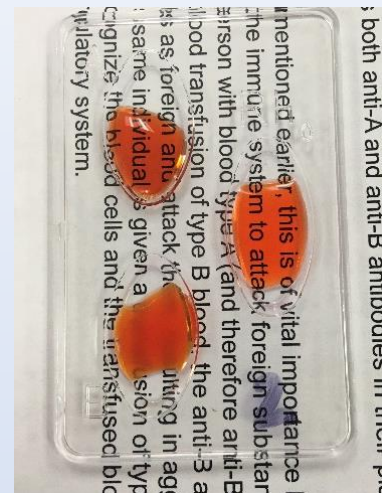
Shared Class Materials:

- Simulated Anti-serum A, B, and Rh
- Simulated blood samples marked “unknown” with #1, #2, #3, or #4



Procedures:

1. Place the 3-well typing tray on your workspace.
2. Use a sharpie to write the number of your unknown on the bottom right corner of your test tray.
3. Add 1 drop of your unknown blood in each of the three wells (A, B, and Rh).
4. Add 2 drops of anti-A serum into well marked “A”
5. Add 2 drops of anti-B serum into well marked “B”
6. Add 2 drops of anti-Rh serum into well marked “Rh”
7. Using a clean tooth pick gently stir the liquid in well “A” for 10-15 seconds.
8. Using a clean toothpick, repeat this process for wells “B” and “Rh”.
9. Determine the blood type of your given sample (see instructions below).
10. Work with your group to check and confirm your analysis.



Continued next page.



Scientific Lab: Blood Type Lab (continued)

Reading your results

- Place your 3-well typing tray on a sheet of paper that has printed material on it. This will provide additional contrast and make it easier to read and interpret your results.
- Examine the sample well marked “A”.
 - Can you easily read the print on the underlying page through the liquid in the well or is the printed material obscured (hard to see or can’t see at all).
- Record your results on the sheet provided to you (table 1 from data analysis section of kit manual).
 - If you can read the printed material (the liquid appears to be transparent), record a “-“ in your results for that well.
 - If you cannot read the printed material (it is blurred or opaque), record a “+“ in your results for that well.
 - When you can see through the liquid, that means there was no reaction between the anti-serum and the sample.
 - A reaction or cloudy well indicates agglutination.
- Repeat this process for the wells marked “B” and “Rh”.

Interpreting your results

“A” well:

- If you recorded a “+” for your “A” well- there are “A” antigens on your unknown blood cells.
- If you recorded a “-” for your “A” well- there are not any “A” antigens on your unknown blood cells.

“B” well:

- If you recorded a “+” for your “B” well- there are “B” antigens on your unknown blood cells.
- If you recorded a “-” for your “B” well- there are not any “B” antigens on your unknown blood cells.

“Rh” well:

- If you recorded a “+” for your “Rh” well- there are “Rh” antigens on your unknown blood cells.
- If you recorded a “-” for your “Rh” well, then there are not any “Rh” antigens on your unknown blood cells.

Antiserum:	O-	O+	A-	A+	B-	B+	AB-	AB+
Anti-A	(-)	(-)	(+)	(+)	(-)	(-)	(+)	(+)
Anti-B	(-)	(-)	(-)	(-)	(+)	(+)	(+)	(+)
Anti-Rh	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)

As a group, record the results to the group samples on the next page.

Blood Typing Analysis:

Agglutination Reactions for each blood sample given to your group:

Group samples:	Anti-A Reaction	Anti-B Reaction	Anti-Rh Reaction
Sample #1			
Sample #2			
Sample #3			
Sample #4			

Blood Type of Each Sample:

Group samples:	Blood Type
Sample #1	
Sample #2	
Sample #3	
Sample #4	



Think About It:

- Based on what you learned in the simulated blood test, what do you think might happen if someone was given the wrong type of blood?
- Why is the O+ blood type the universal donor blood type?
- What blood type can accept any other type of blood?

Module 5: Rover Assembly

There are many rover kits on the market, some already assembled and some that require assembly. STEM SEALs chose to engineer a special robotic rover just for its program. In doing so they were able to utilize local (College) resources and experts to create a unique product that has evolved to meet the purpose of the LAND Challenge. Being able to print 3D or laser cut parts on campus allowed the team to design a custom rover. Allowing you to assemble the rover gives you a more in-depth knowledge and hands-on experience with robotics than an out of the box product.

Let's start the assembly process by attaching the Dc motors and wheels, a motor driver, and power source to allow for simple basic movement of the rover.

5.1 Motor Test

It is important to test the functionality of the motors before attaching them to the chassis. We want to be sure the motors work. Find the two DC gearbox motors and the two wheels from the LAND kit. You will also need the battery case and batteries you that come with the micro:bit for testing.

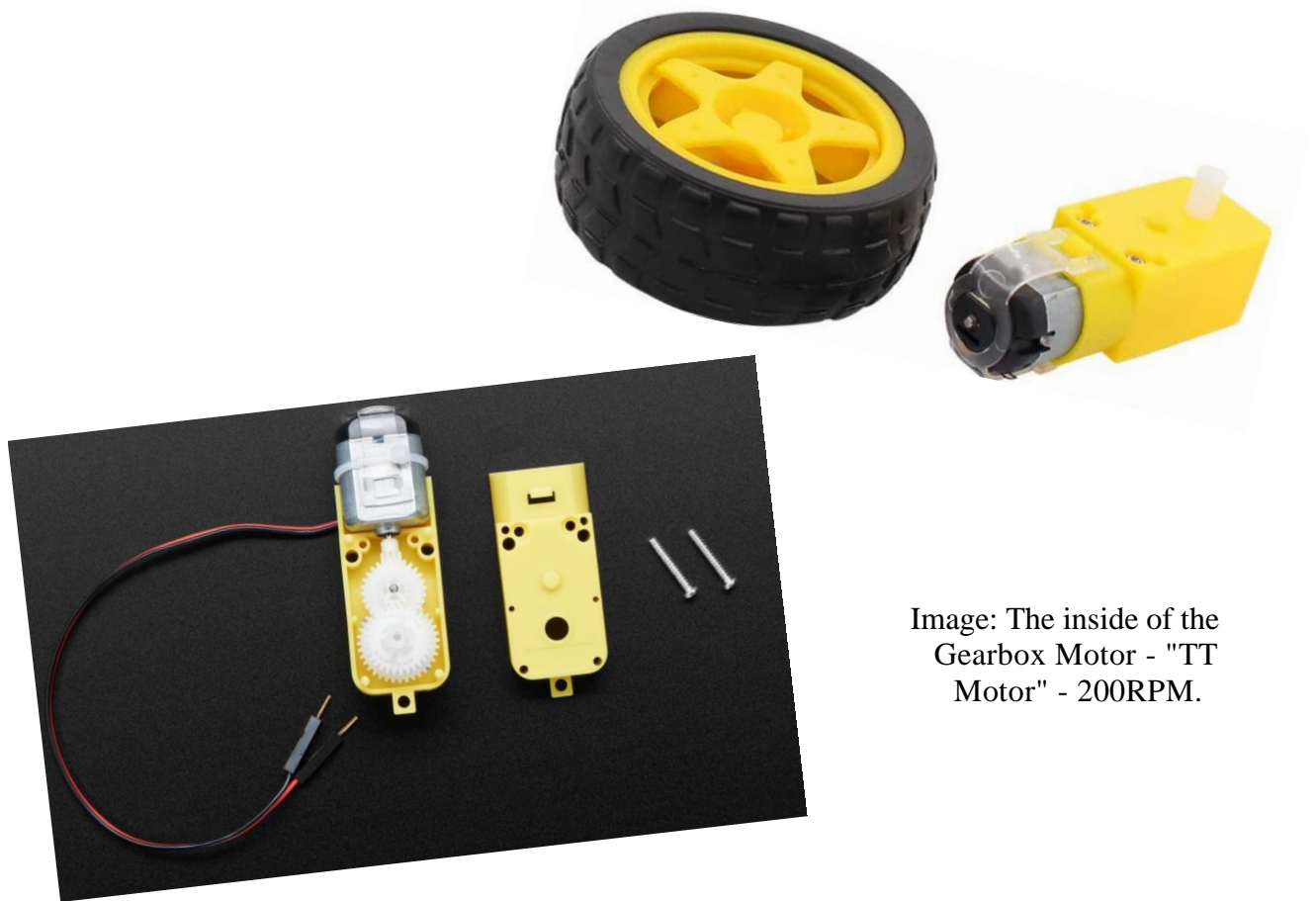


Image: The inside of the Gearbox Motor - "TT Motor" - 200RPM.



Try This: Testing the DC Motors

Follow the instructions at the beginning of the STEM SEALs You Tube video: **LAND Rover Assembly**.

1. Connect a power source to the DC motor. (see video)

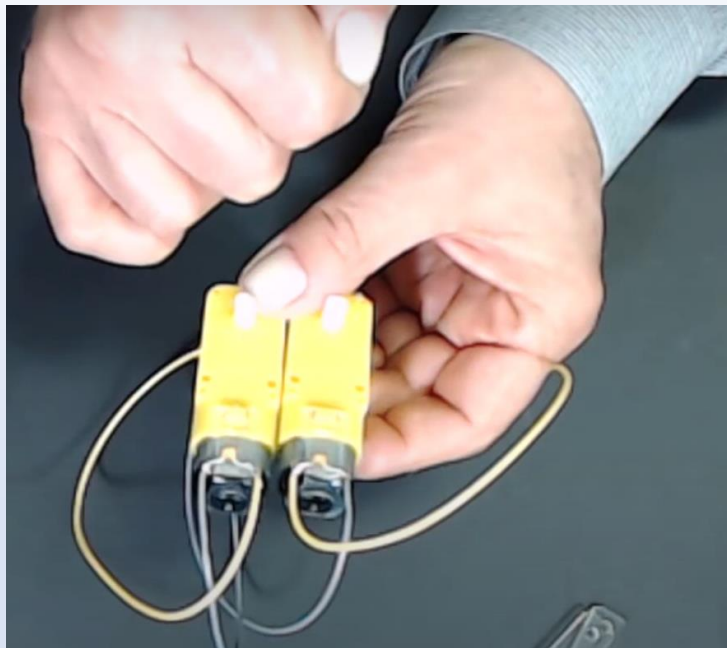
Does the wheel turn?

- Can you tell which direction? Clockwise or counterclockwise? You may need to try it multiple times to determine.

Reverse polarity. (Reverse the red and black wire)

- Which direction does the wheel rotate now?

2. Repeat the same test with the other DC motor and wheel.



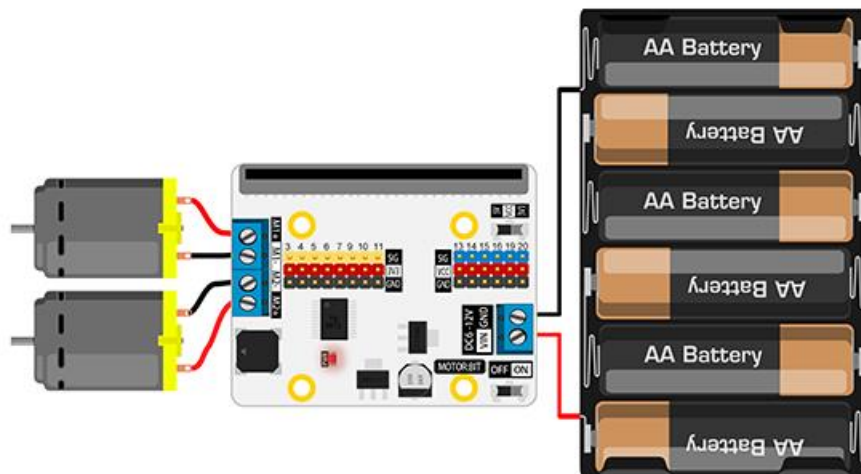
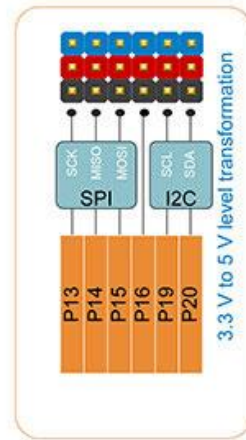
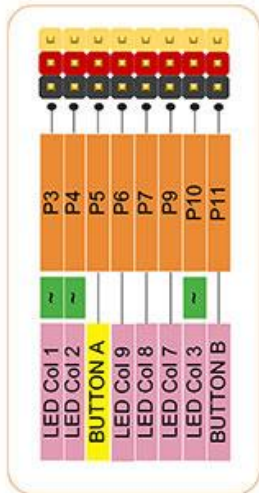
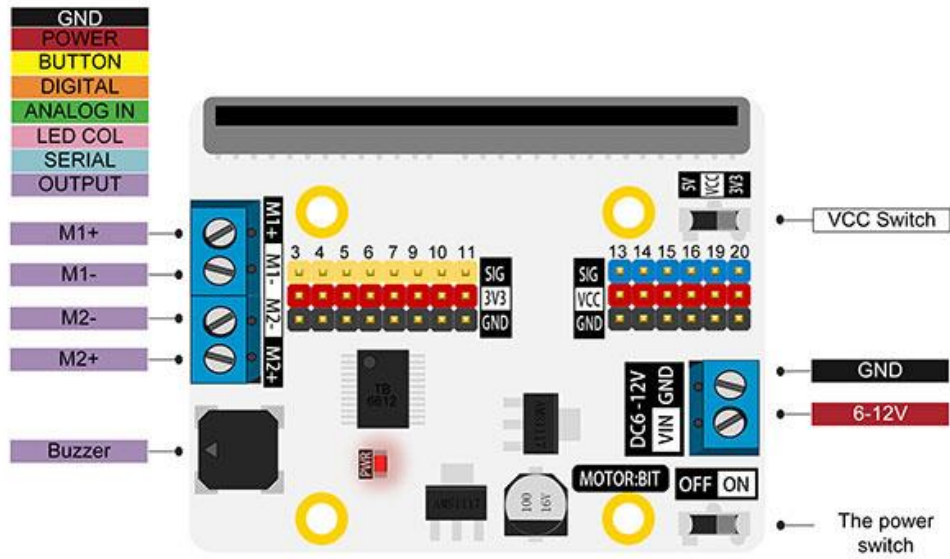
5.2 The Motor Driver

Now that you know that your DC motors are functioning, we can build the basic components of the rover for propulsion. Of course, this will include some type of power source, but it will also need some type of motor driver. The STEM SEALs rover uses an ElecFreaks Motor:bit as the motor driver.

For more information about the Motor:bit refer to this resource:

<https://www.elecFreaks.com/motor-bit-for-micro-bit-motorbit.html>

ELECFREAKS MOTOR:BIT V1.6



5.3 Rover Assembly

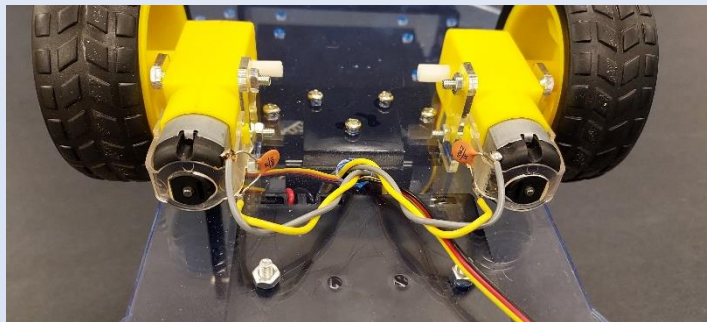
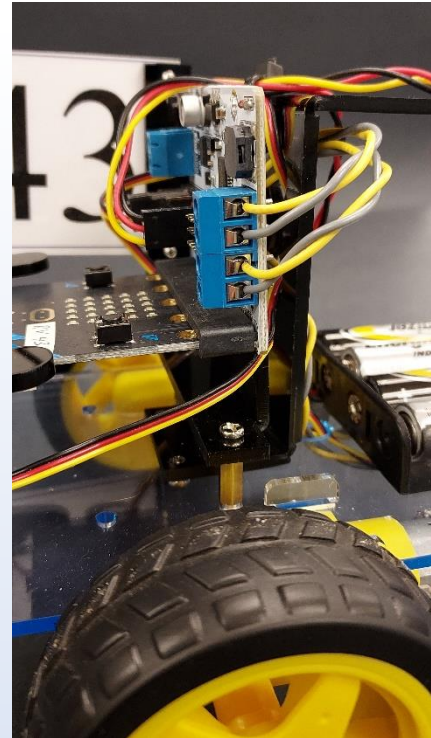
The first part of the rover that will be assembled will allow the rover to move. For this you will need to attach the DC motors and wheel assembly to the rover chassis as well as the front ball caster. A power source will need to be added (battery case and 4 AA batteries). Then some special mounts to hold the motor driver and micro:bit and a wire harness/handle will be added.

Refer to the written directions and the video to assemble the rover.



Assemble: DC Motors, Wheels, Motor:bit, and Micro:bit

1. Refer to the images on pages 10 through 16 to identify the parts needed for this part of the assembly.
2. Watch the STEM SEALs You Tube video: **LAND Rover Assembly**.
3. Step-by step assembly instructions:
 - a. Find the chassis plate.
 - Which side is up?
 - Look for the STEM SEALs logo.
 - b. Assemble the caster wheel.
 - c. Attach the DC motors and wheels.
 - d. Attach the battery case.
 - e. Attach the wire harness/handle.
 - f. Attach the motor:bit and flag mount.
 - g. Connect the wires from the DC motors to the motor:bit.
 - h. Connect the power supply leads to the motor:bit.



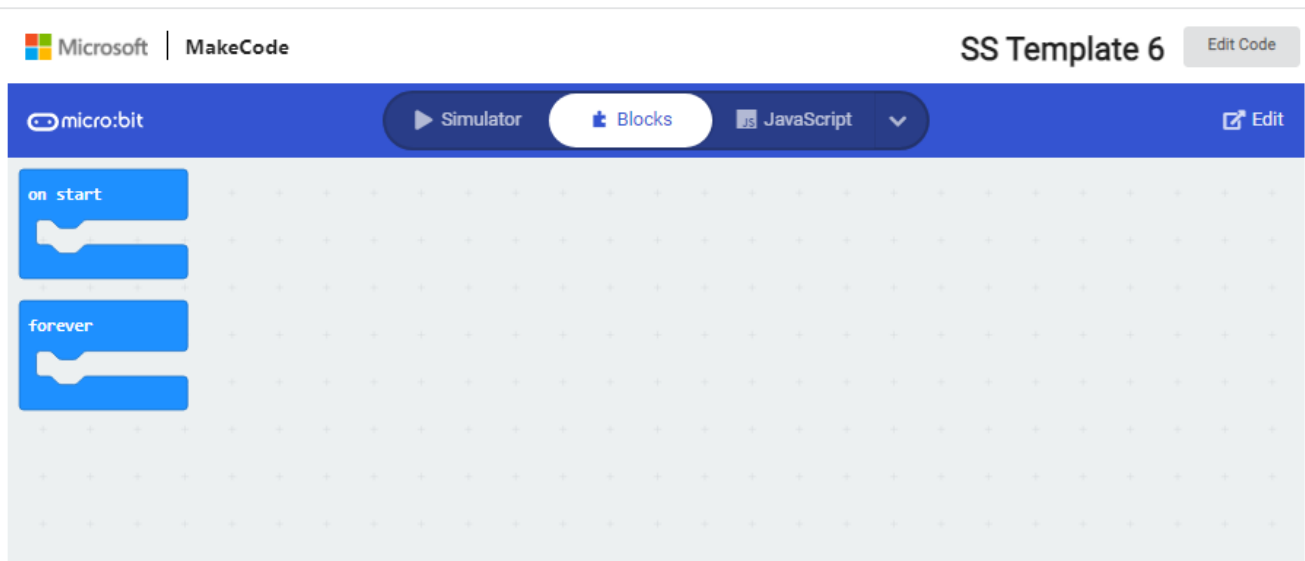
Module 6: Propulsion

Now that the motors have been tested and connected to the motor:bit, it is time to create some code for the micro:bit that will control the rover. The first code will be for **propulsion** which is the mechanism or system used to generate thrust to move the rover forward.

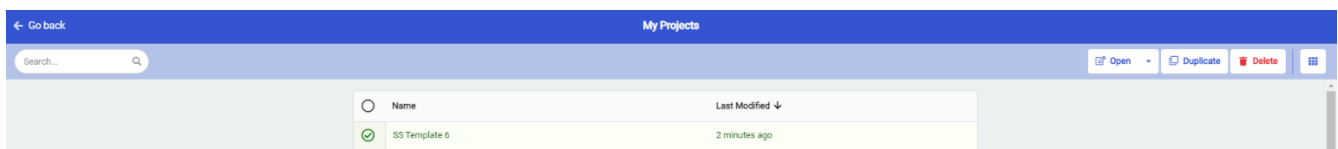
Remember the earlier notes on how you can share programs with others by publishing code in the MakeCode editor? This generates a link that can be shared with others. Special code or blocks have been created by the STEM SEALs team to make controlling the rover easier. To access this code, type the following URL into your computer browser:

SS Template 6: https://makecode.microbit.org/_FMaPxv34CFqs

A new window will appear that looks like this. Click on the “edit” button.



This will open a new project called SS Template 6, in the MakeCode editor. You could rename this and start using it right away but it is suggested to save it in your projects and then duplicate it so you will always have a new blank workspace with this template. To do this go back to the homepage in the editor and click “view all”. Select “SS Template 6” and then in the right-hand corner, the following icons appear: Open, Duplicate, and Delete. Select “duplicate” and a pop-up box will appear asking to rename the newly created duplicate project. Rename the new project “**Propulsion Test 1**”



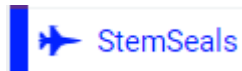
6.1 Propulsion Test



Time to Code: Propulsion Test 1

1. Open MakeCode on your computer, open the “**Propulsion Test 1**” project.

- Make sure the project has a toolbox that is labeled: StemSeals



- If there is not a StemSeals toolbox, go back and follow the directions above.

2. The “**on start**” block:

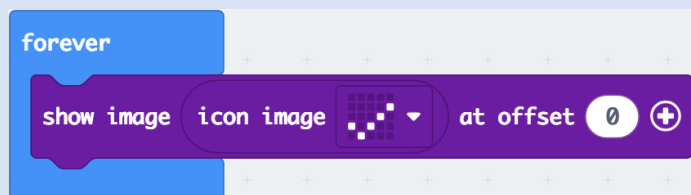
- Drag a "plot x 2 y 2" block from the "Led" toolbox into the "on start" block.
- Place a "play sound giggle until done" block from the "Music" toolbox below it.
- From the "Advanced" blocks area, find the "Pins" toolbox and drag an "analog write pin P0 to 1023" block and drop it at the end of the "on start" block,
 - Change P0 into P1
 - Change the number 1023 to 500.



- Get a "pause (ms) 100" block from the "Basic" toolbox and drop it after the "analog write ..." block.
 - Change the 100 ms into 2000 ms (2 seconds).
 - This pause determines the amount of time the action will last – 2 seconds.
- Duplicate the "analog write ..." block and place it after the "pause ..." block.
 - Duplicate blocks by right clicking on them and then selecting “duplicate” from the drop-down menu.
 - Change the number 500 in that last block to zero.

3. The “**forever**” block:

- Get a "show image ..." block from the ("Advanced") "Images" toolbox and place it into the "forever" block.
 - Get an "icon image" block from the "Images" toolbox and place it inside the "show image ..." block,
 - Change the image of the "icon image" block to a checkmark.



The complete code should look like this:

```
on start
  plot x 2 y 2
  play sound giggle until done
  analog write pin P1 to 500
  pause (ms) 2000
  analog write pin P1 to 0

forever
  show image icon image at offset 0
```



Try This: Propulsion Test 1

Now that you have created some code for the rover, it is time to test it.

1. Connect the micro: bit to the computer using the USB connector.
2. Download the “**Propulsion Test 1**” code to the micro:bit.
3. After downloading, insert the micro: bit into the motor: bit on the rover.
4. Set the rover on the floor and turn the motor:bit ON. Be prepared for the rover to move!
 - Did the rover move? If not press the reset button.
 - Do both wheels turn?

Only the left wheel should have moved because the code only writes to P1.
Let’s see if we can get the right wheel to move.

5. Change the “analog write pin” blocks in the “**on start**” block.
 - Change the P1 in both “analog write...” blocks to P2.
6. Removing the micro:bit from the motor:bit board- always turn the motor:bit OFF first.
7. Download the code to the micro:bit, reinsert it into the motor:bit, and test again.
 - Does the rover move?
 - Which wheel moved?
 - Which direction did it move?

Let’s get both wheels moving at the same time and control the direction!



Time to Code: Propulsion Test 1 (Moving Both Wheels)

1. In the “on start” block:

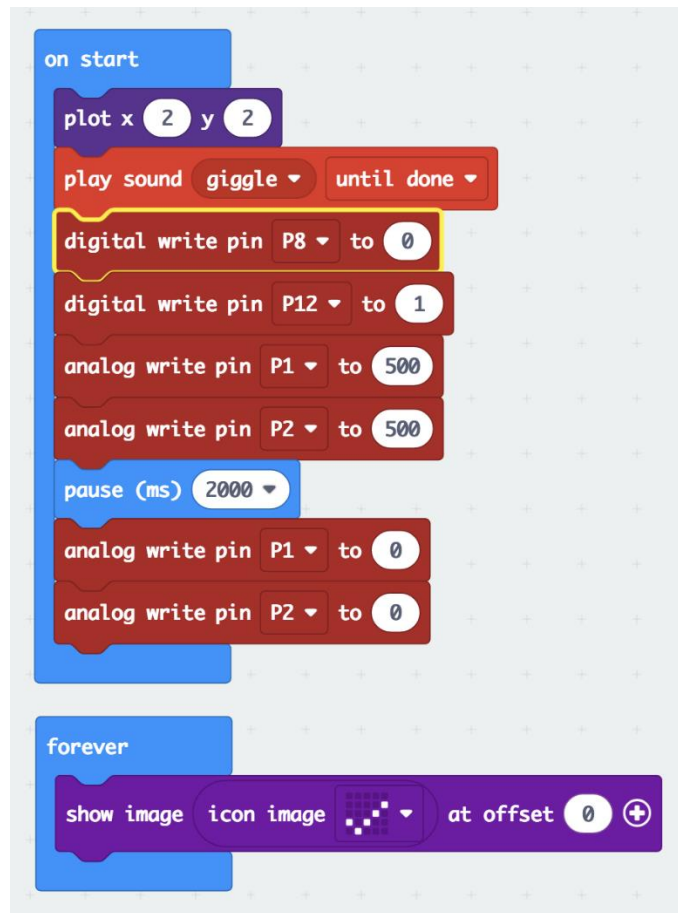
- Get a "digital write pin P0 to 0" block from the "Pins" toolbox ("Advanced") and place it before the first "analog write ..." block.
 - Change pin P0 to P8.
- Duplicate/copy this "digital write ..." block and place it after the block you just placed.
 - Change pin P8 to P12.
 - Change the zero in the "digital write pin P12 to 0" block to 1.
- Duplicate/copy the first "analog write ..." block and place it right after the first “analog write...” block.
- Change pin P2 of the first "analog write ..." block back to P1.
- Copy this new block and place it before the "analog write ..." block at the end of the "on start" block,
 - change the number 500 in this block to zero.

```
on start
  plot x 2 y 2
  play sound giggle until done
  digital write pin P8 to 0
  digital write pin P12 to 1
  analog write pin P1 to 500
  analog write pin P2 to 500
  pause (ms) 2000
  analog write pin P1 to 0
  analog write pin P2 to 0
```

Download the code, upload to the micro:bit, and test:

- Does the rover move?
- Do both wheels turn?
- Which direction does the rover move?

The complete code for “**Propulsion 1**” should look like this:

The image shows a Scratch code editor with a blue 'on start' block containing several blocks: 'plot x 2 y 2', 'play sound giggle until done', 'digital write pin P8 to 0', 'digital write pin P12 to 1', 'analog write pin P1 to 500', 'analog write pin P2 to 500', 'pause (ms) 2000', 'analog write pin P1 to 0', and 'analog write pin P2 to 0'. Below this is a blue 'forever' loop block containing a purple 'show image icon image at offset 0' block.

The code: "**Propulsion Test 1**": <https://makecode.microbit.org/L7HU4b1JaeDC> for forward motion.

This is great! We now have our rover moving forward with both wheels.

How do we make it move backwards?

There is a simple modification to create backwards motion:

- In the “**Propulsion Test 1**” code:
 - Change the zero in the "digital write pin P8 ..." block to 1,
 - Change the 1 in the "digital write pin P12 ..." block to 0.
- Download, upload and test:
 - Does the rover move?
 - Which direction?

We could continue writing code like this for all kinds of motions, forward, backward, turns and spins, but it could be quite confusing keeping the zeros and ones straight.

Let’s try something else!

6.2 Propulsion 2

Let's modify the "Propulsion Test 1" code. Whenever writing code, if you plan to use large portions of the same code as before, it is easier and saves time to modify the previous code as we have been doing. However, in case you want to be able to follow your steps and changes, it is recommended that you save a copy of the previous code "Propulsion Test 1" on your flash drive as well as duplicating the project and renaming in your projects in the MakeCode homepage. This will also give you the STEM SEALs toolbox to use.



Time to Code: Propulsion Test 2

1. Save and duplicate "Propulsion Test 1".
2. Rename the new duplicated project "Propulsion Test 2".
3. In the "on start" block:

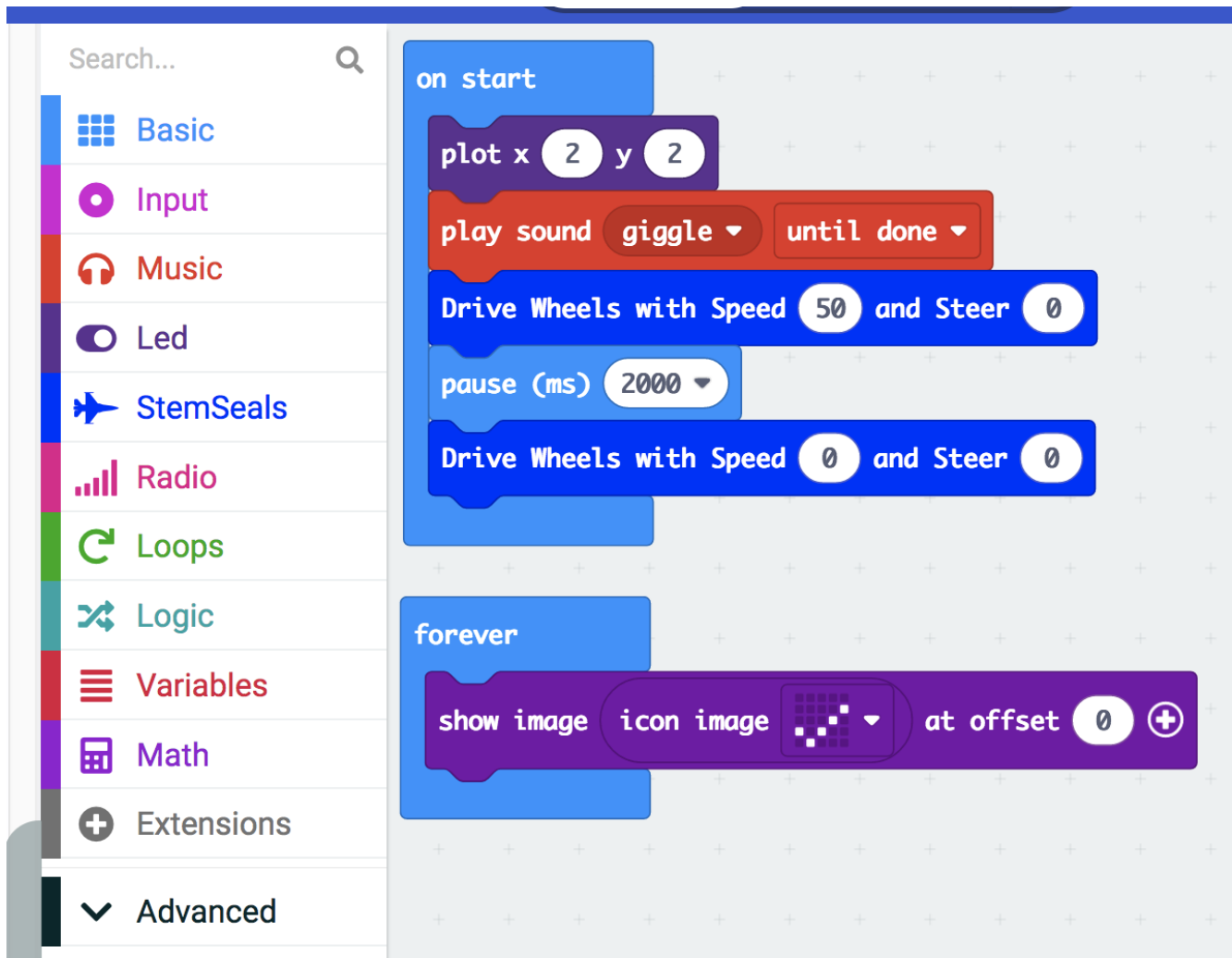
- Delete all "digital write ..." blocks.
- Delete all "analog write ..." blocks
- From the "StemSeals" toolbox, get the "Drive Wheels ..." block and place it before the "pause ..." block .
 - Change the zero after Speed to 50.
- Get another "Drive Wheels ..." block and place it after the "pause ..." block".
- Remember that the pause block determines how long the "Drive wheels with speed 50" command will continue before the next command happens which here turns the speed value back to zero (0). Later you may need to manipulate pauses such as these to get the rover to travel as you wish.

```
on start
  plot x 2 y 2
  play sound giggle until done
  Drive Wheels with Speed 50 and Steer 0
  pause (ms) 2000
  Drive Wheels with Speed 0 and Steer 0
```

Download the code, upload to the micro:bit, and test:

- Does the rover move?
- Do both wheels turn?
- Which direction does the rover move?

Your completed code for “**Propulsion Test 2**” should look like this:



The code: “**Propulsion Test 2**”: <https://makecode.microbit.org/DhmbvsdXIJVT>

6.3 Understanding the Code

In the earlier code, the analog write blocks (P1 and P2) refers to the wheels and 500 determined the speed of the rover. The digital write blocks (P8 and P12) determine the direction.

Therefore, the Speed variable determines how fast the rover moves.

It has two wheels, and at first you would think

$$P1 = 1023, P2 = 1023, P8 = 0 \text{ and } P12 = 1 \text{ would be Speed} = 100 (\%).$$

That is correct, but the rover can move in more ways than just forward:

- forward $P8 = 0 \text{ and } P12 = 1$
- backward $P8 = 1 \text{ and } P12 = 0$

- turn (make a turn)
 - forward P8 = 0 and P12 = 1 but different values for P1 and P2
 - backward P8 = 1 and P12 = 0 but different values for P1 and P2
- spin
 - left P8 = 1 and P12 = 1
 - right P8 = 0 and P12 = 0

In the STEM SEALs “Drive Wheels” block, we can control both the **speed** and **steer**.

We can give the **Speed** variable a sign, and defined Speed > 0 as forward motion, Speed < 0 as reverse motion.

Pin P1 controls the speed of the left wheel, P2 the right wheel.

We can control turns and spins with a second variable we call **Steer**.

With Steer = 0, P1 = P2.

With Steer < 0, the rover turns or spins left.

With Steer = -50 (%) the left wheel should be stopped.

With Steer = -100 (%) the left wheel should rotate backwards with the same speed as the right wheel rotates forward, and vice versa at positive Steer values.

i.e.: (what the directions are behind the code)

if Speed > 0:

```

if Steer < -50 then P8 = 1, P12 = 1, P1 = [abs(Steer) - 50] * abs(Speed) * 1023/5000
                    P2 = abs(Speed) * 1023/100
if Steer < 0 (>=-50) then P8 = 0, P12 = 1, P1 = [50 - abs(Steer)] * abs(Speed) * 1023/5000
                    P2 = abs(Speed) * 1023/100
if Steer < 50 (>0) then P8 = 0, P12 = 1, P1 = abs(Speed) * 1023/100
                    P2 = [50 - abs(Steer)] * abs(Speed) * 1023/5000
if Steer > 50 then P8 = 0, P12 = 0, P1 = abs(Speed) * 1023/100
                    P2 = [abs(Steer) - 50] * abs(Speed) * 1023/5000

```

else if Speed <= 0:

```

if Steer < -50 then P8 = 0, P12 = 0, P1 = [abs(Steer) - 50] * abs(Speed) * 1023/5000
                    P2 = abs(Speed) * 1023/100
if Steer < 0 (>=-50) then P8 = 1, P12 = 0, P1 = [50 - abs(Steer)] * abs(Speed) * 1023/5000
                    P2 = abs(Speed) * 1023/100
if Steer < 50 (>0) then P8 = 1, P12 = 0, P1 = abs(Speed) * 1023/100
                    P2 = [50 - abs(Steer)] * abs(Speed) * 1023/5000
if Steer > 50 then P8 = 1, P12 = 1, P1 = abs(Speed) * 1023/100
                    P2 = [abs(Steer) - 50] * abs(Speed) * 1023/5000

```

Would you like to see the details of code that the STEM SEALs team created in the MakeCode Editor?

Remember you can toggle the workspace to Java Script. Go to the “**Propulsion 2**” project and switch to Java Script. Notice that you see something like this for the “Drive Wheels” block:

`StemSeals.DriveWheels(0, 0)`

Let's look deeper into the code. Stay in the Java Script mode and look under the simulated micro:bit in the MakeCode workspace, there is a drop-down menu : Explorer. Click on "custom.ts". Now in the workspace you will see all the code behind the special STEM SEALS "Drive Wheels" block.

Look starting at line 55.

Does this look familiar to the explanation above of how the rover can move?

Wow!

There is a lot of directions or coding that went into creating what appears to be a very simple block.

Learning coding using blocks can make learning the process of how to organize code much simpler but remember there is much more that goes on behind those graphic blocks.

To return to the block view, click on the "Blocks" tap at the top of the workspace.

6.4 Propulsion 3

Now that we can make the rover move forwards and backwards and we have a simpler way to control both the speed and steer, it is important to know how fast your rover is traveling or how far it will travel in a specific amount of time.

To test this, we will use Buttons A and B to increase or decrease the speed in increments and the Logo Button/icon to allow the rover to drive for a fixed amount of time.

Let's modify our previous code for these changes.



Tip: Modifying Code

Remember MakeCode automatically saves any changes you make to a project. Therefore, if you want to make sure you have all versions of the code you should always save (Share) the URL link to a location for safe keeping or duplicate and create a new project on the MakeCode browser. When you duplicate a project, it will copy all the code from one project to another.



Time to Code: Propulsion Test 3

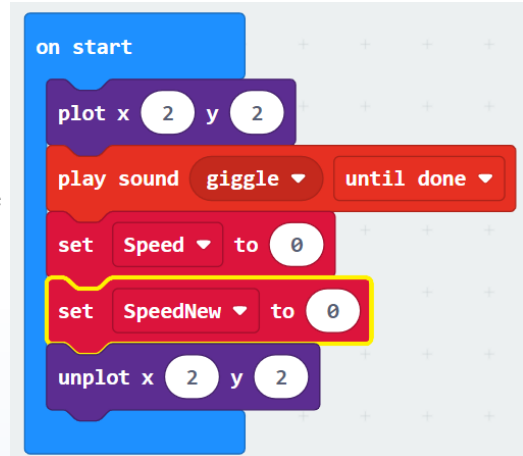
1. Beginning with the **Propulsion Test 2** code, rename it “**Propulsion Test 3**”.

2. Make some new variables:

- Create a variable and name it “Speed”.
- Create a variable and name it “SpeedNew”.

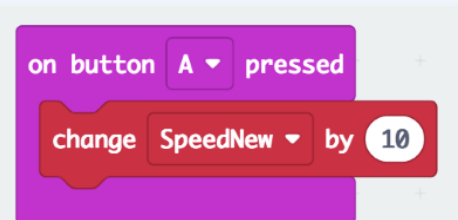
3. Modify the “**on start**” block:

- Drag both of the “Drive Wheels” blocks and the “pause” block to a blank area of the workspace and leave it for use later (it will look ghosted) .
- Drag a “set Speed to 0” block form “Variables” and place it after the “play sound giggle” block.
- Copy this block and paste it right after the one you copied.
 - Change the variable in this new “set” block to “SpeedNew”.
- Drag an “unplot x 0 y 0” block from the “Led” toolbox and place it after the “set SpeedNew to 0” block.
 - Change the x 0 y 0 to x 2 y 2.



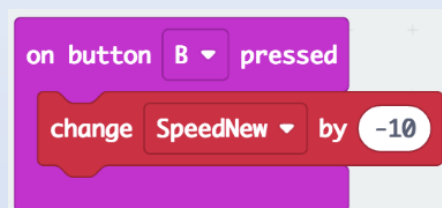
4. Create an “**on button A pressed**” block:

- Drag a “on button A pressed” block from the “Input” toolbox and place it in blank space on the workspace.
- Drag a “change SpeedNew by 1” block from “Variables” and place it in the “on button A” block.
 - Change the number 1 in this block to a 10.
 - This will allow you to change the speed of the rover by increments of 10 by pressing the A button.



5. Create an “**on button B pressed**” block:

- Copy/duplicate the entire “on button A pressed” block.
 - Notice the duplicated block looks ghosted. This is because you can only have one “button A” block.
 - Change the “A” in the drop-down menu at the top of this block to “B”.
 - Now the block should look normal.
 - Change the 10 in the “set SpeedNew to” block to a negative 10 (-10).
 - This will allow you to decrease the speed of the rover in increments of 10.



Continued next page.



Time to Code: Propulsion Test 3 (continued)

6. Create a “on logo pressed” block:

- Drag a “on logo pressed” block from the “Input” toolbox.
- Place it in an empty area of the workspace.
- Drag a “repeat 4 times” block from the “Loops” toolbox and place in the “on logo pressed” block.
 - Get a “play tone Middle C for 1 beat” block from “Music” and place it inside the “repeat...” block.
 - Change the Middle C to a Middle F note.
 - Change the 1 beat to ½ beat.
 - Insert a “pause (ms) 100” block after the “play tone...” block.
 - Change the 100 (ms) to 200 (ms).
- Drag a “set Speed to 0” block from “variables **after** the “repeat...” block and **not inside it**.
 - Instead of the set to zero (0), drag a “SpeedNew” variable into where the zero is.
- Drag both of the “Drive wheels...” blocks and the “pause” block that you set aside from the “on start” block and place them after the “set Speed...” block.
- Move the “show image icon image...” with the checkmark from the “forever” block to the very end of the “on logo pressed” block.
- Add a “pause (ms) 1000” block after this.
- Then place an “unplot x 4 y 1” block.

The “on logo pressed” block should look like this:

```

on logo pressed
  repeat 4 times
    do
      play tone Middle F for 1/2 beat
      pause (ms) 200
  set Speed to SpeedNew
  Drive Wheels with Speed Speed and Steer 0
  pause (ms) 2000
  Drive Wheels with Speed 0 and Steer 0
  show image icon image at offset 0
  pause (ms) 1000
  unplot x 4 y 1
  
```

Continued next page.

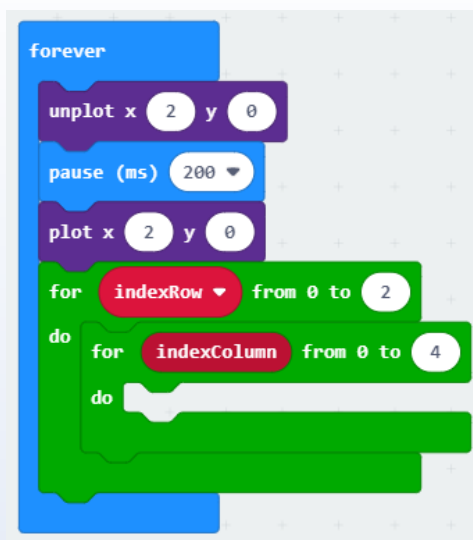


Time to Code: Propulsion Test 3 (continued)

7. The “forever” block: (it should be empty)

- Place an “unplot x 0 y 0”, change the x 0 y 0 to x 2 y 0.
- Place a “pause (ms) 200” block after the “unplot...” block.
- Then place an “plot x 2 y 0” after the “pause...” block.
- Drag a “for index from 0 to 4” block from the “Loops” toolbox.
 - Right click on the “index” variable and rename it “indexRow”.
 - Drag another “for index from 0 to 4” block and place it in the opening of the first “index...” block.

The forever block should look like this so far:



Now we want to add a logic statement:

If “SpeedNew” is greater than (>) [$[\text{“indexColumn”} + (5 \times \text{“indexRow”})]$ integer $\times 10$ **then**

“plot x (4 – “indexColumn”) y (4 – “indexRow”)”

Else “unplot x (4 – “indexColumn”) y (4 – “indexRow”)”

Notice there are some mathematic operations within others.

It is important when setting up these blocks, pay attention to these operations.

Let’s continue with the “forever” block and add this statement.

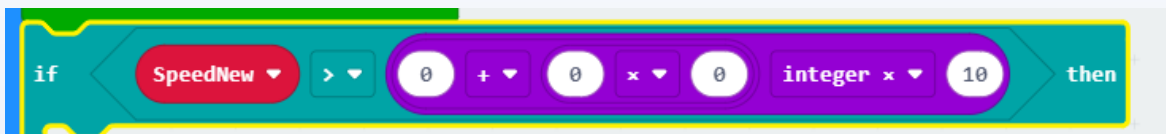
Continued next page.



Time to Code: Propulsion Test 3 (continued)

7. The “forever” block: (continued)

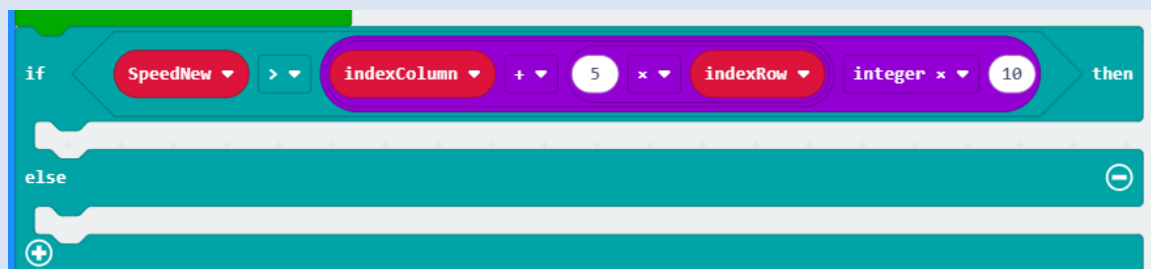
- From the “Logic” toolbox drag a “if true then else” block.
 - Place it inside the “for indexColumn...” block.
 - Replace the “true” in the “if then ...” block with a “ $0 < 0$ ” comparison block from the “Logic” toolbox.
 - o Drag the comparison block over the “true” in the “if then ...” block.
 - Get a “SpeedNew” variable from “Variables” and drop it where the first zero (0) is in the “ $0 < 0$ ” comparison block.
 - Change the sign of the “ $0 < 0$ ” block to greater than ($>$).
 - Grab a “square root 0” block from the “Math” toolbox.
 - o Place it in the second zero (0) of the comparison block.
 - o Change the “square root” to “integer multiplication” by using the drop-down menu of the block.
 - o Get a “ $0 + 0$ ” block from “Math” and place it over the first zero (0) in the “integer multiplication” block.
 - o Get a “ 0×0 ” block from “Math” and place it over the second zero (0) in the “integer multiplication” block.



Notice how the mathematical operation blocks are nested inside of each other according to the order of operations needed. This is important.

- Complete the “Math” blocks by adding the following:
 - o Add the variable “indexColumn” in the over the first zero (0).
 - o Change the second zero (0) to a 5.
 - o Add the variable “indexRow” over the third zero (0).
 - o Change the last zero to 10.

The “if then else” block should now look like this:



Continued next page.



Time to Code: Propulsion Test 3 (continued)

7. The “forever” block: (continued)

- Now drag a “plot x 0 y 0” block from the “Led” toolbox and place it in the first opening of the “if SpeedNew then...” block.
 - o Get a “0 – 0” block from “Math” and put it over the first zero (0) in the “plot...” block.
 - o Change the first zero (0) in the math block to a 4.
 - o Drag a variable “indexColumn” over the second zero (0) in the “4 – 0” block.
 - o Copy this whole math block and place it in the y 0 location of the “plot...” block.
 - o Change the “indexColumn” to “indexRow”
- Now drag a “unplot x 0 y 0” block from the “Led” toolbox and place it in the second opening of the “if “SpeedNew then...” block.
 - o Copy and paste both subtraction blocks from the “plot...” block above and put in the “unplot...” block.

The “if then else” block should now look like this:

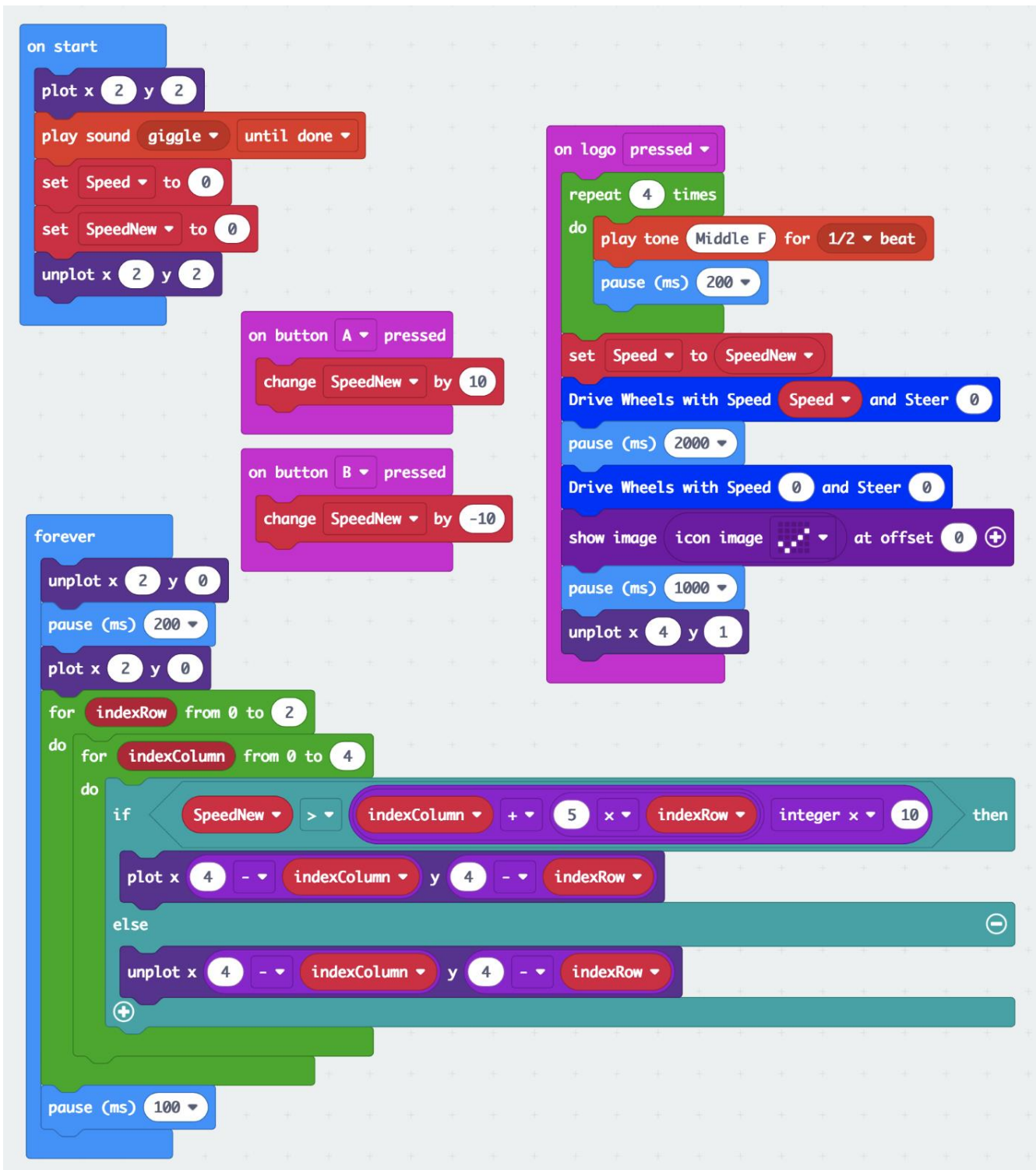


- Finally, add a “pause (ms) 100” at the very end of the “forever” block.

Let’s test this new code!

See the completed code on the next page and check your work before downloading and testing.

Your completed code for “**Propulsion Test 3**” should look like this:



The image shows a Scratch script for a Micro:bit project. The script is organized into several sections:

- on start:** A sequence of blocks including 'plot x 2 y 2', 'play sound giggle until done', 'set Speed to 0', 'set SpeedNew to 0', and 'unplot x 2 y 2'.
- on button A pressed:** A block to 'change SpeedNew by 10'.
- on button B pressed:** A block to 'change SpeedNew by -10'.
- on logo pressed:** A 'repeat 4 times' loop containing:
 - 'play tone Middle F for 1/2 beat' followed by 'pause (ms) 200'.
 - 'set Speed to SpeedNew'.
 - 'Drive Wheels with Speed Speed and Steer 0' followed by 'pause (ms) 2000'.
 - 'Drive Wheels with Speed 0 and Steer 0'.
 - 'show image icon image at offset 0' followed by 'pause (ms) 1000'.
 - 'unplot x 4 y 1'.
- forever:** A loop containing:
 - 'unplot x 2 y 0', 'pause (ms) 200', and 'plot x 2 y 0'.
 - A nested loop for 'indexRow from 0 to 2' and 'indexColumn from 0 to 4'. Inside, an 'if' statement checks if 'SpeedNew > (indexColumn + 5 * indexRow) integer * 10'. If true, it 'plot x 4 - indexColumn y 4 - indexRow'. If false, it 'unplot x 4 - indexColumn y 4 - indexRow'.
 - 'pause (ms) 100'.

The code link: "Propulsion Test 3": <https://makecode.microbit.org/LV3LoeHmuFbH>

Now that you have checked your code, let's test it out.



Try This: Propulsion Test 3

1. Download the “**Propulsion Test 3**” code, upload it to the micro:bit.
2. Insert the micro:bit into the rover and turn on the motor:bit.
3. After the initial “giggle” sound and illumination of the middle LED, is the “rear” LED blinking?
 - Note: What is referred to as the rear of the rover is the top of the micro:bit! Therefore, the LED that is blinking is in row zero of the micro:bit LED display. It makes more sense to look at the rover display from the rear. This also makes row 4 the top. This explains the “4-index” calculations in the “forever” block.
4. Push button A:
 - What happens? (Does a LED appear at the “top” or forward part of the micro:bit)
 - Press button A again.
 - Press it multiple times.
 - Each newly appearing LED represents the increasing SpeedNew variable from 0 to 10 to 20 to 30 and so on.
5. Push button B:
 - What happens? (Does an LED now disappear?)
 - Press button B again.
 - Conclusion: We can look at the LEDs on the display to determine the SpeedNew variable.
 - However, the rover does not move yet!
6. Press the reset button on the underside of the rover. This will reset the sequence and there should be no speed LEDs showing.
7. Press button A once. (There should be one LED on besides the blinking one at the rear.)
8. Set the rover on the floor and touch the micro:bit logo.
 - Does the rover move?
 - You should hear 4 beeps and the DC motors engaging but not moving.
9. Keep pressing button A once to advance/change the speed and then touch the logo.
 - Eventually the rover should move.
 - Continue this process of pressing button A and then the logo.
 - Does the speed of the rover reflect what you would expect?
10. Keep pressing button A until LEDs appear on the third row or middle row of the display.
 - Does the rover move?
 - Each LED represents an increase in speed or really power by 10. Once the LEDs represent over 100, the motors cannot do it.
11. Press button B until there are only 5 LEDs on the display.
 - This represents 50% power.
 - Press the logo. Does the rover move?
12. Press button B repeatedly, even after all the LEDs are off. Then press the logo.
 - Does the rover move in reverse? Can you explain why?

6.5 Propulsion 4

Now we can control the movement of the rover and the power (speed) with which it moves. We also have a handy way to view the forward movement using the LED display. Understanding how your rover moves will be important for the competition. The next coding exercise will allow you to see both the forward and reverse movement on the LED display.



Time to Code: Propulsion Test 4 (Optional)

1. Beginning with the “**Propulsion Test 3**” code, rename it “**Propulsion Test 4**”.
2. Modify the “if then else” block” located in the “**forever**” block:
 - Drag the “SpeedNew variable from the “if then...” block and set it in the workspace.
 - From “Math” get and “ absolute of)” block and place where the “SpeedNew” variable was in the “if then...” block.
 - Now Place the “SpeedNew” variable over the zero (0) in the “absolute” block.
3. Add a new “if then else” block to the “**forever**” block:
 - Get a new “if then else” block from “Logic” and place it at the beginning of the “forever” block.
 - Get a “ $0 < 0$ ” comparison block and drop it over the “true” in the new “if then...” block.
 - Drag a “SpeedNew” variable into the place of the first zero (0) in the comparison block.
 - Click on the “+” sign at the bottom of this block to expand “else” part.
 - Drag the “unplot...” block that is now below this “if then...” block into the “else” part of the “if then...” block
 - Notice how all the remaining blocks moved with the “unplot...” block – this is okay.
 - You want the first three blocks in the “else” part.
 - Grab the “indexRow...” block and the remaining blocks and place them now at the end of the “if then...” block.

```
forever
  if SpeedNew < 0 then
  else
    unplot x 2 y 0
    pause (ms) 200
    plot x 2 y 0
  for indexRow from 0 to 2
  do for indexColumn from 0 to 4
  do
    if absolute of SpeedNew > indexColumn + 5
    plot x 4 - indexColumn y 4 - indexRow
    else
    unplot x 4 - indexColumn y 4 - indexRow
  do
  pause (ms) 100
```

Continued next page.



Time to Code: Propulsion Test 4 (Optional)

4. Continue with the “forever” block to complete the new “if then...” block:
 - Get an “index from...” block from “Loops” and place it in the open slot in the “if then...” block.
 - o Change the “from 0 to 4” to “0 to 2”.
 - Get a “pause (ms)” block and place it right after the “for index...” block.
 - o Change the 100 to 200.
 - Duplicate the “index from...” block and place it after the pause block.
 - Get an “unplot ...” from “LED” and place it into the first “for index...” block.
 - o Get a “0 - 0” block from “Math” and place it in the x 0 space.
 - o Replace the first zero in the subtraction block with a 3.
 - o Get an “index” variable and place over the second zero in the subtraction block.
 - Get a “plot...” block from “LED” and place it in the second “for index...” block.
 - o Duplicate the math block “3 - index” from the “unplot...” block and place it in this “plot...” block.

The completed “forever” block should now look like this:

```
forever
  if SpeedNew < 0 then
    for index from 0 to 2
      do
        unplot x 3 - index y 0
    pause (ms) 200
    for index from 0 to 2
      do
        plot x 3 - index y 0
  else
    unplot x 2 y 0
    pause (ms) 200
    plot x 2 y 0
  for indexRow from 0 to 2
    do
      for indexColumn from 0 to 4
        do
          if absolute of SpeedNew > 5 * indexRow * 10 then
            plot x 4 - indexColumn y 4 - indexRow
          else
            unplot x 4 - indexColumn y 4 - indexRow
        pause (ms) 100
```

Continued next page.



Time to Code: Propulsion Test 4 (Optional)

5. Modify the “on logo pressed” block:

- Get a new “if then else” block from “Logic” and place it at the very beginning of the “on logo pressed” block.
- Drag the “repeat 4 times” block and everything after it (will move automatically) and set it into the “else” part of the new “if ...” block.
- Drag the “Set Speed to SpeedNew” block back out and drop at the very beginning before the “if ...” block.
- Drag the second “Drive Wheels ...” block back out and drop after the “if ...” block.
- Get a Boolean “or” block from “Logic” and drop over the “true” in the “if ...” block.
- Copy the “SpeedNew < 0” comparison block from the top of the “forever” block, paste and drop it in the first place before the “or” of the new Boolean block.
 - change the SpeedNew variable in there to Speed.
 - change the zero in there to negative 100.
- Copy this “Speed < -100” comparison block, paste and drop it over the place after the “or” in the “if ...” block.
 - change the less than sign to a greater than sign,
 - change the number -100 in that last comparison block to 100.
- Add a “play tome Middle C ...” block from “Music” into the first part of the “if ...” block.
 - change the “1 beat” to “1/16 beat”,
 - copy this block, place it right after, and change the Middle C to Low A#,
 - copy this block, place it right after, and change the Low A# to Low A,
 - copy this block, place it right after, and change the Low A to Low G,
 - copy this block, place it right after, and change the Low G to Low F,
 - change the “1/16 beat” of that last block to “1/2 beat”.

The completed “on logo pressed” block should look like this:

```
on logo pressed
  set Speed to SpeedNew
  if Speed < -100 or Speed > 100 then
    play tone Middle C for 1/16 beat
    play tone Low A# for 1/16 beat
    play tone Low A for 1/16 beat
    play tone Low G for 1/16 beat
    play tone Low F for 1/2 beat
  else
    repeat 4 times
      do
        play tone Middle F for 1/2 beat
        pause (ms) 200
      Drive Wheels with Speed Speed and Steer 0
      pause (ms) 2000
    Drive Wheels with Speed 0 and Steer 0
  show image icon image at offset 0
  pause (ms) 1000
  unplot x 4 y 1
```

The completed code for "Propulsion Test 4" should look like this:

```
on start
  plot x 2 y 2
  play sound giggle until done
  set Speed to 0
  set SpeedNew to 0
  unplot x 2 y 2

on button A pressed
  change SpeedNew by 10

on button B pressed
  change SpeedNew by -10
```

```
on logo pressed
  set Speed to SpeedNew
  if Speed < -100 or Speed > 100 then
    play tone Middle C for 1/16 beat
    play tone Low A# for 1/16 beat
    play tone Low A for 1/16 beat
    play tone Low G for 1/16 beat
    play tone Low F for 1/2 beat
  else
    repeat 4 times
      do
        play tone Middle F for 1/2 beat
        pause (ms) 200
      Drive wheels with Speed Speed and Steer 0
      pause (ms) 2000
    Drive wheels with Speed 0 and Steer 0
    show image icon image at offset 0
    pause (ms) 1000
    unplot x 4 y 1
```

```
forever
  if SpeedNew < 0 then
    for index from 0 to 2
      do
        unplot x 3 - index y 0
    pause (ms) 200
    for index from 0 to 2
      do
        plot x 3 - index y 0
  else
    unplot x 2 y 0
    pause (ms) 200
    plot x 2 y 0
  for indexRow from 0 to 2
    do
      for indexColumn from 0 to 4
        do
          if absolute of SpeedNew > indexColumn + 5 * indexRow integer * 10 then
            plot x 4 - indexColumn y 4 - indexRow
          else
            unplot x 4 - indexColumn y 4 - indexRow
        pause (ms) 100
```

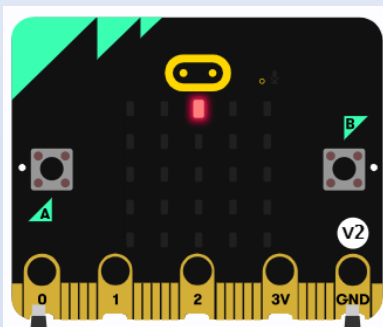
The code link: "Propulsion Test 4": https://makecode.microbit.org/_gU1TwpRcTjky



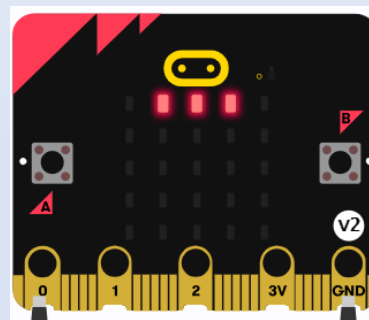
Try This: Propulsion Test 4

1. Download the “**Propulsion Test 4**” code, upload it to the micro:bit.
2. Insert the micro:bit into the rover and turn on the motor:bit.
3. After the initial “giggle” sound and illumination of the middle LED, is the “rear” LED blinking?
4. Press button A four times, then touch the logo.
 - Does it drive ?
 - Which direction?
5. Press button A until there are eleven speed LEDs showing.
 - Does it drive?
 - Does it make a sound?
6. Press the reset button to clear the speed setting.
7. Press button B until there are four LEDs and three Blinking LEDs in the rear (this represents a minus sign).
 - Does it drive?
 - Which direction?
8. Press button B until there are eleven speed LEDs showing.
 - Does it drive?
 - Does it make a sound?

As you should have noticed, there will be one blinking LED in the rear for positive speed / forward motion, and there will be three blinking LEDs in a row in the rear to indicate negative speed / reverse motion.



Positive Speed



Negative Speed

6.6 Propulsion Test (Speed and Distance)

Now that we have created a program that allows the rover to move in both forward and reverse directions, it is important to collect some data on how the speed of the rover affects the distance it will travel in a given time. Knowing this will help you plan for courses that you want the rover to travel and to know what variables to change to get the results you want.

Let's conduct a rover propulsion test!



Data Collection Lab: Propulsion Test

Time Required: 1 hour

Materials:

- Computer Lab (any computers, tablets okay, no cell phones)
- Assembled STEM SEALs LAND robot (partially assembled okay: motors on chassis, battery case, motor:bit and micro:bit)
- four AA batteries
- USB-to-Micro-B-USB cable
- browser on the computer, Fire Fox preferred (some things don't work on Chrome!)
- an open, flat floor space (classroom, hallway)
- masking tape
- meter stick

Prerequisites:

- Students are familiar with programming the Microbit and its LED display.
- Students have (partially) assembled their 2WD rover.
- Rover micro:bit with loaded the “**Propulsion Test 4**” code.
- Check if the “pause (ms) ...” block after the “Drive Wheels with Speed Speed and Steer 0” block in the “on logo pressed” block makes a 2 second pause! (For the first test, for the second you change this block to “pause (ms) 4000”.)

Continued next page.



Data Collection Lab: Propulsion Test

Propulsion Test (2 sec)

1. Set the robot at a place where you can mark the starting position.
2. Turn it on.
3. Push button A, wait and observe if it moves: it shouldn't!
You should see one LED on in the first (front) row.
4. Push the logo button, wait and observe if it moves: it probably does not: Speed = 10 means 10 % power to the DC motors, which is not enough to make them even spin a little.
5. Enter 0 (for zero centimeters) in the "Distance" column and the row for Speed = 10 of Data Table 1.
6. Push A again, the LEDs should indicate Speed = 20.
7. Push the logo button, measure the distance traveled, and enter the value in Data Table 1.
8. Push A again, confirm: 3 LEDs, push the logo, measure the distance traveled, and enter in Data Table 1.
9. Repeat increasing the Speed variable and measure after each time how far the rover moved.
10. Return the rover back to the marked starting position (and don't move the meter stick) to make the distance measurement reproducible.
11. After each time, write down in Data Table 1 how far the rover moved until the table is complete.
12. Set the robot back into its starting position.

Data Table 1 (2 Seconds)

Test Run Number	Speed Display	Speed Variable	Distance / cm
1	1	10	
2	2	20	
3	3	30	
4	4	40	
5	5	50	
6	6	60	
7	7	70	
8	8	80	
9	9	90	
10	10	100	

Continued next page.



Data Collection Lab: Propulsion Test

Propulsion Test (4 sec)

We expect the robot to move twice as far when the motors are on for twice the time. Let's check if this is true:

1. Connect the microbit to the computer and change the number in the "pause (ms) 2000" block to 4000; this is 4 seconds. Leave everything else the same.
2. Disconnect and set robot down at the start position and turn it on.
3. Perform the same procedure as for the previous test:
Increase Speed with button A, press the logo, measure and write down the distance traveled in Data Table 2.
4. Repeat until Data Table 2 is completed.
 - Does the rover move further?
 - Are the values in the "Distance" column of Data Table 2 larger than corresponding numbers in Table 1?
 - Are they twice as big?
 - All of them?

Data Table 2 (4 Seconds)

Test Run Number	Speed Display	Speed Variable	Distance / cm
1	1	10	
2	2	20	
3	3	30	
4	4	40	
5	5	50	
6	6	60	
7	7	70	
8	8	80	
9	9	90	
10	10	100	

Continued next page.



Data Collection Lab: Propulsion Test

Propulsion Test Graphs

The data you collected (Speed variable, Distance / cm) are pairs of numbers. Both types have a different meaning. The Speed variable is a number in code of the rover's micro:bit. The distance is what you measured, which could be centimeters, meters, inches, or feet, of which each of them would result in different numbers.

Even though the data are different types of data, they have a relationship: A larger Speed variable means a greater distance covered per time. The time is different for our two experiments, 2 seconds or 4 seconds. The relationship could be written as

$$distance = velocity \cdot time$$

$$d = v \cdot t$$

We understand, the Speed variable is just a number relating to actual speed, which in physical science, velocity (that's where the letter "v" comes from). Actual speed, or velocity are measured in distance per time, e.g., cm/s (read centimeters per second) or m/s or km/h (read kilometers per hour), or "/s (inches per second), or '/min (feet per minute), or mph (miles per hour, here somebody did not like the slash "/" for "per", we could also use "m/h", which would be correct but very uncommon).

Pairs of numbers can be visualized in graphs. Enter your data as points in the graphs that follow.

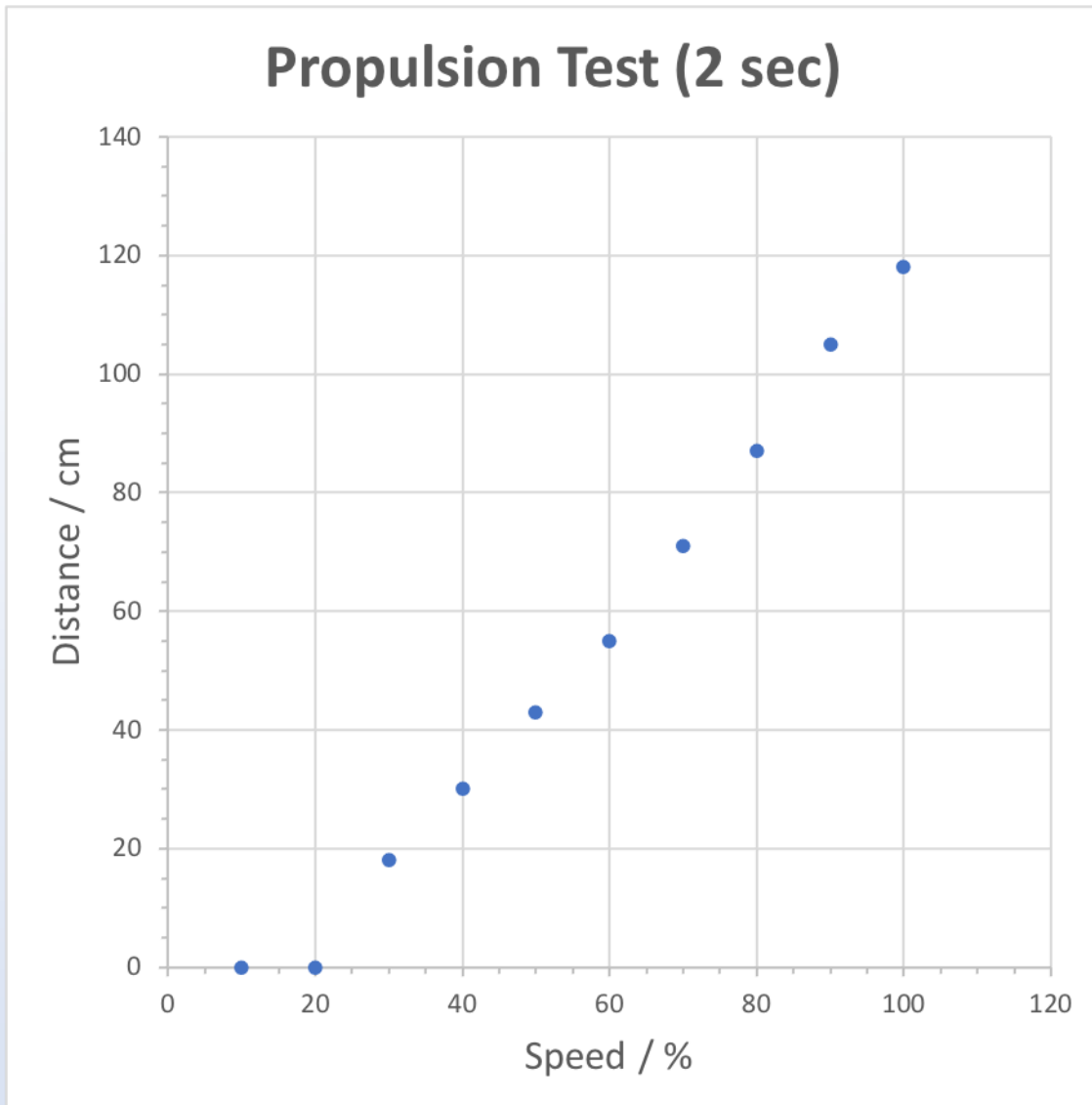
You can compare your data with the example graphs, which contain data from another rover (it was RV44).

Continued next page.



Data Collection Lab: Propulsion Test

Example data for the first propulsion test with driving time = 2 sec

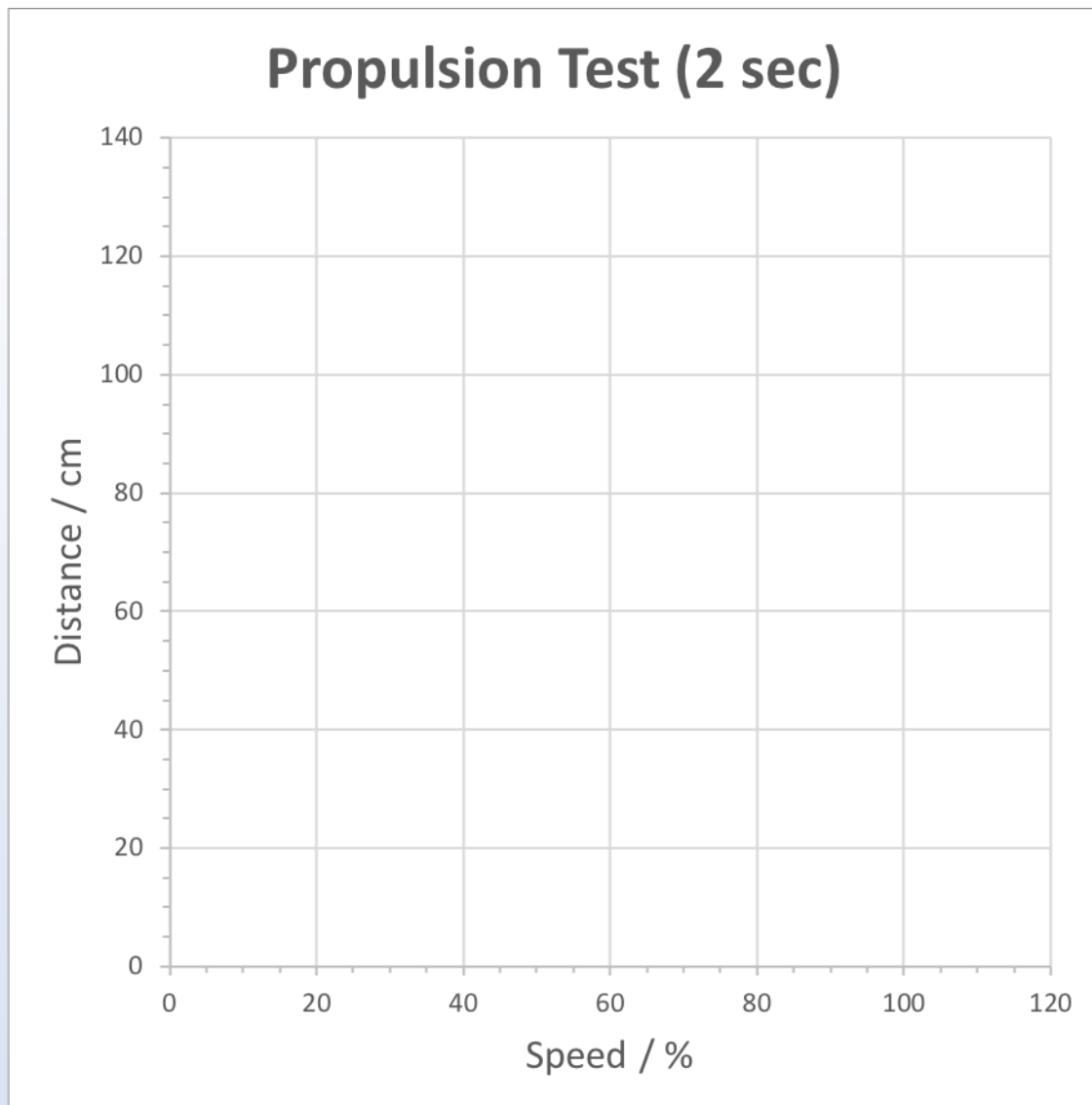


Continued next page.



Data Collection Lab: Propulsion Test

Using the pairs of data that you recorded in Data Table 1 (2 Seconds), plot each ordered pair of data (speed/distance) on the coordinate graph.

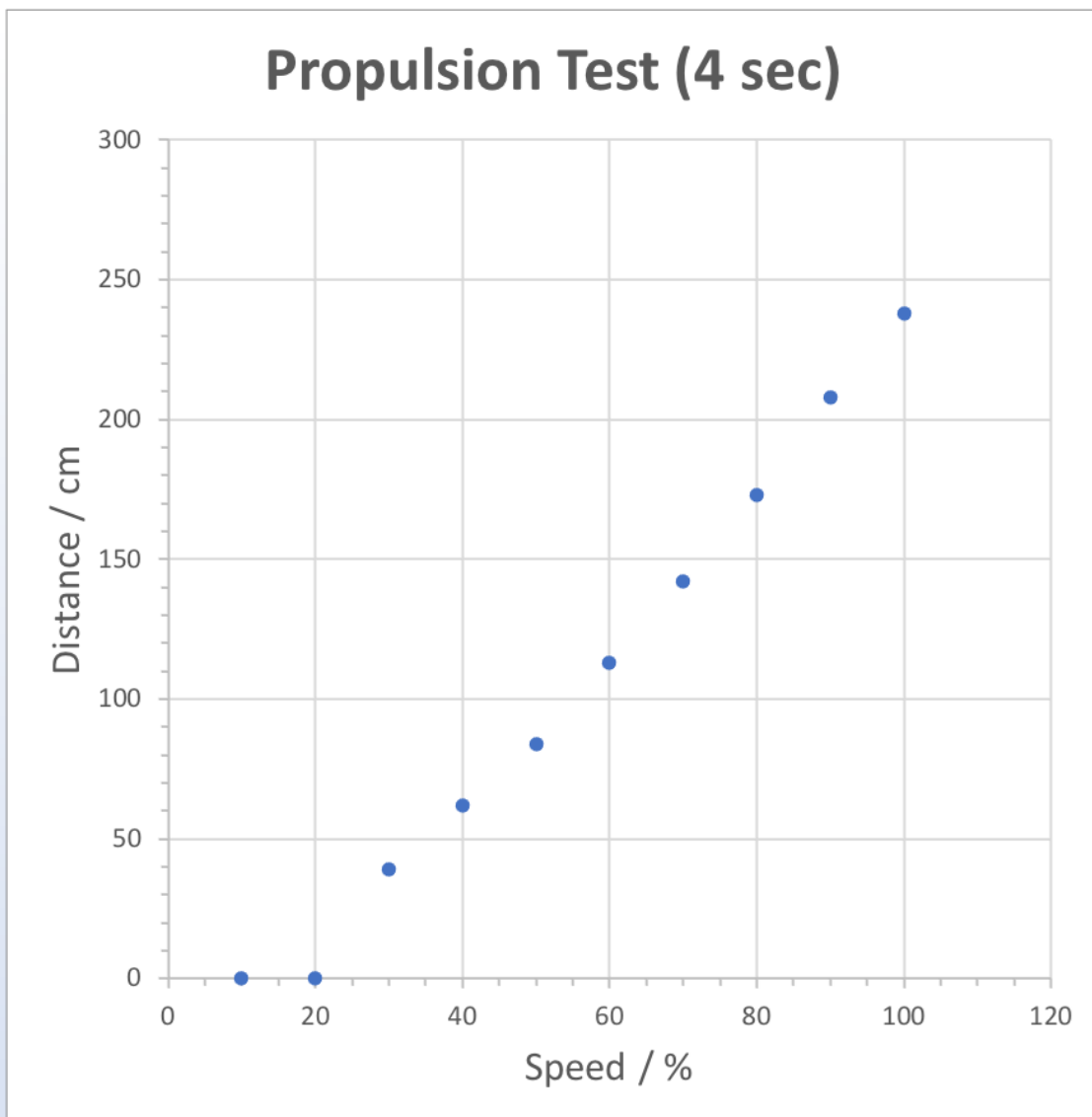


Continued next page.



Data Collection Lab: Propulsion Test

Example data for the second propulsion test with driving time = 4 sec

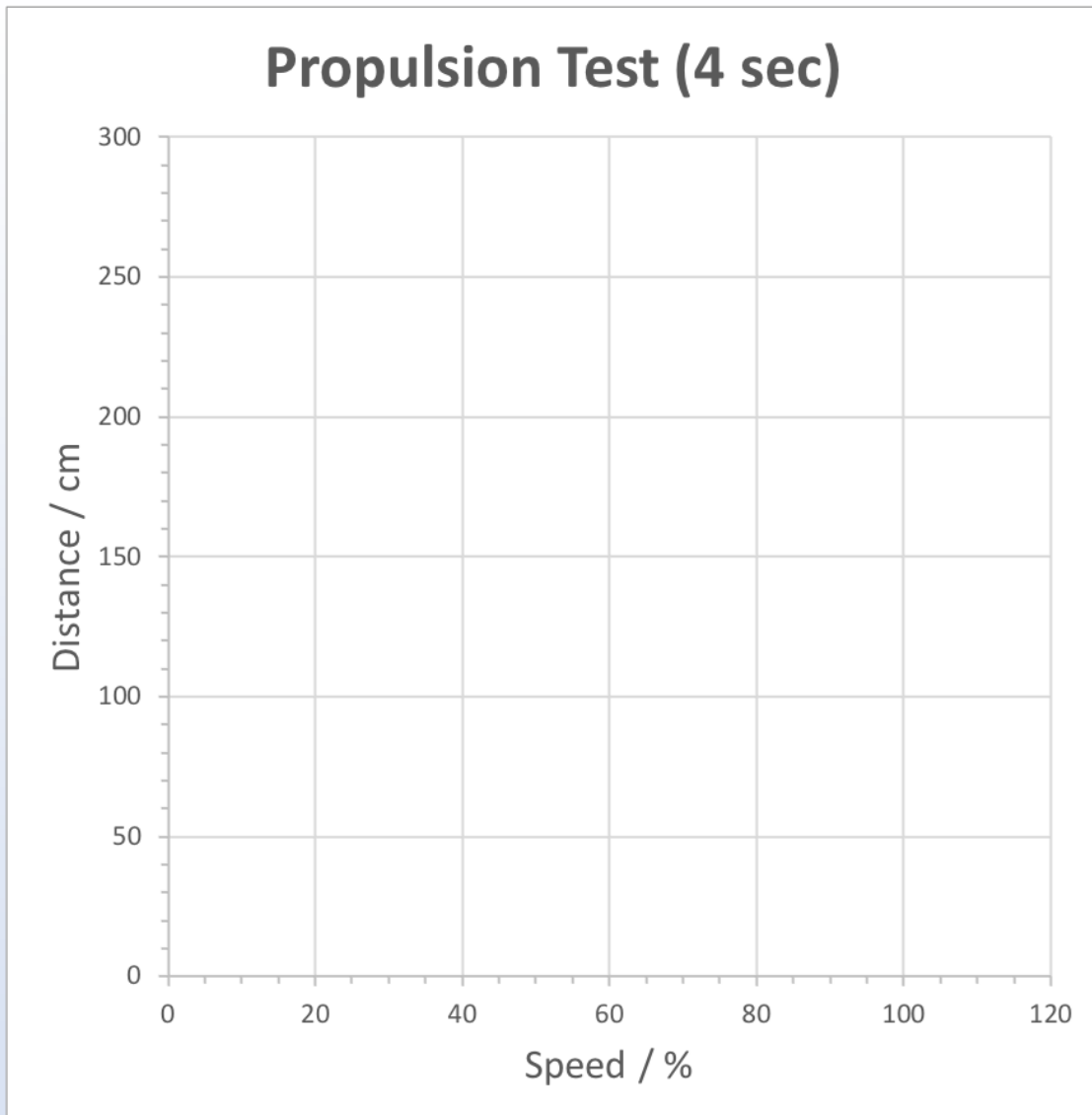


Continued next page.



Data Collection Lab: Propulsion Test

Using the pairs of data that you recorded in Data Table 1 (4 Seconds), plot each ordered pair of data (speed/distance) on the coordinate graph.

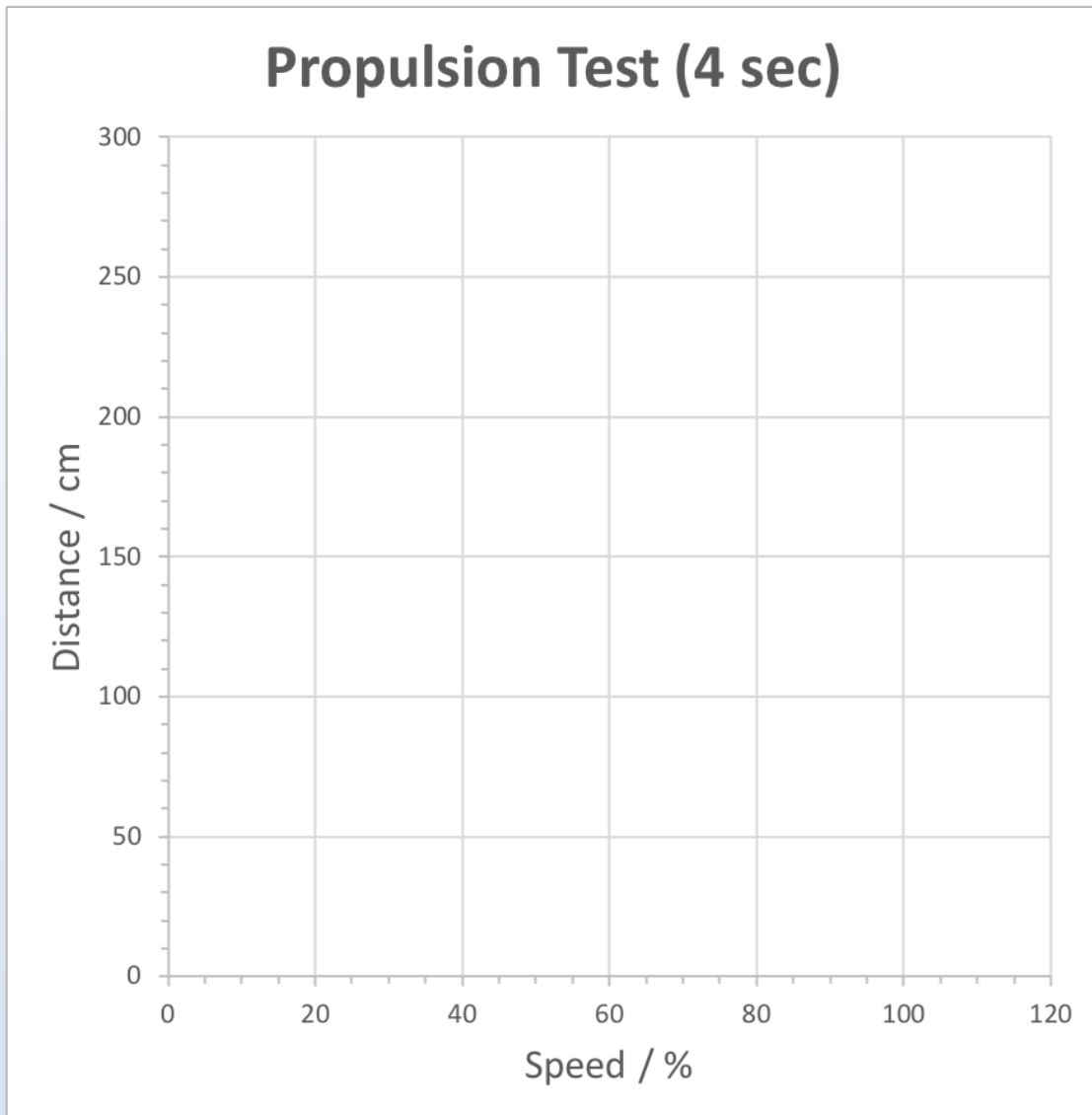


Continued next page.



Data Collection Lab: Propulsion Test

Using the pairs of data that you recorded in Data Table 2 (4 Seconds), plot each ordered pair of data (speed/distance) on the coordinate graph.



Continued next page.



Data Collection Lab: Propulsion Test

Results from the Propulsion Test

- Do you see an (almost) straight line in the graph of the data in Data Tables 1. and 2.?
- Did the rover move at all possible speed settings? If or if not, explain what you think could be the cause. How does this effect show in the graphs?
- Did the rover move twice as far in the 2 second test for all possible speed settings? Why or why not? Do you know how to determine the slope of the two graphs? Try!
- Did the rover go straight at all possible speed settings?

Optional Activity: Reverse Propulsion Test

Your rover drives in reverse as well. We can test this, too, by reversing the meter stick: Place the meter stick in the opposite direction, but still with zero centimeters at the front of the sonar sensor.

Use the A button to choose Speed, and record the distance travelled, now as negative numbers in Data Table 3.

Then, enter your data (together with the data from Data Table 2.) as points in a graph. Is it still an almost straight line?

Data Table 3. 4 Seconds in Reverse

Test Run Number	Speed Display	Speed Variable	Distance / cm
1	0	0	
2	-1	-10	
3	-2	-20	
4	-3	-30	
5	-4	-40	
6	-5	-50	
7	-6	-60	
8	-7	-70	
9	-8	-80	
10	-9	-90	
11	-10	-100	

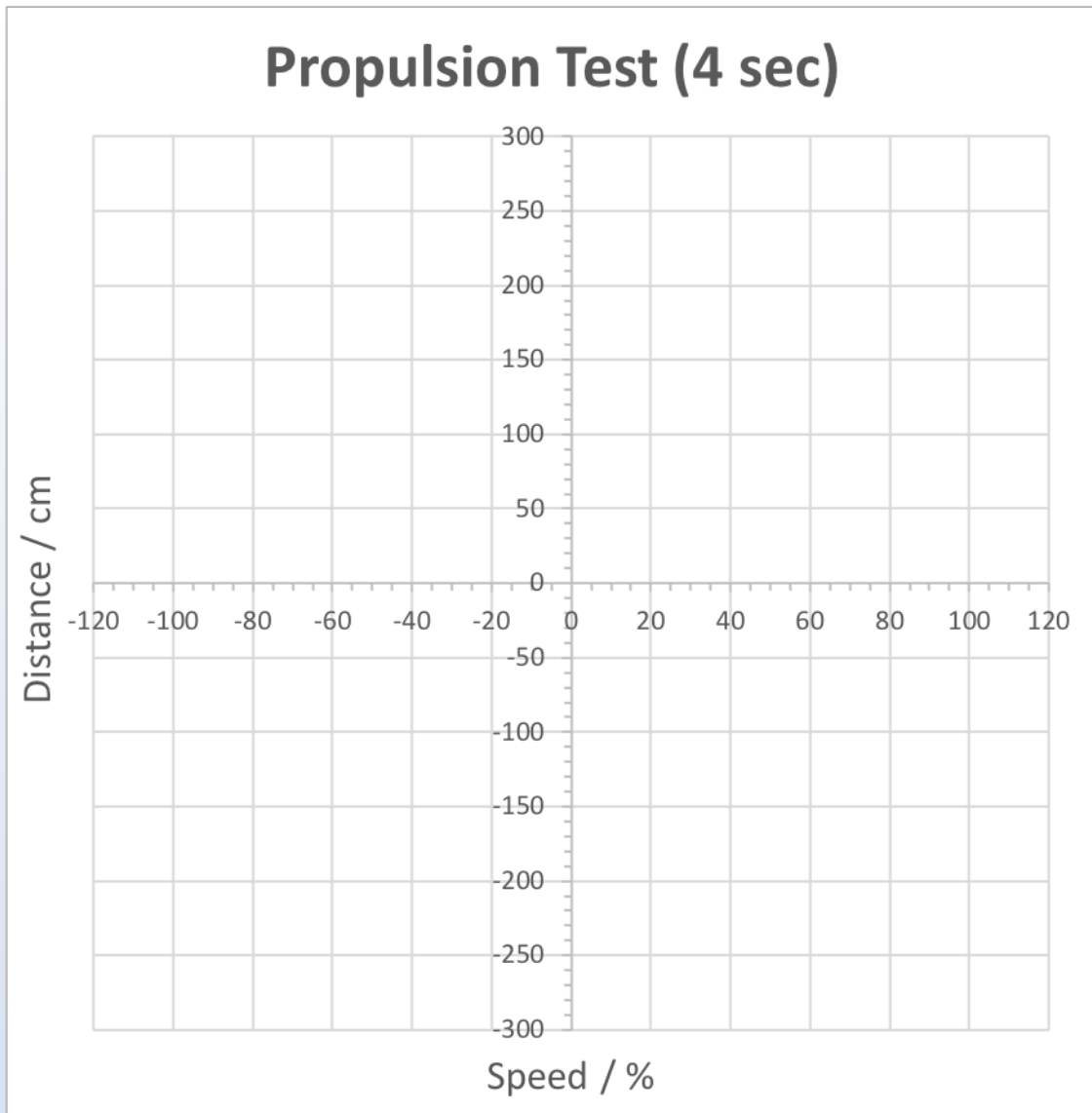
Continued next page.



Data Collection Lab: Propulsion Test

Enter your data in this table for both, forward and reverse driving for 4 seconds.

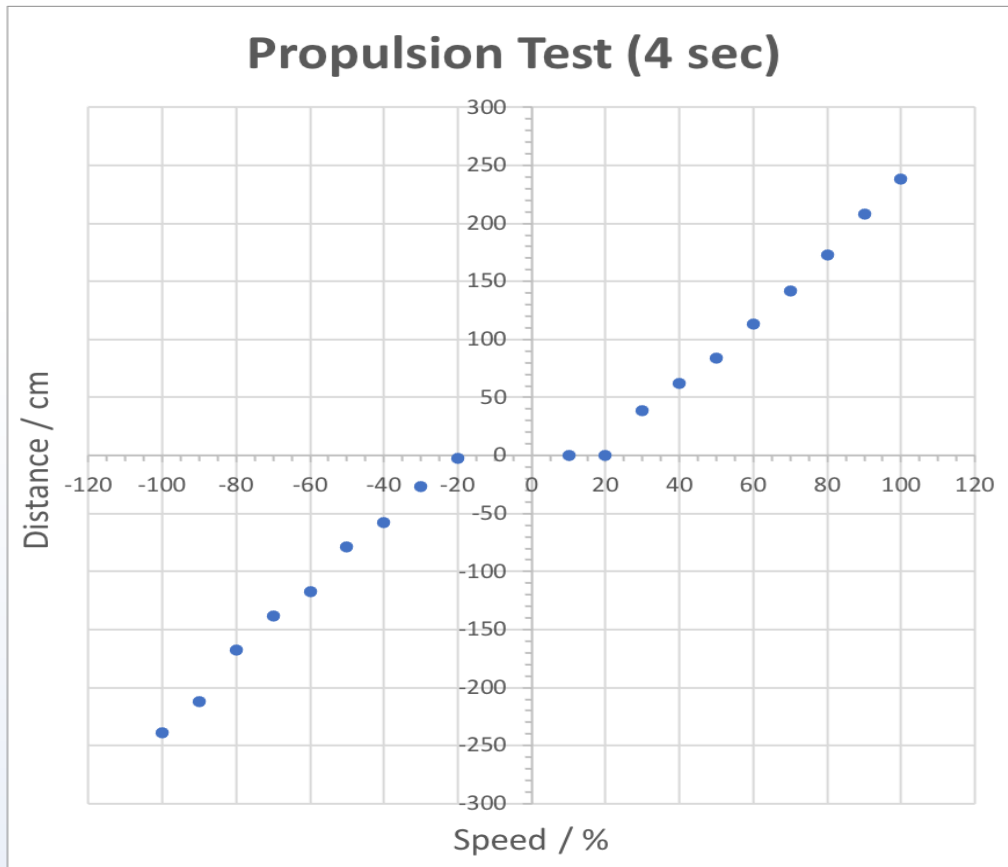
See the example graph from RV 44 on the next page.



Continued next page.



Data Collection Lab: Propulsion Test



Example data from another rover. 4 sec forward and reverse.

Consider forward and reverse driving as two different data sets, and therefore as two different approximately straight lines: Can you determine the slope of each?

The graph shows clearly that around Speed = 0, the Speed variable is not proportional to the real speed of the rover. The power from the motor:bit board must be enough to overcome the reluctance of the motors to turn (friction of the internal gears?). When there is enough power, the data points follow closely a straight line.

But we also see, the data points do not follow exactly a straight line. Also, if you repeat the whole experiment, you might find a different data set and therefore, a different graph. There could be other effects (temperature of the motors? battery charge?) having an influence on the result and leading to slight deviations.

Module 7: Steering with a Remote Control

Now that you understand how speed, power, and distance are related and how you can manipulate them to control how your rover moves, let's look at how we can control the steering of the rover. Remember, we can use code to tell the rover in what direction to travel but we can also create a remote-control for the rover that would also allow us to move the rover.

To create a remote control for the rover, we will use a second micro:bit and the built-in radio and accelerometer features of the micro:bit. The remote will provide an easier way to change the direction of the rover (steer) as well as change the speed.

To “build” the remote-control (RC) – use the extra micro:bit in your kit along with the battery case that connects to the micro:bit. It's as simple as that- now you have a programmable remote-control - once you write the code!

You might ask how?

Let's look at some of the features of the micro:bit in the resources below.



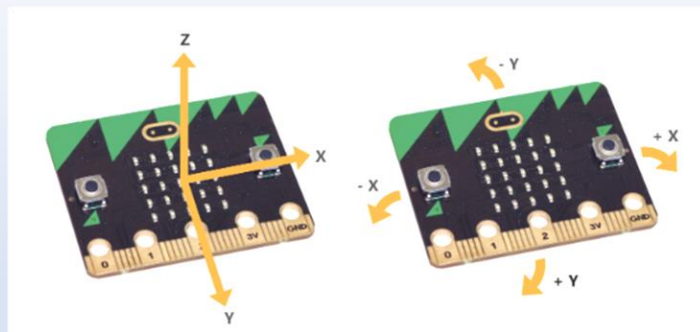
Internet Resources: Radio and Accelerometer Features of the Micro:bit

Check out these cool videos about the radio features of the micro:bit:

- [Micro:bit Radio](#)
- [Behind the MakeCode Hardware: Radio in Micro:bit](#)
- [Micro:bit Accelerometer](#)
- [Behind the MakeCode Hardware: Accelerometer on Micro:bit](#)

The micro:bit measures movement along three axes:

- X - tilting from left to right.
- Y - tilting forwards and backwards.
- Z - moving up and down.



7.1 Adding a Remote Control (Propulsion 5 / Rover RC 1)

To create the code to allow the rover to work with a remote control, we will first modify the “**Propulsion 4**” code for the rover so that it can receive messages from a remote-control micro:bit.

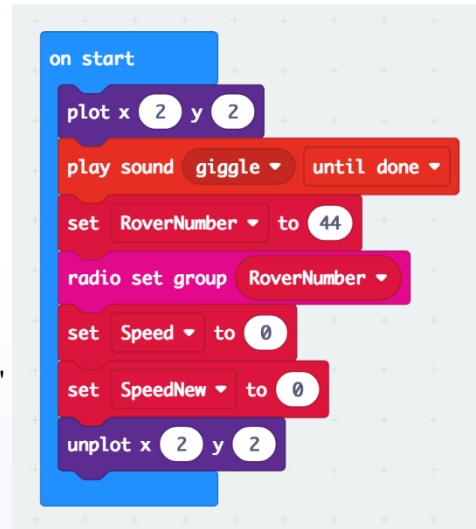


Time to Code: Propulsion Test 5

4. Open the “Propulsion Test 4” project and change the name to “Propulsion Test 5”.

5. Modify the “on start” block:

- Make a new variable and name it “RoverNumber”.
- Place a "set RoverNumber to 0" block after the "play sound ..." block in the "on start" block.
 - Change the zero in this block to the number of **your rover (flag number)**.
- From the "Radio" toolbox, get a "radio set group 1" block and place it right after the "set RoverNumber ..." block.
 - Change the number 1 to the variable “RoverNumber”



3. Create a “on radio received receivedString” block:

- From the "Radio" toolbox, get an "on radio received receivedString" block, and place it beside the "on ... pressed" blocks you already have in use.
- Place an "if ..." block into the new "on radio received ..." block,
 - Get a "0 = 0" comparison block from the "Logic" toolbox and place it over the "true" in the new "if ..." block.
 - From the "Advanced" "Text" toolbox get a "parse to number 123" block and place it over the first zero of the comparison block.
 - From the "Advanced" "Text" toolbox get a "substring of ... from 0 to 10" block and place it over the "123" of the "parse to number" block.
 - Drag the "receivedString" variable from the top on the "of ..." place in the "substring ..." block,
 - Change the number 10 in this block to the number 2,
 - Replace the second zero in the comparison block with the "RoverNumber" variable,

The “on radio received” block should look like this so far:



Continued next page.



Time to Code: Propulsion Test 5 (continued)

4. Continue with the “on radio received receivedString” block:

- Place a new "if ..." block into the previously created "if ..." block in the "on radio received ..." block.
 - Get a " " = " " string comparison block from the "Logic" toolbox and place it over the "true" in the new "if ..." block.
 - From the "Advanced" "Text" toolbox get a "substring of ... from 0 to 10" block and place it over the first " " of the comparison block.
 - Drag the "receivedString" variable from the top on the "of ..." place in the "substring ..." block.
 - Change the number 0 in this block to the number 2,
 - and the number 10 to the number 1.
 - Type an "S" (only the capital character S without the quotation marks!) into the place after the equal sign in the comparison block.
 - Place a "set SpeedNew to ..." block inside the "if ..." block.
 - Get a "parse to number ..." block from "Text" and drop it onto the zero in the "set SpeedNew ..." block.
 - Copy the "substring ..." block from above, paste and place it over the placeholder in the "parse to number ..." block.
 - Change the numbers that the block reads "set SpeedNew to parse to number substring of receivedString from 3 of length 3",
 - Get a “set SpeedNew to” block and place after the above “set SpeedNew” block.
 - Place a math block “0 – 0” in the “set SpeedNew to” block.
 - Place a variable “SpeedNew” in over the first zero.
 - Change the second zero to a 200.
- Get another "if ..." block and place it after the “second “set SpeedNew to” block – it will be inside both previous “if then...” blocks.
 - Get "... or ..." comparison block from “Logic”, place it into the new "if ..." block.
 - Add a comparison block “0 < 0” to each of the open spaces in the “... or...” block.
 - Place a “SpeedNew” variable over both first zeros.
 - Then in the first comparison make it “SpeedNew” ≥ (greater than or equal) -100,
 - and “SpeedNew” ≤ (less than or equal)100 in the second.
- Make a new "set Speed to SpeedNew" block and place it into the new "if ..." block.
- Get a "Drive Wheels with Speed 0 and Steer 0" block from the STEM SEALs toolbox and place it after the "set Speed ..." block.
 - Place a “Speed” variable over the first zero.

See the code on the next page.

The complete code for “**Propulsion Test 5**” looks like this:

It now has code written to receive radio messages from a second micro:bit.

Note: The only blocks that changed were the “**on start**” and “**on radio received**”. All other blocks remain the same.

```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set Speed to 0
  set SpeedNew to 0
  unplot x 2 y 2
```

The "on start" block of "Propulsion Test 5".

Note, the number of the variable RoverNumber must be set to the number of your own rover.

The “on radio received” of Propulsion Test 5.

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = "S" then
      set SpeedNew to parse to number substring of receivedString from 3 of length 3
      set SpeedNew to SpeedNew - 200
      if SpeedNew >= -100 and SpeedNew <= 100 then
        set Speed to SpeedNew
        Drive Wheels with Speed Speed and Steer 0
```

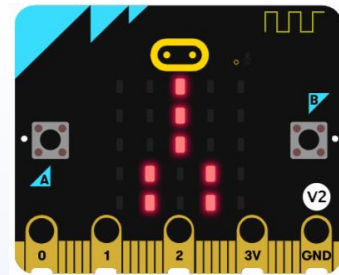
Code link "Propulsion Test 5": https://makecode.microbit.org/_660Wkx9bK9xf

Now we will create the code for the second micro:bit in your kit. This micro:bit will become a remote control for the rover. At this point it might be helpful to find a way to designate which micro:bit you will use for the rover and which you will use for the remote control.



Time to Code: Rover RC 1

1. Open MakeCode, create a new project and name it “**Rover RC 1**”.
 - You will not need to the SS Template for this code.
2. The “**on start**” block:
 - Place seven “plot x 0 y 0” blocks at the beginning of the “on start” block.
 - Change the x y values as listed in this order:
 - x 1 y 4
 - x 1 y 3
 - x 3 y 3 This creates an image to like
 - x 2 y 2 this on the remote control.
 - x 2 y 1 Looks like a transmitting antenna.
 - x 2 y 0



- Place a "play sound giggle ..." block underneath it and change "giggle" to "spring"
- Make a new variable "RoverNumber".
- Place a "set RoverNumber to 0" block after the "play sound ..." block.
 - Change the zero to the number on the flag of your rover.
- From the "Radio" toolbox, get the "radio set group 1" block.
 - Change the number 1 to the variable "RoverNumber".
- Create three variables, "Speed", "Steer", and “Message”
- Place a "set Speed to 0" block underneath the “radio set...” block.
- Place a "set Steer to 0" block after this block.
- Set a "set Message to 0" block at the end,
 - From the "Text" toolbox, get the "" "" string block and drop it over the zero in the "set Message ..." block.



Continued next page.



Time to Code: Rover RC 1 (Continued)

3. The “forever” block:

- Place a "unplot x 2 y 0" block at the beginning of the “forever” block.
- Create a variable "gx" and another "gy".
- Place a "set gx to 0" block after the “unplot...” block.
 - Drag from the "Input" toolbox the "acceleration (mg) x" block and place over the zero.
- Copy and paste this "set gx ..." block and place it underneath the one you copied.
 - Change the "gx" to "gy" and the "x" to "y".
- Place a "set Speed to 0" block next.
 - Place the "square root ..." block from "Math" over the zero.
 - Place a "0 + 0" block from "Math" inside the "square root ...".
 - Place a "0 x 0" block from "Math" over the first zero of the addition block.
 - Replace both zeros in this multiplication block with the variable "gx",
 - Copy, paste and place this multiplication block over the second zero in the addition block.
 - Change in this second multiplication block both variables "gx" to "gy".
- Place a new "set Speed to 0" block after the one with the square root.
 - Place a new "square root 0" block over the zero in this block.
 - Change the "square root" in this block to integer division.
 - Replace the first zero with the "Speed" variable.
 - Replace the second zero with the number 10.

The “forever” block should look like this thus far:



Continued next page.



Time to Code: Rover RC 1 (Continued)

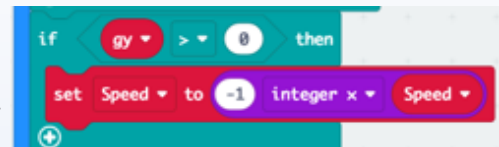
4. Continue with the “forever” block:

- Get an "if ..." block from "Logic" and place it underneath “set Speed...” block.
 - Get a "0 < 0" comparison block and place it over the "true".
 - Replace the first zero with the "Speed" variable.
 - Replace the second zero with the number 100
 - Change the less than sign to a greater than.

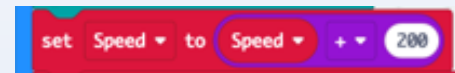


- place a "set Speed to 100" block inside this "if ..." block.

- Copy the "if ..." block you just created, paste, and place it at the end of the "forever" block.
 - Change the "Speed" variable in the comparison block to "gy".
 - Replace the number 100 with a zero.
 - Drag a "square root 0" block from "Math" over the 100 in this "set Speed ..." block,
 - Change the "square root ..." to integer multiplication.
 - Change the first zero to a negative one.
 - Change the second zero to the "Speed" variable.



- Place a "set Speed to 0" block after that last "if ..." block,
 - Drag a "0 + 0" block from "Math" over the zero.
 - Replace the first zero with the "Speed" variable.
 - Replace the second zero with the number 200.



- Place a "set Message to 0" block after the "set Speed ..." block.
 - Drag the "join ..." block from "Text" over the zero.
 - Click on the "+" sign at the end of the "join ..." block.
 - Drag the "convert 0 to text" block from "Text" over the first and third spaces.
 - Place the variable "RoverNumber" in the first “convert...” block.
 - Enter the letter "S" (just the S!) in the second space of the "join ..." block.
 - Place the variable "Speed" in the second “convert...” block.



- Place a "radio send string" block from "Radio" at the end in the "forever" block.
 - Place the variable "Message" into this block.

- Place a "pause (ms) 100" block thereafter.

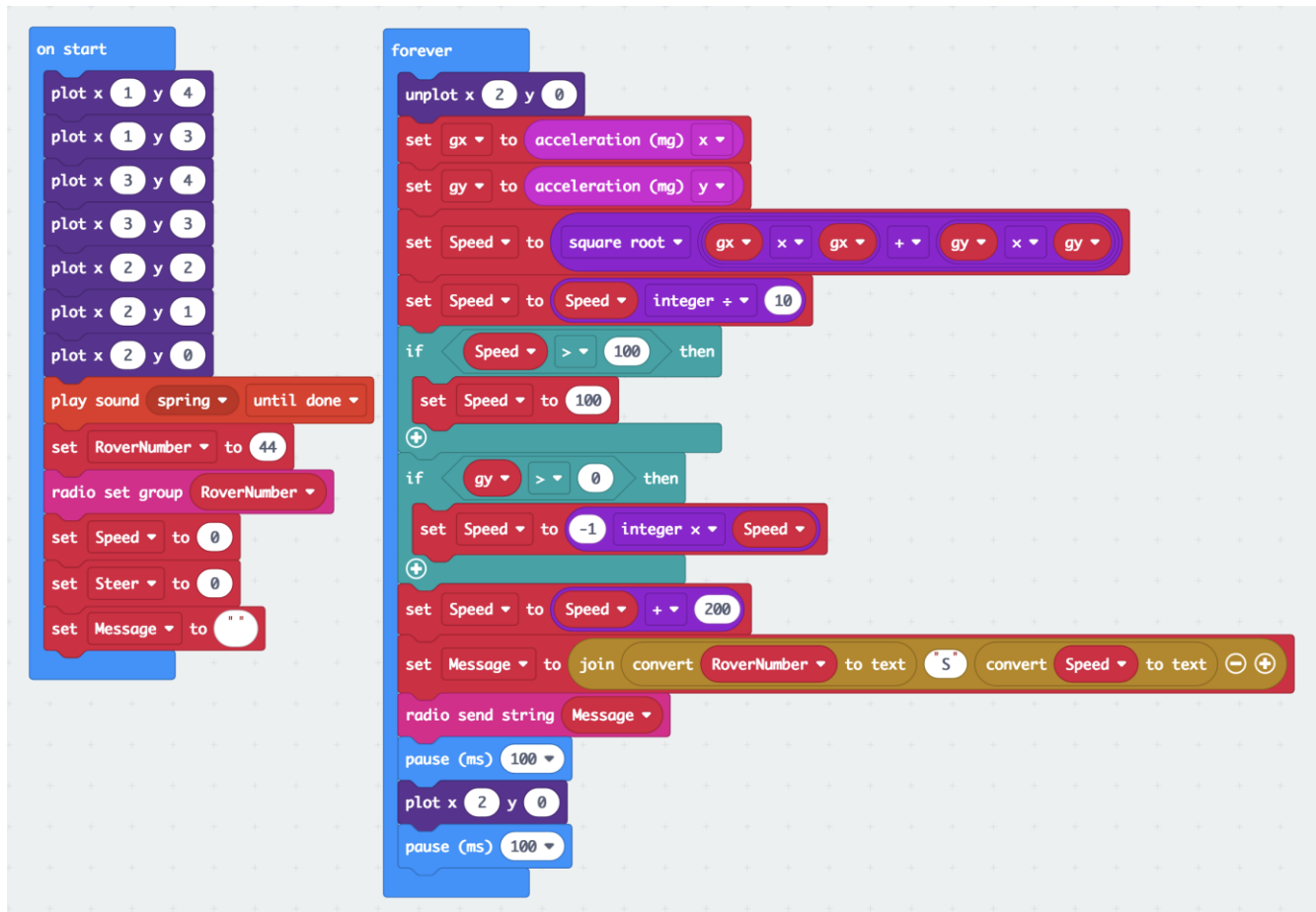
- Place a "plot x 2 y 0" block thereafter.

- Place a "pause (ms) 100" block at the very end.



See full code on the next page.

The complete code for “Rover RC 1” should look like this:



The image shows a Scratch script for a remote control rover. It is divided into two main sections: 'on start' and 'forever'.

on start:

- plot x 1 y 4
- plot x 1 y 3
- plot x 3 y 4
- plot x 3 y 3
- plot x 2 y 2
- plot x 2 y 1
- plot x 2 y 0
- play sound spring until done
- set RoverNumber to 44
- radio set group RoverNumber
- set Speed to 0
- set Steer to 0
- set Message to ""

forever:

- unplot x 2 y 0
- set gx to acceleration (mg) x
- set gy to acceleration (mg) y
- set Speed to square root of (gx x gx + gy x gy)
- set Speed to Speed integer ÷ 10
- if Speed > 100 then
 - set Speed to 100
- if gy > 0 then
 - set Speed to -1 integer x Speed
- set Speed to Speed + 200
- set Message to join (convert RoverNumber to text 'S', convert Speed to text)
- radio send string Message
- pause (ms) 100
- plot x 2 y 0
- pause (ms) 100

The code link: "Rover RC 1" for remote control: https://makecode.microbit.org/_coufkYUTt8R3

Code link "Propulsion Test 5" for rover: https://makecode.microbit.org/_660Wkx9bK9xf



Try This: Remote Control Test (Speed) (Propulsion 5 & Rover RC 1)

1. Turn the motor:bit on the rover on. Make sure the rover micro:bit is inserted into the motor:bit.
 - Place the rover on the floor.
2. Connect the remote-control micro:bit to a power supply (battery case with batteries).
3. **Caution:** Hold the remote-control with both hands and micro:bit as level as you can.
4. Gently tilt the front/top of the remote-control micro:bit down.
 - Does the rover move?
 - In what direction?
 - Look at the speed LEDs on the rover micro:bit as you do this – what do you notice?
5. Now tilt the rear/back of the remote-control micro:bit down.
 - Does the rover move?
 - In what direction?
 - Look at the speed LEDs on the rover micro:bit as you do this – what do you notice?


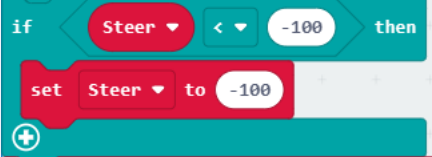
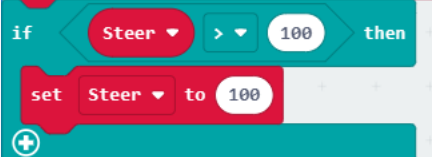
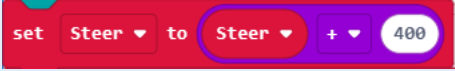

7.2 Propulsion 6 and Rover RC 2

The remote control that you created will now control the forward and reverse motion of the rover. However, it would be nice if we could also steer or turn the rover as well.

Let's modify our code for both the rover (Propulsion 5) and the remote-control (Rover RC 1).

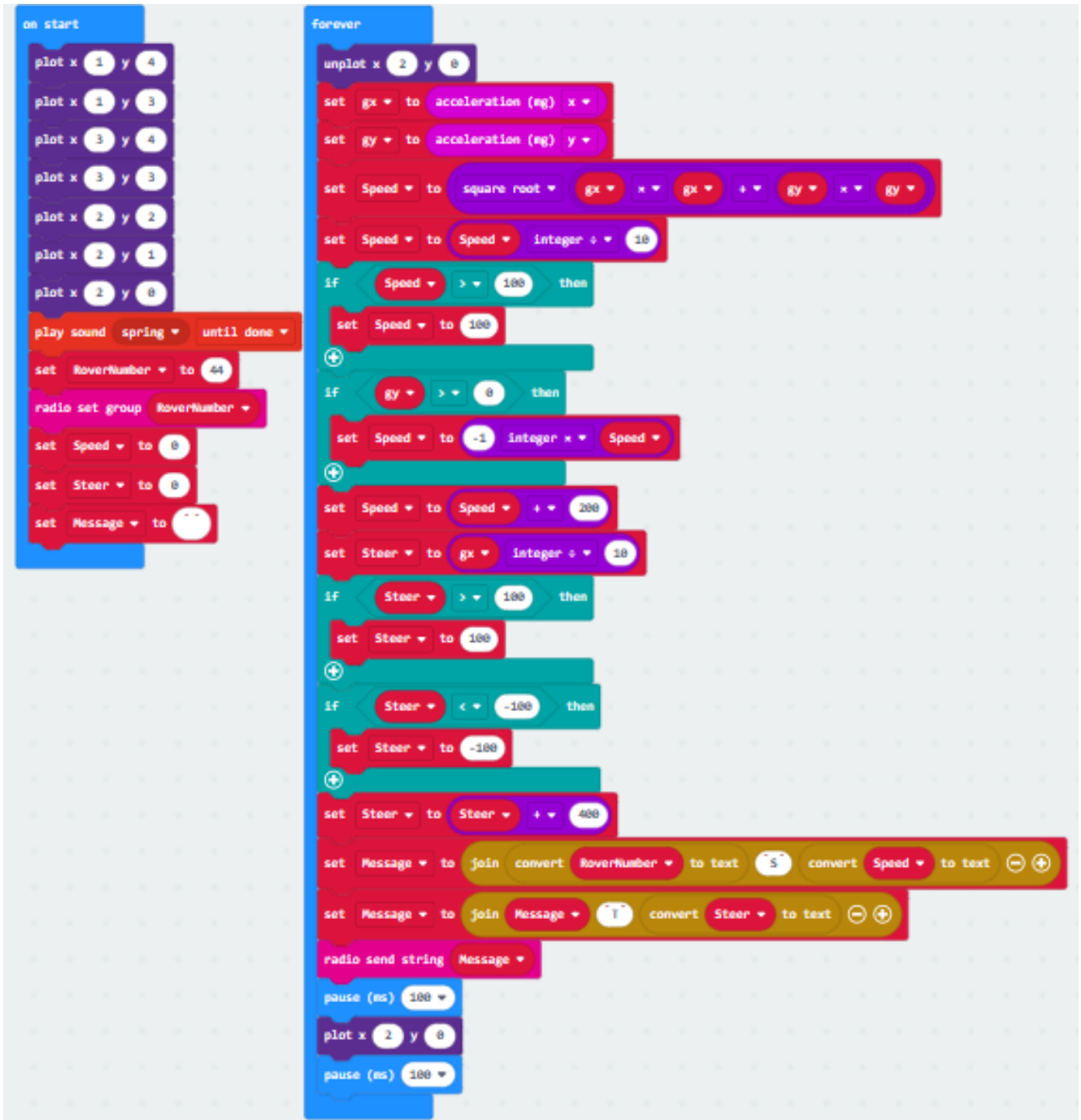


Time to Code: Remote Control 2

1. Start with "**Rover RC 1**" and change the name to "**Rover RC 2**",
2. Modify the "forever" Block:
 - Copy the "set Speed to Speed integer division by 10" block and paste it after the "set Speed to Speed + 200" block.
 - Change the first Speed variable to Steer
 - and the second to gx.
 - Copy the "if Speed > 100 then ..." block from above.
 - Paste and place it after the "set Steer ..." block just created.
 - Change the Speed variable to Steer.
 - Also change the Speed variable inside this new "if ..." block to Steer,
 - Copy this "if Steer > 100 then ..." block, paste and place it underneath,
 - Change the inequality sign from greater to less than.
 - Change the number 100 to a negative 100.
 - Also change the number 100 in the "set Steer ..." block inside the "if ..." to negative 100.
 - Copy the "set Speed to Speed + 200" block.
 - Paste it after this last "if ..." block.
 - Change both Speed variables to Steer.
 - Change the + 200 to + 400.
 - Copy, paste and place the "set Message to join 'S' ..." block below the other "set Message ..." block.
 - Delete the first "convert to text" block and replace it with the variable Message.
 - Change the "S" to a "T".
 - Change the Speed variable in the last "convert..." block to Steer.

Download this code onto the micro:bit used as the remote control.

The complete code for the “Rover RC 2” should look like this:



The image shows a Scratch script for a rover. It is divided into two main sections: 'on start' and 'forever'.

on start:

- Plot x 1 y 4
- Plot x 1 y 3
- Plot x 3 y 4
- Plot x 3 y 3
- Plot x 2 y 2
- Plot x 2 y 1
- Plot x 2 y 0
- Play sound spring until done
- Set RoverNumber to 44
- Radio set group RoverNumber
- Set Speed to 0
- Set Steer to 0
- Set Message to ""

forever:

- Unplot x 2 y 0
- Set gx to acceleration (mg) x
- Set gy to acceleration (mg) y
- Set Speed to square root of (gx * gx + gy * gy)
- Set Speed to Speed integer divided by 10
- If Speed > 100 then set Speed to 100
- If gy > 0 then set Speed to -1 integer multiplied by Speed
- Set Speed to Speed + 200
- Set Steer to gx integer divided by 10
- If Steer > 100 then set Steer to 100
- If Steer < -100 then set Steer to -100
- Set Steer to Steer + 400
- Set Message to join convert RoverNumber to text, 'S', convert Speed to text, '-'
- Set Message to join Message, 'T', convert Steer to text, '-'
- Radio send string Message
- Pause (ms) 100
- Plot x 2 y 0
- Pause (ms) 100

The code link for "Rover RC 2": https://makecode.microbit.org/_APtHmRYAEd3a

Before we can test the newly modified remote control, the rover code will need to be changed.



Time to Code: Propulsion Test 6

1. Begin with "**Propulsion Test 5**" and change the name to "**Propulsion Test 6**".
2. Modify the "on radio received" block:
 - Copy the "set SpeedNew to parse ..." block.
 - Paste and place it after the "Set SpeedNew to SpeedNew – 200" block.
 - Make a new variable "Steer".
 - Change the "SpeedNew" variable to "Steer" in this new block.
 - Change the first number from 3 to 7.
 - Copy the second "set SpeedNew ..." block above.
 - Paste and place it below the "set Steer ..." block just created.
 - Change in this block "SpeedNew" to "Steer".
 - And the number 200 to 400.

The modified code should look like this.

Note: No other blocks will change in the "**Propulsion Test 6**".

```
on radio received receivedString
  if <parse to number substring of receivedString from 0 of length 2 = RoverNumber> then
    if <substring of receivedString from 2 of length 1 = "S"> then
      set SpeedNew to parse to number substring of receivedString from 3 of length 3
      set SpeedNew to SpeedNew - 200
      set Steer to parse to number substring of receivedString from 7 of length 3
      set Steer to Steer - 400
      if <SpeedNew >= -100 and SpeedNew <= 100> then
        set Speed to SpeedNew
        Drive Wheels with Speed Speed and Steer Steer
      +
      +
      +
```

The code link: "**Propulsion Test 6**": <https://makecode.microbit.org/6C0V0U3cjDuX>



Try This: Remote Control Test (Speed and Steer)

1. Turn the motor:bit on the rover on. Make sure the rover micro:bit is inserted into the motor:bit.
 - Place the rover on the floor.
2. Connect the remote-control micro:bit to a power supply (battery case with batteries).
3. **Caution:** Hold the remote-control with both hands and micro:bit as level as you can.
4. Gently tilt the front/top of the remote-control micro:bit down.
 - Does the rover move?
 - In what direction?
 - Did the rover also veer from a straight path?
5. Now tilt the remote-control micro:bit to the left.
 - Does the rover move?
 - In what direction?
 - What happens if you tilt it right?
6. Continue tilting the remote-control micro:bit in different ways.
 - Are you able to control the rover movements in a way you would expect?
 - Look at the LEDs on the rover micro:bit as you do this – what do you notice?
 - Does it make sense to you?

7.3 Rover Remote Control Practice

Now that you have a rover that can move in all directions with the use of a remote control, it is time to get some real practice in with your rover.

Module 8: Actuators and Sensors

Now that you have learned some of the basic functions of your rover propulsion system and how to control it autonomously and with a remote control, it is time to add some new features to the rover. One will be an ultrasonic sensor (sonar) and the other will be a movable cargo bed that will allow the rover to fulfill a purpose. Both will use a servo motor or actuator to control placement or location of the sonar head and the cargo bed.

8.1 What are Servo Motors?

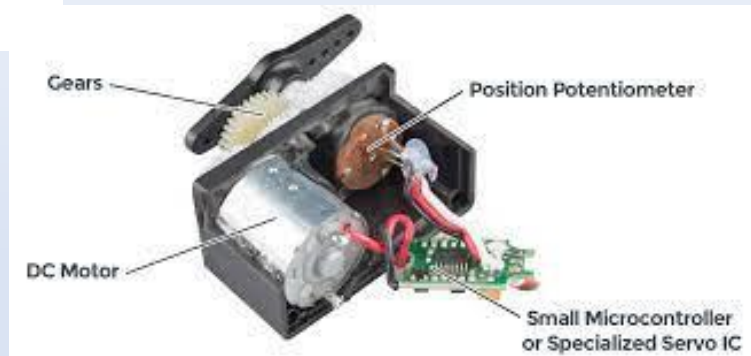
Before we install the servos on the rover, let's learn some more about servo motors.



Internet Resources:

What are servo motors and how do they work?

- Servos Explained
 - <https://www.sparkfun.com/servos>
- What is a Servo Motor and What Does it Do?
 - <https://www.youtube.com/watch?v=tHOH-bYjR4k>

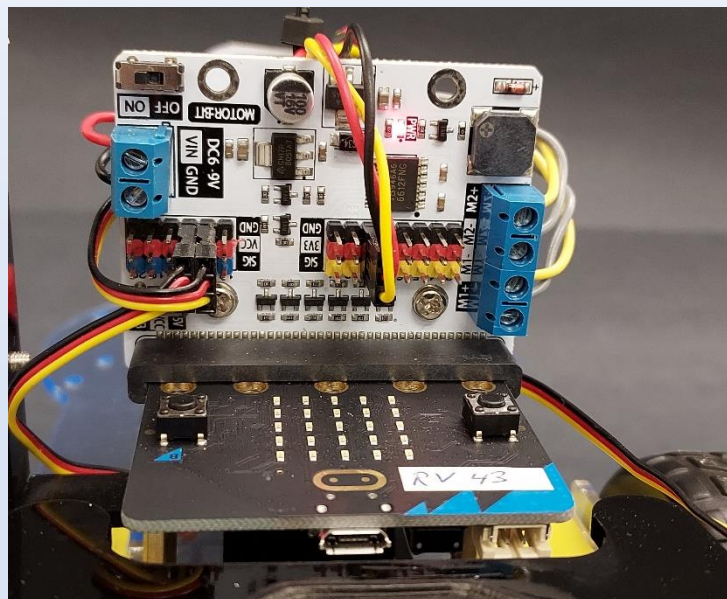
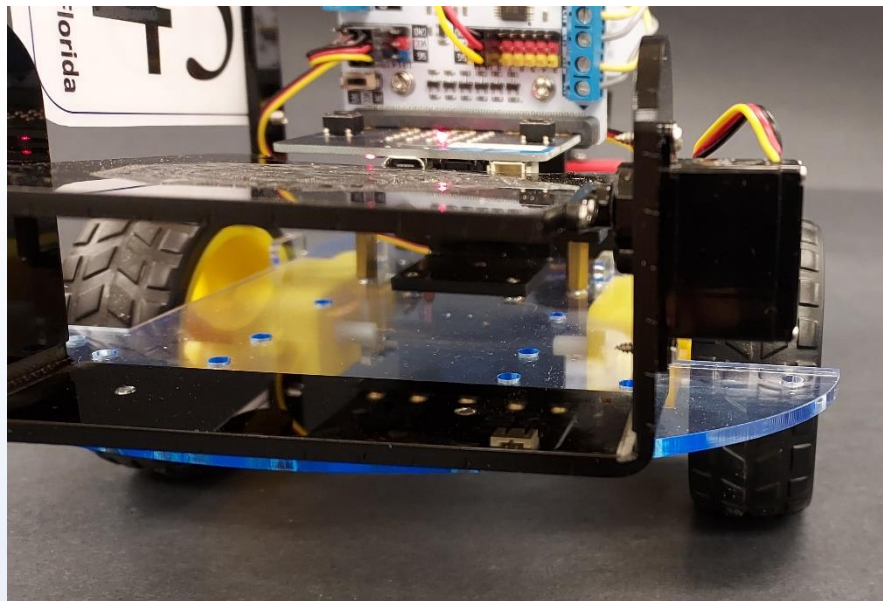


8.2 Cargo Bed Assembly



Assembly: Cargo Bed Assembly

1. Refer to the images on pages 10 through 16 to identify the parts needed for this part of the assembly.
2. Watch the STEM SEALS YouTube video: [LAND Cargo Bed Assembly](#).
3. Step-by step assembly instructions:

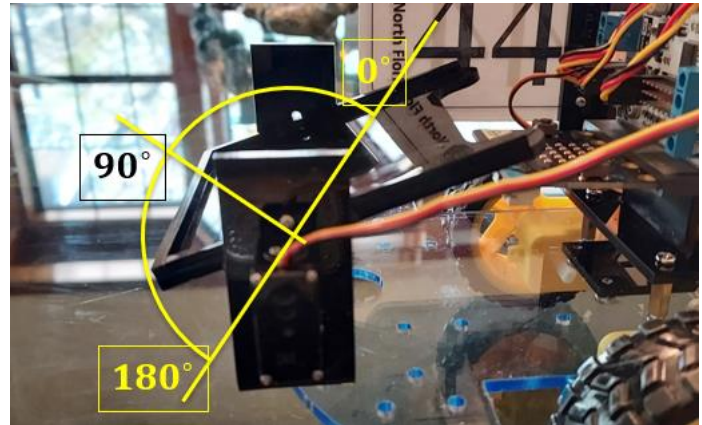


8.3 Cargo Bed Calibration (Cargo Bed Test 1)

After installing or changing out a servo motor in a device, calibration is often needed to make sure the arm position of the servo is in the appropriate place. This can be done mechanically by reattaching the arm of the servo in a different position on the motor's cog wheel. Because there are 24 teeth on the cog wheel, they are placed every 15° degrees (360° divided by 24 teeth = 15°). So large adjustments can be made by reattaching the servo arm in a different position. Small adjustments (less than 15°) can be more accurately completed by modifying the code.

The level position of the cargo bed needs to be determined and should be in the range of 100° to 120°. By writing this number to the “servo write....” block in our code we can set the servo arm to a more precise and desired position. Below is a picture of the cargo bed and the calibration.

We will start with some simple code to first observe the cargo bed to determine the calibration needed.

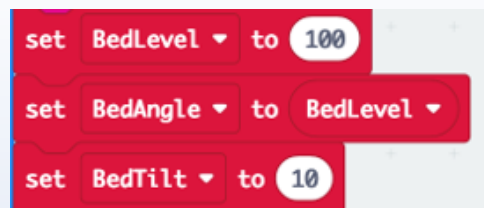


Time to Code: Cargo Bed Test 1

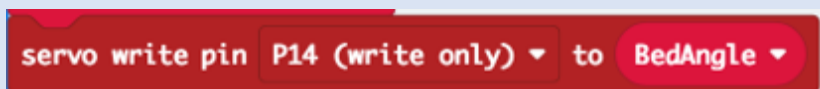
1. Start with the “**Propulsion Test 6**” code, load and rename it “**Cargo Bed Test 1**”.

2. Modify “**on start**” block:

- Create three new variables: “BedLevel”, “BedAngle”, BedTilt”
- Place a new "set BedLevel to 100" block right after the "radio set group ..." block.
- Place a new "set BedAngle to BedLevel" block underneath it.
- Place a new "set BedTilt to 10" block underneath.



- Place a "servo write pin P0 to 0" block from ("Advanced") "Pins" toolbox towards the end right before the “unplot... block.”
 - Change the P0 to P14 using the drop-down menu.
 - Place a “BedAngle” over the zero.

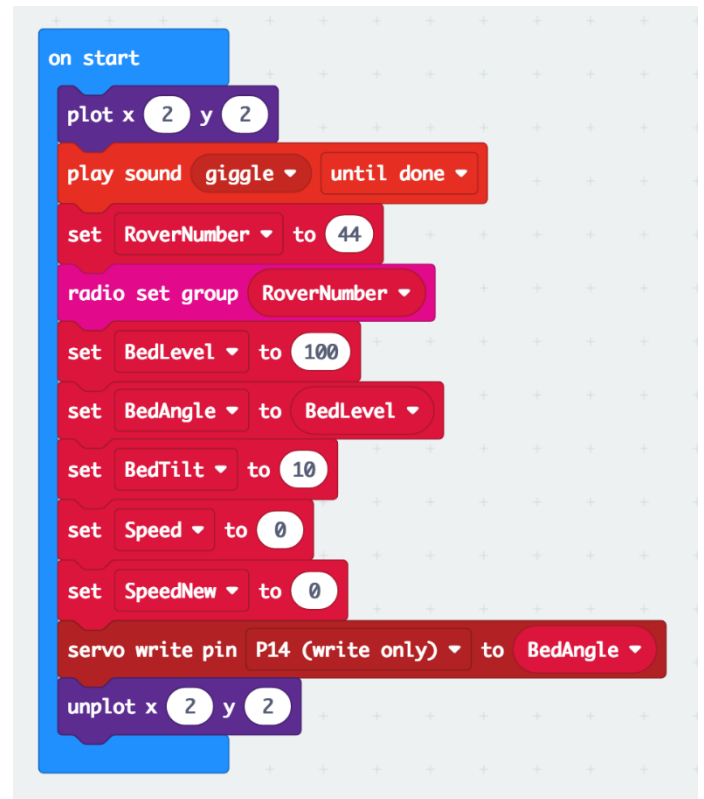


The modified code for “Cargo Bed Test 1” should look like this:

Note the only blocks that changed were in the “on start” block.

Code link "Cargo Bed Test 1":

https://makecode.microbit.org/_RjMdaKHWqh03



```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set BedLevel to 100
  set BedAngle to BedLevel
  set BedTilt to 10
  set Speed to 0
  set SpeedNew to 0
  servo write pin P14 (write only) to BedAngle
  unplot x 2 y 2
```



Try This: Cargo Bed Test 1

1. Download, upload, and test:
 - **Caution:** Do NOT turn the rover ON with the USB cable attached!
 - Remove USB cable from the rover's micro:bit.
 - Turn rover ON:
2. Does the cargo bed move?
 - Is it level?
 - **If the bed is more than 15° off from horizontal:**
 - **First** adjust the servo horn's position on the servo motor's cog wheel - leave as is if the bed is close to horizontal!
 - Turn the rover OFF and then back on: does the cargo bed move?
 - Is it level now?
 - There is some variation in the absolute servo position.
 - It should not be more than a few degrees.
 - **If the bed is less than 15° off from horizontal:**
 - **Change** the number or angle value in the “set BedLevel to 100” block.
 - **If tilted down below the horizontal point**, add degrees to the 100 (change to 110) and reload and see what happens.
 - **If tilted up above the horizontal point**, add degrees to the 100 (change to 90) and reload and see what happens.
 - Keep adjusting this variable, reload, and test until the cargo bed is as level as possible.

8.4 Remote-Controlled Cargo Bed (Cargo Bed Test 1 & 2 / Rover RC3 & 4)

Now that the cargo bed is calibrated, let's modify the code so that the bed can be operated using the remote-control. Start first with the rover code to be able to accept the messages from the remote-control to operate the cargo bed.



Time to Code: Cargo Bed Test 1 (add remote operation)

1. Use the “**Cargo Bed Test 1**” code.
2. Modify the “**on radio received...**” block:
Note: you will move some of the previously created code and modify it in after expanding one of the “if then...” blocks. The picture of the code is on the opposite page for easy reference.
 - Click on the "+" sign **two times** at the end of the "if substring of receivedString ..." block (the second "if ..." block) – this will add an “else if” and an “else” opening.
 - Copy, paste and place the "substring of receivedString from 2 of length 1 = "S"" comparison block from the beginning of the "on radio received ..." block over the "true" in the "else if ..." block.
 - Change the "... from 2 of length 1 = "S"" in the first "if ..." part to "... from 2 of length 1 = "D"" (see picture next page, “D” is for “degrees”).
 - Copy the "set SpeedNew to parse to number ..." from below, paste and place into the new "else if ..." block.
 - Copy the next "set SpeedNew to SpeedNew ..." block and place it also in the "else if ..." block.
 - Move the next "set Steer to parse to number ..." block - and everything after it into the "else if ..." block.
 - From the "StemSeals" toolbox, place a "Drive Wheels with Speed 0 and Steer 0" block at the beginning of the first "if substring ..." block.
 - From the "Basic" toolbox, place "clear screen" block thereafter.
 - In the next "set SpeedNew ..." block, change variable name and the from number such that the block reads "set BedTilt to parse to number substring of receivedString from 3 of length 3".
 - In the next "set SpeedNew ..." block, change the variable and the number to "BedTilt to BedTilt - 700".
 - Add a "set BedAngle to BedAngle + BedTilt" underneath (you can copy the block above and modify the copy!).
 - Place a "servo write pin P14 (write only) to BedAngle" at the end of the "if ..." block.
 - Place a "pause (ms) 200" at the end of the "if ..." block.



Time to Code: Cargo Bed Test 1 (add remote operation)

Modified “on radio received...” block:

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = "D" then
      Drive Wheels with Speed 0 and Steer 0
      clear screen
      set BedTilt to parse to number substring of receivedString from 3 of length 3
      set BedTilt to BedTilt - 700
      set BedAngle to BedAngle + BedTilt
      servo write pin P14 (write only) to BedAngle
      pause (ms) 200
    else if substring of receivedString from 2 of length 1 = "S" then
      set SpeedNew to parse to number substring of receivedString from 3 of length 3
      set SpeedNew to SpeedNew - 200
      set Steer to parse to number substring of receivedString from 7 of length 3
      set Steer to Steer - 400
      if SpeedNew ≥ -100 and SpeedNew ≤ 100 then
        set Speed to SpeedNew
        Drive Wheels with Speed Speed and Steer Steer
      else
        Drive Wheels with Speed Speed and Steer Steer
```

Above: "on radio received ..." block of "Cargo Bed Test 1". All other blocks, "on button A pressed", "on button B pressed", "on logo pressed", and the "forever" block are the same.

Before we test this code, the remote-control code needs to be changed as well.



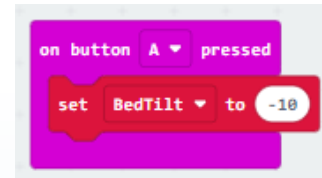
Time to Code: Rover RC 3 (adding the cargo bed)

1. Use the “Rover RC 2” and rename it “Rover RC 3”.
2. Modify the “on start” block:

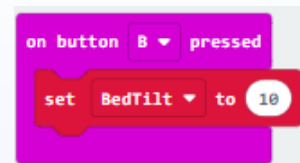
- Create a new variable "BedTilt".
- Place a "set BedTilt to 0" block after the "radio set group ..." block.



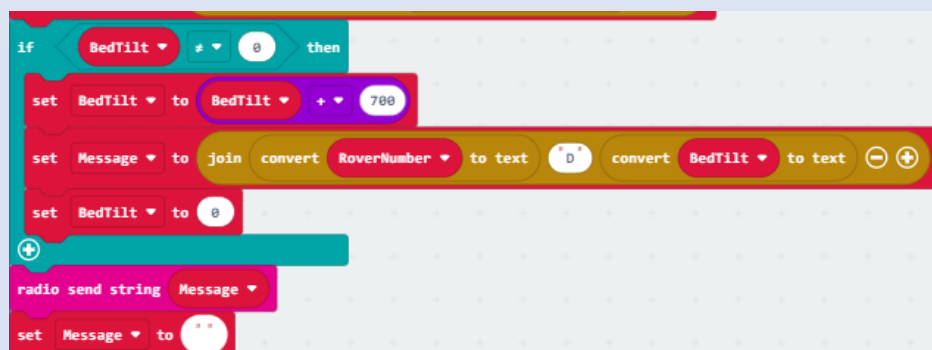
3. Place an "on button A pressed" block in the workspace:
 - Copy the "set BedTilt to 0" block from the "on start" block, paste and place inside the "on button A pressed" block.
 - Change the zero in this block to the number -10.



4. Make an “on button B pressed” block:
 - Copy the “on button A pressed” block, paste.
 - Change the “A” to “B”.
 - Change the number -10 to a positive 10.

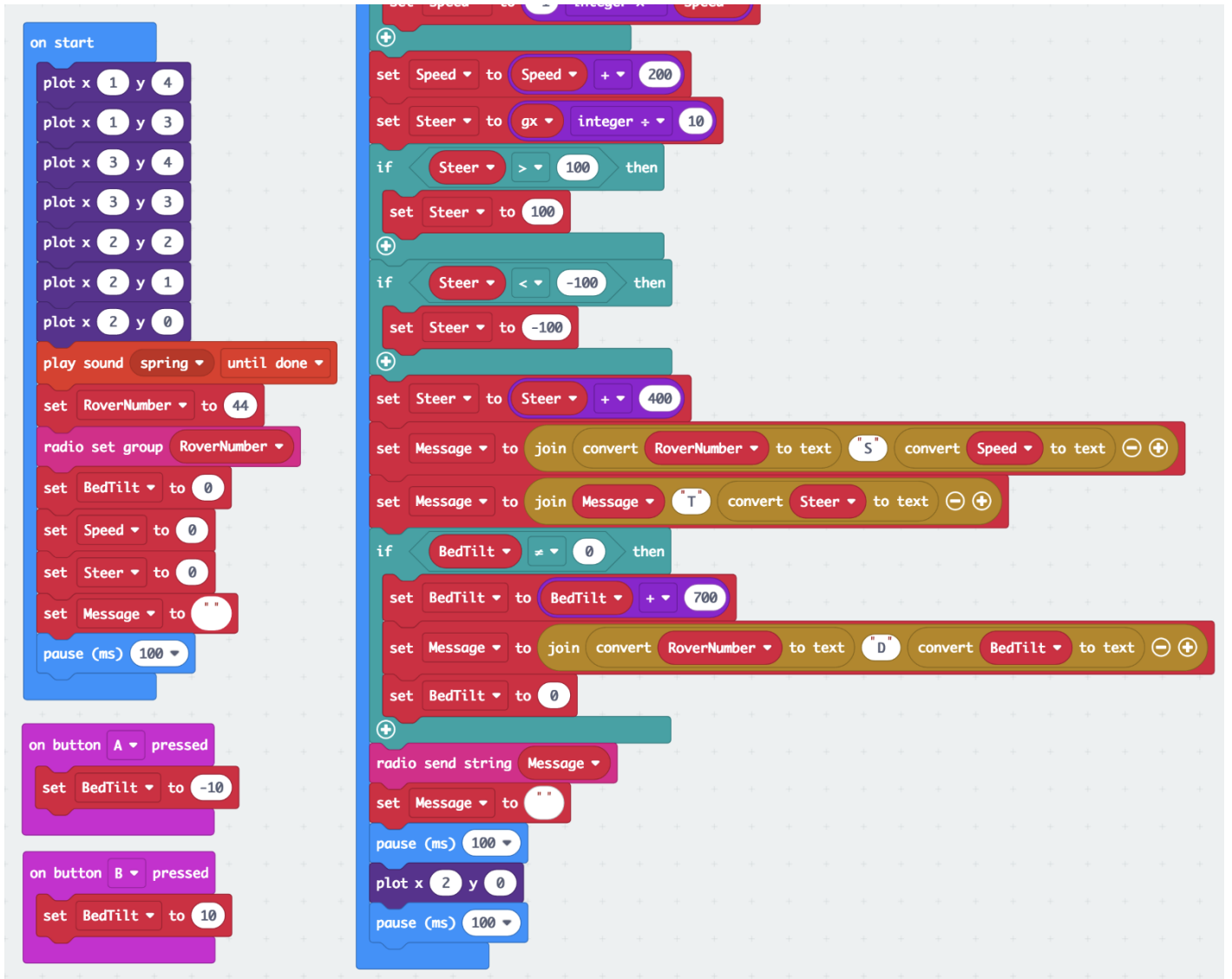


5. Modify the “forever” block:
 - Get a new "if ..." block and place it right before the "radio send string Message" block in the "forever" block.
 - Copy the "Steer > 100" comparison block from above, paste and place it over the "true" of the new "if ..." block.
 - Change the "Steer" variable here to "BedTilt".
 - Change the inequality sign from greater than to unequal.
 - Change the number 100 to zero.
 - Copy the "set Steer to Steer + 400" block from above, place it inside the new "if ..." block.
 - Change the variable name (both!) to BedTilt.
 - Change the number 400 to 700.
 - Copy the "set Message to join ... "S" convert ..." block from above.
 - Place it inside the new "if ..." block.
 - Change the "S" to a "D".
 - Change the variable "Speed" to "BedTilt".
 - Copy the "set BedTilt to 0" block in the "on start" block
 - Paste and place it at the end of new "if ..." block.
 - Place a "set Message to """ block after the "radio send string Message" block.
 - Remember, the zero needs to be replaced by the "" from the "Text" toolbox!



The code for "Rover RC 3":

Note: The "on start" block and second part of the "forever" block of "Rover RC 3" code for the adjustment of "BedLevel" and "BedTilt" variables. The top part of the "forever" block is the same as in the "Rover RC 2" code.



The image shows a Scratch script for a rover. It is divided into three main sections: an 'on start' block, two 'on button pressed' blocks, and a large 'forever' loop.

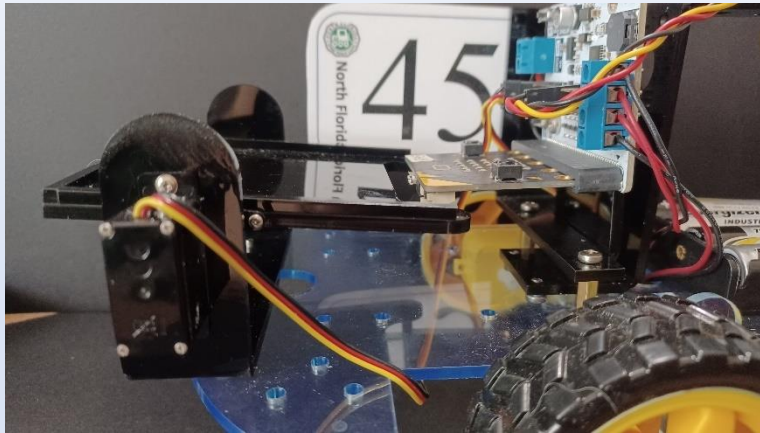
- on start:** Plots a path of points (1,4), (1,3), (3,4), (3,3), (2,2), (2,1), (2,0). Plays a 'spring' sound. Sets RoverNumber to 44, BedTilt to 0, Speed to 0, Steer to 0, and Message to "". Pauses for 100ms.
- on button A pressed:** Sets BedTilt to -10.
- on button B pressed:** Sets BedTilt to 10.
- forever loop:** Sets Speed to Speed + 200 and Steer to gx integer + 10. It has two conditional checks: if Steer > 100, set Steer to 100; if Steer < -100, set Steer to -100. Then it increments Steer by 400. It constructs a Message string: "S" followed by Speed, then "T" followed by Steer. It checks if BedTilt is not 0; if so, it increments BedTilt by 700, updates the Message with "D" and BedTilt, and resets BedTilt to 0. Finally, it sends the Message via radio, resets Message to "", and pauses for 100ms before plotting (2,0) and pausing for 100ms.

Code link "Rover RC 3": <https://makecode.microbit.org/dcFhtJ4h0fW0>



Try this: Remote Controlled Cargo Bed

1. Load the “**Cargo Bed Test 1**” on the rover micro:bit and the “**Rover RC 3**” on the remote-control micro:bit.
2. Turn on the power to the rover.
 - Does the cargo bed move to a level position?
 - If the bed is still not close to level, to back to the “Cargo Bed Test” Try This and try to get the cargo bed reasonably level.
3. Power ON the remote-control.
 - Press button B? Does the front of the cargo bed raise?
 - Try to put the back of the bed as low as it will go (the unload position) by pressing button B multiple times (about 6 presses or 60° lower).
 - Caution: Make sure the bed does not touch the chassis plate or the cargo mount when fully raised.
 - Press button A: Can you level the cargo bed with it?



We are not actually interested in moving the cargo bed in multiple positions with the remote-control. The purpose of the cargo bed is to be able to carry a load/package to a destination and then unload the package. Therefore, we only need two positions of the cargo bed to do this: level and fully tilted to cause the loaded package to unload or slide off the cargo bed when it is tilted.

Let's modify the code of both the rover and remote-control to handle this task.

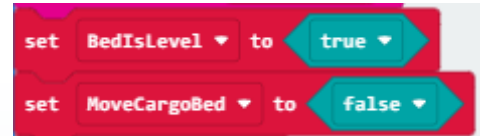


Time to Code: Rover RC 4

1. Use “**Rover RC 3**” and rename it “**Rover RC 4**”.

2. Modify the “**on start**” block:

- Create a new variable name “BedIsLevel”.
- Replace the “BedTilt” in the “set BedTilt...” block to “BedIsLevel”.
 - Replace the zero in this block with a “true” value from the “Logic” toolbox.
- Make a new variable named “MoveCargoBed”
 - Place a “set MoveCargoBed to false” right after the “set BedIsLevel to true” block.



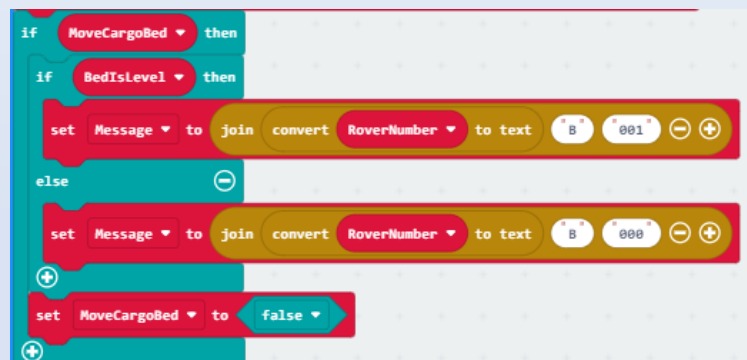
3. Modify the “**on button pressed**” blocks:

- Change the block in the “on button A pressed” block to “set BedIsLevel to true”.
- Change the block in the “on button B pressed” block to “set BedIsLevel to false”.
- Copy the “set MoveCargoBed to false” block twice, place one into the “on button A pressed” block, the other in the “on button B pressed” block.
 - Change the “false” in both these blocks to “true”.

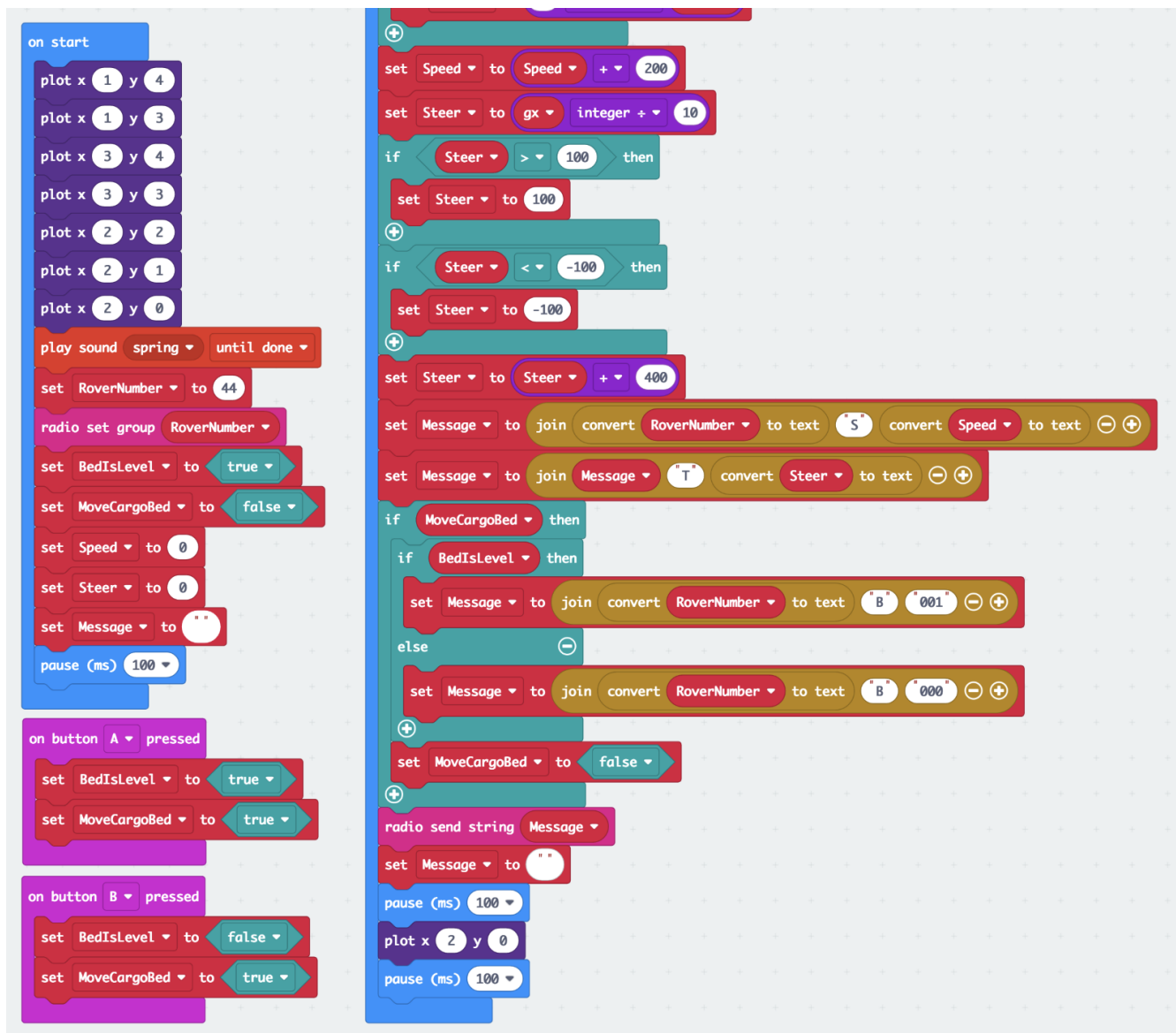


4. Modify the “**forever**” block:

- Delete the **comparison block** “BedIsLevel unequal to zero” from the last “if ...” block.
 - Replace it with just the “MoveCargoBed” variable.
- Get a new “if true then else” block and place it into the “if MoveCargoBed then” block.
 - Replace the “true” in there with the “BedIsLevel” variable.
- Drag the first “set BedTilt ...” block - with everything attached into the new “if ...” block.
 - Delete both of the “set BedTilt...” block in this “if ...” block.
 - Change the “set Message ...” block to “set Message to join “B” “001””.
 - Expand the new “if then...” block by clicking the plus (+) at the bottom.
 - Copy and paste the “set Message ...” block and place it in this else slot.
 - Change the “001” to “000” of the block in the “else” part.
- Place a new “set MoveCargoBed to false” after the “if BedIsLevel ...” block, but still inside the “if MoveCargoBed ...” block.



The code for "Rover RC 4":



The image displays a Scratch script for a Micro:bit rover. It is divided into three main sections: an 'on start' block, two 'on button pressed' blocks (A and B), and a 'forever' loop. The 'on start' block initializes variables: RoverNumber (44), BedIsLevel (true), MoveCargoBed (false), Speed (0), Steer (0), and Message (empty string). It also plays a 'spring' sound and plots several points. The 'on button A pressed' block sets BedIsLevel to true and MoveCargoBed to true. The 'on button B pressed' block sets BedIsLevel to false and MoveCargoBed to true. The 'forever' loop contains logic for speed and steering control. It sets Speed to 200 and Steer to gx + 10. If Steer > 100, Steer is set to 100. If Steer < -100, Steer is set to -100. Steer is then increased by 400. The Message variable is constructed as 'S' followed by Speed and 'T' followed by Steer. An if-else block checks MoveCargoBed; if true and BedIsLevel is true, it appends 'B' and '001' to the message, otherwise it appends 'B' and '000'. The message is sent via radio, Message is reset, and there are two 100ms pauses. The loop ends with a plot at (2, 0) and another 100ms pause.

Above: "on start" and "forever" blocks of bottom part of "Rover RC 4" with additions to control the cargo bed: the remote control's A and B buttons can flip the bed to the unload position and back level. The beginning of the "forever" block is not shown but is the same as in the "Rover RC 3" code.

"Rover RC 4": [https://makecode.microbit.org/ 8mqCk6V1Lgw7](https://makecode.microbit.org/8mqCk6V1Lgw7)

To complete our coding task, we now must modify the rover.

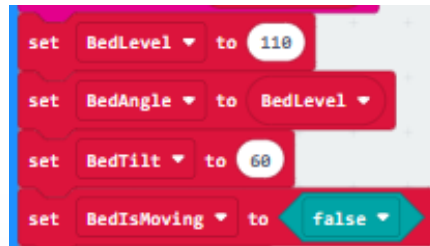


Time to Code: Cargo Bed Test 2

1. Start with “Cargo Best Test 1” and rename it “Cargo Bed Test 2”.

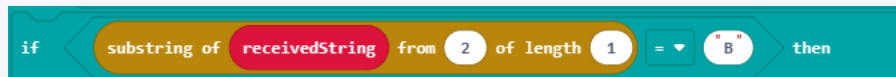
2. Modify the “on start” block:

- Adjust "set BedLevel to 100" in "on start".
 - This is the value that made your cargo bed level.
 - The example here was 110.
- Change the number 10 in "set BedTilt to ..." to 60
 - This is the value your rover needed to make bed tilt enough to unload.
 - Your value may be different.
- Create a new variable "BedIsMoving".
- Place a "set BedIsMoving to false" block after the "set BedTilt ..." block.

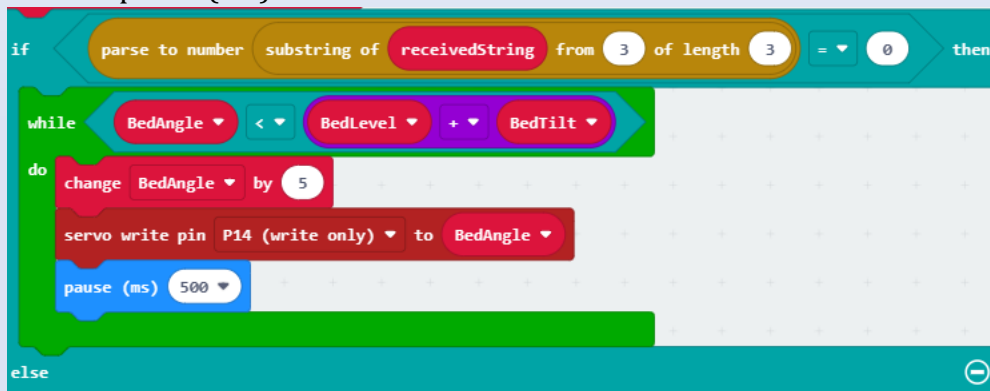


3. Modify the “on radio received....” block:

- Change the letter "D" in the "if substring of receivedString from 2 of length 1 ..." block letter "B".



- Place a "set BedIsMoving to true" block after the clear screen block.
- Place a **new** "if true then else" block after the “set BedIsMoving...” block.
 - Place a new "0 > 0" comparison block over its "true" value.
 - From below, drag the "parse to number substring of receivedString from 3 of length 3" over the first zero in the comparison block.
 - Place a "while false do" block from the “Loops” toolbox into the first blank of the “if then...” block.
 - Place a "0 < 0" comparison block over the "false" in the "while ..." block.
 - Replace the first zero with the variable "BedAngle".
 - Replace the second zero with the sum "BedLevel + BedTilt".
 - Place a "change BedAngle by 5" block from "Variables" inside the "while ..." block.
 - Place a "servo write pin P14 to BedAngle" block after the “change BedAngle...” block.
 - Place a “pause (ms) 500” block after this block



Continued next page.



Time to Code: Cargo Bed Test 2 (continued)

3. Continue modifying the “on radio received...” block:

- Copy, paste and place the whole "while ..." block into the "... else ..." slot of the "if ..." block.
 - o Change the less than sign to a greater than sign.
 - o Replace the “BedLevel” + “BedTilt” with just the "BedLevel" variable.
 - o Change the number five in the "change BedAngle by 5" in the second "while ..." block to a negative five.

```
else
  while BedAngle > 0
  do
    change BedAngle by -5
    servo write pin P14 (write only) to BedAngle
    pause (ms) 500
```

- Delete the next five (5) blocks:
 - o "set BedTilt to)" block
 - o "set BedTilt to BedTilt ..." block
 - o "set BedAngle to BedAngle ..." block
 - o "servo write pin P14..." block
 - o "pause (ms) 200" block

See the complete code on next three pages.



Try This: Cargo Bed Test 2

Now you should be able to press button A on the remote control to unload a package and then return the cargo bed back to level by pressing button B. Try it out!

- Download the “**Rover RC 4**” code to the remote-control and the “**Cargo Bed Test 2**” to the rover micro:bit.
- Turn the rover on. Is the bed level?
- Turn the remote on (hold it level).
 - Press button A
 - Does the cargo bed move?
 - Press button B
 - What happens?

The code for “Cargo Bed Test 2”:



```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  servo write pin P14 (write only) to BedAngle
  unplot x 2 y 2
```

Above: "on start" block of "Cargo Bed Test 2" used for driving the rover (Speed and Steer) and controlling the cargo bed.

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = "B" then
      Drive Wheels with Speed 0 and Steer 0
      clear screen
      set BedIsMoving to true
      if parse to number substring of receivedString from 3 of length 3 > 0 then
        while BedAngle < BedLevel + BedTilt
          do
            change BedAngle by 5
            servo write pin P14 (write only) to BedAngle
            pause (ms) 500
        else
          while BedAngle > BedLevel
            do
              change BedAngle by -5
              servo write pin P14 (write only) to BedAngle
              pause (ms) 500
          set BedIsMoving to false
      else if substring of receivedString from 2 of length 1 = "S" then
```

```
else if <substring of receivedString from 2 of length 1 = "S"> then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if SpeedNew >= -100 and SpeedNew <= 100 then
    set Speed to SpeedNew
    Drive Wheels with Speed Speed and Steer Steer
  else
  +
  +
  +
```

Above: "on radio received ..." block of "Cargo Bed Test 2". All other code blocks, the "forever", "on button A pressed", "on button B pressed", and "on logo pressed" block remain unchanged from "Cargo Bed Test 1".

Code link: "Cargo Bed Test 2": https://makecode.microbit.org/_XFh2qkiD6gfM

8.5 What are Ultrasonic Sensors?

There are many types of sensors that can be used in robotics. Look at the resources below to learn about sensors and specifically ultrasonic sensors.



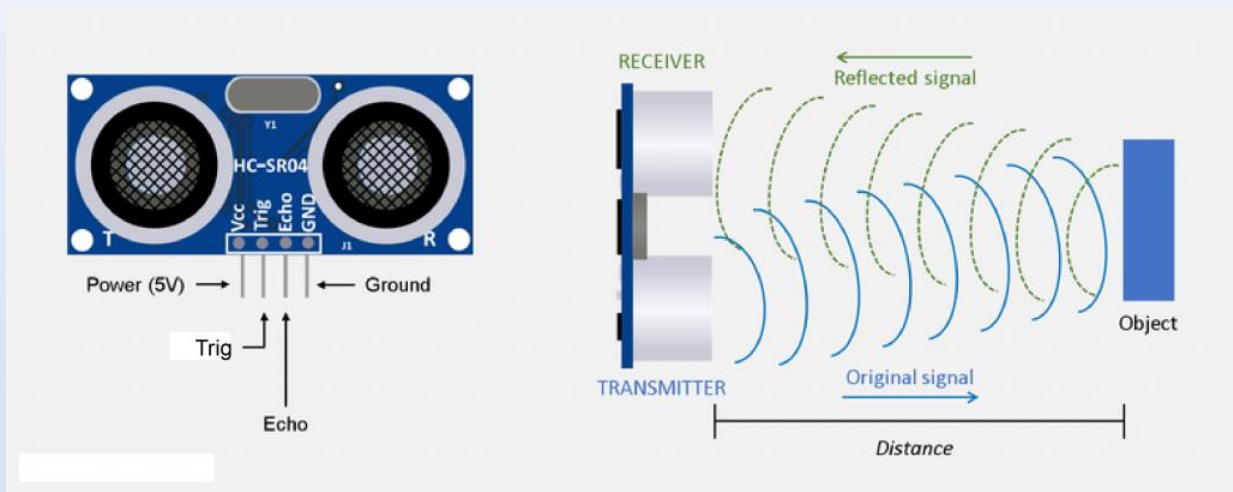
Internet Resources:

Types of sensors used in robotics:

- <https://www.wevolver.com/article/sensors-in-robotics-the-common-types>

How do Ultrasonic Distance Sensors Work?

- <https://youtu.be/2ojWO1QNprw>

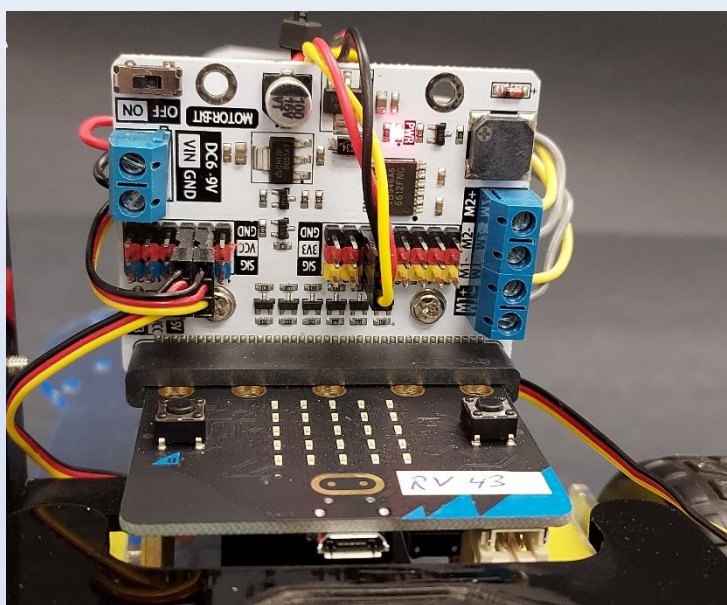


8.6 Sonar Assembly



Assembly: Sonar Assembly

1. Refer to the images on pages 10 through 16 to identify the parts needed for this part of the assembly.
2. Watch the STEM SEALs You Tube video: **LAND Sonar Assembly**.
3. Step-by step assembly instructions:



8.7 Sonar Calibration (Sonar Test 1)

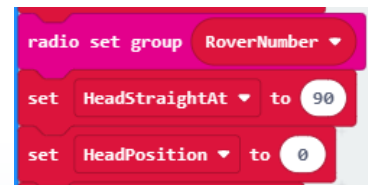
Now that you have the cargo bed calibrated and ready to use, it is time to work with the ultrasonic sensor or sonar. Just like the cargo bed, you will need to calibrate the position of the servo motor arm (with sonar attached). Also like the servo of the cargo bed, large adjustments should first be made by resetting the servo arm on the motor cog wheel. Remember the teeth of the cog wheel are 15° apart. Turn the rover on, does the sonar face forward? If it looks slightly to the left or right, then the calibration code will correct this. If it is larger than 15°, then remove the set screw and reset the sonar arm before continuing to the next coding activity.



Time to Code: Sonar Test 1

1. Start with the “Cargo Bed Test 2” and rename it “Sonar Test 1”.
2. Modify the “on start” block:

- Create a new variable, name it "HeadStraightAt".
- Place a new "set HeadStraightAt to 90" block right after the "radio set group ..." block.
- Create a new variable, name it "HeadPosition".
- Place a "set HeadPosition to 0" after the previous block.



- Place a new "pause (ms) 1000" after the "servo write pin P14 ..." block.
- Copy, paste, and place a new "servo write pin P14 ..." block below the "pause 1000 ms" block.
 - Change pin P14 in this block to P15.
 - Get a "0 + 0" addition block from "Math" and replace the "BedAngle" variable in the new "servo write pin P15 ..." block.
 - Place the variables "HeadStraightAt + HeadPosition" in this “Math” block.
 - Place a "pause 500 ms" block after the “servo write...” block.
 - Place a "set HeadPosition to -70" block next.
 - Copy the "servo write pin P15 ..." block from above and place next.
 - Place a "pause 500 ms" block after it.





Time to Code: Sonar Test 1(continued)

- Get a "while false do" block from "Loops" and place next.
 - Get a " $0 < 0$ " comparison block from "Logic" and place it on the "false" in the "while ..." block.
 - Replace the first zero with the variable "HeadPosition".
 - Type over the second zero there with the number 70.
 - Copy, paste and place a "servo write pin P15 ..." block from above and place it inside the "while ..." block.
 - Place a "pause 100 ms" block next.
 - Place a "change HeadPosition by 5" block at the end of the "while ..." block.
- Place another "pause 100 ms" block after the "while ..." block.
- Place a "set HeadPosition to 0" block next.
- Add another "servo write pin P15 ..." next (before the "unplot ..." block).

```
while HeadPosition < 70 do
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  pause (ms) 100
  change HeadPosition by 5
pause (ms) 100
set HeadPosition to 0
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
unplot x 2 y 2
```



Try This: Sonar Test 1

1. Load the "Sonar Test 1" code on the rover.
2. Turn the Rover on.
3. Is the cargo bed level?
4. Does the sonar head turn?
 - Sweep from side to side?
 - Does it end up looking forward?
5. Calibrating the sonar head.
 - If it is not perfectly straight at 90° , estimate the angle and change the number in this line of code.
 - If it looks more to the right, decrease the number.
 - If it looks more to the left, increase the number.

The code for “**Sonar Test 1**”:

```
radio set group RoverNumber
set HeadStraightAt to 84
set HeadPosition to 0
set BedLevel to 105
set BedAngle to BedLevel
set BedTilt to 60
set BedIsMoving to false
set Speed to 0
set SpeedNew to 0
servo write pin P14 (write only) to BedAngle
pause (ms) 1000
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
pause (ms) 500
set HeadPosition to -70
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
pause (ms) 500
while HeadPosition < -70
do
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  pause (ms) 100
  change HeadPosition by 5
pause (ms) 100
set HeadPosition to 0
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
unplot x 2 y 2
set HeadStraightAt to 90
```

Above: Second part of the “on start” block of “**Sonar Test 1**”, turning the sonar first straight, then left, sweeping to the right, and then looking straight forward again.

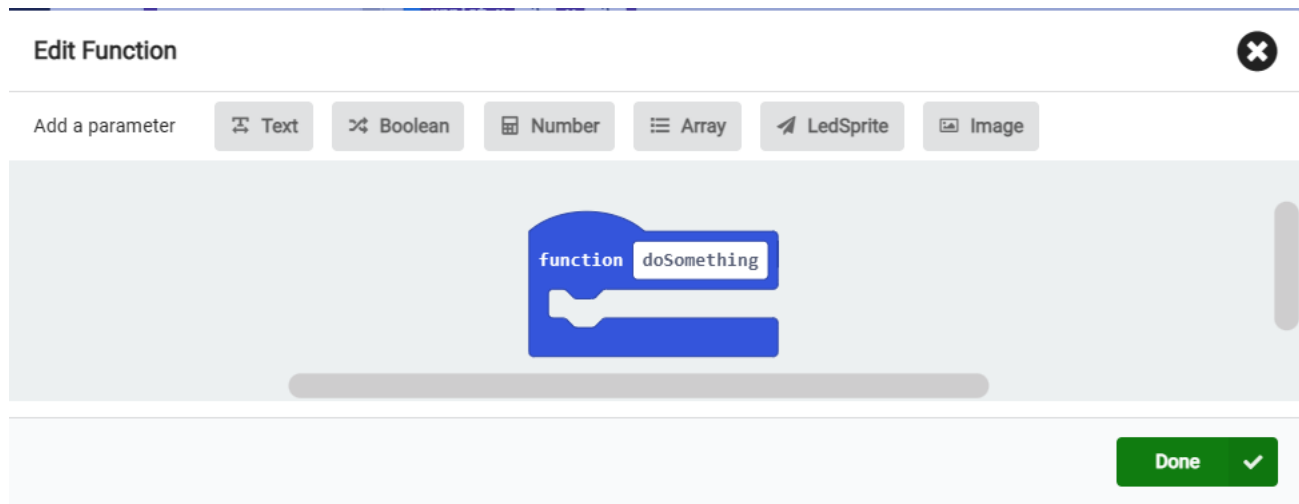
Code link “**Sonar Test 1**”: https://makecode.microbit.org/_Lta7U9Y9seEz

8.8 Improving the Code with Functions (Sonar Test 2)

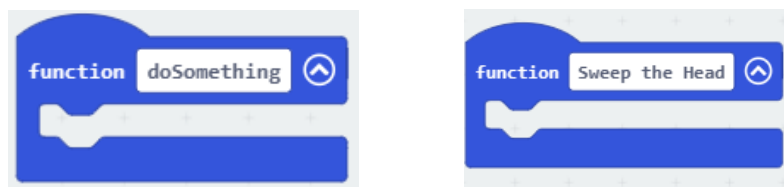
In this section, the code will be improved and simplified by using a special block called a “function” and a feature will be added to the code to allow the sonar head to turn in the direction the rover is driving which will later be useful when detecting obstacles.

A function lets you create a portion of code that you can reuse in various places in your program without having to repeat or copy the sometimes-lengthy code each time. Instead, you can just call the function name in a single block.

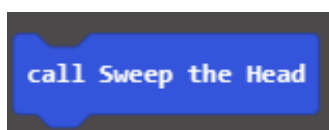
When you are using a function in MakeCode, a new edit box will appear- **just click on the done button.**



In the workspace, you will type the name of the new function where it says “doSomething”.



The name of the function usually gives an indication of what you want the code to do. When you want to call (use) the function, the new function name will be listed in the “functions” toolbox.



Let's use this function now!



Time to Code: Sonar Test 2

1. Start with "Sonar Test 1" and rename "Sonar Test 2".
2. **Create a function** from the ("Advanced") "Functions" toolbox and name it "Sweep the Head".
3. In the "on start" block, drag all blocks beginning with the "pause 500 ms" block after the first "servo write pin P15 ..." block into the function.
4. Drag the last block "unplot x 2 y 2" back into the "on start" block,
5. Get a "call Sweep the Head" block from "Functions" and place it before the "unplot x 2 y 2" block in the "on start" block.

The function should look like this:

```
function Sweep the Head
  pause (ms) 500
  set HeadPosition to -70
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  pause (ms) 500
  while HeadPosition < 70
  do
    servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    pause (ms) 100
    change HeadPosition by 5
  end
  pause (ms) 100
  set HeadPosition to 0
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
```

The end of the "on start" block looks like this:

```
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
call Sweep the Head
unplot x 2 y 2
```

Continued next page.



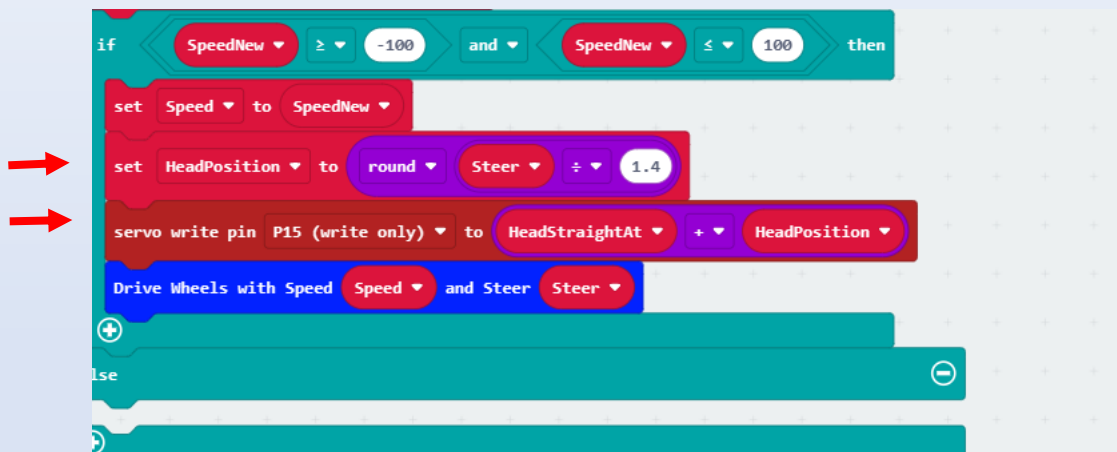
Time to Code: Sonar Test 2 (continued)

We also want the head to turn in the direction the rover drives:

6. Modify the "on radio received..." block:
 - Get a "set HeadPosition to 0" block from "Variables" and place it before the "DriveWheels ..." block at the beginning of the "on radio received ..." block.
 - Get a "round 0" block from "Math" and drop it on the zero in the "set HeadPosition" block.
 - Get a "0 divided by 0" block from "Math" and drop it on the zero in the "round ..." block.
 - Type the number 1.4 over the second zero in this block.
7. Copy the "servo write pin P15 ..." block from the "on start" block.
 - Paste it after the new "set HeadPosition..." block.



8. Copy and paste these two new blocks ("set HeadPosition" and "servo write P15...").
9. Drag them to the end of the "on radio received ..." block and drop them there before the "Drive Wheels with Speed Speed and Steer Steer" block.
10. Place a "Steer" variable in over the zero in the "set HeadPosition to round 0 divided by 1.4".





Try This: Sonar Test 2

1. Load the “Sonar Test 2” code to the rover (make sure it is OFF while loading!)
2. Turn ON the rover.
3. Does the cargo bed more into the horizontal position (level)?
4. Does the sonar head sweep from the left to the right, and then center itself?
5. Turn the RC on.
6. Can you drive and steer the rover?
7. Does the sonar head turn in the direction the rover is traveling?

The code for “Sonar Test 2” should look like this:

```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set HeadStraightAt to 84
  set HeadPosition to 0
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  servo write pin P14 (write only) to BedAngle
  pause (ms) 1000
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  call Sweep the Head
  unplot x 2 y 2
```

Above: "on start" block of "Sonar Test 2" code, sweeping the sonar head left to right, then centering it.
(continued)

```
function Sweep the Head
  pause (ms) 500
  set HeadPosition to -70
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  pause (ms) 500
  while HeadPosition < 70
  do
    servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    pause (ms) 100
    change HeadPosition by 5
  pause (ms) 100
  set HeadPosition to 0
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
```

Above: "function Sweep the Head" block of "Sonar Test 2", enabling the sonar head to sweep left to right, then centering.

(continued)

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = "B" then
      set HeadPosition to round 0 ÷ 1.4
      servo write pin P15 (write only) to HeadStraightAt + HeadPosition
      Drive Wheels with Speed 0 and Steer 0
      clear screen
      set BedIsMoving to true
      if parse to number substring of receivedString from 3 of length 3 > 0 then
        while BedAngle < BedLevel + BedTilt
          do
            change BedAngle by 5
            servo write pin P14 (write only) to BedAngle
            pause (ms) 500
        else
          while BedAngle > BedLevel
            do
              change BedAngle by -5
              servo write pin P14 (write only) to BedAngle
              pause (ms) 500
          set BedIsMoving to false
      else if substring of receivedString from 2 of length 1 = "S" then
```

(continued)

```

else if < substring of receivedString from 2 of length 1 = "S" > then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if < SpeedNew ≥ -100 and SpeedNew ≤ 100 > then
    set Speed to SpeedNew
    set HeadPosition to round Steer ÷ 1.4
    servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    Drive Wheels with Speed Speed and Steer Steer
  else

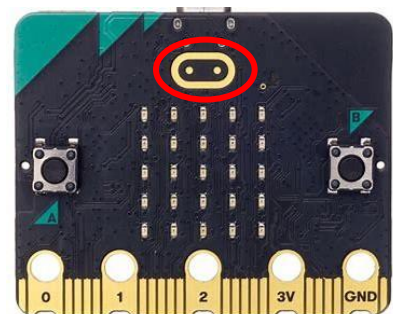
```

Above: "on radio received receivedString" block of "Sonar Test 2" code, enabling the rover to turn the head in the direction it is driving. All other blocks stay the same.

Code link "Sonar Test 2": <https://makecode.microbit.org/F2UTLsUc2UVW>

8.9 Testing the Sonar (Sonar Test 3 & 4, Rover RC 5)

While having a remote control(RC) to drive the rover is handy, it is not always desirable to use the RC when testing code on the rover. We can add a feature to the remote control that will allow it to be turned on and off as needed. The on-button A pressed and on-button B pressed have already been utilized but the new V2 micro:bit has a touch sensitive sensor where the logo is on the face just above the LED display. Let's create some code to use this as an ON/OFF switch for the remote control. Then we will create some code that will allow us to test the functionality of the ultrasonic sensor or sonar.



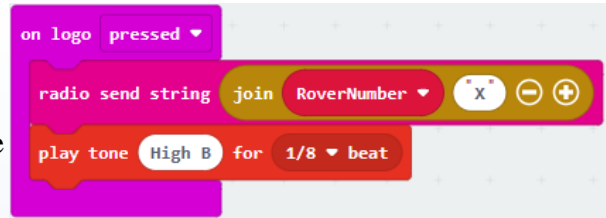


Time to Code: Rover RC 5

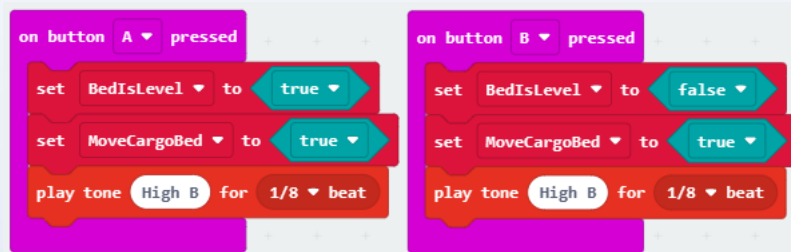
1. Use “Rover RC 4” and rename it “Rover RC 5”.

Code for the “on logo pressed” ON/Off function:

2. Drag an “on logo pressed” block from “Input” to the workspace.
 - Insert a "radio send string ..." block from "Radio" into the "on logo ..." block.
 - Drag a “join Hello World” block from “Text” and drop it over the “ ” place of the "radio send string ..." block.
 - Drag the “RoverNumber” variable from “Variables” and drop it over the “Hello” in the “join ...” block.
 - Write an "X" (just the letter X!) to replace the "World" of the "radio send string ..." block.
 - Add a “play tone High B for 1/8 beat” block thereafter.

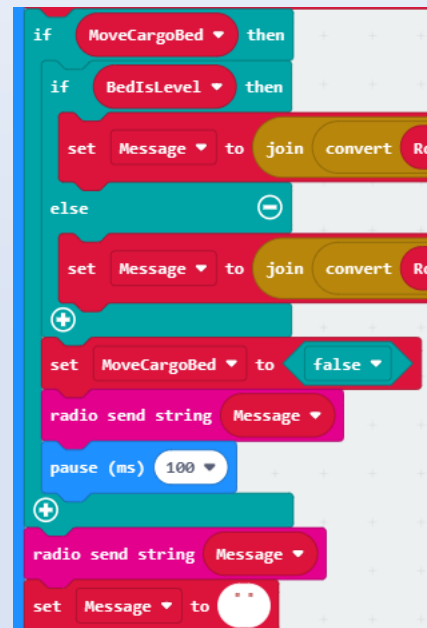


3. Add the same “play tone High B...” block to the end of both the “on A pressed” and “on B pressed” input blocks.
 - Copy twice and paste one to the end of the “on A pressed” and “on B pressed” blocks.



Code to avoid missed radio commands:

4. At the end of the "forever" block, copy, paste and place the "radio send string Message" block.
 - Paste the new “radio send Message...” block inside the “if then” block right after the “set MoveCargoBed...” block.
 - Add a "pause 100 (ms)" block right after (also, inside the "if ..." block).



Continue next page.

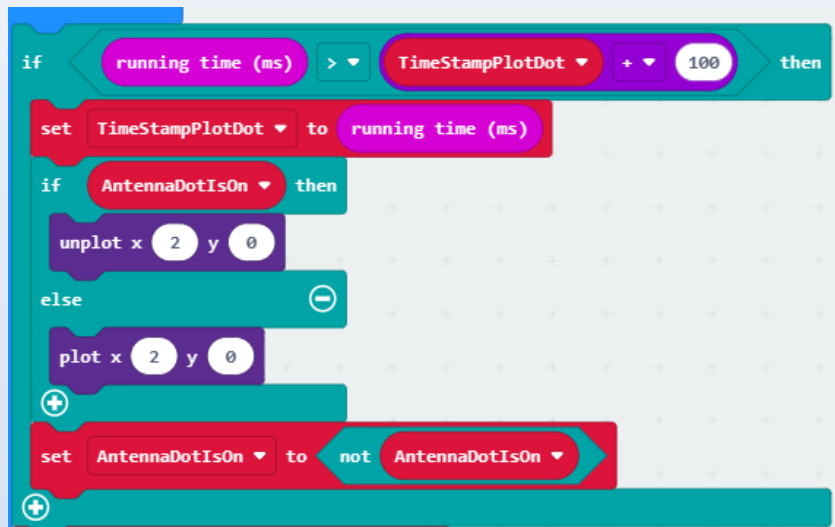


Time to Code: Rover RC 5 (continued)

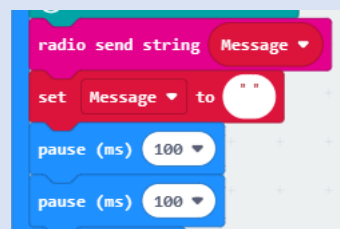
Code for replacing some “pause” block with a timer:

5. Modify the “forever” block:

- Make two new variables: “TimeStampPlotDot” and “AntennaDotIsOn”
- Drag a new “if then” block and place it under the “uplot x 2 y 0” block at the beginning of the “on start” block.
 - Place a comparison “0 > 0” over the “true” in this block.
 - Drag a “running time (ms)” from “Input” “more” over the first zero.
 - Drag a “0 + 0” from “Math” and place over the second zero.
 - Place the variable “TimeStampPlotDot” where the first zero is in the “0 + 0”.
 - Change the 0 to 100 of the second zero.
 - Place a “set TimeStampPlotDot to 0” in the “if then block”
 - Place a “running time (ms)” over zero.
 - Get a new “if then else” block and place after the “set TimeStampPlotDot...” block.
 - Place the variable “AntennaDotIsOn” over the “true”.
 - Drag the “uplot x 2 y 0” block into this new “if then...” block after then.
 - Drag the “plot x 2 y 0” block from the end of the “on start” block into the “else” opening of this “if then else” block.
 - Place a “set AntennaDotIsOn to 0” after the last “if then else” block but still within the first “if then” block.
 - Place “not” from the “Logic” toolbox over the zero.
 - Place the variable “AntennaDotIsOn” in the open space in “not”.



- **Delete** the two “pause (ms) 100” at the very end of the “forever” block.



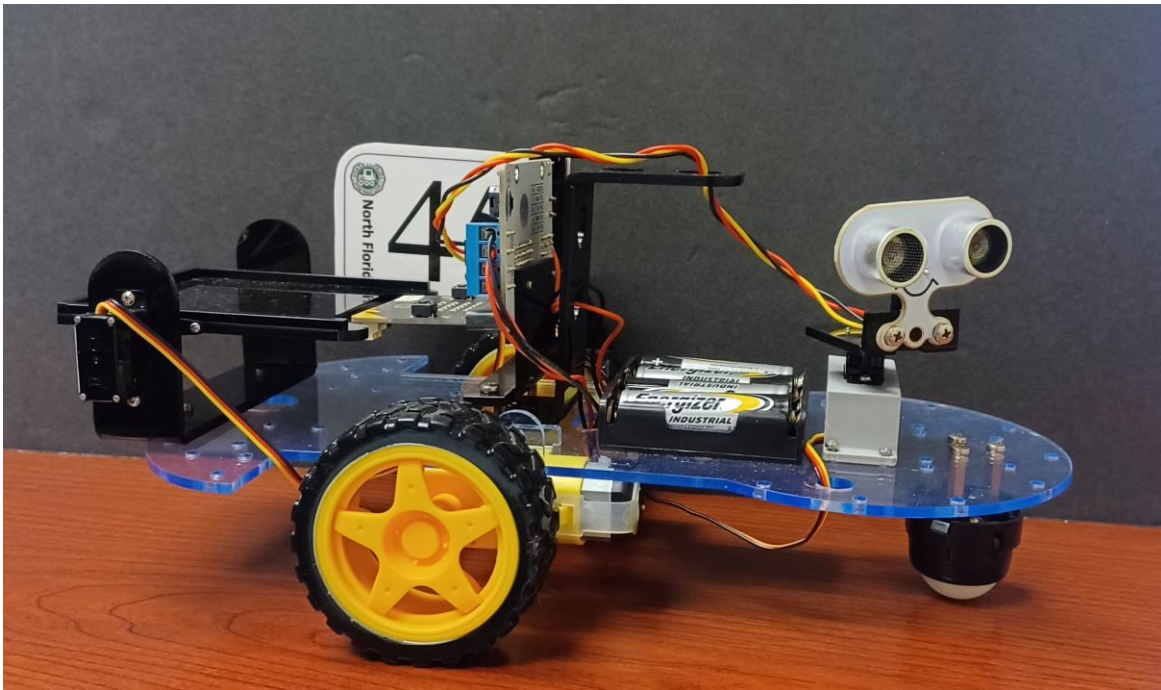
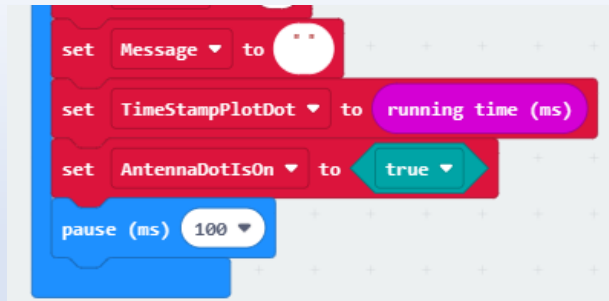
Continued next page.



Time to Code: Rover RC 5 (continued)

3. Modify the “on start” block:

- Copy the “set TimeStampPlotDot to running time (ms)” in the “forever” block.
 - Paste the “set TimeStampPlotDot...” block after the “set Message to...” block.
- Get a “set AntennaDotIsOn to 0” and place after this block
 - Drag a “true” from “Logic” over the zero.



Your code should look like this for “Rover RC 5”:

The image displays two columns of Scratch code blocks. The left column is an 'on start' block containing: seven 'plot x y' blocks with coordinates (1,4), (1,3), (3,4), (3,3), (2,2), (2,1), and (2,0); a 'play sound spring until done' block; 'set RoverNumber to 44'; 'radio set group RoverNumber'; 'set BedIsLevel to true'; 'set MoveCargoBed to false'; 'set Speed to 0'; 'set Steer to 0'; 'set Message to ""'; 'set TimestampPlotDot to running time (ms)'; 'set AntennaDotIsOn to true'; and a 'pause (ms) 100' block. The right column is a 'forever' loop starting with an 'if running time (ms) > TimestampPlotDot + 100 then' block. Inside this loop, it sets 'TimestampPlotDot' to 'running time (ms)'. An 'if AntennaDotIsOn then' block contains 'unplot x 2 y 0'. An 'else' block contains 'plot x 2 y 0'. This is followed by 'set AntennaDotIsOn to not AntennaDotIsOn'. Then, it calculates 'gx' and 'gy' as 'acceleration (mg) x' and 'y' respectively, and 'Speed' as the 'square root' of 'gx x gx + gy x gy'. 'Speed' is then rounded to an 'integer + 10'. An 'if Speed > 100 then' block sets 'Speed' to 100. Another 'if gy > 0 then' block sets 'Speed' to '-1 integer x Speed'.

Above: "on start" block and the beginning of the "forever" block of "Rover RC 5" code. The code is replacing some "pause ..." block with a timer functionality to avoid unresponsiveness when the micro:bit is halted by "pause ..." commands.

(continued)

```

on button A pressed
  set BedIsLevel to true
  set MoveCargoBed to true
  play tone High B for 1/8 beat

on button B pressed
  set BedIsLevel to false
  set MoveCargoBed to true
  play tone High B for 1/8 beat

on logo pressed
  radio send string join RoverNumber "X"
  play tone High B for 1/8 beat

forever loop
  set Steer to Steer + 400
  set Message to join convert RoverNumber to text "S" convert Speed to text
  set Message to join Message "T" convert Steer to text
  if MoveCargoBed then
    if BedIsLevel then
      set Message to join convert RoverNumber to text "B" "001"
    else
      set Message to join convert RoverNumber to text "B" "000"
  set MoveCargoBed to false
  radio send string Message
  pause (ms) 100
  radio send string Message
  set Message to ""

```

Above: "on ... pressed" blocks and the end of the modified "forever" block of the "Rover RC 5" code. The double sending of the "Message" helps to avoid radio commands being missed by the rover because it is busy doing something else.

The "Rover RC 5" code: https://makecode.microbit.org/_DMzFozEx1hq5

Now it is time to modify the rover code to accept the ON/Off functionality now coded on the remote control.

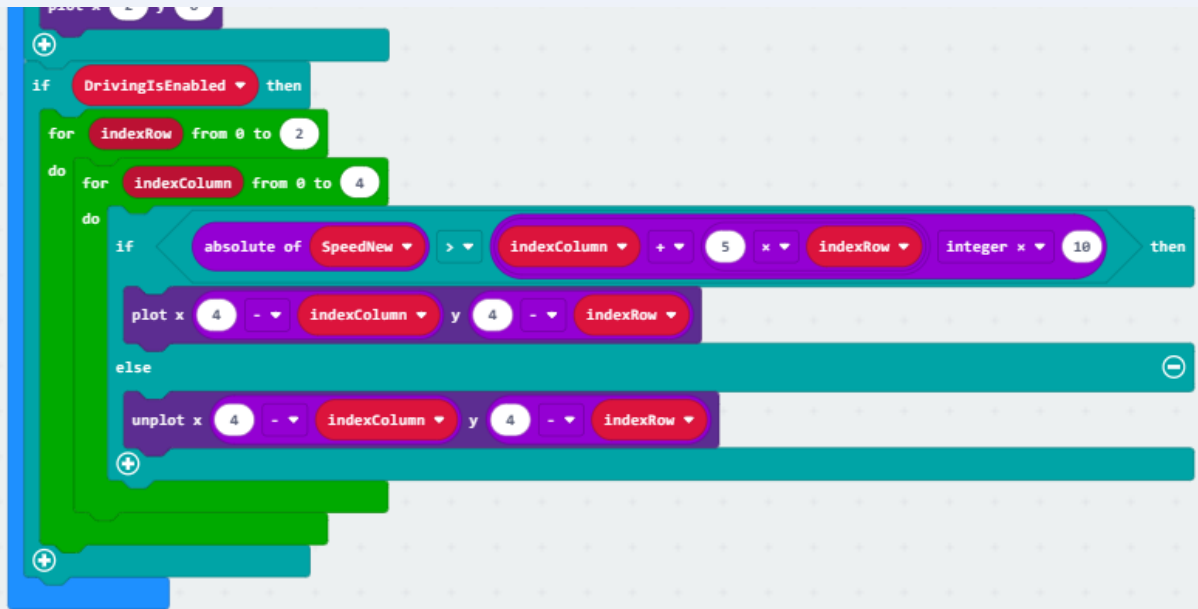


Time to Code: Sonar Test 3

1. Use “**Sonar Test 2**” and rename it “**Sonar Test 3**”
2. Create a new variable and name it “**DrivingIsEnabled**”.
3. Modify the “**on start**” block:
 - Get a “**set DrivingIsEnabled**” block and place it right before the “**servo write pin P14**” block.
 - Place a “**true**” over the zero in the “**set DrivingIsEnabled**” block.



4. Modify the “**forever**” block:
 - Place a new “**if then...**” block **after** the “**if then else**” block and before the “**for indexRow from 0 to 2**” block in the “**forever**” block.
 - Place the variable “**DrivingIsEnabled**” in the new “**if then**” block over the “**true**”.
 - Drag all the remaining blocks after the new “**if then**” block into the “**if then**” block.



Continued next page.

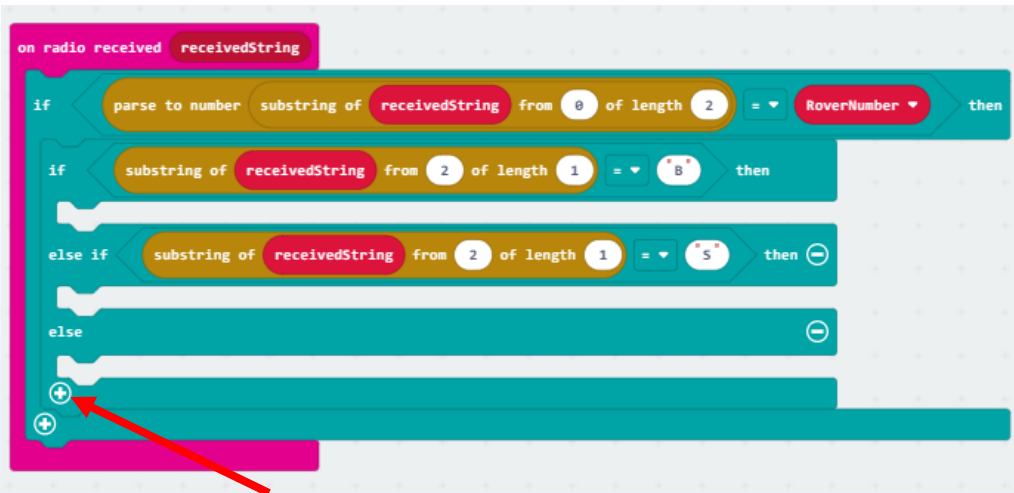


Time to Code: Sonar Test 3 (continued)

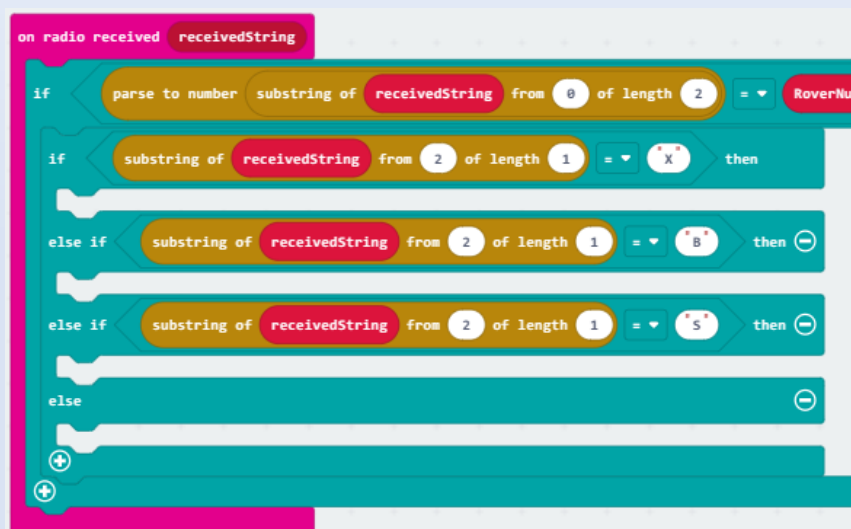
5. Modify the “on radio received” block:

Note: Much of the code in this modification requires rearranging already written code. To make this easier, the code will be pulled out of the block and left on the workspace (in ghost form) and then pulled back in as needed.

- Refer to the picture below and drag most of the blocks out of the “on radio received” block and just place them in an empty spot on the workspace – DO NOT trash them!
 - The blocks should be removed with only two drags, and you want them in big parts.
 - It should resemble this:



- Click on the “+” of the inner “if then else if else” block to add another “else if” space.
- Move the comparison with “substring of receivedString...= S” to the newly opened “else if” block.
- Move the comparison with “substring of receivedString...= B” where the comparison with “substring of receivedString...= S” was.
- Copy the comparison with “substring of receivedString...= B” and place it in the open space just created.
 - Change the “B” to the letter “X”.





Time to Code: Sonar Test 3 (continued)

5. Continue working in the “on radio received” block:

- Add the following new blocks to open space after “if substring of receivedString.... = X”.
- Refer to the picture.

```
if substring of receivedString from 2 of length 1 = "X" then
  set DrivingIsEnabled to not DrivingIsEnabled
  if not DrivingIsEnabled then
    show image icon image [ ] at offset 0
  else
    clear screen
  pause (ms) 100
```

- Add the following new blocks to open space after “if substring of receivedString.... = B”.
- Refer to the picture.

```
else if substring of receivedString from 2 of length 1 = "B" then
  if not DrivingIsEnabled then
    show image icon image [ ] at offset 0
```

- Add the following new blocks to open space after “if substring of receivedString.... = S”.
- Refer to the picture.

```
else if substring of receivedString from 2 of length 1 = "S" then
  if DrivingIsEnabled then
```

Continued next page.



Time to Code: Sonar Test 3 (continued)

- Continue working in the “on radio received” block:
 - For the rest of the code, move the “ghosted” blocks we placed on the workspace into the appropriate places.
 - Start at the top of the code and move towards the bottoms by moving the already written code into place.
 - Refer to the series of pictures.
 - Note that the last line of code is repeated in the next picture for a reference point.

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = 'X' then
      set DrivingIsEnabled to not DrivingIsEnabled
      if not DrivingIsEnabled then
        show image icon image at offset 0
      else
        clear screen
        pause (ms) 100
    else if substring of receivedString from 2 of length 1 = 'B' then
      set HeadPosition to round 0 ÷ 1.4
      servo write pin P15 (write only) to HeadStraightAt + HeadPosition
      Drive Wheels with Speed 0 and Steer 0
      clear screen
      if not DrivingIsEnabled then
```

Continued next page.



Time to Code: Sonar Test 3 (continued)

- Continue working in the “on radio received” block:
 - Continue moving the “ghosted” already written code from the workspace to the appropriate places.

```
if not DrivingIsEnabled then
  show image icon image [X] at offset 0
  set BedIsMoving to true
  if parse to number substring of receivedString from 3 of length 3 > 0 then
    while BedAngle < BedLevel + BedTilt
      do
        change BedAngle by 5
        servo write pin P14 (write only) to BedAngle
        pause (ms) 500
    else
      while BedAngle > BedLevel
        do
          change BedAngle by -5
          servo write pin P14 (write only) to BedAngle
          pause (ms) 500
      set BedIsMoving to false
  else if substring of receivedString from 2 of length 1 = 5 then
```

Continued next page.



Time to Code: Sonar Test 3 (continued)

5. Continue working in the “on radio received” block:

- Continue moving the “ghosted” already written code from the workspace to the appropriate places.

```
else if substring of receivedString from 2 of length 1 = 'S' then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if SpeedNew >= -100 and SpeedNew <= 100 then
    set Speed to SpeedNew
    set HeadPosition to round Steer / 1.4
    servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    if DrivingIsEnabled then
      Drive Wheels with Speed Speed and Steer Steer
  else
  
```

Let's do a quick test to see how the rover operates now:

Load the “**Sonar Test 3**” to the rover micro:bit and the “**Rover RC 5**” to the remote-control micro:bit.

- Turn the rover ON first:
 - Is the bed level?
 - Does the head move, then look straight?
 - The rover does not move but blinks "Ready" on the micro:bit?
- Turn the RC ON:
 - The rover comes "alive" (does the head move a little?)
 - Does the rover drive with the RC?
 - Does the rover steer and move its head?
 - Does the rover indicate the speed it is driving?

- push button A (left) on the RC:
 - Does the rover stop?
 - Does the bed move to unload?
- Push button B (right) on the RC.
 - Does the bed move to level?
- Touch the logo on the RC:
 - Does the rover display the "X"?
 - Does the rover still turn its head (controlled by the RC)?
 - Does the rover move?
- Test buttons A and B again.

The complete code for “**Sonar Test 3**” should look like this:

```

on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set HeadStraightAt to 84
  set HeadPosition to 0
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  set DrivingIsEnabled to true
  servo write pin P14 (write only) to BedAngle
  pause (ms) 1000
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  call Sweep the Head
  unplot x 2 y 2
  
```

Above: "on start" block of "Sonar Test 3" code with the ability to enable or disable remote-controlled driving.

(continued)


```

on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = 'X' then
      set DrivingIsEnabled to not DrivingIsEnabled
      if not DrivingIsEnabled then
        show image icon image [X] at offset 0
      else
        clear screen
        pause (ms) 100
    else if substring of receivedString from 2 of length 1 = 'B' then
      set HeadPosition to round 0 ÷ 1.4
      servo write pin P15 (write only) to HeadStraightAt + HeadPosition
      Drive Wheels with Speed 0 and Steer 0
      clear screen
      if not DrivingIsEnabled then

```

Above: First of three parts of the “on radio received” block of the “Sonar Test 3” code.

(continued)

```

if not DrivingIsEnabled then
  show image icon image [X] at offset 0
  set BedIsMoving to true
  if parse to number substring of receivedString from 3 of length 3 > 0 then
    while BedAngle < BedLevel + BedTilt
      do
        change BedAngle by 5
        servo write pin P14 (write only) to BedAngle
        pause (ms) 500
    else
      while BedAngle > BedLevel
        do
          change BedAngle by -5
          servo write pin P14 (write only) to BedAngle
          pause (ms) 500
      set BedIsMoving to false
  else if substring of receivedString from 2 of length 1 = 'S' then

```

Above: Second of three parts of the “on radio received” block of the “Sonar Test 3” code.

(continued)

```

else if substring of receivedString from 2 of length 1 = 5 then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if SpeedNew >= -100 and SpeedNew <= 100 then
    set Speed to SpeedNew
    set HeadPosition to round Steer / 1.4
    servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    if DrivingIsEnabled then
      Drive Wheels with Speed Speed and Steer Steer
    else
  else

```

Above: Last of three parts of the “on radio received” block of the “Sonar Test 3” code.
 The “Sonar Test 3” code: <https://makecode.microbit.org/4CiAx0YE9L0q>

How does an ultrasonic distance sensor work?

Before we test out the functionality of the sonar, let’s review a bit about how a sonar works. Sonars or ultrasonic distance sensors work on the principles of echo and reflection.

$$\begin{aligned} \text{speed of sound: } 343 \text{ m/s} &= 0.343 \text{ m/ms} = 343 \text{ mm/ms} \approx 1 \text{ ft/ms} \\ &= 34.3 \text{ cm/ms} = 3.43 \text{ cm}/0.1 \text{ ms} = 3.43 \text{ cm}/100 \text{ ms} = 1 \text{ cm}/29 \text{ ms} \end{aligned}$$

$$d \text{ (distance)} = v \text{ (velocity)} t \text{ (time)}$$

The micro:bit measures the time between pulse and echo – that is the time for the sound to get to the obstacle and come back. Then we compute the distance as

$$d = v t / 2 = (v/2) t = t * 1 \text{ cm} / (2 * 29 \text{ } \mu\text{s}) = (t / \mu\text{s}) / 58 \text{ in centimeters.}$$

Note: motion sensors (such as automatic light switches) may interfere with the sonar function!

We also need to modify the code just a bit so we can get a read out of the sonar data.



Time to Code: Sonar Test 4

2. Use “Sonar Test 3” and rename it “Sonar Test 4”.
3. Modify the “forever” block:
 - Rearrange the blocks in the “forever” block.
 - First pull all blocks out of the “forever” block and place them on the workspace as “ghost” blocks. All the blocks will move in one drag.
 - Separate the blocks into two parts by dragging the lower blocks starting with the “if DrivingIsEnabled then” block.
 - Place the “if DrivingIsEnabled then” and all the blocks that follow it at the beginning of the “forever” block.
 - Place the remaining “ghost” block after the blocks in the “if DrivingIsEnabled then” but also inside the “if DrivingIsEnabled then” block.
 - The “forever” block should now look like this:

```
forever
  if DrivingIsEnabled then
    for indexRow from 0 to 2
    do
      for indexColumn from 0 to 4
      do
        if absolute of SpeedNew > indexColumn + 5 * indexRow Integer > 10 then
          plot x 4 - indexColumn y 4 - indexRow
        else
          unplot x 4 - indexColumn y 4 - indexRow
        +
      do
        for index from 0 to 2
        do
          unplot x 3 - index y 0
        +
      pause (ms) 200
      if SpeedNew < 0 then
        for index from 0 to 2
        do
          plot x 3 - index y 0
        +
      else
        plot x 2 y 0
      +
    +
  +
```

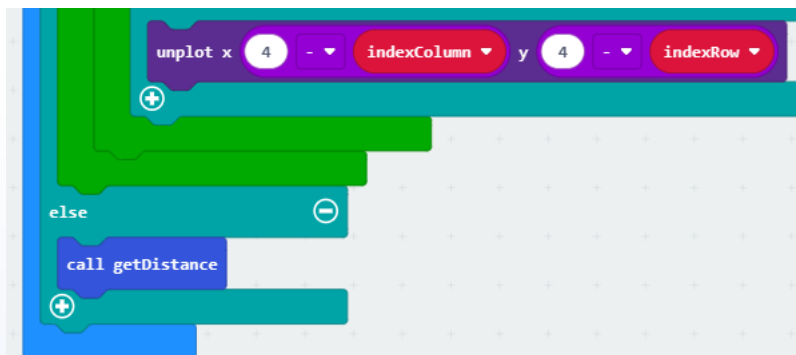
Continued next page.



Time to Code: Sonar Test 4 (continued)

4. Continue to modify the “forever” block:

- Click on the “+” sign of the last “if DrivingIsEnabled...” block.
- Make a function from “Advanced” “functions” toolbox and name it “getDistance”
- From “functions” now get a “call getDistance” and place it in the newly created “else” part of the “if DrivingIsEnabled...” block.



5. Complete the function “getDistance” block: (see next page for picture)

- From the “Advanced” “Pins” “... more” toolbox, place a “set pull pin P0 to up” block into the “function getDistance” block.
 - Change the pin 0 to pin P9.
 - Change the “up” to “none”,
- Place a “digital write pin P9 to 0” from the “Pins” toolbox after the “set pull pin ...” block.
- Place a “wait (µs) 2” block from the “Control” toolbox at the end of the “function ...” block.
 - Change the 4 to 2.
- Copy the “digital write pin P9 ...” block and paste twice, place both after the “wait ...” block.
 - Change the zero in the second “digital write pin P9 ...” block to the number 1.
- Copy the “wait ...” block, paste and place it before the last “digital write pin ...” block.
 - Change the number 2 to the number 10.
- Place a “set Distance to 0” block from “Variables” at the end of the “function getDistance” block.
 - From the “Pins” “... more” toolbox, place a “pulse in (µs) pin P0 pulsed high” block over the zero in the “set Distance ...” block.
 - Change pin P0 to pin P9.
- Get another “set Distance ...” block from “Variables” and place it after the one just created.
 - Get a “square root 0” block from “Math” and place it over the zero in the last “set Distance ...” block.
 - Change the “square root” in there to “integer division”.
 - Drag a “Distance” variable from “Variables” and drop it over the first zero in the integer division.
 - Replace the second zero with the number 58.
 - This will give us the distance from the sonar sensor in centimeters.

Continued next page.



Time to Code: Sonar Test 4 (continued)

- Continue with the “getDistance” block:
 - After the last block add a “show number” from “Basic” toolbox.
 - Place the variable “Distance” over the zero.
 - Next add a “pause (ms) 100”.
 - Next Add a “clear screen”.
 - And lastly add another “pause (ms) 100”.

The function “getDistance” should look like this:

```
function getDistance
  set pull pin P9 to none
  digital write pin P9 to 0
  wait (µs) 2
  digital write pin P9 to 1
  wait (µs) 10
  digital write pin P9 to 0
  set Distance to pulse in (µs) pin P9 pulsed high
  set Distance to Distance integer ÷ 58
  show number Distance
  pause (ms) 100
  clear screen
  pause (ms) 100
```

- Modify the “on start” block:
 - Change the “true” in the “set DrivingIsEnabled ...” block to “false”.
 - Place a “set Distance to 0” block right after the “set DrivingIsEnabled to false” block.

```
set SpeedNew to 0
set DrivingIsEnabled to false
set Distance to 0
servo write pin P14 (write only) to BedAngle
```

The “**Sonar Test 4**” code should look like this:

```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set HeadStraightAt to 84
  set HeadPosition to 0
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  set DrivingIsEnabled to false
  set Distance to 0
  servo write pin P14 (write only) to BedAngle
  pause (ms) 1000
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  call Sweep the Head
  unplot x 2 y 2
```

Above: "on start" block of "Sonar Test 4" including the "Distance" variable.

(continued)

```
function getDistance
  set pull pin P9 to none
  digital write pin P9 to 0
  wait (μs) 2
  digital write pin P9 to 1
  wait (μs) 10
  digital write pin P9 to 0
  set Distance to pulse in (μs) pin P9 pulsed high
  set Distance to Distance integer ÷ 58
  show number Distance
  pause (ms) 100
  clear screen
  pause (ms) 100
```

Above: The "function *getDistance*" block in "Sonar Test 4", which displays the distance obtained from the sonar sensor.

(continued)

```

forever
  if DrivingIsEnabled then
    for index from 0 to 2
      do
        unplot x 3 - index y 0
    pause (ms) 200
    if SpeedNew < 0 then
      for index from 0 to 2
        do
          plot x 3 - index y 0
    else
      plot x 2 y 0
      +
      for indexRow from 0 to 2
        do
          for indexColumn from 0 to 4
            do
              if absolute of SpeedNew > indexColumn + 5 x indexRow integer x 10 then
                plot x 4 - indexColumn y 4 - indexRow
              else
                unplot x 4 - indexColumn y 4 - indexRow
            +
          +
        +
      +
    else
      call getDistance
      +
  +

```

Above: "forever" block of "Sonar Test 4" modified for mode switching using "if DrivingIsEnabled ..." block. The new "call getDistance" block at the end. All other blocks stay the same.

- The "Sonar Test 4" code: https://makecode.microbit.org/_HC4aaM53DDkc
- Works with "Rover RC 5": https://makecode.microbit.org/_DMzFozEx1hq5
-

8.10 Sonar Calibration Test

Now that you have a working sonar for your rover, we want to test how accurate it is. We can do that by conducting a test much like we did with propulsion so that we can gather some data and analyze it.



Data Collection Lab: Sonar Calibration Test

Time Required: 1 hour

Materials:

- Computer Lab (any computers, tablets okay, no cell phones)
- Assembled STEM SEALs LAND robot with sonar
- four AA batteries
- USB-to-Micro-B-USB cable
- browser on the computer, Fire Fox preferred (some things don't work on Chrome!)
- an open, flat floor space (classroom, hallway)
- masking tape
- ruler (metric)
- obstacle

Prerequisites:

- Students are familiar with programming the microbit and its LED display.
- Students have assembled the rover and sonar attachment.
- Rover micro:bit with loaded the “**Sonar Test 4**” code.
- Sonar head on student rover moves, and at rest is facing straight forward.
- If the sonar head is not facing straight, adjust “set HeadStraightAt” parameter as determined earlier in programing the rover.

Directions:

1. Place a ruler in front of the rover, pointing away from the rover, place the 0 cm mark in the plane of the sonarbit's PCB (the board the sensor is mounted on). Ensure that the ruler does not shift for the following measurements (use masking tape).
2. Place the obstacle in front of the rover, start at the 4 cm mark of the ruler.
3. Read the number displayed by the microbit and enter on the table below.
4. Move the obstacle to the 5 cm mark and enter the number displayed by the microbit on the table.
5. Repeat for every cm moving the obstacle away from the rover and enter the microbit's number in the appropriate place in the data table
6. Stop at a distance of 30 cm.
7. Repeat all steps a second time and record in the last column of the table.

Continued next page.



Data Collection Lab: Sonar Calibration Test

Sonar Calibration - Data Table

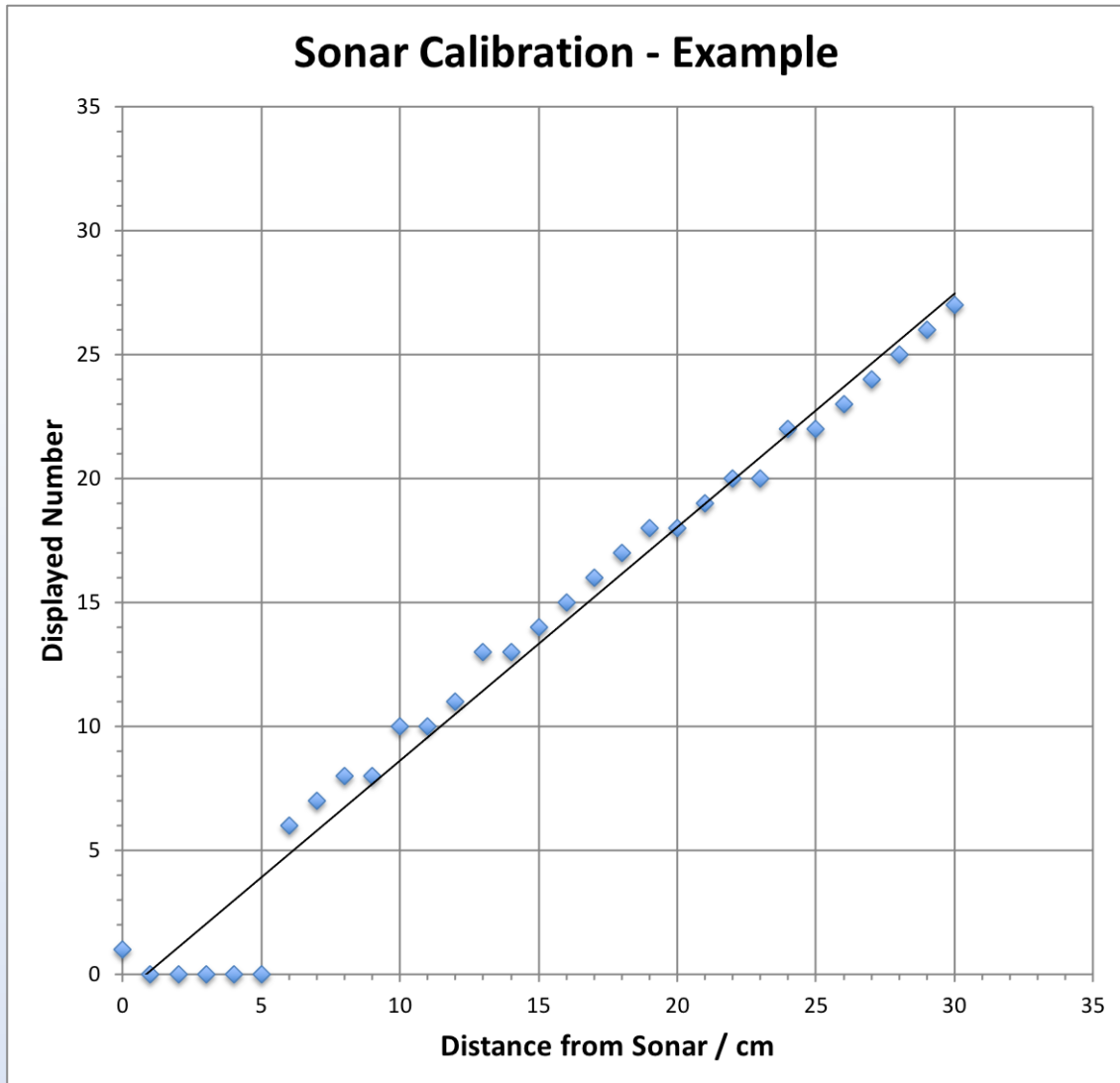
Distance / cm	Display (cm) Trial 1	Display (cm) Trial 2
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		

Continued next page.



Data Collection Lab: Sonar Calibration Test

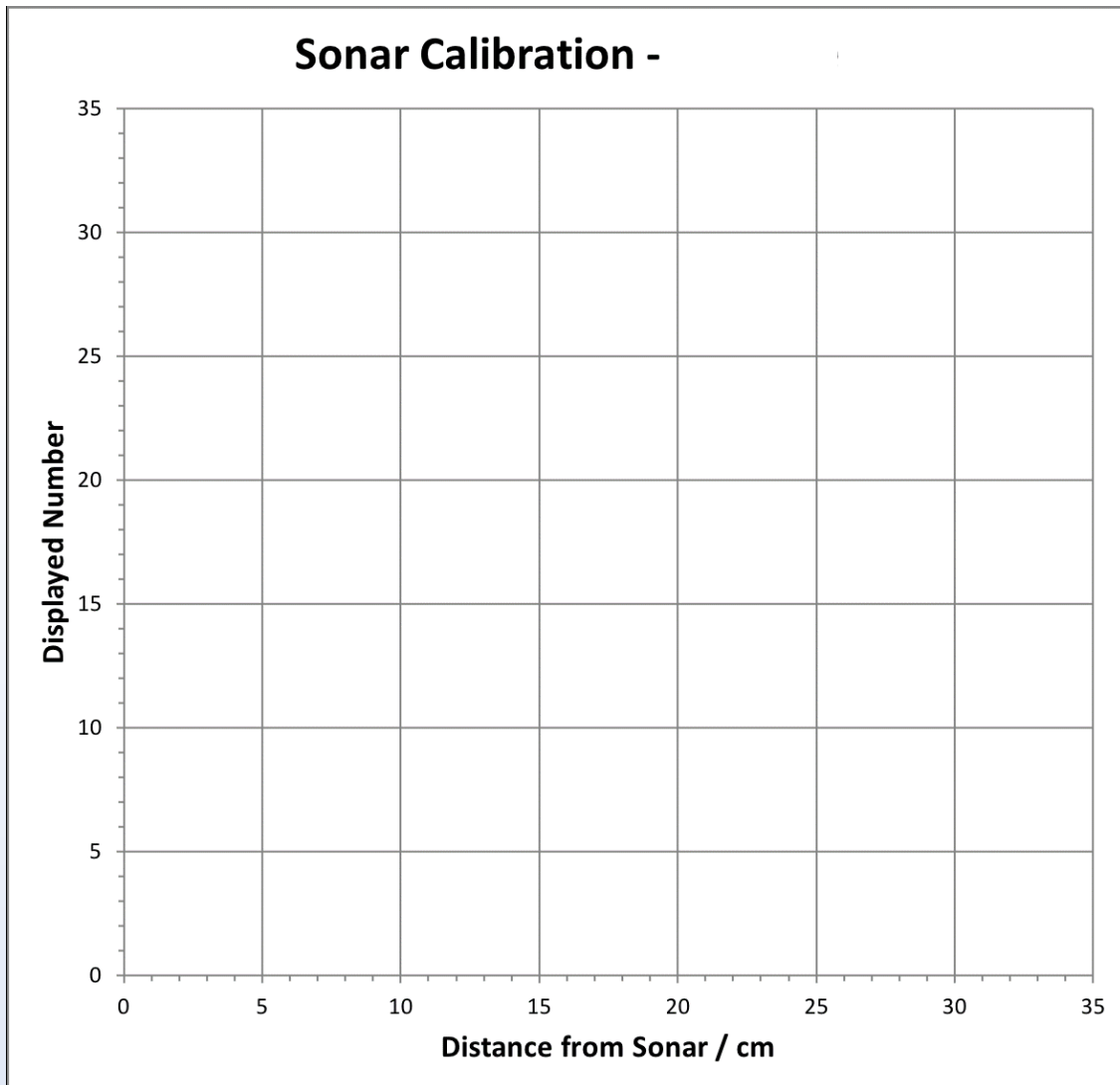
Now that you have collected your data, graph the ordered pairs on the coordinate graph on the next page. Below is a sample graph of rover 44.



Continued next page.



Data Collection Lab: Sonar Calibration Test



The results:

- Are your data close to a straight line?
- Is the slope of the line 1.00? (Why should it be exactly one?)
- Optional activity:
 - make a new table like the one before but place the rover at the whole inch marks.
 - enter your measurements in the table below,
 - and make a graph.
 - Is the slope of these new data different?
 - Why?

Module 9: Navigation - Autonomous Driving

9.1 Writing an Autonomous Program (Navigation Test 1)

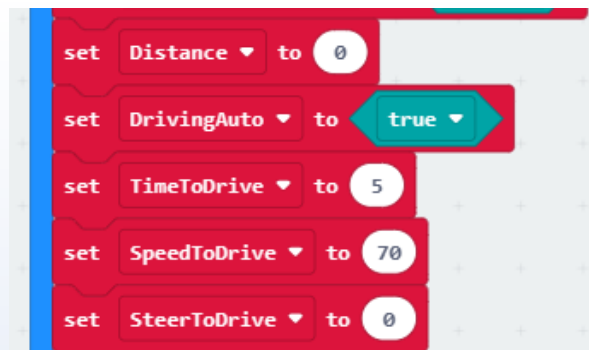
How "intelligent" is your rover? Do you think it can drive on its own?

Let's write a program to get the rover to automatically drive and stop when it encounters an obstacle without hitting the obstacle. We will use the sonar to measure the distance, slow down and have the rover decide when to stop.



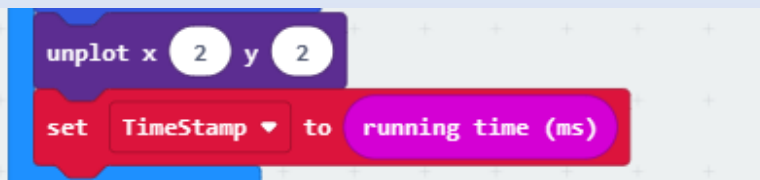
Time to Code: Navigation Test 1

1. Begin with the "Sonar Test 4" code, load and rename it into "Navigation Test 1".
2. Modify the "on start" block:
 - Make a new variable "DrivingAuto" and place a "set DrivingAuto to true" block in to the "on start" block after the "set Distance to 0" block.
 - Make a new variable "TimeToDrive" and place a "set TimeToDrive to 5" block after the "set DrivingAuto ..." block.
 - Make a new variable "SpeedToDrive" and place a "set SpeedToDrive to 70" block thereafter.
 - Make a new variable "SteerToDrive" and place a "set SteerToDrive to 0" block next.



With all this we tell the rover: initially, if autonomous driving is enabled, we want it to drive with Speed = 70 (fast) and Steer = 0 (straight) for TimeToDrive = 5 seconds, but then when the sensor "feels" it gets close to an obstacle, the rover should slow down and stop before hitting the obstacle.

- Make a new variable "TimeStamp" and place a "set TimeStamp to running time (ms)" (from the "Input" toolbox) at the very end of the "on start" block after the "unplot x 2 y 4" block.



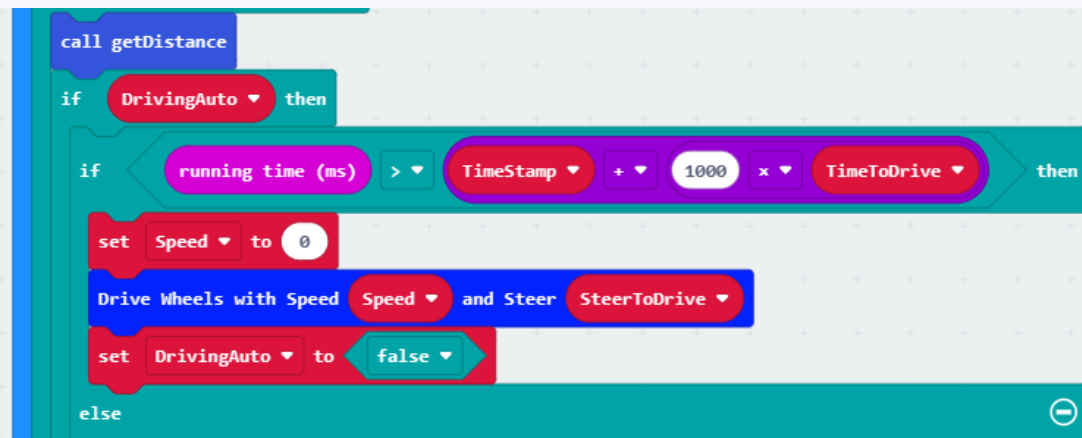
Continued next page.



Time to Code: Navigation Test 1 (continued)

3. Modify the “forever” block:

- At the end of the “forever” block, place an “if DrivingAuto then ...” block after the “call getDistance” block,
- Inside this new “if DrivingAuto then” block, place a new “if then else” block.
 - Place a “0 < 0” comparison block over the “true” in this block.
 - Place a “running time (ms) from the Input toolbox (more) over the first zero.
 - Change the “<” (less than) sign to a “>” (greater than) sign.
 - Create the calculation [TimeStamp + (1000 x TimeToDrive)] and place it over the second zero in the comparison block.
 - Place a “set Speed to 0” block inside the empty “if then” slot.
 - Place a “Drive wheels with Speed Speed and Steer Steer to Drive” block after that.
 - Then place a “set DrivingAuto to false” after that.



- In the empty “else...” slot of the “if running time (ms)...” block you just created place another “if Distance < 7 then” block.
 - Inside this “if Distance ...” block, place a “set Speed to 0” block.
 - After this, a “Drive Wheels with Speed Speed and Steer SteerToDrive” block from the “StemSeals” toolbox.
 - Next place a “set DrivingAuto to false” block.



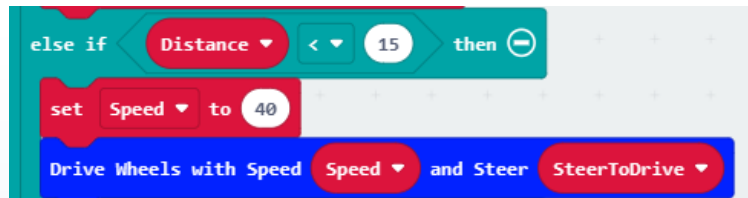
Continued next page.



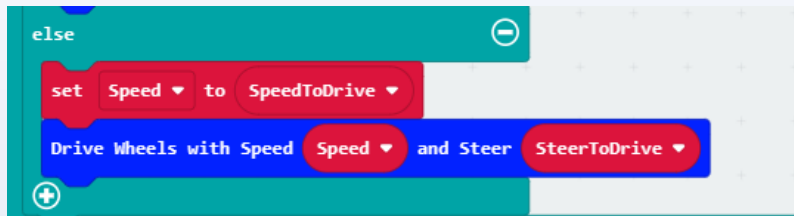
Time to Code: Navigation Test 1 (continued)

4. Continue modifying the “forever” block:

- Click on the twice on the "+" of the "if Distance < 7" block to make an "else if Distance < 15" block.
 - Place a "set Speed to 40" block inside this "else if ..." slot.
 - Copy the "Drive Wheels ..." block from above and place it next.



- In the "else ..." part of the "if Distance ..." block,
 - Place a "set Speed to SpeedToDrive" block.
 - Copy the "Drive Wheels ..." block from above and place it here.

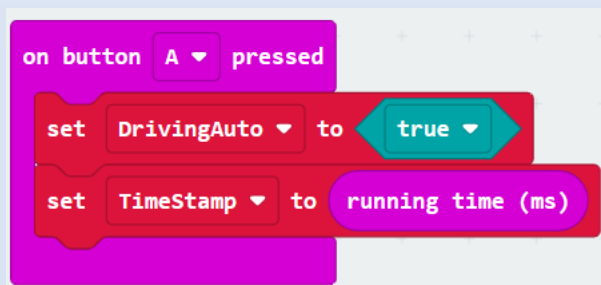


5. Modify the “function getDistance” block:

- Delete the first of the "pause 100 ms" block at the end of the “function” block.
- Insert a "plot x 2 y 4" block from "Led" at the very end of the "function ..." block.

6. Modify the “on button A pressed” block:

- Delete the "change SpeedNew ..." block.
- Replace it with a "set DrivingAuto to true" block.
- Copy the "set TimeStamp ..." block from the end of the "on start" block and place it at the end of the "on button A pressed" block.





Try This: Navigation Test 1

1. Download, upload the “Navigation Test 1” to the rover micro:bit.
2. Do not use the remote control for this test.
3. Place the rover on the ground and turn it on,
 - Does the bed level and the head sweep from left to right?
 - After that, does the rover drive off?
4. Place the rover 50 cm from an obstacle, and push the A button:
 - Does it drive?
 - Does it stop before the obstacle?
5. Try different obstacle types and judge how well the rover stops.
6. If you think your rover is not stopping well before the obstacle, you can adjust certain numbers.
 - You can adjust the numbers (7 and 15) for the distances in the "if Distance ..." blocks of the "forever" block.

```
if Distance < 7 then
  set Speed to 0
  Drive Wheels with Speed Speed and Steer SteerToDrive
  set DrivingAuto to false
else if Distance < 15 then
  set Speed to 40
  Drive Wheels with Speed Speed and Steer SteerToDrive
```

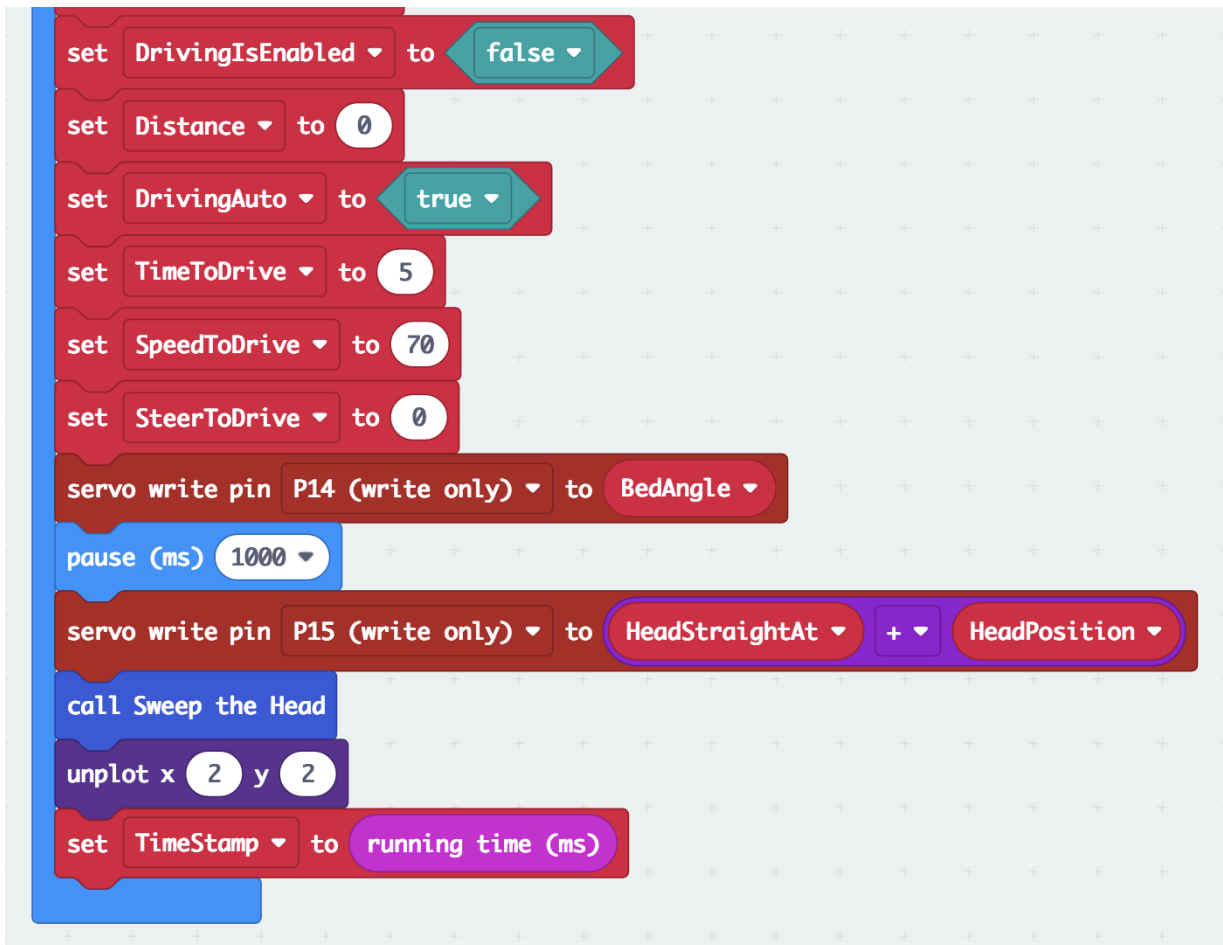
- You can also adjust the speed (40 and 70). On some surfaces, the rover slides a little further when traveling too fast and therefore might collide with the obstacle.

In the “on start” block:

In the “forever” block:

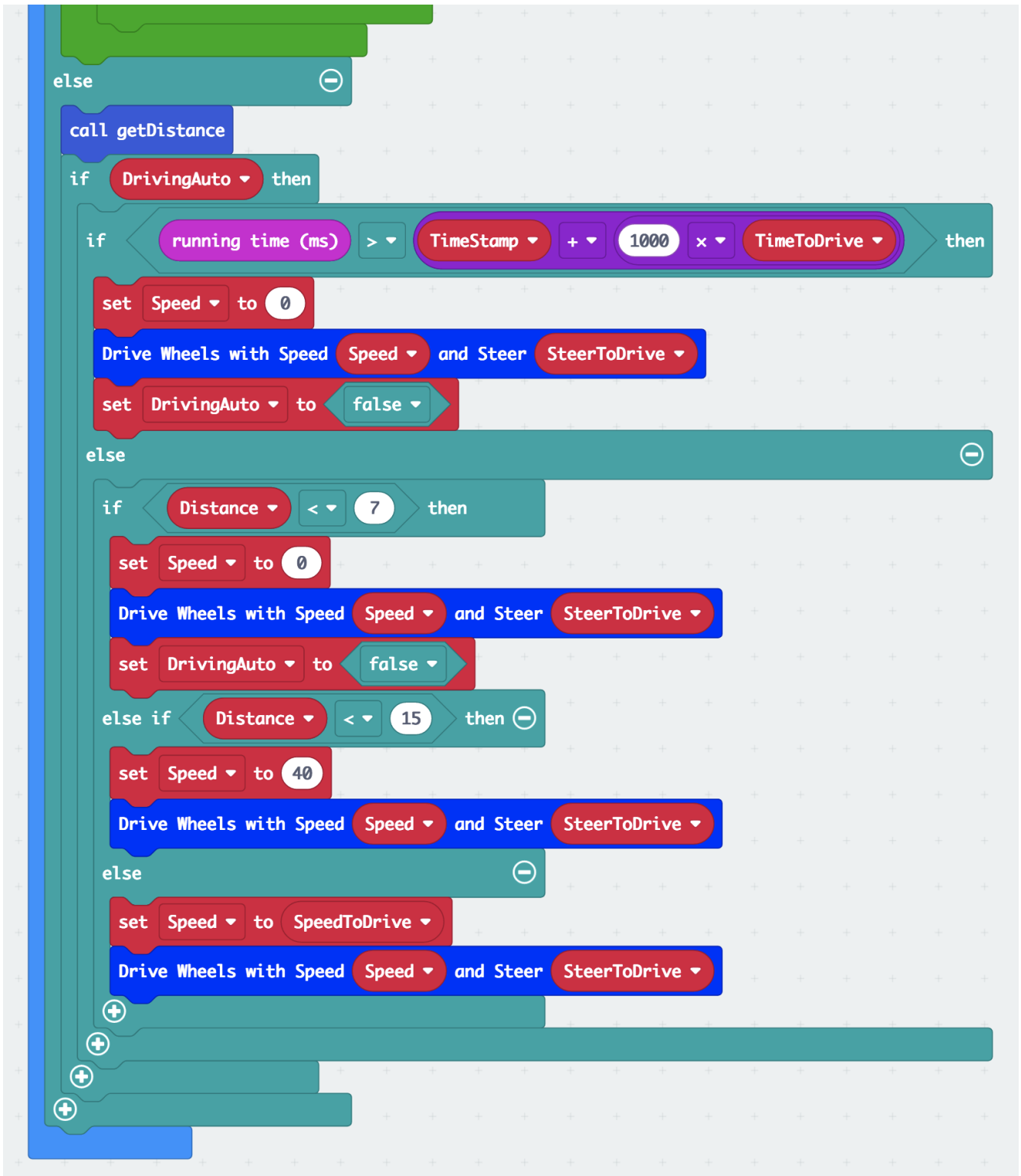
Variables such as these in the code are the ones to take note of as these will be the ones you may need to change for your rover to function at its best. When in competition rover events are judged on speed and accuracy. Completing a task should be done in the quickest speed but also without hitting obstacles (accuracy).

- Your code for “Navigation Test 1” should look like this:



The image shows a sequence of Scratch code blocks for an "on start" event. The blocks are: a red "set DrivingIsEnabled" block to "false"; a red "set Distance" block to "0"; a red "set DrivingAuto" block to "true"; a red "set TimeToDrive" block to "5"; a red "set SpeedToDrive" block to "70"; a red "set SteerToDrive" block to "0"; a red "servo write pin P14 (write only)" block to "BedAngle"; a blue "pause (ms)" block set to "1000"; a red "servo write pin P15 (write only)" block to "HeadStraightAt" plus "HeadPosition"; a blue "call Sweep the Head" block; a blue "unplot x" block with "2" and "y" block with "2"; and a red "set TimeStamp" block to "running time (ms)".

Above: The end of the "on start" block of the "Navigation Test 1" code with a new set of variables to enable autonomous driving



Above: The end of the "forever" block in "Navigation Test 1" enabling autonomous driving.

```

function getDistance
  set pull pin P9 to none
  digital write pin P9 to 0
  wait (µs) 2
  digital write pin P9 to 1
  wait (µs) 10
  digital write pin P9 to 0
  set Distance to pulse in (µs) pin P9 pulsed high
  set Distance to Distance integer ÷ 58
  clear screen
  pause (ms) 100
  plot x 2 y 4

```

Above: Modified "function getDistance" block to make it fast enough for the rover to react before it hits the obstacle.

```

on button A pressed
  set DrivingAuto to true
  set TimeStamp to running time (ms)

```

Above: New "on button A pressed" block in "Navigation Test 1" allowing another test approaching obstacles.

- "Navigation Test 1" code: https://makecode.microbit.org/_d7aXuvF4fMjP

9.2 Navigation Test 2

You have now completed a very simple autonomous navigation task. When you press button A on the rover, it carries out a set of instructions without any further input from you. Throughout the navigation activities in Module 9, you will continue to add on to this simple program to make it more complex so that it can complete more autonomous maneuvers.

Let's start by modifying the code to make the rover not only stop at an obstacle but also to turn around.



Time to Code: Navigation Test 2

1. Start with "Navigation Test 1" and rename it "Navigation Test 2".
2. Modify the "on start" block:
 - Right below the "set TimeToDrive ..." block, place a "set list to array of 0 1" block from the "Advanced" "Arrays" toolbox.
 - Change the "list" to "TimeToDrive".
 - Change the zero to 5 (seconds).
 - Change the number 1 to 2 (seconds).
 - Now delete the "set TimeToDrive to 5" block above.
 - Make a new variable "TurningAuto" and place a "set TurningAuto to false" block before the "set TimeToDrive to array ..." block.



This allows the rover 5 sec to drive straight and find the obstacle, and then 2 sec to turn around. Instead of using two variables to do that; we can use the variable name "TimeToDrive", but declare it as an array of numbers, first 5 sec, then 2 sec.

3. Modify the "forever" block:
 - Delete the variable "TimeToDrive" in the "if running time (ms) > TimeStamp + 1000 * TimeToDrive" block,
 - Place a "list get value at 0" block from ("Advanced") "Arrays" at the zero where the "TimeToDrive" variable was.
 - Change the "list" to "TimeToDrive".



Continued next page.



Time to Code: Navigation Test 2 (continued)

4. Continue modifying the "forever" block:

- Inside the "if running time > TimeStamp ..." block add these three blocks.
 - Add a "set TimeStamp to running time (ms)" block to the end.
 - After that, place a "set TurningAuto to true" block.
 - Then add a "pause 10 ms" block.

The screenshot shows a Scratch script editor. An 'if' block is selected, with the condition 'running time (ms) > TimeStamp + 1000 * TimeToDrive get value at 0'. Below the 'if' block, a 'set Speed to 0' block is present. A 'Drive Wheels with Speed Speed and Steer SteerToDrive' block is also present. A red arrow points to the 'if' block, indicating where the new blocks should be added. The new blocks to be added are: 'set DrivingAuto to false', 'set TimeStamp to running time (ms)', 'set TurningAuto to true', and 'pause (ms) 10'.

- Copy all three of these new blocks, "set TimeStamp ...", "set TurningAuto ...", and "pause ...", paste and place them **inside** the "if Distance < 7 then" block after the "set DrivingAuto to false" block,

The screenshot shows a Scratch script editor. An 'if' block is selected, with the condition 'Distance < 7'. Below the 'if' block, the following blocks are present: 'set Speed to 0', 'Drive Wheels with Speed Speed and Steer SteerToDrive', 'set DrivingAuto to false', 'set TimeStamp to running time (ms)', 'set TurningAuto to true', and 'pause (ms) 10'.

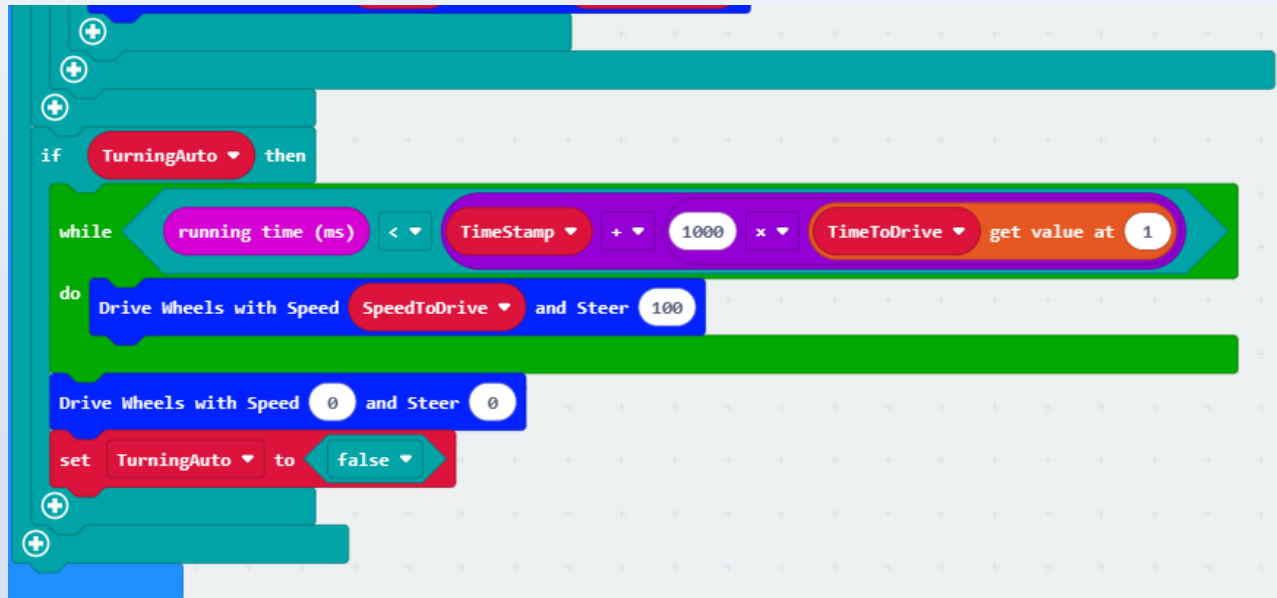
Continued next page.



Time to Code: Navigation Test 2 (continued)

5. Continue modifying the “forever” block:

- Add a new "if ..." block (be careful where) at the end of the "forever" block, but inside the "else ..." part of the "if DrivingIsEnabled" block (look at the image below to be sure where this "if ..." needs to be!).
 - Place the "TurningAuto" variable between the "if" and the "then" in this block.
 - Get a "while ..." block from "Loops" and place it inside the "if TurningAuto then ..." block.
 - Copy the "running time (ms) < Timestamp + 1000" block from above, paste and place it in the space after the "while".
 - o Change the zero at the end to the number 1.
 - Get a new "Drive Wheels" block from "StemSeals" and place it inside the "while ..." block.
 - o Place the variable "SpeedToDrive" over the first zero.
 - o Place the number 100 (to let it spin!) over the second zero.
- Place a new "Drive Wheels with Speed 0 and Steer 0" block after the "while ..." block.
- Place a new "set TurningAuto to false" block thereafter.

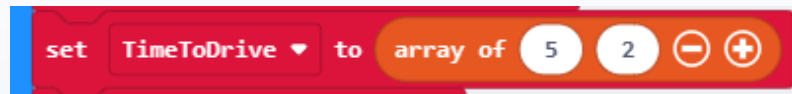




Try This: Navigation Test 2

1. Download, upload and test “Navigation Test 2” to the rover micro:bit.
2. First, place the rover in an open space, and turn it on:
 - Does it drive?
 - Does it stop? After 5 sec?
 - Does it turn? How far?
 - The turn angle depends on the floor material, you need to adjust the number in the TimeToDrive array in the "on start" block to get your rover to turn as far as you need. Ideally you want it to turn 180°.

Array in the “on start block:



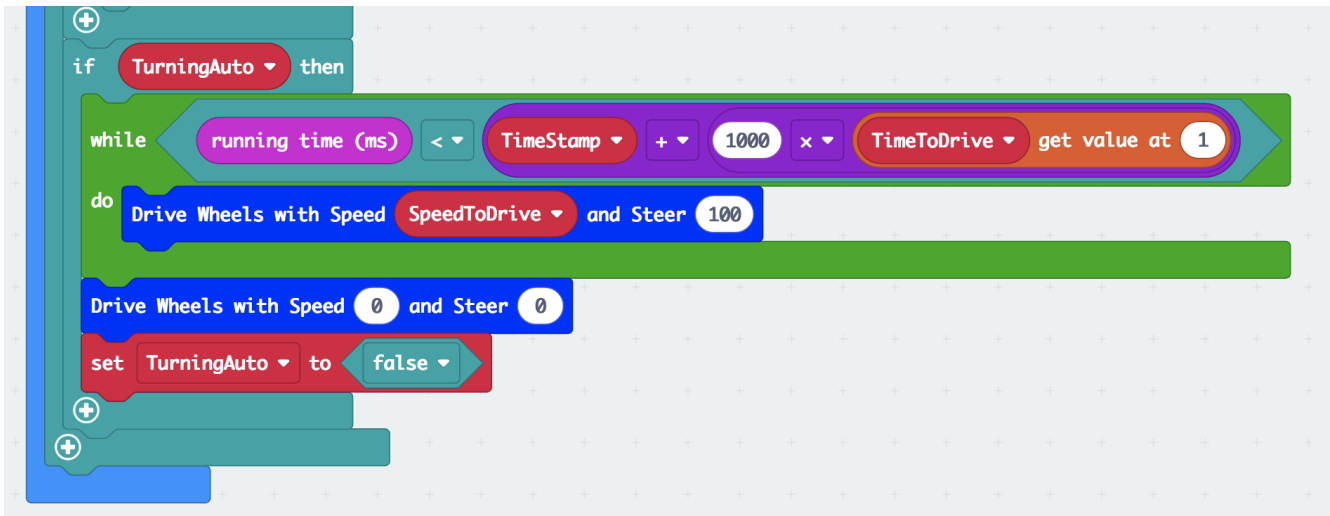
- Remember the 5 (seconds) is the initial driving straight time. The 2 (seconds) is how long the rover drives while turning. If the rover turns past 180° then decrease the 2 (seconds). If the rover turns short of 180° then you would increase the 2 (seconds).
 - What does the pause do?
3. Change the numbers to the following:
 - Change the 2 in "set TimeToDrive to array" to 1.2
 - Test, and improve!
 - Example: On a specific wooden floor, the rover performed best with "set TimeToDrive to array of 5 0.85" in "on start".
 4. Keep trying different numbers to perfect the performance of your rover.
 - The competition will take place in the gymnasium on the wooden basketball court.

The code for “Navigation Test 2” should look like:

The image shows a sequence of Scratch code blocks for the 'on start' event of 'Navigation Test 2'. The blocks are as follows:

- set DrivingIsEnabled to false
- set Distance to 0
- set DrivingAuto to true
- set TurningAuto to false
- set TimeToDrive to array of 5 0.85
- set SpeedToDrive to 70
- set SteerToDrive to 0
- servo write pin P14 (write only) to BedAngle
- pause (ms) 1000
- servo write pin P15 (write only) to HeadStraightAt + HeadPosition
- call Sweep the Head
- unplot x 2 y 2
- set TimeStamp to running time (ms)

Above: the new array in "on start" of "Navigation Test 2"



Above: the new end of the "forever" block in "Navigation Test 2". Here is where the turn (the spin!) happens after finding the obstacle (or just after running straight for 5 seconds).

The code: "Navigation Test 2": https://makecode.microbit.org/_frUDhjMvFecj

9.3 Navigation Test 3

As you can see, doing different types of tasks, at different times, can get complex, and maybe even confusing. Just imagine how the code might look if we wanted the rover go straight, make a 90-degree turn, drive to the left, make another turn and come back!

Did you notice how easy it was to deal with the two different times for the "TimeToDrive" in the array?

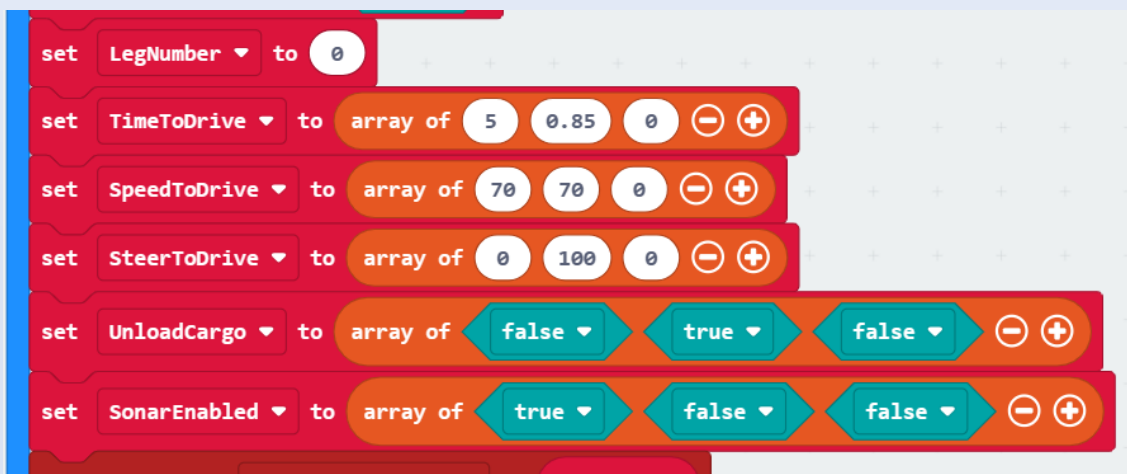
We can make the whole situation clearer if we use arrays for all the task we need: how long? how fast? how sharp the turn? Then we would simply treat each task the same way. The only thing we should watch out for is which element of the arrays we must pick at which time. But that's a simple counting action: The first leg or task of the journey uses the first element in each array, the second the second, and so on.

Let's try!



Time to Code: Navigation Test 3

1. Start with "Navigation Test 2" and rename it "Navigation Test 3".
2. Modify the "on start" block:
 - Click on the variable in the "set TurningAuto to false" block and rename this variable to "LegNumber" by scrolling all the way to the bottom of the variable list where you will find the option to rename. This will change the "TurningAuto" variable to the new name "LegNumber" in all blocks that it is located.
 - Delete the "false" there and leave the number zero, (an exclamation mark appears temporarily because we used it before as a Boolean variable - we will fix this later.)
 - Click on the "+" sign at the end of the "set TimeToDrive to array of 5 2." block; this creates another space for the return leg of the course.
 - Obtain another "set list to ..." block from "Arrays" and place it right underneath.
 - Change the "list" to "SpeedToDrive"
 - Click on the "+" at the end to make it a 3-element array.
 - Enter 70 and 70 for the first two elements,
 - Obtain another "set list to ..." block.
 - Change the "list" to "SteerToDrive".
 - Click on the "+" at the end to make it a 3-element array.
 - Enter 0 and 100 for the first two elements.
 - Obtain another "set list to ..." block.
 - Make a new variable, name it "UnloadCargo", change the "list" to "UnloadCargo".
 - Click on the "+" at the end to make it a 3-element array.
 - From the "Logic" toolbox, get "true"s and "false"s to set the "UnloadCargo" array to false, true, and then false again.
 - Obtain another "set list to ..." block.
 - Make a new variable, name it "SonarEnabled", change the "list" to "SonarEnabled".
 - Click on the "+" at the end to make it a 3-element array.
 - From the "Logic" toolbox, get "true"s and "false"s to set the "SonarEnabled" array to true, false, and false again.
 - Delete the "set SpeedToDrive to 70" block.
 - Delete the "set SteerToDrive to 0" block.



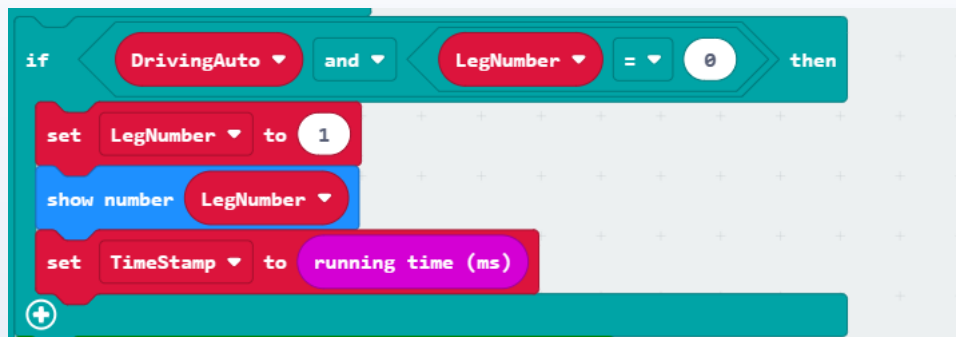
Continued next page.



Time to Code: Navigation Test 3 (continued)

7. Modify the “forever” block:

- Drag all blocks from the "else ..." part of the "if DrivingIsEnabled then ..." block starting with the “call getDistance” block and place them temporarily in the workspace. You will use these again.
- Insert a new "if then" block from "Logic" into the now empty "else ..." part.
 - Get a "... and ..." Boolean block from "Logic" and place it into the "if ..." block.
 - Place the "DrivingAuto" variable into the first empty space in the "... and ..." block.
 - Get a new "0 = 0" comparison block and place it into the second empty space,
 - Place the "LegNumber" variable over the first zero.
 - Place a new "set LegNumber to 1" block inside the newly created "if DrivingAuto and LegNumber = 0 then" block.
 - Place a “shownumber LegNumber” after that.
 - Make a new "set TimeStamp to running time (ms)" block and place it next into the "if DrivingAuto ..." block.



- Get a new "for index from 0 to 4" block and place it after the just completed "if DrivingAuto..." block.
 - Get a new "0 - 0" block from "Math" and place it over the number 4 here.
 - Replace the first zero with the variable "LegNumber".
 - Replace the second zero with the number 1.
 - Starting with the "if running time (ms) > ..." block from the set aside code in the workspace, drag it and the attached blocks that fall below it into the "for index ..." block. There will still be some block left “ghosted” in the workspace.
 - Change the greater than sign (>) in “if running time...” block to a less than sign (<).
 - Copy the “LegNumber - 1” from the “for index...” block and place over the zero in the “TimeToDrive get value at 0” block.
 - Starting with the “set Speed to 0” block, drag it and the five other blocks inside the “if running time...” block to the workspace to use later.



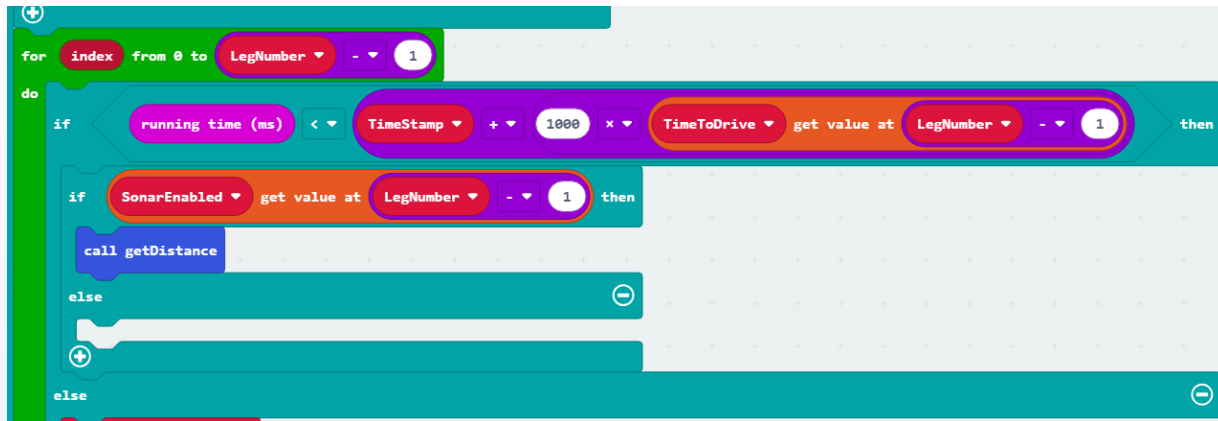
Continued next page.



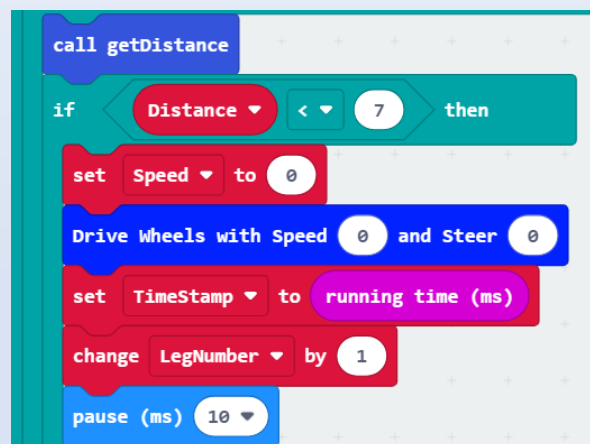
Time to Code: Navigation Test 3 (continued)

3. Continue to modify the “forever” block:

- Get a new “if...then...else” block and place it inside the “if running time...” Block.
 - Copy the “TimeToDrive get value at LegNumber – 1” from the “if...” Block above and place it over the “true” in the newly created “if...then...else” block.
 - Change "TimeToDrive" in here to the "SonarEnabled" variable.
 - Get only the “call getDistance” block you put in the workspace and place it inside the “if SonarEnabled...” block.



- Grab and move the “if Distance < 7 then” block and all the attached blocks and place them under the “call getDistance” block.
 - Change the “Drive Wheels with Speed Speed and Steer SteerToDrive” to “Drive Wheels with Speed 0 and Steer 0”.
 - Delete the “DrivingAuto to false” block.
 - Delete the “set LegNumber to true” block and replace with a “change LegNumber by 1” block.



Continued next page.



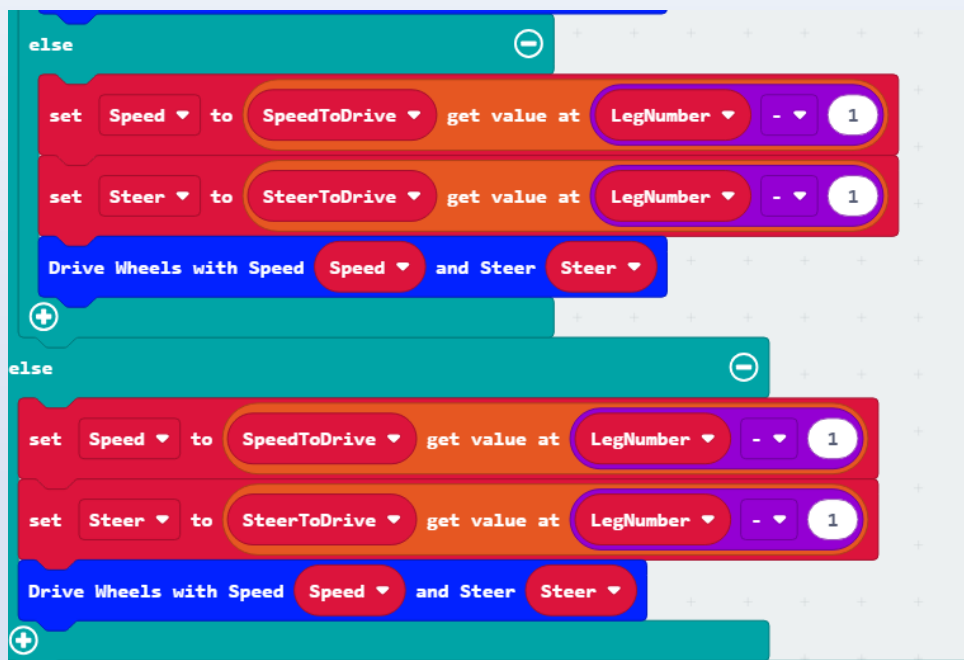
Time to Code: Navigation Test 3 (continued)

4. Continue Modifying the “forever” block:

- In the “else if Distance < 15” block”, create a “set Steer to SteerToDrive get value at LegNumber -1” and place it between the “set speed...” and Drive Wheels...” blocks.
 - Change the “Steer to Drive” to Steer Steer” in the “Drive Wheels...” block.



- In the “else” part of the “if Distance < 7...” block (which is below the “else if distance < 15”), delete the “set Speed to SpeedToDrive” block.
 - Copy the “set Steer to Steer to Drive get value at LegNumber -1” block from above and paste it two times before the Drive Wheels...” block.
 - In the first one change the “Steer” to “Speed” and the “SteerToDrive” to SpeedToDrive”.
 - In the “Drive Wheels...” block change the “SteerToDrive” to “Steer”.
 - Copy **ALL** three of these blocks and paste them into the else part that is empty in the “if SonarEnabled...” block.

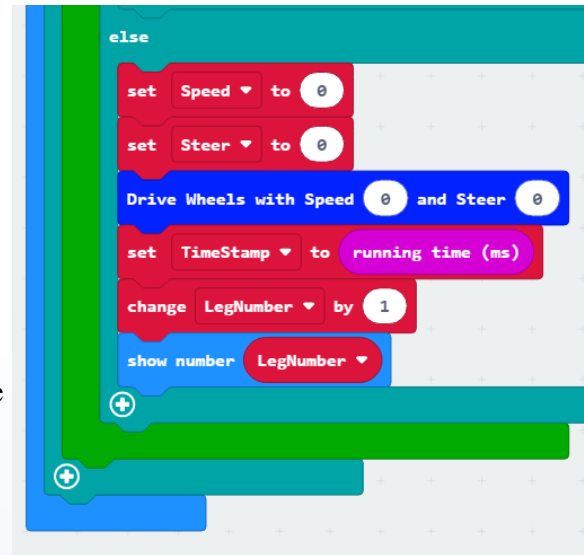


Continued next page.

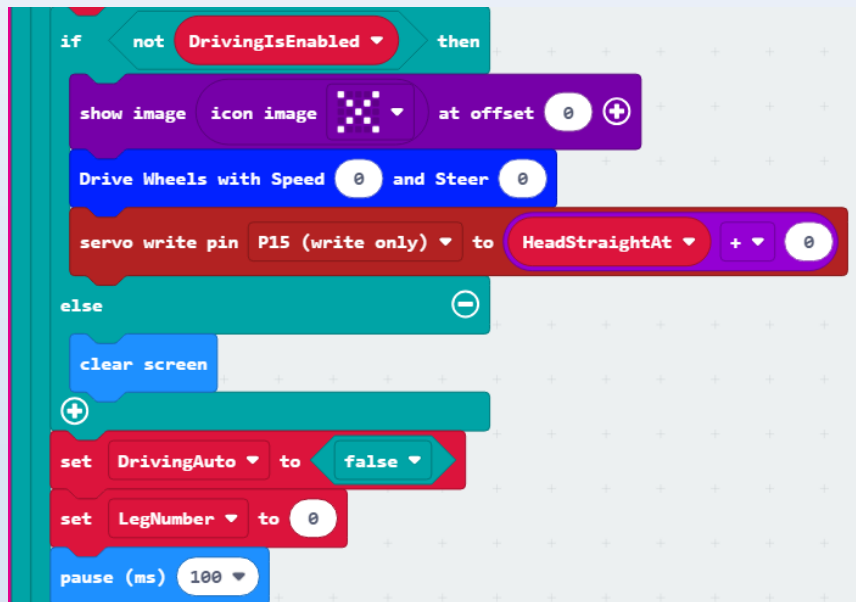


Time to Code: Navigation Test 3 (continued)

5. Finish modifying the “forever” block:
- Finally, to finish in the “forever” block, in the else part of the “if running time...” block (it should be the empty else at the end of the “forever” block), place these blocks. (see picture)
 - You may find some in the workspace that were pulled aside, some you can copy from other places, and some you must create.
 - There will be a left-over “ghosted” block in the workspace you may now delete.



6. Modify the “on radio received...” block:
- At the top of the "on radio received ..." block, inside the "if not DrivingIsEnabled then ..." after the "Drive Wheels with Speed 0 and Steer 0" block, insert (copy from below) a "servo write pin P15 ..." block.
 - Delete the "HeadPosition" variable such that it reads "servo write pin P15 to HeadStraightAt + 0".
 - After this "if ... then ... else" block, before the "pause (ms) 100" block, insert a "set DrivingAuto to false".
 - Insert a "set LegNumber to 0" block after that.

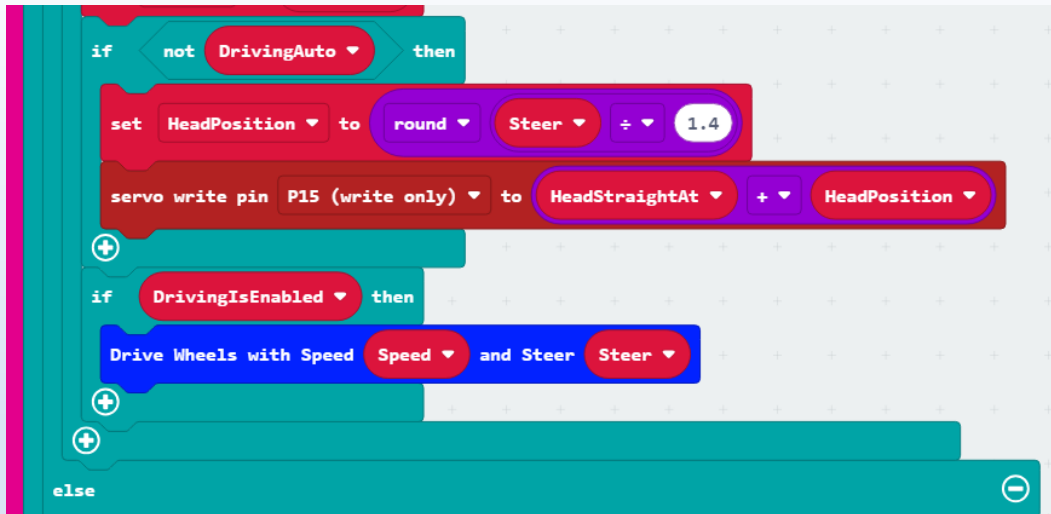


Continued next page.



Time to Code: Navigation Test 3 (continued)

- Continue to modify the “on radio received” block:
 - Create a new “if not DrivingAuto then...” and place it before the “if DrivingIsEnabled then...” block at the end of the “on radio received...” block.
 - Move both blocks, “set HedaPosition to round...” and “servo write pin P15...”, into the newly created “if not DrivingAuto...” block.



Try This: Navigation Test 3

- Download, upload the “Navigation Test 3” code to the rover micro:bit, and test your code:
 - Turn the rover on and set it down in an open space:
 - Does it drive straight for 5 sec?
 - Does it turn around after?
 - Does it stop moving?
 - Turn the RC on (Rover RC 5):
 - Does the rover remain stopped?
 - Touch the logo on the RC:
 - Can you now remote control the rover as you did before?
 - Stop the rover with the RC (easiest is: press A!)
 - Place the rover in front of an obstacle:
 - Press the rover's A button:
 - Does it move?
 - Does it stop before the obstacle?
 - And turn around?
 - Check if you still can unload with the RC's A button.
 - Check if the B button levels the bed again.
 - You should be able to switch with the RC's logo button and the rover's A button between autonomous and remote-controlled driving modes.

➤ Your code for “Navigation Test 3” should look like this:

```
set Speed to 0
set SpeedNew to 0
set DrivingIsEnabled to false
set Distance to 0
set DrivingAuto to true
set LegNumber to 0
set TimeToDrive to array of 5 0.85 0 - +
set SpeedToDrive to array of 70 70 0 - +
set SteerToDrive to array of 0 100 0 - +
set UnloadCargo to array of false true false - +
set SonarEnabled to array of true false false - +
servo write pin P14 (write only) to BedAngle
pause (ms) 1000
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
call Sweep the Head
unplot x 2 y 2
set TimeStamp to running time (ms)
```

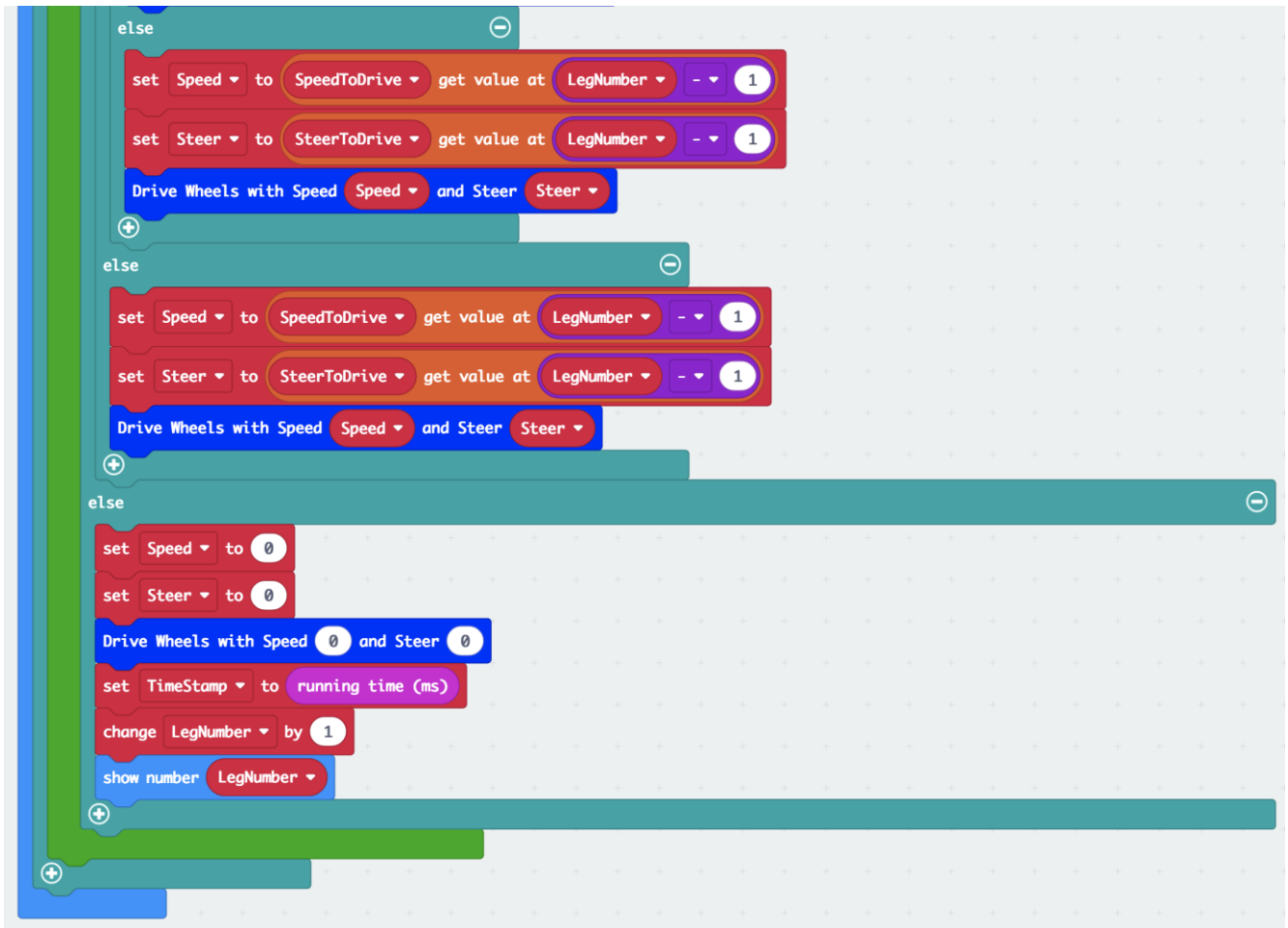
Above: Second half of the "on start" block of "Navigation Test 3" with five arrays of variables to control up to three legs of a course.

Continued next page.

```
else
  if << DrivingAuto >> and << LegNumber >> = << 0 >> then
    set LegNumber to 1
    show number LegNumber
    set TimeStamp to running time (ms)
  for index from 0 to LegNumber - 1
  do
    if << running time (ms) >> << TimeStamp >> + << 1000 >> x << TimeToDrive >> get value at LegNumber - 1 then
      if << SonarEnabled >> get value at LegNumber - 1 then
        call getDistance
        if << Distance >> << 7 >> then
          set Speed to 0
          Drive Wheels with Speed 0 and Steer 0
          set TimeStamp to running time (ms)
          change LegNumber by 1
          show number LegNumber
          pause (ms) 10
        else if << Distance >> << 15 >> then
          set Speed to 40
          set Steer to SteerToDrive get value at LegNumber - 1
          Drive Wheels with Speed Speed and Steer Steer
        else

```

Continued next page.



Above: New "else (if DrivingIsEnabled ...) ..." block in the "forever" block of "Navigation Test 3". The big "for index ..." block handles the course legs in autonomous navigation mode.

Continued next page.

```

on radio received receivedString
  if < parse to number substring of receivedString from 0 of length 2 = RoverNumber > then
    if < substring of receivedString from 2 of length 1 = "X" > then
      set DrivingIsEnabled to not DrivingIsEnabled
      if < not DrivingIsEnabled > then
        show image icon image [dotted] at offset 0
        Drive Wheels with Speed 0 and Steer 0
        servo write pin P15 (write only) to HeadStraightAt + 0
      else
        clear screen
        set DrivingAuto to false
        set LegNumber to 0
        pause (ms) 100
    else if < substring of receivedString from 2 of length 1 = "B" > then
  
```

Above: Needed modifications at the beginning of the "on radio received ..." block of "Navigation Test 3" to allow RC operation as well as autonomous driving.

Continued next page.

```
else if <substring of receivedString from 2 of length 1 = "S"> then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if <SpeedNew >= -100 and SpeedNew <= 100> then
    set Speed to SpeedNew
    if <not DrivingAuto> then
      set HeadPosition to round Steer ÷ 1.4
      servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    if DrivingIsEnabled then
      Drive Wheels with Speed Speed and Steer Steer
  else
```

Above: Needed modifications at the end of the "on radio received ..." block of "Navigation Test 3" to allow RC operation as well as autonomous driving.

The code: "Navigation Test 3": https://makecode.microbit.org/_cqAhxRbsdD8T

9.4 Navigation Test 4

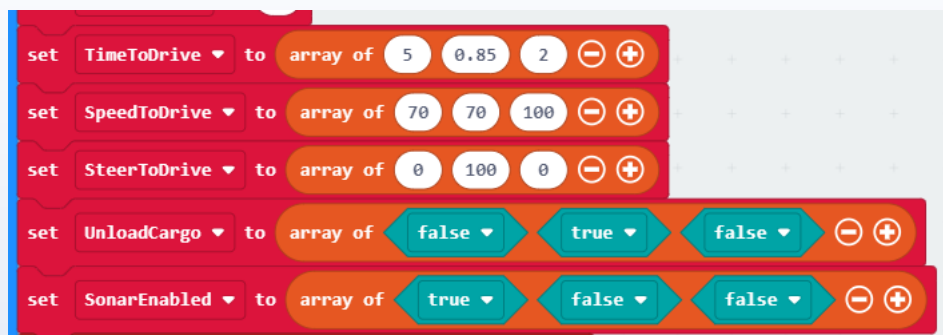
Now with that task completed, we also want to be able to automatically unload the cargo after a certain designated leg (without having to push button A). This is the reason we already added an “UnloadCargo” array in the start block. We will also set the parameters for the 3rd leg so that after unloading the cargo the rover will move forward.

Let’s now set up our code to unload the cargo automatically. This will only take a few minor adjustments to the code.



Time to Code: Navigation Test 4

1. Load the "Navigation Test 3" code and rename it "Navigation Test 4".
2. Modify the “on start” block:
 - Check if the block in the "on start" block reads "set UnloadCargo to array of false true false", meaning that after leg 2 (after the spin) the rover should raise the cargo bed for unloading.
 - Change the last zero in "set TimeToDrive" to 2 (seconds).
 - Change the last zero in "set SpeedToDrive" to 100 (full speed),
 - Leave the last value in "set SteerToDrive" at zero.



3. Modify the “forever” block:
 - After the last "Drive Wheels with Speed 0 and Steer 0" block, insert a new "if ... then" block.
 - Copy a "SteerToDrive get value at LegNumber - 1" block from above, past and place it between the "if" and the "then".
 - Change the "SteerToDrive" variable in this block to "UnloadCargo",
 - Look inside the "on radio received ..." block for the two "while BedAngle ..." blocks.
 - Copy both, paste them, drag them over to the end of the "forever" block, and drop them inside the newly created "if UnloadCargo get value ..." block.

Continued next page.



Time to Code: Navigation Test 4 (continued)

```
Drive Wheels with Speed 0 and Steer 0
if UnloadCargo get value at LegNumber - 1 then
  while BedAngle < BedLevel + BedTilt
  do
    change BedAngle by 5
    servo write pin P14 (write only) to BedAngle
    pause (ms) 500
  while BedAngle > BedLevel
  do
    change BedAngle by -5
    servo write pin P14 (write only) to BedAngle
    pause (ms) 500
```



Try This: Navigation Test 4

1. Download, upload, and test:
 - First, just turn the rover on and set it in an open space.
 - Does it drive?
 - Does it turn?
 - Does it unload?
 - Does drive again (remember: we made a third leg!)
 - Now place the rover in front of an obstacle and press button A:
 - Does it drive?
 - Does it stop before the obstacle?
 - Does it turn and unload?
 - Does it come back?
 - Test with the remote control as well.

Your code for “Navigation Test 4” should look like this:

```
set DrivingIsEnabled to false
set Distance to 0
set DrivingAuto to true
set LegNumber to 0
set TimeToDrive to array of 5 0.85 2
set SpeedToDrive to array of 70 70 100
set SteerToDrive to array of 0 100 0
set UnloadCargo to array of false true false
set SonarEnabled to array of true false false
servo write pin P14 (write only) to BedAngle
pause (ms) 1000
servo write pin P15 (write only) to HeadStraightAt + HeadPosition
call Sweep the Head
unplot x 2 y 2
set TimeStamp to running time (ms)
```

Above: Second half of the "on start" block of "Navigation Test 4", which now has a third leg, driving away from the obstacle.

9.5 Navigation Test 5

What would we do if we wanted to make more than three legs or moves with the rover. Right now, the rover travels straight, turns (180°), and then travels straight again. Adding legs is actually quite easy, we just need to define more legs by extending the arrays.

Let's add two more legs to the code!



Time to Code: Navigation Test 5 (continued)

1. Beginning with "Navigation Test 4", rename into "Navigation Test 5".
2. Modify the "on start" block:
 - For each of the arrays, click twice on the "+" sign at the end, adding two more elements.
 - Enter 3 and 2 to get "set TimeToDrive to array of 5 0.85 2 3 2".
 - Enter 60 and 80 to get "set SpeedToDrive to array of 70 70 100 60 80".
 - Enter -30 in the fourth and leave the zero in the fifth element of "SteerToDrive".
 - Change "set UnloadCargo ..." to "false true false false false".
 - Change "set SonarEnabled ..." to "true false false true true".

Note: When you add five elements to each array, the array will change the way it looks and instead of each element next to each other in a long horizontal block, the block will now be larger with each element under the other.

See the code changes below (what code should look like section).

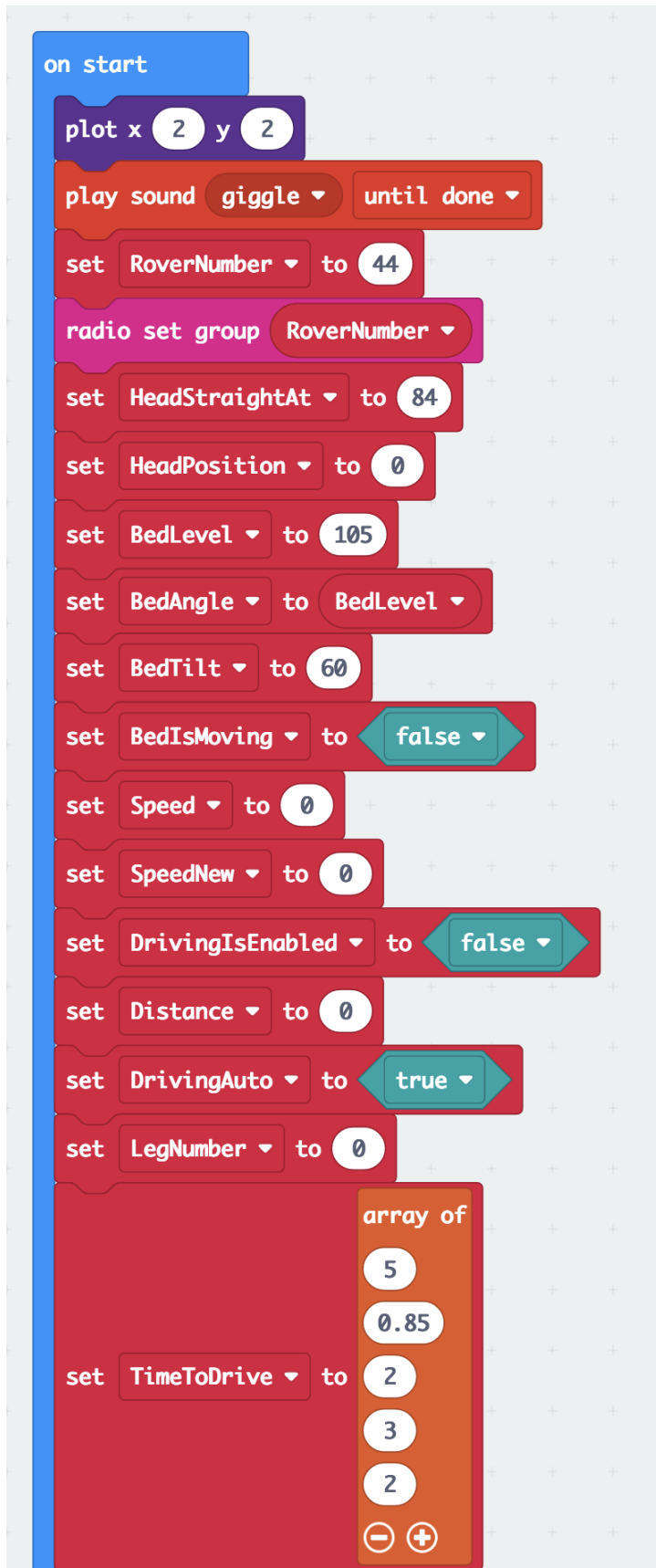


Try this: Navigation Test 5

1. Download, upload, and test:
 - Does your rover drive a 5-legged course?
 - Does it do what you expected?
 - Does the remote control still work?
 - After using the remote control, can you get it to do another course?

Can you see how changing the numbers in the arrays can change the course of the rover and by adding more elements, we can complete a longer course?

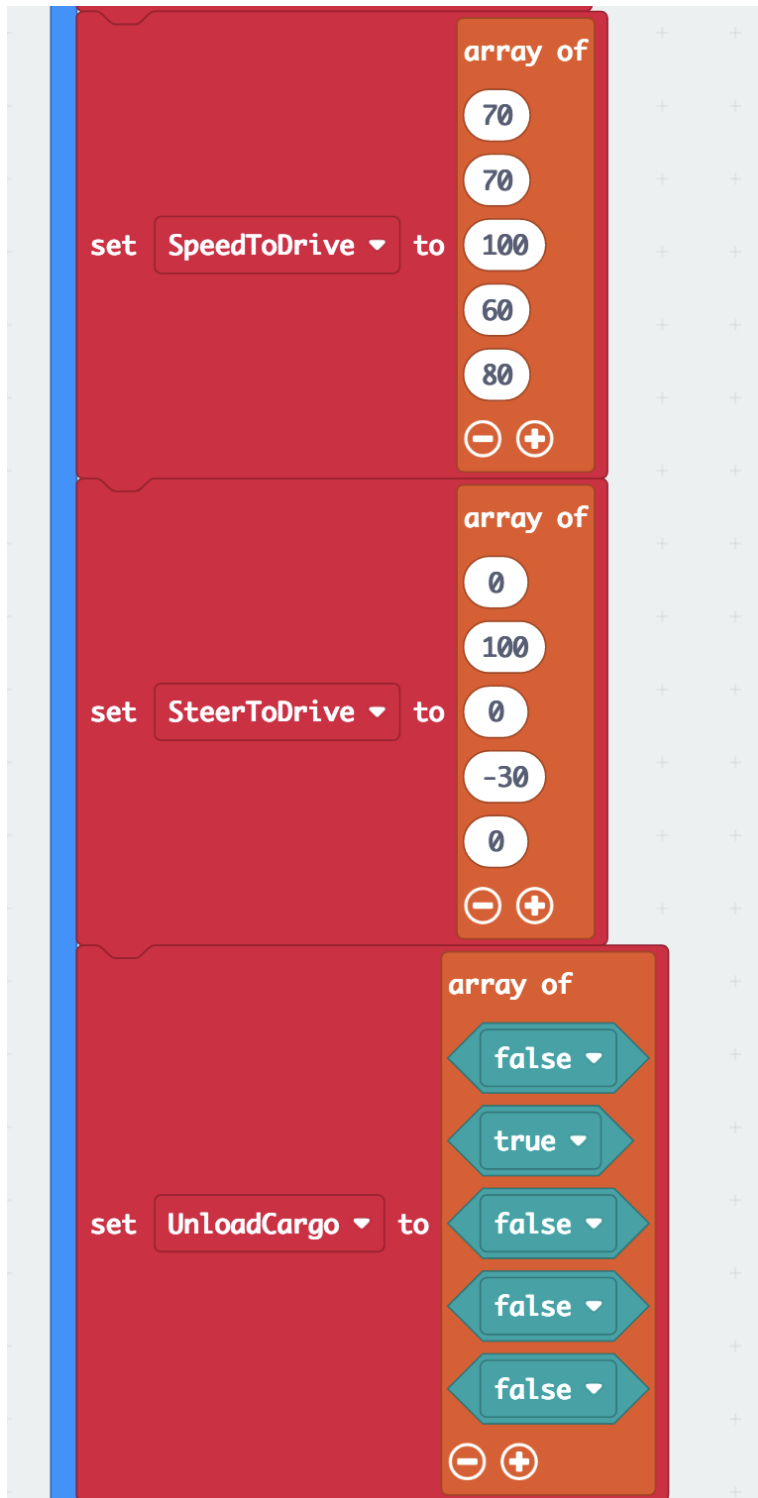
Your code for “Navigation Test 5” should look like this:



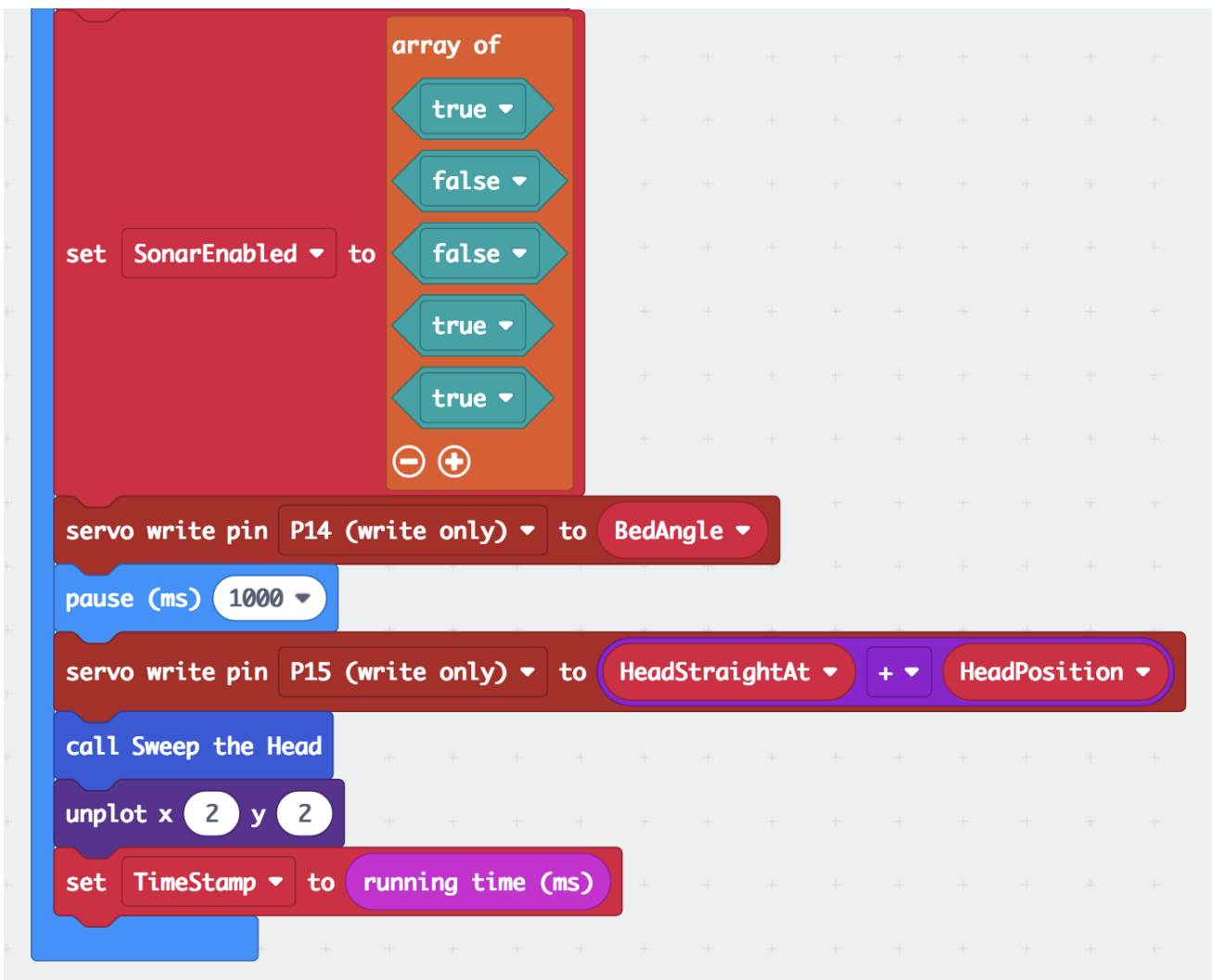
The image shows a Scratch script for a navigation test. It begins with an 'on start' block, followed by a 'plot x 2 y 2' block. The script then plays a 'giggle' sound until done. A series of 'set' blocks initialize variables: RoverNumber to 44, HeadStraightAt to 84, HeadPosition to 0, BedLevel to 105, BedAngle to BedLevel, BedTilt to 60, BedIsMoving to false, Speed to 0, SpeedNew to 0, DrivingIsEnabled to false, Distance to 0, DrivingAuto to true, and LegNumber to 0. A final 'set' block for TimeToDrive is shown with a dropdown menu containing the values 5, 0.85, 2, 3, and 2.

```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set HeadStraightAt to 84
  set HeadPosition to 0
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  set DrivingIsEnabled to false
  set Distance to 0
  set DrivingAuto to true
  set LegNumber to 0
  set TimeToDrive to [array of 5 0.85 2 3 2]
```

Above: The beginning of the "on start" block of "Navigation Test 5" with a 5-legged course for autonomous navigation, including unloading cargo.



Above: Middle part of the "on start" block in "Navigation Test 5" with 5-element arrays.



Above: The ending part of the "on start" block of the "Navigation Test 5" code with a 5-legged course.

The code: "Navigation Test 5": https://makecode.microbit.org/_HgwMviDpEb5w

- Let's summarize what "Navigation Test 5" does:
 - "on radio received ...", it checks the RoverNumber (avoiding potential crosstalk between different "radio groups",
 - receiving an "X", if Driving WAS enabled, it shows and "X", stops, and moves the head straight,
 - receiving an "X", if Driving was NOT enabled, it clears the screen,
 - receiving an "X" in either case, it cancels autonomous driving,
 - receiving a "B", it stops, shows and "X" if Driving is NOT enabled, and moves the cargo bed, either up or down depending on the parameter received after the "B" (this is either 0 or 1),
 - receiving an "S", if Driving is enabled, it drives with the "Speed" and "Steer" variables,
 - receiving an "S", if it is not on an autonomous course, it turns the head according to the "Steer" parameter received from the RC.

9.6 Remote-Controlled Autonomous Navigation (RC 6 & Nav. Test 6)

With the previous code, if we wanted to adjust one of the course parameters, we needed to go back into MakeCode, adjust the numbers in the arrays, download the code, upload the code to the micro:bit of the rover, then start all over with the rover.

We can make all this more flexible if the values of the parameters for the arrays were coming from the remote control. Modifications to both the rover code and the remote-control code are needed for this update. At this point in the camp, time may be more wisely spent learning how to manipulate the code (change array parameters) to operate the rover and have it perform the needed task for the competition events instead of continuing coding practice. The steps for adjusting the code and photos of the completed code are written below for those wanting to continue to hone their coding skills. Otherwise, the links to the new code are posted at the end of this section. For this reason, fewer photos have been provided to show each step of the process.



Time to Code: RC Rover 6

We will first modify the remote-control code.

1. Open the MakeCode code for "**Rover RC 5**" and rename it "**Rover RC 6**".
2. Modify the "on start" block:
 - Make a new variable "RCMode" and place a "set RCMode to 0" block before the "pause 100 ms" block.
 - Drag a "" block from "Text" over the zero, and type "B" (for "(Cargo) Bed") inside.
3. Modify the "forever" block:
 - Drag the "set gx to acceleration ..." block in the "forever" block together with all that follows out of the "forever" block.
 - Get a new "if ... then ... else" block from "Logic" and place it after the "set TimeStampPlotDot to running time ..." block.
 - Get a new "" = "" comparison block and place it between the "if" and "then".
 - Place into the first part of the comparison block the variable "RCMode".
 - Type into the second part "B".
 - Drag the "if AntennaDotIsOn ..." block into the first slot of this "if RCMode = "B" then" block.
 - Click on the "+" sign at the end to create an "else if ... then" slot.
 - Copy the comparison block from above, paste and place it here, change the "B" to an "R"
 - Copy the "if AntennaDotIsOn ..." block from above and place it into the "else if RCMode = "R" then" slot.
 - Change both, the "unplot x 2 ..." and "plot x 2 ..." in the "if RCMode = "B" then" (the first time!!) block to "unplot x 3 ..." and "plot x 3 ...".

Continued next page.



Time to Code: RC Rover 6 (continued)

4. Continue to modify the “**forever**” block:
 - Click one more time on the "+" sign at the end of the "if RCMode = "B" then" block.
 - Copy the comparison block again, place the copy into the new "else if ... then" slot, and change the letter to "A".
 - Copy the "if AntennaDotIsOn ..." block again, place the copy into the new "else if RCMode = "A" then" block, and change the y-value in both "un/plot" blocks from zero to the number 4 (both x-values should be 2).
 - Drag the "set AntennaDotIsOn to not AntennaDotIsOn" block after the last (empty!) "else" slot of the "if RCMode ..." block - this is outside this "if ..." but inside the "if running time ..." block.
 - Copy the "if RCMode = "B" ..." block, paste, and place it after the "if running time ..." block.
 - Delete all three of the "if AntennaDotIsOn then ..." block inside this new "if RCMode ..." block.
 - Drag the "set gx to acceleration ..." together with all its attachments, which we set aside, into the "else if RCMode = "R" then" slot of the new "if RCMode ..." block.
 - Copy the "if MoveCargoBed then ..." block, paste, drag it way up, and place it into the "if RCMode = "B" then" slot of this (second!) "if RCMode ..." block.
5. Modify the “**on start**” block again:
 - Make a new variable and name it "ParameterCode".
 - Place a "set ParameterCode to "Y"" block before the "pause 100 ms" at the end of the "on start" block.
 - Make a new variable and name it "LegNumber".
 - Place a "set LegNumber to 0" block after the "set ParameterCode ..." block.
 - Make a new variable, name it "GotParameter", and place a "set GotParameter to false" block after the "set LegNumber to 0" block.
6. Make new “**function**” blocks:
 - Make a new function and name it "displayRLogo".
 - Place a new "show image ..." block and add a “create image...” block inside.
 - Click on the dots to create an image of an antenna. (see images at the end of the section).
 - Make a new function and name it "displayBLogo".
 - Copy the "show image ..." block from the "function displayRLogo" block and place it into the new function.
 - Change the dots (by clicking on them) in this new "create image" block to a level and raised cargo bed.
 - Make a new function and name it "displayALogo".
 - Copy the "show image ..." block from the "function displayRLogo" block and place it into the new function.
 - Change the dots in this new "create image" block to a base line and a forward arrow.
7. Place a "call displayBLogo" block from "Functions" at the very beginning of the “**on start**” block.

Continued next page.



Time to Code: RC Rover 6 (continued)

8. Modify the “on logo pressed” block:

- Place a new "if ... then ... else" block at the beginning of the "on logo pressed" block.
 - Copy a "RCMode = "R"" comparison block from the "forever" block, paste and place it between the "if" and the "then" of the newly created "if ..." block.
 - Change the "R" in this block to "B".
 - Place a "set RCMode to "R"" block inside the "if RCMode = "B"" block.
 - Get a "call displayRLogo" block from "Function" and place it after the "set RCMode ..." block.
- Click on the "+" sign at the end of the "if RCMode = "B"" block.
 - Copy and paste the "RCMode = "B"" comparison block from above and place it after the "else if".
 - Change the "B" to "R".
 - Copy the "set RCMode to "R"" block from above, paste, and place it inside the new "else if RCMode = "R"" block.
 - Change the "R" to an "A".
 - Place a "call displayALogo" block after the "set RCMode to "A"" block,
- Copy and paste the "if RCMode = "B"" block (yes, the whole thing!), and place it inside the "else" slot of the first "if RCMode = "B"" block.
 - Change the variable in this new "if ..." block to "ParameterCode".
 - Change the letter "B" to "Y".
 - Change the "set RCMode to "R"" block in here to "set LegNumber to 0".
 - Change the "else if RCMode = "R"" block to "else if ParameterCode = "Z"".
 - Change the "set RCMode to "A"" block in here to "set ParameterCode to "t" (lower case "t", that's important: it is for "time" while the uppercase "T" stands for "Steer!")
- Click on the "+" sign four times to create more "else if" slots,
 - Copy and paste the "ParameterCode = "Z"" comparison block four times and place them after the four new "else if" places.
 - Change the first one to "else if ParameterCode = "t"".
 - The second to "else if ParameterCode = "S"".
 - The third to "else if ParameterCode = "T"".
 - The fourth to "else if ParameterCode = "U"".
 - Place a "set Parameter Code to "S" inside the "else if ParameterCode = "t"" block.
 - Place a "set Parameter Code to "T" inside the "else if ParameterCode = "S"" block.
 - Place a "set Parameter Code to "U" inside the "else if ParameterCode = "T"" block.
 - Place a "set Parameter Code to "O" inside the "else if ParameterCode = "U"" block.
 - Place a "set Parameter Code to "Z" inside the "else" slot.
 - Add a "change LegNumber by 1" block right thereafter.

Continued next page.



Time to Code: RC Rover 6 (continued)

9. Continue to modify the “on logo pressed” block:

- Place a new "if true then" block before the "radio send string ..." block.
 - Get a new "0 < 0" comparison block, place it over the "true" in the newly created "if ..." block.
 - Change it to "if LegNumber > 0 then".
 - Move the "radio send string ..." block inside the new "if LegNumber ..." block.
 - Change the "X" to the letter "L".
 - Click twice on the "+" at the end of the "join ..." block.
 - Place a "convert LegNumber to text" after the "L".
 - Place the "ParameterCode" variable in the next place.
- Place a new "set GotParameter to false" block after the "radio send ..." block,
- Place a new "show string "?"" block thereafter.
- Move the "play tone ..." block back to after this "if LegNumber > 0" block - to the very end of the "on logo pressed" block.

10. Modify the “on button A pressed” block”:

- Create a new "if ... then ... else" block and place it at the beginning .
 - Copy and paste from the "on logo pressed" block the "RCMode = "B"" comparison block and place it over the "true" in the new "if ..." block.
 - Drag the "set BedIsLevel to true" block together with the other blocks into the "if RCMode = "B"" then slot.
 - Drag the "play tone ..." block back out and place it after the "if ..." block.
 - Click on the "+" sign at the end of the "if ..." block.
 - Copy the "RCMode = "B"" comparison block, place it after the "else if", and change the letter "B" to "R".
 - Copy both, "set BedIsLevel ..." and "set MoveCargoBed ..." blocks and place the copy into the "else if RCMode = "R" slot.
 - Get a new "radio send string """" block and place it inside the "else" slot of the "if ..." block in the "on button A ..." block.
 - Place a "join "Hello" "World"" block from "Text" over the empty string.
 - Place a "convert 0 to text" block over the "Hello".
 - Replace the zero here with the "RoverNumber" variable.
 - Type "Go!" over the "World".
- Download and upload this code to the RC micro:bit. It will be tested after modifying the rover code.
- See the photos of what the code should look like starting on the next page.

Your code for “RC Rover 6” should look like this:

The image shows a Scratch script with three custom functions and an 'on start' block. The functions are:

- displayBLogo**: A function that creates a 5x5 grid of squares. The top row has 4 squares, the second row has 3, the third row has 2, and the bottom two rows have 1 square each, forming a right-angled triangle. It then shows this image at an offset of 0.
- displayRLogo**: A function that creates a 5x5 grid of squares. The top row has 1 square, the second row has 2, the third row has 3, the fourth row has 4, and the bottom row has 5 squares, forming a right-angled triangle. It then shows this image at an offset of 0.
- displayALogo**: A function that creates a 5x5 grid of squares. The top row has 5 squares, the second row has 4, the third row has 3, the fourth row has 2, and the bottom row has 1 square, forming a right-angled triangle. It then shows this image at an offset of 0.

The 'on start' block contains the following code:

```
on start
  call displayBLogo
  play sound spring until done
  set RoverNumber to 44
  radio set group RoverNumber
  set BedIsLevel to true
  set MoveCargoBed to false
  set Speed to 0
  set Steer to 0
  set Message to ""
  set TimeStampPlotDot to running time (ms)
  set AntennaDotIsOn to true
  set RCMODE to "B"
  set ParameterCode to "Y"
  set LegNumber to 0
  set GotParameter to false
  pause (ms) 100
```

Above: the new display functions and additional variables in the "on start" block of "Rover RC 6".

```
on logo pressed
  if RCMode = "B" then
    set RCMode to "R"
    radio send string join convert RoverNumber to text "RC!"
    call displayRLogo
  else if RCMode = "R" then
    set RCMode to "A"
    radio send string join convert RoverNumber to text "ATD"
    call displayALogo
  else
    if ParameterCode = "Y" then
      set LegNumber to 0
      set RCMode to "B"
      radio send string join convert RoverNumber to text "Bed"
      call displayBLogo
    else if ParameterCode = "Z" then
      set ParameterCode to "t"
    else if ParameterCode = "t" then
      set ParameterCode to "S"
    else if ParameterCode = "S" then
      set ParameterCode to "T"
    else if ParameterCode = "T" then
      set ParameterCode to "U"
    else if ParameterCode = "U" then
```

```

else if ParameterCode = "U" then
  set ParameterCode to "D"
else
  set ParameterCode to "Z"
  change LegNumber by 1
  if LegNumber > 0 then
    radio send string join convert RoverNumber to text "L" convert LegNumber to text ParameterCode
    set GotParameter to false
    show string "?"
    play tone High B for 1/8 beat
  end if
end if

on button B pressed
  set BedIsLevel to false
  set MoveCargoBed to true
  play tone High B for 1/8 beat

```

Above: The "on logo pressed" and "on button B pressed" blocks in "Rover RC 6". The remote-control cycles through B (bed) --> R (RC) --> A (autonomous) --> B (bed) modes on pressing the logo button.

```

on button A pressed
  if RCMODE = "B" then
    set BedIsLevel to true
    set MoveCargoBed to true
  else if RCMODE = "R" then
    set BedIsLevel to true
    set MoveCargoBed to true
  else
    radio send string join convert RoverNumber to text "Go!"
    play tone High B for 1/8 beat
  end if
end on button A pressed

```

Above: "on button A pressed" block of "Rover RC 6" code, capable of starting an autonomous course with the "Go!" command.

```
forever
  if <running time (ms) > > TimestampPlotDot + 100 then
    set TimestampPlotDot to running time (ms)
    if RCMode = "B" then
      if AntennaDotIsOn then
        unplot x 3 y 0
      else
        plot x 3 y 0
    else if RCMode = "R" then
      if AntennaDotIsOn then
        unplot x 2 y 0
      else
        plot x 2 y 0
    else if RCMode = "A" then
      if AntennaDotIsOn then
        unplot x 2 y 4
      else
        plot x 2 y 4
    else
      // Empty block
    set AntennaDotIsOn to not AntennaDotIsOn
```



```
if RCMode = "B" then
  if MoveCargoBed then
    if BedIsLevel then
      set Message to join convert RoverNumber to text "B" "001"
    else
      set Message to join convert RoverNumber to text "B" "000"
    set MoveCargoBed to false
    radio send string Message
    pause (ms) 100
  else if RCMode = "R" then
    set gx to acceleration (mg) x
    set gy to acceleration (mg) y
    set Speed to square root (gx x gx + gy x gy)
    set Speed to Speed integer ÷ 10
    if Speed > 100 then
      set Speed to 100
    if gy > 0 then
      set Speed to -1 integer x Speed
```

```

+
set Speed to Speed + 200
set Steer to gx integer + 10
if Steer > 100 then
  set Steer to 100
+
if Steer < -100 then
  set Steer to -100
+
set Steer to Steer + 400
set Message to join convert RoverNumber to text "S" convert Speed to text
set Message to join Message "T" convert Steer to text
if MoveCargoBed then
  if BedIsLevel then
    set Message to join convert RoverNumber to text "B" "001"
  else
    set Message to join convert RoverNumber to text "B" "000"
+
set MoveCargoBed to false
radio send string Message
pause (ms) 100
+
radio send string Message
set Message to ""
else if RCMODE = "A" then

```

Above: the "forever" block of "Rover RC 6" is almost identical with that of "Rover RC 5".



Time to Code: Navigation Test 6

Now program the rover micro:bit to make the needed adjustments.

1. Load "Navigation Test 5" and rename it "Navigation Test 6",
2. Modify the "on start" block:
 - Change the "true" in the "set DrivingAuto to true" block at the beginning of the "on start" block to "false".
 - Make a new variable, name it "RVMode", and place a "set RVMode to "B"" block before the "set TimeToDrive to array ..." block.
 - Make a new variable, name it "Message", and place a "set Message to """ block right after,
 - Make a new variable, name it "DotIsOn", and place a "set DotIsOn to false" block after the "unplot ..." block at the end of the "on start" block.
 - Make a new variable and name it "TimeStampDot" and place a "set TimeStampDot to running time (ms)" block at the very end of the "on start" block.
3. Modify the "forever" block:
 - Click on the "+" sign of the "if DrivingAuto and LegNumber = 0 then" block to create an "else" slot.
 - Create and place an "if running time (ms) > TimeStampDot + 100 then" block into the "else" slot.
 - Place a "set TimeStampDot to running time (ms)" block inside this "if ..." block.
 - Place a new "if ... then ... else" block right after.
 - Place the variable "DotIsOn" over the "true" in the new "if ..." block.
 - Place an "unplot x 2 y 2" block in the "if DotIsOn then" slot.
 - Place a "plot x 2 y 2" block in the "else" slot.
 - Place a "set DotIsOn to not DotIsOn" block after the "if DotIsOn then ... else ..." block
 - At the end of the big "for index from 0 to LegNumber - 1 then" block, after the "change LegNumber by 1" block, add a new "if ... then ... else" block.
 - Insert between the "if" and "then" a "0 < 0" comparison block.
 - Place the variable "LegNumber" over the first zero.
 - Place a "length of array list" block from "Arrays" over the second zero, Change the "list" in there to the variable "TimeToDrive".
 - Drag the "show number LegNumber" block into the "if ..." slot.
 - Place a "set DrivingAuto to false" block into the "else ..." slot.
 - Place a "set LegNumber to 0" block right thereafter.
 - Place a "plot x 2 y 2" block right after the previous one.

Continued next page.



Time to Code: Navigation Test 6 (continued)

4. Modify the “on radio received” block:
 - Click on the "+" sign above the "+" sign at the very end of the "on radio received ..." block (important: do NOT click on the last one!).
 - Copy from above the "substring of receivedString from 2 of length 1 = "S"" block and place the copy between the new "else if" and "then".
 - Change the number 1 in here to 3.
 - Type over the "S" in here "RC!",
 - Create two more "else if" slots by clicking on that same "+" sign twice.
 - Copy the "substring of receivedString from 2 of length 3 = "RC!"" comparison block and paste it twice, place the two blocks between the two new "else if" and "then" spaces.
 - Change the "RC!" in the first one to "ATD".
 - Change the "RC!" in the second one to "Go!".
 - Create and place a "set RVMode to "R"" block into the "else if substring ... = "RC!"" slot.
 - Place a "set LegNumber to 0" block right after.
 - Place a "set DrivingAuto to false" block next.
 - Place a "set DrivingIsEnabled to true" block next.
 - Copy all four blocks twice and place them into the next two (the "ATD" and "Go!") "else if" slots.
 - Change the letter "R" in both to "A".
 - Change the "set DrivingIsEnabled to true" in both, the "ATD" and "Go!" slots to "false".
 - Change the "set DrivingAuto to false" only in the "Go!" slot to "true".
 - Add a "set TimeStamp to running time (ms)" block at the end of the "Go!" slot.
 - Add a "Drive Wheels with Speed 0 and Steer 0" block from "StemSeals" block at the end of the "ATD" block.
 - Click on the "+" before the last "+" at the very end of the "on radio received ..." block.
 - Place a "substring of receivedString from 2 of length 1 = "L"" comparison block between the new "else if" and "then".
 - Place a "set LegNumber to parse to number substring of receivedString from 3 of length 1" block inside the new "else if" slot.
 - Place a "set Message to join convert RoverNumber to text "L" convert LegNumber to text" block next.
 - Insert a new "if ... then ... else" block right after.
 - Copy the "substring ..." comparison block from above, paste it into the new "if ..." block, and change it to "if substring of receivedString from 4 of length 1 = "t" then".
 - Place inside this new "if ..." slot "set Message to join Message "t" convert 1000 + round 10 * TimeToDrive get value at LegNumber - 1" to text.
 - Click on the "+" sign of the new "if substring ... = "t"" four times to create four "else if" slots.
 - Copy and paste the "substring of receivedString from 4 of length 1 = "t"" comparison block four times and paste each between the "else if" and "then" in the newly created slots.
 - Change the first one to "S" for Speed,
 - The second to "T" for Steer,
 - The third to "U" for unload,
 - and the fourth to "O" for obstacle avoidance.

Continued next page.



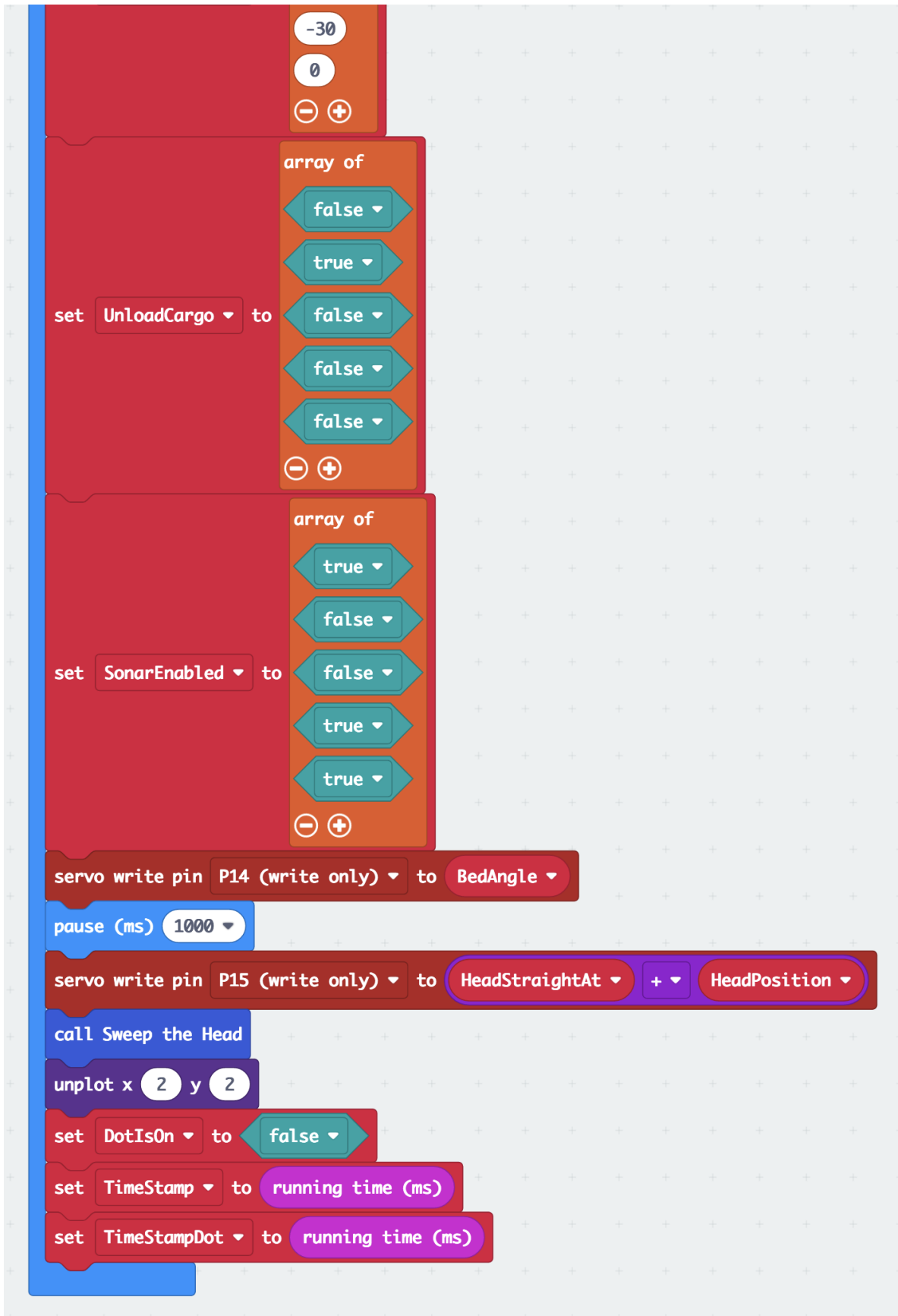
Time to Code: Navigation Test 6 (continued)

5. Continue modifying the “on radio received” block:

- Copy that last "set Message to join Message ..." block and paste it four times.
 - Place each in one of the new "else if ..." slots.
 - Change the "t" in the first one to "S" (should be in the "substring ... = "S" slot!).
 - Change the "t" in the second to "T",
 - Change the "t" in the third to "U",
 - and the last to "O".
 - Change the number 1000 in the first (with the "S") to 200,
 - Change the number 1000 in the second (with the "T") to 400.
 - Delete the "round 10 * ..." block in the first (with the "S").
 - Delete the "round 10 * ..." block in the second (with the "T").
 - Copy the "TimeToDrive get value ..." from above and paste the copy over the zero after the "200 +".
 - Change the "TimeToDrive" to "SpeedToDrive".
 - Copy the "SpeedToDrive get value ..." from above, and paste the copy over the zero after the "400 +",
 - Change the "SpeedToDrive" to "SteerToDrive".
 - Delete the "1000 + round ..." in the third (with the "U"),
 - Delete the "1000 + round ..." in the fourth (with the "O").
 - Copy the "SteerToDrive get value ..." from above and paste the copy over the zero in the "convert ..." block in the "U" Message.
 - Change the "SteerToDrive" to "UnloadCargo".
 - Copy the "UnloadCargo get value ..." from above, and paste the copy over the zero in the "convert ..." block in the "O" Message.
 - Change the "UnloadCargo" variable to "SonarEnabled".

- Download and upload this code to the Rover micro:bit.
- See the photos of what the code should look like starting on the next page.

```
on start
  plot x 2 y 2
  play sound giggle until done
  set RoverNumber to 44
  radio set group RoverNumber
  set HeadStraightAt to 84
  set HeadPosition to 0
  set BedLevel to 105
  set BedAngle to BedLevel
  set BedTilt to 60
  set BedIsMoving to false
  set Speed to 0
  set SpeedNew to 0
  set DrivingIsEnabled to false
  set Distance to 0
  set DrivingAuto to false
  set LegNumber to 0
  set RVMode to "B"
  set Message to ""
  set TimeToDrive to array of
    5
    0.85
    2
    3
    2
```



Above: additional variables in the "on start" block of the "Navigation Test 6" code.


```

plot x 4 - indexColumn y 4 - indexRow
else
  unplot x 4 - indexColumn y 4 - indexRow
else
  if < DrivingAuto > and < LegNumber > = 0 then
    set LegNumber to 1
    show number LegNumber
    set TimeStamp to running time (ms)
  else
    if < running time (ms) > < TimeStampDot > + 100 then
      set TimeStampDot to running time (ms)
      if < DotIsOn > then
        unplot x 2 y 2
      else
        plot x 2 y 2
      set DotIsOn to not DotIsOn
    for index from 0 to LegNumber - 1
    do
      if < running time (ms) < < TimeStamp > + 1000 x < TimeToDrive > get value at LegNumber - 1 then

```

Above: Center part of the "forever" block of "Navigation Test 6" code. When driving autonomously, it displays the LegNumber it is on, otherwise we see a blinking dot.

```
while BedAngle > BedLevel
do
  change BedAngle by -5
  servo write pin P14 (write only) to BedAngle
  pause (ms) 500
+
set TimeStamp to running time (ms)
change LegNumber by 1
if LegNumber < length of array TimeToDrive then
  show number LegNumber
else
  set DrivingAuto to false
  set LegNumber to 0
  plot x 2 y 2
+
+
+
+
```

The image shows a Scratch code block with a 'while' loop. The loop condition is 'BedAngle > BedLevel'. Inside the loop, there is a 'do' block containing: 'change BedAngle by -5', 'servo write pin P14 (write only) to BedAngle', and 'pause (ms) 500'. Below the loop, there are several blocks: a 'set TimeStamp to running time (ms)' block, a 'change LegNumber by 1' block, an 'if LegNumber < length of array TimeToDrive then' block containing a 'show number LegNumber' block, an 'else' block containing 'set DrivingAuto to false', 'set LegNumber to 0', and 'plot x 2 y 2'. The code block is decorated with green and blue vertical bars on the left side.

Above: minor changes to the end of the "forever" block of "Navigation Test 6".

```
on radio received receivedString
  if parse to number substring of receivedString from 0 of length 2 = RoverNumber then
    if substring of receivedString from 2 of length 1 = "X" then
      set DrivingIsEnabled to not DrivingIsEnabled
      if not DrivingIsEnabled then
        show image icon image [dotted] at offset 0 +
        Drive Wheels with Speed 0 and Steer 0
        servo write pin P15 (write only) to HeadStraightAt + 0
      else
        clear screen
        +
        set DrivingAuto to false
        set LegNumber to 0
        pause (ms) 100
    else if substring of receivedString from 2 of length 1 = "B" then
      if substring of receivedString from 2 of length 3 = "Bed" then
        set RVMode to "B"
        set LegNumber to 0
        set HeadPosition to round 0 ÷ 1.4
        servo write pin P15 (write only) to HeadStraightAt + HeadPosition
        Drive Wheels with Speed 0 and Steer 0
      else
```

```
else
  set HeadPosition to round 0 + 1.4
  servo write pin P15 (write only) to HeadStraightAt + HeadPosition
  Drive Wheels with Speed 0 and Steer 0
  clear screen
  if not DrivingIsEnabled then
    show image icon image [grid icon] at offset 0 +
    set BedIsMoving to true
    if parse to number substring of receivedString from 3 of length 3 > 0 then
      while BedAngle < BedLevel + BedTilt
        do
          change BedAngle by 5
          servo write pin P14 (write only) to BedAngle
          pause (ms) 500
      else
        while BedAngle > BedLevel
          do
            change BedAngle by -5
            servo write pin P14 (write only) to BedAngle
            pause (ms) 500
        set BedIsMoving to false
    else if substring of receivedString from 2 of length 1 = "S" then
```

```
else if substring of receivedString from 2 of length 1 = "S" then
  set SpeedNew to parse to number substring of receivedString from 3 of length 3
  set SpeedNew to SpeedNew - 200
  set Steer to parse to number substring of receivedString from 7 of length 3
  set Steer to Steer - 400
  if SpeedNew ≥ -100 and SpeedNew ≤ 100 then
    set Speed to SpeedNew
    if not DrivingAuto then
      set HeadPosition to round Steer ÷ 1.4
      servo write pin P15 (write only) to HeadStraightAt + HeadPosition
    if DrivingIsEnabled then
      Drive Wheels with Speed Speed and Steer Steer
  else if substring of receivedString from 2 of length 3 = "RC!" then
    set RVMode to "R"
    set LegNumber to 0
    set DrivingAuto to false
    set DrivingIsEnabled to true
  else if substring of receivedString from 2 of length 3 = "ATD" then
```

```

else if substring of receivedString from 2 of length 3 = "ATD" then
  set RVMode to "A"
  set LegNumber to 0
  set DrivingIsEnabled to false
  set DrivingAuto to false
  Drive Wheels with Speed 0 and Steer 0

else if substring of receivedString from 2 of length 3 = "Go!" then
  set RVMode to "A"
  set LegNumber to 0
  set DrivingIsEnabled to false
  set DrivingAuto to true
  set TimeStamp to running time (ms)

else if substring of receivedString from 2 of length 1 = "L" then
  set LegNumber to parse to number substring of receivedString from 3 of length 1
  set Message to join convert RoverNumber to text "L" convert LegNumber to text

if substring of receivedString from 4 of length 1 = "t" then

```

```

if substring of receivedString from 4 of length 1 = "t" then
  set Message to join Message "t" convert 1000 + round 10 x TimeToDrive get value at LegNumber - 1 to text

else if substring of receivedString from 4 of length 1 = "S" then
  set Message to join Message "S" convert 200 + SpeedToDrive get value at LegNumber - 1 to text

else if substring of receivedString from 4 of length 1 = "T" then
  set Message to join Message "T" convert 400 + SteerToDrive get value at LegNumber - 1 to text

else if substring of receivedString from 4 of length 1 = "U" then
  set Message to join Message "U" convert UnloadCargo get value at LegNumber - 1 to text

else if substring of receivedString from 4 of length 1 = "0" then
  set Message to join Message "0" convert SonarEnabled get value at LegNumber - 1 to text

else
  +
else
  +
+
+

```

Above: major changes to the whole "on radio received" block of "Navigation Test 6". It includes that the rover, on request by the RC reports current autonomous course parameters. The idea behind is, how easy it may be to edit course parameters on the fly by only using the remote control.



Try this: Navigation Test 6

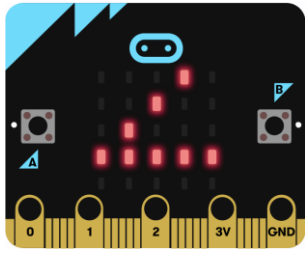
1. Download, upload and test your code (**RC Rover 6 & Navigation Test 6**):

"Rover RC 6": https://makecode.microbit.org/_4zK2jgAEw2tD

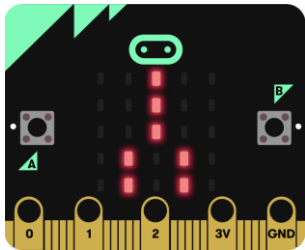
"Navigation Test 6": https://makecode.microbit.org/_b382w4bKqYF3

2. Turn the rover on (with the RC off).
 - Does the bed level?
 - Does the head turn?
 - Does the rover drive?
3. Turn the RC micro:bit on.
 - Does the rover drive?
 - What do the LEDs on the RC show?
 - Push button A: does the bed move?
 - Wait ... push button B: does the bed level?
 - Tilt the RC: does the rover drive?
4. Press the golden logo.
 - What does the RC's display show?
 - Do you hear the rover?
 - Tilt and drive the rover: does it work as you expected?
5. Press the golden logo.
 - Does the rover stop?
 - Tilt the RC: does the rover remain stopped?
 - What does the RC's display show?
6. Press button A,
 - What does the rover do?

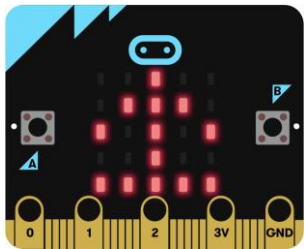
- Let's summarize what the remote control, the RC micro:bit does:
 - with the logo we can cycle through the modes: B --> R --> A --> B,
 - every time we enter another mode, the micro:bit displays



for B mode (move cargo bed),



for R mode (RC, remote control mode),



- for A mode (autonomous driving mode).
- also, each mode has its own blinking LED:
- in B mode the top LED (3,0) blinks,
- in R mode the top LED (2,0) blinks,
- in A mode the bottom LED (2,4) blinks.
- in B mode:
 - button A unloads (raises the cargo bed), button B levels it,
- in R mode:
 - tilting the RC drives the rover
 - buttons A and B stop the rover and move the bed,
- in A mode:
 - button A sends the rover on its autonomous course: note: this is the course predefined by the rover's code in "on start"!

- The code: "Rover RC 6": https://makecode.microbit.org/_4zK2jgAEw2tD
- "Navigation Test 6": https://makecode.microbit.org/_b382w4bKqYF3

9.7 Navigation 7 and Rover RC 7

Navigation 7 will be the last edit of the code and will put the remote control (RC) in A mode when button B is pressed. It will allow the RC and rover to “talk” back and forth to each other and allow for course (leg) changes without having to go back into MakeCode.

Next, we want the RC, in A mode, when button B is pressed

- 1) request from the rover,
 - a. for each leg,
 - b. all parameters,
 - c. first, “t”, then “S”, then “T”, then “U”, then “O”,
- 2) and repeat this for each leg, which does not have $t = 0$
- 3) and receive the (on the rover!) current information,
- 4) and allow to change that information using buttons A and B (on the RC),
- 5) and, when the change is confirmed, transmit that information back to the rover.
- 6) and finally, being able to exit course editing by shaking the RC.

The changes for the Navigation Test 7 and RC Rover 7 will be placed in the boxes below (again with little or no pictures) and the links for both programs will be placed at the end for easy upload. This will allow you more time to focus on how to run the program versus coding it.



Time to Code: RC Rover 7

1. Open the MakeCode code for "**Rover RC 6**" and rename it "**Rover RC 7**".
2. Modify the “forever” block:
 - Insert a new “if ... then ... else” block right underneath the “else if RCMode = “A”” block.
 - Copy the “RCMode = “A”” comparison block, paste and place it over the “true” in the newly created “if ...” block.
 - Change the “RCMode” variable to “ParameterCode”.
 - And change the letter “A” to “Y”.
 - Drag the whole “if AntennaDotIsOn then” block from below the new “if ...” block into its first empty slot.
 - Click on the “+” sign of the new “if ParameterCode ...” block (not the “if AntennaDotIsOn ...”!) to create a new “else if ...” slot.
 - Place a new “ParameterCode = “t”” comparison block into the empty hexagon of this “else if ...”.
 - Copy the whole “if AntennaDotIsOn ...” block, paste, and insert it into the just created “else if ParameterCode = “t” then” slot, and change the number 4 to 3 in both, the “unplot ...” and “plot ...” blocks. This will change the blinking dot from the center of the “baseline” when Autonomous Mode has been selected to a blinking dot one up from the bottom row. This will indicate the decimal point for the time display. We leave the “else ...” slot below empty: No blinking dots will be desired for the other editing options, Speed, Steer, Unload and Obstacle Avoidance.

Continued next page.



Time to Code: RC Rover 7 (continued)

3. Continue modifying the “forever” block:

- Scroll all the way down to the end of the “forever” block until you find the “else if RCMODE = “A” then ...” block.
- Place a new “if ... then ... else ...” block into the first slot of the “else if RCMODE = “A” then ...” slot.
 - Get an “... and ...” block from the “Logic” toolbox and place it over the “true” of the new “if ...” block
- Make a new variable and name it “NeedParameter”.
 - Change the “... and ...” block to a “NeedParameter and not GotParameter” block.
- Get a new “if ... then ... else ...” block and place it in the first slot of the just created “if NeedParameter and not GotParameter then ...” slot.
 - Get a “0 < 0” comparison block and place it over the “true” of the new “if ...” block.
 - Create a new “AttemptsToReceive” variable and place it over the first zero.
 - Type over the second zero the number 10. All this means: If the RC needs to know a parameter from the rover, then it will try to receive this parameter. But, for example, if the rover is turned off, the RC can never receive anything. Therefore, it only will try ten times and then give up and tell you something is wrong.
- Create a new “set Message to join convert RoverNumber to text “L” convert LegNumber to text” block and place it into the slot of the “if AttemptsToReceive < 10 then” block (you can copy the one you find ~ 16 block above and edit it).
- Create a new “set Message to join Message ParameterCode “?”” block and place it right after the previous “set Message ...” block (you can copy from above and edit).
- Create a new “if not GotParameter then ... else ...” block and place it right after the previous “set Message ...” block.
- Place a new “radio send string Message” block into the first empty slot of the new “if ...” block.
- Place a “change AttemptsToReceive by 1” block right after.
- Place a new “show string “?”” block right after.
- Create a new variable named “ShowsQuestionMark”.
 - Place a “set ShowsQuestionMark to true” block right after the “show string “?”” block.
- Place a “set Message to “”” block into the empty “else ...” slot. This all means: The RC, if it did not yet get the requested parameter, will send the request to the rover, and display a question mark. If it already got the requested parameter, it would clear the Message variable and not send a request.
- In the “else ...” slot of the “if AttemptsToReceive < 10”, place a new “set AttemptsToReceive to 0” block.
 - Add a “play tone Low F for 1/2 beat.
 - Add a “play tone Low D# for 1/2 beat.
 - Add a “play tone Low C# for 1 beat right after the “set AttemptsToReceive to 0” block. This means, if all ten attempts asking for the parameter fail, then you will hear three low tones in descending order.

Continued next page.



Time to Code: RC Rover 7 (continued)

4. Continue modifying the “forever” block:

Now we need to get the display to display always helpful information, for each of the tasks while editing a course:

- In the empty “else ...” slot of the “if NeedParameter and not GotParameter”, place a new “if ShowsQuestionMark then ... else ...” block.
- Place another new “if ParameterCode = “t”” block into the empty “if ShowsQuestionMark then ...” block.
- Make a new function named “**displayTimeOfLeg**”, move the “function ...” block out of the way for now, but place the “call displayTimeOfLeg” block into the slot of the “if ParameterCode = “t” then ...” block.
- Place a “set ShowsQuestionMark to false” block right after.
- Click on the “+” of the “if ParameterCode ...” block to create an “else if ...” slot.
 - Place a “ParameterCode = “S”” comparison block into the empty hexagon (you can copy from above and edit).
- Make a new function named “displaySpeed”, move the “function ...” block out of the way for now, but place the “call displaySpeed” block into the slot of the “if ParameterCode = “S” then ...” block.
- Place a “set ShowsQuestionMark to false” block right after.
- Click on the “+” of the “if ParameterCode ...” block to create an “else if ...” slot.
 - Place a “ParameterCode = “T”” comparison block into the empty hexagon (you can copy from above and edit).
- Make a new function named “displaySteer”, move the “function ...” block out of the way for now, but place the “call displaySteer” block into the slot of the “if ParameterCode = “t” then ...” block.
- Place a “set ShowsQuestionMark to false” block right after.
- Click on the “+” of the “if ParameterCode ...” block to create an “else if ...” slot.
 - Place a “ParameterCode = “U”” comparison block into the empty hexagon (you can copy from above and edit).
- Make a new function named “displayCargo”, move the “function ...” block out of the way for now, but place the “call displayCargo” block into the slot of the “if ParameterCode = “U” then ...” block.
- Place a “set ShowsQuestionMark to false” block right after.
- Click on the “+” of the “if ParameterCode ...” block to create an “else if ...” slot.
 - place a “ParameterCode = “O”” comparison block into the empty hexagon (you can copy from above and edit).
- Make a new function named “displaySonar”, move the “function ...” block out of the way for now, but place the “call displaySonar” block into the slot of the “if ParameterCode = “O” then ...” block.
- Place a “set ShowsQuestionMark to false” block right after.

Continued next page.



Time to Code: RC Rover 7 (continued)

5. Search for the place where you left the empty “**function displayTimeOfLeg**” earlier when you put it out of the way.
 - Place a “clear screen” block in the empty slot,
 - Make two new variables, “DigitOfTime” and “TimeOfLeg”, and place a “set DigitOfTime to truncate TimeOfLeg” block after the “clear screen” block. You find the “truncate ...” in Math, it is the “round 0” block.
 - Place a “for index from 0 to 4” block from “Loops” after, rename the “index” variable to “indexColumn” (renaming an index variable needs to be done in the “Variables” toolbox) and change the number 4 to 1.
 - Place another “for index from 0 to 4” block inside the first loop, rename the “index” variable to “indexRow”, leave the number 4.
 - Place a new “if ... then ... else ...” block inside the empty slot of the second loop, and drag a “ $0 < 0$ ” block over the “true”.
 - Change the inequality sign to a greater than.
 - Replace the first zero with the “DigitOfTime” variable.
 - Replace the second zero with a “ $0 + 0$ ” block from “Math”.
 - Replace the first zero in here with “indexRow” (you can simply drag it from the loop).
 - Replace the second zero in here with “ 0×0 ” (multiplication!) from “Math”.
 - Replace the first zero here with the number 5.
 - Replace the second zero here with “indexColumn” (you can drag from the loop).
 - In the first empty slot of the “if DigitOfTime > indexRow ...” block, place a “plot x indexColumn y indexRow” block.
 - Place a “unplot x indexColumn y indexRow” block in the empty “else ...” slot.
 - Place a “set DigitOfTime to round($10 \times (\text{TimeOfLeg} - \text{DigitOfTime})$)” block after the two loops. Watch for the parenthesis of the calculation!
 - Copy the “for indexColumn ...” loop (and all in it) and place it at the end of the “function ...” block.
 - Change in both, “plot ...” and “unplot ...” blocks the “indexColumn” to “ $3 + \text{indexColumn}$ ”.

The RC micro:bit can now display the “TimeOfLeg” faster than it would be able to display the (scrolling!) number, let’s say 1.5: The 1 of 1.5 is displayed as a single dot on the left side of the RC, the 5 is displayed as five dots on the right side of the RC, and in between them will be the blinking dot, indicating the decimal point.

Similarly, the time 5 sec will be displayed as the number 5.0, i.e. five LEDs left, no LED right, with the decimal point in between. This way all times from 0.0 to 9.9 sec can be accurately and quickly displayed.

Continued next page.



Time to Code: RC Rover 7 (continued)

6. Display the Speed on the RC:

- Find the **“function displaySpeed”** block,
- Place a “clear screen” block at the beginning,
- Place a “pause (ms) 100” block next,
- Place an “if Speed < 0 then ...” block next,
- Place a “for index from 0 to 2” block inside the “if ...”,
- Place a “plot x 3 - index y 4” block next inside (remember: that’s the minus sign for negative Speed!),
- Place a “plot x 2 y 4” block into the empty “else ...” slot.
- Create a “for indexRow from 0 to 2” block next after (outside!) the “if ...” block,
- Create a “for indexColumn from 0 to 4” block inside the first loop,
- Create a new “if absolute of Speed > (indexColumn + (5 x indexRow)) x 10 then ...” block and place it in the inner loop.
- Place a “plot x indexColumn y indexRow” block into the empty slot of the “if ...” block,
- and an “unplot x indexColumn y indexRow” block in the empty “else ...” slot.

7. Display the Steer variable on the RC

- Find the **“function displaySteer”** block,
- Place a “clear screen” block at the beginning,
- Place a “pause (ms) 100” block next,
- Place a “plot x 2 y 0” block next. Notice: the front center LED is always on when the Steer variable is displayed. This makes it easily distinguishable from any of the other variables.
- Create an “if absolute of Steer > 90 then ...” block (no “else ...” needed),
 - and place a “plot x 2 y 2” block inside.
- Create an “if absolute of Steer > 80 then ...” block, place it next,
 - and place a “plot x 2 y 3” block inside. Notice: most of the many following blocks are best created by copying, pasting, placing them and editing them.
- Create an “if absolute of Steer > 70 then ...” block, place it next,
 - and place a “plot x 2 y 4” block inside.
- Create an “if Steer < -60 then ...” block, place it next, and place a “plot x 1 y 4” block inside, notice: no absolute!
- Create an “if Steer < -50 then ...” block, place it next, and place a “plot x 0 y 4” block inside,
- Create an “if Steer < -40 then ...” block, place it next, and place a “plot x 0 y 3” block inside,
- Create an “if Steer < -30 then ...” block, place it next, with a “plot x 0 y 2” block inside,
- Create an “if Steer < -20 then ...” block, place it next, with a “plot x 0 y 1” block inside,

Continued next page.



Time to Code: RC Rover 7 (continued)

8. Continue in the “**function displaySteer**” block,
- Create an “if Steer < -20 then ...” block, place it next, with a “plot x 0 y 1” block inside,
 - Create an “if Steer < -10 then ...” block, place it next, with a “plot x 0 y 0” block inside,
 - Create an “if Steer < 0 then ...” block, place it next, with a “plot x 1 y 0” block inside,
 - Create an “if Steer > 60 then ...” block, place it next, with a “plot x 3 y 4” block inside,
 - Create an “if Steer > 50 then ...” block, place it next, with a “plot x 4 y 4” block inside,
 - Create an “if Steer > 40 then ...” block, place it next, with a “plot x 4 y 3” block inside,
 - Create an “if Steer > 30 then ...” block, place it next, with a “plot x 4 y 2” block inside,
 - Create an “if Steer > 20 then ...” block, place it next, with a “plot x 4 y 1” block inside,
 - Create an “if Steer > 10 then ...” block, place it next, with a “plot x 4 y 0” block inside,
 - Create an “if Steer > 0 then ...” block, place it next, with a “plot x 3 y 0” block inside.

For spins we will see a spiral-shape, either on the right or the left, and for milder turns, only part of the spiral will indicate the value of the Steer variable.

9. **Display if cargo is unloaded at the end of the leg, or if obstacle avoidance is enabled during driving the leg.**

- Find the “function displayCargo” block,
- Place a “clear screen” block at the beginning,
- Add an “if ... then ... else ...” block next,
- Create a “UnloadCargo” variable and place it over the “true” in the “if ...” block,
 - Place a “show string “U”” block (capital “U”!) in the first slot,
 - and a “show string “u”” block in the “else ...” slot.
- Find the “function displaySonar” block,
- Place a “clear screen” block at the beginning,
- Add an “if ... then ... else ...” block next,
- Create a “SonarEnabled” variable and place it over the “true” in the “if ...” block,
 - Place a “show string “O”” block (capital “U”!) in the first slot,
 - and a “show string “o”” block in the “else ...” slot.

An uppercase “U” or “O” letter will indicate a “true”, i.e., “do it!” for either unloading the cargo or activating obstacle avoidance. If the same letter is lowercase, it means a “false”, or “don’t do it.”

10. **Cleanup the “on start” block.**

- Find the “on start” block,
- add a “set TimeOfLeg to 0” block after the “set MoveCargoBed to false” block,
- add a “set UnloadCargo to false” block after the “set Steer to 0” block,
- add a “set SonarEnabled to false” block after the “set UnloadCargo to false” block,
- add a “set NeedParameter to false” block after the “set LegNumber to 0” block,
- add a “set ShowQuestionMark to false” block after the “set GotParameter to false” block,
- add a “set AttemptsToReceive to 0” block after the “set ShowQuestionMark to false” block.

Continued next page.



Time to Code: RC Rover 7 (continued)

11. Find the “on button A pressed” block:

- Add a “play tone High G# for 1/8 beat at the very beginning.
- Add a new “if ... then ... else ...” block after the “else ...” of the “if RCMode = “B” then ...” block, before the “radio send string ...” block,
 - Replace the “true” in the new block with “ParameterCode = “Y””,
- Drag (move!) the “radio send string ...” from after the new “if ...” block inside, into the “if ParameterCode = “Y” then ...” slot.
- Click on the “+” of the new “if ...” block to create a new “else if ...” slot,
 - Place a new “ParameterCode = “t”” block (copy and edit from above) into the empty hexagon of the “else if ...”,
- Place a new “change TimeOfLeg by -0.1” block inside the “else if ParameterCode = “t”” block and add a “call displayTimeOfLeg” right after.
- Click on the “+” of the new “if ...” block to create a new “else if ...” slot,
- Place a new “ParameterCode = “S”” block into the empty hexagon of the “else if ...”,
- Place a new “change Speed by -10” block inside the “else if ParameterCode = “S”” block and add a “call displaySpeed” right after.
- Click on the “+” of the new “if ...” block to create a new “else if ...” slot,
 - Place a new “ParameterCode = “T”” block (capital “T” for Steer!) into the empty hexagon of the “else if ...”,
- Place a new “change Steer by -10” block inside the “else if ParameterCode = “T”” block and add a “call displaySteer” right after.
- Click on the “+” of the new “if ...” block to create a new “else if ...” slot,
 - Place a new “ParameterCode = “U”” block into the empty hexagon of the “else if ...”,
- Place a new “set UnloadCargo to not UnloadCargo” block inside the “else if ParameterCode = “U”” block and add a “call displayCargo” right after.
- Click on the “+” of the new “if ...” block to create a new “else if ...” slot,
 - Place a new “ParameterCode = “O”” block into the empty hexagon of the “else if ...”,
- Place a new “set SonarEnabled to not SonarEnabled” block inside the “else if ParameterCode = “O”” block and add a “call displaySonar” right after.
- Delete the “play tone High B for 1/8 beat” block at the end.

Continued next page.



Time to Code: RC Rover 7 (continued)

Now we need to do a similar modification for the B button for changes in the other direction.

12. Modify the “on button B pressed” block:

- Copy and paste the whole “if RCMODE = “B” then ...” block in the “on button A pressed” block and drag it to the “on button B pressed” block. Insert it at the very end.
- Delete the “set BedIsLevel to false” block at the beginning,
- Delete the “set MoveCargoBed to true” block at the beginning,
- Change the “play tone High B for 1/8 beat” at the beginning to “play tone High G# for 1/8 beat”.
- In the “if RCMODE = “B” then ...” slot, change the “set BedIsLevel to true” to “... false”,
- Do the same in the “else if RCMODE = “R” then ...” slot. (Leave both “set MoveCargoBed to true” blocks as they are.)
- In the “if ParameterCode = “Y” then ...” slot, add a new “change LegNumber by 1” block,
- Right after this new block, add a new “set ParameterCode to “t”” block,
- After this, add a new “set Message to join convert RoverNumber to text “L” convert LegNumber to text” block,
- And change the “radio send string ...” there to “radio send string join Message ParameterCode “?””.
- Add after this a new “set NeedParameter to true”,
- Add after this a new “set GotParameter to false”,
- Add after this a new “set AttemptsToReceive to 1”,
- And add after this a new “show string “?””.
- In the “else if ParameterCode = “t”” slot, change “-0.1” to positive “0.1” in the “change TimeOfLeg by ...” block,
- In the “else if ParameterCode = “S”” slot, change “-10” to positive “10” in the “change Speed by ...” block,
- In the “else if ParameterCode = “T”” slot, change “-10” to positive “10” in the “change Steer by ...” block,
- No changes are needed in the “else if ParameterCode = “U”” or “O”” slots.

13. Find the “on logo pressed” block:

- And move the “play tone High B for 1/8 beat” from the very end to the very beginning.
- About in the middle of the “on logo pressed” block, in the slot of the “else if ParameterCode = “t” then ...” block, add a new “set Message to join convert RoverNumber to text “L” convert LegNumber to text” block,
- And a “radio send string join Message ParameterCode convert $1000 + (10 \times \text{TimeOfLeg})$ to text” right after it.

Continued next page.



Time to Code: RC Rover 7 (continued)

14. Continue modifying the “on logo pressed” block:

- Add a new “if ... then ... else ...” block right after,
 - Replace the “true” in there with “TimeOfLeg > 0.01”,
- Move the “set ParameterCode to “S”” block inside the first empty slot of the new “if ... “ block,
- Place a new “set ParameterCode to “Y”” block into the empty “else ...” slot,
- Place a new “set LegNumber to 0” block right after (into the “else ...”)
- Place a new “call displayALogo” block right next.
- Copy and paste the “set Message to join ...” block from above and place it at the beginning of the “else if ParameterCode = “S” then ...” slot,
- Copy and paste the “radio send string ...” block from above and place it right after, change the “1000 + (10 x TimeOfLeg)” in the new “radio send ...” block to “200 + Speed”.
- Copy and paste the “set Message to join ...” block from above and place it at the beginning of the “else if ParameterCode = “T” then ...” slot,
- Copy and paste the “radio send string ...” block from above and place it right after, change the “200 + Speed” in the new “radio send ...” block to “400 + Steer”.
- Copy and paste the “set Message to join ...” block from above and place it at the beginning of the “else if ParameterCode = “U” then ...” slot,
- Add a new “if ... then ... else ...” block right after,
 - Replace the “true” in the “if ...” block with the “UnloadCargo” variable,
- Place a new “radio send string join Message ParameterCode “1”” block inside the first slot of the “if ...” block,
- And place another “radio send string join Message ParameterCode “0”” block inside the “else ...” block. (hint: copy, paste, place and edit the previous block!).
- Click the “+” sign at the end to create one more “else if ParameterCode = “O” then ...” block,
- Copy and paste the “set Message to join ...” block from above and place it at the beginning of the “else if ParameterCode = “O” then ...” slot,
- Copy and paste the “if UnloadCargo then ...” block from above, place it right after the new “set Message ...” block, and change the variable “UnloadCargo” to “SonarEnabled”.
- Add a new “change LegNumber by 1” block right after the “if SonarEnabled ...” block, (outside the “if ...” block!)
- Add a new “if ... then ... else ...” block next,
 - Replace the “true” in there with “LegNumber ≤ 5”,
 - And place a “set ParameterCode to “t”” in the first slot.
- Place a “set ParameterCode to “Y”” into the “else ...” slot,
- Add a “set LegNumber to 0” block right after,
- And add a “call DisplayALogo” next.

Continued next page.



Time to Code: RC Rover 7 (continued)

15. Continue modifying the “on logo pressed” block:

- At the end of the “on logo pressed” block, inside the “if LegNumber > 0 then ...” block, add a “set Message to join convert RoverNumber to text “L” convert LegNumber to text” block (same as above: copy and paste!).
- And change the “radio send string ...” block to “radio send string join Message ParameterCode “?””.
- Add a “set NeedParameter to true” block right after,
- Add a “set AttemptsToReceive to 1” block after the “set GotParameter to false” block,
- Leave the “show string “?”” block.

16. How to obtain course variables from the rover?

- Now we need to get the variables sent by the rover on the requests of the RC. To do that, we place a new “**on radio received receivedString**” block somewhere on our workspace.
- Add an “if parse to number substring of receivedString from 0 of length 2 = RoverNumber then ...” block.
- Add an “if length of receivedString > 3 then ...” block inside the just created “if ...” block: This ensures that short, fragmented strings are not considered.
- Add an “if LegNumber = parse to number substring of receivedString from 3 of length 1 then ...” block inside the previous “if ...” block. This ensures that only data of the current leg can be edited.
- Add a new “if ... then ... else ...” block,
- Click four times on the “+” sign at the end to create four “else if ...” slots.
- Get a new string comparison block, and place it in this last “true” place and edit to obtain an “if substring of receivedString from 4 of length 1 = “t” then ...”,
- Copy, paste, and place this comparison block in each of the empty “else if ...” hexagons,
- Edit the first one to “... = “S””,
- The second to “... = “T””,
- The third to “... = “U””,
- And the fourth to “... = “O””.
- Now we have to fill each of the slots, “t”, “S”, “T”, “U”, and “O” with the appropriate handling of the received string: For each we check if the correct ParameterCode is received (and don’t do anything if not!):
- Add a new “if ParameterCode = substring of receivedString from 4 of length 1 then ...”,
- Add a new “set TimeOfLeg to parse substring of receivedString from 5 of length 4” and place it inside the empty “if ...” slot,
- Add a new “set TimeOfLeg to 0.1 * (TimeOfLeg - 1000)” and place it next,
- Add a new “if TimeOfLeg ≥ 0 and TimeOfLeg < 10 then ... else ...” block and place it next,
- Add a new “set GotParameter to true” block and place it inside the new “if ...” block,
- Add a new “set NeedParameter to false” block and place it next,
- Place a “call displayTimeOfLeg” block next,

Continued next page.



Time to Code: RC Rover 7 (continued)

16. Continue with the “on radio received” block:

- Place a “play tone High F# for 1/2 beat” next,
- And place a “play tone Low F for 1/2 beat” into the empty “else ...” slot. We are done dealing with the received radio if the message is the TimeOfLeg. Next:
- Copy the whole “if ParameterCode = substring ...” block we just made, paste, and place it in the “else if substring ... = “S”” block,
- Change the TimeOfLeg variable in both “set ...” blocks to the Speed variable,
- Change the number 4 in the “set Speed to ... of length 4” block to the number 3,
- Change the final Speed calculation to “set Speed to Speed - 200”,
- In the next “if ...” block change both TimeOfLeg variables to Speed,
- Change the numbers in this block to “if Speed \geq -100 and Speed \leq 100 then ...”, note: Also change the inequality sign!
- Replace the “call displayTimeOfLeg” block with “call displaySpeed”. We are done with dealing with the Speed, next is Steer:
- Copy the whole “if ParameterCode = substring ...” block we just made, paste, and place it in the “else if substring ... = “T”” block,
- Change the Speed variable in both “set ...” blocks to the Steer variable,
- Change the final Steer calculation to “set Steer to Steer - 400”,
- In the next “if ...” block change both Speed variables to Steer,
- Replace the “call displaySpeed” block with “call displaySteer”. We are done with dealing with the Steer, next is Unload:
- Copy the whole “if ParameterCode = substring ...” block we just made, paste, and place it in the “else if substring ... = “U”” block,
- Add a new “if ... then ... else ...” block immediately before the first “set Steer ...” block,
- Place a new “0 = 0” comparison block over the “true” in this block,
- Drag the “parse to number ...” block from the “set Steer ...” block below over the first zero in the comparison block,
- Change the numbers 3 and 0 in the new “if ...” block such that it reads “if parse to number substring of receiveString from 5 of length 1 = 1 then ...”,
- Drag the “set Steer to 0” block from below into the first slot of the newly created “if ...” block, (it does not matter here that several blocks come with it - that’s okay!)
- Change the variable Steer to UnloadCargo,
- Place a “true” over the zero, such that you have now “set UnloadCargo to true”,
- Delete the “set Steer to Steer - 200” block,
- Drag the “set GotParameter to true (and everything attached to it) to outside the newly created “if ...” block: watch out: don’t drop it into the empty “else ...” slot! Drop it below the first “+” underneath that “else”!
- Delete the “if Steer -100 and ...” block,
- Copy the “set UnloadCargo to true” block, paste, and place the copy into the empty “else ...” slot,
- Change the “true” in this second “set Unload Cargo ...” block to “false”,
- Replace the “call displaySteer” block in here with “call displayCargo”.

Continued next page.



Time to Code: RC Rover 7 (continued)

17. Continue with the “**on radio received**” block:

- Copy the whole “if ParameterCode = substring ...” block we just made, paste, and place it in the “else if substring ... = “O”” block,
- Change both UnloadCargo variables in the “if parse ...” block to SonarEnabled,
- And replace the “call displayCargo” block with a “call displaySonar” block.

There is one more small thing left: We would like to exit course editing any time ... but there is no button left, which could be assigned to do this! Instead, we use the “on shake” block from the “Input” toolbox.

18. Place an “**on shake**” block somewhere on the workspace.

- Place a new “set ParameterCode to “Y”” block inside,
- Place a new “set LegNumber to 0” block next,
- Place a “call displayALogo” block next,
- Place a “NeedParameter to false” block next,
- Place a “GotParameter to false” block next,
- Place a “set Speed to 0” block next,
- Place a “set Steer to 0” block next,
- Place a “set Message to join convert RoverNumber to text “S” convert Speed to text” block next,
- Place a “set Message to join Message “T” convert Steer to text” block next,
- Place a “radio send string Message” block next.



Time to Code: Navigation Test 7

Now code must be written to teach or allow the rover to respond to the parameter request from the remote control.

1. We start with the rover's "**Navigation Test 6**" code.
2. Rename the project into "**Navigation Test 7**".
3. Modify the "**on start**" block:
 - Create a new variable "TimeOfLeg"
 - Place a "set TimeOfLeg to 0" block after the "set Message to """ block (right before the "set TimeToDrive ..." array) in the "on start" block,
 - Create another variable "UnloadCargoLeg" and place a "set UnloadCargoLeg to false" block right after the "set TimeOfLeg to 0" block,
 - Create another variable "SonarEnabledForLeg" and place a "set SonarEnabledForLeg to false" block next.
 - At the very end of the "on start" block, add five blocks for serial communication (these blocks can be used for debugging, but also help to control the timing for the radio communication and therefore are essential for correct function!):
 - Add a "serial redirect to USB" block from the "Serial" toolbox,
 - Add a "pause (ms) 500" block next,
 - Add a "serial set baud rate 115200" block from the "Serial" toolbox next,
 - Add another "pause (ms) 500" block next,
 - And add a "serial write line join "Rover RV" convert RoverNumber to text " online".
4. Modify the "**forever**" block:
 - At the beginning of the "**forever**" block, drag temporarily the "for index from 0 to 2 do unplot ..." block and all blocks attached to it out of the "if DrivingIsEnabled then ..." slot and place it on the workspace,
 - Create a function "displaySpeed" (in the "Function" toolbox),
 - And place the "call function displaySpeed" block into the (now) empty slot of the "if DrivingIsEnabled then ..." block.
 - Move the (still empty) "function displaySpeed" block to an empty area of the workspace (but don't move it far!),
 - And move the "for index from 0 to 2 ..." block we just placed there temporarily into the "function displaySpeed" block;

Continued next page.



Time to Code: Navigation Test 7 (continued)

5. Continue modifying the “forever” block:
 - Get a new “if ... then ...” block and place it at the beginning of the “else ...” slot of the “if DrivingAuto and LegNumber = 0 then ...” block, before the “if running time ...” block,
 - Place a “DrivingAuto” variable over the “true” in this new “if ...” block,
 - Drag the “if running time (ms) > TimeStampDot + 100 then ...” block with all that is attached to it into the empty slot.
 - Click on the first “+” sign after the green “for index from 0 to LegNumber - 1 ...” block (this is very close to the end of the “forever” block, but don’t confuse the green “for ...” block with the also green “while ...” blocks!) to create an “else ...” slot,
 - Copy the whole “if running time (ms) > TimeStampDot + 100 then ...” block from immediately before the “for index from 0 ...” block, paste, and place the copy into the empty “else ...” slot at the end of the “forever” block (the one we just created).
 - Inside the big green “for index4 from 0 to LegNumber – 1” block, far down, right before the “else ...” part of the “if running time (ms) < TimeStamp + ...” block, insert a “set HeadPosition to round(Steer / 1.4)” block,
 - And right after that place a “servo write pin P15 to HeadStraight + HeadPosition” block.
 - Place a “play tone High G for 1 beat” block after the “change BedAngle by 5” block (close to the end of the “forever” block, inside one of the green “while ...” blocks).
 - Make a copy of this block and place it after the “change BedAngle by -5” block in the other “while ...” block.
6. Modify the “on radio received” block:
 - Make another copy of the last block and place it after the “change BedAngle by -5” block in the “on radio received receivedString” block, (it is about in the middle of the block),
 - And place another copy of the “play tone High G ...” block after the “change BedAngle by 5” block in the “while ...” block above it.
 - Add a “clear screen” block immediately before the “set BedIsMoving to false” block below the two “while ...” blocks.
 - Almost at the very beginning of the “on radio received ...” block, right inside, at the beginning of the “if parse to number substring of receivedString from 0 of length 2 = RoverNumber then ...” add a “serial write line receivedString” block.
 - Almost at the end of the “on radio received ...” block, look for the “else if substring of receivedString from 2 of length 1 = “L” then ...” block,
 - Add a “set RVMode to “A”” block right at the beginning (inside!), before the “set LegNumber to parse ...” block,
 - Add a “set DrivingAuto to false” block right after the “set RVMode ...” block, (you can copy the block from above!)
 - Add a “set DrivingIsEnabled to false” block next.
 - Get a new “if ... then ... else ...” block from the Logic toolbox, and place it into the “if substring of receivedString from 4 of length 1 = “t” then ...” slot before the “set Message ...” block,
 - Replace the “true” in this block with a “substring of receivedString from 5 of length 1 = “?”” comparison block (you can copy from above and edit!),

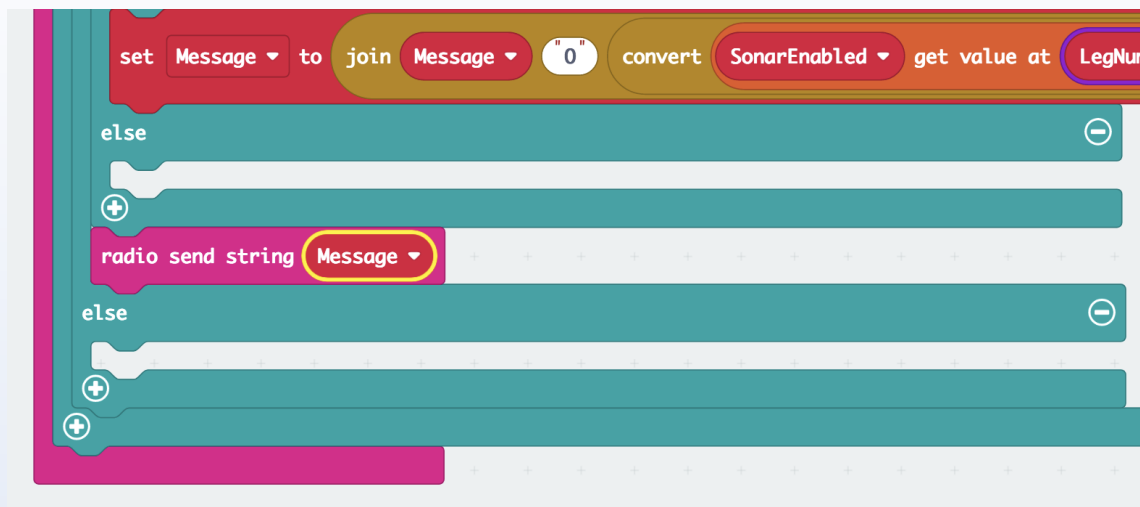
Continued next page.



Time to Code: Navigation Test 7 (continued)

7. Continue modifying the “on radio received” block:

- The “set Message ...” should have automatically snapped into the “if ... “ slot; if it didn’t, move it there.
- Place a “set TimeOfLeg to [0.1 * (parse to number substring of receivedString from 5 of length 4 - 1000)] in the empty “else ...” slot. Watch parentheses!
- Place a “TimeToDrive set value at LegNumber - 1 to TimeOfLeg” block (from the Arrays toolbox) next,
- And place a “play tone High B for 1/16 beat right after.
- Close to the end of the “on radio received ...” block, after the first empty “else ...” slot, place a “radio send string Message” block. Here is the precise location:



- Make a new function (there is space to the left on the workspace) and name it “displayTimeOfLeg”,
- Place a “clear screen” block in there,
- Create a new variable “DigitOfTime” and place a “set DigitOfTime to truncate TimeOfLeg” block next (the “truncate” comes from the “round ...” block from the “Math” toolbox),
- Place a new “for index from 0 to 4 ...” block next,
- And change the “index” variable to “indexColumn” (right-click the red “index”),
- And change the number 4 to 1.
- Get another new “for index from 0 to 4 ...” block and place it inside the previous “for ...” block,
- Change the “index” variable to “indexRow” (and do NOT change the 4!).

Continued next page.



Time to Code: Navigation Test 7 (continued)

8. Continue modifying the “on radio received” block:

- Place a new “if ... then ... else ...” block inside that last “for ...” block,
- Over the “true” there, place a comparison block “DigitOfTime > [indexRow + (5 * indexColumn)]. Watch the parentheses!
- Place a “plot x (4 - indexColumn) y (4 - indexRow)” block in the empty “if ...” slot,
- And place a “unplot x (4 - indexColumn) y (4 - indexRow)” block in the empty “else ...” slot (you can copy the two calculations!).
- Place a new “set DigitOfTime to round[10 * (TimeOfLeg - DigitOfTime)]” block after the two “for ...” blocks,
- Copy the two “for ...” blocks (and everything in them), paste and place the copy after the new “set DigitOfTime ...” block,
- Change the first number 4, the one in the (4 - indexColumn) calculation, to the number 1 in both, the “plot ...” and “unplot ...” blocks. Do not change the other 4!
- Go back to the “if substring of receivedString from 5 of length 1 = “?”” then slot in the “on radio received ...” block. Find the “play tone High B ...” block underneath. Place a “call displayTimeOfLeg” block (from “Functions” toolbox) right before the “play tone High B 1/16 beat” block.

We now added the capability of replacing the value of the TimeToDrive in one leg of the course with a new one as received from the RC. Before, the code only responded with a message containing the current value of the variable. Now, when a “?” is received from the RC, the message with the parameter is transmitted, but if there is not a “?”, then the current value of the parameter is replaced by the one transmitted from the RC. The rover also now displays the value of the received variable for verification. It should be the same as the one edited on the RC - - - but we don’t always know that the communication works. When we see it on the rover, then this is our confirmation it received the correct value. But, so far, we have only dealt with the TimeToDrive. We must do the same now for all the other parameters.

- Place a new “set Message to “”” block at the very end of the “on radio received ...” block.
- Get a new “if ... then ... else ...” block and place it immediately after the “else if substring of receivedString from 4 of length 1 = “S” then ...” block,
- Copy the “substring of receivedString from 5 of length 1 = “?”” comparison block from the “t” section and place it over the “true” here.
- Copy and paste this “if ... = “?”” then ... else ...” block three times and place them at the beginning of each of the following “else if ...” slots (before the “set Message ...” block).
- Move the “set Message to join Message “S” convert ...” block into the empty “if ... then ...” slot of the new “if ...” block,
- Place a new “SpeedNew to (parse to number substring of receivedString from 5 of length 3 - 200)” block into the empty “else ...” slot. Look before you create all the code new: you can copy some similar parts from above, and then edit. Just make sure you get all variable names and numbers correct!

Continued next page.



Time to Code: Navigation Test 7 (continued)

9. Continue modifying the “on radio received” block:

- Place a new “SpeedToDrive set value at LegNumber -1 to SpeedNew” block next,
- Place a new “call displaySpeed” block next,
- And add a “play ton High B for 1/16 beat” block after that.
- Copy the “play tone ...” block from above and place it next.
- Move the “set Message to join Message “T” convert ...” block into the empty “if ... then ...” slot of the “... ?” block inside the “else if ... “T”” block.
- Copy, paste and place the “set SpeedNew to parse ...” block from above into the empty “else ...” slot,
- Change “SpeedNew” to “Steer”, and change the number 200 to 400
- Copy, paste and place the “SpeedToDrive set value at ...” block into the empty “else ...” slot,
- Change the “SpeedToDrive” variable to “SteerToDrive”,
- Change the number 200 to 400,
- Create a new function “displaySteer”, place the function block itself out of the way for now, and place a “call displaySteer” block after the “SteerToDrive set value ...” block,
- And place a “play tone High B ...” next.
- Get a new “if ... then ... else ...” block and place it into the empty “if ... = “?” then ...” slot in the “else if ... = “U” then ...” block,
- Move the “UnloadCargo get value at LegNumber - 1” block from the “set Message ...” block below over the “true” in this new block,
- Move the “set Message to join Message “U” ...” block into the empty “if UnloadCargo ...” slot,
- Delete the “convert to text” in here and write the number “1” at the end of the block. See how it should look like:

```
else if substring of receivedString from 4 of length 1 = "U" then
  if substring of receivedString from 5 of length 1 = "?" then
    if UnloadCargo get value at LegNumber - 1 then
      set Message to join Message "U"
    else
  
```

Continued next page.



Time to Code: Navigation Test 7 (continued)

10. Continue modifying the “on radio received” block:

- Copy the “set Message ...” block you just edited, paste, and place it into the empty “else ...” slot of the “if ...” block. Change the number “1” to a zero.
- Place a new “if ... then ... else ...” block into the empty “else ...” slot,
- Copy the “substring of receivedString from 5 of length 1 = “?”” comparison block and place it over the “true” of the new “if ...” block.
- Change the “?” to “1” (the string of the number one!).
- Create a new “set UnloadCargoLeg to true” block and place it into the “if substring of receivedString from 5 of length 1 = “1” then ...” slot,
- Copy this “set UnloadCargoLeg ...” block, paste and place it into the empty “else ...” slot,
- And change the “true” in here to “false”.
- Copy the “SteerToDrive set value ...” block from above and place it after the “if substring ... = “1” then ... else ...” block,
- Change the variable “SteerToDrive” here to “UnloadCargo”,
- And change the “Steer” variable at the end to “UnloadCargoLeg”.
- Create a new function “displayUnloadCargo”, place the “function ...” block out of the way for now,
- And place a “call displayUnloadCargo” block after the “UnloadCargo set value at ...” block,
- And place a “play tone High B ...” block next.
- Copy the whole “if substring of receivedString from 5 of length 1 = “?” then ...” block (the one under the “else if ... = “U” then ...”, paste and place the copy before the “set Message to join Message “O” ...” block below.
- Change the “UnloadCargo” variable in the “if UnloadCargo get value ...” block to “SonarEnabled”,
- Change the “U” in both “set Message to ...” blocks below to an “O”,
- Change the “UnloadCargoLeg” variable below, in both “set UnloadCargoLeg to ...” blocks in the “if substring ... = “1” then ... else ...” block to the “SonarEnabledForLeg” variable,
- Change the “UnloadCargo” variable in the “UnloadCargo set value ...” block to “SonarEnabled”,
- And change the “UnloadCargoLeg” variable there to “SonarEnabledForLeg”.
- Delete the “call displayUnloadCargo” block,
- Make a new function “displaySonarEnabled”, move the “function ...” block out of the way for now,
- And place a “call displaySonarEnabled” block before the “play tone High B ...” block.
- Delete the “set Message to join Message “O” convert SonarEnabled ...” block.

Whoow! Almost done. But we must fill the empty functions we created to get here with action!

Continued next page.



Time to Code: Navigation Test 7 (continued)

11. Find the empty "function displaySteer" block:

- Place a "clear screen" block into the function slot,
- Place a "pause (ms) 100" block next,
- Place a "plot x 2 y 4" block next,
- Create an "if absolute of Steer > 90 then ..." block, and place a "plot x 2 y 2" block inside.
- Copy this "if ..." block twice and place both of them next,
- Edit the first (of the two copies - that is the second "if ..." block of all in this function (yet) to replace the number 90 with 80, and the "plot x 2 y 2" by "plot x 2 y 1",
- Edit the other copy to replace the number 90 with 70, and the "plot x 2 y 2" by "plot x 2 y 0".
- Make a new "if Steer < -60 then ..." block and place a "plot x 3 y 0" block inside,
- Copy and paste this block seven times,
- Place one copy after the other at the end of the "function displaySteer" block,
- And edit the first copy replacing -60 with -50, and "plot x 3 y 0" with "plot x 4 y 0",
- Edit the second copy replacing -60 with -40, and "plot x 3 y 0" with "plot x 4 y 1",
- Edit the third copy replacing -60 with -30, and "plot x 3 y 0" with "plot x 4 y 2",
- Edit the fourth copy replacing -60 with -20, and "plot x 3 y 0" with "plot x 4 y 3",
- Edit the fifth copy replacing -60 with -10, and "plot x 3 y 0" with "plot x 4 y 4",
- And edit the sixth copy replacing -60 with 0, and "plot x 3 y 0" with "plot x 3 y 4".
- Edit the seventh copy changing the "<" (less than) sign to a ">" (greater than) sign, replacing the number -60 with positive 60, and replacing "plot x 3 y 4" with "plot x 1 y 0",
- Copy and paste this last block six times,
- Place one copy after the other at the end of the "function ..." block,
- And edit the first copy replacing 60 with 50, and "plot x 1 y 0" with "plot x 0 y 0",
- Edit the second copy replacing 60 with 40, and "plot x 1 y 0" with "plot x 0 y 1",
- Edit the third copy replacing 60 with 30, and "plot x 1 y 0" with "plot x 0 y 2",
- Edit the fourth copy replacing 60 with 20, and "plot x 1 y 0" with "plot x 0 y 3",
- Edit the fifth copy replacing 60 with 10, and "plot x 1 y 0" with "plot x 0 y 4",
- And edit the sixth copy replacing 60 with 0, and "plot x 1 y 0" with "plot x 1 y 4".

12. Find the empty "function displayUnloadCargo" block:

- Place a "clear screen" block inside,
- Create a new "if UnloadCargoLeg then ... else ..." block,
- Place a new "show leds" (from the "Basic" toolbox) block,
- And click the LEDs on until you see and upside-down capital "U". Remember, we look at the microbit of the rover most of the time from the rear, which means we see the microbit upside-down!)
- Copy this "show leds upside-down U" block, paste and place into the "else ..." slot,
- Edit the LEDs until you see an upside-down lowercase "u".

Continued next page.



Time to Code: Navigation Test 7 (continued)

13. Find the empty “function displaySonarEnabled” block:

- Place a “clear screen” block inside,
- And copy the “if UnloadCargoLeg then ... else ...” block from the “function displayUnloadCargo” block and place the copy in here.
- Change the “UnloadCargoLeg” variable to “SonarEnabledForLeg”,
- And edit both, the “U” and “u” for an “O” and “o” ... all upside-down.

14. This code is too big to load on the microbit, we need to reduce its size:

- Delete in the “on logo pressed” block the “if Speed < -100 or Speed > 100 then ... else ...” block.

Here are the links to the final code:

“Rover RC 7”: https://makecode.microbit.org/_WcXDXW8Vg9ks

“Navigation Test 7”: https://makecode.microbit.org/_1TyC4Kd01c1s



Try this: Navigation Test 7

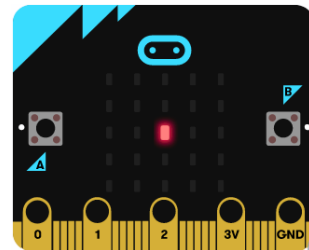
1. Upload the RC Rover 7 & the Navigation 7 code to the correct micro:bits and test:
 - **Turn the rover on:**
 - It giggles, the bed levels, the sonar head scans left to right, then centers, and the center dot (LED) blinks.
 - **Turn the RC on:**
 - The RC plays a scale up and down, it displays the “Bed logo”, and the “top” LED blinks.
 - **Push button A on the RC:**
 - The cargo bed rises to the unload position,
 - it beeps at every step,
 - the rover displays a large “X” with a blinking center dot,
 - when done there is only the blinking dot.
 - **Push button B on the RC:**
 - the cargo bed lowers into the level position,
 - it beeps at every step,
 - the rover displays a large “X” with a blinking center dot,
 - when done there is only the blinking dot.
 - **Push the logo button on the RC:**
 - the RC displays the “RC logo” with a blinking dot at the top of the antenna,
 - the rover is ready to drive,
 - displays Speed with sign,
 - the head turns in driving direction,
 - the cargo bed is somewhat nervous at low speeds.
 - The rover drives forward and in reverse.
 - **Push the logo button on the RC:**
 - the RC displays the “Autonomous logo” with a blinking dot at the base,
 - **Push button A on the RC:**
 - the rover drives the default course,
 - and briefly displays the leg number for each leg,
 - pushing button A again repeats the same course.
 - **Push button B on the RC:**
 - the RC enters course edit mode and displays t, S, T, U, O in sequence for each leg,
 - with each push of the logo button, the RC transfers the current parameter to the rover,
 - the rover displays the same current parameter,
 - while the RC offers to edit the next parameter.
 - Pushing either button A or B increases or decreases the current parameter,
 - only with the logo button, the parameter is sent to the rover,
 - and also displayed by the rover.
 - **Shake the RC:**
 - The RC quits edit mode and displays the “Autonomous logo”,
 - here there is the choice again between driving the course or editing it.

Final Code TEST:

1. Open **Rover RC 7** code: https://makecode.microbit.org/_WcXDXW8Vg9ks
2. Change “set RoverNumber to 44” in the “on start” block to your rover’s number
3. Download the code to your remote control.
4. Open **Navigation Test 7** code: https://makecode.microbit.org/_1TyC4Kdo1c1s
5. Change “set RoverNumber to 44” in the “on start” block to your rover’s number
6. Change “set HeadStraightAt to 84” in the “on start” block to your rover’s adjusted number
7. Change “set BedLevel to 105” in the “on start” block to your rover’s adjusted angle

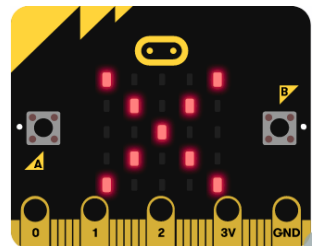
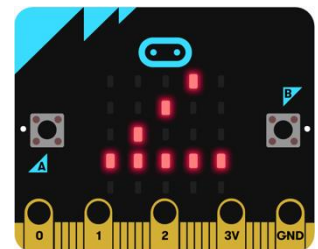
8. Turn the rover ON

9. Does the sonar head move?
10. Is the cargo bed level?
11. Does it show the blinking center dot?
12. Push button A (on the rover)
13. Does the rover drive the (default) course?



14. Turn the RC ON

15. Does it show the “Bed” icon (cargo bed level and unloading) with the blinking top dop?
16. Press button A: Does the cargo bed move?
17. Press button B: Does the bed go back to level position?
18. Adjust the “set BedLevel to ...” angle if needed



19. Press the touch logo (on the RC): does the rover come “alive”?

20. Does it display the “RC” icon (antenna) with the blinking dot front center?

21. Drive the rover using the RC: tilting it forward causes the rover to move forward, leveling the RC stops it, tilting it backward drives it reverse, tilting left and right steers. Make sure you hold the RC with both hands and with the connector side facing you.

22. Stop the rover (hold RC level).

23. Press the touch logo again:

24. Does it display the “Autonomous” icon (arrow-with-baseline) with the blinking dot rear center?

25. Press button A for “Go!”

26. Does it drive the default course? (Leg 1: go straight at Speed = 70, Leg 2: spin right at Speed = 50 and unload, Leg 3: go straight at Speed = 100, Leg 4: turn left (Steer = -30) at Speed = 60, Leg 5: go straight at Speed = 80.)

27. Place an obstacle about 3 feet in front of the rover.

28. Press button A on the RC (still in Autonomous mode, for “Go!”)

29. Does it stop in front of the obstacle, spin around and unload right there?

30. Press button B on the RC (still in Autonomous mode, for “Edit Course”)

31. Does the RC double beep (ti-ra)?

32. Does the RC display the time for leg 1 (should be 6 sec)?

33. Increase the time to 8 sec (using button A for down, B for up).

34. Press the logo button.

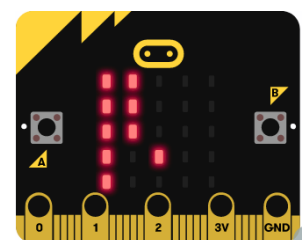
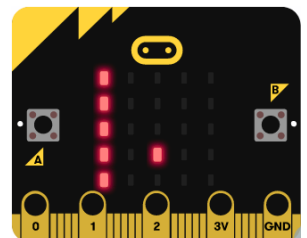
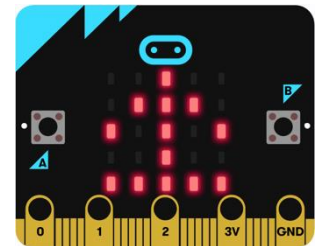
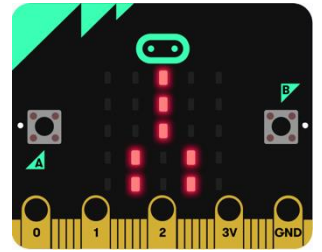
35. Does the RC display the speed for leg 1 (should be 70)?

36. Change the speed to 100 (using button A for down, B for up).

37. Press the logo button.

38. Does the RC display the steer variable for leg 1 (should be 0)?

39. Change the steer variable to +50 (A is right, B is left).

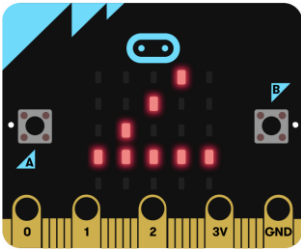


40. Press the logo button.
41. Does the RC display unload settings (should be lowercase “u” for don’t, would be uppercase “U” for unload after completing the driving for this leg).
42. Press the logo button.
43. Does the RC display obstacle avoidance settings (should be uppercase “O” for stop leg before obstacle, would be lowercase “o” for ignore the sonar).
44. Press the reset button of the RC: the rover will remember the edited course.
45. Press the logo button until the RC is in A mode.
46. Press the A button to send the rover on its course.
47. Does it drive the new course?

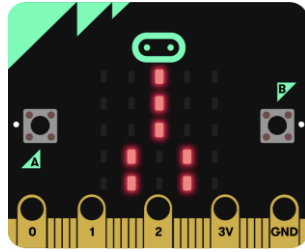
Summary of Rover (RC) Functions:

- The **logo “button”** toggles between 3 modes:
 - Bed, “B” mode,
 - RC, “R” mode, and
 - Autonomous, “A” mode.
- In B and R modes, the A button “unloads” (rises the cargo bed), the B button lowers it back to level,
- In A mode,
 - the A button (at first) means “Go!”
 - the B button (at first) means “Edit Course”.
 - When being in “Edit Course” mode,
 - Time, (duration of the current leg in seconds)
 - Speed, (-100 ... +100 in increments of 10)
 - Steer, (-100 ... +100 in increments of 10)
 - Unload, and (raising the bed after the motion, and then leveling it)
 - Sonar (enable/disable sonar stop during the motion)
 - for each leg can be determined.
 - the logo “button” cycles through the variables and legs
 - when pressing the logo, the RC beeps first, then the rover with the same pitch.
 - expect (and wait for!) two low beeps from the RC: then edit!!!
 - a leg with zero time is not executed.

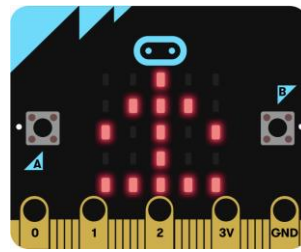
- **Rover Modes:**



“Bed” mode icon

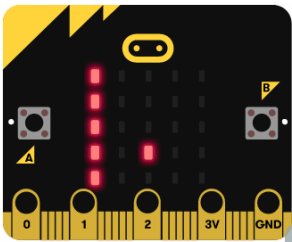


“RC” mode icon

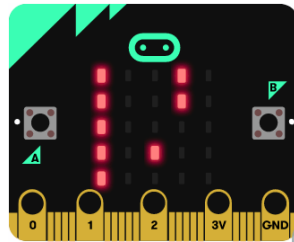


“Autonomous” mode icon

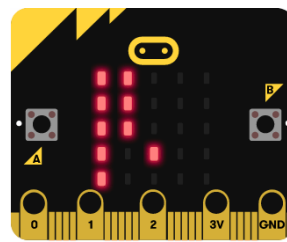
- **Time Display:**



5 sec



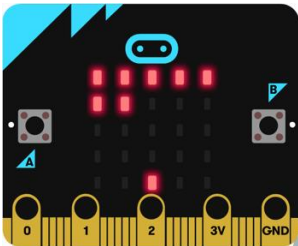
5.2 sec



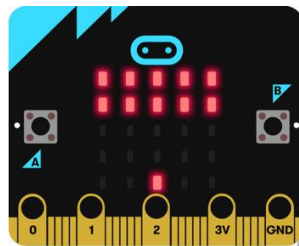
8 sec

The blinking dot below the center acts as a decimal point between full seconds on the left and tenths of seconds to the right.

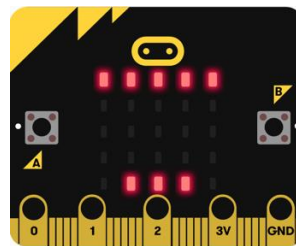
- **Speed Display:**



Speed = 70



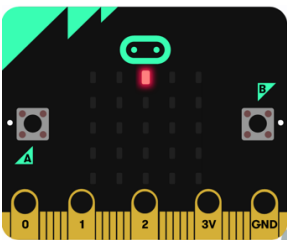
Speed = 100



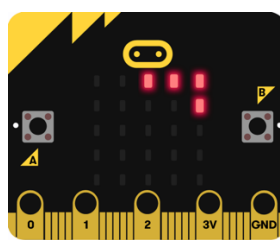
Speed = -50

The solid dot at the bottom means “+”; three dots mean “-“ (reverse)

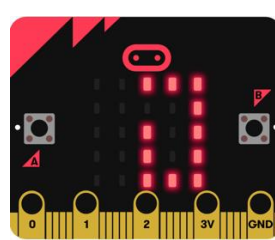
- **Steer Display:**



Steer = 0 (straight)

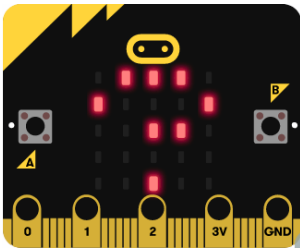


Steer = 30 (turn right)

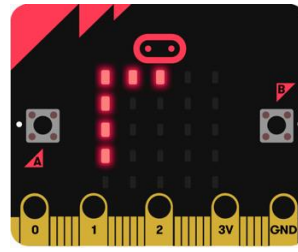


Steer = 100 (spin right)

- **Error Display:**



waiting for rover to respond (is it ON?)



Steer = -50 (turn left)

Practice Course

	Time	Speed	Steer	Unload	Sonar
Leg 1					
Leg 2					
Leg 3					
Leg 4					
Leg 5					

Default Course

	Time	Speed	Steer	Unload	Sonar
Leg 1	6.0 s	70	0	no	yes
Leg 2	2.2 s	50	100	yes	no
Leg 3	2 s	100	0	no	no
Leg 4	3 s	60	-30	no	no
Leg 5	2 s	80	0	no	no

Code Summary so far:

- "**Blink Timer**" code: https://makecode.microbit.org/_KFPTALMYDa10
- "**Count Timer**" code: https://makecode.microbit.org/_6XCXtm4ibKyu
- SS Template 6:** https://makecode.microbit.org/_FMaPxv34CFqs
- "**Propulsion Test 1**": https://makecode.microbit.org/_L7HU4b1JaeDC
drive rover wheels with analog write pin
- "**Propulsion Test 2**": https://makecode.microbit.org/_DhmbvsdXJJVT
drive rover wheels with StemSeals' Drive Wheels
- "**Propulsion Test 3**": https://makecode.microbit.org/_LV3LoeHmuFbH
change speed with rover's A (up) and B (down) buttons
- "**Propulsion Test 4**": https://makecode.microbit.org/_gU1TwpRcTJkY
display reverse speeds as well
- "**Propulsion Test 5**": https://makecode.microbit.org/_660Wkx9bK9xf
remote controlled rover, using
- "**Rover RC 1**": https://makecode.microbit.org/_coufkYUTt8R3
speed control only
- "**Propulsion Test 6**": https://makecode.microbit.org/_6C0V0U3cjDuX
remote controlled speed and steer, using:
- "**Rover RC 2**": https://makecode.microbit.org/_APtHmRYAEd3a
RC for speed and steer
- "**Cargo Bed Test 1**": https://makecode.microbit.org/_RJmDakHWqh03
for bed alignment (BedLevel and BedTilt angles), using:
- "**Rover RC 3**": https://makecode.microbit.org/_dcFhtJ4h0fW0
helps to quickly align the cargo bed parameters
- "**Cargo Bed Test 2**": https://makecode.microbit.org/_XFh2qkiD6gfM
for RC driving the rover and controlling unloading and level cargo bed positions,
using:
- "**Rover RC 4**": https://makecode.microbit.org/_8mqCk6V1Lgw7
RC driving the rover and controlling the cargo bed
- "**Sonar Test 1**": https://makecode.microbit.org/_Lta7U9Y9seEz
sonar head sweeps, then centers on Reset
- "**Sonar Test 2**": https://makecode.microbit.org/_F2UTLsUc2UVW

sonar head "looks" in driving direction

"**Rover RC 5**" code: <https://makecode.microbit.org/ DMzFozEx1hqs>

fewer missed radio commands and ability to enable/disable driving with RC

"**Sonar Test 3**" code: <https://makecode.microbit.org/ 4CiAx0YE9L0q>

works with "Rover RC 5", improved functionality

"**Sonar Test 4**" : <https://makecode.microbit.org/ HC4aaM53DDkc>

works with "Rover RC 5", reads first the sonar, but can drive and unload

"**Navigation Test 1**" code: <https://makecode.microbit.org/ d7aXuvF4fMjP>

rover drives for 5 sec but stops if encountering an obstacle

"**Navigation Test 2**": <https://makecode.microbit.org/ frUDhjMvFecj>

rover spins after driving straight to an obstacle and stopping

"**Navigation Test 3**": <https://makecode.microbit.org/ cqAhxRbsdD8T>

code simplification for autonomous 2-leg course, works with "Rover RC 5"

"**Navigation Test 4**": <https://makecode.microbit.org/ FR9PL3HouU1X>

3-legged course, including unloading cargo after the spin

"**Navigation Test 5**": <https://makecode.microbit.org/ HgwMviDpEb5w>

easy expansion to a 5-legged course, works with "Rover RC 5"

"**Rover RC 6**": <https://makecode.microbit.org/ 4zK2jgAEw2tD>

RC cycles through B --> R --> A --> B modes

"**Navigation Test 6**": <https://makecode.microbit.org/ b382w4bKqYF3>

"**Rover RC 7**": <https://makecode.microbit.org/ WcXDXW8Vg9ks>

"**Navigation Test 7**": <https://makecode.microbit.org/ 1TyC4Kdo1c1s>

Module 10: Competition Practice

Practice Missions

(Events may change. A complete list of events and rules will be handed out and practiced.)

1. From Lab to Hospital and Back (Remote Controlled)

- Time (Lowest time)
- Accuracy (points in seconds added for hitting obstacles or going out of bounds)
- Free style (you may follow the rover)

2. From Lab to Hospital and Back (Remote Controlled)

- Time (Lowest time)
- Accuracy (points in seconds added for hitting obstacles or going out of bounds)
- You must stay in the designated box for navigation.

3. From Lab to Hospital, Unload and Back (Remote Controlled)

- Time (Lowest time)
- Accuracy (points in seconds subtracted for target zone)

4. Complete an Autonomous 5-leg Course (Open Square)

- Time (Lowest time)
- Accuracy (points in seconds subtracted for target zone)

5. From Lab to Hospital, Unload and Back (Autonomous) w/blood test

- Time (Lowest time)
- Accuracy (points in seconds subtracted for target zone)

6. Relay Race over the Bridge (Remote controlled)

- Relay race for teams of 4
- Time (Lowest time)

Module 11: The Competition

Today is the time to put all your efforts and hard-earned skills to use. Use this space to compile any last reminders to help you during the competition phase of the camp. You will receive an extra packet with all the competition events, rules, and judging criteria.

Rover Specifics:

Rover # _____

Head Straight at: _____

Bed Cargo Level at: _____

References:

- *BBC Micro:Bit Overview*. www.microbit.org/get-started/user-guide/overview.
- Micro:bit Educational Foundation. “Introduction to the BBC Micro:Bit.” *YouTube*, 19 Jan. 2021, www.youtube.com/watch?v=u2u7UJSRuko&feature=youtu.be.
- Electronics, SparkFun. “Getting Started With Micro:Bit Part 1: Say Hello.” *YouTube*, 10 Apr. 2017, www.youtube.com/watch?v=kaNtg1HGxbY&feature=youtu.be.
- Hymel, Shawn. “Micro:Bit Tutorial Series Part 1: Getting Started.” *YouTube*, 14 Aug. 2016, www.youtube.com/watch?v=ZIW_6rxYNBg&feature=youtu.be.
- Micro:bit Educational Foundation. “Input and Output Devices.” *YouTube*, 19 Jan. 2021, www.youtube.com/watch?v=NkoS2JXaBuM&feature=youtu.be.
- ---. “Micro:Bit Processor.” *YouTube*, 28 Feb. 2020, www.youtube.com/watch?v=Y9tk07CzTAA&feature=youtu.be.
- *Set up Your Micro:Bit*. microbit.org/get-started/first-steps/set-up.
- “Introduction.” *Microsoft MakeCode*, makecode.microbit.org/courses/csintro/coordinates/overview.
- ---. “Micro:Bit LEDs.” *YouTube*, 28 Feb. 2020, www.youtube.com/watch?v=eRhlaXqT-0w&feature=youtu.be.
- Aethon. “TUG Autonomous Mobile Robots for Healthcare and Hospitality.” *Aethon*, 14 Mar. 2023, aethon.com/products.
- Swisslog Healthcare. “Swisslog Healthcare.” *Swisslog Healthcare*, www.swisslog-healthcare.com/en-gb/products/transport/relay#.
- Dr Matt & Dr Mike. “ABO Blood Types Made Easy!” *YouTube*, 29 Nov. 2021, www.youtube.com/watch?v=Amn2EWTY2Lk.
- *Aldon - Innovating Science® - Simulated ABO Blood Typing Kit*. www.alдон-chem.com/product_simulated-abo-blood-typing-kit.php.
- Parveen, Shah Nawaz. “Create a Variable in Makecode Microbit.” *YouTube*, 22 Apr. 2020, www.youtube.com/watch?v=vD4ywGNsZTA&feature=youtu.be.
- *ELECFREAKS Micro:Bit Motor:Bit*. www.electfreaks.com/motor-bit-for-micro-bit-motorbit.html.
- Micro:bit Educational Foundation. “Micro:Bit Radio.” *YouTube*, 28 Feb. 2020, www.youtube.com/watch?v=rwymAr6WqrQ.

- Microsoft MakeCode. “Behind the MakeCode Hardware - Radio in Micro:Bit.” *YouTube*, 11 Mar. 2019, www.youtube.com/watch?v=Re3H2ISfQE8.
- Microsoft MakeCode. “Behind the MakeCode Hardware - Accelerometer on Micro:Bit.” *YouTube*, 1 Dec. 2018, www.youtube.com/watch?v=byngcwjO51U.
- *Accelerometer* — *UCL BBC Micro:Bit Tutorial*. microbit-challenges.readthedocs.io/en/latest/tutorials/accelerometer.html.
- *Servos Explained - SparkFun Electronics*. www.sparkfun.com/servos.
- The Engineering Mindset. “What Is a Servo Motor and What Does It Do?” *YouTube*, 12 Dec. 2022, www.youtube.com/watch?v=tHOH-bYjR4k.
- “Types of Sensors in Robotics.” *Wevolver*, 13 Jan. 2023, www.wevolver.com/article/sensors-in-robotics-the-common-types.
- element14 presents. “How Do Ultrasonic Distance Sensors Work? - the Learning Circuit.” *YouTube*, 27 Jan. 2021, www.youtube.com/watch?v=2ojWO1QNprw.