



STEM SEALS

SEA Lectionary



A Student Guide to the SEA Challenge

STEM SEALs
SEA Lectionary

A Student Guide to the SEA Challenge

Copyright © 2021 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 G. Maresch and Lura L. Murfee

ISBN: 979-8-9874652-0-2

Printed by North Florida College

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

Synopsis

This document will guide you through the STEM SEALs SEA challenge. You will build a boat capable of navigating in the water autonomously and by remote control.

Your challenge is to assemble the boat, test it and improve it where you can, and then use it to collect a water quality sample and analyze it.

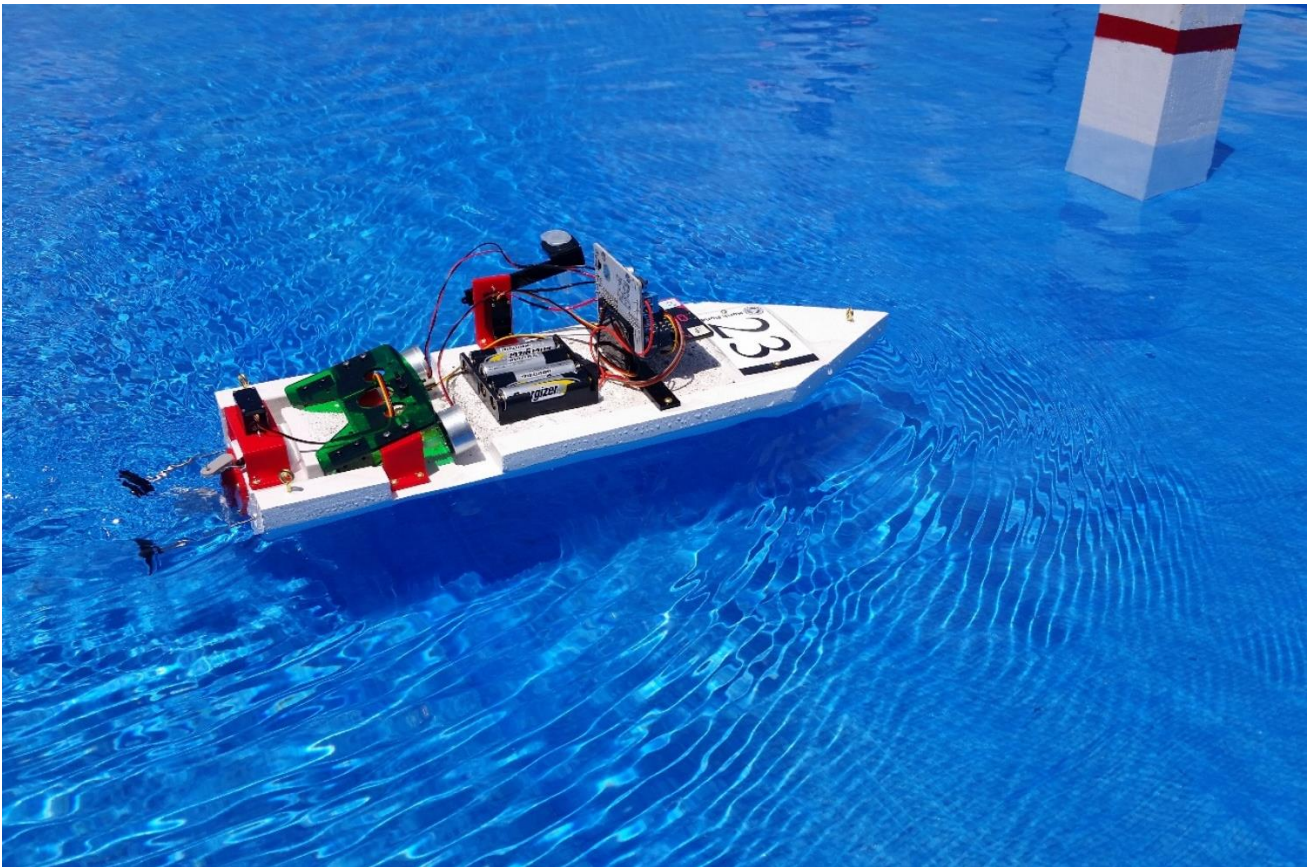


Table of Contents

Preface – The STEM SEALs Project.....	10
Module 1: The SEA Challenge.....	13
1.1 What is the SEA Challenge?.....	13
1.2 Unpacking the SEA Challenge Kit.....	14
Module 2: What Does a Microcomputer Do?.....	27
2.1 Exploring the Micro:bit	27
2.2 Accessing and Using the MakeCode Editor	28
2.3 Writing Code	30
2.4 Sharing or Publishing Your Code.....	34
2.5 Variables	34
Module 3: Micro:bit Code	35
3.1 How Does Code Work?.....	35
3.2 Interruptions in Code	35
3.3 The Essential MakeCode Blocks	43
3.4 Advanced MakeCode Blocks	43
3.5 MakeCode Java Script	44
Module 4: Testing Water Quality	45
4.1 Why are we testing water samples?.....	45
4.2 What are the things we will we be testing in our water samples?	45
4.3 Testing for nitrates, nitrites, and sulfates.....	46
4.4 Testing for pH (Hydrogen Ion).....	49
Module 5: Floatation Test.....	52
5.1 Mass and Volume	52
5.2 Floatation of the Boat Hull	53
5.3 Buoyancy	54
5.4 Load and Balance	54
5.5 Center of Gravity	59
Module 6: Boat Assembly	61
6.1 Twin Motor Assembly.....	61
6.2 Motor Test	62
6.3 Assembly of the Motor:bit Mount with Micro:bit.....	63
Module 7: Boat Propulsion.....	65

7.1 Propulsion.....	65
7.2 Propulsion 2.....	67
7.3 Adding a Remote Control (Propulsion 3 / Remote Control 1 & 2).....	69
7.4 Measuring Propulsion (Thrust).....	76
7.5 Reversing the Engines (Propulsion 4).....	78
7.6 Stop Engines (Propulsion 5 /Remote Control 3).....	85
7.7 Remote Control 4.....	88
Module 8: Rudder Assembly and Test.....	92
8.1 Attaching the Rudder Servo and the Rudder.....	92
8.2 Code for the Rudder Servo Motor.....	94
Module 9: Remote-Controlled Boat with Rudder.....	98
9.1 Remote Control with Rudder.....	98
9.2 Remote Control with Speed and Rudder.....	101
9.3 Remote Controlled Boat with Reversible Engines and Rudder.....	105
Module 10: Steering Test with Remote Control.....	109
10.1 Mobile Pond Practice Notes.....	109
10.2 Lake Osceola Practice Notes.....	109
Module 11: Water Sampling Assembly and Code.....	110
11.1 Assembling the Water Sampler.....	110
11.2 Water Sampler Code.....	111
11.3 Automatic Water Sampling.....	118
11.4 Completing the Boat.....	126
11.5 Function Summary: Remote Controlled Boat.....	126
Module 12: Floatation Test of the Finished Boat.....	127
12.1 Weigh the Completed Boat.....	127
12.2 Center of Gravity.....	127
12.3 Buoyancy.....	127
Module 13: Autonomous Navigation.....	128
13.1 The Process of Navigation.....	128
13.2 Planning a Course.....	131
13.3 Using the Micro:bit to Set a Course.....	133
13.4 How does the SEA Navigation Program work?.....	137
13.5 Navigating Lake Osceola.....	139

Module 14: Navigation Practice	140
Mobile Pond Events:.....	140
A) Buoy Race (one buoy):.....	140
B) Figure Eight:.....	140
C) Bumper Boat Rally:.....	140
Lake Osceola Events:	140
A) Buoy Race (two buoy):.....	140
B) Water Sample Grand Finale:	141
Module 15: The Competition.....	142
Resources:.....	143

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.

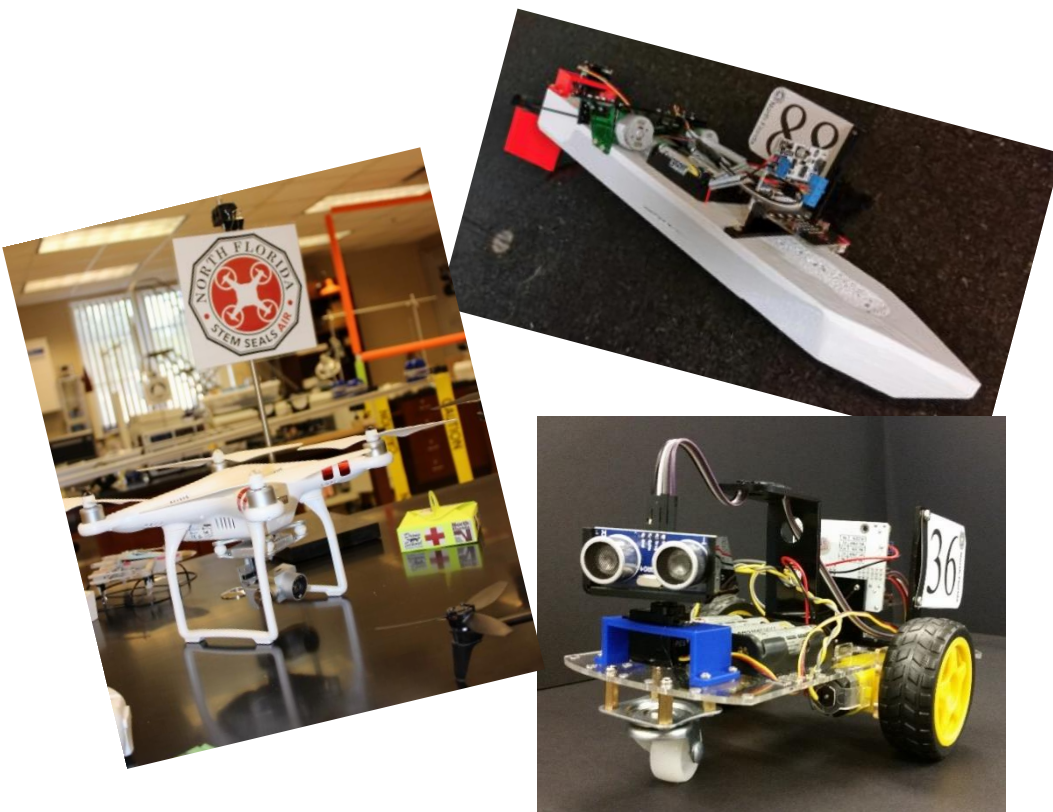
For each of the three disciplines, SEA, AIR and LAND two documents provide instructions. The first, the “Lectonary” is the student guide; it is the textbook needed by each student, containing modules describing each activity but also providing plans and images for clarification and troubleshooting when needed. A second document, the “Dictionary” was used as a guide for facilitators, i.e., 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. The Dictionary has not been published, yet.

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 the STEM SEALs team, including the researchers at Cynosure Consulting, and the evaluators at Indikus Evaluation and Planning, gratefully acknowledge funding by the National Science Foundation and its Discovery Research PreK-12 (DRK12) program. Also, we thank Tri-County Electric Cooperative of Madison, Florida for their support. We are grateful to the Maintenance Department, Duplication, College Advancement, Computer Services and to the leadership of North Florida College for their help, their insightful contributions, their participation, and for being available and creative when in need. However, the most important people for the successful deployment of the STEM SEALs challenges are the teachers from the schools in our area. This project would not be possible without active participation and the guiding contributions of those educators.

Dr. Guenter Maresch
Professor for Physics and Astronomy



Module 1: The SEA Challenge

1.1 What is the SEA Challenge?

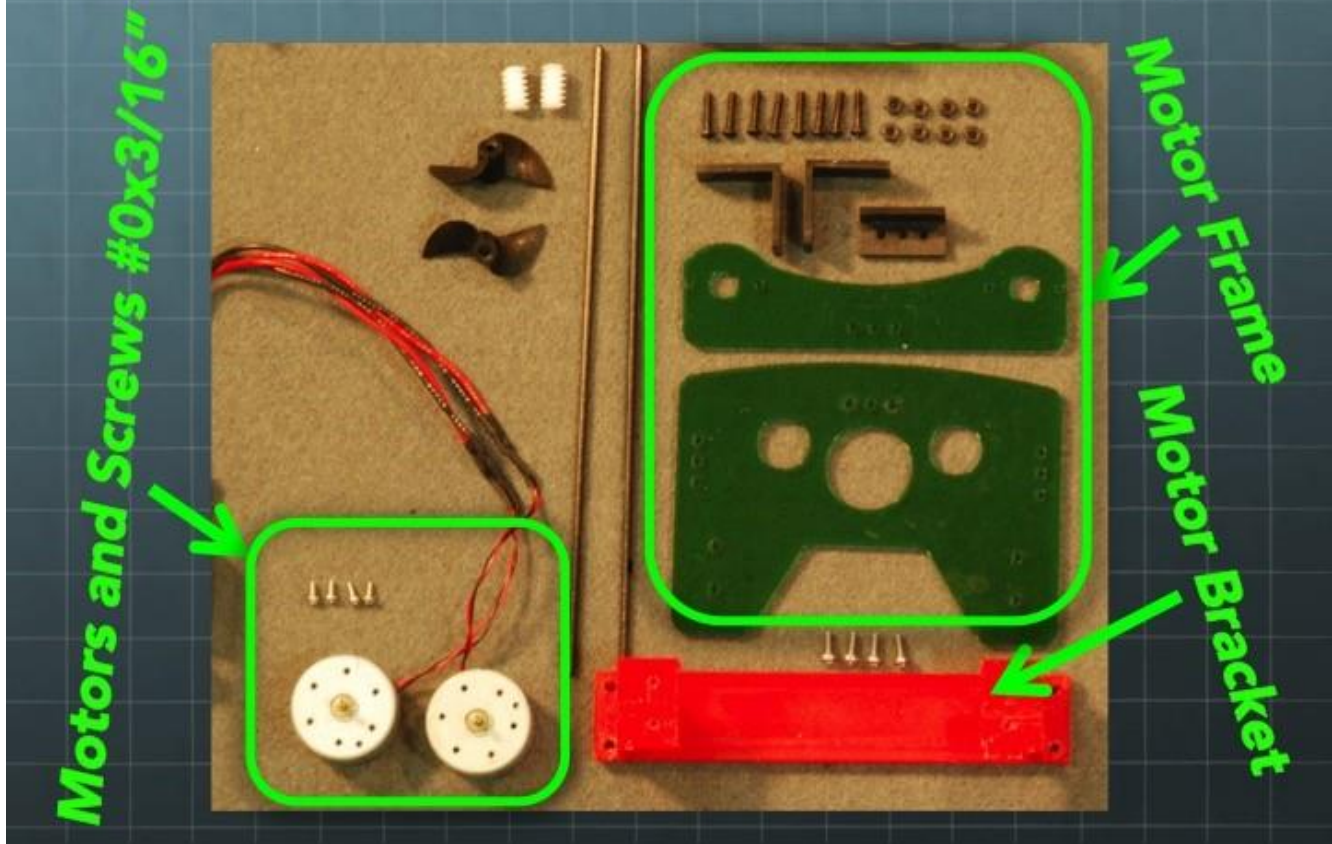
What is the goal for the SEA Challenge?

The STEM SEALs project is about remotely controllable vehicles, and the SEA challenge allows students to build a robotic boat that is capable of autonomous navigation and being controlled by a remote-control device.

You will explore design features of boats, the component parts to create a robotic boat, and the principles of how boats operate. You will apply engineering basics in the building of the boat, develop your physical, chemical, and mathematic principals through learning about the buoyancy, propulsion, steering, and the energy/power source of the boat. Using a microcomputer, you will learn how to code your device to produce desired outcomes. Lastly, you will compile all of these skills to create a robotic boat that will be able to complete a given course on an open body of water (Lake Osceola) to acquire a water sample of the lake and test it for particular analytes.



Twin Motor-Propeller Assembly

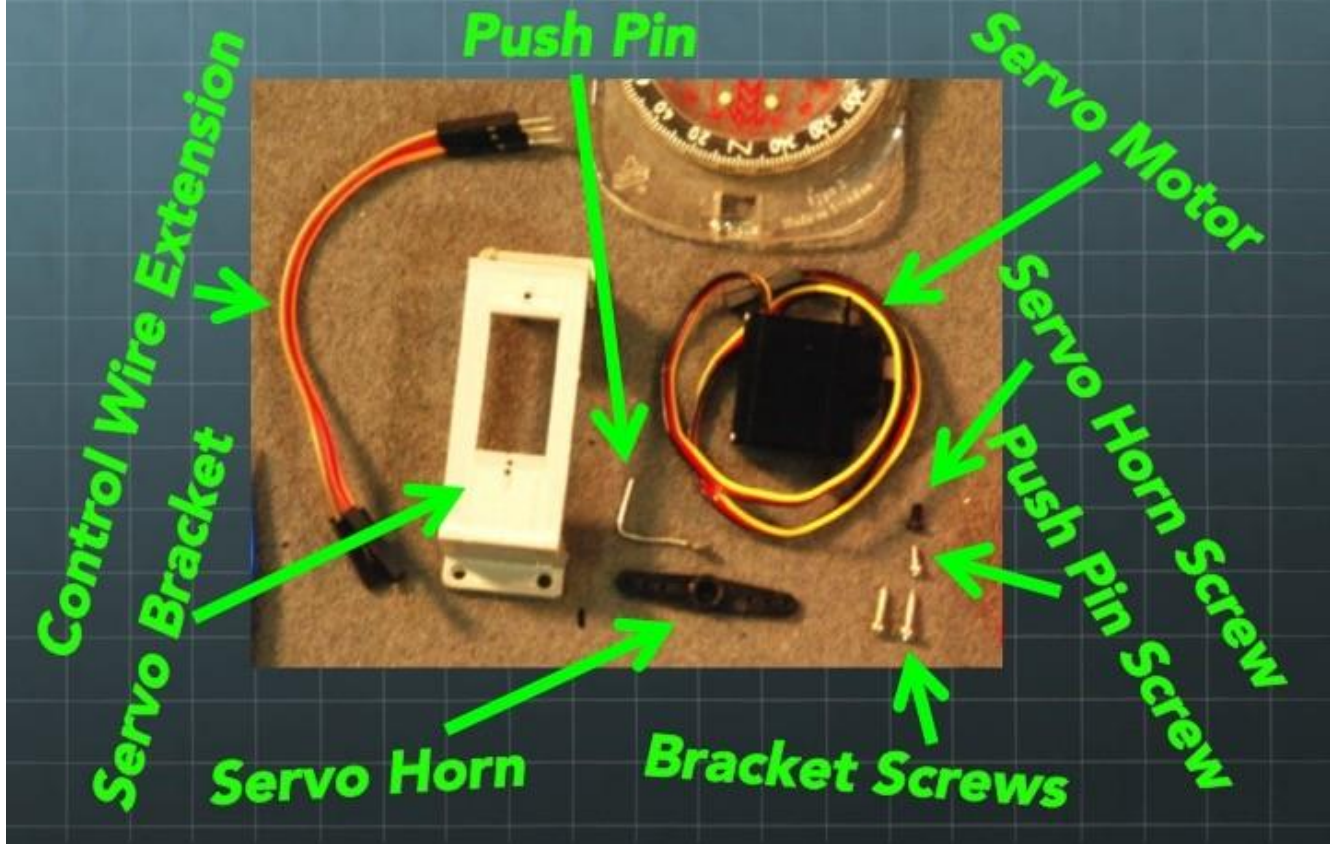


Twin Motor-Propeller Assembly Parts:

- A motor frame which consists of two green acrylic sheets and three small black brackets.
- 8 black (M2 x 10 mm) screws and hex nuts used to assemble the green motor frame.
- two DC motors and four tiny (#0 x 3/16") screws to attach motors to the small acrylic plate in the motor frame.
- two small white plastic worm gears used to join the motor shafts to the propeller shafts.
- two propellers placed at the opposite end of the shafts from the motors. They are not the same and have a small number on them to help with proper placement.

CAUTION: The propeller shafts (long rods) are thin. Be careful not to bend them to avoid later problems with your propellers.

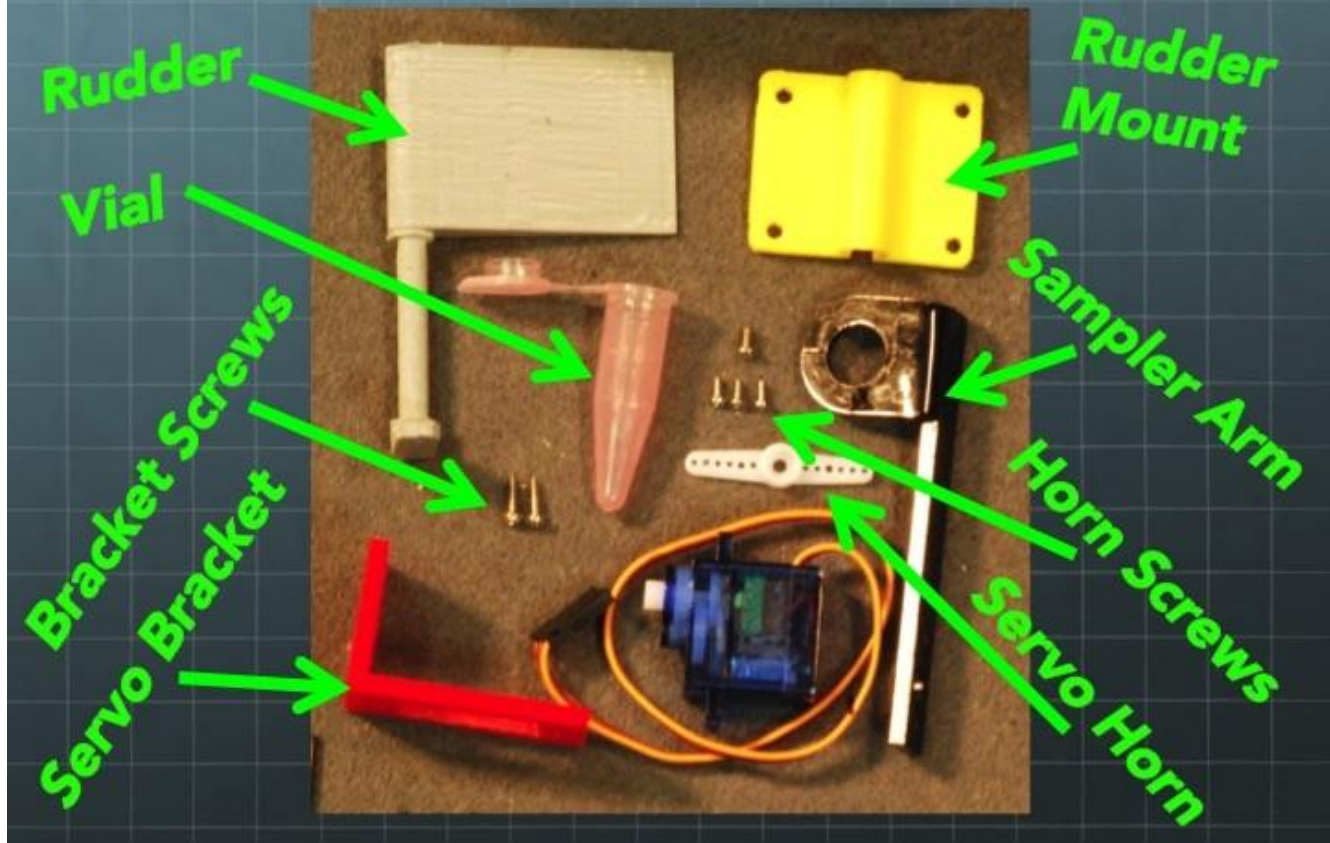
Rudder Servo Assembly



Rudder Servo Assembly Parts:

- servo motor with three control wires
- 3D printed servo bracket
- extension (M – MF jumper wires) for the servo control wires
- servo horn (large)
- bracket screws, which mount the servo motor on the 3D printed servo bracket are the two long, pointed screws.
- one (#0 x 3/16") screw to attach the loop of the push pin to the servo horn.
- one servo horn screw (tiny black) to secure the servo horn on the cog wheel of the servo motor.

Rudder and Water Sampler

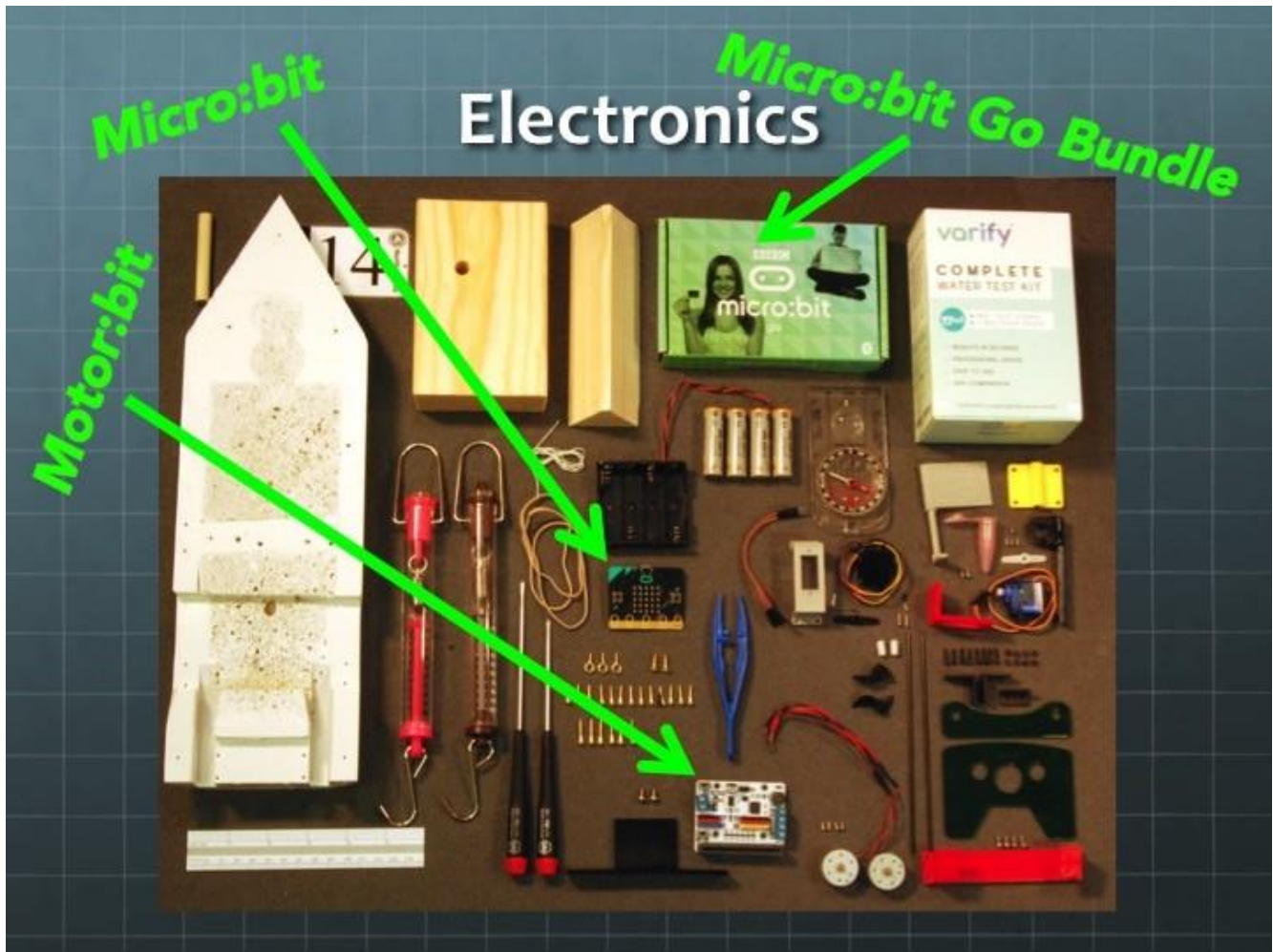


Rudder

- 3D printed rudder with rudder shaft
- 3D printed rudder mounting bracket, the rudder shaft snaps into the bearing.

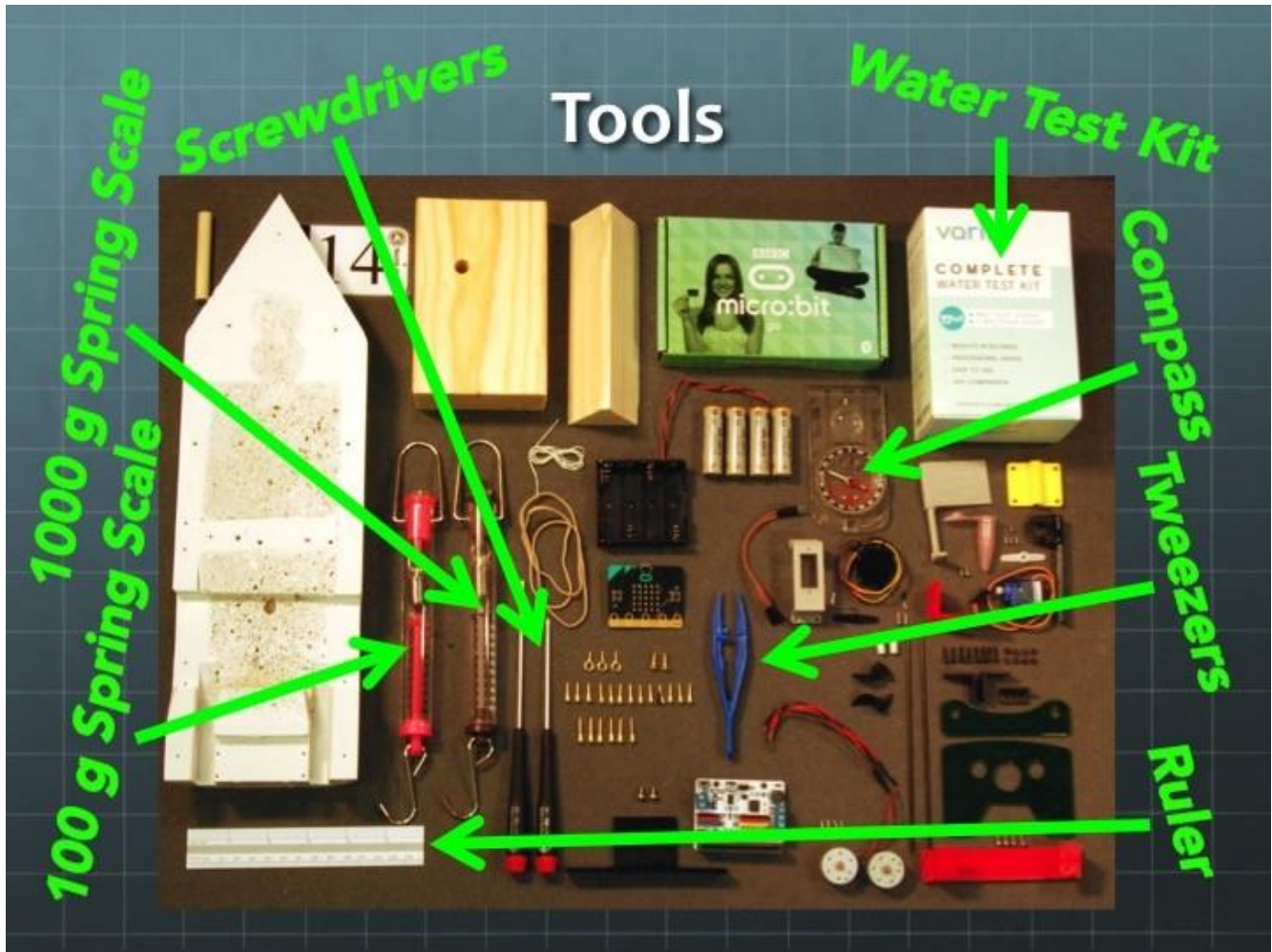
Water Sampler

- servo motor with three control wires.
- servo bracket mounts the servo motor to the hull.
- bracket screws are the two long, pointed screws to attach the servo to the bracket.
- long servo horn.
- three (#0 x 3/16") horn screws mount the sampler arm to the horn.
- sampler arm is the laser cut and bent black acrylic piece.
- sampler vial fits into the opening of the sampler arm.



Electronics

- Micro:bit Go Bundle Kit: a micro:bit board, a USB cable, a battery case and two AAA batteries.
- loose micro:bit is used as a remote control.
- motor:bit board provides power to the micro:bit, has terminals for the two DC motors, and has many pins for connecting the servo motors.



Tools

- 100 g tubular spring scale
- 1000 g tubular spring scale
- screwdrivers, one slotted, one Phillips
- water test kit (you will not receive a full kit)
- compass
- tweezers
- ruler

Brass Screws

Eyebolts



#4 x 1/2



#2 x 1/2



#3 x 5/8

Brass Fasteners:

Eyebolts

- three eyebolts with 1/8" inner diameter

Screws

- two (#4 x 1/2") flat head screws (top right) for mounting the battery case.
- six (#3 x 5/8") rounded head screws (bottom row)
- thirteen (#2 x 1/2") rounded head screws (middle row)

Machine and Plastic Screws



Machine Screws and Nuts

- two (M3 x 6) for mounting the motorbit board to the motorbit mount
Note: M3 means it is a metric screw with a thread which fits through a 3 mm diameter hole, the "x 6" means the screw (without head) is 6 mm long
- four (M2 x 6) for mounting the twin motor assembly to the motor mount
- four (M1.5 x 4) screws to mount the motors to the motor frame
- eight (M2 x 10) screws to assemble the twin motor frame
- eight (M2) hex nuts for the same purpose

Plastic Screws (note they have a coarser thread than machine screws)

- four (7 mm) long plastic screws for mounting the servo motors to the servo motor mounts
- two servo horn screws – these are the screws which come with the servo motors; often they are attached already; each servo motor has only one screw; make sure you do not lose this screw!
- four (#0 x 3/16") plastic screws one is for mounting the rudder push rod to the servo horn, the three others for attaching the water sampler arm to the sampler's servo horn
Note: #0 stands for the thread size in the American imperial system, "x 3/16" means they are 3/16 inches long

Complete Parts List

Number	Part Name	Part Details	Packed	Received
1	hull	yellow pine/spruce, painted		
2	dry dock	yellow pine 2x4 with hole		
3	3/8" dowel	poplar hardwood, 50 mm long		
4	fulcrum	yellow pine/spruce, plain		
5	microbit	plain		
6	microbit	Go Bundle		
7	battery case	for 2x AAA, comes with microbit Go Bundle		
8	2x AAA	batteries, come with microbit Go Bundle		
9	4x AA	batteries		
10	motorbit	made by ElecFreaks		
11	motorbit bracket	for mounting the motorbit		
12	battery case	for 4x AA		
13	2x DC motors	part of twin motor-propeller kit		
14	2x propellers	L/R part of twin motor-prop kit		
15	2x worm gear	part of twin motor-propeller kit		
16	large base plate	green acrylic, part of the motor frame		
17	small front plate	green acrylic, part of the motor frame		
18	wide bracket	part of twin motor-propeller kit		
19	2x long bracket	part of twin motor-propeller kit		
20	motor bracket	3D printed part		
21	servo motor	for rudder		
22	servo horn	for rudder		
23	push pin	for rudder		
24	servo bracket	for mounting the servo motor of the rudder, 3D printed part		
25	control wire	extension for servo motor wires		
26	rudder	3D printed part		
27	rudder mount	3D printed part		
28	servo motor	for water sampler		
29	servo horn	for water sampler		
30	sampler arm	laser cut part		
31	sampler container	centrifuge vial, 1.5 ml		
32	servo bracket	for mounting the servo motor of the water sampler, 3D printed		
33	competition sign	laminated number		
34	2x servo horn screw	comes with servo		
35	pushrod screw	comes with servo (rudder)		
36	sampler screws	3x #0 x 3/16", on sampler arm		
37	2x #4 x 1/2" brass screws	brass wood, for battery case		
38	4x #3 x 5/8" brass screws	for mounting the rudder bracket		

39	15x #2 x 1/2", brass screws	for servo motor mount (4x), sampler servo mount (3x), motorbit mount (2x), twin motor mount (4x), number mount (2x)		
40	3x eyebolts	1/8" eye, brass		
41	2x M3 x 6 mm	Phillips, stainless, for mounting motorbit		
42	4x M2 x 6 mm	for mounting twin-motors on motor bracket		
43	4x M1.5 x 4 mm	for mounting the motors to the motor frame		
44	8x M2 x 10 mm	for assembling the motor frame		
45	8x M2 hex nuts			
46	4x long plastic screws	for mounting servo motors on their brackets		
47	2x servo horn screws			
48	4x #0 x 3/16"	screws for mounting on servo horn		
49	string	60 cm		
50	spring scale	tubular, 1000 g / 10 N (brown)		
51	spring scale	tubular, 100 g / 1 N (pink)		
52	5 x rubber band	for 100 g spring scale / boat		
53	magnetic compass			
54	12 inch ruler			
55	Phillips #0	screwdriver		
56	slotted screwdriver	2 mm		
57	container			
58	container lid			
59	flash drive	STEM SEALs 8 GB		

Module 2: What Does a Microcomputer Do?

STEM SEALs devices 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)

2.1 Exploring the Micro:bit

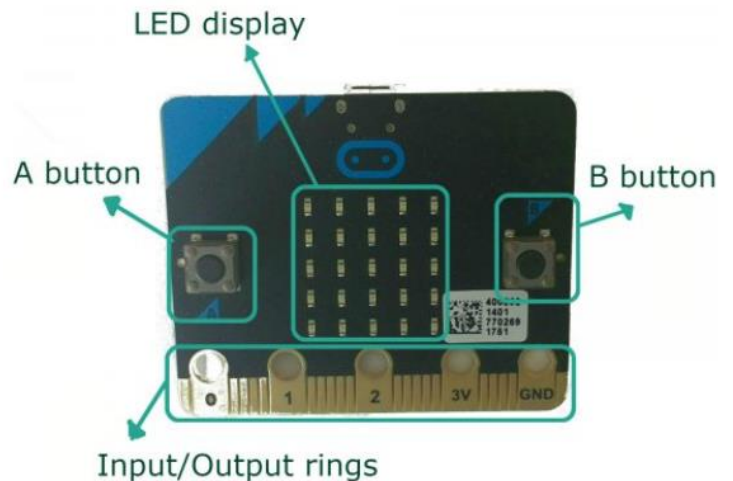
Open the micro:bit. (Go Bundle Kit). It contains the following items:

- 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.

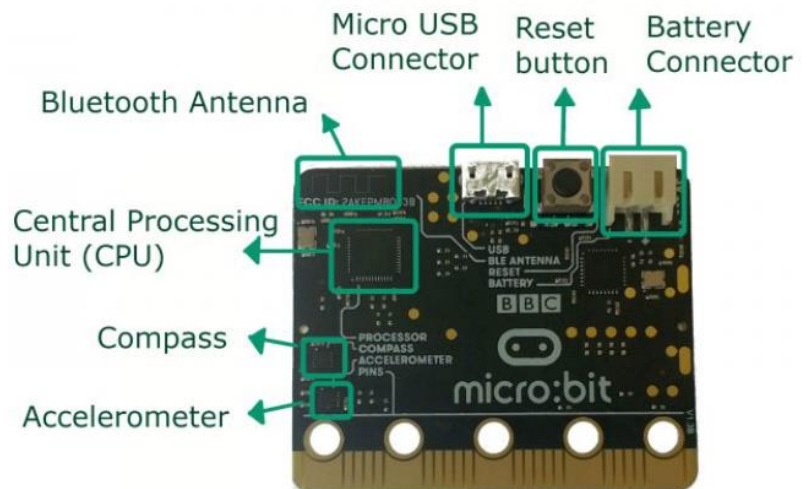
The front of the micro:bit contains the following features:

- 25 LEDs in a 5 X 5 array
- 2 input buttons (A and B)
- Input/Output rings



The back of the micro:bit contains these features:

- a battery connector for the battery case with J2 connector
- a reset button
- a micro-USB connector
- a bluetooth antenna
- the CPU (Central Processing Unit)
- a compass
- an accelerometer





Internet Resources:

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

- [Introducing the BBC Micro:bit – BBC Make it Digital](#) (video)
- [Introduction to the Micro:bit](#) (video)
- [Getting Started with the Micro:bit Part 1: Say Hello](#) (video)
- [Micro:bit Tutorial Series Part 1: Getting Started](#) (video)

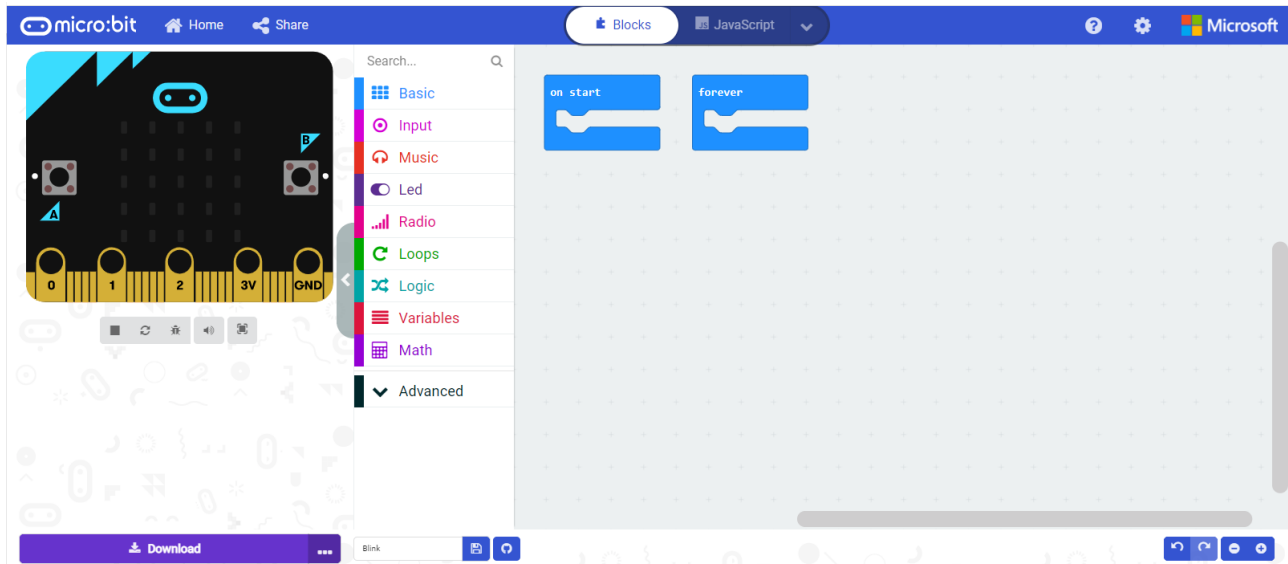


Try This: Run the DEMO Program on the Micro:bit

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, shaking the screen to light up every LED, and playing a chase the blinking dot game.
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.

2.2 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 SEA 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.



Explore the features of the MakeCode Editor:

- 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.
- 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: Log into MakeCode

1. On a computer connected to the internet, open a browser (Edge, Firefox, Safari, Chrome, etc.) and go to www.microbit.org.
2. In the tab bar at the top of the microbit.org page select “Let’s Code”.
3. On the new page, select the button labeled “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. On the new page, select the “New Projects” icon (+).
5. A popup box will appear (Create a Project) and prompt you to give the project a name.
6. Name the project “Blink” 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” 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 edit it. “Save” downloads a copy of the code to your computer device. This file will be saved as a .hex file.

2.3 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.

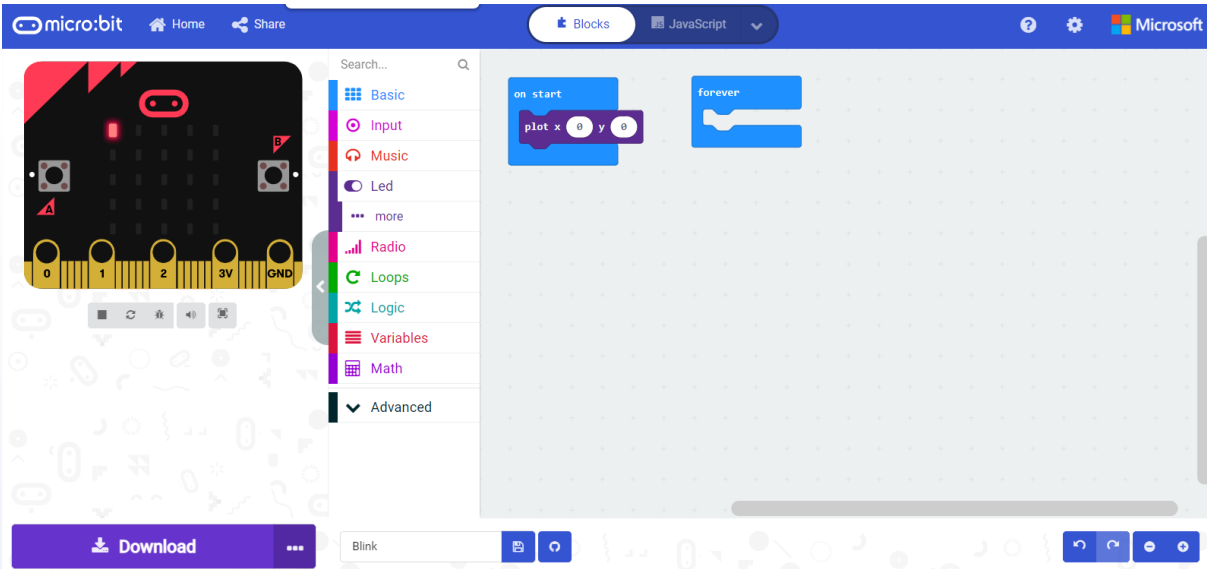
Follow the steps on the next pages to make the LEDs of the micro:bit blink.



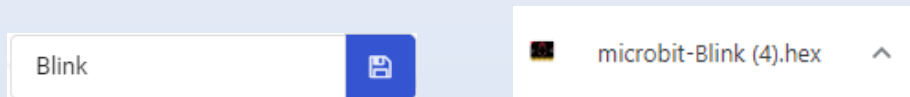


Time to Code: Create Simple Code and Download on the Micro:bit

1. Use the project you named “**Blink**”
 - To locate the “**Blink**” project, go to the home page of MakeCode.
2. Click on the “Led” toolbox and drag the “plot x 0 y 0” into the “on start” block.



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 producing light when an electric current flows through it.
4. Click on the “Save” button at the bottom. A window may open giving you choices: Choose “Save File”. Watch the top corner or bottom corner of your browser while you are doing it. Most browsers show the “Downloads” folder in these locations. The download will be a .hex file like below.



5. Click on your Downloads folder (or open it from FileExplorer (Windows) or Finder (Apple)).
6. Connect your micro:bit to your computer using the mini-USB cable. Your computer should recognize this connection.
7. Drag “microbit-Blink.hex” (the file you just downloaded from MakeCode) to “MICROBIT” in the FileExplorer or Finder. This moves the code from the download file onto the micro:bit. Remember the code you create in MakeCode is always saved on the server unless you delete it.
8. Look at the display of the micro:bit. Is the upper left LED on?
9. Notice the LED is just illuminated. Can you make it blink?




Time to Code: Making the LED Blink

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

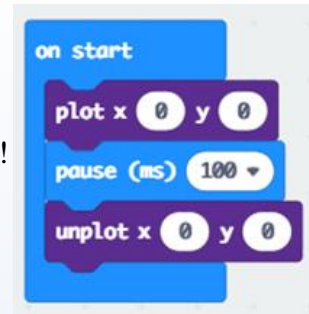
1. Open up MakeCode, select the “**Blink**” project on the home page. notice that changes are saved automatically in the program.
2. Click on “Basic” in the toolbox, find and drag the “pause (ms) 100” block and place it directly underneath the “plot x 0 y 0”.
3. Drag an “unplot x 0 y 0” block from the “LED” toolbox and place it underneath the pause.

4. Look at the micro:bit simulator on the left.

- Do you see anything?

5. Click on the “Restart the simulator” icon  below the simulated micro:bit in the browser. Watch the display!

- Does that make sense?
- Of course, if we want to make the LED blink that means we must turn it on and off and on and ... so on.
That seems like a horrible task! We can do better!



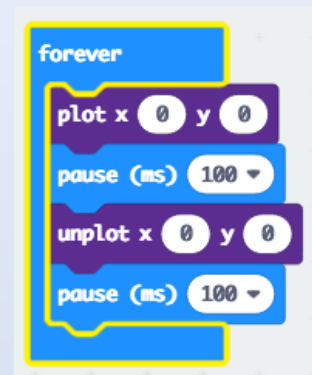
6. Copy the three blocks in “on start” and paste them in the “forever” block.

- You could get new ones from the toolbox. But try to copy and paste: Select what you want to copy, hold the Control key (Mac: Command key) and press the C key, let both go, hold the Control key (Command) again and press the V key: You duplicated the block, which you now drag wherever it needs to go.
- You can also do this by hovering over the block you want to duplicate, right click with the mouse, and select the option in the pop-up box.

7. Copy the “pause (ms) 100” and paste it as the last block into the “forever” block (it does not matter which one!).

8. Watch the display of the simulator!

- The upper left LED blinks forever!
(Not really: only until you change the code, or until you close the browser.)

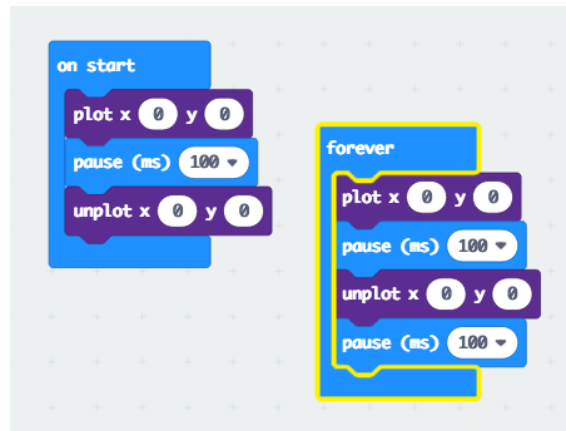


9. Now, “Save” (download) and upload the new code to your micro:bit.

Does the LED on your micro:bit blink?

You just successfully completed your first coding project. Congratulations!

Your complete code should look like the image below on the MakeCode workspace:



“Blink” project code as displayed by the MakeCode editor.

Once code has been created, remember it is automatically saved on the MakeCode browser but you must be on the same device.

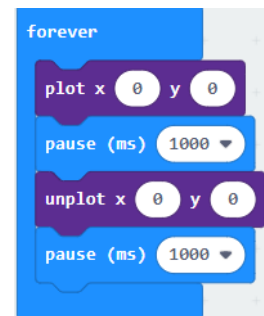
Code can be edited and shared for others to work with.

Let’s edit your code.

Do you know how long 100 ms is?

Click on the drop-down arrow beside the number 100 in the pause block in the forever block. In the drop-down menu select 1 second. Do this for both the pause blocks in the forever block.

Look at the simulator display. Does this make sense? What changed?

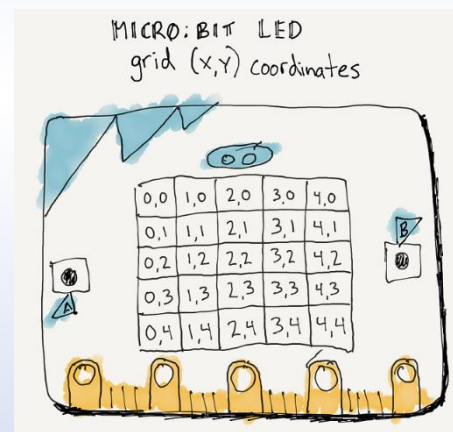


Note: Micro:bit LEDs

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>



2.4 Sharing or Publishing Your Code

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.

Here is the URL link to the SEA Challenge “Blink” project created by the STEM SEALs team.

https://makecode.microbit.org/_Jf1RDhVoWY4R (Blink)

Your project even though it has the same name would have its own unique URL.



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.

2.5 Variables

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 names 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

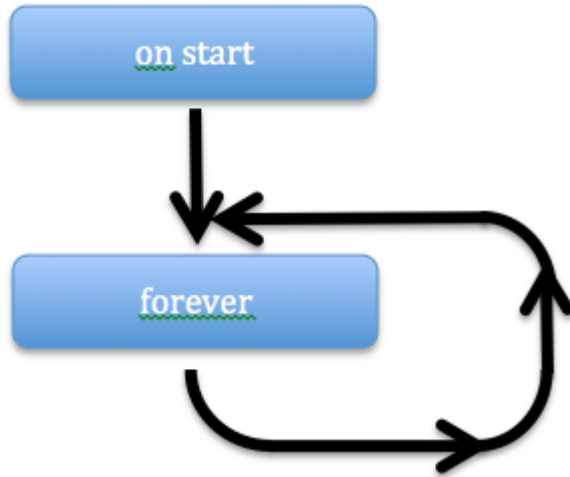
Check out the video to learn more about variables.

- [Create a Variable in MakeCode](#) (video 1:22)
- [Micro:bit – Why variables?](#) (video 8:55)

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”.

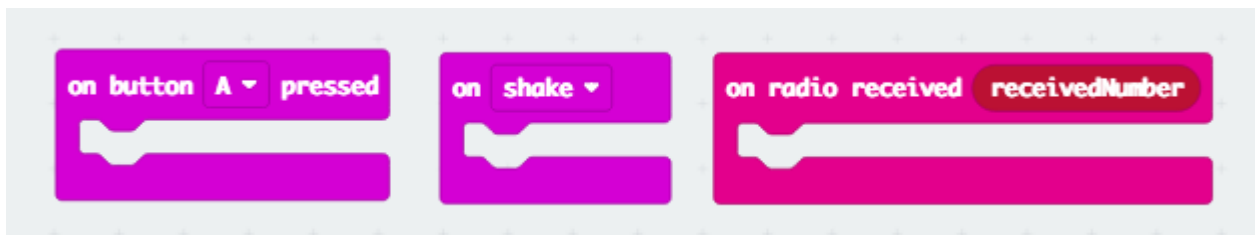
The micro:bit only stops running the “forever” block commands when it is turned off or restarted with the reset button.

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.

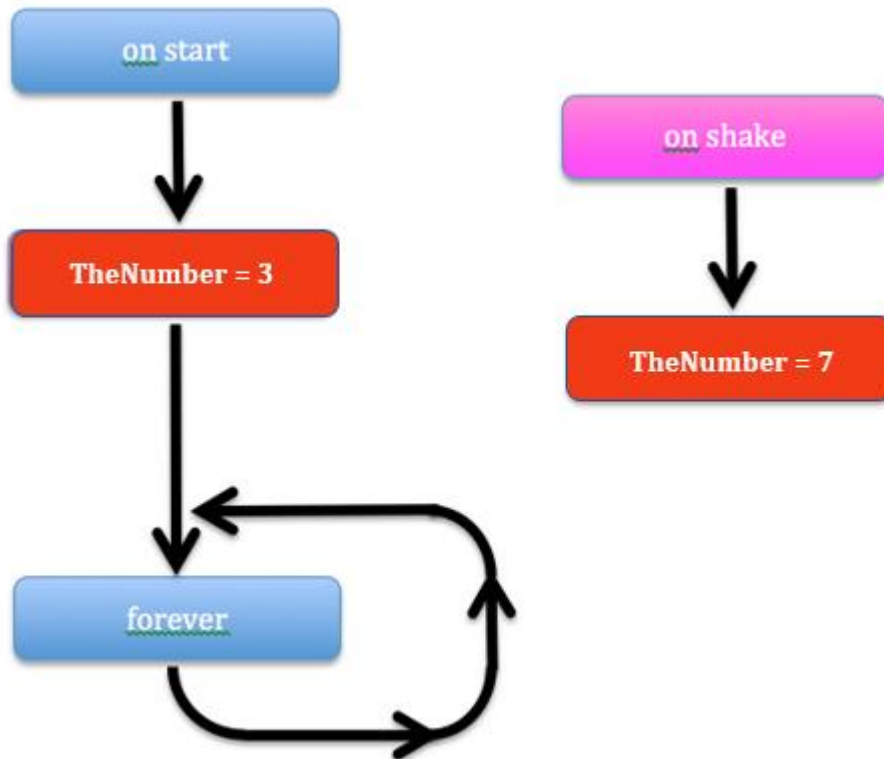
3.2 Interruptions in Code

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.

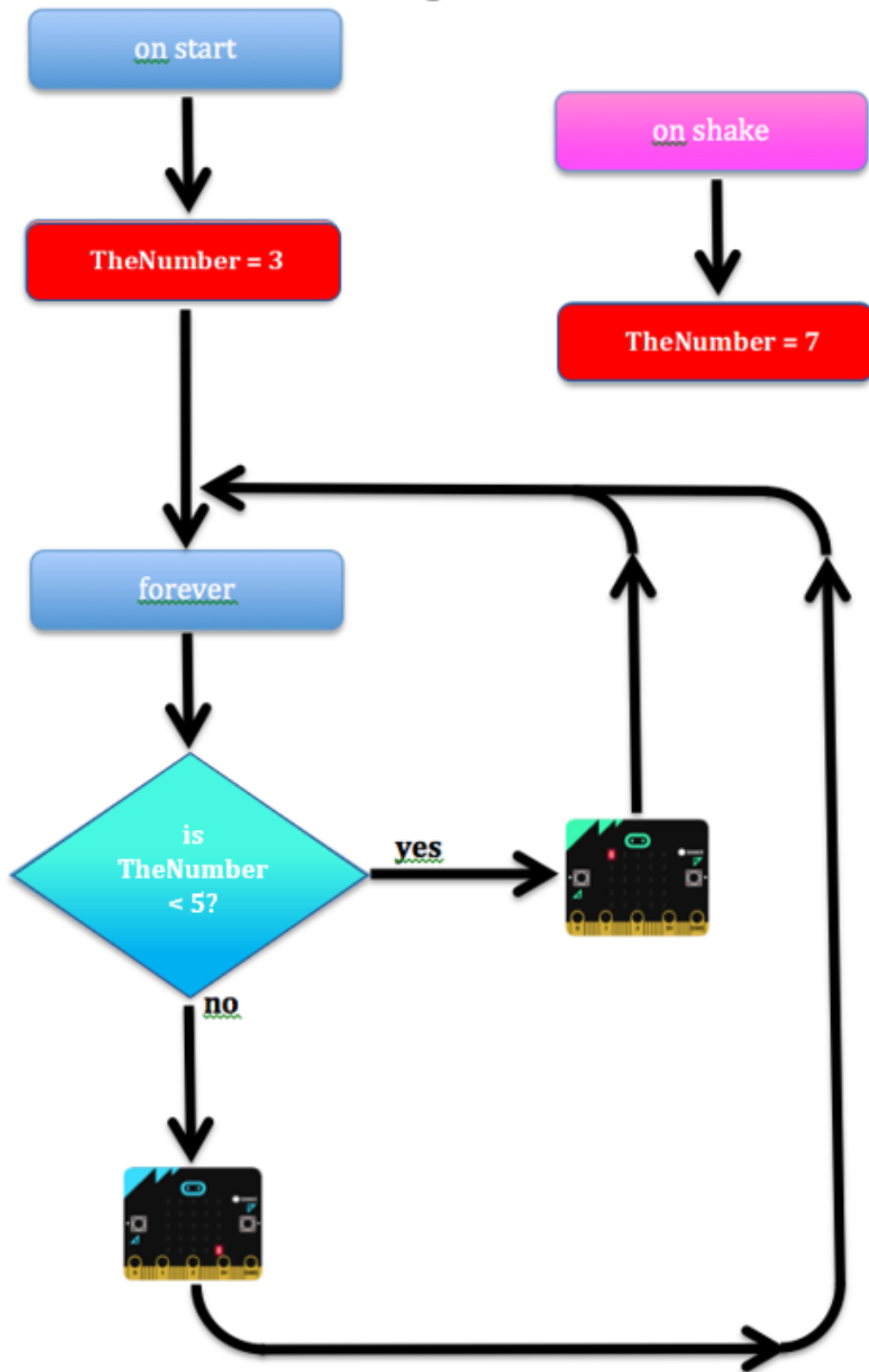


We can place blocks inside these input blocks, which then are executed once each time the event happens such as pressing the button or shaking the micro:bit. After the input commands are completed within the input blocks, the forever loop resumes with whatever it was supposed to do. The empty blocks 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.

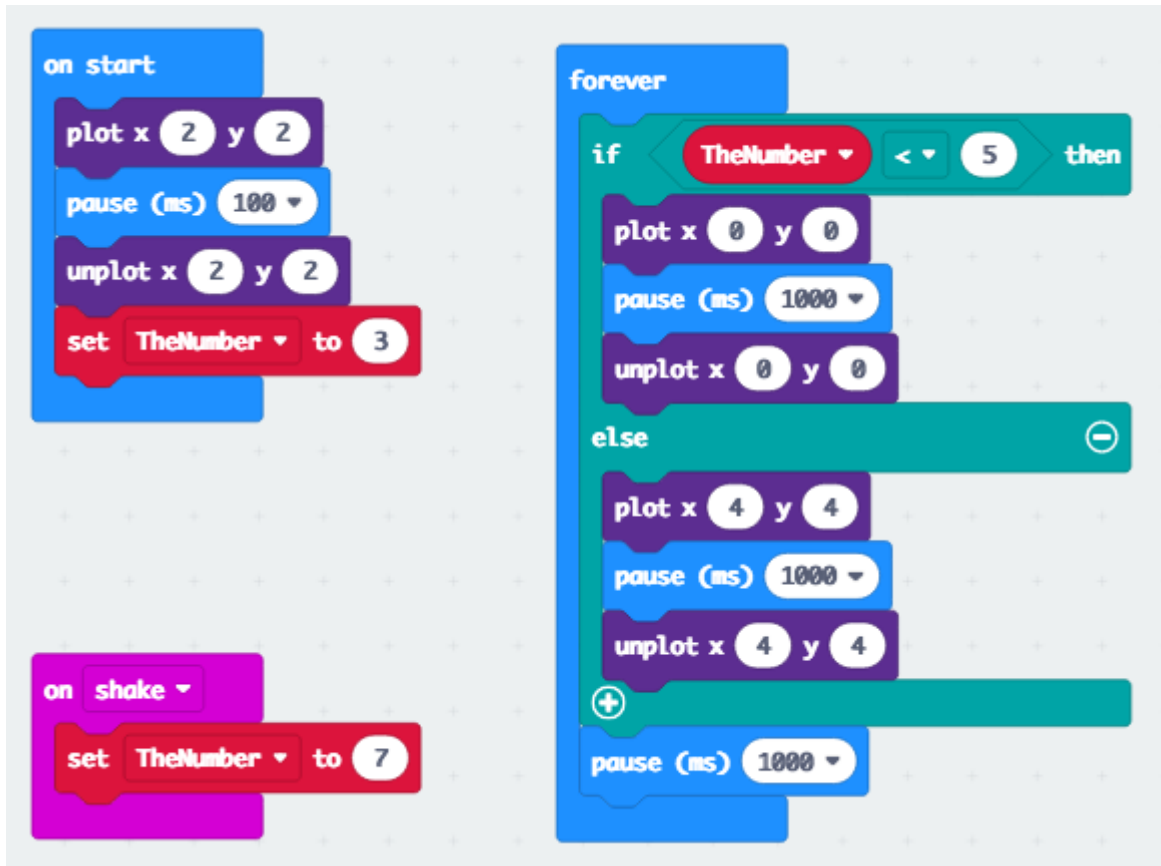


For example, if we set the variable “TheNumber” to 3 in the “on start” block, the forever loop “TheNumber” will always be 3 ... until we shake the micro:bit: Then the “on shake” block will set “TheNumber” to 7 and every time thereafter “TheNumber” will be 7 and the forever loop will “know” this.

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)

1. Open the “Blink” project – see the note on the next page. 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?



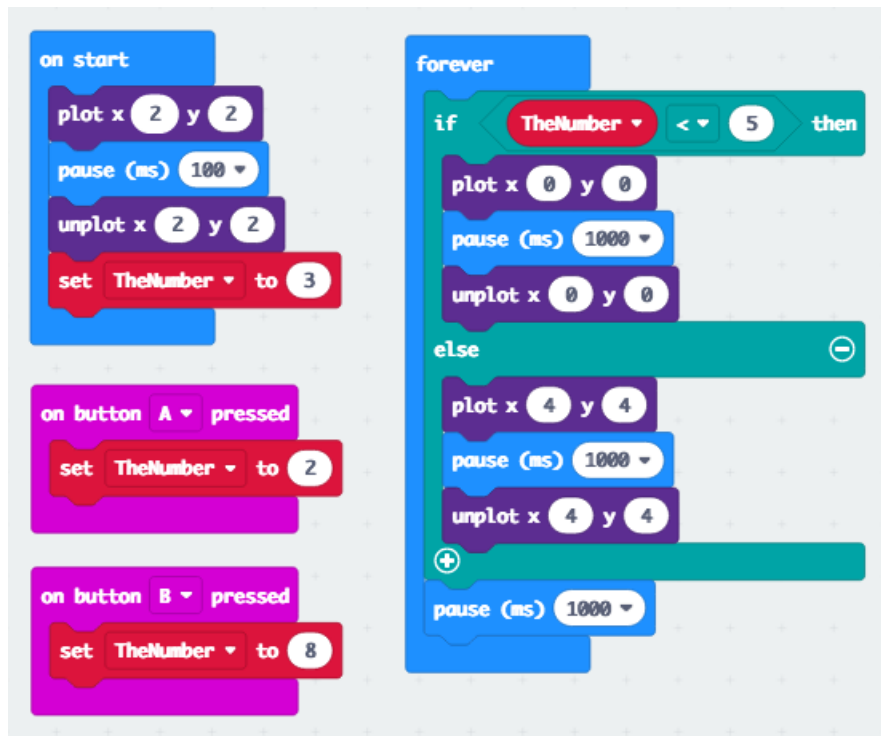
Tip: Using Previous MakeCode Projects

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 and duplicate and rename for extra safety precautions.

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:



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:

The image shows a Scratch code editor with the following blocks:

- on start** block containing:
 - `plot x 1 y 2`
 - `pause (ms) 100`
 - `unplot x 1 y 2`
 - `set TheNumber to 0`
- forever** loop containing:
 - `plot x TheNumber y 2`
 - `pause (ms) 1000`
 - `unplot x TheNumber y 2`
 - `pause (ms) 1000`
- on button A pressed** block containing:
 - `change TheNumber by -1`
- on button B pressed** block containing:
 - `change TheNumber by 1`

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 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!

```
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
  unplot x TheNumber y 2
  change TheNumber by -1

on button B pressed
  unplot x TheNumber y 2
  change TheNumber by 1
```

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!

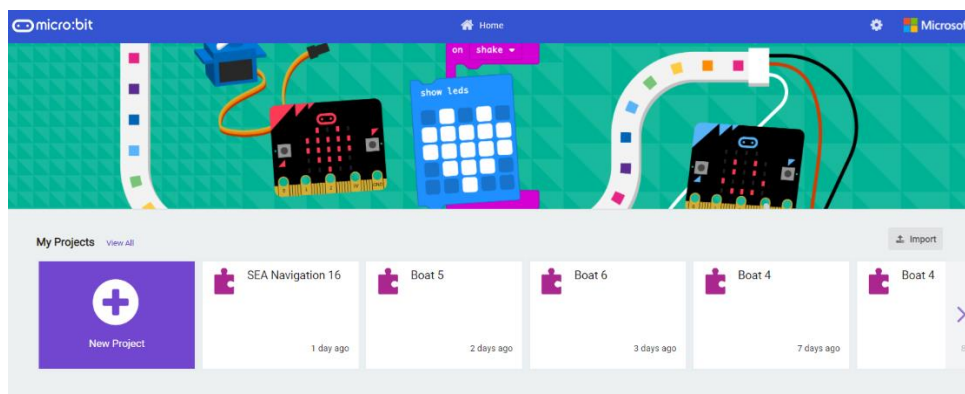


Here we use the “if ... then” block without the “else” part to set the number to a value of an existing LED if the “TheNumber” variable gets out of range.

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.

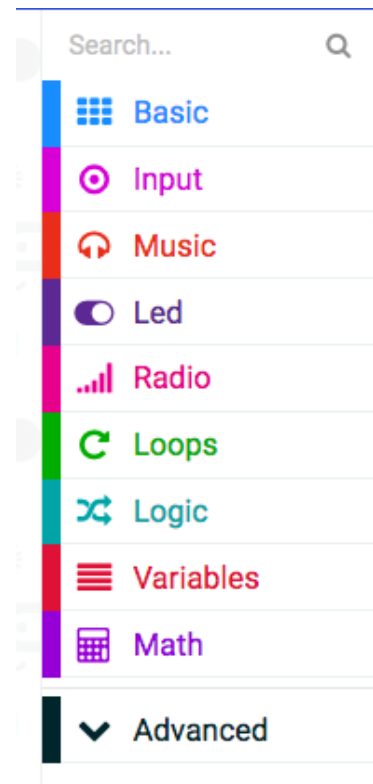
The possibilities are numerous when you combine blocks with many functions!

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



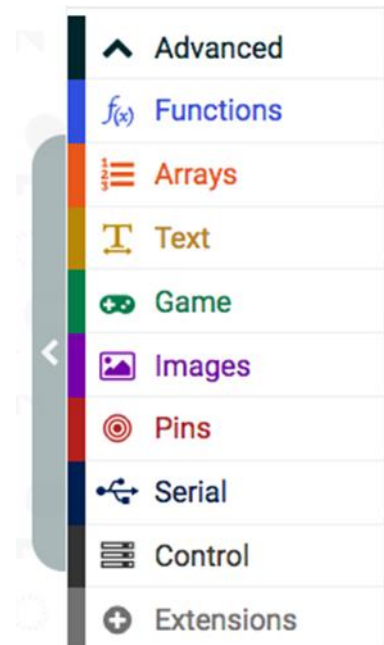
3.3 The Essential MakeCode Blocks

- 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.



3.5 MakeCode Java Script

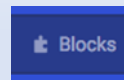
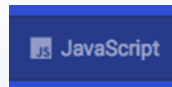
The MakeCode editor you have seen so far is a graphics-oriented editor. The blocks you drag, and drop must be assembled to a meaningful graphical image, which then is translated when you click “Save” into code, which the computer can read.

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 SEA Challenge but will utilize it to easily copy some code for a couple of activities.



Try this: MakeCode JavaScript

1. Load https://makecode.microbit.org/_Jf1RDhVoWY4R (**Blink**) or open on the MakeCode homepage. This is the Blink 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.substring(0, 100)) + "...");
17     },
18     onComplete: function(request) {
19       if (request.name == 'log_error') return;
20       log_fatal(request.url + " " + request.name + " " + request.responseText);
21     },
22     onException: function(request, e) {
23       if (request.name == 'log_error') return;
24       log_fatal(request.url + " " + request.name + " " + request.responseText + " " + e.message);
25     }
26   })();
27 }
28
```

Module 4: Testing Water Quality

4.1 Why are we testing water samples?

It is fun to make a robotic boat, but remember robotic boats and crafts usually serve a purpose. Water is essential for the life and economy of Florida. While Florida is surrounded by water on three sides, it is the aquifer systems under Florida that provide most of the state's water. Keeping our surrounding bodies of water and our aquifers healthy is important to the wellbeing of the plants, animals, and humans. Florida has more than 7,700 lakes. These lakes and other freshwater areas provide habitats for plants, birds, fish, and animals. The watersheds provide aquifer recharge, natural flood protection, water purification, preservation of wildlife habitat, and public recreation. Robotic boats and devices are valuable resources to help manage Florida's water resources. Using the robotic boats, we will sample water from Lake Osceola and then test, analyze, and report the results and conclusions of the sampled water.



Internet Resources: Use of Robotic Boats

Check out these resources to learn more:

- FAU to Develop Robotic Boats with a “Mind of Their Own”(article)
<https://www.fau.edu/newsdesk/articles/fau-usv.php>
- Robotic Boats Have a Mind of Their Own on Water (article)
<https://www.nbcnews.com/technolog/robotic-boats-have-mind-their-own-water-6c10643235>
- Engineering Smarter Robotic Boats for Safer, Cheaper Work on the Water (video, 3:03)
<https://youtu.be/R4CAZCCaUUg>
- Students Build Underwater Robot- Use it to Study Indian River Lagoon Problems (mixed media)
<https://www.mynews13.com/fl/orlando/news/2021/06/18/students-build-underwater-robot--learn-about-steam>

4.2 What are the things we will we testing in our water samples?

One way to determine the health status of waters is by testing for minerals and small **molecules**, bacterial and algal **load** and types. The four analytes or samples that we will focus on for the competition portion of the camp will be **nitrate**s, **nitrite**s, **sulfate**s, and the **hydrogen ion** (pH). The kit we use will test more than these four, but we will focus on how these can affect our water resources. When we do the competition, we will test the first three analytes together with a single test strip. Then pH will be tested separately using litmus paper. To gain experience with the testing procedures we will test each analyte separately (one test strip per analyte) in our practice sessions. So, let's begin to familiarize ourselves with the test and analytes with some background and practice.

4.3 Testing for nitrates, nitrites, and sulfates.

We will use the Varify Complete Water Test Kit (<https://varifytest.com/products/complete-drinking-water-test-kit>) to gather data on the concentrations of nitrates, nitrites, and sulfates. The figure below shows the complete list of analytes (and bacteria) that can be tested with your kit. Each analyte has a brief description and some information regarding health hazards and/or other complications associated with the analyte. When doing the practice phase using this kit, we will take a sample from some common household items or solutions prepared for you that are known to contain our analytes. These will be mixed with water in a total volume of 15 **milliliters** (ml) and then tested.

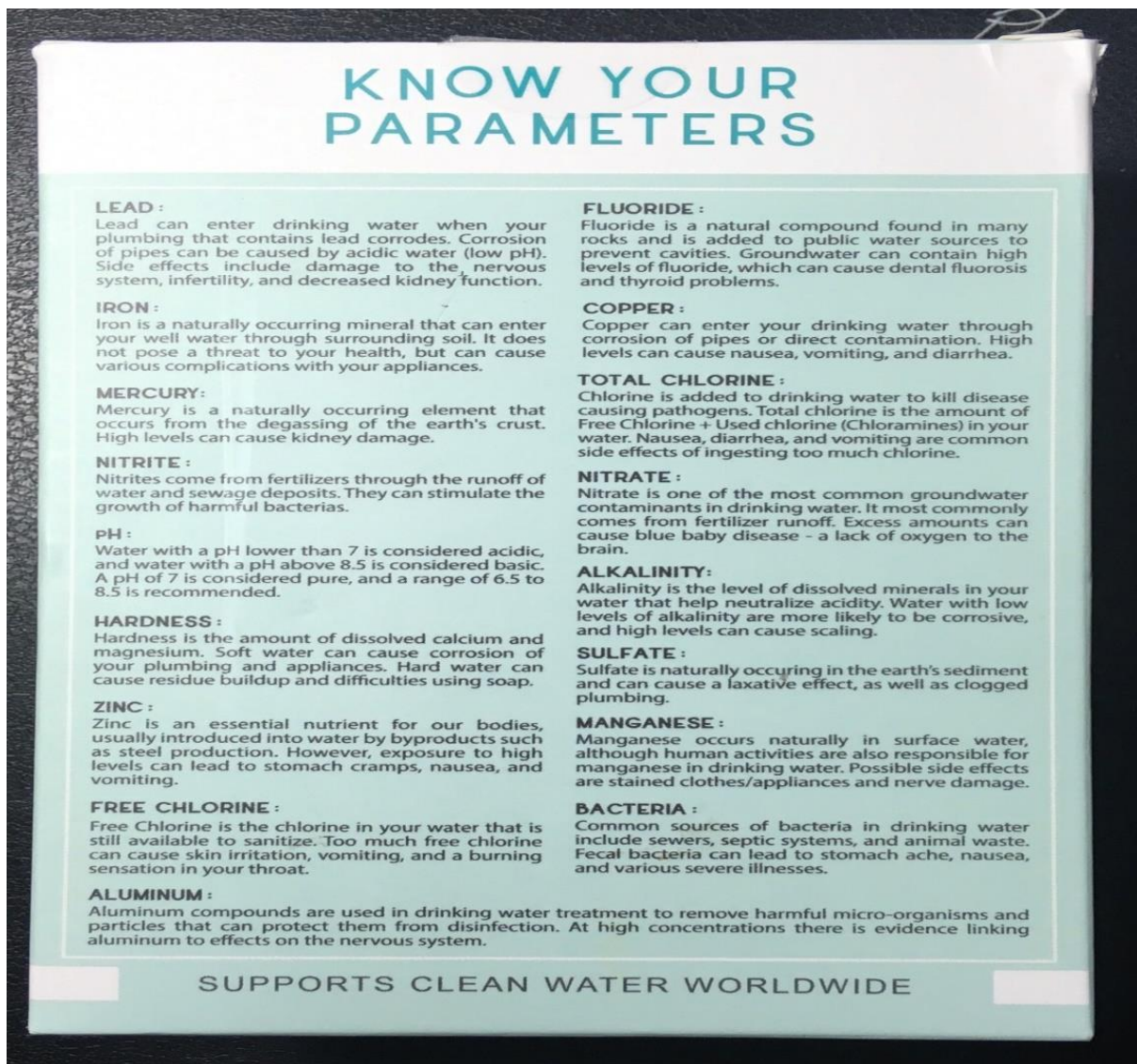


Figure 1. Shown above are analytes that can be tested with your Varify test strips and complications that can arise when their concentrations are too high in your water supply.

Where can we find substances that contain nitrates, nitrites, and sulfate?

Nitrate (NO_3^-) is found in common household items such as toothpastes, fertilizers, fireworks, and pesticides.

Nitrite (NO_2^-) is found in common household items such as spray paint, deodorant and hairsprays, vegetable oil cooking sprays, and static cling sprays.

Sulfate (SO_4^{2-}) is found in detergents, or surfactants. These are commonly found in products like shampoo, body wash, face cleanser, and toothpaste (in addition to household cleaning products, like laundry and dish detergent).



Think About It:

How do you think these contaminants get in our water?

What do you think you could do to prevent water contamination?

Materials needed for testing nitrate, nitrite, sulfate, and pH:

1. Samples for testing (provided on site)
2. Disposable 15 ml plastic tubes with caps (4)
3. Disposable 3 ml plastic transfer pipet (2)
4. Disposable cups (small)
5. Stirring sticks (4)
6. Bottled water
7. Varify test strips (6)
8. Analyte Concentration Key (small and large copy)
9. Litmus paper
10. pH Interpretation Key
11. Data record sheet
12. Pencil
13. Digital camera (optional)



Figure 2. From left to right: test strip, test tube and cap, transfer pipet.



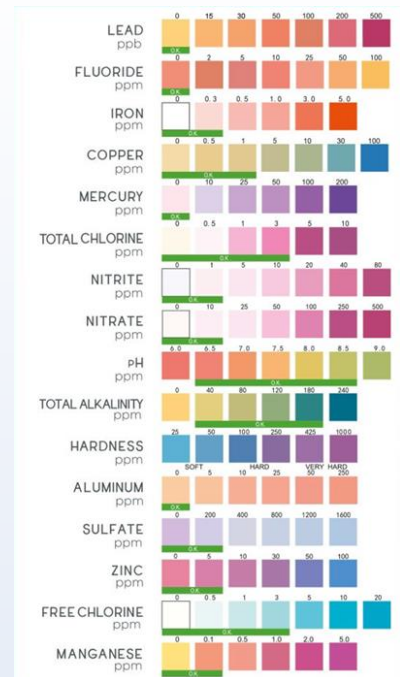
Try This: Testing for Nitrates, Nitrites, and Sulfates.

Note: When using the Verify test strips (nitrates, nitrites, and sulfates) It is important to be ready to read the results immediately after removing the Verify test strip from the liquid in your sample tube. Use the small, laminated concentration key for this.

1. Is your item a solid or powder?

- Using a disposable cup, place a small amount (1/2 teaspoon) of your test item in water and stir vigorously for a few seconds.
- Allow for undissolved powder or solids to settle.
- Add 1.5 ml of the solution from step 1b into your 15 ml tube and add bottled water to 15 ml mark.
- Mix well by inverting the capped tube 5-10 times.
- Test this using the protocol for the test strips.
 - Immerse for 2 seconds.
 - Remove and shake off excess liquid.
 - Compare analyte area of test strip (nitrate, nitrite, or sulfate) to the concentration key to determine concentration (in ppm) of the analyte.
- Remember you are testing a 1:10 dilution of your original sample from step c, so your test result should be multiplied by 10 to account for this.
- Record your results (analyte and ppm) on your Date Record Form.

Note: Between each item you test it will be important to clean (flush out at least 3 times) your transfer pipette and test tube with fresh bottled water.



2. Is your item a liquid?

- Carefully remove 1.5 ml and dilute to 15 ml with bottled water.
- Mix well by inverting the capped tube 5-10 times.
- Test this using the protocol for the test strips.
 - Immerse for 2 seconds.
 - Remove and shake off excess liquid.
 - Compare analyte area of test strip (nitrate, nitrite, or sulfate) to the concentration key to determine concentration (in ppm) of analyte.
- Remember you are testing a 1:10 dilution of your sample from step c, so your test result should be multiplied by 10 to account for this.
- Record your results (analyte and ppm) on your Date Record Form.

For any given analyte a concentration change is demonstrated by shift in color from left to right on the line associated with that analyte. Analyte concentrations are given as parts per million (ppm)

Note: Between each item you test it will be important to clean (flush out at least 3 times) your transfer pipette and test tube with fresh bottled water.

4.4 Testing for pH (Hydrogen Ion)

A pH test measures the hydrogen ion (H^+) concentration or pH of a solution. The more hydrogen ions in a solution, the lower the pH and the solution is said to be **acidic**. The fewer hydrogen ions in a solution, the higher the pH, and the solution is said to be **alkaline** (or **basic**). The pH scale (shown below) is a $-\log_{10}$ scale that goes from 0-14. In practical terms this means for example that a solution with pH of 2 has 100 times more H^+ than a solution at pH of 4.

At a pH of 0, there is one mole (about 6×10^{-23}) of hydrogen ions in a one-liter solution. At a pH of 14, there is one mole (about 6×10^{-8}) of hydrogen ions in a one-liter solution. A neutral solution has a pH of 7.0. Acidic solutions are things such as bleach, colas (Pepsi, Coke, Dr. Pepper, Sprite, etc.), orange juice, and coffee. Alkaline products around the house include things such as baking soda, baking powder, antacids, and toothpastes. Biological systems work best at a pH of around 7. The figure below shows the pH scale and the pH values of various liquids/solutions that you are probably familiar with.

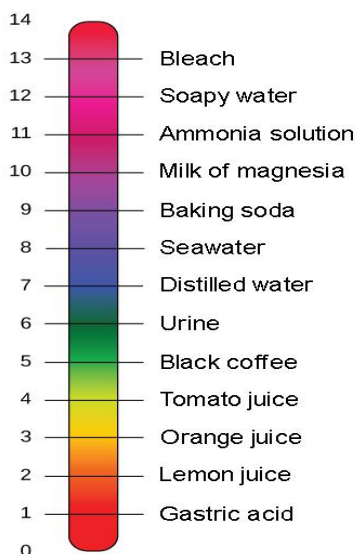


Figure 3. pH scale and pH for some common liquids/solutions.

When you perform your pH testing of your selected solutions/liquids you can use the reference standard immediately below to compare against the color resulting from your tests of the liquids.



Figure 4. Detail of pH color change key.

When you test a solution on the litmus paper, you can then compare the color change on your test strip to the key in this figure to determine the pH of your test solution.

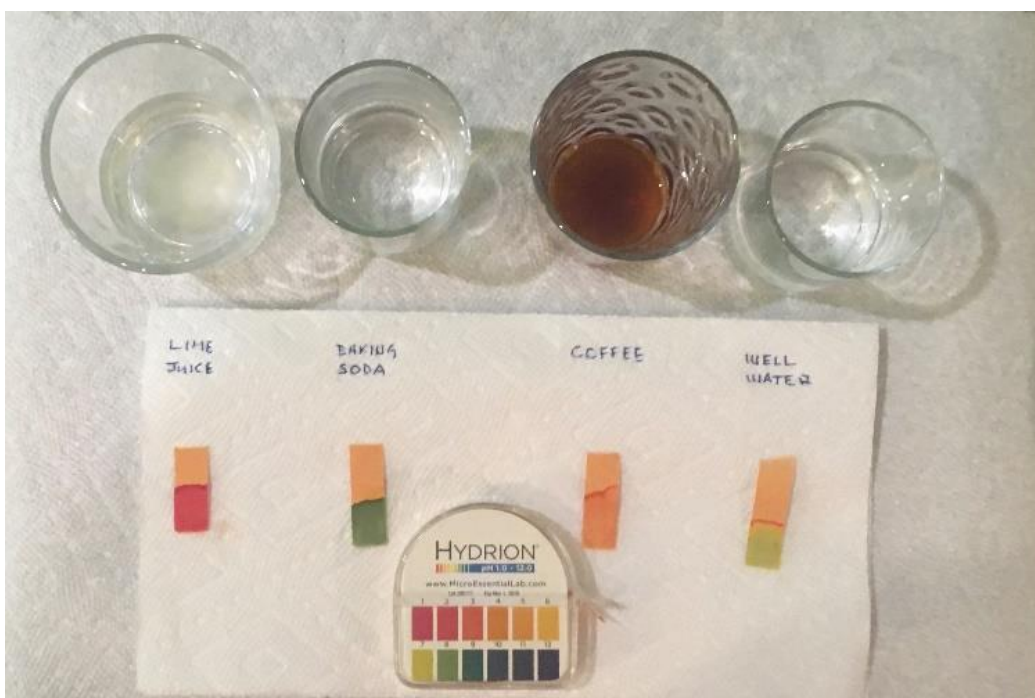


Figure 5. Test results for pH of four liquids (lime juice, baking soda, coffee, and well water).



Figure 6. What pH values would you assign to each of these four test liquids? Compare the color to the pH color key. How do your results compare to the results provided in Figure 3?



Try This: Testing for pH (Hydrogen Ions)

Collect several household items for testing (you may be provided these onsite). It may be best to select one that is known to be acidic (citrus juice, coffee, cola, etc.), one that is alkaline (baking soda, antacid, etc.), and one that is neutral (well water or bottled water).

Note: Between each item you test it will be important to clean (flush out at least 3 times) your transfer pipette and test tube with fresh bottled water.

1. Is your item a solid or powder?

If the answer is yes, follow the steps immediately below. If not, skip to number 2 (below).

- a. Place a small amount (about 1/2 tsp) of the powdered substance in a few ounces of bottled water and stir vigorously for a few seconds.
- b. Allow the undissolved powder or solids to settle.
- c. Remove 1.5 ml with your transfer pipet and dilute to 15 ml with bottled water in your test tube.
- d. Mix well by inverting the capped tube 5-10 times.
- e. Test a drop of this solution. Draw up about 1 ml of solution in the transfer pipet and dispense one drop onto the end of your pH test paper.
- f. Place on clean paper towel to blot dry.
- g. Remember you are testing a 1:10 dilution of your sample from step c, so if your sample shows a pH 6 result, the original solution from the tube is pH 5 (one pH unit lower, but 10 X more hydrogen ions).
- h. Record your results (analyte and pH).

2. Is your item a liquid?

- a. Carefully remove 1.5 ml of your test liquid with the transfer pipette and dilute to 15 ml with bottled water in your test tube.
- b. Test a drop of this solution. Draw up about 1 ml of solution in the transfer pipet and dispense one drop onto the end of your pH test paper.
- c. Place on clean paper towel to blot dry.
- d. Remember you are testing a 1:10 dilution of your sample from step b, so if your sample shows a pH 6 result, the original solution is pH 5 (one pH unit lower, but 10 X more hydrogen ions).
- e. Record your results (analyte and pH).

3. Does your result correspond to what we know about the pH of your test item?

Module 5: Floatation Test

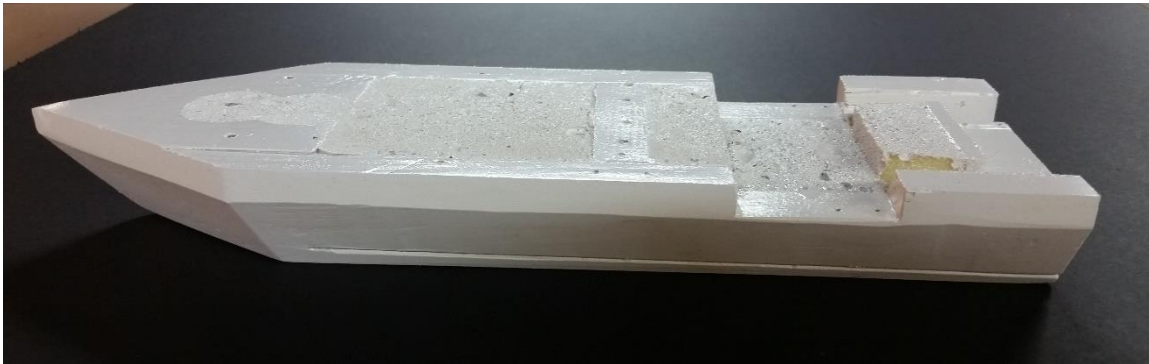


Think About It

Look at the **hull** of the boat.

What do you think it is made of?

Why do you think these materials were used?



The boat is mostly made of wood, two-by-fours (2 x 4) and one-by-fours (1 x 4). It is not solid wood though. It is also filled with foam. What **physical properties** do you think make this a good or poor boat design?

5.1 Mass and Volume

All matter (an object or substance) has both **mass** and **volume**. Mass is the amount of material in a substance. The mass of an object can be determined by the **weight** with a scale. Mass can be measured in different units: ounces (oz) is the **imperial unit**, grams (g) and kilograms (kg) are international standard units. Volume is the amount of three-dimensional space an object takes up. The volume of a solid is measured in cubic centimeters (cm³).

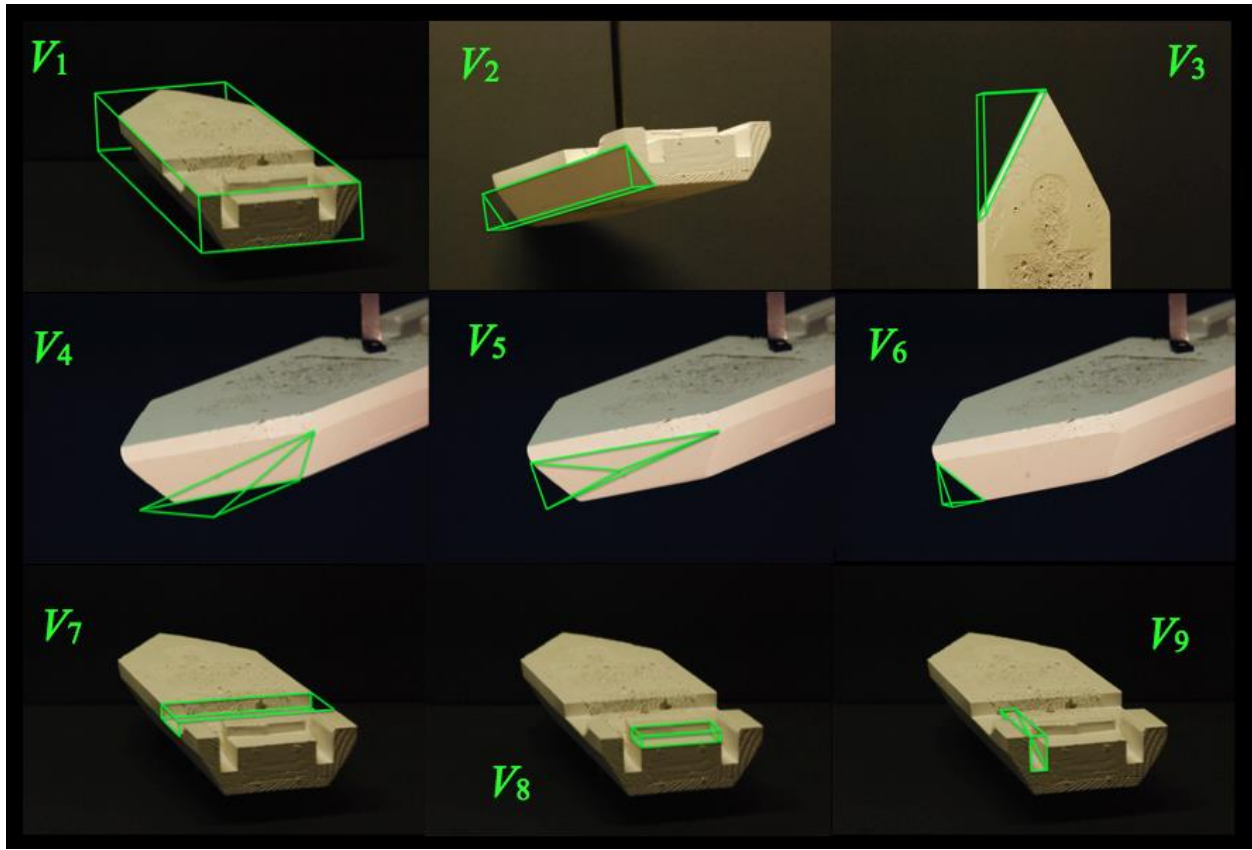
Let's explore the mass and volume of the boat hull.

What is the mass of your boat hull? _____

What method did you use to measure the mass? _____

What are two ways to measure the volume of an object? _____

Why might the volume of the boat hull be difficult to measure? (Hint: look at the diagram on next page) _____



5.2 Floatation of the Boat Hull

Do you think the boat will float well? When an object **floats**, it means that part of it is submerged in the water and part of it is above the water surface.

Fill your plastic container with water and place the hull into the water.

Does the boat float? _____

Now place a small 2 x 4 piece of wood (dry dock) on top of the boat.

What do you observe? _____

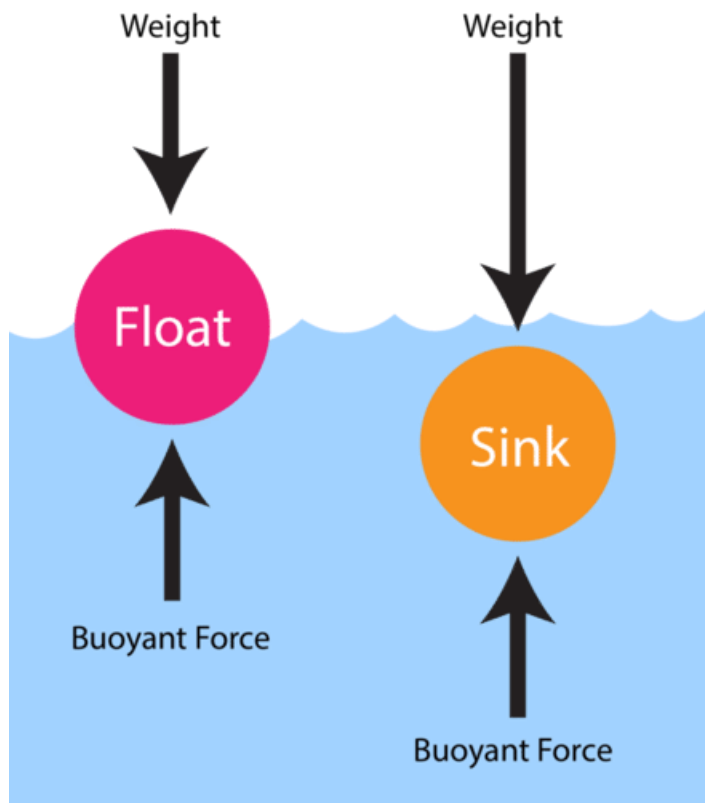
Later we will add motors, batteries, and other attachments to the boat, and we need to ensure that it does not sink. You should be able to place about 500 g more mass on top of the boat before begins to sink. There are bigger boats which weigh much more than ours ... and they do not sink! How come?

Bigger boats have a larger volume and displace much more water, which in turn can support more weight.

5.3 Buoyancy

Buoyancy is an upward force which makes a log float and birthday balloons fly. Even the balloons have a weight even if a small weight. **Gravity** acts on all objects and causes them to fall, even birthday balloons if they are filled with air. (Cold) air filled balloons do not have enough buoyancy to overcome their mass and the force of gravity pulling down on them and therefore do not float.

Hot air balloons and helium-filled balloons float in the air because they displace enough air to compensate their weight. The same holds for logs and boats, they also displace water.



Buoyancy can be explained by **Archimedes' Principle**: The (upward) buoyant force is equivalent to the weight of the fluid that would otherwise occupy the submerged volume of the object.

Consequently, a rock or a piece of solid steel also have a buoyant force. They just cannot float because they cannot displace enough water to compensate for their weight. This is because they are denser than water.

Most of our boat is made from wood and foam. Together with more dense motors, batteries, wires and screws, the whole boat floats because it displaces enough water to compensate for its total weight. If we add more weight, the boat will sink deeper in the water.

5.4 Load and Balance

The boat can support additional loads. The boat will eventually be fitted with a water sampler. When the sample tube is filled with water there will be an additional weight off-center, which principally throws the boat off balance.

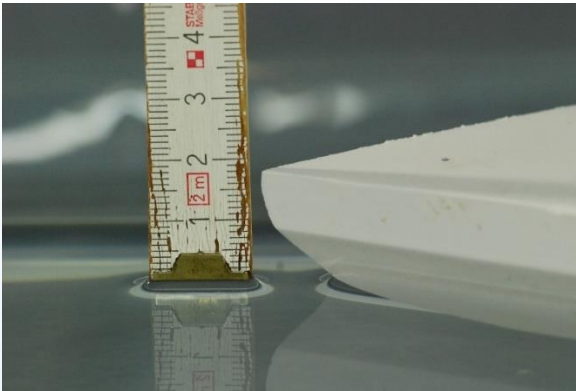
We can check the effect of additional loads on the buoyancy of the boat with a small 2 x 4 (**dry dock**). The dry dock is the additional 2 x 4 that will be used later as a stand to help with the assembly process.

Let's see how much **load** the boat can withstand and check its balance.



Try This: Boat Hull Load and Balance Test

1. Place the **hull** into the water and measure the distance from the waterline to the top of the **deck**. Record in *Data Table 1* the measurements in **millimeters (mm)**. Measurements should be taken from the middle of each part.
 - Measure the **bow** of the boat (the front).
 - Measure the **stern** of the boat (the rear).
 - Measure the **starboard side** of the boat (the right side when looking from the rear of boat).
 - Measure the **port side** of the boat (the left side when looking from the rear of the boat).



Waterline measurement of the boat hull without a load. This boat measures 18 mm at the bow.

2. Add an additional load (dry dock) to the **center** of the boat, measure, and record. Use the images on page 53 to help you with the placement of the dry dock.
 - Place the dry dock on top of the hull. Center the block as best you can.
 - Measure how far each of the points (bow, port, starboard, and stern) are now above the water surface.
 - Measure how far the bow, the port side (left), the starboard side (right) and the stern (rear) are out of the water.
 - Be sure to measure at the same places each time.
 - Record in *Data Table 1*.
3. Move the additional load (dry dock) to the **front** of the boat, measure, and record.
 - Push the dry dock toward about two inches from the center position toward the bow.
 - Take your four measurements.
 - If the bow is under water enter a negative number.
 - Record in *Data Table 1*.



Waterline measurement at the bow with load two inches in the front. This boat: 4 mm.



Try This: Boat Hull Load and Balance Test (continued)

4. Move the additional load (dry dock) toward the **rear (stern)** of the boat, measure, and record.
 - Push the block toward the stern until it is about two inches towards the stern from the center position.
 - Take your four measurements.
 - If the stern is under water enter a negative number.
 - Record in *Data Table 1*.

5. Move the additional load (dry dock) toward the **port side** of the boat, measure, and record.
 - Place the block back to the middle of the boat.
 - Then shift the block by 5 mm toward the port side (to the left).
 - Take the four measurements.
 - Enter a negative number if under the water.
 - Record in *Data Table 1*.



*Bow waterline measurement with load shifted towards the port side from the center of the boat.
(This boat: 14 mm)*

6. Place the additional load (dry dock) toward the **starboard side** of the boat, measure, and record.
 - Place the block back to the middle of the boat.
 - Then shift the block by 5 mm toward the starboard side (to the right).
 - Take the four measurements.
 - Enter a negative number if under the water.
 - Record in *Data Table 1*.

Data Table 1

Record measurements in mm	bow	port	starboard	stern*	stern (* + 7 mm)	pitch	roll
1. hull only							
2. dry dock in center							
3. dry dock in front							
4. dry dock in rear							
5. dry dock 5 mm left							
6. dry dock 5 mm right							

**The deck in the middle of the stern is lower than the main deck of the boat. We will consider this later – enter in this column the distance to the top of the wood in the center of the stern.*

Now we can calculate how much load the boat can take in one area before the performance is affected.

1. First, calculate the adjustment needed for the lower stern deck:
 - Add 7 mm to your measurements and enter this number in the second column for the stern measurement.
2. Calculate pitch for each of the five conditions.
 - Pitch = (bow number – stern number).
 - Enter the difference in the pitch column.
3. Calculate roll for each of the five conditions.
 - Roll = (port number – stern number) [the adjusted stern number!].
 - Enter this difference in the roll column.

What information do pitch, and roll provide? _____

Notice the two yellow shaded cells in the pitch column: if the load is toward the front, the front pitch number will be lower than the rear pitch number. This is what the pitch number tells you.

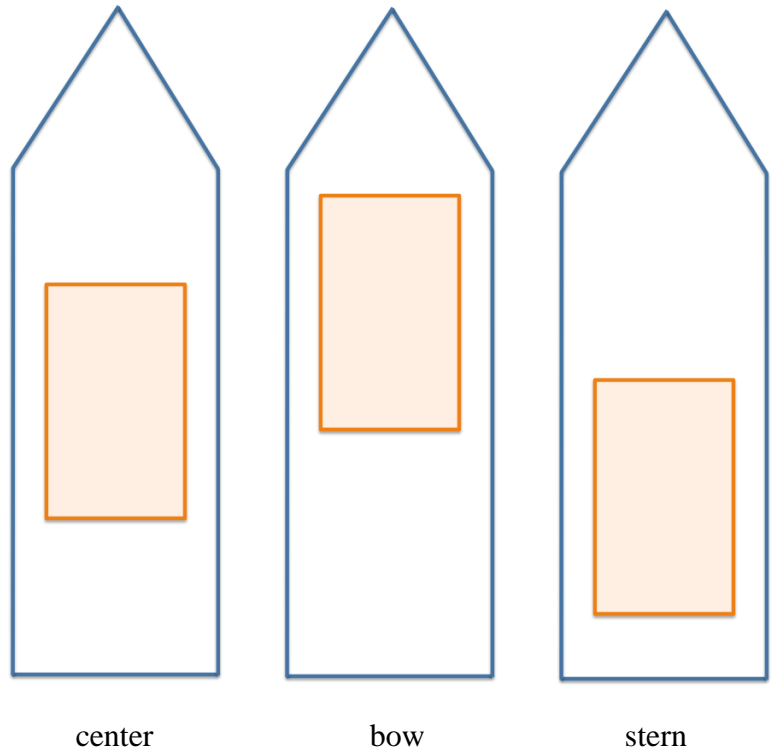
But may also notice that even without the dry dock block the pitch is a negative number meaning the hull has its nose down or front heavy. Do not worry! This is by design. Later the motors for the propellers and the rudder assembly will balance the boat.

Notice the two yellow shaded cells in the roll column: if the load is shifted to one side of the boat, the boat will roll (lean) towards that same side. These measurements (**pitch** and **roll**) give us an idea of how much any load may be out of balance before the performance of the boat will be affected.

Floatation and Balance Test Dry Dock Diagrams:

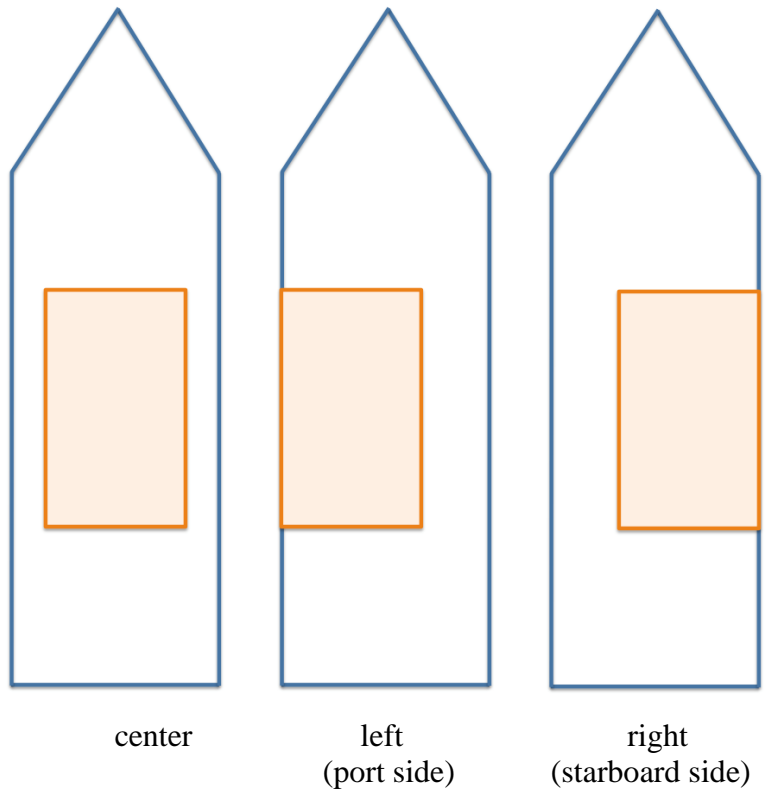
Testing Pitch

Small 2x4 (dry dock) on top of the floating hull simulation balance or unbalance of a load. The optimal position along the long axis of the boat is about two inches in front or behind the center position.



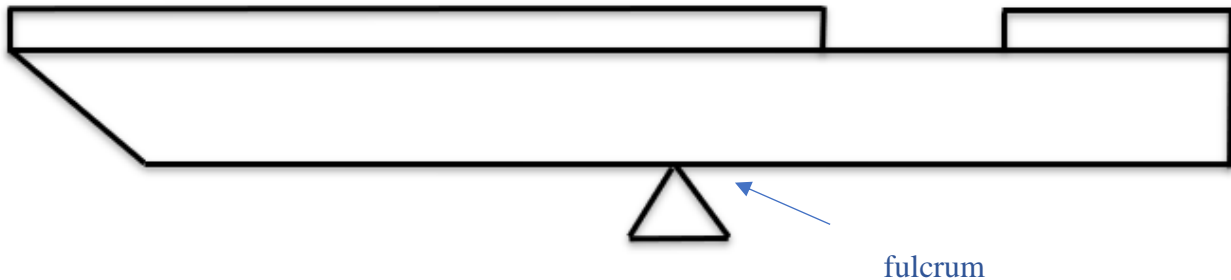
Testing Roll

Place the small 2x4 left or right of the center position. Choose a position at least 5 mm off center. Can you move the block such that one side lines up with the side of the boat without capsizing the boat?



5.5 Center of Gravity

We determine the center of gravity by balancing the boat on a fulcrum. The **center of gravity** of an object is the point at which weight is evenly dispersed and all sides are balanced.



Can you determine the center of gravity for the boat hull?

Using the pyramid or triangular shaped piece of wood, place the boat perpendicular to the **fulcrum**. A fulcrum is the point on which a lever rest and pivots.

Balance the boat on the fulcrum by shifting it back and forth until you find the place where it tips neither forward nor backward.

Measure the distance from the stern end of the boat to the fulcrum: _____ mm

Measure the distance from the bow tip to the fulcrum: _____ mm

Are the two measurements the same or different? Why? _____

As you add additional parts to the boat (motors, rudders, batteries, water sampler), do you think the center of gravity will change? Why? _____

The center of gravity of the boat is somewhere above the top line of the fulcrum, in the middle between the port and starboard sides.



Supplemental Resources: Why is my boat designed this way?

Strength of Ships

- <https://www.britannica.com/technology/naval-architecture/Strength-of-ships>

Why Do Ships Float?

- <https://letstalkscience.ca/educational-resources/stem-in-context/why-do-ships-float>

The Basics of Boat Hull Design

- <https://www.powerandmotoryacht.com/boat-design/the-basics-of-hull-design-explained>

Boat Hull Design Concepts

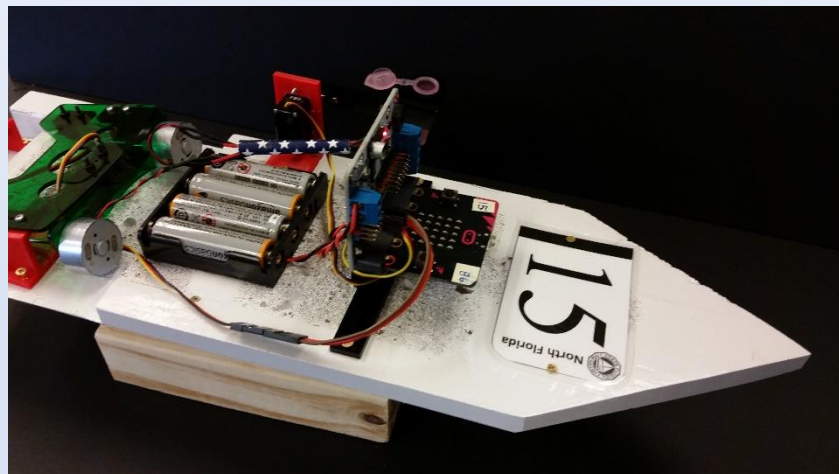
- <https://youtu.be/cQiY5rea0kA>

Buoyancy Science

- <https://untamedscience.com/buoyancy-science/>

Buoyant Boats (Hands-on Activity – Teach Engineering)

- https://www.teachengineering.org/activities/view/duk_boat_mary_act



Module 6: Boat Assembly

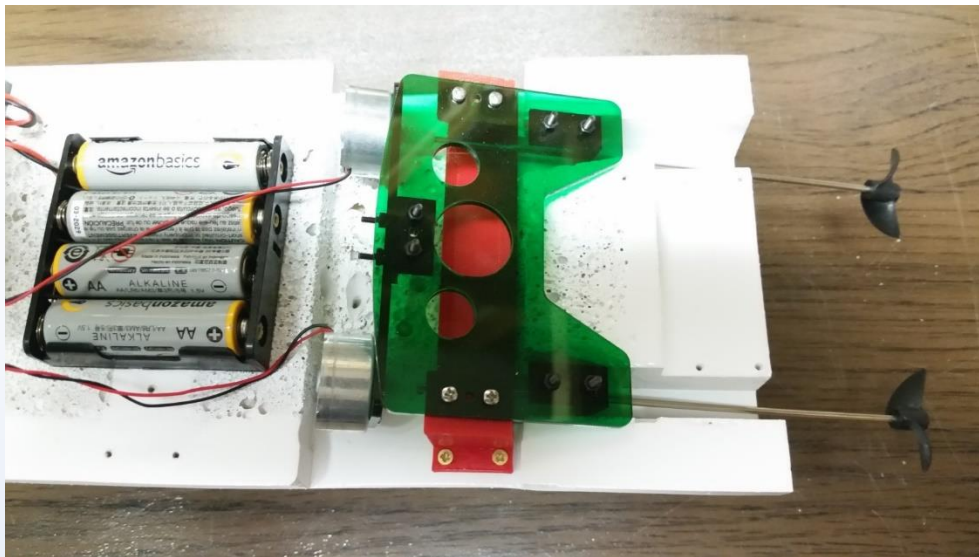
6.1 Twin Motor Assembly

Now it is time to begin assembling some of the parts that will attach to the hull of the boat. We will first assemble the twin motors, shafts, and propellers on the motor mount.



Assembly: Twin Motors

1. Refer to the image on page 12 to see which parts you will need for this assembly.
2. Watch the **STEM SEALS YouTube video: SEA Twin-Motor Assembly**



Stern Closeup with Motors Installed

3. Step-by-step twin motor assembly:
 - a. Attach the two green acrylic plates to each other using the wide bracket.
 - b. Attach the two long brackets for guiding the propeller shafts.
 - c. Fasten the DC motors to the motor frame.
 - d. Attach the white worm gears to the motor shafts.
 - e. Attach the propellers to the shafts: take great care not to bend the shafts! There are small numbers on the propellers - #1 is right / starboard, #2 is left / port.
 - f. Feed the propeller shafts through the long brackets and attach them to the white worm gears.
 - g. Attach the motor mounting bracket to the twin motor assembly.
 - h. Attach the motor assembly to the hull.

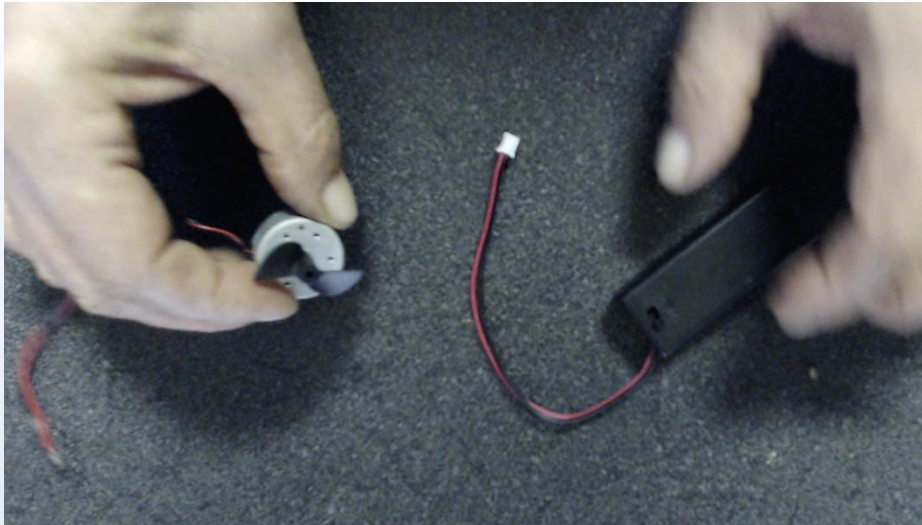
6.2 Motor Test

It is important to test the functionality of the motors before moving on and adding more parts to the boat. We want to be sure the motors work.



Try This: Testing the Twin Motors and Propellers

1. Use the battery case with the 2 AAA batteries inserted.
Make sure the batteries are inserted correctly and leave the cover off.
2. Take the red and black wires from one motor in one hand and the white battery connector in the other.
3. Simultaneously stick both motor wires (the bare metal ends) into the two tiny holes at the end of the connector. Be sure to place one wire in one hole and the other wire into the other hole of the battery connector. Keep the wires from touching each other at the bare ends.



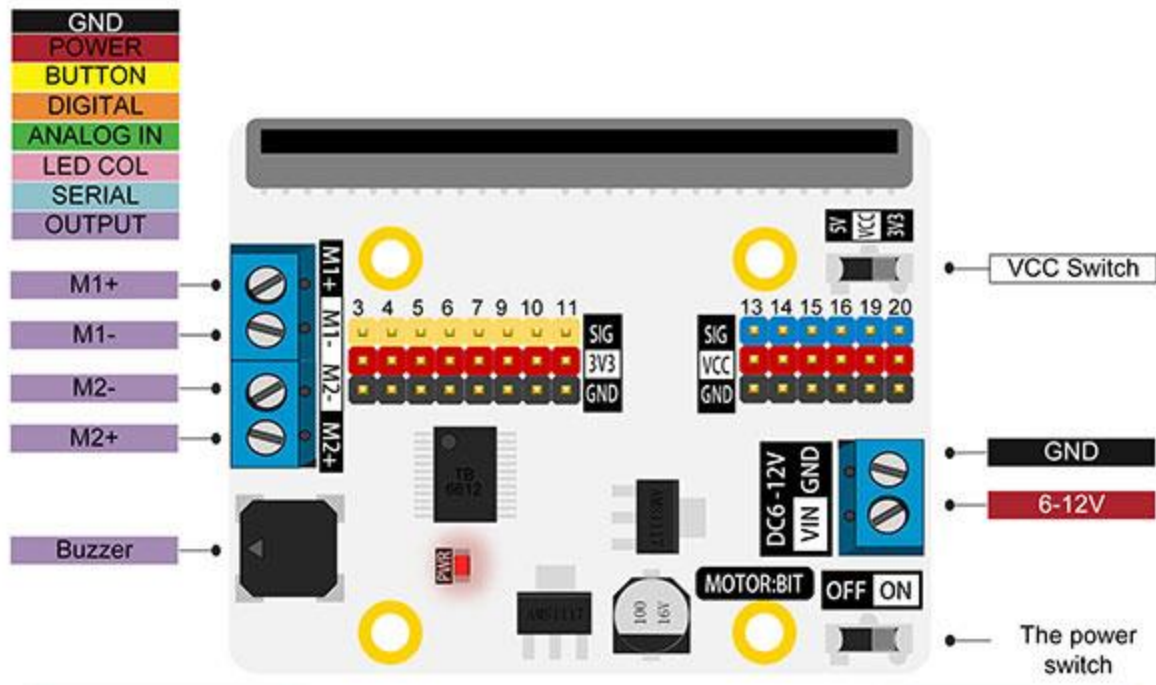
4. Does the propeller turn?
 - Can you tell which direction? You may need to try it multiple times to determine.
 - By looking at the propeller decide: Will this cause forward or backward motion of the boat?
5. Reverse polarity. (Reverse the red and black wire)
 - Which direction do the propellers rotate now?
6. Repeat the same test with the other motor.

Note that the propellers are different. The motors should rotate in the same direction. But one propeller will drive the boat forward, the other one backward. We will adjust for this later when we connect the wires to the motor:bit board.

6.3 Assembly of the Motor:bit Mount with Micro:bit

Now we will assemble the motor:bit mount and the motor:bit. The motor:bit is a kind of motor drive board that has a terminal that the micro:bit fits into which can drive two DC motors (our twin motors). It also has integrated series' sensor connectors where we will plug in various sensors. We will also need to add a battery supply to the boat.

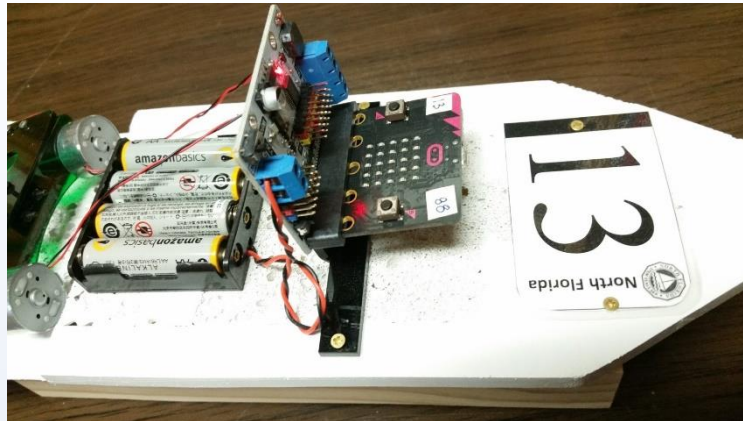
ELECFREAKS MOTOR:BIT V1.6





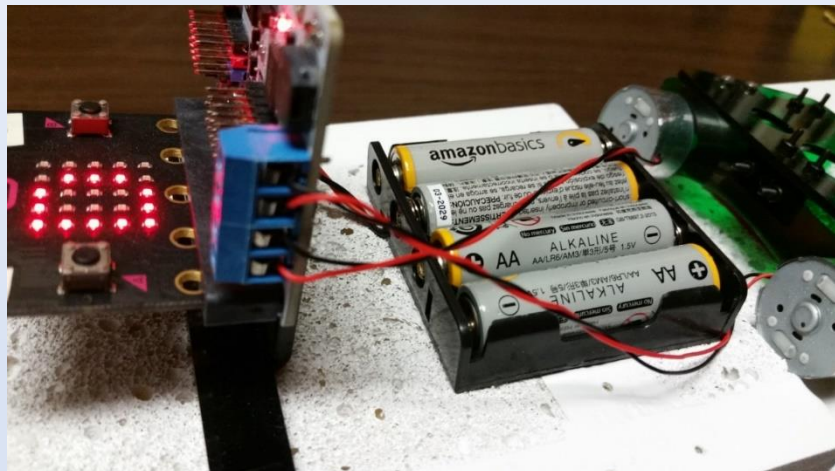
Assemble: Motor:bit and Micro:bit (continued)

1. Refer to the images on pages 15 and 17 to identify the parts needed for this assembly.
2. Watch the **STEM SEALs YouTube video: SEA Motor:bit Assembly**.
3. Step-by-step assembly instructions:
 - a. Using two (M3) screws attach the motor:bit to the motor:bit bracket.
 - b. Stick the micro:bit into the moto:rbit's terminal connector (LEDs facing up).
 - c. Using two (#4 x 1/2") brass screws fasten the battery case to the hull. Ensure that the leads come out in the front of the case.
 - d. Fasten motor:bit and micro:bit to the hull using two (#3 x 5/8") brass screws.
 - e. Connect the battery leads to the motor:bit. (Red wire in the top terminal- see image)



Motor:bit/micro:bit mount, battery case and twin motors mounted to hull, seen from starboard side (right)

- f. Connect the motor leads to the motor:bit. (See image)



Motor:bit/micro:bit mount, battery case and twin motors mounted to hull, seen from port side (left). Notice: starboard motor wires are at the bottom of the terminal on the motor:bit board, port motor wires at the top – and both have the red wire at the bottom and the black wire at the top.

Module 7: Boat 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 boat. The first code will be for boat **propulsion** which is the mechanism or system used to generate thrust to move the boat forward or across the water.

7.1 Propulsion

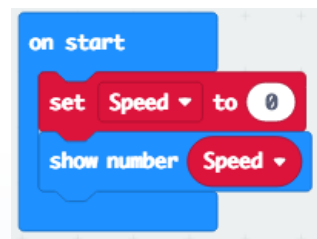


Time to Code: Propulsion

1. Open MakeCode on your computer, go to MakeCode Home and click “New Project”.
 - Name it “**Propulsion**”

2. The “**on start**” block:

- Click on the “Variable” toolbox
 - Make a Variable
 - Name it “Speed”
- Drag the “set Speed to 0” block into the “on start” block.
- Drag the “show number 0” block from the “Basic” toolbox after the “set Speed” block.
 - Get the “Speed” variable from the “Variables” toolbox and drag it over the number zero in the “show number 0” block.



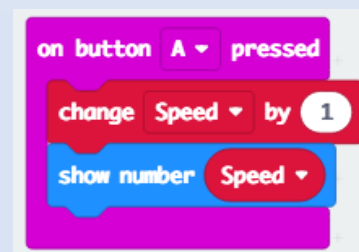
3. The “**forever**” block:

- Click on “Advanced” then “Pins”
- Drag the “analog write pin P0 to 1023” block into the “forever” block
 - Change P0 to P1 – the left motor of the boat is connected to pin P1.
 - Click on the “Math” toolbox and drag a “0 x 0” open block over the number 1023 in the “analog write ...” block.
 - Change the first zero to 100.
 - Get the “Speed” variable (from “Variables”) and drag it over the second zero in the Math multiplication block.



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

- Drag the “on button A pressed” block from “Input” onto the workspace.
 - It does not matter where– let’s say right underneath the “on start” block – not in the “on start” block!
- Drag a “Change Speed by 0” block from “Variables” into the “on button A pressed” block.
- Copy the “show number Speed” block in “on start” and paste and drag into the “on button A pressed” block.



The complete code should look like this:

```
on start
  set Speed to 0
  show number Speed

forever
  analog write pin P1 to 100 x Speed

on button A pressed
  change Speed by 1
  show number Speed
```

https://makecode.microbit.org/_DVjMkjUTHWdJ (Propulsion)



Try This: Test Propulsion (Speed)

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

1. Connect the micro:bit to the computer using the USB connector and download the “**Propulsion**” code.
2. After downloading, insert the micro:bit into the motor:bit on the boat.
3. Set the boat on the dry dock and turn the motor:bit on.
4. When the micro:bit starts, Speed is at zero and the forever loop writes 100 times 0 to pin P1 and the motor does not move.
5. Press button A.
 - Speed changes from zero to 1 and now the forever loop writes 100 to P1. The motor may move or not. Be sure the motor:bit is on.
6. Press button A again.
 - Speed changes from 1 to 2. 200 is written to P1 and the motor should slowly turn.
7. Press button A again.
 - Each time you press button A, Speed will be increased by 1 and the motor should turn faster.
 - Once Speed is larger than 10, writing numbers larger than 1023 does not affect the motor anymore and the propeller speed does not increase anymore.

What observations did you make during the speed test?

At what Speed variable did the propeller start to turn? _____

Were you able to see what direction the propeller was turning? _____ Left, or right? _____

7.2 Propulsion 2

It would be helpful if we could increase and decrease the speed of the boat!

We can accomplish this by using another button: button B.

Let's modify the code to make some improvements and add features.



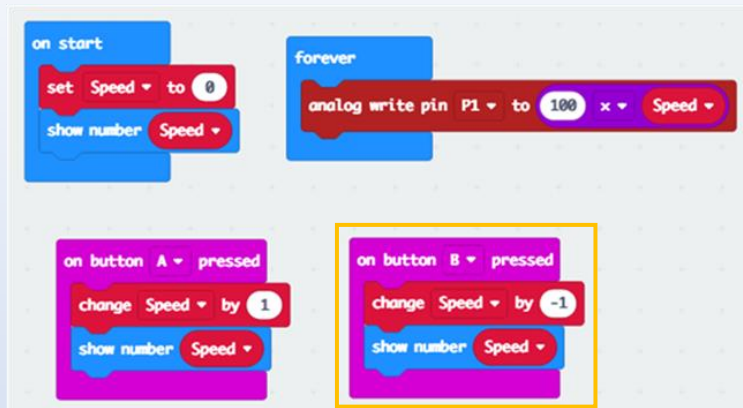
Tip: Modifying Code

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 2

1. Open the **Propulsion** project (save or duplicate – see note above).
2. Rename the project “**Propulsion 2**”
3. Create an “**on button B pressed**” block:
 - Copy the “on button A pressed” block and paste in into the workspace.
 - Notice the block turns grey – this is to alert you that you have a duplication in coding and the micro:bit would not know what to do with two “on button A pressed” blocks.
 - Change the “A” to a “B” in this new block.
 - Change the number 1 in the “on button B pressed” block to a negative 1 (-1).



Download on the micro:bit and test out this modified code.

Can you increase speed (button A) and decrease speed (button B)?

Did you wonder why only one propeller was turning?

Of course, we want both propellers to work. Remember in the forever loop we only wrote code to the pin P1 which controls the left motor. Pin P2 controls the right motor.

Also, if you tested your boat to Speed = 10 (the maximum speed for the motors), you may have noticed that the number 10 scrolls across the LED screen and can be difficult to read.

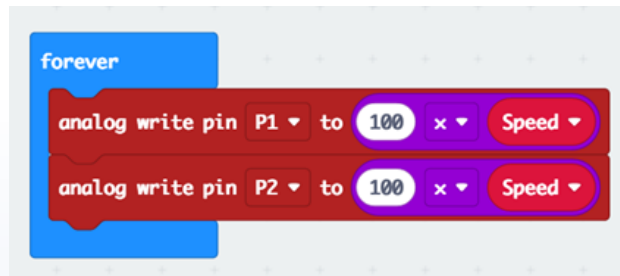
Let's fix that and make both propellers work!



Time to Code: Propulsion 2 (continued)

3. Modify the “forever” block:

- Copy the “analog write pin...” block in the forever block and paste it right under the one you copied.
 - Change the new block P1 to P2.
- The right motor is connected to the P2 pin on the motor:bit.
- Now both propellers should work.

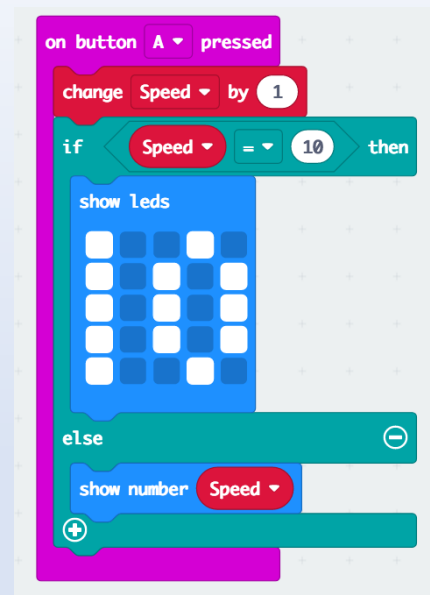


4. Test out the new code to see if both propellers work.

Now let's fix the scrolling 10!

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

- Click on the “Logic” toolbox and drag a “if true then else” block and drop it after the “change Speed by 1” block in the “on button A pressed” block.
 - Drag a “0 = 0” comparison block into the “if ...” block and place it over the true
 - Drag a “Speed” variable over the first zero 0
- Click on the “Basic” toolbox and drag the “show leds” block and drop it into the first slot of the “if ...” block.
 - By clicking on the small squares, you can turn on any LED you like.
 - Create the number “10”.
- Drag the “show number Speed” block into the second slot of the “if ...” block after the “else”.

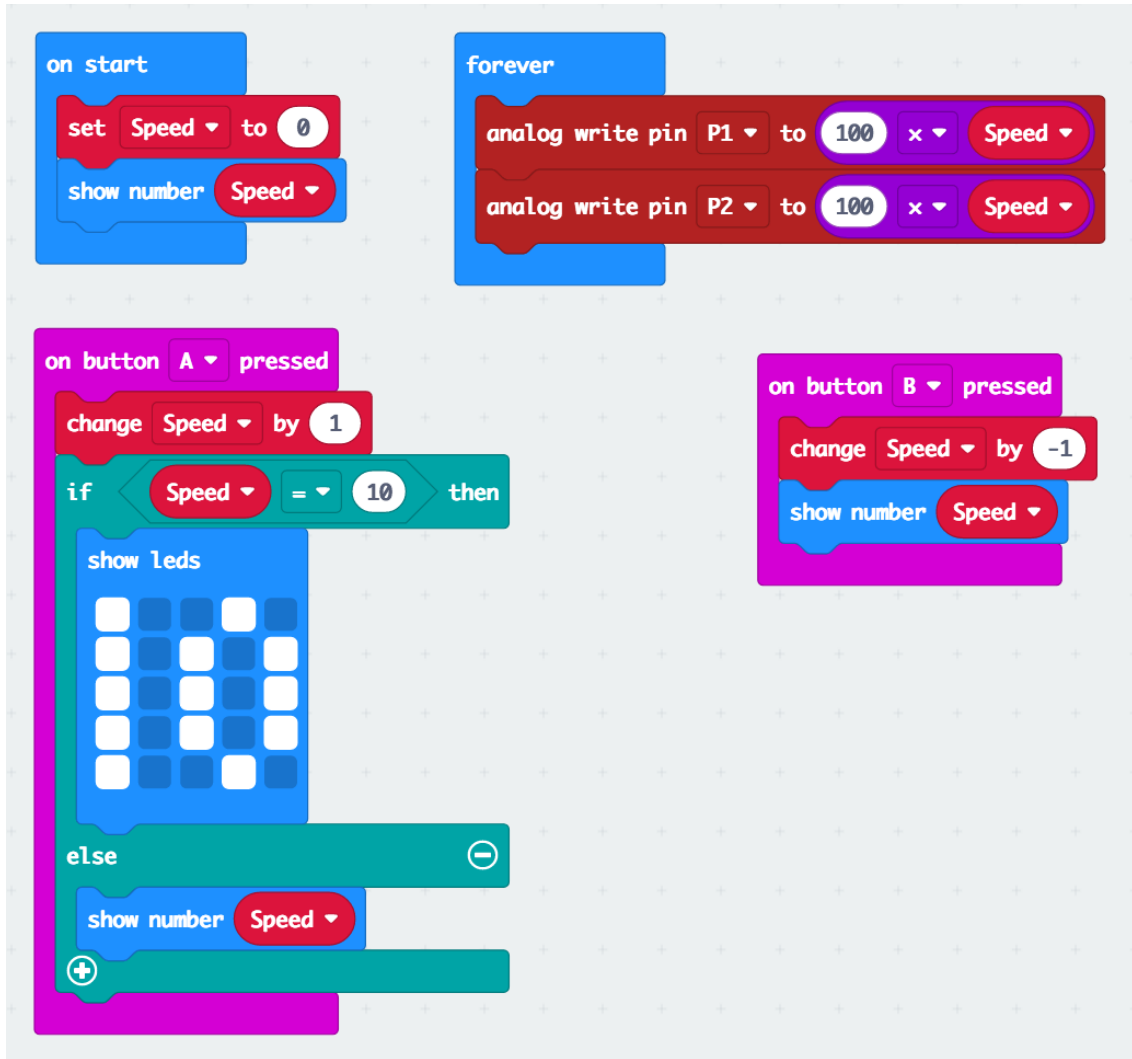


6. Change the name of this project to “Propulsion 2” if you did not at the beginning.

7. Save and upload the code to the micro:bit.

8. Insert the micro:bit into the motor:bit and test the code with the boat.

The complete code should look like this:



https://makecode.microbit.org/_deC2gV3MHDUK (Propulsion 2).

7.3 Adding a Remote Control (Propulsion 3 / Remote Control 1 & 2)

With the current code, the boat would still need an onboard captain to operate the buttons!

Wouldn't it be nice to have a remote control?

The micro:bit comes with built-in radio features which can be used to relay messages from another micro:bit thus allowing us to make a remote control. One micro:bit will be on board the boat and another micro:bit can be used from shore to relay messages to the micro:bit on the boat.



Internet Resources: Radio 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](#)

To create the code, we will first modify the “**Propulsion 2**” code for the boat so that it can receive messages from a remote control micro:bit.

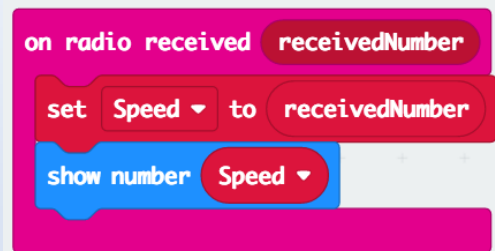
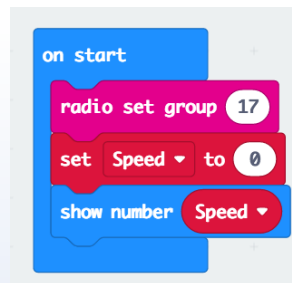


Time to Code: Propulsion 3

1. Open the “**Propulsion 2**” project and change the name to “**Propulsion 3**”.
2. Modify the “**on start**” block:
 - Drag a “radio set group 1” block from the “Radio” toolbox to the beginning of the “on start” block.
 - Change the number 1 to the ID number of your boat.
3. Create an “**on radio received**” block:

Drag an “on radio received receivedNumber” block from the “Radio” toolbox onto the workspace.

 - Drag a “set Speed to 0” block from the “Variables” toolbox into the “on radio received ...” block.
 - Drag the “receivedNumber” variable from the “on radio received receivedNumber” block (it duplicates itself) over the zero in the “set Speed...” block.
4. Be sure to save this code and then load it onto the micro:bit you will insert into the motor:bit on the boat.



The complete code for “**Propulsion 3**” looks like this:

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

```
on start
  radio set group 17
  set Speed to 0
  show number Speed

forever
  analog write pin P1 to 100 x Speed
  analog write pin P2 to 100 x Speed

on button A pressed
  change Speed by 1
  if Speed = 10 then
    show leds
  else
    show number Speed

on button B pressed
  change Speed by -1
  show number Speed

on radio received receivedNumber
  set Speed to receivedNumber
  show number Speed
```

https://makecode.microbit.org/_1LWDgzaH4civ (**Propulsion 3**)

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



Time to Code: Remote Control 1

- Use the “**Propulsion 3**” code and modify it for the remote control.
 - Be sure to use a saved URL or copy and duplicate the Propulsion code otherwise you will change the code you will need later for the boat.
- Rename this project “**Remote Control 1**”.
- Modify the **forever loop**:
 - Delete the two “analog write pin ...” blocks.
 - Drag a “plot x 0 y 0” block from “Led”.
 - Change the first zero to 4.
 - Drag a “pause (ms) 100” block from “Basic”.
 - Drag an “unplot x 0 y 0” block from “Led”.
 - Change the first zero to 4.
 - Copy the “pause ...” block and place at the end.
- Delete the “**on radio received ...**” block.
- Modify the “**on button A pressed**” block:
 - Drag a “radio send number 0” from “Radio” to the end of the “on button A pressed” block.
 - Copy the “Speed” variable and drag over the other zero in the last block.
- Modify the “**on button B pressed**” block:
 - Copy the “radio send number Speed” block from the “on button A” block and drop at the end of the “on button B pressed” block.
- Save this code on the second micro:bit (the one which is not connected to the motors/motor:bit).

```

forever
  plot x 4 y 0
  pause (ms) 100
  unplot x 4 y 0
  pause (ms) 100
  
```

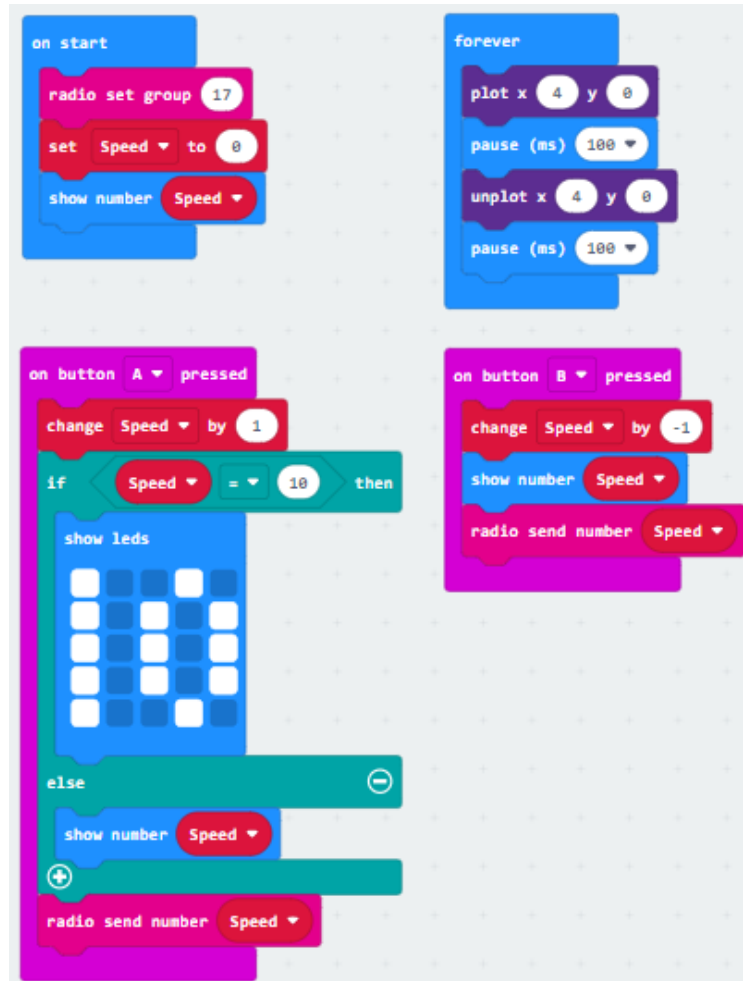
```

on button A pressed
  change Speed by 1
  if Speed = 10 then
    show leds
  else
    show number Speed
  radio send number Speed
  
```

```

on button B pressed
  change Speed by -1
  show number Speed
  radio send number Speed
  
```

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



The image shows four Scratch code blocks for a remote control project. The 'on start' block contains: 'radio set group 17', 'set Speed to 0', and 'show number Speed'. The 'forever' loop contains: 'plot x 4 y 0', 'pause (ms) 100', 'unplot x 4 y 0', and 'pause (ms) 100'. The 'on button A pressed' block contains: 'change Speed by 1', an 'if Speed = 10 then' block with 'show leds' (a 5x5 grid of blue squares), and an 'else' block with 'show number Speed' and 'radio send number Speed'. The 'on button B pressed' block contains: 'change Speed by -1', 'show number Speed', and 'radio send number Speed'.

https://makecode.microbit.org/_0K68FAhUMWm9 (Remote Control 1)



Try This: Remote Control Test

1. Turn the motor:bit on the boat on. Make sure the boat micro:bit is inserted into the motor:bit.
 - The boat micro:bit should show “0”.
2. Connect the remote control micro:bit to a power supply. It is probably still connected with USB cable to the computer. If not, connect it to the battery case.
 - This micro:bit also shows a “0” with a blinking dot in the upper right corner.
3. Press button A on the remote control: It shows “1”
 - Look at the boat! That micro:bit also shows “1” and the motors might begin to turn.
4. Press button A again on the remote control: the boat also displays “2” and the propellers should turn faster.
 - With buttons A and B, you have full remote control over the speed of the boat.
 - Remember button A increases the speed and button B decreases the speed.
5. Use the remote control to turn the motors off! (Decrease Speed to zero.)

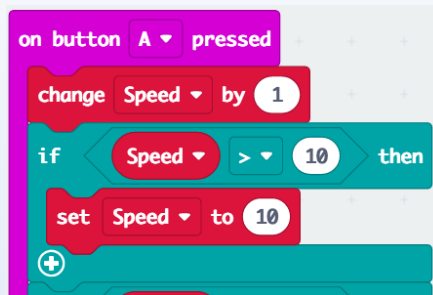
Remember the issue of scrolling number “10” in the propulsion code? We now have the same issue with the remote control. Also, any numbers above 10 produce invalid Speed numbers.

Let’s fix these!



Time to Code: Remote Control 2

1. Use the “**Remote Control 1**” project.
 - Rename “**Remote Control 2**”.
2. Modify the “**on button A pressed**” block:
 - Drag an “if true then” block from the “Logic” toolbox right below the “change Speed by 1” block and before the other “if...” block.
 - Drag a comparison block from “Logic” and drop in place of the “true” in the “if ...” block.
 - Get the variable “Speed” and drop it in place of the first zero in the comparison block.
 - Change the comparison sign (if needed) to a greater than sign (>).
 - Write the number 10 in place of the second zero.
 - Drag a “set Speed to 0” block (from “Variables”) and drop it into the first slot in the “if ...” block; change the zero to 10.



3. Upload the code to the (remote control) micro:bit.
 - You can test this attached to the computer and without the boat.
4. Test your code:
 - Press button A and look at the display!
 - The numbers are displayed as before ... until you reach 10:
 - Is the “10” scrolling now?
 - Press A again: Do you get numbers larger than 10? Why not?



Time to Code: Remote Control 2 (continued)

Let's fix the similar problem with negative 10's as well:

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

- Copy, paste and drag both “if ... “blocks from the “on button A pressed” into the “on button B pressed” block.
- In the first “if ...” block we need to change the comparison statement to “Speed < -10”.
- Change the 10 in the “set Speed to 10” block to a negative 10.
- Change the 10 in the second “if ...” block to a negative 10.
- Change some of the LEDs and make the zero smaller, like a lowercase “o” to indicate this is not a 10 (as in the “on button A pressed” block) but a negative 10. We do not have enough space on the LED matrix to write a full “-“sign.

Upload and test your code. It should prevent the “10’s” from scrolling. It also does not allow Speed to get larger than 10 or less than negative 10 (the maximum Speed value).

```
on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
  show leds
  else
    show number Speed
  radio send number Speed
```

The complete “Remote Control 2” code:

```
on start
  radio set group 17
  set Speed to 0
  show number Speed

forever
  plot x 4 y 0
  pause (ms) 100
  unplot x 4 y 0
  pause (ms) 100

on button A pressed
  change Speed by 1
  if Speed > 10 then
    set Speed to 10
  show leds
  else
    show number Speed
  radio send number Speed

on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
  show leds
  else
    show number Speed
  radio send number Speed
```

https://makecode.microbit.org/_Cq70uwd1d4io (Remote Control 2)

7.4 Measuring Propulsion (Thrust)

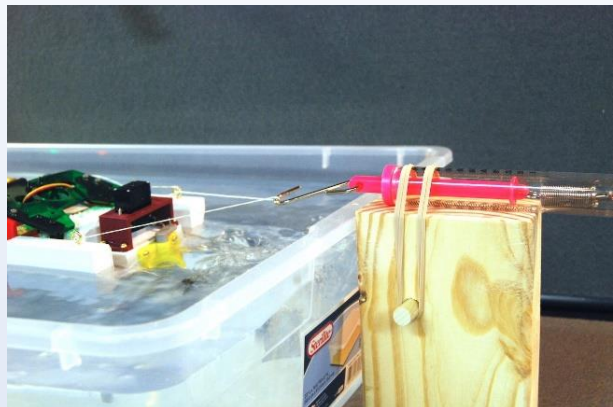
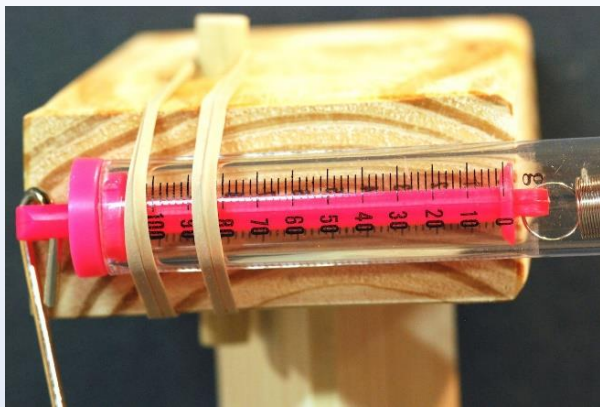
It is time to put the boat in the water! You have tested to see if your motors and propeller's function and created a remote to be able to control the boat without having to physically touch the boat after placing it in the water. Before testing on a large body of water, you want to be sure the propellers can push the boat through the water by measuring its thrust. **Thrust** is a reaction force described quantitatively by Newton's third law. A boat generates thrust (or reverse thrust) when the propellers are turned to accelerate water backwards (or forwards). The resulting thrust pushes the boat in the opposite direction to the sum of the momentum change in the water flowing through the propeller.



Try This: Propulsion Test

Materials: Large container, 100 g spring scale, spring scale support, string, boat, remote control

1. Fill the container with water up to about one inch below the top.
2. Set the boat in the water.
3. Tie the string to the two rear eyehooks on stern of boat.
 - You may have to install the eye hooks. (see photo)
4. Secure the 100 g tube spring scale horizontally on the support with a rubber band and the dowel inserted into the dry dock. (see photo)
 - Ensure the gram scale is up and you can read it.
 - Test if the spring scale reads zero when nothing is attached to the metal hook. The scale indicator (the washer-type looking wider part at the top) should be within 2 or 3 grams of the zero mark of the scale. To adjust rotate the nut at the top of the scale.



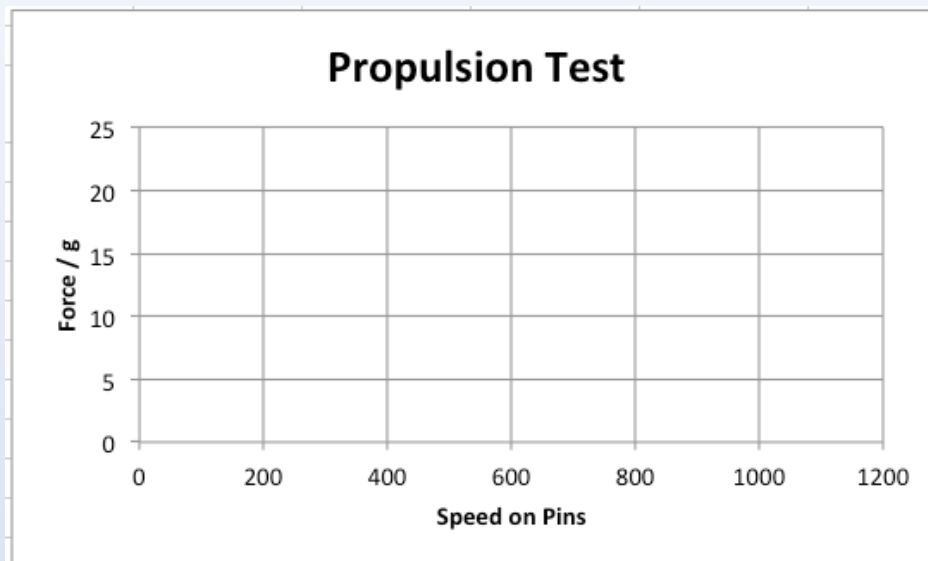
5. Using the remote control micro:bit, start the engines.
 - Begin with Speed = 4
 - Remember that is the displayed number on the micro:bit.
 - The number for the motors is really 400.
6. Read the pulling force of the boat on the scale.
7. Enter the value in Data Table 2.
8. Increase speed step by step and enter the pulling force in the Data Table.



Try This: Propulsion Test (continued)

	Speed (on pins)	Force / g
S = 4	400	
S = 5	500	
S = 6	600	
S = 7	700	
S = 8	800	
S = 9	900	
S = 10	1000	

Make a graph of your data. The independent variable Speed is the horizontal axis; the dependent variable on the vertical axis is the pulling force. Note that gram is a unit of mass not force but we can measure the force equivalent to a specific mass in Earth's gravitational field.



- The motors turn faster the larger the Speed variable is. You would expect the force be stronger at larger speeds. How can you see this in your diagram? _____

- The motors have a difficult time or do not turn at all at low values of Speed. Does your diagram reflect this? _____
- If not, what needs to be done to add this information to your data? _____

7.5 Reversing the Engines (Propulsion 4)

It would be useful if the boat could go both forward and in reverse.

We can modify the code to allow the boat to sail forward or backward.



Time to Code: Adding Reverse Engines (Propulsion 4)

1. Open the “**Propulsion 3**” project and rename it “**Propulsion 4**”.
2. Modify the “**on-start**” block:
 - Create a new variable “EnginesInReverse”.
 - Drag the “set EnginesInReverse to 0” block and drop it after the “set Speed to 0” block.
 - Click on the “Logic” toolbox and drag the “false” value from the bottom and drop it over the zero in the “set EnginesInReverse to 0” block.
 - The “EnginesInReverse” is not a number! It is the value “false” because at the beginning we want to start with engines NOT in reverse and we will change that later as appropriate.
 - Drag a “digital write pin P0 to 0” block from the “Pins” toolbox (under “Advanced”) and drop it after the “set EnginesInReverse to false” block.
 - Change the pin P0 to P8.
 - Copy this new block, paste, and drop it after the first one.
 - Change pin P8 to P12.
 - The micro:bit pins P8 and P12 control the polarity of the motors. Setting them both to zero means “drive forward”.

```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
```



Time to Code: Propulsion 4 (continued)

3. Modify the “forever” loop:

- Drag an “if true then else” block from “Logic” and drop it at the beginning of the forever loop.
 - From “Variables” get the “EnginesInReverse” variable and drop it onto the “true” in the “if ...” block.
 - Copy the two “digital write pin ...” blocks in the “on start” block, paste and drop them in the first slot of the “if EnginesInReverse ...” block.
 - Change the zero in both to the number 1.
 - Now drag the two existing “analog write pin ...” blocks into the first slot in the “if EnginesInReverse ...” block and place after the “digital write pin” blocks.
 - Change the number 100 in the multiplication of both “analog write pin” blocks to -100 (negative 100).
- In the “else” slot:
 - Copy and paste all four blocks from the first slot of the “if ...” block.
 - Change the number -100 (negative 100) in the multiplication of both “analog write pin ...” blocks in the else slot to number 100 (positive 100).

```
forever
  if EnginesInReverse then
    digital write pin P8 to 1
    digital write pin P12 to 1
    analog write pin P1 to -100 x Speed
    analog write pin P2 to -100 x Speed
  else
    digital write pin P8 to 0
    digital write pin P12 to 0
    analog write pin P1 to 100 x Speed
    analog write pin P2 to 100 x Speed
```

- Check to make sure: the “digital write pin ...” blocks come in pairs setting both, P8 and P12 because we want both propellers switched to reverse and back.
- Check to make sure: the “analog write pin ...” blocks come in pairs setting both, P1 and P2 because we want to have both propellers the same speed.

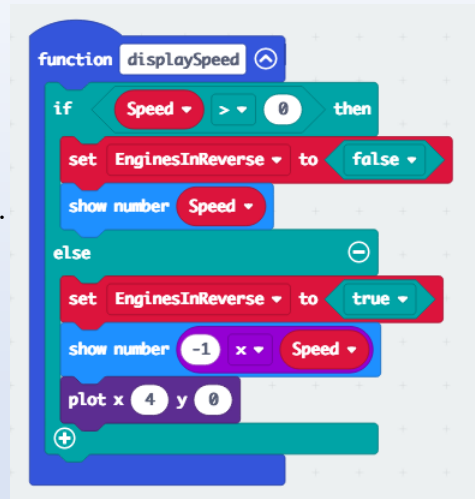
Before you continue, it is worth noting that you will be using a **function** in the next part of the code. A function lets you create a portion of code that you can reuse in your program. Instead of having to copy the same code in many places in your program, you can just call the function name in a single block. **When using a function, a new edit box will appear- just click done. In the workspace, type the name of the function where it says “doSomething” and you are all set.** The name of the function will now be listed in the functions toolbox for you to call up as needed.



Time to Code: Propulsion 4 (continued)

4. Create a **function**:

- Click on the “Functions” toolbox.
- Make a function and name it “displaySpeed”.
 - A new blue “function displaySpeed” block appears on your workspace toolbox in functions.
- Get a new “if true then else” block from “Logic” and drag it into the “function ...” block.
 - Click on “Logic” and drag a comparison “0 < 0” block and drop it over the “true” in the “if ...” block.
 - Get the “Speed” variable from “Variables” and drop it over the first zero in the comparison block.
 - Change the sign (if needed) of the comparison to larger than zero, “>”.
 - Copy the “set EnginesInReverse to false” block from the “on start” block and paste it in the **first** slot of the “if ...” block.
 - Copy a “show number Speed” block from somewhere (it doesn’t matter) and drop it after the “set EnginesInReverse ...” block.
 - Copy the “set EnginesInReverse to false” block and paste into the **second** slot of the “if ...” block. Change “false” in this block to “true”.
 - Copy the “show number Speed” block and drop after the “set EnginesInReverse to true” block.
 - Drag the “Speed” variable outside of this block.
 - Get a multiplication from “Math” and place it instead of the “Speed” onto the zero in the “show number 0” block.
 - Change the first zero in the multiplication to a negative one, -1.
 - Drag the “Speed” variable you just placed outside the block back in over the second zero.
 - Get a “plot x 0 y 0” block from “Led” and drop after the last “show number -1 * Speed” block and change the first zero to 4.



As you see, the function checks if Speed is positive. If so it sets “EnginesInReverse” to false because with positive “Speed” we want the boat go forward.

The if else, meaning if Speed is zero or negative sets “EnginesInReverse” to true. But then we still want to see how fast the boat goes backward. Instead of displaying a negative number (which the micro:bit can do but it takes a long time to do that), we display the negative of the negative number, which is a positive number again; and to indicate the reversal we show a dot in the upper right corner of the micro:bit display, which is done by the “plot x 4 y 0” block.



Time to Code: Propulsion 4 (continued)

Now we can continue modifying our code:

5. Simplify the “on button A pressed”, “on button B pressed”, and the “on radio received” blocks by using the function “call displaySpeed”:
 - Delete the “show number Speed” and the “if...” blocks in the “on button A pressed” block.
 - Click on the “Functions” toolbox and drag the “call displaySpeed” block and place it after the “change Speed by 1” block.
 - Replace the “show number ...” block in the “on button B pressed” block as well.
 - Also replace the “show number ...” block in the “on radio received receivedNumber” block.

```
on button A pressed
  change Speed by 1
  call displaySpeed
  show number Speed
```

```
on button B pressed
  change Speed by -1
  call displaySpeed
  show number Speed
```

```
on radio received receivedNumber
  set Speed to receivedNumber
  call displaySpeed
  show number Speed
```




Time to Code: Propulsion 4 (continued)

Speeds larger than 10 or smaller than -10 do not make sense: the micro:bit cannot use the pins P1 and P2 for numbers larger than 1023 or smaller than -1023.

Let's do *input validation*:

6. Change the “if then else” block in the function “displaySpeed” block to

- “if Speed \geq 10 then
 - set Speed to 10” (don't let it get bigger!),
 - “set EnginesInReverse to false”
 - “show image” the special “10”
- “else if Speed \geq 0 then
 - set EnginesInReverse to false
 - unplot x 4 y 0
 - show number Speed”
- “else if Speed $>$ -10 then
 - set EnginesInReverse to true
 - show number (-1 * Speed)
 - plot x 4 y 0”
- “else
 - set Speed to -10
 - set EnginesInReverse to true
 - show image”, make a special image which you can recognize as a negative ten.

```
function displaySpeed
  if Speed >= 10 then
    set Speed to 10
    set EnginesInReverse to false
    show leds
  else if Speed >= 0 then
    set EnginesInReverse to false
    unplot x 4 y 0
    show number Speed
  else if Speed > -10 then
    set EnginesInReverse to true
    show number -1 * Speed
    plot x 4 y 0
  else
    set Speed to -10
    set EnginesInReverse to true
    show leds
```

Good Job! You have completed the “**Propulsion 4**” coding!

Your complete code for the boat micro:bit should look like this:

```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed

function displaySpeed
  if Speed >= 10 then
    set Speed to 10
    set EnginesInReverse to false
    show leds
  else if Speed >= 0 then
    set EnginesInReverse to false
    unplot x 4 y 0
    show number Speed
  else if Speed > -10 then
    set EnginesInReverse to true
    show number -1 * Speed
    plot x 4 y 0
  else
    set Speed to -10
    set EnginesInReverse to true
    show leds

forever
  if EnginesInReverse then
    digital write pin P8 to 1
    digital write pin P12 to 1
    analog write pin P1 to -100 * Speed
    analog write pin P2 to -100 * Speed
  else
    digital write pin P8 to 0
    digital write pin P12 to 0
    analog write pin P1 to 100 * Speed
    analog write pin P2 to 100 * Speed

on button A pressed
  change Speed by 1
  call displaySpeed

on button B pressed
  change Speed by -1
  call displaySpeed

on radio received receivedNumber
  set Speed to receivedNumber
  call displaySpeed
```

<https://makecode.microbit.org/b8Ce0Jgdoaae> (**Propulsion 4**)



Try This: Testing the Propellers

Now that you have completed the “**Propulsion 4**” code, it is time to test it.

Upload the “**Propulsion 4**” code onto the micro:bit for the boat and insert it into the motor:bit.

Turn the motor:bit on and place the boat on the dry dock.

- Press A and A again on the boat’s micro:bit:
 - Do the propellers turn? _____
 - Do they turn pushing the boat forward? _____
- Press A again:
 - Do they spin faster? _____
- Press B:
 - Do the propellers slow down? _____
- Press B and again until the micro:bit shows zero:
 - Do the propellers stop? _____
- Press B and B again:
 - Do the propellers spin? _____
 - Do they spin backwards? _____
- Press B again:
 - Do the propellers spin faster backwards? _____
- Press B again:
 - Do the propellers spin faster backwards? _____

7.6 Stop Engines (Propulsion 5 /Remote Control 3)

There is one more practical code addition: Often you want to stop the motors from whatever the Speed is. To do so you must press buttons A or B multiple times until you reach zero. A “Stop Motors” button would be useful.

Let’s do it!

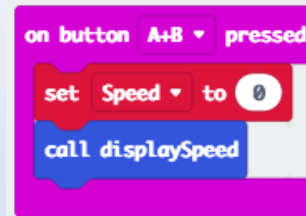


Time to Code: Propulsion 5

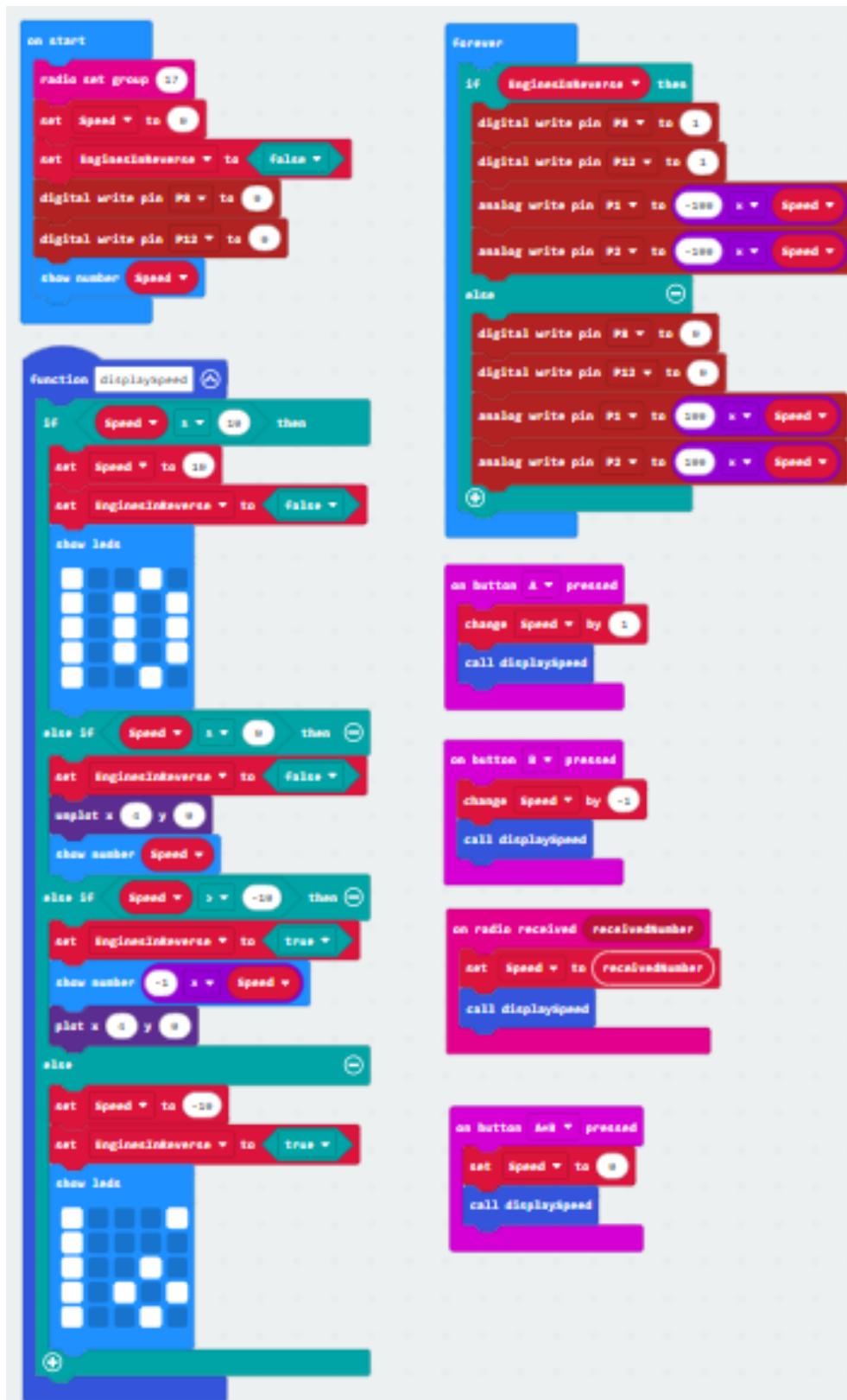
1. Open “**Propulsion 4**” and rename it and save it as “**Propulsion 5**”.
2. Get a new “**on button A pressed**” block from “Input”.
 - Change the “A” to “A+B”.
 - Place a “set Speed to 0” block inside.
 - Copy, paste and drop a “call displaySpeed” block after the “set Speed to 0” block.

Upload and test your code with the boat and its micro:bit only.

- Start motors by pressing button A multiple times.
- Then press button A and B at the same time.
- Do the motors stop immediately?
- Can you restart the motors?
- Do the motors still reverse (press button B)?



This is the complete “Propulsion 5” code:



The image displays a block of Scratch code for a micro:bit project named "Propulsion 5". The code is organized into several sections:

- on start:** Initializes the radio group to 17, sets Speed to 0, EngineInReverse to false, and configures digital pins P8 and P12 to output mode. It also shows the current Speed value.
- function displayspeed:** A custom function that checks the Speed value. If Speed is 10, it sets EngineInReverse to false, updates the LED matrix, and shows the Speed. If Speed is 0, it sets EngineInReverse to false, updates the LED matrix, and shows the Speed. If Speed is greater than -10, it sets EngineInReverse to true, updates the LED matrix, and shows the Speed. Otherwise, it sets Speed to -10, sets EngineInReverse to true, updates the LED matrix, and shows the Speed.
- forever loop:** Checks if EngineInReverse is true. If true, it sets digital pins P8 and P12 to 1 and analog pins P1 and P2 to -100 * Speed. If false, it sets digital pins P8 and P12 to 0 and analog pins P1 and P2 to 100 * Speed.
- on button A pressed:** Increases Speed by 1 and calls displayspeed.
- on button B pressed:** Decreases Speed by 1 and calls displayspeed.
- on radio received receivedNumber:** Sets Speed to receivedNumber and calls displayspeed.
- on button A+B pressed:** Sets Speed to 0 and calls displayspeed.

<https://makecode.microbit.org/C35Yzr2kthVi> (Propulsion 5)

Let's do the same with the remote control.



Time to Code: Remote Control 3

1. Open **Remote Control 2** and rename **Remote Control 3**.
2. Add an **“on button A+B pressed”** block.
 - Place a **“set Speed to 0”** block inside.
 - Add a **“show number”** with **“Speed”** variable
 - Copy, paste and drag a **“radio send number Speed”** block to the end.

```
on button A+B pressed
  set Speed to 0
  show number Speed
  radio send number Speed
```

Your completed **“Remote Control 3”** code should now look like this:

```
on start
  radio set group 17
  set Speed to 0
  show number Speed

forever
  plot x 4 y 0
  pause (ms) 100
  unplot x 4 y 0
  pause (ms) 100

on button A+B pressed
  set Speed to 0
  show number Speed
  radio send number Speed

on button A pressed
  change Speed by 1
  if Speed > 10 then
    set Speed to 10
  if Speed = 10 then
    show leds
  else
    show number Speed
  radio send number Speed

on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
  if Speed = -10 then
    show leds
  else
    show number 0 - Speed
  radio send number Speed
```

https://makecode.microbit.org/_8byF166zbJrA (Remote Control 3)

7.7 Remote Control 4

The “**Remote Control 3**” code allows you to operate the boat at all possible speeds but does not always display negative speeds correctly. It would be good to use the same “displaySpeed” function as the boat does. To accomplish this task, we can use the “displaySpeed” function from the “**Propulsion 5**” code and insert a modified version into the **Remote-Control** code. However, you will need to copy it in the JavaScript form.



Time to Code: Remote Control 4

1. Open the project “**Remote Control 3**” and rename it and save it as “**Remote Control 4**”.
2. Leave this code open and open a new tab your browser in MakeCode.
3. Open “**Propulsion 5**”.
 - Click Edit Code,
 - Click the JavaScript button,
 - Select ALL of the “function displaySpeed” code – find the last curly bracket and include it in your selection, and copy (yes, the text of the code!).
4. Go back to “**Remote Control 4**”,
 - Click the JavaScript button
 - Paste the code for the function “displaySpeed” at the very beginning. You will need to click on line 1 and hit space so you do not delete the first line of the code.
 - Delete each of the individual four lines containing the text “EnginesInReverse = ...”.
 - Then click on the Blocks button and look for the “function displaySpeed” block.
5. Modify the “**displaySpeed**” block:
 - Since the remote control already has the LED (x 4 y 0) blinking to let you know it’s the remote, the LED validation for a negative speed needs to be placed at the (x 4 y 1) position.
 - Make sure you are back into the block view on MakeCode.
 - Change both the “unplot x 4 y 0” and the “plot x 4 y 0” to x 4 y 1 in the “displaySpeed” block.

```
function displaySpeed () {  
  if (Speed >= 10) {  
    Speed = 10  
    EnginesInReverse = false  
    basic.showLeds(`  
      # . . #  
      # . . #  
      # . . #  
      # . . #  
      # . . #  
      # . . #  
    `)  
  }  
  else if (Speed >= 0) {  
    EnginesInReverse = false  
    led.unplot(4, 0)  
    basic.showNumber(Speed)  
  }  
  else if (Speed > -10) {  
    EnginesInReverse = true  
    basic.showNumber(-1 * Speed)  
    led.plot(4, 0)  
  }  
  else {  
    Speed = -10  
    EnginesInReverse = true  
    basic.showLeds(`  
      # . . . #  
      # . . . #  
      # . . . #  
      # . . . #  
      # . . . #  
    `)  
  }  
}
```

```
else if Speed >= 0 then  
  unplot x 4 y 1  
  show number Speed  
else if Speed > -10 then  
  show number -1 * Speed  
  plot x 4 y 1
```



Time to Code: Remote Control 4 (continued)

6. Modify the “**on button A + B pressed**” block:
 - Delete the “show number Speed” block
 - Drag a “call displaySpeed” block from the (Advanced) “Functions” toolbox in its place.

7. Modify the “**on button A pressed**” block:
 - Delete the **second** “if...” block, “if Speed = 10”
 - Place a “call displaySpeed” block in its place.

8. Modify the “**on button B pressed**” block:
 - Delete the **second** “if...” block, “if Speed = -10”
 - Place a “call displaySpeed” block in its place.

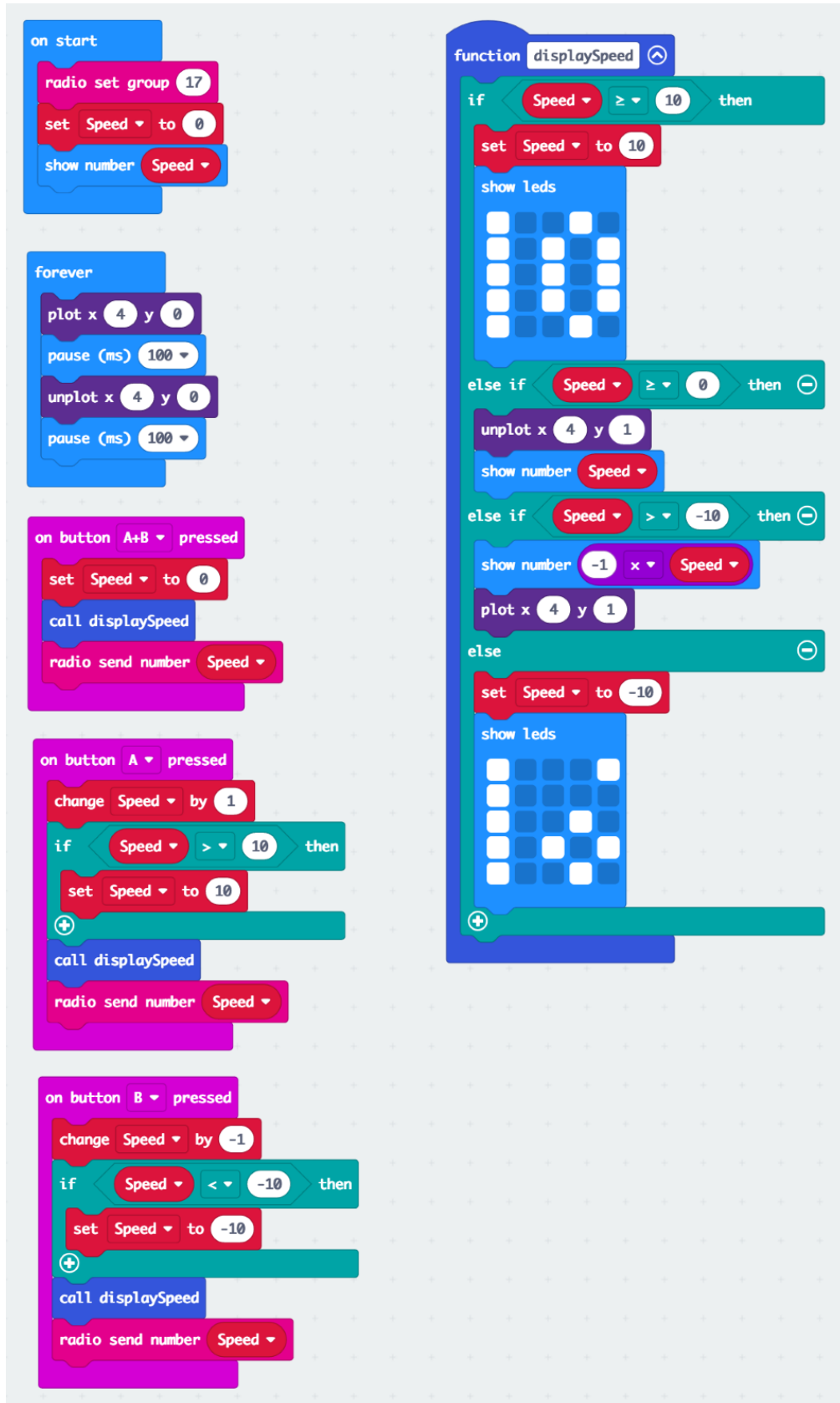
9. Save code the on your remote control micro:bit.
10. See the **Propellers Test with Remote Control** to test it. (below)

```
on button A+B pressed
  set Speed to 0
  call displaySpeed
  radio send number Speed

on button A pressed
  change Speed by 1
  if Speed > 10 then
    set Speed to 10
    call displaySpeed
  radio send number Speed

on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
    call displaySpeed
  radio send number Speed
```

Your complete “Remote Control 4” code should look like this:



The image displays a collection of Scratch code blocks for a remote control project. On the left, there are four event-driven blocks: 'on start', 'on button A+B pressed', 'on button A pressed', and 'on button B pressed'. On the right, there is a function block named 'displaySpeed' which contains conditional logic for speed limits and LED patterns.

```
on start
  radio set group 17
  set Speed to 0
  show number Speed

forever
  plot x 4 y 0
  pause (ms) 100
  unplot x 4 y 0
  pause (ms) 100

on button A+B pressed
  set Speed to 0
  call displaySpeed
  radio send number Speed

on button A pressed
  change Speed by 1
  if Speed > 10 then
    set Speed to 10
  call displaySpeed
  radio send number Speed

on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
  call displaySpeed
  radio send number Speed

function displaySpeed
  if Speed >= 10 then
    set Speed to 10
    show leds
  else if Speed >= 0 then
    unplot x 4 y 1
    show number Speed
  else if Speed > -10 then
    show number -1 x Speed
    plot x 4 y 1
  else
    set Speed to -10
    show leds
```

https://makecode.microbit.org/_V3U6bpMHfCq1 (Remote Control 4)



Try This: Propellers Test with the Remote Control

1. For this test you should have “**Remote Control 4**” loaded onto the remote micro:bit and “**Propulsion 5**” loaded onto the micro:bit for the boat.
2. Place the boat on the dry dock and turn the motor:bit on.
3. Connect the remote control micro:bit to the battery case.
4. Test the remote control:
 - Do the propellers turn at a positive Speed of 3? _____
 - Would they propel the boat forward or backward? _____
 - Do remote control and boat show the same Speed on their LED displays? _____
 - Increase Speed:
 - Do the propellers spin faster? _____
 - Do both spin in the same way? _____
 - Press button B multiple times until you have a Speed of negative 3 (note the dot!).
 - Would the propellers now propel the boat in reverse? _____
 - Do remote control and boat show the same Speed? _____
 - Can you go all the way up to Speed 10? _____
 - Push button A again.
 - What do you see? _____
 - Does input validation work? _____
(Hint: Does the display show the correct speed?)

Module 8: Rudder Assembly and Test

8.1 Attaching the Rudder Servo and the Rudder

Now that your boat has both forward and reverse propulsion, wouldn't it be great to be able to steer the boat?

By adding a rudder to the boat and controlling it with the micro:bit, you will be able to steer or turn the boat. A servo motor will need to be added which will allow the rudder to change positions. Both the servo motor mount and the rudder are 3D printed parts.

Before adding these parts to your boat, check out the resources below to learn about these parts.



Internet Resources:

What are rudders and how do they work?

- How Does a Rudder Work?
 - <https://youtu.be/jWD-IxjdFrs>
- How the Rudder Actually Moves the Boat, Generates “Lift”
 - <https://youtu.be/L-UkOvb6TK8>
- How Does a Ship Turn in Water?
 - <https://youtu.be/k1YxpOg3r6U>

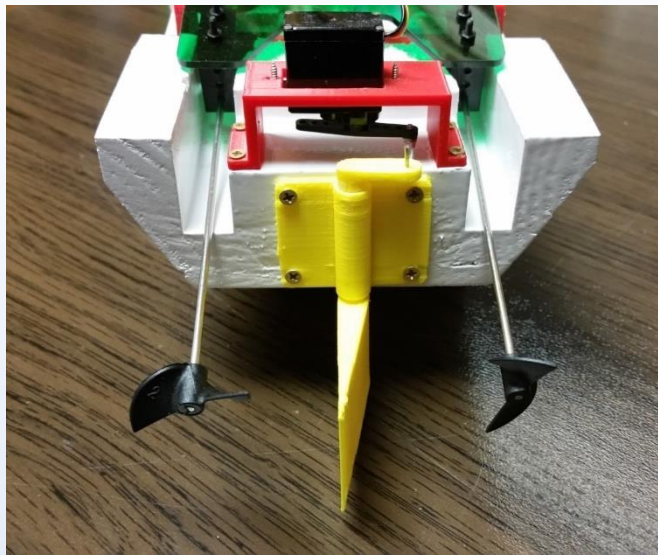
What are servo motors and how do they work?

- Servos Explained
 - <https://www.sparkfun.com/servos>



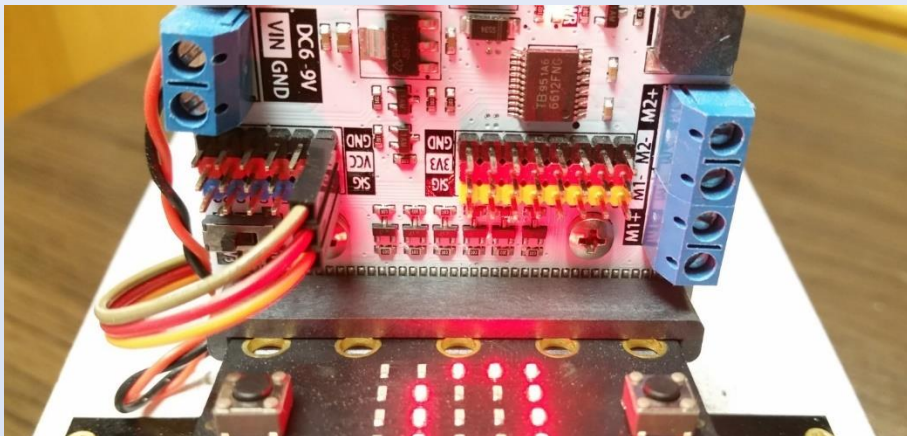
Assembly: Rudder Servo and Rudder

1. Refer to pages 13 and 14 for images of the parts you will need.
2. Watch the **STEM Seals YouTube video: SEA Rudder Assembly**.
3. Step-by-step instructions:
 - a. Attach the servo motor to the servo mount bracket.
 - b. Attach the servo push rod to the servo horn. Change the servo horn type if needed.
 - c. Attach the servo motor/mount assembly to the boat.
 - d. Attach the rudder to the boat.
 - e. Test the rudder mobility by turning the servo horn – do not twist the rudder blade!
Readjust servo horn position if needed.



Rudder and rudder servo motor installed to hull.

- f. Connect the servo wire to pin 13 on the motor:bit board using a 10 cm extension wire.



Rudder and rudder servo motor installed to hull.

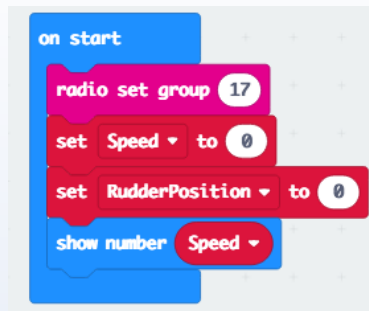
8.2 Code for the Rudder Servo Motor

Remember the code you used to test the propulsion of the boat? You used buttons A and B to increase and decrease the speed of the motors. Now you will modify that code to test the performance of the rudder allowing you to move the rudder one direction with button A and the opposite direction with button B.

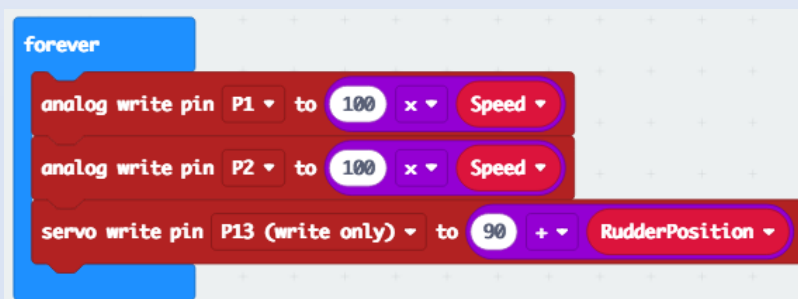


Time to Code: Rudder Servo Motor

1. Use your “**Propulsion 3**” project in MakeCode and rename it “**Boat 2**”.
2. Make a new variable named “RudderPosition”.
3. Modify the “**on start**” block:
 - Drag the “set RudderPosition to 0” block into the “on start” block. Place it right after the “set Speed to 0” block.



4. Modify the **forever** loop:
 - Drag a “servo write pin P0 to 180” block from the “Advanced” > “Pins” toolbox and drop it at the end of the forever loop.
 - Change pin P0 to P13. This is where we connected the rudder servo motor.
 - Drag a “Math” > “0 + 0” block into the last block and drop it onto the number 180.
 - Change the first zero in this block to 90.
 - Drag the variable “RudderPosition” from “Variables” and drop it on the second zero.

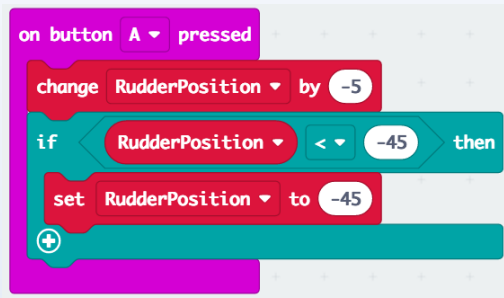




Time to Code: Rudder Servo Motor (continued)

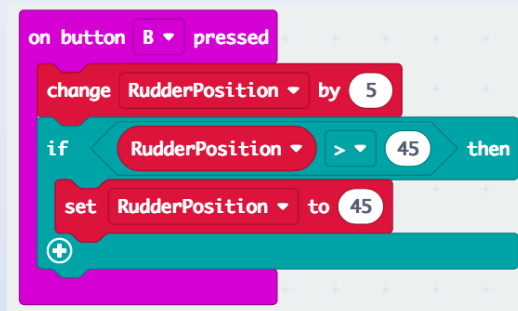
5. Modify “on button A pressed”:

- Click on the “Speed” variable in the “change Speed ...” block.
 - In the drop-down menu, select “RudderPosition”.
This changes the variable in the “change ...” block.
 - Change the number 1 to -5 (negative).
- Delete the “if Speed = 0 ...” block and everything in it.
- Drag a new “if true then” block from the “Logic” toolbox and drop it underneath the “change RudderPosition by -5” block.
 - Drag a “if 0 < 0 then” comparison block from “Logic” and drop it onto the “true” in the “if ...” block.
 - Get the “RudderPosition” variable from “Variables” and drop it onto the first zero in the comparison block.
 - Change the other zero in this block to -45 (negative).
 - Get a “set RudderPosition to 0” block from “Variables” and drop it into the “if ...” block you just created.
 - Change the zero in this block to -45 (negative).



6. Modify the “on button B pressed”:

- Delete the “show number ...” block
- Click on the “Speed” variable in the “change Speed ...” block.
 - In the drop-down menu, select “RudderPosition”.
 - Change the number -1 in the “change RudderPosition ...” block to 5.
- Copy all of the “if RudderPosition < -45 then” block in the “on button A ...” block and drop it at the end in the “on button B ...” block.
 - Change the “<” sign to “>”.
 - Change the number -45(negative) to 45 in both, the comparison block and the “set RudderPosition ...” block.



7. Load this code onto the boat micro:bit.

Your completed code for “Boat 2” should look like the following:

```
on start
  radio set group 17
  set Speed to 0
  set RudderPosition to 0
  show number Speed

forever
  analog write pin P1 to 100 x Speed
  analog write pin P2 to 100 x Speed
  servo write pin P13 (write only) to 90 + RudderPosition

on button A pressed
  change RudderPosition by -5
  if RudderPosition < -45 then
    set RudderPosition to -45

on button B pressed
  change RudderPosition by 5
  if RudderPosition > 45 then
    set RudderPosition to 45

on radio received receivedNumber
  set Speed to receivedNumber
  show number Speed
```

https://makecode.microbit.org/_g0eTyqcU7H9X (Boat 2)



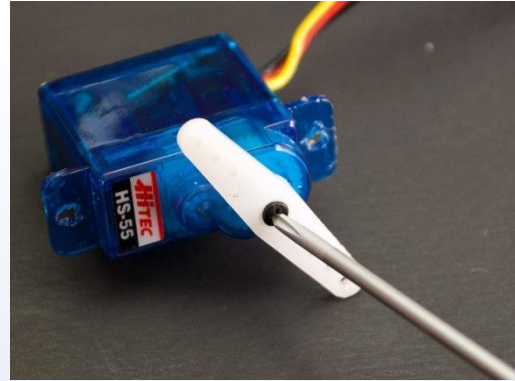
Try This: Test the Rudder

1. Load the “Boat 2” code on the boat micro:bit.
2. Turn the boat on (with the power switch on the motor:bit board).
 - Did you hear the rudder move?
3. Press button A.
 - Does the rudder move? Which direction?
4. Press button A again multiple times and observe the rudder.
 - Which direction would the boat turn?
5. Press button B and observe the rudder.
 - The rudder should move in the opposite direction because of the changed signs in the code.
6. Press button B until nothing new happens anymore.
 - Do you know why?
7. Press button A.
 - Does the rudder still move?
8. Press button A multiple times until nothing new happens anymore.
 - The “if ...” blocks in our code prevent “RudderPosition” from becoming larger than 45 (degrees) or smaller than -45 (degrees). This should prevent the rudder push pin from getting stuck in a position it cannot come back from.
9. Press the Reset button on the micro:bit.
 - The rudder should go in the middle. Does it do that?
 - If it does not reset exactly in the middle, you need to adjust the number 90 in the “servo write pin P13 ...” block in the forever loop. (See tip below)



Tip: Adjusting the Rudder to the Middle (90 °)

When assembling a servo motor there is a small gear that the servo arm sets on. Large adjustments can best be made by resetting the arm in this gear.



However, small adjustments can be made by adjusting the number in the “servo write pin P13” block. If the rudder is more to the right (looking from the stern of the boat), decrease the number. If the rudder is more to the left, increase the number. Load the new code with rudder adjustment on your micro:bit and test. You may wish to write this number down.



Module 9: Remote-Controlled Boat with Rudder

9.1 Remote Control with Rudder

The code for “**Boat 2**” allowed you to check the rudder function using buttons A and B on the micro:bit for the boat. To make this code more usable, let’s modify it to be able to use the remote control to control both the speed and rudder of the boat now that you have checked out your boat rudder. Both the boat code and the remote-control code will have to be modified. We will start with the boat code first.



Time to Code: Remote Controlled Boat (Speed and Rudder)

1. Use the “**Boat 2**” project and rename “**Boat 3**”.
2. Modify the “**on start**” block:
 - Copy the “servo write P13...” block from the forever block and place it after the “set RudderPosition” block.

```
on start
  radio set group 17
  set Speed to 0
  set RudderPosition to 0
  servo write pin P13 (write only) to 90 + RudderPosition
  show number Speed
```




Time to Code: Remote Controlled Boat (continued)


3. Modify the “on radio received” block:

- Drag an “if true then ... else” block from “Logic” and drop it at the beginning.
 - Drag a “ $0 < 0$ ” comparison block and drop it onto the “true”.
 - Drag receivedNumber from that block and replace the first zero with it.
 - Change the second zero to 20.
- In the **first** slot of the “if... then”:
 - Move both, the “set Speed ...” and “show number ...” blocks into this “if...then” block.
- Add two more “else” slots to the “if...then” block by clicking on the (+) sign at the bottom of the “if...then” block :
 - Copy and paste the “receivedNumber < 20 ” comparison block from the **first** “if” statement and place it in the first “else” statement.
 - o Change the number 20 to the number 300.
 - Copy and paste the “receivedNumber < 20 ” comparison block from the first “if” statement and place it in the **second** “else” statement.
 - o Change the number 20 to the number 500
 - o Drag a “set RudderPosition to” block.
 - o Drag a “round” block from “Math” and place in the “set RudderPosition” block.
 - o Drag a “Math” $> “0-0”$ in to the “round” block.
 - o Drag a “receivedNumber” from the “on radio received” block replace the first zero.
 - o Change the second zero to a 400.
 - Leave the **third** “else” statement empty.

4. Load this code onto the micro:bit for the boat.

```
on radio received receivedNumber
  if receivedNumber < 20 then
    set Speed to receivedNumber
    show number Speed
  else if receivedNumber < 300 then
  else if receivedNumber < 500 then
    set RudderPosition to round(receivedNumber - 400)
  else
```

The complete code for “Boat 3” should look like this:



The image shows a Scratch code editor with the following blocks:

- on start**
 - radio set group 17
 - set Speed to 0
 - set RudderPosition to 0
 - servo write pin P13 (write only) to 90 + RudderPosition
 - show number Speed
- forever**
 - analog write pin P1 to 100 x Speed
 - analog write pin P2 to 100 x Speed
 - servo write pin P13 (write only) to 90 + RudderPosition
- on button A pressed**
 - change RudderPosition by -5
 - if RudderPosition < -45 then
 - set RudderPosition to -45
- on radio received receivedNumber**
 - if receivedNumber < 20 then
 - set Speed to receivedNumber
 - show number Speed
 - else if receivedNumber < 300 then
 - else if receivedNumber < 500 then
 - set RudderPosition to round receivedNumber - 400
 - else
- on button B pressed**
 - change RudderPosition by 5
 - if RudderPosition > 45 then
 - set RudderPosition to 45

<https://makecode.microbit.org/YXmJhF6ipFyb> (Boat 3)

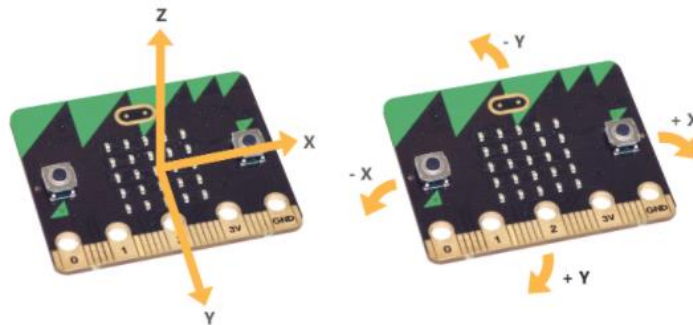
9.2 Remote Control with Speed and Rudder

On the “**Boat 3**” code for the micro:bit - buttons A and B will still control the rudder manually since we did not change their function. Next you will modify the remote-control code to be able to control the speed of the boat using button A (to increase) and button B (to decrease) speed. We will create a new function block called “detectTilt” that will be used to control the rudder. This new function “detectTilt” will use the built-in **accelerometer** on the micro:bit. The accelerometer measures acceleration. See the following diagram and resources.



The micro:bit measures movement along three axes:

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



Internet Resources: The Micro:bit Accelerometer

Check out these accelerometer resources:

- Micro:bit Accelerometer (video 1:28)
 - <https://youtu.be/UT35ODxvmS0>
- Behind the MakeCode Hardware: Accelerometer on the Micro:bit (video 5:47)
 - <https://youtu.be/byngewjO51U>

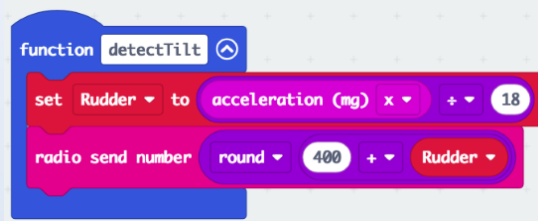
Now let's modify the code for the remote control.



Time to Code: Remote Control 5 (Controlling Speed and Rudder)

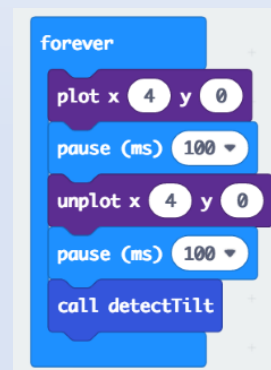
Since the micro:bit only has two buttons, we will use a tilt and the micro:bit's accelerometer to control the rudder, leaving the buttons for speed.

1. Use the project “**Remote Control 4**” and rename it “**Remote Control 5**”.
2. Create a new function called “**detectTilt**”.
 - Make a function by clicking on the “Advanced” > “Functions” toolbox. Name it “detectTilt” by typing detectTilt over “doSomething”.
 - The function will now show on your workspace.
 - Make a variable name it “Rudder”.
 - Drag the “set Rudder to 0” block from “Variables” and place it into the function.
 - Drag a “Math” > “0 / 0” (divided) block over the zero in the “set Rudder” block
 - Drag an “acceleration” block from “Input” over the first zero
 - Change the second zero to 18.
 - Drag a “radio send number 0” block from “Radio” and drop it after the “set Rudder ...” block.
 - Drag a “round” block over the zero
 - Drag a “0 + 0” block from “Math” and drop it in the “round” block.
 - Replace the first zero in the math block with 400.
 - Drag the “Rudder” variable from “Variable” and drop it onto the other zero.

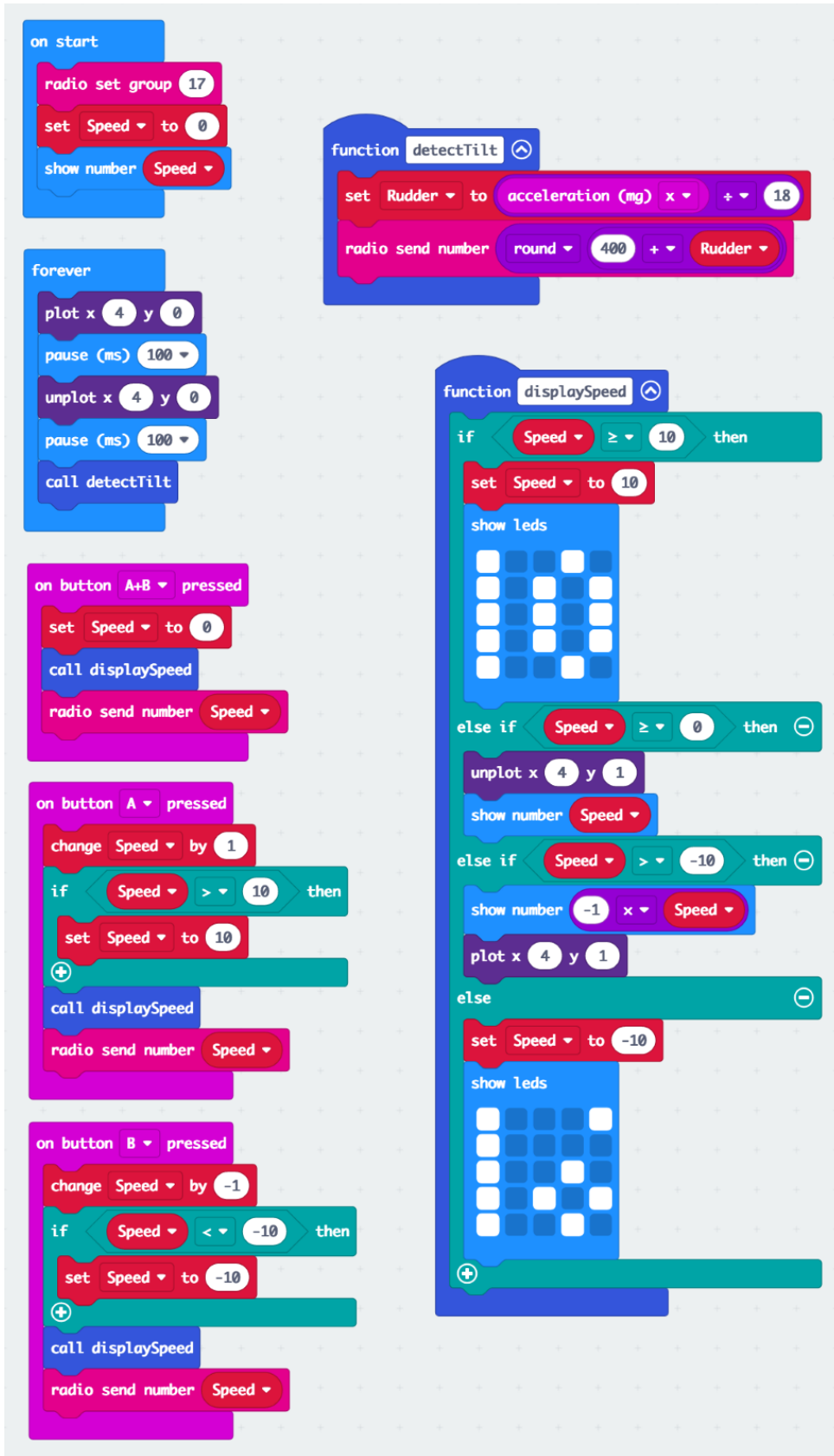


round (400 + Rudder)

3. Drag the “call detectTilt” block from “Functions” and drop it at the end of the **forever loop**.
4. Load this onto the remote control micro:bit.
5. Test the new remote-control functions in the **Remote Controlled Rudder and Speed) Test** below.



The complete code for “Remote Control 5” should look like this:



The image displays a collection of Scratch code blocks for a project titled "Remote Control 5". The code is organized into several functional sections:

- on start:** A block to set the radio group to 17, initialize the Speed variable to 0, and show the Speed variable.
- forever loop:** A loop that plots the Speed variable on a coordinate system (x=4, y=0), pauses for 100ms, unplots the variable, pauses again for 100ms, and then calls the detectTilt function.
- detectTilt function:** A function that calculates the Rudder value based on acceleration (mg) and a multiplier of 18, then sends the rounded Rudder value via radio.
- on button A+B pressed:** A block that resets Speed to 0 and calls the displaySpeed function.
- on button A pressed:** A block that increments Speed by 1. If Speed is greater than 10, it resets Speed to 10. It then calls displaySpeed and sends the Speed value via radio.
- on button B pressed:** A block that decrements Speed by 1. If Speed is less than -10, it sets Speed to -10. It then calls displaySpeed and sends the Speed value via radio.
- displaySpeed function:** A function that checks the Speed variable against several conditions:
 - If Speed is greater than or equal to 10, it sets Speed to 10 and shows the LED display.
 - Else if Speed is greater than or equal to 0, it unplots the Speed variable, shows the Speed variable, and shows the LED display.
 - Else if Speed is greater than -10, it shows the Speed variable multiplied by -1, plots the Speed variable on the coordinate system, and shows the LED display.
 - Else, it sets Speed to -10 and shows the LED display.

https://makecode.microbit.org/_CKDbKwKAPJTX (Remote Control 5)



Try This: Remote Controlled Boat (Speed and Rudder) Test

1. Make sure the “**Boat 3**” code is loaded on the boat micro:bit and the “**Remote Control 5**” code is loaded on the remote micro:bit.
2. Place the boat on the dry dock and turn it on.
3. Take the remote control micro:bit (connected to battery power) in both of your hands and hold it level.
4. Now tilt the remote slowly to the left and observe the rudder.
 - Does it move?
5. Tilt the remote control to the right.
 - Does the rudder do what you expect?
6. Does the tilt of the micro:bit turn the boat correct direction?
7. Now turn the engines on by pressing button A two or three times.
 - Do the propellers work?
8. Can you control the propellers (drive/speed) and rudder (steer) with the remote?
9. Practice these maneuvers... and don't forget to turn the boat off when you are finished and disconnect the remote control from the battery.



At this point, you could even test the boat in a small body of water such as a cow trough!

9.3 Remote Controlled Boat with Reversible Engines and Rudder

Your boat is very navigable now! You can now control the speed (drive) and rudder (direction). However, while the boat should turn both left and right, it can only move in a forward direction with the remote control. It might be handy to be able to go in reverse if needed. To do this, we will need to modify only the boat micro:bit code.



Time to Code: Remote Controlled Boat (Reversible Engines with Rudder)

1. Use the “Boat 3” code copy it and rename “Boat 4”.
2. Modify the “on-start” block:
 - Create a variable “EnginesInReverse”
 - Remember, we started the “Boat” codes with “Propulsion 3” code, and it did not have the “EnginesInReverse” variable yet.
 - Add a “set EnginesInReverse to” block after the “set Speed...” block in the “on-start” block.
 - Set the “EnginesInReverse” to “false” from the “Logics” toolbox.
 - Add the two “digital write pin ...” blocks for P8 and P12 after the “set Rudder to...” block.

```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
```

3. Create a “function displaySpeed” block from the JavaScript of “Propulsion 5”.
 - Open “Propulsion 5” in a new tab of your browser.
 - Toggle to the JavaScript.
 - Copy from the beginning of the function “displaySpeed” to the last curly line for that function.
 - Go back to the “Boat 4” opened browser, toggle to JavaScript.
 - At the top of the JavaScript code, paste the copied code. Be sure to add a space at the top before pasting it.
 - Toggle back to the Block view and check to see if the function “displaySpeed” is there.
 - See the picture of the complete code below for reference.



Time to Code: Remote Controlled Boat (Reversible Engines with Rudder)

(continued)

4. Modify the “on radio received” block:

- Replace the “show Number Speed” block with the “call displaySpeed” block.

```
on radio received receivedNumber
if receivedNumber < 20 then
  set Speed to receivedNumber
  call displaySpeed
else if receivedNumber < 300 then
  // empty
else if receivedNumber < 500 then
  // empty
else
  set RudderPosition to round(receivedNumber / 400)
```

5. Modify the “forever” block:

- Place an “if ... then ... else” block from the “Logic” toolbox at the beginning
- Place the EnginesInReverse variable into the new “if ...” block in the first slot.
- Drag the two “analog write pin P1 or P2” blocks into the **first** slot of the “if ...” block.
- Leave the “servo write pin P13 ...” block outside.
- Copy, paste and place both “digital write pin P8 and P12” blocks from the “onStart” block and paste above the two “analog write pin ...” blocks in the “if ...then” block.
- Copy all four blocks (two “digital write ...” and two “analog write ...”) from the **first** slot into the second slot of the “if ...” block.
- Change the zeros in the two “digital write pin ...” blocks in the first slot to “1”.
- Change the number “100” in both “analog write pin ...” blocks in the first slot in the “if ...” block to the number “-100”.

```
forever
if EnginesInReverse then
  digital write pin P8 to 1
  digital write pin P12 to 1
  analog write pin P1 to -100 x Speed
  analog write pin P2 to -100 x Speed
else
  digital write pin P8 to 0
  digital write pin P12 to 0
  analog write pin P1 to 100 x Speed
  analog write pin P2 to 100 x Speed
servo write pin P13 (write only) to 90 + RudderPosition
```

6. Add the “on button A+B pressed” block

- Drag a “set Speed to 0” and “call displaySpeed” into this block .

```
on button A+B pressed
  set Speed to 0
  call displaySpeed
```

7. Upload this code to the boat micro:bit and test.

- Can you reverse the direction of the motors?
- Does the rudder still work?

The complete code for “Boat 4” should look like this:

```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed

forever
  if EnginesInReverse then
    digital write pin P8 to 1
    digital write pin P12 to 1
    analog write pin P1 to -100 x Speed
    analog write pin P2 to -100 x Speed
  else
    digital write pin P8 to 0
    digital write pin P12 to 0
    analog write pin P1 to 100 x Speed
    analog write pin P2 to 100 x Speed
  servo write pin P13 (write only) to 90 + RudderPosition

function displaySpeed
  if Speed > 0 then
    set EnginesInReverse to false
    if Speed = 10 then
      show leds
    else
      unplot x 4 y 1
      show number Speed
  else
    set EnginesInReverse to true
    if Speed = -10 then
      show leds
    else
      show number -1 x Speed
      plot x 4 y 1
```

(continued on the next page)

The image displays three Scratch code blocks on a grid background. The first block, 'on radio received receivedNumber', is a large teal block with a pink border. It contains an 'if' statement with a condition 'receivedNumber < 20'. If true, it sets 'Speed' to 'receivedNumber' and calls 'displaySpeed'. It then has an 'else if' block with condition 'receivedNumber < 300', followed by another 'else if' block with condition 'receivedNumber < 500' which sets 'RudderPosition' to 'round receivedNumber - 400'. The final 'else' block is empty. The second block, 'on button A pressed', is a purple block with a pink border. It changes 'RudderPosition' by -5, then has an 'if' statement with condition 'RudderPosition < -45' which sets 'RudderPosition' to -45. The third block, 'on button B pressed', is a purple block with a pink border. It changes 'RudderPosition' by 5, then has an 'if' statement with condition 'RudderPosition > 45' which sets 'RudderPosition' to 45. A fourth block, 'on button A+B pressed', is a purple block with a pink border. It sets 'Speed' to 0 and calls 'displaySpeed'.

```
on radio received receivedNumber
  if receivedNumber < 20 then
    set Speed to receivedNumber
    call displaySpeed
  else if receivedNumber < 300 then
  else if receivedNumber < 500 then
    set RudderPosition to round receivedNumber - 400
  else

on button A pressed
  change RudderPosition by -5
  if RudderPosition < -45 then
    set RudderPosition to -45

on button B pressed
  change RudderPosition by 5
  if RudderPosition > 45 then
    set RudderPosition to 45

on button A+B pressed
  set Speed to 0
  call displaySpeed
```

https://makecode.microbit.org/_RePHTKdwtE97 (Boat 4)

Module 10: Steering Test with Remote Control

It is now time to test out your boat! Before moving to the practice sites, the mobile shallow pond, or Lake Osceola make sure you have all of your needed materials.

- Your boat with micro:bit (loaded with “**Boat 4**” code).
- Your remote control micro:bit and the battery case (loaded with “**Remote Control 5**”).
- Your Lectionary, clipboard, and a pencil.

10.1 Mobile Pond Practice Notes

Use the space below to make detailed notes on how your boat functioned and operated.

10.2 Lake Osceola Practice Notes

Use the space below to make detailed notes on how your boat functioned and operated.

Module 11: Water Sampling Assembly and Code

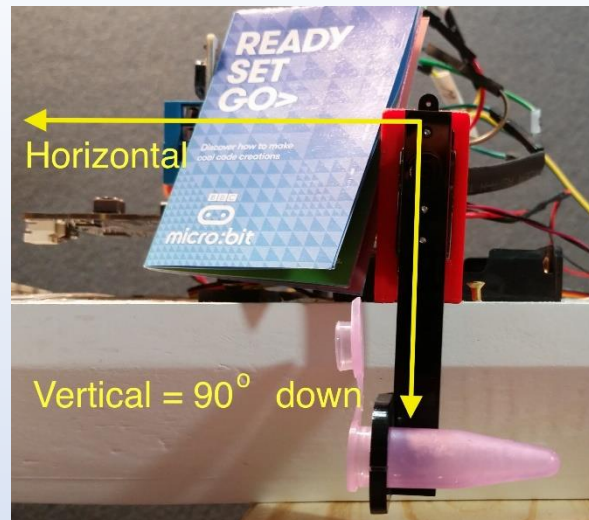
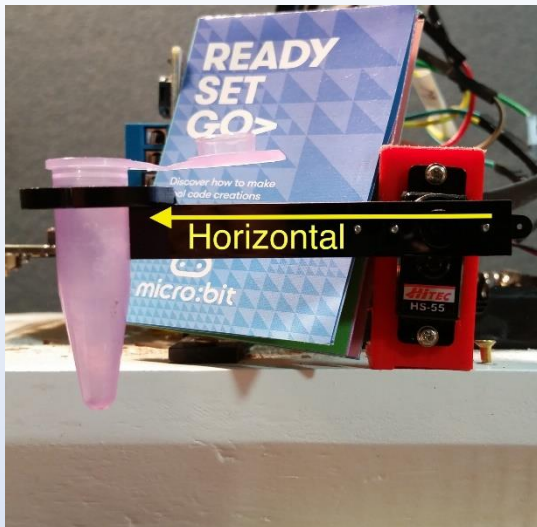
11.1 Assembling the Water Sampler

Now that we have a fully functioning boat that can move forward and backwards and steer, it is time to add a water sampler. We will use another servo motor to control the motion of the sampler arm and allow it to lower into the water to take a sample at the desired location.



Assembly: Water Sampler and Mount

1. See the image on page 14 for the parts needed.
2. Watch the **STEM SEALS YouTube video: SEA Water Sampler Assembly**.
3. Assemble and mount the water sampler.
4. Detach the battery case and reattached to balance the boat. Test balance using a plastic tub filled with water.



Left: Water Sampler Arm in the up position, “SamplerPosition” variable should be about 45 (degrees). Right: The down position is 90° different; it should be around 135 (degrees).

11.2 Water Sampler Code

Like all moving parts on the boat, the water sampler must be controlled by the micro:bit, which receives instructions from the code which we create in MakeCode and then download onto the micro:bit.

Let's create the code to make the water sampler function!



Time to Code: Water Sampler (Boat 5)

1. Use the “**Boat 4**” project, copy, and rename to “**Boat 5**”.
2. Modify the “**on-start**” block:
 - Create a new variable and name it SamplerPosition.
 - Drag and place a “set SamplerPosition” block after the “set RudderPosition” block.
 - Change the zero to a 1 in the “set SamplerPosition” block.
 - Drag a “repeat 4 times” block from the “Loops” toolbox and place it after the “show number” block.
 - Change the 4 to 20 in the “repeat 4 times” block.
 - Drag a “change SamplerPosition” block from “Variables” and place it inside the “repeat” block.
 - Add a “pause 100 ms” after the “Change SamplerPosition” block.

- Copy the “servo write pin P13 ...” block and place it after the “pause” block.
 - Change pin P13 to pin P15.
 - Change the “RudderPosition” variable in this block to “SamplerPosition”.

```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  set SamplerPosition to 1
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
  repeat 20 times
    do
      change SamplerPosition by 1
      servo write pin P15 (write only) to 90 + SamplerPosition
      pause (ms) 100
```



Time to Code: Water Sampler (Boat 5) – (continued)

3. Modify the “on radio received” block:

- At the bottom of the “if receivedNumber < 20...” block, there should be an empty “else” slot. If not click on the plus sign (+) to add another else slot.
- Copy the “set RudderPosition...” block and place in this “else” slot.
- Change “RudderPosition” to “SamplerPosition”
- Change the 400 to 800 in this block.

```
on radio received receivedNumber
  if receivedNumber < 20 then
    set Speed to receivedNumber
    call displaySpeed
  else if receivedNumber < 300 then
  else if receivedNumber < 500 then
  set RudderPosition to round receivedNumber - 400
  else
  set SamplerPosition to round receivedNumber - 800
```

Sometimes the servo motors don't like to be moved to the same position every time the forever loop is executed. It should not matter, but it may lead to jittering of the servos, which is unnecessary. This can be improved by remembering the previous position as RudderPositionOld or SamplerPositionOld and by writing to the servo only when the position is different than the previous position.

Let's fix this also.



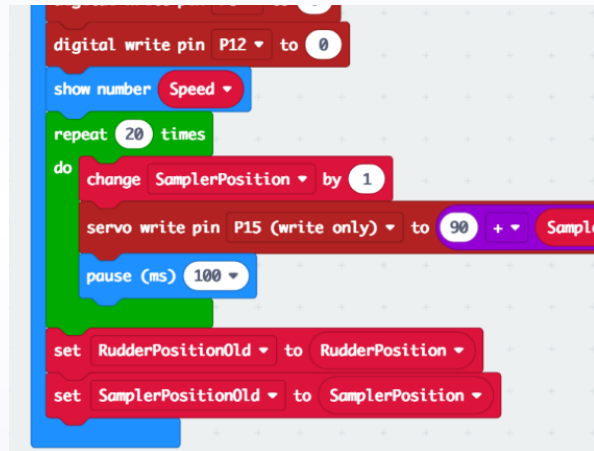


Time to Code: Water Sampler (Boat 5) – (continued)

4. Make two new variables: “RudderPositionOld” and “SamplerPositionOld”.

5. Add to the end of the “on-start” block:

- Drag a “set RudderPositionOld to 0”
 - Set “RudderPositionOld” to “RudderPosition”
- Drag a “set SamplerPositionOld to 0” after the previous block..
 - Set “SamplerPositionOld” to “SamplerPosition”

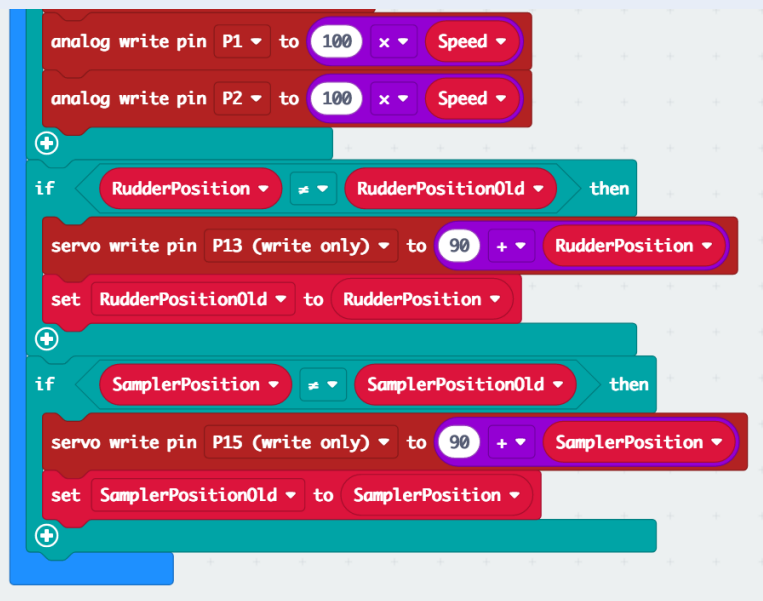


Picture cropped to show end of “on-start” block.

6. Add to the end of the “forever” loop:

- Insert an “if, then...” block at the end of the “forever” loop.
 - Drag an $(0 \neq 0)$ comparison into the “if.....”
 - Drag the variables to create: “if RudderPosition \neq RudderPositionOld”
 - Drag the “servo write pin P13 to 90 + RudderPosition” block and place in the slot of the “if...then” block.
 - Drag a “set RudderPositionOld to 0” after this block.
 - Set “RudderPositionOld” to “RudderPosition”
- Repeat the same steps above with the variable “SamplePositionOld” by adding another “if, then...” block.

6. Load “Boat 5” code on the micro:bit for the boat. See the activity below for testing.



Picture has been cropped to show the end of the “forever” loop.

The complete code for the water sampler, “Boat 5” should look like this:

```

on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  set SamplerPosition to 20
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
  repeat 20 times
    do
      change SamplerPosition by 1
      servo write pin P15 (write only) to 90 + SamplerPosition
      pause (ms) 100
  set RudderPositionOld to RudderPosition
  set SamplerPositionOld to SamplerPosition

forever
  if EnginesInReverse then
    digital write pin P8 to 1
    digital write pin P12 to 1
    analog write pin P1 to -100 x Speed
    analog write pin P2 to -100 x Speed
  else
    digital write pin P8 to 0
    digital write pin P12 to 0

function displaySpeed
  if Speed > 0 then
    set EnginesInReverse to false
    if Speed = 10 then
      show leds
    else
      unplot x 4 y 1
      show number Speed
  else
    set EnginesInReverse to true
    if Speed = -10 then
      show leds
    else
      show number -1 x Speed
      plot x 4 y 1
  
```

Code continues on the next page. The forever loop has been cropped, see below for the rest.

```

analog write pin P1 to 100 * Speed
analog write pin P2 to 100 * Speed

if RudderPosition != RudderPositionOld then
  servo write pin P13 (write only) to 90 + RudderPosition
  set RudderPositionOld to RudderPosition

if SamplerPosition != SamplerPositionOld then
  servo write pin P15 (write only) to 90 + SamplerPosition
  set SamplerPositionOld to SamplerPosition

on radio received receivedNumber
  if receivedNumber < 20 then
    set Speed to receivedNumber
    call displaySpeed
  else if receivedNumber < 300 then
  else if receivedNumber < 500 then
    set RudderPosition to round(receivedNumber - 400)
  else
    set SamplerPosition to round(receivedNumber - 800)

on button A pressed
  change RudderPosition by -5
  if RudderPosition < -45 then
    set RudderPosition to -45

on button B pressed
  change RudderPosition by 5
  if RudderPosition > 45 then
    set RudderPosition to 45

on button A+B pressed
  set Speed to 0
  call displaySpeed

```

https://makecode.microbit.org/_5gqXYecgDi4E (Boat 5)



Try This: Water Sampler (Level) Test

1. Make sure that “Boat 5” is loaded on the boat micro:bit. Turn on the motor:bit.
2. When the boat powered up, did the sampler arm move?
3. Press the micro:bit reset button.
 - Did the sampler arm move after reset?
4. Do buttons A and B still control the rudder movement?
 - When you press button A, which direction would the boat turn?
 - When you press button B, which direction will the boat turn?
5. Is the sampler arm and collection vial level after you press the reset button?
 - If not- correct excessive tilt by removing the set screw of the servo horn, pulling the horn (with connected sampler arm) off the cog and resetting it level or horizontal. Do not forget to replace the set screw.

Now that you know the sampler arm works, let's add it to the remote control. This will allow you to take a water sample when your boat is in the lake. Remember how we used the micro:bits' accelerometer to detect a tilt to move the rudder? We used the x-axis to detect a tilt left or right to move the rudder in the same direction as the tilt. Now we will use the y-axis to detect a tilt to move the sampler arm up and down. We just need to add a few simple blocks to the "detectTilt" function on the remote control.



Time to Code: Remote Control Water Sampler (Remote Control 6)

1. Use "**Remote Control 5**", copy and rename it "**Remote Control 6**".
2. Make a new variable "Sampler".
3. Modify the function "**detectTilt**" block:
 - Drag the "set Sampler to 0" block to the end of the "function detectTilt" block.
 - Copy the "(acceleration (mg) x / 18)" block from the "set Rudder.." block and drop it over the zero in the "set Sampler ..." block.
 - Change the "x" in this block to "y".
 - Copy the "radio send number ..." block from above the "set Sampler ..." block and paste it after the "set Sampler ..." block.
 - Change the number 400 in this block to 800.
 - Change the "Rudder" variable in this block to "Sampler".
4. Load the code onto the micro:bit used as remote control. See the activity below to test it.

```
function detectTilt
  set Rudder to acceleration (mg) x / 18
  radio send number round 400 Rudder
  set Sampler to acceleration (mg) y / 18
  radio send number round 800 Sampler
```


The complete code for “Remote Control 6” should look like this:

```

on start
  radio set group 17
  set Speed to 0
  show number Speed

function detectTilt
  set Rudder to acceleration (mg) x + 18
  radio send number round 400 + Rudder
  set Sampler to acceleration (mg) y + 18
  radio send number round 800 + Sampler

function displaySpeed
  if Speed >= 10 then
    set Speed to 10
    show leds
  else if Speed >= 0 then
    unplot x 4 y 1
    show number Speed
  else if Speed > -10 then
    show number -1 x Speed
    plot x 4 y 1
  else
    set Speed to -10
    show leds

forever
  plot x 4 y 0
  pause (ms) 100
  unplot x 4 y 0
  pause (ms) 100
  call detectTilt

on button A+B pressed
  set Speed to 0
  call displaySpeed
  radio send number Speed

on button A pressed
  change Speed by 1
  if Speed > 10 then
    set Speed to 10
  call displaySpeed
  radio send number Speed

on button B pressed
  change Speed by -1
  if Speed < -10 then
    set Speed to -10
  call displaySpeed
  radio send number Speed
  
```

https://makecode.microbit.org/_VUCDqxW46WuH (Remote Control 6).

You should now be able to control the movement of the water sampler with the remote control.

Let's test it out!



Try This: Water Sampler Test (Remote Controlled)

1. Make sure you have “**Boat 5**” loaded on the boat micro:bit and “**Remote Control 6**” on the remote micro:bit.
2. Turn the boat (motor:bit) on.
 - Does the sampler arm still move on start up?
3. Hold the remote micro:bit level and plug it into the battery case to turn it on.
4. Tilt the remote control forward.
 - Does the water sampler go down?
5. Tilt the remote control back towards you.
 - Does the sampler go up?
6. Can you hold the remote level?
7. Tilt the remote control left and right.
 - Does the rudder still move correctly?
8. Press buttons A and B.
 - Do the propellers still work correctly?

11.3 Automatic Water Sampling

It is not necessary to always control all positions of the water sampler. It would be better if we could tilt the remote forward and the sampler would go down, dip into the water, wait until the vial is filled, and then automatically return to the up position with the vial in the level position.

Let's modify our sampler code to do just that!



Time to Code: Automatic Water Sampler (Boat 6)

1. Use the project “**Boat 5**”, copy, save, and rename it “**Boat 6**”.
2. Modify the “**on-start**” block:
 - Create a new variable “**SamplerUpPosition**”.
 - Change the “**set SamplerPosition to 20**” block to “**set SamplerUpPosition to 130**”.
 - Create a new block “**set SamplerPosition to (SamplerPosition – 20)**”.
 - In the “**servo write pin P15...**” block, delete the “**90 + SamplerPosition**” part and replace with the variable “**SamplerPosition**”. (see arrow)
 - Delete the “**ser SamplerPositionOld to SamplerPosition**” block at the end of the “**on start**” block.
 - Create a new variable named “**SamplerIsUp**”
 - Place a new “**set SamplerIsUp to true**” block at the end of the “**on-start**” block.

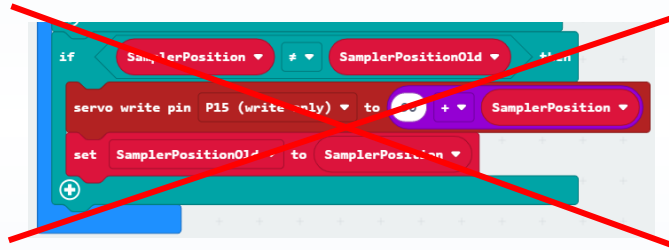
```
on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  set SamplerUpPosition to 130
  set SamplerPosition to SamplerUpPosition - 20
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
  repeat 20 times
    do
      change SamplerPosition by 1
      servo write pin P15 (write only) to SamplerPosition
      pause (ms) 100
  set RudderPositionOld to RudderPosition
  set SamplerIsUp to true
```



Time to Code: Automatic Water Sampler (Boat 6) (Continued)

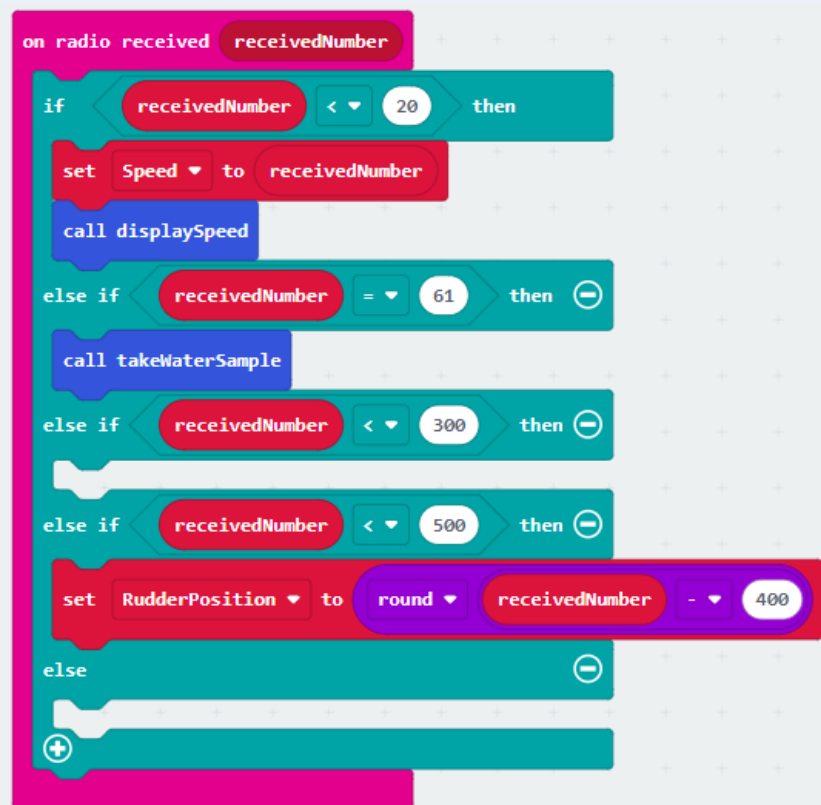
3. Modify the “forever” block:

- Delete the last “if SamplerPosition....” block.



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

- Change the condition for the first “else if ...” to “else if (receivedNumber = 61) then”.
- Create a new function from the “Functions” toolbox. Name it “takeWaterSample”.
- Drag the “call takeWaterSample” from the “Functions” toolbox into the slot of the “else if (receivedNumber = 61) then” slot.
- Delete the “set SamplerPosition ...” block in the “else ...” slot of the “if ...” at the end of the “on radio received” block.





Time to Code: Automatic Water Sampler (Boat 6) (Continued)

5. Finish creating the function "takeWaterSample":

- Create a "set SamplerPosition to SamplerUpPosition" block and place at beginning.
- Drag a "set SamplerIsUp to 0" block.
 - Replace the "0" in this block with "false" from the "Logic" toolbox.
- Create a new "repeat 90 times" block and place it into the function.
 - Create (or copy from the "on start" block) a "servo write pin P15 to SamplerPosition" and place it into the "repeat ..." block.
 - Create a new "change SamplerPosition by -1" block and place it into the "repeat ..." block.
 - Create a new "pause (ms) 100" block and place it into the "repeat ..." block.
- Copy the "pause ..." block from the "repeat ..." block and place after the "repeat ..." block. Change the time from 100 to 500 ms.
- Copy the "repeat 90 times ..." block and place it after the "pause (ms) 500" block.
 - Change the "-1" in the "change SamplerPosition by -1" in the second "repeat ..." block to a positive "1".
- Copy the "set SamplerIsUp to false" block from the beginning of the function and place it at the end.
 - Change "false" to "true".

```
function takeWaterSample
  set SamplerPosition to SamplerUpPosition
  set SamplerIsUp to false
  repeat 90 times
    do
      servo write pin P15 (write only) to SamplerPosition
      change SamplerPosition by -1
      pause (ms) 100
    do
      pause (ms) 500
  repeat 90 times
    do
      servo write pin P15 (write only) to SamplerPosition
      change SamplerPosition by 1
      pause (ms) 100
  set SamplerIsUp to true
```

The complete code for the “Automatic Water Sampler (Boat 6)” should look like this:

```

on start
  radio set group 17
  set Speed to 0
  set EnginesInReverse to false
  set RudderPosition to 0
  set SamplerUpPosition to 130
  set SamplerPosition to SamplerUpPosition - 20
  servo write pin P13 (write only) to 90 + RudderPosition
  digital write pin P8 to 0
  digital write pin P12 to 0
  show number Speed
  repeat 20 times
    do
      change SamplerPosition by 1
      servo write pin P15 (write only) to SamplerPosition
      pause (ms) 100
  set RudderPositionOld to RudderPosition
  set SamplerIsUp to true

forever
  if EnginesInReverse then
    digital write pin P8 to 1
    digital write pin P12 to 1
    analog write pin P1 to -100 * Speed
    analog write pin P2 to -100 * Speed
  else
    digital write pin P8 to 0
    digital write pin P12 to 0
    analog write pin P1 to 100 * Speed
    analog write pin P2 to 100 * Speed
  if RudderPosition ≠ RudderPositionOld then
    servo write pin P13 (write only) to 90 + RudderPosition
    set RudderPositionOld to RudderPosition

on button A pressed
  change RudderPosition by -5
  if RudderPosition < -45 then
    set RudderPosition to -45

on button B pressed
  change RudderPosition by 5
  if RudderPosition > 45 then
    set RudderPosition to 45

on button A+B pressed
  set Speed to 0
  call displaySpeed
  
```

(Continued on the next page)

(Continued: Automatic Water Sampler (Boat 6))

```
function displaySpeed
  if Speed > 0 then
    set EnginesInReverse to false
    if Speed = 10 then
      show leds
    else
      unplot x 4 y 1
      show number Speed
    else
      set EnginesInReverse to true
      if Speed = -10 then
        show leds
      else
        show number -1 * Speed
        plot x 4 y 1

function takeWaterSample
  set SamplerPosition to SamplerUpPosition
  set SamplerIsUp to false
  repeat 90 times
    do
      servo write pin P15 (write only) to SamplerPosition
      change SamplerPosition by -1
      pause (ms) 100
  pause (ms) 500
  repeat 90 times
    do
      servo write pin P15 (write only) to SamplerPosition
      change SamplerPosition by 1
      pause (ms) 100
  set SamplerIsUp to true

on radio received receivedNumber
  if receivedNumber < 20 then
    set Speed to receivedNumber
    call displaySpeed
  else if receivedNumber = 61 then
    call takeWaterSample
  else if receivedNumber < 300 then
  else if receivedNumber < 500 then
    set RudderPosition to round receivedNumber - 400
  else
```

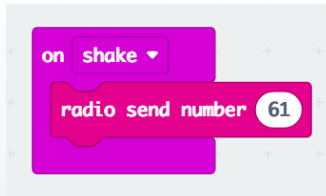
https://makecode.microbit.org/_1q5i6o8qH5uo (Boat 6)

Now we need to do a simple modification to the remote control to add the automatic sampling function.

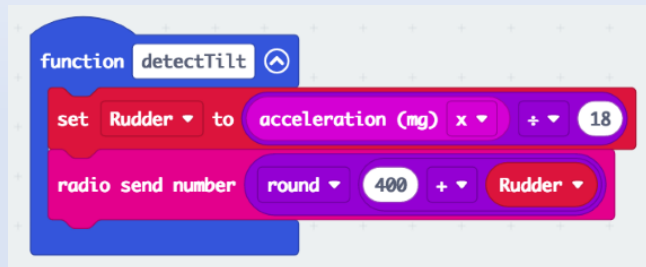


Time to Code: Automatic Water Sampler (Remote Control)

1. Use the “**Remote Control 6**” project and rename it “**Remote Control 7**”.
2. Add a “**on shake**” block from the “Input” toolbox.
 - Place a “radio send number 61” block inside the “on shake” block.



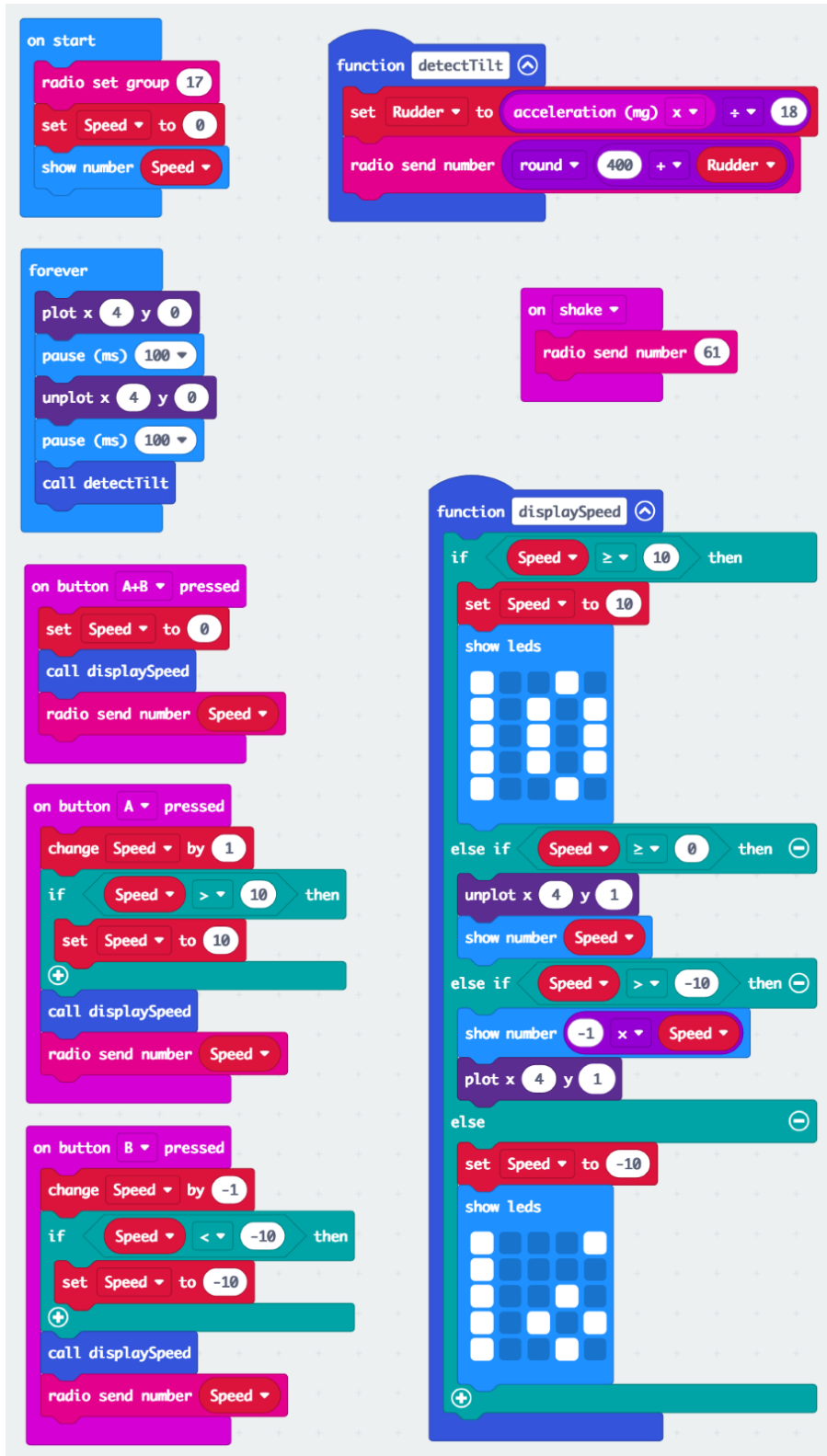
3. Modify the function “**detectTilt**” block:
 - Delete the “set Sampler ...” block in the “function detectTilt”.
 - Delete the “radio send number round(800+Sampler)” block in the “function detectTilt”.



Try This: Automatic Water Sampler Test

1. Load the “**Boat 6**” project on the boat micro:bit and “**Remote Control 7**” on the remote micro:bit.
2. Turn on the boat. Does the sampler move?
3. Turn on the remote control micro:bit. Hold it level.
 - Does the sampler move? It should not.
4. Tilt the remote control left and right. Does the rudder move?
5. Shake the remote control quickly back and forth. Experiment with it! Does the sampler move?
6. Operate the engines with buttons A and B. Does everything work as expected?

The complete code for the “**Remote Control 7**” should now look like this:



The image displays a Scratch script for a remote control project. The script is organized into several functional blocks:

- on start:** A radio group is set to 17, Speed is initialized to 0, and the Speed variable is shown on the screen.
- detectTilt function:** A function that sets the Rudder variable to the acceleration (mg) multiplied by 18, and then sends the rounded Rudder value (multiplied by 400) via radio.
- forever loop:** A loop that plots a point at (4, 0), pauses for 100ms, unplots the point, pauses for 100ms, and then calls the detectTilt function.
- on shake:** A block that sends the number 61 via radio.
- on button A+B pressed:** Resets Speed to 0, calls displaySpeed, and sends the Speed value via radio.
- on button A pressed:** Increments Speed by 1. If Speed is greater than 10, it is set to 10. It then calls displaySpeed and sends the Speed value via radio.
- on button B pressed:** Decrements Speed by 1. If Speed is less than -10, it is set to -10. It then calls displaySpeed and sends the Speed value via radio.
- displaySpeed function:** A function that checks the Speed value and updates the LED display and plot accordingly:
 - If Speed is greater than or equal to 10, it sets Speed to 10, shows the LED display, and unplots the point at (4, 1).
 - If Speed is greater than or equal to 0, it unplots the point at (4, 1), shows the Speed value, and plots the point at (4, 1).
 - If Speed is greater than -10, it shows the Speed value multiplied by -1 and plots the point at (4, 1).
 - Otherwise, it sets Speed to -10, shows the LED display, and plots the point at (4, 1).

<https://makecode.microbit.org/1Aj4yi90Kh18> (Remote Control 7)

11.4 Completing the Boat

Now it is time to make sure your boat is complete. You may have already added these parts.

Refer to the **STEM SEALS YouTube video: SEA Final Boat Assembly**.

Your boat should now have all these parts:

- twin motors with propellers
- motor:bit, micro:bit, and power supply
- rudder
- water sampler
- eye hooks to tie the boat
- boat competition number

To complete your boat, it took the following:

- mechanical engineering to get the parts together to make the boat;
- electrical engineering to connect wires, sensors and actuators to make it functional;
- computer software engineering to write the code enabling to control the boat;
- physical science to test performance and operation.

Now we can use the boat for the missions it is designed for.

11.5 Function Summary: Remote Controlled Boat

Now that we have a fully functioning remote control that can control the speed, rudder position, and the water sampler, let's review the functions of the remote and what actions or inputs initiate the functions.

List of Functions		
Number	Action	Function
1	press A	increase speed
2	press B	decrease speed
3	press A & B	stop engines
4	tilt RC left	turn left
5	tilt RC right	turn right
6	shake RC	deploy sampler
7	press reset	start all over

The final code for our remote-controlled boat:

- https://makecode.microbit.org/_1q5i6o8qH5uo (**Boat 6**)
- https://makecode.microbit.org/_1Aj4yi90Kh18 (**Remote Control 7**)

Module 12: Floatation Test of the Finished Boat

Now that we have a completed boat and have added all of the parts such as the battery case and batteries, twin motor assembly, servo mounts and servos for both the rudder and the water sampler – we have added quite a bit of weight to the boat. We should reassess our boat’s weight, center of gravity, and buoyancy.

12.1 Weigh the Completed Boat

Let’s weigh the boat:

- Use the brown tubular spring scale (10 N / 1 kg / 1000 g), eye hooks, and rubber band to weigh your boat again.
- Measure in grams.

What is the weight of your boat now? _____

12.2 Center of Gravity

Let’s check the center of gravity:

- Balance the boat on the fulcrum. Place the boat with its axis perpendicular to the fulcrum.
- Measure the distance of the fulcrum from the bow tip: _____
- Measure the distance of the fulcrum from the stern: _____

12.3 Buoyancy

Let’s check the buoyancy of the boat:

- Fill the container with water to about 1” from the top.
- Place the boat in the water.
- Measure again, as you did with the hull from the top of the “deck” to the water line in four different places and record the measurements in mm (millimeters):

Data Table 2

All numbers in mm	bow	port	starboard	stern*	stern (* + 7)	pitch	roll
hull only							
completed boat							

- Copy your measurements from the hull only and enter the values in this table.
- Enter the values for the completed boat in the bottom row of the table.
- Compute the stern correction and pitch and roll of the completed boat.

Is your boat balanced? _____

Module 13: Autonomous Navigation



Think About It

- Do you know what autonomous navigation is?
- When might a boat need autonomous navigation?

13.1 The Process of Navigation

According to the Merriam-Webster Dictionary,

navigation (noun) is

- the act or practice of navigating,
- the science of getting ships, aircraft, or spacecraft from place to place,
- ship traffic or commerce;

the verb **navigates** means

- to travel by water, to sail,
- to steer a course through a medium such as operating an airplane,
- to get around, to move.

Let's suppose you want to take a trip from Miami to the Virgin Islands, but you want to pass through the Bahamas. Suppose this was the course you wanted to take:

1. Leave from Miami, FL
2. Pass by the Bahamas
3. Turn SE
4. Turn S
5. Arrive at the Virgin Islands

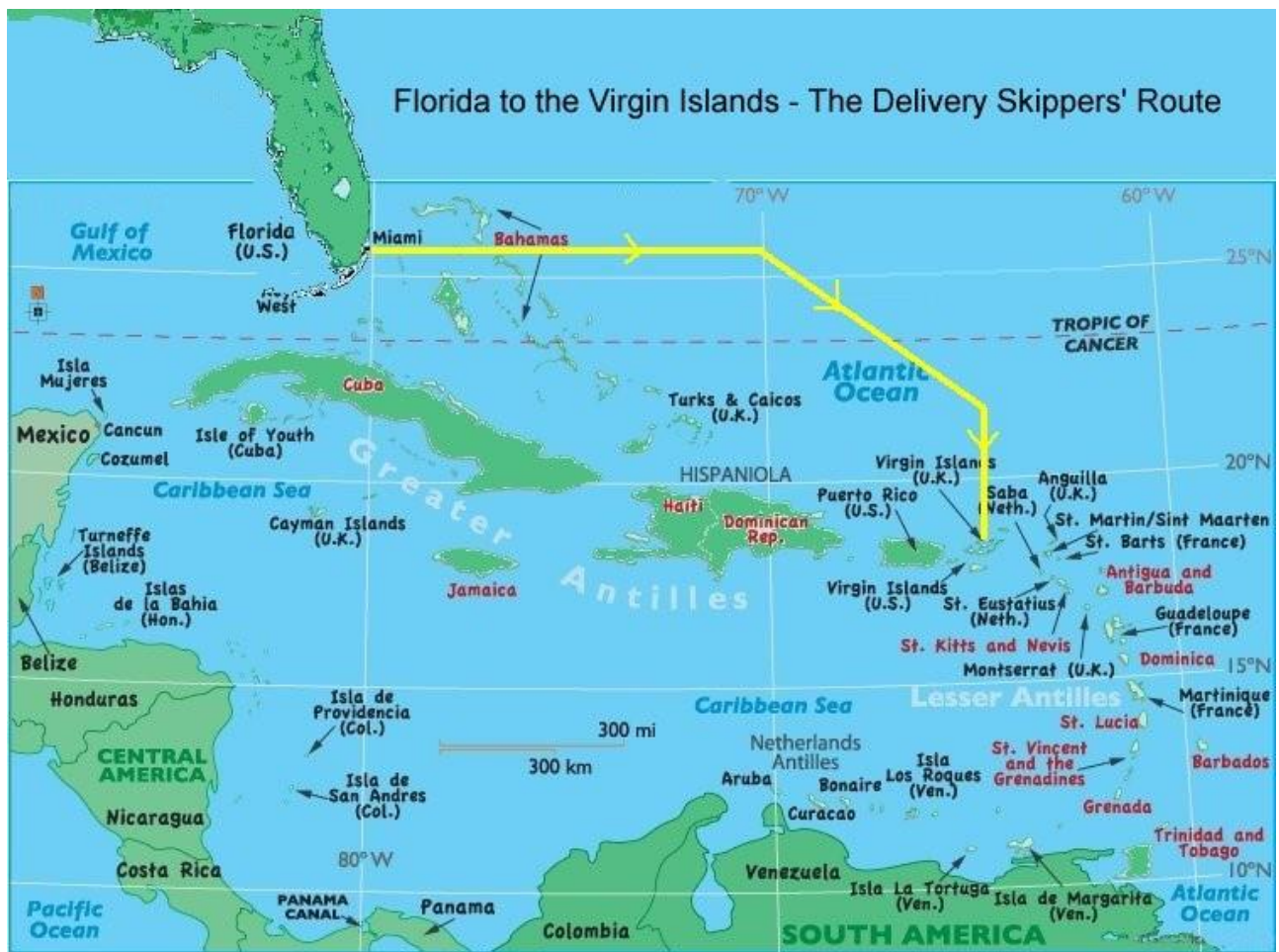
On land we would just follow the road or hike. But in the water or air – there is NO road!

How would you navigate this course?

You might think – Just steer in the right direction and you would get to your destination.

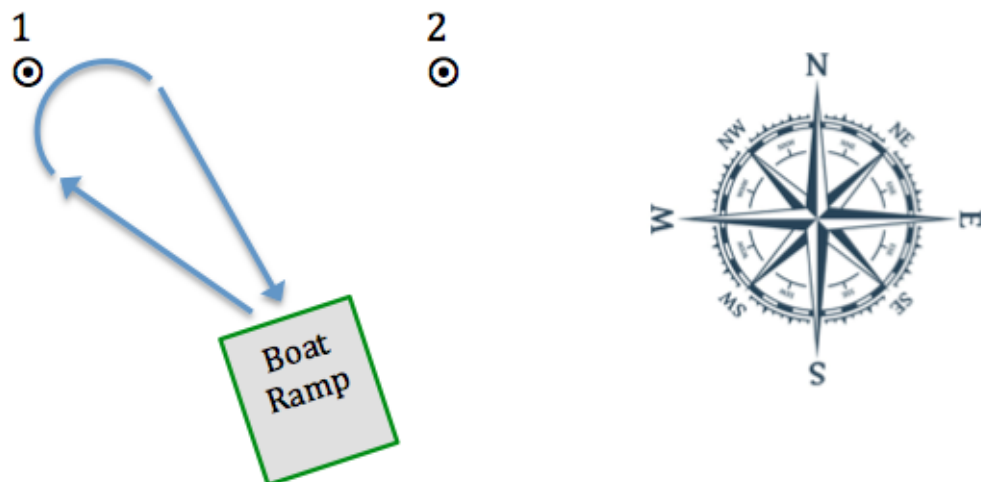
There is much more to it than that!

What do you think some of the obstacles might be to this?



We will not be sailing to a tropical island with our boat but navigating Lake Osceola. Up until now, we have used the remote control to navigate or steer our boat. Now, we will set the boat up to navigate autonomously.

Let's consider a simple course in the lake – leaving the launch point or boat ramp, traveling around a buoy, and returning to the ramp.



Here are some of the things you will need to consider:

- What direction to sail when leaving ramp?
- How long sailing straight?
- Which way to turn?
- How long to turn?
- Direction for return?
- How long sailing straight?

Those several things to consider, but not everything – you should also consider winds or currents.

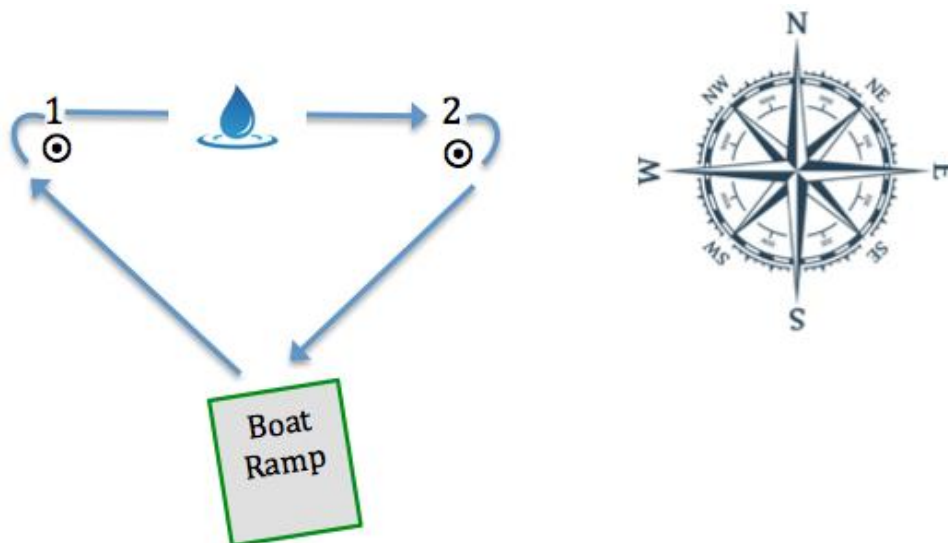
With wind, you need to think about:

- Wind direction?
- Wind Speed?
- How steady is the wind?

With current you need to think about:

- Currents in the water are difficult to judge.
- Current can change from location to location.
- Have a stronger effect on the boat.

When sailing around one buoy, we would need to consider these things on the trip out and the trip back. So, each part of the trip (out and back) would be considered a leg of the trip or course. If you plan a course around two buoys such as this:



You would have three legs in the course: Leg 1 to the buoy #1, leg 2 from buoy #1 to #2, and leg 3 back to the ramp. Other things to consider are which direction are the turns (left or right) and do you wish to take a water sample and when.

13.2 Planning a Course

It is helpful to plan your course based on the items from the last section.

Let's plan out a course.

On a piece of paper draw a big circle.

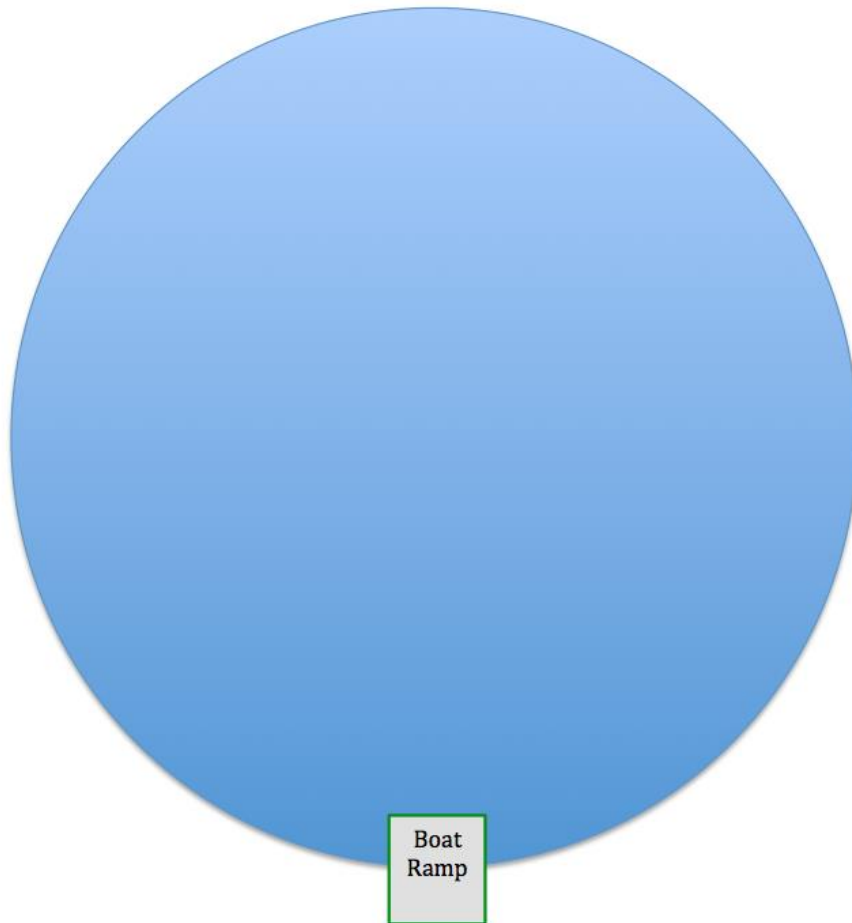
Let's imagine this is the lake we want our boat to navigate on.

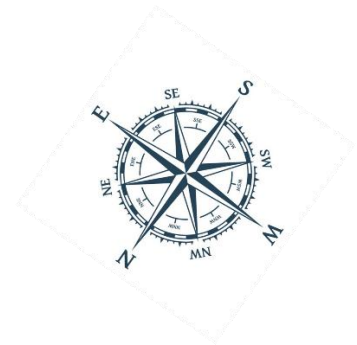
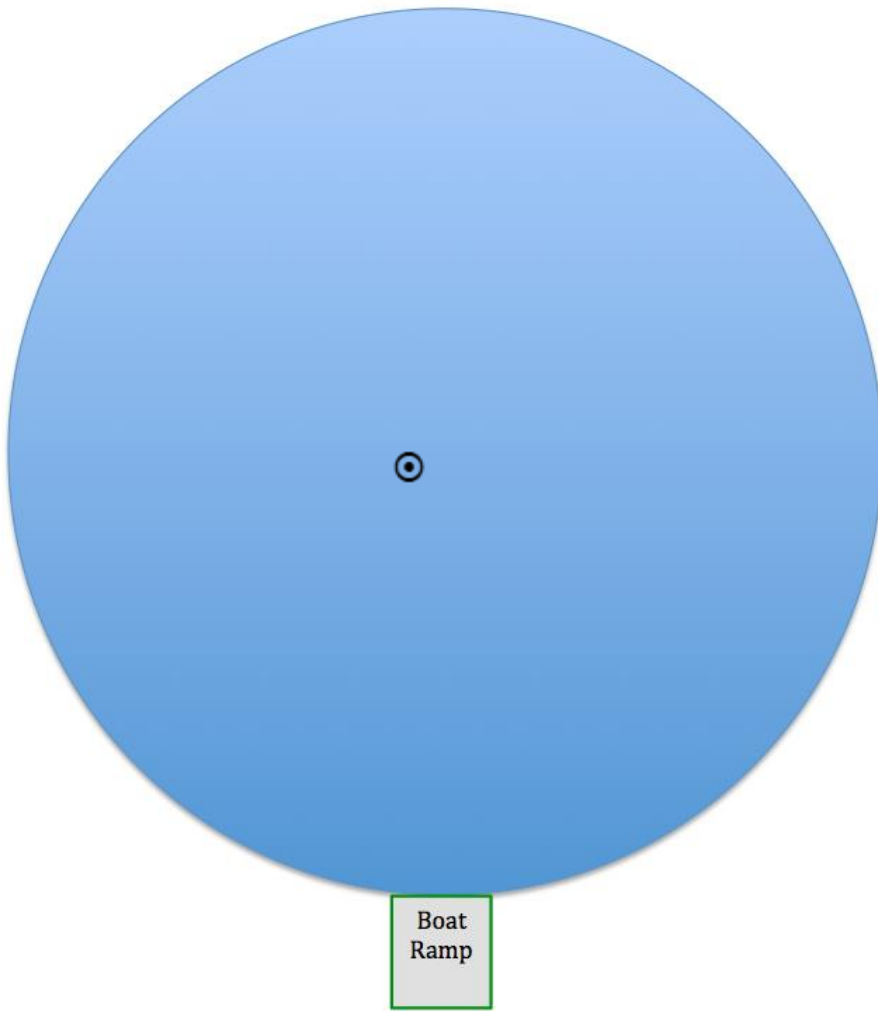
Use a protractor or ruler if needed for the following tasks.

Place the boat ramp at the bottom of the circle.

Place a cross with a small circle in the lake, a little more on the left side.

Let's mark the direction where North is somewhere in the corners.





- Determine the direction of the buoy as seen from the boat ramp: _____
- Determine the direction in which you want your boat to sail on the first leg if you want to circle around the buoy with a right turn: _____
- How much of a right turn does your boat have to make before it heads back to the ramp? _____
- Determine the direction the boat must sail to get back to the boat ramp: _____

13.3 Using the Micro:bit to Set a Course

Up until now, you have used the remote control to move your boat in the desired way, direction, or to perform a specific function.

Let's see if we can teach the micro:bit how to navigate a course! since Lake Osceola is large and it is easy to lose a wayward boat, we will first program our boat to navigate the mobile pond.



Time to code: Autonomous Navigation (SEA Navigation 20 Pond)

1. For the Autonomous Navigation, you will use the supplied link and make two adjustments.
2. **SEA Navigation 20 Pond:** [https://makecode.microbit.org/ DzXdfjAdTJeV](https://makecode.microbit.org/DzXdfjAdTJeV)
3. Click on “edit” to open the project in MakeCode.
4. In the “on-start” block:
 - Find the “radio set” block and change the number to your boat ID #.
 - Find the “RudderStraightAt...” block and enter the number that you found earlier that set your rudder straight.

5. Load the code on the micro:bit for the boat.

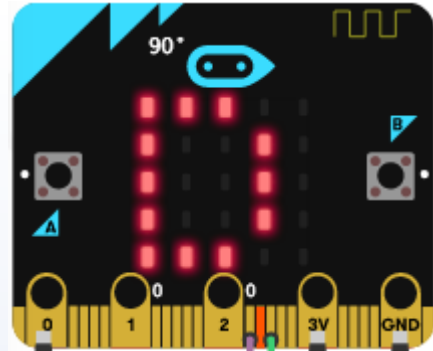
```
on start
  play scale up middle
  radio set group 17
  set RudderStraightAt to 93
  set TimeStep to 1
  set SamplerUp to 131
  set SamplerDown to 51
  set RudderGainControl to 0.9
  set RudderMax to 45
  analog write pin P1 to 0
```



Try This: Setting a Course on the Micro:bit

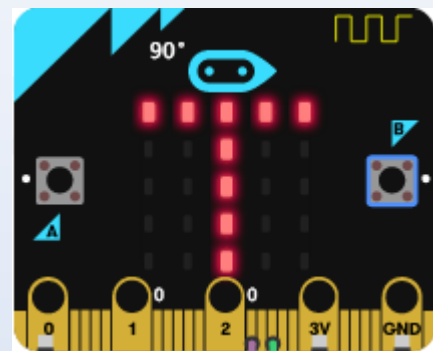
1. Make sure you have “SEA Navigation 16” loaded on your micro:bit.
2. Insert it into the boat and turn the motor:bit on.
3. You will need to calibrate the compass on start up.
 - Take the boat in both hands and tilt it until all 25 of the LEDs are on.
4. The blinking “D”:

- “D” is for the **direction** you wish the boat to go on the first leg of the course.
- Set the boat on the dry dock and point the bow in the direction you want the boat to travel.
- Then press button B to confirm your direction.
- The moment you press button B the microbit will read the compass and take this direction as the direction for the first leg. So, make sure you point the boat in the direction you want it to go. Then press button B.



5. The blinking “T”:

- “T” is for the **time** you wish the boat to travel on the first leg of the course.
- Press button A to set the time increment.
- A number 1 will be displayed which means 1 second.
- If you press button A again the number 2 will be displayed which means going out for 2 seconds.
- The 10 displayed after pressing button A ten times means you want your boat go out for (10 X 1 sec = 10 seconds).
- After pressing button A 20 times the micro:bit shows a zero for 0 sec!
- Then pressing button A will start all over again with 1 for 1 sec, 2 for 2 sec, etc.
- Therefore, with button A you choose how long you want your boat to sail on the first leg. The choices are 1 sec, 2 sec, 3 sec, ... all the way to 19 sec!
- Once you decide how long you want the boat to travel on the first leg by pressing button A, you confirm your choice by pressing button B.

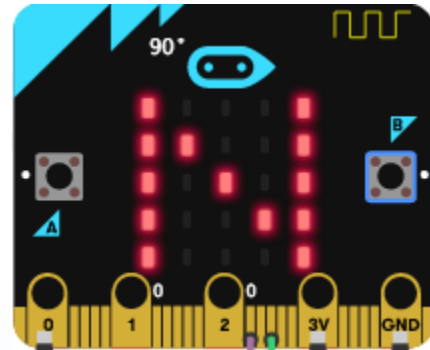




Try This: Setting a Course on the Micro:bit (continued)

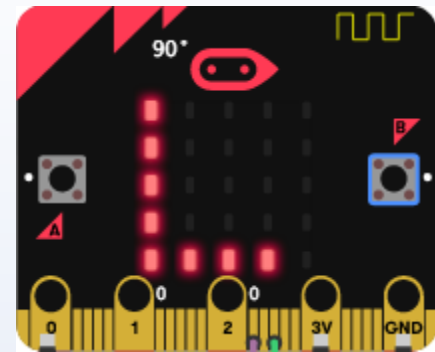
6. The blinking “N”:

- “N” stands for no water sample to be taken on this leg.
- If you do not want to take a water sample on the first leg, press button B to confirm the “N” or no.
- If you press button A, you can toggle between a “N” for no sample or a “Y” to take a sample.
- Once you make your choice, press button B to confirm.



7. The blinking “L”:

- “L” means turn left at the end of the leg.
- Again, you can press button A to toggle between “L” for left turn or “R” for right turn.
- Once you make your choice, press button B to confirm.



You have now successfully set the course for leg one:

“D” for direction

“T” for time

“N” or “Y” for a water sample

“L” or “R” for direction of the turn

The micro:bit now will display the letter “E”. It is asking you for the direction of the second leg. “D” was for the direction for the first leg and since “E” is next in the alphabet, we will use it for the direction for the second leg. “F” comes after “E” and therefore would be used for a third leg and so on.

For practice, let’s assume only one leg for the course.

8. To confirm you do not wish to program any more legs, you would press button A.

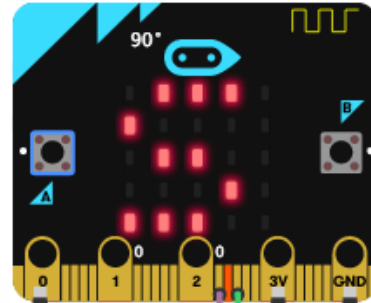
- If you press button B at this point (at a letter prompt for the next leg direction such as “E” or “F”) you will have set that next leg or press reset and start over the process.



Try This: Setting a Course on the Micro:bit (continued)

9. The blinking “S”:

- “S” is for **setting the speed**.
- After pressing A to confirm you are finished programming any more legs, you will get a blinking “S”.
- By pressing button A multiple times you can choose a speed for the course.
- The micro:bit displays numbers 1, 2, 3, ... 9.
- Slow speeds are low numbers and the fastest speed is number 9.
- When pressing button A after number 9, the cycle starts all over again beginning at speed zero, then 1, 2, ... etc.
- You probably know already: By pressing button B you make your choice. But WAIT!!! The moment you press the B button after speed selection the micro:bit will start autonomous navigation. You better be ready!

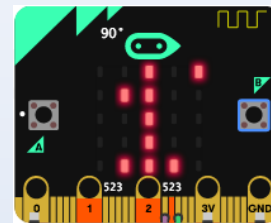


10. Set the boat on the dry dock (for practice) and press button B.

- A countdown will begin on the micro:bit display beginning with 9.
- This will allow you time to place the boat in the water or dry dock.

After the countdown, the micro:bit shows the number 1 – the number of the leg it is sailing, and a blinking dot to tell it is actively navigating the course.

- The engines come on with the selected speed,
- The rudder corrects for potential course corrections,
- The engines stop after course completion, and
- The micro:bit displays a cross to indicate it is done.



After it is finished running this course, you can choose to run the same course by pressing button B. Press button B and observe the rudder while your boat is sitting on the dry dock.

- Do you see the rudder moving?
- Reading the compass yields slightly different directions, which results in correctional movements of the rudder.

While running the course turn the heading of your boat about 30 degrees to the left and watch the rudder.

- Do you see that the rudder moves to navigate the boat to turn right?
- Of course, on the dry dock, the boat will not change direction. But it is an easy way to test it the boat engines, rudder, and micro:bit are working correctly.
- Now turn the heading of the boat about 30 degrees to the right.
- Does the boat steer in the opposite direction?

13.4 How does the SEA Navigation Program work?

At first it might seem confusing to program multiple legs of a course since the letter designations change for direction (D, E, F.....) and time (T, U, V...) but with practice you will soon get the hang of it. Practice makes perfect!

Below are some tables that will help by showing you what letter is displayed on the micro:bit and what action you should complete for a specific task.

Example: One-Leg Course

	Direction	Time	Sample	Turn	Speed
Display Leg 1	D	T	N / Y	L / R	E / S
Action Leg 1	turn boat	choose with A	choose with A	choose with A	choose with A
Confirm by	press B	press B	press B	press B	press B

- Pressing button B means “confirm choice”.
- Pressing button A means, for Time, Sampling, Turns and Speed selection “display next choice”.
- The choice for Direction is made by turning your boat physically!
- Pressing button A in this case means “I don’t want to add another leg. Let me choose the speed!”
- The letters E, F, G, H, I, J, K are the prompts for directions for legs 1, 2, 3, 4, etc., respectively, up to seven legs.
- The letters T, U, V, W, X, Y, Z are the prompts for times for legs 1, 2, 3, 4, etc., respectively, up to seven legs.

Example: Two-Leg Course

	Direction	Time	Sample	Turn	Speed
Display Leg 1	D	T	N / Y	L / R	E
Action Leg 1	turn boat	choose with A	choose with A	choose with A	
Confirm by	press B	press B	press B	press B	(next leg)
Display Leg 2	E	U	N / Y	L / R	F / S
Action Leg 2	turn boat	choose with A	choose with A	choose with A	choose with A
Confirm by	press B	press B	press B	press B	press B

- By NOT pressing button A when “E” for the Direction of leg 2 is displayed, we need to turn the boat again to let the micro:bit know in which direction we think the boat will come back to the ramp.

- We again confirm the Direction for leg 2 with button B.
- The micro:bit displays “U”, the letter following “T” in the alphabet for Time selection.
- We then choose the time on leg 2 with button and confirm this time with button B.
- The choices if we want a water sample taken and which direction to turn after this leg are made the same way as we did for leg 1.
- At the end of the leg, letter “F” is displayed for the choice of leg 3. By pressing button A instead of confirming with button B we enter the selection for the Speed.
- The confirmation of the chosen speed with button B also starts the autonomous course navigation with the countdown.

Example: Three-Leg Course

	Direction	Time	Sample	Turn	Speed
Display Leg 1	D	T	N / Y	L / R	E
Action Leg 1	turn boat	choose with A	choose with A	choose with A	
Confirm by	press B	press B	press B	press B	(next leg)
Display Leg 2	E	U	N / Y	L / R	F
Action Leg 2	turn boat	choose with A	choose with A	choose with A	
Confirm by	press B	press B	press B	press B	(next leg)
Display Leg 3	F	V	N / Y	L / R	G / S
Action Leg 3	turn boat	choose with A	choose with A	choose with A	choose with A
Confirm by	press B	press B	press B	press B	press B

- Summary: Toggling through choices are made with button A – including the choice for adding a new leg or instead finalizing the course. Confirmation of the choices are made by pressing button B.
- Remember: Pressing button B after Speed selection starts the course!

13.5 Navigating Lake Osceola

Now that you have the autonomous navigation and understand how the programming of the micro:bit functions as well as practiced navigating in the mobile pond, it is time to move to Lake Osceola! Remember that when pressing button A at the prompt for time (T), each time you pressed the button equaled 1 second the boat would travel for a maximum time of 19 seconds. We will need more time for the boat to make a course in the lake.

By changing the TimeStep variable in the “on start” block to 30 instead of 1, we can increase the time the boat travels on each leg of the course. This small change multiplies the numbers we used for the duration of the legs with 30. A leg with duration 1 now lasts 30 seconds, with 2 it travels 60 seconds, etc. The maximum time for a leg is then $19 \times 30 = 570 \text{ sec} = \text{almost } 10 \text{ minutes!}$



Use this code for practice in the lake. The TimeStep variable has already been changed but you will need to change the “RudderStraight” and the “radio set” to your Boat ID #. (See page 28)

SEA Navigation 20 Lake: <https://makecode.microbit.org/Lta2XCR6KimW>

Module 14: Navigation Practice

Now it is time to practice your boat maneuvers at both the mobile pond and Lake Osceola. Below is a description of each of the competition events. You should practice each event before the competition day.

Mobile Pond Events:

A) Buoy Race (one buoy):

- The goal of the buoy race is to control the boat using autonomous navigation and travel around one buoy and back to the starting position with the fastest time.
- Use the SEA Navigation 20 Pond code for the race.
- Scoring for this race will be for time and accuracy.
- Total score equals total seconds plus penalty seconds. A second will be added to the total time for each hit or touch that the boat makes if it encounters the buoy or the mobile pond during its course.

B) Figure Eight:

- The goal of the figure eight race is to control the boat using the remote control and make a figure eight pattern around two buoys located in the mobile pond with the fastest time.
- Use the Boat 6 and Remote Control 7 code for this race.
- Scoring for this race will include both time and accuracy.
- Total score equals total seconds plus penalty seconds. A second will be added to the total time for each hit or touch that the boat makes if it encounters the buoy or the mobile pond during its course.

C) Bumper Boat Rally:

- The goal of this race is to be the last boat with your plastic golf ball still perched atop the wooden golf tee in the front of your boat.
- Students will compete four at a time by placing their boat into the mobile pond with the golf ball on the wooden tee.
- Students will race around the mobile pond trying to avoid collisions with other boats or the walls of the pond. Collisions are likely to cause the golf ball to fall and thus eliminating that student from the race.
- Several elimination rounds will ensue until there is one winner over all.

Lake Osceola Events:

A) Buoy Race (two buoy):

- The goal of the buoy race is to control the boat using remote control and travel around two buoys and back to the starting position with the fastest time.
- Use the Boat 6 and the Remote Control 7 code for this race.
- Scoring for this race will be for time and accuracy.

- Total score equals total seconds. Boats that do not travel around both buoys will not be scored.

B) Water Sample Grand Finale:

- The goal of this race is to travel in the lake using autonomous navigation. the course is to travel around two buoys, take a water sample, return back to the dock, collect the sample and test.
- Below is a sample scoring card for this race:
 - Grand Finale: Water Sample Rally
 - Name:
 - Boat #:
 - Did they pass the buoy? Yes (no penalty/ zero)
 - Did they pass the buoy? No (penalty add 30 secs)
 - Did they go around the buoy? Yes (no penalty/zero)
 - Did they go around the buoy? No (penalty add 30 secs)
 - Complete the water sample analysis? Yes / No
 - Time: (for collection and analysis)
 - Total time (including penalties)
- The student with the shortest time after penalties are deducted will win the race.

Module 15: 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.

Resources:

Page 23:

BBC. "Introducing the BBC Micro:Bit - BBC Make It Digital." *YouTube*, 8 July 2015, www.youtube.com/watch?v=Wuza5WXiMkc.

Micro:bit Educational Foundation. "Introduction to the BBC Micro:Bit." *YouTube*, 19 Jan. 2021, www.youtube.com/watch?v=u2u7UJSRuko.

SparkFun Electronics. "Getting Started With Micro:Bit Part 1: Say Hello." *YouTube*, 10 Apr. 2017, www.youtube.com/watch?v=kaNtg1HGxbY.

Shawn Hymel. "Micro:Bit Tutorial Series Part 1: Getting Started." *YouTube*, 15 Aug. 2016, www.youtube.com/watch?v=ZIW_6rxYNBg.

Page 28

"Introduction." *Microsoft MakeCode*, makecode.microbit.org/courses/csintro/coordinates/overview.

---. "Micro:Bit LEDs." *YouTube*, 28 Feb. 2020, www.youtube.com/watch?v=eRhlaXqT-0w.

Page 29

Shahnawaz Parveen. "Create a Variable in Makecode Microbit." *YouTube*, 22 Apr. 2020, www.youtube.com/watch?v=vD4ywGNsZTA.

STEM SEALs North Florida. "Microbit - Why Variables?" *YouTube*, 1 July 2020, www.youtube.com/watch?v=HntpSgJIAKE.

Page 40

Florida Atlantic University, and GISELE GALOUSTIAN. *FAU TO DEVELOP ROBOTIC BOATS WITH a 'MIND OF THEIR OWN.'* 17 Feb. 2017, www.fau.edu/newsdesk/articles/fau-usv.php.

Roach, John. "Robotic Boats Have Mind of Their Own on Water." *NBC News*, 16 July 2013, www.nbcnews.com/technolog/robotic-boats-have-mind-their-own-water-6c10643235.

National Science Foundation. "Engineering Smarter Robotic Boats for Safer, Cheaper Work on the Water - Science Nation." *YouTube*, 4 June 2018, www.youtube.com/watch?v=R4CAZCCaUUg.

Pallone, Greg. "Students Build Underwater Robot, Use It to Study Indian River Lagoon Problems." *Spectrum News 13*, 19 June 2021, www.mynews13.com/fl/orlando/news/2021/06/18/students-build-underwater-robot--learn-about-steam?web=1.

Page 49

Buoyancy Diagram, Christopher AuYeung, CK-12 Foundation, CC BY-NC 3.0, accessed April 2021, <https://flexbooks.ck12.org/cbook/ck-12-middle-school-physical-science-flexbook-2.0/section/12.6/primary/lesson/buoyancy-ms-ps/>

Page 55

“Naval Architecture | Development and Principles.” *Encyclopedia Britannica*, 13 Apr. 2018, www.britannica.com/technology/naval-architecture/Strength-of-ships.

“Why Do Ships Float?” *Let’s Talk Science*, 24 Sept. 2019, letstalkscience.ca/educational-resources/stem-in-context/why-do-ships-float.

Prince, Bill. “The Basics of Hull Design Explained.” *Power & Motoryacht*, 11 Mar. 2022, www.powerandmotoryacht.com/boat-design/the-basics-of-hull-design-explained.

engineeringdotcom. “Boat Hull Design Concepts.” *YouTube*, 31 Mar. 2011, www.youtube.com/watch?v=cQiY5rea0kA.

Science Filmmaking Tips. “How Do Kayakers Use Buoyancy?” *YouTube*, 27 Apr. 2012, www.youtube.com/watch?v=O8buGJ-XdBI.

“Buoyant Boats.” *TeachEngineering.org*, 30 Aug. 2022, www.teachengineering.org/activities/view/duk_boat_mary_act.

Page 65

---. “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.

Page 87

Casual Navigation. “How Does a RUDDER Work?” *YouTube*, 7 Sept. 2018, www.youtube.com/watch?v=jWD-IxjdFrS.

ROWINTEL.COM. “How The Rudder Actually Moves the Boat, Generates ‘Lift.’” *YouTube*, 26 Feb. 2015, www.youtube.com/watch?v=L-UkOvb6TK8.

marineinsight. “How Does a Ship Turn in Water? #Rudder #Ship #Shipturn.” *YouTube*, 14 Apr. 2020, www.youtube.com/watch?v=k1YxpOg3r6U.

Servos Explained - SparkFun Electronics. www.sparkfun.com/servos.

Page 96

---. "Micro:Bit Accelerometer." *YouTube*, 28 Feb. 2020,
www.youtube.com/watch?v=UT35ODxvmS0.

Microsoft MakeCode. "Behind the MakeCode Hardware - Accelerometer on Micro:Bit." *YouTube*, 1
Dec. 2018, www.youtube.com/watch?v=byngcwjO51U.

Page 124

Tor Pinney's Homepage - a Cruising Sailor's Homeport. www.tor.cc/articles/flcarib.htm.

