Student Robotics 2019 Microgames

Welcome to the Student Robotics 2019 microgames! You will be learning how to use our kit by completing a series of challenges.

You must start with the "**A** good place to start" challenge (below). Once you've done this, you can move on to task sheet "**B**". Once you're done, show it to a volunteer, and they will check you've done it correctly. You must also show a volunteer after you've done each individual task to make sure they tick you off, otherwise they might not believe that you've done all the challenges in the task!

A good place to start

Before you start building your fantastic robots, we need to get you used to the kit.

1. Battery Charging

The really exciting part first! You **MUST** know how to safely charge the batteries we provide to you. The LiPo (Lithium Polymer) batteries we give you can contain ~120 kilojoules of energy! Before you plug a battery into the charger make sure you've shown a Blueshirt that you can safely charge a battery.

The best place to start if you don't know how to do something is always the docs, at srobo.org/docs. Find the instructions on how to use the battery charger in your kit. Note that there are two different models of charger and they may get hot, so be careful. Make sure you use the instructions for the one you have in your kit - check that your charger looks like the image at the top of the instructions.

You **MUST** show a volunteer before you plug a battery in.

Once you have shown a volunteer and started charging a battery, move on to the next task.

2. Assemble your kit

You should start by assembling your kit and getting used to what's in the box. If you need some help, the first place you should always look at is the docs at srobo.org/docs. At srobo.org/docs/kit/assembly there's a video tutorial on how to put it all together.

Your kit won't turn on unless you have a jumper cable or switch plugged into the on|off terminal! If there isn't already one in your kit, you can make a jumper cable by wiring both terminals of a 5mm camcon (medium-sized green connector) together, and plugging it into the slot next to the on|off button.

Take your time, once you're happy that you've assembled your kit, and you know what all of the components so you can move on to the next task.

3. Hello World

Firstly, you should get your code running on your kit. The traditional program to test things are working is to print "Hello, World!" to the logs. You'll need to write the code in the IDE at srobo.org/ide/ (your teachers should have been sent these in an email), and you can view the logs on the USB stick, or on your teams tablet, or even on your own mobile phones if you connect to the robot's wifi.

#See srobo.org/docs/kit/tablet for information on how to connect your phones / tablets.

Once you're printing "Hello, World!", you can add the lines below. This bit of code isn't meant to be human readable, so don't worry if you haven't a clue what it does!

```
import sys;
print('{1}{0}N{2}{3}{4:01X}{0}{5}{0}L'.format(sys.version_info[0] - 2,
chr(ord('[')-1), 5-2, chr(32), 12, 'RTN'[::-1]))
```

Too lazy to type? Code is at goo.gl/WPnL5w

As an additional challenge make the program print the same message to the log 10 times. **Hint**: Use a loop!

Once you're done, show it to a volunteer, telling them what answer you got.
The volunteer will then recommend which task to do next.

B prepared

1. Servo Wiggle

Note: *collect a servo from the volunteer desk*
Servos are pretty important to a successful robot design, they allow you to rotate objects to a specific angle. Servos are useful for robot arms (with a counter-weight), though we'll let your imaginations run wild for any other uses you can think of.

Connect up a servo to your robot (ask a volunteer for one of these) and then get it moving by following the instructions on the docs. Show a volunteer once you're finished. Try and see what the movement limits are of the servos, they don't turn all the way.

Note: *Make sure the servo is connected to the servo board!*

Please return the servos afterwards as other teams may need them.

2. Use the robot simulator

Although you don't have any motors (or even a robot) yet you can still familiarise yourselves with the software environment using the simulator! Head to the docs to find the simulator.

Motors can be controlled by a simple interface. A good place to start would be the docs at srobo.org/docs/programming/sr/motors/.

The simulator will run most robot code you give it, so use it to make the robot drive in a circle, and if you really fancy a challenge make it pick up a cube. Use the docs to find out how.

When you're finished, show a volunteer.

3. Post on the forum

The forums are the best place to go to ask volunteers for help if things aren't working or to ask any other questions you may have. They're also a great place to chat with other competitors about their robots, and exchange ideas on strategy, design or construction. Post on the forums introducing yourselves to the other competitors, then show a volunteer.

4. Broken Code

Developers often come across many problems, be it missing brackets in code or a cat jumping onto their keyboard — hopefully you'll only be dealing with the former.

To help prepare you for this, we would like to look at a piece of code, and point out any errors, and then fix them. Copy the code below (also found on pastebin.com/XZifM0F4) into the IDE and look at the code. Fix it, and then run your code in the simulator, which can be found in the docs.

This can be a very tricky task for people who are new to Python, if you need help, first look in the docs at srobo.org/docs/troubleshooting/python. If you have any other issues, ask a volunteer!

Once you've fixed this, show it to a volunteer, and they'll recommend which challenge to do next.

```
from sr.robot import *

R = robot()


def drive_forward(speed):
    r.motors[].m0.power = speed
    r.motors[].m1.power = speed


def drive_backwards(speed):
    r.motors[].m0.power = -speed
    R.motors[].m1.power = -speed


def look_for_token_markers()
    markers = R.see()
   arena_markers = []
    for m in markers:
        if m.info.marker_type = MARKER_ARENA:
            arena_markers.append(m)
    return arena_markers

while True:
    markers = look_for_token_markers()
    if markers:
        Drive_forward(20)
```

C the code

1. Vision

Using your robot's camera, you'll be able to find your robot's position in the arena and locate markers. You don't need to write any code to identify the markers in an image though - we've done that for you. For this task, you should connect the camera and write a program that looks for markers and displays information about any it can see. There is a page on the docs about using the vision system.

To complete this challenge you should collect the pattern of four markers from a volunteer and then use your webcam and code to find out what the markers are spelling out from left to right.

The volunteer will also give you a numbered conversion sheet from numbers to letters. Each team will receive a different key so no cheating!

You can convert the token number to a letter using this key (using the `code` property of `MarkerInfo`).

Once you're done, tell a volunteer the word and the number of your key.

2. Corners

It can also be handy to know which corner you are in. Make your robot blink an LED depending on which corner you are assigned to.

Look at srobo.org/docs/programming/sr/#OtherRobotAttributes for more information. The corner you start in will be set with a special USB key during the competition; in the meantime, this attribute can be set manually like so:

Get an LED from a volunteer and connect it to your Ruggeduino using a suitable output. We suggest using output 13 and GND. Use the docs to work out how to control the pin so that the LED flashes.

For corner 0, make the LED flash once a second. For corner 1, once every two seconds; corner 2, once every three seconds; corner 3, once every four seconds.

Once you've demonstrated this to a volunteer, move on to the next task.

3. Bug Hunting!

Developers often come across issues with their code. It may run, it just might not be running how you expect it to!

Debugging is a very useful skill to have. In this case, you should be able to use the code but it won't work properly. We have placed a few small errors in the code that will prevent it from running as we'd expect.

To help prepare you for this, we would like to look at a piece of code, and find any errors, and then fix them. Copy the code from http://pastebin.com/CrjdBsxj into the IDE and run it on the simulator. If you haven't used the simulator yet then head to the docs to find out how to use it.

The first time you run it you should have a few errors appear. Use these errors to find the problems and fix them. Don't worry if you have no idea what they mean, either ask a volunteer or feel free to google it!

The expected output on the logs should include: 3, 4, 31, 45, 53, 212 **(in order)**

This can be a very tricky task for people who are new to Python, if you need help, first look in the docs at srobo.org/docs/troubleshooting/python. If you have any other issues, ask a volunteer!

Once you've fixed this, show it to a volunteer, and they'll recommend which challenge to do next.

Z oom to success

1. Lift off!

A useful feature of python is the ability to pause your code for a given amount of time.
To do this, we will use a function in the "`time`" module. You can look-up how to do this and more in the Python documentation which is available at srobo.org/docs/python/.

The power board has a small buzzer on it. You may be able to spot it, it's right next to the fan. It's primarily used for diagnostics (the beeps you hear when you switch the kit on indicates the version of the software it's running) but you can also program your robot to beep at a particular tone for a set length of time. In other words, your robot is a musical instrument!

We would like you to make a rocket countdown that starts by beeping 10 times with a one second delay between each beep and then one long beep at the end. Once that's working, show it off to a volunteer.

2. Listen for the competition

Note: *collect an ultrasound sensor and four hookup wires from the volunteer desk*

Sometimes you may find yourself limited by the capabilities of our kit, so we try to make it easy for you to add your own extensions to it. In this task, you'll add the ability to measure the distance of an object from an ultrasonic proximity sensor, using the Ruggeduino. This will be useful for finding the distance to cubes or other objects in the final competition.

You'll first need to download and install the Arduino IDE, if you don't already have it (it is already installed on the University computers). You can get it from arduino.cc/en/Main/Software. If you have trouble installing it, check the Arduino website. If you're still stuck, ask a volunteer!

Next, try adding a custom command to the Ruggeduino that does something simple, like blink an LED. The docs page titled "Ruggeduino custom firmware" may be very helpful for this.

The ultrasonic sensors you'll be using have four pins on them, labelled Vcc, Trig, Echo and Gnd. Their connections to the Ruggeduino should be made as follows:

- Vcc (sensor) to 5V (Ruggeduino)
- Trig (sensor) to 2 (Ruggeduino)
- Echo (sensor) to 3 (Ruggeduino)
- Gnd (sensor) to Gnd (Ruggeduino)

It's operated by sending a short pulse to the sensor's Trig pin. The sensor then generates a pulse on the Echo pin in response; the duration of this pulse indicates how far the away an object in front of the sensor is. Specifically, the duration of the pulse you send should be 10 microseconds, and the duration of the pulse you receive in response is equal to **round-trip**

distance between the object and sensor divided by the speed of sound (343 metres per second).

Some Arduino functions that you might find useful for implementing this are:

- pinMode
- digitalWrite
- pulseIn
- delayMicroseconds

Their names are fairly self-explanatory, but more detailed information can be found in the Arduino documentation: https://www.arduino.cc/reference/en/

3. The path to victory

This challenge involves navigating a 7 by 7 grid using only the markers provided to you, for this game you will be provided with a sequence of markers that will give you a direction to follow. Using functions in the vision section of the docs. In this challenge you start in the centre of the grid, and trace out a path, this path is given by a sequence of markers, these markers relate to one of 4 directions you can move, up, down, left or right. If you see markers 0,1 or 2 move up one square in the grid, if you see markers 3,4 or 5 move right one square, if you see markers 6,7 or 8 more down on square and finally if you see markers 9,10 or 11 move left one square, fill in the squares you have been in to trace out a path.

To start this challenge come to the front and take a 7 by 7 blank square (it should have one black square in the centre, this is your starting square) and a sequence of markers (these are the ones to read and follow). There are 5 sequences available so remember which one you chose. Show this to a volunteer so they can check its correct!

Hint : check the marker object to find out which number the marker is