Student Robotics 2017 Microgames

Welcome to the Student Robotics 2017 microgames! You will be learning how to use our kit by completing a series of challenges. This year, the game is '**Easy as ABC**'.

You must start with the "**A** good place to start" challenge and then you can work through the other challenges at any order you want (we don't recommend you do the "**C** the code" one first!). You must complete each challenge in full before moving on to the next one, a volunteer will check this as you ask them for help. 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.

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

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. 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.

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 do 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/, 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 line 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!

```
print('S{2!s}{0:02X}{1:+4.0f}'.format(176, 504.25*-4, b'\x52\x4f'))
```

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), 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 from pastebin.com/370TSWRw 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.

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 a camera 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've 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.
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.

Make your robot blink an LED depending on which corner you are assigned to.

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, twice a second; corner 2, three times a second; corner 3, four times a second.

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.

A to Z

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. The sound of ~~Music~~ Buzzers!

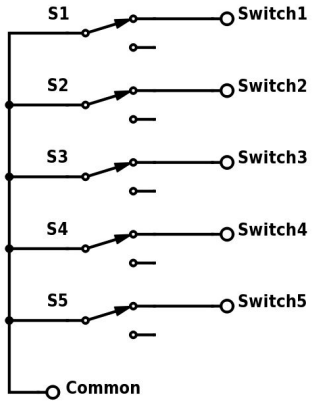
note: *collect a microswitch keyboard from the volunteer desk*

Let's start off by writing a program for your robot that will play a short tune. Check out the documentation for the power board to find out how to operate the buzzer. We recommend you use the pentatonic scale for your tune. If you don't have much musical experience, the pentatonic scale is the five notes (C, D, E, G and A). It usually sounds nice when played in any order, so for the sanity of those around you, you should keep it to these.

So now you've made your robot emit some noises that could vaguely be described as "music", but your instrument is quite cumbersome to use; you change and redownload your program every time you want to play a different note! Let's see if we can make it a bit simpler...

We've made some keyboards that you can use to control your robot's buzzer. Each board has five microswitches on it. Can you connect up the keyboard to your robot and program each key to play a different note when pressed?

The keyboard has six wires attached to it that can be plugged into your robot's Ruggeduino: one "Common" wire, and another wire for each switch. Normally, each switch's wire is electrically connected to the Common wire; when a switch is pressed, it will break the connection. You'll also want to use the Ruggeduino's builtin pull-up resistors.



Keyboard schematic

When you're done, please return the keyboard so that another team can use it.

3. Keyboard Orchestra!

Keep your friends close, and your enemies closer…

It's very useful to communicate with other teams during the competition. To complete this challenge, another team and your own must produce a musical rendition between yourselves and present it to a volunteer. If no other team has been as speedy as your own then let a volunteer know and they'll give you another task to do while you wait.

Once the volunteer is satisfied with your musical prowess you can move on to the next task.