ALBIR OPENMV\_BOT MANUAL

Contents

[Module repository 2](#_Toc158230524)

[Software development notes 2](#_Toc158230525)

[General notes 2](#_Toc158230526)

[Moving files to your robot 2](#_Toc158230527)

[Using OpenMV IDE 2](#_Toc158230528)

[Running scripts over wifi 3](#_Toc158230529)

[Colour detection threshold tuning 4](#_Toc158230530)

[Hardware usage notes 5](#_Toc158230531)

[General notes 5](#_Toc158230532)

[Power cycling 5](#_Toc158230533)

[Servo zero-point offset tuning (important) 5](#_Toc158230534)

[Wheel speed tuning instructions (optional) 6](#_Toc158230535)

[Common errors and fixes 7](#_Toc158230536)

[Trying to run a script gives me Err19! 7](#_Toc158230537)

# Module repository

<https://github.com/dragonflyneuro/ICL-BioEng-ALBiR-PixyBot>

# Software development notes

## General notes

* OpenMV development is best done in micropython, which has some important differences to standard python. As a result, some common modules used with python are not available for use.
* Please read through all the classes and functions we provide in our default codebase and have a look through the OpenMV documentation (<https://docs.openmv.io/index.html>).
* For the first assignment, you will not be able to use your own bots and will have to rely on the default classes and functions we provide.
* For the rest of the course, we highly recommend modifying and experimenting with the default codebase and making it yours – get the most out of your bot!
* It may be helpful to write code to have one of the LEDs on the OpenMV turn on when the script is running so you can tell that the bot is active.
* Do most of your testing over USB cable for ease of development and to keep the Lipo battery charged - you will not be able to access the internet if connecting to the bot wirelessly (see below).

## Moving files to your robot

When you plug the OpenMV\_bot into your computer via USB, you should be able to access the contents of the SD card in the OpenMV as a USB device. All files on board the SD card can safely be deleted if you want a clean slate.

***We HIGHLY suggest you edit and run files directly in the SD card to avoid potential problems with cached modules not updating every time you run a script.***

## Using OpenMV IDE

All compiling of code is done on-board the OpenMV, so you will not be able to test code when not connected to the OpenMV\_bot. OpenMV IDE allows you to connect to the bot over USB cable with ease. The USB button on the bottom left of the IDE is used to open serial connection (serial monitor can be opened using a tab just to its right), and scripts can be run and stopped using the button below this button (Figure 1, red and blue circles).

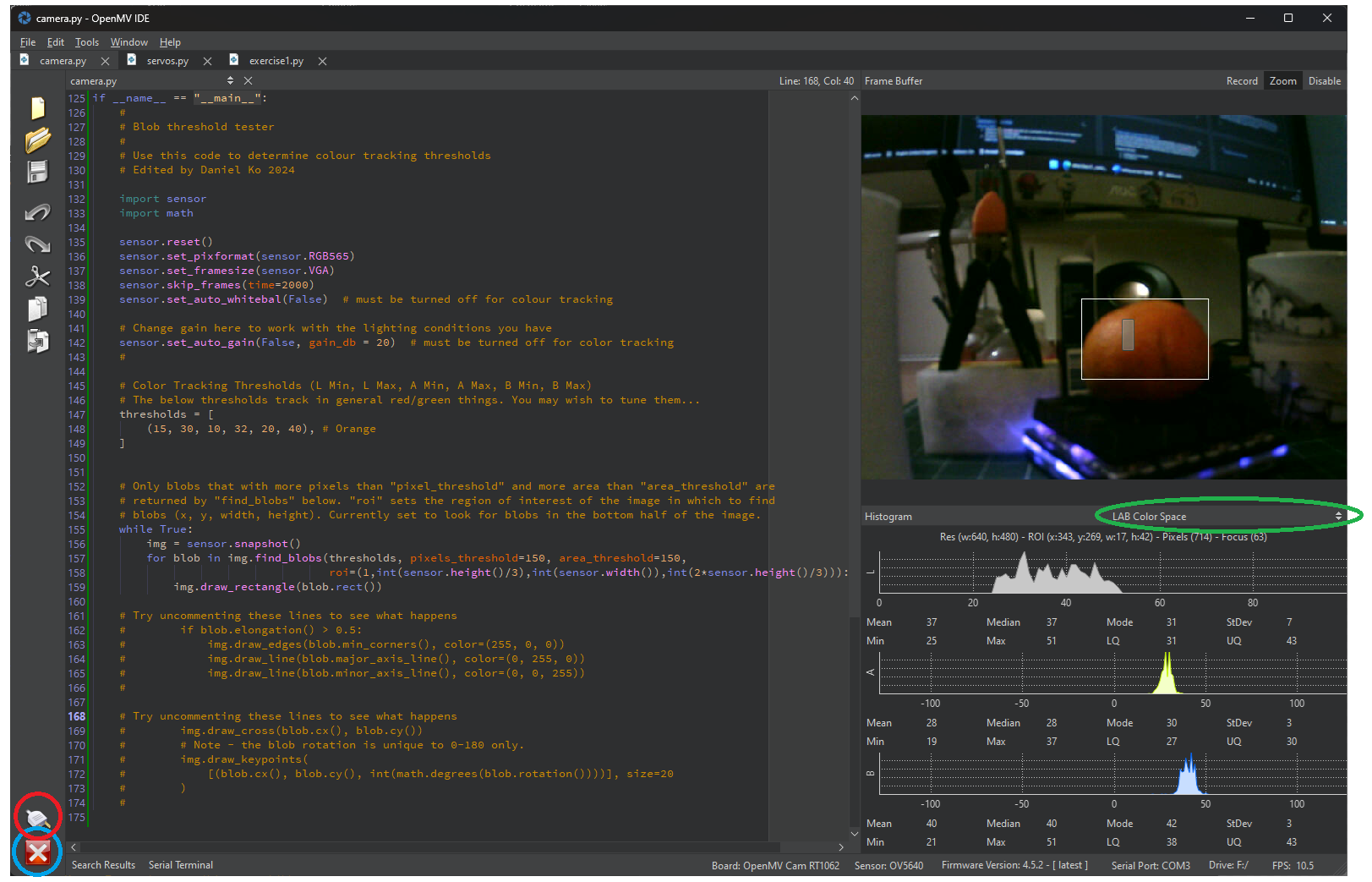


Figure 1: OpenMV IDE running colour tracking example. The buttons at the bottom left corner of the IDE are used to connect/disconnect to/from the OpenMV (red circle) and run/stop the current script open on the IDE (blue circle). When camera.py is run as a script, the frame buffer will be streamed from the OpenMV for viewing. In this window, the white no-fill rectangle denotes a detected colour’s boundary, and the grey rectangle is the region of interest marked out by the user to focus the colour space histogram below. The colour space should be set to LAB using the dropdown (green oval) as the colour tracking thresholds are set using minimum and maximum values in LAB space. The histogram window has basic statistics printed out just below the histograms for each colour channel that you can use to determine thresholds.

## Running scripts over wifi

For cable-free testing, you can execute scripts on the OpenMV\_bot over wifi. You can save ***MV\_remote\_exec.py*** as ***main.py*** onto the bot’s SD card to make sure the bot runs it on startup, regardless of if it is connected to a computer or not. This sets up the OpenMV to act as a wifi access point for your compuer to connect to. From the next power cycle, the bot will listen over wifi for scripts to be sent over and executed. You should modify ***MV\_remote\_exec.py lines 11-12*** to set the SSID and password of your access point so you do not accidentally connect to someone else’s bot.

Once your computer is connected to the access point, you need to find the IP address of the bot (e.g. ipconfig on command line) and enter it into ***send\_script.py line 19***. Additionally, you need to enter the path of the script file on your computer that you want to execute (***line 21***). Finally, run ***send\_script.py*** on your device. You will not be able to use OpenMV IDE for this purpose so will need another way to run python scripts (e.g. terminal with python installed, Visual Studio Code).

***MAKE SURE TO NOT HAVE ANY INFINITE LOOPS IN THE SCRIPT YOUR SEND OVER!!! YOU WILL NOT BE ABLE TO STOP EXECUTION VIA SOFTWARE SO YOU WILL HAVE TO CATCH YOUR ROBOT AND HARDWARE RESET!!!***

## Colour detection threshold tuning

You can run **camera.py** as a script using OpenMV IDE to tune colour detection thresholds (check the end of the file to set thresholds). Any colours detected using the specified LAB colour space (Figure 2) thresholds will be marked using a rectangle in the image buffer at the top right of the IDE (Figure 1). You can set the colour space histograms shown below the image buffer using a dropdown (Figure 1, green oval). The colour histograms can be focused on specific regions of interest by clicking and dragging rectangles in the image buffer.

You can also use Tools->Machine Vision->Threshold editor to adjust the thresholds using sliders in OpenMV IDE.

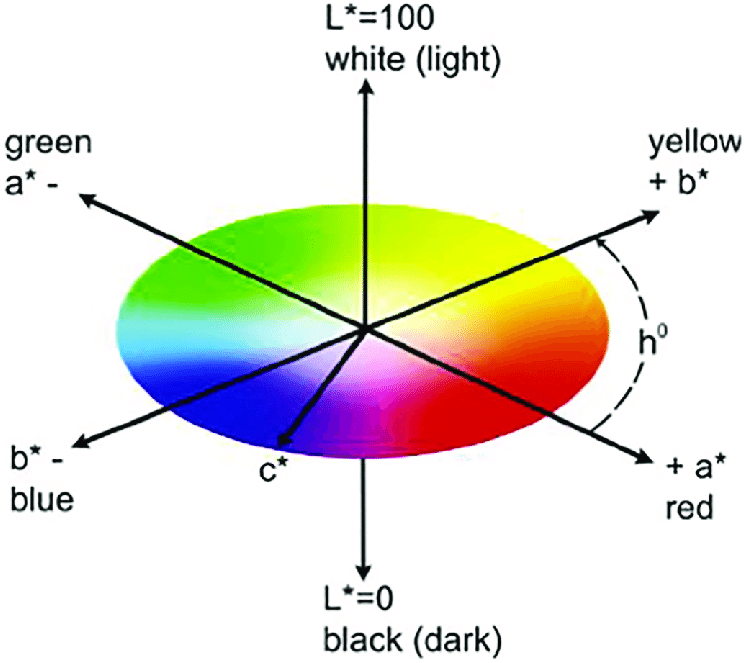


Figure 2 LAB Colour space. Reproduced from Ebeneezar et al 2020 (10.1016/j.aquaculture.2020.735728)

# Hardware usage notes

## General notes

* Please treat the OpenMV\_bot with care, covering the lens when not in use and making sure the SD card isn’t lost/damaged. Using a small box to transport the bot is a great way to avoid accidents.
* Turn the OpenMV off (see below) when not in use to conserve battery – you don’t want to run out of battery in the middle of an assignment!
* You should not have to take apart the bot at all throughout the course, but if things break, do let one of the GTAs know so we can troubleshoot and hand out spares.

## Power cycling

Please avoid unplugging and re-plugging the Lipo battery by pulling on the cable. Instead, the OpenMV\_bot can be turned off by holding the power button down for 5 seconds (Figure 2). It can be turned back on by holding it down for 1 second. ***This is an effective way of resetting your robot in the case of an infinite loop (which should be avoided regardless).***

A purple object with wires

Description automatically generated

Figure 3: OpenMV reset button

## Servo zero-point offset tuning (important)

Each of your wheels may have different zero-points, meaning even when the drive setting is set to 0, the wheel may move slowly in either the backwards or forwards direction. To fix this, you can adjust the zero-point offsets for each wheel in ***servos.py******lines 16-17*** incrementally (try +-0.05 steps) until both wheels stay still at drive setting of 0. Running ***servos.py*** as a script may help you tune these parameters.

Similarly, you should set ***servos.py******line 15*** to centre the camera pan servo when pan angle is set to zero.

## Wheel speed tuning instructions (optional)

You may notice that the wheel speed does not scale linearly between -1 and 1 drive setting in the control code. If you would like to have better control over the speed of your OpenMV\_bot, you may choose to perform wheel speed tuning experiments to model the drive setting-actual speed curve (e.g. sinusoidal, sigmoid). You will need a printout of a turning pad to perform these experiments.

Instructions:

1. Place the bot on the turning pad with one of the wheels placed in the centre and aligned with the 0° line. You will be testing the dynamics of the other wheel (the “test wheel”) (Figure 3, left).
2. Test how many degrees the bot rotates around the turning pad at several different drive settings for the test wheel (Figure 3, right). Note these values down. We suggest you sample the lower drive settings more densely. For example, you may test with drive settings of -1,-0.7, -0.4, -0.2, -0.15, -0.1, -0.05 for backwards drive.
3. Plot the drive settings on the x-axis and the angular distance values on the y-axis. What relationship do you see? Can you fit a function to it? Maybe you need to fit four separate functions for each of left wheel forwards/backwards and right wheel forwards/backwards.
4. Code the drive setting-actual speed curve onto your bot as a class method of *Servo* in ***servos.py***. You can modify the default drive methods to call on this new method.

A circular red circle with numbers and a circular object with a red circle

Description automatically generated with medium confidenceA circular diagram of a machine

Description automatically generated with medium confidence

Figure 4: Using a turning pad to tune the drive settings of the left wheel. (left) the bot is placed with the non-test wheel in line with the mark in the centre of the circle to begin. (right) the bot has ended up with the non-test wheel in line with the 140° mark after driving the test wheel forwards. Write this value down. Note that the bot may rotate more than 360° for higher drive settings.

# Common errors and fixes

## Trying to run a script gives me Err19!

Make sure the servo shield on the OpenMV has not become unconnected from the main board – be careful when unplugging/replugging the USB cable.