CSC 4120/6120 INTRODUCTION TO ROBOTICS

### MODULE 3

### All source codes for GoPiGo 3 robot are available in the GitHub page: <https://github.com/DexterInd/GoPiGo3>

Note: To change the Robot SSID/Name follow the link below:

<https://www.dexterindustries.com/howto/change-the-hostname-of-your-pi/>

### **Planned Path Robot Moving**

**Exercise 1**: **Learn the easygopigo3 library**

The GoPiGo3 robot has its own dedicated library, called easygopigo3. Familiarize with the library in this exercise. We can move the robot based on time, distance and even wheel rotation (say one wheel rotation, 360 degree)

From the Dashboard, go to **Learn > Lessons in Python**

Go through below exercise: **1\_Moving\_Around.ipynb**

Result:

Drive forward 10 cms followed by 10 inches.

Try below methods as suggested in the script by adding new cells along.

**my\_gopigo.forward()**

**my\_gopigo.backward()**

**my\_gopigo.stop()**

**my\_gopigo.right()**

**my\_gopigo.left()**

**my\_gopigo.turn\_degrees(90)**

**and**

**my\_gopigo.turn\_degress(-90)**

**Exercise 2**: **A. Motor Encoders**

Create a folder in the Home directory. Press on new folder icon. Right click on the newly created untitled folder and rename as “Module3”. Double click to enter the folder.

Click on the third icon (the upload icon) below the menu bar. Upload the file “Motor\_Encoder.py” present in the Module 3 -> Exercise 2 of the support files.

On the terminal, type:

#### cd Module3

#### python3 Motor\_Encoder.py

This code is an example for reading the GoPiGo3 Motors' encoders

Results: When you run this program, the GoPiGo3 Motors' position will be printed.

**Exercise 2**: **B. Motor Encoders**

Create a folder in the Home directory. Press on new folder icon. Right click on the newly created untitled folder and rename as “Module3”. Double click to enter the folder.

Click on the third icon (the upload icon) below the menu bar. Upload the file “Motor\_Follow.py” present in the Module 3 -> Exercise 2 of the support files.

On the terminal, type:

#### cd Module3

#### python3 Motor\_Follow.py

This code is an example for controlling the GoPiGo3 Motors

Results: When you run this program, the GoPiGo3 Motors will rotate back and forth and the GoPiGo3 Motors' position will be printed.

**Exercise 2**: **C. Motor Encoders**

Create a folder in the Home directory. Press on new folder icon. Right click on the newly created untitled folder and rename as “Module3”. Double click to enter the folder.

Click on the third icon (the upload icon) below the menu bar. Upload the file “Motor\_Position.py” present in the Module 3 -> Exercise 2 of the support files.

On the terminal, type:

#### cd Module3

#### python3 Motor\_Position.py

This code is an example for controlling the GoPiGo3 Motors

The robot on a loop, moves two wheel rotation forward followed by two wheel rotation backward.

Results: When you run this program, the GoPiGo3 Motors will rotate back and forth.

Go ahead and try printing the motors’ encoder position values as observe how it changes (the convention being + forward, - negative) during forward and backward wheel rotation. You can add the below line to print the same as seen in 2A.

#### print("Encoder L: %6d R: %6d" % (GPG.get\_motor\_encoder(GPG.MOTOR\_LEFT), GPG.get\_motor\_encoder(GPG.MOTOR\_RIGHT)))

**Exercise 2**: **D. Motor Encoders**

Create a folder in the Home directory. Press on new folder icon. Right click on the newly created untitled folder and rename as “Module3”. Double click to enter the folder.

Click on the third icon (the upload icon) below the menu bar. Upload the file “Motor\_Speed.py” present in the Module 3 -> Exercise 2 of the support files.

On the terminal, type:

#### cd Module3

#### python3 Motor\_Speed.py

This code is an example for controlling the GoPiGo3 Motors

Results: When you run this program, manually rotate the left motor to control the speed of the right motor accordingly.

**Exercise 2**: **E. Motor Encoders**

Create a folder in the Home directory. Press on new folder icon. Right click on the newly created untitled folder and rename as “Module3”. Double click to enter the folder.

Click on the third icon (the upload icon) below the menu bar. Upload the file “Motor\_Turn.py” present in the Module 3 -> Exercise 2 of the support files.

On the terminal, type:

#### cd Module3

#### python3 Motor\_Turn.py

This code is an example for making the GoPiGo3 turn accurately

Results: When you run this program, the GoPiGo3 should turn 90 degrees to the right, 180 to the left, and then 90 to the right, ending where it started.

Go ahead and try different degrees of wheel rotation and speed which can be passed as arguments as in the code snip below:

TurnDegrees(arg1, arg2)

# turn arg1 degrees to the right, at a wheel speed of arg2 degrees per second

# + arg1 is the convention for right turn ; - arg1 is the convention for left turn

**Exercise 3**: **Keyboard Robot Control**

Go through the below program to move the robot around using python methods.

Go to : **Home > GoPiGo3-Projects**

To run python scripts, open a terminal from the launcher:

File > New > Terminal

Once in the terminal, go to the directory to run corresponding script from.

On the terminal, type:

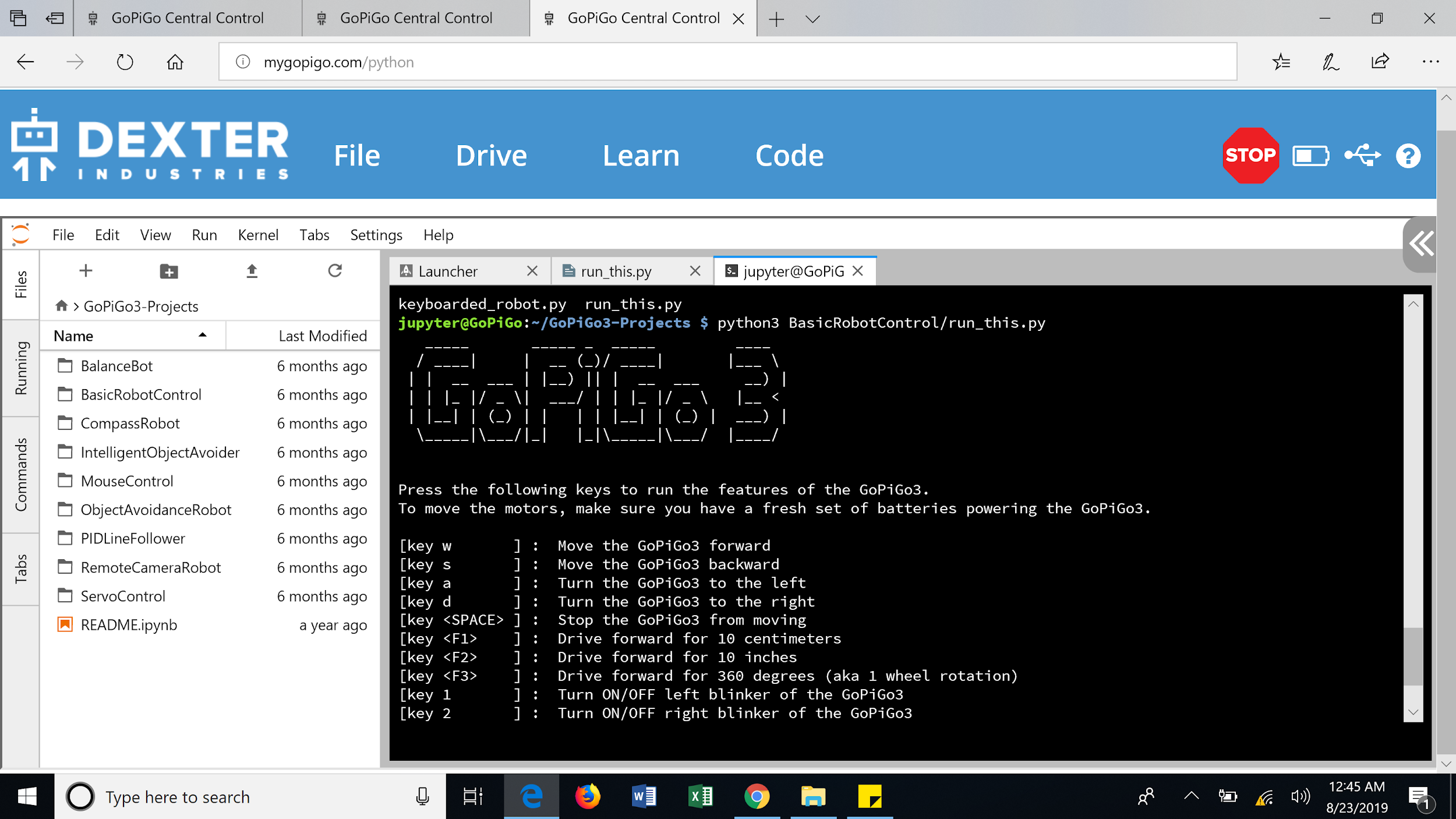
#### cd GoPiGo3-Projects/BasicRobotControl

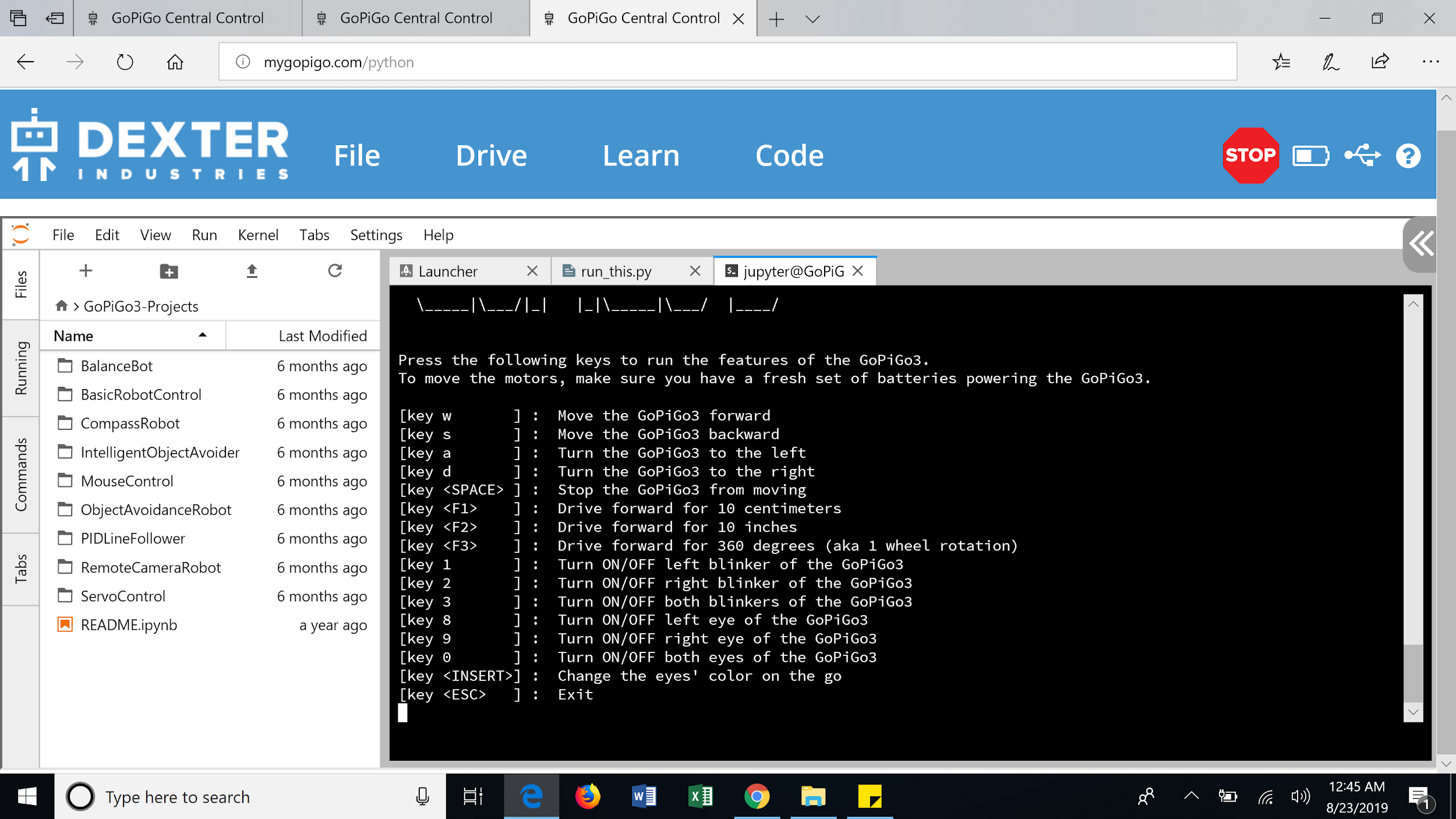
#### python3 run\_this.py

Result:

Should be able to control the robot’s movements through key press as directed through terminal console instructions. Should also be able to control blinkers and robot head eye.

Console looks like this once you run the program:





Refer to the videos in module 3 for robo in action performing various movement and blinking functions.

**Exercise 4**: **Mouse Robot Control**

Connect a wireless mouse via the Raspberry Pi’s USB port. You'll be able to control your GoPiGo3 robot with a wireless mouse either by mouse clicks or by mouse movements.

On the Terminal, type:

#### cd GoPiGo3-Projects/MouseControl

#### python3 mouse\_control\_robot.py

Result:

Choice 1 : Control your GoPiGo3 robot with a wireless mouse by mouse clicks.

Choice 2 : Control your GoPiGo3 robot with a wireless mouse by mouse movements.

**Assignment Problems for Demonstration**

**1. Autonomously move the robot along the perimeter of a square path of dimension 0.5 m x 0.5 m. If the robot senses an obstacle within 25cm on its way then it must stop, rotate by 180 degrees and trace the path along the other direction. Store the path trace into a file.**

**Expected output:** Create a folder named Module 3. The path trace should be saved into a file named as *problem1\_pathtrace.csv.* The path trace should include 3 columns in each row: *index or row number, wheel encoder value, distance value from distance sensor.* The path trace must be collected at **at-the-least** 2 samples/second.

**Bonus (2pts): The entire episode must be streamed using the remote camera option.**

**(Hint:** [**https://github.com/DexterInd/GoPiGo3/tree/master/Projects/RemoteCameraRobot**](https://github.com/DexterInd/GoPiGo3/tree/master/Projects/RemoteCameraRobot)**)**

-------------------------------------------------------------------------------------------------------------------------------

**2. Autonomously move the robot forward along the four cardinal directions. The path will be: turn towards North-->move 50cm--> turn towards East>move 50cm→ turn towards South→ move 50cm-->turn towards West-->move 50cm-->turn towards North and REPEAT. If it detects an obstacle within 25cm along the path it must circle around the obstacle and get back to its original forward path and continue moving. Store the path trace into a file.**

**Expected output:** Save the path trace as *problem2\_pathtrace.csv* in the Module 3 folder. The path trace should include 4 columns in each row: *index or row number, wheel encoder value, distance value from distance sensor, tracing path direction.* The path trace must be collected at **at-the-least** 2 samples/second.

Note: The tracing path direction can be noted as N, S, E, W, representing the four cardinal directions.

**Bonus (2pts): The entire episode must be captured as a video and the video must be downloaded onto the laptop and shown. It is OKAY to have a pre-run demo already saved, but I will also check for the video creation in real-time.**