

After setting up the raspberry pi and setting up the ssh, follow the below-mentioned steps to run code for multiple tasks:

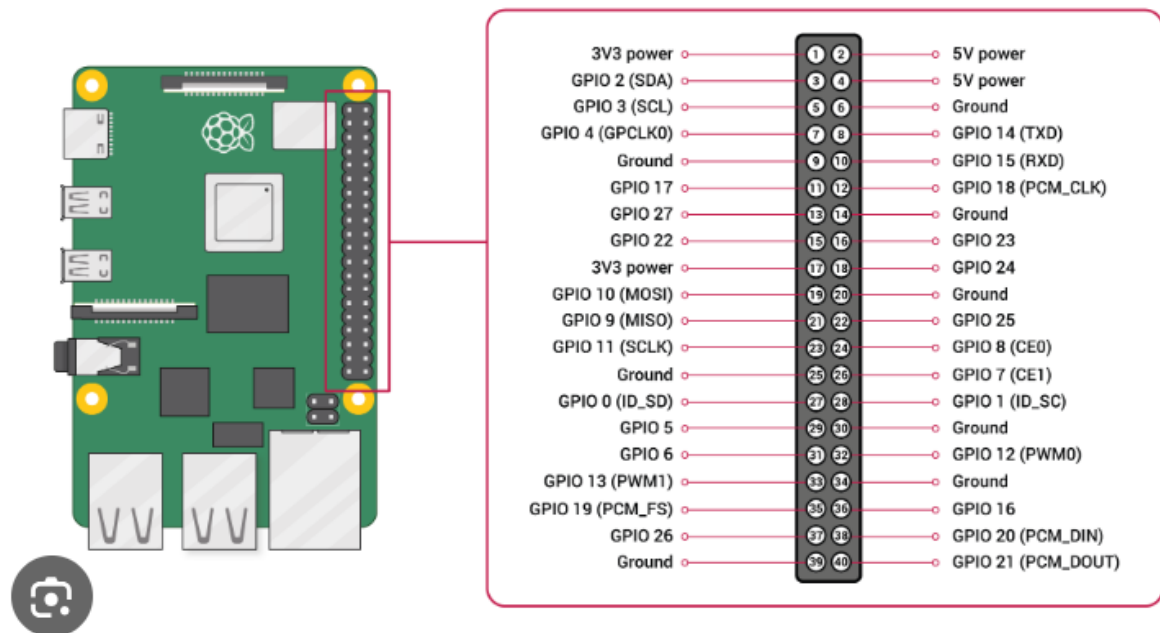
### 1. Motor with speed control:

#### ***File - motor\_with\_speed\_control.py***

All the motor and linear actuator-related tasks can be accomplished using this file.

Verify that all the Raspberry Pi GPIO pins are connected to their respective motors and linear actuator.

You can use the following diagram as a reference:



The GPIO pin numbers will depend upon what `GPIO.setmode(GPIO.BOARD)` or `GPIO.setmode(GPIO.BCM)`, as seen in the following image.

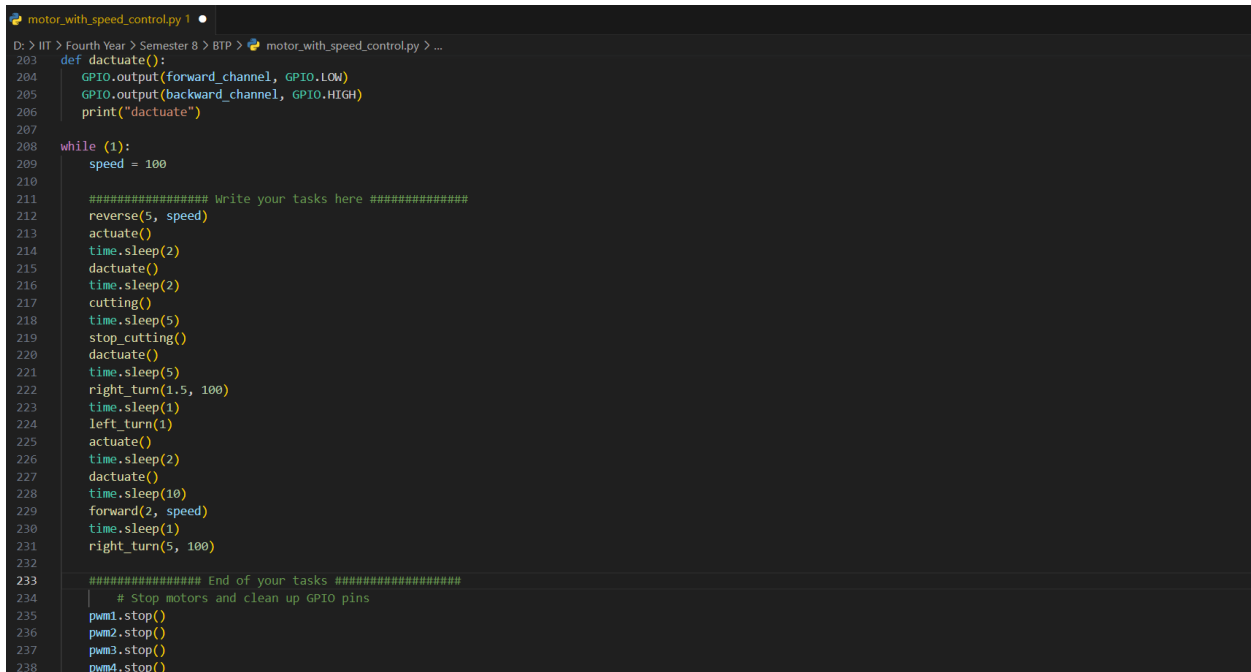
[In the image as well as by default in the code file: `GPIO.BOARD` mode is set]

```
motor_with_speed_control.py 1 • linear_actuator.py 1
D: > IIT > Fourth Year > Semester 8 > BTP > motor_with_speed_control.py > ...
1 import RPi.GPIO as GPIO
2 import time
3 #from camera_capture import *
4
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setmode(GPIO.BOARD)
8 sleeptime=1
9
```

Eight tasks can be accomplished using this Python file -

- a. Forward
- b. Reverse
- c. Left\_turn
- d. Right\_turn
- e. Cutting
- f. Stop\_cutting
- g. Actuate
- h. Dactuate

We can progressively mention the task we want to accomplish in the while loop of the file, as can be seen in the image below:



```
motor_with_speed_control.py 1
D:\> IIT > Fourth Year > Semester 8 > BTP > motor_with_speed_control.py > ...
203 def dactuate():
204     GPIO.output(forward_channel, GPIO.LOW)
205     GPIO.output(backward_channel, GPIO.HIGH)
206     print("dactuate")
207
208 while (1):
209     speed = 100
210
211     ##### Write your tasks here #####
212     reverse(5, speed)
213     actuate()
214     time.sleep(2)
215     dactuate()
216     time.sleep(2)
217     cutting()
218     time.sleep(5)
219     stop_cutting()
220     dactuate()
221     time.sleep(5)
222     right_turn(1.5, 100)
223     time.sleep(1)
224     left_turn(1)
225     actuate()
226     time.sleep(2)
227     dactuate()
228     time.sleep(10)
229     forward(2, speed)
230     time.sleep(1)
231     right_turn(5, 100)
232
233     ##### End of your tasks #####
234     # Stop motors and clean up GPIO pins
235     pwm1.stop()
236     pwm2.stop()
237     pwm3.stop()
238     pwm4.stop()
```

Note:

You have to mention the `time.sleep(x)` [x being the number of seconds] after every task, as seen in the image.

Speed can be mentioned from range (0, 100), though, for speeds < 15, the robot might not move due to its inability to overcome static friction.

## 2. For all the tasks:

**File: BTP\_main.py [GitHub], BTP folder, file name: main.py (or similar) [Raspberry Pi memory card]**

This file contains the entire code for the weeding operation apart from the path traversal. The code initially captures the webcam video with `cv2.VideoCapture(0)` command.

The object detection model identifies the weed and cotton plants based on their training and operates accordingly.

If the weed is found to be with  $> 0.5$  confidence, the robot will start its cutting operation. No code is written for the cotton plant to replan the robot's path based on its identification.

### 3. Computer Vision Model Scripts - Resnet:

**File: *CV\_model.ipynb* [Github] - very well-organized and easy-to-use Python notebook**

This file contains the entire code required for computer vision model training.

The file also contains code for testing the model on any video or webcam input based on the `cv2.VideoCapture(0)` command.

The data on which it is trained is attached with the google drive link below.

### 4. Object Detection Scripts - YOLOv7:

**File: *yolo\_training.ipynb* [Github]**

This file contains the entire code required for YOLO model training.

The code for model testing is also mentioned in the Python notebook.

### 5. Path Traversal Scripts - A-star:

**File: *Astar\_weed\_killing\_robot.ipynb***

All the heuristics and conditions have already been implemented for the path traversal.

[For more info about the heuristics, you can refer the BTP report file]

The only input required for the operation of the code is `start_node` and `end_node`.

Call the `run` function from the initialized Robot class, giving it all the arguments.

The image displays the entire path that the algorithm has designed.

```
import numpy as np
cotton_plants = np.where(np.array(grid) > 0)
cotton_plants = np.array(cotton_plants).transpose().tolist()

start = Node(0, 9)
goal = Node(10, 7)

# Define the robot
robot = Robot(grid, cotton_plants, start, goal)

# Run the robot
robot.run()

Start_Node: Node(0, 9)
Goal_Node: Node(10, 7)
[Node(0, 9), Node(1, 9), Node(1, 8), Node(1, 7), Node(2, 7), Node(3, 7), Node(4, 7), Node(5, 7), Node(6, 7), Node(7, 7), Node(8, 7), Node(9, 7), Node(10, 7)]
Task completed, all weeds have been cut!
```

The object detection and path traversal files haven't been integrated fully into the motor operation codes, which can be a task of future developers. However, it has been attempted in a few files found in the raspberry pi memory card but hasn't been tested.

**For any information, contact:**

Vansh Rai Saini: +91 96544 58060

Dhiraj Pimparkar: +91 98501 92860

Nishtha Gupta: +91 94525 01659

Shivank Kapila: +91 97977 11244