# piRover – Motor Controller Introduction Rev 1.4

## Overview:

In this activity you will get a first look at how to make the piRover move. You will investigate an online resource that describes the use of an L298N motor controller component. While this is not exactly what is found on the Yahboom controller board, you’ll get a look at the typical hardware required for driving a motor and the software configuration and control in this example is identical to what is required for your piRover.

You will review the code provided in the blog post and compare to our coding patterns used in the course so far. The sample code uses user defined functions. The instructor will introduce this software pattern and then work with the class to produce your first drive code.

## Prerequisites:

Prior to beginning the instruction provided in this lesson you must have completed the following:

1. Line Follower Input

## Performance Outcomes:

1. Identify motor controller concepts of “drive” and “control.”
2. See a discreet motor controller solution using the Raspberry Pi and compare it to the Yahboom controller board.
3. Identify GPIO ports used to control piRover motors.
4. Create and use functions in your code.
5. Create basic motion with the piRover.

## Resources:

1. [Motor Control - L298N Dual H-Bridge](http://www.piddlerintheroot.com/l298n-dual-h-bridge/)

## Materials:

1. piRover

## Part 1 – Set Up

1. Prepare your workspace for this activity. Connect to your piRover using VNC and open VS Code
2. Create a directory for this week’s work if this was not in the prior session
3. Using the VS Code terminal window, change to this week’s directory.
4. Download the starter files for the activity using links provided on this week’s page.

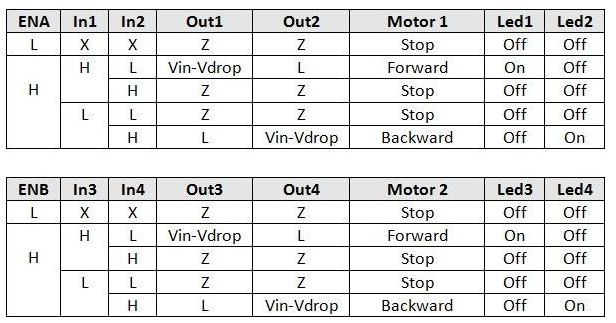
## Part 1 – Investigate the hardware

1. Use the [Motor Control - L298N Dual H-Bridge](http://www.piddlerintheroot.com/l298n-dual-h-bridge/) blog post to get an introduction to the circuity required for motor control. The instructor will review the concepts of “drive” and “control.” In this example, what voltage level or source is being used to “drive” the motors? What voltage level is being used to “control” the motors? Why is a ground connection required between the Raspberry Pi and the motor controller component?
2. Use the [Yahboom Expansion Board Manual](https://1drv.ms/b/s!AuNrQVMLdN3-mblEVZDg7JQaYSEJgw?e=5EdQJs) to identify the GPIO pins allocated for motor control output. Table 1 entries are missing below. Determine the values using the documentation.

|  |  |  |
| --- | --- | --- |
| **Input** | **Board Pin** | **GPIO Reference** |
| PWMA |  |  |
| PWMB |  |  |
| AIN1 |  |  |
| AIN2 |  |  |
| BIN1 |  |  |
| BIN2 |  |  |

Table

1. The following function table comes from the L298N data sheet. The function is not obvious for a beginner. The instructor will discuss. At this point you should understand the meaning of H and L (High and Low), but what about X and Z? Your instructor will define.

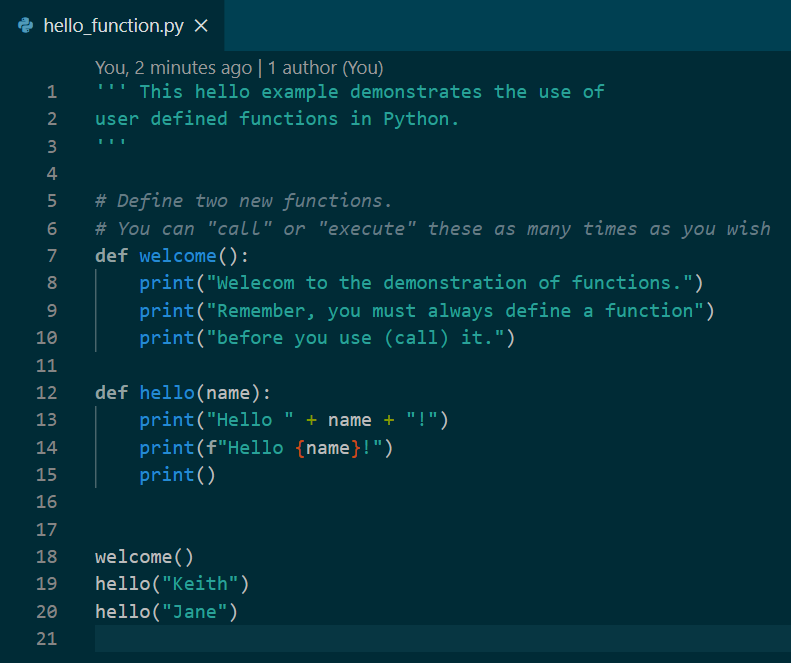


1. Based on the function table above and the instructor’s review, complete the simplified function table for a single channel below indicating if the motor is stopped, turning forward, or turning backward.

|  |  |  |  |
| --- | --- | --- | --- |
| **EN** | **IN1** | **IN2** | **Motor Action** |
| 0 | 0 | 0 |  |
| 0 | 0 | 1 |  |
| 0 | 1 | 0 |  |
| 0 | 1 | 1 |  |
| 1 | 0 | 0 |  |
| 1 | 0 | 1 |  |
| 1 | 1 | 0 |  |
| 1 | 1 | 1 |  |

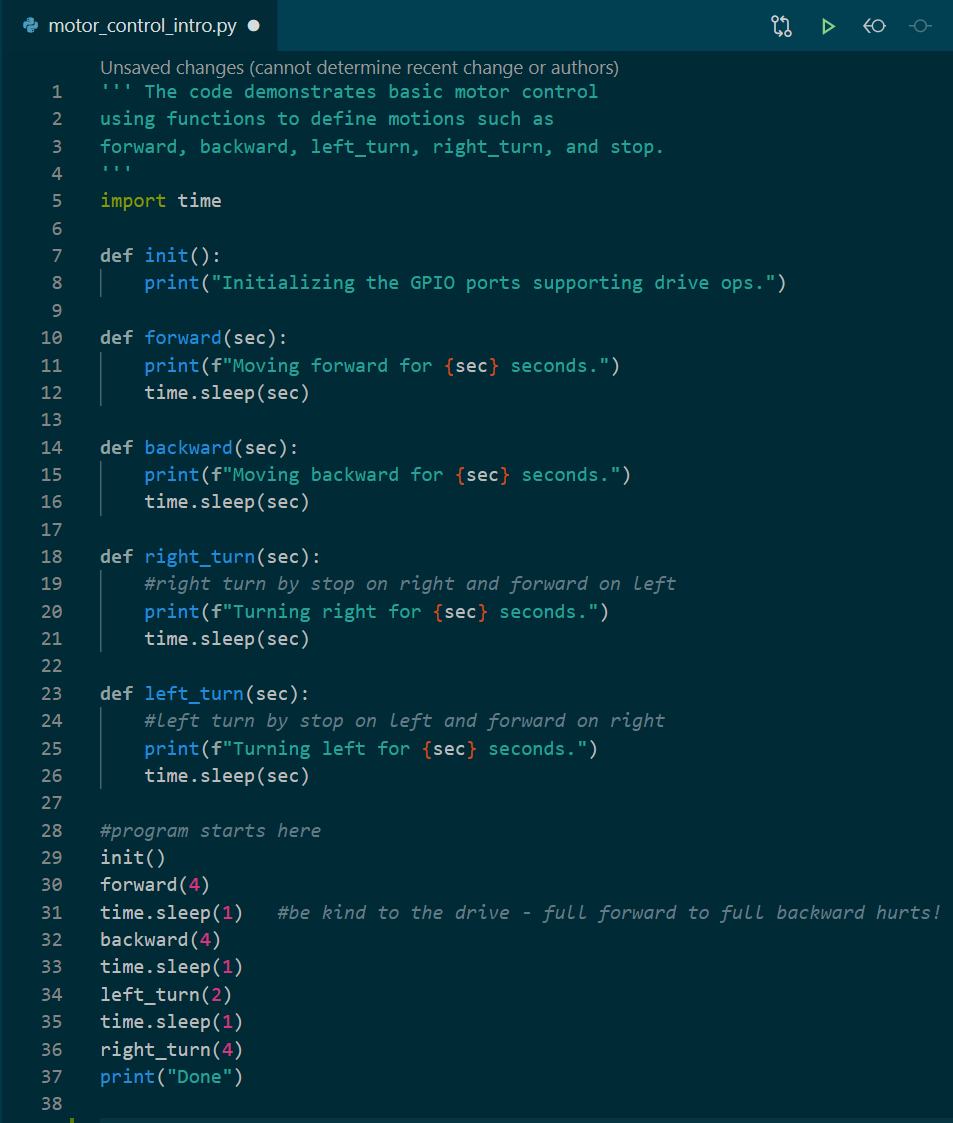
1. Note that “forward” and “backward” are relative. What if the motor is turning clockwise and is positioned on the right side of the rover? Is this forward or backward? Now consider the same motor turn clockwise on the left side of the rover. Is this forward or backward motion? You will need to experiment with your piRover to get direction control working correctly.

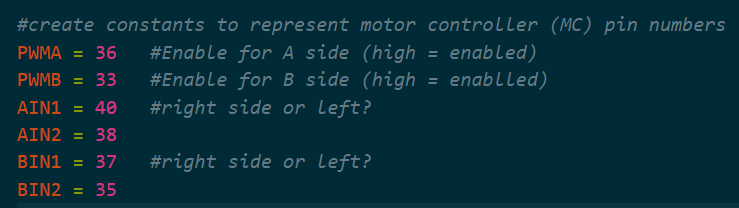
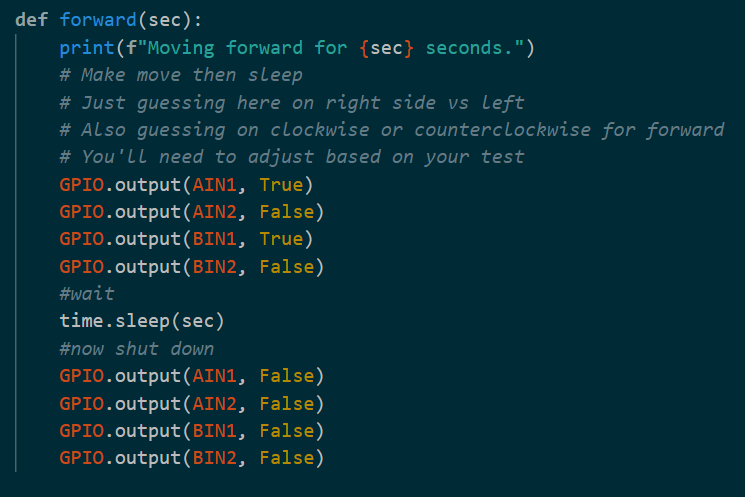
## Part 2 – Investigate the user defined functions

1. The instructor will introduce the concept of functions. Create the HelloFunction.py below. Use the debugger as a tool to understand how Python is processing and executing this code. Be sure to use both Step Over (F10) and Step Into (F11) in the debug session.

## Part 3 – Investigate the software – motor\_control\_intro.py

1. **NOTE:** You will be testing the piRover drive function. Be sure to have your piRover suspended using a prop such as the original kit box.
2. Open the new motor\_control\_intro.py file copied to the activity directory during the setup. The current file does not include any GPIO code. Use the debugger to run the solution. Be sure to use the F11 – Step Into operation and note how the sec delay values are used.



1. Review table 1 and create constants to represent the six pins required for motor control. Note the comments included below. Is A or B the right or left side?
2. Create the GPIO initialization function as shown below. Do you recognize this code? Does it make sense to have this segment defined in a function called init()?
3. Let’s take a guess at some code to make it move forward. Is the A side the right or the left? Do you set AIN1 high and AIN2 low for forward? Run the code with only the forward function defined to test. BE SURE TO HAVE THE ROVER TRACKS SUSPENDED USING A PROP SUCH AS THE ORIGINAL KIT BOX.

1. Use the debugger to single step the code. Determine which controller (A or B) controls the left track and which controls the right track.
2. Adjust IN1 and IN2 as required to get forward motion.
3. Complete the following function table to document your findings on FORWARD motion.

|  |  |  |
| --- | --- | --- |
| **Output** | **Location (left or right)** | **Control Signal for FORWARD motion** |
| PWMA |  |  |
| AIN1 |  |  |
| AIN2 |  |  |
| PWMB |  |  |
| BIN1 |  |  |
| BIN2 |  |  |

1. Copy the code created for forward motion into the backward() function. Adjust the code as required to create backward or reverse motion. Test your code
2. Copy motion code into left and right turn functions. Note the comment on how to create turns. There are other options that you will investigate later. Adjust the code as required and test.
3. Add a few additional movements at the end of the file, such as a series of left and right turns. See the comment on the dangers of going from full forward to full backward motion. There is no speed control at this point. You will create speed control in future activities.

## Assessment:

Submit this file with table entries completed and your final motor\_control\_intro.py solution by zipping along with other weekly requirements. Submit to the weekly assignment link provided in this week’s Moodle section.