LESSON D-11

ACTIVITY #2:
ADD PWM DRIVE CONTROL TO MOTORS

In this activity you will modify the program from the last activity to include PWM control of the enable lines on the motor drive IC. This will enable independent speed control of the left and right motors. This program modification will be quick and can be done on battery power.

Step #1

If not already running from the last activity, power up the robot on battery power and connect to the onboard VNC server using another device on your network.

Once at the Desktop, open the folder named **`robot`** and double-click on the file named **`motor_fwd_rev.py`** to open the file in Thonny.

Save a new copy of this program in Thonny by selecting **`File`**, and then **`Save As`** from the upper-left menu. If needed, double-click on the **`robot`** folder in the displayed list of files to save your file in the same location as the original, which is inside the folder named **`/home/pi/Desktop/robot`**. Type **`motor_pwm.py`** in for the file name and click on the **`OK`** button in the lower-right of the Thonny window.

You will now be working in a new copy of the file so the original can be used again later, if needed.

Step #2

The first modification that will be made to this program will be to add variables that contain PWM values that will be used for the right and left motors. We will use the variable names **r_speed** and **l_speed** to store values of **99** for both motors as a starting point. Add these variables and their values to the list of variables at the beginning of the program:

```
motors = [13,19,26,12,23,24]
r_enable = 12
r_for = 24
r_rev = 23
r_speed = 99
l_enable = 13
l_for = 19
l_rev = 26
l_speed = 99
```

Step #3

The next modification will be to configure the PWM signals that will be used to drive the **l_enable** and **r_enable** pins. We will use the names **l_pwm** and **r_pwm** for the left and right PWM signals and we use 1000Hz for the frequency. Add the highlighted code below to the program just before the **drive_motor** function:

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(motors, GPIO.OUT)
GPIO.output(motors, GPIO.LOW)

l_pwm = GPIO.PWM(l_enable, 1000)
l_pwm.start(l_speed)
r_pwm = GPIO.PWM(r_enable, 1000)
r_pwm.start(r_speed)

def drive_motor(pin1, pin2, state):
    GPIO.output(pin1, state)
    GPIO.output(pin2, state)
```

This will configure and start the PWM signals on the left and right enable pins using the **l_speed** and **r_speed** duty cycle values that were specified in earlier in the program.

Step #4

The goal of this test is to see how straight the robot is driving, which may be a little difficult if the robot only drives for one second. Change the **time.sleep** value in the **drive_motor** function from **1** to **2** so you will be able to get a better view of how straight the robot is driving:

```
def drive_motor(pin1, pin2, state):
    GPIO.output(pin1, state)
    GPIO.output(pin2, state)
    time.sleep(2)
```

Step #5

The program is now ready to drive the motors using PWM speed control values of 99 for both the left and right motors. In a perfect world, these equal values would cause the robot to drive perfectly straight, but as you learned in this lesson, many factors are working against these being the best values.

Ensure the area around the robot is clear of any obstacles and safe for driving. Keep in mind that the robot will be driving twice as far as the first test since we doubled the **`time.sleep`** value in the **`drive_motor`** function.

Run the program and monitor the robot's accuracy when driving forward and then in reverse. Here are the ways to handle adjustments as needed:

- The robot is driving straight – No adjustment is needed, proceed to the next step.
- The robot is turning slightly to the left – This means the right motor is driving too quickly. Substitute a slightly lower number for the value of **`r_speed`** and run the test again. Keep adjusting this value until the robot is driving straight.
- The robot is turning slightly to the right – This means the left motor is driving too quickly. Substitute a slightly lower number for the value of **`l_speed`** and run the test again. Keep adjusting this value until the robot is driving straight.

Step #6

Now that the robot is driving straight using your PWM values, make note of these values as they will be useful for future programs that you will create that use PWM speed control of the left and right motors on your robot.

Since the PWM speed control test and calibration is now complete, power off the Raspberry Pi. Once the SD card activity LED has stopped flashing, disconnect the battery pack from the Voltage Regulator Module to fully remove power from the robot.