piRover Builds with K2

piRover – Blink with VS Code   Rev 1.0

Overview:

In this activity you learn to access and control the input/output (I/O) pins on the Pi using Python and the GPIO library. You start by first reviewing the Raspberry Pi GPIO header. The Raspberry Pi was introduced in an earlier lesson, but here you take a much closer look at the interfacing that is possible with its GPIO ports.

In "Blink", you work with the GPIO port as an output that controls an LED light. Using the Yahboom documentation, you need to determine the connections from the Raspberry Pi GPIO header to the RGB LED module. Once the GPIO pins for each LED color are known, you configure that pin as an output and control the LED by turning it on or off.

Prerequisites:

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

1. Visual Studio Code – Getting Started

Performance Outcomes:

1. Recognize a light emitting diode – LED
2. Interpret documentation to determine electrical connections
3. Write code to control state using a timer
4. Configure a GPIO port on the Raspberry Pi as an output.
5. Write logic levels to a GPIO port configured as an output.

Resources:

1. Light-Emitting Diodes (LEDs)
2. GPIO Outputs
3. Yahboom Expansion Board Manual

Materials:

1. piRover

Part 1 – Investigate the hardware

1. This is the first activity where your code (software) will control electronics (hardware). Start by reviewing the overview of Light-Emitting Diodes (LEDs) These devices will light when a current flows through the device. To create a current, you will need to produce a voltage drop across the device. You will learn more about this device in the second course, but for now you know that you need to control the voltage applied to the device if you want to turn it on and off.

2. Next, review the GPIO connector on the Raspberry Pi. The image below shows the Raspberry Pi 3 B+. You are using Raspberry Pi 4, but the GPIO port pin out is the same. The gray text under each pin description shows some special function of the pin. For now, you are just focused on the basics of using GPIO.

3. First, identify the 5-volt power connections on pins 2 and 4. These are not typically used because the Raspberry Pi is a 3.3-volt device, meaning its GPIO circuitry works between 0-volts and 3.3-volts.

4. Next, identify the 3.3-volt power connections on pins 1 and 17. These pins can be used to supply 3.3V power to external circuits.

5. Now locate the ground pins. These pins provided the ground or 0-volt reference for external circuits. Having a "good ground" is important in electronic circuits so there are multiple pins available – 6, 9, 14, 20, 25, 30, 34, 39. These are all electrically the same point, so it does not matter which pin that you use when creating an external circuit.

GPIO Pinout Diagram
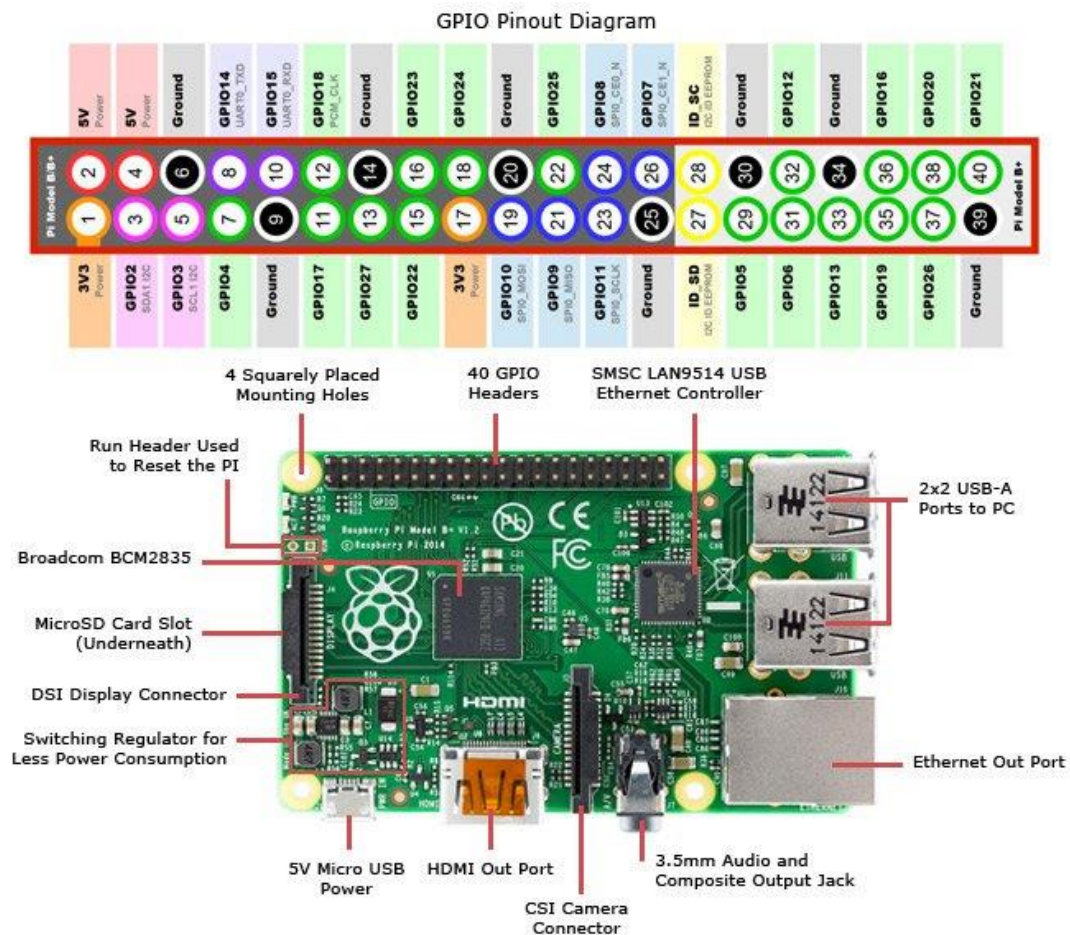
piRover Builds with K2

2. The remainder of the pins GPIO signal pins and available for both input (sensing) and output (control). Again, many have a special purpose as well, but for this course you will be working with only the basic operation.

3. In this initial GPIO activity, you will work with the GPIO port as an output. This means that in software you will be able to "write" a high or a low logic level to the pin. For almost all systems a low logic level is 0 volts. The Raspberry Pi is a 3.3-volt device, so a high logic level is 3.3-volts. By writing highs and lows (3.3V and 0V) to a pin connected to the LED, you can turn the light on and off.

4. Now review the documentation in the Yahboom Expansion Board Manual. On page 10 the detail for the RGB LED Module is provided. You are only interested in the Raspberry Pi connections so ignore details on the Arduino, 51 MCU, and STM32 interfaces. The Raspberry Pi detail in image 5-2 is duplicated below.
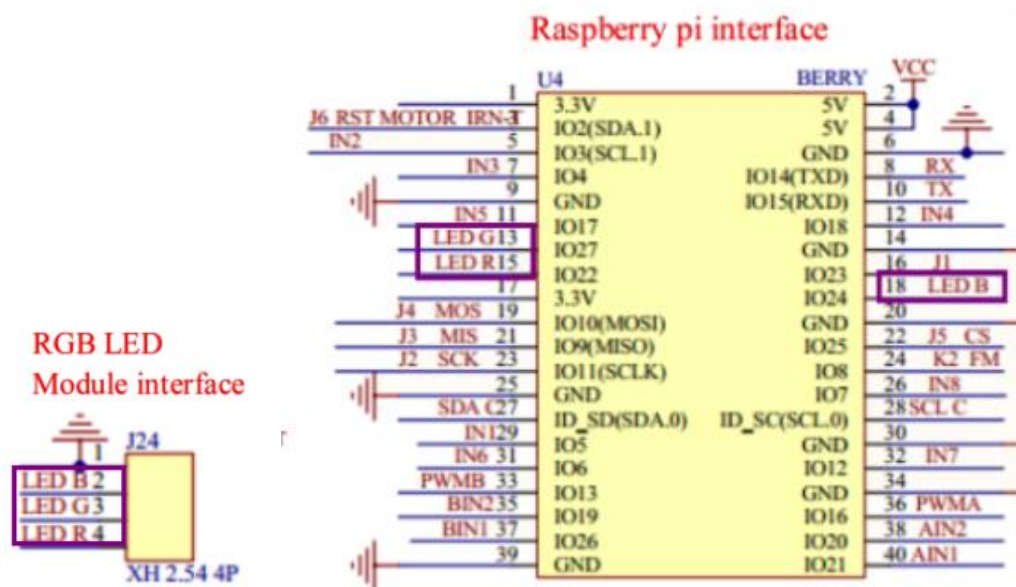
5. J24 on the left represents the connector on the expansion board that you plugged the RGB LED module into. The 40-pin Raspberry Pi connector is represented on the right. You will normally reference documentation such as this to determine the required "pinout". In this first activity, this has been done for you and is recorded in the Table 1. Review Table 1 to be sure that you understand how this connection detail is determined. You will be expected to locate information on pinouts on your own in future activities.

piRover Builds with K2

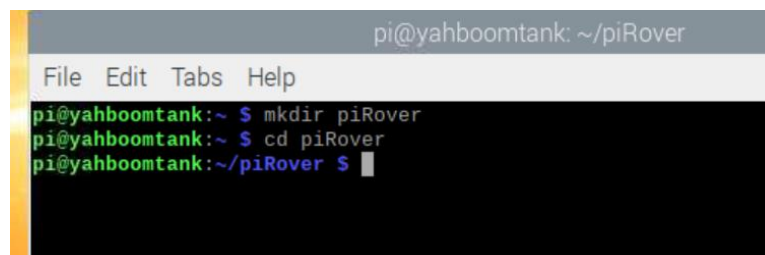| LED Color	Physical Pin	GPIO Reference
Red	15	GPIO22
Green	13	GPIO27
Blue	18	GPIO24

Table 1

6. Your goal in this activity is to blink the RGB module first with red light and then with white. In this class you reference the connections using the physical pins on the connector, which means that you will first be controlling pin 15 and then eventually all three – 13, 15, and 18.

Part 2 – Investigate the software

7. With an understanding of the hardware and connections, it is time to transition to the software. Here, you download code that represents the basic structure of blinking an LED. You will then revise the Python code to include the specific GPIO code required to "drive" the LED, turning it on and off.

8. Connect to your piRover and open a terminal window.

9. Create a directory in your home directory named piRover. The command is provided below. This will be the location for all your Python solutions supporting the piRover. Move to this directory using the cd command.

   mkdir piRover

   cd piRover

10. Create a directory in your piRover directory for this blink exercise named **01.Blink**. Move to this directory. The commands are provided below.

   mkdir 01.Blink

   cd 01.Blink

piRover Builds with K2

11. Download the starter code for the exercise by entering the following command. Use copy and paste (Note: you must have an Internet connection established on the piRover. See the prior "Configuring" activity.)

curl https://raw.githubusercontent.com/k2controls/piRover/master/01.Blink/blink.py > blink.py

12. Use the **ls** command to verify that the blink.py file was copied to your folder.

13. Launch Visual Studio Code by entering the following command at the prompt. The dot at the end tells VS Code to open with this folder loaded in the Explorer view.
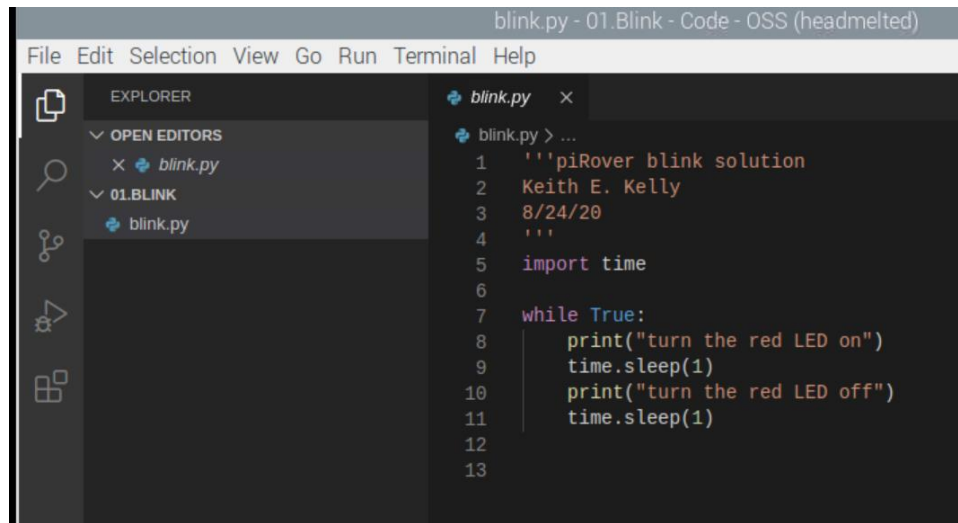
code-oss .

14. The Visual Studio Code editor is now visible. Close the Welcome tab and click on Blink.py to open in the editor.

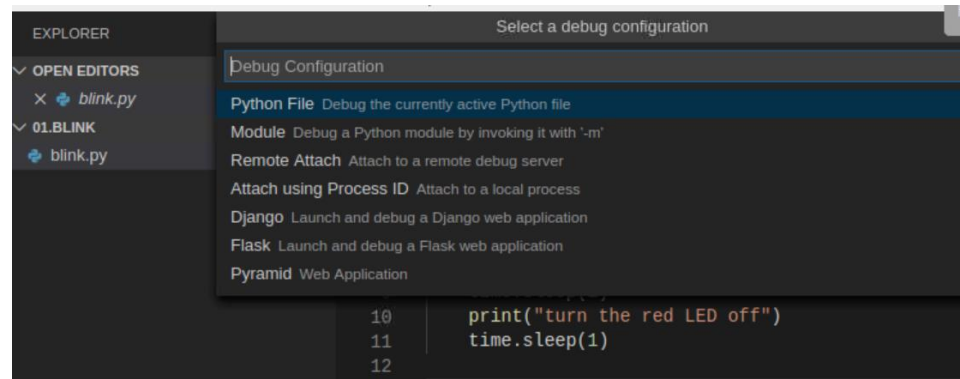# piRover Builds with K2

15. Edit lines 2 and 3 to include your name and the current date.

16. Use F5 to run the program. The debugger needs additional information about what to run. Select Python File from the "Select a debug configuration" prompt shown below.

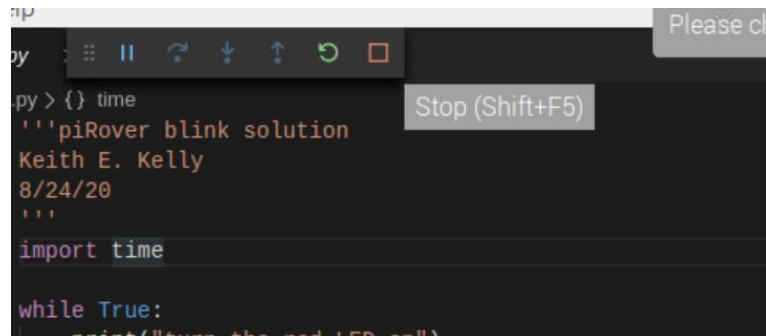17. The program runs in the terminal window at the bottom of VS Code.

18. Click the Stop button (red square) in the debug toolbar to stop execution.

piRover Builds with K2

19. Review each line of code. The "while True:" statement makes the block of code that follow loop or repeat forever.

20. Run the code again, but this time access the "Start Debugging" option from the Run menu.
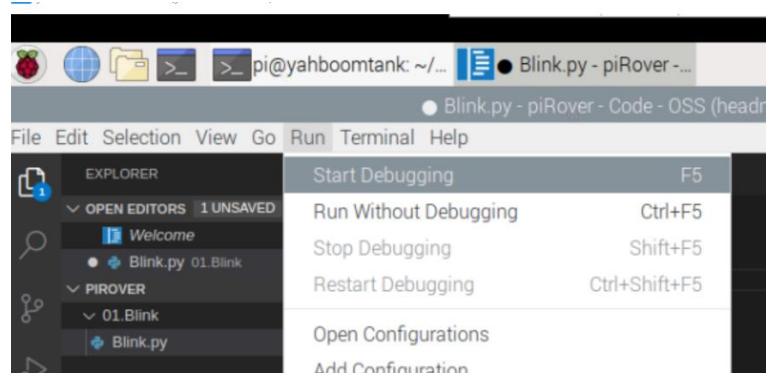
Part 3 – Control GPIO Outputs

21. Add the code required to control the GPIO port connected to the Red LED shown below.

    a. Lines 9 and 10 will always be included. Unlike Yahboom you will always use pin numbers, so line 11 uses GPIO.BOARD rather than the GPIO.BCM seen in Yahboom code.

    b. Line 13 is required configuration code. You are specifying that pin 15 (the pin connected to the Red LED) is an output, meaning you can "write" voltage levels to this pin.

    c. Lines 18 and 21 set the voltage on the pin either high (True) or low (False).

7

© 2020, Keith E. Kelly. All rights reserved.

piRover Builds with K2

```
home > pi > Desktop > 🐍 blink_solution.py > ...
  1    '''piRover blink solution
  2    Keith E. Kelly
  3    8/24/20
  4    '''
  5    import time
  6    import RPi.GPIO as GPIO
  7
  8    # Ignore warnings
  9    GPIO.setwarnings(False)
 10    # Use board pin numbers
 11    GPIO.setmode(GPIO.BOARD)
 12    # Set pin 15 as output
 13    GPIO.setup(15, GPIO.OUT)
 14
 15
 16    while True:
 17        print("turn the red LED on")
 18        GPIO.output(15, True)
 19        time.sleep(1)
 20        print("turn the red LED off")
 21        GPIO.output(15, False)
 22        time.sleep(1)
 23
 24
```

22. Run the code and test using the F5 key. If the LED blinks only for a moment, you still have the Yahboom Bluetooth software running in the background. Refer to the "Disabling Yahboom Bluetooth" document to resolve.

23. On your own, modify the code to use the Green and Blue LEDs as well. Change or add output statements in the Blink.py loop so that the LEDs cycle from Red to White (white is created by turning red, green, and blue on).

24. Run your revised code and test.

25. When you are satisfied with your solution, upload the 01.Blink folder with the blink.py file into a piRover folder on your favorite cloud storage service. Below shows Microsoft OneDrive being used. The

piRover Builds with K2

01.Blink folder can be dragged from the Pi's Explorer window to the piRover folder shown in the browser. Google drive or DropBox will work as well.
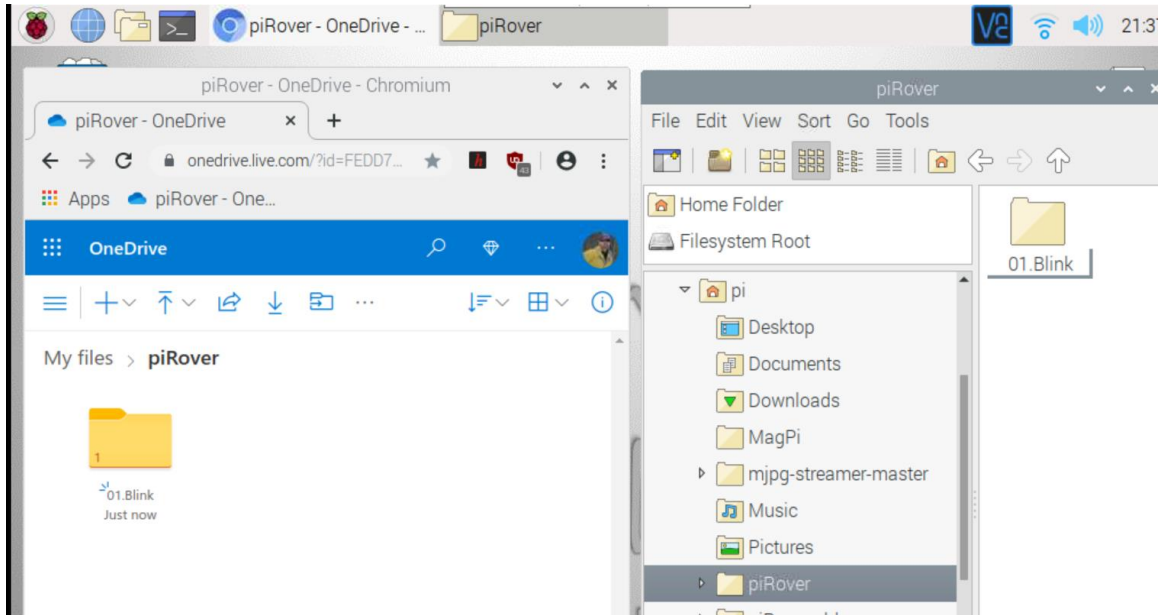
Assessment:

Submit your final **Blink.py** file to Moodle along with other files in this week's zip file.