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 – 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

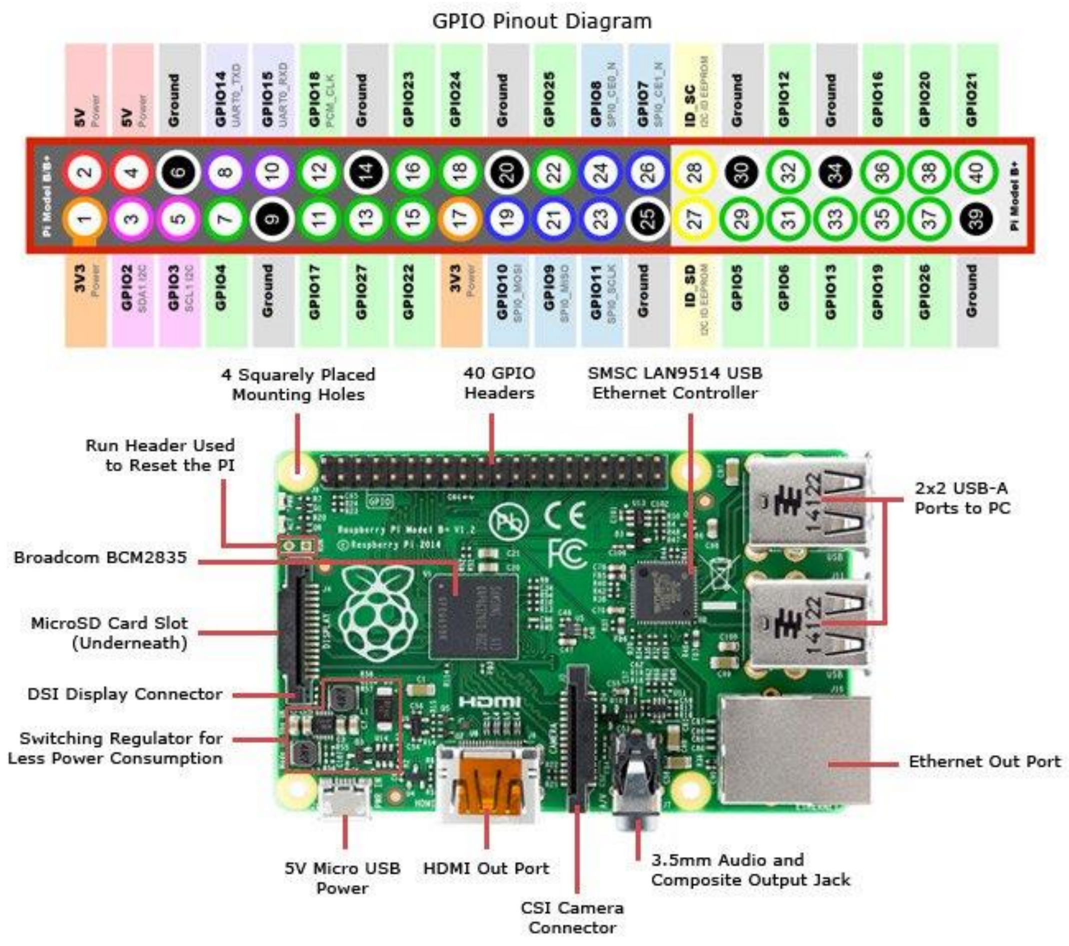
Materials:

1. piRover

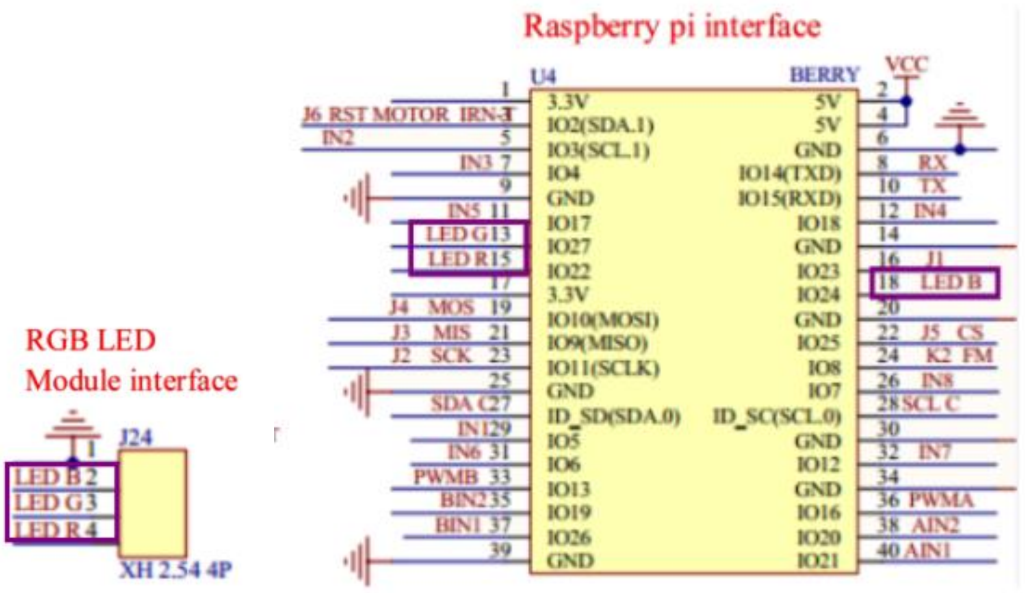
Part 1 – Investigate the hardware

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.



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.

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

Table 1

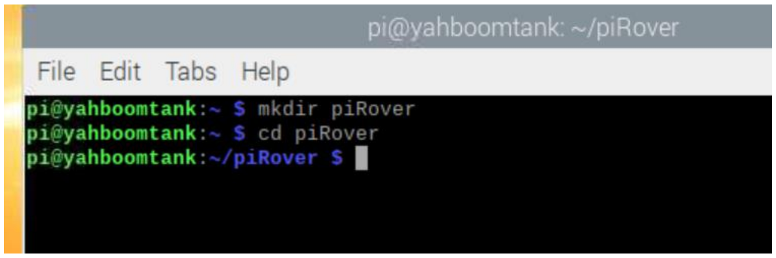
6. Your goal in this activity is to blink the RGB module first with red light and then with white. In this class you reference the connections using the physical pins on the connector, which means that you will first be controlling pin 15 and then eventually all three – 13, 15, and 18.

Part 2 – Investigate the software

7. With an understanding of the hardware and connections, it is time to transition to the software. Here, you will first create the control logic and then make the connections to the hardware in Part 3.
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 of our Python code that supports the piRover. Move to this directory using the cd command.

mkdir piRover

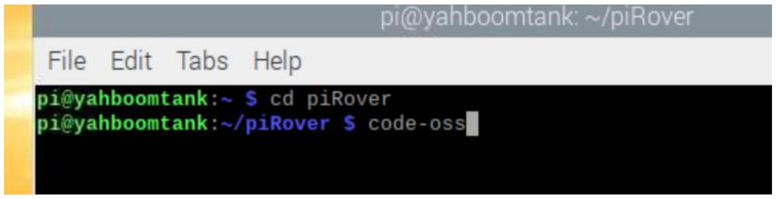
cd piRover



```
pi@yahboomtank:~ $ mkdir piRover
pi@yahboomtank:~ $ cd piRover
pi@yahboomtank:~/piRover $ 
```

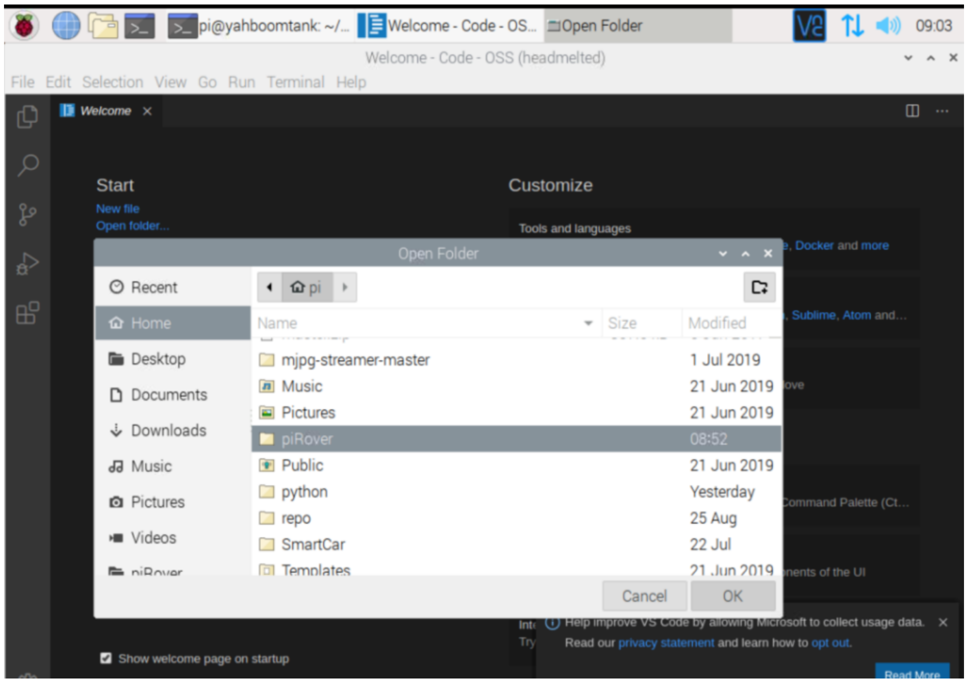
10. Launch Visual Studio Code by entering the following command at the prompt.

code-oss



```
pi@yahboomtank:~ $ cd piRover
pi@yahboomtank:~/piRover $ code-oss
```

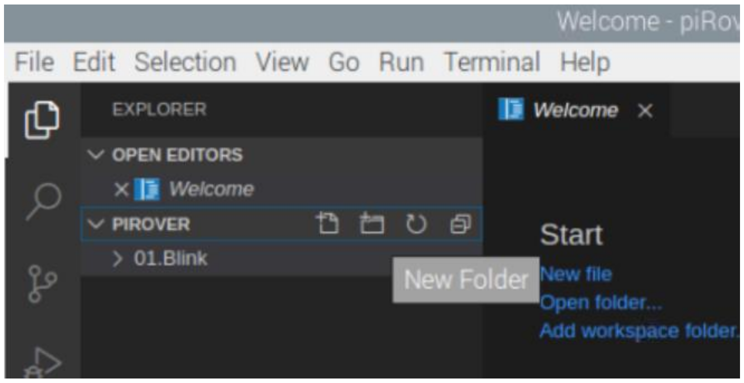
11. The Visual Studio Code editor is now visible. Use the Open Folder option from the File menu or the Welcome window to open the piRover directory that you just created.



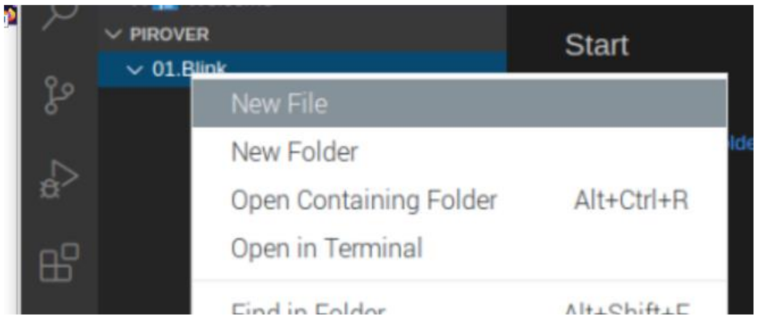
Name	Size	Modified
mjpg-streamer-master		1 Jul 2019
Music		21 Jun 2019
Pictures		21 Jun 2019
piRover		08:52
Public		21 Jun 2019
python		Yesterday
repo		25 Aug
SmartCar		22 Jul
Templates		21 Jun 2019

Help improve VS Code by allowing Microsoft to collect usage data.
Read our privacy statement and learn how to opt out.

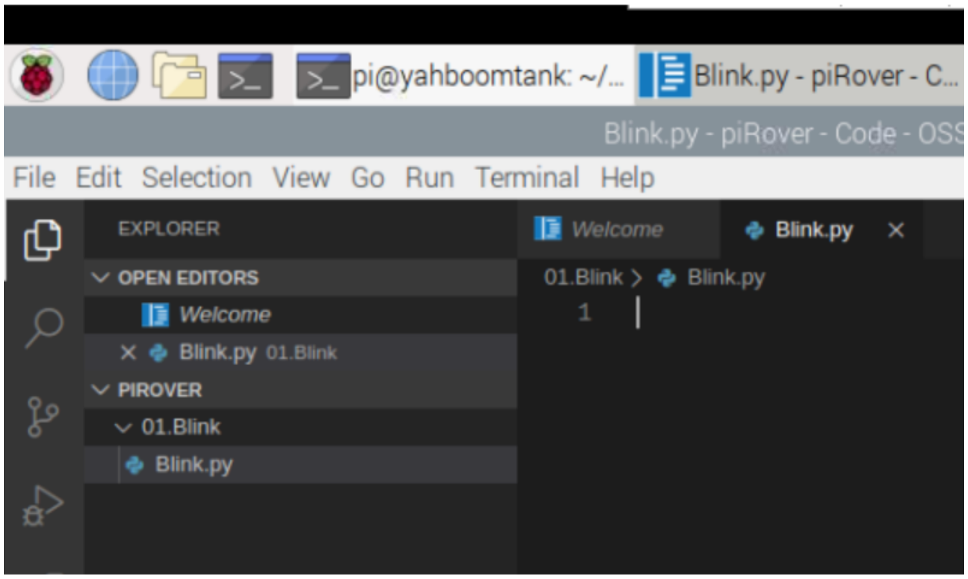
12. Use the New Folder tool in the Explore pane to create a new folder under the piRover folder named 01.Blink



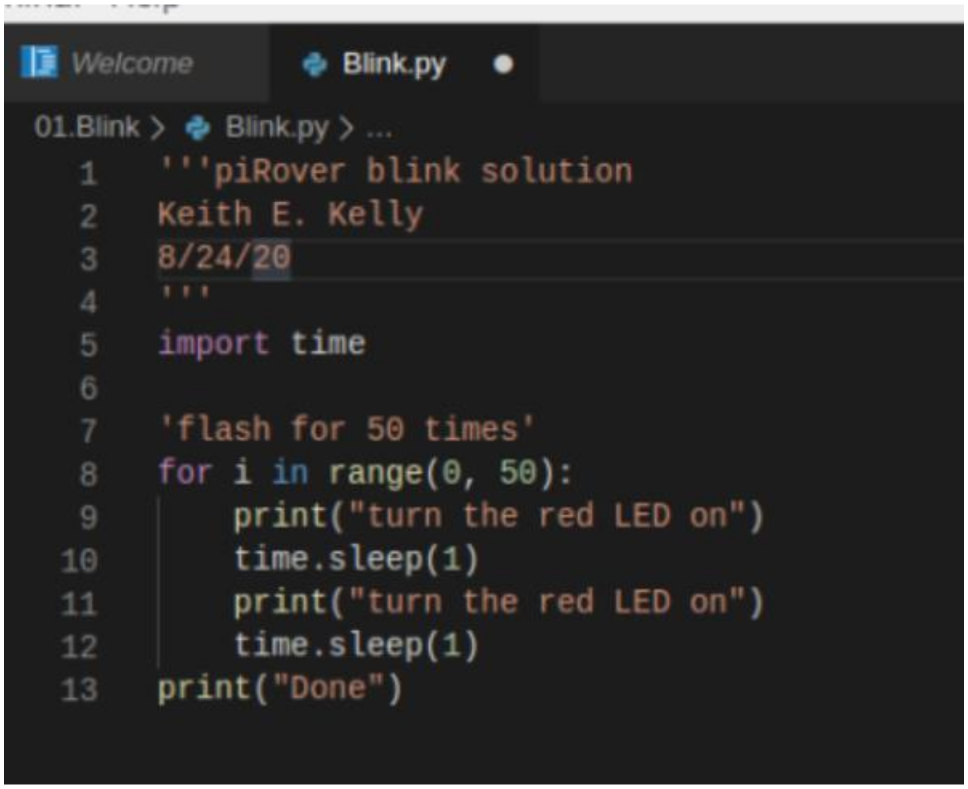
13. Right-click on the 01.Blink folder and select New File. Create a new file named Bink.py



14. The blink.py file is open in the editor. Note the icon on the file. VS Code recognizes this file with a .py extension as a Python code file.

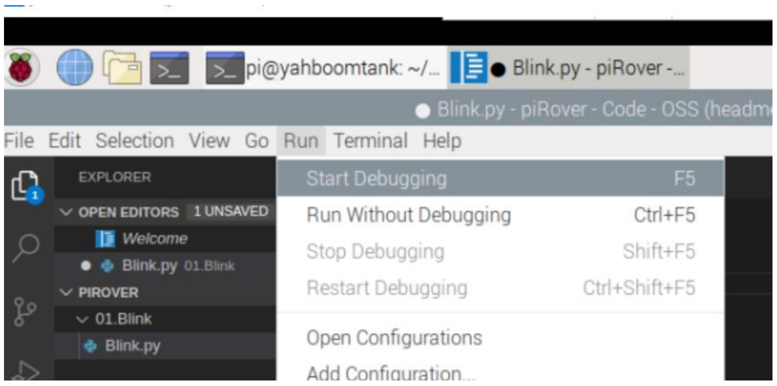


15. Focus just on the logic of the code to start. Create a loop that executes 50 times and prints "LED on" and "LED off" every second. you'll add the GPIO code after you get the basic structure of the file created and have had a chance to run the code. Enter the code below. Be sure to enter your name and the date rather than the instructor's in the example.

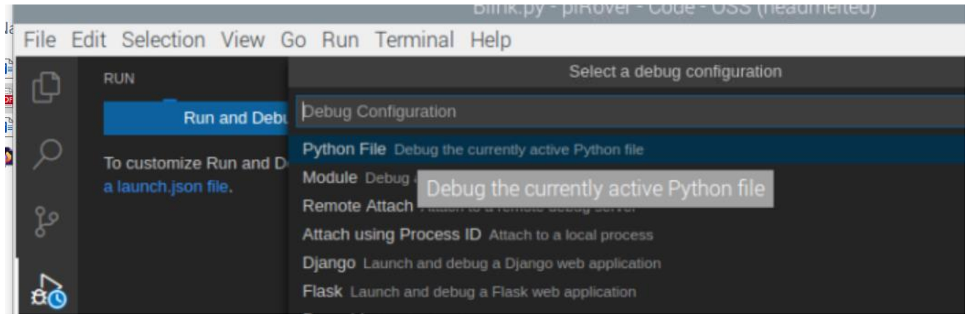


```
'''piRover blink solution
Keith E. Kelly
8/24/20
'''
import time

'flash for 50 times'
for i in range(0, 50):
    print("turn the red LED on")
    time.sleep(1)
    print("turn the red LED on")
    time.sleep(1)
print("Done")
```

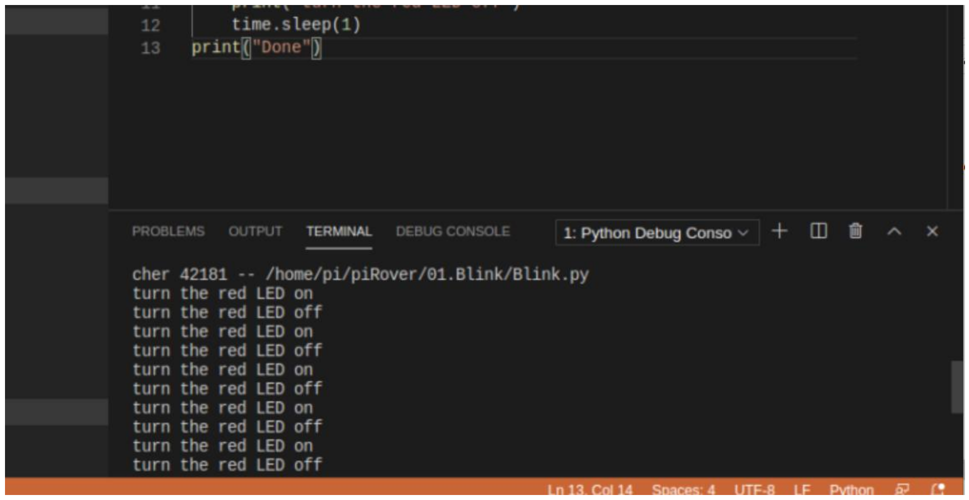
16. Run your code by selecting Start Debugging from the Run menu or just press F5 as shown on the menu.



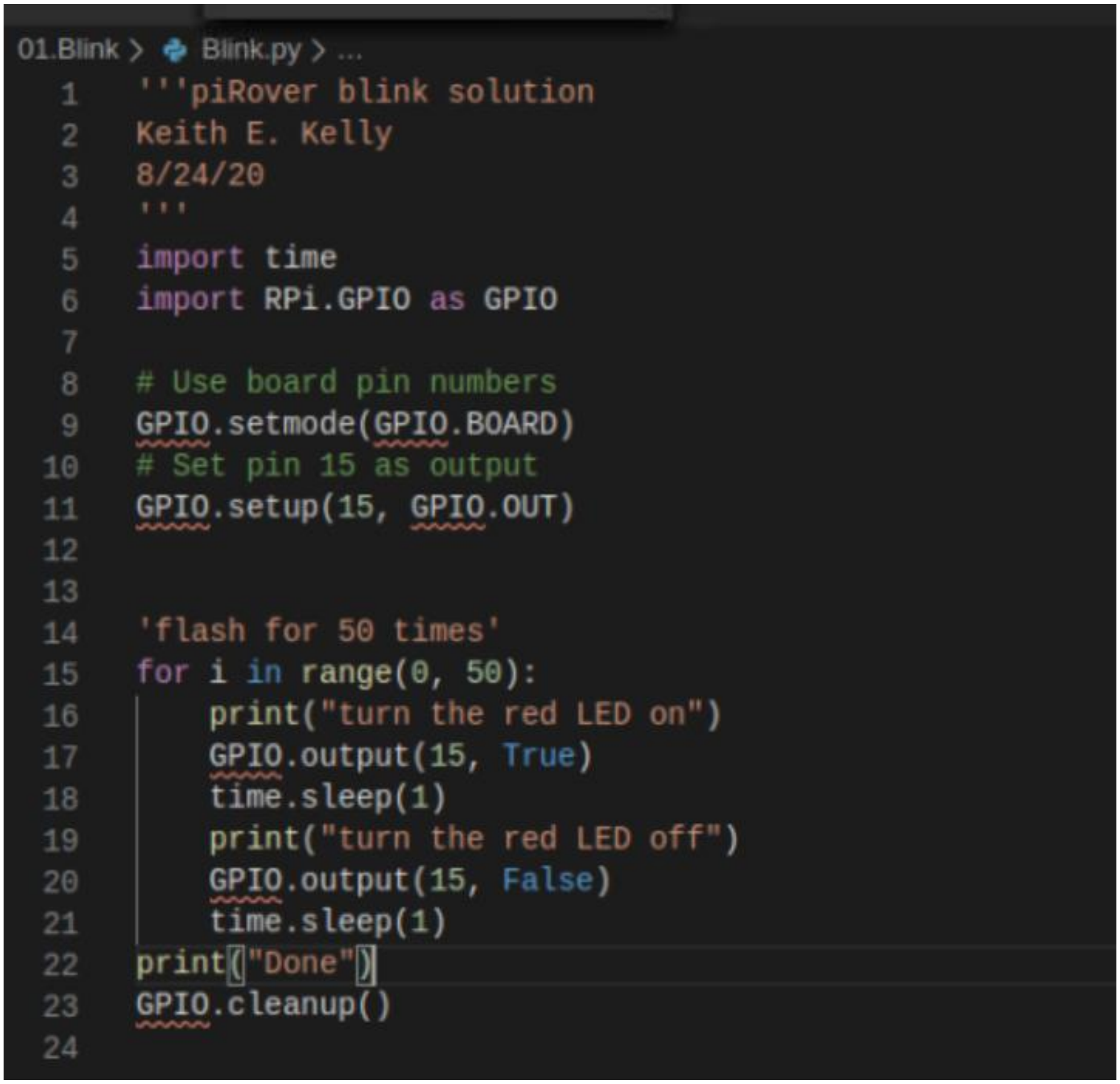
17. Visual Studio needs more detail to determine what you are trying to run. Click "Python File" from the Debug Configuration options shown.


File Edit Selection View Go Run Terminal Help
RUN
Run and Debu
To customize Run and D
a launch.json file.
Select a debug configuration
Debug Configuration
Python File Debug the currently active Python file
Module Debug
Debug the currently active Python file
Remote Attach
Attach using Process ID Attach to a local process
Django Launch and debug a Django web application
Flask Launch and debug a Flask web application

18. The output of the file is shown in the Terminal window displayed at in the lower section of the VS Code window.


```
        time.sleep(1)
    print("Done")
```

```
cher 42181 -- /home/pi/piRover/01.Blink/Blink.py
turn the red LED on
turn the red LED off
turn the red LED on
turn the red LED off
turn the red LED on
turn the red LED off
turn the red LED on
turn the red LED off
turn the red LED on
turn the red LED off
```

19. 

```python
'''piRover blink solution
Keith E. Kelly
8/24/20
'''
import time
import RPi.GPIO as GPIO

# Use board pin numbers
GPIO.setmode(GPIO.BOARD)
# Set pin 15 as output
GPIO.setup(15, GPIO.OUT)


'flash for 50 times'
for i in range(0, 50):
    print("turn the red LED on")
    GPIO.output(15, True)
    time.sleep(1)
    print("turn the red LED off")
    GPIO.output(15, False)
    time.sleep(1)
print("Done")
GPIO.cleanup()

```

Part 3 – Make the connection

Let's start by reviewing the Pi's GPIO connections. The instructor will review the GPIO pin out shown below. Keep in mind that although a 5V connection is available on the header, the Raspberry Pi is a 3.3V device.

piRover Builds with K2

Introduction to GPIO Code   V 1.2

Overview:

In this lesson, you will learn to how to access the input/output (I/O) pins on the Pi using Python and the GPIO library. You'll use the bread board attached to the PiRover to connect an LED with a current limiting resistor to a GPIO pin. You'll configure the pin as an output and blink the LED. The lesson continues by adding a standard logic switch with pull-up resistor to a second GPIO pin. You'll configure this second GPIO pin as an input and enter code to continually read the switch and update the LED based on the state of the switch.

Prerequisites:

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

1. Remote Access Setup

Performance Outcomes:

By the end of this lesson you will be able to:

1. Connect a LED to a GPIO pin using a current limiting resistor.
2. Write code to configure a GPIO output and to control the GPIO output state.
3. Connect a standard logic switch using a pull-up resistor.
4. Write code to configure a GPIO pin for input and to read the state of the GPIO input.

Resources:

The following resources are required to complete this lesson

1. PiRover
2. LED
3. Resistors 220Ω and 10KΩ
4. Push button switch
5. Connection wire
6. blink.py

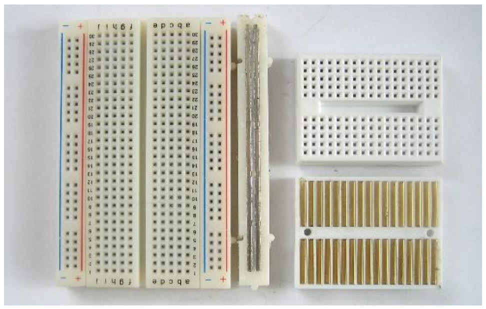
7. InOut.py

Lesson:

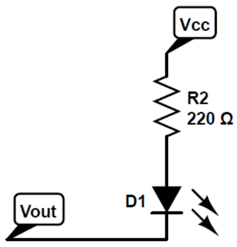
1. This is the first lesson where your code (software) will control electronic hardware. We'll start by building the required circuits. We'll test this hardware manually. Once we're sure we have circuits completed, we'll start writing code.
2. Let's start by reviewing the Pi's GPIO connections. The instructor will review the GPIO pin out shown below. Keep in mind that although a 5V connection is available on the header, the Raspberry Pi is a 3.3V device.

Figure 1 - GPIO pin out (source Jameco.com)

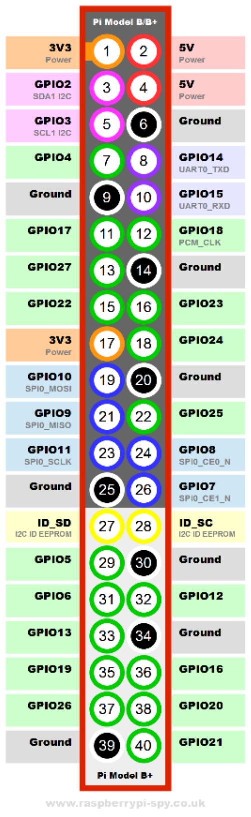
3. Now let's look at the breadboard provided on the initial PiRover configuration. Here's the deconstructed breadboard. The + (red) and – (blue) rails are connected across the entire board. The inner board has rows of 5 holes connected.

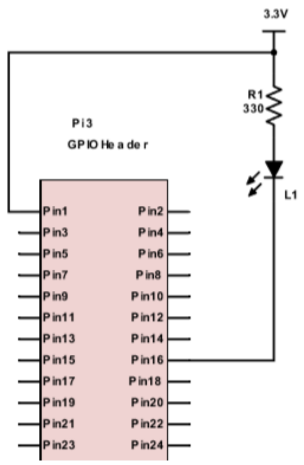


4. Here's the schematic for an LED. Vcc is the supply voltage. In the case of the Pi, it's 3.3V. The Vout point is connected to a GPIO pin. As the output of the GPIO pin is taken high (3.3V) and low (0V), the LED will turn on and off.

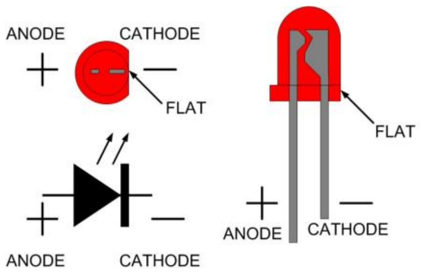


5. Assume the voltage drop across the LED is 1.5 V. How much current is flowing though the GPIO pin when the LED is lit? The maximum current draw from a pin is 16mA. Are we okay connecting this circuit to an output pin?
6. The schematic below shows the final connections for the LED. 3.3 volt power will be provided by Pin 1 of the GPIO header. We'll eventually connect the cathode of the LED to GPIO23 (Pin 16)

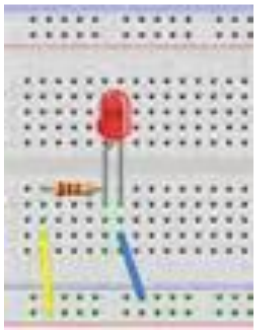

Pi Model B/B+
3V3 Power 1 2 5V Power
GPIO2 SDA1 I2C 3 4 5V Power
GPIO3 SCL1 I2C 5 6 Ground
GPIO4 7 8 GPIO14 UART0_TXD
Ground 9 10 GPIO15 UART0_RXD
GPIO17 11 12 GPIO18 PCM_CLK
GPIO27 13 14 Ground
GPIO22 15 16 GPIO23
3V3 Power 17 18 GPIO24
GPIO10 SPI0_MOSI 19 20 Ground
GPIO9 SPI0_MISO 21 22 GPIO25
GPIO11 SPI0_SCLK 23 24 GPIO8 SPI0_CE0_N
Ground 25 26 GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM 27 28 ID_SC I2C ID EEPROM
GPIO5 29 30 Ground
GPIO6 31 32 GPIO12
GPIO13 33 34 Ground
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
Ground 39 40 GPIO21
Pi Model B+
www.raspberrypi-spy.co.uk



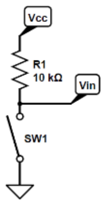
7. Connect the + (red) rail of your bread board to 3.3V pin 1 of the GPIO header and the – (blue) rail to the ground pin 14 of the GPIO header using the connection wires provided.
8. Connect R2 resistor to the + rail and the positive side (anode) of the LED. The negative side (cathode) of the LED will be connected to the GPIO output.



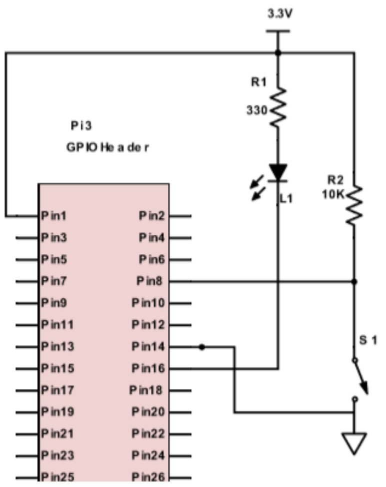
9. Test your LED by removing the cathode's GPIO connection and then adding a connection to ground on the breadboard. A sample is shown below. Turn the Pi control power on (green switch). The LED should be lit.



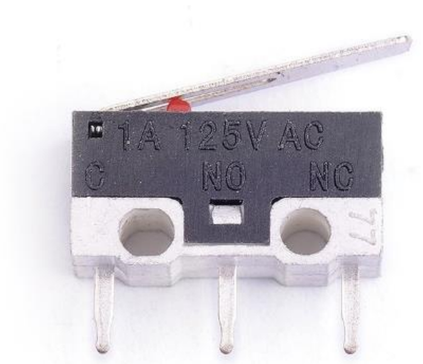
10. Let's build the logic switch. The 10K resistor shown below is known as a pull-up resistor. When the SW1 switch is open, the Vin test point is "pulled-up" to the Vcc voltage level, in this case 3.3V. Closing the switch "pulls" the Vin test point to ground (0V).



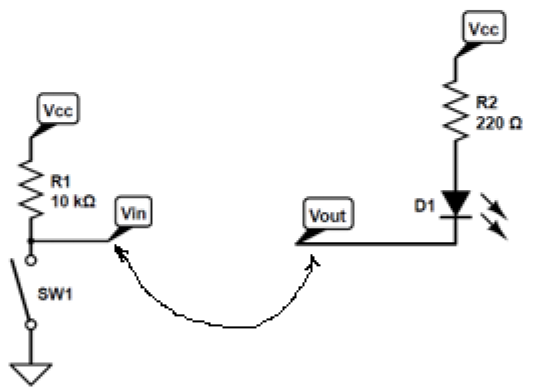
11. Here's the final switch circuit connected to the GPIO headers.

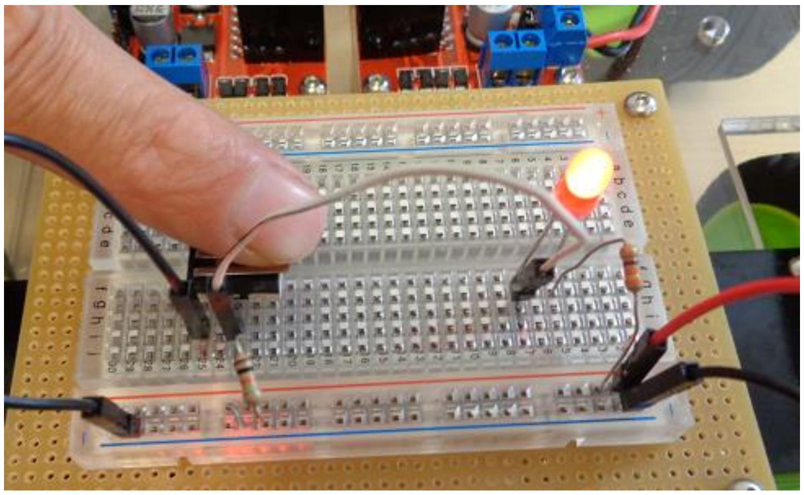


12. When connecting the switch, note the normally closed (NC) and normally open (NO), as well as the common (C) connection on the momentary switch. Connect the common connection point the ground (blue rail) of the breadboard and the normally open (NO) connection point to the 10K pull up resistor. Connect the other side of the 10K to the 3.3V rail.

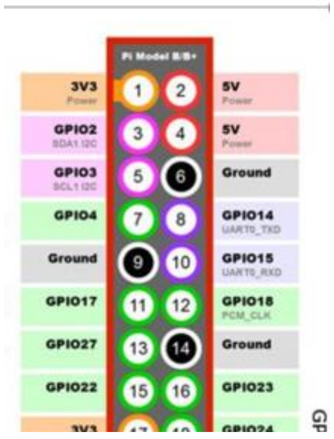


13. Now test the switch by connecting a jumper between the Vin test point of the switch to the Vout test point of LED. Pressing the switch should light the LED.





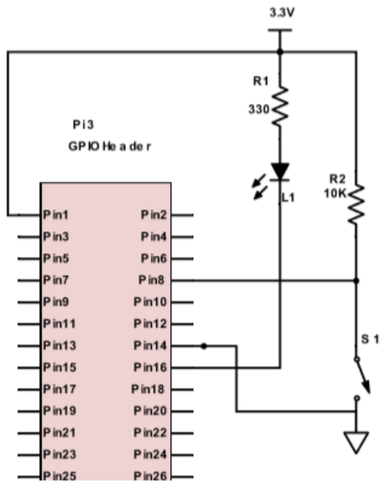
14. Now let’s have the Raspberry Pi read the switch state and control the LED. We’ll start by controlling the LED. Disconnect the switch from the LED and connect the cathode of the LED to GPIO23 on pin 16 of the Pi header.



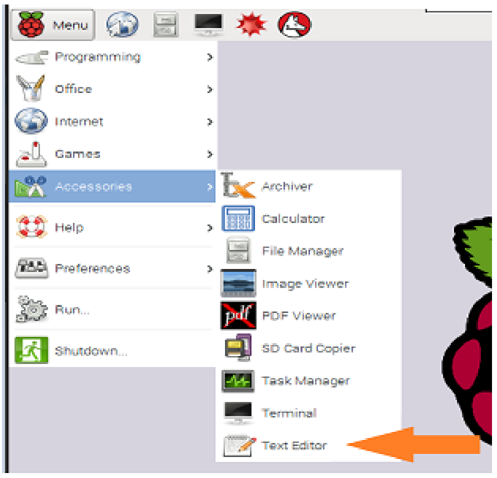
15. Review the schematic below and double check your wiring.

  a. Pin 1 provides the 3.3-volt source to the breadboard

b. Pin 14 provides ground or common to the breadboard
c. Pin 8 is input from the logic switch
d. Pin 16 is the output that controls the LED.



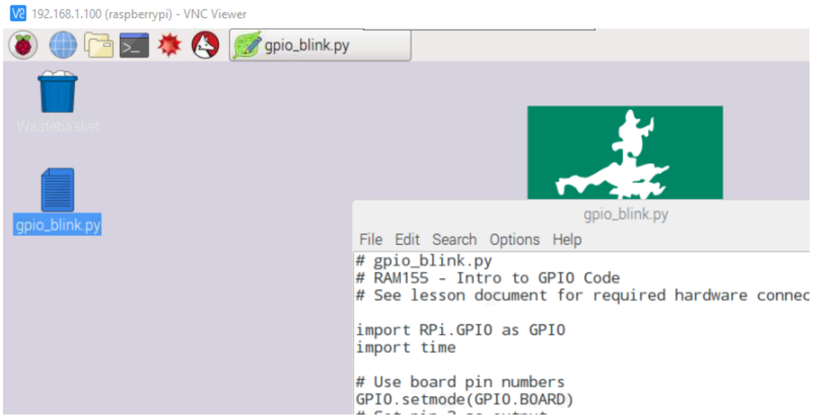
16. Boot your piRover and use the VNC application to access the desktop.
17. Open the text editor on the Pi as shown below. You can also type leafpad at a command prompt.



18. Copy the code below by opening a copy available here.

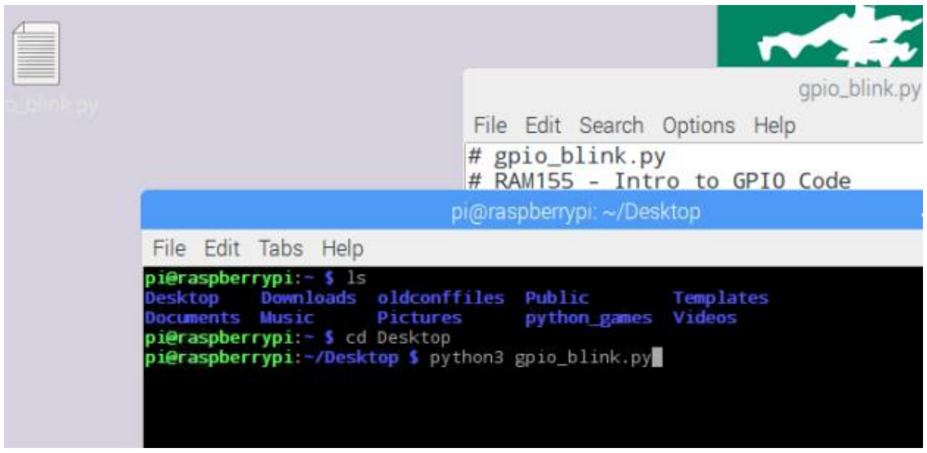
```
# gpio_blink.py
# RAM155 - Intro to GPIO Code
# See lesson document for required hardware connections

import RPi.GPIO as GPIO
import time

# Use board pin numbers
GPIO.setmode(GPIO.BOARD)
# Set pin 16 as output
GPIO.setup(16, GPIO.OUT)

for i in range(0, 50):
    GPIO.output(16, True)
    time.sleep(1)
    GPIO.output(16, False)
    time.sleep(1)
GPIO.cleanup()
```

19. Paste the code into the Pi's text editor. Save the file to the desktop using the name gpio_blink.py as shown below.


192.168.1.100 (raspberrypi) - VNC Viewer
gpio_blink.py
Wastebasket
gpio_blink.py
gpio_blink.py
File Edit Search Options Help

```
# gpio_blink.py
# RAM155 - Intro to GPIO Code
# See lesson document for required hardware connec

import RPi.GPIO as GPIO
import time

# Use board pin numbers
GPIO.setmode(GPIO.BOARD)
```

8. Open a command window and use the Linux "ls" and "cd" commands to navigate to the Desktop directory (folder) as shown below. The path from the root directory is /home/pi/Desktop (For additional help with Linux command line see - https://www.youtube.com/watch?v=AO0jzD1hpXc)



```
# gpio_blink.py
# RAM155 - Intro to GPIO Code
```

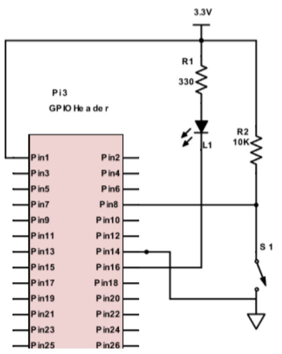
```
pi@raspberrypi:~ $ ls
Desktop    Downloads  oldconffiles  Public        Templates
Documents  Music      Pictures      python_games  Videos
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ python3 gpio_blink.py
```

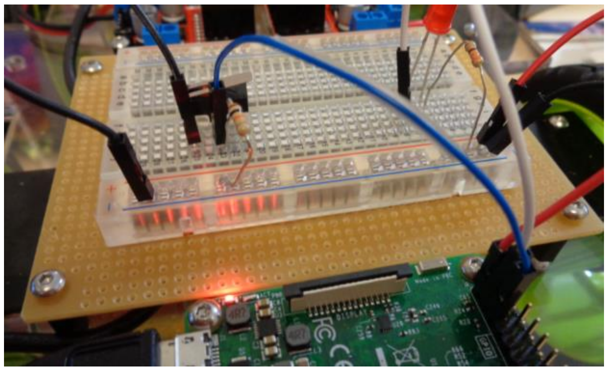
9. Run your blink code by entering the following at the command line.

Python3 gpio_blink.py

10. The LED should blink on and off fifty times.

11. Now let's add the switch to control the LED but through the Pi controller. Verify the switch is connected to GPIO 14 (pin 8) of the GPIO header.





20. Open the text editor and copy the code below by opening the listing available here.

```
# gpio_inout.py
# RAM155 - Intro to GPIO Code
# See lesson document for required hardware connections

import RPi.GPIO as GPIO
import time

# Use board pin numbers
GPIO.setmode(GPIO.BOARD)
# Set pin 16 as output
GPIO.setup(16, GPIO.OUT)
# Set pin 8 as input
GPIO.setup(8, GPIO.IN)

state = False

for i in range(0, 50):
    state = GPIO.input(8)  #get switch state
    GPIO.output(16, state)   #use state value to update LED
    time.sleep(.5)
    GPIO.cleanup()
```

12. Save the file as gpio_inout.py. This code reads the value of the switch and updates the LED.

13. Run your code using the following command and test.

```
Python3 gpio_inout.py
```

14. What if the opposite operation was required? In other words, you want the LED to normally be on but turn off when the switch is pressed. Make the required software modification and demonstrate to the instructor. How much easier is this than updating hardware connections?

Assessment

1. **Complete the Introduction to GPIO Code Activity.**

Introduction to Visual Studio Code

Visual Studio Code is a powerful code editor. It is a free, lightweight, open-source editor that helps you write professional code for Windows, Mac, and Linux platforms. It includes support for debugging, Git source control, and code completion. Its simplified edit, build, and debug cycle makes it ideal for both the beginner and the serious professional. VS Code comes with built-in support for JavaScript, TypeScript and Node.js, but you will be installing an extension providing support for Python.

Review the Stack Overflow 2019 Survey at https://insights.stackoverflow.com/survey/2019 and you will see that it is one of the top, if not the top, editors used by developer. See additional information on why VS Code is preferred at this link – Why VS Code? In short, this is a great tool. You will want to learn it well!

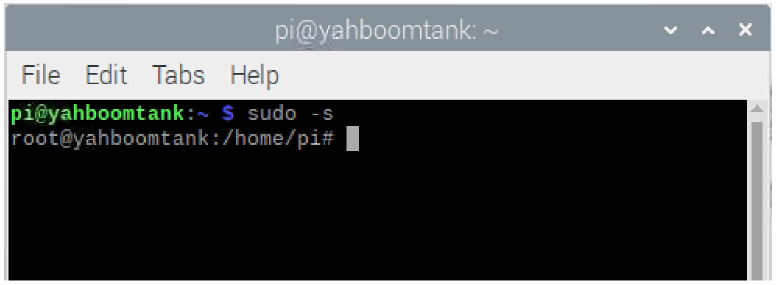
Installing VS Code

Thanks to Jay Rodgers at code.headmelted.com the installation of Visual Studio Code on the ARM processor of the Raspberry Pi is easy. See Scott Hanselman's post How to install Visual Studio Code on a Raspberry Pi 4 in minutes for additional details.

1. Open and view your piRover desktop using VNC Viewer.

2. Open a terminal window and enter the following command. This command enables the following script to run with root access. Recall that sudo is "super-user-do" and is just like elevating your rights to administrator on a Windows workstation.

   sudo -s

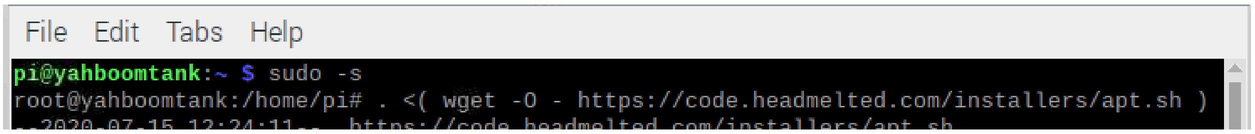


```
pi@yahboomtank:~ $ sudo -s
root@yahboomtank:/home/pi#
```

3. Note the change in the terminal window prompt when you enter this mode. You are now running a script as "root" rather the "pi" user.
4. Copy the following to the command line to execute the installation script from code.headmelted.

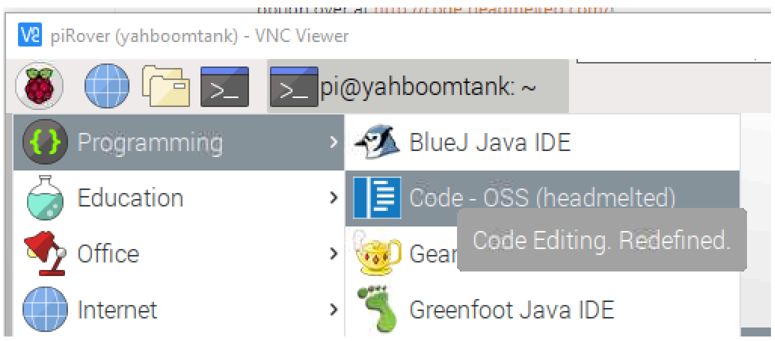
   . <(wget -O - https://code.headmelted.com/installers/apt.sh)

5. It is best to copy the line above and paste it into the terminal window. If you are typing it out, the command starts with the period and then a space. Refer to the screen capture below.



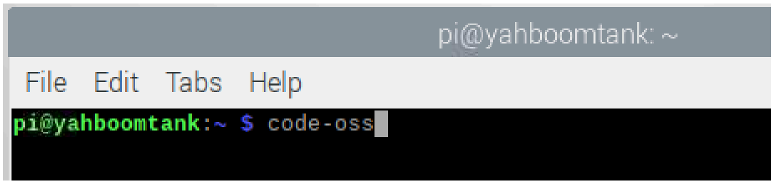
```
pi@yahboomtank:~ $ sudo -s
root@yahboomtank:/home/pi# . <( wget -O - https://code.headmelted.com/installers/apt.sh )
```

6. The installation runs, and you will see an installation complete! message. Close the terminal window to remove root access.
7. Visual Studio Code is now available as Code-OSS. Open the Raspberry Pi menu and navigate to the Programming group. Code-OSS (headmelted) is available.



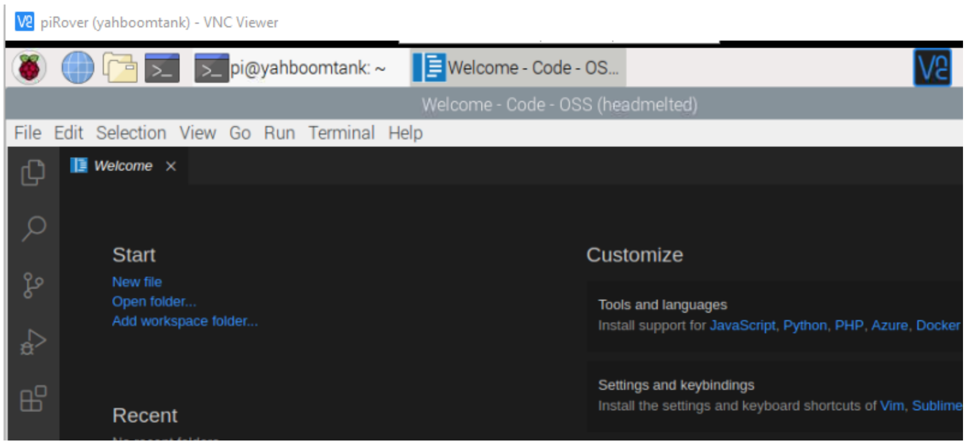
8. You can also launch the VS Code environment from any terminal prompt by entering **code-oss**. Let's open VS Code and run the ColorLED_v3.py program introduced in the Python Getting Started activity.

9. Open the terminal window and launch VS Code using the **code-oss** command.

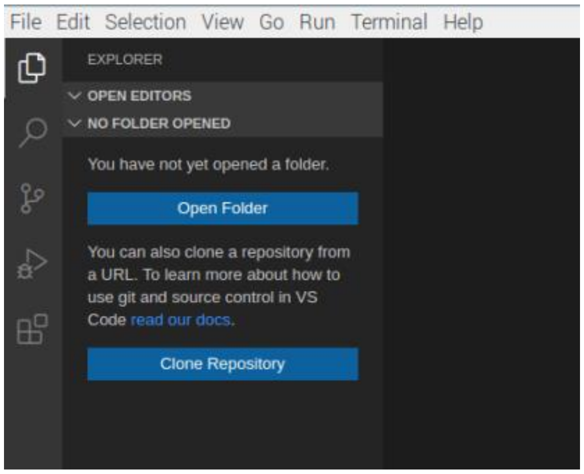


```
pi@yahboomtank:~ $ code-oss
```

10. VS Code launches with the Welcome screen visible. There is a lot of good information on the Welcome screen but close for now.


piRover (yahboomtank) - VNC Viewer
pi@yahboomtank: ~
Welcome - Code - OS...
Welcome - Code - OSS (headmelted)
File Edit Selection View Go Run Terminal Help
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
Customize
Tools and languages
Install support for JavaScript, Python, PHP, Azure, Docker
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime

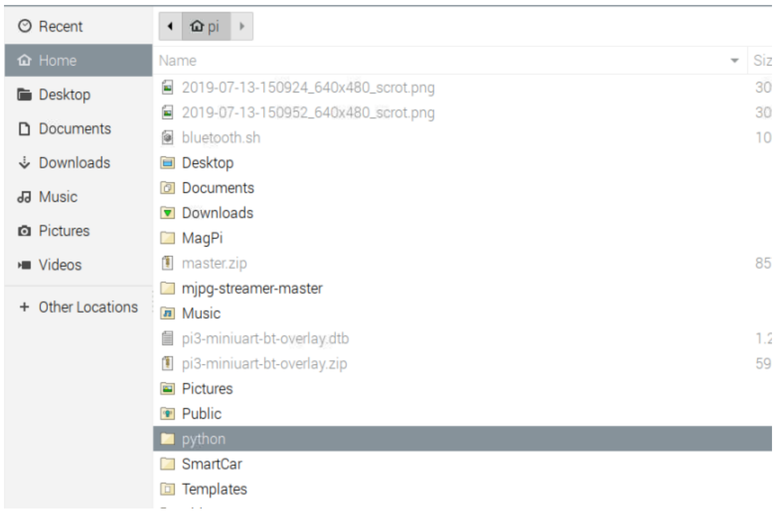
11. Open the Python folder that contained the ColorLED_v3.py file using the File Explorer icon at the top of the Activity Bar at the left.



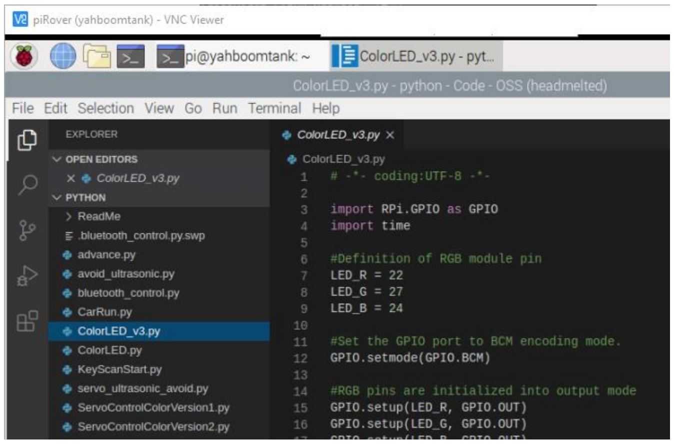
You have not yet opened a folder.

You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

12. Click the Open Folder button and then select Home from the directory list on the left and then python from the list of folders in the main window.

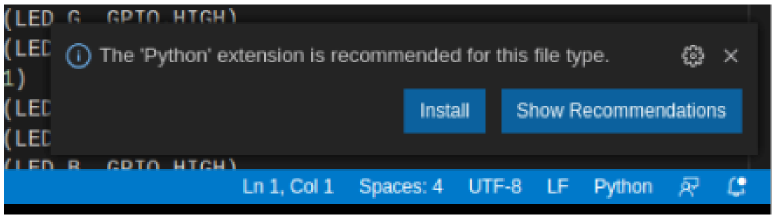


13. The folder contents are now available in the Explorer pane. Click the ColorLED_v3.py file created in the prior assignment. It opens in the editor.

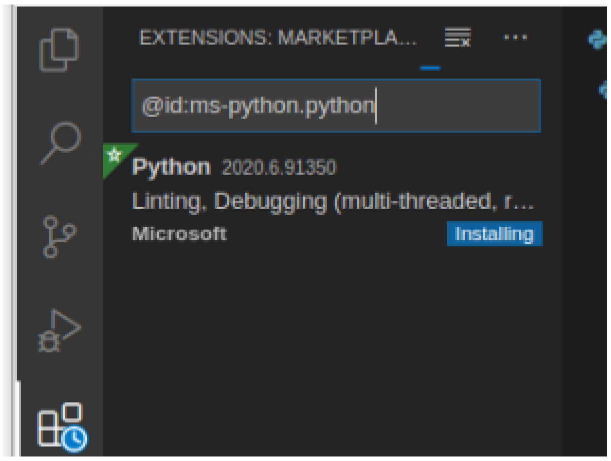

piRover (yahboomtank) - VNC Viewer
pi@yahboomtank: ~
ColorLED_v3.py - pyt...
ColorLED_v3.py - python - Code - OSS (headmelted)
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
ColorLED_v3.py
PYTHON
ReadMe
.bluetooth_control.py.swp
advance.py
avoid_ultrasonic.py
bluetooth_control.py
CarRun.py
ColorLED_v3.py
ColorLED.py
KeyScanStart.py
servo_ultrasonic_avoid.py
ServoControlColorVersion1.py
ServoControlColorVersion2.py
ColorLED_v3.py

```
# -*- coding:UTF-8 -*-

import RPi.GPIO as GPIO
import time

#Definition of RGB module pin
LED_R = 22
LED_G = 27
LED_B = 24

#Set the GPIO port to BCM encoding mode.
GPIO.setmode(GPIO.BCM)

#RGB pins are initialized into output mode
GPIO.setup(LED_R, GPIO.OUT)
GPIO.setup(LED_G, GPIO.OUT)
```

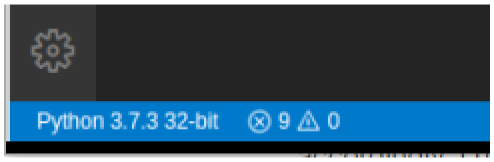
14. Look at the notification that is displayed in the lower right-hand corner of the editor. VS Code has recognized this file as Python code and is prompting you to install the related extension that enables Python in VS Code. Click Install to add this extension to VS Code.


The 'Python' extension is recommended for this file type.

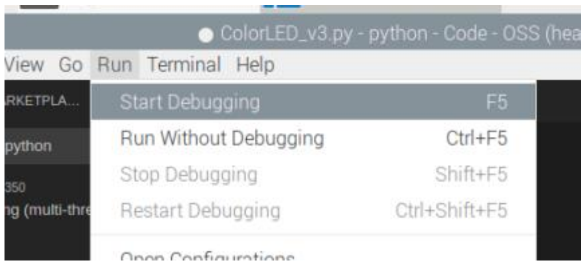
15. The Activity Bar switches to the Extensions tool and support for Python is installed.



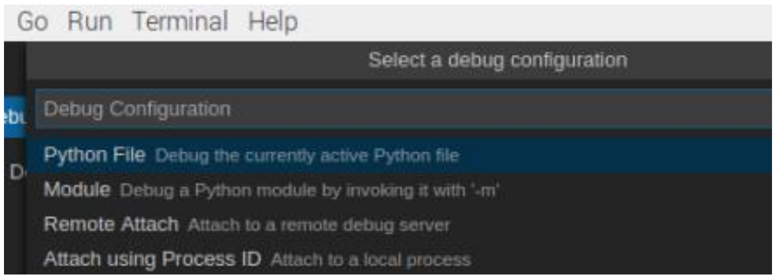
16. Now view the status bar at the bottom of the VS Code windows. This shows that you are currently operating in the version 3.7 environment.



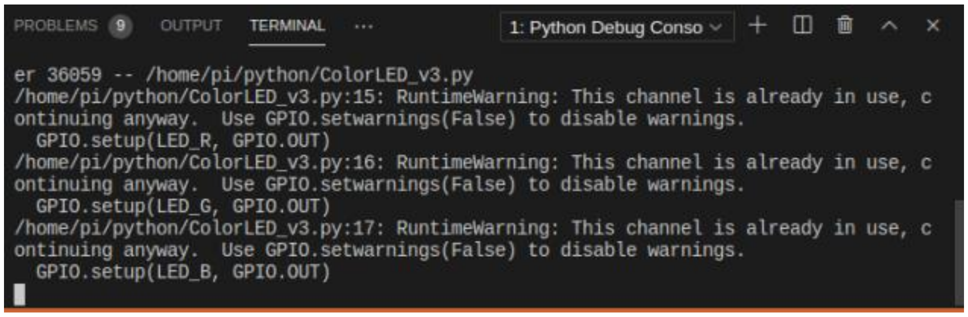
17. Run the ColorLED_v3.py program by selecting Run from the menu and then Start Debugging. Note that you can also use the F5 function key to start the program.


● ColorLED_v3.py - python - Code - OSS (hea
View Go Run Terminal Help
Start Debugging F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations

18. The Command Palette opens prompting for you to select a configuration for the debug session. Select the first option "Python File" and the program will run.

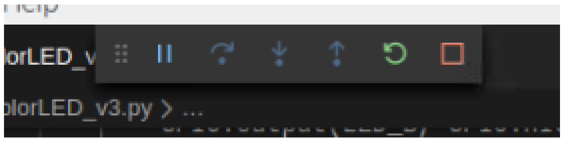

Go Run Terminal Help
Select a debug configuration
Debug Configuration
Python File Debug the currently active Python file
Module Debug a Python module by invoking it with '-m'
Remote Attach Attach to a remote debug server
Attach using Process ID Attach to a local process

19. The output from the program is displayed in the terminal pane. You will review these warning statements later, but for now, the code runs and the LEDs flash.



```
er 36059 -- /home/pi/python/ColorLED_v3.py
/home/pi/python/ColorLED_v3.py:15: RuntimeWarning: This channel is already in use, c
ontinuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED_R, GPIO.OUT)
/home/pi/python/ColorLED_v3.py:16: RuntimeWarning: This channel is already in use, c
ontinuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED_G, GPIO.OUT)
/home/pi/python/ColorLED_v3.py:17: RuntimeWarning: This channel is already in use, c
ontinuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED_B, GPIO.OUT)
```

20. To stop the execution, click the red square icon in the debug tools at the top of the window.



21. There is a lot more to investigate and learn both about Python and the Visual Studio Code environment. But, for now, you were able to load a Python file into the VS Code editor and run the program.

22. This activity is the final action in the first section (Sprint) in the course. At this point in the course, you must have the software and other tools installed so that you can write and execute Python code on your system. If you are having difficulty be sure that you are in contact with the instructor. Again, you must have a functional system as you move into the next section of the course.

Assessment:

Run the ColorLED_v3.py file again in VS Code. Use a screen capture tool to record an image of the VS Code window with the Python code running. Save this file as **VSCodeWithPython.jpg**. Submit this file to Moodle along with other files in this week's zip file.