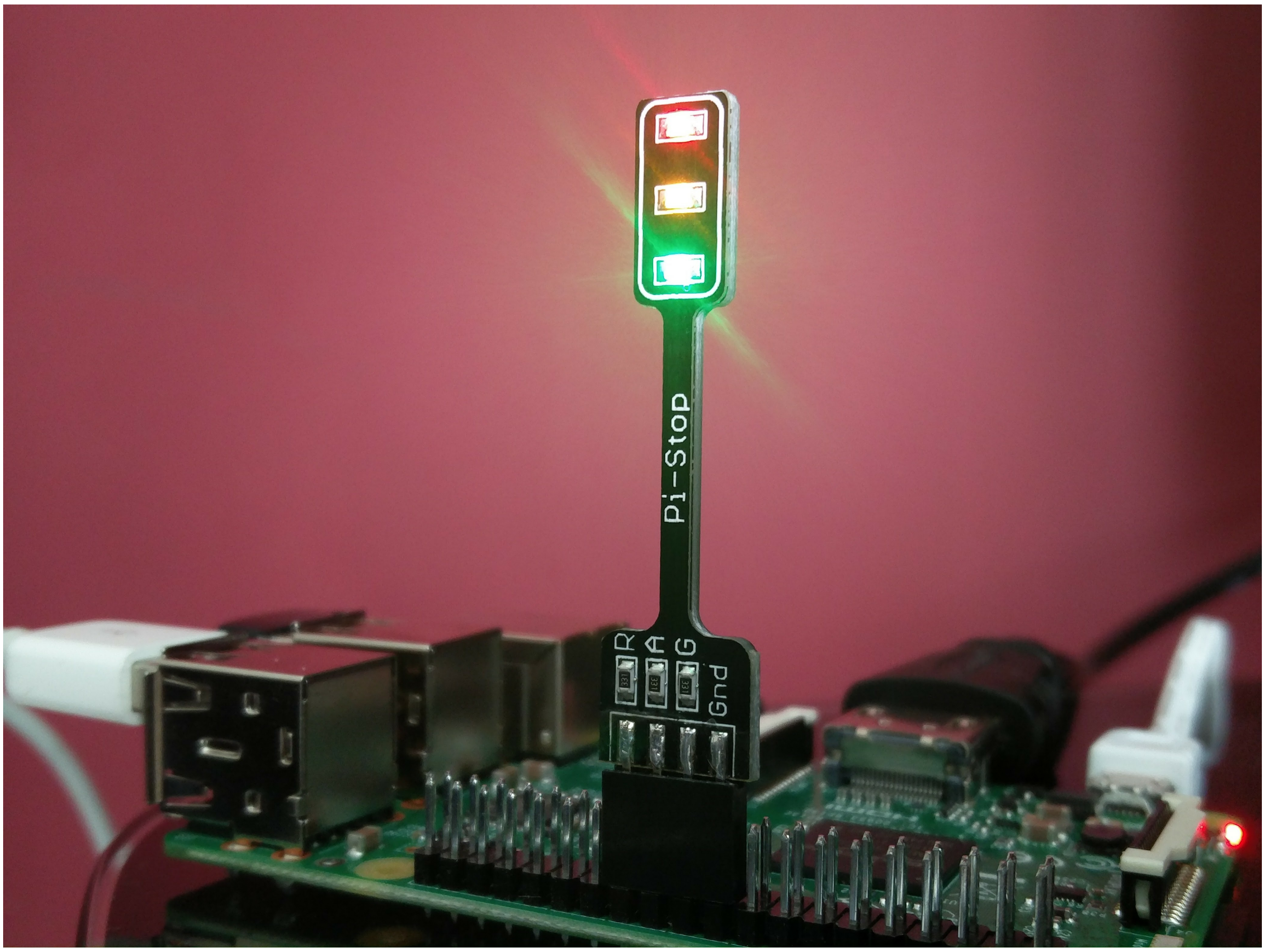
Controlling the Pi-Stop with Python



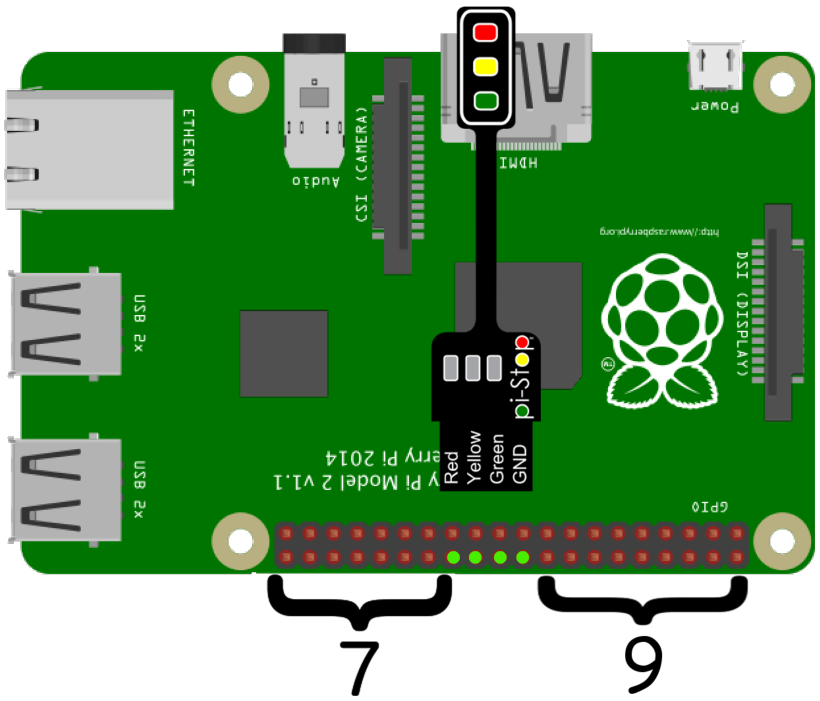
What is the Pi-Stop?

The Pi-Stop is a mini traffic light that uses four pins to interact with the Pi, three of which are inputs for the 3 different LEDs (red, amber and green) and the fourth is for ground to complete the circuit.

What are we doing in this worksheet?

We are going a control the Pi-Stop using the programming language Python. We will go through using Python, to interact with the Raspberry Pi's GPIO to turn on and off the LEDs whilst introducing `while` loops and and the `time.sleep()` function.

Attaching the Pi stop

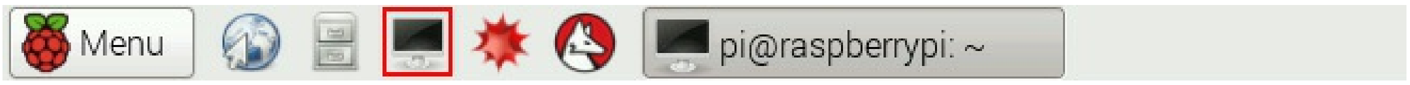


Firstly ensure that your Pi is **powered off and unplugged**, this is very important. Make sure when attaching the Pi-Stop that the LEDs are **facing away** from your Pi. The diagram above shows the pins that you attach the Pi-Stop to, as well as which direction it should be facing.

TIP : There should be **7** pins **to the left** of the Pi-Stop so you could count seven pins across and then insert one end of the Pi-Stop on the eighth pin.

After you have attached the Pi-Stop the LEDs should light up dimly.

Opening and setting up Python



To open the Python 3 shell (IDLE3) we will first need to click the terminal icon on the top bar (highlighted by the box in the picture above). Once terminal has opened type in `sudo idle3` and press enter. Python IDLE should now open.

TIP : Be sure to create a new script for your code by going to File > New File

How to use the Pi-Stop

In Python we need to import the GPIO functions from the `RPi.GPIO` library to talk to the GPIO pins. We will import this library using the `from` and `import` functions to only import the part of the GPIO library we need.

```
from RPi.GPIO import GPIO
```

Next we need to make sure that we setup RPi.GPIO correctly with the right mode. To do this, add.

```
GPIO.setmode(GPIO.BOARD)
```

Then to talk to the LEDs, we need to set up the pins before we can turn them on or off. To do this, we use the `GPIO.setup` function which takes a pin number and a setup value of either

- `GPIO.IN` for an input pin (eg. for buttons or sensors).
- `GPIO.OUT` for an output pin (eg. for LEDs).

Since we will be outputting the the LEDs, we will use the `GPIO.OUT` value.

An example for the red LED is show below

```
GPIO.setup(26, GPIO.OUT)
```

So now that we have set everything up, your code should look something like below.

```
from RPi.GPIO import GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(26, GPIO.OUT)
```

Turning one LED on and off

For now, we'll focus on the red LED. The red LED matches to pin 26 and to output power to pin 26 we will use the `GPIO.output` function. This function takes a pin number and a `True` or `False` value to enable (or disable) a power output to it. We will need to repeat this each time we want to turn on of off the LED!

Output function for red LED

```
GPIO.output(26, True)
```

Now try running your code (**hit F5 on the keyboard, or click on Run-> Run Module**)

And to turn it off, change the `True` to `False`.
If it worked, Awesome! Now lets try and turn the other two coloured LEDs on and off.

GPIO Pin key for LEDs

To help, here is a table with all of the lights with matching pin numbers.

Light	Pin	Variable Name
Red	26	red_led
Amber	24	amber_led
Green	22	green_led

To make referencing them easier, we are going to create some variables as a variable (in this case, basically a shortcut) for each of the LEDs, here is the example for the red LED and using it in the function...

```
red_led = 26

GPIO.output(red_led, True)
```

Repeat this for each LED pin.

To light up the different LEDs we will repeat the `GPIO.output` function another four times, two to turn on the LEDs and two to turn off the LEDs.

The other two LEDs work the same way as the red LED's output function, just with different pin numbers. Go ahead and make the other two output functions for the amber and green LEDs

Making the LEDs Loop

So now that we can turn the LEDs on and off, let's try to make them run by themselves. To do this we are going to use a `while` loop.
A while loop uses a condition to check if it should continue looping. Since we want it to loop forever we will set the condition to `True`.

```
while True:
    GPIO.output(red_led, True)
```

We will also need to use the `time.sleep` function. To do this, we need to import the `time` library and to use it you set the amount of time in seconds you want the script to sleep for.

```
time.sleep(1)
```

So lets focus on the red LED first...

To tell the LED to turn on, we again use the `GPIO.output` function that was mentioned in the last section. Then we will make it wait for three seconds before turning it off and waiting for another three seconds. This entire section will then loop forever.

Try making and testing the code to make the red LED flash!

TIP : **If you are having trouble**, don't be scared to grab one of the volunteers to help you out.

Making the Pi-Stop into a traffic light

What can we do with the Pi-Stop now? Why not make it act like a real traffic light? Lets replicate the work we did before and make the Pi-Stop act like a real traffic light!

This is how the Sequence for our traffic light works...

Action	Sleep Time After(in seconds)
Green light on	2
Green off and then amber light on	1
Amber off and then red light on	3
Amber on	1
Red and amber off then green on	1
Green light off	1

What next?

- Perhaps try hooking up multiple Pi-Stops and control then all at the same time? (ask a jam volunteer for help for where to connect extra PiStops to).
- Why not connect Wizard Signal (traffic light) to the Pi-Stop and control both using Python?
- Maybe try different ways to turn Pi-Stop on and off? (eg. touch, PIR sensor).