**Programming the Physical World**

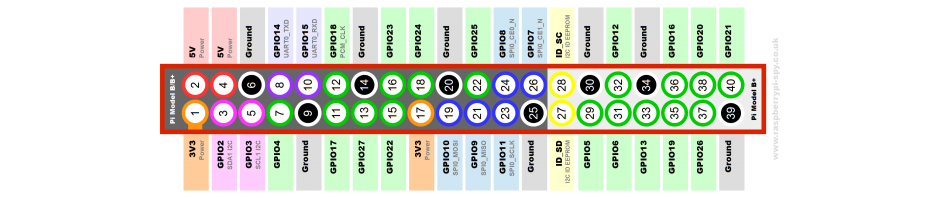
**Stop Light Simulator**

In this lab we are going to write a program that interfaces with computer hardware to make something physical occur, specifically blink Morse code on a Light Emitting Diode (LED).

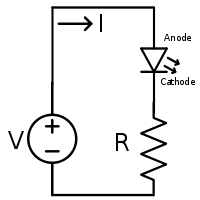
For this activity we will require the following:

1. Raspberry Pi
2. Four male/female jumper wires
3. Solderless breadboard
4. Red LED, Yellow LED, Green LED
5. 330 Ω resistor
6. HDMI to DVI cable

The hardware we will be using will be the Raspberry Pi which has 18 GPIO (General Purpose Input Output) pins which can be easily programmed to send output signals to or receive input signals.



For this activity, we will be using pins 6, 8, 10 and 12. These correspond to Ground and GPIO Pin 14, 15, and 18 respectively. Here is a diagram of an LED setup. To get the LED to light we need approximately 2.1 volts across it. The purpose of the resistor is to limit the current through the LED. Pin 18 is capable of supplying 5 volts. If we do not use a current limiting resistor, the LED will light brightly for a while and then go dark. For a nominal brightness we want about 10 mA to flow through the LED. This means a 330 Ω resistor should do. Actually we should be OK with a resistor or 180 Ω to 550 Ω.



We will be using a solderless breadboard to hook up the components. Pin 6 (Ground) should hook to one end of the resistor. Pin 12 (GPIO 18) should hook to the Anode of the LED (the longer lead).

**Hardware Assembly Procedure**

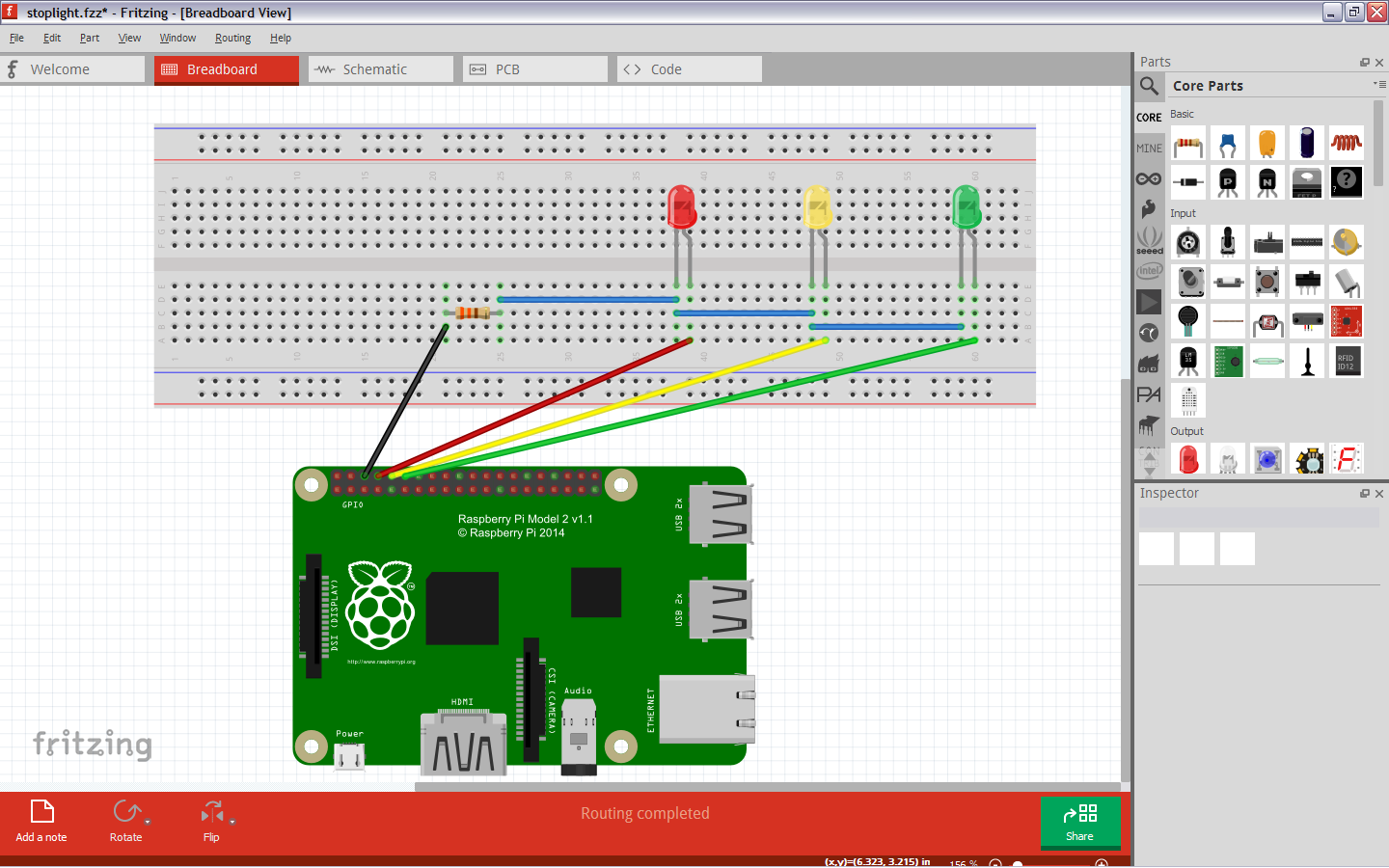
1. Add 330 Ω resistor to the breadboard
2. Add Red LED to breadboard so Cathode of LED (shorter lead) is connected to resistor
3. Add Yellow LED to breadboard so Cathode of LED (shorter lead) is connected to resistor
4. Add Green LED to breadboard so Cathode of LED (shorter lead) is connected to resistor
5. Connect Pin 6 on Raspberry Pi GPIO header to unconnected lead of resistor
6. Connect Pin 8 on Raspberry Pi GPIO header to Anode of Red LED (longer lead)
7. Connect Pin 10 on Raspberry Pi GPIO header to Anode of Yellow LED (longer lead)
8. Connect Pin 12 on Raspberry Pi GPIO header to Anode of Green LED (longer lead)

Note: If you have the LED hooked up backwards it will not light and you can correct this by reversing its connections.

Green

Yellow

Red



Now we are ready to program the GPIO pins. In order to do this we will write a short Python script. The RPi.GPIO library is the software that allows us easy access to the hardware. The Operating System running on the Raspberry Pi is a variation of Debian Linux. What this means to you is to directly access the hardware, you must have root. This means we must run IDLE as root. Linux has a mechanism to do this called sudo which allows you to run programs with the security privileges of another user (typically the superuser). This is a concatenation for superuser do. Start IDLE as follows:

1. Open the Run Dialog (Start|Run)
2. Type sudo idle
3. Hit Enter

Once you have the IDLE shell opened, open a new window (File|New Window)

Note: Make a folder on your desktop called MyPython for storing your programs.

**First Experiment**

Type in and run the following script to verify our LEDs are hooked up correctly:

#Turn LED on

import RPi.GPIO as GPIO

GPIO.setwarnings(False) #Disable warnings for any not reset pins

GPIO.setmode(GPIO.BCM) #Allow ARM processor to access GPIO

GPIO.setup(14, GPIO.OUT) #set pin 14 to output

GPIO.setup(15, GPIO.OUT) #set pin 15 to output

GPIO.setup(18, GPIO.OUT) #set pin 18 to output

GPIO.output(14, True) #Turn Red LED on

GPIO.output(15, True) #Turn Yellow LED on

GPIO.output(18, True) #Turn Green LED on

Now let's make our stoplight simulation:

Type in the following script:

#Stop light simulation

import RPi.GPIO as GPIO

import time

GPIO.setwarnings(False) #Disable warnings for any not reset pins

GPIO.setmode(GPIO.BCM) #Allow ARM processor to access GPIO

GPIO.setup(14, GPIO.OUT) #set pin 14 to output (Red LED)

GPIO.setup(15, GPIO.OUT) #set pin 15 to output (Yellow LED)

GPIO.setup(18, GPIO.OUT) #set pin 18 to output (Green LED)

#set initial condition of all lights off

GPIO.output(14, False) #Turn Red LED off

GPIO.output(15, False) #Turn Yellow LED off

GPIO.output(18, False) #Turn Green LED off

time.sleep(2)

#Make LED blink

while True:

#Green

GPIO.output(18, True)

time.sleep(6)

GPIO.output(18, False)

#Yellow

GPIO.output(15, True)

time.sleep(2)

GPIO.output(15, False)

#Red

GPIO.output(14, True)

time.sleep(6)

GPIO.output(14, False)