Using Pulse-Width Modulation (PWM) and the Distance Sensor Module

1. Background Information

What is PWM?

Pulse Width Modulation (PWM) is a method of approximating an analog signal, using a digital source. PWM is used in a wide range of applications from dimming an LED to controlling the direction of a DC motor.

A PWM signal has two key parameters: **duty cycle** and **frequency**.

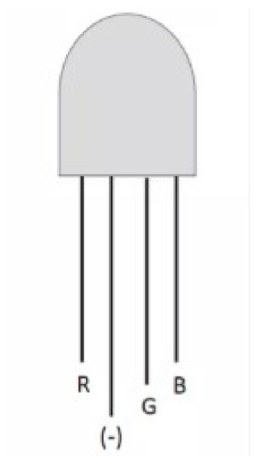
The **duty cycle** describes the amount of time the signal is high (on) as a percentage of the total time of it takes to complete one cycle (period).

The **frequency** determines how many pulses are sent each second (in Hertz).

When we turn a digital signal off and on at a sufficient rate, the voltage seen by certain devices (like an LED) will be proportional to the duty cycle. Thus, we can approximate voltages between the "HI" and "LOW" values provided by the RPi.

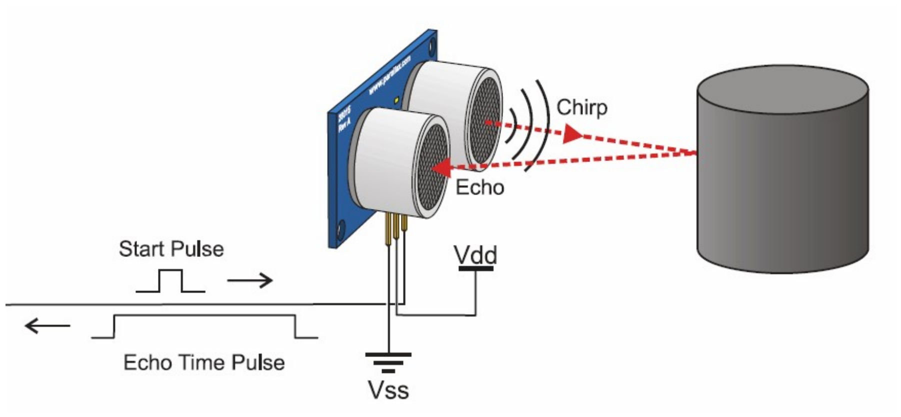
What is an RGB LED and how does it work?

An RGB LED is a device that contains 3 types of LEDs (Red, Blue, and Green) in the same casing. These 3 colours are combined at various intensities (using PWM) to produce the colours of the RGB gamut. RGB LEDs have 4 pins (R, B, G, and ground).



WHAT IS THE HC-SR04 DISTANCE SENSOR AND HOW DOES IT WORK?

The ultrasonic sensor sends out a high-frequency sound pulse and measures how long it takes for the echo of the sound to reflect back. The sensor had 2 openings on its front face: one transmits ultrasonic waves (like a tiny speaker), and the other receives them (like a tiny microphone). The echo pin produces a HI signal for the time it takes the output signal to travel to the object and back. Thus, the width of the echo pulse is proportional to the distance from the object.

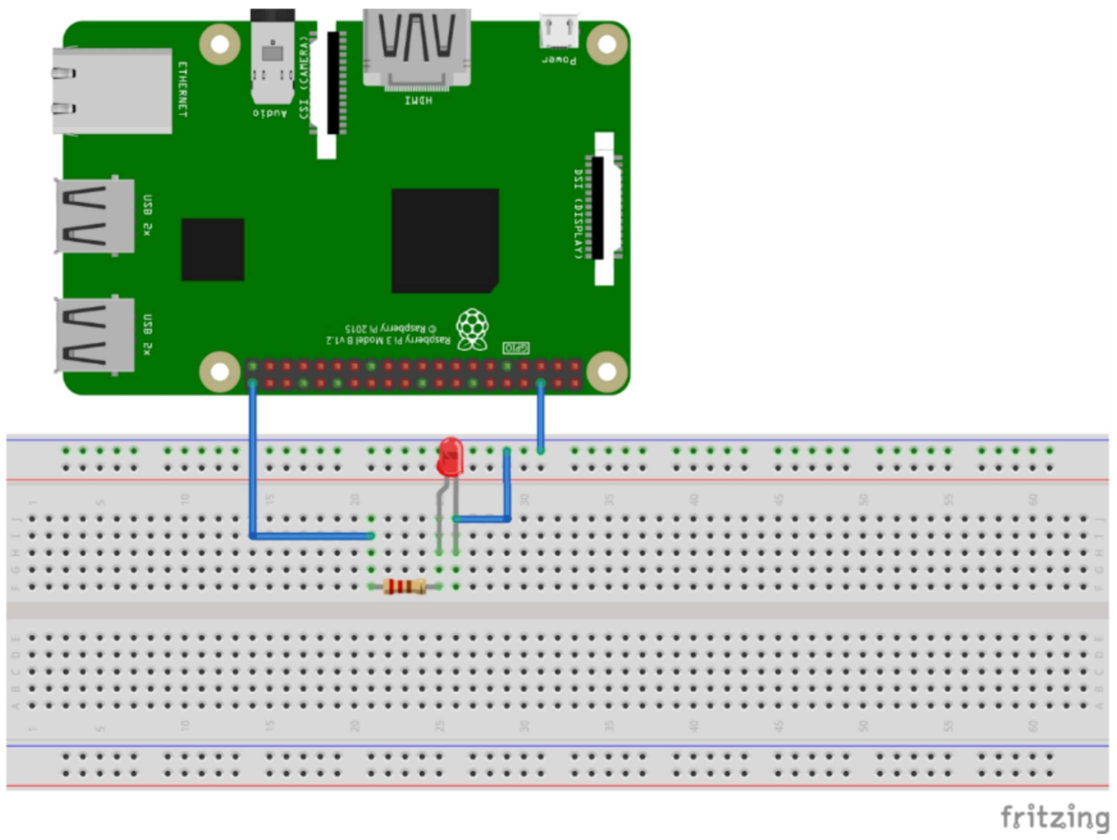


Hints:

- We've found that using the IDLE IDE(for Python 3.X) conserves system resources
- Print statements are your friend if the code you wrote is not performing as expected. Include them at key parts of your code to help you debug.
- Always connect LEDs through resistors as shown on the circuit diagrams.
- Not all RPi pins support PWM. Only pins GPIO2 to GPIO27 can be used for this purpose. PWM pins must also be initialized in a particular way.
- LEDs have both a negative lead and a positive lead. The positive lead is always longer than the negative lead. Connect negative to ground. **If your LED doesn't light up, try reversing the polarity.**

2. Use PWM to dim an LED

Familiarize yourself with the RPi PWM functions by writing a program which uses PWM to alter the brightness of an LED. Use the circuit diagram and code snippet below to get started.



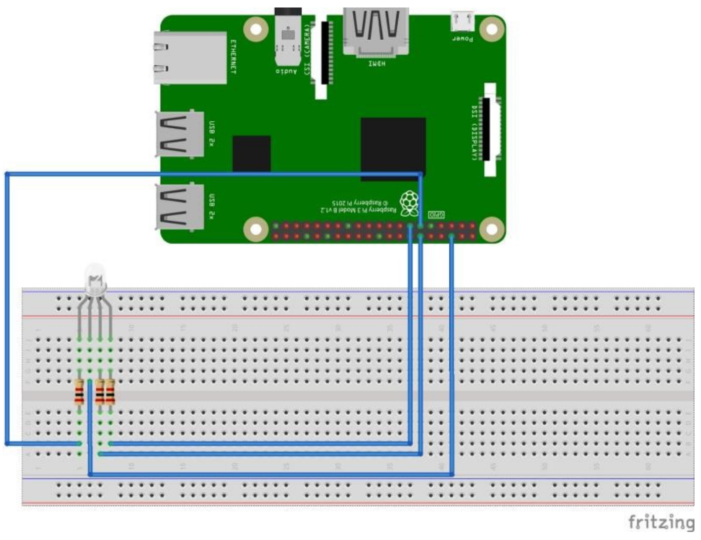
PWM pins must be initialized in a particular way, as shown in the snippet below. Change the "duty cycle" to alter the brightness of the LED.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(led_R, GPIO.OUT)           #initialize pin as output
pwm_R = GPIO.PWM(led_R, 5000)         #initialize PWM pin, frequency = 5kHz
pwm_R.start(0)                        #start pin at 0% duty cycle
pwm_R.ChangeDutyCycle(??)             #modify this statement and look at LED
```

3. Use PWM to change the colour of an RGB LED

You can use PWM to alter the colour of the RGB LED. Use circuit depicted below and the example code on the next page to get started. It might be helpful to keep the colour combinations below in mind.

Colour	Colour Combination Needed
Yellow	Green + Red
Cyan	Green + Blue
Magenta	Red + Blue



Hints:

- On the RGB led the longest lead is negative (Opposite to the regular LED), so connect this lead to ground
- All three pins must be connected to the LED through their own resistor.
- The duty cycle values accepted are floats between 0 and 100.0

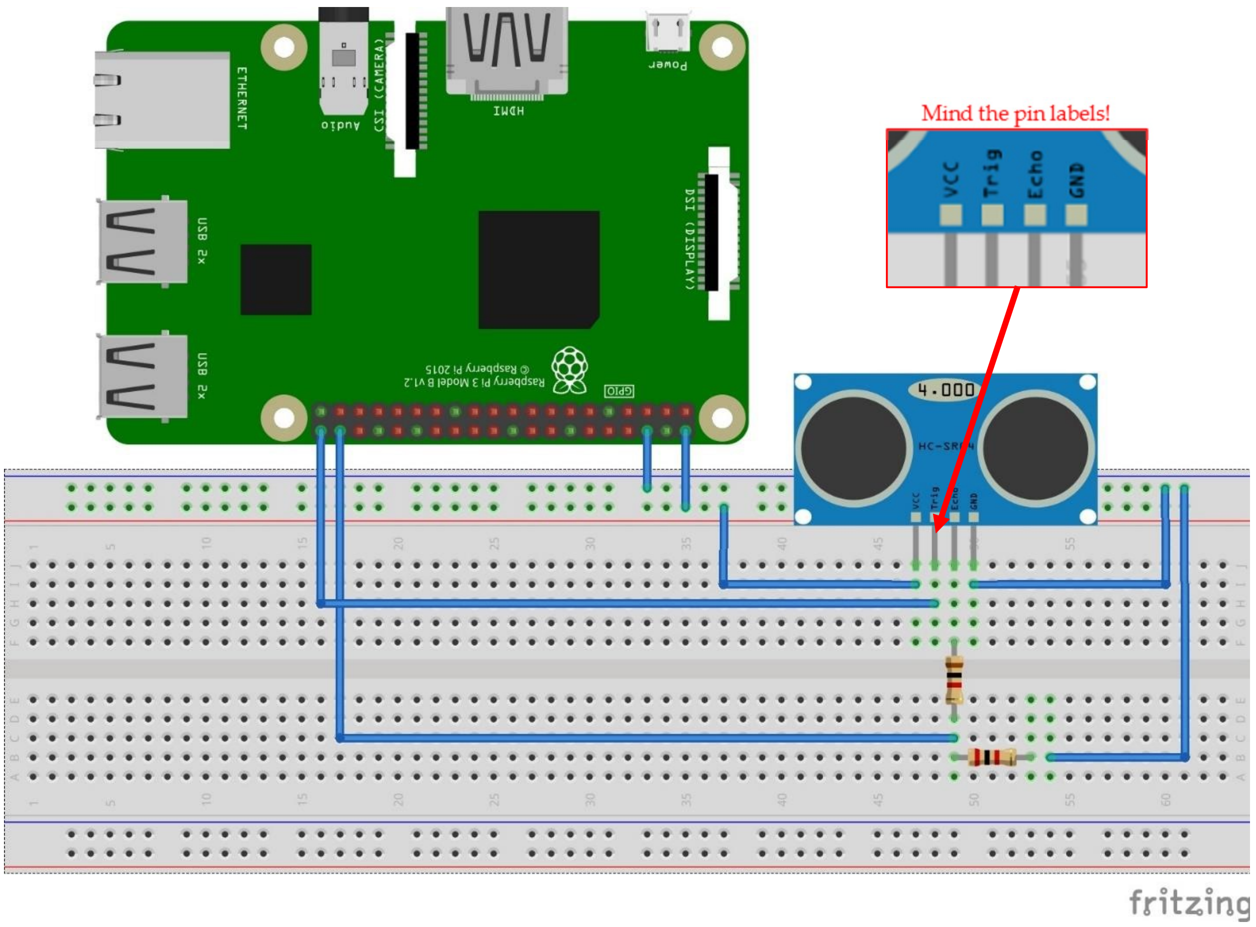
EXAMPLE CODE

Use the following code to experiment with the RGB LED. Change the brightness/intensity of each of the colour components (Red, Green and Blue) to obtain different colours.

```
import RPi.GPIO as GPIO
import time

#GPIO pin 2 to 27 are PWM compatible.
#We assign names to the pins for convenience
led_R = 11
led_G = 12
led_B = 13

GPIO.setmode(GPIO.BOARD)  #physical numbering scheme
GPIO.setup(led_R, GPIO.OUT) #set pin 11 as output, required for PWM
GPIO.setup(led_G, GPIO.OUT) #set pin 12 as output
GPIO.setup(led_B, GPIO.OUT) #set pin 13 as output

pwm_R = GPIO.PWM(led_R, 5000) #enable PWM, specify frequency of 5kHz
pwm_G = GPIO.PWM(led_G, 5000)
pwm_B = GPIO.PWM(led_B, 5000)

pwm_R.start(0) #start PWM pins at 0% duty cycle (off)
pwm_G.start(0)
pwm_B.start(0)

#MODIFY THESE VALUES
R = 0  #set the duty cycle (and colour). Values between 0 and 100
G = 0
B = 0

#apply the changes to the duty cycle.
pwm_R.ChangeDutyCycle(R)
pwm_G.ChangeDutyCycle(G)
pwm_B.ChangeDutyCycle(B)

time.sleep(10) #leave the light on for 10 seconds
GPIO.cleanup() #free up all pins
```

4. Display the Output of the Distance Sensor

Familiarize yourself with the operation of the distance sensor by displaying its output on the console. You can use the code provided in this section to get the readings. You will be able to reuse the code in the following section. Try to establish a range within which the sensor operates reliably.



Because the Rpi cannot handle a voltage of 5v we must divide the voltage. We use Resistors with the values of 1k and 2k in the divider.

Important Notes:

- The distance sensor requires 5V. There is a single pin on the RPi which provides this. Be sure not to short (connect) this pin with any other pins on the RPi.
- The signal output on the ECHO pin will be 5V. The RPi cannot accept voltages higher than 3.3V. **DO NOT DIRECTLY CONNECT THE ECHO PIN TO THE RPI**. We must make use of a voltage divider to take the 5V output down to 3.3V. **Be sure that you are using the correct resistor values when assembling the circuit!**

EXAMPLE CODE

The code below can be used to read values from the distance sensor. Once it is working, try varying the some of the values and note their impact. You can use this code as a base for the next task.

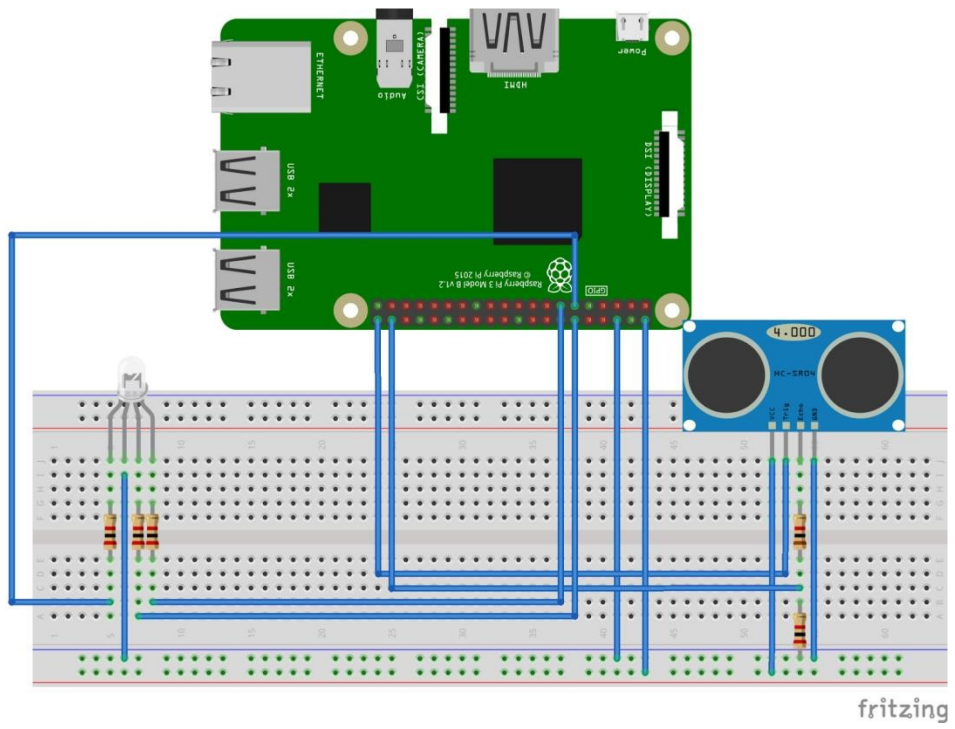
```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)     #physical numbering scheme
trig = 40
echo = 38
GPIO.setup(trig, GPIO.OUT)  #set pin 38 as output
GPIO.setup(echo, GPIO.IN)   #set pin 40 as input
GPIO.output(trig, False)    #start trig at 0
time.sleep(1)               # delay for setup

def distance():
    GPIO.output(trig, True)
    time.sleep(0.00001) #send a 10 microsecond pulse
    GPIO.output(trig, False)

    #keep resetting the start time until an echo is detected
    while GPIO.input(echo)==0:
        start = time.time()
    #keep updating the stop time until the falling edge of the echo pulse
    while GPIO.input(echo)==1:
        stop = time.time()

    #the constant comes from the speed of sound in cm/s, divided by 2
    distance = 17014.5*(stop-start)
    return distance     #in centimeters

scanON = True
while (scanON):
    dist = distance()   # function call
    time.sleep(0.5)
    if dist < 1000:     # print condition
        print(dist)

    if (dist < 2):      # exit condition
        scanON = False

GPIO.cleanup()
```

5. Mini-project: Change the colour of an RGB LED using the distance sensor

Combine what you have learned in the previous sections to represent the output of the distance sensor on the RGB LED. Choose a "middle" point at some distance from the sensor. If the sensor detects an object at the "middle" point, the LED will be off. The further the object moves from the "middle" point, the brighter the LED. Use different colours to tell the user whether they are closer than or further away than the "middle". You can this of this as a "Hot/Cold" indicator to help the user find the "middle" point. Use the circuit diagram and the code provided in the previous section to get started.

Hints:

- Be careful to keep the pin connections the same if you are reusing code
- Reduce the sleep time to improve the response of your device
- Get creative to eliminate outliers in the sensor data
- BONUS: you can turn this into a guessing game by generating a random "middle" point and asking the user to enter a guess. The program could tell the user how far off they are.


ETHERNET
Audio
CSI (CAMERA)
HDMI
Power
DSI (DISPLAY)
USB 2x
USB 2x
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
GPIO
4.000
HC-SR04
fritzing