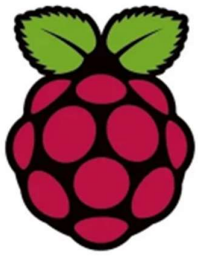
Led Breathing Using Raspberry Pi



Aditi Vaidya

vaidya19598@mail.npu.edu

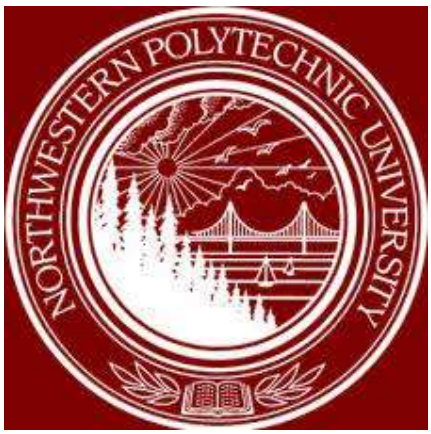


Table of Contents

1.0 Introduction

1.1 Objective

The main objective is to built the LED breathing circuit with Python programming using the Raspberry Pi 3 model B development board.

1.2 Requirement

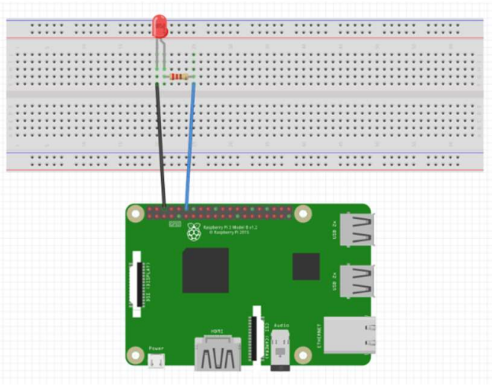
- ✓ 1 * Breadboard
- ✓ Jumper wires
- ✓ 1 * Raspberry Pi (I am using Raspberry Pi 3 Model B)
- ✓ 1 * 220Ω Resistor
- ✓ 1 * LED

1.3 Principle

The on-off pattern can simulate voltages in between full **on** (3.3 Volts) and **off** (0 Volts) by changing the portion of the time when the signal is on versus the time that the signal is off.

2.0 Hardware connections

Use the bread board and make connections as shown in visual representation below :



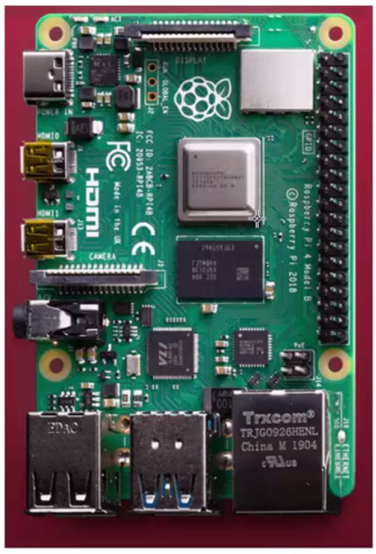
For calculating the resistor value , use ohms law :

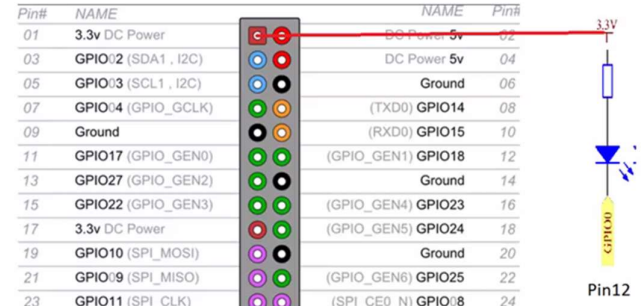
$$V_{in}=IR$$

'V_{in}' through GPIO is 3.3 V and current should not exceed the load of 16mA.Substituting values we get **R= 220Ω.**

3.0 Working

1. Once the wireless connection is established then open thonny Python IDE which mostly is pre installed in raspberry Pi's.





2. Make sure the Python library is uploaded and running .
3. Type in the program, and check for any errors.
4. Run the program .
5. Check for LED .

3.1 Programming

Enter the following program on Thonny Python IDE :

```
import RPi.GPIO as GPIO
import time

LedPin = 12

def setup():
        global   p
        GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
        GPIO.setup(LedPin, GPIO.OUT)    # Set LedPin's mode is output
        GPIO.output(LedPin, GPIO.LOW)   # Set LedPin to low(0V) to switch on LED

        p = GPIO.PWM(LedPin, 1000)      # set Frequece to 1KHz
        p.start(0)                      # Duty Cycle = 0%; DC(Length of Voltage Hi)
                                        # p.start(20)=20%

def loop():
        while True:
                for dc in range(0, 101, 4):         # Increase duty cycle: 0~100 step=4
                        p.ChangeDutyCycle(dc)       # Change duty cycle
                        time.sleep(0.05)            # time.sleep(sec)
                time.sleep(1)
                for dc in range(100, -1, -4):       # Decrease duty cycle: 100~0
                        p.ChangeDutyCycle(dc)
                        time.sleep(0.05)
                time.sleep(1)

def destroy():
        p.stop()
        GPIO.output(LedPin, GPIO.HIGH)     # turn off all leds
        GPIO.cleanup()

setup()
try:
        loop()
except   KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be  executed.
        destroy()
```

3.1 Debugging and Execution

Connect the hardware and run the application to execute the program.