Co-funded by the
Erasmus+ Programme
of the European Union

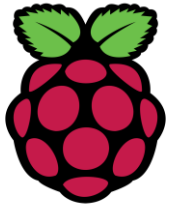
Al Hussein
Technical University
جامعة الحسين التقنية

Remote Lab For Engineering Project
Erasmus+ Program

Raspberry Pi -Take Home Lab

Lab Experiment # 5

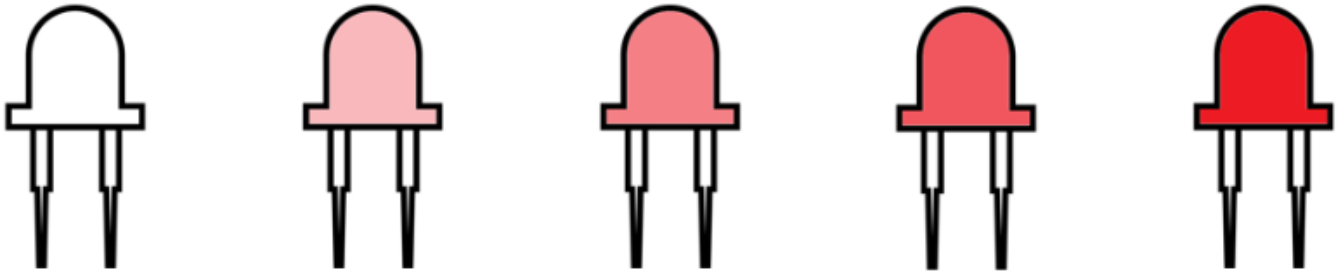
Analog Output Using Pulse Width Modulation(PWM)



Al Hussein Technical University جامعة الحسين التقنية
Co-funded by the Erasmus+ Programme of the European Union

PWM - Pulse Width Modulation

It is a commonly used control technique that generates analog signals from digital devices such as microcontrollers.
In PWM technique, the signal's energy is distributed through a series of pulses rather than a continuously varying (analog) signal.
It’s one of the primary means by which microcontrollers drive analog actuators such as variable-speed motors, dimmable lights, and speakers

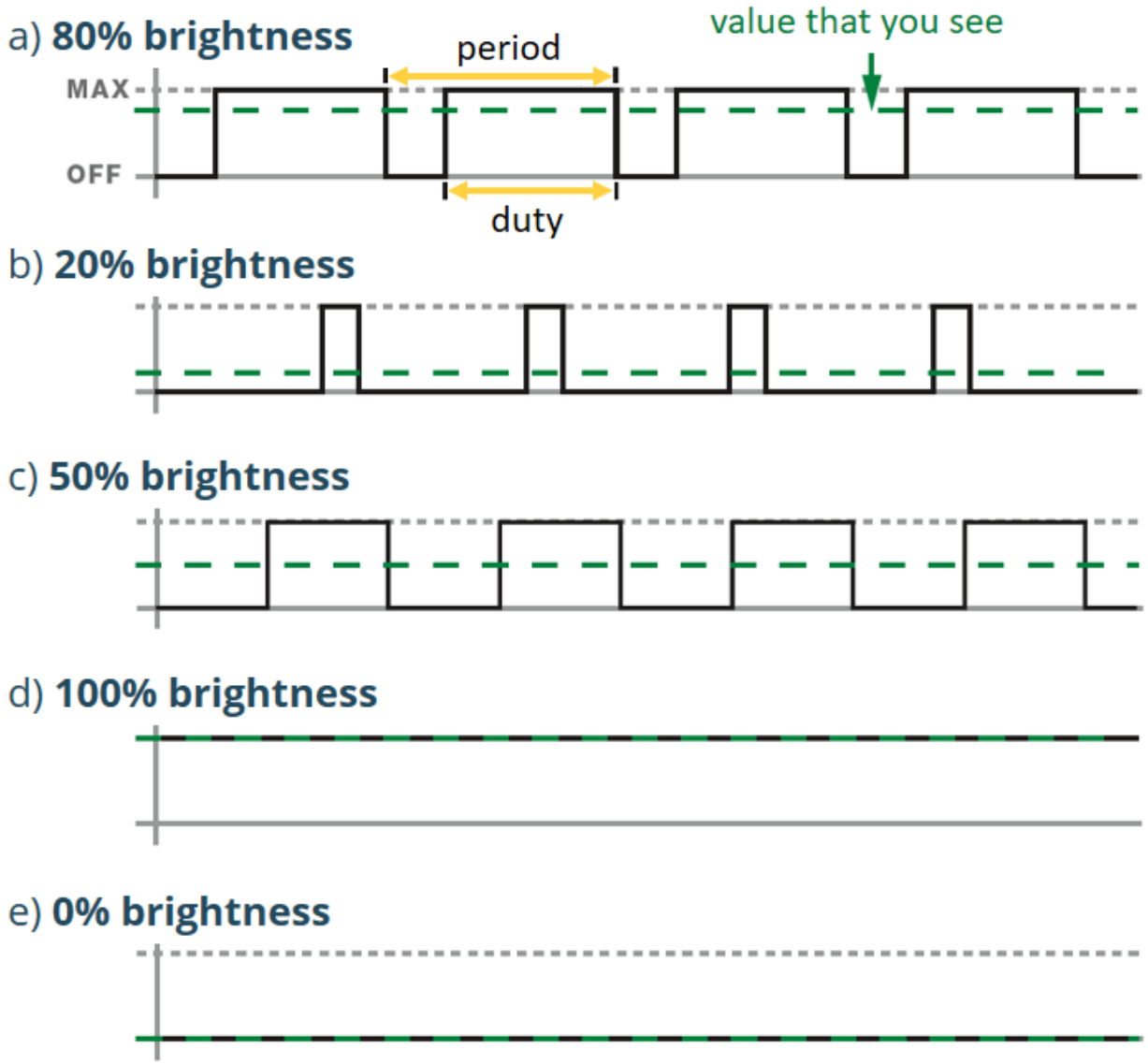


Changing the LED Brightness using PWM

Al Hussein Technical University جامعة الحسين التقنية
Co-funded by the Erasmus+ Programme of the European Union

PWM- LED Brightness

- The Raspberry Pi has 4 hardware PWM pins: GPIO 12, GPIO 13, GPIO 18, GPIO 19
- PWM works — by producing an output that changes between HIGH and LOW at a very high frequency.
- The duty cycle is the fraction of the time period at which LED is set to HIGH.
- A duty cycle of 50 percent results in 50 percent LED brightness, a duty cycle of 0 means the LED is fully off, and a duty cycle of 100 means the LED is fully on. Changing the duty cycle is how you produce different levels of brightness.

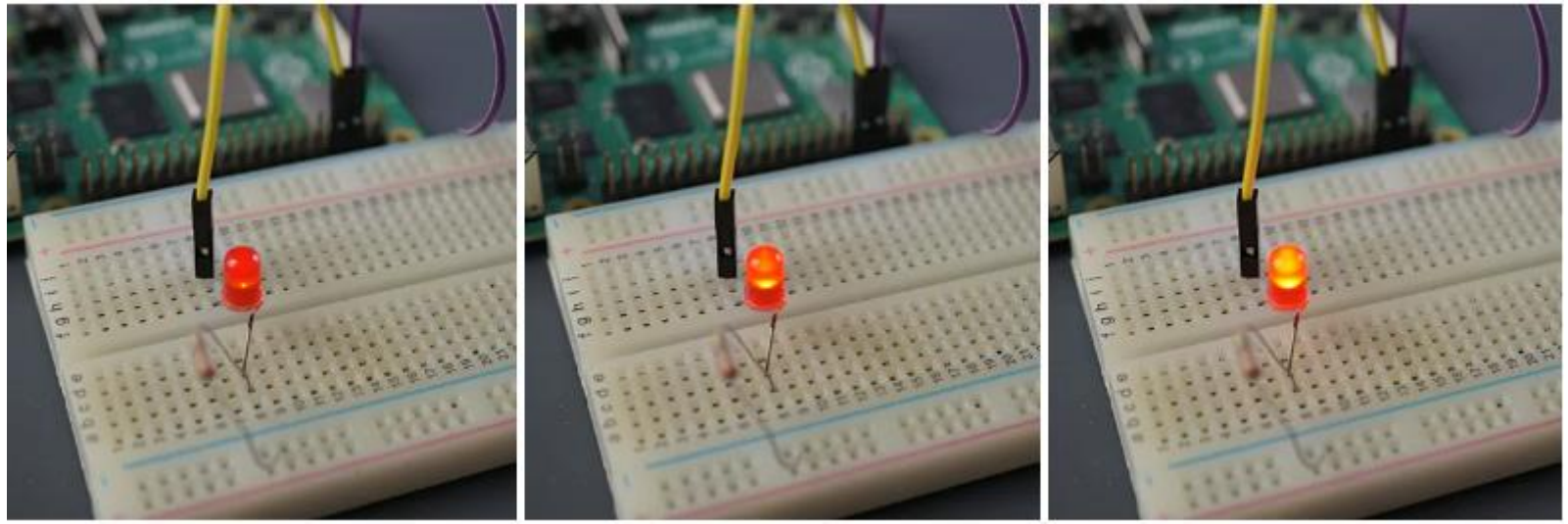


Al Hussein Technical University
جامعة الحسين التقنية

Co-funded by the Erasmus+ Programme of the European Union

LED With Analog Output

Build a circuit to change the LED brightness using one of the Raspberry Pi 's analog outputs.



Al Hussein Technical University جامعة الحسين التقنية

Co-funded by the Erasmus+ Programme of the European Union

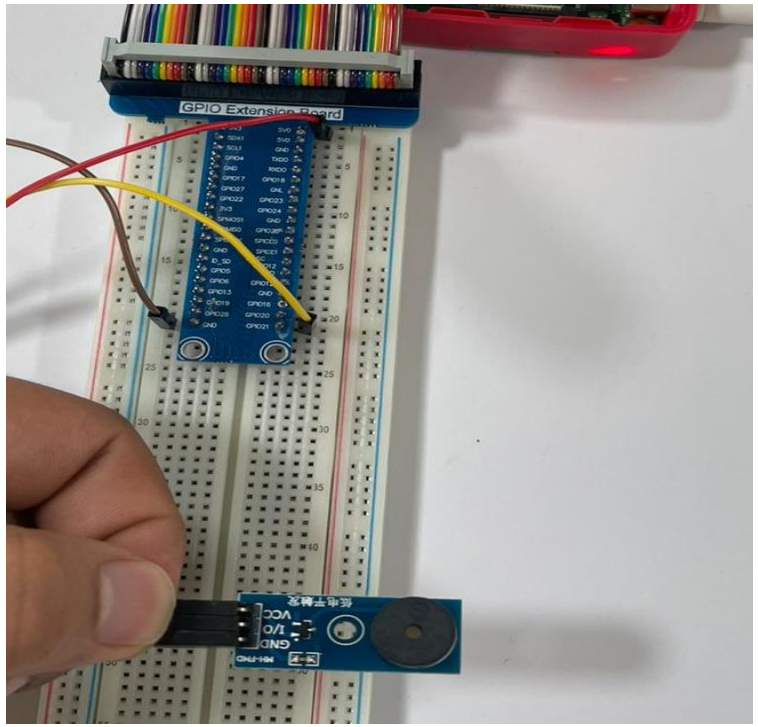
LED With Analog Output- Code

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

# Pin configuration
LED_PIN = 40  # Physical pin 40 (BOARD numbering)

# GPIO setup
GPIO.setmode(GPIO.BOARD)  # Use physical pin numbering
GPIO.setup(LED_PIN, GPIO.OUT)  # Set the pin as an output

# Set up PWM for the LED
pwm = GPIO.PWM(LED_PIN, 100)  # 100 Hz frequency
pwm.start(0)  # Start with 0% duty cycle (LED off)

try:
    while True:
        # Gradually increase brightness
        for duty_cycle in range(0, 101, 5):  # 0% to 100% in steps of 5
            pwm.ChangeDutyCycle(duty_cycle)
            time.sleep(0.5)  # Small delay for smooth transition

        # Gradually decrease brightness
        for duty_cycle in range(100, -1, -5):  # 100% to 0% in steps of 5
            pwm.ChangeDutyCycle(duty_cycle)
            time.sleep(0.5)

except KeyboardInterrupt:
    print("Exiting program.")
finally:
    pwm.stop()  # Stop PWM
    GPIO.cleanup()  # Reset GPIO settings
```

Al Hussein Technical University جامعة الحسين التقنية
Co-funded by the Erasmus+ Programme of the European Union

Buzzer With Analog Output

Build a circuit to change the Buzzer tone level using one of the Raspberry Pi 's analog outputs.



Al Hussein Technical University جامعة الحسين التقنية

Co-funded by the Erasmus+ Programme of the European Union

Buzzer With Analog Output- Code

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

# Pin configuration
BUZZER_PIN = 40  # Physical pin 40 (BOARD numbering)

# GPIO setup
GPIO.setmode(GPIO.BOARD)  # Use physical pin numbering
GPIO.setup(BUZZER_PIN, GPIO.OUT)  # Set the pin as an output

# Set up PWM for the buzzer
pwm = GPIO.PWM(BUZZER_PIN, 1000)  # Start with 1000 Hz frequency
pwm.start(0)  # Start with 0% duty cycle (buzzer off)

try:
    while True:
        # Generate tones with varying frequencies
        for freq in [262, 330, 392, 494]:  # C4, E4, G4, B4 (notes)
            pwm.ChangeFrequency(freq)  # Change to the desired frequency
            pwm.ChangeDutyCycle(50)  # 50% duty cycle for a consistent tone
            print(f"Playing tone at {freq} Hz")
            time.sleep(1)  # Play each tone for 1 second
        pwm.ChangeDutyCycle(0)  # Turn off the buzzer for a pause
        time.sleep(1)
except KeyboardInterrupt:
    print("Exiting program.")
finally:
    pwm.stop()  # Stop PWM
    GPIO.cleanup()  # Reset GPIO settings
```

Al Hussein Technical University
جامعة الحسين التقنية

Co-funded by the Erasmus+ Programme of the European Union

Take-Home Task

Build a circuit that reads the intensity of light using LDR senor then change the buzzer tone level as the following:

Completely dark: high tone level

Dark: medium tone level

Normal Light: buzzer oFF

Flashlight: low tone level