|  |
| --- |
|  |

**Aim:**

**Familiarization with Pulse Width Modulation (PWM) using various sensors and actuators interfacing with Raspberry Pi Pico and LEDs.**

**.Objectives:**

1. Introduction to **Pulse Width Modulation (PWM)** Technique, its behaviour and applications.
2. Implementation of PWM on Micro-python with Raspberry Pi Pico: **LED Breathing** using **Duty Cycle and Brightness Modification** of **inbuilt LED** of Raspberry Pi Pico.
3. Implementation of PWM on Micro-python with Raspberry Pi Pico: **LED Breathing** using **Duty Cycle and Brightness Modification** of **external LED**.
4. Implementation of **LED Breathing: Complete the challenge** with Minimum, Intermediate and Maximum Brightness Level.
5. Implementation of **Colorful Light: Alternately flashing red, green, and blue** colour without PWM technique and **Colorful Light: Mystery of additive color mixing** with PWM technique using Common Cathode type RGB LED and Raspberry Pi Pico.
6. Implementation of **Custom Tone: Create a Melody** using active buzzer, transistor and Raspberry Pi Pico.
7. Implementation of a **Custom Tone: Classic Happy Birthday Melody** using active buzzer, transistor and Raspberry Pi Pico.

**Summary of Experiment - 6 Goals and Outcomes**

By the end of this experiment, students will have gained a solid understanding of IoT fundamentals, Raspberry Pi Pico, Python programming, and practical experience in building and controlling IoT-related hardware. These skills will serve as a strong foundation for more complex IoT projects in the future.

**Pre-Lab Questionnaire:**

1. What is the maximum frequency that can be achieved with the PWM module on the Raspberry Pi Pico?
2. How many PWM channels are available on the Raspberry Pi Pico?
3. How can you configure the PWM frequency on the Raspberry Pi Pico using MicroPython?
4. How can you set the duty cycle of a PWM channel on the Raspberry Pi Pico using MicroPython?
5. What is the range of the duty cycle on the Raspberry Pi Pico?
6. How can you enable or disable a PWM channel on the Raspberry Pi Pico using MicroPython?
7. How can you change the polarity of the PWM output on the Raspberry Pi Pico using MicroPython?
8. How can you create a software PWM on the Raspberry Pi Pico using MicroPython?
9. How can you synchronize multiple PWM channels on the Raspberry Pi Pico using MicroPython?
10. How can you measure the frequency and duty cycle of an input PWM signal on the Raspberry Pi Pico using MicroPython?

**Answers to Pre-Lab Questions**

**Components/Equipment Required:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Name of the**  **Component / Equipment** | **Specification** | **Quantity** |
|  | Raspberry Pi Pico | RP2040 microcontroller chip, 125MHz | 1 |
|  | Raspberry Pi Pico cable | USB Type A to Micro-B | 1 |
|  | Resistors (carbon type) | ¼ watt (330 Ω) | 16 |
|  | LED | 3mm, Red | 16 |
|  | RGB LED | Common Cathode (CC) | 1 |
|  | Buzzer | Active | 1 |
|  | Transistor | NPN (2N2222) | 1 |
|  | Breadboard | 840 Tie points | 1 |
|  | Jumper Wire | ----------------------- | As per requirement |

**Objective 1**

## Pulse Width Modulation (PWM) is a technique used in electronics to control the average power delivered to a load by modulating the width of the pulse signal. This modulation is achieved by varying the duty cycle of the signal, which is the ratio of the on-time to the total period of the signal.

In a PWM signal, a periodic square wave, often referred to as the carrier signal, is used. The key characteristic is the duty cycle, which determines the percentage of time the signal is in the "on" state compared to the total period of the signal. A high duty cycle means the signal is on for a large portion of the time, while a low duty cycle indicates a smaller on-time.

## Frequency: The frequency of the PWM signal refers to how often the pulse repeats within a given time frame. It is measured in Hertz (Hz) and determines how quickly the system responds to changes in the control signal.

Duty Cycle: The duty cycle, usually expressed as a percentage, defines the proportion of time the signal is in the "on" state. It is a crucial parameter, affecting the average power delivered to the load. A 50% duty cycle means the signal is on for half the time, resulting in an average power of 50%.

## Applications of PWM are :

* Motor Speed Control - Analog Signal Generation,

Audio Signal Generation - Temperature Control

**Objective 2**

**Inbuilt LED Blinking with Raspberry Pi Pico and PWM:**

Implementation of PWM on Micro-python with Raspberry Pi: LED Breathing using Duty Cycle and Brightness Modification of inbuilt LED**.**

**Code**

**2.1: write a program that implements PWM on Micro-python with Raspberry Pi Pico: LED Breathing using Duty Cycle and Brightness Modification of inbuilt LED.**

**Observation**

**Figure 1: (Simulation-based LED Breathing using Duty Cycle and Brightness Modification of inbuilt LED using PWM on Micro-python with Raspberry Pi Pico)**

**Figure 2: (Hardware Implementation based LED Breathing using Duty Cycle and Brightness Modification of inbuilt LED using PWM on Micro-python with Raspberry Pi Pico)**

**Objective 3**

**External LED Blinking with Raspberry Pi Pico and PWM:**

Implementation of PWM on Micro-python with Raspberry Pi: LED Breathing using Duty Cycle and Brightness Modification of external LED**.**

***Circuit/Schematic Diagram:***



(Figure 3: Circuit diagram for implementation of PWM on Micro-python with Raspberry Pi: LED Breathing using Duty Cycle and Brightness Modification of external LED.)

**Code**

**3.1: write a program that implements PWM on Micro-python with Raspberry Pi Pico: LED Breathing using Duty Cycle and Brightness Modification of external LED.**

**Observation**

**Figure 4: (Simulation-based LED Breathing using Duty Cycle and Brightness Modification of external LED using PWM on Micro-python with Raspberry Pi Pico)**

**Figure 5: (Hardware Implementation based LED Breathing using Duty Cycle and Brightness Modification of external LED using PWM on Micro-python with Raspberry Pi Pico)**

**Objective 4**

**Implementation of LED Breathing: Complete the challenge with Minimum, Intermediate and Maximum Brightness Level.**

**Circuit / Schematic Diagram**



(Figure 6: Circuit diagram for implementation of LED Breathing: Complete the challenge with Minimum, Intermediate and Maximum Brightness Level.)

**Code**

**4.1: write a program to complete the challenge of LED breathing with Minimum, Intermediate and Maximum Brightness Level.**

**.**

**Observation**

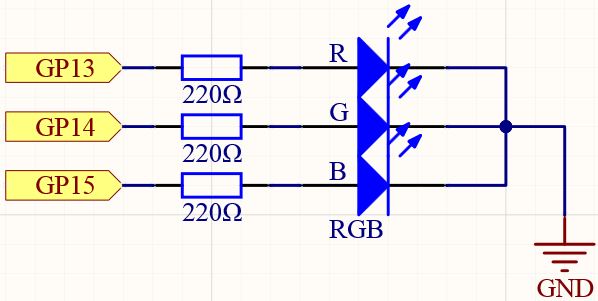
**Figure 7: (Simulation-based LED Breathing: Complete the challenge with Minimum, Intermediate and Maximum Brightness Level)**

**Figure 8: (Hardware Implementation based LED Breathing: Complete the challenge with Minimum, Intermediate and Maximum Brightness Level)**

**Objective 5**

**Implementation of Colorful Light:**

**Alternately flashing red, green, and blue** colour without PWM technique and **Colorful Light: Mystery of additive color mixing** with PWM technique using Common Cathode type RGB LED and Raspberry Pi.



(Figure 9: Circuit diagram for implementation of Colorful Light: Alternately flashing red, green, and blue colour without PWM technique and Colorful Light: Mystery of additive color mixing with PWM technique using Common Cathode type RGB LED and Raspberry Pi.)

**Code**

**5.1: write a program to implement colourful light alternately flashing red, green, and blue colour without PWM technique and Colorful Light: Mystery of additive color mixing with PWM technique using Common Cathode type RGB LED and Raspberry Pi.**

**Observation**

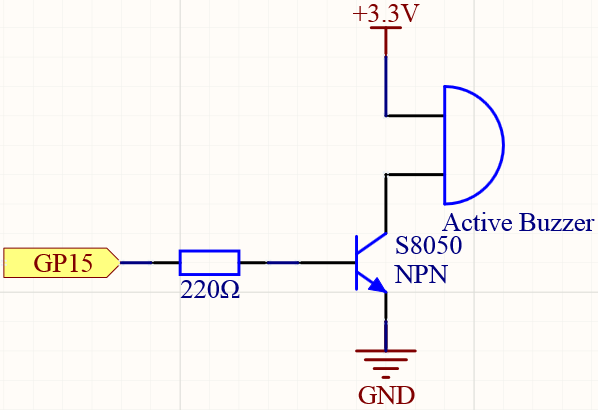
**Figure 10: (Simulation-based Colorful Light: Alternately flashing red, green, and blue and Mystery of additive color mixing using PWM on Micro-python with Raspberry Pi Pico)**

**Figure 11: (Hardware Implementation of Colorful Light: Alternately flashing red, green, and blue and Mystery of additive color mixing using PWM on Micro-python with Raspberry Pi Pico)**

**Objective 6**

**Implementation of Custom Tone:**

**Implementation of Custom Tone: Create a Melody using active buzzer, transistor and PWM on Micro-python with Raspberry Pi Pico.**



(Figure 12: Circuit diagram for implementation of Custom Tone: Create a Melody using active buzzer, transistor and Raspberry Pi Pico.)

**Code**

**6.1: write a program for implementation of Custom Tone: Create a Melody using active buzzer, transistor and PWM on Micro-python with Raspberry Pi Pico.**

**Observation**

**Figure 13: (Simulation-based electronic circuit for implementation of Custom Tone: Create a Melody using active buzzer, transistor and Raspberry Pi Pico)**

**Figure 14: (Hardware implementation based electronic circuit for implementation of Custom Tone: Create a Melody using active buzzer, transistor and Raspberry Pi Pico)**

**Objective 7**

**Implementation of Custom Tone:**

**Implementation of Custom Tone: Classic Happy Birthday Melody using active buzzer, transistor and PWM on Micro-python with Raspberry Pi Pico.**



**(Figure 15: Circuit diagram for implementation of Custom Tone: Classic Happy Birthday Melody using active buzzer, transistor and PWM on Micro-python with Raspberry Pi Pico.**

**Code**

**7.1: write a program for implementation of Custom Tone: Classic Happy Birthday Melody using active buzzer, transistor and PWM on Micro-python with Raspberry Pi Pico.**

**Observation**

**Figure 16: (Simulation-based electronic circuit for implementation of Custom Tone: Classic Happy Birthday Melody using active buzzer, transistor and Raspberry Pi Pico)**

**Figure 17: (Hardware implementation based electronic circuit for implementation of Custom Tone: Classic Happy Birthday Melody using active buzzer, transistor and Raspberry Pi Pico)**

**Conclusion:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Precautions:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Post Experiment Questionnaire:**

**Answer all the Questions in brief with some appropriate examples.**

1. Write a Micro-Python code which turns on the buzzer, connected to GPIO16 of the Raspberry Pi Pico by PWM signal having 200 Hz frequency and 20% duty cycle by pressing the push button connected to GPIO21.
2. A PWM signal with a frequency of 4kHz is to be generated on the Raspberry Pi Pico using MicroPython. What value should be passed to the freq() function to set this frequency?
3. A PWM signal with a duty cycle of 75% is to be generated on the Raspberry Pi Pico using MicroPython. What value should be passed to the duty() function to set this duty cycle?
4. A PWM signal with a frequency of 2kHz is currently being generated on a specific channel of the Raspberry Pi Pico using MicroPython. The frequency needs to be changed to 1kHz. What value should be passed to the freq() function to set this new frequency?
5. A PWM signal with a duty cycle of 50% is currently being generated on a specific channel of the Raspberry Pi Pico using MicroPython. The duty cycle needs to be changed to 25%. What value should be passed to the duty() function to set this new duty cycle?
6. A software PWM signal with a frequency of 1kHz and a duty cycle of 50% is to be generated on a specific digital output pin of the Raspberry Pi Pico using MicroPython. How many times per second should the digital output pin be toggled to achieve this?
7. An input PWM signal with a frequency of 3kHz and a duty cycle of 33% is being received on a specific digital input pin of the Raspberry Pi Pico using MicroPython. What is the width of the input pulse in microseconds?
8. A PWM signal with a frequency of 2MHz is to be generated on the Raspberry Pi Pico using MicroPython. What value should be passed to the freq() function to set this frequency?
9. A PWM signal with a duty cycle of 90% is to be generated on the Raspberry Pi Pico using MicroPython. What value should be passed to the duty() function to set this duty cycle?
10. A PWM signal with a frequency of 1kHz is currently being generated on a specific channel of the Raspberry Pi Pico using MicroPython. The frequency needs to be changed to 500Hz. What value should be passed to the freq() function to set this new frequency?
11. A PWM signal with a duty cycle of 20% is currently being generated on a specific channel of the Raspberry Pi Pico using MicroPython. The duty cycle needs to be changed to 10%. What value should be passed to the duty() function to set this new duty cycle?
12. A software PWM signal with a frequency of 5kHz and a duty cycle of 25% is to be generated on a specific digital output pin of the Raspberry Pi Pico using MicroPython. How many times per second should the digital output pin be toggled to achieve this?
13. An input PWM signal with a frequency of 1kHz and a duty cycle of 50% is being received on a specific digital input pin of the Raspberry Pi Pico using MicroPython. What is the width of the input pulse in microseconds?

**Answers to Post-Lab Questions**

|  |  |  |  |
| --- | --- | --- | --- |
| **(Signature of the Faculty)** | |  | **(Signature of the Student)** |
|  | | **Name:** |  |
| **Date:** |  | **Registration No.:** |  |
|  |  | **Branch:** |  |
|  | | **Section** |  |