VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# School of Electronics Engineering (SENSE)

## B. Tech – Electronics & Communication Engineering

# BECE403E – EMBEDDED SYSTEMS DESIGN

# LAB RECORD

# (lab slot L27+L28)

## Submitted By


**22BEC1205 – Jayakrishnan Menon**


## Submitted To

**Dr. S.Muthulakshmi**

**DATE: 13/01/2025**

**Slot:** L27+L28

**Date: 13/01/2025**

# LAB – 03: Working with PWM

**AIM:** To understand the PWM feature of Nucleo64-STM32L152RE Board for applications like controlling the brightness of an LED as well as producing various tones from a buzzer, and also to perform the following tasks:

Lab Task-1: Write a C++ program with mbed APIs to control the brightness of the LED using a PWM signal with duty cycle 75%. Assume PWM period as 2 Seconds. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Lab Task-2: Write a C++ program with mbed APIs to gradually increase and decrease LED brightness using PWM signal with variable duty cycle. Assume the LED is connected to PC_8 PWM pin, time period of the PWM signal as 10ms, increment/decrement duty cycle value is 10% and delay between each increment/decrement duty cycle is 0.5s.

Lab Task-3: Write a C++ code with mbed APIs to generate a music note SA-RE-GA-MA-PA-THA-NEE on the buzzer using PWM signal.

Lab Task-4: Write a C++ program with mbed APIs to design a warning signals system for an Automotive

**SOFTWARE REQUIRED:** ARM Keil Studio (Mbed Online Compiler)

**HARDWARE REQUIRED:** Micro USB cable, NUCLEO64-STM32L152 Board, LED, Piezo Buzzer, Jumper Wires (M-F and M-M), Breadboard

**PROCEDURE:**

1. Go to ARM Keil Studio (https://studio.keil.arm.com) and log in
2. Select File → New → Mbed Project
3. Click the Example project drop-down list and select “mbed2-example-blinky”
4. In Project name field, provide the name of the new project and click Add project
5. Double click on the “main.cpp” file from the newly created project folder
6. Modify the code in the editor window as per the logic of your application
7. Check for any errors in the program under the “Problems” tab of the panels window
8. If no errors, connect the Nucleo Board to the computer using Micro USB Cable
9. Click Play icon (Run project) to upload and start the code execution on the board.

## PROGRAMS:

**Lab Task 1:** Write a C++ program with mbed APIs to control the brightness of the LED using a PWM signal with duty cycle 75%. Assume PWM period as 2 Seconds. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

**Code:**

```
#include "mbed.h"
PwmOut led(PC_8);
int main(){
    while(1){
        led.period(2.0f);
        led.write(0.750f);
    }
}
```

**Output:**

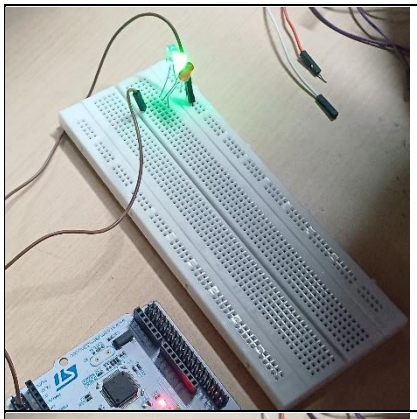


**Fig. 1.1:** LED connected to the PWM pin is in the "ON" state for 1.5 seconds (Duty Cycle of 75% with the time period of 2 seconds)

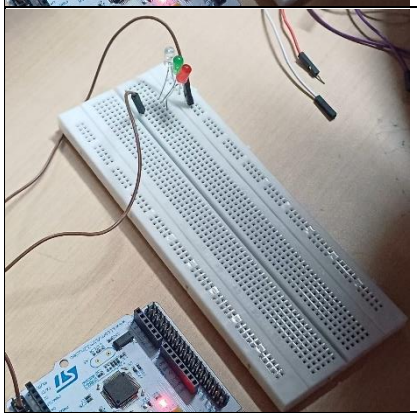


**Fig. 1.2:** LED connected to the PWM pin is in the "OFF" state for 0.5 seconds

**Lab Task 2:** Write a C++ program with mbed APIs to gradually increase and decrease LED brightness using PWM signal with variable duty cycle. Assume the LED is connected to PC_8 PWM pin, time period of the PWM signal as 10ms, increment/decrement duty cycle value is 10% and delay between each increment/decrement duty cycle is 0.5s. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

**Code:**

```
#include "mbed.h"
PwmOut led(PC_8);
int main(){
    led.period_ms(10);
    led=0.0f;
    while(1){
        for(float val=0.0f; val<1.0f; val+=0.1f){
            led=val;
            wait(0.5f);
        }
        for(float val=0.0f; val>1.0f; val-=0.1f){
            led=val;
            wait(0.5f);
        }
    }
}
```

**Output:**

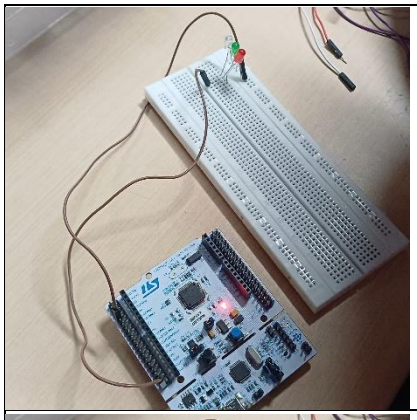


**Fig. 2.1:** LED connected to the PWM pin is not glowing, indicating a Duty Cycle of around 0% (or else, the RMS voltage value of the PWM signal is below the threshold voltage of the diode). From here, the LED starts to glow, progressively increasing in brightness

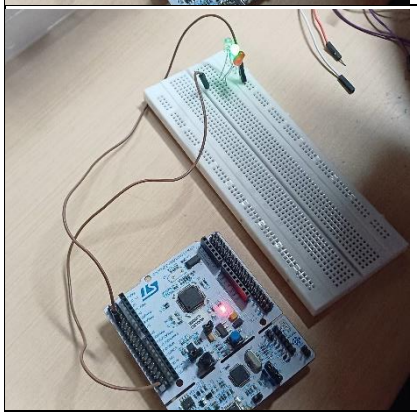
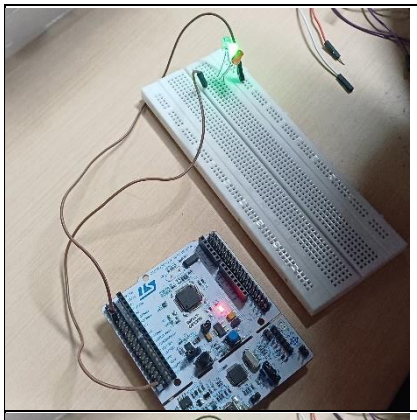
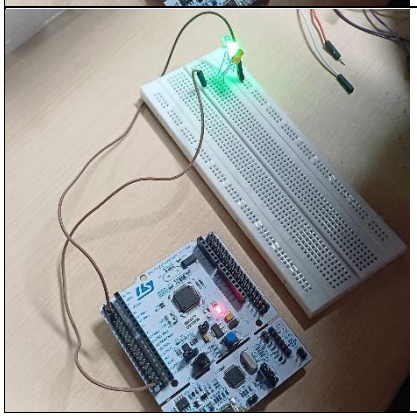


**Fig. 2.2:** LED connected to the PWM pin is glowing more bright than previously observed, indicating a higher PWM duty cycle than the previously observed output

| | |
|---|---|
|  | **Fig. 2.3:** LED connected to the PWM pin is glowing more bright than previously observed, indicating a higher PWM duty cycle than the previously observed output |
|  | **Fig. 2.4:** LED connected to the PWM pin is glowing at maximum brightness indicating a Duty Cycle of almost 100%. After this, the LED starts to dim from 100% Duty Cycle to 0% and so on, indefinitely. |

**Lab Task 3:** Write a C++ code with mbed APIs to generate a music note SA-RE-GA-MA-PA-THA-NEE on the buzzer using PWM signal. Frequency for each music tone is listed below. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

| Notes | Frequency in Hz |
|---|---|
| Sa | 240 |
| Re | 270 |
| Ga | 300 |
| Ma | 337.5 |
| Pa | 360 |
| Dha | 400 |
| Ni | 450 |

**Code:**

```
#include "mbed.h"
PwmOut speaker(PC_8);
int main(){
    while(true){
    speaker.period(1.0/240.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/270.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/300.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/337.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/360.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/400.0);
    speaker=0.5;
    wait(0.5);
    speaker.period(1.0/450.0);
    speaker=0.5;
    wait(0.5);
    speaker=0.0;
    }
}
```

**Output:**

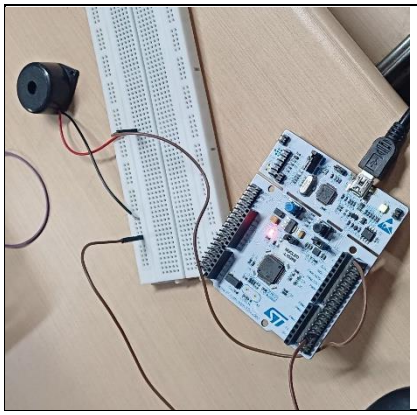


**Fig. 3.1:** Once the code is uploaded, the Buzzer starts to play the 7 distinct musical tones/notes sequentially, one after the other.

**Lab Task 4 (Challenging Task):** Write a C++ program with mbed APIs to design a warning signals system for Automotive with the following logic.

1. Use 4 switch and one buzzer to generate four warning signals such as indicator signal, horn sound, seat belt warning and reversing signal.
2. Assume each of these warning signals must have 50% duty cycle and 0.5s delay between their ON and OFF state.
3. Activate the respective warning signal as per the following switch status,
    - If Switch-1 is HIGH, generate a indicator signal (2 Hz)
    - If Switch-2 is HIGH, generate a Horn signal (400 Hz)
    - If Switch-3 is HIGH, generate a Seat belt warning signal (612 Hz)
    - If Switch-4 is HIGH, generate a Reversing signal (1000 Hz)
    - If all switches are LOW, the buzzer should be in OFF state

Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

**Code:**

```
#include "mbed.h"
BusIn myswitch(PC_4,PB_13,PB_14,PB_15);
PwmOut speaker(PC_8);
int main() {
  while(1) {
      if(myswitch==1){
        speaker.period(1.0/2.0);
        speaker=0.5;
        wait(0.5);
      }
      else if (myswitch==2) {
        speaker.period(1.0/400.0);
        speaker=0.5;
        wait(0.5);
      }
      else if (myswitch==4) {
        speaker.period(1.0/612.0);
        speaker=0.5;
        wait(0.5);
      }
      else if (myswitch==8) {
        speaker.period(1.0/1000.0);
        speaker=0.5;
        wait(0.5);
      }
      else{
        speaker.period(1.0/240.0);
        speaker=0;
        wait(0.5);
      }
  }
}
```

**Output:**

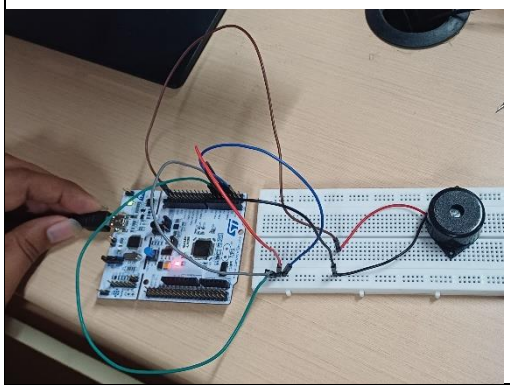
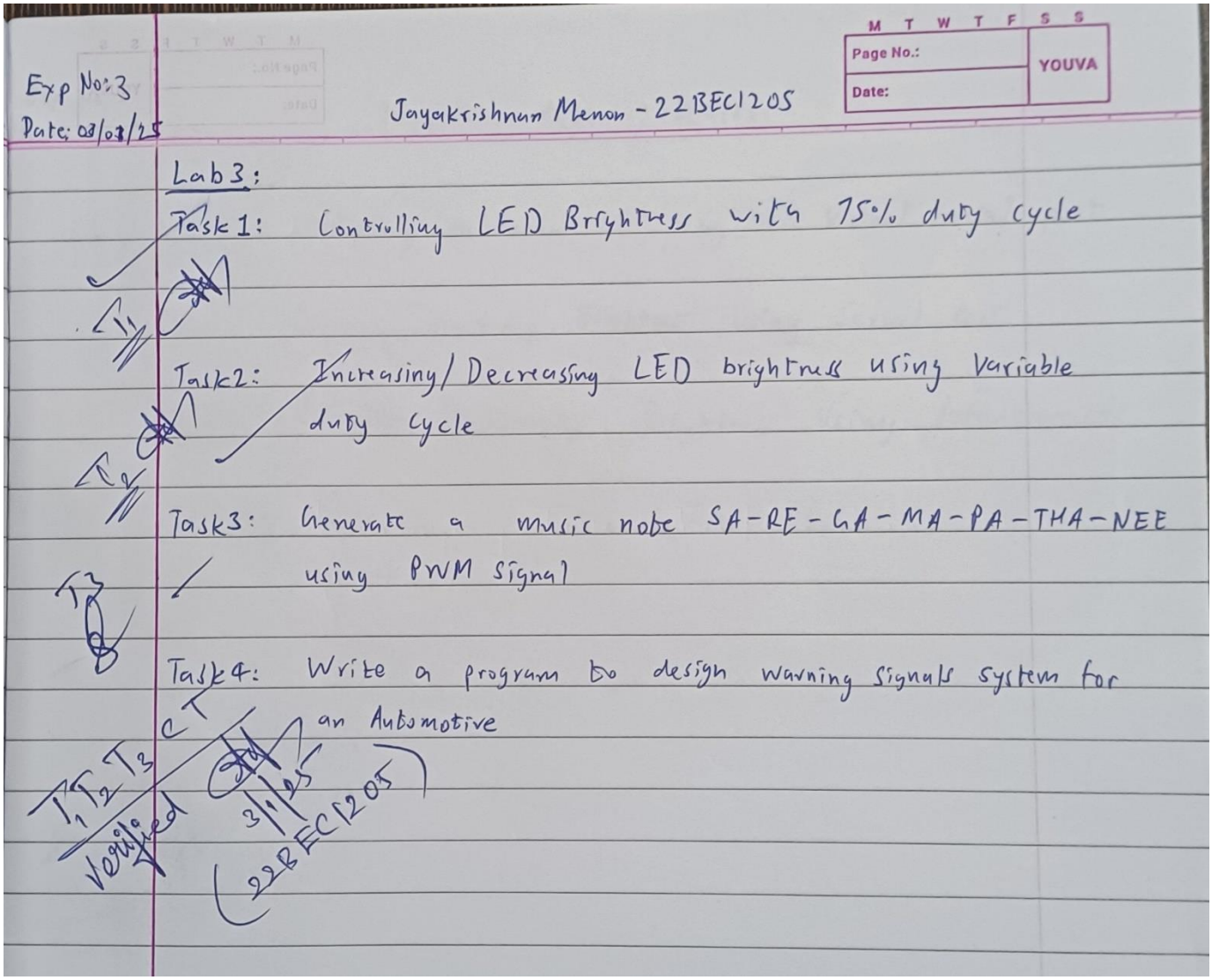


**Fig. 4.1:** When the Switches are made to be in the "HIGH" state one by one, the Buzzer starts to play one of the 4 distinct warning signals. The switches were constructed using Green, Gray, Red, and Blue wires, with only one wire connected to "HIGH" voltage at any given time.

**OUTPUT VERIFICATION:**




**INFERENCE:**

1. The PwmOut API in mbed allows you to generate Pulse Width Modulated (PWM) signals on pins that support PWM functionality. This is particularly useful for controlling devices such as LEDs, Buzzers, and servos. Its Syntax is: “PwmOut Identifier(PinName)”.
2. The PwmOut API provides two methods, period and period_ms, to configure the period of the Pulse Width Modulation (PWM) signal. These methods are critical for setting the frequency of the PWM signal.
3. The write method in the PwmOut API sets the duty cycle of the PWM signal. The duty cycle determines the proportion of the PWM period for which the signal is HIGH (ON) versus LOW (OFF). This is essential for controlling the intensity, speed, or position of devices like LEDs, motors, and servos.

4. The BusIn API in mbed provides an efficient way to group multiple DigitalIn pins together and read their states as a single value. This simplifies working with multiple digital input pins, such as switches or sensors. Its Syntax is: “BusIn Identifier(PinNames)”.
5. The “identifier” of a “BusIn” object can be used to read the combined state of all the pins in the bus. Each bit in the returned value corresponds to the state of a specific pin in the bus, starting from the least significant bit (LSB).
6. In ARM Keil IDE, the f suffix is often required when working with floating-point literals to explicitly denote them as single-precision (float) values instead of double-precision (double). Appending an f to the literal explicitly marks it as a single-precision float. This avoids implicit conversions and ensures compatibility with float variables and functions.
7. It was observed that by merely changing the period and duty cycle of the PWM signal, the audio output of a piezo buzzer could be controlled, in a similar way to how the brightness of an LED was controlled by modifying the PWM signal.
8. Programmable Delays can be implemented through the use of the “wait()” function.
9. A program can be set to run indefinitely using the “while(1)” loop

**RESULT:**

Thus, the usage of the PWM feature of Nucleo64-STM32L152RE Board for applications like controlling the brightness of an LED as well as producing various tones from a buzzer, were understood and the tasks were performed successfully.