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: 16/01/2025

Slot: L27+L28

Date: 16/01/2025

LAB – 04: Working with Serial Port and ADC

AIM: To understand the Serial Ports and Analog to Digital Converters of Nucleo64-STM32L152RE Board for applications like controlling the state of an LED and varying its brightness as well as to read Analog voltage values through ADC pins. We will also perform the following tasks:

Lab Task-1: Write a C++ code with mbed APIs to receive a character (H) from PC via host terminal application (Tera term) & switch ON the on-board LED1. For all other character LED1 must be in OFF state. 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 on-board LED brightness by receiving a character ("i" for increase and "d" for decrease) from PC via host terminal application (Tera term) using PWM signal with variable duty cycle. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Lab Task-3: Write a C++ program with mbed APIs to gradually increase and decrease LED brightness by changing the position of the Potentiometer via ADC. Also print it's voltage value on serial monitor. Use Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Lab Task-4: Write a C++ program with mbed APIs to design a battery level indicator system using potentiometer and LEDs.

SOFTWARE REQUIRED: ARM Keil Studio (Mbed Online Compiler), Tera Term

HARDWARE REQUIRED: Micro USB cable, NUCLEO64-STM32L152 Board, LEDs, Potentiometer, 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.

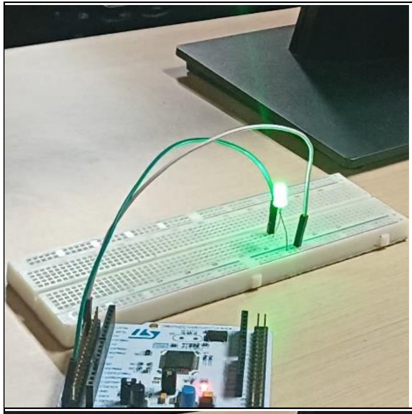
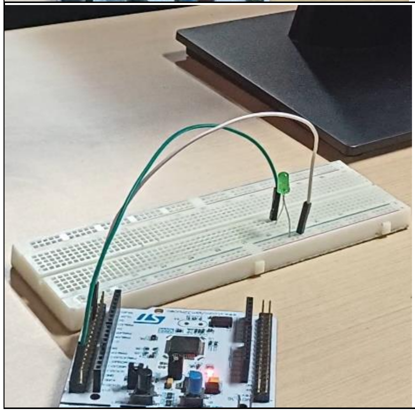
PROGRAMS:

Lab Task 1: Write a C++ code with mbed APIs to receive a character (H) from PC via host terminal application (Tera term) & switch ON the on-board LED1. For all other character LED1 must be in OFF state. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Code:

```
#include "mbed.h"
Serial pc(USBTX,USBRX);
DigitalOut myled(PC_8);
int main(){
    printf("Press a character");
    while(1){
        int y=pc.getc();
        printf("%c\n",y);
        if(y=='H'){
            myled=1;
            wait(0.2);
        }
        else{
            myled=0;
        }
    }
}
```

Output:

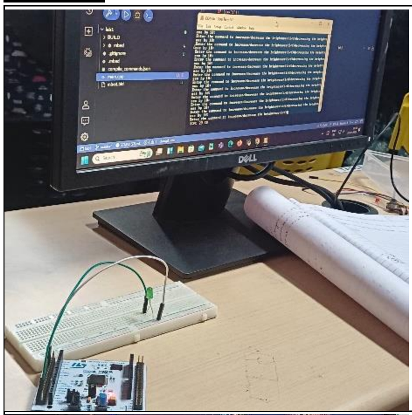
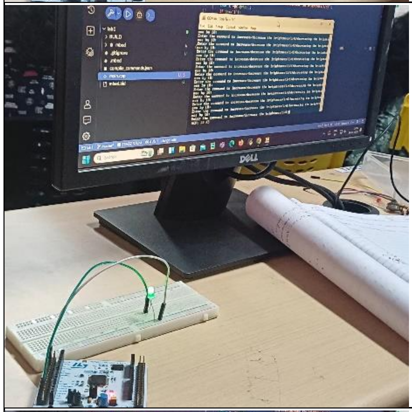
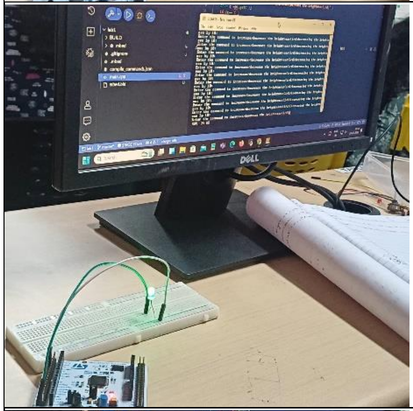
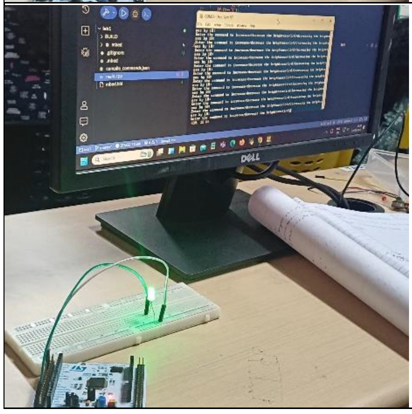
	**Fig. 1.1:** LED connected to PC_8 turns "ON" when the character "H" is entered on the serial terminal
	**Fig. 1.2:** LED connected to PC_8 turns "OFF" when any character other than "H" is entered on the serial terminal

Lab Task 2: Write a C++ program with mbed APIs to gradually increase and decrease on-board LED brightness by receiving a character (“i” for increase and “d” for decrease) from PC via host terminal application (Tera term) using PWM signal with variable duty cycle. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Code:

```
#include "mbed.h"
Serial pc(USBTX,USBRX);
PwmOut led(PC_8);
float brightness=0.0;
int main(){
    while(1){
    pc.printf("Enter the command to increase/decrease the brightness(i/d)");
    char c =pc.getc();
    if((c=='i'))
    {
        pc.printf("Increasing the brightness by 10%: \r\n");
        brightness+=0.1;
        led = brightness;
    }
    if((c=='d')&&(brightness>0.0))
    {
        pc.printf("decreasing the brightness by 10%: \r\n");
        brightness -=0.1;
        led = brightness;
    }
    }
}
```

Output:

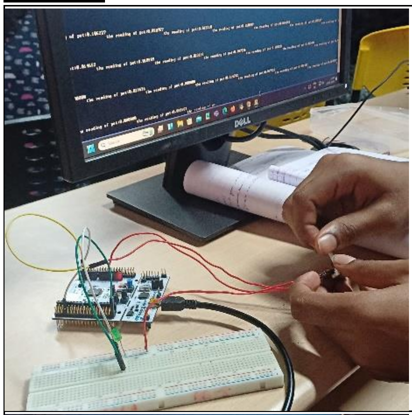
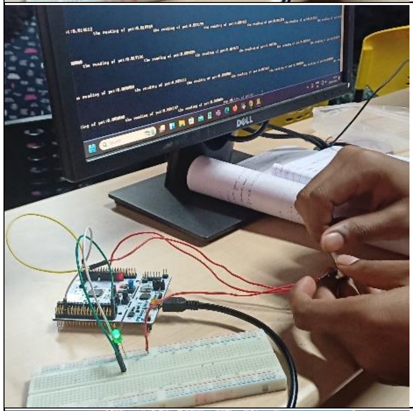
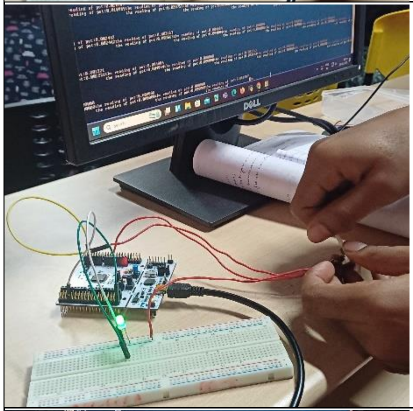
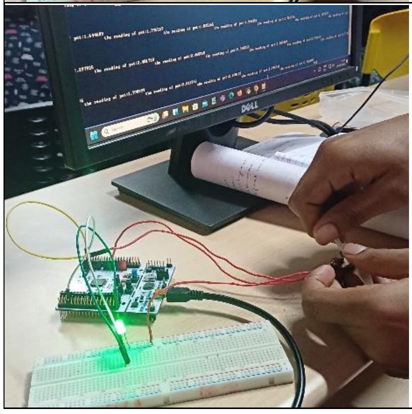
	**Fig. 2.1:** LED connected to the PWM pin is not glowing, indicating that character "d" had been pressed till the brightness reduced and the LED turned "OFF"
	**Fig. 2.2:** On entering character "i" to the terminal screen, the LED's brightness increases by 10%
	**Fig. 2.3:** On entering character "i" consecutively to the terminal screen, the LED's brightness increases more
	**Fig. 2.4:** After a few presses of character "i", the LED will be at maximum brightness. Entering the character "d" will decrease the brightness

Lab Task 3: Write a C++ program with mbed APIs to gradually increase and decrease LED brightness by changing the position of the Potentiometer via ADC. Also print it's voltage value on serial monitor. Use Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Code:

```
#include "mbed.h"
Serial pc(USBTX,USBRX);
AnalogIn ain(PC_3);
PwmOut PWM1(PC_8);
int main(){
    while(1){
        float value =ain*5;
        pc.printf("The reading of pot:%f",value);
        wait(1);
        PWM1.period(0.01);
        PWM1=ain;
    }
}
```

Output:

	**Fig. 3.1:** The Potentiometer's wiper is on one extreme and hence, at the maximum resistance value. The corresponding voltage value to this resistance is read using the AnalogIn API and a PWM with certain duty cycle (with voltage similar to the input analog voltage) is produced and fed to the LED.
	**Fig. 3.2:** Turning the wiper of the potentiometer leads to increase in input analog voltage and an increase in the PWM duty cycle.
	**Fig. 3.3:** Further Turning of the wiper of the potentiometer leads to further increase in input analog voltage and a corresponding increase in the PWM duty cycle.
	**Fig. 3.4:** When the input analog voltage is maximum, the PWM duty cycle is also maximum.

Lab Task 4 (Challenging Task): Write a C++ program with mbed APIs to design a battery level indicator system using potentiometer and LEDs. The system must display the different level of the voltage with the help of 5 LEDs as per following conditions.

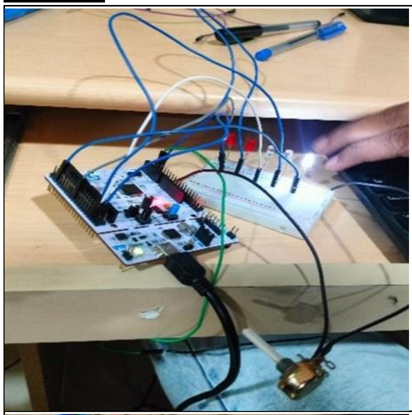
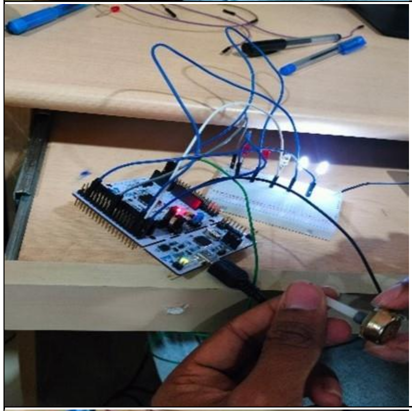
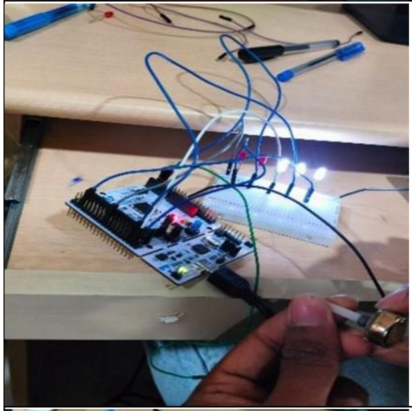
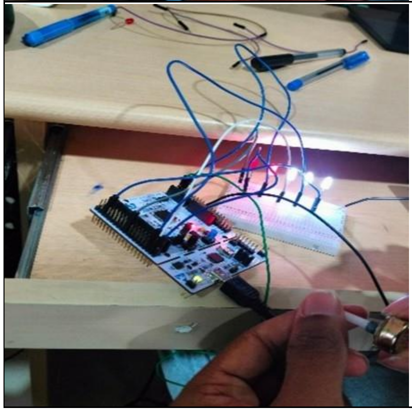
- If the voltage is between 0 to 1V glow LED1 and display “0 to 1V” in serial monitor
- If the voltage is between 1 to 2V glow LED1 and LED2 “1 to 2V” in serial monitor
- If the voltage is between 2 to 3V glow LED1 to LED3 “2 to 3V” in serial monitor
- If the voltage is between 3 to 4V glow LED1 to LED4 “3 to 4V” in serial monitor
- If the voltage is between 4 to 5V glow LED1 to LED5 “4 to 5V” in serial monitor

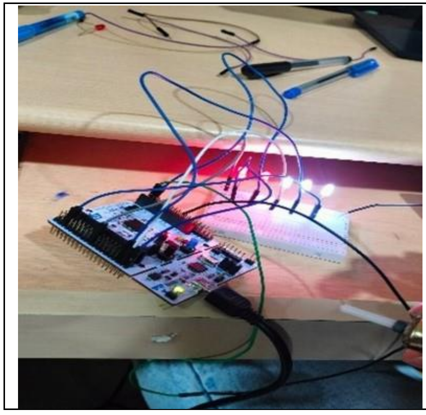
Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Code:

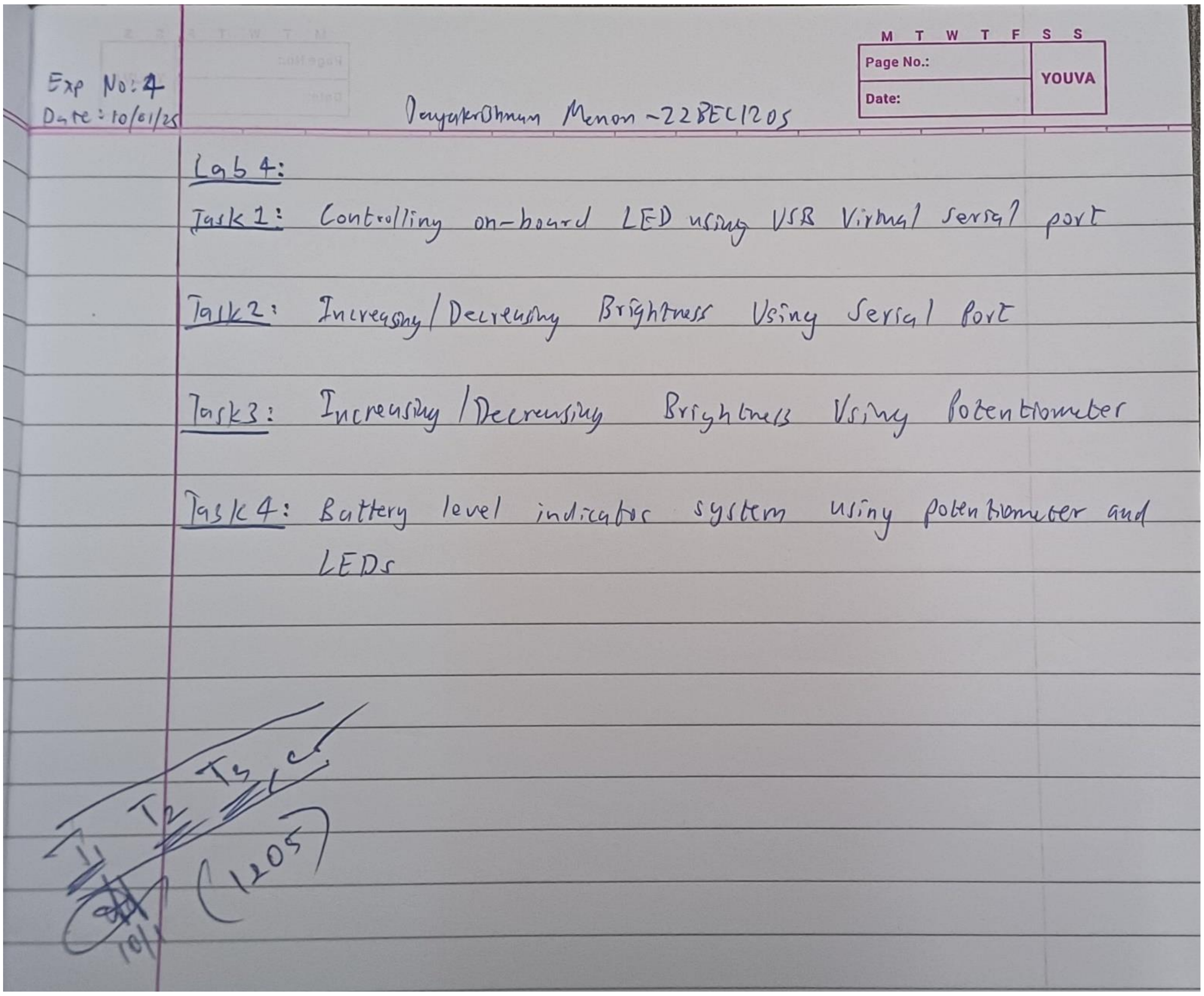
```
#include "mbed.h"
Serial pc(USBTX,USBRX);
AnalogIn ain(PC_3);
BusOut led(PC_8,PC_2,PC_6,PC_5,PC_4);

int main() {
    while(1){
    float value=ain*5;
    pc.printf("The reading of pot:%f\r\n",value);
    if(value>0 && value<=1){
    pc.printf("0 to 1V\r\n");
    led=16;//10000
    wait(0.5);
    }
    if(value>1 && value<=2){
    pc.printf("1 to 2V\r\n");
    led=24;//11000
    wait(0.5);
    }
    if(value>2 && value<=3){
    pc.printf("2 to 3V\r\n");
    led=28;//11100
    wait(0.5);
    }
    if(value>3 && value<=4){
    pc.printf("3 to 4V\r\n");
    led=30;//11110
    wait(0.5);
    }
    if(value>4 && value<=5){
    pc.printf("4 to 5V\r\n");
    led=31;//11111
    wait(0.5);
    }
    }
}
```

Output:

	**Fig. 4.1:** Only the 1st LED is glowing (10000), indicating very low battery power level
	**Fig. 4.2:** The starting 2 LEDs are glowing (11000), indicating a low battery power level
	**Fig. 4.3:** The starting 3 LEDs are glowing (11100), indicating a medium battery power level
	**Fig. 4.4:** The starting 4 LEDs are glowing (11110), indicating a high battery power level

	**Fig. 4.5:** All the 5 LEDs are glowing (11111), indicating maximum battery power level

OUTPUT VERIFICATION:



Exp No: 4
Date: 10/01/25

Jayakrishnan Menon - 22BEC1205

M T W T F S S
Page No.:
Date:
YOUVA

Lab 4:

Task 1: Controlling on-board LED using USB Virtual serial port

Task 2: Increasing / Decreasing Brightness Using Serial Port

Task 3: Increasing / Decreasing Brightness Using Potentiometer

Task 4: Battery level indicator system using potentiometer and LEDs

T1 T2 T3 C

(1205)

10/1

INFERENCE:

1. The “Serial” API in mbed provides a mechanism for serial communication, enabling the microcontroller to send and receive data over a serial interface. Its Syntax is: “Serial Identifier(PinName tx, PinName rx, int baud)”.
2. The Serial API provides two methods, getc and putc. The getc() function is used to receive a single character from the serial interface. It returns the received character. The putc() function is used to transmit a single character over the serial interface. It takes the character to be transmitted as its argument.
3. The USBTX, USBRX arguments (in “Serial Identifier(USBTX, USBRX)”) allow Serial communications to happen through the micro USB cable on the Development board (“USB

Virtual Serial Port”). Hence, serial communication doesn’t require any additional pins to be configured.

4. Using the “BusOut” API, we can combine a number of “DigitalOut” pins to write values on all of them using a single statement. Its Syntax is: “BusOut Identifier(PinNames)”.
5. The “identifier” of a “BusOut” can be assigned values, inside the main function. Through this, the user can assign different output values to the Output Bus, based on the requirements. It is possible to assign hexadecimal values (eg: 0x08) or decimal values (eg: 8) to it.
6. The “AnalogIn” API in mbed allows you to read analog signals from an analog input pin. This is useful for interfacing with sensors that produce analog outputs, such as potentiometers, light sensors, and temperature sensors. Its Syntax is: “AnalogIn Identifier(PinName)”.
7. The “read()” function in the AnalogIn API reads the analog value from the specified pin. It returns a value representing the analog reading, typically in the range of 0 to 1.
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 Serial Ports and Analog to Digital Converters feature of Nucleo64-STM32L152RE Board for applications like controlling the state of an LED and varying its brightness as well as reading Analog voltage values through ADC pins, were understood and the tasks were performed successfully.