



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: 13/02/2025

Slot: L27+L28

Date: 13/02/2025

LAB – 06: Working with a Servo Motors and LDRs

AIM: To understand the interfacing of Servo Motors and Light Dependent Resistors with Nucleo64-STM32L152RE Board for various applications involving light intensity detection and actuation of angular displacement. We will also perform the following tasks:

- Lab Task-1: Write a program to control the servo motor by rotating slowly from 0 degrees to 180 degrees, 45 degree at a time. When the motor has to be rotated 180 degrees, it will return to the initial position. Implement and verify this logic on STM32 board.
- Lab Task-2: Write a program to design auto intensity street light controller. This system helps the street light to get switched on automatically as per surrounding brightness. For example, sometimes when the weather become hazy its quite difficult to see anything then at that point this auto intensity street light gets switched ON based on present lighting condition. Implement and verify this logic on STM32 board.
- Lab Task-3: Write a program to design a solar tracking system for harvesting solar energy efficiently by the solar panels. This system is constructed by fitting two LDRs, angled away from each other by around 90, to a servo. Continuously read the light value sensed by the two LDRs and rotates the servo so that each is receiving equal light. As a sun-tracking system will be located to track the sun from sunrise to sunset, i.e. not more than 180. Also, display both LDR values and present Servo motor position on LCD. Implement and verify this logic on STM32 board.

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

HARDWARE REQUIRED: Micro USB cable, NUCLEO64-STM32L152 Board, Potentiometer, Jumper Wires (M-F and M-M), Breadboard, LCD, LDRs and a Servo Motor

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 program to control the servo motor by rotating slowly from 0 degrees to 180 degrees, 45 degree at a time. When the motor has to be rotated 180 degrees, it will return to the initial position. Implement and verify this logic on STM32 board.

Code:

```
#include "mbed.h"
PwmOut PWM1(PC_8);
int main(){
while(1){
PWM1.period_ms(20);
PWM1.pulsewidth_us(500); // 45 degree
wait(1);
PWM1.pulsewidth_us(1000); // 45 degree
wait(1);
PWM1.pulsewidth_us(1500); // 90 degree
wait(1);
PWM1.pulsewidth_us(2000); // 135 degree
wait(1);
PWM1.pulsewidth_us(2500); // 180 degree
wait(1);
}
}
```

Output:

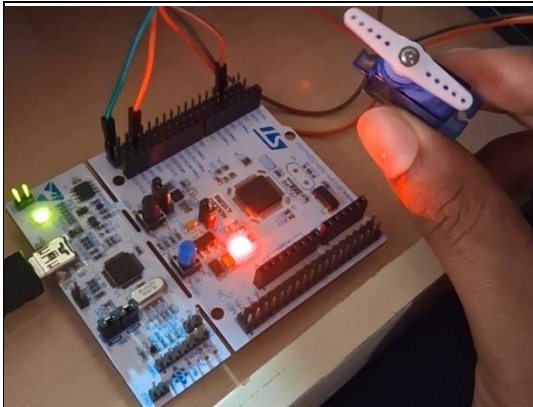


Fig. 1.1: Servo Motor Blade at 0 degrees.

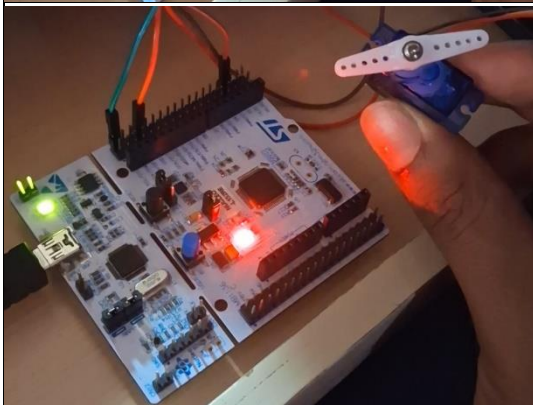


Fig. 1.2: Servo Motor Blade at 45 degrees.

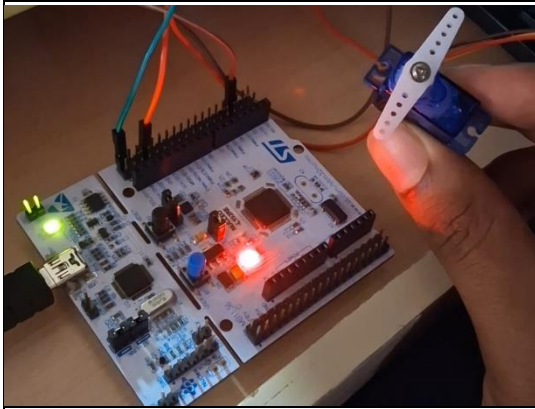


Fig. 1.3: Servo Motor Blade at 90 degrees.

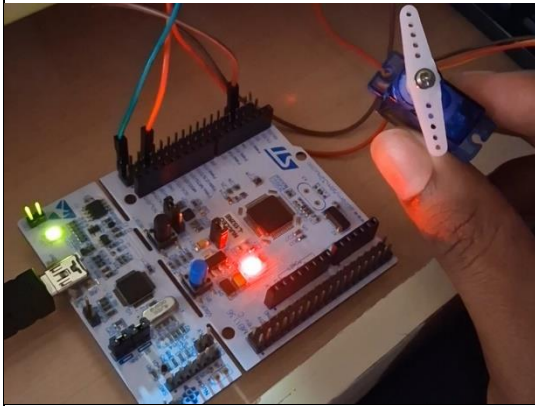


Fig. 1.4: Servo Motor Blade at 135 degrees.

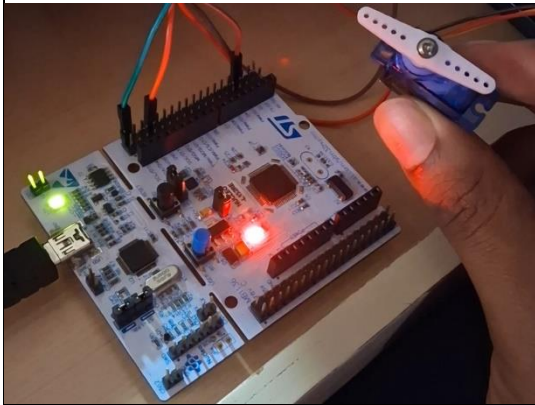


Fig. 1.5: Servo Motor Blade at 180 degrees.

Lab Task 2: Write a program to design auto intensity street light controller. This system helps the street light to get switched on automatically as per surrounding brightness. For example, sometimes when the weather become hazy its quite difficult to see anything then at that point this auto intensity street light gets switched ON based on present lighting condition. Implement and verify this logic on STM32 board.

Code:

```
#include "mbed.h"
PwmOut PWM1(PC_8);
AnalogIn Ain(PC_3);
int main(){
while (1){
PWM1.period(0.010); //PWM period = 10 milli seconds
PWM1=1-Ain;
wait(0.1);
}
}
```

Output:

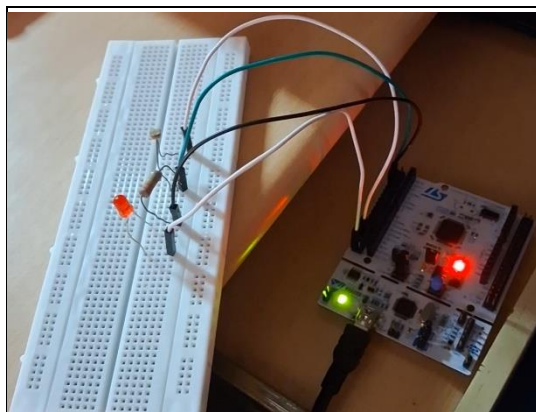


Fig. 2.1: Maximum Light is received by the LDR, hence no illumination of the LED, due to very less PWM duty cycle.

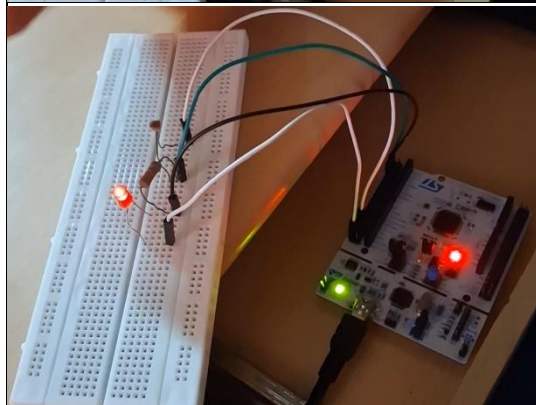


Fig. 2.2: Hand slightly obstructs the light received by the LDR, hence there is a slight illumination of the LED, due to increased PWM duty cycle.

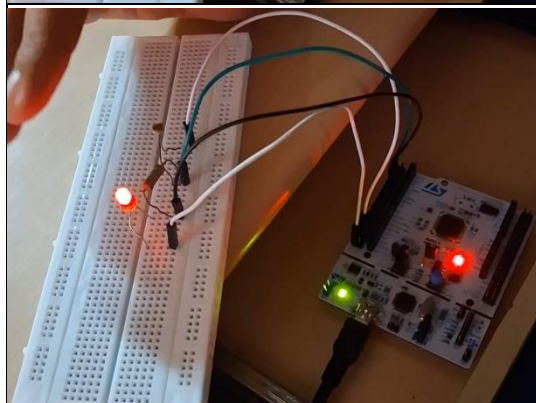


Fig. 2.3: More obstruction of the light causes the LED to turn more bright.

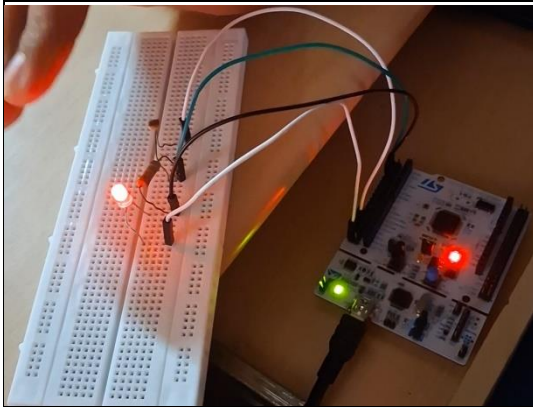


Fig. 2.4: More obstruction of the light causes the LED to turn more bright.

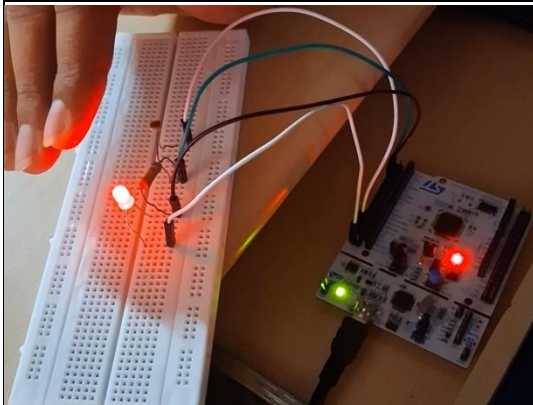


Fig. 2.5: When the Hand completely obstructs the light received by the LDR, the LED glows the brightest to maximum PWM duty cycle.

Lab Task 3: Write a program to design a solar tracking system for harvesting solar energy efficiently by the solar panels. This system is constructed by fitting two LDRs, angled away from each other by around 90, to a servo. Continuously read the light value sensed by the two LDRs and rotates the servo so that each is receiving equal light. As a sun-tracking system will be located to track the sun from sunrise to sunset, i.e. not more than 180. Also, display both LDR values and present Servo motor position on LCD. Implement and verify this logic on STM32 board.

Code:

```
#include "mbed.h"
#include "TextLCD.h"
TextLCD lcd(PC_0,PC_1,PB_0,PA_4,PA_1,PA_0);
AnalogIn Ain1(PC_3);
AnalogIn Ain2(PC_2);
PwmOut PWM1(PC_8);
int main(){
    while(1){
        lcd.locate(0,0);
        PWM1.period_ms(20);
        float ain1 = Ain1*5;
        float ain2 = Ain2*5;
        lcd.printf("%0.2f",ain1);
        wait(0.1);
        lcd.locate(7,0);
        lcd.printf("%0.2f",ain2);
        wait(0.1);
        lcd.locate(5,1);
        if(Ain1 == Ain2){
            PWM1.pulsewidth_us(1500);
            lcd.printf("90");
        }
        else if(Ain1>Ain2){
            PWM1.pulsewidth_us(1000);
            lcd.printf("45");
        }
        else{
            PWM1.pulsewidth_us(2000);
            lcd.printf("135");
        }
        wait(1);
        lcd.cls();
    }
}
```

Output:

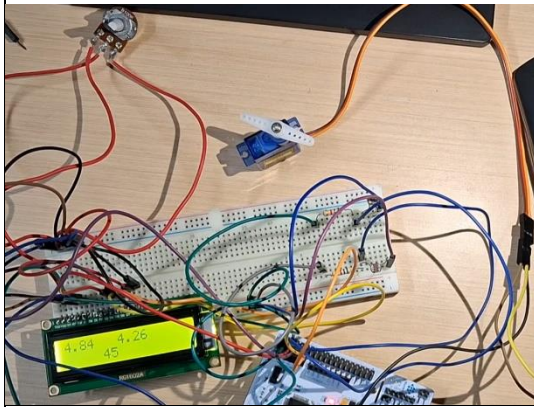


Fig. 3.1: More light falls on the top LDR than the bottom one. The top left value on the LCD is also greater due to less voltage drop. The Servo Motor is at a 45 degree angle and the same is displayed on the LCD as well.

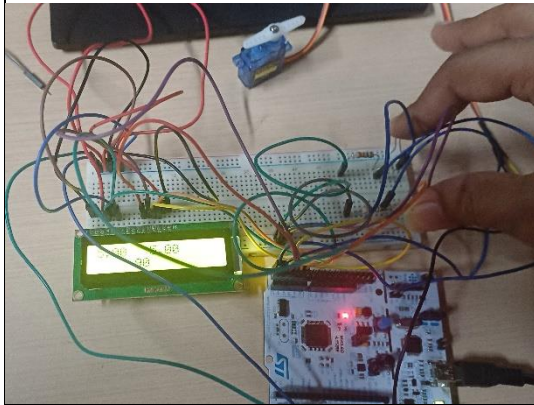


Fig. 3.2: Equal amounts of light falls on both LDRs than the bottom one. The Servo Motor is at a 90 degree angle and the same is displayed on the LCD as well.

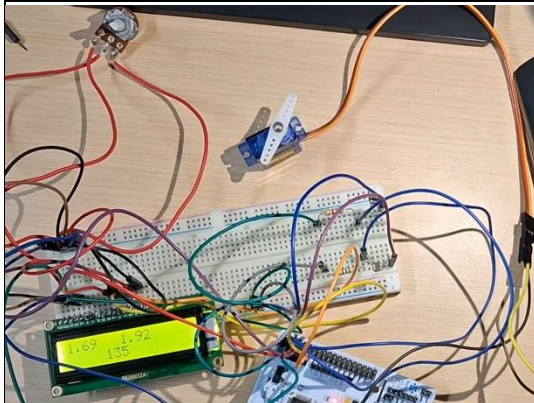
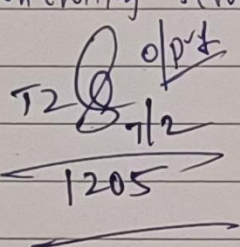
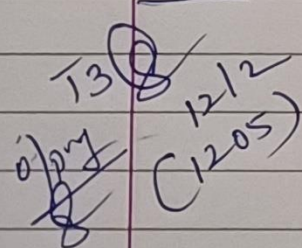


Fig. 3.3: More light falls on the bottom LDR than the top one. The top right value on the LCD is also greater due to less voltage drop. The Servo Motor is at a 135 degree angle and the same is displayed on the LCD as well.

OUTPUT VERIFICATION:

M T W T F S S	
Exp No: 6	Page No.:
Date: 7/2/25	Date:
YOUVA	
Jayakrishnan Menon - 22BEC1205	
Lab 6	
Task 1: Controlling servo motor, 45° at a time	
Task 2:  Auto intensity street light controller	
Task 3: Solar tracking system	
	

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 "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)".
5. 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.

6. The TextLCD library allows a Nucleo board to control Liquid Crystal Displays (LCDs) based on the Hitachi HD44780 chipset in 4-bit mode. It can be used by including the header file, "TextLCD.h". The mbed TextLCD library performs the laborious LCD setup routines and also tells the LCD object which pins are used for which functions.
7. The Syntax for using this API is: "TextLCD lcd(rs, enable, d4, d5, d6, d7)".
8. It also provides 4 main useful functions. The cls() function clears the screen and locates the cursor to 0,0. The locate() function can be used to move the cursor to any of the 16x2 positions available on the LCD. The putc() function writes a character to the LCD. The printf() function writes a formatted string to the LCD.
9. Programmable Delays can be implemented through the use of the "wait()" function.
10. A program can be set to run indefinitely using the "while(1)" loop

RESULT:

Thus, the interfacing of Servo Motors and Light Dependent Resistors with Nucleo64-STM32L152RE Board for various applications involving light intensity detection and actuation of angular displacement was understood and the tasks were also performed successfully.