THE NATIONAL INSTITUTE OF ENGINEERING

(Autonomous Institute affiliated to VTU, Belagavi)
Manandavadi Road, Mysuru – 570018
Phone: 0821-2403733/214 Website: www.nie.ac.in Email: officeece@nie.ac.in

ARM MICROCONTROLLER LAB

Project Title: "Analog Voltage-Based Control of Servo and Buzzer Using STM32F103C8T6"

SUBJECT CODE: BEC404

SEMESTER: IV A

PAVAN R	**4NI24EC405**
SAMPREETH G R	**4NI23EC133**
POOJA K N	**4NI23EC072**
POOJA R K	**4NI24EC406**

Faculty in charge:

Dr. Lokesha HR

Assistant Professor

DEPARTMENT

OF

ELECTRONICS AND COMMUNICATION ENGINEERING

2024-2025

THE NATIONAL INSTITUTE OF ENGINEERING
Mananthawadi Road
Mysuru-570001

DEPARTMENT
OF
ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the LIC lab work is carried out by **PAVAN R -4NI24EC405**, **SAMPREETH G R – 4NI23EC133, POOJA K N –4NI23EC072, POOJA R K -4NI24EC406** a Bonafide student of IV Semester ECE, NIE Mysore during the year 2024-2025 It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Lab reports. The lab record has been approved as it satisfies the academic requirements in respect **of Skill Lab: ARM Microcontroller lab** prescribed in the fourth semester in the Academic year **2024-25.**

Name & Signature of the Lab in Charge/ Faculty in Charge:

Abstract

This project introduces a voltage-activated peripheral control system utilizing the STM32 microcontroller platform. The system is designed to interpret real-time analog voltages applied to specific input pins and initiate defined hardware actions accordingly. In this configuration, the analog signals are fed into the microcontroller's ADC (Analog-to-Digital Converter) channels and interpreted through software logic to determine the state of operation. Depending on the input voltage levels, the microcontroller activates either a servo motor, a buzzer, or both. For instance, when a specific voltage is detected on an input pin, the servo may be triggered to rotate continuously, or the buzzer may be turned on to signal an event.

This voltage-based control eliminates the need for external switching mechanisms or amplification components, such as transistors or MOSFETs. All decisions and actions are handled internally by the microcontroller, simplifying the circuit and reducing external dependencies. A 16x2 I2C LCD is used to present the current ADC readings and system state, while a serial monitor provides additional debugging and visualization support. This report elaborates on the system architecture, pin configuration, implementation logic, threshold calibration, results, and real-time applications. The approach is tailored to support education, prototyping, and signal-based decision-making in embedded systems

Index Terms: *STM32, ADC, Voltage Control, Servo Motor, Buzzer, LiquidCrystal_I2C*

Introduction

This project presents a simple yet powerful demonstration of how voltage levels can control peripheral behaviour in embedded systems, specifically using the STM32F103C8T6 microcontroller (commonly known as the STM32 Blue Pill). Without using external amplification or switching components, this system relies solely on internal microcontroller logic to respond to analog voltage inputs and control hardware peripherals. The ADC (Analog-to-Digital Converter) channels continuously monitor three different analog inputs. Based on defined voltage thresholds, the microcontroller activates one or more output devices — namely a servo motor and a buzzer. The output status and corresponding ADC readings are displayed in real-time on a 16x2 I2C LCD and are also printed to the serial monitor for debugging and analysis purposes.

Objective

To build a microcontroller-based system that activates a servo motor and/or buzzer in response to analog voltage inputs without the use of transistors or MOSFETs. The system should read three analog voltages through the STM32's ADC pins, process the data in software, and provide real-time feedback via an LCD screen and serial monitor.

Overview

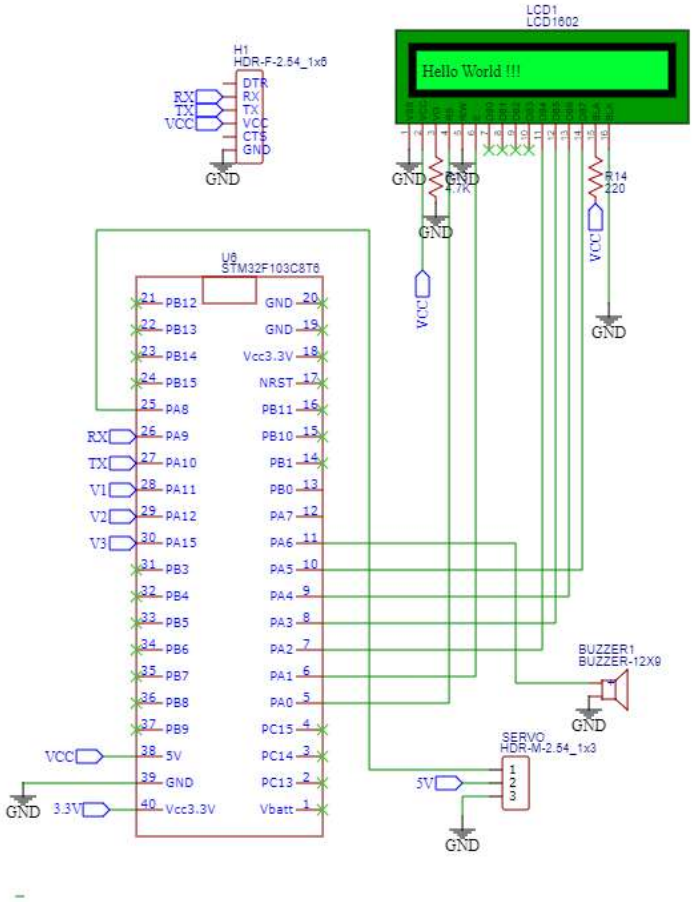
The complete setup includes the STM32 Blue Pill microcontroller board, a 16x2 LCD for user interface, a standard 5V buzzer, and a servo motor. Analog voltage signals are simulated using external sources such as variable power supplies or potentiometers. The microcontroller interprets the voltages and drives the appropriate peripherals accordingly. This configuration is ideal for illustrating real-time control and monitoring mechanisms in embedded systems education and prototyping environments.

.

Materials and Tools Used

- STM32 Blue Pill (STM32F103C8T6)
- Liquid Crystal I2C Display (16x2)
- SG90 Servo Motor
- 5V Buzzer
- External adjustable power supply / potentiometers for voltage simulation
- Arduino IDE with STM32 board support

Circuit Diagram



Pin Configuration

Function	STM32 Pin	Type	Description
LCD SDA	PB7	I2C	I2C Data line for LCD communication
LCD SCL	PB6	I2C	I2C Clock line for LCD communication
Servo Control	PB0	PWM	Controls Servo Motor angle
Buzzer	PB11	Digital Out	Activates buzzer when voltage condition met
Triode Analog Input	PA3	Analog In	Voltage input to trigger servo motion
Saturation Analog Input	PA2	Analog In	Voltage input to trigger both buzzer & servo
Cutoff Analog Input	PA4	Analog In	Voltage input to activate buzzer only
UART TX (Serial Output)	PA9	UART	Sends ADC data to serial monitor
UART RX (Optional)	PA10	UART	Receives data if needed (not used here)

Software

The firmware for this project is developed using the Arduino IDE, a user-friendly and widely supported platform for embedded system programming. Despite being designed for Arduino boards, the IDE supports STM32 microcontrollers through the installation of STM32 board packages. This makes it a convenient and efficient choice for rapid development and testing. The main program logic revolves around continuously reading three analog voltage inputs using the STM32's ADC (Analog-to-Digital Converter) channels.

These inputs are interpreted based on empirically defined ADC threshold values. If a certain voltage level is detected on a particular analog pin, corresponding hardware actions are triggered. Specifically, a servo motor may be activated to rotate continuously, a buzzer may be sounded to indicate a voltage threshold event, or both may be triggered simultaneously.

The LCD connected via I2C is used to display the real-time ADC readings and the active control states. Additionally, the UART module transmits the ADC values to the serial monitor, providing an alternative method for real-time monitoring and debugging. This combination of visual and serial outputs enhances the observability and reliability of the system during development and demonstration phases.

Code Implementation Highlights

```
if (V2_Active) {
  digitalWrite(BUZZER_PIN, HIGH);
  myServo.write(180);
} else if (V3_Active) {
  digitalWrite(BUZZER_PIN, HIGH);
  myServo.write(90);   // Cut
} else if (V1_Active) {
  digitalWrite(BUZZER_PIN, LOW);
  myServo.write(180);
} else {
  digitalWrite(BUZZER_PIN, LOW);
  myServo.write(90);
}
```

Operation Flow

1. System boots and displays welcome message on LCD.
2. Reads analog voltages on PA2, PA3, PA4.
3. Compares ADC values to defined thresholds.
4. Activates buzzer, servo based on conditions.
5. Displays ADC readings and states on LCD and serial monitor.

Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#define BUZZER_PIN PB11
#define SERVO_PIN PB0
#define ANALOG_V1_PIN PA3  // Triode -> V1
#define ANALOG_V2_PIN PA2  // Saturation -> V2
#define ANALOG_V3_PIN PA4  // Cutoff -> V3
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo myServo;
void setup() {
 lcd.init();
 lcd.backlight();
 pinMode(BUZZER_PIN, OUTPUT);
 digitalWrite(BUZZER_PIN, LOW);
 myServo.attach(SERVO_PIN);
 myServo.write(90);  // Stop servo
 // Welcome message
 lcd.setCursor(0, 0);
 lcd.print("ARM PROJECT");
 delay(2000);
 lcd.clear();
 Serial.begin(9600);
}
```

```
void loop() {
  int v1ADC = analogRead(ANALOG_V1_PIN);  // PA3
  int v2ADC = analogRead(ANALOG_V2_PIN);  // PA2
  int v3ADC = analogRead(ANALOG_V3_PIN);  // PA4

  lcd.setCursor(0, 0);
  lcd.print("V1:");
  lcd.print(v1ADC);
  lcd.print(" V2:");
  lcd.print(v2ADC);

  lcd.setCursor(0, 1);
  lcd.print("V3:");
  lcd.print(v3ADC);
  lcd.print("     "); // clear extra chars

  Serial.print("V1: "); Serial.print(v1ADC);
  Serial.print(" | V2: "); Serial.print(v2ADC);
  Serial.print(" | V3: "); Serial.println(v3ADC);

  bool V1_Active = (v1ADC >= 3970 && v1ADC <= 4010);  // Triode
  bool V2_Active = (v2ADC >= 2450 && v2ADC <= 2510);  // Saturation
  bool V3_Active = (v3ADC >= 2450 && v3ADC <= 2510);  // Cutoff

  if (V2_Active) {
    digitalWrite(BUZZER_PIN, HIGH);
    myServo.write(180);
  } else if (V3_Active) {
    digitalWrite(BUZZER_PIN, HIGH);
```

```
  myServo.write(90);
 } else if (V1_Active) {
  digitalWrite(BUZZER_PIN, LOW);
  myServo.write(180);
 } else {
  digitalWrite(BUZZER_PIN, LOW);
  myServo.write(90);
 }

 delay(300);
}
```

UART Communication

The STM32 microcontroller in this project utilizes UART (Universal Asynchronous Receiver Transmitter) communication for transmitting analog-to-digital conversion results to an external serial monitor. Specifically, the TX (transmit) line on pin PA9 is configured to send the processed voltage values (from pins PA2, PA3, and PA4) at a baud rate of 9600. This serial output is a critical diagnostic and debugging tool during both development and deployment phases. UART communication is chosen for its simplicity, ubiquity in embedded systems, and minimal hardware requirement—only a TX and RX line are necessary.

By displaying real-time data in the Arduino IDE's Serial Monitor or Plotter, developers can quickly validate whether the analog voltages are being interpreted correctly. This is particularly useful when setting or adjusting voltage thresholds for triggering peripheral actions (such as activating the buzzer or rotating the servo motor). Unlike using LCD alone, which is constrained by limited character space, UART enables continuous streaming of comprehensive sensor data for visualization and analysis. Additionally, since UART is natively supported in STM32 microcontrollers and seamlessly integrated within the Arduino IDE ecosystem, it provides an efficient and reliable method for ensuring system accuracy and responsiveness during execution.

Working Principle

The voltage-controlled peripheral system implemented using the STM32 microcontroller operates by continuously monitoring three distinct analog input lines labeled V1 (PA3), V2 (PA2), and V3 (PA4). Each input line corresponds to a different functional output behavior based on the voltage thresholds defined in the program logic. V1 is programmed to activate the servo motor when the analog voltage detected on pin PA3 falls within a specified range, typically associated with a higher analog voltage level. This mimics the servo motor's activation in a voltage-dominant control logic.

Similarly, V2 (PA2) represents a condition where both the servo motor and the buzzer are activated simultaneously when the analog voltage input matches a mid-range threshold value. This dual-actuation behavior ensures that the system can alert the user audibly and visibly when this condition occurs. On the other hand, V3 (PA4) is designated to trigger only the buzzer, providing an auditory alert when the input voltage matches a different threshold range.

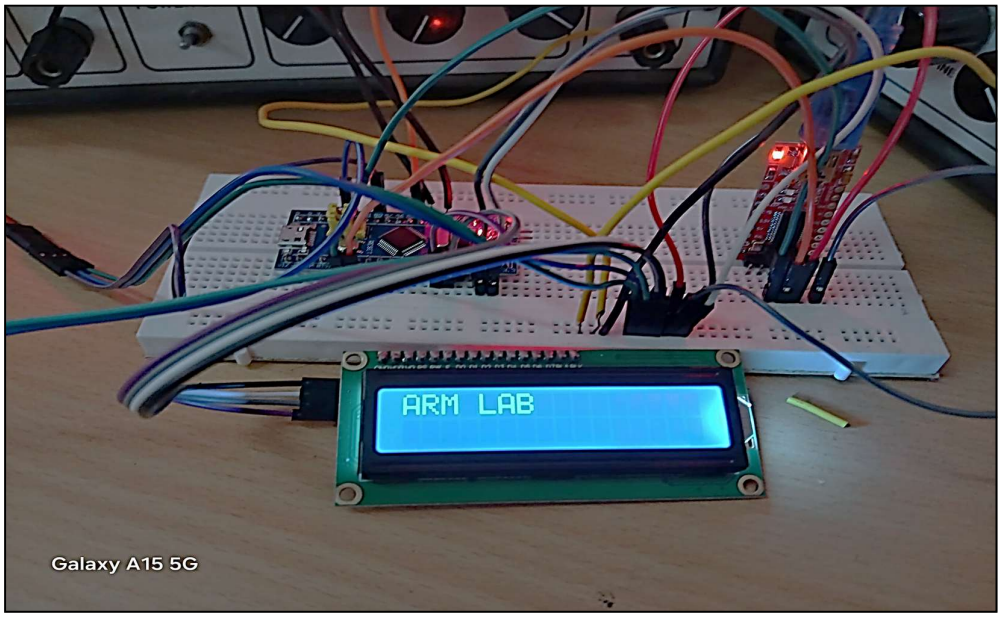
The loop function embedded within the firmware constantly reads the analog values from these three inputs using the STM32's ADC channels. It then compares the obtained digital values with predefined bounds. If a value lies within the expected range for a specific action, the corresponding actuator(s) are triggered accordingly. This efficient and continuous scanning mechanism enables real-time system response to varying input conditions, and it effectively demonstrates how a simple analog voltage can be utilized for conditional peripheral control in embedded systems without any external switching components.

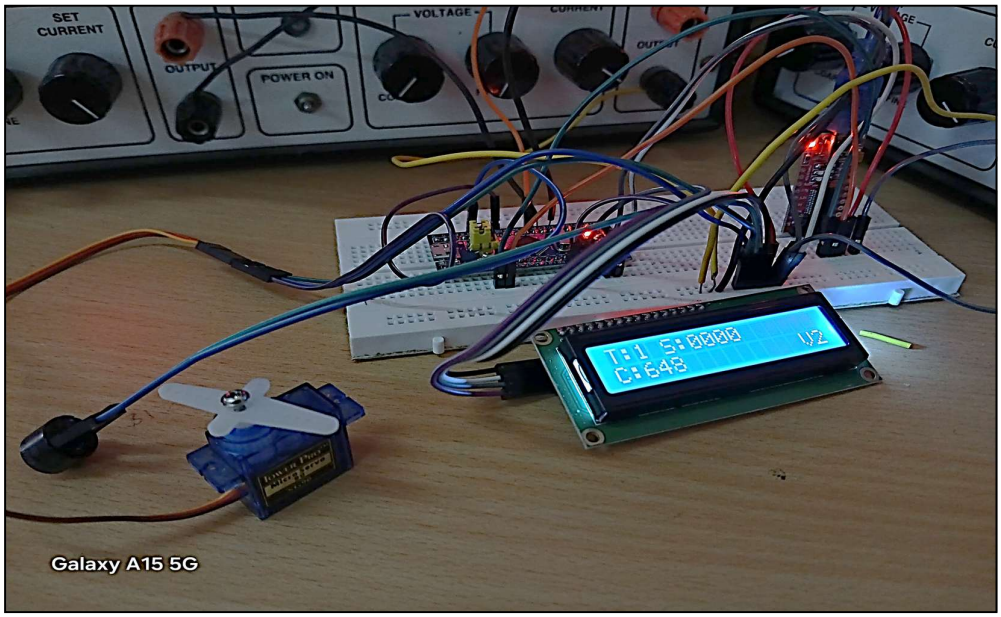
Testing And Results

To validate the operational logic of the voltage-controlled system, each of the analog input pins (V1, V2, and V3) was thoroughly tested using a regulated variable power supply. These controlled voltage inputs allowed systematic calibration of the ADC threshold ranges, ensuring that each input triggered the intended hardware response. The empirical process involved gradually adjusting voltage levels while monitoring both LCD and UART outputs for confirmation. This method ensured that the response boundaries were accurate and that the peripherals—servo motor and buzzer—reacted reliably to voltage changes.

During the trials, it was observed that V1 effectively initiated continuous servo rotation upon detecting voltage within the specified high threshold range. Similarly, V2 correctly activated both the servo and buzzer when its mid-range threshold condition was met. This confirmed that the dual-output logic embedded in the program was functioning correctly. For V3, an isolated buzzer activation was confirmed when the input voltage aligned with its predefined threshold. These results indicated that the decision logic based purely on voltage levels performed robustly without any external control components.

In addition, the UART interface, configured via PA9 at 9600 baud, played a pivotal role in debugging and fine-tuning the threshold values. Real-time feedback on ADC readings helped correlate actual voltage values with digital equivalents, streamlining the calibration process. The UART monitor thus served as an essential tool for verifying operational reliability and achieving consistent system behavior under varying input conditions.





Applications

- **Educational demonstration of analog signal processing**: This project serves as an excellent hands-on example for students and hobbyists learning about analog-to-digital conversion, signal interpretation, and peripheral control using microcontrollers. It simplifies complex electronics by eliminating external components like MOSFETs and demonstrates how software logic alone can drive actuators. The visual display via LCD and debugging via UART further reinforce core embedded system concepts, making this ideal for classroom demonstrations and workshops.
- **Simple alarm and automation systems**: The voltage-triggered responses showcased here can easily be extended to basic alarm systems in home or industrial settings. For example, if voltage from a sensor surpasses a threshold, a buzzer can alert users of a breach or abnormality. Similarly, servo-driven mechanisms can be used to open or close valves, doors, or hatches, making this setup adaptable for straightforward automation without complex circuitry.
- **Voltage-based trigger systems**: This configuration exemplifies the principle of voltage-based actuation. It can be adapted into environments where sensor readings (like temperature, light, or gas levels) are converted to voltage and used to activate specific outputs. This makes it valuable for control panels or decision-making logic in industrial, medical, or agricultural applications where specific voltages correspond to actionable events.
- **Prototyping smart sensing interfaces**: Developers and engineers prototyping IoT or embedded systems can use this architecture to test different sensing responses. By replacing the voltage input with real-world sensors, the project becomes a rapid prototyping tool to assess how sensor data can be translated into actuator control. It provides a scalable foundation to build more complex smart systems and serves as a stepping stone toward automation and remote-control applications.
- **Voltage threshold-based safety systems**: In critical safety applications where exceeding or falling below a particular voltage level could signal a dangerous condition (such as overheating or chemical leak), this type of control logic provides a straightforward and dependable solution. Integrating a buzzer or visual alert in such scenarios ensures that early warnings are provided without requiring a complex processing unit or expensive hardware.
- **Low-cost automation control**: In regions or applications where cost constraints prohibit the use of high-end automation solutions, this design can be employed as a low-cost alternative. Its reliance on voltage input simplifies the sensor interface, while the use of widely available components like STM32 Blue Pill, LCDs, and buzzers ensures affordability and accessibility without compromising on functionality.

Conclusion

This project successfully demonstrates a simple yet powerful implementation of voltage-based peripheral control using the STM32 Blue Pill microcontroller. By utilizing internal ADC channels and digital output pins, it eliminates the need for external switching components such as MOSFETs or transistors. The system effectively interprets three distinct analog voltage signals and triggers corresponding outputs: a servo motor, a buzzer, or both. The integration of a 16x2 I2C LCD and UART communication provides comprehensive real-time feedback, enhancing user interaction and debugging capabilities.

The design showcases the flexibility of embedded systems where logic and response behavior are entirely governed by software thresholds and internal microcontroller peripherals. The use of the Arduino IDE further simplifies the development and deployment process, making the project accessible to students, hobbyists, and rapid prototyping scenarios. The clear separation of functions based on voltage thresholds reflects a scalable approach to analog signal processing, potentially adaptable to sensor-based decision-making systems.

Future Scope

This voltage-controlled system provides a solid foundation for expansion into more sophisticated applications. In the future, the design can be extended to accommodate wireless control using Bluetooth or Wi-Fi modules (e.g., HC-05 or ESP8266), enabling remote monitoring and actuation. Additionally, integration with real-world analog sensors such as gas, temperature, or light sensors would allow the project to serve in smart automation and environmental monitoring applications.

Another significant enhancement would be to implement interrupt-based ADC sampling to improve power efficiency and responsiveness, especially in battery-powered scenarios. The system can also be connected to a cloud-based dashboard via IoT platforms, making it viable for remote industrial or agricultural automation systems. Lastly, replacing the 16x2 LCD with a graphical OLED display could further improve the visual interface.

Limitations

While the system performs effectively within its scope, it does have a few constraints. First, the voltage thresholds used for decision-making are hardcoded and may need to be recalibrated for different environmental or supply conditions. Secondly, the reliance on analog voltage sources (like potentiometers or lab power supplies) makes the setup less ideal for direct deployment in real-world sensing environments without additional interfacing.

The use of the Arduino IDE, though accessible, abstracts certain low-level functionalities of STM32, potentially limiting performance optimization. Moreover, without hardware-level voltage protection or filtering, the ADC readings could be affected by noise or unstable input sources. The system also lacks memory storage or logging capabilities, making it unsuitable for data persistence or trend analysis without external additions.

References

STM32 Arduino Core Documentation: https://github.com/stm32duino

Servo Motor Control with Arduino IDE: https://www.electronicshub.org/servo-motor

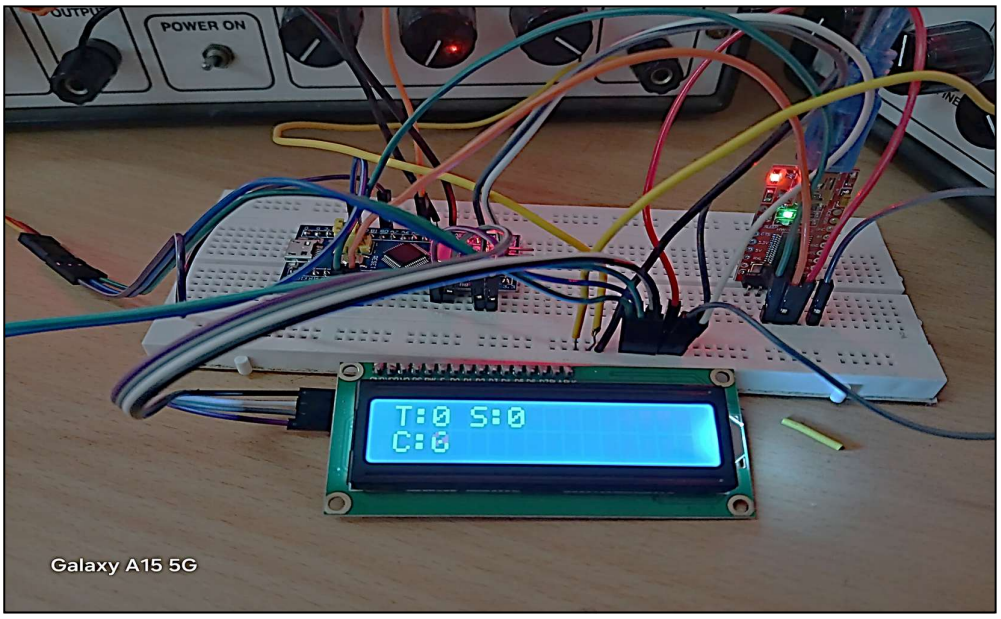
LiquidCrystal_I2C Library Documentation: https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c

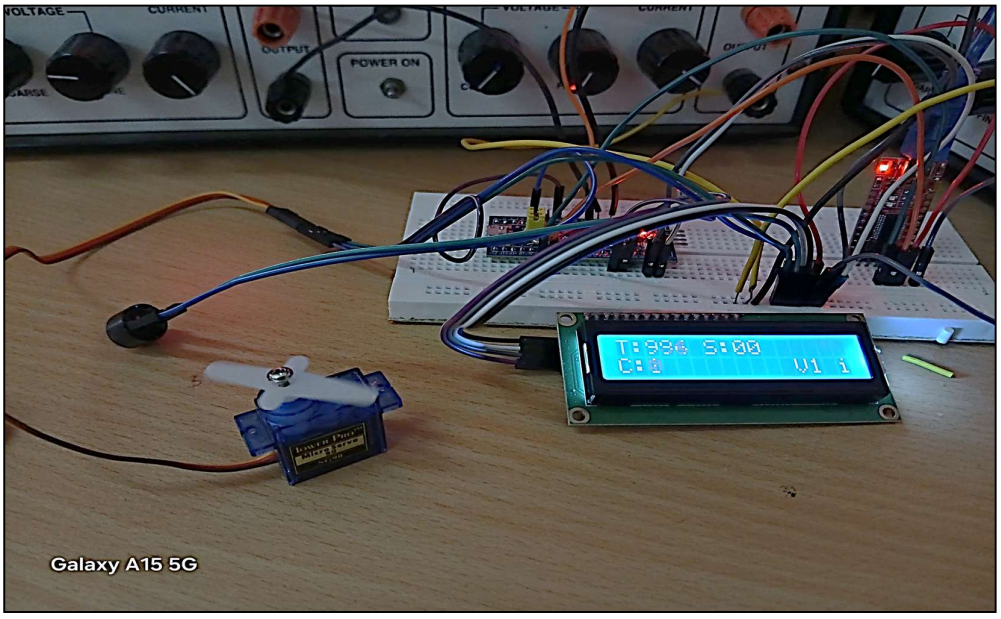
STM32 Blue Pill Getting Started Guide: https://circuitdigest.com/microcontroller-projects/getting-started-with-stm32-using-arduino-ide

UART Communication in STM32:

https://deepbluembedded.com/stm32-uart-serial-communication-tutorial

Analog-to-Digital Conversion Principles: https://www.ti.com/lit/an/slyt176/slyt176.pdf


POWER ON
T:0 S:0
C:0
Galaxy A15 5G


VOLTAGE
CURRENT
OUTPUT
POWER ON
Galaxy A15 5G