ECEN 260 - Final Project

Victor's Final Project

Victor Santana

Instructor: Brother Jason Allred

April 6, 2025

Contents

List of Tables

List of Figures

1 Final Project Overview

The Battery Voltage Meter project is designed to provide an efficient and accurate method for monitoring the voltage levels of commonly used batteries (e.g., 3V, AAA, 9V, A23). With the increasing reliance on battery-powered devices, it is essential to ensure that batteries operate within safe voltage ranges. This project aims to measure, display, and communicate the battery voltage, allowing users to assess battery health in real time.

The system is based on a microcontroller that reads the voltage from a battery through an ADC (Analog-to-Digital Converter) and displays the value on an I2C-based LCD display. Additionally, UART communication enables data/command transfer from a laptop, allowing for remote monitoring of the battery status. The system includes various safety features, such as voltage range monitoring, to ensure that devices powered by these batteries operate within specified limits. The project also integrates essential engineering principles like ADC, UART, and display communication, creating a practical solution that can be used in various applications, from personal devices to troubleshooting circuits.

By providing a simple interface and real-time data, this battery voltage meter ensures optimal battery performance, reduces the risk of device failure, and contributes to sustainable energy practices.

2 Final Project Design

My project is designed to meet important needs in electronics and electrical systems, specifically for measuring battery voltage. This helps to: (1) Monitor voltage levels in commonly used batteries to ensure they are within operational limits. (2) Assist in troubleshooting electrical issues, identifying faulty or underperforming batteries. (3) Aid in circuit design, ensuring voltage values remain optimal for circuit functionality and safety.

While not solving all issues, my project contributes to public safety and economic factors by ensuring devices operate within safe voltage levels. Public Safety: It helps prevent malfunctions, overheating, or damage to circuits by ensuring batteries are within safe voltage ranges, reducing the risk of electrical fires or failure. Economic Factors: By providing accurate voltage readings, industries can ensure their products function reliably and avoid premature device failures, ultimately saving costs. The project promotes sustainability by encouraging efficient battery use and recycling, raising awareness about energy conservation and responsible battery disposal.

My design incorporates several engineering principles: (1) ADC (Analog-to-Digital Conversion): Used to accurately measure battery voltage by converting the analog signal into a digital value. (2) UART Communication: Facilitates communication between the laptop and the microcontroller for easy remote monitoring of battery levels. (3) Display (I2C Communication): An LCD connected via I2C displays the battery voltage and percentage, providing clear, real-time data to the user.

3 Final Project Specifications

3.1 System Functionality:

The Battery Voltage Meter system is designed to measure and display the voltage and charge percentage of commonly used batteries. It supports three different types of batteries: 1.5V, 3V, and 9V. The system allows the user to select the battery type they want to measure or reset the system via UART commands in PuTTY, with the following available commands:

- **VIEW_1.5V:** Measures and displays voltage for a 1.5V battery.
- **VIEW_3V:** Measures and displays voltage for a 3V battery.
- **VIEW_9V:** Measures and displays voltage for a 9V battery.
- **RESET:** Resets the entire system.

The user can input one of these commands to view the voltage and percentage of the selected battery type. The system will then display the corresponding battery's voltage and percentage on the LCD1602 display. Since the display is limited to two rows, only the voltage and percentage of one battery type can be shown at a time. The system can be reset if the appropriate command is give. If a incorrect command is written through PuTTY, the system will send an error message back to the user.

3.2 How to Use the System:

1. **Power on the system:** Connect the system via USB to a laptop or PC. The system will be powered through the USB connection.
2. **Connect the battery:** Use alligator clips to connect the selected battery (1.5V, 3V, or 9V) to the system. The voltage divider is used for the 9V battery to protect the system.

3. **Select battery type:** In PuTTY, enter the corresponding command based on the battery type:

    - Enter "VIEW_1.5V" to measure a 1.5V battery.
    - Enter "VIEW_3V" to measure a 3V battery.
    - Enter "VIEW_9V" to measure a 9V battery.

4. **Read the output:** The system will display the battery voltage and percentage on the LCD1602 display.

5. **Reset the system:** The system will be reset if the user inputs the "RESET" command in PuTTY.

3.3 Operating Constraints:

- **Voltage Range:** The system is designed to measure 1.5V, 3V, and 9V batteries only. Any voltage outside of these ranges will not be supported.
- **Display Limitations:** The LCD1602 display has two rows, limiting the amount of information that can be displayed at once. Only the voltage and percentage for the selected battery type will be shown.
- **User Input:** The system can only measure one type of battery voltage at a time. The user must input the correct command in PuTTY before measuring another type of battery.

3.4 Design Limitations:

- **Single Measurement:** The system is limited to measuring only one battery type at a time. If the user wants to switch between different battery types, they must input the correct command through PuTTY.

- **Voltage Divider:** To safely measure the 9V battery, a voltage divider consisting of three 100-ohm resistors is used. This ensures that the 9V battery does not exceed the system's input voltage limits and prevent damage.

3.5 Parts List

- 1x 9V Battery.
- 1x 1.5V Battery.
- 1x 3V Battery.
- 2x Alligator Clips.
- x10+ Jumper Wires
- 1x Breadboard.
- 6x 100-ohm Resistors.
- 3x LEDs.
- 1x USB Cable (for Nucleo Board).
- 1x STM32 Nucleo-L476RG
- 1x LCD1602 Display

4 Schematics

The circuit diagram shows the battery voltage meter system, using an STM32 microcontroller (NUCLEO-L476RG) to measure the voltage of 1.5V, 3V, and 9V batteries. The system uses an ADC to convert the battery voltage to a digital value, which is displayed on an LCD1602 screen via I2C. UART communication is used to send the voltage data to a laptop or PC for remote monitoring.

The circuit includes 100-ohm resistors for voltage division to protect the system when measuring high voltages, like the 9V battery.

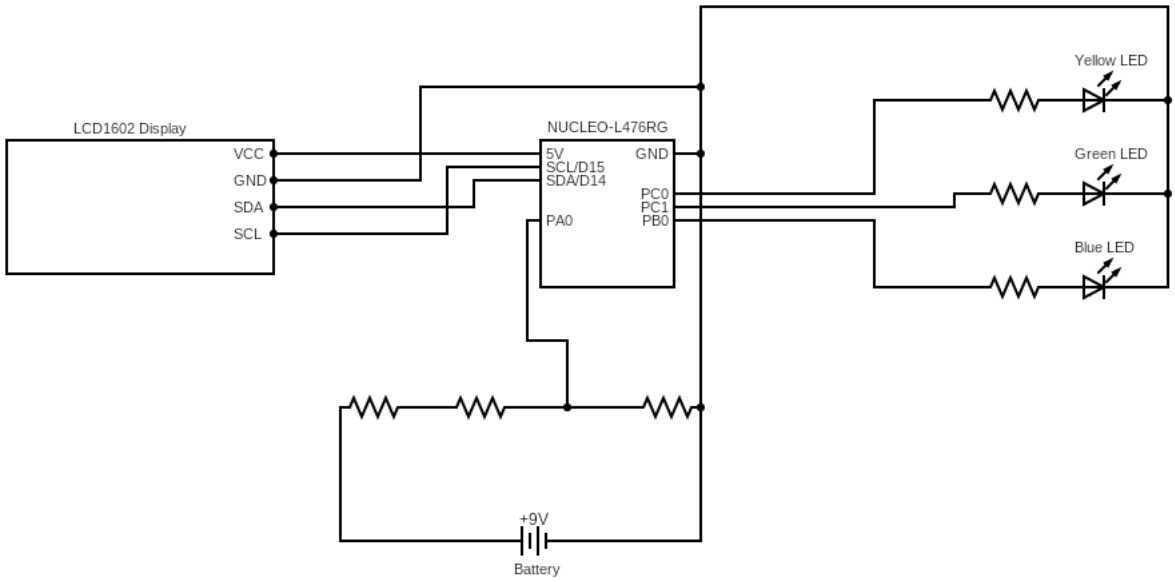

Figure 1: Schematic diagram for the Final Project.

5 Test Plan and Test Results

This section details the test scenarios conducted to verify the functionality and reliability of the Battery Voltage Meter system. The system was tested across a series of steps to ensure it correctly measures and displays the voltage and charge percentage for 1.5V, 3V, and 9V batteries, while also interacting with the user via the PuTTY interface. The test cases include both common scenarios and edge cases to validate the system's behavior under different conditions.

5.1 Test Scenarios Overview:

- **Test Scenarios 1-3:** These tests check the system's ability to correctly measure the voltage and display the corresponding values for the 1.5V, 3V, and 9V batteries. Each test also verifies that the correct LED is lit (yellow, green, or blue).
- **Test Scenarios 4-5:** These tests check the RESET functionality and functionality when no command is given, ensuring that the system can clear the display, turn off the LEDs when instructed, and not display values when not instructed.
- **Test Scenarios 6-8:** These tests assess the system's behavior when an invalid command is issued or when the system is asked to measure a battery that doesn't match the selected type. The expected result in such cases is for the system to display invalid values, while the correct LED should remain on.

Each test case includes the expected results and the actual results, ensuring the system behaves as expected and is capable of handling various user inputs and edge cases.

5.2 Test Plan Procedure

- Test Scenario #1

  - Step 1: In PuTTY type the command "VIEW_9V".
  - Step 2: See if the LCD display displays the following: "Voltage: 0.0V" and "Charge: 0.0%".
  - Step 3: See if the yellow LED turned on and all other LEDs are off.
  - Step 4: Connect the alligator clips to the 9V battery.
  - Step 5: See the displayed results.

- Test Scenario #2
  - Step 1: In PuTTY type the command "VIEW_1.5V".
  - Step 2: See if the LCD display displays the following: "Voltage: 0.0V" and "Charge: 0.0%".
  - Step 3: See if the green LED turned on and all other LEDs are off.
  - Step 4: Connect the alligator clips to the 1.5V battery.
  - Step 5: See the displayed results.

- Test Scenario #3
  - Step 1: In PuTTY type the command "VIEW_3V".
  - Step 2: See if the LCD display displays the following: "Voltage: 0.0V" and "Charge: 0.0%".
  - Step 3: See if the blue LED turned on and all other LEDs are off.
  - Step 4: Connect the alligator clips to the 3V battery.
  - Step 5: See the displayed results.

- Test Scenario #4
  - Step 1: In PuTTY type the command "RESET".
  - Step 2: See if the LCD display is completely cleared up.
  - Step 3: See if all LEDs are turned off.

- Test Scenario #5
  - Step 1: Connect alligator clips to the 9V battery.
  - Step 2: See the status of the LCD and check if displays the following: "Voltage: (value)" and "Charge: (value)%".
  - Step 3: See the status of the yellow LED, and check if it is on.
  - Step 4: Connect alligator clips to the 1.5V battery.
  - Step 5: See the status of the LCD and check if displays the following: "Voltage: (value)" and "Charge: (value)%".
  - Step 6: See the status of the green LED, and check if it is on.
  - Step 7: Connect alligator clips to the 3V battery.
  - Step 8: See the status of the LCD and check if displays the following: "Voltage: (value)" and "Charge: (value)%".
  - Step 9: See the status of the blue LED, and check if it is on.
  - Step 10: Reset the system by typing the "RESET" command in PuTTY
- Test Scenario #6
  - Step 1: In PuTTY type the command "VIEW_9V".
  - Step 2: See if the yellow LED turned on.
  - Step 3: Connect alligator clips to the 1.5V battery.
  - Step 4: See the displayed values in the LCD display.
  - Step 5: See if the green LED turned on.
  - Step 6: Connect alligator clips to the 3V battery.
  - Step 7: See again the displayed values in the LCD display.
  - Step 8: See if the blue LED turned on.
  - Step 9: Reset the system by typing "RESET" in PuTTY.
- Test Scenario #7

  - Step 1: In PuTTY type the command "VIEW_1.5V".
  - Step 2: See if the green LED turned on.
  - Step 3: Connect alligator clips to the 9V battery.
  - Step 4: See the displayed values in the LCD display.
  - Step 5: See if the yellow LED turned on.
  - Step 6: Connect alligator clips to the 3V battery.
  - Step 7: See again the displayed values in the LCD display.
  - Step 8: See if the blue LED turned on.
  - Step 9: Reset the system by typing "RESET" in PuTTY.

- Test Scenario #8
  - Step 1: In PuTTY type the command "VIEW_3V".
  - Step 2: See if the blue LED turned on.
  - Step 3: Connect alligator clips to the 9V battery.
  - Step 4: See the displayed values in the LCD display.
  - Step 5: See if the yellow LED turned on.
  - Step 6: Connect alligator clips to the 1.5V battery.
  - Step 7: See again the displayed values in the LCD display.
  - Step 8: See if the green LED turned on.
  - Step 9: Reset the system by typing "RESET" in PuTTY.

5.3 Expected and Observed Results

This section should include the the expected and actual results of each test.

- Test Scenario #1

  - Expected Result: A battery voltage and percentage for the 9V battery is shown in the LCD display. The yellow LED turned on, and the other LEDs are off.
  - Actual Result: Both the battery percentage and voltage where shown in the LCD display. Voltage read: 8.8V — Charge read: 97.6%. The yellow LED turned on, and the other ones stayed off.

- Test Scenario #2
  - Expected Result: A battery voltage and percentage for the 1.5V battery is shown in the LCD display. The green LED turned on, and the other LEDs are off.
  - Actual Result: Both the battery percentage and voltage where shown in the LCD display. Voltage read: 1.5V — Charge read: 99.9%. The green LED turned on, and the other ones stayed off.

- Test Scenario #3
  - Expected Result: A battery voltage and percentage for the 3V battery is shown in the LCD display. The blue LED turned on, and the other LEDs are off.
  - Actual Result: Both the battery percentage and voltage where shown in the LCD display. Voltage read: 2.9V — Charge read: 99.5%. The green LED turned on, and the other ones stayed off.

- Test Scenario #4
  - Expected Result: The whole system resets when the command "RESET" is typed in PuTTY. All LEDs turn off and LCD display clears.
  - Actual Result: The system was reset after typing the appropriate command. The LEDs turned off, and the LCD was cleared.

- Test Scenario #5
  - Expected Result: Nothing will be displayed if measurement is done before writing a command. LEDs should not turn on.

- Actual Result: After measuring all batteries before sending any command through PuTTY, no voltage or charge was displayed. LEDs did not turn on.

- Test Scenario #6

  - Expected Result: Invalid values will display in the LCD when attempting to measure batteries that were not intended for the "VIEW_9V" command. Yellow LED should always stay on.

  - Actual Result: After typing the "VIEW_9V" command in PuTTY and measuring other batteries instead of the 9V battery, unexpected and inconclusive values appeared. The yellow LED stayed on always.

- Test Scenario #7

  - Expected Result: Invalid values will display in the LCD when attempting to measure batteries that were not intended for the "VIEW_1.5V" command. Green LED should always stay on.

  - Actual Result: After typing the "VIEW_1.5V" command in PuTTY and measuring other batteries instead of the 1.5V battery, unexpected and inconclusive values appeared. The green LED stayed on always.

- Test Scenario #8

  - Expected Result: Invalid values will display in the LCD when attempting to measure batteries that were not intended for the "VIEW_3V" command. Blue LED should always stay on.

  - Actual Result: After typing the "VIEW_3V" command in PuTTY and measuring other batteries instead of the 3V battery, unexpected and inconclusive values appeared. The blue LED stayed on always.

6 Code

6.1 Code for main.c

This section of the code is the core of the Battery Voltage Meter system, running inside the while loop. It is responsible for calculating the battery voltage and percentage, and displaying the results on the LCD screen.

Since the system supports three types of batteries: 1.5V, 3V, and 9V, with the user selecting the battery type using the "VIEW_1.5V", "VIEW_3V", or "VIEW_9V" commands. Based on the selected battery, the code calculates the voltage and percentage and updates the display.

This part of the code also ensures that only one battery type is measured at a time, clearing the display when switching between types. Additionally, the reset functionality allows users to clear the display and reset the system using the "RESET" command.

I like this code the most because it is central to the system's operation. It handles all the measurements, display updates, making it the backbone of the project. It is also the part I am most proud of since I wrote most of the functions and variables involved.

```
#include "string.h"  // library for C strings
#include <stdbool.h> // library for bool data type
#include <stdio.h>

    // If the view 9v battery message was displayed
    if (ninev_battery)
    {
      // Calculate the percentage
      battery_percentage = ninev_percentage(analog_measurement);

      // Correct bounds of battery percentage
      battery_percentage = percentage_bounds(battery_percentage);

```

```
14      // Calculate the voltage
15      battery_voltage = ninev_voltage(battery_percentage);
16
17      // if the other types were activated
18      if (one_fivev_battery || threev_battery)
19      {
20        // Clear LCD and turn off other battery types
21        lcd_clear();
22        one_fivev_battery = false;
23        threev_battery = false;
24
25      } // End of if
26
27      // Show the voltage on LCD
28        display_voltage(battery_voltage);
29      // Show the charge percentage on LCD
30      display_charge(battery_percentage);
31
32    } // End of if
33
34    // If the view 1.5v battery message was displayed
35    if (one_fivev_battery)
36    {
37      // Calculate the percentage
38      battery_percentage = onefivev_percentage(analog_measurement);
39
40      // Correct bounds of battery percentage
41      battery_percentage = percentage_bounds(battery_percentage);
42
43      // Calculate the voltage
44      battery_voltage = onefivev_voltage(battery_percentage);
45
46      // if the other types were activated
47      if (ninev_battery || threev_battery)
48      {
49        // Clear LCD and turn off other battery types
```

```
50          lcd_clear();
51          ninev_battery = false;
52          threev_battery = false;
53
54        } // End of if
55
56        // SHow the voltage on LCD
57          display_voltage(battery_voltage);
58        // Show the charge percentage on LCD
59        display_charge(battery_percentage);
60
61      } // End of if
62
63      // If the view 9v battery message was displayed
64      if (threev_battery)
65      {
66        // Calculate the percentage
67        battery_percentage = threev_percentage(analog_measurement);
68
69        // Correct bounds of battery percentage
70        battery_percentage = percentage_bounds(battery_percentage);
71
72        // Calculate the voltage
73        battery_voltage = threev_voltage(battery_percentage);
74
75        // if the other types were activated
76        if (one_fivev_battery || ninev_battery)
77        {
78          // Clear LCD and turn off other battery types
79          lcd_clear();
80          one_fivev_battery = false;
81          ninev_battery = false;
82
83        } // End of if
84
85        // Show the voltage on LCD
```

```
86          display_voltage(battery_voltage);
87        // Show the charge percentage on LCD
88        display_charge(battery_percentage);
89
90      } // End of if
91
92      // If the reset sequence was activated
93      if(reset_system)
94      {
95        // Clear the screen
96        lcd_clear();
97
98        // Stop reset sequence
99        reset_system = false;
100
101     } // End of if
```

7 Conclusion

In this project, I developed a Battery Voltage Meter system using an STM32 microcontroller to measure the voltage and charge percentage of 1.5V, 3V, and 9V batteries. Through this, I learned how to better implement the ADC (Analog-to-Digital Converter) to read analog voltage levels and convert them into digital values, then display the results on an LCD screen. I also integrated UART communication to send the commands from a laptop.

One of the most useful parts of this lab was learning how to manage multiple inputs (different battery types) and ensure that only one type of battery was measured at a time. I had to deal with some challenges, like handling voltage scaling and preventing errors when measuring higher voltages (like the 9V battery). I also had some issues with handling invalid inputs, but I found that managing the state of the system (like clearing the display and updating LEDs based on the command input) was essential for ensuring the system behaved as expected.

The part I struggled most with was the calculations for battery percentage and voltage. Initially, I had trouble figuring out how to correctly calculate the voltage from the ADC readings, particularly for the 1.5V and 3V batteries. To solve this, I met with a tutor who helped me understand the correct way to approach the calculations. The tutor introduced me to the formula:

Expected Voltage / Reference Voltage = X / 4095

In this equation, I needed to solve for X to find the appropriate ADC values that correspond to the voltages for the 1.5V and 3V batteries. The tutor helped me first work through the 9V battery calculation, which gave me a better understanding of how to apply the formula for the other batteries. This was a turning point for me in mastering how to read voltage from the ADC properly.

This project builds on the concepts I've already learned in previous labs about microcontroller programming and interfacing with peripherals like displays. It's a good stepping stone

for me in terms of applying theory to real-world applications, like battery management, and voltage monitoring, which is something I can definitely see myself working on in the future, whether in consumer electronics or other projects that require voltage monitoring.

I'm proud of how it turned out, especially the real-time updates on the display and the clear feedback provided to the user through the LEDs. This lab has definitely helped me gain a deeper understanding of embedded systems and how to create useful tools for practical problems.

References