Project Artefact Report: Battery Capacity Tester

Contents

INTRODUCTION

Background And Problem Statement

My project focuses on addressing the challenge of assessing the capacity of used 18650 batteries from electronic devices, such as laptops, power banks and flashlights. Currently, when we acquire these batteries, whether from recycled electronics or other sources, we lack accurate information about their remaining capacity. This uncertainty poses a significant problem as we cannot efficiently utilize these batteries in new applications without knowing their true capacity. By addressing this issue, we can reduce the risk of them ending up in landfill.

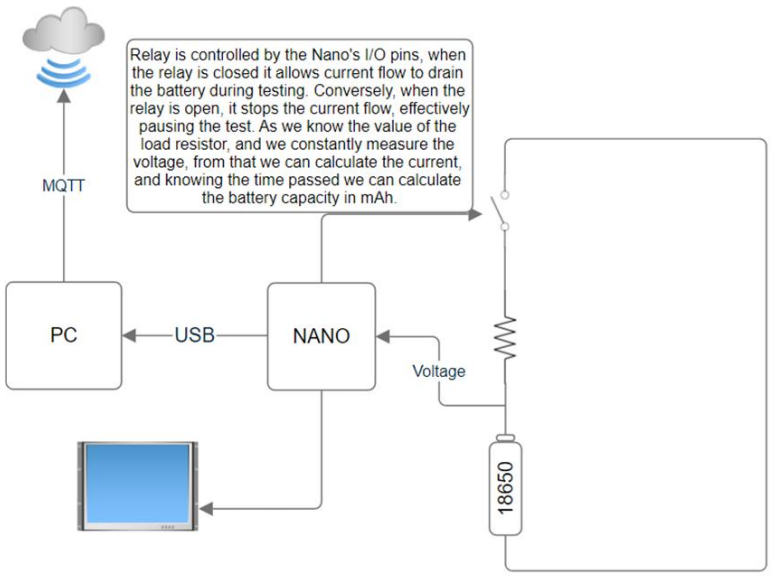
Existing Work/Traditional Testing Methods

Traditional battery capacity testing methods involve manual processes and often lack real-time monitoring capabilities. These methods often require significant time and effort to obtain accurate results. Some existing solutions include:

1. **Multimeter Testing –** Time consuming, unable to integrate with another exclusive device and prone to human error.
2. **Dedicated Battery Testers –** Expensive and might not offer real-time monitoring or integration with other systems.
3. **Commercial Solutions –** Highly expensive making them inaccessible for individual DIY projects or for day-to-day use.

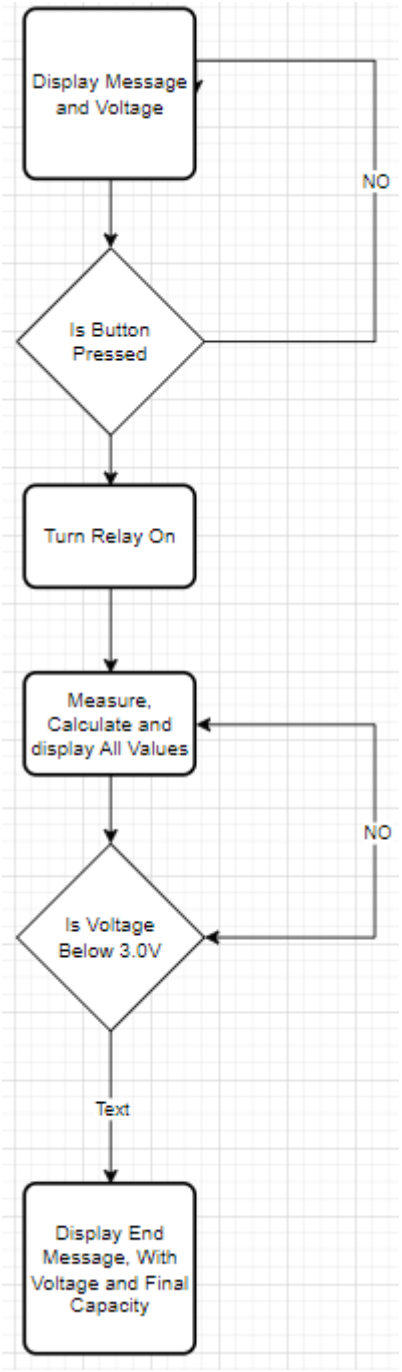
My project and the use of embedded systems can fill this gap by continuously monitoring key battery parameters such as voltage and current. Real-time data collection allows for immediate feedback and analysis, enabling proactive maintenance and optimization of battery performance.

Original Block Diagram Plan



In my finished product instead of transmitting data from the Nano's to my PC over USB, I utilized the Nano's Wi-Fi capabilities by transmitting the voltage drop over MQTT (Message Queuing Telemetry Transport) to ThingSpeak.

Flowchart Diagram



Requirements/Plan

1. **Sensing**
   - Use the Arduino Nano's analogue-to-digital converters (ADC) to measure battery voltage accurately.
   - Incorporate battery sensors to monitor key parameters such as voltage and current in real time.
2. **Computation**
   - Develop a program to process the sensed data (Voltage) and determine battery capacity
   - Manage the testing process by controlling relays to toggle the load on and off during testing.
3. **Communication**
   - Use the MQTT protocol to upload battery data to ThingSpeak to monitor declining voltage on a graph and to the IFTTT server for notifying userd when test is complete.
4. **User Interface**
   - Incorporate an intuitive graphical user interface on the OLED display to show real-time battery parameters and testing progress.
   - Allow interactive controls via push buttons to start and stop tests
5. **Cost-Effectiveness**
   - Develop a solution that is affordable and accessible for individual DIY projects and hobbyists.
   - Utilize readily available components and open-source software to minimize costs.

By meeting these requirements, the project aims to provide an efficient, cost-effective, and user-friendly solution.

Design Principles

Modularity

The system is divided into distinct components, each handling a specific function such as reading voltage, updating the display, communicating with the internet, and managing user input. This implementation enhanced maintainability and scalability, allowing individual modules to be updated or replaced without impacting the entire system

Reliability

Reliability is another critical principle, ensuring the system consistently performs its functions. This includes accurate voltage readings and maintaining a stable Wi-Fi connection, which are crucial for obtaining trustworthy test results. The system provides a user-friendly interface through an OLED display that offers real-time feedback, showing voltage, current, capacity, and elapsed time in a clear and concise manner

Fault Tolerance

To enhance robustness, fault tolerance is incorporated through error handling and recovery mechanisms, such as Wi-Fi reconnection logic and timeouts for network requests. This fault tolerance ensures the system can recover from transient error without human intervention.

Prototype Architecture

The prototype architecture of the battery capacity test system is designed to manage key functions, including user interaction, data acquisition, data processing, and remote data logging.

The microcontroller processes the raw sensor data to determine the battery's capacity and updates the display with real-time information. Every 60 seconds, the microcontroller sends the latest readings to ThingSpeak for remote logging. When the battery voltage drops below the predefined threshold of 3.0V, the system turns off the relay, sends a notification via IFTTT, and displays the final results continuously after the test is completed.

System Testing and Evaluation

This part of my report outlines the methodologies and processes used in evaluating a voltage monitoring system designed to test battery performance.

Testing with Potentiometer

The Potentiometer was used to stimulate different load conditions safely during initial testing phases. By adjusting the potentiometer, I could finely tune the resistance and observe how the system behaved under different scenarios. This controlled adjustment helped ensure the system operated safely and within the Arduino's input voltage

Using a Voltage Divider for Safe Battery Testing

The Arduino 33 IoT microcontroller can only read voltages up to 3.3V. To read higher voltages, such as those from a typical 18650 battery (range up to 4.2V when fully charged), a voltage divider has been implemented. A voltage divider reduces the input voltage to a level that can be safely read by the microcontroller.

The Voltage divider consisted of two resistors configured in series. The battery voltage was applied across the entire series, and the microcontroller read the voltage across one of the resistors. This configuration allowed the higher battery voltage to be scaled down to with the 0-3V range acceptable for the Arduino's ADC.

Obtaining Digital Values and Calculating Actual Voltage

The system's microcontroller reads the analogue voltage from the voltage divider using its ADC, which converts this analogue voltage into a digital value ranging from 0 to 1023. This digital value is essential for calculating the actual battery voltage.

The function `readBattVolt` reads the analogue value from the battery connected to the analogue pin (A1) and it converts this digital value to a proportional voltage by dividing it by factor (313). The `calcVoltage` function then doubled this divided voltage, yielding the actual voltage. This process ensured that the voltage displayed was an accurate representation of the battery's actual voltage.

Integration with ThingSpeak for Data Logging

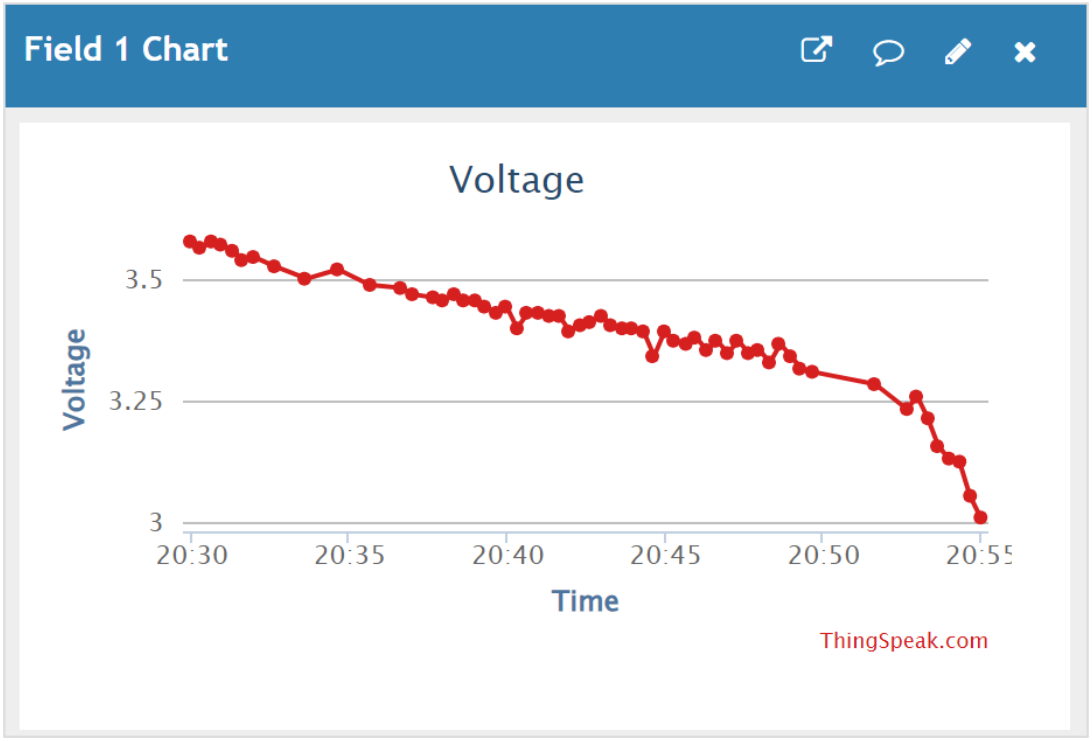
ThingSpeak has been utilized to log and monitor the voltage readings remotely. This IoT analytics platform allows continuous data collection, facilitating long term analysis and trend monitoring.

The microcontroller is connected to a Wi-Fi network, and it sends voltage data to ThingSpeak at regular intervals.

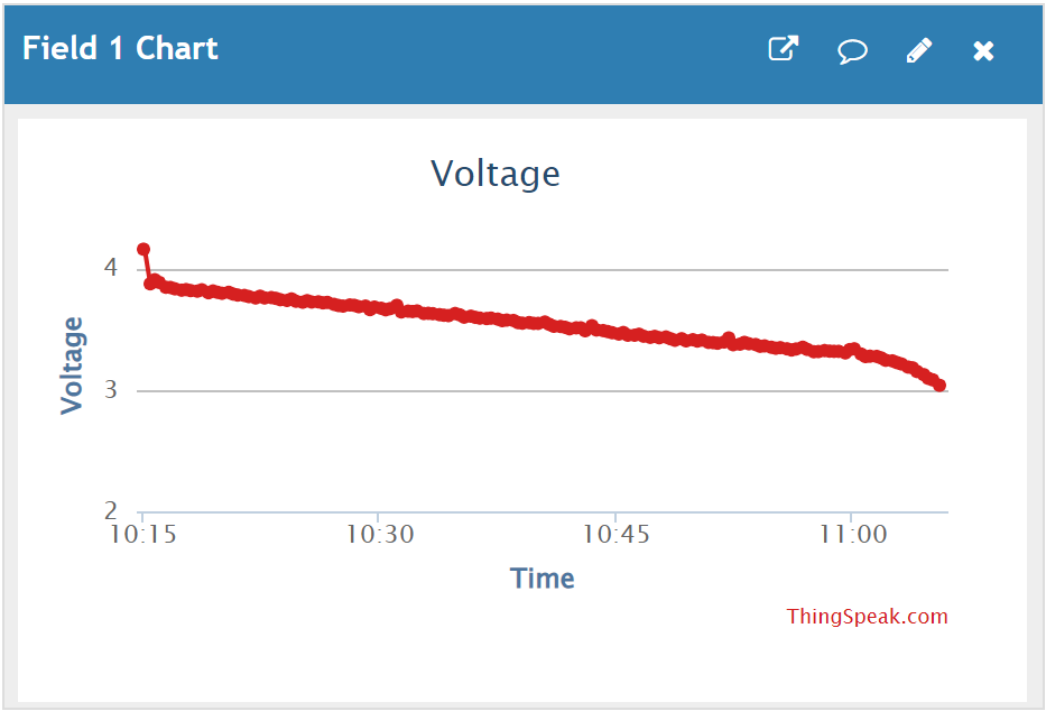
To ensure I sent the voltage values to the designated ThingSpeak channel I had to use corresponding channel ID and write API key. By regularly updating ThingSpeak it ensured accurate readings and allowed users/clients to visualize the data through graphs.

Improving Accuracy by Averaging Multiple Readings

In my initial implementation of measuring the voltage, I relied on a single reading to determine the battery's voltage, I found that this approach was susceptible to inaccuracies due to electrical noise and transient fluctuations, as seen below.



To address this issue, I revised the `calcVoltage` function to take multiple readings (four in total) and average them. By averaging four readings, the function effectively smooths out the noise and random fluctuations, providing a more precise estimate of the true voltage. This adjustment ensures that each measurement is more reliable and reflective of the actual battery condition, enhancing the overall performance and dependability of the system.



User Manual

Installation

1. **Hardware Setup:**
   - Connect 18650 battery to the Arduino Nano via the ADC pins
   - Connect relays to the Nano's I/O pins and the load (resistors) to the relays
   - Connect the OLED display and push buttons to the Nano
   - Ensure all connections are secure and power is supplied to the Arduino Nano 33 IoT
2. **Software Setup:**
   - Install the Arduino IDE on your PC
   - Install and include the necessary libraries (WiFiNINA, ThingSpeak, Wire and Adafruit_GFX and Adafruit_SSD1306)
   - Upload the Arduino code to the nano

Using the Embedded Solution

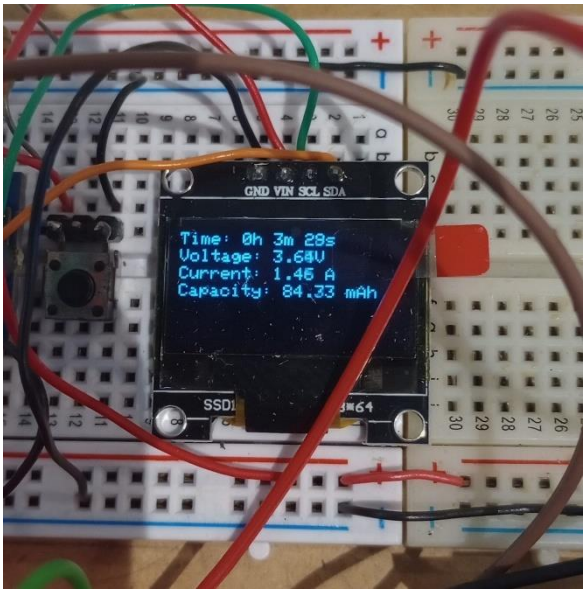
1. **Starting a Test**
   - Press the push button on the display to initiate the battery testing process
   - The LCD will show real-time battery parameters and testing progress
2. **Remote Monitoring**
   - Ensure your PC is connected to the local network and can communicate with the Arduino Nano
   - Monitor the decreasing battery voltage via ThingSpeak.
   - Once complete you will be notified through the IFTTT.

Conclusion

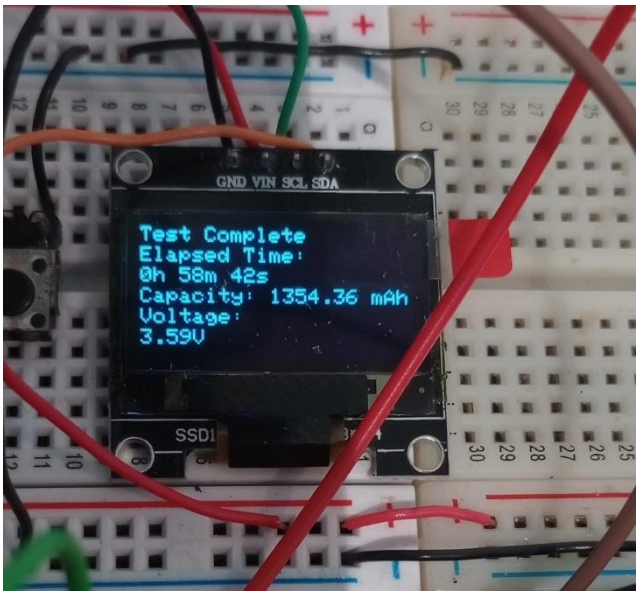
Working on this battery capacity tester project was a challenging yet rewarding experience. One of the major issues I faced was ensuring the accuracy of the voltage reading while managing the constraints of the Arduino's input limits. Additionally, maintaining a stable Wi-Fi connection fot continuous data logging posed some challenges due to intermittent connectivity issues. If given a second chance to work on this project. I would explore more robust networking solutions and perhaps integrate additional sensors for a more comprehensive battery health analysis. I would also focus on refining the user interface to provide even more intuitive controls and better visual feedback. Overall, this project has deepened my understanding of embedded systems and IoT applications, providing valuable insights into practical problem-solving and system design.

Project Videos

1. https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=ac4a1786-d40b-41da-8b09-b17d000dde8d



2. https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=f9796f9d-3fa8-424b-8360-b17d001c7e6e



GitHub Repository

https://github.com/ezza2000/BatteryCapacityProject