CAI/AEC 2025
Programming

Solar Energy Control System

L. Daigle, M. Hutchinson, L. MacLennan, C. White

January 26, 2025

Meet the Team

Lauren Daigle
Project Manager

Matthew Hutchinson
Data Acquisition
and Debugging

Liam MacLennan
Networking Lead

Colby White
Mechatronics Lead

Background

CAI AEC

- In the face of climate change, **deep decarbonization** is necessary to satisfy **increasing energy demands**.
- Solar and wind are **low-cost solutions,** however**, wind is unpredictable**
- Solar is more reliable, and work can be done to **optimize the energy output**

Problem Statement

CAI AEC

Participants must design a **client-server application** to **monitor and control** the operations of a **solar power plant** equipped with motorized photovoltaic panels and safety mechanisms. The system must include:

1. **Data monitoring and storing**
2. **Control mechanisms**
3. **Wireless communication**

Requirements

CAI AEC

1. Data Monitoring and Storing

a. **Collect data** from sensors to **calculate the power generated**.

b. **User interface** - display the collected data.

2. Control Mechanisms

a. **Adjust the angle** of solar panels **manually** and **automatically**.

b. **Emergency shutdown** to remotely cut off the current flow to the grid.

3. Wireless Communication

a. **Store** and **display** data sent from **client** (ESP32) to **server**.

b. Errors – server must manage and reconnect.

Constraints

1. Hardware

- Only the assigned hardware can be used (ESP32 for sensor and Raspberry Pi for server) and cannot be modified.

2. System Network

- The system must be wireless and communicate through Wi-Fi. The network must be a hotspot hosted by a computer.

Specific Objectives

1 | **Read and Display Data**

-Voltage, Current, Angle

2 | **Interface Hardware**

-Manually adjust panel angle with an interface

-Fault checking (button)

-Fault LED

3 | **Automation**

-Shut off servo upon a fault and re-enable once resolved.

-Automatically track the sun angle

Specific Objectives cntd.

CAI AEC

4 | **Wireless Data Acquisition**

-Wirelessly communicate with Pi Server

-Notify on disconnect, attempt to resolve and reconnect

5 | **Web Based Interface**

-Website interface

-Graph real-time data (no refreshing)

-Interface scales to mobile devices and desktop

Design Process

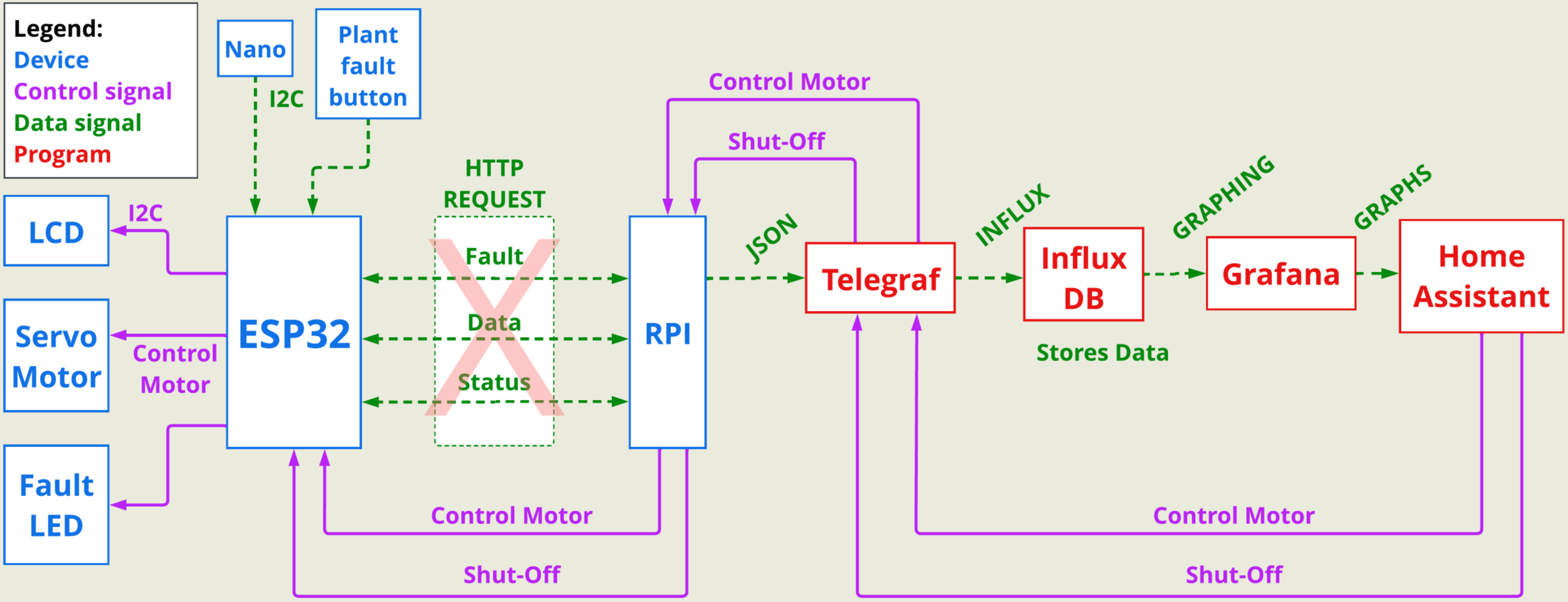
1 — Problem Definition
- Defined requirements based on prior knowledge and skills

2 — Preliminary Design
- Divided tasks based on strengths for concurrent engineering
- Scheduled check-ins to manage time and maximize productivity

3 — Iterative Design
- Prototyped and tested design
- Iterated on server communication methods

4 — Final Design
- Improved functional components

Solution



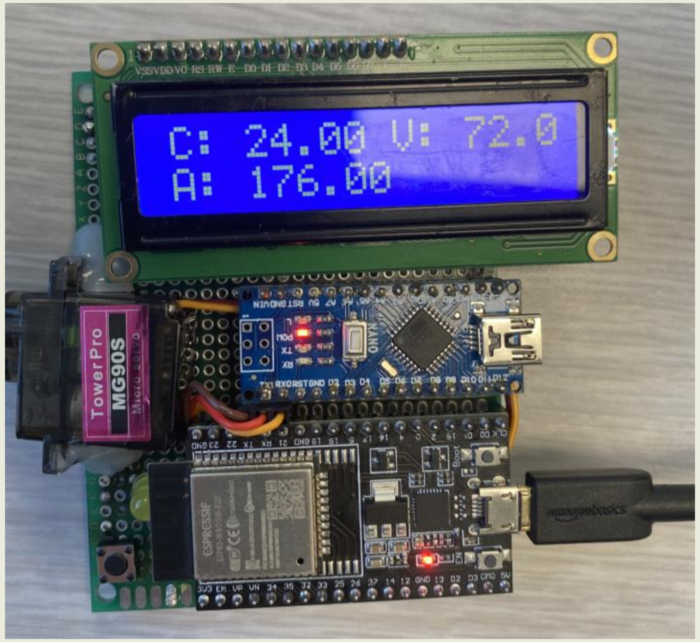
Legend:
Device
Control signal
Data signal
Program
Nano
I2C
Plant fault button
LCD
I2C
ESP32
Servo Motor
Control Motor
Fault LED
HTTP REQUEST
Fault
Data
Status
RPI
Control Motor
Shut-Off
JSON
Telegraf
INFLUX
Influx DB
Stores Data
GRAPHING
Grafana
GRAPHS
Home Assistant
Control Motor
Shut-Off
Control Motor
Shut-Off

S. Objective 1

CAI AEC

1 | Read and Display Data

-Voltage, Current, Angle



Utilized the LiquidCrystal I2C library by Frank de Barbander [1]
And the Wire library by Expressif Systems [4]

-Previous experience

-Confidence in a successful outcome

```
//read I2C data from nano for voltage, current, and angle
void readI2CData() {
  Wire.requestFrom(0x55, 32); // Request 32 bytes from I2C address 0x55
  jsonData = "";
  while (Wire.available()) {
    char c = Wire.read();
    if (c == 0xFF) break; // End of data
    jsonData += c;
  }
```

S. Objective 2

CAI AEC

2 | **Interface Hardware**

-Manually adjust panel angle with an interface

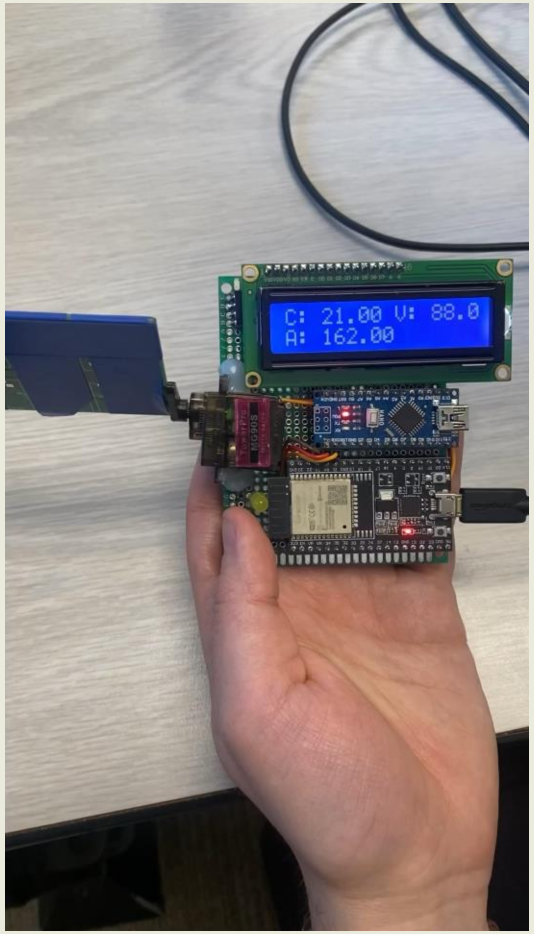
-Fault checking (button)

-Fault LED

```
int importedUserAngle = 0; // from the website
```

```
void manualControl(int userAngle) {
  //This function will take in the user's input and move the servo to that angle
  if (userAngle < minAngle) {
    myservo.write(minAngle);
  } else if (userAngle > maxAngle) {
    myservo.write(maxAngle);
  } else {
  myservo.write(userAngle);
  }
}
```

Used ESP32Servo library from Kevin Harrington and John K. Bennett [2]



S. Objective 3

CAI AEC

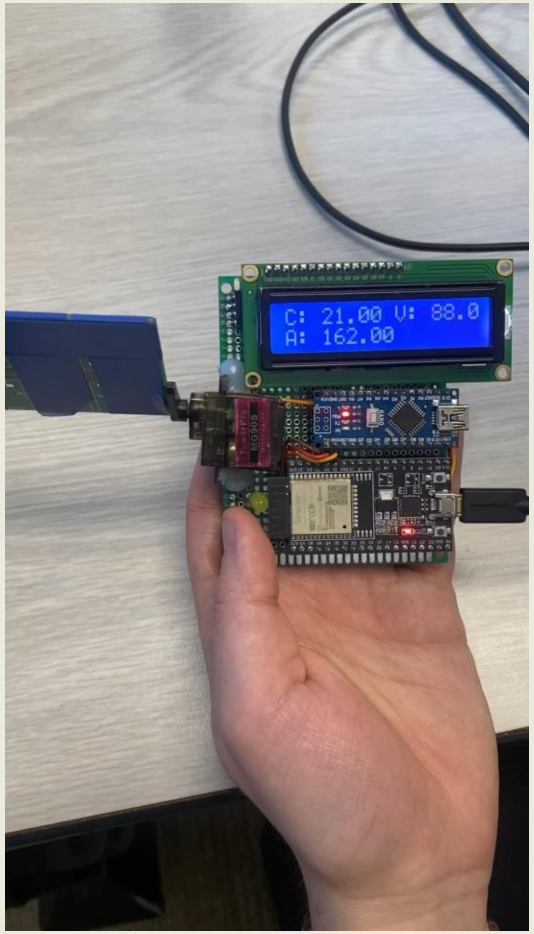
3 | **Automation**

-Shut off servo upon a fault and re-enable once resolved.

-Automatically track the sun angle

```
// Check fault status from server
void checkFaultStatus() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin("http://pi.com/plant1/fault");

    int httpCode = http.GET();
    if (httpCode > 0) {
      String payload = http.getString();
      StaticJsonDocument<200> doc;
      DeserializationError error = deserializeJson(doc, payload);
      if (error) {
        Serial.print(F("deserializeJson() failed: "));
        Serial.println(error.f_str());
        return;
      }
      fault = doc["fault"];
    } else {
      Serial.println("Error on HTTP GET request for fault status: " + String(httpCode));
    }
    http.end();
  } else {
    Serial.println("WiFi not connected");
  }
}
```



S. Objective 4

4 | Wireless Data Acquisition

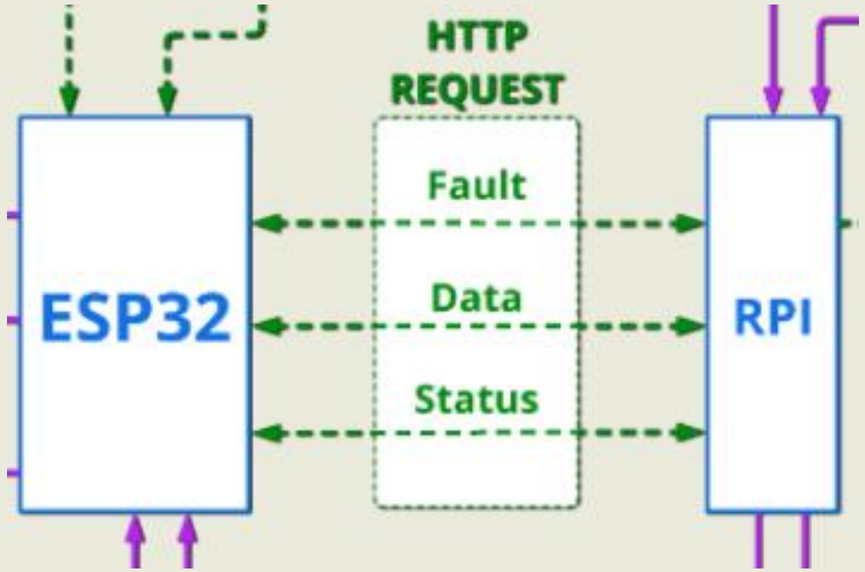
-Wirelessly communicate with Pi Server

-Notify on disconnect, attempt to resolve and reconnect

Originally focused on MQTT communication due to its flexibility in IOT applications.

Moved to HTTP/S upon discovery of Raspberry Pi Backend Issues

Unable to resolve connection issues



CAI AEC 2025

S. Objective 5

5 | **Web Based Interface**

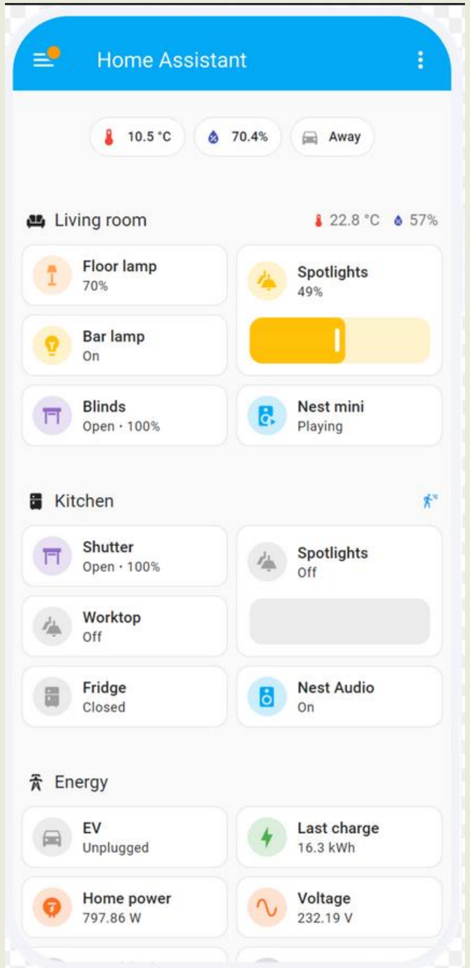
-Website interface

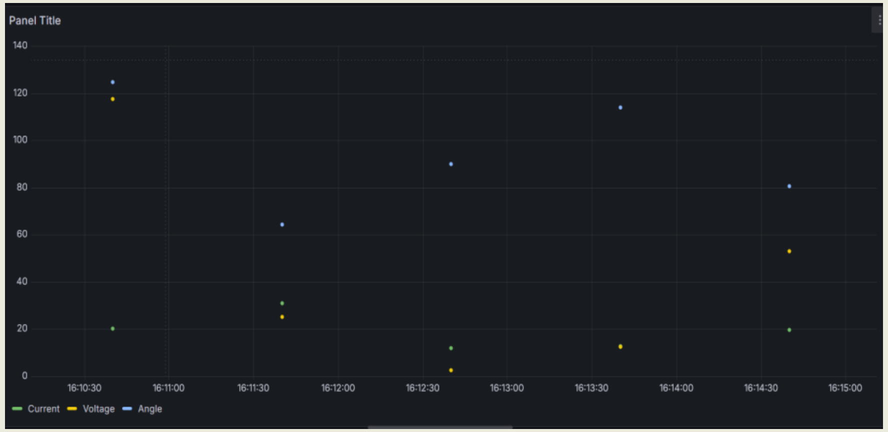
-Graph real-time data (no refreshing)

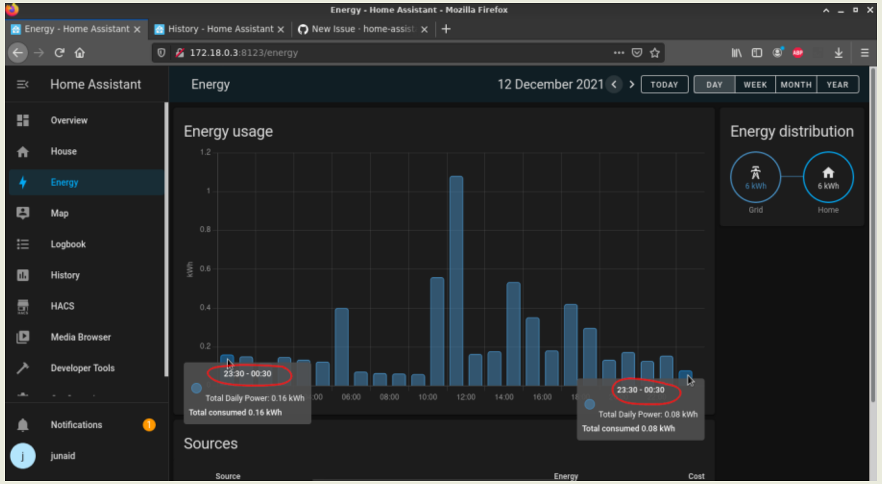
-Interface scales to mobile devices and desktop

Home Assistant

- Scales to either web clients or mobile devices
- Allows for real-time visualization of graphed data
- Allows for an all-in-one platform for data visualization and monitoring







Testing

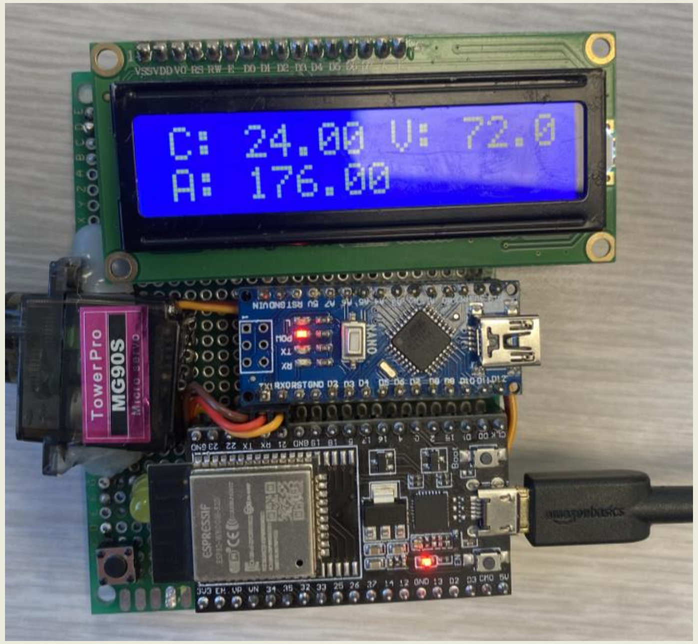
- Tried out the data reading, LCD display, and motor control. Minor adjustments to limits corrected their functionality
- Attempted multiple connection strategies, encountering persistent errors throughout. Explored various methods to successfully broadcast the data, but challenges persisted.
- Generated random test data for database validation and visualization within the platform

Errors and Critiques

CAI AEC

Server side (Raspberry Pi):

- Unable to resolve peer connection refusal by telegraf
- Unable to implement front end graphing of real-time data
- Unable to implement front-end fault reset and manual override from front end.



Solar Energy Control System

L. Daigle, M. Hutchinson, L. MacLennan, C. White

References

[1] LiquidCrystal_I2C: https://github.com/johnrickman/LiquidCrystal_I2C (Author: johnrickman)

[2] ESP32Servo: https://github.com/madhephaestus/ESP32Servo (Author: madhephaestus)

[3] ArduinoJson: https://github.com/bblanchon/ArduinoJson (Author: Benoît Blanchon)

[4] Wire: https://github.com/espressif/arduino-esp32/tree/master/libraries/Wire (Author: Espressif Systems)

[5] WiFi: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi (Author: Espressif Systems)

[6] HTTPClient: https://github.com/espressif/arduino-esp32/tree/master/libraries/HTTPClient (Author: Espressif Systems)

[7] WiFiClientSecure: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFiClientSecure (Author: Espressif Systems)

[8] GitHub Copilot: Used for autocompletion assistance