energies

MDPI

Article

An Open-Source Supervisory Control and Data Acquisition Architecture for Photovoltaic System Monitoring Using ESP32, Banana Pi M4, and Node-RED

Wei He, Mirza Jabbar Aziz Baig * and Mohammad Tariq Iqbal *

Department of Electrical and Computer Engineering, Memorial University of Newfoundland, 230 Elizabeth Ave, St. John's, NL A1C 5S7, Canada; weih@mun.ca

* Correspondence: mjabaig@mun.ca (M.J.A.B.); tariq@mun.ca (M.T.I.)

Abstract: To overcome the issues of the existing properties and the non-configurable supervisory control and data acquisition (SCADA) architecture, this paper proposes an IoT-centered open-source SCADA system for monitoring photovoltaic (PV) systems. The system consists of three voltage sensors and three current sensors for data accumulation from the PV panel, the battery, and the load. As a part of the system design, a relay is used that controls the load remotely. An ESP32-E microcontroller transmits the collected data to a Banana Pi M4 Berry (BPI-M4 Berry) through the Message Queuing Telemetry Transport (MQTT) protocol over a privately established communication channel using Wi-Fi. The ESP32-E is configured as the MQTT publisher and the BPI-M4 Berry serves as the MQTT broker. Locally installed on the BPI-M4 Berry, the Node-RED platform creates highly customizable dashboards as human–machine interfaces (HMIs) to achieve real-time monitoring of the PV system. The proposed system was successfully tested to collect the PV system voltage/current/power data and to control the load in a supervisory way under a laboratory setup. The complete SCADA architecture details and test results for the PV system data during the total eclipse on 8 April 2024 and another day are presented in this paper.

Keywords: renewable energy; PV system; SCADA; Node-RED; Internet of Things; MTU; RTU; MQTT

Citation: He, W.; Baig, M.J.A.; Iqbal, M.T. An Open-Source Supervisory Control and Data Acquisition Architecture for Photovoltaic System Monitoring Using ESP32, Banana Pi M4, and Node-RED. *Energies* **2024**, *17*, 2295. https://doi.org/10.3390/en17102295

Academic Editor: Oscar Barambones

Received: 25 April 2024
Revised: 5 May 2024
Accepted: 8 May 2024
Published: 10 May 2024

1. Introduction

To monitor and control industrial processes in different sectors, such as power generation systems, agriculture [1], and biogas [2], supervisory control and data acquisition (SCADA) systems have been developed. Explicitly in the case of photovoltaic (PV) systems, the SCADA systems can optimize PV system performance [3]. The main components of a SCADA system incorporate field instrumentation devices (FIDs), remote terminal units (RTUs), the main SCADA server or master terminal unit (MTU), and human–machine interface (HMI) devices. FIDs are used as a part of the SCADA system to collect data that are critical for use in industrial production, such as current, pressure, temperature, and voltage. FIDs and RTUs can communicate over wireless or wired communication protocols, resulting in efficient data communication. In general, RTUs are physically placed in distributed regions for data communication and processing. In addition to serving the purpose of communication with MTUs, RTUs can control FIDs as well. In contrast, MTUs are at a central location and serve the purpose of data communication, data storage, and hosting the main server.

The architecture of the SCADA system has gone through a significant change since its first utilization. Monolithic SCADA, as the first generation, represents a standalone framework where the RTUs are connected to the MTU via wide area networks (WANs). RTUs and MTUs, if not manufactured by the same vendor, often cannot communicate with each other due to proprietary communication protocols. Distributed SCADA stands for a SCADA system that uses a local area network (LAN) to achieve communication between

each operation station and uses WANs between the RTUs and the corresponding operation stations. This architecture facilitates system miniaturization by assigning each operation station different functions. For example, an operation station that solely exchanges data with RTUs or provides the HMI to the operator can be made smaller. Ref. [4] Networked SCADA lifts the vendor limitation by adopting standard communication protocols, e.g., Ethernet and Transmission Control Protocol (TCP)/Internet Protocol (IP). IoT SCADA, as the fourth generation, introduces IoT technology and cloud services into the traditional SCADA field, promoting the modernization of industry or Industry 4.0 [5]. Unlike networked SCADA, more diverse devices can be utilized in IoT SCADA, such as smart sensors. Moreover, cloud services, including storage, servers, software, and computing can be integrated in IoT SCADA. Applications of IoT in the industry context, including digital twins and the prediction of system failures, have advanced industrial automation to a large extent [6]. Conventional SCADA systems come with high cost and high power consumption [7], leaving a challenge for small-scale and cost-sensitive PV systems. Additionally, scalable and flexible SCADA systems are gaining attention in response to the development of the renewable energy industry. Renewable resources can be abundant in remote areas, where the integration of IoT adds to the information flow. Moreover, the high cost of proprietary SCADA systems prevents the massive adoption of small-scale PV installations, while low-cost solutions can be proposed by using IoT architectures. The adoption of the Internet of Things (IoT) in SCADA systems utilizes connected devices and sensors via Internet Protocol (IP) networks to collect and analyze data, offering improved flexibility, scalability, and cost-effectiveness [8]. As a result of the IoT, more PV installations with different nominal capacity, from small rooftop arrays to large solar farms located in remote or urban areas, can employ scalable SCADA systems.

Message Queuing Telemetry Transport (MQTT) is a communication protocol running over TCP/IP. Based on the publish/subscribe mechanism, MQTT is a lightweight, open, and easy-to-implement solution. The MQTT broker separates the publisher from the subscriber, routing messages by topic [9]. This architecture encourages the integration of low-bandwidth devices in the IoT context, where numerous RTUs and MTUs with limited processing power can be present [10]. Prevalent in the IoT field, MQTT in IoT-based SCADA systems can be achieved by Node-RED, an open-source programming tool. With the flow-based interface, developers can quickly connect devices and create interactive dashboards to display monitored variables [11]. The reduced complexity makes Node-RED a practical option for efficiently handling data transmission via MQTT and visualization in real-life scenarios.

As a part of this study, an open-source SCADA system based on the IoT for monitoring a PV system has been proposed. Voltage sensors and current sensors are specified as the F031-06 0–25 V voltage sensor module and the ACS712 current sensor module. Monitored voltages and currents are first collected by these analog sensors and the connected RTU ESP32 module, and then data are transmitted to the MTU Banana Pi M4 Berry via Wi-Fi connection. By utilizing the open-source programming tool Node-RED running on the MTU, our solution for PV system monitoring can be easily replicated by developers. The adoption of the MQTT protocol between the RTU and the MTU requires minimum bandwidth while ensuring reliable and real-time communication within the SCADA system. The designed SCADA system incorporates low-cost components such as FIDs, RTUs, and MTUs, making the proposed system cost-effective as a whole. To add to this, the proposed SCADA system is experimentally validated, which ensures the reliability of this work.

The following is the outline of the rest of this paper. A review of the related literature is presented in Section 2. In Sections 3 and 4, the system description and the components used throughout the course of this study are introduced. In Section 5, the implementation methodology is covered. In Section 6, the experimental setup and results are demonstrated. Section 7 highlights the discussion part of this paper. The conclusion of this paper is provided in Section 8. Limitations and the future work are described in Section 9.

2. Related Work

The authors of [12] designed a remote monitoring and control system for a PV-based water-pumping system. For field data acquisition, a light-dependent resistor (LDR) module to measure the sunlight and an ultrasonic sensor to measure the water tank level are connected to an Arduino UNO R3. Also equipped with an SIM800L module, global system for mobile communication (GSM) connectivity is added to the Arduino UNO R3. The Raspberry Pi 2 works as a server hosting the Node-RED platform, and the data storage unit collects sensor readings from the Arduino UNO R3 via SMS commands. Another GSM module is added to the Raspberry Pi 2 to enable the SMS updates between it and the Arduino UNO R3. Received data are processed to CSV files, including timestamps, water level readings, and the pump status. An Android application can check the PV pumping system status and turn the pump on/off over SMS and query the historic data from the Raspberry Pi 2 using the MQTT protocol via Ethernet. Their research utilizes GSM to achieve remote monitoring as a part of their work. In [13], an IoT-based wireless data acquisition and control system for PV modules was introduced. For the hardware setup, the authors used three NodeMCU Wi-Fi modules flashed with Tasmota firmware that are responsible for collecting sensor readings, servo motor control, and controlling relays for ON/OFF control. Sensors include a voltage and a current sensor, a luminosity sensor, an ambient temperature sensor, a humidity sensor, and four waterproof temperature sensors. They are all mounted on one PCB. Another hardware component is a Raspberry Pi running the Raspbian OS with software packages installed: NodeRED for data handling, InfluxDB for data storage, Grafana for data visualization, and WireGuard VPN for remote access. The overall system performance was validated in terms of data accuracy, latency times in communications, CPU computational and thermal performance benchmarks, and sampling rate. The sole focus of the study was PV module performance under open-circuit conditions, not considering the realistic performance impact from other common parts such as the charging controller, the battery, and the load.

The researchers in [14] developed a novel system to track the maximum power point (MPP) of PV panels under partial shading based on artificial vision. A wireless network node (WSN) powered by the PV panel consists of an MCU ATmega328P, an RF transceiver ATmega128RFA1, an external ADC, a battery, and a buck-boost converter. The coordinator node is an ATmega128RFA1 microcontroller that receives sensor readings from or sends the control signal to the WSN. The coordinator node connects with a webcam and a Raspberry Pi. The artificial vision algorithm requires the input voltage, the input current of the DC-DC converter, the ambient temperature, and the webcam images of the PV panel to generate the reference voltage of the DC-DC converter. The duty cycle control signal can also be generated and sent to the WSN via the coordinator node. The Raspberry Pi runs the artificial vision algorithm, obtains the reference voltage for the MPP, and calculates the duty cycle of the DC-DC converter. The authors claim their remote monitoring and control system has efficiency of more than 99%, even under partial shading conditions.

An open-source SCADA system to monitor a PV panel and a backup battery was designed in [15]. The microcontroller ESP32 Thing measures and collects the PV panel voltage, the PV panel current, and the backup battery voltage through the connected sensors. The sensor readings are then sent to the Thinger.IO local server running on the Raspberry Pi 2 via a local Wi-Fi network for saving data, monitoring in real time and controlling remotely. HMI dashboards are created over the Thinger.IO server to facilitate the monitoring and supervisory control. Two system configurations were tested, where the Raspberry Pi was connected to either Ethernet or the LAN port of a local Wi-Fi router, with the latter being an industrial network. This work adopts the modern SCADA architecture, which is IoT-based. The authors of this study claim their work is low-cost, low-power, and open-source. As a part of [16], an IoT platform for online monitoring of renewable energy systems was developed. A grid-connected PV station in the Saharan environment was taken as the case study to validate the effectiveness of the proposed system. An LA-25NP current sensor and an LV-25P voltage sensor, as the FIDs, are connected to an ESP32 (RTU) with Wi-Fi module.

An ADSL Wi-Fi router as the gateway achieves communication between the ESP32 and the main server through the MQTT protocol. A website displays the monitored data in graphic form, and a MySQL database manages and stores the collected data. According to the researchers, this work is low-cost, IoT-based, and suitable for harsh environments, such as the desert. An industrial IoT system with redundant network architecture was presented in [17] to remotely monitor a solar plant. The computing module of a Raspberry Pi 4 combined with a Waveshare board acts as the MTU and the RTU at the same time. Collected data from the pyranometer and environmental sensors are transmitted to the MTU via RS485 for Modbus communication. An SIM7600G-H-PCIE module resolves the cellular communication from the MTU to the operation center by deploying VPN tunnels. Irradiance, temperature, generated power, and process flow are presented by Grafana visualization. In this study, the authors suggest that resilience can be ensured by using a redundant mechanism in which the backup node takes over the master role when the master node is unreachable.

Another study [18] proposes an IoT-based system for the monitoring of a PV fuel cell system. The DC power output from the combination of the PV array and the fuel cell was monitored by sampling the voltage and current. An ATMega2560 as the RTU first collected the monitored voltage and current, and then sent the data to the NodeMCU ESP8266 module through serial communication. Finally, a ThingSpeak platform received the data from the ESP8266 and displayed them graphically. This work enables the real-time and remote monitoring for a PV fuel cell system that can be applied in residential areas. As reported in [19], the authors used supervisory control and data acquisition (SCADA) data to check the soundness of the blades following a lightning strike in order to increase up-time by quickly resuming operations in response to a number of reports about blade damage following lightning strikes on wind turbines. In [20], the authors present a framework for developing an Intrusion Detection System (IDS) for SCADA-based power systems using machine learning, which incorporates effective modeling methods, such as data preprocessing, data augmentation, automated feature selection, rigorous training, and testing. For the purpose of substantiating our proposed design framework, we used a publicly available ORNL (Oak Ridge National Laboratory) dataset.

An approach to apply SCADA systems in industrial control systems is presented in [21]. A multilayer architecture for SCADA control is presented, along with aspects of interoperability and interconnectivity within the architecture reference models, as well as research opportunities and challenges arising from the recent rapid increase in industrial control system complexity and digital transformation. In this study, the authors investigate the issue of proprietary SCADA systems and demonstrate how SCADA quality requirements are related to new technology adoption in steel manufacturing. In important infrastructure such as power, telecommunications, transportation, and manufacturing plants, supervisory control and data acquisition (SCADA) systems provide monitoring and control. The authors of [22] believe that the SCADA system operating in connection with the Internet is vulnerable compared to isolated SCADA systems. As a result, the security of SCADA systems is gaining attention. In addition, they discuss some high-impact security incidents, objectives, and threats. The authors in [23] use SCADA to connect a lower central controller to an upper WEB monitoring system, which is the hub of a microgrid intelligent monitoring platform. Using Java as middleware, the designed system provides communication and control functions between the central controller and the upper monitoring system. As part of the microgrid's security and stability, the SCADA system executes real-time data acquisition, storage, load balancing, resource recovery, and security processing simultaneously. The authors in [24] utilized a SCADA system for development of energy management systems for intelligent buildings. The design system integrates information data regarding illumination, temperature, and ventilation, which is important in modern building design technology. In this study, they used PLC controllers that can be controlled by the central SCADA system. The authors of [25] developed a SCADA system for data logging and control of a solar backup power system for a grid connected system with

an annual capacity of 516.75 MWH. The system is designed for a village in Lebanon for irrigation and domestic use.

The authors of [26] proposed a highly customizable SCADA platform based on the industrial IoT (IIoT) concept. Modbus TCP protocol was used to achieve communication between the PLC and Node-RED, while the MQTT protocol was used for communication between Node-RED and two MQTT clients (MQTT.fx in Windows OS and MQTTClient in iOS). Experimental results indicate that (1) the PLC simulated by Mod_RSsim can successfully publish the status code of the motors to the Node-RED dashboard via Modbus TCP and (2) the two MQTT clients can successfully publish and subscribe to the Node-RED dashboard via MQTT. This paper contributed to the conceptualization of applying IoT in a SCADA system, providing a framework for future practical scenarios. A Node-RED-based SCADA architecture for a hybrid power system was implemented in [27]. One voltage sensor and five current sensors collected the hybrid power system parameters, sending them to the Node-RED platform running on the Windows OS via a USB cable from an Arduino Mega2560. A Wio terminal was also utilized to display the monitored parameters. This work serves as an alternative to implementing the SCADA system without IoT architecture. Furthermore, this work focused on monitoring more than control in SCADA implementation.

During the course of this study, a considerable amount of literature has been reviewed, and some of the useful parts have been summarized in the above section of this article. According to the literature reviewed and to the best of authors knowledge, no such SCADA system has been identified that uses a Banana Pi M4 Berry to host a Node-RED-based IoT server and uses the MQTT protocol over a private network for remote monitoring. The designed SCADA system during the course of this study is the first of its kind, with the following key contributions.

- The proposed open-source SCADA system utilizes the latest IoT architecture configured over a Banana Pi M4 Berry that provides all remote monitoring and control capabilities essential for the operation of PV systems in remote locations.
- Node-RED, an open-source platform, and the commercially available components utilized in the system design facilitate the scalability of the system, including the choice of communication channels and the number of sensors.
- A two-step load cut-off mechanism is provided in the designed system: (1) An operator of the SCADA system can manually shut down the relay with just a tap using the intuitive user interface (UI) when necessary. (2) When the monitored battery voltage falls below a pre-set threshold, the relay will automatically turn off, featuring over-discharge protection of the battery.

3. System Description

An illustration of the proposed SCADA system can be found in this section of the paper. Figure 1 provides an overview of the SCADA system design presented in the article. An ESP32-E (FireBeetle 2 ESP32-E, DFRobot®, Shanghai, China) serves as the RTU in the system design; on the other hand, a Banana Pi M4 Berry (BPI-M4 Berry, Guangdong Bipai Technology®, Dongguan, China) acts as the MTU. In the designed SCADA system, the ESP32-E is used for data collection from the FIDs, where voltage sensors are connected in parallel arrangements, while the current sensors are connected in serial settings to the PV panel, battery, and load. A control signal is generated by the ESP32-E when the battery voltage falls below the threshold level, thereby turning off the relay and the load. As a part of system design, we have used a Wi-Fi router to establish a private network, enabling wireless communication between the ESP32-E and the BPI-M4 Berry. Under the MQTT protocol, the ESP32-E publishes messages with different topics; meanwhile, the BPI-M4 Berry subscribes to the same topics, receiving the messages in real time. The Node-RED-based local server configured on the BPI-M4 Berry communicates with the ESP32-E and stores the data received using FIDs on the BPI-M4 Berry's local storage, which

is an 8G eMMC flash. It is also accessible for data visualization over UI by browsing https://localhost:1880/.

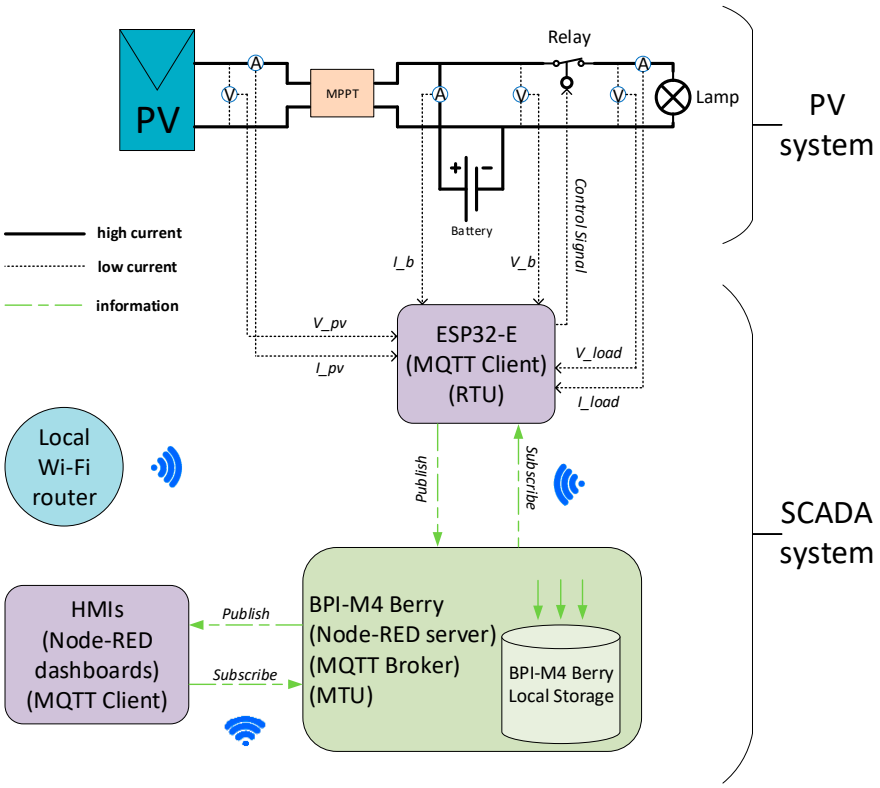

Figure 1. Overview of proposed SCADA design.

4. SCADA System Components

This section of the paper describes the hardware and software components used as part of the system design. It includes a Banana Pi (BPI) M4 Berry, which serves as the master terminal unit (MTU), ESP32-E remote terminal unit (RTU), and Node-RED, which, as a part of the system, is used to develop a locally accessible UI for the proposed SCADA system and to implement the MQTT protocol. Furthermore, the system also includes three voltage sensors and three current sensors for measuring voltage and current, respectively.

4.1. Banana Pi M4 Berry

The BPI-M4 Berry development board is a single-board computer (SBC) powered by the Allwinner H618 System-on-Chip (SoC). Figure 2 shows the physical representation of the BPI-M4 Berry. Compared to the well-regarded Raspberry Pi 4b, the BPI-M4 Berry is more powerful and commonly available at a price much lower than that of the Raspberry Pi 4b. The BPI-M4 Berry has a full HDMI connector and M.2 PCIe 2.0 slot, features missing in other single-board computers. Equipped with a comparable CPU capacity, it also includes 2 GB of LPDDR4 memory, 8 GB of embedded Multi-Media Controller (eMMC) storage, and integrated Wi-Fi and Bluetooth capabilities. The Banana Pi M4 Berry also offers a Gigabit Ethernet (GbE) RJ45 connector, a 40-pin header, and four USB ports. The specifications of the BPI-M4 Berry are listed in [28]. With its features, the BPI-M4 Berry board supports a broad range of applications, including media processing, the Internet of Things (IoT), and entertainment sectors. As a part of the proposed SCADA system design, we used the BPI-M4 Berry as the MTU to host the main data server over a locally established Wi-Fi network, establishing communication with the ESP32-E, and to store the collected sensor readings locally.

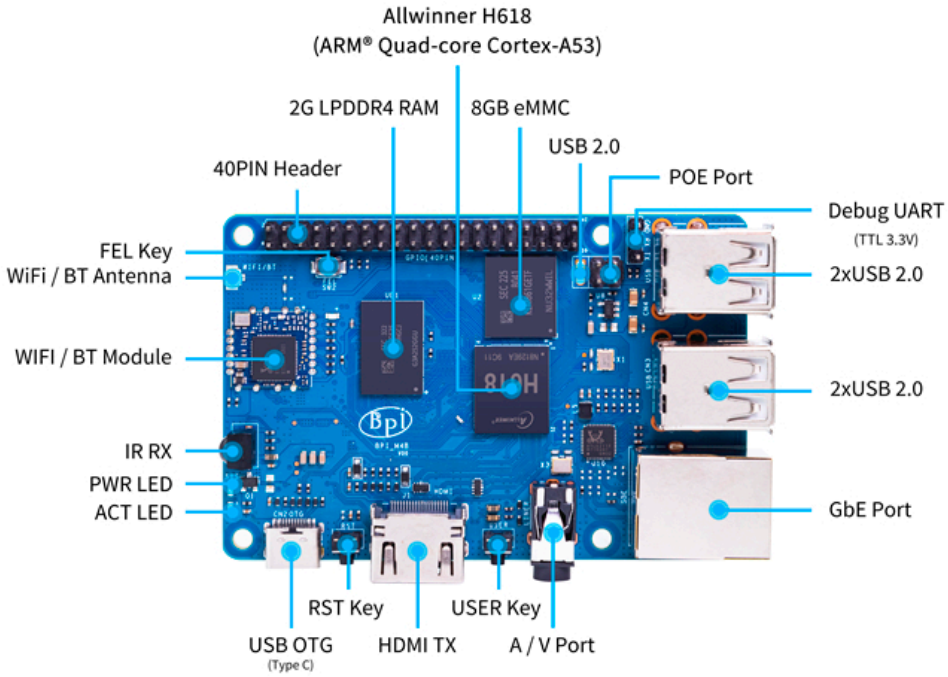

Figure 2. BPI-M4 Berry [28].

4.2. ESP32-E

In the ESP32 series, the FireBeetle 2 ESP32-E is a competent micro-controller board using an ESP-WROOM-32E module, featuring the dual-core processor architecture and Wi-Fi and Bluetooth communications [29]. Its compact size, high energy efficiency, and inclusion of an integrated charging circuit are advantages for multiple applications such as smarthomes and industrial Internet of Things (IoT) solutions. The significant factors contributing to its selection for this research are the capability of wireless communication and the availability of analog pins. Figure 3 presents the pinout of the FireBeetle 2 ESP32-E.

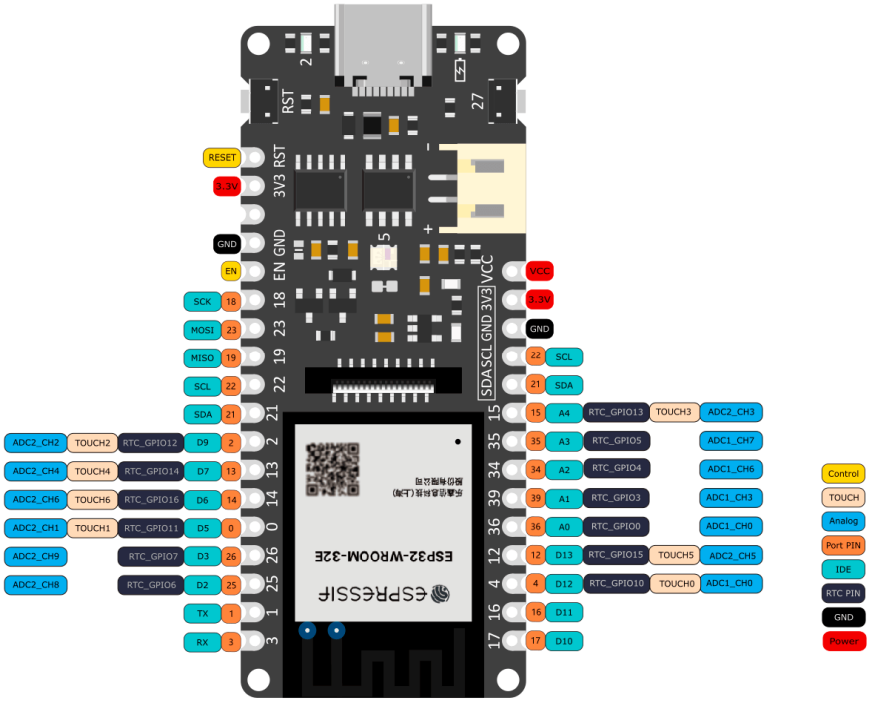

Figure 3. Pinout of the FireBeetle 2 ESP32-E [29].

4.3. Node-RED

Node-RED (Version 3.1.5) epitomizes a paradigm of flow-based programming, a concept emphasizing a methodology for delineating an application's functionality through a network of encapsulated units or "nodes" [30]. Figure 4 shows the edit dialog for the MQTT configuration node. Originating from IBM's Emerging Technology Services team and presently under the stewardship of the OpenJS Foundation, Node-RED serves as an ideal tool that enables the construction of applications as a series of interconnected nodes. Each node is tasked with a specific function: receiving data, performing an operation on these data, and then forwarding the processed data onwards. This architecture not only facilitates the consistent flow of data across the network but enhances the accessibility and comprehension for more customers. The inherent design of Node-RED, with its emphasis on visual representation, significantly lowers the barrier to entry for users across different skill levels. Through the course of this study, we used Node-RED for easy implementation of the MQTT protocol by Mosquitto Broker (Version 2.0.18) to transfer data and create dashboards that enable real-time monitoring and control.

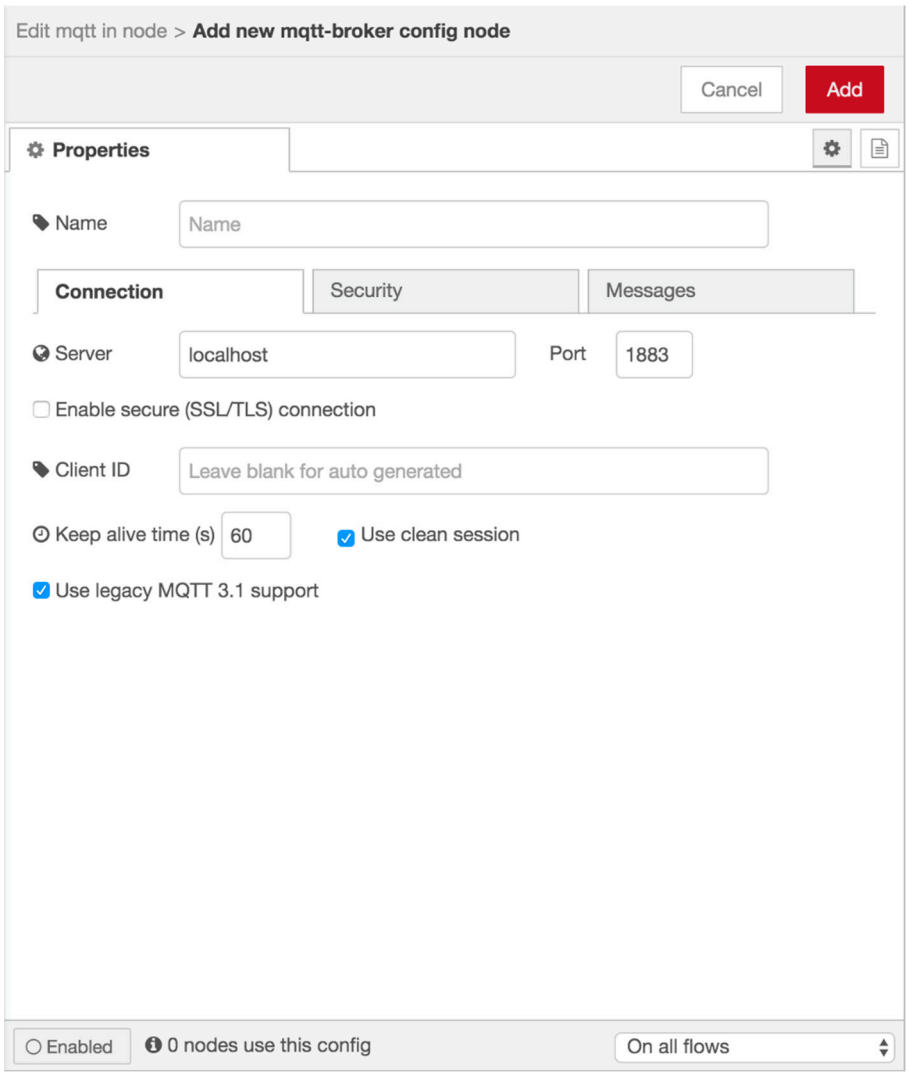

Figure 4. The Node-RED MQTT node configuration interface [30].

4.4. Voltage Sensor

The voltage sensor (voltage detection module, HiLetgo®, Shenzhen, China), capable of measuring DC voltages from 0 to 25 V, develops a resistive voltage divider strategy for its operational methodology. Figure 5 depicts the actual image of the voltage sensor.

The module interfaces with the voltage source through VCC and GND terminals. It incorporates a series configuration of two resistors with values of 30 K Ohms and 7.5 K Ohms, respectively. The output terminal, denoted as "S", is positioned between these resistors and is intended for connection to an analog input pin on the ESP32-E development board. The output terminal marked "-" aligns with the voltage source's ground, establishing a common ground with the development board. The division ratio of 7.5 K Ohms to the cumulative resistance (7.5 K + 30 K Ohms) results in a fraction of one-fifth, indicating that the voltage read by the ESP32-E represents a fifth of the actual voltage. Consequently, this requires an adjustment within the programming to ensure the accuracy of the voltage measurements [15]. We used voltage sensors in this paper to measure the voltages within the PV system.

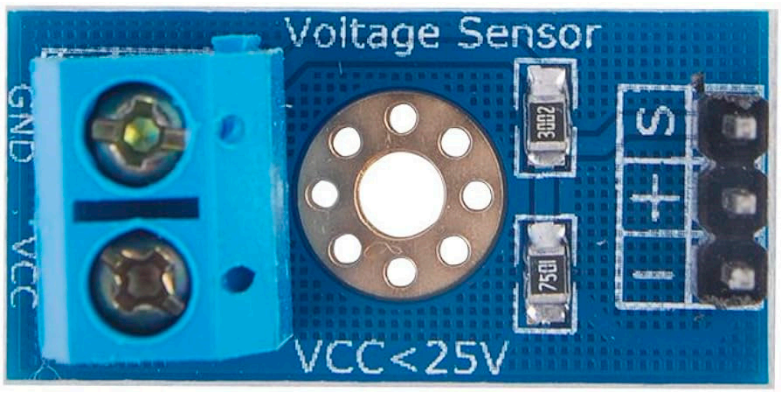

Figure 5. Voltage sensor illustration, 0–25 V [15].

4.5. Current Sensor

Based on the Hall effect, the ACS712 (ACS712ELCTR-30A-T, Allegro Microsystems®, Manchester, NH, USA), as a cost-effective and precise current sensing solution, is suitable for a broad spectrum of applications within both the industrial and commercial sectors. Figure 6 provides a visual representation of the ACS712 current sensor, which is capable of accommodating currents up to ±20 A. For implementations involving the ESP32—characterized by a nominal voltage of 3.3 volts—it is worth noting that the Vcc and ground pin of the ACS712 sensor interface with a voltage source supplying 5 volts and ground, respectively. This setup ensures that the ground of the external voltage source is also interconnected with the ESP32's ground, fostering a common ground system. To safely interface the ACS712's Vout with an ESP32 analog pin, a voltage divider is recommended to mitigate the risk of exposing the ESP32 to voltages exceeding 3.3 volts [31]. The ACS712 sensor is available in three variants, differentiated by their optimized current measurement ranges of ±5 A, ±20 A, and ±30 A. These variants exhibit distinct voltage–current sensitivities, spanning from 185 mV/A to 66 mV/A, with minor errors. As a part of this paper, we used the ACS712 current sensors with ±30 A to measure the current in the PV system.

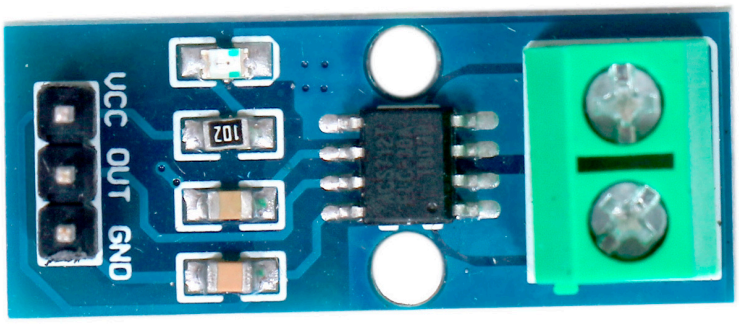

Figure 6. Icon of ACS 712 current sensor [31].

5. Implementation Methodology

In order to implement the proposed SCADA system, FIDs, including voltage and current sensors, are connected to an ESP32-E, which serves as an RTU as a part of the system design. Voltage sensors are connected in parallel with the PV panel, the battery, and the load. The current sensors are connected in series with the PV panel, the battery, and the load. Furthermore, the relay used in this study is connected in series with the load, and the components of the system are grounded properly. The Vcc pin of the current sensor is connected to the VCC pin of the ESP32-E, since the VCC pin outputs about 4.7 volts, which meets the voltage requirement of 4.5–5.5 volts by the current sensor. Table 1 presents the pin configuration of the RTU.

Table 1. Sensors and their connections with the ESP32-E.

Device #	Specification	A/D	ESP32-E Pin #
1	PV panel voltage sensor	Analog	36
2	PV panel current sensor	Analog	39
3	Battery voltage sensor	Analog	35
4	Battery current sensor	Analog	34
5	Load voltage sensor	Analog	12
6	Load current sensor	Analog	15
7	Relay	Digital	26

Figure 7 depicts the system flowchart. After the ESP32-E collects the sensor data, they are transmitted to the Node-RED platform configured on the BPI-M4 Berry using the MQTT protocol. The platform then displays the data using dashboard nodes from an installed palette named *node-red-dashboard* and stores them in the BPI-M4 Berry local 8 G flash. Algorithm 1 represents the pseudocode programmed in the ESP32-E using Arduino IDE software (Version 2.3.2).

Algorithm 1: Data acquisition, automatic control, display, and logging

Initialization;

1. ESP32-E connects to the local TCP/IP Wi-Fi network;
2. Voltage/current sensors measure voltages/currents from the PV panel, the battery, and the load;
3. ESP32-E reads sensor values on Pin 36, 39, 35, 34, 12, 15;
4. ESP32-E (MQTT publisher) sends a connect request to the Node-RED server (MQTT broker) over IP;

While *ESP32-E receives a return code of 0* **do**

5. ESP32-E publishes the sensor data to the Node-RED server over the Wi-Fi network;
6. Node-RED logs the sensor data in the local BPI-M4 Berry flash memory;
7. Display the sensor data in Node-RED dashboards, and any other devices over the Wi-Fi network;

If *the battery voltage is less than 13 volts or subscription message from Node-RED is OFF* **then**

8. Turn off the relay;

Else

9. Turn on the relay;

End

End

If *ESP32-E does not receive a return code of 0* **then**

10. Print "Attempting MQTT connection" in Arduino IDE Serial Monitor, reconnect in 5 s;

Else

11. Go to Step 1;

End

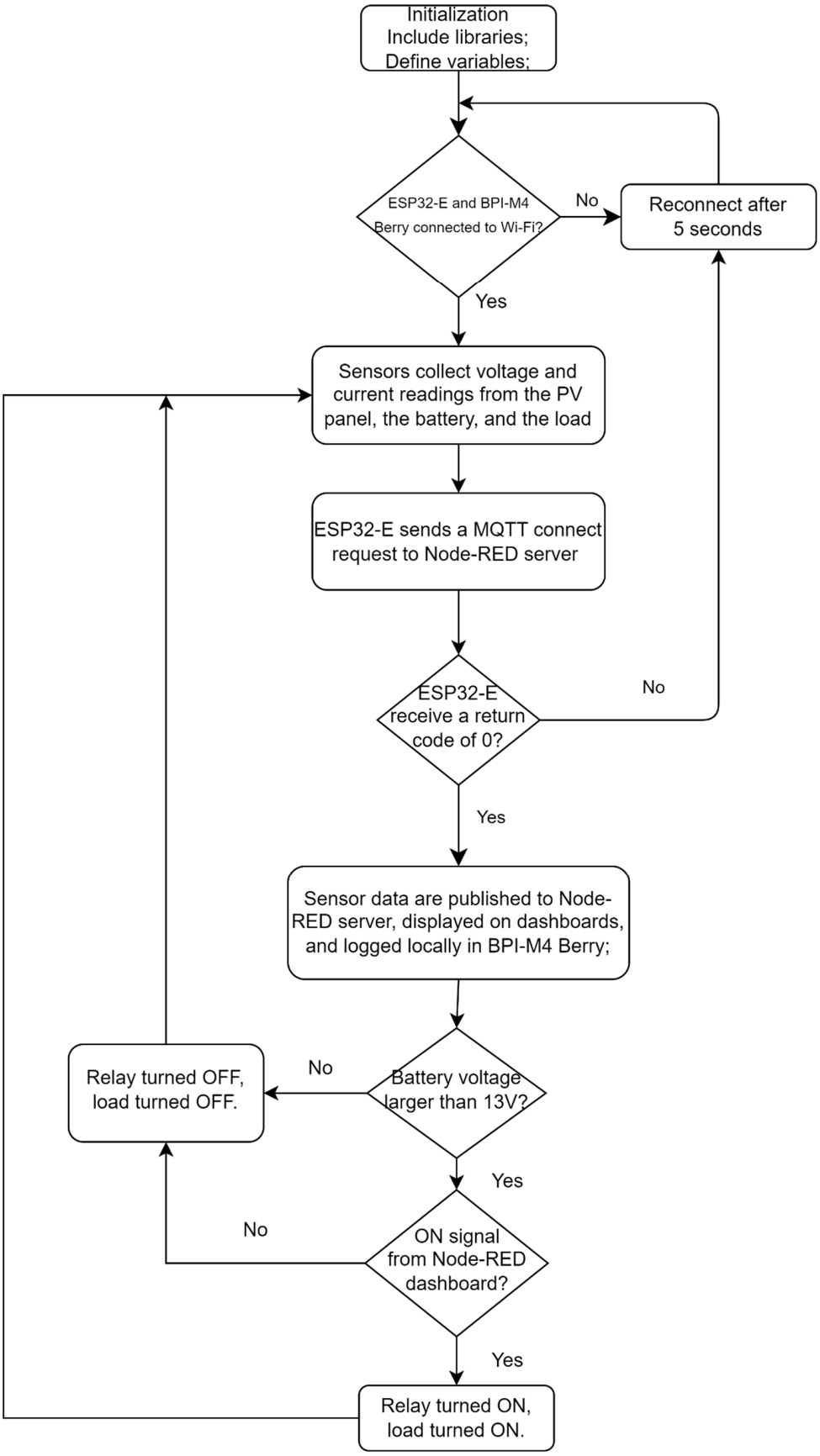

Figure 7. System flowchart.

6. Experimental Setup and Results

To evaluate the performance and capabilities of the developed open-source SCADA architecture, it has been configured to monitor solar PV metrics—current, voltage, and power—within the PV installation at the Electrical Engineering Laboratory at Memorial University. The hardware connections were built based on the system description in Section 3. Figure 8 depicts the experimental setup of the proposed SCADA system. The hardware configuration is illustrated in Figure 9, where CS stands for current sensor and VS stands for voltage sensor.

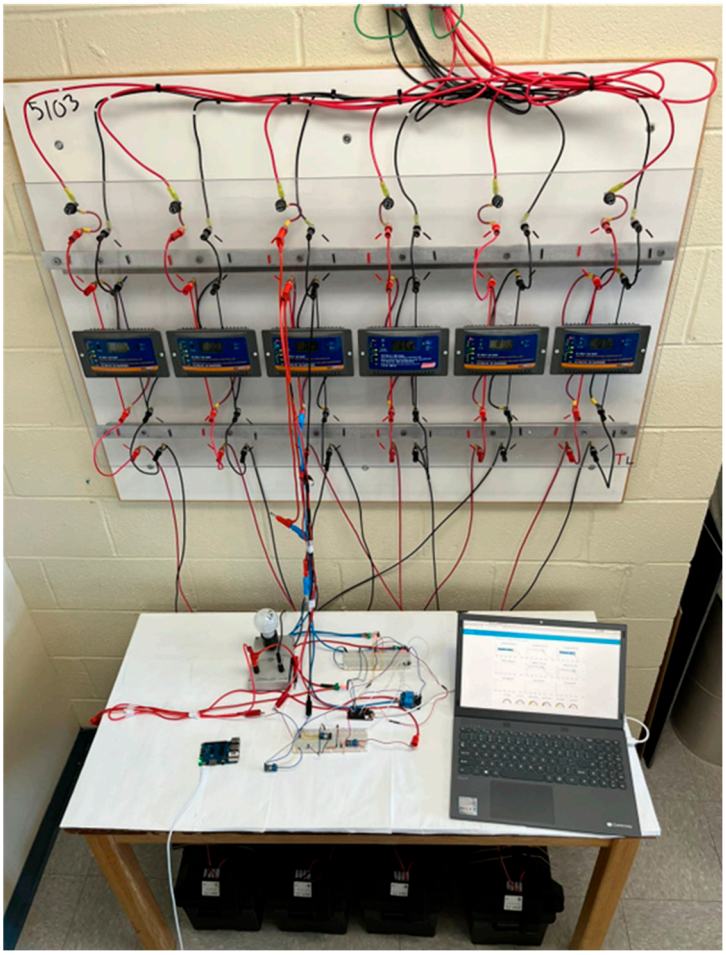

Figure 8. Experimental setup at MUN ECE laboratory.

Figure 10 represents the PV installation, which consists of 12 solar panels, spanning a collective area of 14 square meters, with each panel capable of producing approximately 130 W and 7.6 A at maximum. For the purposes of this test, the SCADA system was interfaced with two panels of the array, which consists of an output of 260 W and 15.2 A at maximum, to provide a focused assessment of the system's data acquisition and control functionalities. Additionally, the system incorporated six maximum power point tracking (MPPT) controllers and a battery bank comprising six lead–acid batteries to optimize and maintain energy efficiency.

Node-RED (MQTT Broker), which is installed on the BPI-M4 Berry machine locally and hosted on a local Wi-Fi network, is configured to receive the sensor readings collected and published by the ESP32-E microcontroller (MQTT Client). Using the MQTT protocol, the data are transmitted in real time. In the Node-RED platform, the node flow, as shown in Figure 11, is responsible for calculating electrical power, subscribing and publishing messages, and creating HMIs (dashboards). Dashboards are built for the remote monitoring of

the PV panel data, including the PV panel voltage/current, the battery voltage/current, and the load voltage/current. The power of each component is calculated by multiplying the voltage and the current. The data trend can be easily monitored remotely. HMIs connected to the local Wi-Fi network access the dashboards by browsing the URL "192.168.x.x:1880/ui" in a web browser for real-time monitoring. 192.168.x.x should be replaced by the IP address of the BPI-M4 Berry. The Banana Pi Debian 1.0.0 OS was installed on the eMMC of the BPI-M4 Berry. During the setup, two Chrome webpages and one terminal running Node-RED were open in the OS, with the dashboard webpage on the top. CPU usage varied between 1% and 34%, corresponding to idle status (about 80% of the time) and data update status (about 20%). The memory used was around 870 MB, being 43.8% of the total available memory. The hardware resources required in this setup were notably less than the total available resources, ensuring safe and continuous system operation. Due to the volatile nature of the weather in St. John's, data vibrations were observed during the test of the proposed SCADA system. To validate the accuracy of the measured data, six digital multimeters are used to measure the PV system variables. Measurement results from the proposed SCADA system align with those of the digital multimeters, showcasing that our system accurately measures the PV system variables.

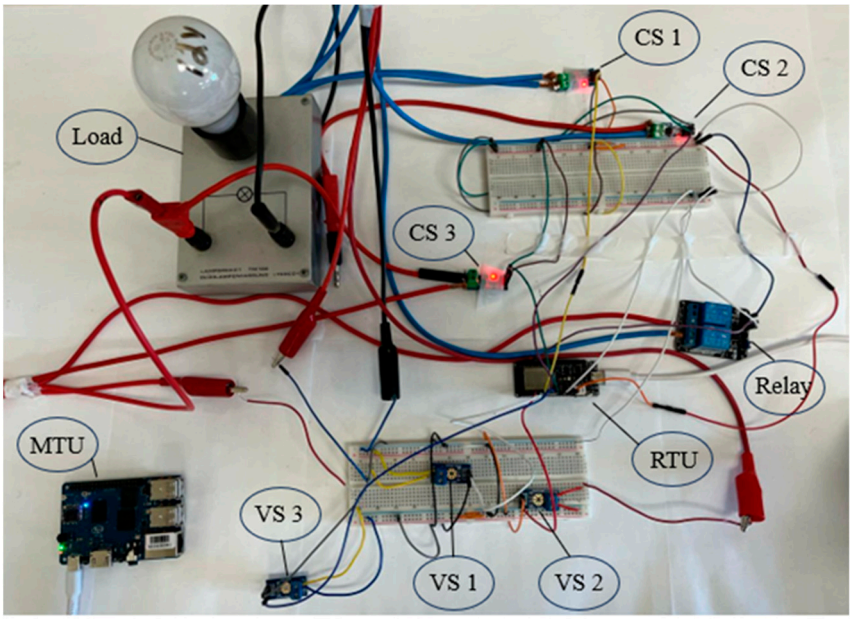

Figure 9. Hardware implementation of the proposed SCADA system.

Figure 10. MUN ECE laboratory PV installation.

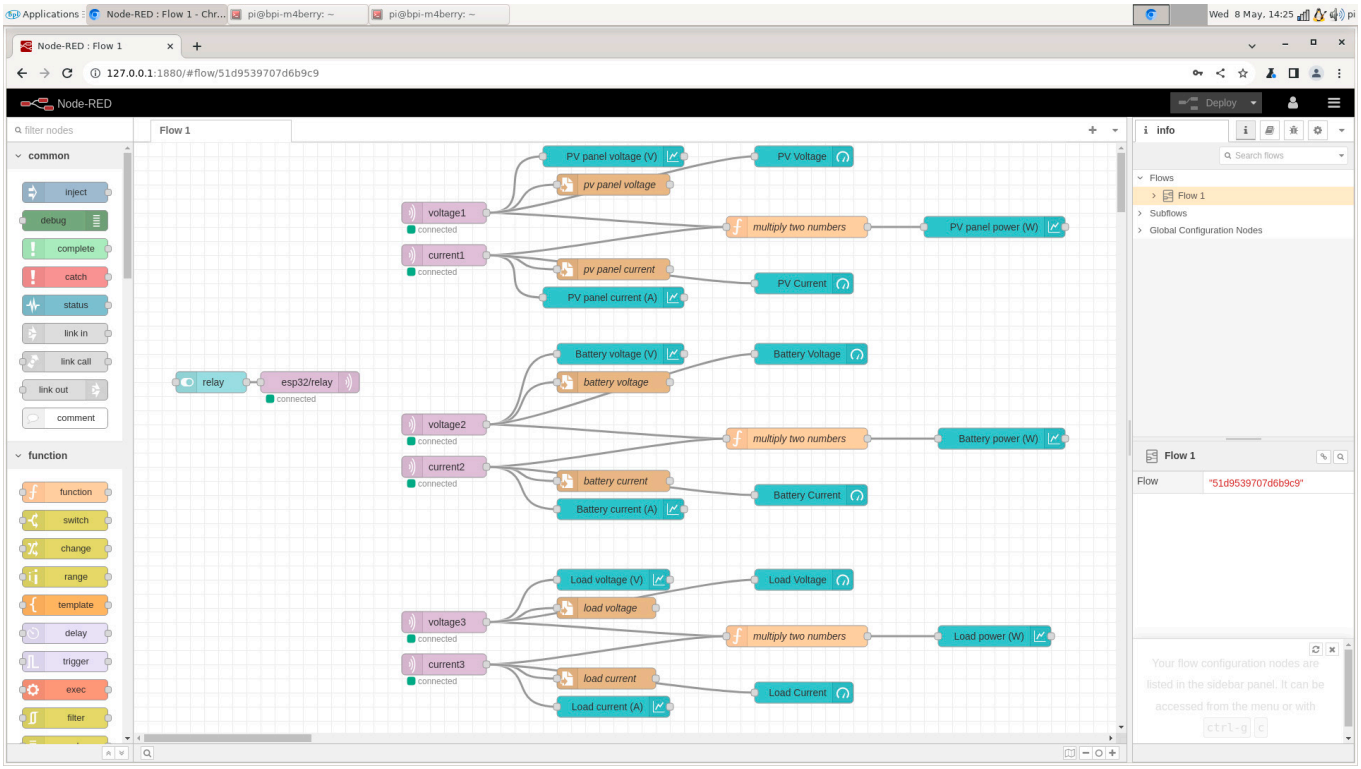

Figure 11. The developed Node-RED flow.

Figure 12 is the screenshot of the Node-RED dashboard showing the logged PV system data during a day. Starting from 10:12, the PV panel voltage fluctuates between 14.1 V and 22.5 V. The PV current remained steady with a smaller fluctuation between 3 A and 3.4 A. The PV panel power ranged from 38.50 W to 74.54 W. The battery voltage was kept around 14.5 V, while the battery current was about −2.8 A, indicating that the battery was charging the load. The battery power ranged from −44 W to −37 W. The load voltage was the same as the battery voltage due to the parallel circuit configuration. The load current was varying in the center of 4.88 A. The load power ranged from 67.38 W to 72.07 W. After 17:07, the PV voltage began to decrease from 21.5 V, entering a relatively stable stage ranging from 11 V to 12 V. The PV panel current dropped from 3.27 A to 0.62 A at 18:24. The PV panel power was 7.38 W at 18:24. The battery voltage also dropped to 12.59 V, since the power from the PV panel cannot compensate for the load power. The battery current increased to −4.06 A, trying to output the maximum power for the load when the battery voltage was dropping. The battery power increased to −51.16 W at 18:24 PM. The load voltage, current and power decreased to 12.59 V, 4.66 A, 58.72 W at 18:24. At 20:11, the PV voltage and the PV power were 0. The battery voltage and the load voltage were both 12.28 V. The battery current and power were −4.22 A and −51.82 W. The load power was solely from the battery.

The solar eclipse of 8 April 2024 was a total solar eclipse visible from places in Canada, the United States, and Mexico. In St. John's, NL, Canada, we monitored the solar PV panel voltage during this eclipse, as shown in Figure 13. From 17:14 to 17:16, the PV voltage decreased rapidly from 11.47 volts to 3.73 volts due to the blockage of the sunlight. After 17:16, the voltage increased to 12.27 volts at 17:18 since the Moon was no longer stopping the sunlight shed to the St. John's area. This also validates the system performance. Figure 14 is the PV output current during the eclipse. Similarly, the PV current dropped dramatically around 17:16, while it remained at a near-zero level longer than the PV voltage.

Figure 12. Node-RED dashboards showing the monitored values of the PV system.

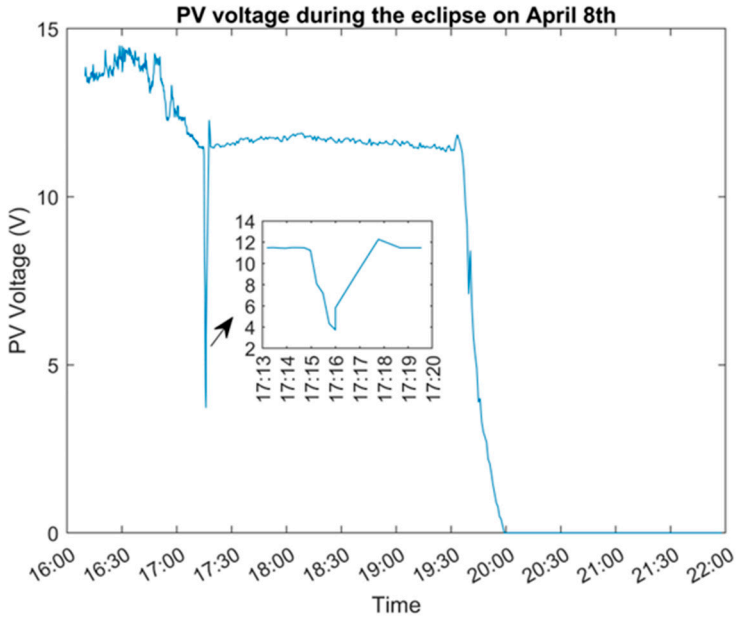

Figure 13. PV output voltage during the total solar eclipse on 8 April 2024.

Furthermore, to test the supervisory control capability of our system, a button was added to the dashboard to manually turn ON/OFF the relay and the load. A switch node was created and added to the Node-RED flow to enable supervisory control. The relay will be turned off when the switch node is set to be OFF by a simple click. Moreover, the local control of the battery was configured in Arduino IDE so that when the battery voltage is less than 13 V, the relay turns off. The relay will only remain ON when the supervisory switch is ON and the battery voltage is larger than 13 V. From Figure 15, the switch node was set to be OFF at 21:25, and subsequently both the battery current and the load current went to a value near zero since the relay and the load were turned off. Figure 16 presents the experimental results for local control. Despite the switch node being ON, the battery current and the load current were still near zero due to the functionality of local control since the battery voltage went below 13 V.

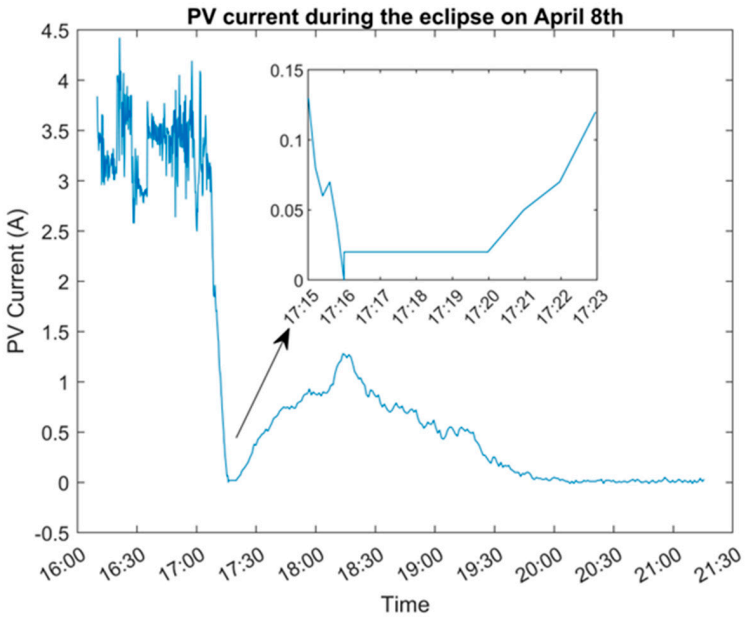

Figure 14. PV output current during the total solar eclipse on 8 April 2024.

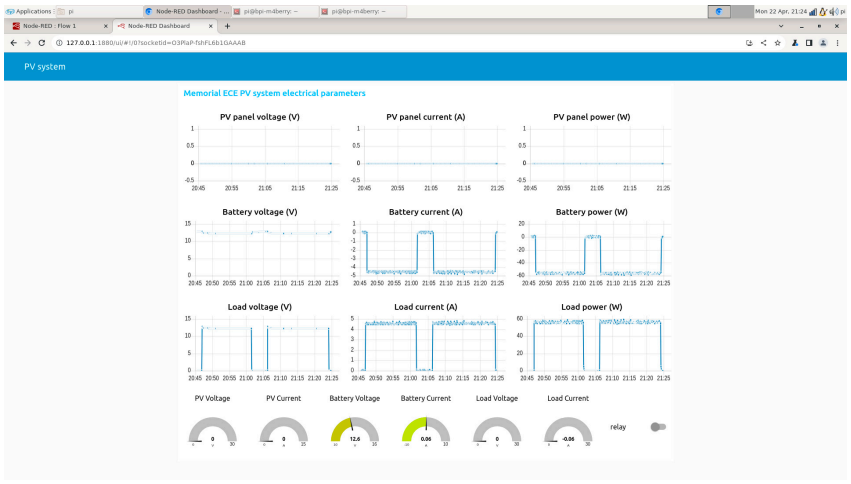

Figure 15. Test results for supervisory control.

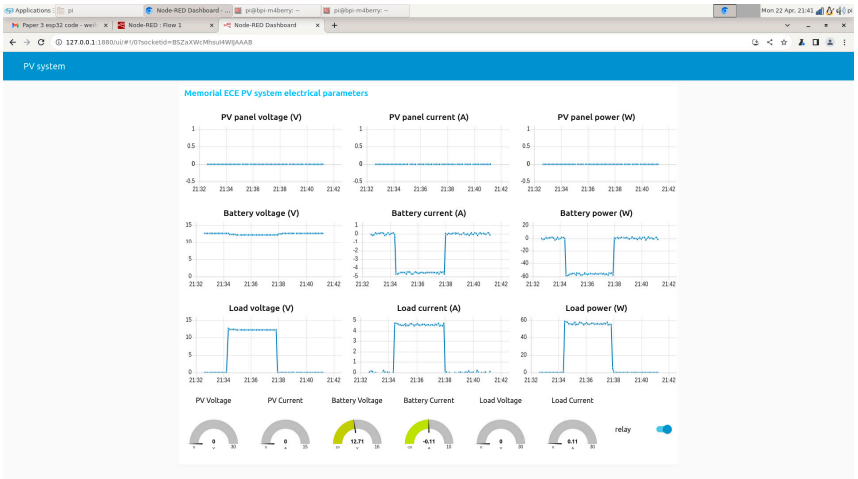

Figure 16. Test results for local control.

7. Discussion

Some of the key features of the proposed SCADA system for monitoring the PV system are outlined below based on the successful experimental results.

- **Architecture**: The proposed SCADA system is the most recent IoT-based SCADA architecture comprising crucial elements for a SCADA system. It includes FIDs, a remote terminal unit (RTU) based on an ESP32-E, a master terminal unit (MTU) configured over a Banana PI M4, Node-RED, and the MQTT protocol over a local network.
- **User Interface (UI):** The Node-RED platform is used in the development of the SCADA architecture, which simplifies the complex process of data communication over MQTT and the HMI design facilitating the easy implementation of the proposed SCADA system.
- **Remote monitoring:** The proposed IoT-based SCADA system features remote monitoring and access. The user can access the SCADA system using HMIs over the local network.
- **Supervisory and automatic control:** The SCADA system operator can remotely turn off the load whenever necessary. In addition, the load will be turned off automatically when the battery voltage is below a threshold to ensure the safety of the battery.
- **System safety:** The sensors and the relay (FIDs) are connected to the ESP32-E (RTU), separating the BPI-M4 Berry (MTU) from the PV system and thus avoiding the RTU being damaged under extreme situations.
- **System flexibility:** The RTU is wirelessly connected to the MTU, lifting the limitation of the physical position of both the RTU and the MTU.
- **Data Storage:** The proposed SCADA system features local storage based on a BPI-M4 Berry. In turn, historic data can be visualized over the UI, and these data trends are crucial for decision making.
- **Open-source and low-cost:** Utilizing the Node-RED platform and the Arduino IDE, the software used in the SCADA system is open-source and free of cost, eliminating the operational cost of the proposed design. Moreover, the hardware includes FIDs and an MTU commonly available in the local market, all at affordable prices. Table 2 breaks down the power consumption and the prices of the system components. The total system costs only CAD 94.5. The design system consumes 3.19 W on average, which is another low power consumption feature of the SCADA system. Since the SCADA system is meant to operate 24/7, power consumption is a critical component of the SCADA system.

Table 2. Itemized price and power consumption of the system components.

Components	QTY	Price (CAD)	Power Consumption (W)
BPI-M4 Berry	1	54	2.2
FireBeetle 2 ESP32-E	1	14	0.33
HiLetgo voltage detection module	3	3.5	0.006
ACS712 current sensor	3	3.5	0.064
5 V DC–250 V 10 A relay module	1	5.5	0.45
Total		94.5	3.19

8. Conclusions

Crucial operational facilities are distributed across vast areas, in often inhospitable terrains, and are increasingly reliant on a mix of traditional and renewable energy sources, including solar PV systems and wind turbines. The need for an open-source and scalable system to oversee these diverse energy assets is of paramount importance. Proprietary and non-configurable SCADA systems, while having enabled some degree of centralized control and monitoring, present significant downsides. Only limited companies produce these SCADA systems, which leads to relatively high prices and high maintenance costs. Moreover, the system compatibility of such proprietary systems with the components manufactured by other companies remains a question. Therefore, an open-source and

customizable SCADA system as a solution can reduce the dependence on specific vendors and improve system compatibility.

In this paper, based on a comprehensive arrangement of FIDs, an RTU, an MTU, and a robust SCADA communication channel (MQTT) to track and manage data flow, the designed system has shown excellent performance in laboratory settings. The system facilitates (1) real-time monitoring of essential parameters such as voltage and current for the PV panel, the battery, and the load and (2) remote and local control of the load through a network of sensors, the ESP32-E microcontroller, and the Banana Pi M4 Berry single-board computer. The PV system data during the total solar eclipse on 8 April 2024, in addition to regular validation of the system performance in real time, also testify to the reliability of the proposed SCADA system. The supervisory control and local control are also tested to be effective in the experimental results. Furthermore, it provides easy-to-implement methods for displaying data, using the Node-RED platform to increase operational reliability. Overall, the implementation of this SCADA system not only fulfills the need for an affordable monitoring solution—as evidenced by its total cost of CAD $94.5, zero running costs, and a running power of 3.19 W—but also demonstrates versatility and precision in handling real-time data in robust environments.

9. Limitations and Future Work

There may exist some potential limitations in this study. The system operational time is limited due to time constraints. A longer period of monitoring would add to the credibility of the work. Moreover, other open-source SCADA software and dashboards can be undertaken as a future research direction and comparison, e.g., Home Assistant with ESPHome. Furthermore, the PV system monitored in this paper is single string and low power. The current research work can also be extended for multi-string, larger PV systems. In addition, data storage can be considered for implementation in the cloud, e.g., Google Cloud, to align with the IoT context. Finally, the SCADA system can also have its application for different system hardware, e.g., a Li-ion battery-based PV system including interaction with battery bank BMS or a hybrid system with a grid-connected inverter, AC load, and higher DC bus voltage. Additional security features, such as user accounts, passwords, and local data logging in the server can be added in Node-RED. Using additional measurements for the PV system, system fault diagnosis can be added in Node-RED.

Author Contributions: W.H.: system design, experimental results, and writing of the manuscript under the supervision of M.T.I.; M.J.A.B.: carried out writing of the Introduction, Literature Review, and Discussion; funding acquisition; and overall review/editing of the manuscript; M.T.I.: provided the required resources and components and contributed research ideas throughout the research. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the School of Graduate Studies (SGS) at Memorial University.

Data Availability Statement: Data are contained within the article and references.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

Abbreviations

The following abbreviations are used in this manuscript:

SCADA	Supervisory Control and Data Acquisition
RTUs	Remote Terminal Units
MTUs	Master Terminal Units
FIDs	Field Instrumentation Devices
HMI	Human–Machine Interface
IoT	Internet of Things

GPIO	General-purpose Input/Output
IDE	Integrated Development Environment
PV	Photovoltaic
MPPT	Maximum Power Point Tracker
VS	Voltage Sensor
CS	Current Sensor
OS	Operating System
RAM	Random Access Memory
UART	Universal Asynchronous Receiver–Transmitter
USB	Universal Serial Bus

References

1. Lunstad, N.T.; Sowby, R.B. Smart Irrigation Controllers in Residential Applications and the Potential of Integrated Water Distribution Systems. *J. Water Resour. Plan. Manag.* **2024**, *150*. [CrossRef]
2. Ling, J.Y.X.; Chan, Y.J.; Chen, J.W.; Chong, D.J.S.; Tan, A.L.L.; Arumugasamy, S.K.; Lau, P.L. Machine learning methods for the modelling and optimisation of biogas production from anaerobic digestion: A review. *Environ. Sci. Pollut. Res.* **2024**, *31*, 19085–19104. [CrossRef] [PubMed]
3. Kumar Dalapati, G.; Ghosh, S.; Sherin P A, T.; Ramasubramanian, B.; Samanta, A.; Rathour, A.; Kin Shun Wong, T.; Chakrabortty, S.; Ramakrishna, S.; Kumar, A. Maximizing solar energy production in ASEAN region: Opportunity and challenges. *Results Eng.* **2023**, *20*, 101525. [CrossRef]
4. Alanazi, M.; Mahmood, A.; Chowdhury, M.J.M. SCADA vulnerabilities and attacks: A review of the state-of-the-art and open issues. *Comput. Secur.* **2023**, *125*, 103028. [CrossRef]
5. Folgado, F.J.; Calderón, D.; González, I.; Calderón, A.J. Review of Industry 4.0 from the Perspective of Automation and Supervision Systems: Definitions, Architectures and Recent Trends. *Electronics* **2024**, *13*, 782. [CrossRef]
6. Babayigit, B.; Abubaker, M. Industrial Internet of Things: A Review of Improvements Over Traditional SCADA Systems for Industrial Automation. *IEEE Syst. J.* **2024**, *18*, 120–133. [CrossRef]
7. de Arquer Fernández, P.; Fernández Fernández, M.Á.; Carús Candás, J.L.; Arboleya Arboleya, P. An IoT open source platform for photovoltaic plants supervision. *Int. J. Electr. Power Energy Syst.* **2021**, *125*, 106540. [CrossRef]
8. Jose, J.; Mathew, V. Internet of Things—A Model for Data Analytics of KPI Platform in Continuous Process Industry. *Informatica* **2024**, *48*, 119–130. [CrossRef]
9. Abdelatti, M.; Sodhi, M. Lab-Scale Smart Factory Implementation Using ROS. In *Robot Operating System (ROS): The Complete Reference*; Koubaa, A., Ed.; Springer International Publishing: Cham, Switzerland, 2023; Volume 7, pp. 119–143.
10. Flamini, A.; Ciurluini, L.; Loggia, R.; Massaccesi, A.; Moscatiello, C.; Martirano, L. A Prototype of Low-Cost Home Automation System for Energy Savings and Living Comfort. *IEEE Trans. Ind. Appl.* **2023**, *59*, 4931–4941. [CrossRef]
11. Rattanapoka, C.; Chanthakit, S.; Chimchai, A.; Sookkeaw, A. An MQTT-based IoT Cloud Platform with Flow Design by Node-RED. In Proceedings of the 2019 Research, Invention, and Innovation Congress (RI2C), Bangkok, Thailand, 11–13 December 2019; pp. 1–6.
12. Ahmed, O.; Iqbal, M.T. Remote Monitoring, Control and Data Visualization for a Solar Water Pumping System. *Eur. J. Electr. Eng. Comput. Sci.* **2023**, *7*, 71–77. [CrossRef]
13. Radia, M.A.A.; Nimr, M.K.E.; Atlam, A.S. IoT-based wireless data acquisition and control system for photovoltaic module performance analysis. *e-Prime—Adv. Electr. Eng. Electron. Energy* **2023**, *6*, 100348. [CrossRef]
14. Martin, A.D.; Cano, J.M.; Medina-García, J.; Gómez-Galán, J.A.; Hermoso, A.; Vazquez, J.R. Artificial vision wireless PV system to efficiently track the MPP under partial shading. *Int. J. Electr. Power Energy Syst.* **2023**, *151*, 109198. [CrossRef]
15. Aghenta, L.O.; Iqbal, M.T. Low-Cost, Open Source IoT-Based SCADA System Design Using Thinger.IO and ESP32 Thing. *Electronics* **2019**, *8*, 822. [CrossRef]
16. Ziane, A.; Dabou, R.; Necaibia, A.; Rouabhia, A.; Bouchouicha, K.; Sahouane, N.; Lachtar, S.; Bouraiou, A.; Larbi, A.A. IoT Platform For Online Monitoring Of Renewable Energy Systems. In Proceedings of the 2022 3rd International Conference on Embedded & Distributed Systems (EDiS), Oran, Algeria, 2–3 November 2022; pp. 55–60.
17. Voicu, V.; Petreus, D.; Cebuc, E.; Etz, R. Industrial IoT (IIOT) Architecture for Remote Solar Plant Monitoring. In Proceedings of the 2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet), Sovata, Romania, 15–16 September 2022; pp. 1–4.
18. Sagayaraj, R.; Priya, S.; Malathi, S.; Sujith, S. IoT Monitoring for Hybrid Photovoltaic Fuel Cell System. In Proceedings of the 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 14–16 June 2023; pp. 949–953.
19. Matsui, T.; Yamamoto, K.; Sumi, S.; Triruttanapiruk, N. Detection of Lightning Damage on Wind Turbine Blades Using the SCADA System. *IEEE Trans. Power Deliv.* **2021**, *36*, 777–784. [CrossRef]
20. Zaman, M.; Upadhyay, D.; Lung, C.H. Validation of a Machine Learning-Based IDS Design Framework Using ORNL Datasets for Power System with SCADA. *IEEE Access* **2023**, *11*, 118414–118426. [CrossRef]

21. Sverko, M.; Grbac, T.G.; Mikuc, M. SCADA Systems With Focus on Continuous Manufacturing and Steel Industry: A Survey on Architectures, Standards, Challenges and Industry 5.0. *IEEE Access* **2022**, *10*, 109395–109430. [CrossRef]
22. Pliatsios, D.; Sarigiannidis, P.; Lagkas, T.; Sarigiannidis, A.G. A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1942–1976. [CrossRef]
23. Li, S.; Jiang, B.; Wang, X.; Dong, L. Research and Application of a SCADA System for a Microgrid. *Technologies* **2017**, *5*, 12. [CrossRef]
24. Figueiredo, J.; Sá da Costa, J. A SCADA system for energy management in intelligent buildings. *Energy Build.* **2012**, *49*, 85–98. [CrossRef]
25. Khadra, A.; Rammal, R. SCADA System for Solar Backup Power System Automation. In Proceedings of the 2022 International Conference on Smart Systems and Power Management (IC2SPM), Beirut, Lebanon, 10–12 November 2022; pp. 75–79. [CrossRef]
26. Nițulescu, I.-V.; Korodi, A. Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions. *IoT* **2020**, *1*, 5. [CrossRef]
27. Omidi, S.A.; Baig, M.J.; Iqbal, M.T. Design and Implementation of Node-Red Based Open-Source SCADA Architecture for a Hybrid Power System. *Energies* **2023**, *16*, 2092. [CrossRef]
28. BananaPi. BananaPi BPI-M4 Berry Documentation. Available online: https://docs.banana-pi.org/en/BPI-M4_Berry/BananaPi_BPI-M4_Berry (accessed on 22 April 2024).
29. DFRobot. FireBeetle 2 ESP32-E IoT Microcontroller. Available online: https://www.dfrobot.com/product-2195.html (accessed on 12 April 2024).
30. Node-RED. Node-RED User Guide—Concepts. Available online: https://nodered.org/docs/user-guide/concepts (accessed on 12 April 2024).
31. SparkFun_Electronics. ACS712. Available online: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf (accessed on 12 April 2024).