energies

MDPI

Article

Open-Source Internet of Things-Based Supervisory Control and Data Acquisition System for Photovoltaic Monitoring and Control Using HTTP and TCP/IP Protocols

Wajahat Khalid [1], Mohsin Jamil [1], Ashraf Ali Khan [1,*] and Qasim Awais [1,2]

[1] Department of Electrical & Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada; wkhalid@mun.ca (W.K.); mjamil@mun.ca (M.J.); qasim.awais@fjwu.edu.pk (Q.A.)

[2] Department of Electronics & Computer Science, Fatima Jinnah Women University, Old Presidency, Rawalpindi 46000, Pakistan

* Correspondence: ashrafak@mun.ca

Abstract: This study presents a cost-effective IoT-based Supervisory Control and Data Acquisition system for the real-time monitoring and control of photovoltaic systems in a rural Pakistani community. The system utilizes the Blynk platform with Arduino Nano, GSM SIM800L, and ESP-32 microcontrollers. The key components include a ZMPT101B voltage sensor, ACS712 current sensors, and a Maximum Power Point Tracking module for optimizing power output. The system operates over both Global System for Mobile Communications and Wi-Fi networks, employing universal asynchronous receiver–transmitter serial communication and using the transmission control protocol/Internet protocol and hypertext transfer protocol for data exchange. Testing showed that the system consumes only 3.462 W of power, making it highly efficient. With an implementation cost of CAD 35.52, it offers an affordable solution for rural areas. The system achieved an average data transmission latency of less than 2 s over Wi-Fi and less than 5 s over GSM, ensuring timely data updates and control. The Blynk 2.0 app provides data retention capabilities, allowing users to access historical data for performance analysis and optimization. This open-source SCADA system demonstrates significant potential for improving efficiency and user engagement in renewable energy management, offering a scalable solution for global applications.

Keywords: renewable energy; IoT; SCADA (supervisory control and data acquisition); Arduino Nano; GSM SIM800L; ESP-32; TCP/IP (transmission control protocol/Internet protocol); HTTP (hypertext transfer protocol); UART (universal asynchronous receiver–transmitter); Blynk; Arduino IDE 1.8.19; sensors; low cost and low power

check for updates

Citation: Khalid, W.; Jamil, M.; Khan, A.A.; Awais, Q. Open-Source Internet of Things-Based Supervisory Control and Data Acquisition System for Photovoltaic Monitoring and Control Using HTTP and TCP/IP Protocols. *Energies* **2024**, *17*, 4083. https://doi.org/10.3390/en17164083

Academic Editors: Sumin Kim and Sojung Kim

Received: 10 July 2024
Revised: 12 August 2024
Accepted: 14 August 2024
Published: 16 August 2024

1. Introduction

In recent years, global energy demand has surged due to population growth and rapid industrial advancement. As a result, renewable energy sources have gained widespread adoption in both the industrial and residential sectors [1]. Solar energy, being unlimited, eco-friendly, and secure, is a highly sought-after form of green energy [2]. Global photovoltaic (PV) capacity has dramatically increased over the past twenty years, from 1288 MW in 2000 to 1,177,000 MW in 2022, reflecting a significant rise in solar energy use [3]. Pakistan receives abundant solar irradiance, averaging 2400 kWh/m^2 annually and 5–7 kWh/m^2 daily, with over 2300–2700 sunshine hours and more than 300 sunny days per year. Leveraging this resource could significantly address the country's energy shortfall [4]. Pakistan's energy sector faces severe power shortages, about 2.5 outages daily, lasting 13.2 h each. Over 75% of businesses cite unreliable electricity as a major obstacle, primarily due to technical malfunctions [5]. Load shedding severely affects Pakistan's industrial sector, reducing its GDP by 2–3% annually, or USD 7.5 to USD 11.25 billion. The textile industry, responsible for 60% of exports, loses about USD 1.3 billion yearly due to power shortages, decreasing manufacturing output by 15–20%. Industries spend over USD 1 billion annually on diesel for generators, with backup systems costing USD 5000 to USD 100,000 each. Small and Medium Enterprises (SMEs) face revenue declines of 20–30%, with 10–15% closing or downsizing. Load shedding also results in the loss of over 400,000 jobs each year and deters USD 1–2 billion in foreign investment, highlighting the urgent need for reliable energy solutions to support economic stability [6]. Figure 1 shows the electricity demand and generation of Pakistan.

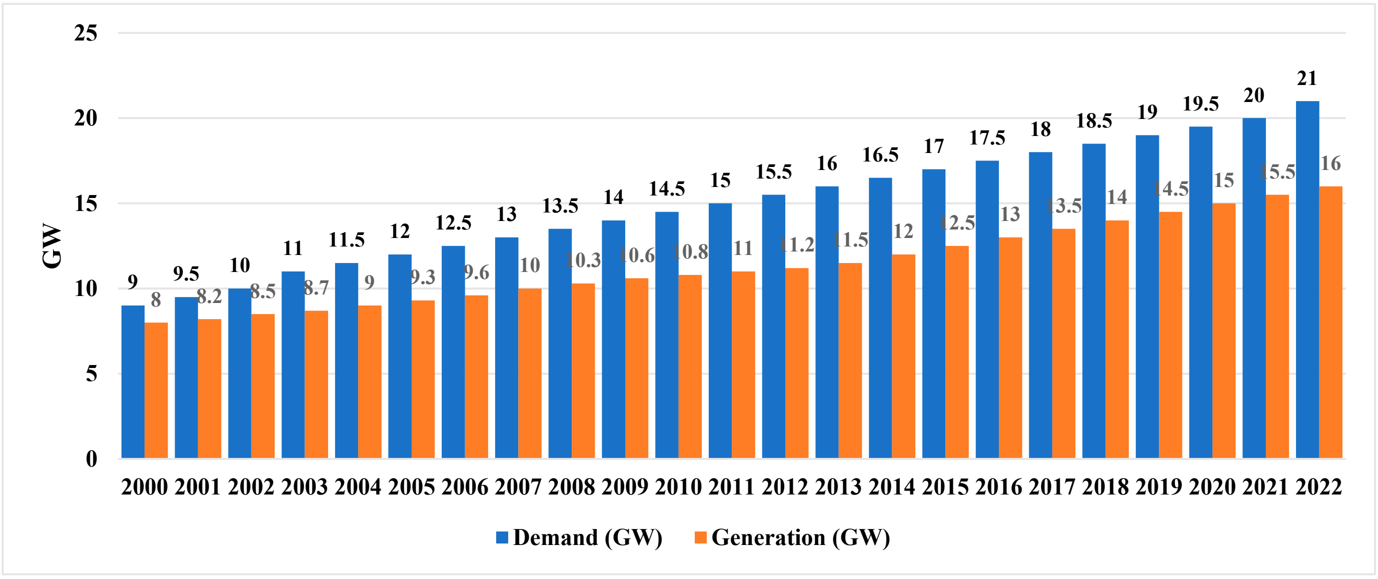


Figure 1. Electricity demand and generation of Pakistan [7].

In Pakistan, 24% of the population, about 51 million people, lack reliable electricity access and often rely on expensive fossil fuels. To address this, cost-effective distributed renewable sources like PV systems are increasingly used to power these remote areas [8]. A standalone PV system for rural homes offers reliable, off-grid electricity with solar panels, batteries, charge controllers, and inverters. Ideal for remote areas, it provides a sustainable, cost-effective alternative to fossil fuels, powering appliances and lighting [9].

There is growing interest in smart homes with interconnected devices for remote control. Users can monitor and manage lighting, climate, entertainment, and appliances via a smartphone from anywhere. IoT energy monitoring devices offer remote control and data transmission over the Internet [10]. SCADA (Supervisory Control and Data Acquisition) technology retrieves data from remote facilities and administers limited control commands [11], and integrates sensors, actuators, and software like Human Machine Interface (HMI) for data collection, networked communication, display, and monitoring/control. It uses Remote Terminal Units (RTUs) such as microcontrollers and Master Terminal Units (MTUs) like the Arduino IoT Cloud, alongside communication networks [12]. SCADA systems are typically categorized as proprietary or open source. Proprietary systems use components from a single manufacturer, which can limit flexibility and pose vulnerabilities if the supplier fails [13].

The Internet of Things (IoT) connects physical entities via sensors and software for online data exchange. Originating from the RFID community, IoT has expanded with advances in communication and cloud computing. Billions of devices, from household items to industrial machinery, communicate over IP networks, enabling real-time monitoring, enhanced security, and intelligent decision-making [14]. This supports real-time identification, location tracking, monitoring, and event automation [15]. The IoT platform integrates smart sensors, PLCs, actuators, and IEDs from industrial control systems (ICS) and SCADA networks [16]. Home PV systems utilize IoT for the real-time monitoring of voltage, current levels, and safety via sensors, enhancing energy production and usage. Wireless data transmission facilitates analysis to optimize solar capture and efficient battery storage management through automated controls and intelligent algorithms [17]. This proactive approach allows for timely interventions such as maintenance or adjustments to optimize performance and minimize disruptions [18]. Remote access via smartphone apps or web interfaces enables homeowners to manage PV systems from anywhere with Internet access, enhancing convenience and facilitating informed decisions on energy usage, troubleshooting, and cost-effectiveness optimization. Communication protocols are crucial in heterogeneous systems, providing a standardized framework for device interaction. The key protocols for machine-to-machine (M2M) communication and IoT systems include MQTT, AMQP, HTTP, and CoAP [19] and TCP/IP-based frameworks to connect devices to designated applications, enabling data transfer over networks like the Internet, Intranet, WiFi, WiMax, and LTE to control centers for processing and management [20]. Figure 2 depicts the SCADA system structure and layer scheme of SCADA.

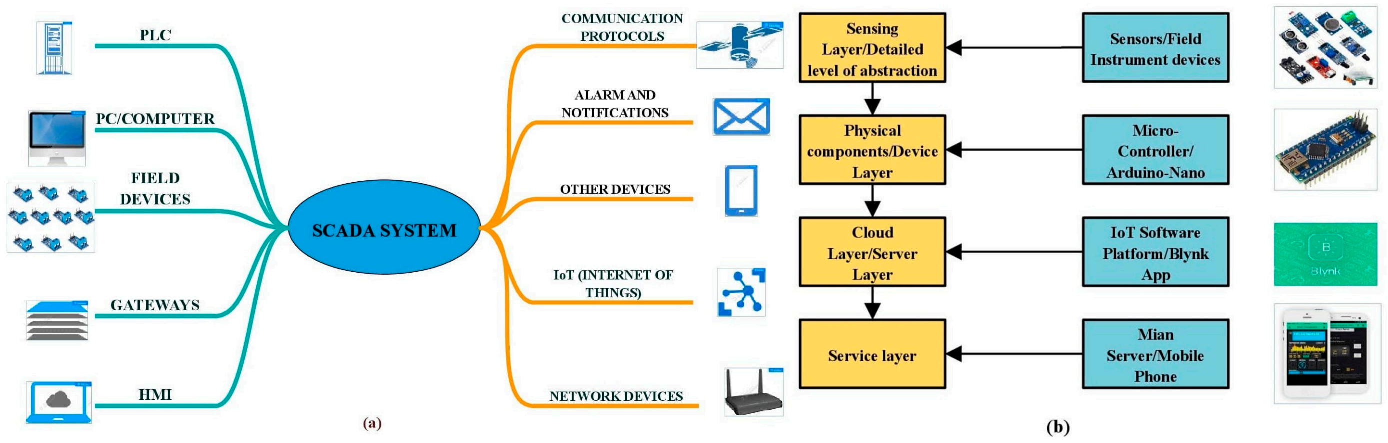


Figure 2. (**a**) Structure of SCADA system. (**b**) Layer scheme of SCADA system.

Given the escalating environmental impact of greenhouse gas (GHG) emissions from fossil fuels, evidenced by events like floods and climate change, the development of a Hybrid Power System (HPS) for rural communities in Pakistan focuses on integrating renewable energy sources, particularly solar power, with conventional generators. This aims to reduce GHG emissions while enhancing energy accessibility in rural areas. The HPS includes components such as a buck converter (DC-DC), Maximum Power Point Tracking (MPPT), an LCL Filter, and a DC-AC inverter. Advanced software tools like PVsyst 7.4 and HOMER Pro-3.18.1 are used for system sizing, evaluating energy usage patterns, and optimization under site-specific conditions. Dynamic modeling in MATLAB/Simulink r2023b and hardware-in-the-loop (HIL) validation assess the system's performance in response to changes in solar irradiance and temperature, ensuring operational efficiency and reliability. An open-source SCADA system utilizing the Blynk 2.0 app enables real-time monitoring, local data storage, and integration with GSM and WiFi networks.

The structure of this paper is outlined as follows: A literature review and proposed system comparison are presented in Section 2, an overview of the system is given in Section 3, and the components used are discussed in Section 4. The implementation method and the experimental setup/hardware and results are detailed in Section 5, the analysis is presented in Section 6, and the conclusion is provided in Section 7.

2. Literature Review and Proposed System Comparison

A considerable amount of work has been carried out in the field of IoT-based SCADA systems regarding their improvement in terms of communication and data transfer. Table 1 shows a brief overview of the work previously conducted during recent years in the field of IoT SCADA.

Table 1. An overview of prior work in the field of IoT SCADA.

Ref./Year	Methodology
[21] 2021	A grid-independent IoT SCADA system at a BTS site uses ESP32 and the Arduino IoT Cloud for monitoring current, voltage, temperature, and humidity data over Wi-Fi.
[21] 2022	An economical SCADA system for a PV plant using Arduino, Raspberry Pi, sensors, serial cables, and an open-source web interface on a Debian OS via Emoncms, processing and displaying data.
[22] 2022	An ESP32 microcontroller as the RTU and the Cayenne IoT platform with an MQTT protocol enable efficient wireless data transfer, power allocation, and detailed performance insights.
[23] 2022	An open-source SCADA system for solar-powered reverse osmosis uses FIDs, RTU, and MTU with Node-RED on Grafana.
[24] 2023	An affordable IoT-based SCADA system monitors offshore aquaculture HPS using sensors, an Arduino Leonardo RTU, and a LoRa gateway.
[25] 2023	A cost-effective IoT monitoring system was developed for an off-grid PV system in Algeria's Sahara region, supporting a small greenhouse farm.
[26] 2024	With three voltage and three current sensors, the system uses an ESP32-E to transmit data to a Banana Pi M4 via MQTT with Node-RED on the BPI-M4.

Throughout this research, an extensive review of the existing literature in the field of IoT-based SCADA systems is conducted. Despite this thorough examination, it is determined that no low-cost, low-powered SCADA system has been developed or identified that incorporates both GSM and Wi-Fi communication, utilizes TCP/IP and HTTP protocols, and provides monitoring and control of a photovoltaic (PV) system via the Blynk app and web dashboard. Furthermore, this system aims to be open-source, making it accessible for widespread adoption and customization. Table 2 presents a detailed comparison of the proposed SCADA system with previous work, highlighting its unique combination of features and capabilities not found in other systems.

Table 2. Comparison of the proposed SCADA system with those used in previous work.

Proposed SCADA System		[12]	[21]	[22]	[23]	[24]	[25]	[26]
SCADA platform	Blynk	Things Speak	Emoncms	Grafana, Node-RED	Cayenne Server	Grafana, AWS	Locally Developed	Node-RED
Open source	✔	✔	✔	✔	✔	✔	✔	✔
Power consumption (W)	3.462	Not mentioned	32.31	Not mentioned	Not mentioned	Not mentioned	13.5	3.19
Cost (USD)	35.52	88.34	761.72	N.A	N.A	N.A	107.77	94.5
Comm. protocol	HTTP, TCP/IP	MQTT	MQTT, HP	MQTT	MQTT	LORA, QTT	MQTT	MQTT
Control	✔	X	X	✔	X	X	✔	✔
Delay (ms)	50~300	50~300	50~500	50~500	50~500	50~100	160~180	50~500
GSM	✔	X	X	X	X	X	X	X
Wi-Fi	✔	✔	✔	✔	✔	✔	✔	✔
Microcontroller	Arduino Nano, ESP-32, GSM SIM800L	ESP-32	Arduino Mega 2586	Arduino Mega 2560	ESP-32	Arduino Leonardo	ESP-8266	ESP-32

The proposed SCADA system, developed as part of this research, introduces a novel approach with several distinctive features:

I. System Architecture: The proposed SCADA system employs an Arduino Nano as the RTU, utilizing voltage and current sensors as Field Instruments (FIDs), and GSM SIM800L and ESP-32 work to extend RTU communication.

II. Utilization of Open Source Platforms: The system offers thorough monitoring of essential photovoltaic (PV) parameters and enables the remote control of electrical loads to optimize energy usage. Open-source SCADA platforms, such as the Blynk app and console, exemplify this capability. Through remote access via smartphone applications or web interfaces, users can achieve real-time data access and system management, thereby enhancing user engagement and operational efficiency.

III. Data Transfer Mechanism: Data transfer in the SCADA system uses an Arduino Nano communicating with GSM SIM800L and ESP-32 via UART. TCP/IP and HTTP protocols ensure secure, reliable, and encrypted data transfer from the RTU to the MTU (Blynk), guaranteeing robust and seamless integration with the web infrastructure.

IV. Low Cost and Low Power: The developed SCADA system is designed to be low-cost, with a total expense of CAD 35.52, and have low power consumption of 3.462 W. This makes it an affordable solution for rural communities, promoting wider adoption and accessibility. The use of open-source platforms like Arduino IDE further enhances its accessibility and encourages community-driven innovation.

V. Dual-Mode Communication: Traditional SCADA systems often rely on single-mode communication and proprietary components, resulting in higher costs and lower flexibility. The system supports both GSM and Wi-Fi communication, ensuring reliable data transfer under various network conditions. This dual-mode capability enhances the flexibility and robustness of the system, making it suitable for diverse environments.

3. Site and System Description

3.1. Site Description

The chosen location, "Berru Bandi", is a small community of 10 houses in the rural area of Abbottabad District, around 25 km from Abbottabad city, Pakistan. Positioned on a mountain at coordinates 34°16′38″ N 73°15′18″ E and at an elevation of 1456.79 m above sea level, accessing this site is difficult due to the absence of road infrastructure and basic amenities. Figure 3 shows an overview of the selected site from Google Maps.

Figure 3. Site overview from Google Maps [27].

3.2. PV System Description

The PV system envisioned for the chosen location integrates multiple elements, including a solar photovoltaic system, an MPPT controller, a battery bank, a DC-DC buck converter, a DC-AC inverter, an LCL filter, and an AC power source. This setup features both AC and DC buses for increased operational versatility and easier upkeep, ensuring uninterrupted power provision. The system is configured with a DC-AC inverter that converts the DC electricity generated by the solar panels into AC electricity for use in standard applications. A three-phase multi-level inverter is used, which offers benefits such as reduced switching losses and improved voltage waveform quality. The PV system also includes a battery bank with a maximum capacity of 201 Ah and a round-trip efficiency of 85%. The battery bank's maximum charge and discharge currents are 68.4 A and 342 A, respectively. The PV system's orientation and tilt angle are optimized using PVsyst software, with the tilt angle set at 34° to maximize energy production. This ensures maximum efficiency and minimizes loss factors associated with solar radiation incidence on the panels. The designed system has a load of 137.48 kWh/d and a peak load of 33.54 kW. To optimize performance and cost-effectiveness, the system design includes simulations conducted with HOMER Pro software. This simulation explored 982 different configurations of power sources, with the best setup incorporating solar panels, a converter, a battery bank, and a diesel genset, leading to a low net present cost (NPC) of USD 0.102 million and a cost of energy (COE) of USD 0.158.

The Maximum Power Point Tracking (MPPT) technique employed for the PV system in this study is the Incremental Conductance Algorithm (ICA). The ICA was chosen for its ability to dynamically adjust the operating voltage and current of the photovoltaic system to ensure operation at or near its Maximum Power Point (MPP). This optimization is crucial for maximizing the efficiency and power output of the PV system, especially under varying environmental conditions such as changes in solar irradiance and temperature. Unlike simpler algorithms like Perturb and Observe (P&O), the ICA evaluates both the instantaneous conductance and the incremental changes in conductance of the PV array. This allows it to more accurately track the MPP and respond effectively to rapid changes in weather conditions.

3.3. System Description

Software and hardware components are essential and integral to the design of the proposed system, ensuring its successful and smooth operation. In this SCADA system designed for monitoring and controlling a PV system in a rural community, the electrical parameters are continuously monitored by voltage and current sensors located at key points within the system. Figure 4 depicts the brief of the proposed SCADA system.

The proposed management and operation algorithm enhances the SCADA system for photovoltaic (PV) microgrids by integrating advanced data acquisition, processing, and control mechanisms. It is monitored by sensors, including the ZMPT101B for voltage and the ACS712 for current, as well as a relay for monitoring voltage and current in the PV panel and battery storage unit. These sensors interface with an Arduino Nano microcontroller, serving as an RTU, which processes the acquired data. The RTU then transmits the processed data to an MTU at a central monitoring station via the SIM800L GSM module or the ESP-32. The MTU, implemented using a cloud-based server interfaced with the Blynk app, aggregates and visualizes the data for real-time monitoring and control. In the absence of GSM connectivity, the ESP-32 transmits the processed data to the central monitoring unit using a Wi-Fi network through HTTP interfacing with the Blynk app. At the central monitoring station, the SCADA Blynk 2.0 software platform receives and processes the data, employing communication protocols such as TCP/IP over the GSM network and HTTP in the case of ESP-32 to integrate seamlessly with the Blynk app and web console. Through the Blynk app, users can remotely oversee and manage the PV system using their smartphones or computers, visualizing real-time data. Additionally, the system offers control capabilities, allowing users to control electrical devices and manage the operation of the PV system for optimal performance and energy efficiency. This integration supports essential functionalities, including dynamic load management and threshold alerts for critical parameters, helping to maintain system stability.

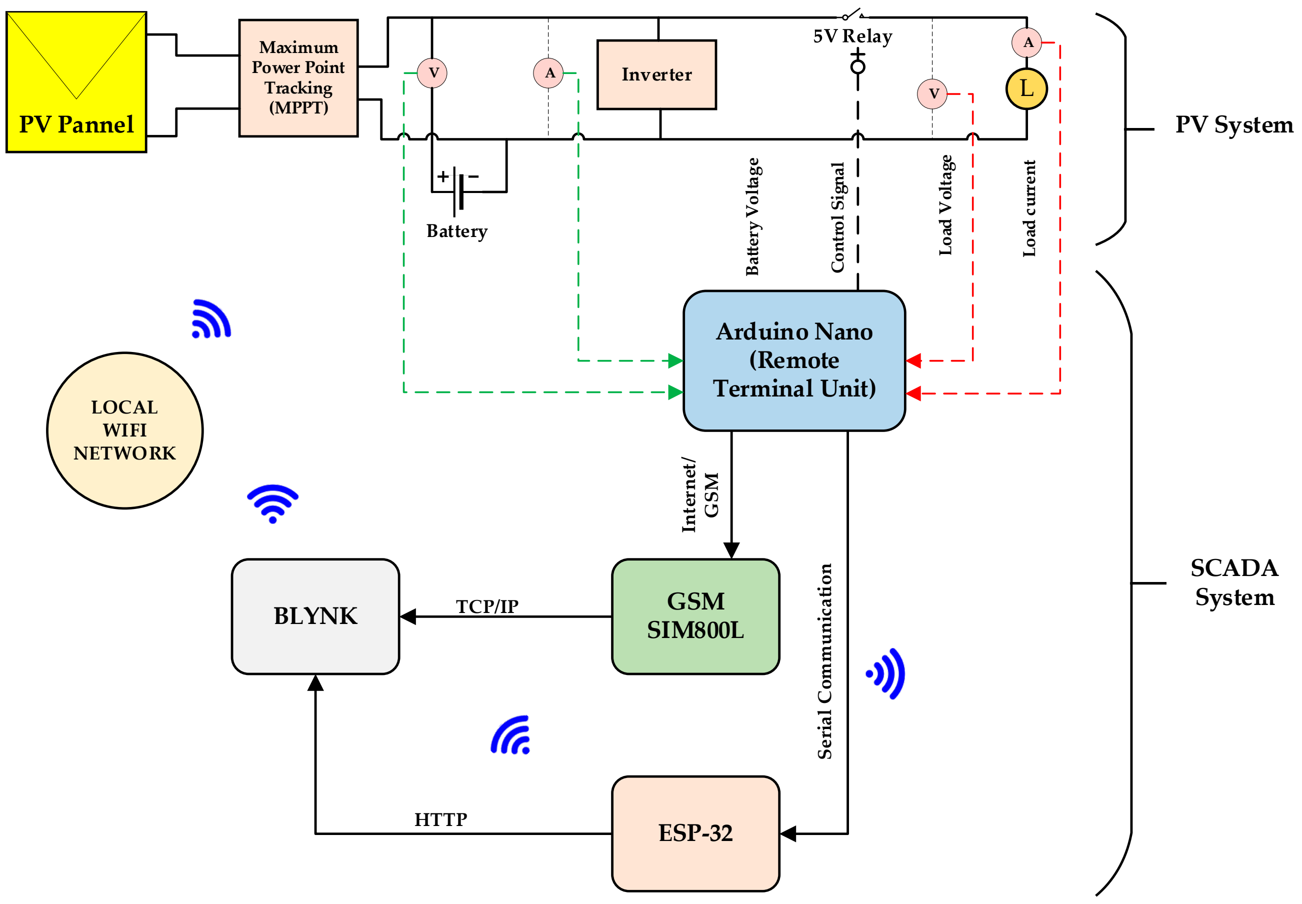


Figure 4. Brief of the proposed SCADA system.

4. Components Used in the Proposed SCADA System

This section discusses the hardware components of the proposed SCADA system. It uses an Arduino Nano as the RTU, with voltage (ZMPT101B) and current (ACS712) sensors for measuring electrical parameters and a relay for control. The SIM800L GSM module and ESP-32 facilitate remote data transmission and control, ensuring efficient SCADA system operation.

4.1. Arduino Nano

The microcontroller is a digital system extensively used in both household and industrial electronics. Its popularity stems from its affordability. Microcontrollers are commonly employed for control systems, signal processing, instrumentation, and a range of other applications [28]. The Arduino Nano was selected for the SCADA system because of its compact size, cost-effectiveness, and versatile features. Its small form factor facilitates easy integration into systems with limited space, while its affordability makes it a practical choice for large-scale deployments. Compared to other microcontrollers, such as the Raspberry Pi Pico, the Arduino Nano is distinguished by its simplicity and ease of use, making it an ideal solution for developers seeking a straightforward and reliable option.

Figure 5 shows a pinout diagram of the Arduino Nano, and Table 3 shows the technical specifications of the Arduino Nano.

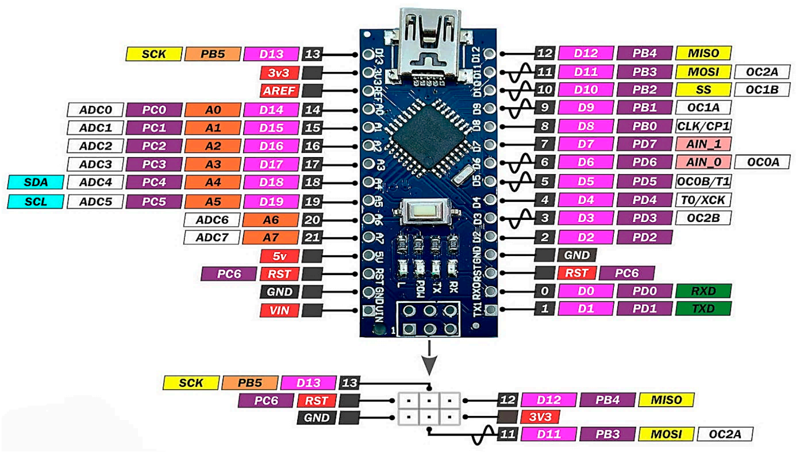


Figure 5. Pin layout of Arduino Nano [29].

Table 3. Technical specifications of Arduino Nano.

Specifications	Details
Microcontroller	ATmega328P
Operating Voltage and clock speed	5 V and 16 MHz
Input voltage	7–12 V
Digital I/O pins	14 (of which 6 provided PWM)
PWM digital I/O pins	6 (D3, D5, D6, D9, D10, D11)
Analog input pins	8 (A0 to A7)
Flash memory	32 KB of which 2 KB was used by the bootloader
EPROM	1 KB (ATmega328P)
Communication	UART, 12C, SPI, Mini-USB

4.2. Voltage Sensor ZMPT101B

The ZMPT101B is a high-precision voltage sensor module designed for measuring DC voltages ranging from 0 to 25 V, with an analog output voltage proportional to the input, readable by an Arduino's ADC [30]. It is ideal for applications such as battery level monitoring and power supply voltage tracking, and requires precise voltage measurements. Interfacing with an Arduino involves connecting the sensor's VCC to 5 V, the GND to the ground, and the output to an analog input pin (e.g., A0). In the Arduino code, the sensor value read from the analog pin is converted to the input voltage using a calibration factor, mapping the 0–5 V output to the 0–25 V input range. The ZMPT101B voltage sensor is widely used for real-time monitoring in applications such as battery monitoring, power supply verification, and solar systems.

4.3. Current Sensor ACS712

The ACS712 current sensor, utilizing Hall-effect technology, is essential in PV systems for the precise measurement of DC and AC within ±5 A, ±20 A, or ±30 A ranges. It outputs 66 to 185 mV per ampere and operates on a single 5-volt supply. This sensor detects current through a conductor via a magnetic field, converted to voltage by its integrated Hall IC [31]. In conjunction with an Arduino Nano microcontroller board, the ACS712 becomes a vital component for monitoring current flow within the system. The Arduino Nano's analog input pins facilitate easy interfacing with the ACS712 sensor, enabling the reading of analog voltage output proportional to the measured current. Using Arduino programming, it converts the voltage into precise current values. These values can be displayed on LCD screens, transmitted wirelessly for remote monitoring, or used for tasks like load balancing and fault detection in PV systems.

4.4. Buck Converter LM2596

Buck converters are DC-DC step-down switching regulators known for their high power efficiency, especially when the input voltage exceeds the desired output voltage [32]. Essential in PV systems, they adjust the output of PV modules to match the load specifications [33]. The LM2596 is a popular type of buck converter IC used with charge controllers in PV systems to regulate panel voltage. The LM2596 efficiently steps down high DC voltage from PV panels to a stable level suitable for battery charging or powering loads. This is crucial for optimal battery performance and longevity, preventing overcharging and over-discharging. Adjusting its output voltage to match the charge controller and load requirements enhances system efficiency and reliability.

4.5. Inverter (DC to AC)

In a PV system, an inverter plays a key role in converting the direct current (DC) electricity generated by PV panels into alternating (AC) electricity at 220 V, which is suitable for operating household appliances and devices. The process initiates with the DC electricity produced by the solar panels being supplied to the input of the inverter. Inside the inverter, electronic components such as transistors, capacitors, and diodes work together to convert the DC electricity into an AC signal using pulse-width modulation (PWM) or square-wave modulation techniques. This AC signal is then passed through a transformer to increase the voltage to the required 220 V, while also ensuring isolation and safety. Finally, the output undergoes filtering and shaping to produce a clean and stable AC output, ready for use in powering lights, appliances, and other electrical devices within the household.

4.6. Liquid Crystal Display

An LCD (Liquid Crystal Display) with an I2C module is an external device used to display an output on a screen [34]. In PV applications, an LCD is commonly used alongside an Arduino Nano to provide a user-friendly interface for monitoring and controlling the system. The Arduino Nano collects data from various sensors, such as voltage and current sensors, which measure the performance and status of the PV system. These data are then processed and displayed on the LCD screen, allowing users to easily read important information such as the current output, battery charge level, and system status.

4.7. Five-Volt Single-Channel Relay

A 5 V single-channel relay is an electromechanical switch used to control high-power devices with a low-voltage signal from microcontrollers like the Arduino Nano or Raspberry Pi. It operates on a 5 V DC supply, switching up to 250 V AC or 30 V DC at 10 A. A microcontroller's digital signal activates its internal coil, altering the contact states to control components such as motors, lights, and household appliances [35]. Featuring Normally Open (NO) and Normally Closed (NC) contacts, it provides flexibility in managing various loads. A 5 V single-channel relay, when integrated with an Arduino Nano, provides intelligent control and automation for PV systems, enhancing overall performance and reliability.

4.8. GSM Module SIM800L

GSM is a global standard for digital cellular communication, defining functionalities and interface requirements in networks like Base Station Systems (BSSs), Switching Systems (SSs), and Operation and Support Systems (OSSs) [36]. The SIM800L module enables data communication over GPRS in machine-to-machine (M2M) systems [37], supporting voice calls, SMS, and GPRS on GSM frequencies (850/900/1800/1900 MHz). It communicates with microcontrollers via UART. To interface with an Arduino Nano, we must connect SIM800L's VCC to 5 V, the GND to the ground, the TX to the Arduino RX, and the RX to the Arduino TX.

4.9. ESP-32 System-on-Chip (SOC) Microcontroller

The ESP-32, developed by Espressif Systems, is a versatile, low-cost System-On-Chip (SOC) microcontroller known for its integrated Wi-Fi and dual-mode Bluetooth capabilities. Launched in 2016 as a successor to the ESP8266, it features a dual-core Tensilica Xtensa LX6 microprocessor (Cadence Design Systems, San Jose, CA, USA), clock speeds up to 240 MHz, and advanced power management with various sleep modes, making it ideal for energy-efficient IoT applications. Figure 6 shows a pinout of the ESP-32, and Table 4 shows the technical details of the ESP-32.

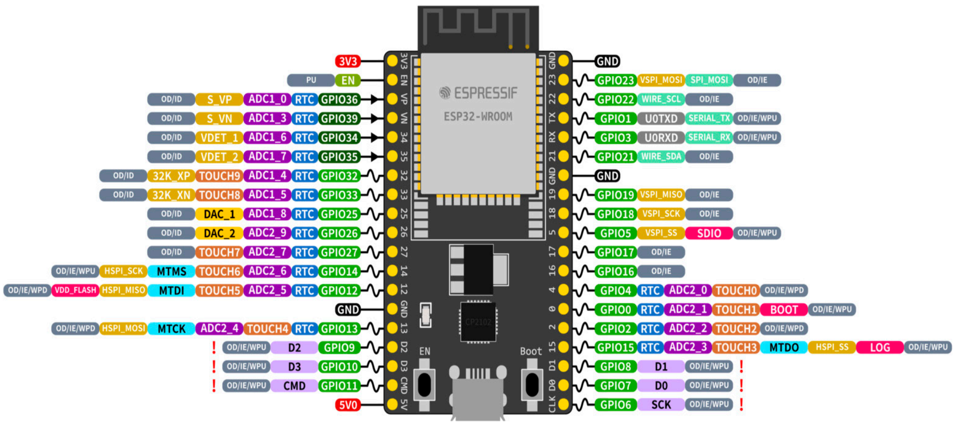

3V3
PU EN
OD/ID S_VP ADC1_0 RTC GPIO36
OD/ID S_VN ADC1_3 RTC GPIO39
OD/ID VDET_1 ADC1_6 RTC GPIO34
OD/ID VDET_2 ADC1_7 RTC GPIO35
OD/ID 32K_XP TOUCH9 ADC1_4 RTC GPIO32
OD/ID 32K_XN TOUCH8 ADC1_5 RTC GPIO33
OD/ID DAC_1 ADC1_8 RTC GPIO25
OD/ID DAC_2 ADC2_9 RTC GPIO26
OD/ID TOUCH7 ADC2_7 RTC GPIO27
OD/IE/WPU HSPI_SCK MTMS TOUCH6 ADC2_6 RTC GPIO14
OD/IE/WPD VDD_FLASH HSPI_MISO MTDI TOUCH5 ADC2_5 RTC GPIO12
GND
OD/IE/WPD HSPI_MOSI MTCK ADC2_4 TOUCH4 RTC GPIO13
! OD/IE/WPU D2 GPIO9
! OD/IE/WPU D3 GPIO10
! OD/IE/WPU CMD GPIO11
5V0
3V3 EN VP VN 34 35 32 33 25 26 27 14 12 GND 13 D2 D3 CMD 5V
ESPRESSIF
ESP32-WROOM
CP2102
EN
Boot
GND 23 22 TX RX 21 GND 19 18 5 17 16 4 0 2 15 D1 D0 CLK
GND
GPIO23 VSPI_MOSI SPI_MOSI OD/IE
GPIO22 WIRE_SCL OD/IE
GPIO1 U0TXD SERIAL_TX OD/IE/WPU
GPIO3 U0RXD SERIAL_RX OD/IE/WPU
GPIO21 WIRE_SDA OD/IE
GND
GPIO19 VSPI_MISO OD/IE
GPIO18 VSPI_SCK OD/IE
GPIO5 VSPI_SS SDIO OD/IE/WPU
GPIO17 OD/IE
GPIO16 OD/IE
GPIO4 RTC ADC2_0 TOUCH0 OD/IE/WPD
GPIO0 RTC ADC2_1 TOUCH1 BOOT OD/IE/WPU
GPIO2 RTC ADC2_2 TOUCH2 OD/IE/WPD
GPIO15 RTC ADC2_3 TOUCH3 MTDO HSPI_SS LOG OD/IE/WPU
GPIO8 D1 OD/IE/WPU !
GPIO7 D0 OD/IE/WPU !
GPIO6 SCK OD/IE/WPU !

Figure 6. Pin layout of the ESP32 [38].

Table 4. Technical specifications of the ESP-32.

Specifications	Details
Processor	Xtensa dual-core 32-bit LX6, up to 240 MHz
WiFi	IEEE 802.11 b/g/n, 2.4 GHz
Supply voltage	3.3 V
GPIO pins	34 GPIO pins
Analog channels	18 channels (12-bit SAR ADCs)
Digital to analog	2 channels (8-bit DACs)
Communication interfaces	SPI, I2C, I2S, UART, CAN, PWM, ADC, DAC

The ESP-32 is widely used in home automation, wearable electronics, industrial automation, and smart energy devices, and can be programmed using multiple development frameworks like Arduino, MicroPython v1.20.0, and ESP-IDF v5.x, making it a favored option for both hobbyists and professional developers [39].

5. Implementation Approach

To implement the proposed SCADA system, the following strategy was employed. The designed system followed two approaches:

1. The IoT system was developed using an Arduino with a GSM SIM800L module interfaced with the Blynk app.
2. The IoT system was developed using an Arduino serially interfaced with an ESP-32, utilizing the Blynk console.

The process starts by creating a circuit diagram using EasyEDA software, incorporating components like the Arduino Nano, GSM SIM800L, ESP-32, current and voltage sensors, a relay, buck converters, and an LCD. The components are then soldered onto the PCB. Next, the Blynk app, an open-source SCADA design system, is configured. The Arduino code is developed to manage the SCADA system parameters, uploaded via Arduino IDE

1.8.19, and verified using a serial monitor. Hardware testing follows to enable live data monitoring from FIDs and the control of electrical loads using the IoT Blynk app and Blynk console web dashboard. Figure 7 shows a flow chart of the system process.

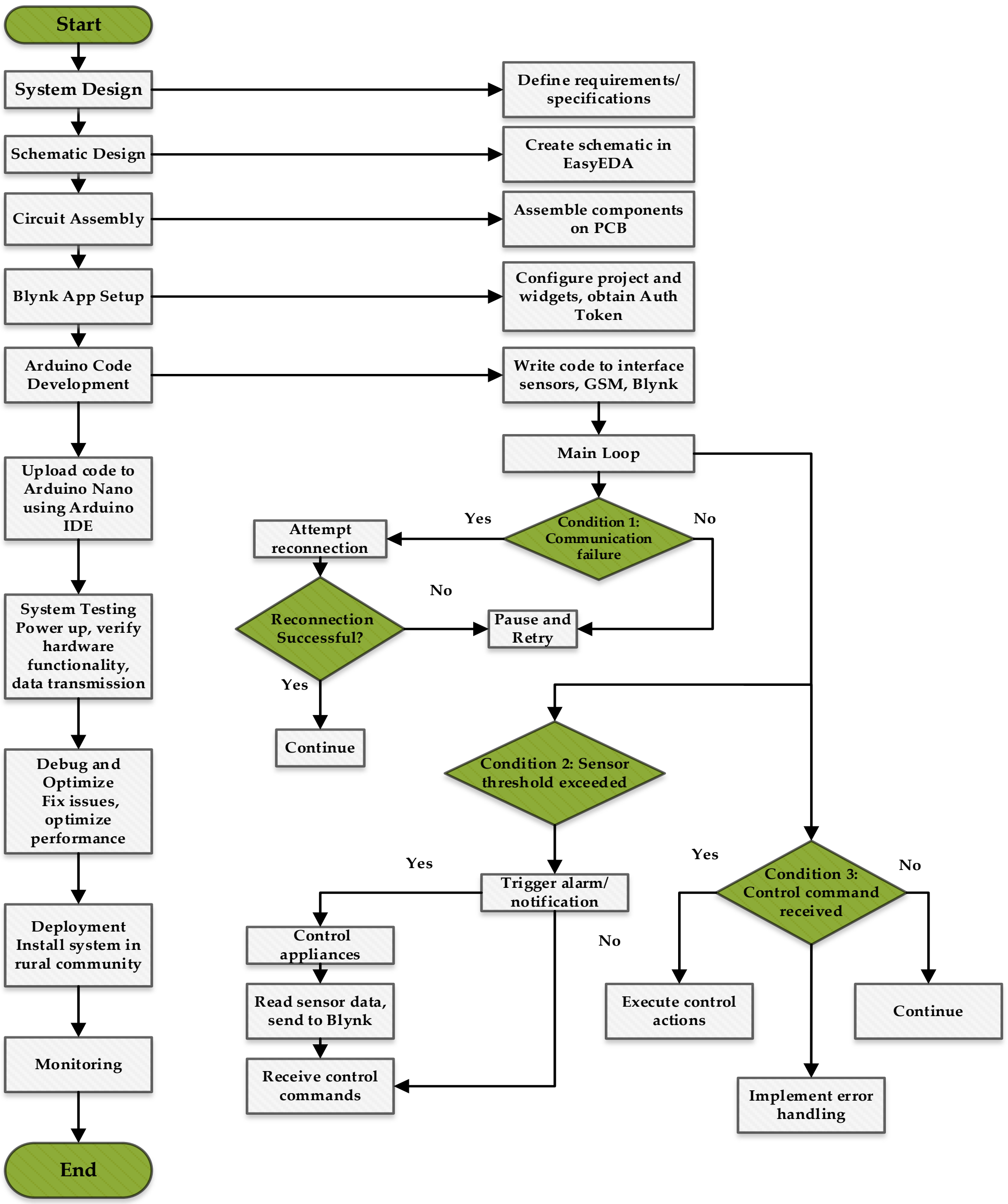

Start
System Design
Define requirements/ specifications
Schematic Design
Create schematic in EasyEDA
Circuit Assembly
Assemble components on PCB
Blynk App Setup
Configure project and widgets, obtain Auth Token
Arduino Code Development
Write code to interface sensors, GSM, Blynk
Main Loop
Upload code to Arduino Nano using Arduino IDE
Yes
Attempt reconnection
Condition 1: Communication failure
No
No
Reconnection Successful?
Pause and Retry
Yes
Continue
System Testing Power up, verify hardware functionality, data transmission
Debug and Optimize Fix issues, optimize performance
Condition 2: Sensor threshold exceeded
Yes
Trigger alarm/ notification
Yes
Condition 3: Control command received
No
Deployment Install system in rural community
Control appliances
No
Read sensor data, send to Blynk
Execute control actions
Continue
Monitoring
Receive control commands
Implement error handling
End

Figure 7. Flow chart of SCADA system process.

5.1. SCADA-Based PV System Design Using Arduino Nano and GSM SIM800L

5.1.1. Schematic Design

The schematic design of the SCADA system was implemented in EasyEDA (Electronic Design Automation). EasyEDA is an online tool that facilitates the design, simulation, and creation of schematics and printed circuit boards (PCBs) [40]. The platform includes an extensive component library, allowing users to access a wide variety of electronic parts and symbols. Figure 8 shows a schematic diagram of the proposed SCADA system using the Arduino Nano and GSM SIM800L. The solar panel is interfaced with a solar charge controller, which, in turn, is connected to a 12-volt, 5-ampere battery.

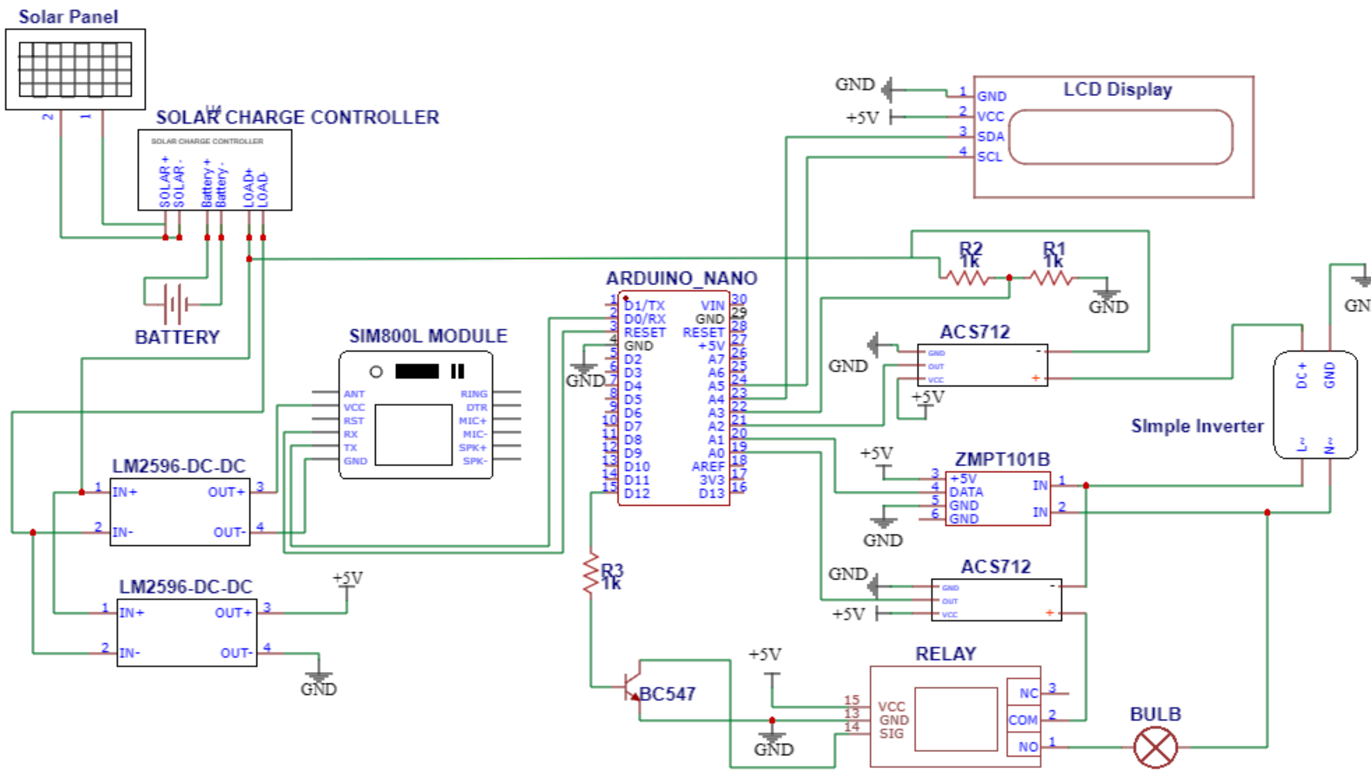


Figure 8. Circuit diagram of proposed SCADA system using Arduino Nano and GSM Sim800L (SIMCom Wireless Solutions, Shanghai, China).

The output from the charge controller is directed to two LM2596 buck converters, tasked with stepping down the higher voltage from the PV panel to a stable lower voltage (from 12 V to 5 V). Table 5 shows the connections of the FIDs to the RTU.

Table 5. FIDs connections with Arduino Nano and GSM SIM800L.

Component #	Description	Analog/Digital	Arduino Nano Pin #
1	GSM SIM800L	Digital	2.3
2	PV Voltage	Analog	22
3	LCD Display	Analog	23.24
4	ACS 712 current sensors	Analog	21.19
5	ZMPT101B	Analog	20
6	5 V Relay	Digital	15

Furthermore, a basic inverter is integrated to convert the DC to an AC, essential for the site's location in Pakistan, where the standard operating voltage is a 220 V AC. A 5 V relay is connected to an Arduino through a transistor and resistor, along with a load to switch it on and off as needed. The Arduino operates on a 5-volt power supply, with all necessary sensors connected to it. To implement the design in EasyEDA, the project is initiated in the schematic editor. Subsequently, electronic components including the Arduino Nano, a ZMPT101B voltage sensor (Winsen Electronics Technology Co., Ltd., Zhengzhou, China), an ACS712 current sensor (Allegro MicroSystems, LLC, Manchester, NH, USA), resistors, a SIM800L GSM module, a 5 V relay, and a buck converter are selected from the extensive library and placed accordingly. Following component placement, connections are established between them using a wiring tool.

5.1.2. Blynk App Setup

Blynk is a versatile IoT platform enabling users to create custom mobile apps for remote hardware monitoring and control. It supports a wide range of microcontrollers including Arduino, Raspberry Pi, ESP32, and others. With Blynk, users can easily design graphical interfaces using a drag-and-drop interface on smartphones or tablets. These interfaces communicate with connected hardware via the Blynk Cloud, enabling real-time data exchange and control.

Figure 9 depicts the Blynk app setup with the Arduino Nano and GSM SIM800L. Integrating Blynk enhances the PV system's functionality and user experience by aggregating sensor data on the solar panel output and battery status. The data are transmitted to the Blynk app via Wi-Fi or GSM, enabling the real-time visualization and control of components via smartphones or tablets.

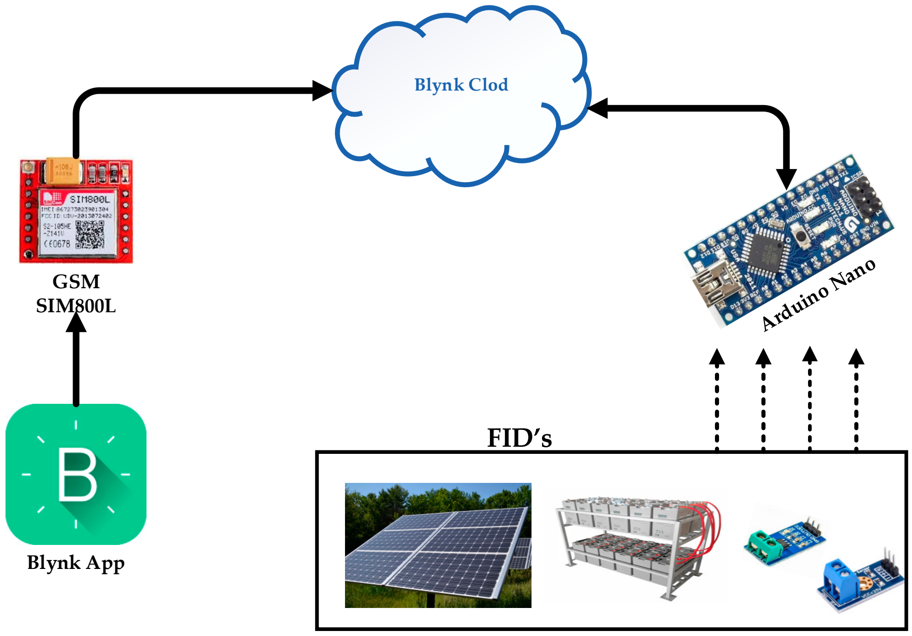


Figure 9. Blynk app setup using Arduino Nano and GSM SIM800L.

5.1.3. Arduino Code Development

The Arduino IDE allows for programming in Sketch, a C-based language. Arduino microcontrollers feature a bootloader for communication with the Arduino compiler during development [41]. The free, open-source Arduino IDE runs on Java and works on Mac, Windows, and Linux. Supporting C and C++, it simplifies program uploads and communication with Arduino hardware [42]. To write and upload a program, launch the IDE and create a new sketch. Code is written using the Arduino programming language, derived from C/C++, which simplifies development with built-in functions and libraries. Algorithm 1 details the sequence of instructions for reading data from the GSM module, voltage sensor, and current sensor.

This initializes a SoftwareSerial interface (gsm) for communication with the SIM800 modem and an LCD for local display, and the Blynk authentication details are defined for connection to the Blynk cloud server. Sensors for AC and DC voltage and AC and DC readings are interfaced using analog inputs (ACvolpin, ACcurpin, DCvolpin, DCcurpin). The data are transmitted to Blynk using Blynk.virtualWrite() in the blynkupdate() function for real-time monitoring on the Blynk mobile app. Additionally, the system allows for remote control of a relay connected to a relay pin through Blynk app commands (V4), enabling the remote switching of connected devices. After writing the code, verify it by clicking the checkmark icon to detect syntax errors. Plug the Arduino Nano into the computer using a USB cable, and select the "Arduino Nano" board and the appropriate

COM port from the "Tools" menu. Click the upload arrow icon to compile and transfer the code to the Arduino board. The IDE will compile the code, establish communication with the board, and upload the program. The above Algorithm represents the pseudocode programmed using Arduino IDE in the Arduino Nano and GSM SIM800L.

Algorithm 1: GSM, Voltage, and Current Sensor Data Reading Algorithm

```
1. Start
    2. Initialization
        2.1 Include necessary libraries (ZMPT101B, Tiny Gsm Client, Blynk Simple Tiny GSM,
            SoftwareSerial, LiquidCrystal_I2C), set pin modes for sensor and relay pins.
        2.2 Define constants and pins (Blynk authentication token, APN settings, sensor, and relay pins).
        2.3 Initialize the GSM module, LCD, and modem, set sensitivity for the voltage sensor.
    3. Setup Function
        3.1 Begin serial communication at 9600 bps, initialize GSM and LCD, display "Initializing GSM" on
            LCD, initialize the modem, and print the modem info to the serial monitor.
        3.2 Connect to Blynk with the provided credentials, display "Connecting To BLYNK" on LCD.
        3.3 Set pin modes for AC and DC voltage/current pins and relay pins, turn off the relay initially.
    4. Main Loop (Loop Function)
        4.1 Run Blynk.
        4.2 Call function: 'takeDCvol' to measure DC voltage, 'takeDCcur' to measure DC.
        4.3 Call function: 'takeACvol' to measure AC voltage, 'takeACcur' to measure AC.
        4.4 Call function: update LCD with the latest sensor values using 'LCD update'.
    5. Function: takeDCvol
        5.1 Read DC voltage from an analog pin, calculate actual DC voltage
        5.2 Delay for 500 ms.
    6. Function: takeDCcur,
        6.1 Read DC from analog pin, calculate actual DC, print DC to the serial monitor, delay for 500 ms.
    7. Function: takeACvol
        7.1 Obtain RMS voltage from the voltage sensor, adjust AC voltage to be within the specified range.
        7.2 Print AC voltage to the serial monitor, delay for 500 ms.
    8. Function: takeACcur
        8.1 Obtain peak-to-peak voltage for AC using 'getVPP1', calculate RMS voltage and current.
        8.2 Adjust AC to be within the specified range, print AC to the serial monitor, delay for 500 ms.
    9. Function: blynkupdate
        9.1 Update Blynk virtual pins with DC voltage, DC, AC voltage, and AC.
    10. Function: LCD update
        10.1 Clear LCD, display DC voltage and DC on the first row,
        10.2 Display AC voltage and AC on the second row.
    11. BLYNK_WRITE Function
        11.1 Check if the value from pin V4 is 1. If true, turn on the relay. Otherwise, turn off the relay.
    12. Function: obtain VPP1
        12.1 Initialize variables for reading sensor values, start timing for sampling period (500 ms).
        12.2 While the sampling period is not over, read sensor value.
        12.3 Update max and min values if the current reading is higher or lower than the current max or min.
        12.4 Calculate peak-to-peak voltage.
        12.5. Return peak-to-peak voltage.
13. End
```

5.1.4. Hardware Setup and Commissioning with Blynk App

Hardware development is essential in research involving physical systems. It enables real-world validation, prototyping, and testing, ensuring functionality and performance. In the proposed system, the hardware setup is implemented by connecting the Charge controller, MPPT, Arduino Nano Microcontroller, FIDs, 10 Watts PV panel, and 12 V DC battery. The circuit is powered by the DC voltage produced by the PV panels that have been stored by the 12 V DC battery connected through the MMPT. There is also a simple inverter that will convert the 12 V DC to a 220 V AC.

This setup enables the SIM800L to communicate with the Blynk app, transmitting real-time data from the PV system, including voltage and current measurements, to the Blynk cloud. Figure 10 shows the hardware setup using the Arduino Nano and GSM SIM800L. When no load is connected to the power source, the terminal voltage is at its maximum, 12 V, known as the open-circuit voltage (Voc), while the current is effectively zero due to the lack of a closed path for current flow. Upon switching on a load through the Blynk app, the voltage across the terminals drops from its open-circuit value due to the internal resistance of the power source and the voltage drop across the load. Concurrently, according to Ohm's Law, the current begins to flow through the circuit, with its magnitude determined by the load's resistance and the applied voltage. Figure 11 shows the display of the FID's parameters on the LCD.

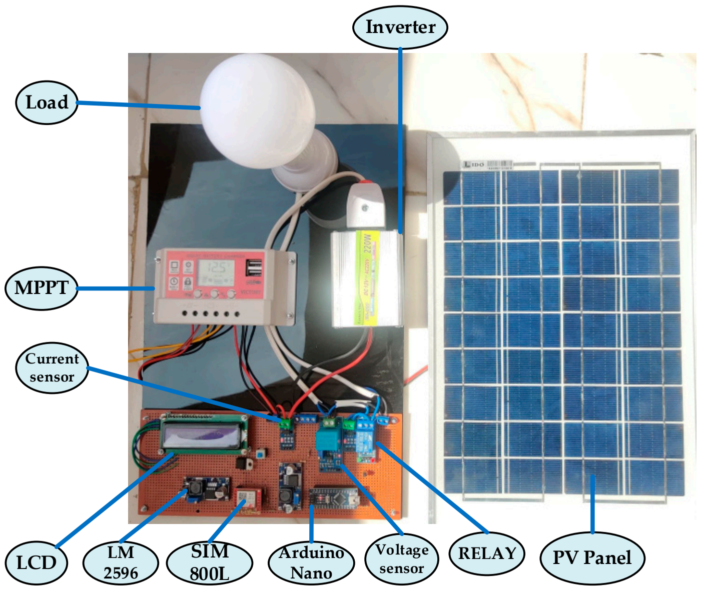


Figure 10. Hardware setup using Arduino Nano and GSM SIM800L.

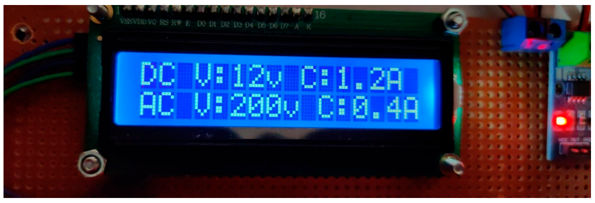
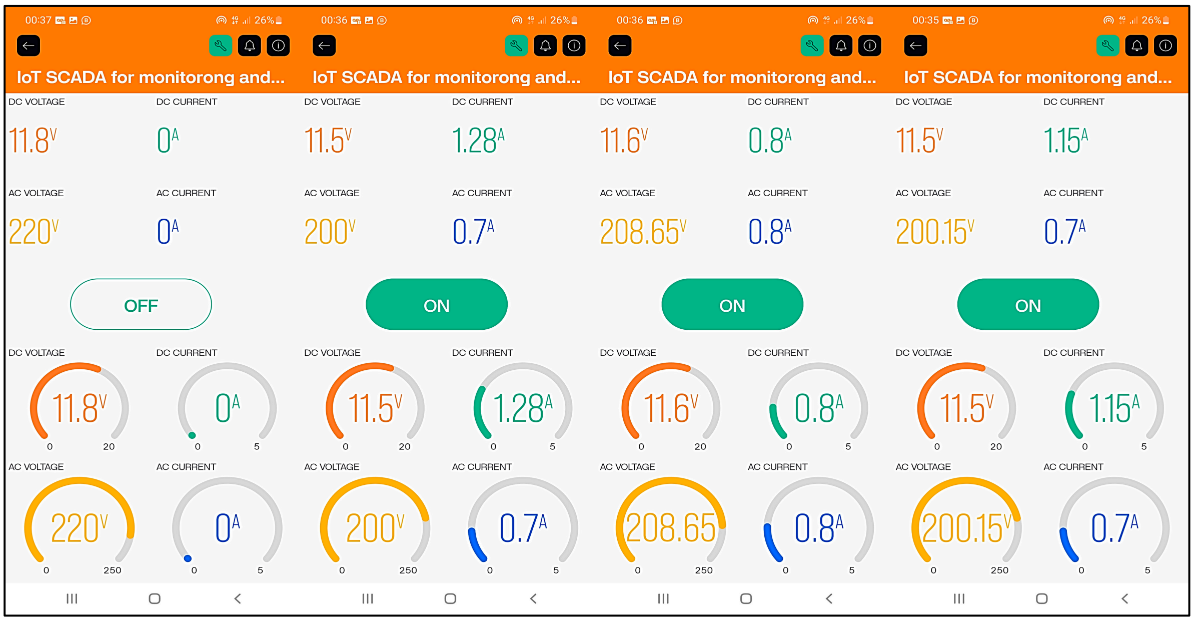


Figure 11. Display of FID's values on LCD.

The SIM800L module uses GSM for data communication, connecting to the Internet via the cellular network using the APN credentials provided by the SIM card carrier. Communication between the microcontroller and the SIM800L is established through AT commands sent over a serial interface. Figure 12 shows the PV system's FID values on the Blynk app dashboard in the "ON" and "OFF" states.



Screen	DC VOLTAGE	DC CURRENT	AC VOLTAGE	AC CURRENT	Load
00:37	11.8 V	0 A	220 V	0 A	OFF
00:36	11.5 V	1.28 A	200 V	0.7 A	ON
00:36	11.6 V	0.8 A	208.65 V	0.8 A	ON
00:35	11.5 V	1.15 A	200.15 V	0.7 A	ON

Figure 12. PV system FID values on the Blynk app dashboard.

Upon successful Internet connection, the microcontroller firmware's Blynk library facilitates data transfer to the Blynk cloud server, utilizing the TCP/IP protocol over the GSM network for reliable transmission. The Blynk app presents real-time data of the PV system, including DC and AC voltage readings and current measurements, providing a comprehensive view of system performance. Furthermore, the Blynk app enables remote control of the load connected to the PV system. The app's interface enables live monitoring, allowing for the real-time observation of electrical parameters and load status, as shown in Figure 13.

The connected load is turned "ON" or "OFF" directly through the app, using virtual pins mapped to specific digital pins on the microcontroller that control relays or switches connected to the load. The turning on and off of the connected load is managed by the Serial2.write(val); command within the BLYNK_WRITE(V4) function. With this function, the Arduino receives a value from the Blynk application and forwards this value to the relay for switching the load. Here is how this works in terms of controlling a load such as a relay:

- Receiving Input from Blynk App: The BLYNK_WRITE(V4) function is activated when a widget linked to virtual pin V4 in the Blynk app is interacted with, such as a button press. The state of this widget (usually 0 or 1) is then passed as a parameter to the function.
- Load Control on Receiving Device: The Arduino code controls a relay by sending a digital signal at a specific voltage level. In the setup() function, the relay pin is configured as an output and initially set to LOW, ensuring the relay is off. In the main loop(), the code continuously checks for incoming data via the SoftwareSerial connection. When a command is received, it reads the value; if the value is 0, it sets the relay pin to LOW (0 V), turning the relay off, and if the value is 1, it sets the relay pin to HIGH (5 V), turning the relay on. This means the Arduino operates the relay using a digital signal of either 0 V (to turn it off) or 5 V (to turn it on). This control logic allows the Arduino to dynamically manage the relay's state based on the external input.

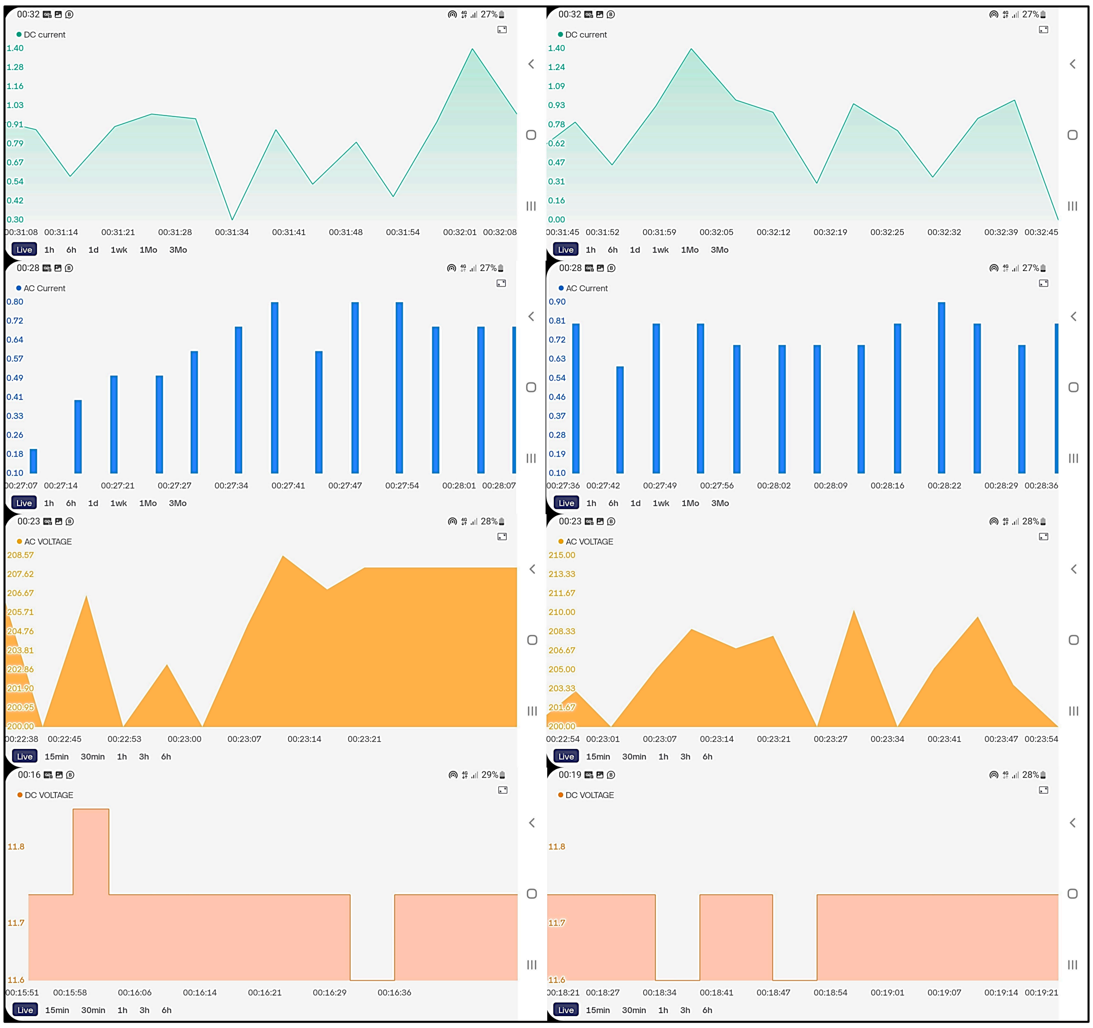


Figure 13. PV system FID monitoring on the Blynk app mobile interface.

5.2. SCADA-Based PV System Design Using Arduino Nano and ESP-32

5.2.1. Schematic Design

The schematic diagram of the proposed system was generated using EasyEDA software. This diagram mirrors Figure 11, with the SIM800L being substituted by the ESP-32. While the circuit requires a 5 V power supply, the ESP-32 operates at 3 V. Hence, to provide the ESP-32 with the requisite voltage, a voltage divider comprising 1 kΩ and 2 kΩ resistors is employed.

Figure 14 shows a circuit diagram of the Arduino Nano interface with the ESP-32 for Wi-fi connectivity. To facilitate communication between the two microcontrollers, serial communication via UART (universal asynchronous receiver–transmitter) is utilized. UART is a protocol designed for direct device communication, enabling data exchange through dedicated transmit (TX) and receive (RX) pins. Pin 10 of the Arduino Nano is directly connected to pin 17 of the ESP-32, while pin 9 of the Arduino Nano is linked to pin 16 of the ESP-32 via a voltage divider. Both devices are configured to operate at the same baud rate of 9600 bps for serial communication, ensuring seamless data transfer. The Arduino Nano sketch employs the serial or SoftwareSerial library to transmit data to the ESP32, whereas the ESP32 sketch utilizes the HardwareSerial library to receive incoming data from the Arduino Nano, facilitating efficient inter-device communication.

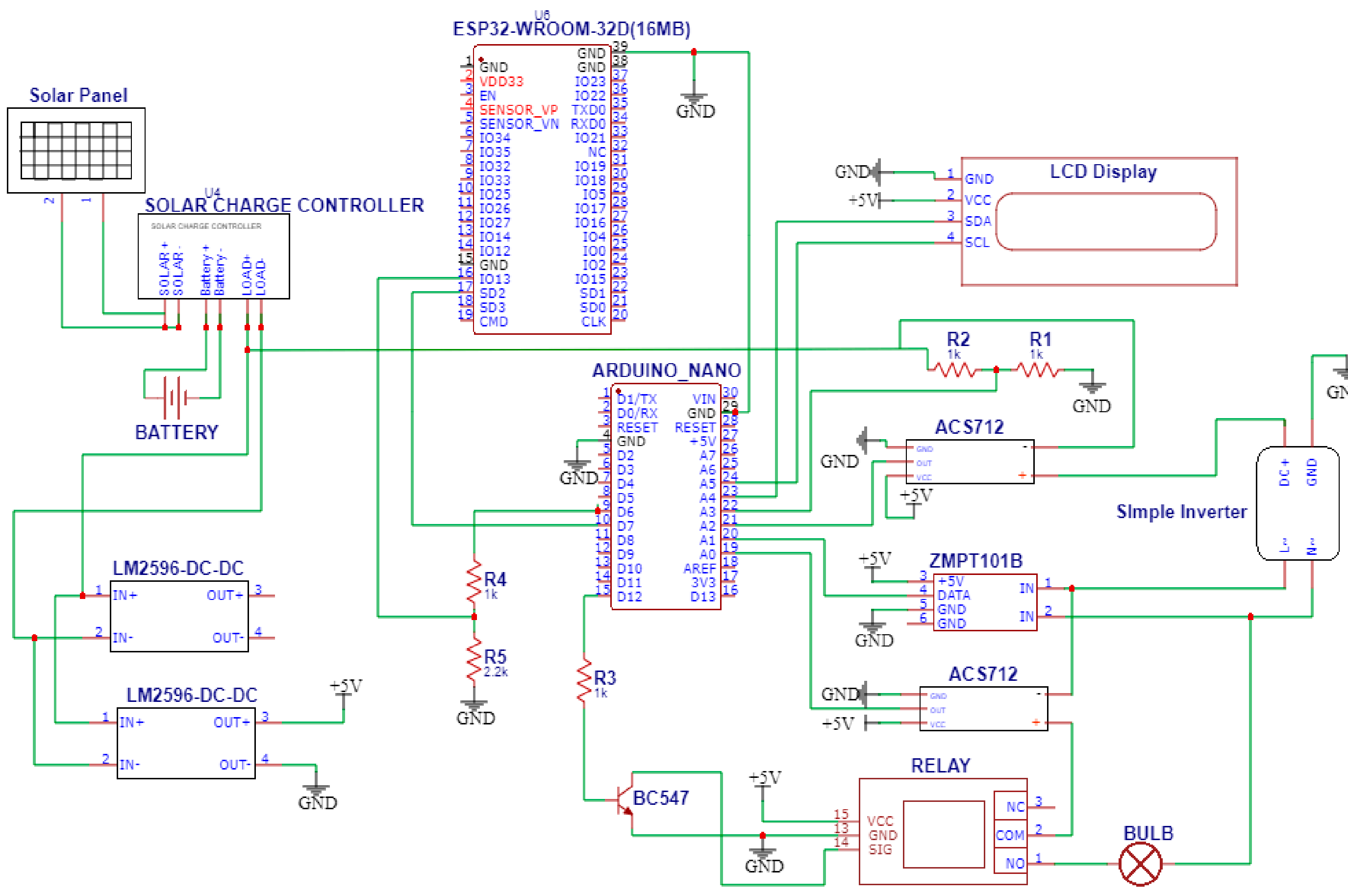


Figure 14. Circuit diagram of proposed SCADA system using Arduino Nano and ESP-32.

5.2.2. Arduino IDE Code Development for ESP-32

To code for the ESP-32 using the Arduino IDE 1.8.19, start by adding the ESP-32 Board Manager URL to the IDE Preferences under "Additional Board Manager URLs." Install the ESP32 package from the Board Manager and connect the ESP-32 to your computer via USB, selecting the appropriate COM port. Use the Library Manager to install Wi-Fi, Blynk, and HTTP Client libraries for Wi-Fi connectivity, sensor readings, and app communication. In your Arduino sketch, configure the setup function to initialize connections and settings, and use the loop function for continuous operations. This setup is tailored for a load monitoring system with an ESP32 microcontroller, integrating with the Blynk platform over Wi-Fi. The system connects to the Blynk cloud server via HTTP on port 80 using the BlynkSimpleEsp32.h library, with the Wi-Fi credentials set for network access. The sketch processes JSON data received via SoftwareSerial (Serial2), containing DC voltage (DC vol), DC (DC cur), AC voltage (AC vol), and AC (AC cur) readings. These values are displayed on the serial monitor for debugging and updated in real time on the Blynk app using Blynk.virtualWrite() in the blynk update() function. Furthermore, the sketch handles commands from the Blynk app through virtual pin V4, enabling the remote control of connected devices. This ensures accurate programming and operational functionality across diverse applications.

The Arduino code handles control commands and implements safety mechanisms through a structured approach, with robust error handling integrated into the system. During initialization, the setup function sets up serial communication, the LCD display, and sensor pins, ensuring the relay starts in a safe, known state by explicitly setting it to LOW. In the main loop function, the code checks for incoming data from the ESP module using espSerial.available() and reads the data with espSerial.read(). Based on the received value (0 or 1), the relay is controlled via digitalWrite(relay, HIGH) or digitalWrite(relay, LOW), allowing for remote operation.

The error handling mechanisms are embedded in the sensor calibration functions, such as takeDCvol(), takeDCcur(), takeACvol(), and takeACcur(). These functions read analog values from sensors and convert them to accurate voltage and current readings using calibration constants. The code includes checks for spurious readings; for example, in takeDCvol(), if the DC voltage reading is below 0.1 V, it is set to 0.0 V. This prevents erroneous low readings from affecting system operations. Further, error handling is reinforced by the design of the lcdupdate() function, which updates the LCD display with real-time sensor values, ensuring continuous and accurate monitoring. The senddata() function transmits sensor data, possibly to a remote monitoring system, allowing for real-time data logging and error detection. Additionally, the initial state setting and continuous monitoring help in identifying and mitigating errors promptly, maintaining the reliability and safety of the system. These comprehensive error handling strategies ensure that the Arduino code manages control commands effectively, maintains accurate sensor data, and prevents incorrect operations due to spurious or erroneous inputs. The complete code for the ESP-32 is available in Appendix A, providing further insight into the implementation details.

5.2.3. Hardware Setup

To evaluate the performance of the proposed Supervisory SCADA system using the ESP-32, the experimental setup was established within the Electrical and Computer Engineering Laboratory at Memorial University of Newfoundland and Labrador (MUN), Canada. On the rooftop of the building, twelve PV panels each generate 130 Watts with a maximum current of 7.6 Amperes. For the SCADA system's implementation, two PV panels were used to evaluate performance in monitoring and controlling the electrical load.

The system also includes Maximum Power Point Tracking (MPPT) and a battery bank of six 12 V, 25 A lead–acid batteries to enhance operational efficiency and effectiveness. Figure 15 illustrates the installation of the PV panels on the rooftop of the ECE building at MUN, and Figure 16 shows the hardware setup in the ECE Lab. The battery is connected to the circuit's input to power the entire system. A multimeter is attached to verify the voltage supply to the circuit. The GSM SIM800L is removed from the circuitry and an ESP-32 is installed in the circuit using a voltage divider. The Arduino Nano acts as the main controller interfacing with the FIDs and relays, while the ESP-32 provides the Wi-Fi connectivity for remote monitoring and control via the Blynk app. In cases when there is no load ("OFF"), the DC voltage is 13 V, DC is 0 A, AC voltage is 220 V, and AC is 0.1 A, as shown in Figure 17.

As soon as the load is "ON" on the web interface of the Blynk, the current starts flowing through the circuit and there is a drop of the voltage at the Load side. The LCDs FIDs values of DC voltage are 13 V, DC 1.3 A, AC Voltage 217 V, and AC 0.5 A.

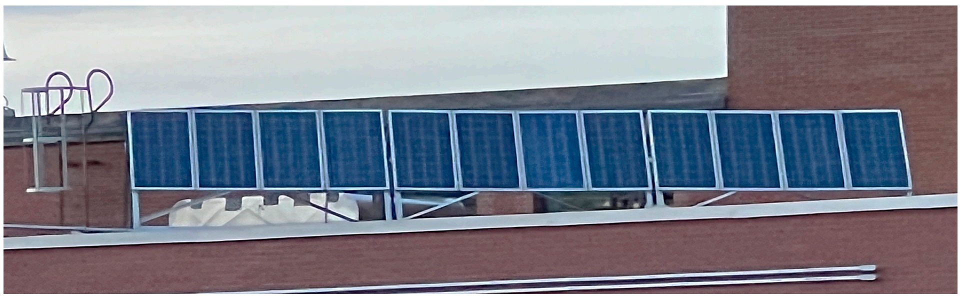


Figure 15. PV panel installation on the rooftop of the ECE building.

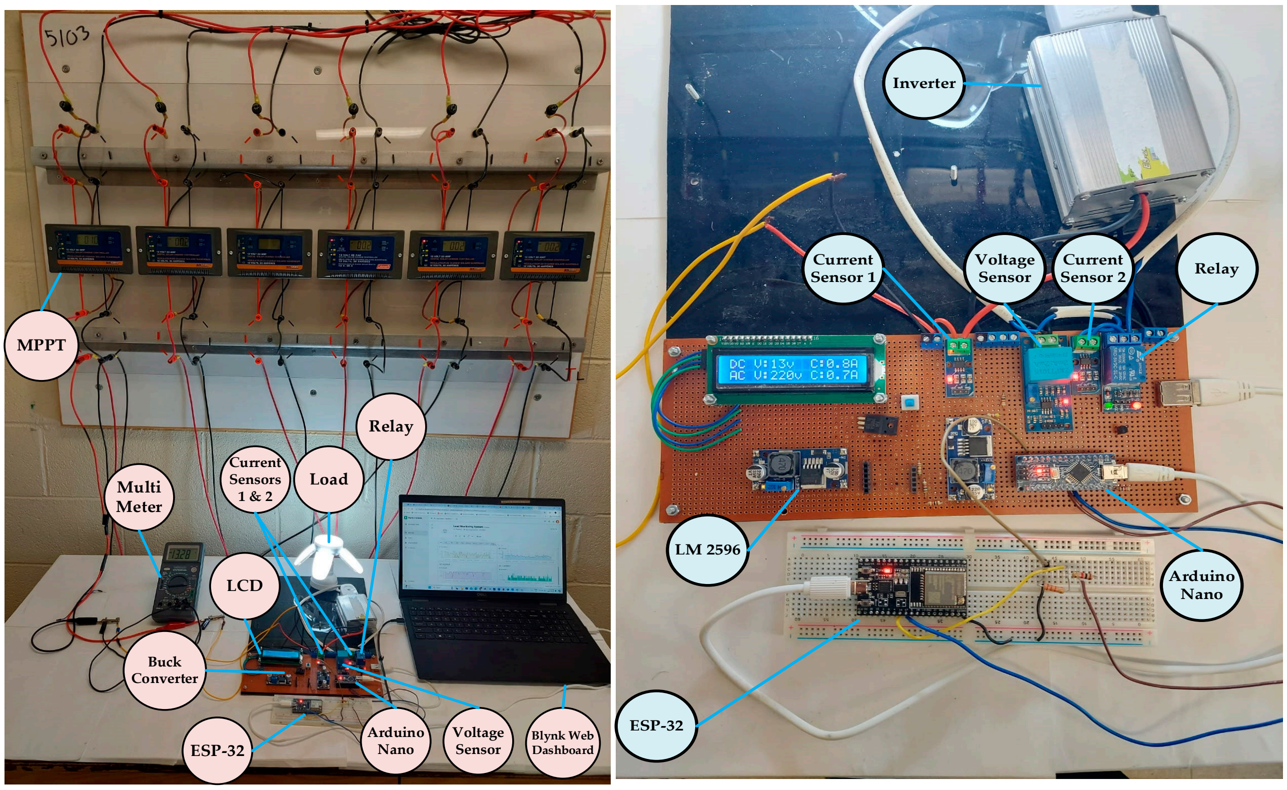


Figure 16. Experimental setup at MUN ECE building.

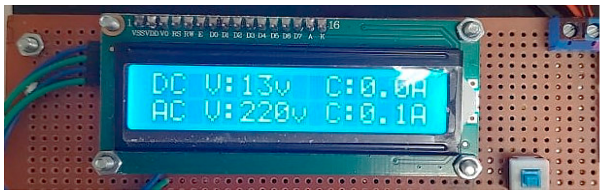


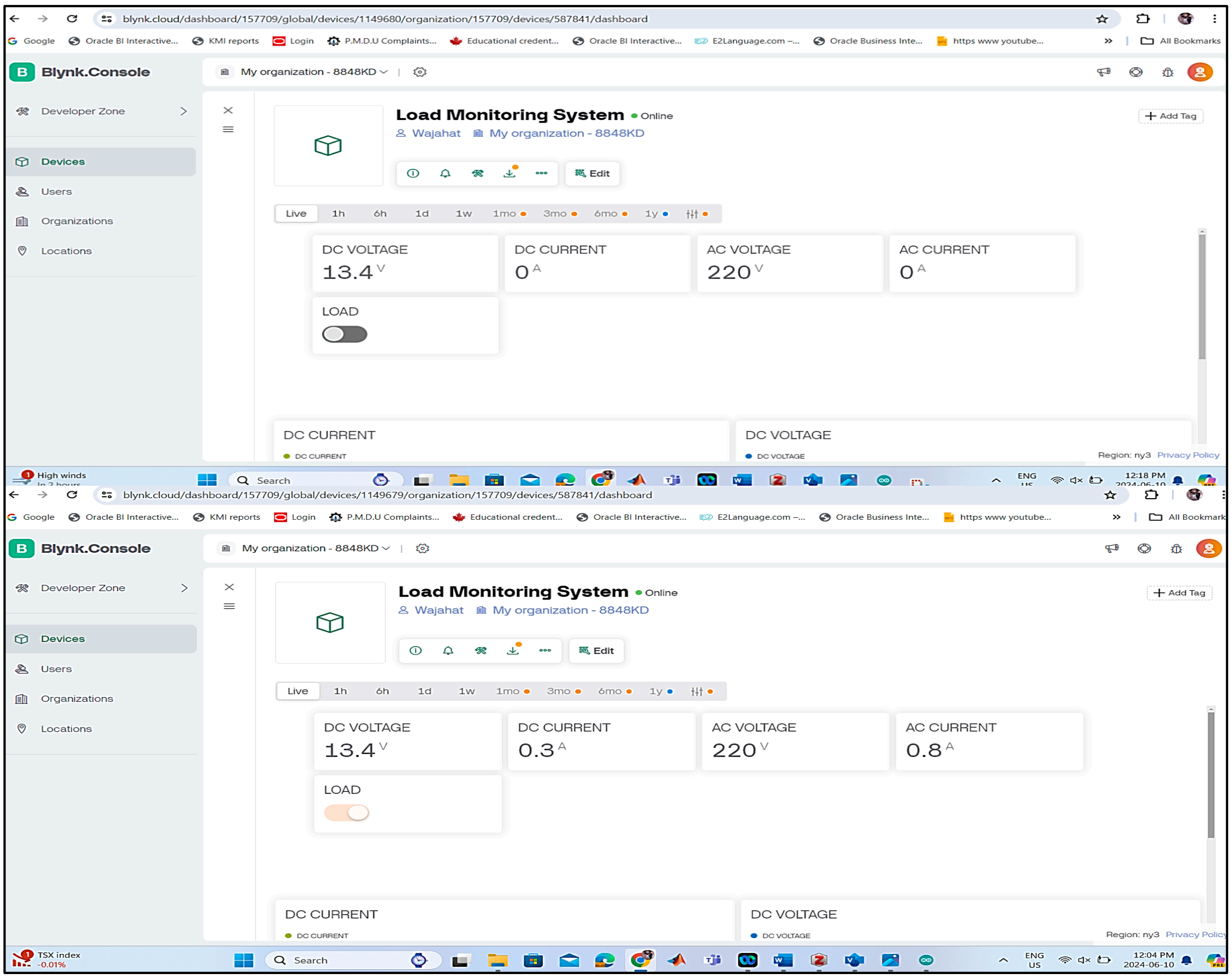
Figure 17. FID parameters on the LCD in the "OFF" state.

The voltage and current sensors measure the data and transmit them to the analog pins of the Arduino Nano, which processes these analog signals to determine the actual voltage and current values using the necessary conversion factors. Based on these data, the Arduino Nano controls load switching through a 5 V relay. The processed data are then transmitted to the ESP-32 via a UART interface, with data packets sent at a baud rate of 9600 bps between the two microcontrollers. Figure 18 shows the status of the Blynk web dashboard interface in the "OFF" and "ON" states.

Figure 19 shows the live monitoring and control of PV data on the Blynk console.

The ESP-32 receives the data from the Arduino Nano, processes them, and prepares them for transmission over the Wi-Fi network. Communication with the Blynk cloud server is established using HTTP protocols, enabling the ESP-32 to send FID data to the Blynk server, which updates the user interface on the Blynk console in real time. Control commands such as "ON" and "OFF" from the Blynk console are sent to the ESP-32, which then relays the commands to the Arduino Nano via the serial interface. The Blynk console displays real-time voltage and current readings of the PV panels, load, and battery, facilitating monitoring and control of the PV system and identifying anomalies.

The system was tested during a time of day when sunlight exposure was minimal, specifically in the evening, as shown in the attached figure. As the sunlight began to diminish, noticeable fluctuations in the DC voltage were observed, starting at approximately 6:58 PM. These fluctuations caused the DC voltage to drop from an initial value of 11 V to 10.8 V, and this downward trend continued as the light exposure further decreased. Figure 20 illustrates the system's performance under these reduced-sunlight conditions.


blynk.cloud/dashboard/157709/global/devices/1149680/organization/157709/devices/587841/dashboard

Load Monitoring System • Online
Wajahat My organization - 8848KD

DC VOLTAGE 13.4 V
DC CURRENT 0 A
AC VOLTAGE 220 V
AC CURRENT 0 A
LOAD

blynk.cloud/dashboard/157709/global/devices/1149679/organization/157709/devices/587841/dashboard

Load Monitoring System • Online
Wajahat My organization - 8848KD

DC VOLTAGE 13.4 V
DC CURRENT 0.3 A
AC VOLTAGE 220 V
AC CURRENT 0.8 A
LOAD

Figure 18. Status of Blynk web dashboard interface in "OFF" and "ON" states.

This reduction in DC voltage directly impacted the DC, which also began to exhibit abrupt fluctuations. The graph clearly shows a correlation between the decrease in DC voltage and the instability of the DC. As the voltage continued to drop, the current became increasingly erratic, reflecting the instability of the power generation due to the reduced sunlight.

Figure 21 shows the status of the voltages and currents as a result of low PV exposure. These observations underline the sensitivity of the PV system to changes in environmental conditions, such as varying sunlight intensity. These fluctuations in both the DC voltage and DC demonstrate the dynamic nature of the system's performance under less-than-optimal lighting conditions, providing valuable insights into how the system behaves in real-world scenarios.

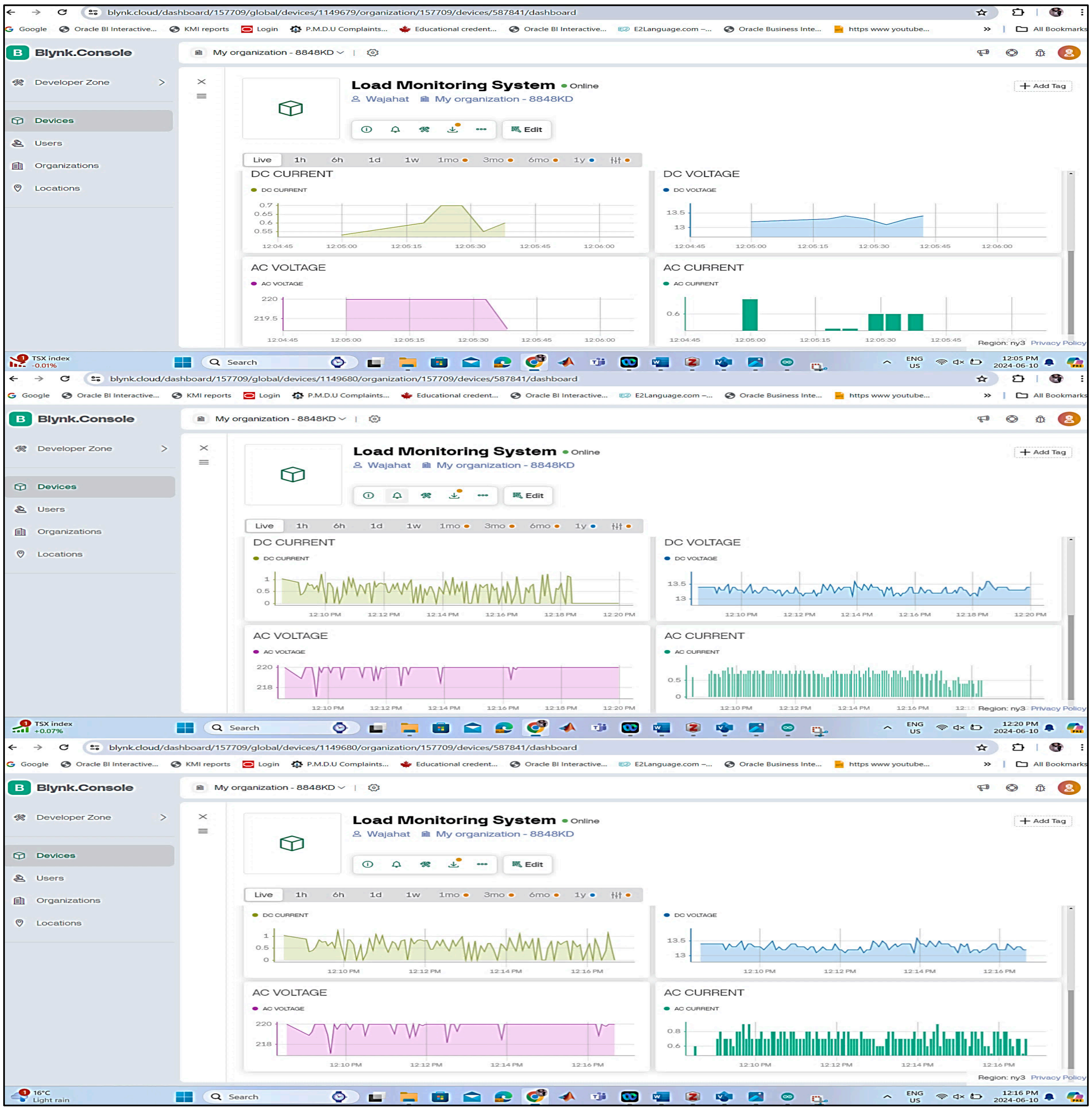


Figure 19. Monitoring and control of PV system on Blynk console dashboard using ESP-32 and Arduino Nano.

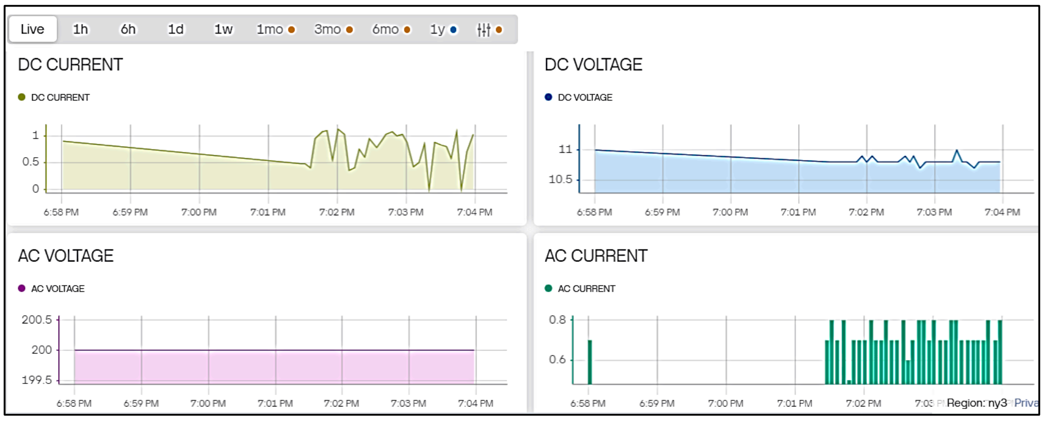


Figure 20. PV system parameters under reduced sunlight.

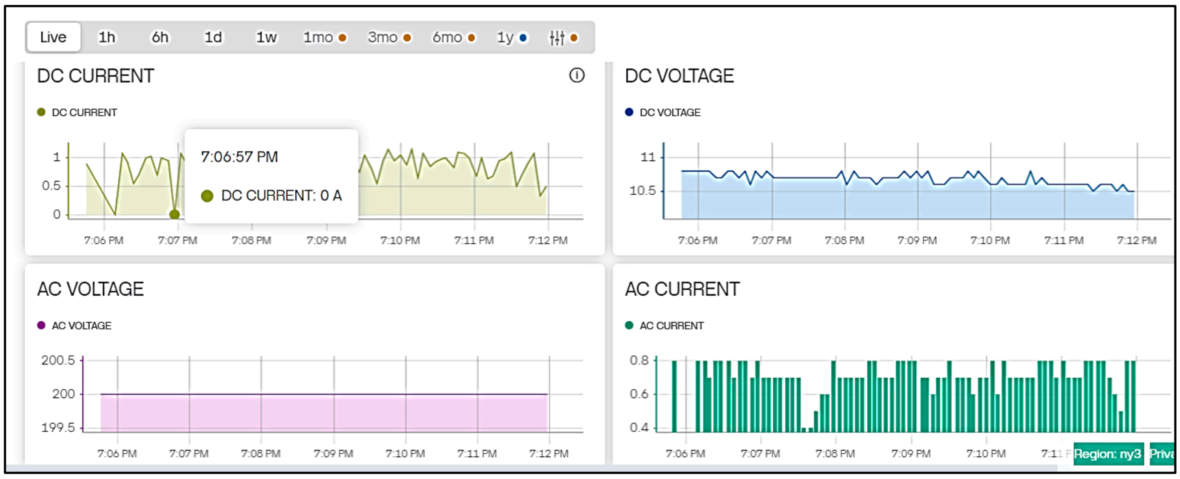


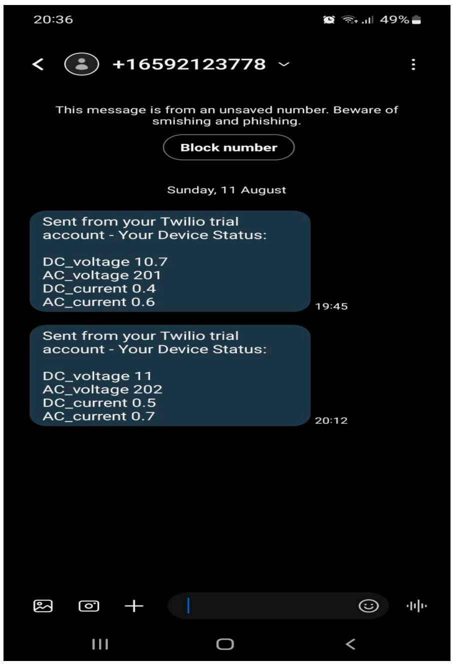
Figure 21. Status of DC Voltage and DC.

5.2.4. Notification of PV System Parameters

The designed system is fully capable of notifying users about the PV system parameters both through SMS notifications and via visual inspection using an LCD installed in the circuit. The LCD provides a real-time display of critical parameters, including DC and AC voltage, as well as DC and AC. Any malfunction or abnormal behavior in the system is immediately displayed, enabling the user to identify and rectify faults promptly.

In addition to on-site monitoring, the system also supports remote notification via SMS, ensuring that users can stay informed about system performance even when they are away. This functionality is enabled through the integration of the Twilio platform, an open-source cloud communication service. Twilio's robust Application Program Interface (API) allows for the seamless integration of messaging, voice, and video capabilities into applications. Specifically, within this SCADA system, Twilio's SMS API is utilized to send real-time alerts regarding system status, faults, or operational changes directly to the user's mobile phone, ensuring immediate awareness and a quick response. By incorporating Twilio's services, the system significantly enhances its monitoring and communication capabilities, facilitating more responsive and efficient management of the PV system's operational environment. Figure 22 shows the notification of system parameters via SMS under testing conditions.

The sendTwilioMessage() function is designed to transmit an SMS containing critical electrical measurements—such as AC and DC voltage and AC and DC readings—to a specified phone number via the Twilio API. Initially, the function verifies the presence of an active WiFi connection; if WiFi is not connected, it exits and notifies the user via the serial monitor. If connected, the function proceeds by preparing the necessary components for message transmission. This includes constructing the Twilio API endpoint URL and encoding the account SID and authentication token for authorization. The electrical measurements are formatted into a clear and readable message, which is then embedded in the body of the HTTP POST request alongside the recipient's and sender's phone numbers. Upon sending the HTTP request to Twilio's API, the function captures and outputs the response status code and content to the serial monitor, providing feedback on the success of the message transmission. If the WiFi connection is absent, the function immediately reports the issue without attempting to send the SMS. Figure 23 illustrates the Arduino IDE code implementation of the Twilio function.


20:36
+16592123778
This message is from an unsaved number. Beware of smishing and phishing.
Block number
Sunday, 11 August
19:45
20:12

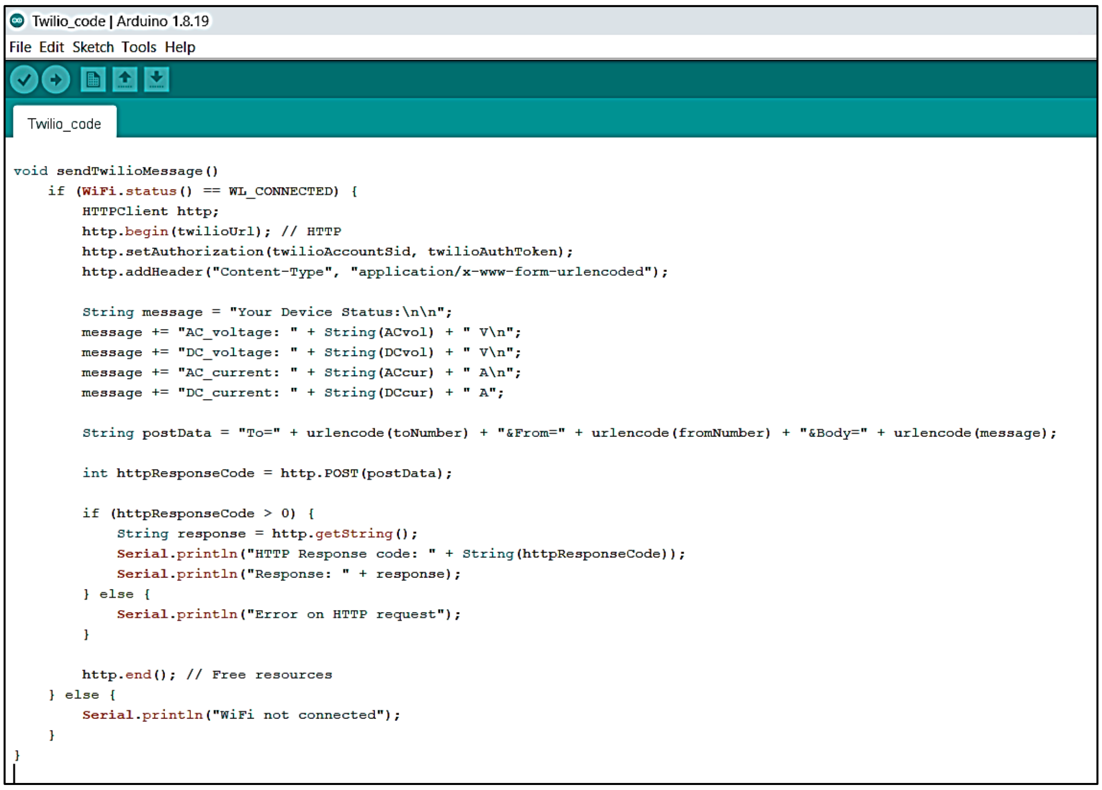
Sent from your Twilio trial account - Your Device Status:

DC_voltage 10.7
AC_voltage 201
DC_current 0.4
AC_current 0.6

Sent from your Twilio trial account - Your Device Status:

DC_voltage 11
AC_voltage 202
DC_current 0.5
AC_current 0.7

Figure 22. Notification of PV system parameters via SMS under testing conditions.



```
void sendTwilioMessage()
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(twilioUrl); // HTTP
        http.setAuthorization(twilioAccountSid, twilioAuthToken);
        http.addHeader("Content-Type", "application/x-www-form-urlencoded");

        String message = "Your Device Status:\n\n";
        message += "AC_voltage: " + String(ACvol) + " V\n";
        message += "DC_voltage: " + String(DCvol) + " V\n";
        message += "AC_current: " + String(ACcur) + " A\n";
        message += "DC_current: " + String(DCcur) + " A";

        String postData = "To=" + urlencode(toNumber) + "&From=" + urlencode(fromNumber) + "&Body=" + urlencode(message);

        int httpResponseCode = http.POST(postData);

        if (httpResponseCode > 0) {
            String response = http.getString();
            Serial.println("HTTP Response code: " + String(httpResponseCode));
            Serial.println("Response: " + response);
        } else {
            Serial.println("Error on HTTP request");
        }

        http.end(); // Free resources
    } else {
        Serial.println("WiFi not connected");
    }
}
```

Figure 23. Arduino IDE code of the Twilio API.

6. Discussion

Building upon our successful experimental outcomes, here are the key attributes of the proposed IoT-enabled SCADA system for overseeing and managing the PV system.

- Framework: The proposed SCADA system represents the latest IoT-based framework, incorporating essential components for a SCADA system. It consists of FIDs (ZMPT101B, ACS712, relay) and an RTU built on an Arduino Nano, a GSM SIM800L, and an ESP-32, all serially interfaced (UART) with the RTU for communication over GSM and Wi-Fi. Additionally, it features an MTU configured via the Blynk and the TCP/IP for GSM and HTTP protocols over a local network.

- Graphical User Interface (GUI): The Blynk platform is utilized in developing the SCADA framework, simplifying the intricate process of data communication via HTTP and the HMI design, thereby enabling straightforward implementation of the proposed SCADA system.
- Remote Supervision and Monitoring: The proposed IoT-driven SCADA system incorporates remote monitoring and control features. Users can access the SCADA system through the Blynk app and console HMIs via the local network and mobile application.
- Reliability of the System: The sensors and relays (FIDs) are interfaced with the Arduino Nano (RTU), ensuring that the Blynk app (MTU) remains isolated from the PV system. This setup prevents potential damage to the RTU during extreme conditions.
- Data Retention: The proposed SCADA system integrates local storage via the Blynk app, enabling the visualization of historical data through its graphical user interface (GUI). This feature facilitates the analysis of essential data trends to support informed decision-making.
- Enhanced Remote Monitoring and Notification System: The system successfully integrates a function to send real-time SMS alerts containing critical electrical measurements, such as AC and DC voltage and AC and DC, directly to the user. This capability enhances remote monitoring and ensures the immediate notification of system parameters, significantly improving the responsiveness and overall reliability of the PV system.
- Cost-Effective and Open-Source: By integrating the Blynk platform with the Arduino IDE, our SCADA system optimizes cost-effectiveness through freely available open-source software. This strategy eliminates the burden of costly proprietary licenses while providing a flexible framework for development and maintenance. Hardware components such as FIDs and the RTU are readily accessible in local markets, streamlining deployment and enhancing economic viability. The detailed cost breakdown in Table 6 reveals a total system cost of CAD 35.52, coupled with a low average power consumption of 3.39 watts, ensuring efficient, sustainable operation 24/7.

Table 6. Price and power consumption of elements used in SCADA system.

Quantity (No.)	Elements	Price (CAD)	Power Consumption (W)
1	Arduino Nano	4.91	0.2
1	ESP-32	7.37	0.66
1	LCD display	1.39	0.135
2	ACS 712 current	6.38	0.012
1	ZMPT101B	2.45	0.025
1	GSM-SIM800L	5.92	1.5
1	5 V relay	2.28	0.05
2	Buck converter LM2596	4.82	0.88
	Total	35.52	3.462

7. Conclusions

Rural communities in Pakistan face electricity shortages due to high initial infrastructure costs. Given these communities' diverse terrains, photovoltaic (PV) systems are crucial for electrification—components like PV panels, MPPT controllers, batteries, inverters, and loads from the PV system. Field Interface Devices (FIDs) monitor and control PV parameters, aiding in effective electrical consumption management. There is a growing need for a versatile, scalable system to handle these tasks efficiently. This study achieves significant technical milestones in developing an economical IoT-based SCADA system for PV monitoring and control. Integrating IoT technology with traditional SCADA functionalities, the system enables remote monitoring and control through the Blynk app and console, leveraging Arduino Nano, GSM SIM800L, and ESP-32 microcontrollers. It offers a cost-effective solution with a total expense of CAD 35.52 and low power consumption of 3.462 W, making it an affordable alternative for rural communities. The dual-mode communication capability via GSM and Wi-Fi ensures reliable data transfer in various network conditions, enhancing flexibility and robustness. Utilizing open-source platforms like Blynk and Arduino IDE promotes accessibility and customization, encouraging community-driven innovation. The system provides comprehensive monitoring of critical PV parameters, allows for the remote control of electrical loads, and features real-time SMS notifications for critical system alerts, ensuring immediate user awareness and quick response to any issues. Real-world validation demonstrated its reliability and effectiveness, particularly in rural settings. The user-friendly interface of the Blynk app and web dashboard facilitates easy access to real-time data and system management, enhancing user engagement and operational efficiency. These achievements underscore the potential of IoT-based SCADA systems to revolutionize renewable energy management, particularly in underserved areas.

Future research should focus on enhancing the system's ability to integrate a broader range of energy sources and leverage advanced predictive analytics to optimize energy usage and generation. Additionally, there is significant potential to support the cybersecurity measures of IoT components, ensuring the reliability and security of the energy management system. Although the current user interface is functional, it would benefit from improved visualization tools to facilitate the interpretation of complex data.

Research should also explore the system's scalability by deploying it in larger environments and implementing advanced cybersecurity protocols to protect against unauthorized access and data breaches. Developing solutions for long-term data storage and incorporating machine learning algorithms for predictive maintenance and optimization will further enhance the system's efficiency. Moreover, expanding the system to include redundancy features and automated fault detection will significantly improve its reliability and robustness.

Author Contributions: Conceptualization, W.K.; System Design, W.K.; Methodology, W.K.; software, W.K.; validation, W.K.; formal analysis, W.K., M.J. and Q.A.; investigation, W.K.; resources, A.A.K., W.K. and M.J.; data curation, W.K.; writing—original draft preparation, W.K.; writing—review and editing, A.A.K., M.J. and Q.A.; visualization, W.K.; supervision, A.A.K. and M.J.; project administration, A.A.K., M.J. and Q.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: This study incorporates its original findings within the article, and any additional questions can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript.

SCADA	Supervisory Control and Data Acquisition
RTU	Remote Terminal Unit
MTU	Master Terminal Unit
HTTP	Hypertext transfer protocol
FIDs	Field Instrument Devices
UART	Universal asynchronous receiver–transmitter
IoT	Internet of Things
GSM	Global System for Mobile Communications
HPS	Hybrid Power System
IDE	Integrated Development Environment

Appendix A

```
#define BLYNK_PRINT Serial; #include <WiFi.h>; #include <BlynkSimpleEsp32.h>
#include <ArduinoJson.h>; #define BLYNK_TEMPLATE_ID “TMPL2th_r2ugs”
#define BLYNK_TEMPLATE_NAME “Load Monitoring System”; #define BLYNK_AUTH_TOKEN
“6EF9dE0xZalWaVG-AtZ-q9ZHGcGtuvUa”; char auth [] = “6EF9dE0xZalWaVG-AtZ-
q9ZHGcGtuvUa”; char ssid [] = “xxxxxx”; char pass[] = “xxxxxx”; float DCvol,DCcur,
ACvol,ACcur; void setup () {Serial. Begin(9600); Serial2.begin(9600); Blynk.begin(auth,
ssid, pass,”blynk.cloud”,80); void loop() {Blynk.run(); if (Serial2.available() > 0); // Read
the data from Arduino String jsonString = Serial2.readStringUntil(‘\n’); // Parse JSON;
DynamicJsonDocument doc(200); deserializeJson(doc, jsonString); DCvol = doc[“DCvol”];
DCcur = doc[“DCcur”]; ACvol = doc[“ACvol”]; ACcur = doc[“ACcur”]; // Extract
values Serial.print(“DCvol = ”); Serial.println(DCvol,1); Serial.print(“DCcur = ”); Se-
rial.println(DCcur,1); Serial.print(“ACvol = ”); Serial.println(ACvol,1); Serial.print(“ACcur
= ”); Serial.println(ACcur,1); blynkupdate(); delay(1000); // Adjust delay as needed void
blynkupdate(); Blynk.virtualWrite(0, DCvol); Blynk.virtualWrite(1, DCcur); Blynk.virtualWrite(2,
ACvol); Blynk.virtualWrite(3, ACcur); BLYNK_WRITE(V4) {
int val = param. asInt(); Serial.print(“Recieved “); Serial.println(val); Serial2.write(val);
```

References

1. Ahmed, M.M.; Qays, M.O.; Abu-Siada, A.; Muyeen, S.M.; Hossain, M.L. Cost-Effective Design of IoT-Based Smart Household Distribution System. *Designs* **2021**, *5*, 55. [CrossRef]
2. Abouobaida, H.; De Oliveira-Assis, L.; Soares-Ramos, E.P.P.; Mahmoudi, H.; Guerrero, J.M.; Jamil, M. Energy management and control strategy of DC microgrid based hybrid storage system. *Simul. Model. Pract. Theory* **2023**, *124*, 102726. [CrossRef]
3. He, W.; Iqbal, M.T. A Novel Design of a Low-Cost SCADA System for Monitoring Standalone Photovoltaic Systems. *J. Electron. Electr. Eng.* **2024**. [CrossRef]
4. Stökler, S.; Schillings, C.; Kraas, B. Solar Resource Assessment Study for Pakistan. *Renew. Sustain. Energy Rev.* **2016**, *58*, 1184–1188. [CrossRef]
5. Grainger, C.A.; Zhang, F. Electricity Shortages and Manufacturing Productivity in Pakistan. *Energy Policy* **2019**, *132*, 1000–1008. [CrossRef]
6. Asian Development Bank. Energy Crisis in Pakistan: Implications for Economic Growth. In *Asian Development Outlook April 2023*; Asian Development Bank: Mandaluyong, Philippines, 2023. Available online: https://www.adb.org/sites/default/files/publication/863591/pak-ado-april-2023.pdf (accessed on 31 July 2024).
7. International Energy Agency (IEA). *Energy Policies beyond IEA Countries: Pakistan 2022*; IEA: Paris, France, 2023. Available online: https://www.iea.org/countries/pakistan (accessed on 31 July 2024).
8. Khan, H.A.; Ahmad, H.F.; Nasir, M.; Nadeem, M.F.; Zaffar, N.A. Decentralized Electric Power Delivery for Rural Electrification in Pakistan. *Energy Policy* **2018**, *120*, 312–323. [CrossRef]
9. Nasir, M.; Anees, M.; Khan, H.A.; Khan, I.; Xu, Y.; Guerrero, J.M. Integration and Decentralized Control of Standalone Solar Home Systems for Off-Grid Community Applications. *IEEE Trans. Ind. Appl.* **2019**, *55*, 7240–7250. [CrossRef]
10. Akhtar, T.; Rehman, A.U.; Jamil, M.; Gilani, S.O. Impact of an Energy Monitoring System on the Energy Efficiency of an Automobile Factory: A Case Study. *Energies* **2020**, *13*, 2577. [CrossRef]
11. Aghenta, L.O.; Iqbal, M.T. Development of an IoT-Based Open-Source SCADA System for PV System Monitoring. In Proceedings of the 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 5–8 May 2019; pp. 1–4. [CrossRef]
12. Oton, C.N.; Iqbal, M.T. Low-Cost Open Source IoT-Based SCADA System for a BTS Site Using ESP32 and Arduino IoT Cloud. In Proceedings of the 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 1–4 December 2021; pp. 0681–0685. [CrossRef]
13. Kao, K.-C.; Chieng, W.-H.; Jeng, S.-L. Design and development of an IoT-based web application for an intelligent remote SCADA system. IOP Conf. *Ser. Mater. Sci. Eng.* **2018**, *323*, 012025. [CrossRef]
14. Pramudhita, A.N.; Asmara, R.A.; Sirajuddin, I.; Rohadi, E. Internet of Things Integration in Smart Grid. In Proceedings of the 2018 International Conference on Applied Science and Technology (iCAST), Manado, Indonesia, 26–27 October 2018; pp. 718–722. [CrossRef]
15. Wang, Q.; Zhu, X.; Ni, Y.; Gu, L.; Zhu, H. Blockchain for the IoT and industrial IoT: A review. *Internet Things* **2020**, *10*, 100081. [CrossRef]
16. Huda, S.; Yearwood, J.; Hassan, M.M.; Almogren, A. Securing the operations in SCADA-IoT platform based industrial control system using an ensemble of deep belief networks. *Appl. Soft Comput.* **2018**, *71*, 66–77. [CrossRef]
17. Al-Ali, A.R.; Zualkernan, I.A.; Rashid, M.; Gupta, R.; Alikarar, M. A smart home energy management system using IoT and big data analytics approach. *IEEE Trans. Consum. Electron.* **2017**, *63*, 426–434. [CrossRef]

18. Devi, M.; Muralidharan, S.; Elakiya, R.; Monica, M. Design and Implementation of a Smart Home Energy Management System Using IoT and Machine Learning. *E3S Web Conf.* **2023**, *387*, 04005. [CrossRef]
19. Moraes, T.; Nogueira, B.; Lira, V.; Tavares, E. Performance Comparison of IoT Communication Protocols. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 3249–3254. [CrossRef]
20. Jaloudi, S. Open-source software of smart city protocols status and challenges. In Proceedings of the 2015 International Conference on Open-Source Software Computing (OSSCOM), Amman, Jordan, 10–13 September 2015; pp. 1–6. [CrossRef]
21. Ahsan, L.; Baig, M.J.A.; Iqbal, M.T. Low-Cost, Open-Source, Emoncms-Based SCADA System for a Large Grid-Connected PV System. *Sensors* **2022**, *22*, 6733. [CrossRef]
22. Duair, J.J.; Majeed, A.I.; Ali, G.M. Design and Implementation of IoT-Based SCADA for a Multi Microgrid System. *ECS Trans.* **2022**, *107*, 17345–17359. [CrossRef]
23. Uddin, S.U.; Baig, M.J.A.; Iqbal, M.T. Design and Implementation of an Open-Source SCADA System for a Community Solar-Powered Reverse Osmosis System. *Sensors* **2022**, *22*, 9631. [CrossRef] [PubMed]
24. Asgher, M.N.; Iqbal, M.T. Development of a Low-Cost, Open-Source LoRA-based SCADA System for Remote Monitoring of a Hybrid Power System for an Offshore Aquaculture Site in Newfoundland. *Eur. J. Electr. Eng. Comput. Sci.* **2023**, *7*, 65–73. [CrossRef]
25. Hamied, A.; Mellit, A.; Benghanem, M.; Boubaker, S. IoT-Based Low-Cost Photovoltaic Monitoring for a Greenhouse Farm in an Arid Region. *Energies* **2023**, *16*, 3860. [CrossRef]
26. 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. [CrossRef]
27. Khalid, W.; Awais, Q.; Jamil, M.; Khan, A.A. Dynamic Simulation and Optimization of Off-Grid Hybrid Power Systems for Sustainable Rural Development. *Electronics* **2024**, *13*, 2487. [CrossRef]
28. Santosa, E.S.B.; Waluyanti, S. Teaching Microcontrollers using Arduino Nano Based Quadcopter. *J. Phys. Conf. Ser.* **2019**, *1413*, 012003. [CrossRef]
29. Arduino Nano Pinout. Available online: https://www.electronicshub.org/arduino-nano-pinout/ (accessed on 2 June 2024).
30. El Hammoumi, A.; Motahhir, S.; Chalh, A.; El Ghzizal, A.; Derouich, A. Low-Cost Virtual Instrumentation of PV Panel Characteristics Using Excel and Arduino in Comparison with Traditional Instrumentation. *Renew. Sustain. Energy Rev.* **2018**, *5*, 3. [CrossRef]
31. Omidi, S.A.; Baig, M.J.A.; Iqbal, M.T. Design and Implementation of Node-Red Based Open-Source SCADA Architecture for a Hybrid Power System. *Energies* **2023**, *16*, 2092. [CrossRef]
32. Alsumady, M.O.; Alturk, Y.K.; Dagamseh, A.; Tantawi, M. Controlling of DC-DC Buck Converters Using Microcontrollers. *Int. J. Circuits Syst. Signal Process.* **2021**, *15*, 197–202. [CrossRef]
33. Abidin, Z.; Muttaqin, A.; Maulana, E.; Ramadhan, M.G. Buck Converter Optimization Using P&O Algorithm for PV System Based Battery Charger. *Int. J. Power Electron. Drive Syst. IJPEDS* **2020**, *11*, 844. [CrossRef]
34. Zaveri, K.A.; Amin, M.H.; Amin, M.S.; Patel, M.R. IoT Based Real Time Low Cost Home Quarantine Patient Aid System Using Blynk App. *J. Phys. Conf. Ser.* **2021**, *2007*, 012014. [CrossRef]
35. Jamlos, M.A.; Moorali, J.; Mustafa, W.A.W.; Idrus, S.Z.S. Automotive Collision Avoidance System (ACAS) Application. *J. Phys. Conf. Ser.* **2021**, *1874*, 012037. [CrossRef]
36. Budijono, S. Margaretta Smart Warning System Using SIM800L and ESP32. *IOP Conf. Ser. Earth Environ. Sci.* **2021**, *794*, 012132. [CrossRef]
37. Sugiyanti, I. Design of ATM Crime Monitoring System Based on MQTT Protocol Using SIM800L and Arduino Mega 2560. *INA-Rxiv Pap.* **2019**. [CrossRef]
38. Espressif. ESP-IDF Programming Guide: Get Started with ESP32 DevKitC. Available online: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html (accessed on 8 June 2024).
39. Hercog, D.; Lerher, T.; Truntič, M.; Težak, O. Design and Implementation of ESP32-Based IoT Devices. *Sensors* **2023**, *23*, 6739. [CrossRef]
40. Hossain, J.; Algeelani, N.A.; Al-Masoodi, A.H.H.; Kadir, A.F.A. Solar-Wind Power Generation System for Street Lighting Using Internet of Things. *Indonesia. J. Electr. Eng. Comput. Sci.* **2022**, *26*, 639. [CrossRef]
41. Vidhya, R.G.; Rani, B.K.; Singh, K.; Kalpanadevi, D.; Patra, J.P.; Srinivas, T.A.S. An Effective Evaluation of SONARS Using Arduino and Display on Processing IDE. In Proceedings of the 2022 International Conference on Computer, Power and Communications (ICCPC), Chennai, India, 14–16 December 2022; IEEE: New York, NY, USA, 2022; pp. 500–505. [CrossRef]
42. Divya, P.; Bhavana, N.; George, M. Arduino Based Obstacle Detecting System. *SSRN Electron. J.* **2020**. [CrossRef]