

# **Design of Supervisory Controllers and Ultra-low Power Data Loggers for Hybrid Power Systems**

by © Wei He

A Thesis submitted to the School of Graduate Studies  
in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October 17, 2024

St. John's Newfoundland and Labrador

## **Abstract**

Standalone photovoltaic (PV) systems are crucial in the transition towards sustainable energy, offering reductions in fossil fuel dependence and lower electricity costs. Optimizing these systems' performance requires effective data loggers to monitor and record operational data. Furthermore, proprietary and non-configurable data logger available in the current market impede their massive adoption. In this thesis four different designs of the data logging system are proposed to solve the existing data logging system problems. The first design emphasizes low power consumption and comprehensive data management. Key low-power strategies are presented, including reducing supply voltage and CPU frequency, using a data buffering mechanism, and optimizing the Wi-Fi connection interval. In the second design the novel Human-Machine Interface (HMI) and data storage solution are proposed. A mobile application as the HMI receives data via Bluetooth Low Energy, and then carries out historical data analysis and trend identification. Data is also intermittently transferred to a website via Wi-Fi for substantial remote storage free of charge. Following is the third design, which put emphasis on supervisory control of the load by using a relay, the deployment of Master Terminal Unit (MTU), and the use of the middleware Node-RED. Data is transmitted from an ESP32-E microcontroller to a Banana Pi M4 Berry (MTU) via the Message Queuing Telemetry Transport (MQTT) protocol. The Node-RED platform on the MTU provides customizable dashboards. In the last design, two Internet of Things (IoT) platforms are utilized to build a scalable IoT-Supervisory Control and Data Acquisition (SCADA) system, which facilitates the implementation of a complete five-layer IoT architecture. The load images are available on a web camera server for inspection. Verified under the laboratory setup, the ultra-low power, open-source, IoT-based data

loggers have showcased their great potential in monitoring, controlling, and logging PV system operational data and process.

## **Acknowledgements**

First and foremost, I want to express my deepest gratitude for Prof. Tariq Iqbal's guidance, support, and encouragement throughout my academic journey. His expertise, patience, and dedication have been invaluable to me, and I am profoundly thankful for the opportunity to have worked under his supervision. Sometimes I delayed my submission, but he always gave his feedback to me as soon as possible. His insightful feedback and constructive criticism have significantly contributed to the development of my research skills and the quality of my work. The knowledge and experience I have gained through his mentorship have been instrumental in shaping my academic and professional growth. I greatly appreciate the time and effort he has invested in reviewing my work, providing detailed feedback, and offering thoughtful suggestions. His commitment to my success has been evident in every interaction we have had, and it has motivated me to strive for excellence in my studies and research. I want to thank Prof. Tariq Iqbal once again for his unwavering support and guidance. I look forward to continuing our collaboration and learning from you in the future. In addition, I want to thank Dr. Mirza Jabbar Aziz Baig for his selfless tutorial sincerely. He is another important person in my research career. Without him, I could not achieve what I have done. Besides, I want to thank my parents, the other family members and close friends for their encouragement. I am eternally grateful to have such amazing people in my life. Without them, I cannot walk through this incredible journey myself. As I move forward, I carry with me the lessons, love, and strength you have given me, and I look forward to sharing future successes with you all.

# Table of Contents

<b>ABSTRACT.....</b>	<b>II</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>III</b>
<b>TABLE OF CONTENTS .....</b>	<b>IV</b>
<b>LIST OF TABLES .....</b>	<b>IX</b>
<b>LIST OF FIGURES .....</b>	<b>XI</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>XV</b>
<b>CHAPTER 1. INTRODUCTION .....</b>	<b>16</b>
1.1 Introduction .....	16
1.1.1 Non-renewable Energy and Renewable Energy.....	16
1.1.2 Photovoltaic System .....	18
1.1.3 Data Logging.....	21
1.2 Literature Review .....	24
1.3 Research Objectives.....	42
1.4 Thesis Structure .....	43
1.5 Co-authorship Statement.....	43
<b>CHAPTER 2. POWER CONSUMPTION MINIMIZATION OF A LOW-COST IOT DATA LOGGER FOR PHOTOVOLTAIC SYSTEM .....</b>	<b>45</b>
2.1 Introduction .....	45
2.2 Literature Review .....	47



2.3 Data Logger.....	50
2.3.1 Components Connection.....	50
2.3.2 Microcontroller FireBeetle 2 ESP32-E.....	51
2.3.3 Peripherals.....	55
2.4 Low Power Techniques.....	59
2.4.1 Impact of Voltage.....	60
2.4.2 Impact of Frequency.....	62
2.4.3 Power Saving Modes of ESP32 and the applications.....	63
2.5 System Design.....	73
2.5.1 Algorithm Flowchart.....	73
2.5.2 System Hardware Assembly.....	76
2.6 Experimental Validation.....	78
2.7 Conclusion.....	85
2.8 Co-authorship Statement.....	86
<b>CHAPTER 3. A NOVEL DESIGN OF A LOW-COST SCADA SYSTEM FOR MONITORING STANDALONE PHOTOVOLTAIC SYSTEMS.....</b>	<b>88</b>
3.1 Introduction.....	88
3.2 Materials and Methods.....	91
3.2.1 Arduino® UNO R4 Wi-Fi.....	92
3.2.2 INA3221 Voltage Monitor.....	93

3.2.3 BlueTooth Terminal eDebugger.....	94
3.2.4 PVOutput.org .....	95
3.2.5 Experimental Setup.....	97
3.3 Experimental Results .....	99
3.4 Discussion .....	102
3.5 Conclusion.....	104
3.6 Co-authorship Statement.....	105
<b>CHAPTER 4. AN OPEN-SOURCE SCADA ARCHITECTURE FOR PHOTOVOLTAIC SYSTEM MONITORING USING ESP32, BANANA PI M4 AND NODE-RED.....</b>	<b>106</b>
4.1 Introduction .....	106
4.2 Related Works .....	109
4.3 System Descriptions .....	115
4.4 SCADA System Components .....	116
4.4.1 Banana Pi M4 Berry .....	116
4.4.2 ESP32-E.....	118
4.4.3 Node-RED.....	119
4.4.4 Voltage Sensor.....	121
4.4.5 Current Sensor .....	121
4.5 Implementation Methodology .....	123
4.6 Experimental Setup and Results.....	126

4.7 Discussion .....	133
4.8 Conclusion.....	135
4.9 Limitations and Future Work .....	136
4.10 Co-authorship Statement.....	137
<b>CHAPTER 5. AN IOT-SCADA ARCHITECTURE FOR PHOTOVOLTAIC SYSTEM MONITORING, CONTROL, AND INSPECTION IN REAL TIME .....</b>	<b>138</b>
5.1 Introduction .....	138
5.2 Related Work.....	140
5.3 System Descriptions .....	147
5.4 SCADA System Components .....	152
5.4.1 ESP32-S3 and ESP32-E.....	152
5.4.2 Arduino Cloud.....	153
5.4.3 ThingSpeak.....	155
5.4.4 Voltage Sensor and Current Sensors .....	156
5.4.5 OV2640 Camera.....	159
5.5 Implementation Methodology .....	160
5.6 Experimental Setup.....	167
5.7 Results.....	173
5.8 Discussion .....	177
5.9 Conclusion.....	180

5.10 Co-authorship Statement.....	181
<b>CHAPTER 6. CONCLUSIONS.....</b>	<b>182</b>
6.1 Conclusion.....	182
6.2 Research Contributions.....	186
6.3 Future Work .....	187
6.4 List of Publications.....	187
<b>REFERENCES.....</b>	<b>188</b>

## List of Tables

Table 1-1 Data logging parameters.....	23
Table 1-2 Commercial products of data logger in PV systems. ....	23
Table 1-3 Summary of the used components. ....	29
Table 1-4 Characteristic of Wireless Connectivity Technologies .....	32
Table 1-5 Power calculations.....	33
Table 1-6 Security Requirements in IoT Levels.....	33
Table 1-7 Charging Controller Specifications.....	35
Table 1-8 Battery Specifications.....	35
Table 1-9 Components specifications. ....	38
Table 2-1 Common commercial PV system data loggers. ....	46
Table 2-2 Comparison of FireBeetle 2 ESP32-E and Arduino UNO. ....	53
Table 2-3 Comparison of three types of ACS712 current sensors with different ranges.....	58
Table 2-4 ESP32 power modes with their active components and corresponding current.....	64
Table 2-5 Wake-up sources and their active parts. ....	65
Table 2-6 Current and charge consumption comparison between SD card and the flash memory. .....	68
Table 2-7 Average time and current for building Wi-Fi connection. ....	70
Table 2-8 The ratio of monitoring times to average current at different Wi-Fi connection intervals. ....	72
Table 2-9 Budget for the developed low-power and low-cost data logger. ....	84
Table 3-1 Comparison of HMI designs and data storage solutions in the literature review. ....	89
Table 3-2 Comparison of technical specifications between Arduino® UNO R4 Wi-Fi and.....	92

Table 3-3 Characteristics of CSV loader and Live loader in for updating data in PVOutput.org.	96
Table 3-4 BLE service information between BTeD and the RTU.....	99
Table 3-5 Change of the battery voltage during two days with different PV energy generation. .....	102
Table 3-6 Power consumption and price breakdown of the proposed system. ....	103
Table 4-1 Sensors and their connections with the ESP32-E. ....	123
Table 4-2 Itemized Price and power consumption of the system components. ....	134
Table 5-1 Types of monitored parameters and IoT platforms used in related papers. ....	146

## List of Figures

Figure 1-1 World Annual Oil Production 1900-2021 and Peak Oil 2005-2020 Scenarios.....	17
Figure 1-2 PV cell, module, panel, and array.....	19
Figure 1-3 Diagram of the prototype including improved modules (analog and digital sensors ..	25
Figure 1-4 Schematic design of the data logger system.....	26
Figure 1-6 Energy monitoring system architecture.....	27
Figure 1-7 Block Diagram of the Proposed IoT-based SCADA System.....	28
Figure 1-8 System architecture of a remote RFID tag embedded with temperature sensor. ....	30
Figure 1-9 Schematic illustration of a PV-RFID sensor. ....	31
Figure 1-10 Illustration potential application as building environment sensor. ....	31
Figure 1-11 Data logger installation connection diagram. ....	35
Figure 1-14 SOC of optimum power system size at solar cell area of.....	36
Figure 1-15 Profit of power consumption for the same program running in RAM comparing ....	37
Figure 1-16 Schematic diagram of experimental setup.....	38
Figure 1-18 Designed data logger circuit.....	39
Figure 1-19 Adjustable regulator flow chart. ....	40
Figure 1-20 Reduced saving events from 6 (left) to 1 (right) in 6 cycles. ....	41
Figure 1-21 Sleep current drop.....	41
Figure 2-1 Connection diagram of PV system and developed data logger.....	51
Figure 2-2 FireBeetle 2 ESP32-E pinout. Cut off the small line pointed by the red arrow to ....	53
Figure 2-3 DC 0-25V voltage sensor. (a) Real image. (b) Working diagram. ....	56
Figure 2-4 ACS712-20A Current Sensor. (a) Real image. (b) Connection schematic. ....	57
Figure 2-5 SD card transflash breakout connection diagram with FireBeetle 2 ESP32-E. ....	59

Figure 2-6 FireBeetle 2 ESP32-E current and power consumption at different levels of voltages, .....	61
Figure 2-7 FireBeetle 2 ESP32-E current consumption at different levels of CPU frequencies,..	62
Figure 2-8 Current consumption during Wi-Fi connection.....	70
Figure 2-9 The flowchart of developed power saving algorithm. ....	75
Figure 2-10 Experiment test setup. ....	78
Figure 2-11 Voltage logging scatter plot over 24 hours by DI-145.....	80
Figure 2-12 Current logging scatter plot over 24 hours by DI-145.....	80
Figure 2-13 Voltage scatter plot over 24 hours using the developed low-power data logger.....	81
Figure 2-14 Current scatter plot over 24 hours using the developed low-power data logger. ....	81
Figure 2-15 Current consumption including all scenarios of the data logger tasks.....	83
Figure 2-16 Web page showing electrical parameters of monitored PV system.....	84
Figure 3-1 INA3221 schematic diagram. ....	94
Figure 3-2 Comparison of technical specifications between Bluetooth Classic and BLE [54]. ...	95
Figure 3-3 (a) The designed SCADA system block diagram. (b) Experimental setup.....	98
Figure 3-4 The SCADA system program flowchart. ....	98
Figure 3-5 BTeD live recording of the PV panel output power. (a) Increased power.....	100
Figure 3-6 PV panel output power on Dec. 10, 2023, recorded in PVOutput.org.....	101
Figure 4-1 Overview of proposed SCADA Design. ....	116
Figure 4-2 BPI-M4 Berry. ....	117
Figure 4-3 Pinout of the FireBeetle 2 ESP32-E.....	119
Figure 4-4 The Node-RED MQTT node configuration interface.....	120
Figure 4-5 0-25 V voltage sensor illustration.....	121
Figure 4-6 Icon of ACS 712 current sensor.....	122



Figure 4-7 System flowchart. ....	125
Figure 4-8 Experimental setup at MUN ECE laboratory. ....	127
Figure 4-9 Hardware implementation of the proposed SCADA system. ....	127
Figure 4-10 MUN ECE laboratory PV installation. ....	128
Figure 4-11 The developed Node-RED flow. ....	129
Figure 4-12 Node-RED dashboards showing the monitored values of the PV system. ....	130
Figure 4-13 PV output voltage during the total solar eclipse on April 8, 2024. ....	131
Figure 4-14 PV output current during the total solar eclipse on April 8, 2024. ....	131
Figure 4-15 Test results for supervisory control. ....	132
Figure 4-16 Test results for local control. ....	133
Figure 5-1 Proposed design of 5-layer IoT-based SCADA architecture. ....	150
Figure 5-2 SCADA components of the proposed design. ....	151
Figure 5-5 Arduino Cloud Things setup interface with ESP32-E. ....	155
Figure 5-6 ThingSpeak channel private view. ....	156
Figure 5-7 Voltage sensor. ....	157
Figure 5-8 Image of ACS712 current sensor. ....	158
Figure 5-9 OV2640 camera module. ....	160
Figure 5-10 Flowchart of programs on ESP32-E. ....	162
Figure 5-11 Flowchart of programs on ESP32-S3. ....	163
Figure 5-12 Overview of the experimental setup. ....	168
Figure 5-13 Hardware schematic diagram. ....	170
Figure 5-14 Hardware setup of IoT-SCADA system. ....	170
Figure 5-15 MATLAB program for email alert when the battery voltage is below 12.5 V. ....	172
Figure 5-16 Camera webserver interface. ....	173

Figure 5-17 Arduino Cloud dashboards on July 11 and 12, with 1D time scale. ....	174
Figure 5-18 Arduino Cloud dashboards on July 12, with 1H time scale. ....	175
Figure 5-19 ThingSpeak dashboards displaying PV system parameters. ....	177
Figure 5-20 Load on when battery voltage is larger than 13.0 V. ....	177

## List of Abbreviations

PV	Photovoltaic
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
BLE	Bluetooth Low Energy
GPIO	General-purpose Input/Output
IDE	Integrated Development Environment
MPPT	Maximum Power Point Tracker
OS	Operating System
RAM	Random Access Memory
UART	Universal Asynchronous Receiver Transmitter Pins
USB	Universal Serial Bus
CMOS	Complementary Metal-Oxide Semiconductor
SD card	Secure Digital card

# Chapter 1. Introduction

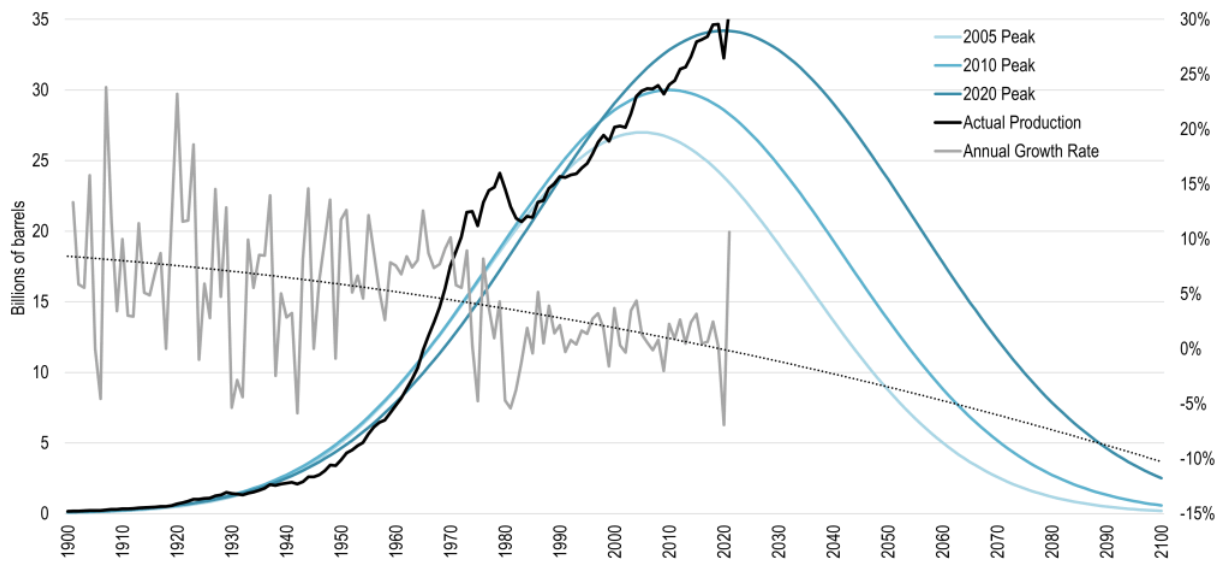
## 1.1 Introduction

### 1.1.1 Non-renewable Energy and Renewable Energy

Non-renewable energy is produced by sources that cannot be supplemented in short time. Fossil fuels like coal, petroleum, and natural gas are the three main forms of non-renewable energy sources. They are formed gradually during hundreds of millions of years, from the remains of plants and animals that are buried underground with high pressure and temperature as for coal, and from micro-organisms under the sea floor at a substantial depth as for petroleum and gas. Since the high pressure and temperature cannot be achieved in laboratory, fossil fuels are considered non-renewable in terms of the long period of formation process. The other non-renewable energy source is nuclear energy. The reason it is considered as non-renewable is the material used in nuclear power plants, which is the element uranium in form of U-235. Uranium is a non-renewable resource, since it is rare and cannot be replenished in a short period of time.

By its nature, a non-renewable energy resource production follows a bell curved shape as Figure 1-1 [1] shows. Fossil fuels production will peak and then decrease over time, leaving the environment damaged and having to find another spot before the end of the curve. Another disadvantage of fossil fuels is the introduction of many harmful particles to the air. Insufficient burning of fossil fuels like motor vehicle exhaust results in harmful Carbon Monoxide (CO) emission to human body, while sufficient burning of them like in thermal power plants still produce massive Carbon Dioxide (CO<sub>2</sub>) and thus amplify the greenhouse effect [2] and contributes to uprising of global temperature. Usage of nuclear energy can be dangerous and

bring irreparable harm to humans since the fission produces radioactive waste which is extremely toxic to the body.



*Figure 1-1 World Annual Oil Production 1900-2021 and Peak Oil 2005-2020 Scenarios.*

Renewable energy is the energy that can be replenished over time. Common renewable energy sources include solar, wind, geothermal, hydropower, biomass. Solar energy is the energy from radiant light and heat of the Sun; thus, it is renewable when the Sun exists. Solar energy can be utilized by various ways, such as using solar power to generate electricity, using solar power to heat, and solar architecture. Wind energy refers to the kinetic energy of air in motion. Wind is formed by the air that moves across Earth surface due to pressure difference. Air above the land gets heated up by the Sun quicker than that above the sea during daytime, resulting in density difference of the air. Warmer air rises and expands, while colder air falls and exchanges positions with warmer air. During night, this process remains similar with the only change that air above land cools down quicker, creating the wind as well. If the air and the Sun exist, wind energy can be supplemented and renewable. Geothermal energy is from the heat produced by the

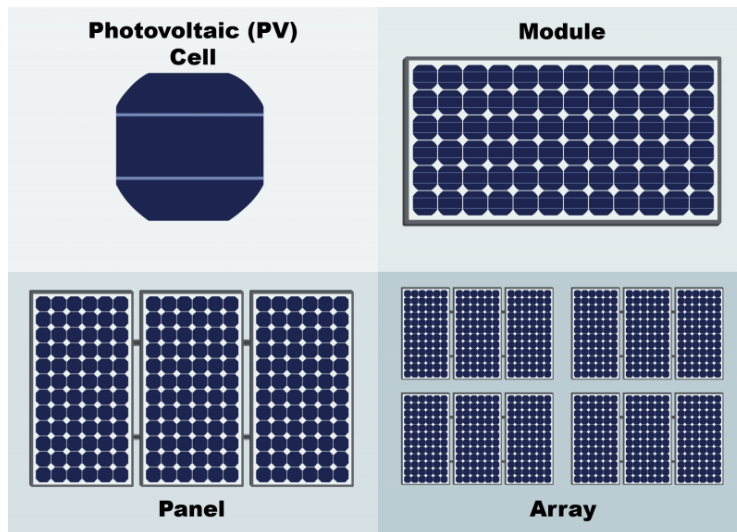
core of Earth. Hydropower utilizes the kinetic energy of falling water to rotate the turbine blades to generate electricity. Biomass is a kind of plant material which can be transformed into biofuel. Though burning biofuel releases CO<sub>2</sub> just like fossil fuels, it would not produce huge amounts of CO<sub>2</sub> since the plants utilized have a lifetime ranging from only one growing season to two decades. The CO<sub>2</sub> the plants absorb is significantly less than fossil fuels. Geothermal, hydropower, and biomass are renewable energy since they can be replenished if Earth core, the atmosphere, and the Sun exist.

### **1.1.2 Photovoltaic System**

To convert solar energy into electricity, one of the most common methods is by photovoltaic (PV) systems. This technology has been around for several decades and has seen rapid growth in recent years due to advancements in chemistry and materials domain, as well as an increased policy focus on reducing greenhouse gas emissions and dependence on non-renewable energy sources. Coming in a variety of sizes and configurations, PV systems range from small-scale residential systems to large-scale utility-scale projects. Residential systems are typically installed on the roofs of homes and can range in size from a few panels to several kilowatts. Utility-scale systems are typically installed on the ground and can range in size from several megawatts to hundreds of megawatts.

PV systems work by converting sunlight into direct current (DC) electricity. A PV system consists of solar panels, inverters, charge controllers, batteries and other components. These components work together to convert sunlight into usable electricity. A PV cell is the basic building block of a PV system, which is a semiconductor device that converts sunlight into

electricity. These cells are made of semiconducting materials such as silicon, and absorb photons from the sun and free electrons, creating an electrical current. They are connected to form PV modules. The modules are mounted on a frame and covered to protect the cells. The cells in a PV module are connected in series, so that the voltage of the module is equal to the sum of the voltages of the individual cells. Furthermore, a PV panel is a collection of PV modules that are gathered to form a larger unit. PV modules and panels are both designed to produce a specific amount of power, and are typically rated in watts, and kilowatts/megawatts, respectively. These panels are normally mounted on the roof of a building or on the ground. A PV array is a large-scale installation of interconnected PV panels. The panels are wired together to increase the total power output of the system, ranging from several kilowatts for a residential installation to hundreds of megawatts. The cost of PV cells, modules, panels, and arrays has dropped dramatically in recent years, making PV systems more affordable for both residential and commercial applications. Illustration of PV cell, module, panel, and array is as Figure 1-2 shows [3].



*Figure 1-2 PV cell, module, panel, and array.*

One of the main advantages of PV systems is their low environmental impact. Unlike traditional energy sources such as coal and natural gas, PV systems do not produce greenhouse gases or other harmful pollutants during operation. In addition, the materials used to produce photovoltaic cells are non-toxic and recyclable, making them an environmentally friendly energy source. Also, PV systems are becoming increasingly cost-competitive compared to traditional energy sources. The cost of photovoltaic cells and other components has dropped dramatically in recent years, making PV systems more accessible and affordable for both residential and commercial applications. Additionally, many countries and states offer incentives and subsidies to encourage the adoption of renewable energy sources, making PV systems even more cost-effective. In addition to their environmental and economic benefits, PV systems also offer increased energy independence and security. By generating their own electricity, homes and businesses can reduce their dependence on the grid and become less vulnerable to power outages and other disruptions. As more homes and businesses adopt PV systems, the demand for non-renewable energy sources will decrease, reducing the dependence on importing energy sources and increasing energy security.

Despite the many benefits of PV systems, challenges still exist before their widespread adoption. One of the main challenges is the intermittent sunlight, which can result in varying levels of electricity generation throughout the day and year. This can make it difficult to integrate PV systems into the existing grid, particularly in areas with limited grid capacity. Another challenge is the upfront cost of PV systems, which can be a barrier to adoption for some homeowners and businesses. While the cost of PV systems has decreased dramatically in recent years, the initial investment can still be substantial, particularly for larger systems.



The trend towards renewable energy sources is clear, and photovoltaic systems will likely play a critical role in the transition to a clean energy future, even with the challenges above. With continued advancements in technology, the adoption of PV systems is likely to continue to grow, providing a clean and reliable source of electricity for generations to come.

### **1.1.3 Data Logging**

Data logging refers to the process of measuring, recording, and storing data over time. This data can be used for analysis and decision-making purposes. Data logging provides several benefits for PV systems in terms of performance optimization, maintenance, monitoring and control, and energy management. Data logging gives detailed information about the performance of the photovoltaic system, which can be used to optimize the system performance. For example, by monitoring the battery voltage and state of charge, it is possible to determine if the battery is being over- or under-utilized. This can help to adjust the system settings to optimize the battery's performance. Another benefit is that it can also be used to monitor the performance of individual components within a PV system. Identification of any issues with the components and schedule of maintenance can be much faster. Also, it enables monitoring and controlling a PV system remotely, especially useful for large-scale systems where it may not be possible to physically inspect the system on a regular basis. By monitoring the load power consumption by end-user, optimization of the energy usage of the system can be achieved. For example, shift energy-intensive activities to times when the PV system is generating the most electricity.

In PV systems, data logging plays a crucial role in monitoring and optimizing the performance of the system. Data logging is used to monitor various parameters of a PV system, including:

- Solar panel output: The amount of electricity generated by the solar panels is an important parameter that is monitored and logged. This information can be used to identify any issues with the panels or to determine if they are operating at peak efficiency.
- Battery voltage and state of charge: Batteries are an important component of a PV system, as they store the electricity generated by the solar panels for use during periods of low or no sunlight. Data logging is used to monitor the voltage of the battery and its state of charge to ensure that it is functioning properly and to optimize its performance.
- Inverter performance: The inverter is responsible for converting the direct DC electricity generated by the solar panels into Alternating Current (AC) electricity that can be used by the end-user. Data logging is used to monitor the performance of the inverter to ensure that it is working properly and to optimize its performance.
- System load: The amount of electricity being used by the end-user is also an important parameter that is monitored and logged. This information can be used to determine the efficiency of the system and to identify any issues with the electrical load.

Specific parameters logged are shown below in Table 1-1 [4].

Table 1-1 Data logging parameters.

General parameters	Specific parameters	Symbol
Meteorology	Total irradiance, in the plane of the array	$G_i$
	Ambient temperature in a radiation shield	$T_{amb}$
	Wind speed (optional)	-
	Rainfall (optional)	-
	Humidity (optional)	-
PVarray	Output voltage	$V_A$
	Output current	$I_A$
	Output power	$P_A$
	PV module temperature	$T_{mod}$
Energy storage	Operating voltage	$V_S$
	Current to storage <sup>a</sup>	$I_{TS}$
	Current from storage <sup>a</sup>	$I_{FS}$
	Power to storage <sup>a</sup>	$P_{TS}$
Load	Power from storage <sup>a</sup>	$P_{FS}$
	Load voltage	$V_L$
	Load current	$I_L$
Utility grid	Load power	$P_L$
	Utility voltage	$V_U$
	Current to utility grid <sup>a</sup>	$I_{FU}$
	Current from utility grid <sup>a</sup>	$I_{FU}$
Back-up sources	Power to utility grid <sup>a</sup>	$P_{TU}$
	Power from utility grid <sup>a</sup>	$P_{FU}$
	Output voltage	$V_{BU}$
	Output current	$I_{BU}$
	Output power	$P_{BU}$

<sup>a</sup>A single current or power sensor can be used for the measurement of current or power for directions or both input and output.

Several commercial data loggers are presented in Table 1-2 with their specifications. Only one product from the same company is picked out with recent release year and lowest cost, if the company owns multiple products.

Table 1-2 Commercial products of data logger in PV systems.

No.	Product	Company	Power consumption	Cost range (€) (sensors not included)	Number of inputs	Internal data storage	Data sending/saving interval	Market launch year
1	Delta Solivia Gateway M1 G2	Delta Energy Systems GmbH	1 W @ 5 V	174	8	\	15 min by def.	2019
2	Fronius Datamanager 2.0	Fronius	< 2 W	222-259	6-10	\	\	2013
3	Q_reader	Gantner Instruments GmbH	5 W	600	2	8 GB	\	2010
4	Smartlogger 3000A	Huawei	8 W	1495	4 DI, 4 AI	\	\	2020
5	Powador-proLOG M	Kaco	\	549-664	4DI, 4 AI	\	300-3600 s	2009
6	ADL-MXSpro	Meier-NT GmbH	0.1 W-0.76 W	980-1100	max. 8	1 GB	24 h	2017
7	SMA Data Manager M Lite	SMA Solar Technology AG	4 W	285	\	\	\	2022
8	Solar-Log 2000	Solare Datensysteme GmbH	3 W	1031-1248	8	\	connected inverters < 30: 5min connected inverters 30-59: 10min connected inverters > 60: 15min	\
9	MaxWeb XPN	SolarMax	24 W (maximum)	500-600	1 DI, 4 AI	\	15 min by def.	2016
<b>Methods and rate of communication</b>								
No.	<b>RS 485</b>	<b>RS 422</b>	<b>Ethernet</b>	<b>WLAN</b>	<b>USB</b>	<b>2G/3G/4G</b>	<b>Measurement on main connection point</b>	<b>Open interfaces for SCADA integration</b>
1	19200 baud rate, by def.	\	Yes	\	\	\	Yes	No
2	Yes	Yes	Yes	Yes	\	\	Yes	Yes
3	Up to 115.2 kbs	\	Yes	\	1s to 24h	\	Yes	Yes
4	1200-115200 bps	\	Yes	Yes	Yes	Yes	Yes	Yes
5	Yes	\	Yes	\	\	\	\	\
6	300-115200 baud	\	10/100 Mbit	\	Yes	GSM module integrated	Yes	Yes
7	Yes	\	Yes	Yes	Yes	\	Optional	Yes
8	2400-115200 bps	Yes	10/100 Mbit	Yes	Yes	\	Yes	Yes
9	Yes	\	Yes	Yes	Yes	\	Yes	No

## 1.2 Literature Review

A detailed design of low-cost data logger for remote monitoring of standalone solar systems is given in [4]. In this study, ATmega328 microcontroller serves as processing the data from analog and digital sensors. The prototype diagram is as Figure 1-3 shows [4]. Considering the disadvantages of Arduino UNO board, such as the low resolution (10-bit) of the internal analog-to-digital converter (ADC), low flash memory of 32K bytes, and lack of display, an ad hoc printed circuit board (PCB) is developed to lift restrictions mentioned above. Two 18-bit ADCs MCP3424, a 4GB SD card, and a 16×2 liquid crystal display (LCD) are the components of the PCB other than the peripherals in Arduino UNO board. Extensive meteorological sensors and electrical sensors are adopted and tested to be in alignment with the IEC61724 standard. The measured data is displayed instantaneously on LCD and stored on an SD card once every day. Several techniques to reduce power consumption are also stated, including decreasing measuring frequency during inactive periods of microcontroller, sleep modes that disables onboard ADC and brown-out detection, and non-lasting display of LCD. Statistical comparison between newly developed data loggers and commercial data loggers reveals its accuracy.

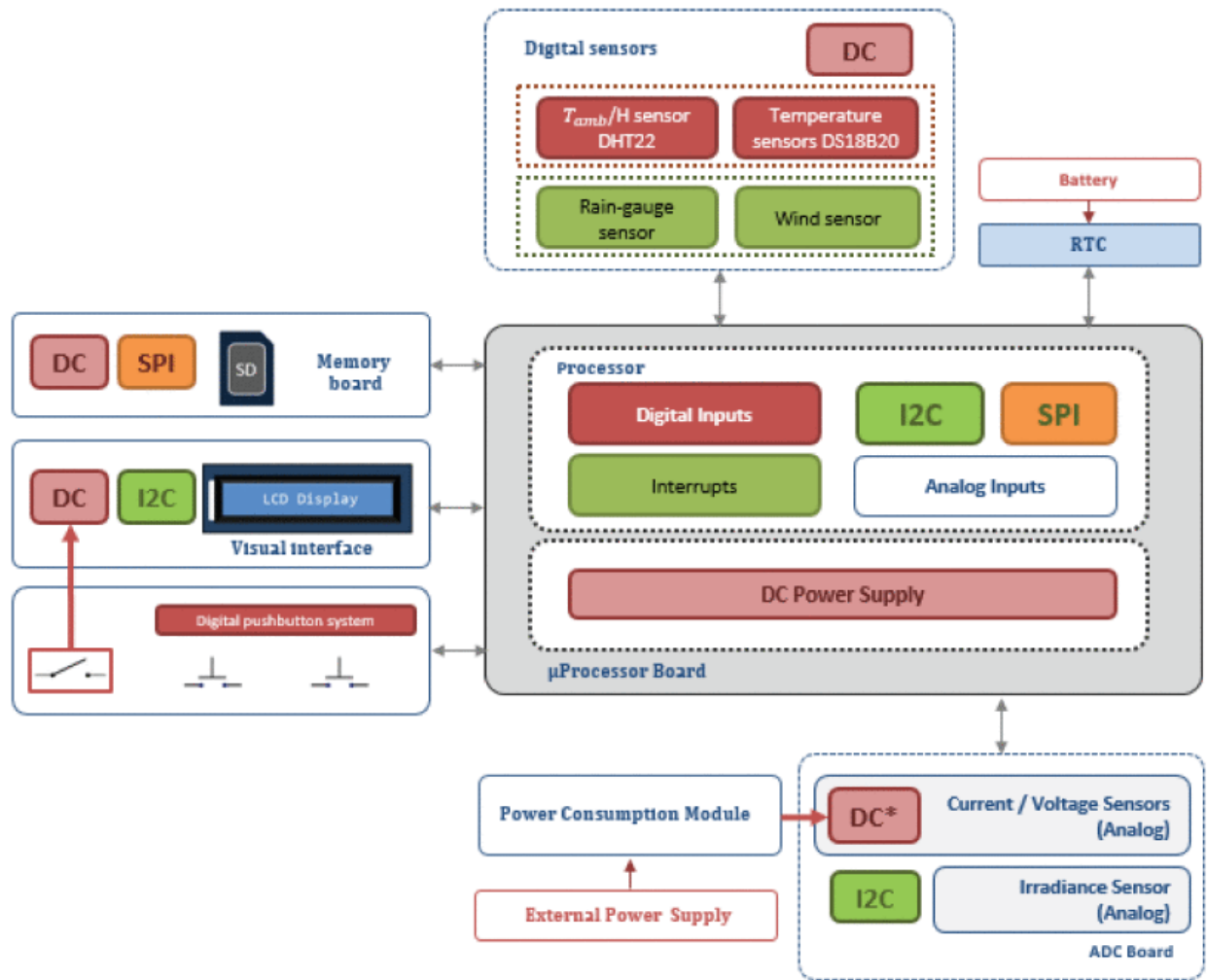


Figure 1-3 Diagram of the prototype including improved modules (analog and digital sensors modules, ADC board) and new modules (visual interface, digital pushbutton system and power consumption module).

Authors of [5] designed a data logger for home-scale hybrid renewable energy system of which the system power sources are solar photovoltaic power and wind turbine power. The system schematic design is shown in Figure 1-4 [5]. For DC monitoring, voltage sensor and current sensor are used. AC monitoring by PZEM-004 module includes voltage, current, power and energy at the output of the inverter. Real time clock (RTC) and anemometer are also used to collect the data of time and wind velocity. Arduino Mega 2560 is the microcontroller for

processing the data from sensors and modules, and then store the data in local SD card. The recording duration of the data logger depends on the size of the chosen SD card. This design does not require internet connection, which makes it suitable for remote sites. However, it lacks the ability to automatically present the data graph without retrieval of the SD card by human intervention.

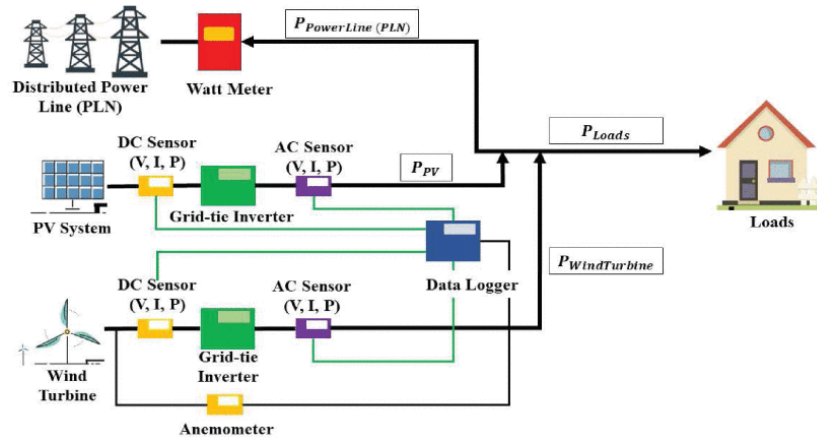
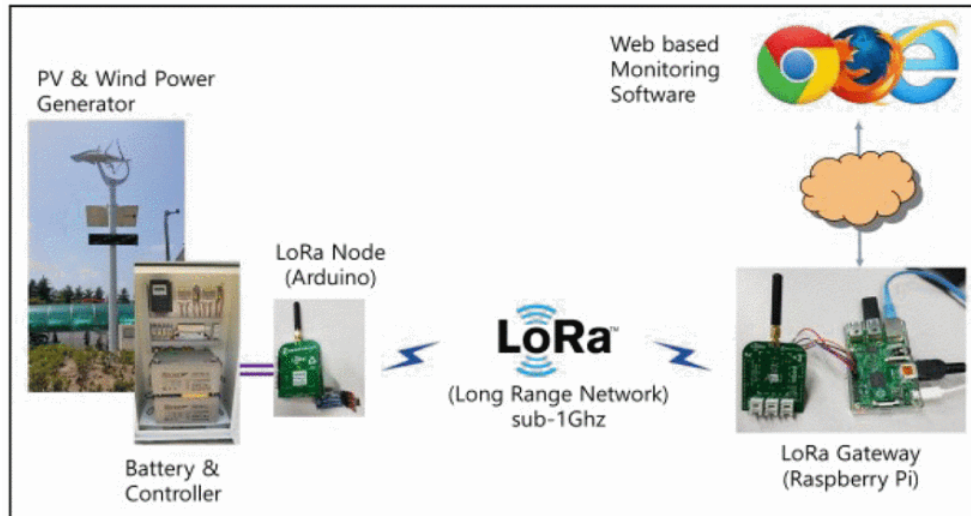


Figure 1-4 Schematic design of the data logger system.

Work presented in [6] described an energy IoT monitoring system on solar and wind power generation facilities. The system architecture is shown in Figure 1-5 [6]. Data of voltage, current, temperature, and battery are collected by a Long Range (LoRa) node based on Arduino. The data is then sent to LoRa gateway based on Raspberry Pi by applying an end-to-end modem, without using base station. Through serial interface, Raspberry Pi receives the data from LoRa modem and then stores it at the MongoDB developed for big data with NoSQL method. A web server is implemented to display the trend graph and statistical results with min/max/avg values. Low cost and low power consumption monitoring system is achieved by adopting Arduino board and Raspberry Pi 2 Model B V1.1. However, the power consumption of hardware is not indicated in the paper.



*Figure 1-5 Energy monitoring system architecture.*

An open-source, low-cost Supervisory Control and Data Acquisition (SCADA) system to monitor and control a solar PV panel at Memorial University is presented in [7]. The block diagram of the system is shown in Figure 1-6 [7]. ACS 712 Hall Effect current sensor and MH Electronic voltage sensor are used to measure the current and voltage at the output of PV system, and voltage across the battery. ATmega328P microcontroller processes the data from sensors and then parses the data to Raspberry Pi2 model B through USB cable. Node-RED is a programming tool for wiring together hardware devices, which is installed on the Raspberry Pi2 to post the data to EmonCMS local server. Remote monitoring and supervisory control can be achieved by creating dashboards and setting alarms on EmonCMS IoT platform. Nevertheless, the overall cost and power loss of the SCADA system are not specified in this paper.

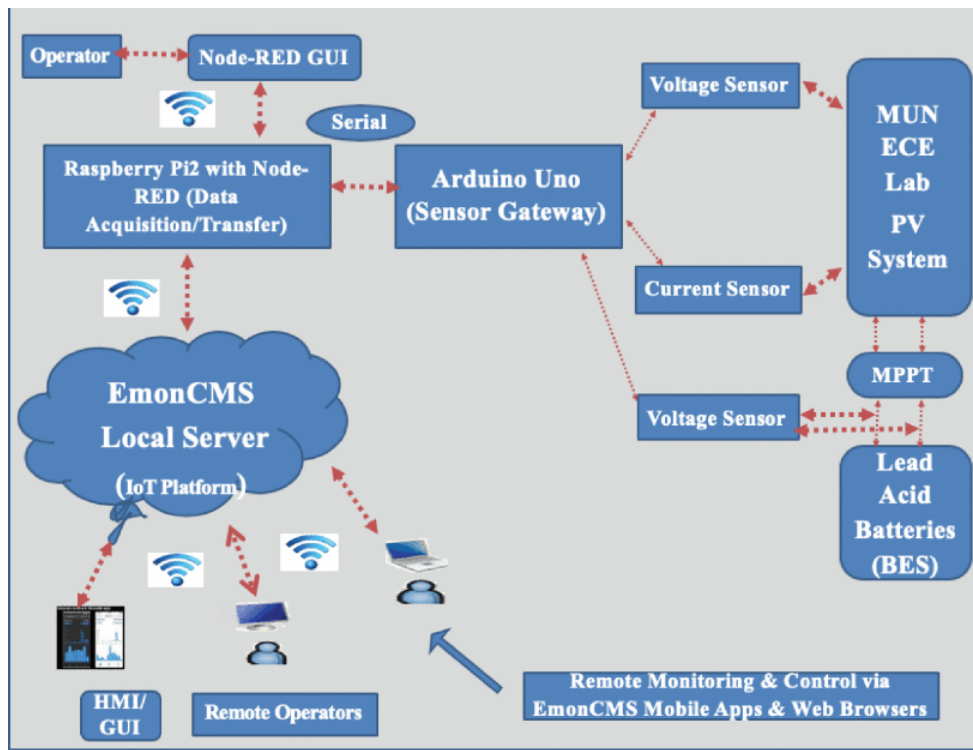








Figure 1-6 Block Diagram of the Proposed IoT-based SCADA System.

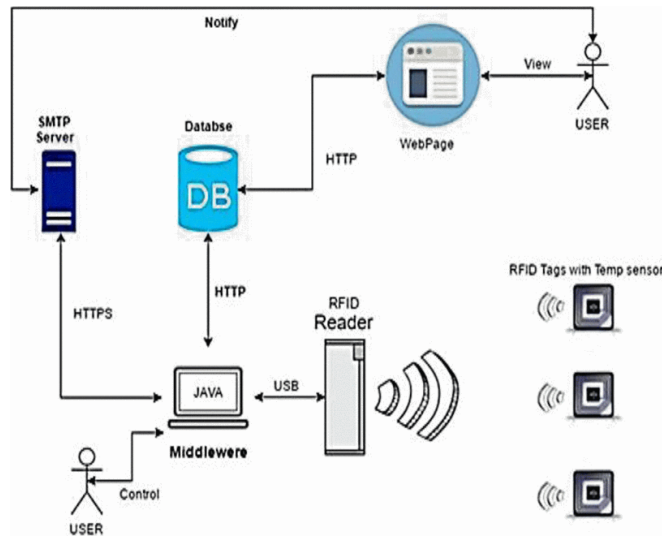
A low-cost monitoring system for a PV system using maximum power point tracking (MPPT) technique is designed by authors in [8]. In this PV system, a PV module, a DC-DC Boost converter, and 1 k $\Omega$  load are connected in series. A data acquisition board and a DC-DC converter board are used. The summary of used components is shown in Table 1-3 [8]. ACS 712 current sensor and LM 24 voltage sensor are used to sample the current and voltage at the output of the PV module and the load. Thus, duty ratio of DC-DC converter can be obtained. Arduino Mega 2560 recovers analog data from sensors and sends data to an LCD display to show monitored data in real time, and an ESP-01 WiFi module to transmit data as an HTTP request to host personal computer (PC). The WiFi module is powered by PV module with a voltage regulator. The experimental result shows MPPT is achieved successfully by observing steady output power. The cost for the designed monitoring system is 90 Euro.



Table 1-3 Summary of the used components.

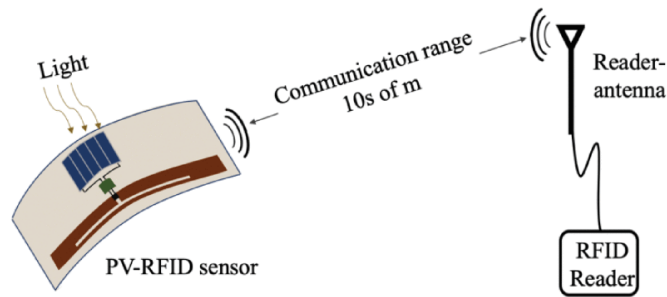
Components	Principals Features	Figures
<b>Sensor current AllegroACS712</b>	Bandwidth= 50 kHz Extremely stable output offset voltage Nearly zero magnetic hysteresis	
<b>Sensor voltage LM24</b>	Input voltage =3 to 32V True diffentielle input stage Four amplifiers per package Short circuited protected output	
<b>Micro-controller Arduino mega 2560</b>	Micro= ATmega1280 Input Voltage: 7-12V Operating Voltage:5V Digital I/O pins: 54 Analog pins : 16	
<b>Module WIFI ESP 8862-01</b>	WiFi Direct (P2P), soft-AP Integrated TCP/IP protocol stack Dimension total de 1,4*2,4cm Nombre the PINs=8	
<b>LCD DISPLAY</b>	Display format: 16 x 4 dots includes cursor 5*8 power supply: 5V Module Dimension 70.6 x 60.0	
<b>Voltage regulator Mini BEC</b>	Input voltage: 5 V to 23 V Output voltage: 3.3 V switching frequency: 330 kHz Small size: (38 × 15 × 10 mm)	

In reference [9], the authors proposed a low cost IoT based solution for monitoring hot spots of PV cells. The system architecture is shown in Figure 1-7. Battery less radio-frequency identification is adopted to harvest energy to power temperature sensors. Inside of a Radio Frequency Identification (RFID) tag, an antenna and an integrated circuit (IC) chip are embedded. When the sensors integrated in RFID tags get the temperature data, it sends to the RFID readers, which are connected to PC via Universal Serial Bus (USB). The middleware developed using Java then broadcasts the data on the web. In case of a hotspot, an alert will be sent to an email address using simple mail transfer protocol (SMTP). The benefit of RFID technology is the elimination of battery and object identification. However, the impact on the energy harvest induced by attached RFID tag on the PV panel surface has not been studied. Besides, the RFID readers should be fixed in vicinity of the tag which limits the application in real world.

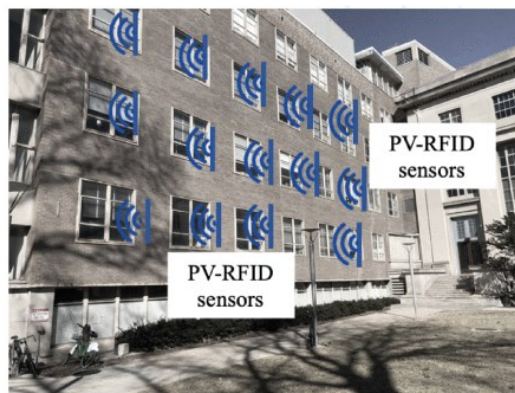


*Figure 1-7 System architecture of a remote RFID tag embedded with temperature sensor.*

A PV-powered passive RFID tag to enhance its read range is developed in [10]. RFID tags can be classified into passive, semi-passive, and active tags. The classification criterion is whether the tags are powered by external energy source instead of Radio Frequency (RF) energy. Passive RFID tags purely rely on RF energy transmitted by interrogator, while semi-passive or active tags partly or entirely are powered by external energy source. The minimum power to power up the IC is also called the IC's sensitivity. EM 4325 IC has -31 dBm sensitivity which allows for a longer range of transmission. Two applications are shown in this paper. The first one is PV-RFID augmented building environment sensing network. PV-RFID tags are attached on the windows or vents to monitor environment parameters such as temperature and air flow. The schematic illustration is shown in Figure 1-8, and the real application scenario is shown in Figure 1-9. Another application is PV-RFID augmented golf field, where golf ball is equipped with a PV-RFID tag to be tracked in real time. The IC consumes 10  $\mu$ A at 1.5V/3V to read/write Electrically Erasable Programmable Read-only Memory (EEPROM), which is 15 $\mu$ W/30 $\mu$ W power cost. However, the price of RFID reader can be as high as hundreds of dollars.



*Figure 1-8 Schematic illustration of a PV-RFID sensor.*



*Figure 1-9 Illustration potential application as building environment sensor.*

In reference [11] the authors presented an IoT application for real-time monitoring of solar home systems. An analysis of IEC61724 Standard is given in detail. A comparison of wireless communication technologies regarding range, bandwidth and cost is also shown as Table 1-4. 3G mobile communications are selected to add connectivity to designed datalogger based on Arduino Ethernet Rev 3 board with ATmega328 microcontroller. This is because 3G mobile communication offers affordable and long distance data transfer, with vast majority of population as users around the world. Meteorological and electrical parameters are measured and then sent and stored in a local server or a cloud server via 3G. TL-MR3020 router from TP-LINK with SIM card inserted in the modem provides the connection between the Arduino board

and the internet. A 12-month outdoor experiment was taken to test the robustness and reliability of the designed system.

*Table 1-4 Characteristic of Wireless Connectivity Technologies*

Type	Technology	Maximum range	Maximum bandwidth	Module cost (\$)
WPAN	ANT+	30 m	1 Mbps	1 – 15
	Bluetooth 4.0 LE	50 m	24 Mbps	1 – 15
	RFID	10 m (passive)	100 kbps	< 1 – 15
		100 m (active)		5 – 25
	NFC	10 cm	424 kbps	< 1
	802.15.4g	200 m	200 kbps	1 – 15
Zigbee	10 – 100 m	250 kbps	1 – 15	
WLAN	Wi-Fi	300 m	250 Mbps (802.11n)	10+
			54 Mbps (802.11a/g)	
			11 Mbps (802.11b)	
			1 Gbps (802.11ac)	
WWAN	LoRa	2 – 10 km	200 kbps	1 – 15
	Weightless	2 – 10 km	200 kbps	1 – 15
	Dash 7	2 km	200 kbps	1 – 15
	WiMax	40 km	34 Mbps – 1 Gbps	1 – 15
	2G (GSM, GPRS, EDGE)	35 km	9.6 – 384 kbps	1 – 15
	3G (UTMS, HSPA)	Up to 100 km	384 kbps – 10 Mbps	35 – 50
	cellular 4G/LTE	Up to 100 km	3 Mbps – 100 Mbps	80 – 120

A low-cost energy monitoring system of conditions of solar panels on acoustic buoy is described in [12]. The monitoring methods are mentioned as non-invasive and invasive monitoring. For non-invasive monitoring, one infrared camera and one megapixel camera are used to capture time-stamped images. For invasive monitoring, electrical parameters from MPPT charge controllers including PV voltage, battery voltage, charge state and daily energy yield are collected. Raspberry Pi Model B+ V1.2 as the microprocessor receives the images and electrical parameters and then transmits them into smart-connected parent buoy via ultra-high frequency (UHF) wave. They are demodulated after the ashore transmission via subsea Ethernet cable. The monitoring system is powered by an RS Pro Li-ion portable power bank of which the capacity is 10,400 mAh at 5V. Power calculations are presented in Table 1-5.

*Table 1-5 Power calculations.*

<i>Status</i>	<i>Components</i>	<i>Average Current (A)</i>	<i>Power (W)</i>
Standby	Raspberry Pi B+	0.25	1.25
Standby	GPS, Active Antenna, Pi Camera Module	0.04	0.2
Active	Pi, GPS and Pi Camera taking image and Tx.	0.33	1.65
Active	FLIR Lepton®	N/A	0.15*

Authors of [13] proposed a secure data acquisition architecture on PV systems using IoT techniques. Different layers of IoT security are presented in Table 1-6. Arduino UNO and Raspberry Pi are used as the microcontroller board and the single chip computer. Data of PV system including current, voltage, temperature, solar irradiance, and other information are routed to Arduino UNO. Then, the data is transmitted to Raspberry Pi to be stored locally and sent to cloud. Virtual Private Network (VPN) is developed for safe and efficient data exchange through a secure tunnel created by encapsulation and encryption. A successful connection between PV monitoring system and IoT gateway requires known local area network (LAN) Internet Protocol (IP), 3G/4G IP, and Tunnel IP. The safety of the proposed data acquisition architecture is achieved by connecting all sensors to VPN concentrator.

*Table 1-6 Security Requirements in IoT Levels.*

<b>Application layer</b>	Key agreement and authentication Security education Password management User's privacy protection
<b>Support layer</b>	Secure cloud computing Secure multiparty computation Anti-virus
<b>Network layer</b>	Communication security Anti-ddos Identity authentication Encryption mechanism
<b>Perceptual layer</b>	Sensors protection Key agreement Lightweight encryption technology

Street light system is monitored by a data logger which is designed in [14]. The system connection diagram is shown in Figure 1-10. Present monitoring of streetlights is done manually, and the data logger can automatically collect the data, only requiring regular retrieval of the memory. The system consists of PV modules, streetlamp, charging controller, battery and data logger. The charging controller specifications are shown in Table 1-7. The battery specifications are presented in Table 1-8. The data logger includes ATmega 8 microcontroller, Dallas DS 1307 Real Time Clock (RTC) and 64 k serial EEPROM. The time interval of recording voltage and current can be set and 15 mins are chosen as default in this paper. The data logger relates to a personal computer via RS 232 protocol to send recorded data to PC. Besides, different user commands are achieved by keyboard input from PC as interrupts to the microcontroller, allowing users to upload the data to PC, set date in RTC, etc. Nevertheless, this design of data logger requires cable connection between data logger and PC.

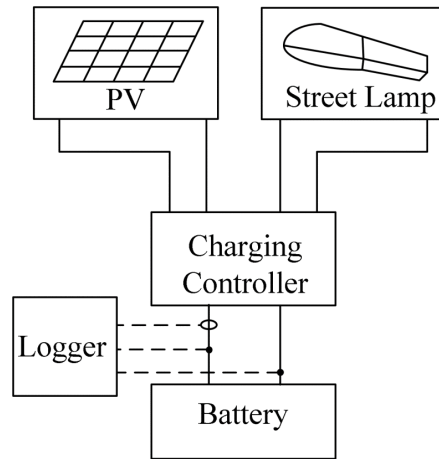


Figure 1-10 Data logger installation connection diagram.

Table 1-7 Charging Controller Specifications.

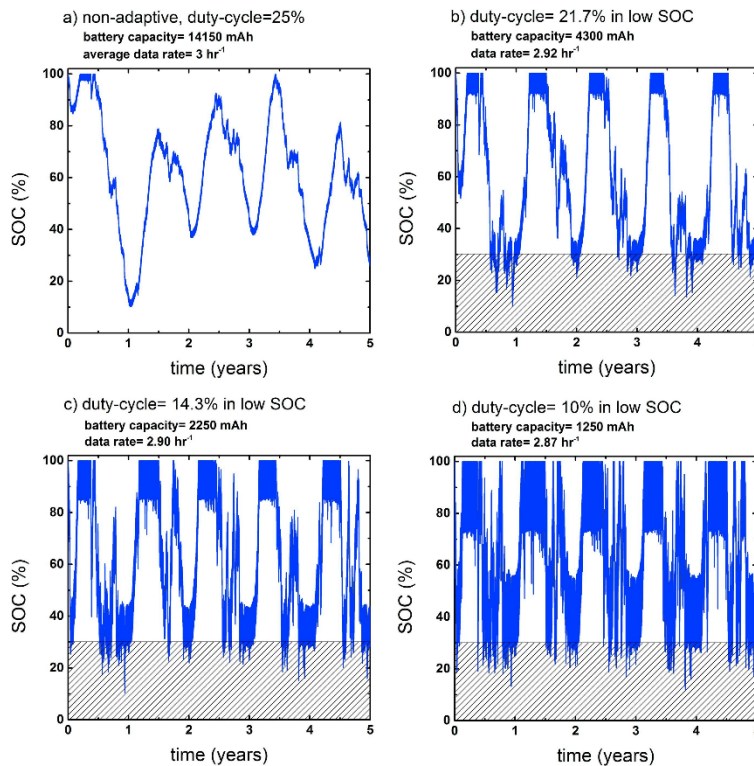
Charging method	PWM
Solar input capacity	$\pm 10$ A
Load capacity	$\pm 10$ A
Overload capacity	$12 \pm 0.5$ A
Regulator voltage	$14 \pm 0.5$ V
Self consumption	8 mA
Operating temperature	$-40^{\circ}\text{C} - 85^{\circ}\text{C}$

Table 1-8 Battery Specifications.

Type	Absorbent glass material
Acid	$\text{H}_2\text{SO}_4$
Capacity	80 Ah
Nominal Voltage	12 V

The work in [15] described an adaptive power consumption scheme to improve the reliability of PV powered devices. PV powered data logging device have the autonomy of not using bulky back battery to fulfill the need to data acquisition and transmission. To improve system reliability and deal with outlier conditions like battery state-of-charge becoming anomalously low, a methodology is stated in this work. The availability of energy is considered compromised if the battery State of Charge (SOC) is less than 30% and yesterday's average solar irradiance is

less than 15 mW/cm. In such case, the duty cycle of transmitting the data and the data acquisition rate is reduced to be adaptive to the energy scarcity. From Figure 1-11, in low SOC the lower the duty-cycle, the less battery capacity is required to maintain operation for three hours. The result shows that 90% of battery capacitor can be reduced and 10% of the system cost can be saved by using the adaptive power scheme, whilst at the minor price of 4.3% of reduction of data transmitted.

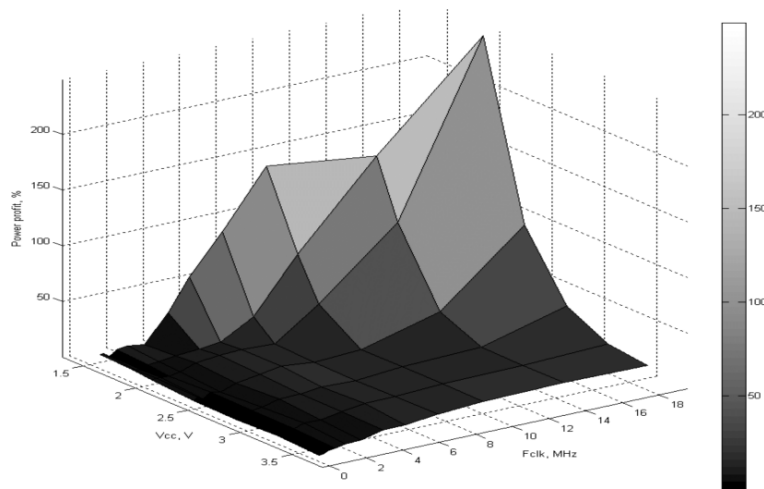


*Figure 1-11 SOC of optimum power system size at solar cell area of 26.6 cm<sup>2</sup> for various power adaptive schemes. The hibernation time increases by (a) 0%, (b) 20%, (c) 100%, (d) 200% when the battery SOC is less than 30%.*

Authors in [16] investigated the optimal method to determine frequency and supply voltage to maintain required performance with minimum energy cost. Texas Instruments (TI) MSP430 microcontroller is studied since it is designed for low power scenarios. MSP430 has three



internal clock sources, which are low frequency oscillator (XT1), high frequency (HF) oscillator (XT2), and digital controlled RC type oscillator (DCO). Under the active mode and idle mode of microcontroller, using DCO results in 62% higher power consumption than using HF as clock source. In each experiment, CPU frequency is set to the minimal possible value to undertake 100% CPU load, and the supply voltage is set according to MSP430 datasheet. The power consumption difference between running with RAM and running with Flash is shown in Figure 1-12. After experiments of executing three tasks, clock frequency of 4MHz and supply voltage of 1.8V achieves the minimal power consumption.



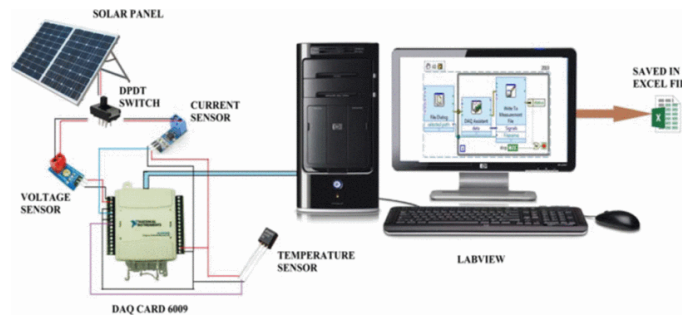
*Figure 1-12 Profit of power consumption for the same program running in RAM comparing with running in Flash memory.*

The degradation of PV system due to dust deposition by recording electrical and environmental parameters of the system is investigated in [17]. Though PV system enjoys many benefits of perpetuality of energy source and minor side effect, many other factors can impact its efficiency, such as dust, wind, and humidity. In this design, LM 35 as temperature sensor, ACS712 as current sensor, and DC voltage sensor are used to obtain temperature, open circuit voltage, and short circuit current, respectively. The components specifications are shown in Table 1-9.

Realistic I-V curve in comparison with nameplate curve of the studied PV system helps to determine the degree of degradation. DAQ 6009 as the interfacing device is connected with the PV system and the sensor data can be viewed in LabVIEW. The experimental setup is presented in Figure 1-13. The result shows that after 55 days of setting two identical PV panels in the same place, the PV panel without regular cleaning loses 9% of power output compared to the one with regular cleaning.

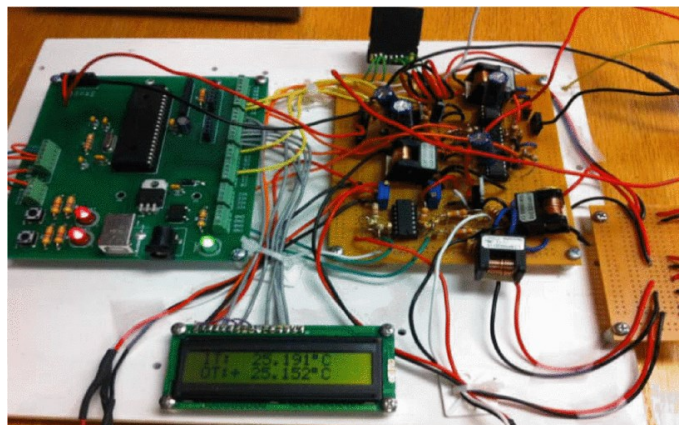
*Table 1-9 Components specifications.*

S. No.	Component	Type	Specifications
1	Temp. sensor	LM 35	Range -50°C to150 °C, Operates from 4 V to 30 V
2	Current Sensor	ACS712	Range -5A to+5A, Operates from 5V,Sensitivity 6mA/V
3	Voltage Sensor	DC Voltage Sensor	Range 0 to25V
4	Interfacing Device	DAQ 6009	Analog output generate outputs from 0-5 V
5	Software	LABVIEW	2013 Version
6	Variable Rheostat	-	0-290 OHM, 5A



*Figure 1-13 Schematic diagram of experimental setup.*

A data logger to record temperature and power consumption of two houses in St. John's, Newfoundland is designed in [18]. This data logger is able to record four AC current and two temperature values, helping to verify the electricity bill and improve the awareness of saving energy use. Two LM35 series temperature sensors are used to measure the indoor and outdoor temperature of the houses. Four Hall effect based current sensors are used to log incoming current to the houses. The designed circuit is shown in Figure 1-14. The sampling interval is 2 minutes. Operating at 5V, PIC18F4550 microcontroller programmed by Mikrobasic is responsible for ADC of collected sensor data and display them on 2×16 LCD. After analysis and comparison between power consumption from data logger and Newfoundland Power®, the mismatch is due to missing recording in some winter months by Newfoundland Power®. However, the power consumption of the data logger itself is not indicated.



*Figure 1-14 Designed data logger circuit.*

In reference [19] authors developed a dynamic voltage and frequency scaling technique to process electrocardiogram (ECG) signal. By this technique, the optimal frequency of microcontroller can be determined to minimize the power cost and thus achieve longest battery life. MCP41100 digital potentiometer interacts with MSP430 microcontroller via serial peripheral interface (SPI) bus to be able to change the supply voltage in real time. DCO is used

to provide adjustable frequency source, with a specified synchronization to avoid instability of DCO under different temperatures. When the execution time exceeds the threshold, MSP430 will adjust its operation mode. The system adjustable flow chart is as Figure 1-15 shows. A series of measurements is carried out and the 3-D plot is created to find the relations between voltage (X-axis), frequency (Y-axis), and performance (Z-axis). 8MHz is found out to be the optimal frequency to have maximum million instructions per second per Watt. In another experiment to determine the optimal frequency to have longest battery life, 2MHz is the desired value. This is because the saved energy by lower voltage and current outweighs prolonged time to finish the tasks carried out by CPU.

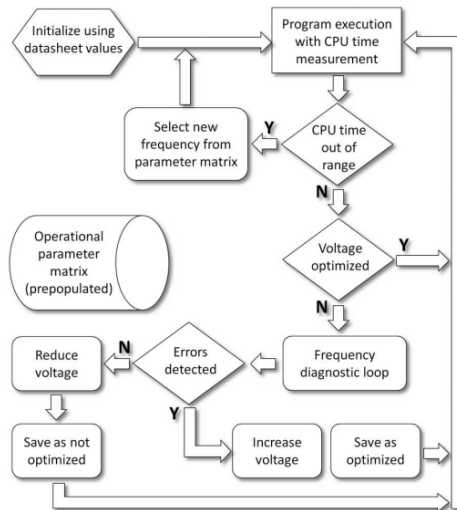


Figure 1-15 Adjustable regulator flow chart.

Authors of [20] discussed various ways to maximize the battery life of data logger by optimizing the saving events of SD card. The measured data is the falling edge pulses sent by PCF8523. The researcher did a series of experiments under the condition of supply voltage and frequency being 5 V/16 MHz and 3.3 V/8 MHz. Two special techniques are used to minimize the power consumption of data logging. First is to store variables in the embedded static random access

memory (SRAM) between sleep modes. The writing event to SD card will only happen when a defined string variable reaches a certain length. With the metal–oxide–semiconductor field-effect transistor (MOSFET), the sleep current of the microcontrollers drops from 800 and 750  $\mu\text{A}$  to 21.1 and 18.6  $\mu\text{A}$  respectively as shown in Figure 1-16. The other technique is to add a MOSFET to reduce the leakage current of SD card reader. A BS170 N-channel MOSFET is set between the SD card reader and the Arduino UNO board. Figure 1-17 shows the comparison of the CS patterns on the SD card when using the MOSFET to save every time (left) compared to saving once every 6 RTC cycles (right). As a result, Atmega328P operating at 5V/16MHz and 3.3V/8MHz can sample data with the measurement period as 2 seconds and 1.7 seconds and store the data in SD card, solely powered by a 2400 mA H battery for 1 year.

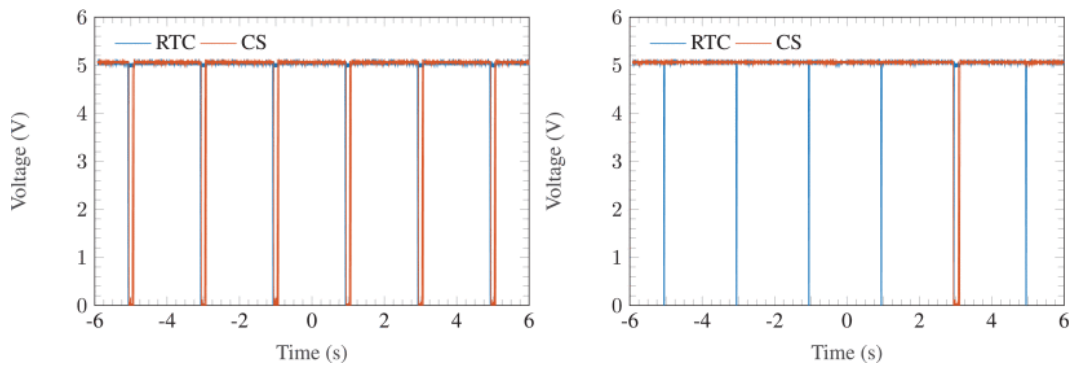


Figure 1-16 Reduced saving events from 6 (left) to 1 (right) in 6 cycles.

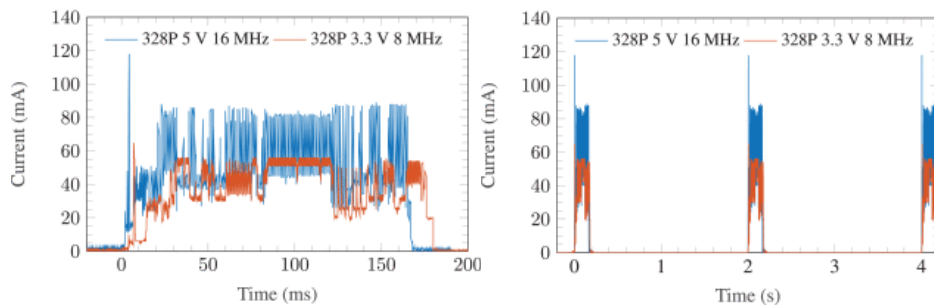


Figure 1-17 Sleep current drop.

## 1.3 Research Objectives

Commercial PV system data loggers typically have relatively high power consumption, imposing a substantial challenge to solar energy efficiency, particularly in winter when sunlight availability is reduced. A further limitation of these data loggers is they are mostly proprietary. The communication compatibility with components manufactured by other companies remains a question. Lastly, the data loggers on the market feature high cost, which can notably prolong the payback period of a PV system. The prices range from C\$ 250 to C\$ 2151 in the researched products.

The need for effective monitoring of PV systems, coupled with the limitations of commercial data loggers, highlights the demand for an IoT-based data logger that is both low-cost and low-power. The research objectives are listed below.

- Design a PV system data logger, using different low-power strategies to minimize the power consumption without compromising its performance.
- Design a PV system data logger, using an HMI that consumes low power and displays historical data, as well as a remote data storage solution that supports large data storage without extra charge.
- Design an open-source SCADA system based on IoT for a PV system, using a locally installed middleware to manage the data flow, provide customizable HMIs, and control the PV system load.
- Design an open-source IoT-SCADA system for monitoring and controlling a PV system, using two IoT platforms increased the system robustness by the data redundancy. Images of the load can be accessed on a web server, enabling the visual verification

of the load status.

## **1.4 Thesis Structure**

This thesis follows the manuscript format, with each chapter being a standalone document.

Chapter 1 introduces the background, literature review and the research objectives.

Chapter 2 presents the low-power consumption strategies in a low-cost IoT data logging for a PV system.

Chapter 3 discusses a novel design of HMI and data historian in a low-cost data logger for a standalone PV system.

Chapter 4 explains an open-source SCADA architecture for a PV system data logger, using ESP32, Banana Pi M4 and Node-RED.

Chapter 5 offers an IoT-SCADA architecture used to monitor, control, and inspect with Arduino Cloud and ThingSpeak platforms.

Chapter 6 highlights the thesis, providing the conclusion, research contributions, and the future work.

## **1.5 Co-authorship Statement**

Wei He is the first author of all chapters. Dr. Mohammad Tariq Iqbal is the corresponding author of all chapters. Dr. Mirza Jabbar Aziz Baig is also the corresponding author of Chapter 4. In Chapter 2, 3, and 5, I am responsible for Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, and Visualization. Dr. Mahammad Tariq Iqbal is responsible for Conceptualization, Resources, Writing - Review & Editing, Supervision, Project Administration, Funding Acquisition. In Chapter 4, Dr. Mizra co-writing of the

Introduction, Literature Review, and Discussion; funding acquisition; and overall review/editing of the manuscript. In Chapter 4, Dr. Mirza Jabbar Aziz Baig is responsible for co-writing of the Introduction, Literature Review, and Discussion; funding acquisition; and overall review/editing of the manuscript.



# **Chapter 2. Power Consumption Minimization of a Low-Cost IoT Data Logger for Photovoltaic System**

## **2.1 Introduction**

Due to Earth's limited fossil fuel reserves, renewable energy has emerged as a vital alternative power source [21]. Renewable energies, in contrast to fossil fuels which are associated with air pollution and carbon emissions, exhibit minimal environmental impact during their production phase. Among these, solar energy, captured by photovoltaic (PV) panels, stands out as a particularly promising option owing to its simplicity in installation and maintenance, especially when compared to wind energy. The proliferation of solar installations is evident in Canada and globally. In 2022 alone, solar energy accounted for 25.9%—over a quarter—of Canada's newly installed capacity, cumulating nearly 4 GW in total installed capacity. Specifically, in Ontario province, solar energy provides power to approximately 517,000 homes [22]. The share of solar energy within the renewable energy portfolio is poised for rapid growth, heralding a promising future for its widespread adoption.

Standalone PV systems typically consist of several key components: PV panels for energy harvesting, an inverter to convert the generated DC power into AC, a load as the energy consumer, and a backup battery to maintain load operation in the absence of sunlight. However, certain common failures in PV systems, such as hot spots resulting from manufacturing defects or partial shading, and leakage currents caused by parasitic capacitance between the PV panel and the ground, can markedly diminish their efficiency. Regular monitoring of PV systems, facilitated by widespread sensor networks and data logging, can lead to enhanced system

performance [23]. Should monitored voltage or current data reveal anomalies, technicians can promptly intervene to inspect and address issues in the PV system, thereby preventing further deterioration.

Commercial PV system data loggers may offer a viable solution, thanks to features such as user-friendly operation, integrated software, and scalability. Nevertheless, the typically high cost of these commercially available data loggers can notably prolong the payback period of a PV system. Table 2-1 presents a compilation of common commercial PV system data loggers, detailing their power consumption and prices, which vary between C\$ 250 and C\$ 2151. A further limitation of these data loggers is their relatively high power consumption, with a minimum of 0.1 watts even in power-saving mode [24]. Such power usage, although minimal, is still significant for a battery-powered device and imposes a substantial challenge to solar energy efficiency, particularly in winter when sunlight availability is reduced.

*Table 2-1 Common commercial PV system data loggers.*

<b>Product and Company</b>	<b>Power Consumption</b>	<b>Cost range (CAD) (Sensors not included)</b>
ADL-MXSpro by Meier-NT GmbH[24]	0.1 -0.76 W	1410-1583
Fronius Datamanager 2.0 by Fronius[25]	< 2 W	319-373
Q.reader by Gantner Instruments GmbH[26]	5 W	863
Smartlogger 3000A by Huawei[27]	8 W	2151
Powador-proLOG M by Kaco[28]	\	790-955
Delta Solivia Gateway M1 G2 by Delta Energy Systems GmbH[29]	1 W @ 5 V	250
SMA Data Manager M Lite by SMA Solar Technology AG[30]	4 W	410
Solar-Log 2000 by Solare Datensysteme GmbH[31]	3 W	1483-1796
MaxWeb XPN by SolarMax[32]	24 W (maximum)	719-863

## 2.2 Literature Review

The need for effective monitoring of PV systems, coupled with the limitations of commercial data loggers, highlights the demand for an IoT-based data logger that is both low-cost and low-power. Various researchers have explored alternative PV system data logger designs to address these challenges. In one study [4], a specialized printed circuit board (PCB) was developed to surpass the limitations of the Arduino UNO board, which suffers from a low-resolution (10-bit) internal analog-to-digital converter (ADC), restricted 32K bytes of flash memory, and the absence of a display. The custom PCB featured enhanced capabilities, including two 18-bit ADCs (MCP3424), a 4GB SD card, and a 16×2 liquid crystal display (LCD), along with the standard peripherals of the Arduino UNO board. However, this design's complexity may be challenging for those with limited electronics and circuitry expertise. Additionally, this data logger did not align with the International standard IEC61724 for PV monitoring and evaluation in terms of time recording.

Another study [5] presented a data logger for a small-scale hybrid renewable energy system, combining solar photovoltaic and wind turbine sources. This design incorporated sensors for DC voltage and current monitoring, and a PZEM-004 module for AC monitoring at the inverter's output, measuring voltage, current, power, and energy. It also included a Real-time Clock (RTC) and an anemometer for time and wind velocity data measurement, respectively. The Arduino Mega 2560 microcontroller processed and stored data on a local SD card. This design's independence from the internet made it suitable for remote locations, but it lacked the capability for automatic data graphing without manual input. A different approach [33] introduced 3G mobile communications for connectivity, using an Arduino Ethernet Rev 3 board with an

ATmega328 microcontroller. This system utilized a TL-MR3020 router from TP-LINK with an inserted SIM card to connect the Arduino board to the internet. The 3G communication was selected for its global availability and efficient long-distance data transfer. This data logger recorded meteorological and electrical parameters, storing or transmitting the data on a local server or a cloud server via 3G. However, it is important to note that the total cost of this system exceeded C\$ 574, which might be a consideration for certain projects.

The research community has recommended various methods to reduce the power consumption of data loggers. In [4], several techniques were outlined, such as decreasing the measuring frequency during inactive periods of the microcontroller, enabling sleep modes to disable onboard ADC and brown-out detection, and employing a non-permanent display for the LCD. However, the study did not specify the overall power consumption of the monitoring system. Another approach, detailed in [15], introduced an adaptive power consumption scheme aimed at enhancing the lifespan of PV-powered devices. This scheme was activated when the battery's State of Charge (SOC) fell below 30%, and the previous day's average solar irradiance was under  $15 \text{ mW/cm}^2$ . In such a scenario, the duty cycle for transmitting data and the data acquisition rate are reduced to conserve energy. Results indicated that using this adaptive scheme could decrease battery usage by 90% and reduce system costs by 10%, with only a minor 4.3% reduction in data transmission loss. A notable limitation, however, is the inability to notify users when the SOC drops below the threshold, potentially compromising sensor data integrity until the battery is completely drained.

In [20], researchers explored methods to maximize battery life in data loggers by optimizing SD card save events. Experiments were conducted under conditions of 5V/16MHz and 3.3V/8MHz

supply voltage and frequency, respectively. Two specific techniques were employed: storing variables in the embedded static random-access memory (SRAM) between sleep modes and using a metal–oxide–semiconductor field-effect transistor (MOSFET) to cut off the SD card's power supply, thereby reducing leakage current. A BS170 N-channel MOSFET was positioned between the SD card reader and the Arduino UNO board. This setup allowed the Atmega328P, operating at 5V/16MHz and 3.3V/8MHz, to sample data every 2 seconds and 1.7 seconds respectively. Then the data was stored on the SD card, which was able to be powered solely by a 2400 mAh battery for a year. However, this system required manual data retrieval due to the absence of IoT capabilities for remote monitoring.

In [34], an IoT-based PV monitoring system was proposed, utilizing Long Range (LoRa) technology to transmit PV current, solar radiation, and surface temperature values to data acquisition cards, with an Ethernet connection to upload data to the Internet. The system's power consumption was recorded at 6.11 Wh per day, averaging 254 mW. While this level of consumption was relatively low, there is still room for improvement, especially considering the system was powered by Li-ion batteries.

The novel PV system data logger developed in this study is distinguished by its low power consumption, cost-effectiveness, and IoT integration, offering an optimized solution for PV system monitoring and data logging. This design demands minimal maintenance, is economically efficient, and enables remote monitoring. Significantly, our data logger eliminates the need for PCB assembly, and all its components are readily available in the market at very low costs. The software employed is the open-source Arduino Integrated Development Environment (IDE), compatible with various operating systems, including Windows, macOS, and Linux. The

paper's focus on low power design not only extends the battery's lifespan, thereby reducing the frequency of human intervention for battery replacement but also decreases the risk of overheating. Additionally, the data logger utilizes a 32GB SD card for data storage, ensuring that sensor data is securely saved over an extended period in a non-volatile format.

The remainder of this chapter is structured as follows: Chapter 2.3 details the monitoring system architecture and introduces the data logger's components such as the ESP32, voltage sensor, current sensor, and microSD card. Chapter 2.4 delves into low-power techniques, examining strategies like reduced supply voltage and CPU frequency, the data buffer mechanism in local storage, the optimum Wi-Fi connection interval, and the deployment of deep-sleep modes. In Chapter 2.5, the configuration of the data logger and the monitored PV system is elucidated. This section also describes the software routines executed by the ESP32, programmed to alternate between active and deep-sleep states to minimize power consumption. Additionally, PV system parameters are outlined, along with a schematic of the connection between the PV system and the data logger. Chapter 2.6 presents experimental results, including data storage on the SD card and real-time monitoring capabilities via a web portal. The section concludes with a summary in Chapter 2.7. Co-authorship statement is presented in Chapter 2.8.

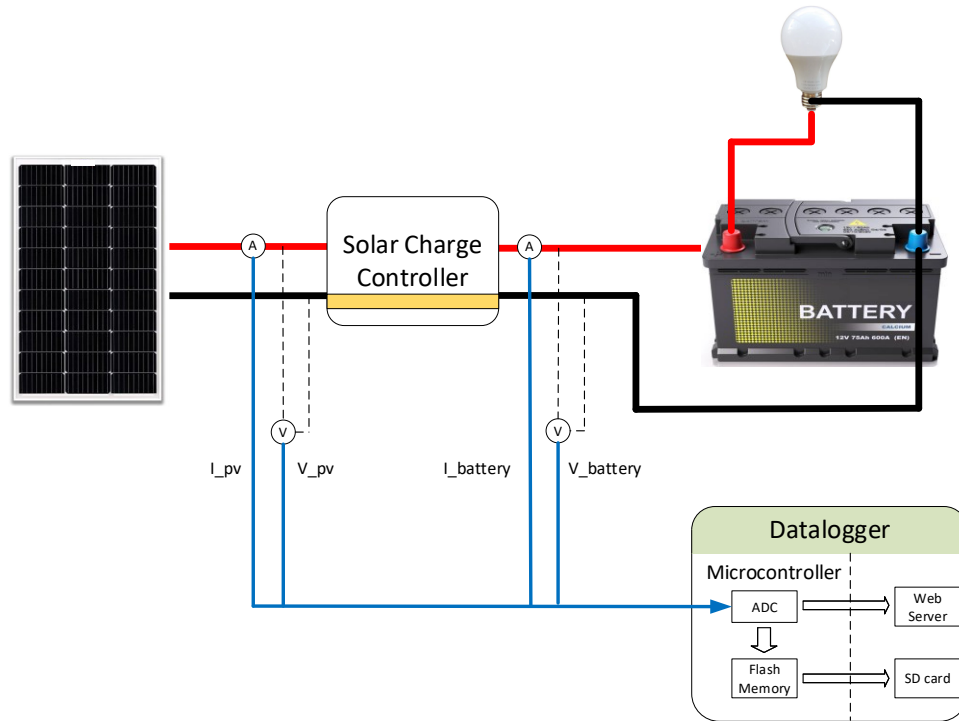
## **2.3 Data Logger**

This section introduces the components of the proposed data logger and the connection diagram.

### **2.3.1 Components Connection**

The data logger is comprised of the FireBeetle 2 ESP32-E and peripherals. FireBeetle 2 ESP32-E as the main microcontroller is used to store collected sensor readings locally and send the

readings to a local web server for the remote monitoring purpose. Peripherals connected to the microcontroller include voltage sensors, current sensors, and one SD card that is for local data storage. The system connection diagram is shown in Figure 2-1. The PV system includes the PV panel, a charging controller, and a 12 V 8.5 W LED lamp as the load. Voltages and currents of the PV panel and the battery are to be logged.



*Figure 2-1 Connection diagram of PV system and developed data logger.*

### 2.3.2 Microcontroller FireBeetle 2 ESP32-E

The microcontroller used in this section belongs to ESP32 family. ESP32 is a series of low-cost, low-power system-on-a-chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth, which makes it one of ideal candidates for IoT applications. ESP32 microcontrollers made by different vendors may have different power consumption mainly due to the choice of on-board voltage regulator. DFRobot FireBeetle ESP32 microcontroller is a low-power consumption

microcontroller designed for IoT projects [35]. The FireBeetle Board integrates a dual-core ESP-WROOM-32 ESP32 WiFi-BT-BLE MCU module for Wi-Fi and Bluetooth dual-mode communication. The board has a 10  $\mu$ A electric current in deep-sleep mode and the main controller supports USB and 3.7 V external lithium battery power supply methods. The USB and external DC can charge the LiPo battery directly. The board has a special hardware design for Arduino IDE to make a download without switching boot-mode manually.

Among FireBeetle series, FireBeetle 2 ESP32-E is an ESP-WROOM-32E-based main controller board with dual-core chips, that supports Wi-Fi and Bluetooth dual-mode communication. Additionally, it features compact dimensions, high energy efficiency, an integrated charging circuit, and an intuitively accessible interface. These attributes collectively make it an optimal choice for applications including smart home IoT, industrial IoT, and wearable devices. It is chosen in this section due to its extremely low power consumption during power saving modes. In deep-sleep mode, consumed electric current is only 10  $\mu$ A according to the datasheet [36]. Arduino UNO is a classic microcontroller board based on the ATmega328P, and it is used in many research papers [4,14,20]. Table 2-2 illustrates the comparison between FireBeetle 2 ESP32-E and Arduino UNO. When counting analog and digital pin numbers of ESP32-E and Arduino UNO, pins with dual usages belong to the analog category. Pins occupied for USB transmission, USB power supply, serial printing, and on-board LED are not counted. Figure 2-2 shows the pinout the FireBeetle 2 ESP32-E. Pointed by red arrow is the low-power solder jumper pad, which is designed for low power mode and default to be connected. Slightly cut off the thin wire



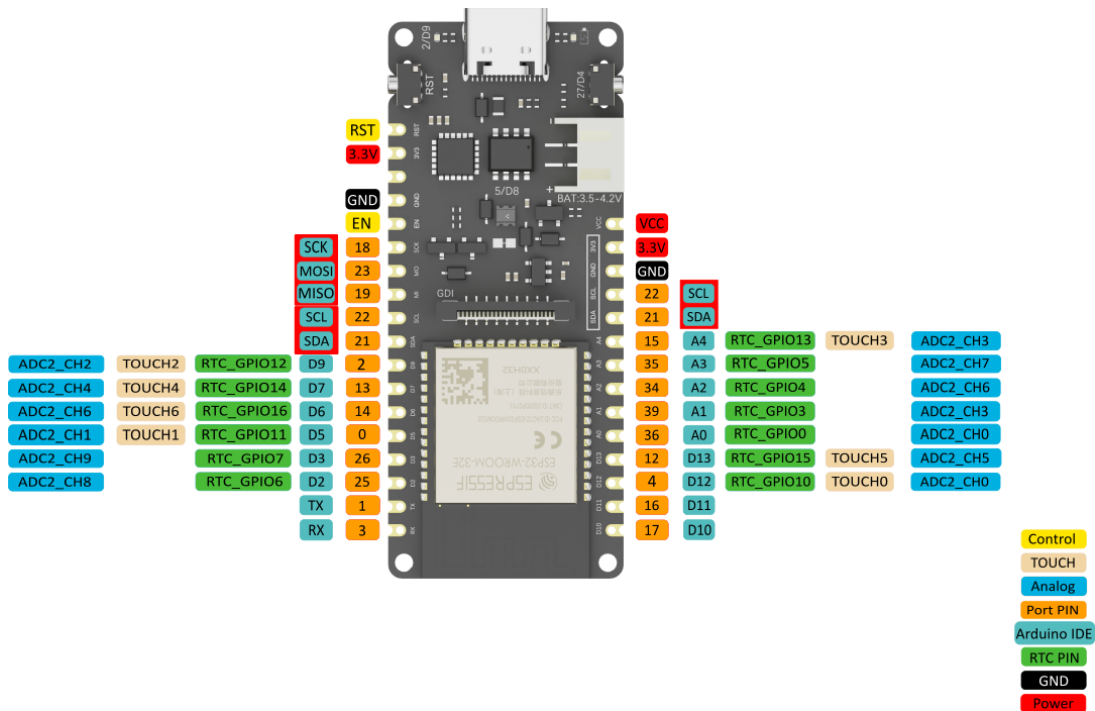


Figure 2-2 FireBeetle 2 ESP32-E pinout. Cut off the small line pointed by the red arrow to disable onboard RGB LED to reduce power consumption.

Table 2-2 Comparison of FireBeetle 2 ESP32-E and Arduino UNO.

Components	FireBeetle 2 ESP32-E	Arduino UNO
Processor	dual-core Tensilica LX6	ATmega328P
Operating Voltage	3.3 V	5 V
CPU Speed	Up to 240MHz	Up to 16MHz
Analog In/Out/Dual Pins	4/0/9	6/0/0
Digital In/Out/Dual Pins	0/0/10	0/0/8
Flash	4 MB	32 kB
SRAM	520 kB	2 kB
EEPROM	0 kB	1 kB

to disconnect the resistor between the battery and onboard RGB LED. When disconnected, RGB LED can only be turned on when the board is powered by USB. Static power consumption can be reduced by 500  $\mu$ A.

Reasons that FireBeetle 2 ESP32-E is selected as the microcontroller are justified below. From Table 2-2 we can easily conclude that FireBeetle 2 ESP32-E features lower supply voltage, higher CPU frequency, and larger Flash/SRAM size than Arduino UNO. Also, Arduino UNO does not have Wi-Fi connectivity to access a wireless local area network (WLAN) to reach the Internet if without an extension shield board, while FireBeetle 2 ESP32-E has such ability due to the inbuilt Wi-Fi module. Finally, FireBeetle 2 ESP32-E consumes ultra-low power during deep-sleep mode, nonetheless no power-saving modes are compatible with Arduino UNO. These advantages of FireBeetle 2 ESP-E can be leveraged to contribute to all functionality of the microcontroller of data logger.

Although ESP32 has 12-bit resolution ADC which in theory is more precise than 10-bit resolution of Arduino UNO, the stability is questionable that in practice ESP32 ADC shows non-linearity. Therefore, the ESP32 ADC output should be adjusted instead of direct use without modification. By connecting one DAC output of the microcontroller with one ADC channel, the corresponding relation can be calculated to form the look-up table for future calibration use [37].

### 2.3.3 Peripherals

In this section the peripherals connected with the FireBeetle 2 ESP32-E are introduced. Voltage sensors and current sensors sample the readings regularly, and the SD card is responsible for storing these sensor readings.

#### 2.3.3.1 Voltage Sensor

The HiLetgo voltage detection module DC 0-25 V is based on resistive voltage divider to measure the voltage. In Figure 2-3 the module image and its working diagram are shown. VCC and GND are ports of the voltage to be measured. Two resistors, 30 k $\Omega$  and 7.5 k $\Omega$ , are connected in series. Output “S” is between the two resistors and should be connected with an analog pin of FireBeetle 2 ESP32-E. Output “-” is the same ground as voltage ground, which is connected with the ground of the development board. Since  $7.5 \text{ k}\Omega / (7.5 \text{ k}\Omega + 30 \text{ k}\Omega) = 1/5$ , the measured voltage by FireBeetle 2 ESP32-E is one fifth of the real voltage, which should be modified in the program to get accurate results.

To reduce the power consumption, a larger resistor of 470 k $\Omega$  is placed in series with the positive side of the voltage to be measured. The power consumption of two voltage sensors is reduced to **0.67 mW**, assuming that the measured voltages are both 13 V.

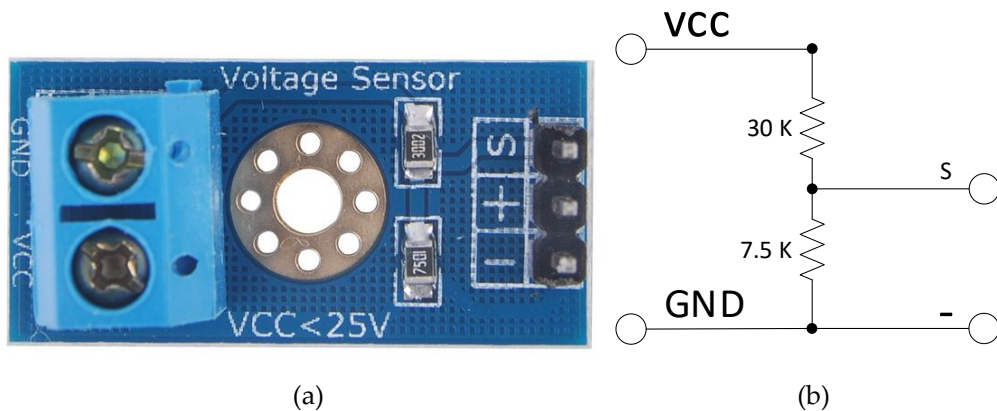


Figure 2-3 DC 0-25V voltage sensor. (a) Real image. (b) Working diagram.

### 2.3.3.2 Current Sensor

Based on Hall effect, ACS712 is a precise current sensor features low cost and easy implementation in both industrial and commercial applications. Figure 2-4(a) is the image of ACS712 current sensor with  $\pm 20$  A range. The connection schematic is illustrated in Figure 2-4(b). Pin 1 and pin 2 are the positive side of measured current, while pin 3 and pin 4 are the negative side. Vcc with the typical value of 5.0 V are connected with pin 8, and GND is attached to pin 5. Pin 7 stands for output (Vout) signal and should be hooked up with analog pin of the microcontroller. For ESP32 of which the nominal voltage is 3.3 V, the Vcc and the ground pin of ACS712 sensor should be connected with 5.0 V and ground of an external voltage source. Since the Vcc pin of current sensor is attached to one of ESP32 pins, the ground of that external voltage source should also be connected with the ground of ESP32. When connecting Vout of ACS712 to ESP32 analog pin, a voltage divider should be used to prevent the ESP32 from taking a voltage more than 3.3 V.

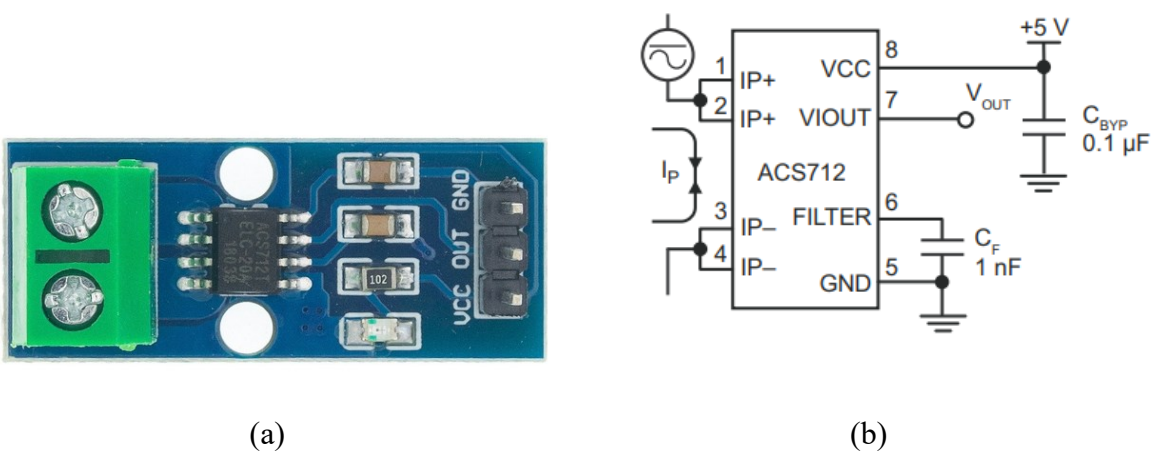


Figure 2-4 ACS712-20A Current Sensor. (a) Real image. (b) Connection schematic.

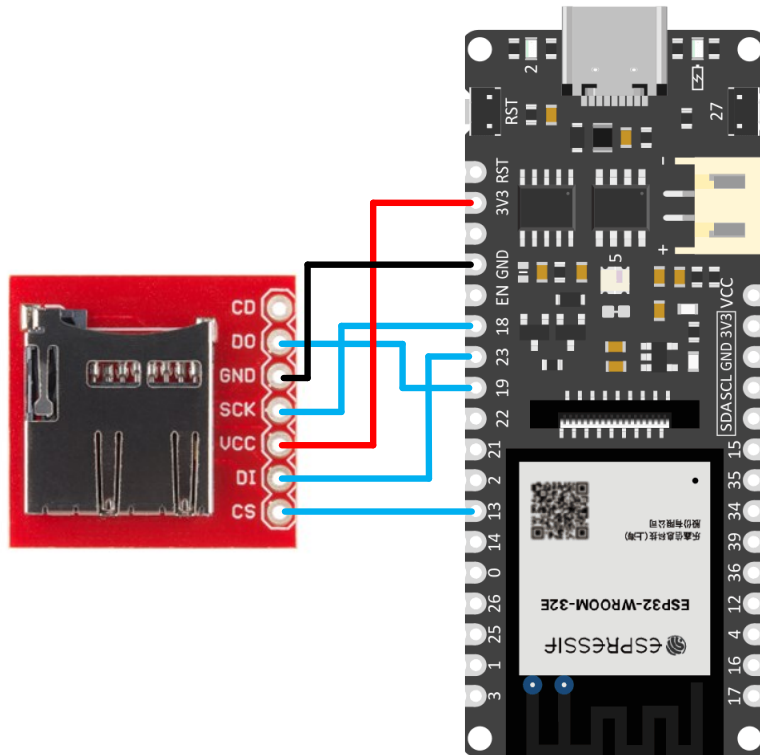
Classified by the optimized current, there are three types of ACS712 sensor with ranges of  $\pm 5$  A,  $\pm 20$  A, and  $\pm 30$  A. The supply voltage to ACS712 series is all 5 V, with a fluctuation of  $\pm 0.5$  V. Different types of ACS712 current sensor have different voltage-current sensitivity, ranging from 185 mV/A to 66 mV/A, with small fluctuations. Peak to peak noise should also be considered while measuring the current with ACS712 series. Table 2-3 lists the properties of the three kinds of current sensors in detail. To reduce the power consumption, an IRF510 MOSFET is placed between the current sensor and the common ground of external voltage and MCU supply voltage. GND pin of the current sensor is connected with Drain pin of MOSFET. Source pin of MOSFET is connected with GND. Gate pin of MOSFET is connected with MCU control pin. When the current is not being measured, the MOSFET is put in cut-off region so that the integrated circuits of the current sensor are not powered. The self-current consumption of one working current sensor is 64.35 mW since the working current is 12.87 mA by testing results. By cutting off the current sensors 25 seconds every 30 seconds, two current sensors consume **21.45 mW** on average.

*Table 2-3 Comparison of three types of ACS712 current sensors with different ranges.*

Part Number	Optimized Current Range (A)	Supply Voltage (V)			Sensitivity (mV/A)			Noise (mV), peak to peak
		Minimum	Typical	Maximum	Minimum	Typical	Maximum	
ACS712ELCTR-05B-T	±5				180	185	190	21
ACS712ELCTR-20A-T	±20	4.5	5	5.5	96	100	104	11
ACS712ELCTR-30A-T	±30				64	66	68	7

### 2.3.3.3 SD Card Transflash Breakout

Figure 2-5 illustrates the connection diagram between Sparkfun® microSD Transflash Breakout and FireBeetle 2 ESP-E. Compatible with SPI protocol, six pins of the breakout are attached with their counterparts on the microcontrollers. DO on the breakout board corresponds to Master Input Slave Output (MISO) pin (GPIO 19) of the microcontroller; DI is related to Master Out Slave Input (MOSI) pin (GPIO 23). VCC pin is with 3.3V pin of FireBeetle 2 ESP32-E; GND pin is with GND pin of the microcontroller board. SCK pin connects to SCK pin (GPIO 18) of the development board. Finally, it is worth noting that by the try-and-error method, the CS data selection pin should be bridged with D7 (GPIO 13) of the FireBeetle 2 board, even though the DFROBOT tutorial website claims D6 is the right choice.



*Figure 2-5 SD card transflash breakout connection diagram with FireBeetle 2 ESP32-E.*

## 2.4 Low Power Techniques

The power consumption of a data logger consists of the power consumption of the microcontroller and the peripherals. In most cases, most of the power consumption is from the microprocessor that is on the microcontroller board. Early microprocessors are made of pMOS or nMOS logic gates, while nowadays CMOS gates become the primary method of building microprocessors since they overcame the scaling challenge and reduced the size of devices [38]. For microcontrollers made of CMOS circuits, static and dynamic are the two types of power consumptions. Static power consumption is significantly less than its dynamic counterpart, therefore only dynamic power consumption is discussed in this section. According to [39] it is expressed as

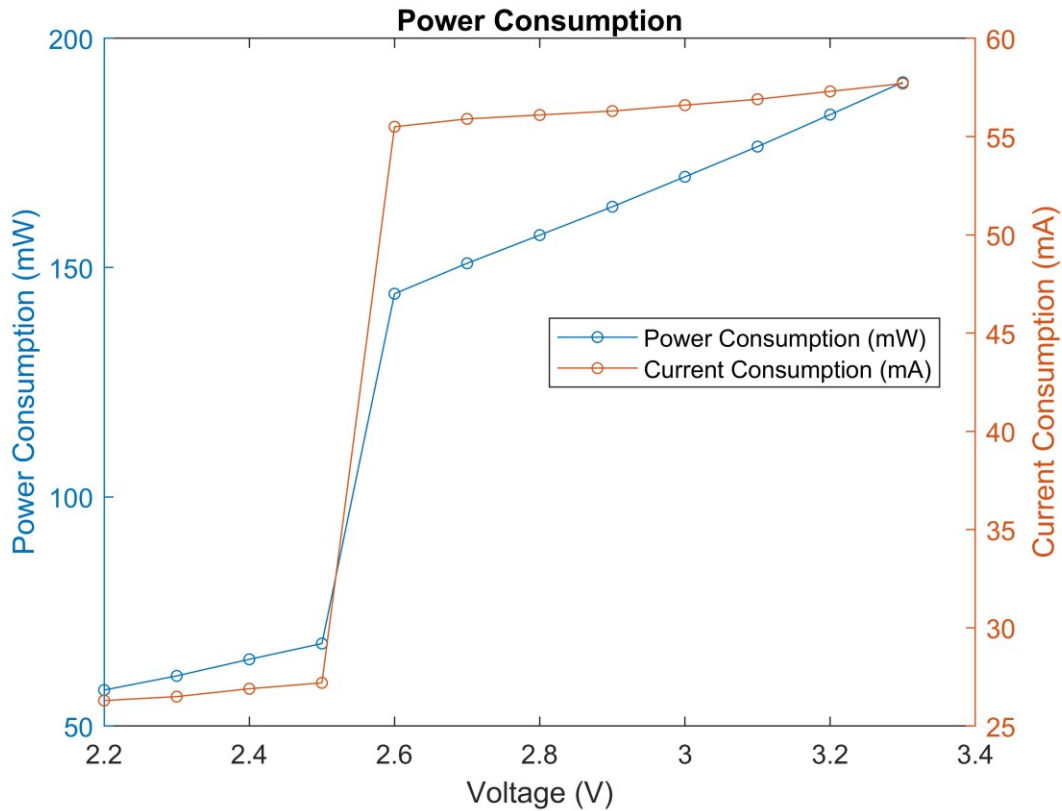
$$p = CV^2 f \quad (1)$$

where  $C$ ,  $V$ ,  $f$  is the total capacitance, supply voltage, and operating frequency. Since the capacitance is hard to change after fabrication, varied voltage and frequency are investigated in the following subsections.

### **2.4.1 Impact of Voltage**

From Equation 1, the power consumption is proportional to the supply voltage squared, which suggests the supply voltage has significant impact on the power dissipation. To gain a basic understanding of the effect of reduced voltage, DC power supply AB-3300 is used to provide regulated voltage ranging from 2.2 V to 3.3 V to Firebeetle 2 ESP32-E. It is achieved by providing V+ and Ground from AB-3300 to 3.3 V and ground pins in FireBeetle 2 ESP32-E. A LED blink program from Arduino IDE built-in library is uploaded to test its power consumption at different levels of voltage.



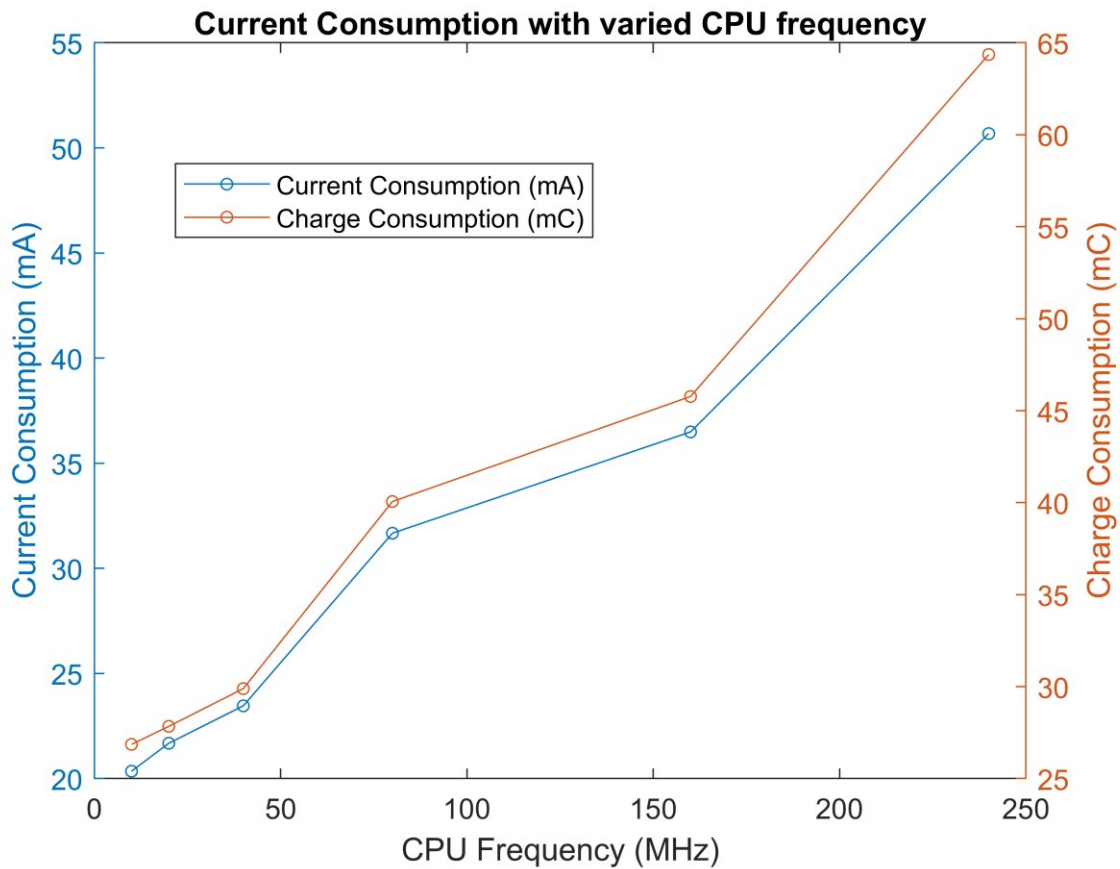


*Figure 2-6 FireBeetle 2 ESP32-E current and power consumption at different levels of voltages, with a LED blink program running on the board.*

Figure 2-6 demonstrates that power consumption decreases with lower voltage, while it does not follow the relationship stated in Equation 1 but in a linear way. The reason is that it is tested with a complete board including a Tensilica LX6 dual-core processor and onboard peripherals. Besides, when supply voltage drops below 2.6 V, a sudden decrease of current and power consumption is observed. The possible explanation is that some components such as RF module cease to work, leading to an unrealistic power saving since the system stability is compromised. For battery powered data logger, a suitable battery will extend the replacement period significantly.

## 2.4.2 Impact of Frequency

According to Equation 1, the CPU frequency is linearly proportional to the total power consumption. To test the impact of FireBeetle 2 ESP32-E operating frequency on power consumption, a program saving variables to the flash memory is uploaded and the current consumption is measured at different CPU frequencies which is shown in Figure 2-7. Adjusting CPU frequency can be achieved by calling `setCpuFrequencyMhz` function. The supply voltage is fixed at 5 volts.



*Figure 2-7 FireBeetle 2 ESP32-E current consumption at different levels of CPU frequencies, with a saving event program running on the board.*

From Figure 2-7, with the increasing of CPU frequency at fixed supply voltage, the current and power consumption grow linearly, while a change of slope rate is observed across the figure. It is mostly in alignment with Equation 1 that power consumption is proportional to the switching frequency. The time of finishing one saving event is the charge consumption divided by current consumption, which remains nearly the same for all CPU frequencies. According to Figure 2-7, it can be concluded that when the data logger is executing saving event task, CPU frequency should be set to the minimum values, which is 10 MHz.

For other tasks more complicated than saving in flash memory, CPU frequency setting should be cautious. Since the communication protocol between SD card transflash breakout and the microcontroller is SPI, higher CPU frequency will accelerate the data transmission process and reduce the overall power and energy consumed [40]. Thus, if executing other tasks such as communication with SD card or local web server, CPU frequency should be set to a value enabling the system to function first, instead of considering saving power consumption.

### **2.4.3 Power Saving Modes of ESP32 and the Applications**

This section introduces different kinds of power saving modes of ESP32. Furthermore, their applications are also explained.

#### **2.4.3.1 Power Saving Modes with Further Power Saving Options**

Different power saving modes are designed to save power consumption and extend the battery lifespan. Active mode is the default operation mode when all the ESP 32 components are running. This is the least energy efficient operation mode, while it is necessary since Wi-Fi is needed to connect to the Internet. Minor power saving options can still be applied to active mode to further

reduce the power consumption, such as reducing CPU frequency and changing RF operation mode. In modem sleep mode, Bluetooth, radio, and Wi-Fi are disabled. CPU clock frequency is configurable, resulting in different current/power consumption accordingly. The next operation mode is light sleep mode. The only difference it has with modem sleep is the CPU, most of RAM, and the digital peripherals are clock-gated, with supply voltage also reduced. Considering frequent connection with the Internet is not required, modem sleep mode and light sleep mode would not be used in this section. Deep sleep mode and hibernation mode are switched to when ESP32 is idle. In deep sleep mode, the only active parts are RTC memories and RTC controller, with the ULP coprocessor in some cases. Hibernation mode disables most of the ESP32 components except for the RTC timer and part of the RTC GPIOs. Table 2-4 illustrates the power modes with their active components and corresponding current consumptions.

*Table 2-4 ESP32 power modes with their active components and corresponding current consumptions.*

<b>Power Modes</b>	<b>Active Components</b>	<b>Current Consumption</b>
Active	ESP 32 core, ULP coprocessor, RTC, peripherals, Bluetooth, Radio, Wi-Fi	95-240 mA
Modem Sleep	ESP 32 core, ULP coprocessor, RTC, peripherals	2-4 mA @ 2 MHz 20~25 mA @ 80 MHz 30~50 mA @ 240 MHz
Light Sleep	ULP coprocessor, RTC, peripherals	~0.8 mA
Deep Sleep	ULP coprocessor, RTC RTC	150 $\mu$ A 10 $\mu$ A
Hibernation	RTC (timer, part of RTC GPIOs)	5 $\mu$ A

To exit the power saving modes, wake-up sources are mandatory. Options of wake-up sources are briefly introduced here. A built-in timer in the RTC controller can wake up the chip after a pre-defined amount of time. Touchpad in the RTC IO of RTC peripherals will trigger wakeup when a touch sensor interrupt occurs. Single or multiple RTC GPIOs of RTC peripherals can be

chosen as external wake-up sources, when one or all the RTC GPIOs reach the predefined logic level depending on the configuration. RTC GPIOs are indicated as green blocks in Figure 2-2. External wake-up source of single RTC GPIO is controlled by RTC IO, and external source of multiple RTC GPIOs are controlled by RTC controller. Non RTC GPIOs can be configured as wake-up source, while which is only supported by light-sleep mode. The last while not least wake-up source is the ULP coprocessor in RTC peripherals, which can be programmed to wake up the ESP32 when specific events are detected to occur. Careful wake-up source configuration can help to reduce power consumption even further. Different wake-up sources are controlled by different parts of RTC module. The inactive part of RTC module can be powered off to save energy. For example, RTC IO and RTC peripherals can be disabled when the timer is configured as the only wake-up source. Wake-up sources, short descriptions, and active parts are listed in Table 2-5.

*Table 2-5 Wake-up sources and their active parts.*

<b>Wake-up Sources</b>	<b>Wake-up Description</b>	<b>Parts to Power ON</b>
Timer	After predefined time	RTC controller
Touch Pad	After touching a pin	RTC IO
External 0	After toggling a pin	RTC IO
External 1	After toggling multiple pins	RTC controller
ULP coprocessor	Per program uploaded	ULP coprocessor

The second power-saving option is to either power down the flash entirely, or just pull up flash CS pin in light sleep. The former one is not recommended by ESP-IDF programming guide since

the wake-up time for flash is unpredictable and flash may not be working properly. Instead, pulling up the CS pin of flash makes the flash be in non-selected state, to avoid a large current leakage. The final option is to isolate IOs in deep-sleep mode. Some pins might be pulled up externally and pulled down internally at the same time, creating a current path from high voltage to the ground through pull-up and pull-down resistors, thus adding to total current consumption.

#### **2.4.3.2 Local Data Storage**

In several research papers [41-44], the SD card is chosen as the data storage solution. Though the SD card offers large storage capacity up to tens of gigabytes, the current consumption when the SD card is active can be as high as 100 mA, which impedes the reduction of the data logger power consumption. To optimize the overall power consumption, an effective way is to reduce the frequency at which the data is transferred to the SD card by utilizing the internal flash memory of ESP32 due to the low current consumption. Instead of storing the collected variables into the SD card after each batch, flash memory serves as the data buffer. In [20] the authors took advantage of the internal SRAM of the Atmega328P to store variables to reduce the power consumption. Only after exceeding a certain length will the data saved in SRAM be transferred to the SD card. Authors of [20] set the threshold length as six sampling cycles without stating the reason. However, SRAM is volatile and thus cannot guarantee the integrity of saved data. Flash memory is used in this section, considering the relatively low current consumption and the non-volatile nature that ensures data integrity even without the supply power.

Table 2-6 evaluates the current consumption and lasting period when saving events happens in SD card and the ESP32 flash memory. Measured data points are from Power Profiler Kit II [45] which is developed by Nordic Semiconductor®. Different operations including initialization,

read, and write will result in different levels of current consumption and the overall electrical discharge. For writing operation in SD card and the flash, the total electric discharge is 8.38 mC and 0.421 mC, where the former is nearly as 20 times as the latter. Frequent data storage in SD card should thus be avoided, of which the alternative solution is to buffer the sensor readings in the flash memory. One of the easiest methods to achieve data storage in the flash memory is to facilitate the Preference library. Functions are `preferences.putULong()` and `preferences.getULong()` to write and read the variable to/from the flash.

The strategy for determining the appropriate time interval for data transfer from flash memory to the SD card requires careful consideration. The longer this interval, the less frequent the transfer process, resulting in lower power consumption. A seemingly straightforward approach is to maximize data storage until the flash memory reaches its capacity. In [46], the buffering event is triggered only after the EEPROM is filled. However, this method risks data overflow in the flash memory unless meticulously managed. Moreover, incorporating additional sensors into the data logger in the future might necessitate extensive programming modifications, adding to time consumption. There is also an increased risk of data loss in the event of system malfunction or failure. Therefore, it is crucial to find a balance between power consumption and data security when selecting the data transfer interval. Given that the data sampling interval in this study is set at 30 seconds, mirroring the approach in [11], it is reasonable to establish the data transfer interval at 120 cycles, equivalent to one hour. This duration strikes a practical balance, ensuring efficient power usage while minimizing the risk of data loss and accommodating future system enhancements without significant programming alterations.

*Table 2-6 Current and charge consumption comparison between SD card and the flash memory.*

		<b>Current</b>		
<b>Memory</b>	<b>Operation</b>	<b>Consumption (mA)</b>	<b>Period (ms)</b>	<b>Discharge (mC)</b>
	Initialization	65.49	4620	302.4
SD card	Read	62.10	8.34	0.516
	Write	58.15	144.1	8.38
Flash	Read	38.84	13.05	0.507
	Write	41.64	10.11	0.421

### **2.4.3.3 Remote Monitoring**

To grant access to remote monitoring, wireless communication must be built between the data logger and the SCADA supervisory computers. Wi-Fi as the wireless connectivity technology in this section has its unique advantages. To begin with, automatic time synchronization in logged data is suggested by IEC61724, which can be provided by Internet connection through Wi-Fi. Also, the Wi-Fi module is integrated in ESP32 requiring no extra components to build the connection. Bluetooth technology also features the above characteristics, while the maximum range and bandwidth is 50 m and 24 Mbps, which are quite limited compared to 300 m and 250 Mbps (802.11n protocol) for Wi-Fi.

Building the Wi-Fi connection is highly simplified in Arduino core with high level APIs. After including WiFi.h library and providing ssid/password, WiFi.begin() function will try to connect to the access point specified by the ssid/password pair. However, this function returns before the connection is really built, which usually takes seconds. One extra verification step is required



after `WiFi.begin()` to make sure the following codes which are dependent on the successful connection can execute as expected. `WiFi.status()` function is able to check the connection status. When the returned value is `WL_CONNECTED`, the rest of the codes can safely proceed. The measured current and electric charge consumption is measured and depicted in Figure 2-8. During the period of trying to connect to the access point, the average current and time required is 117.27 mA and 3.636 s. After the connection is built, the average current consumption is 80.99 mA.

An effective method to reduce the power consumption on Wi-Fi connection is to turn off unactive parts of ESP32. Deep sleep suits this scenario since everything except for RTC module and ULP coprocessor are powered off. While this method seems effective, one of the biggest challenges is the uncertainty of the time needed to build the Wi-Fi connection. Due to the impacts of physical distance between ESP32 and the Wi-Fi access point, and the transmission traffic caused by using certain channels, received Wi-Fi signal strength will vary which results in variance of connection time. A sketch is composed and uploaded to make ESP32 follow a routine that after it connects to assigned access point, delay it for two seconds, and go to deep-sleep mode for five seconds. Table 2-7 below records eight experiments of the time required to connect to the access point under identical conditions. They are also recorded using Power Profiler Kit II.

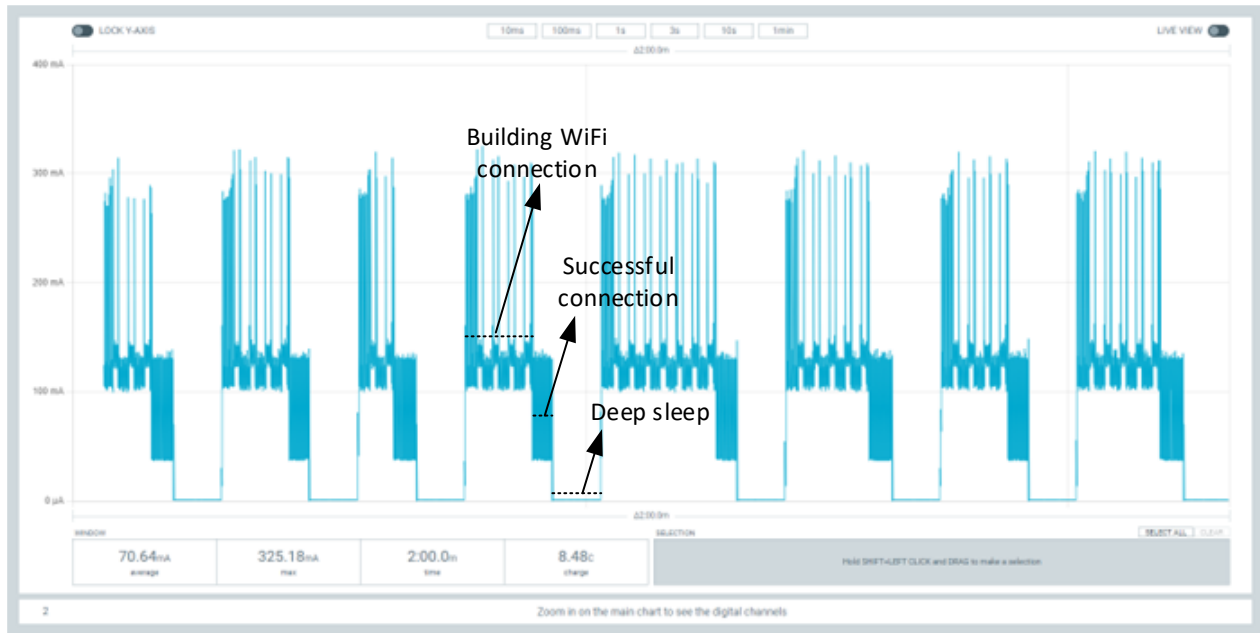


Figure 2-8 Current consumption during Wi-Fi connection.

Table 2-7 Average time and current for building Wi-Fi connection.

Quantity\Count	1	2	3	4	5	6	7	8	Average
Time required to									
build connection	5.236	7.049	4.047	7.049	12.04	9.078	7.049	9.051	<b>7.575</b>
(s)									
Current during									
building									
connection	117.26	119.65	114.92	119.50	119.85	117.57	119.58	117.79	<b>118.27</b>
(mA)									
Current after									
connection	is 61.46	60.56	60.83	61.94	60.53	61.26	61.17	64.27	<b>61.50</b>
built (mA)									

---

Current during	1.35	1.35	1.35	1.35	1.35	1.35	1.35	1.35	1.35	<b>1.35</b>
deep sleep (mA)										

---

To find the shortest Wi-Fi connection interval in terms of power saving when the power saving mode is deployed, a simple equation can be proposed to calculate. Suppose there are two scenarios for power consumption comparison: the first one (left side of Equation 2) is the Wi-Fi connection remains ON all the time with a relatively lower current consumption of 61.50 mA for  $T$  seconds; the second scenario (right side of Equation 2) is to build the Wi-Fi connection at 118.27 mA current for 7.575 seconds, then to consume 1.35 mA for  $(T-7.575)$  seconds representing the deep sleep mode when Wi-Fi connection is turned OFF. The marginal monitoring interval would make the electrical discharge identical for both scenarios, which is as Equation 2.

$$\begin{aligned}
 T \times 61.5 \text{ mA} &= 7.575 \text{ s} \times 118.27 \text{ mA} + (T - 7.575 \text{ s}) \times 1.35 \text{ mA} \\
 T &= 14.717 \text{ s}
 \end{aligned}
 \tag{2}$$

We can conclude that for any monitoring interval that is longer than 14.717 seconds, switching to deep-sleep mode after successful Wi-Fi transmission is a better choice in terms of power saving.

Frequent activation of Wi-Fi connectivity can lead to extended connection attempts due to fluctuations in radio signal strength, resulting in considerable power consumption. Assuming an equal likelihood of prolonged connection times in two identical scenarios, with the only difference being the frequency of Wi-Fi connection attempts, the scenario with more frequent

Wi-Fi attempts will inevitably lead to longer connection times and thus higher power usage. Table 2-8 illustrates the relationship between Wi-Fi connection intervals and the average current at a supply voltage of 3500 mV. It shows that with longer Wi-Fi connection intervals, the average current increases. However, identifying the optimal Wi-Fi connection interval isn't solely based on the average current; the monitoring times is also a critical factor. Ideally, a higher number of monitoring times, which corresponds to the shortest monitoring interval, coupled with lower average current consumption is preferable. Therefore, the best Wi-Fi connection interval is indicated by the largest ratio of monitoring times to average current. As presented in Table 2-8, the highest ratio of 2.86 is observed at the 45-second interval. Based on this data, the monitoring event in this study is set to occur every 45 seconds, balancing efficient power use with effective system monitoring.

*Table 2-8 The ratio of monitoring times to average current at different Wi-Fi connection intervals.*

Wi-Fi connection interval (s)	10	30	45	60
Monitoring times within 600 seconds	60	20	13.33	10
Average current (mA)	23.6	7.34	4.66	4.02
Monitoring times/ average current	2.54	2.72	<b>2.86</b>	2.49

A timeout mechanism should also be added into the sketch to prevent the battery is drained by restlessly trying to connect to Wi-Fi. Due to network issues the Wi-Fi will be down for from a few minutes to a few hours, leading to dramatic power loss of the battery. The timeout mechanism adopted in this section is simple: if it spends more than 45 seconds or any user-

defined time period on Wi-Fi connection which suggests potential network or the data logger problems, further tries on Wi-Fi connection is suspended until human investigation is finished.

## **2.5 System Design**

This section introduces the system design, including the algorithm flowchart and the system hardware assembly.

### **2.5.1 Algorithm Flowchart**

The entire code used in this section is developed with C++ language in Arduino IDE 2.2.1 version. Figure 2-9 presents the program flowchart. The program sets the data logger in deep-sleep power saving mode most of the time, and intermittently wakes it up to process tasks of data saving and displaying. One on-board LED is utilized to indicate the on-going data logging process. Two interactive buttons are in this system to enable human intervention: one added slide button is to control the start or stop of data logging, one push button from the microcontroller board is to restart the program if an unexpected error occurs during running. The slide button is on active low mode when data logging is enabled to reduce the power consumption by high side resistor, since most of the time the data logger should be ready to log sensor readings.

During initialization, libraries such as Preference.h, SD.h and WiFi.h are included to provide high-level functions and data structures to facilitate easy programming. Intervals including sensor sampling (30 s) and remote monitoring (45 s) are defined in setup stage. Also, in this stage pins for SPI connection with SD card and for collecting sensor readings are specified. Wi-Fi credentials are provided to ensure a successful connection.

After the setup, the microcontroller is programmed into deep-sleep mode to minimize power consumption. Only RTC modules are powered on to wake up the microcontroller every 30 seconds. After 30 seconds since initialization, it is activated again to collect sensor readings, and buffer them into the flash memory of ESP32. Multiple data types can be candidates for the data logger applications, an unsigned long integer variable is chosen in this section since the size of it is 4 bytes compared to 8 bytes of a float variable. More sensor readings can be stored in the form of `uint32_t` without compromising the data precision and with an upper limit large enough to record sensor readings. After choosing `ULong` as the data type in `preference.h`, function `putULong` will save values in the flash memory. The data structure of `preference.h` is key/value pair, thus the statement `preferences.putULong("Voltage1", Voltage1);` is able to save values in the flash. Also, the local time obtained from `ntpServer` will be saved according to the suggestion of IEC 61724-1:2021 Standard (Photovoltaic system performance - Part 1: Monitoring). If the flash memory is full and cannot buffer any new sensor readings, the data transfer process from flash to SD card will be initiated. Statement `preferences.getULong("Voltage1", 0);` will first retrieve the stored values from flash, and write the values into SD card after the SPI connection is checked to be successful. Saving event to SD card can be achieved by `appendFile(SD, "/Voltage1.txt", String(Voltage1).c_str());`, which appends the latest sensor readings to the existing file. Deep-sleep mode will be switched to if the displaying interval is not reached yet.

Displaying sensor data in web server is for real-time monitoring of the solar system status, to help technicians to identify faults in the early stage and ensure the system functionality and security. The first step is to connect to Wi-Fi access point. After `WiFi.status()` becomes `WL_CONNECTED` which suggests the successful connection between ESP32 and the access point, `configTime(gmtOffset, daylightOffset, ntpServer)` from library `time.h` will output local time

(GMT -2:30, Newfoundland Daylight Saving Time) to be saved along with sensor data.

Following is the customization of HTML web page that is displayed by entering local IP address.

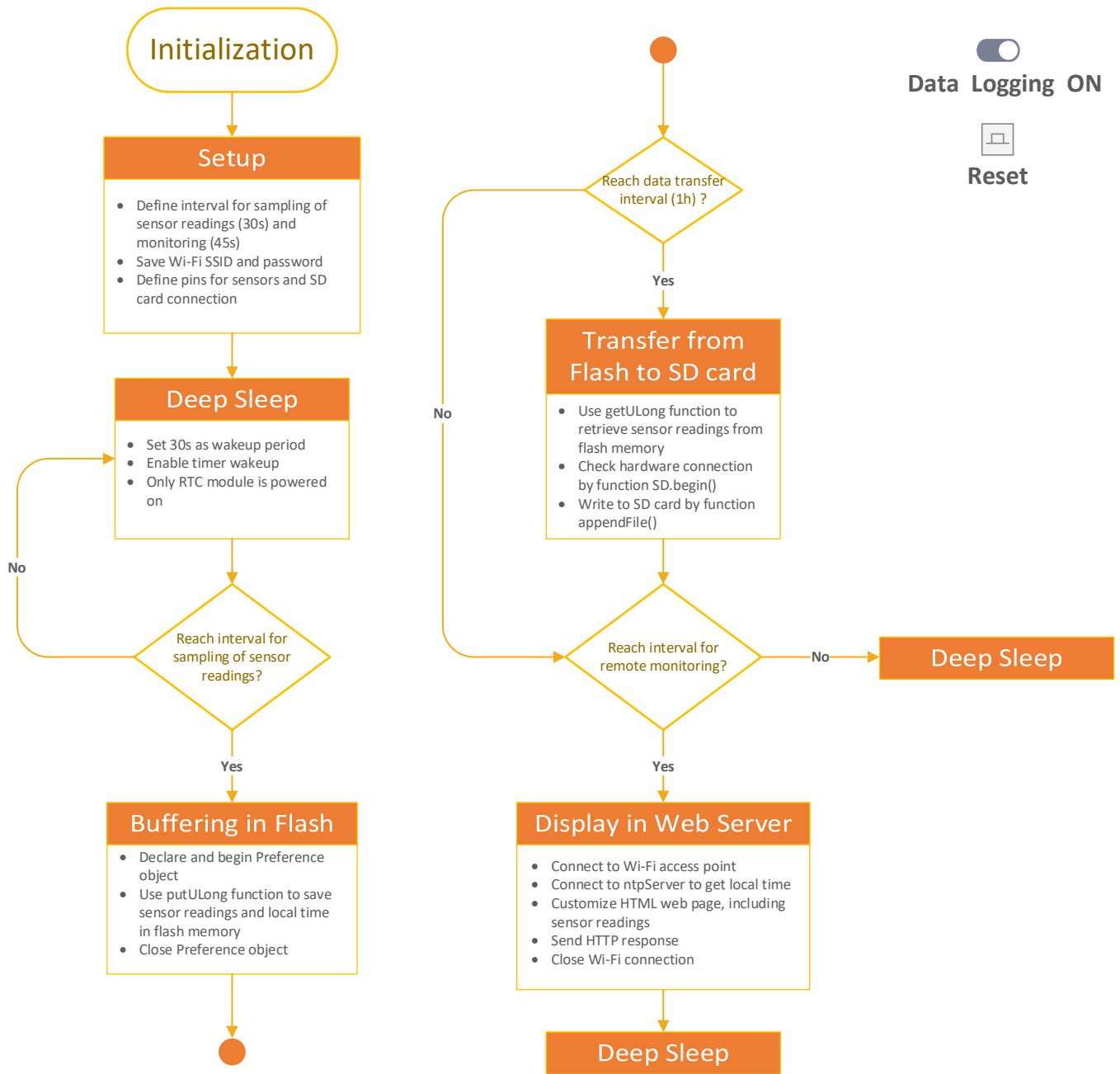


Figure 2-9 The flowchart of developed power saving algorithm.

## 2.5.2 System Hardware Assembly

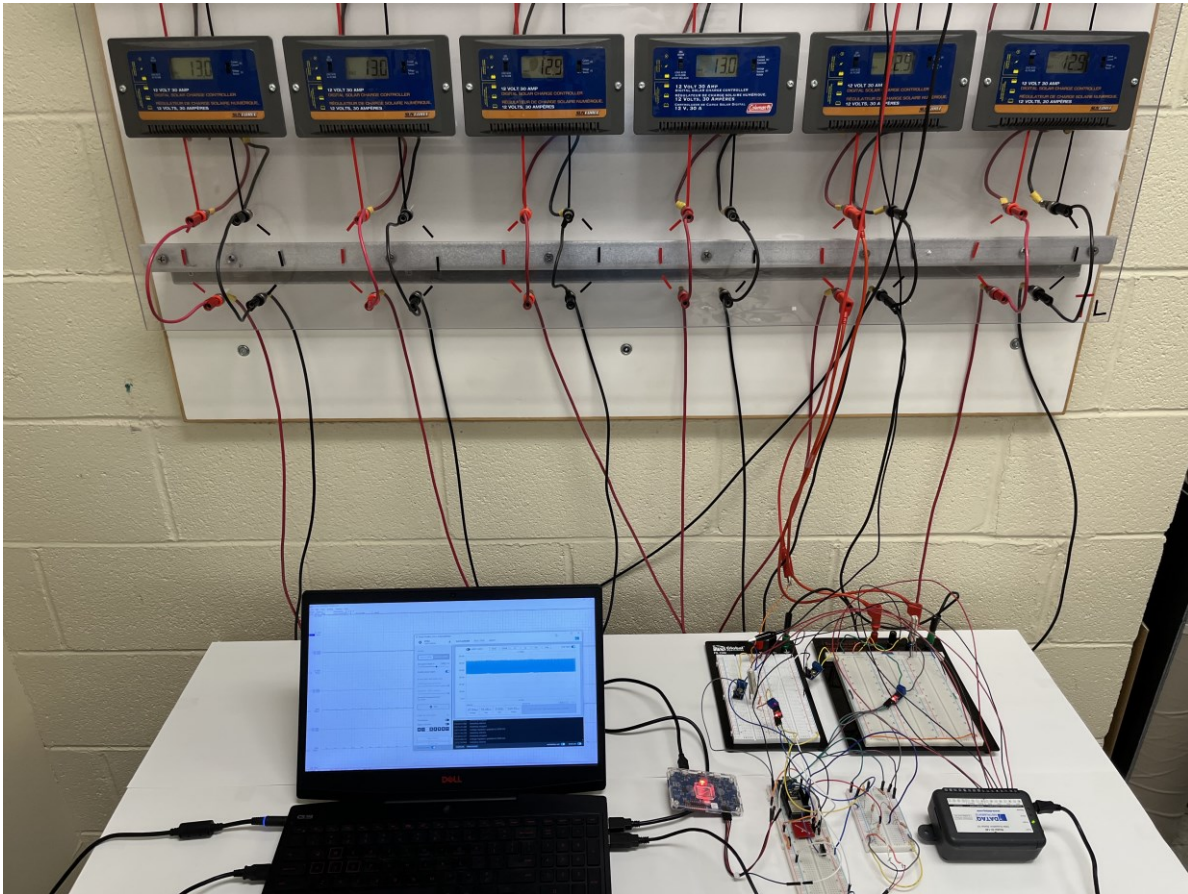
The standalone PV system includes PV panels, one MPPT controller, backup batteries, and an LED lamp. The data logger to monitor the standalone PV system consists of one microcontroller, two voltage sensors, two current sensors, and one SD card. According to Table 2-2, the maximum number of analog pins is 13. Considering four analog pins are necessary for four sensors to monitor one set of a PV panel and its corresponding backup battery, three sets of PV panels and batteries can be monitored by one single FireBeetle 2 ESP32-E microcontroller theoretically. The maximum discharge current is 600 mA, which also suffices to power the sensors. By measuring more sets of PV panels and batteries, the unit cost can be reduced largely. In this section only one set is monitored, due to the limitation of DATAQ® DI-145 to be introduced, which serves as the control group to verify the accuracy of the data logger developed in this section. DI-145 has only four analog input channels, which cannot verify the results of three sets of PV panels and batteries.

To complete a control group of the experiments and to measure the overall power consumption of developed data logger in this section, DATAQ® DI-145 and Power Profiler Kit II from Nordic Semiconductor® are added in the system hardware, respectively. DATAQ® DI-145 is a data acquisition device with USB connectivity. With four-channel analog inputs that can measure  $\pm 10$  volts and two-channel digital inputs which can measure  $\pm 30$  volts, DI-145 is capable of being the data logger for the PV system in this section. Real-time display can also be achieved by the included software *WinDaq* with USB cable. The only modification is the division of voltages to be measured since the PV panel voltage and the backup battery voltage are usually around 13 volts, larger than the upper limit of DI-145. Nordic Semiconductor® Power Profiler Kit II serves as a configurable voltage source ranging from 800 mV to 5.0 V for



power source. With USB communication and the application *nRF Connect for Desktop*, Power Profiler Kit II records the current consumption down to near 200 nA with a high resolution of 200 nA. The maximum measurable current of Power Profiler Kit II is 1 A. Featuring 8 digital pins for accurate tracking, several events can be easily labeled as they commence and terminate. For low-power consumption scenarios including low-power microcontroller development, Power Profiler Kit II is quite suitable for precise current and power measurements.

Figure 2-10 presents the final setup for the experiment. Two voltage sensors are connected in parallel to the PV panel and the battery to measure the voltages, while two current sensors are in series with the PV panel and the battery to measure the currents. Minor modifications are made to let DI-145 log sensor data correctly. Channel 1 and Channel 2 of DI-145 are allocated to log voltages. Since the maximum measurable voltage is 10 volts, two 1k Ohms resistors should be placed between the VCC and ground of PV panel and battery each to ensure DI-145 can function properly when channel 1+ and 2+ are connected at the middle of the resistors respectively. For current logging shunt resistors are used where Channel 3 and Channel 4 are connected. Power Profiler Kit II does not need such modifications. One side of it is connected to a personal computer to enable current waveform displaying and logging, and the other side is to provide power to the microcontroller by connecting to VCC pin and ground pin of the microcontroller.



*Figure 2-10 Experiment test setup.*

## **2.6 Experimental Validation**

This section introduces the data logging results compared with those of DATAQ DI-145, the power consumption cost, and the remote monitoring and costs for the developed data logger.

### **2.6.1 Datalogging Results Comparison with DATAQ DI-145**

Figure 2-11 and 2-12 are the monitoring results by DATAQ DI-145 over 24 hours, where voltages of PV panel and the battery, currents of PV panel and the battery, are collected once every second. At 01:30, a 12 V 8.5 W LED lamp is turned on as the electrical load added into the system. The battery voltage drops immediately from around 13 volts to about 11.7 volts. The battery current becomes negative meaning it is supplying current to the LED lamp. Until 07:00,

the PV panel voltage remains near zero due to inadequacy of solar radiation. The battery voltage also remains at a low level. After 07:00, the PV panel voltage increases rapidly, while the battery voltage delays around 40 minutes to increase since the PV panel voltage needs to be larger than the battery voltage to achieve charging process. The PV panel current and the battery current increases at the same rate, while the former is always 0.7 A larger than the latter since the PV panel is charging the battery and powering the load. At 13:30, the LED lamp is turned off. PV panel voltage rises for around 4 volts, and the battery voltage declines instantly. The battery current is also near zero. After 18:00, the voltage of PV panel starts to decline; the voltage of battery also follows the same pattern.

Recordings of voltage and current sensor readings by the low power data logger developed in this section are revealed in Figure 2-13 and 2-14. The datalogging event happens every 30 seconds. Over the monitored period, the voltages and currents of PV panel and the battery demonstrates a good fitting with the results by DI-145. This comparison manifests that the developed low power data logger in this section is well-functioning during this experiment.

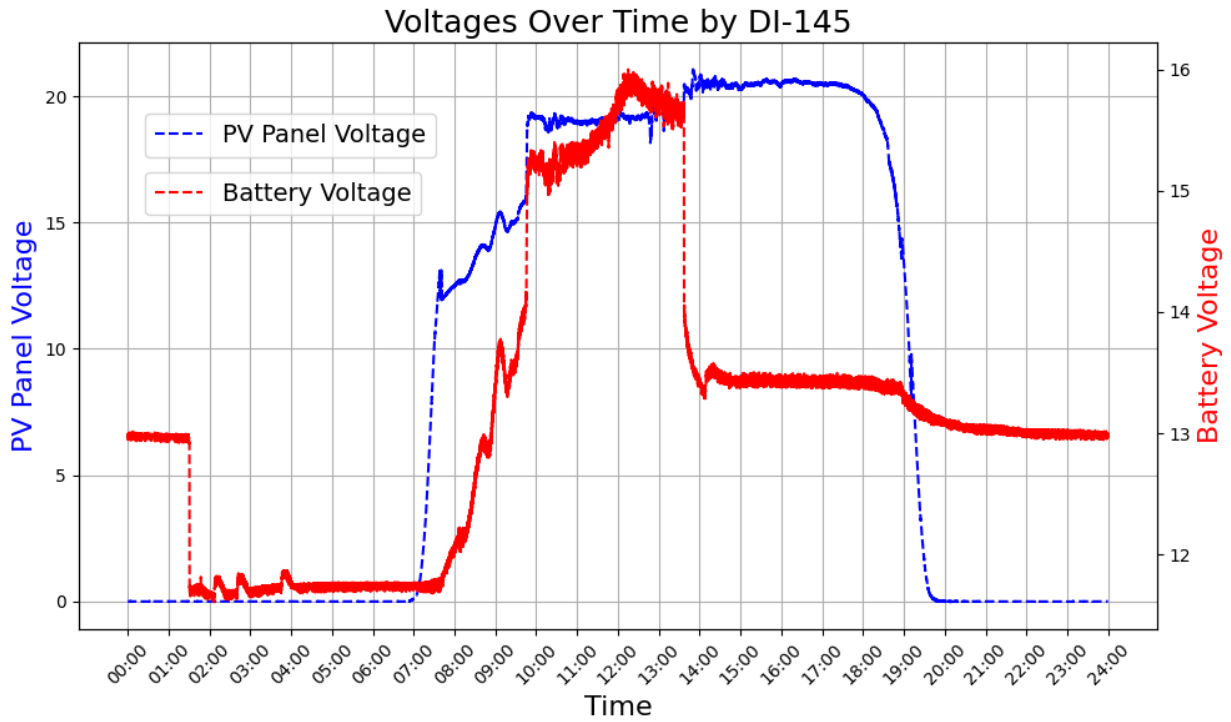


Figure 2-11 Voltage logging scatter plot over 24 hours by DI-145.

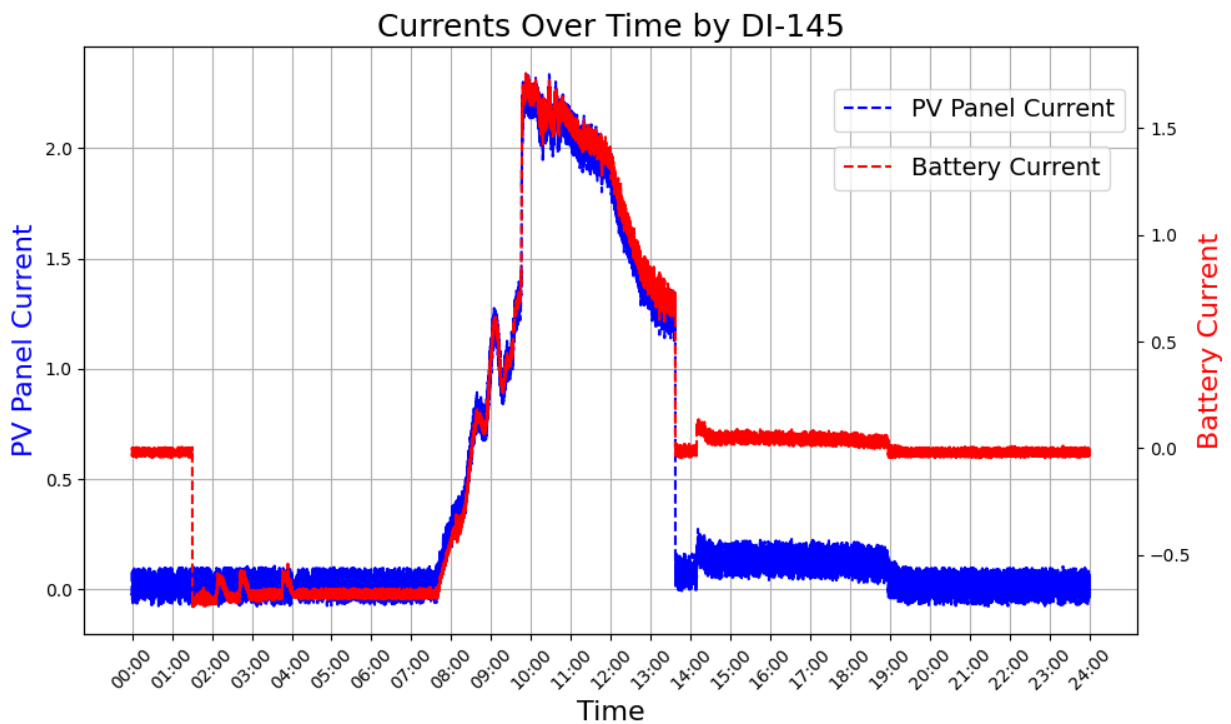


Figure 2-12 Current logging scatter plot over 24 hours by DI-145.

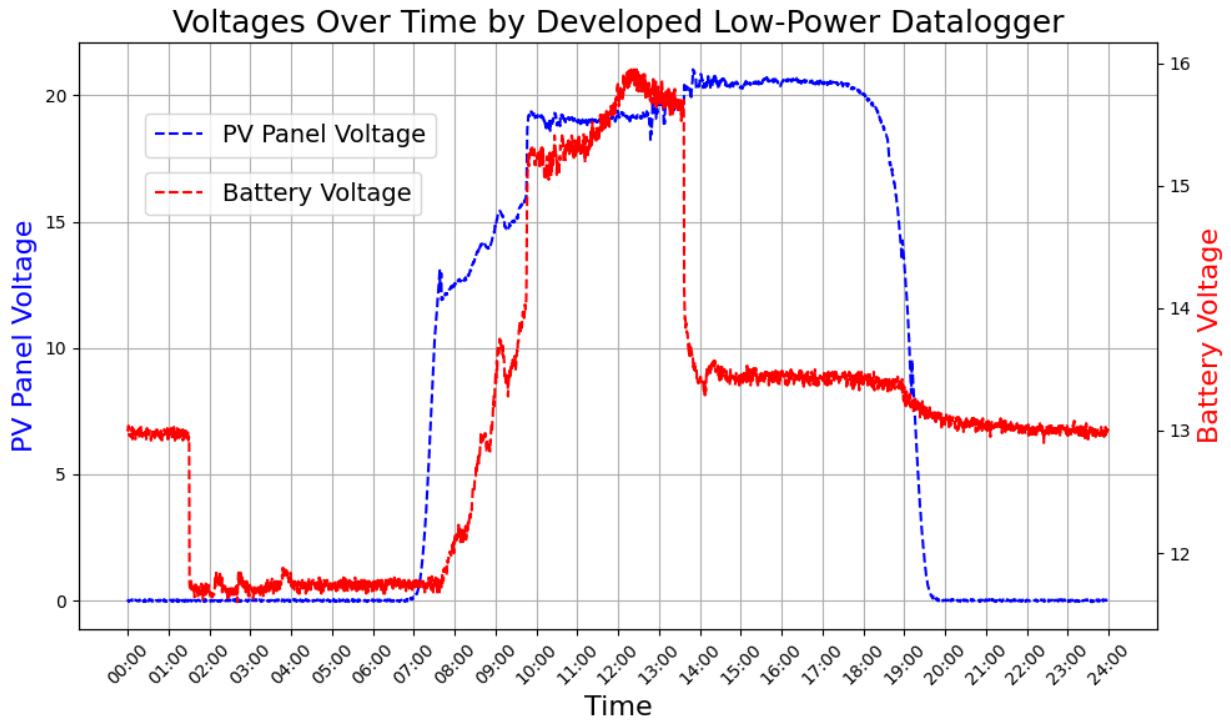


Figure 2-13 Voltage scatter plot over 24 hours using the developed low-power data logger.

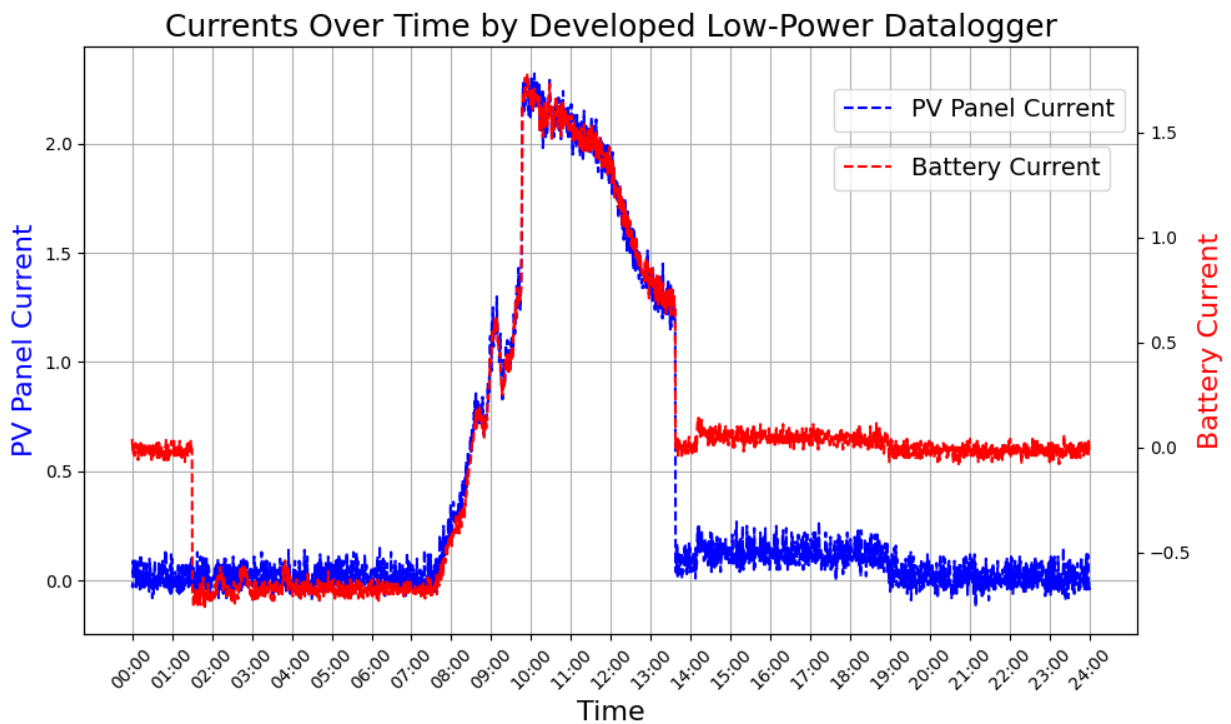


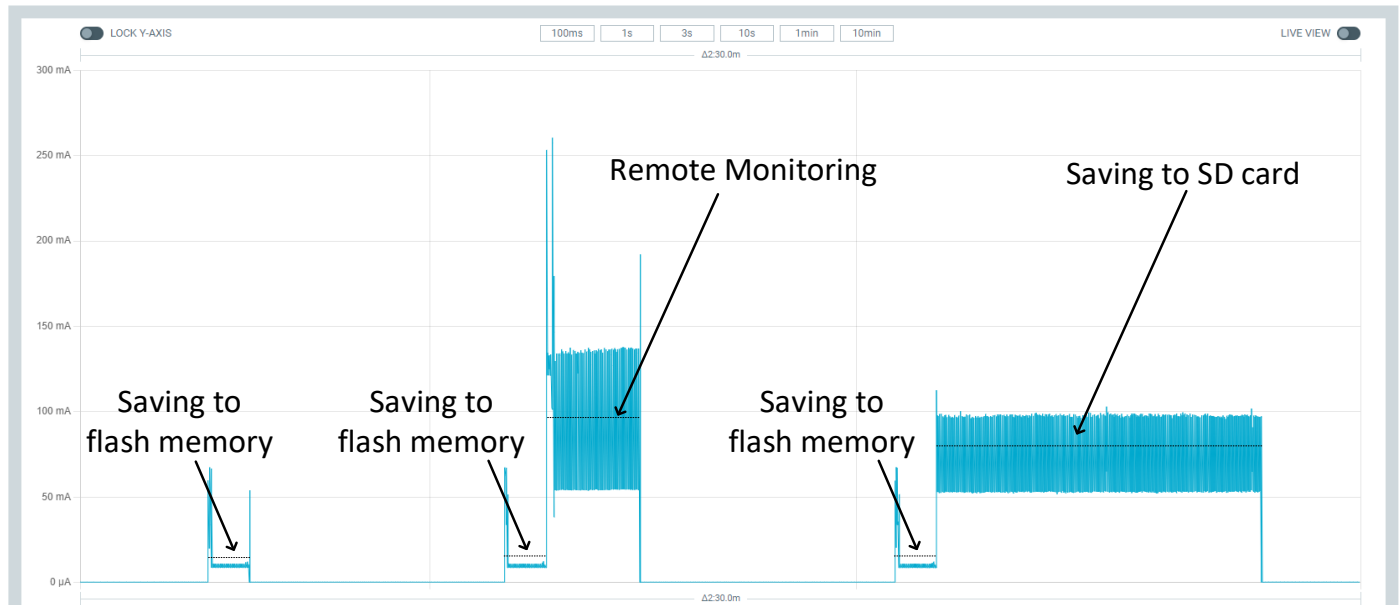
Figure 2-14 Current scatter plot over 24 hours using the developed low-power data logger.

## 2.6.2 Power Consumption Result

In this section the detailed power consumption during one complete period is explained and measured. Considering the impact from supply voltage and operating frequency on power consumption discussed in Section 3.1 and 3.2, 3500 mV is the supply voltage. Any supply voltage below 3500 mV will result in the failure to initiate Wi-Fi connection. However, real battery voltage decreases along with the discharge progress, thus 3500 mV cannot be maintained. This may vary the final power consumption obtained by assuming the supply voltage is constant. 10 MHz as the CPU frequency during events that sensor readings are saved into the flash memory and 240 MHz for the remaining of the time are chosen as the operating frequencies. Voltage that is below 3500 mV, such as 3300 mV, is not sufficient to power SD card transflash breakout, thus is not suitable as a power saving method in this section. Voltages larger than 3500 mV can finish each task precisely, while a larger power consumption is also observed. For the choice of CPU frequency, it is more flexible than supply voltage since the CPU frequency can be adjusted according to the specific task to be executed. Before the execution of saving sensor readings to flash memory, a minimum frequency (10 MHz) would be switched to for minimization of power consumption. However, the other tasks (remote monitoring and saving flash memory contents to SD card) require maximum CPU frequency. Otherwise, the microcontroller would not be capable of accomplishing the datalogging in terms of the high system latency by low CPU frequency.

Figure 2-15 demonstrates the complete progress of the datalogging. Flash memory saving events happen every 30 seconds, costing 60.8 mC. Every 45 seconds, sensor readings are displayed on the local web server, with the charge consumption of 1.17 C. Data transferring from the flash memory to the SD card occurs every 1 hour at the expense of 2.63 C. Combining the three types

of tasks together, 28.76 mA is the average current consumption and 100.66 mW is the average power consumption of the developed low power data logger excluding the sensors. Taking sensors into consideration, the average power consumption is **122.78 mW**. As a comparison, the power consumption of DI-145, which does not have ability to remotely display collected sensor readings, is 0.30 watts, according to the datasheet [47].

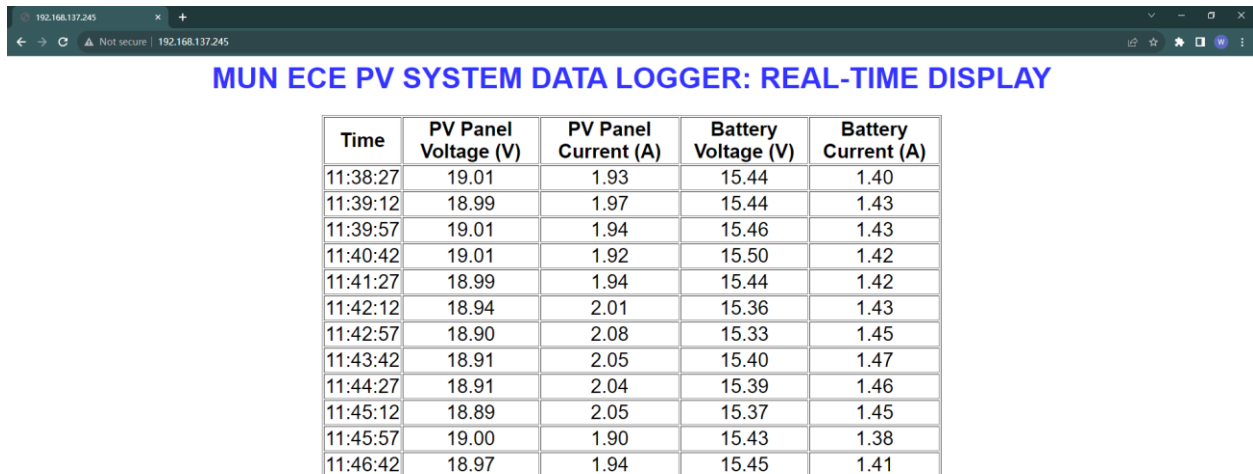


*Figure 2-15 Current consumption including all scenarios of the data logger tasks.*

### 2.6.3 Remote Monitoring and Costs for the developed data logger

The mA-level current consumption of developed low power data logger is mainly due to the need for remote monitoring, which allows users to access system parameters in real time. The interface is illustrated in Figure 2-16, showing the real-time display on Nov.19. The PV panel voltage, PV panel current, battery voltage, and battery current can be viewed via the web page with renewing frequency of 1/45 Hz. This web server could be accessed by entering 192.168.137.245 in the browser during the experiment, while no longer accessible currently.

Low-cost is one of the principal features of the developed PV system data logger. Table 2-9 lists the prices of all components of the data logger including sensors. The total price is C\$ 55.05, which is budget-friendly and beneficial for laboratories to research and conduct experiments in related fields.



Time	PV Panel Voltage (V)	PV Panel Current (A)	Battery Voltage (V)	Battery Current (A)
11:38:27	19.01	1.93	15.44	1.40
11:39:12	18.99	1.97	15.44	1.43
11:39:57	19.01	1.94	15.46	1.43
11:40:42	19.01	1.92	15.50	1.42
11:41:27	18.99	1.94	15.44	1.42
11:42:12	18.94	2.01	15.36	1.43
11:42:57	18.90	2.08	15.33	1.45
11:43:42	18.91	2.05	15.40	1.47
11:44:27	18.91	2.04	15.39	1.46
11:45:12	18.89	2.05	15.37	1.45
11:45:57	19.00	1.90	15.43	1.38
11:46:42	18.97	1.94	15.45	1.41

Figure 2-16 Web page showing electrical parameters of monitored PV system.

Table 2-9 Budget for the developed low-power and low-cost data logger.

Components	Price [C\$]
FireBeetle 2 ESP32-E	10.70
2 HiLetgo voltage detection module DC 0-25V	9.99
2 ACS712 Hall effect current sensor module 5A	12.58
SanDisk 32GB Ultra SDHC UHS-I Memory Card	11.90
SparkFun® MicroSD transflash breakout	7.88
Miscellaneous parts (wires, resistors, one slide button)	2.00
<b>Total</b>	<b>55.05</b>



## 2.7 Conclusion

This chapter introduces an IoT data logger specifically designed for PV system monitoring, characterized by its low power consumption and affordability. The PV system under study includes a single PV panel, a charging controller, and a backup battery. The primary focus is on monitoring the voltages and currents of both the PV panel and the battery. To achieve this, a comprehensive sensor network consisting of two voltage sensors and two current sensors is employed. The data gathered are not only stored on an SD card but are also dynamically displayed on a web server. The FireBeetle 2 ESP32-E microcontroller was selected for processing sensor data, primarily due to its efficient power usage in deep-sleep mode. Several low-power strategies are implemented in this study. Firstly, following the power consumption formula for CMOS circuits, a reduction in supply voltage and CPU frequency is shown to significantly lower power consumption. For optimal functionality, a supply voltage of 3500 mV and an auto-adjusted CPU frequency are chosen. During flash memory save events, operating at a minimum CPU frequency of 10 MHz efficiently reduces power consumption by 60% compared to the maximum frequency. Then, the microcontroller's deep-sleep mode is activated to deactivate non-essential components during periods of inactivity. Additionally, a data buffering mechanism is in place, where sensor readings are initially stored in the microcontroller's flash memory and transferred to the SD card every one hour. This approach is taken because writing to the SD card requires significantly more power than storing data in flash memory. Choosing one hour as the transferring interval is based on a practical balance ensuring efficient power usage, minimization of risk of data loss, and accommodating future system enhancements. Another consideration is the variability in Wi-Fi connection frequencies. Thus, a critical data display interval of 14.717 seconds is established by finding the threshold value to

make the two scenarios' power consumption identical. If the interval is shorter than this threshold, the deep-sleep mode is not utilized to optimize power efficiency. Monitoring events happens every 45 seconds to balance efficient power use with the shortest possible system monitoring interval. Finally, the total power consumption of the developed data logger is 122.78 mW on average. A comparative analysis with the commercial data logger DI-145 validates the functionality of this novel system. The correlation of the collected sensor data with practical observations confirms the efficacy of this low-power PV system data logger. The overall cost of the system is a modest C\$ 55.05, making it accessible to most research groups. For future suggestions and the limitation, while this study demonstrates the monitoring of one PV panel and one battery with a single microcontroller, the system is scalable to monitor up to three PV panels and batteries simultaneously, potentially reducing the per-unit cost. Also, the power consumption obtained in this section is by assuming that the supply voltage keeps constant at 3500 mV, while the battery voltage decreases over time in practical applications.

## **2.8 Co-authorship Statement**

This chapter has been published in Journal of Electronics and Electrical Engineering (JEEE, ISSN: 2972-3280) by Wei He and Dr. Mohammad Tariq Iqbal. Dr. Mohammad Tariq Iqbal is the academic supervisor of Wei He in his graduate program at Memorial University. The link is <https://doi.org/10.37256/jeee.2220233795>.

All authors contributed significantly to the work presented in this chapter. Their individual contributions are outlined below.

Wei He: Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Visualization.

Mohammad Tariq Iqbal: Conceptualization, Resources, Writing - Review & Editing, Supervision, Project Administration, Funding Acquisition.

# **Chapter 3. A Novel Design of a Low-cost SCADA System for Monitoring Standalone Photovoltaic Systems**

## **3.1 Introduction**

Increased installation capacity of photovoltaic (PV) systems worldwide in the last two decades suggests that a higher green energy harvest from the Sun has been achieved. In 2000, cumulative installed solar PV capacity worldwide is 1,288 MW, while this number in 2022 was 1,177,000 MW [48]. The energy production performance of PV systems does degrade naturally with time; but any system faults should be detected and alarmed in time. Monitoring the performance of PV systems is thus crucial, since human intervention at an early stage can prevent serious damage from occurring.

Supervisory control and data acquisition (SCADA) is an industrial concept that helps to monitor and control industrial processes remotely, integrating hardware and software components to store and analyze real-time data [49]. Typically, SCADA system comprises of several components: one master terminal unit (MTU) to manage and monitor the other components in the SCADA network, remote terminal units (RTUs) to gather field information or control field devices, human-machine interface (HMI) to allow operators to view system parameters intuitively, sensors and actuators to collect information and execute physical actions, data historian to store historical data and display trends, and communication network to transfer data between MTU and RTUs.

Applications of SCADA system in PV system monitoring have been researched in many papers. Reference [7] investigated an Internet of Things (IoT) based SCADA system for PV system monitoring and control, featuring low cost and open source. Node-RED programming tool was used to present graphical interface that can be easily interacted. In [50] the authors developed a low-cost SCADA system to monitor a standalone PV system using Reliance SCADA software and MODBUS RTU protocol. These findings provide insightful contributions to SCADA system in PV system monitoring, however, none of them illustrate an HMI design that consumes low power and displays historical data; none of them proposed a remote data storage solution that supports large data storage without extra charge.

Predominantly, the research community developed different designs of SCADA systems for monitoring PV systems. Table 3-1 presents the comparison between related work.

*Table 3-1 Comparison of HMI designs and data storage solutions in the literature review.*

Reference	HMI Design				Data Storage	
	Customized Web Server	Website/ Software	LCD	Mobile App	SD Card	Website
[41]	√		√		√	
[42]		√	√		√	
[51]				√		√
[43]	√				√	
[44]		√			√	
[52]		√	√	√		√

For HMI designs, authors were using the customized web server [41,43], the third-party website [42] or software [43,52] and the LCD [41,42,52] to display the monitored parameters. A customized web server was built to display monitored electrical parameters from a PV panel rated at 15 W and environmental parameters [41]. The Arduino UNO Wi-Fi Rev2 built the web server via the embedded Wi-Fi chip. Every eight seconds, the HTML codes that corresponded to the web page were generated by a program running on the Arduino board. However, this method can only display a limited amount of data at a certain moment. The monitored parameters were also displayed on a Liquid Crystal Display (LCD) LM041L module, while lacking the ability to show the data trend. ThingSpeak is an open IoT platform that collects sensor data, shows the data in charts, and provides various plugins and mobile applications. This platform was used to store and display environmental and electrical parameters from a 30 W polycrystalline PV panel in [42]. While ThingSpeak platform integrates multiple services and functions for IoT applications, only eight channels of data were available for free to be transferred to the website. A 4×20 LCD with I2C interface was connected to the Arduino UNO board to be turned on with each iteration to display monitored results. Authors of [51] proposed that the PV panel current and power was first transferred from STM32 chip to Raspberry Pi 3 module via UART Tx/Rx interface. Then, the data was forwarded to a cloud server by Wi-Fi module on Raspberry Pi 3 and displayed on an Android app. For data visualization in an online monitoring system for PV panels, HTML containing real-time data was created by Raspberry Pi 3 [43]. Then, the HTML page was sent to a web server via Wi-Fi to display the monitored parameters. The HMI design by LabVIEW GUI software enabled users to retrieve the performance data of the PV system in [44]. The interface could start data logging, check Bluetooth connection, and download logged data into the MTU. However, no data trend could be observed in this work. In [52], the monitored parameters were displayed on an Android app, ThingSpeak platform, and an LCD. Each Bluetooth message

comprised of values of parameters and fixed prompt symbols, such as “I = ; V = ”. No historical data could be seen from the mobile app and the LCD.

For data storage solutions, the SD card was massively adopted by the previous works [41] [42] [43] [44]. ThingSpeak is another popular choice for remote data storage [51] [52]. Nevertheless, data stored in the SD card are difficult to retrieve, especially when the standalone PV system locates in remote areas. Although ThingSpeak provides limited free service for messages under 3 million/year, only eight channels are available to upload the monitored variables.

In this chapter, a novel design of HMI using a free Bluetooth Low Energy (BLE) mobile app to display monitored PV system parameters in live plot, and a new method of data storage using a website to massively and freely store the historical data are proposed. Chapter 3.2 introduces the materials and methods required to implement the experimental setup. Chapter 3.3 presents the experiment results, while Chapter 3.4 justifies the effectiveness of the proposed design. Chapter 3.5 concludes the section. Co-authorship statement comprises Chapter 3.6.

## **3.2 Materials and Methods**

For hardware components, an Arduino® UNO R4 Wi-Fi development board functioned as the RTU, an INA3221 module served as the voltage and current sensor. Software components comprised an Android application BlueTooth Terminal eDebugger as the HMI, a website PVOutput.org as the remote data historian. Section 2.1-2.4 introduces hardware and software components used in this section, and Section 2.5 presents the experiment method.

### 3.2.1 Arduino® UNO R4 Wi-Fi

With UNO form factor, Arduino® UNO R4 Wi-Fi maintains the same pinout as the classic UNO Rev3. Projects can be transitioned from UNO Rev3 to UNO R4 Wi-Fi effortlessly. The faster clock than UNO Rev3 enhances its ability to handle complex project tasks. Furthermore, developers can add wireless connectivity to the projects due to the ESP32-S3 module, which supports Wi-Fi and Bluetooth. The larger flash and RAM capacity also guarantee larger data storage and faster computational process. Table 3-2 lists the technical specifications of UNO Rev3 and UNO R4 Wi-Fi.

*Table 3-2 Comparison of technical specifications between Arduino® UNO R4 Wi-Fi and Arduino® UNO Rev3.*

	Arduino® UNO R4 Wi-Fi	Arduino® UNO Rev3
SKU	ABX00087	A000066
Microcontroller	Renesas RA4M1 (Arm® Cortex®-M4)	ATmega328P
Digital I/O Pins	14	14
Analog Input Pins	6	6
Operating Voltage (V)	5 (3.3, for ESP32-S3)	5
Input Voltage (V)	6-24	7-12
DC Current per	8	20



I/O Pin (mA)		
Clock Speed (MHz)	48 (up to 240, for ESP32-S3)	16
Memory	256 kB Flash, 32 kB RAM	32 kB Flash, 2 kB SRAM
Wi-Fi & Bluetooth connectivity	Yes	No
I <sup>2</sup> C	Yes	Yes

### 3.2.2 INA3221 Voltage Monitor

Featuring three-channel and high-side, the INA3221 can monitor both voltage drops across the shunt resistor and the bus supply voltage with an I2C or SMBUS compatible interface. The range of sensed bus voltage varies from 0 V to 26 V, with a gain error of 0.25% at maximum. The INA3221 is powered at a voltage from 2.7 V to 5.5 V, consuming typically 350  $\mu$ A. Figure 3-1 illustrates the schematic diagram of the INA3221 module.

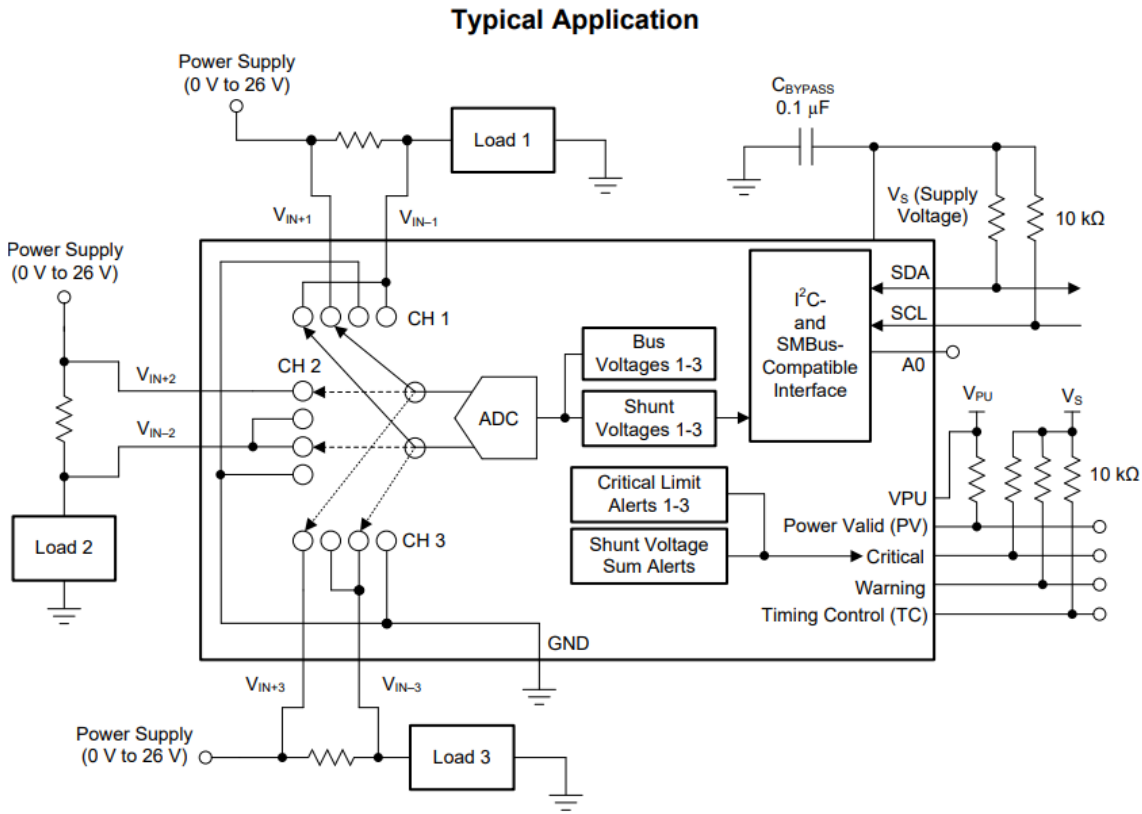


Figure 3-1 INA3221 schematic diagram.

### 3.2.3 Bluetooth Terminal eDebugger

Released in April 2023, Bluetooth Terminal eDebugger (BTed) [53] is a Bluetooth debugging assistant application running on Android OS. BTed supports both Bluetooth Classic and BLE protocols, offering flexibility to Bluetooth developers. The technical specifications of Bluetooth Classic and BLE are presented in Figure 3-2 [54]. With considerably less power consumption down to 0.01-0.50 W, the BLE protocol has the edge over Bluetooth Classic in IoT projects where the power consumption is one of the most crucial aspects. A highlighted function of BTed is to visually display the changes of data, by drawing the received data into a waveform diagram in real time. Since the maximum number of displayed data points is fixed, the higher the received data rate, the faster the plot scrolls.

<i>Specifications</i>	<i>Classic Bluetooth</i>	<i>Bluetooth Low Energy</i>
Range	100 m	Greater than 100 m
Data rate	1–3 Mbps	125 kbitps – 1 Mbps – 2 Mbps
Application throughput	0.7–2.1 Mbps	0.27 Mbps
Active slaves	7	Not defined
Frequency	2.4 GHz	2.4 GHz
Security	56/128-bit	128-bit AES with Counter Mode CBC-MAC
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	24-bit CRC, 32-bit Message Integrity Check
Latency	100 ms	6 ms
Time Lag	100 ms	3 ms
Voice capable	Yes	No
Network topology	Star	Star
Power consumption	1 W	0.01 - 0.50 W
Peak current consumption	less than 30mA	less than 15mA

*Figure 3-2 Comparison of technical specifications between Bluetooth Classic and BLE [54].*

### **3.2.4 PVOutput.org**

PVOutput.org is a free service for sharing and comparing PV output data. 2,515,654 solar panels are monitored, and 61,480,020 PV panel outputs are recorded [55]. Two forms of uploading data are supported by PVOutput.org: CSV loader and Live loader. The former records data daily, without historical limits. The latter allows intraday data upload at an interval of up to every five minutes. Only data from within the previous 14 days can be stored by Live loader. Data from an older date should thus be stored by CSV loader. Table 3-3 presents the differences between the two uploading modes.

Table 3-3 Characteristics of CSV loader and Live loader in for updating data in PVOutput.org.

	CSV loader	Live loader
Maximum Previous Days	No Limit	14 (90, in Donation Mode)
Data Interval	Up to one day	Up to five minutes
Maximum Number for per upload	200	288
Supported Parameters by both	Output Date, Energy Generation, Energy Consumption	Output Date, Energy Generation, Energy Consumption
Supported Parameters by CSV loader exclusively	Energy Exported, Peak Power, Peak Time, Conditions, Temperature Min, Temperature Max, Comments, Import Peak, Import Off Peak, Import Shoulder, Import High Shoulder, Export Peak, Export Off Peak, Export Shoulder, Export High Shoulder	/
Supported Parameters by Live	/	Output Time, Power Generation, Power

loader exclusively		Consumption, Temperature, Voltage
-----------------------	--	--------------------------------------

### 3.2.5 Experimental Setup

A PV test system in Memorial University comprised of two PV muls at 130 W rating each, an MPPT charging controller to regulate the PV output current, a backup battery as the energy storage, and a 12 V /50 W light bulb as the load. Sunforce 260 W Crystalline Solar Kit contains two of the 130 W PV panels, where each has a maximum voltage of 12 V. Each panel has the dimensions of 63.5 x 34 x 13 inches. Figure 3-3(a) describes the SCADA system block diagram, and Figure 3-3(b) displays the experimental setup in the PV lab at Memorial University of Newfoundland. A PV panel with nominal output power of 130 W and current of 7.6 A was monitored, where the output voltage and output current were collected by an INA3221 module. Arduino® UNO R4 Wi-Fi saved the collected voltage and current in the flash memory, then transferred them via Wi-Fi and BLE. For data display in BTeD, collected data would scroll continuously on the interface when a mobile device running BTeD was available nearby. For data storage in PVOutput.org, data within 14 days were uploaded through Live loader every five minutes; while data before 14 days would be reshaped and uploaded according to the requirements of CSV loader. Figure 3-4 describes the flowchart for the data monitoring process.

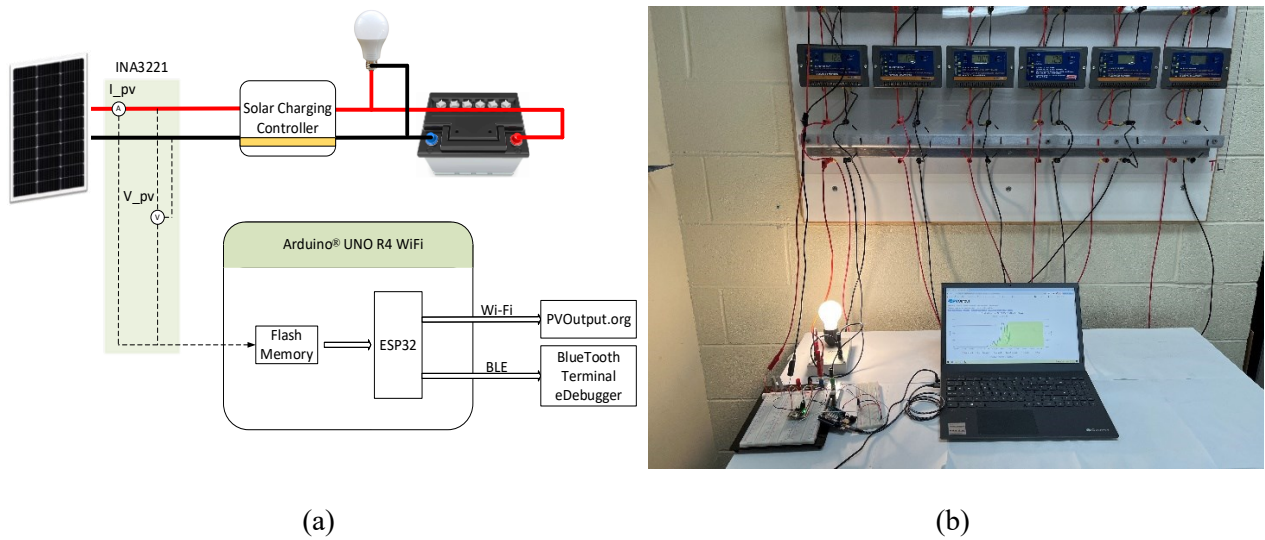


Figure 3-3 (a) The designed SCADA system block diagram. (b) Experimental setup.

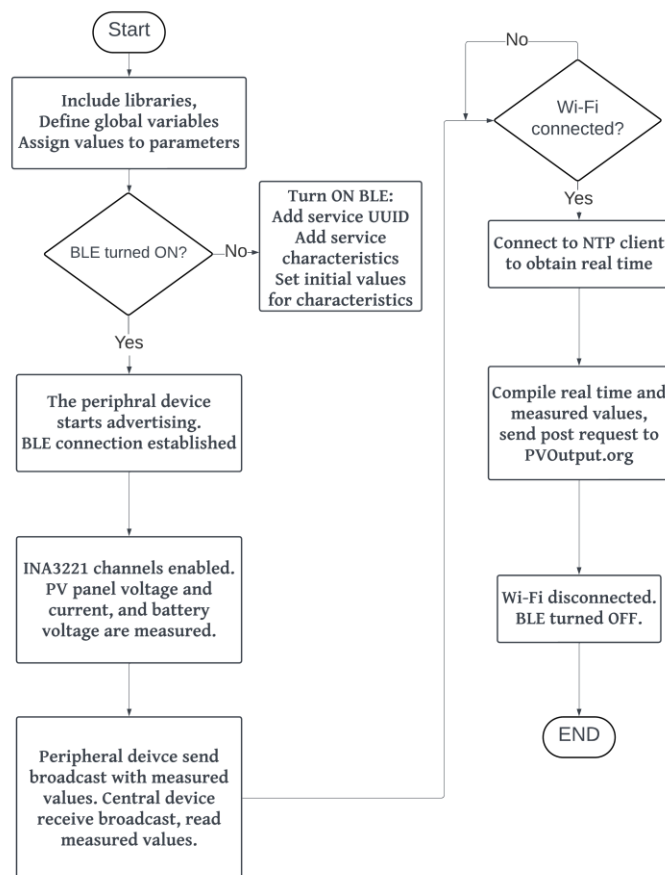


Figure 3-4 The SCADA system program flowchart.

### 3.3 Experimental Results

Table 3-4 lists the BLE service information. A Universally Unique Identifier (UUID) is a globally unique 16-byte number to identify profiles, services, and data types in a Generic Attribute (GATT) profile. The BLE specification supports shortened 16-bit UUIDs for the purpose of efficiency. (0x)19 represents 19 W power produced by the PV panel. Figure 3-5 illustrates the output power of the PV panel in BTeD in a real-time manner. Data received within 55 seconds were displayed on the screen simultaneously. Power data received after 55 seconds scrolled to the right side of the screen, while the data received earliest disappeared from the left side. In Figure 3-5(a) the PV panel power rose from 16.4 W to 23.1 W at 27 seconds. Conversely, the PV panel power decreased to 18.2 W from 21.0 W at 41 seconds as shown in Figure 3-5(b). From the start to 55 seconds, the PV output power remained around 27 W, with a fluctuation of 3 W. During the data measurement, a digital multimeter was also added to verify the accuracy of the collected data. The results from the multimeter were observed to be the same as the power shown in BTeD. The received rate also fluctuated between 2 B/s to 5 B/s.

*Table 3-4 BLE service information between BTeD and the RTU.*

PV System BLE Service	UUID	Result
Generic Access	00001800-0000-1000-8000-00805f9b34fb	\
Generic Attribute	00001801-0000-1000-8000-00805f9b34fb	\
PV power Service	0000180f-0000-1000-8000-00805f9b34fb	\
Power Characteristic	00002a19-0000-1000-8000-00805f9b34fb	\
Property	\	Read, Notify
Value	\	(0x)19

The output power of the PV panel on Dec. 10, 2023, was recorded on PVOutput.org website, as shown in Figure 3-6. The deep green line represented the power generated by the PV panel, while the light green area represented the energy generated by the PV panel. The red line indicated the battery voltage during the 24 hours. Before 8:15 am, there was no output power from the PV panel, nor was the energy output. After 8:15 am, starting from 1 W, the output power rapidly increased to 23 W at 10:15 am. The output power peaked at 12:15 pm, with a value of 40 W. After noon, the output power generally decreased with occasional upward fluctuations. After 3:40 pm, the PV panel did not generate any power.

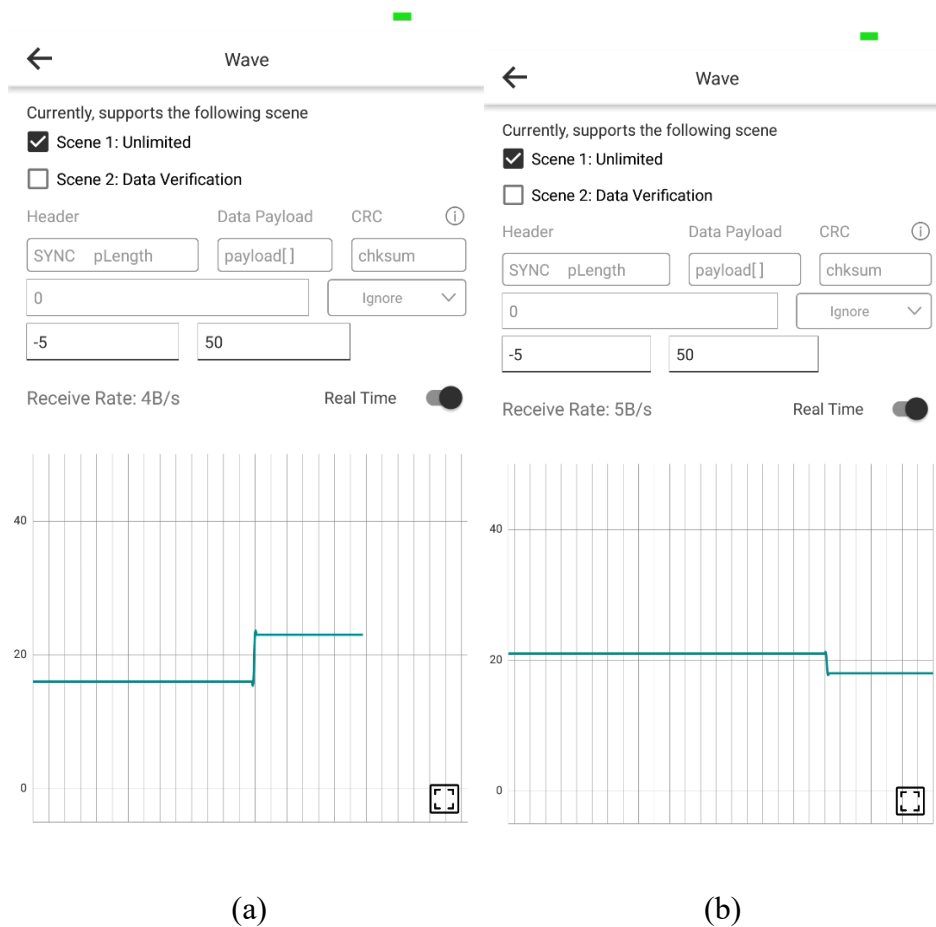
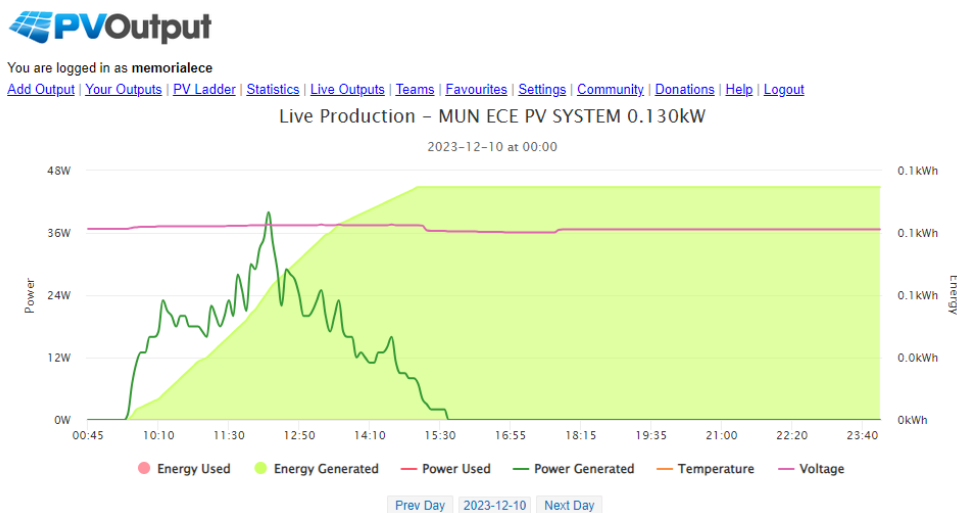


Figure 3-5 BTed live recording of the PV panel output power. (a) Increased power (b) Decreased power.



Meanwhile, the generated energy stopped growing at the same time. The voltage of backup battery also had variations during the day. From 8:15 am to 12:15 pm, the voltage increased slowly from 12.8 V to 13.6 V. At 3:15 pm, at which a 12 V /50 W bulb was turned on, the battery voltage dropped to 12.5 V from 13.4 V immediately. Decreasing steadily, it was 12.1 V at 5:45 pm after 2.5 hours of lighting the bulb. The lighting bulb was turned off thereafter, bringing the voltage back to 12.6 V instantly and finally 12.7 V at the steady state. Data from 14 days ago were automatically reduced to limited parameters as Table 3 lists. All data in Figure 3-6 are stored in PVOutput.org and can be accessed free of charge after a simple registration process on the website, by searching “MUN ECE PV SYSTEM 130W” within Teams in PVOutput.org and choosing December 10<sup>th</sup>, 2023 as the date.



*Figure 3-6 PV panel output power on Dec. 10, 2023, recorded in PVOutput.org.*

The battery voltage discharged at a different rate on Dec. 10 and Dec. 12, 2023. During the a few hours near noon, the output power of the PV panel was greater than that of the afternoon on Dec. 10. Table 3-5 lists the time when the load was turned on and off, the respective period, the corresponding generated energy, and the battery voltage. Both starting from 12.5 V and ending

with 12.1 V, it took 255 minutes for the discharging process on Dec. 12, while 150 minutes for Dec. 10. Energy generated during each period was 0.122 kWh and less than 0.001 kWh.

*Table 3-5 Change of the battery voltage during two days with different PV energy generation.*

	Load turned ON	Load turned OFF	Energy generated by the PV panel during the period (kWh)	Period (minutes)
Time	11:10, Dec-12	15:25, Dec-12	/	255
Voltage (V)	12.5	12.1	0.122	/
Time	15:15, Dec-10	17:45, Dec-10	/	150
Voltage (V)	12.5	12.1	< 0.001	/

### 3.4 Discussion

Upon the BLE connection was built, the output power from the PV panel was plotted in BTeD interface. Within 55 seconds, any trend of the output power was intuitively observed by this method. The upward and downward change of the output power were easily observed in Figure 3-5. Thus, the effectiveness of BTeD as the HMI design in the developed SCADA system was verified.

From the power plot recorded in PVOutput.org, the peak power happened around the noon on Dec. 10, 2023, which coordinated with the common solar irradiation pattern. Since it was a day during winter in Canada, the period when solar power was available was relatively short: only between 8:15 am and 3:10 pm, there was output power from the PV panel. At 3:15 pm when the light bulb was turned on, the battery voltage immediately dropped 0.9 V from 13.4 V. This is due to the voltage drop by the battery internal resistance with the increased load. The battery steadily decreased overtime when the bulb was on was due to the discharge of the battery. Since the PV panel output power was not comparable to the power consumed by the bulb (50 W), the battery voltage declined at the rate of 0.1 V every 20 minutes. After the bulb was disconnected with the

circuit, the battery voltage rose to 12.6 V, and finally 12.7 V during the steady state. During Dec. 12, 2023 noon when the output power from the PV panel was larger, the decreasing rate of the battery was 0.4 (=12.5-12.1) V in 255 minutes, which was slower than 0.4 V in 150 minutes on Dec. 10, 2023. This is attributable to the PV panel which was also charging the bulb, leading to lesser power consumed from the battery.

The power consumption of the components is listed in Table 3-6. Nordic Semiconductor® Power Profiler Kit II was used to measure the power consumption of Arduino® UNO R4 Wi-Fi with and without establishing Wi-Fi connection. The power consumption of INA3221 voltage module is from its datasheet. The overall power consumption is therefore between 100.35 mA and 120.35 mA, which is energy efficient compared to commercial products of which the power consumption is at watt level [29-31]. The cost breakdown of the proposed SCADA system monitoring standalone PV systems is also presented in Table 6. In [56-58], the system costs are C\$ 107, C\$ 210, and C\$ 760, respectively. They are less cost-effective than the proposed SCADA system in this section, which costs only C\$ 42.15. The usage of PVOutput.org and BTeD application are free of charge.

*Table 3-6 Power consumption and price breakdown of the proposed system.*

Components	Current Consumption	Unit Price (C\$)	QTY
Arduino® UNO R4 Wi-Fi	~120 mA (with Wi-Fi connection) ~100 mA (without Wi-Fi connection)	36.99	1

INA3221			
Voltage Monitor	~350 $\mu$ A	5.16	1
Overall	100.35 mA – 120.35 mA	<b>C\$ 42.15</b>	

### 3.5 Conclusion

In this chapter, a new design of HMI and a data storage solution featuring remote, extensive, and low-cost was proposed in the SCADA system for monitoring standalone PV systems. A PV system and its SCADA system were built to verify the effectiveness of this design. The PV system comprised a PV panel, a solar charging controller, a backup battery, and a bulb. The SCADA system consisted of the Arduino UNO R4 Wi-Fi as the RTU, the BTeD as the HMI, the PVOutput.org as the data historian, an INA3221 voltage monitor as the sensor, and a PC as the MTU. By using BTeD, the output power from the PV panel was successfully displayed in the application interface via the BLE connection between the mobile and the RTU. A 55 seconds was the maximum time period to allow the data to be presented at the same screen. The remote and extensive data storage was achieved by PVOutput.org. The PV panel output power and the battery voltage were uploaded to this website via Wi-Fi every five minutes. Data before 14 days were automatically simplified by the website into daily generated energy, peak power, peak time, etc. The recorded solar power matched the solar radiation pattern, and the recorded battery voltage varied accordingly with the load status and the PV output power. This novel design of the SCADA system was revealed to be effective on monitoring the standalone PV system.

### **3.6 Co-authorship Statement**

This chapter has been published in Journal of Electronics and Electrical Engineering (JEEE, ISSN: 2972-3280) by Wei He and Dr. Mohammad Tariq Iqbal. Dr. Mohammad Tariq Iqbal is the academic supervisor of Wei He in his graduate program at Memorial University. The link is <https://doi.org/10.37256/jeee.3120244132>.

All authors contributed significantly to the work presented in this section. Their individual contributions are outlined below.

Wei He: Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, and Visualization.

Mohammad Tariq Iqbal: Conceptualization, Resources, Writing - Review & Editing, Supervision, Project Administration, and Funding Acquisition.

# **Chapter 4. An Open-Source SCADA Architecture for Photovoltaic System Monitoring using ESP32, Banana Pi M4 and Node-RED**

## **4.1 Introduction**

To monitor and control industrial process in different sectors, such as power generation systems, agriculture [59], and biogas [60], Supervisory Control and Data Acquisition (SCADA) systems are developed. Explicitly in the case of photovoltaic (PV) systems, the SCADA systems can optimize PV system performance [61]. 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 SCADA system to collect data that is critical to be used in industrial production, such as current, pressure, temperature, voltage. FIDs and RTUs can communicate over wireless or wired communication protocol 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 central location and serve the purpose of data communication, data storage and hosts the main server.

The architecture of 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). The RTUs and the MTU, if not manufactured by the same vendor, often cannot communicate with each other due to the

proprietary communication protocols. Distributed SCADA stands for the SCADA system that uses local area network (LAN) to achieve communication between each operation station and uses WANs between the RTUs and the corresponding operation stations [62]. This architecture facilitates system miniaturization by assigning each operation station different functions. For example, an operation station that solely exchange data with the RTUs or provides the HMI to the operator can be made smaller. [63] 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 [64]. Unlike networked SCADA, more diverse devices can be utilized in IoT SCADA, such as smart sensors. Moreover, cloud services including storage, servers, software, computing can be integrated in IoT SCADA. Applications of IoT in industry context including digital twins and prediction of system failure have advanced the industrial automation to a large extent [65]. The conventional SCADA systems every so often offers high costs and complexity [66] leaving a challenge for small scale and cost-sensitive PV systems. Besides, scalable, and flexible SCADA systems are gaining attention in response to the development of 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 solution can be proposed by using IoT architecture. The adoption of Internet of Things (IoT) in SCADA systems utilize connected devices and sensors via Internet Protocol (IP) networks to collect and analyze data, offering improved flexibility, scalability, and cost-effectiveness [67]. As a result of 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 system.

Message Queuing Telemetry Transport (MQTT) is a communication protocol running over TCP/IP. Based on publish/subscribe mechanism, MQTT is a lightweight, open, and easy-to-implement solution. The MQTT broker separates the publisher from the subscriber, routing the message by their topic [68]. This architecture encourages the integration of low-bandwidth devices in IoT context where numerous RTUs and MTUs with limited processing power can be present [69]. Prevalent in 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 the monitored variables [70]. 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 IoT for monitoring a PV system has been proposed. Voltage sensors and current sensors are specified as F031-06 0-25 V voltage sensor module and ACS712 current sensor module. Monitored voltages and currents are first collected by these analog sensors and the connected RTU ESP32 module, and then data is 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 MQTT protocol between the RTU and the MTU requires minimum bandwidth while ensuring reliable and real-time communication within the SCADA system. The design SCADA system incorporates the low-cost components such as FIDs, RTUs and MTU make the proposed cost-effective as a whole. To add to this, the proposed SCADA system is experimentally validated which ensures the reliability of this work.



Following is the outline of the rest of the section. A review of related literature is presented in Section 4.2. In Section 4.3 and 4.4, the system description and the components used throughout the course of this study are introduced. In Section 4.5, the implementation methodology is covered. In Section 4.6, the experimental setup and results are demonstrated. Section 4.7 highlights the discussion part of this section. The conclusion of this section is provided in Section 4.8. Co-authorship statement comprises Section 4.9.

## **4.2 Related Works**

The authors of [71] 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 Arduino UNO R3. Also equipped with a SIM800L module, the global system for mobile communication (GSM) connectivity is added to the Arduino UNO R3. The Raspberry Pi 2 works as a server hosting Node-RED platform and the data storage unit collecting 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 on/off the pump over SMS and query the historic data from the Raspberry Pi 2 using the MQTT protocol via Ethernet. This research utilizes GSM to achieve remote monitoring as a part of their work. In [72], an IoT-based wireless data acquisition and control system for PV modules was introduced. For the hardware setup, the authors use three NodeMCU Wi-Fi modules flashed with Tasmota firmware and that are responsible for collecting sensor readings, servo motor control, and controlling relays for

ON/OFF control, respectively. Sensors include a voltage and a current sensor, a luminosity sensor, an ambient temperature, and humidity sensor, and four waterproof temperature sensors. They are all mounted on one PCB. Another hardware component is a Raspberry Pi running 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 is validated in terms of data accuracy, latency times in communications, CPU computational and thermal performance benchmark, and sampling rate. The sole focus of this study was PV module performance under open circuit condition, not considering the realistic performance impact from other common parts such as charging controller, the battery, and the load.

The researchers in [73] developed a novel system to track 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 a MCU ATmega328P, a RF transceiver ATmega128RFA1, an external ADC, a battery, and a buck-boost converter. A coordinator node is 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 computer vision algorithm requires the input voltage, 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 be also generated and sent to the WSN by the coordinator node. The Raspberry Pi runs the computer 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 the PV panel and the backup battery was designed in [74]. 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 Thingier.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 Thingier.IO server to facilitate the monitoring and supervisory control. Two system configurations are tested, where the Raspberry Pi is connected to either Ethernet or the LAN port of a local Wi-Fi router, with the latter being 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 [75], an IoT platform for online monitoring of renewable energy systems is developed. A grid-connected PV station in Saharan environment is taken as the case study to validate the effectiveness of the proposed system. LA-25NP current sensor and 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 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 [76] to remotely monitor a solar plant. The computing module of 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. A 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 [77] proposes an IoT based system for monitoring of PV fuel cell system. The DC power output from the combination of PV array and the fuel cell were monitored by sampling the voltage and current. ATmega2560 as the RTU first collected the monitored voltage and current, and then sent to the NodeMCU ESP8266 module through serial communication. Finally, 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 [78], 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 [79], the authors present a framework for developing 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 the industrial control systems is presented in [80]. 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 as power, telecommunication, transportation, and manufacturing plants, SCADA systems provide monitoring and control. The authors of [81] believe that the SCADA system operating in connection of internet is vulnerable as compared to the 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 [82] use SCADA to connect the lower central controller to the 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 researchers in [83,84] developed a monitoring and control system for renewable energy generation in context of Peer-to-Peer (P2P) energy trading. The proposed system in their study is hosted on a private network, while they used ESP32 as an IoT server. The authors of this study believe that their system is low-cost and low power.

Authors of [85] 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 MQTT protocol was used for communication between Node-RED and two MQTT clients (MQTT.fx in Windows OS and MQTTClient in iOS). Experiment 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; 2) the two MQTT clients can successfully publish to and subscribe from the Node-RED dashboard via MQTT. This paper contributed to the

conceptualization of applying IoT in SCADA system, providing a framework for future practical scenarios. A Node-RED based SCADA architecture for a hybrid power system was implemented in [56]. One voltage sensor and five current sensors were collecting the hybrid power system parameters, sending to Node-RED platform running on Windows OS via a USB cable from Arduino Mega2560. A Wio terminal was also utilized to display the monitored parameter. 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 the best of authors knowledge no such SCADA system has been identified that uses Banana Pi M4 Berry to host Node-RED based IoT server and uses 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 the 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 commercially available components utilized in the system design facilitate 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 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 the over discharge protection of the battery.

### **4.3 System Descriptions**

An illustration of the proposed SCADA system can be found in this section of the section. Figure 4-1 provides an overview of the SCADA system design presented in the chapter. An ESP32-E serves as RTU in the system design, on the other hand Banana Pi M4 Berry acts as the MTU. In the design SCADA system ESP-32-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 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 ESP32-E and stores the data received using FIDs on the BPI-M4 Berry's local storage, which is 8G eMMC flash. It is also accessible for data visualization over UI by browsing localhost.

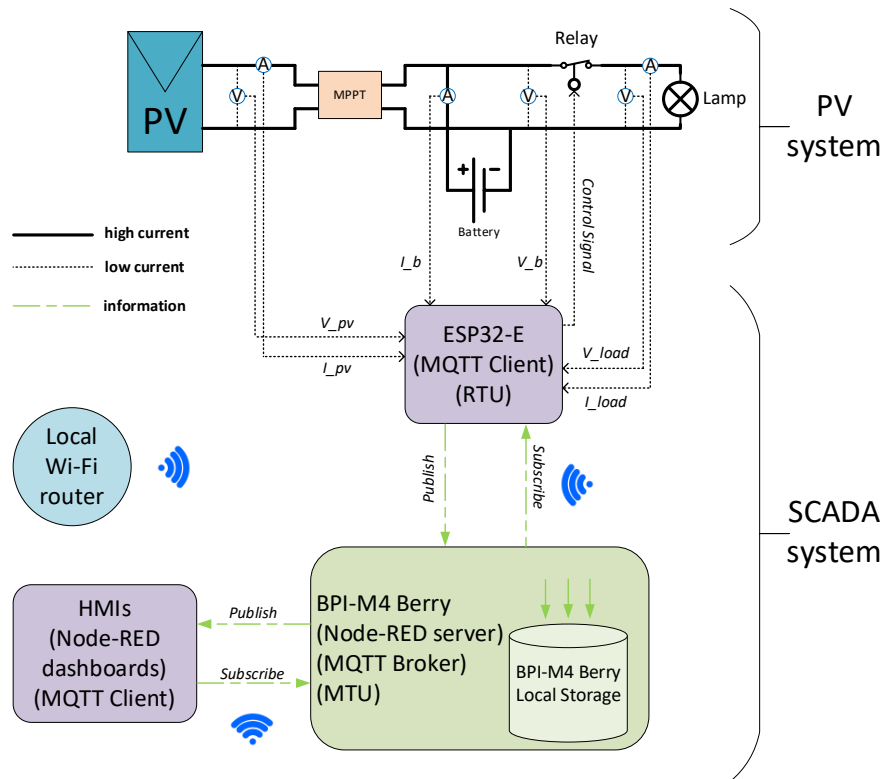


Figure 4-1 Overview of proposed SCADA Design.

## 4.4 SCADA System Components

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

### 4.4.1 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 4-2 shows the physical representation of BPI-M4 Berry. Compared to the well-regarded Raspberry Pi 4b, BPI-M4 Berry is more powerful and



commonly available at a price much lower than that. BPI-M4 Berry has full HDMI connector and M.2 PCIe 2.0 slot 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. Banana Pi M4 Berry also offers a Gigabit Ethernet (GbE) RJ45 connector, a 40-pin header, and four USB ports. The specifications of BPI-M4 Berry are listed in [86]. With its features, the BPI-M4 Berry board supports a broad range of applications including media processing, the IoT, and entertainment sectors. As a part of the proposed SCADA system design, we used 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 collect sensor readings locally.

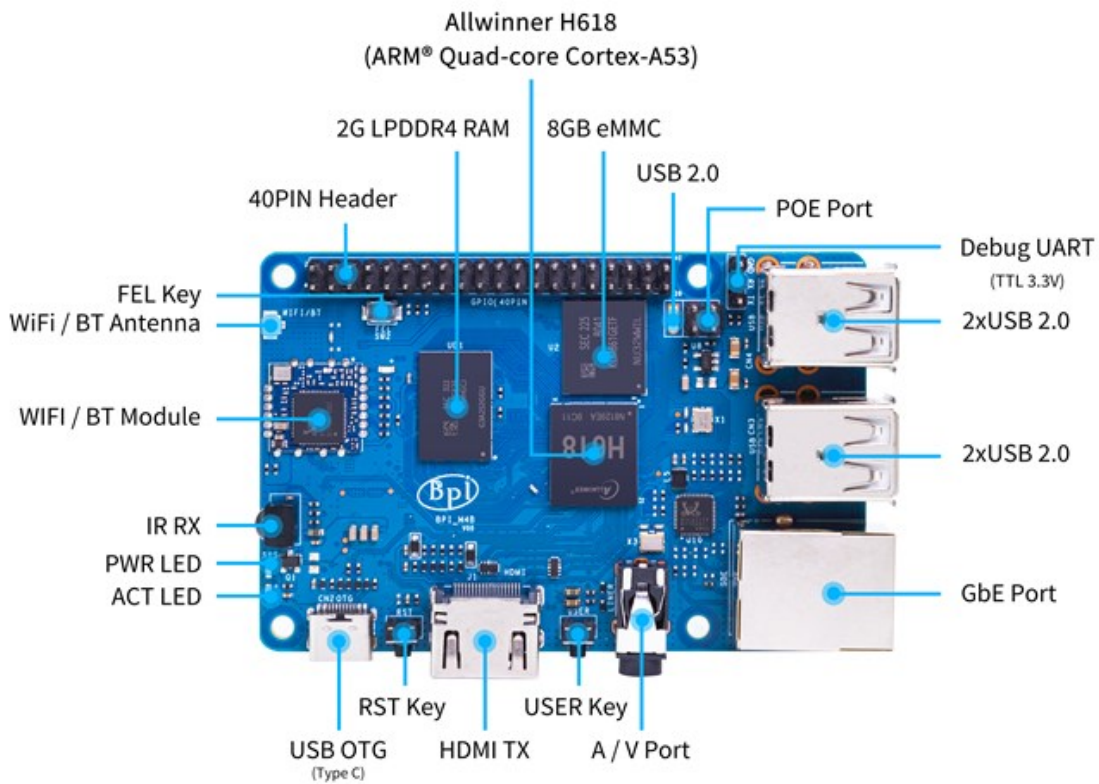


Figure 4-2 BPI-M4 Berry.

#### **4.4.2 ESP32-E**

In the ESP32 series, the FireBeetle 2 ESP32-E is a competent micro-controller board using ESP-WROOM-32E module, featuring the dual-core processor architecture, and Wi-Fi and Bluetooth communications [35]. Its compact size, high energy efficiency, and inclusion of an integrated charging circuit are advantages for multiple applications such as smart home and IIoT solutions. The significant factors contributing to its selection for this research are the capability of wireless communication, the availability of analog pins, low cost, availability in the lab, and resourceful open-source tutorials. Figure 4-3 presents the pinout of FireBeetle 2 ESP32-E.

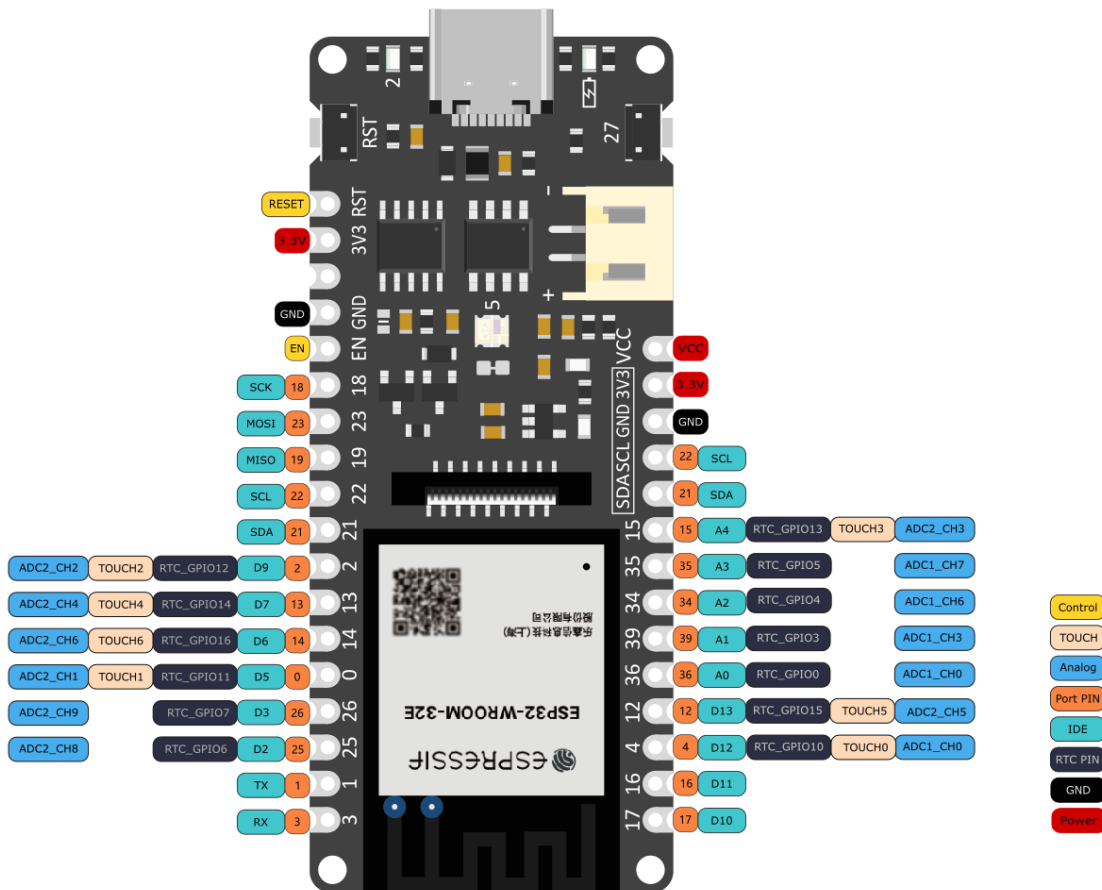


Figure 4-3 Pinout of the FireBeetle 2 ESP32-E

#### 4.4.3 Node-RED

Node-RED epitomizes a paradigm of flow-based programming, a concept emphasizes a methodology for delineating an application's functionality through a network of encapsulated units, or "nodes" [87]. Figure 4-4 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 this data, and then forwarding the

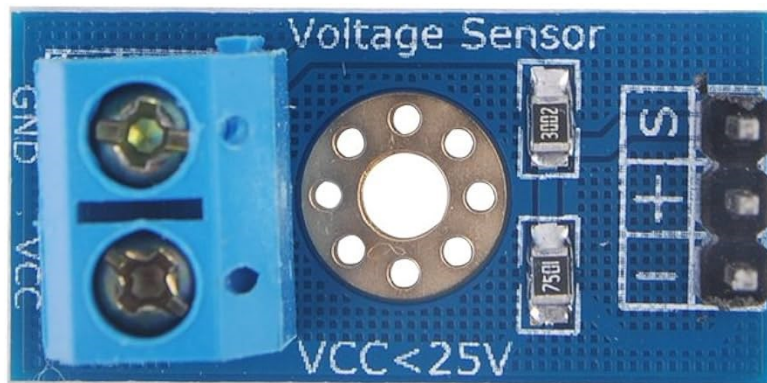
processed data onwards. This architecture not only facilitates the consistent flow of data across the network but enhances accessibility and comprehension to more customers. The inherent design of Node-RED, with its emphasis on visual representation, significantly lowers the barrier to entry for users across different skill levels. Through the course of this study, we used Node-RED for easy implementation of MQTT protocol by Mosquitto Broker to transfer data and creating dashboards that enable real-time monitoring and control.

The image shows the configuration interface for a new MQTT broker node in Node-RED. The title bar reads "Edit mqtt in node > Add new mqtt-broker config node". At the top right, there are "Cancel" and "Add" buttons. Below the title bar is a "Properties" section with a gear icon and a document icon. The main configuration area is divided into three tabs: "Connection" (selected), "Security", and "Messages". Under the "Connection" tab, there are several fields and options: "Name" (text input), "Server" (text input with "localhost" selected), "Port" (text input with "1883" selected), "Enable secure (SSL/TLS) connection" (checkbox, unchecked), "Client ID" (text input with "Leave blank for auto generated" selected), "Keep alive time (s)" (text input with "60" selected), "Use clean session" (checkbox, checked), and "Use legacy MQTT 3.1 support" (checkbox, checked). At the bottom, there is a status bar with "Enabled" (radio button), "0 nodes use this config" (info icon), and "On all flows" (dropdown menu).

Figure 4-4 The Node-RED MQTT node configuration interface.

#### 4.4.4 Voltage Sensor

The HiLetgo voltage detection module, capable of measuring DC voltages from 0 to 25V, develops a resistive voltage divider strategy for its operational methodology. Figure 4-5 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 $\Omega$  and 7.5 k $\Omega$  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 $\Omega$  to the cumulative resistance (7.5 k $\Omega$  + 30 k $\Omega$ ) results in a fraction of 1/5, 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 [74]. We used voltage sensors in this section to measure the voltages within the PV system.

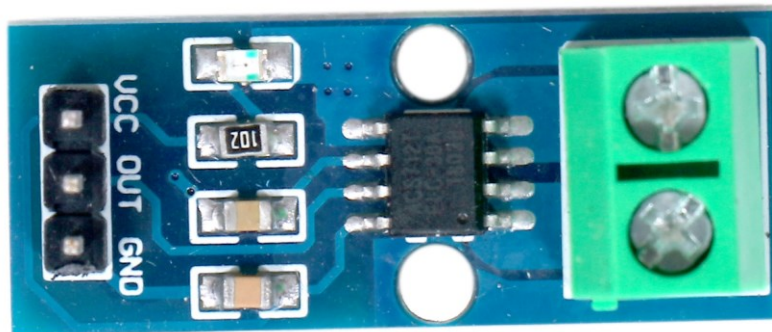


*Figure 4-5 0-25 V voltage sensor illustration.*

#### 4.4.5 Current Sensor

Based on the Hall effect, the ACS712 as a cost-effective and precise current sensing solution, is suitable for a broad spectrum of applications within both industrial and commercial sectors.

Figure 4-6 provides a visual representation of the ACS712 current sensor, which is capable of accommodating currents up to  $\pm 20$  A. For implementations involving the ESP32—characterized by a nominal voltage of 3.3 V—it is worth noticing that the VCC and ground pin of the ACS712 sensor interface with an voltage source supplying 5 V 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 output pin with an one of the analog pins of ESP32, a voltage divider is recommended to mitigate the risk of exposing the ESP32 to voltages exceeding 3.3 V [88]. The ACS712 sensor is available in three variants, differentiated by their optimized current measurement ranges of  $\pm 5$  A,  $\pm 20$  A, and  $\pm 30$  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 section, we used the ACS712 current sensors with  $\pm 30$  A to measure the current in the PV system.



*Figure 4-6 Icon of ACS 712 current sensor.*

## 4.5 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 VCC pin outputs about 4.7 volts that meets the voltage requirement of 4.5 – 5.5 volts by the current sensor. Table 4-1 presents the pin configuration of RTU.

*Table 4-1 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 4-7 depicts the system flow chart. After the ESP32-E collects the sensor data, they are transmitted to the Node-RED platform configured on BPI-M4 Berry using 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 8G flash. Algorithm 1 represents the pseudocode programmed in ESP32-E using Arduino IDE software.

---

**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 device 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 seconds;
- Else**

11. Go to Step 1;
- End**
-



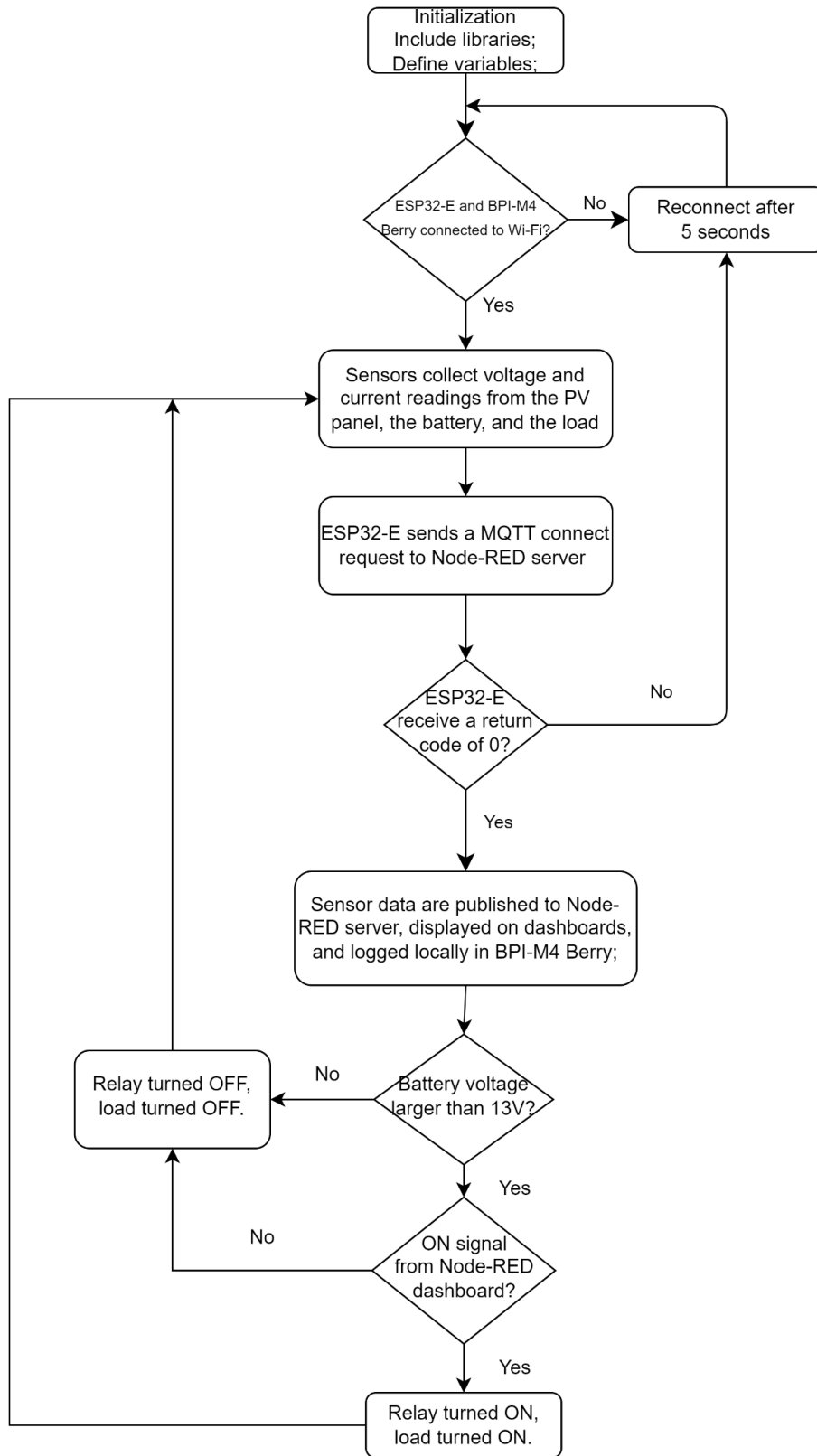


Figure 4-7 System flowchart.

## 4.6 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 PV lab in Memorial University. The hardware connections were built based on the system description in Chapter 4.3. Figure 4-8 depicts the experimental setup of the proposed SCADA system. The hardware configuration is illustrated in Figure 4-9, where CS stands for current sensor and VS stands for voltage sensor.

Figure 4-10 represents PV installation that 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.

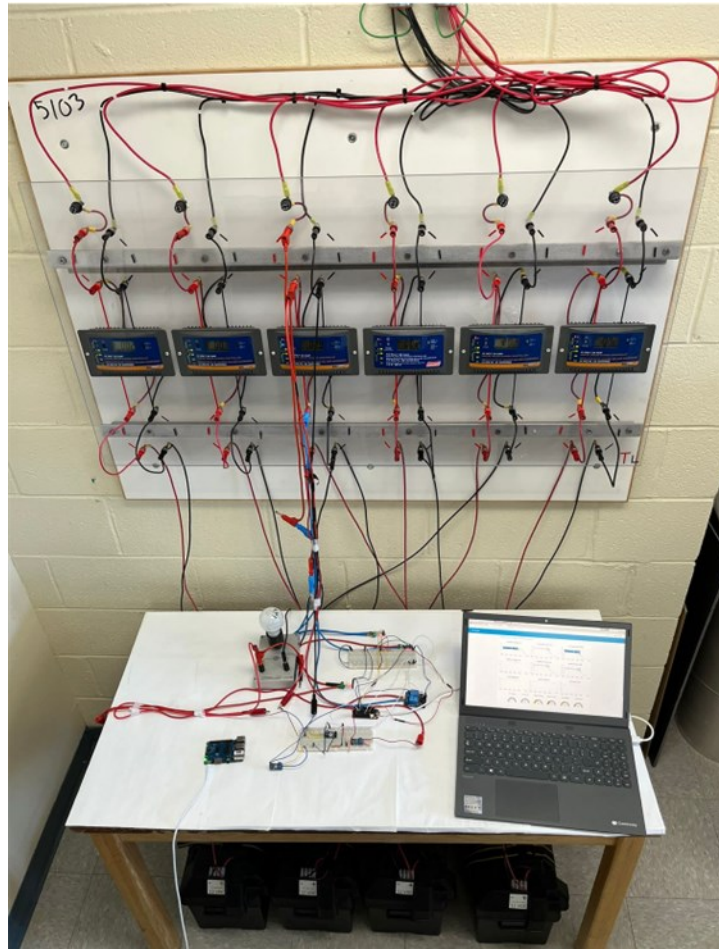


Figure 4-8 Experimental setup at MUN ECE laboratory.

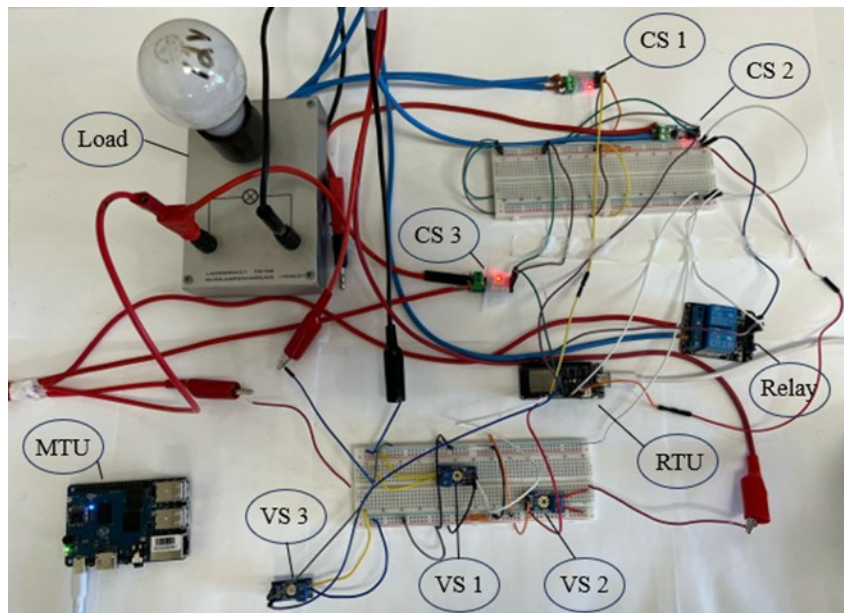


Figure 4-9 Hardware implementation of the proposed SCADA system.



*Figure 4-10 MUN ECE laboratory PV installation.*

Node-RED (MQTT Broker) that 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 ESP-32 E microcontroller (MQTT Client). Using MQTT protocol, the data is transmitted in real time. In Node-RED platform, the node flow as shown in Figure 4-11 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. Banana Pi Debian 1.0.0 OS was installed into the eMMC of 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 used memory was around 870 MB, being 43.8% of total available memory. The hardware resources required in this setup were notably less than the total available resources, ensuring the safe and continuous system operation. Due to the volatile

weather nature in St. John’s, data vibrations are observed during the test of the proposed SCADA system. To validate the accuracy of measured data, six digital multimeters are used to measure the PV system variables. Measurement results by the proposed SCADA system align with that of digital multimeters, showcasing that our system accurately measures the PV system variables.

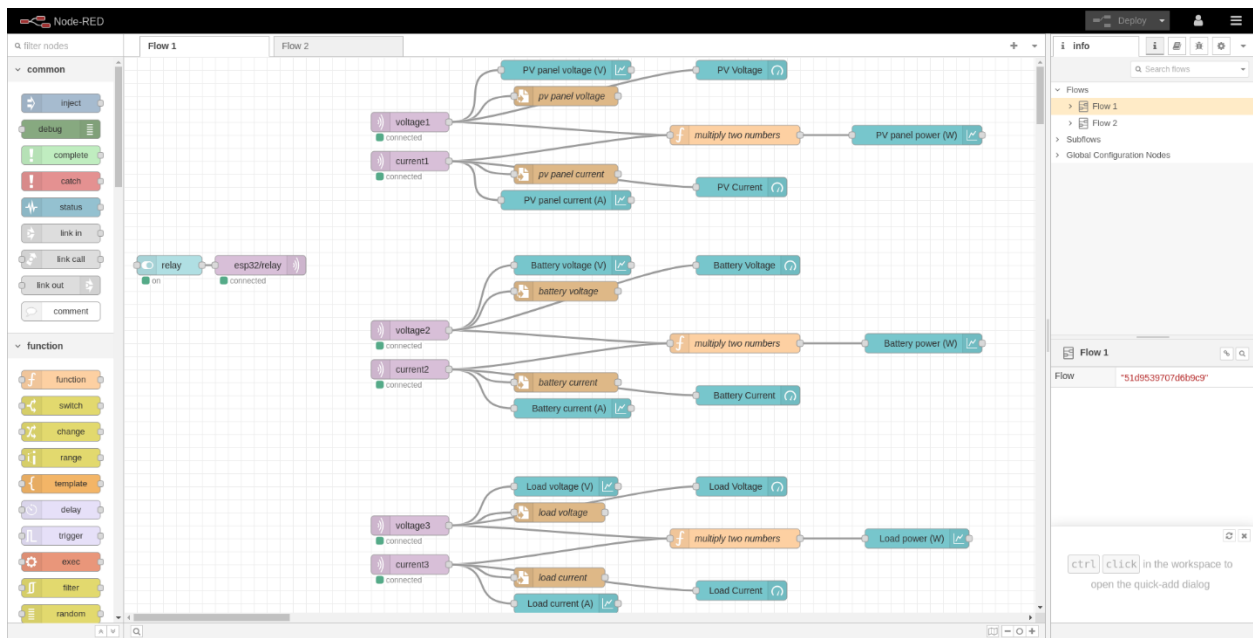


Figure 4-11 The developed Node-RED flow.

Figure 4-12 is the screenshot of the Node-RED dashboard showing the logged PV system data during a day. Starting from 10:12 AM, 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 center of 4.88 A. The load power ranged from 67.38 W to 72.07 W. After 17:07 PM, 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 PM. The PV panel power was 7.38 W at 18:24 PM. 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 PM. At 20:11 PM, the PV voltage and the PV power was 0. The battery voltage and the load voltage were both 12.28 V. The battery current and power was -4.22 A and -51.82 W. The load power was solely from the battery.



Figure 4-12 Node-RED dashboards showing the monitored values of the PV system.

The solar eclipse of April 8, 2024, was a total solar eclipse visible to places from Canada, United States, and Mexico. In St. John's, NL, Canada, we have monitored the solar PV panel voltage during this eclipse as shown in Figure 4-13. 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 St. John's area. This also validates the system performance. Figure 4-14 is the PV output current during the eclipse. Similarly, the PV current dropped dramatically around 17:16, while it remained at near zero level longer than the PV voltage.

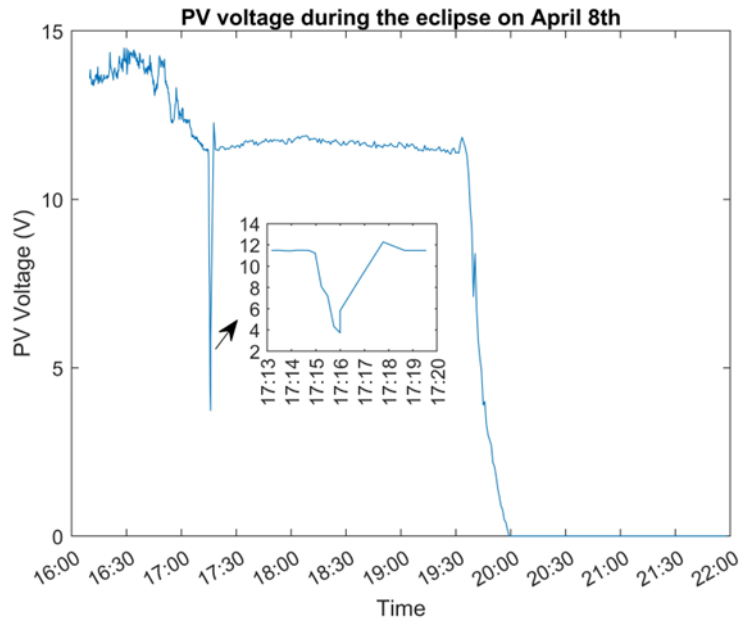


Figure 4-13 PV output voltage during the total solar eclipse on April 8, 2024.

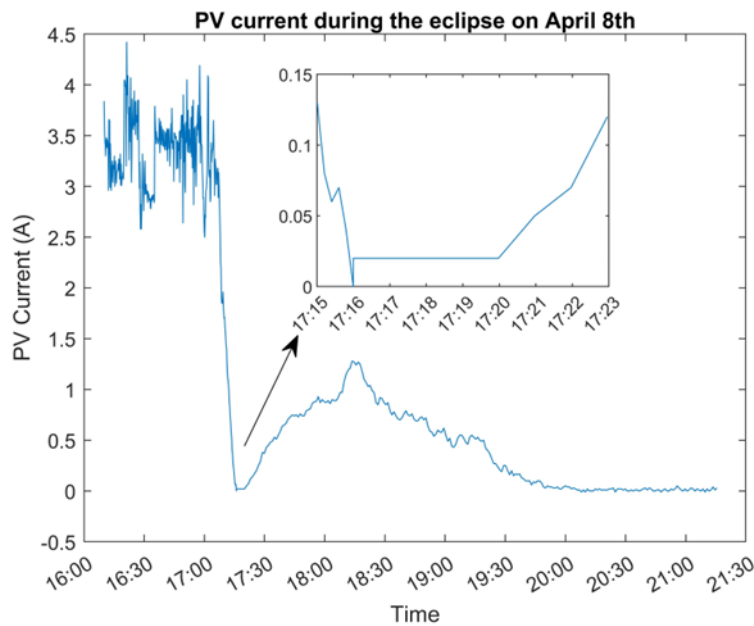


Figure 4-14 PV output current during the total solar eclipse on April 8, 2024.

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 4-15, the switch node was set to be OFF at 21:25 PM, 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 4-16 presents the experimental results for local control. Despite the switch node is 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.

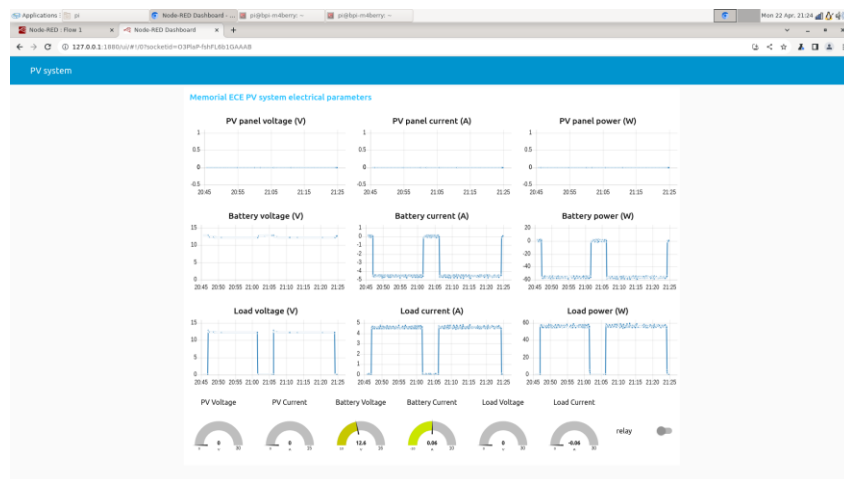


Figure 4-15 Test results for supervisory control.



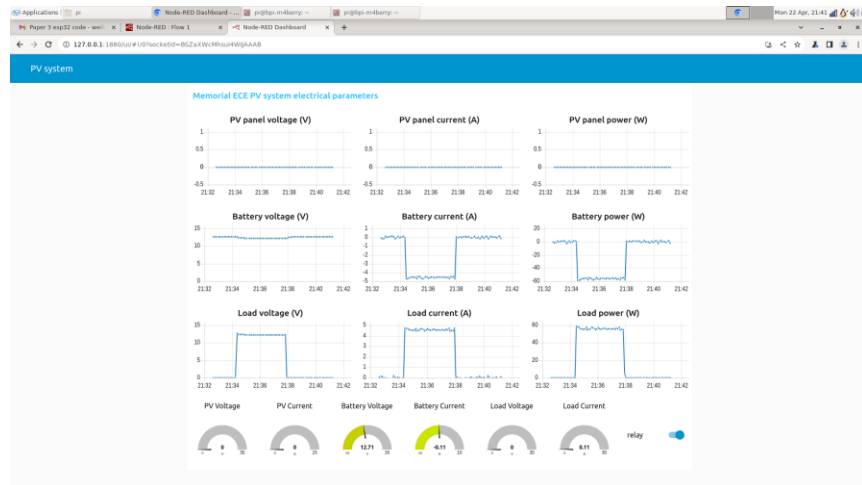


Figure 4-16 Test results for local control.

## 4.7 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, RTU based on ESP32-E, MTU configured over Bana PI M4, Node-RED and MQTT protocol over a local network.
- **User Interface (UI):** The Node-RED platform is used in the development of SCADA architecture that 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 physical position of both the RTU and the MTU.
- **Data Storage:** The proposed SCADA system features local storage based on 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 commonly available FIDs and the MTU in the local market, all at affordable prices. Table 4-2 breaks down the power consumption and the prices of the system components. The total system costs only 94.5 CAD. 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 4-2 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
5VDC - 250V 10A relay module	1	5.5	0.45
Total		94.5	3.19

## 4.8 Conclusion

Crucial operational facilities are distributed across vast areas, 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 they have 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 section, based on a comprehensive arrangement of FIDs, an RTU and 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 April 8, 2024, in addition to regular validation of the system performance in real time, which also testifies 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 - however also demonstrates versatility and precision in handling real-time data in robust environments.

## **4.9 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 section is single string and low power. The current research work can also be extended for multi string larger PV system. In addition, data storage can be considered to implement 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., Li-ion batteries-based PV system including interaction with battery bank BMS or hybrid system with grid connected inverter, AC load and higher DC bus voltage. Additional security features, such as users accounts, passwords and local data logging in server can be added in Node-RED. Using additional measurements of PV system, system faults diagnosis can be added in Node-RED.

## 4.10 Co-authorship Statement

This chapter has been published in MDPI Energies (ISSN: 1996-1073) by Wei He, Dr. Mirza Jabbar Aziz Baig, and Dr. Mohammad Tariq Iqbal. Dr. Mohammad Tariq Iqbal is the academic supervisor of Wei He in his master's program. Dr. Mirza Jabbar Aziz Baig was also supervised by Dr. Mohammad Tariq Iqbal in his Ph.D. program at Memorial University. The link is <https://doi.org/10.3390/en17102295>.

All authors contributed significantly to the work presented in this section. Their individual contributions are outlined below.

W.H.: system design, experimental results, and writing of the manuscript.

M.J.A.B.: co-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.

# **Chapter 5. An IoT-SCADA Architecture for Photovoltaic System Monitoring, Control, and Inspection in Real Time**

## **5.1 Introduction**

The IoT is a technology that integrates objects with the internet, enabling them to collect, exchange, and act upon data. This interconnected network of devices—ranging from household appliances [89] and wearable technology [90] to industrial machinery [91] [92] and smart city infrastructure [93] —facilitates seamless communication and automation. By leveraging advanced sensors, actuators, and connectivity technologies, IoT enhances efficiency, improves decision-making, and fosters innovation across various sectors. The transformative potential of IoT is the ability to create intelligent systems that optimize operations, enhance user experiences, and drive significant economic growth.

SCADA refers to a control system architecture that utilizes computers, communication networks, graphical user interfaces to supervise machines and processes at a high level. A few taxonomies of SCADA components should be briefly introduced. The RTU collects data and information from sensors, and then forwards them to the MTU. The HMI is responsible for visually presenting the data to human operators. Communication networks can be wired or wireless, providing communication services across the other SCADA system components.

During the evolution of SCADA systems, there are typically four stages: monolithic, distributed, networked, and IoT-based [62,63]. Using the proprietary communication protocol (wide area

network, WAN), MTU and RTU must be from the same vendor to communicate with each other. Similarly in distributed SCADA, the communication server uses WAN to communicate with RTUs. However, local area networks (LANs) bridge the gap between the communication server and the multiple operation stations. Furthermore, in networked SCADA, internet protocol (IP) replaces the proprietary protocols, allowing the SCADA system components to be from different vendors and separated geographically. Finally, IoT-based SCADA utilizes cloud computing, trying to extract insightful information from the collected data. Due to the advantages of flexibility, easy development, cost efficiency, and scalability, IoT-based SCADA systems are becoming popular.

However, new vulnerabilities emerge when IoT is integrated into the traditional SCADA systems. Management of large-scale systems and cross-layer collaborations [94], significant required storage space, the data security, along with high power consumption [65], big data analytics [81] are challenges to be overcome. Furthermore, system metrics of great importance such as bandwidth overload and latency rely on the cloud service providers [62]. Production loss could be made if the cloud service is down.

In this study, we proposed an open-source and low power consumption IoT-SCADA system to monitor the PV installations at the PV lab at MUN. Two voltage sensors collect the PV module voltage and the battery voltage data, while three current sensors are responsible for fetching the PV module current, the battery current, and the lamp load current. ESP32-E assembles the voltage and current data, forwarding to ESP32-S3 via HTTP and the Arduino Cloud platform via MQTT. Furthermore, ESP32-E controls the ON/OFF state of the relay, turning ON/OFF the load based on the battery voltage. In addition, ESP32-S3 transmits the received data to the

ThingSpeak platform via HTTP and sends load images to the web server it built. The effectiveness of the proposed IoT-SCADA system was verified through experiment results.

The rest of the chapter is organized as follows. Related works are reviewed in Chapter 5.2. Chapter 5.3 describes the overview of the system. Chapter 5.4, 5.5, and 5.6 present the system components, implementation methodology, and the experimental setup. In Chapter 5.7, the experimental results are demonstrated and analyzed. Discussions of the designed system are covered in Chapter 5.8. Chapter 5.9 concludes this total chapter. Co-authorship statement comprises Chapter 5.10.

## **5.2 Related Work**

In [95] a IoT-based SCADA system for supervising two connected microgrids was designed and implemented. The power sources of the load were changed among local microgrid, neighbor microgrid, national power line, and the diesel generator, regarding the energy production situation of the microgrids. ESP32 microcontroller collected voltage, current, and temperature data from the corresponding sensors via I<sup>2</sup>C, and sent them to Cayenne Web Server cloud via MQTT protocol over Wi-Fi connectivity. A dashboard showing the monitored system values and the power source was created by Widgets application. The authors claimed that an efficient and reliable power sharing can be achieved under power shortages.

In [96] the authors developed a new approach using an improved artificial bee colony (IABC) algorithm based on IoT based SCADA to achieve MPPT for a solar power plant. The duty ratio of the DC-DC converter was determined by the inputs of PV panel voltage and current using



IABC. After the initialization of random duty ratio values, each employed bee evaluated the fitness and searched nearby. Onlook bees then searched for the neighborhood of the positions of employed bees to find a higher fitness value. The employed bees became scout bees to find new solutions if the fitness cannot increase. Smart sensors published PV voltage, PV power, and the duty ratio to ThingSpeak platform organized by a microcontroller, via MQTT. Subscribers of the topics can view the values on mobile devices and laptops. The authors claimed that the steady-state restoration times are improved by 89%, 87%, and 88% for the system's power, voltage, and duty cycle, better than incremental conductance and cuckoo search method.

An IoT open-source platform for monitoring a 3 MW PV plant was proposed in [97]. A personal computer with Eclipse Kura installed was the IoT Gateway, while an Eclipse Kapua instance received the current and voltage data from the industrial devices via MQTT. Several integrations can easily extract the received data, such as Elasticsearch, MQTT broker, Kapua API, Kura API, and camel routes. The authors claimed that the interoperability of the proposed solution was proved by the adaptability of industrial protocols (Modbus, OPC, S7) and communication systems (Ethernet, Wi-Fi, Bluetooth).

A SCADA system for a PV plant based on open-source software was designed in [98]. IoT-ready inverters sent the electrical parameters via Modbus TCP/IP, while gauges that are not IoT-ready sent environmental parameters via Modbus RS-485, both to the communication server. The communication server then processed and sent the data via MQTT protocol to the application and database server which run Emoncms dashboards. The authors claimed that the developed SCADA system using IoT devices achieved the easy customization of the existing supervisory system and increased the polling frequency.

A low-cost real-time PV monitoring system was developed using IoT architecture in [99]. Two data loggers both based on ESP32 microcontroller were implemented: the first one collected meteorological data and the second collected PV generation data. The received transducer readings were first sent to the LoRa gateway (ESP32) via LoRa, then sent via MQTT to the Google Cloud Platform (GCP) MQTT broker. Hosted on Heroku, two webpages were built as the subscribers to display the real-time monitoring meteorological data and the PV generation history data, respectively. The authors claimed that some of the advantages of the proposed monitoring system were complete meteorological measurements, wireless communication, and data stored both locally and in the cloud.

Authors of [100] used a SCADA system to acquire operating parameters of a PV plant, to calculate its quality indicators for energy management. Meteorological parameters including solar irradiation, ambient temperature, PV module temperature and wind direction were collected by the weather station. Electrical parameters including power, current, and voltage of each array box were collected from the inverters. The HMI was able to show the monitored parameters, along with alarm status. Combined with the SUNTECH module characteristics, the authors calculated a series of performance indicators for the PV plant during 2019 and 2020. The performance ratio, the most important indicator for a PV plant, was 83.41% in 2019 and 81.24% in 2020, showing a 2.17% decrease and suggesting that the maintenance can be performed.

In [101] a solution for monitoring and management of distributed PV systems using cloud infrastructure at IoT devices has been proposed. Analog sensors collected environmental parameters and DC current/voltage of PV panels by data acquisition level, and then sent them to

process level. The implementation of the network infrastructure depended on the location of the PV installation. In remote areas, cellular network and satellite network were suitable for sending data to the data center. After inspections by the firewall, the data would be displayed in real time and stored for future use. Inverters were able to receive commands from the data center and run in the under-excited condition to compensate reactive power. The authors claimed that the remote controlling of PV systems' active power and reactive power was achieved.

In [102] the authors designed and implemented a wireless sensor network (WSN) and an IoT platform for a rooftop PV system. Current sensor, voltage sensor, temperature and humidity sensor, and optical dust sensor were deployed to collect key parameters of the PV system. The Arduino UNO microcontroller sent the data to Node MCU through serial communication. Via Wi-Fi network, Node MCU transferred the data to Ubidots, a cloud server. Dashboards for displaying the collected data were created in Ubidots. Statistical analysis including average value in different intervals and maximum/minimum value was also achieved by the platform.

Using IoT architecture, the authors of [103] proposed an anomaly detection methodology for non-ideal operating conditions of PV plants. Irradiance, ambient temperature, PV module temperature, and output current/voltage/power from the inverter are collected by Raspberry Pi Zero and ADC1115. The parameters were then published to the MQTT broker and sent to dashboards which served as the subscriber. The core of the anomaly detection algorithm is whether the deviation from the mean value is larger than 2-sigma. If the short-window average value of power, voltage, and PV module temperature all deviated during the same period, non-ideal conditions were identified. The authors claimed that this algorithm only used real-time data, not requiring historical system data.

The inverter performance was monitored and analyzed in real time by SCADA Haiwell in [104]. Input parameters of the inverter including DC voltage/current/power, and output parameters including energy yield, AC power, AC phase voltage/current were collected by PLC via Modbus TCP/IP protocol. Haiwell Cloud also received the data via Ethernet and created HMI to show all the collected data, using Haiwell HMI C7S. Verified by the measurements of existing iSolarCloud system, the newly developed system was claimed to precisely monitor the inverter online via mobile phones or personal computers.

Authors of [105] proposed a smart embedded system to remote monitor a PV array and diagnose the faults via machine learning algorithms. PV current, PV voltage, ambient temperature, PV cell temperature, and solar irradiance were collected by a NodeMCU ESP8266 module, and then sent to an open-source IoT application Blynk for storage and remote display. Furthermore, these data were also sent to a Raspberry Pi 4 running fault detection, fault classification, and user notification. The authors claimed that the fault detection showed a 97.5% accuracy using multilayer perceptron neural networks, and the fault classification showed a 96.8% accuracy using stacking ensemble algorithm.

A P2P-based method to protect the SCADA communication channels was proposed in [106]. Multiple copies of data are stored at different nodes in the network, and data is transmitted along multiple paths. Even if one node or one path fails, the data can still be transmitted. Achieved through data replication and path redundancy, the availability and the integrity of the system are improved when faults or attacks are present. Evaluated by experiments, the discovery and recovery ratios of data were improved from above 60% to above 70%.

The design and implementation of a low-cost and open-source SCADA system using Thingier.IO were presented in [74]. ESP32 Thing microcontroller collected the PV panel voltage, the PV panel current, and the battery voltage. The data were then sent to the Thingier.IO IoT platform running locally on a Raspberry Pi microcontroller. HMI dashboards were created on the platform to visualize the data and monitor remotely. However, some drawbacks of the design exist. The overall power consumption of the SCADA system is as high as 5.0 W, excluding the wireless router power consumption; the system will cease to log and display data if the communication between the ESP32 Thing and the Raspberry Pi fails; the Thingier.IO Raspberry Pi ISO Image is not free of charge.

In [107] the authors proposed an IoT-based SCADA system to monitor the PV system parameters. Node-RED running on the single board computer Banana Pi M4 Berry serves as the MTU to aggregate, display data and control the load. However, the system design only allows local access to the server. Devices that are not on the same LAN fail to read the logged data. Furthermore, the designed system will stop logging and displaying the PV system parameters if the Node-RED ceases to function.

Previously reviewed papers are part of the bibliography we composed to comprehensively study the latest research results. Table 5-1 lists the monitored parameter types and deployed IoT platforms in related papers. To the best of our knowledge, there is no IoT-based SCADA system where two IoT platforms are utilized for improved system robustness by data redundancy, and graphic images are collected. To be specific, ESP32-S3 hosts a camera web server and forwards data to ThingSpeak via HTTP for analysis and real-time display, and ESP32-E collects data from the PV module, controls the load, and displays data in Arduino Cloud via MQTT. While the

images in this design are from the load, it can be easily transferred into other applications, such as identification of dust on PV panels [108,109] and short term weather forecast by sky imagery [110]. The main contributions of this newly proposed design are listed below.

- Real-time monitoring and control of a PV system are achieved using Internet of Things architecture. The integration of IoT platforms facilitates the system implementation by providing functions such as data aggregation, communication security, and data-driven applications.
- The design of using two IoT platforms increased the system robustness by the data redundancy. When one platform shuts down its service by schedule or accident, the other platform can continue to function.
- Images of the load can be accessed on a web server, enabling the visual verification of the load status. Anomalies that might not be detectable through voltage or current sensors alone, such as a burned-out lamp can be observed and detected. Visual surveillance also provides versatility regarding environmental changes around the lamp and intuitions about the status of multiple lamps.

*Table 5-1 Types of monitored parameters and IoT platforms used in related papers.*

<b>Reference</b>	<b>Monitored Parameter Types</b>	<b>IoT platforms</b>
[95]	Voltage, Current, Temperature	Cayenne
[96]	Voltage, Power, MQTT duty cycle	ThingSpeak
[97]	Voltage, Current, Power, Solar Irradiation	Eclipse Kapua
[98]	Voltage, Current, Power, Energy,	Emoncms

---

Environmental Measurements		
[99]	Voltage, Current, Power, Meteorological Variables	Google Cloud Platform
[102]	Voltage, Current, Power, Temperature and Humidity, Dust	Ubidots Cloud
[103]	Voltage, Current, Power, Solar Irradiance, Ambient and PV module temperature	Apache (web server)
[104]	Voltage, Current, Active/Reactive Power, Grid Frequency	Haiwell Cloud
[105]	Voltage, Current, Power, Ambient and PV module temperature	Blynk IoT
[74]	Voltage, Current, Power	Thingier.IO
[107]	Voltage, Current, Power	Node-RED
This design	Voltage, Current, Power, Graphic Image	Arduino Cloud and ThingSpeak

---

### 5.3 System Descriptions

The conceptual representation of the developed IoT-based SCADA architecture in five layers, along with the schematic diagram of the PV system and the SCADA system, are presented in this section. In Figure 5-1, the perception layer, the network layer, the security layer, the middleware layer, and the application layer consist of the overall IoT architecture. Within the perception layer, an ESP32-E (FireBeetle 2 ESP32-E, DFRobot<sup>®</sup>, Shanghai, China) collects voltage

readings and current readings from the respective sensors. Moreover, the ESP32-E sends the control signal to the actuator (a relay) for turning on or off the load. Another ESP32 series board ESP32-S3 (ESP32-S3-WROOM CAM Board, Freenove<sup>®</sup>, Shenzhen, China) builds a webserver to display the load images from a 2-Megapixel OV2640 camera mounted on the development board. Network layer as the second layer ensures data packets are transmitted from the source IoT devices to other devices, central servers, or the cloud. ESP32-E forwards the collected sensor data to Arduino Cloud using MQTT protocol, and to ESP32-S3 webserver through HTTP. ESP32-S3 transmits the parsed sensor data from the webserver to ThingSpeak through HTTP, afterwards. The security layer, middleware layer, and the application layer are integrated and implemented by Arduino Cloud and ThingSpeak platform. In Arduino Cloud, the device authentication is achieved by matching device login name and the device key generated by the platform. Similarly, Read/Write API keys are provided in ThingSpeak to grant read/write access to the ThingSpeak channel data. The two IoT platforms can also function as the middleware layer. Arduino Cloud aggregates all the sensor data from ESP32-E, stores the historical sensor data, and serves as a bridge between the hardware devices and the application layer. In addition to these functions, ThingSpeak also features easy-to-implement data processing, such as average, median, sum, rounding. The final application layer consists of several applications, including dashboards, email alert, and integration with other software (MATLAB for ThingSpeak).

Figure 5-2 illustrates the developed system from the view of SCADA system. Voltage sensors are placed in parallel with the PV panel, and the battery. Current sensors are in series arrangements with the PV panel, the battery, and the load. A relay is also connected in series with the load, controlling its ON/OFF state. The sensors and the relay comprise the FIDs. The RTU (ESP32-E) collects the sensor data and controls the relay according to the battery voltage,



where low voltage leads to turn off the load. Additionally, the RTU sends the data to the MTU (ESP32-S3) and to the middleware (Arduino Cloud). The MTU forwards the received data to the middleware (ThingSpeak) and sends the images of the load to the webserver. Two middleware both provide HMIs (dashboards) and data historians (online data storage).

## 5-layer IoT-based SCADA architecture

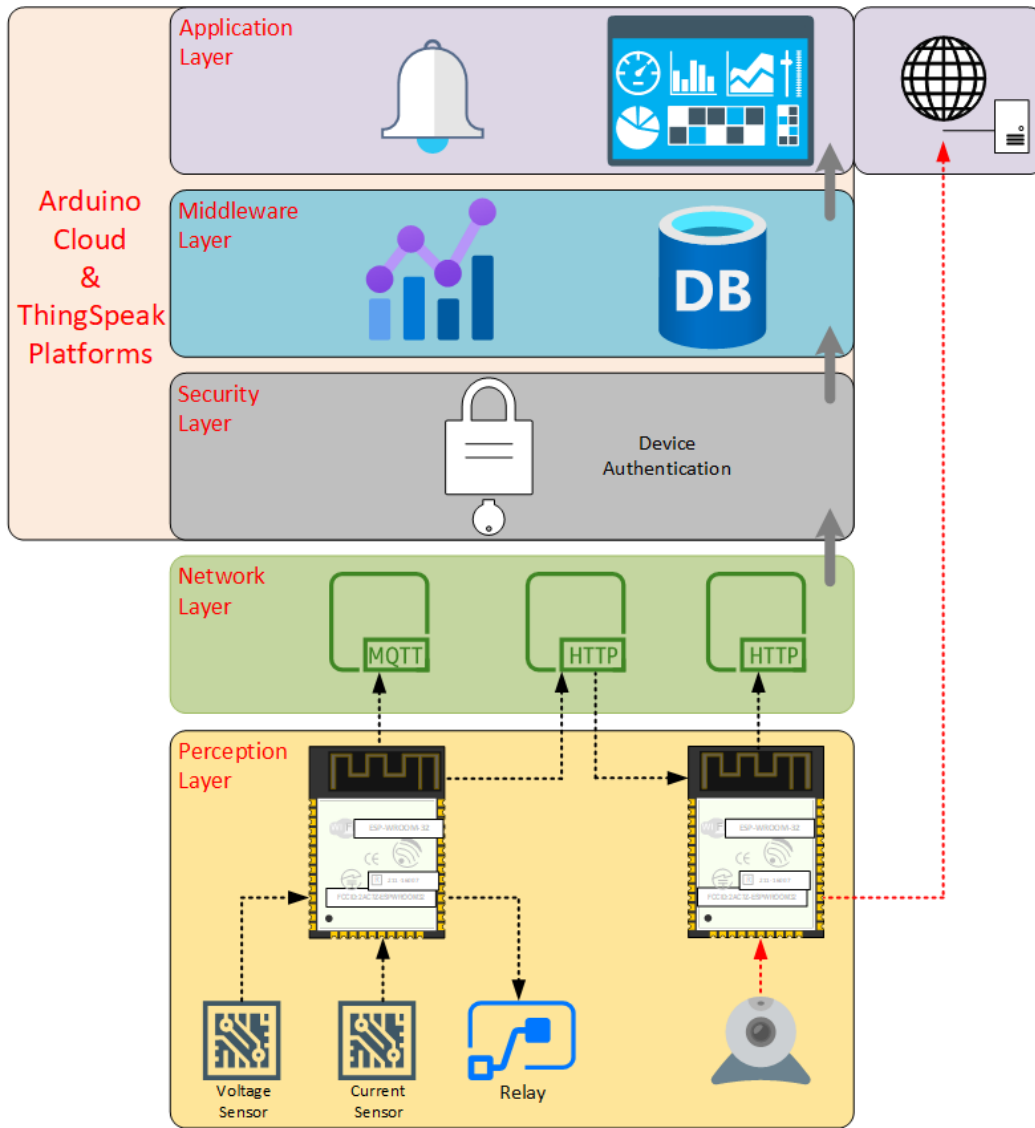


Figure 5-1 Proposed design of 5-layer IoT-based SCADA architecture.

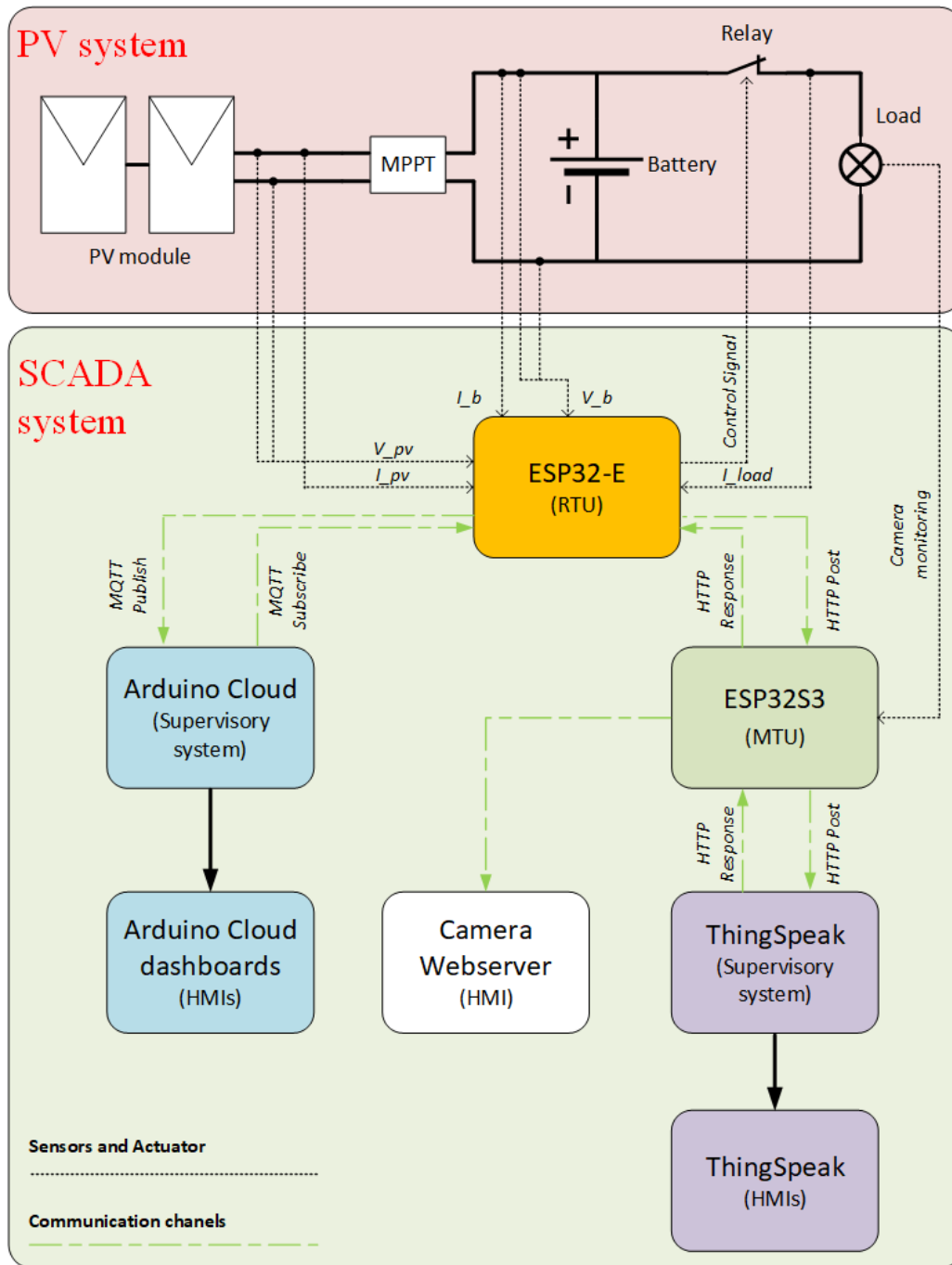


Figure 5-2 SCADA components of the proposed design.

## 5.4 SCADA System Components

In this section, we introduce the hardware and IoT platforms crucial to the system design. The ESP32-E as the MTU and part of the perception layer, controls the relay and collects the sensor data to forward to the ESP32-S3 and the Arduino Cloud. The MTU, ESP32S3, captures the images of the monitored load and sends the sensor data to ThingSpeak. The current sensors, voltage sensors, the relay, and the camera comprise the perception layer, which serve as the FIDs.

### 5.4.1 ESP32-S3 and ESP32-E

The ESP32-S3-WROOM-1-N8R8 is a powerful microcontroller module, that comprise the compact ESP32-S3-DEV-KIT-N8R8 development board. For hardware, equipped with an Xtensa® 32-bit LX7 dual-core processor, the module has a main frequency that can be up to 240 MHz [111]. PSRAM provides extra memory space to increase the usable ESP32 system memory, which is helpful for accelerating access to memory and expanding the buffer. The 8 MB PSRAM on the ESP32-S3 module allows the seamless integration of the camera. It also has 512 KB SRAM, 384 KB ROM, 16 KB SRAM in RTC, and 8 MB flash memory. Its working voltage is from 3.0 V to 3.6 V. Resourceful peripherals including up to 45 GPIOs, two 12-bit ADC, two I2C/I2S, one camera interface, four SPI, one USB OTG, three UART interfaces, and so on. 2.4 GHz Wi-Fi (802.11 B/G/N) and Bluetooth® LE are adopted to enable wireless communication with superior RF performance. On the development board, USB and UART development can be achieved simultaneously by using onboard CH343 and CH334 chips. Excelling in its performance and versatility, ESP32-S3 is an excellent choice for developing IoT applications.

The superior support for vector instructions also shows great potential for computer vision tasks such as facial recognition, object detection, and image classification.

Another ESP32 module employed in this design is ESP32-WROOM-32E-N4, powering the FireBeetle 2 ESP32-E development board. Having Xtensa® 32-bit LX6 dual-core processor, the module has a clock frequency up to 240 MHz [113]. It also has 448 KB ROM, 520 KB SRAM, and 16 KB SRAM in RTC. The operating voltage is from 3.0 V to 3.6 V. As for wireless communication, it supports 2.4 GHz Wi-Fi (802.11 B/G/N) and Bluetooth V4.2 and BLE. Peripherals are supported such as SD card, UART, SPI, SDIO, I2C, GPIOs, and so on. ESP32-WROOM-32E is more of a generic and powerful MCU module that can target various applications including low-power sensor networks, when compared to ESP32-WROOM-S3.

### **5.4.2 Arduino Cloud**

Developed by Arduino® company, the Arduino Cloud is an online open-source platform facilitating the creation, deployment, and monitoring of IoT projects. Key features of the platform are briefly introduced.

- **Device Management:** It allows users to manage multiple Arduino boards or third-party devices on one platform.
- **Security:** To set up a new device in the platform, a device ID and a secret key are provided to realize the authentication process.
- **Data Visualization:** Dashboards can be easily created and customized according to users. Sensor data from different devices can be monitored on one webpage in real time.

- **Cloud Programming:** Users can write, compile, and upload codes to the devices from the web browser, instead of installing a local programming IDE environment.
- **Remote Control:** Widgets that link to pre-defined variables can control the states of these variables.
- **Over-the-Air Updates:** Users can update the firmware of the devices over the air, without removing the field devices from their deployment place.
- **Cloud Services Integration:** Arduino Cloud supports integration with other cloud services, such as Google Cloud, AWS, and IFTTT, allowing for complex and automated workflows.

*Things* handle the communication between IoT devices and the Arduino Cloud. Cloud variables, the associated device, network SSID and password, sketch, and metadata comprise Things. Figure 5-3 is the Arduino Cloud Things setup interface with ESP32-E. Arduino Cloud free plan supports two Things, with five cloud variables each Thing. Cloud data retention is one day on a rolling basis.

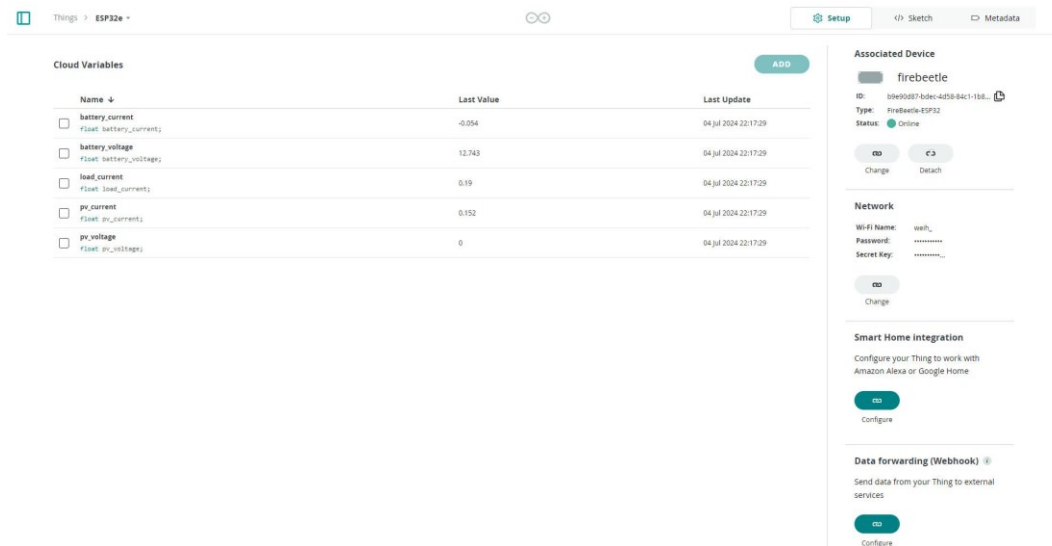


Figure 5-3 Arduino Cloud Things setup interface with ESP32-E.

### 5.4.3 ThingSpeak

ThingSpeak is an IoT analytics platform that supports the aggregating, visualization, and analysis of real-time data streams in the cloud. Users can transmit data from their devices to ThingSpeak, generate immediate visualizations, and send alerts via web services. By incorporating MATLAB<sup>®</sup> analytics, ThingSpeak allows users to write and execute MATLAB codes for data preprocessing, visualization, and analysis, simplifying the process for engineers and scientists to prototype and develop IoT systems without the need for server setup or web software development. The free plan supports three million messages per year, 15 seconds updating interval, and four channels.

A typical workflow in ThingSpeak includes several steps. Starting by configuring the accounts and channels, each channel can have eight fields at maximum to store numeric or alphanumeric. Subsequently, users can read/write data from/to the channel, by using HTTP calls from the REST API or MQTT subscribe/publish method. After that, MATLAB Analysis app helps to prepare,

filter, and analyze the channel data. Calculating averages values, eliminating data outliers, and replacing missing values in data are some common functions that MATLAB Analysis app can realize. Users can then visualize the data and trigger an action such as sending an email. Finally, MATLAB Add-On Toolboxes can be utilized to do specialized analysis including prediction with data. Figure 5-4 is the ThingSpeak channel created on June 5, 2024, with more than 33000 entries.

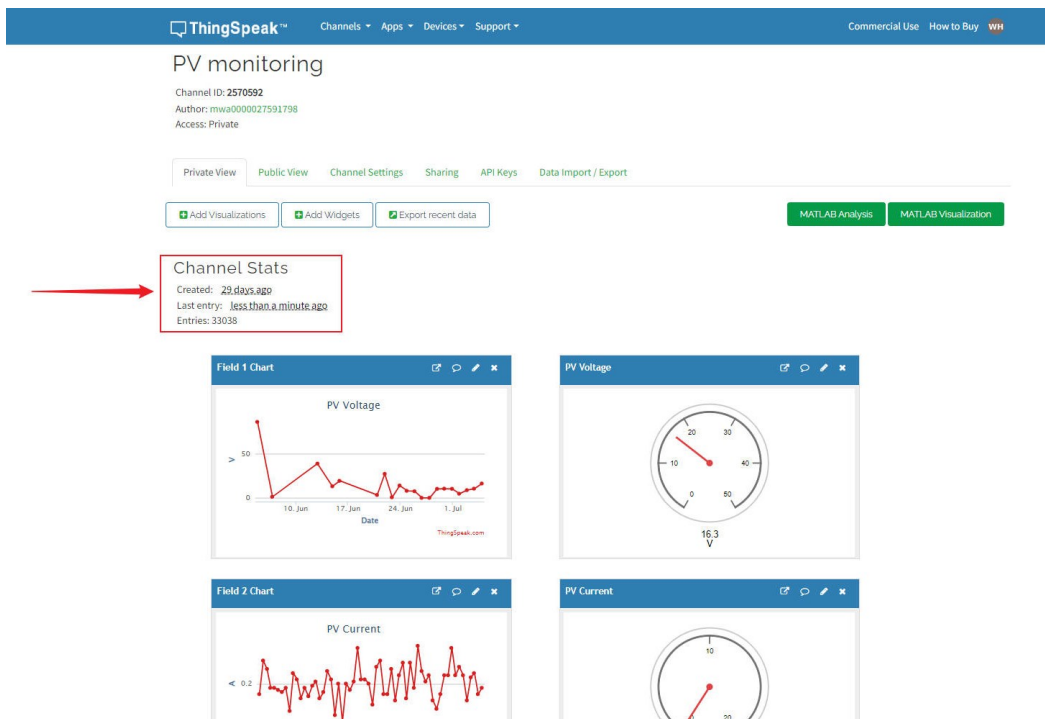


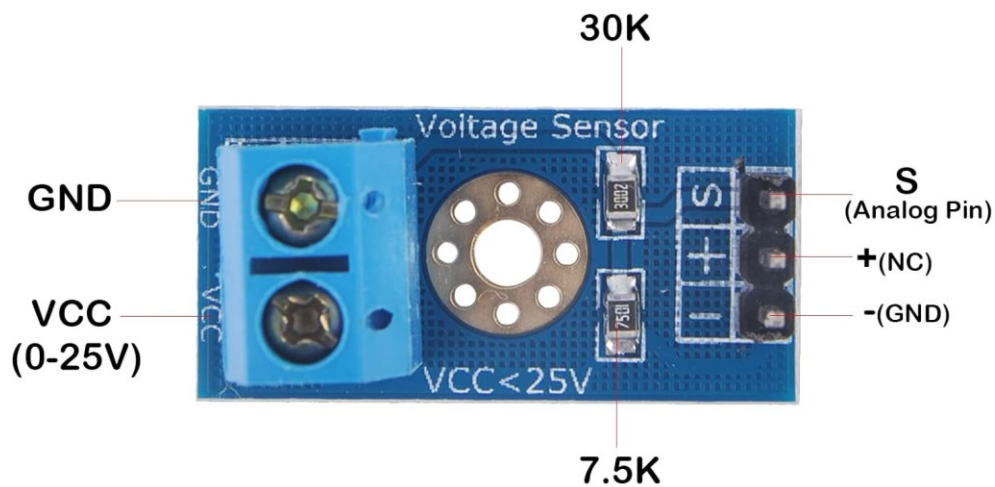
Figure 5-4 ThingSpeak channel private view.

#### 5.4.4 Voltage Sensor and Current Sensors

The voltage sensor used in this design is suitable for 0-25 V DC voltage measuring. It is essentially a voltage divider that reduces the input voltage by a factor of 5 and generates a



corresponding analog output voltage signal. Manufactured by HiLetgo<sup>®</sup>, Shenzhen, China, this is an open-source electronic device featuring low price, down to about 1.7 USD per item [114]. Figure 5-5 is the marked visual representation of the voltage sensor. The left side of the sensor is the electrical side, with the voltage to be measured connected to the VCC and GND port. The right side is the electronic part, with the analog pin connected to the microcontroller analog pin and the GND pin connected to the microcontroller GND pin.



*Figure 5-5 Voltage sensor.*

Calibration process is needed to implement voltage measuring. One fifth of the source voltage divided by the ESP32-E reference voltage, is equal to the voltage analog signal divided by 4096 (12-bit ADC for ESP32). The corresponding relationship between the measured voltage and ESP32-E analog voltage readings is given in Equation 3, where  $V_s$  is the source voltage to be measured,  $V_{ref}$  is the reference voltage for ESP32-E (3.26 V), and  $V_{sig}$  is the analog value read by the ESP32-E.

$$\frac{V_s / 5}{V_{ref}} = \frac{V_{sig}}{4096}$$

$$V_s = \frac{5 \times V_{ref}}{4096} \times V_{sig} \quad (3)$$

The current sensor deployed in our implementation is ACS712 current sensor [88] manufactured by Allegro Microsystems<sup>®</sup>, Manchester, NH, USA. This sensor is comprised of a linear current sensor IC based on Hall Effect, 2.1 kVRMS voltage isolation, and a low-resistance current conductor. Figure 5-6 is the image of ACS712 current sensor. The left side of the sensor is connected in series to the current to be measured. On the right side, VCC/GND pin is supposed to be connected to 5 V/ground to power the sensor. OUT pin is connected to the analog pin of ESP32-E.



*Figure 5-6 Image of ACS712 current sensor.*

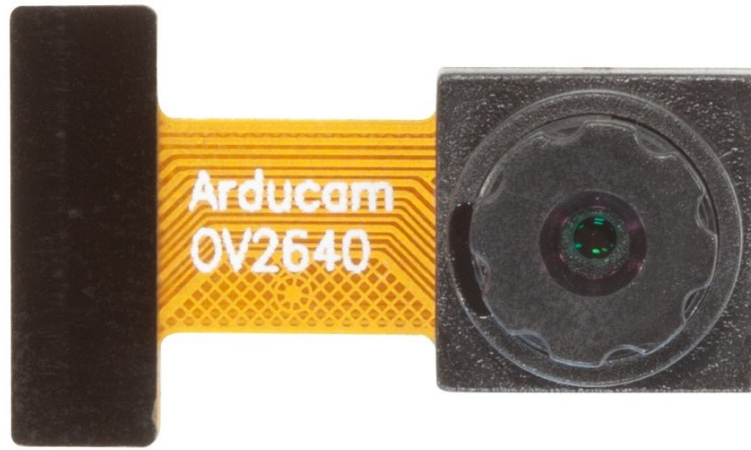
Similarly, the ACS712 current sensor requires calibration to make results accurate. When VCC and GND are connected to 5 V and ground, the OUT pin has an AcsOffset analog reading on ESP32-E if no current is flowing through the sensor. In addition, the sensitivity of ACS712-30A is 66 mV/A, meaning that every change of 1 A leads to the change of 66 mV on OUT pin. Therefore, we have Equation 4 giving the correspondence between the measured current and the analog readings on ESP32-E, where sensitivity is 66 mV/A, AcsOffset is the analog reading on ESP32-E when the measured current is zero, and A<sub>sig</sub> is the analog reading of the measured current.

$$\frac{I_s \times \text{Sensitivity}}{V_{ref}} = \frac{A_{sig} - \text{AcsOffset}}{4096}$$

$$I_s = \frac{V_{ref}}{4096 \times \text{Sensitivity}} (A_{sig} - \text{AcsOffset}) \quad (4)$$

### 5.4.5 OV2640 Camera

The OV2640 is a small (1/4 inch) CMOS UXGA image sensor, that is functioning as the combination of a single-chip UXGA (1600×1200) camera and an image processor [115]. The single-chip camera supports image sizes including UXGA, SXGA (1280×1024), SVGA (800×600), and any size down to 40×30. The maximum image transfer rate is 15 frames per second at UXGA resolution. The signal to noise ratio (SNR) and dynamic range are 40 dB and 50 dB, respectively. In addition, the image processor can be programmed to select settings of resolution, frame rate, exposure control, white balance, gain control, noise reduction, and so on. In terms of power supply and power consumption, the maximum voltage and power needed are 3.3 V and 140 mW. Having the powerful on-chip ISP with JPEG encoding and low power consumption, the OV2640 image sensor is popular for IoT projects. Figure 5-7 is the visual image of Arducam<sup>®</sup> OV2640 camera module, with DVP-friendly interface which makes it easy to connect to microcontrollers.



*Figure 5-7 OV2640 camera module.*

## **5.5 Implementation Methodology**

In the developed IoT-based SCADA system, one microcontroller (ESP32-E) sends parameters of the monitored PV system, which are collected by sensors, to another microcontroller (ESP32-S3) and two IoT platforms (Arduino Cloud and ThingSpeak). Furthermore, ESP32-E also controls the relay to turn ON/OFF the load, while ESP32-S3 sends load images to the web server it built. Figure 5-8 and Figure 5-9 describe the flowchart for algorithms run in ESP32-E and ESP32-S3, respectively. Two processes are executed in parallel in ESP32-E: one is translating sensor readings to usable sensor values, controlling the relay, and sending sensor values to ESP32-S3, the other is to update the sensor values to Arduino Cloud regularly. This asynchronous programming ensures that multiple tasks will run concurrently without blocking each other,

leading to an improved system efficiency. Similarly, in ESP32-S3, there are also Process A and Process B that are executed simultaneously. Process A consists of two steps: Web server A listens to port 81 for incoming HTTP POST requests sent by ESP32-E; post sensor data to ThingSpeak platform. Meanwhile, Process B is that web server B monitors default port 80 for any HTTP GET requests for the latest image from a web browser. Using different port numbers can separate services to avoid conflicts. Algorithm 1 is the pseudocode flashed into ESP32-E by Arduino Cloud. Furthermore, algorithm 2 is the pseudocode running on ESP32-S3, which is uploaded by Arduino IDE (version 2.3.2).

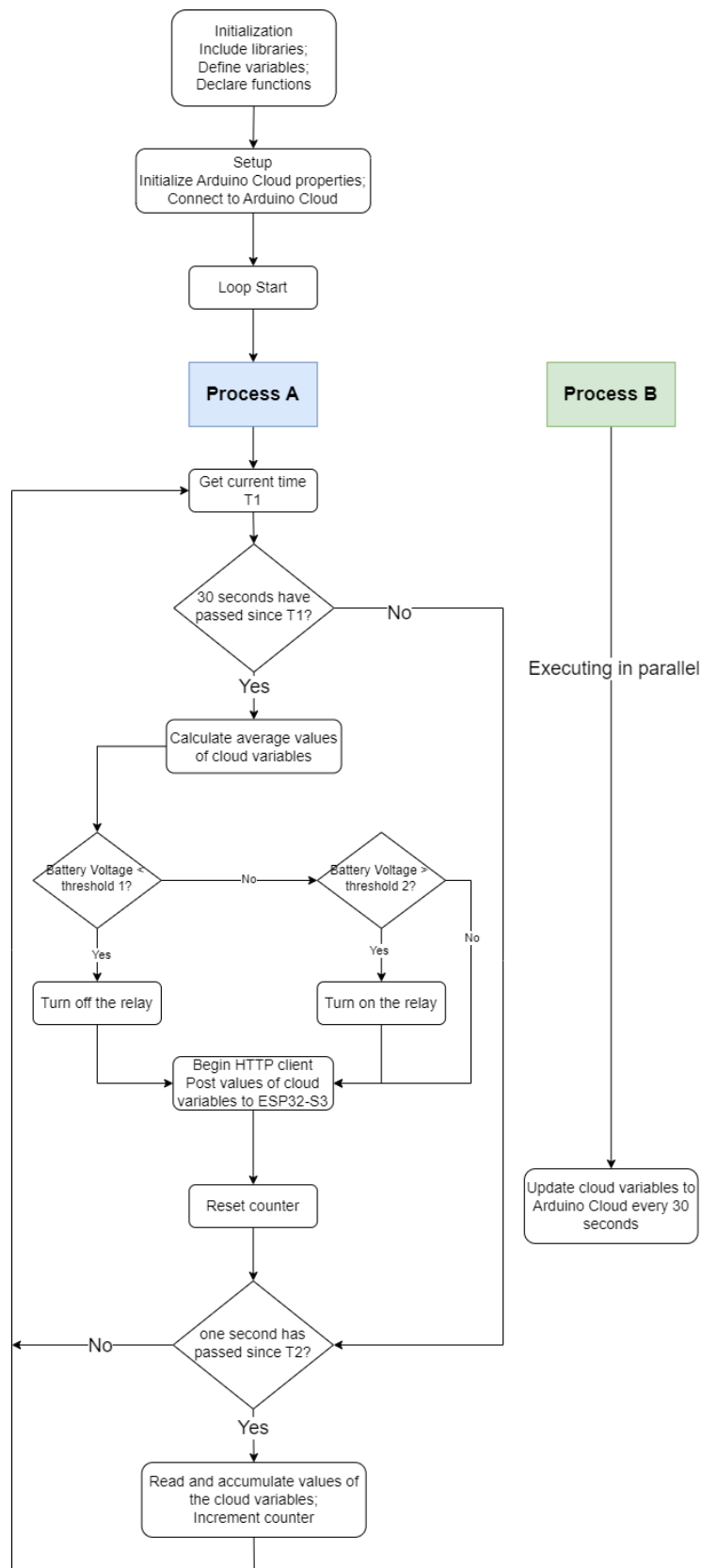


Figure 5-8 Flowchart of programs on ESP32-E.

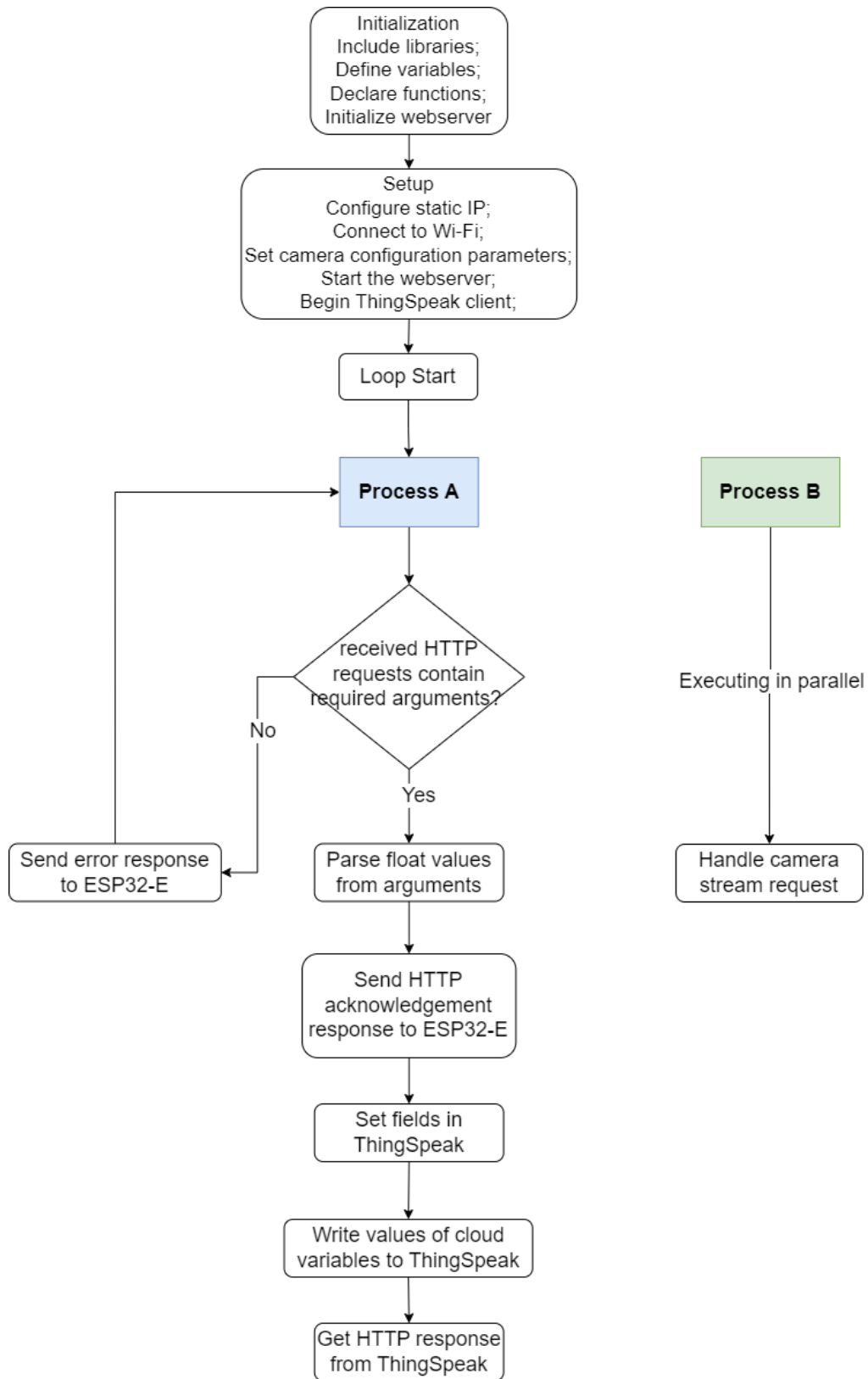


Figure 5-9 Flowchart of programs on ESP32-S3.

---

**Algorithm 1:** Data acquisition, automatic control, and data communication by ESP32-E.

---

Initialization;

1. Include libraries for Arduino Cloud, HTTP client, and Arduino Cloud connection handler;
2. Define constants, including device login name, device key, Wi-Fi network SSID and password, server URL, thresholds, intervals, and counter for averaging;
3. Define cloud variables including PV panel voltage/current, battery voltage/current, and the load current;
4. Declare sensor calibration functions;

Setup Function (executed only once);

5. Initialize cloud properties with read permissions and update intervals;
6. Connect to Arduino Cloud;

Loop Function (executed repeatedly);

7. Call a callback function to update cloud variables at predefined update intervals to Arduino Cloud;
8. Get current time;

**If *interval\_1* (30 seconds) has passed then**

9. Calculate average values (dividing the accumulated values by counter) of cloud variables;

**If *battery voltage* is below *threshold\_1* then**

10. Cut the load by turning off the relay;
-



---

**Else If** *battery voltage is above threshold\_2* **then**

11. Power the load;

**Else:**

12. Continue;

**End**

13. Begin HTTP client;

14. Send values of cloud variables to server (ESP32-S3) via HTTP POST request;

15. Get HTTP response code;

16. Reset counter;

**Else If** *interval\_2 (one second) has passed* **then**

17. Read and accumulate values of the cloud variables;

18. Increment counter;

**Else:**

19. Continue;

**End**

---

---

**Algorithm 2:** Data communication, and camera web server by ESP32-S3.

---

Initialization;

1. Include libraries for ThingSpeak, Wi-Fi, Webserver, and ESP Camera;

2. Define constants, including static IP, ThingSpeak channel number and write API key, Wi-Fi network SSID and password, and ThingSpeak client;

3. Define cloud variables including PV panel voltage/current, battery voltage/current, and

---

---

the load current;

4. Initialize webserver on port 81;
5. Declare the function for starting the camera server, and the callback function for handling float-type HTTP requests;

Setup Function (executed only once);

6. Configure static IP;
7. Connect to Wi-Fi network;
8. Set camera configuration parameters based on the model;
9. Start the webserver for camera on port 80, and for receiving sensor data from ESP32-E on port 81;
10. Begin ThingSpeak client;

Loop Function (executed repeatedly);

11. Handle incoming client (ESP32-E) POST requests to the server;

Handle function;

**If** *received requests contain required arguments (such as “pv\_voltage”)* **then**

12. Parse and convert the arguments to float values;
  13. Send HTTP acknowledgement response to the client (ESP32-E);
  14. Set fields in ThingSpeak;
  15. Write values of cloud variables to ThingSpeak;
  16. Get HTTP response from ThingSpeak;
-

---

**Else**

17. Send error response to the client (ESP32-E);

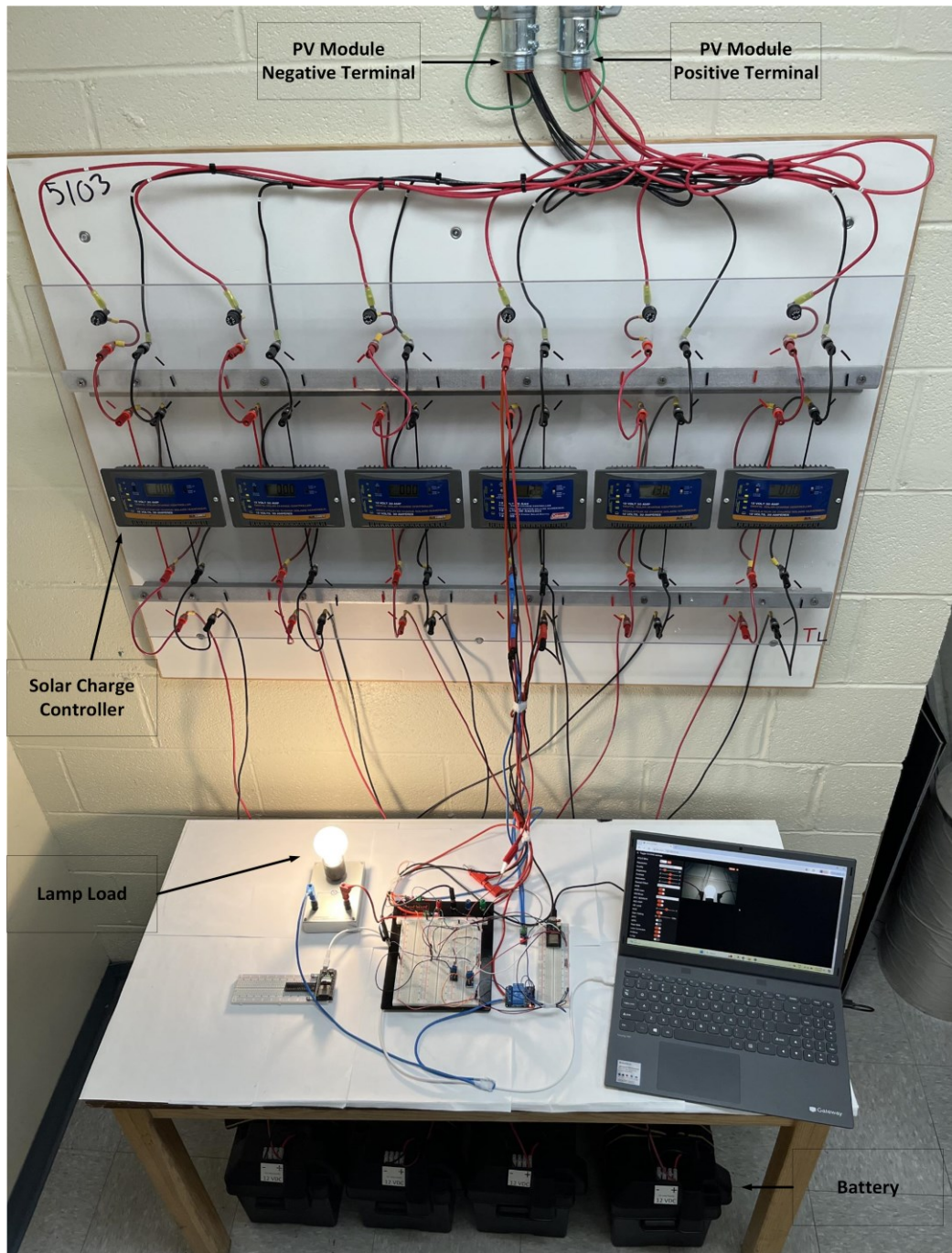
**End**

---

## **5.6 Experimental Setup**

This section explains the details of the experiments carried out to prove the effectiveness of the IoT-SCADA system design. At Memorial University ECE laboratory, the IoT-SCADA system is integrated into the installed PV modules for monitoring and control. The PV module contains two panels, with each one producing 130 W at 17.1 V under ideal conditions.

Figure 5-10 demonstrates the overview of the designed system setup. Positive and negative power terminals of the PV modules on the rooftop are connected to the digital solar charge controllers, the batteries and the load that are placed indoor, through red and black power wires.



*Figure 5-10 Overview of the experimental setup.*

Figure 5-11 presents the hardware schematic diagram of the IoT-based SCADA system. The voltage signal output pin VS of the PV voltage sensor is connected to SENSOR VN pin of ESP32-E, meanwhile the VS pin of the battery voltage sensor is connected to SENSOR VP pin.

GPIO15, GPIO34, and GPIO35 of ESP32-E are connected to the voltage output pins of the battery current sensor, the PV module current sensor, and the load current sensor. Additionally, the DVP interface is used to connect the OV2640 camera module to ESP32-S3. The interface consists of several signal lines. To be specific, 8-bit data lines (D2-D9) carry pixel data from the camera to the microcontroller; vertical synchronization (VSYNC) and horizontal reference (HREF) indicate the start of a new frame and a new row of pixels, respectively; pixel clock (PLCK) is responsible for providing a clock signal when data lines have valid data; external clock (XCLK) is an input clock signal provided to the camera by the microcontroller to drive its internal logic; serial clock (SIOC) synchronizes the data transfer over the I<sup>2</sup>C bus, while serial data (SIOD) enables bi-directional data transfer (camera configuration commands and settings) between the master ESP32-S3 and the slave camera. Figure 5-12 is the implementation of the hardware schematic diagram.

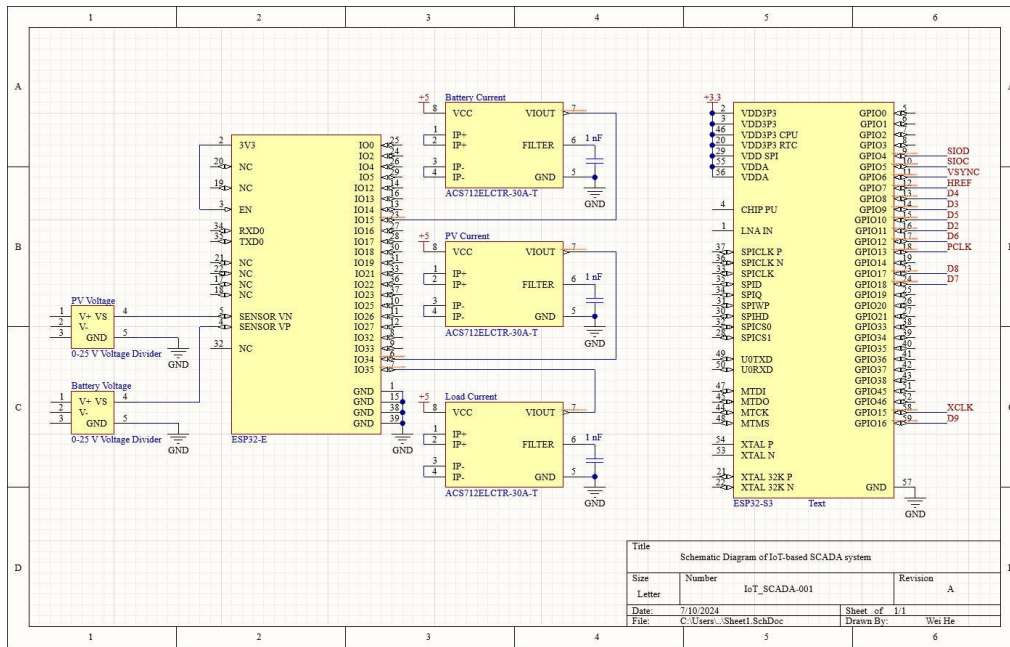


Figure 5-11 Hardware schematic diagram.

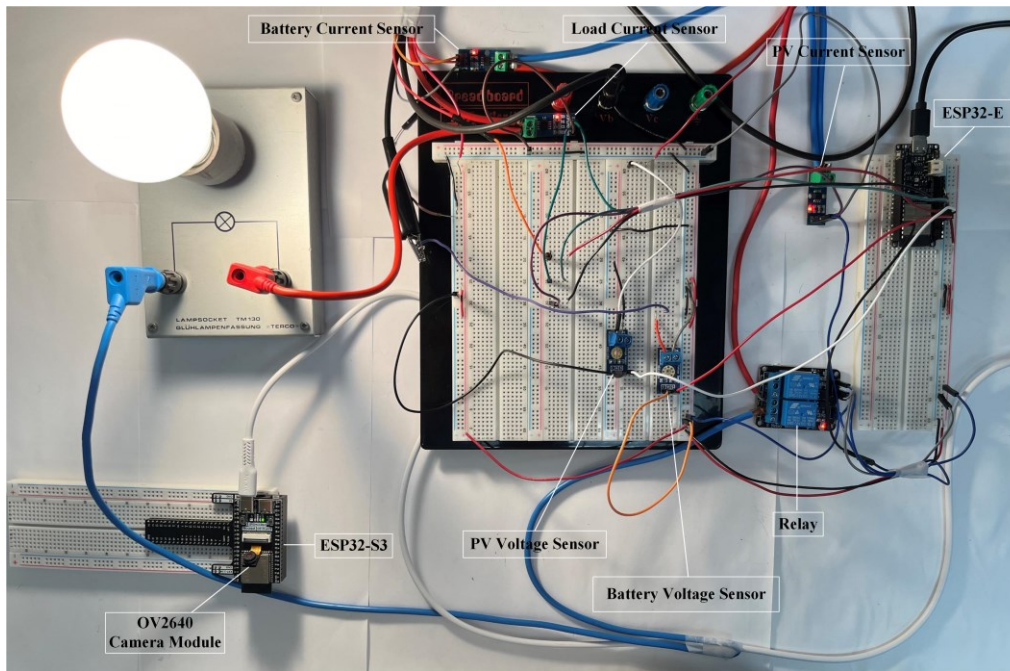


Figure 5-12 Hardware setup of IoT-SCADA system.

In spite of the hardware experimental setup, the application layer which is implemented in Arduino Cloud, ThingSpeak, and the camera web server is also introduced.

Arduino Cloud provides different widgets in the dashboard application, including Switch, Push Button, Slider, Chart, etc. By simply linking the cloud variables to the widgets, the values of the cloud variables can be shown in the dashboard without much effort. In this setup, the Chart widget is chosen to display the changing values. Different time scales of the data are available in Chart, which are 15/7/1 day(s), 1 hour, and live. However, with the free plan, the maximum length of data displayed is one day. If 15 days are selected, it will only present the one-day data 15 days ago, instead of displaying all 15-day data.

Another function is the email alert by ThingSpeak. Users can first read the ThingSpeak channel data, by *thingSpeakRead* function. Channel ID, read API key, numbers of data points, and field number in the channel should be configured in the function. Furthermore, an email alert can be configured with HTTP POST method and triggered when conditions are met. To achieve this, Headers that include the user-specific ThingSpeak Alerts API key, and Body parameters containing the monitored value, are sent to the URL <https://api.thingspeak.com/alerts/send>. Figure 5-13 is the MATLAB program that sends an email alert when the battery voltage is below 12.5 V. To enable the automatic alert function, TimeControl application in ThingSpeak helps to trigger an action at a specific time. The recurrence can be down to minute level. By combining TimeControl application and the email alert function, the battery voltage LOW condition can be alerted to the email account in no more than five minutes.



```

1 % Store the channel ID for the PV monitoring channel.
2 channelID = 2570592;
3 readAPIKey = 'GW5DUUKETI09WJTU';
4
5 % Provide the ThingSpeak alerts API key. All alerts API keys start with TAK.
6 alertApiKey = 'TAKzIUH98ogbRUuzpzu';
7
8 % Set the address for the HTTP call
9 alertUrl="https://api.thingspeak.com/alerts/send";
10
11 % webwrite uses weboptions to add required headers. Alerts needs a ThingSpeak-Alerts-API-Key header.
12 options = weboptions("HeaderFields", ["ThingSpeak-Alerts-API-Key", alertApiKey ]);
13
14 % Set the email subject.
15 alertSubject = sprintf("Low Battery Voltage");
16
17 % Read the recent data.
18 BV_Data = thingSpeakRead(channelID,'ReadKey',readAPIKey,'NumPoints',30,'Fields',3);
19
20 % Check to make sure the data was read correctly from the channel.
21 if isempty(BV_Data)
22     alertBody = ' No data read from battery. ';
23 else
24     % Get the most recent point in the array of battery voltage data.
25     lastBV_Value = BV_Data(end);
26     lastBV_Value
27
28     % Set the outgoing message
29     if (lastBV_Value <= 12.5)
30         alertBody = ' The battery voltage is low! ';
31
32     else (lastBV_Value > 12.5)
33         alertBody = 'The battery voltage is normal. ';
34     end
35 end
36
37 % Catch errors so the MATLAB code does not disable a TimeControl if it fails
38 try
39     webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);
40 catch someException
41     fprintf("Failed to send alert: %s\n", someException.message);
42 end
43

```

*Figure 5-13 MATLAB program for email alert when the battery voltage is below 12.5 V.*

Finally, the camera web server is built on ESP32-S3. By typing the configured ESP32-S3 IP address with a default port number 80 in a web browser, the camera web server processes the HTTP request from the browser and sends back a response to the browser. Figure 5-14 shows the camera web server interface. On the left side there is a settings column, providing options to alter the OV2640 settings. Clock frequency, image resolution/brightness/contrast, H-Mirror, V-Filp, etc. are subject to users' preferences.



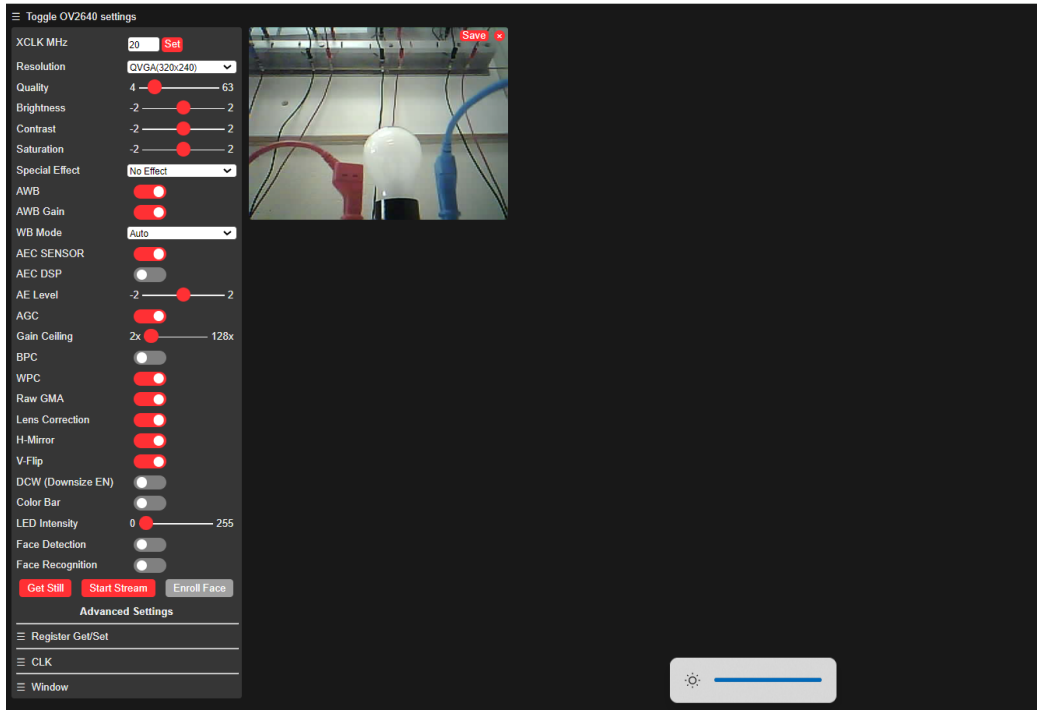


Figure 5-14 Camera webserver interface.

## 5.7 Results

From June 23, 2024, to July 13, 2024, 20-day results were recorded to validate the correct performance of the developed IoT-SCADA system. On July 1, there was a scheduled maintenance for Arduino Cloud from 08:17 to 12:00, July 2 in UTC [116]. During that time, no data was recorded on the Arduino Cloud platform since the device connectivity was cut off. However, ThingSpeak received, stored, and displayed the data on July 1 as normal, without being affected. By the design of data redundancy, our developed system successfully maintained to function, even when Arduino Cloud shut down the service.

Figure 5-15 shows the dashboards displaying the recorded PV system parameters on July 11 and July 12 on Arduino Cloud. Five digital meters (five Velleman DVM890L) were utilized to measure the PV system parameters simultaneously, the readings of which matched the IoT-

SCADA system results.  $\pm 0.5\%$  accuracy for DC voltage ranging from 200 mV to 200 V, and  $\pm 0.8\%$  accuracy for DC current ranging from 2 mA to 200 mA ( $\pm 2\%$  for 20 A) provide reliable measurement results. Furthermore, the validation process based on commercially available measurement instrument DI-145 was went through in our previous paper [36]. To avoid repetitiveness, the details of the data verification are omitted in this paper. On July 11, the PV module voltage dropped from 16.25 V to 0, at around 20:18 and 21:00. This is due to the decrease of the solar irradiance at St. John’s, Canada. The PV voltage remained at zero until the next morning. At 04:50 on July 12, the PV voltage started to increase due to the sunrise, to a value of 16.3 V at 07:13. The battery voltage was decreasing until 05:46, due to the self-discharge current. After that, the battery voltage and the PV module current increased rapidly, showcasing that the PV module was charging the battery. The PV current entered a stable stage which kept the battery voltage also invariant. There are two peaks for PV current, one is for charging the battery, and the other is for charging the load.



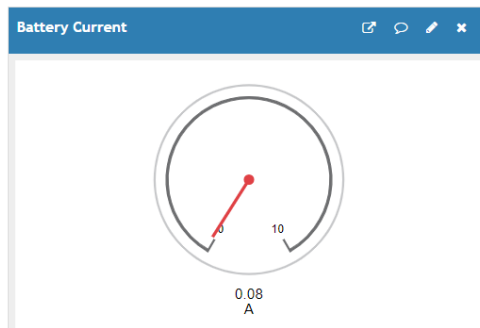
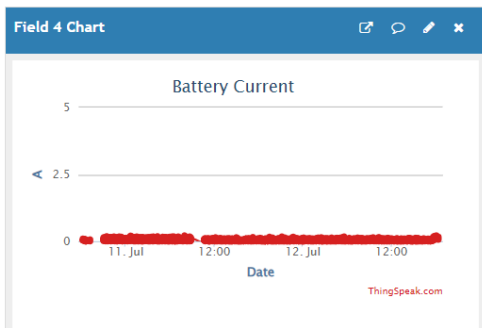
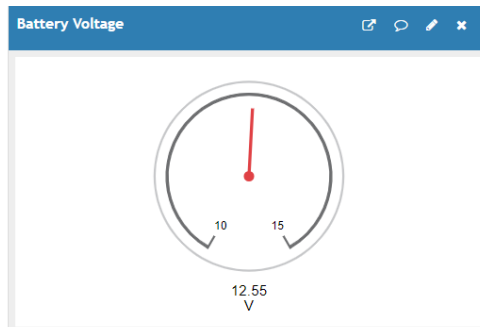
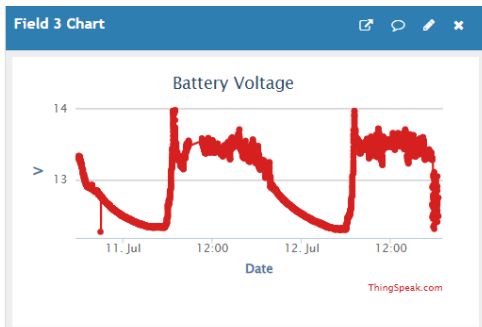
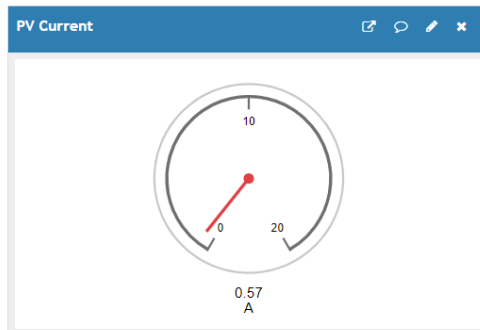
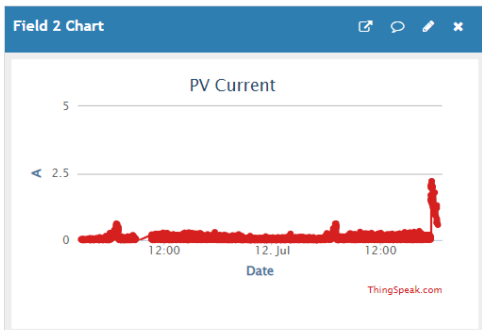
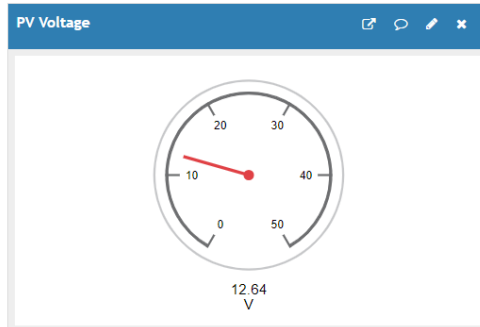
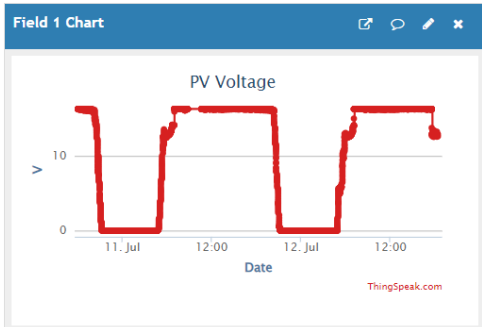
Figure 5-15 Arduino Cloud dashboards on July 11 and 12, with 1D time scale.

To test if the relay can be controlled by ESP32-E, after 17:38 on July 12, the load and the relay were integrated into the system. The relay was designed to turn on when the battery voltage is larger 13.0 V and turn off below 12.5 V. Figure 5-16 is the one hour time scale data to present the details. The load current varied between 4.42 A to 0, corresponding to the ON/OFF state of the relay. Due to the relay mechanism, the battery voltage was oscillating between around 13.1 V and 12.3 V. The battery current was negative, indicating that it was charging the load.



*Figure 5-16 Arduino Cloud dashboards on July 12, with 1H time scale.*

Furthermore, the data logging results were also displayed on ThingSpeak dashboards in Figure 5-17. The readings were presented in real time in the right column. A clear periodic pattern of the PV voltage and the battery voltage can be observed. The battery voltage was 12.55 V, which is larger than 12.5 V. The relay was still cut off, since the battery voltage must be larger than 13.0 V to turn it on. The relay state did not change when the battery voltage fell within 12.5 V and 13.0 V.



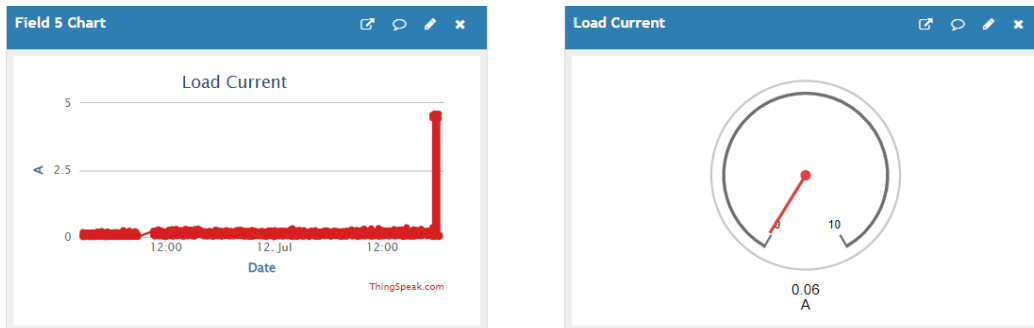


Figure 5-17 ThingSpeak dashboards displaying PV system parameters.

Figure 5-18 shows the lamp load to be turned on when the battery voltage is larger than 13.0 V.

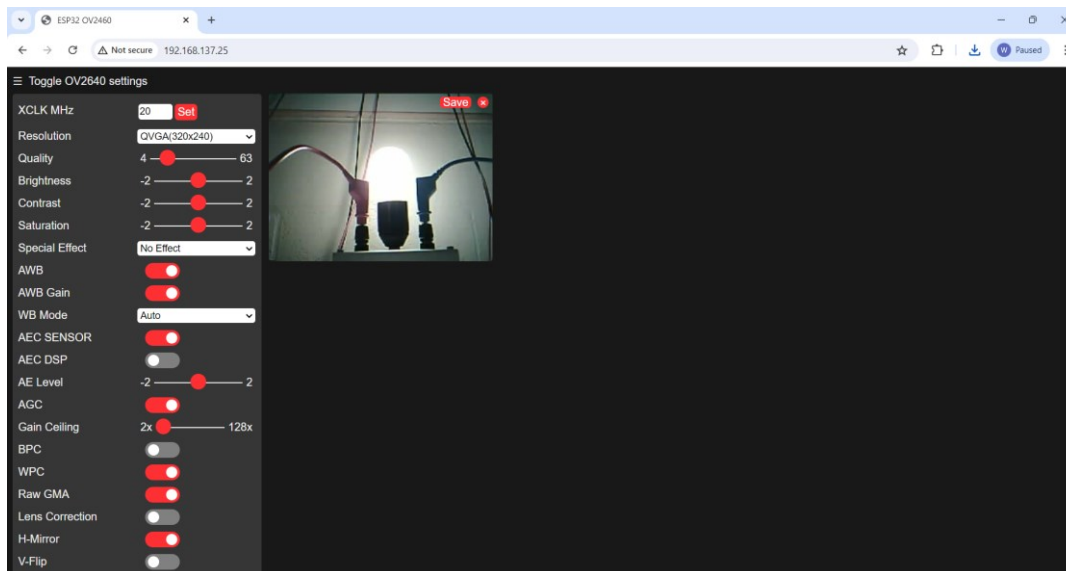


Figure 5-18 Load on when battery voltage is larger than 13.0 V.

## 5.8 Discussion and Future Work

Several features of the proposed IoT-SCADA system are listed below, providing that our system design is practical and innovative.

- **Integration of Dual IoT Platforms:** Arduino Cloud and ThingSpeak are two of the most popular IoT platforms, which facilitate easy and powerful IoT integration into SCADA systems. Communication layer, security layer, and application layer can all be implemented on the two platforms. Arduino Cloud offers exclusive benefits over ThingSpeak, including cloud programming environment, device management and OTA updates. Comparatively, ThingSpeak provides advanced data analytics, customizable dashboards, integration with MATLAB, and powerful plugins. Both cloud services retain some data that could be accessed later. Using two cloud services at the same time adds highly desirable redundancy and diversity features to the design. This design also has great potential to develop new features in the future with the development and updates of the two IoT platforms.
- **Dual Microcontrollers:** ESP32-E focuses on collecting the PV system parameters and control, which requires electrical connections with the PV system circuit through sensors and the relay. Meanwhile, ESP32-S3 is not involved in the physical connection with the PV system but handles image collection and data transmission. This separation of duties improves the system's reliability at the hardware level. Furthermore, the use of dual microcontrollers provides flexibility in monitoring tasks. While it is possible to mount the camera module on the microcontroller that collects the system parameters, this setup would significantly limit the range of inspection objects. This limitation arises from the need to align both the system's electrical ports and the inspection area simultaneously. For instance, if the inspected lamp load (or a PV panel in other scenarios) is located outdoors and the electrical ports are indoors, a single microcontroller would be insufficient to meet these requirements.

- **Real-time Monitoring and HMI design:** Arduino Cloud and ThingSpeak receive voltage and current data every 30 seconds. Fast responses to incidents can be made to prevent potential losses. Furthermore, the two platforms provide customizable and easy-to-use dashboards for HMI.
- **Data Redundancy:** When Arduino Cloud was offline, the IoT-SCADA system continued to log data on ThingSpeak, which ensures the overall system reliability and accessibility.
- **Image-based Load Monitoring:** A low-cost camera web server allows users to capture images of the load. Visual surveillance ensures that no lamp is burned out, the surrounding environment remains safe, and provides more intuitive feedback than electrical parameters alone. By observing the load image, operators obtain awareness of the load status without being onsite, saving on unnecessary technical service calls. Live feedback is a great tool to remotely oversee the monitored system.
- **Automatic Control and Alert:** A control method is employed in ESP32-E, that controls the relay and the load locally. This operation protects the battery from over-discharging automatically.
- **System Security:** Devices must provide their corresponding keys assigned by Arduino Cloud to connect to it. Moreover, Thingspeak requires the channel read/write API key to allow read/write operation. This method guarantees that only authenticated devices have access to the platforms.
- **Cloud Data Storage:** In our design, 5 messages are sent to ThingSpeak every 30 seconds. Since ThingSpeak supports 3 million messages per year free of charge, it can store up to 208 days of data.

- **Open-source:** IoT platforms, microcontrollers, and actuators are all open source. Free software guides and hardware at a low price are available on the Internet and the market. This removes the barrier of replicating this design, facilitating its promotion.
- **Low Power Consumption:** The average power consumptions of the ESP32-S3 with the camera and ESP32-E during working conditions are 0.92 W and 0.81 W, respectively. The total power consumption of the system is merely 2.38 W.

This design lays a solid foundation for the development of advanced fault diagnosis for PV systems using artificial intelligence (AI). Future research could explore the integration of AI and computer vision algorithms to analyze the data collected by the camera, enabling predictive maintenance, real-time fault detection, and automatic alerts to the end user.

## 5.9 Conclusion

Our IoT-SCADA system design comprised of a collection of sensors and the relay, ESP32-E and ESP32-S3 as the RTU and the MTU, various communication protocols including HTTP and MQTT, and Arduino Cloud with ThingSpeak as the IoT platforms. This design is demonstrated to have exceptional performance by experimental verification. The PV system was continuously monitored for 20 days, showcasing the robustness of the developed IoT-SCADA system even when Arduino Cloud was offline during July 1, 2024. Real-time monitoring of the PV system parameters, including the PV module voltage and current, the battery voltage and current, and the load current, was achieved on both Arduino Cloud and ThingSpeak platforms. The load images were also available on the camera server built on ESP32-S3. On July 11 and 12, the



recorded parameters were displayed and analyzed. The periodic pattern of the parameters can be observed in 1D time scale. When the load was turned on, the detailed changes of system parameters were analyzed in 1H time scale. Additionally, ESP32-E also controls the load to turn off when the battery voltage is below 12.5 V, ensuring the safety of the battery. The email reporting the battery status was also received from ThingSpeak. The complete system realization proves its robustness, effectiveness, versatility, and precision against its rigorous design.

## **5.10 Co-authorship Statement**

This chapter has been submitted to MDPI Electronics (ISSN: 2079-9292) by Wei He and Dr. Mohammad Tariq Iqbal. Currently it is pending editor's decision. Dr. Mohammad Tariq Iqbal is the academic supervisor of Wei He in his master's program.

All authors contributed significantly to the work presented in this section. Their individual contributions are outlined below.

Wei He: Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, and Visualization.

Mohammad Tariq Iqbal: Conceptualization, Resources, Writing - Review & Editing, Supervision, Project Administration, and Funding Acquisition.

## Chapter 6. Conclusions

### 6.1 Conclusion

Crucial operational facilities are often spread across vast, inhospitable terrains and increasingly rely on a mix of traditional and renewable energy sources, including solar PV systems. The need for an open-source and scalable system to manage these diverse energy assets is paramount. Proprietary and non-configurable SCADA systems have enabled centralized control and monitoring to some extent but present significant downsides. These systems are produced by only a few companies, resulting in high prices and maintenance costs. Moreover, compatibility with components from other manufacturers is often uncertain. Therefore, an open-source and customizable SCADA system can reduce dependence on specific vendors and improve system compatibility.

To start, emphasizing its low power consumption and affordability, an IoT data logger specifically designed for PV system monitoring was studied. The PV system in focus consists of a single PV panel, a charging controller, and a backup battery, with the primary aim of monitoring the voltages and currents of both the PV panel and the battery. To accomplish this, a sensor network with two voltage sensors and two current sensors is utilized. The collected data are stored on an SD card and dynamically displayed on a web server. The FireBeetle 2 ESP32-E microcontroller is chosen for processing sensor data due to its efficient power use in deep-sleep mode. Several low-power strategies are implemented in this study. Firstly, based on the power consumption formula for CMOS circuits, reducing the supply voltage and CPU frequency significantly lowers power consumption. For optimal performance, a supply voltage of 3500 mV and an auto-adjusted CPU frequency are selected. During flash memory save events, operating at

a minimum CPU frequency of 10 MHz reduces power consumption by 60% compared to the maximum frequency. Then, the microcontroller's deep-sleep mode is activated to deactivate non-essential components during periods of inactivity. Additionally, a data buffering mechanism stores sensor readings in the microcontroller's flash memory, transferring them to the SD card every hour. This approach is taken because writing to the SD card requires significantly more power than storing data in flash memory. Choosing one hour as the transfer interval balances efficient power usage, minimizes the risk of data loss, and accommodates future system enhancements. Another consideration is the variability in Wi-Fi connection frequencies. Thus, a critical data display interval of 14.717 seconds is established to make the two scenarios' power consumption identical. If the interval is shorter than this threshold, the deep-sleep mode is not utilized to optimize power efficiency. Monitoring events occur every 45 seconds to balance efficient power use with the shortest possible system monitoring interval. Finally, the total power consumption of the developed data logger is 122.78 mW on average. A comparative analysis with the commercial data logger DI-145 validates the functionality of this novel system. The correlation of the collected sensor data with practical observations confirms the efficacy of this low-power PV system data logger. The overall cost of the system is a modest C\$ 55.05, making it accessible to most research groups. For future suggestions and limitations, while this study demonstrates the monitoring of one PV panel and one battery with a single microcontroller, the system is scalable to monitor up to three PV panels and batteries simultaneously, potentially reducing the per-unit cost. Additionally, the power consumption obtained in this section assumes a constant supply voltage of 3500 mV, while the battery voltage decreases over time in practical applications.

Secondly, a new design of HMI and a data storage solution featuring remote, extensive, and low-cost characteristics was proposed for the SCADA system to monitor standalone PV systems. To validate the effectiveness of this design, a PV system and its SCADA system were constructed. The PV system included a PV panel, a solar charging controller, a backup battery, and a bulb. The SCADA system comprised the Arduino UNO R4 Wi-Fi as the RTU, BTeD as the HMI, PVOutput.org as the data historian, an INA3221 voltage monitor as the sensor, and a PC as the MTU. Using BTeD, the output power from the PV panel was successfully displayed in the application interface via the BLE connection between the mobile device and the RTU. A maximum period of 55 seconds was allowed for the data to be presented on the same screen. Remote and extensive data storage was achieved through PVOutput.org, where the PV panel output power and battery voltage were uploaded every five minutes via Wi-Fi. Data older than 14 days were automatically simplified by the website into daily generated energy, peak power, peak time, etc. The recorded solar power corresponded with the solar radiation pattern, and the recorded battery voltage varied according to the load status and PV output power. This novel SCADA system design proved to be effective in monitoring the standalone PV system.

Furthermore, based on a comprehensive arrangement of FIDs, an RTU and MTU, and a robust SCADA communication channel (MQTT) to track and manage data flow, another 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 April 8, 2024, in addition to regular real-time validation of system performance, testifies to the reliability of the proposed SCADA system. The supervisory

control and local control have also proven effective in experimental results. Additionally, 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 power consumption of 3.19 W—but also demonstrates versatility and precision in handling real-time data in robust environments.

Finally, a new IoT-SCADA system design consisting of a collection of sensors and a relay, with the ESP32-E and ESP32-S3 serving as the RTU and MTU, was proposed to monitor the PV system operating parameters. Various communication protocols, including HTTP and MQTT, are utilized alongside Arduino Cloud and ThingSpeak as the IoT platforms. This design has demonstrated exceptional performance through experimental verification. The PV system was continuously monitored for 20 days, showcasing the robustness of the IoT-SCADA system even when Arduino Cloud was offline on July 1, 2024. Real-time monitoring of PV system parameters—including the PV module voltage and current, battery voltage and current, and load current—was achieved on both Arduino Cloud and ThingSpeak platforms. Load images were also accessible via a camera server built on the ESP32-S3. On July 11 and 12, 2024, the recorded parameters were displayed and analyzed. The periodic pattern of the parameters was observed in a one day time scale, and detailed changes in system parameters were analyzed in a one hour time scale when the load was turned on. Additionally, the ESP32-E controls the load, turning it off when the battery voltage falls below 12.5 V to ensure battery safety. An email reporting the battery status was also received from ThingSpeak. The complete system realization proves its robustness, effectiveness, versatility, and precision, reflecting its rigorous design.

## 6.2 Research Contributions

- A low-power and low-cost data logger was designed to monitor the PV system. A reduction of supply voltage and CPU frequency, activating deep-sleep mode the microcontroller, adopting a data buffering mechanism, and the optimized monitoring interval were shown to significantly lower power consumption. The cost was as low as C\$ 55.05 due the affordable components.
- A new design of HMI and a data storage solution featuring remote, extensive, and low-cost was proposed for monitoring a standalone PV system. By using BTeD as the HMI, the output power from the PV panel was successfully displayed in the application interface via BLE. The remote and extensive data storage was achieved by PVOutput.org as the data historian. The PV panel output power and the battery voltage were uploaded to this website via Wi-Fi every five minutes.
- A SCADA system was implemented to facilitate real-time monitoring of essential parameters of the PV system and controlling 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 was successfully logged during the total solar eclipse on April 8, 2024. Furthermore, it provides easy-to-implement methods for displaying data, using the Node-RED platform to increase operational reliability.
- An IoT-SCADA system design incorporated Arduino Cloud and ThingSpeak as the two IoT platforms. The PV system was continuously monitored for 20 days, showcasing the robustness of the developed IoT-SCADA system even when Arduino Cloud was offline during July 1, 2024. Real-time monitoring of the PV system parameters was achieved on both Arduino Cloud and ThingSpeak. The email reporting the battery status was also

received from ThingSpeak. The load images were available on the camera server built on ESP32-S3.

### 6.3 Future Work

- Smart sensors and actuators that can directly communicate wireless with the MTU can be employed. In this way, no physical connection would be built between the sensors and the remote terminal microcontroller.
- Non-invasive sensor incorporation into the existing PV system can be explored.
- One unified and reliable solution that achieves local HMI, local data storage, cloud HMI, and cloud storage can be proposed.

### 6.4 List of Publications

He, W.; Iqbal, M. T. Power Consumption Minimization of a Low-Cost IoT Data Logger for Photovoltaic System. *J. Electron. Electric. Eng.* **2023**, *2*, 241 – 261.

He, W.; Iqbal, M. T. A Novel Design of a Low-Cost SCADA System for Monitoring Standalone Photovoltaic Systems. *J. Electron. Electric. Eng.* **2024**, *3*, 101 – 109.

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>

He, W.; Iqbal, M.T. An IoT-SCADA architecture for photovoltaic system monitoring, control, and inspection in real time. Manuscript submitted for publication.

## References

1. Rodrigue, J.-P. World Annual Oil Production and Peak Oil. In *The Geography of Transport Systems.* ; 2024.
2. Greenhouse Effect 101. Available online: <https://www.nrdc.org/stories/greenhouse-effect-101#what-is> (accessed on July 20, 2024.).
3. Florida's Premier Energy Research Center at the University of Central Florida. Cells, Modules, Panels and Arrays. Available online: <https://energyresearch.ucf.edu/consumer/solar-technologies/solar-electricity-basics/cells-modules-panels-and-arrays/> (accessed on July 20, 2024).
4. Lopez-Vargas, A.; Fuentes, M.; Garcia, M.V.; Munoz-Rodriguez, F.J. Low-Cost Datalogger Intended for Remote Monitoring of Solar Photovoltaic Standalone Systems Based on Arduino™. *IEEE Sensors Journal* **2019**, *19*, 4308-4320, doi:10.1109/jsen.2019.2898667.
5. Agustira, Y.M.; Ismail, N.; Rachmildha, T.D.; Fadilla, R.M.; Sartika, N.; Muharja, D. Data Logger System of Hybrid Renewable Energy System at Home-Scale. In Proceedings of the 2022 8th International Conference on Wireless and Telematics (ICWT), 21-22 July 2022, 2022; pp. 1-5.
6. Choi, C.S.; Jeong, J.D.; Lee, I.W.; Park, W.K. LoRa based renewable energy monitoring system with open IoT platform. In Proceedings of the 2018 International Conference on Electronics, Information, and Communication (ICEIC), 24-27 Jan. 2018, 2018; pp. 1-2.
7. 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), 5-8 May 2019, 2019; pp. 1-4.
8. Rouibah, N.; Barazane, L.; Mellit, A.; Hajji, B.; Rabhi, A. A low-cost monitoring system for maximum power point of a photovoltaic system using IoT technique. In Proceedings of the 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), 3-4 April 2019, 2019; pp. 1-5.
9. Deriche, M.; Raad, M.W.; Suliman, W. An IOT based sensing system for remote monitoring of PV panels. In Proceedings of the 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD), 21-24 March 2019, 2019; pp. 393-397.
10. Kantareddy, S.N.R.; Mathews, I.; Bhattacharyya, R.; Peters, I.M.; Buonassisi, T.; Sarma, S.E. Long Range Battery-Less PV-Powered RFID Tag Sensors. *IEEE Internet of Things Journal* **2019**, *6*, 6989-6996, doi:10.1109/JIOT.2019.2913403.
11. Lopez-Vargas, A.; Fuentes, M.; Vivar, M. IoT Application for Real-Time Monitoring of Solar Home Systems Based on Arduino™ With 3G Connectivity. *IEEE Sensors Journal* **2019**, *19*, 679-691, doi:10.1109/jsen.2018.2876635.
12. Hegarty, A.; Westbrook, G.; Glynn, D.; Murray, D.; Omerdic, E.; Toal, D. A Low-Cost Remote Solar Energy Monitoring System for a Buoyed IoT Ocean Observation Platform. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), 15-18 April 2019, 2019; pp. 386-391.
13. Zedak, C.; Lekbich, A.; Belfqih, A.; Boukherouaa, J.; Haidi, T.; Mariami, F.E. A proposed secure remote data acquisition architecture of photovoltaic systems based on the Internet of Things. In Proceedings of the 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), 10-12 May 2018, 2018; pp. 1-5.



14. Purwadi, A.; Haroen, Y.; Farianza Yahya, A.; Heryana, N.; Nurafiat, D.; Assegaf, A. Prototype development of a Low Cost data logger for PV based LED Street Lighting System. In Proceedings of the Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, 17-19 July 2011, 2011; pp. 1-5.
15. Sahraei, N.; Looney, E.E.; Watson, S.M.; Peters, I.M.; Buonassisi, T. Adaptive power consumption improves the reliability of solar-powered devices for internet of things. *Applied Energy* **2018**, *224*, 322-329, doi:<https://doi.org/10.1016/j.apenergy.2018.04.091>.
16. Mikhaylov, K.; Tervonen, J. Optimization of microcontroller hardware parameters for Wireless Sensor Network node power consumption and lifetime improvement. In Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems, 18-20 Oct. 2010, 2010; pp. 1150-1156.
17. Gupta, V.; Raj, P.; Yadav, A. Investigate the effect of dust deposition on the performance of solar PV module using LABVIEW based data logger. In Proceedings of the 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 21-22 Sept. 2017, 2017; pp. 742-747.
18. Jesha, T.A.; Iqbal, M.T. Data logging and energy consumption analysis of two houses in St. John's, Newfoundland. In Proceedings of the 8th International Conference on Electrical and Computer Engineering, 20-22 Dec. 2014, 2014; pp. 816-819.
19. Raskovic, D.; Giessel, D. Dynamic Voltage and Frequency Scaling For On-Demand Performance and Availability of Biomedical Embedded Systems. *IEEE Transactions on Information Technology in Biomedicine* **2009**, *13*, 903-909, doi:10.1109/TITB.2009.2030181.
20. Bradley, L.J.; Wright, N.G. Optimising SD Saving Events to Maximise Battery Lifetime for Arduino™/Atmega328P Data Loggers. *IEEE Access* **2020**, *8*, 214832-214841, doi:10.1109/ACCESS.2020.3041373.
21. Yokwana, X.P.; Yusuff, A.A.; Moseitlhe, T.C. Faults in a Large-scale photovoltaic installation: A Review. In Proceedings of the 2020 6th IEEE International Energy Conference (ENERGYCon), 28 Sept.-1 Oct. 2020, 2020; pp. 474-478.
22. NEWS RELEASE: Canada added 1.8 GW of wind and solar in 2022. Available online: <https://renewablesassociation.ca/news-release-canada-added-1-8-gw-of-wind-and-solar-in-2022/> (accessed on 15 August).
23. Madeti, S.R.; Singh, S.N. Monitoring system for photovoltaic plants: A review. *Renewable and Sustainable Energy Reviews* **2017**, *67*, 1180-1207, doi:<https://doi.org/10.1016/j.rser.2016.09.088>.
24. Gmbh, D. ADL-MXSpro Multifunctional solar data logger. Available online: <https://www.meier-nt.de/en/photovoltaics/products/adl-mxspro-en> (accessed on Sep. 15).
25. International, F. FRONIUS DATAMANAGER 2.0. Available online: <https://www.fronius.com/en-us/usa/solar-energy/installers-partners/technical-data/all-products/system-monitoring/hardware/fronius-datamanager-2-0/fronius-datamanager-2-0> (accessed on Sep. 15).
26. Gmbh, G.I.E.S. Q.reader 602 + Extension 01. Available online: [https://www.gantner-environment.com/fileadmin/user\\_upload/q.reader\\_602\\_20190928.pdf](https://www.gantner-environment.com/fileadmin/user_upload/q.reader_602_20190928.pdf) (accessed on Sep. 15).
27. HUAWEI. SmartLogger3000A. Available online: <https://solar.huawei.com/-/media/Solar/attachment/pdf/en/datasheet/SmartLogger3000A.pdf> (accessed on Sep. 15).

28. energy, K.n. Powador-proLOG Operating instructions. Available online: <https://kaco-newenergy.com/index.php?eID=dumpFile&t=f&f=6615&token=fda0ce4ca172c6c9356974075baa80235ae7af27> (accessed on Sep. 15).
29. GmbH, D.E.S. SOLIVIA Gateway M1 G2 Operation and installation manual. Available online: <https://support.delta-es.com.au/wp-content/uploads/2020/03/Manual-SOLIVIA-Gateway.pdf> (accessed on Sep. 15).
30. Technology, S.S. SMA Data Manager M. Available online: <https://files.sma.de/downloads/EDMM-10-DS-en-40.pdf> (accessed on Sep. 15).
31. GmbH, S.D. Solar-Log Manul V.3.6.0.G. Available online: [https://www.europe-solarstore.com/download/solarlog/SolarLog\\_products\\_manual.pdf](https://www.europe-solarstore.com/download/solarlog/SolarLog_products_manual.pdf) (accessed on Sep. 15).
32. GmbH, S.P. MaxWeb XPN Installation instructions. Available online: <https://docplayer.net/65483810-Maxweb-xpn-installation-instructions.html> (accessed on Sep. 15).
33. López-Vargas, A.; Fuentes, M.; Vivar, M. IoT Application for Real-Time Monitoring of Solar Home Systems Based on Arduino™ With 3G Connectivity. *IEEE Sensors Journal* **2019**, *19*, 679-691, doi:10.1109/JSEN.2018.2876635.
34. Kalay, M.Ş.; Kılıç, B.; Mellit, A.; Oral, B.; Sağlam, Ş. IoT-Based Data Acquisition and Remote Monitoring System for Large-Scale Photovoltaic Power Plants. In Proceedings of the Proceedings of the 3rd International Conference on Electronic Engineering and Renewable Energy Systems, Singapore, 2023//, 2023; pp. 631-639.
35. DFRobot. FireBeetle 2 ESP32-E IoT Microcontroller. Available online: <https://www.dfrobot.com/product-2195.html> (accessed on April 12, 2024).
36. FireBeetle ESP32 IOT Microcontroller (Supports Wi-Fi & Bluetooth) SKU: DFR0478. Available online: <https://www.application-datasheet.com/pdf/dfrobot/dfr0478.pdf> (accessed on August 18).
37. esp32-adc-calibrate. Available online: <https://github.com/e-tinkers/esp32-adc-calibrate> (accessed on August 14).
38. Vangal, S.; Paul, S.; Hsu, S.; Agarwal, A.; Kumar, S.; Krishnamurthy, R.; Krishnamurthy, H.; Tschanz, J.; De, V.; Kim, C.H. Wide-Range Many-Core SoC Design in Scaled CMOS: Challenges and Opportunities. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2021**, *29*, 843-856, doi:10.1109/TVLSI.2021.3061649.
39. Crowe, J.; Hayes-Gill, B. 9 - Choosing a means of implementation. In *Introduction to Digital Electronics*, Crowe, J., Hayes-Gill, B., Eds.; Newnes: Oxford, 1998; pp. 191-239.
40. Lukas, M.B.; Christian, H.; Dominik, K. IoT on an ESP32: Optimization Methods Regarding Battery Life and Write Speed to an SD-Card. In *Edge Computing*, Sam, G., Ed.; IntechOpen: Rijeka, 2023; p. Ch. 7.
41. Bouzguenda, M.; Chtourou, S.; Alarfaj, M.; Sumsudeen, R.M.; Shwehdi, M. Arduino Uno Wi-Fi DeMilitarized Zone-based monitoring of solar photovoltaic systems. *Measurement and Control* **2022**, *55*, 136-145, doi:10.1177/00202940221090553.
42. Khaleel, K.; Atyia, T.H.; Al-Naib, A.M.I. Design and Development of Real-Time Data Acquisition of Photovoltaic Panel Parameters via IoT. *NTU Journal of Renewable Energy* **2022**, *3*, 1-8.
43. Inner, B. Data monitoring system for solar panels with bluetooth. In Proceedings of the 2017 25th Signal Processing and Communications Applications Conference (SIU), 15-18 May 2017, 2017; pp. 1-4.
44. C, B.; H, M.-A.; H. B. A, S.; A. Bennani-Ben, A.; I, S.-B.; H, S. A real time, wireless and low cost data acquisition system for residential PV modules. In Proceedings of the 2020

- 6th IEEE International Energy Conference (ENERGYCon), 28 Sept.-1 Oct. 2020, 2020; pp. 417-422.
45. Nordic Semiconductor. Power Profiler Kit II. Available online: [https://docs.nordicsemi.com/bundle/ug\\_ppk2/page/UG/ppk/PPK\\_user\\_guide\\_Intro.html](https://docs.nordicsemi.com/bundle/ug_ppk2/page/UG/ppk/PPK_user_guide_Intro.html) (accessed on September 11, 2024).
  46. Beddows, P.A.; Mallon, E.K. Cave Pearl Data Logger: A Flexible Arduino-Based Logging Platform for Long-Term Monitoring in Harsh Environments. *Sensors* **2018**, *18*, doi:10.3390/s18020530.
  47. DI-145 User's Manual. Available online: <https://www.dataq.com/resources/pdfs/manuals/145-manual.pdf> (accessed on September 2).
  48. Cumulative installed solar PV capacity worldwide from 2000 to 2022 (in megawatts) [Graph]. **2023**.
  49. Maldonado-Correa, J.; Martín-Martínez, S.; Artigao, E.; Gómez-Lázaro, E. Using SCADA Data for Wind Turbine Condition Monitoring: A Systematic Literature Review. *Energies* **2020**, *13*, doi:10.3390/en13123132.
  50. Allafi, I.; Iqbal, T. Low-Cost SCADA System Using Arduino and Reliance SCADA for a Stand-Alone Photovoltaic System. *Journal of Solar Energy* **2018**, *2018*, 3140309, doi:10.1155/2018/3140309.
  51. Le, P.T.; Tsai, H.L.; Le, P.L. Development and Performance Evaluation of Photovoltaic (PV) Evaluation and Fault Detection System Using Hardware-in-the-Loop Simulation for PV Applications. *Micromachines* **2023**, *14*, doi:10.3390/mi14030674.
  52. Didi, Z.; El Azami, I. Experimental Analysis and Monitoring of Photovoltaic Panel Parameters. *International Journal of Advanced Computer Science and Applications* **2023**, *14*, 151-157, doi:10.14569/IJACSA.2023.0140219.
  53. BetterTools. Bluetooth Terminal eDebugger. Available online: <https://play.google.com/store/apps/details?id=com.e.debugger&hl=en> (accessed on September 11, 2024.).
  54. Tan, L. A Survey on Internet of Things for Smart Health Technologies. 2018.
  55. PVOutput.org. Available online: (accessed on 02/12).
  56. Omid, 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*, doi:10.3390/en16052092.
  57. Lawrence, O.A.; Iqbal, M.T. Design and implementation of a low-cost, open source IoT-based SCADA system using ESP32 with OLED, ThingsBoard and MQTT protocol. *AIMS Electronics and Electrical Engineering* **2020**, *4*, 57-86, doi:10.3934/ElectrEng.2020.1.57.
  58. Ahsan, L.; Baig, M.J.; Iqbal, M.T. Low-Cost, Open-Source, Emoncms-Based SCADA System for a Large Grid-Connected PV System. *Sensors* **2022**, *22*, doi:10.3390/s22186733.
  59. Lunstad, N.T.; Sowby, R.B. Smart Irrigation Controllers in Residential Applications and the Potential of Integrated Water Distribution Systems. *Journal of Water Resources Planning and Management* **2024**, *150*, doi:10.1061/JWRMD5.WRENG-5871.
  60. 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. *Environmental Science and Pollution Research* **2024**, *31*, 19085-19104, doi:10.1007/s11356-024-32435-6.

61. Kumar Dalapati, G.; Ghosh, S.; Sherin P A, T.; Ramasubramanian, B.; Samanta, A.; Rathour, A.; Kin Shun Wong, T.; Chakraborty, S.; Ramakrishna, S.; Kumar, A. Maximizing solar energy production in ASEAN region: Opportunity and challenges. *Results in Engineering* **2023**, *20*, doi:10.1016/j.rineng.2023.101525.
62. Yadav, G.; Paul, K. Architecture and security of SCADA systems: A review. *International Journal of Critical Infrastructure Protection* **2021**, *34*, 100433, doi:<https://doi.org/10.1016/j.ijcip.2021.100433>.
63. Alanazi, M.; Mahmood, A.; Chowdhury, M.J.M. SCADA vulnerabilities and attacks: A review of the state-of-the-art and open issues. *Computers and Security* **2023**, *125*, doi:10.1016/j.cose.2022.103028.
64. 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*, doi:10.3390/electronics13040782.
65. Babayigit, B.; Abubaker, M. Industrial Internet of Things: A Review of Improvements Over Traditional SCADA Systems for Industrial Automation. *IEEE Systems Journal* **2024**, *18*, 120-133, doi:10.1109/JSYST.2023.3270620.
66. He, W.; Iqbal, M.T. Power Consumption Minimization of a Low-Cost IoT Data Logger for Photovoltaic System. *Journal of Electronics and Electrical Engineering* **2023**, doi:10.37256/jee.2220233795.
67. Jose, J.; Mathew, V. Internet of Things - A Model for Data Analytics of KPI Platform in Continuous Process Industry. *Informatika (Slovenia)* **2024**, *48*, 119-130, doi:10.31449/inf.v48i1.3826.
68. Abdelatti, M.; Sodhi, M. Lab-Scale Smart Factory Implementation Using ROS. In *Robot Operating System (ROS): The Complete Reference (Volume 7)*, Koubaa, A., Ed.; Springer International Publishing: Cham, 2023; pp. 119-143.
69. 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 Transactions on Industry Applications* **2023**, *59*, 4931-4941, doi:10.1109/TIA.2023.3271618.
70. 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), 11-13 Dec. 2019, 2019; pp. 1-6.
71. Ahmed, O.; Iqbal, M.T. Remote Monitoring, Control and Data Visualization for a Solar Water Pumping System. *European Journal of Electrical Engineering and Computer Science* **2023**, *7*, 71-77, doi:10.24018/ejece.2023.7.5.552.
72. 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 - Advances in Electrical Engineering, Electronics and Energy* **2023**, *6*, 100348, doi:<https://doi.org/10.1016/j.prime.2023.100348>.
73. 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. *International Journal of Electrical Power & Energy Systems* **2023**, *151*, 109198, doi:<https://doi.org/10.1016/j.ijepes.2023.109198>.
74. Aghenta, L.O.; Iqbal, M.T. Low-Cost, Open Source IoT-Based SCADA System Design Using Thingier.IO and ESP32 Thing. *Electronics* **2019**, *8*, doi:10.3390/electronics8080822.

75. 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), 2-3 Nov. 2022, 2022; pp. 55-60.
76. 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), 15-16 Sept. 2022, 2022; pp. 1-4.
77. 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), 14-16 June 2023, 2023; pp. 949-953.
78. Matsui, T.; Yamamoto, K.; Sumi, S.; Triruttanapiruk, N. Detection of Lightning Damage on Wind Turbine Blades Using the SCADA System. *IEEE Transactions on Power Delivery* **2021**, *36*, 777-784, doi:10.1109/TPWRD.2020.2992796.
79. 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, doi:10.1109/ACCESS.2023.3326751.
80. 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, doi:10.1109/ACCESS.2022.3211288.
81. Pliatsios, D.; Sarigiannidis, P.; Lagkas, T.; Sarigiannidis, A.G. A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics. *IEEE Communications Surveys & Tutorials* **2020**, *22*, 1942-1976, doi:10.1109/COMST.2020.2987688.
82. Li, S.; Jiang, B.; Wang, X.; Dong, L. Research and Application of a SCADA System for a Microgrid. *Technologies* **2017**, *5*, doi:10.3390/technologies5020012.
83. Baig, M.J.; Iqbal, M.T.; Jamil, M.; Khan, J. A Low-Cost, Open-Source Peer-to-Peer Energy Trading System for a Remote Community Using the Internet-of-Things, Blockchain, and Hypertext Transfer Protocol. *Energies* **2022**, *15*, doi:10.3390/en15134862.
84. Baig, M.J.; Iqbal, M.T.; Jamil, M.; Khan, J. Blockchain-Based Peer-to-Peer Energy Trading System Using Open-Source Angular Framework and Hypertext Transfer Protocol. *Electronics* **2023**, *12*, doi:10.3390/electronics12020287.
85. Nițulescu, I.-V.; Korodi, A. Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions. *IoT* **2020**, *1*, 76-91, doi:10.3390/iot1010005.
86. BananaPi. BananaPi BPI-M4 Berry Documentation. Available online: [https://docs.banana-pi.org/en/BPI-M4\\_Berry/BananaPi\\_BPI-M4\\_Berry](https://docs.banana-pi.org/en/BPI-M4_Berry/BananaPi_BPI-M4_Berry) (accessed on 22 April, 2024).
87. Node-RED. Node-RED User Guide - Concepts. Available online: <https://nodered.org/docs/user-guide/concepts> (accessed on April 12, 2024).
88. SparkFun Electronics. ACS712 Datasheet. Available online: <https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf> (accessed on July 5, 2024).
89. Ahelerooff, S.; Xu, X.; Lu, Y.; Aristizabal, M.; Pablo Velásquez, J.; Joa, B.; Valencia, Y. IoT-enabled smart appliances under industry 4.0: A case study. *Advanced Engineering Informatics* **2020**, *43*, 101043, doi:<https://doi.org/10.1016/j.aei.2020.101043>.



90. Stavropoulos, T.G.; Papastergiou, A.; Mpaltadoros, L.; Nikolopoulos, S.; Kompatsiaris, I. IoT Wearable Sensors and Devices in Elderly Care: A Literature Review. *Sensors* **2020**, *20*, doi:10.3390/s20102826.
91. Franco, J.; Aris, A.; Canberk, B.; Uluagac, A.S. A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems. *IEEE Communications Surveys & Tutorials* **2021**, *23*, 2351-2383, doi:10.1109/COMST.2021.3106669.
92. Wang, Q.; Zhu, X.; Ni, Y.; Gu, L.; Zhu, H. Blockchain for the IoT and industrial IoT: A review. *Internet of Things* **2020**, *10*, 100081, doi:<https://doi.org/10.1016/j.iot.2019.100081>.
93. Rejeb, A.; Rejeb, K.; Simske, S.; Treiblmaier, H.; Zailani, S. The big picture on the internet of things and the smart city: a review of what we know and what we need to know. *Internet of Things* **2022**, *19*, 100565, doi:<https://doi.org/10.1016/j.iot.2022.100565>.
94. Sajid, A.; Abbas, H.; Saleem, K. Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges. *IEEE Access* **2016**, *4*, 1375-1384, doi:10.1109/ACCESS.2016.2549047.
95. Duair, J.J.; Majeed, A.I.; Ali, G.M. Design and Implementation of IoT-Based SCADA for a Multi Microgrid System. *ECS Transactions* **2022**, *107*, 17345, doi:10.1149/10701.17345ecst.
96. Alhasnawi, B.N.; Jasim, B.H.; Alhasnawi, A.N.; Sedhom, B.E.; Jasim, A.M.; Khalili, A.; Bureš, V.; Burgio, A.; Siano, P. A Novel Approach to Achieve MPPT for Photovoltaic System Based SCADA. *Energies* **2022**, *15*, doi:10.3390/en15228480.
97. de Arquer Fernández, P.; Fernández Fernández, M.Á.; Carús Candás, J.L.; Arboleya Arboleya, P. An IoT open source platform for photovoltaic plants supervision. *International Journal of Electrical Power & Energy Systems* **2021**, *125*, 106540, doi:<https://doi.org/10.1016/j.ijepes.2020.106540>.
98. Silva, F.M.Q.; Filho, B.J.C.; Pires, I.A.; Maia, T.A.C. Design of a SCADA System Based on Open-Source Tools. In Proceedings of the 2021 14th IEEE International Conference on Industry Applications (INDUSCON), 15-18 Aug. 2021, 2021; pp. 1323-1328.
99. Melo, G.C.; Torres, I.C.; Araújo, Í.B.; Brito, D.B.; Barboza, E.D. A Low-Cost IoT System for Real-Time Monitoring of Climatic Variables and Photovoltaic Generation for Smart Grid Application. *Sensors* **2021**, *21*, doi:10.3390/s21093293.
100. Hoarcã, I.C. Energy management for a photovoltaic power plant based on SCADA system. In Proceedings of the 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 1-3 July 2021, 2021; pp. 1-9.
101. Vujović, I., Mladen Koprivica, and Željko Đurišić. Centralized controlling of distributed PV systems using cloud and IoT technologies. *Telfor Journal* **2023**, *15*, 6, doi:<https://doi.org/10.5937/telfor2302038V>.
102. Sarkar, S.; Rao, K.U.; Bhargav, J.; Sheshaprasad, S.; C.A, A.S. IoT Based Wireless Sensor Network (WSN) for Condition Monitoring of Low Power Rooftop PV Panels. In Proceedings of the 2019 IEEE 4th International Conference on Condition Assessment Techniques in Electrical Systems (CATCON), 21-23 Nov. 2019, 2019; pp. 1-5.
103. Dupont, I.M.; Carvalho, P.C.M.; Jucá, S.C.S.; Neto, J.S.P. Novel methodology for detecting non-ideal operating conditions for grid-connected photovoltaic plants using Internet of Things architecture. *Energy Conversion and Management* **2019**, *200*, 112078, doi:<https://doi.org/10.1016/j.enconman.2019.112078>.

104. Seflahir, D.; Ahmad Faisal Mohamad, A.; Aliashim, A.; Abdurahman; Nurul, H.; Rivaldi, K.; Ojak Abdul, R. Real-time Analysis of Inverter Performance via SCADA Haiwell Online Monitoring. *Journal of Advanced Research in Applied Sciences and Engineering Technology* **2024**, *37*, 99-114, doi:10.37934/araset.37.1.99114.
105. Mellit, A.; Benghanem, M.; Kalogirou, S.; Massi Pavan, A. An embedded system for remote monitoring and fault diagnosis of photovoltaic arrays using machine learning and the internet of things. *Renewable Energy* **2023**, *208*, 399-408, doi:<https://doi.org/10.1016/j.renene.2023.03.096>.
106. Khelil, A.; Germanus, D.; Suri, N. Protection of SCADA Communication Channels. In *Critical Infrastructure Protection: Information Infrastructure Models, Analysis, and Defense*, Lopez, J., Setola, R., Wolthusen, S.D., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp. 177-196.
107. He, W.; Baig, M.J.; 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*, doi:10.3390/en17102295.
108. Cui, Y.; Liu, M.; Li, W.; Lian, J.; Yao, Y.; Gao, X.; Yu, L.; Wang, T.; Li, Y.; Yin, J. An exploratory framework to identify dust on photovoltaic panels in offshore floating solar power stations. *Energy* **2024**, *307*, doi:10.1016/j.energy.2024.132559.
109. Yang, M.; Javed, W.; Guo, B.; Ji, J. Estimating PV Soiling Loss Using Panel Images and a Feature-Based Regression Model. *IEEE Journal of Photovoltaics* **2024**, *14*, 661-668, doi:10.1109/JPHOTOV.2024.3388168.
110. David, M.; Alonso-Montesinos, J.; Le Gal La Salle, J.; Lauret, P. Probabilistic Solar Forecasts as a Binary Event Using a Sky Camera. *Energies* **2023**, *16*, doi:10.3390/en16207125.
111. Espressif Systems. ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet. Available online: [https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1\\_wroom-1u\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf) (accessed on July 3, 2024).
112. Freenove ESP32-S3-WROOM CAM Board (Compatible with Arduino IDE), Onboard Camera Wireless, Python C Code, Detailed Tutorial, Example Projects. Available online: <https://www.amazon.ca/Freenove-ESP32-S3-WROOM-Compatible-Wireless-Detailed/dp/B0BMQ8F7FN> (accessed on July 3, 2024).
113. Espressif Systems. ESP32-WROOM-32E ESP32-WROOM-32UE Datasheet. Available online: [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e\\_esp32-wroom-32ue\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf) (accessed on July 4, 2024).
114. Geekstory Voltage Tester Sensor Measurement Detection Module DC 0-25V Terminal Sensor Module for Arduino UNO Mega Robot Smart Car Geekstory (Pack of 10). Available online: <https://www.amazon.ca/Voltage-Measurement-Detection-Arduino-Geekstory/dp/B07FVVSYYH> (accessed on June 5, 2024).
115. OmniVision. OV2640 Color CMOS UXGA (2.0 MegaPixel) CameraChip with OminiPixel2 Technology. Available online: [https://www.uctronics.com/download/cam\\_module/OV2640DS.pdf](https://www.uctronics.com/download/cam_module/OV2640DS.pdf) (accessed on July 5, 2024).
116. Arduino Cloud. Scheduled Maintenance for Arduino Cloud. Available online: <https://status.arduino.cc/incidents/3zbnwd7k7vl> (accessed on July 12, 2024).