Journal of Electronics and Electrical Engineering
https://ojs.wiserpub.com/index.php/JEEE/

Article

Power Consumption Minimization of a Low-Cost IoT Data Logger for Photovoltaic System

Wei He and Mohammad Tariq Iqbal*

Department of Electrical and Computer Engineering, Memorial University of Newfoundland, St. John's, NL, Canada
E-mail: tariq@mun.ca

Received: 19 October 2023; **Revised:** 26 November 2023; **Accepted:** 28 November 2023

Abstract: This paper introduces an innovative IoT-based data logger for photovoltaic (PV) system monitoring, emphasizing low power consumption and affordability. The system comprises a PV panel, a charging controller, and a backup battery, focusing on monitoring their voltages and currents through a network of voltage and current sensors. Data is stored on an SD card and displayed in real-time on a web server. The FireBeetle 2 ESP32-E microcontroller, chosen for its efficient deep-sleep mode power management, is central to the data logger's design. This study employs several low-power strategies, notably reducing supply voltage and CPU frequency to decrease power consumption significantly. A data buffering mechanism stores sensor readings in the microcontroller's flash memory, transferring them to the SD card hourly to balance power efficiency and data security. Wi-Fi connection intervals are optimized to 45 seconds, balancing power use and system monitoring frequency. The data logger averages a power consumption of 122.78 mW and demonstrates its efficacy against the commercial DI-145 model. Priced at C$ 55.05, the system is both cost-effective and scalable, capable of monitoring multiple PV panels and batteries, reducing per-unit costs. The study underscores the successful integration of affordability, low-power operation, and efficient monitoring in a PV system data logger, showcasing its potential in future renewable energy research.

Keywords: data logger, low power, ESP32, low cost, Internet of Things, ubiquitous sensor network, photovoltaic system

1. Introduction

Due to Earth's limited fossil fuel reserves, renewable energy has emerged as a vital alternative power source [1]. 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 [2]. 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

DOI: https://doi.org/10.37256/jeee.2220233795

performance [3]. 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 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 [4]. 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 1. Common commercial PV system data loggers.

Product and Company	Power Consumption	Cost range (CAD) (Sensors not included)
ADL-MXSpro by Meier-NT GmbH [4]	0.1 -0.76 W	1410-1583
Fronius Datamanager 2.0 by Fronius [5]	< 2 W	319-373
Q.reader by Gantner Instruments GmbH [6]	5 W	863
Smartlogger 3000A by Huawei [7]	8 W	2151
Powador-proLOG M by Kaco [8]	\	790-955
Delta Solivia Gateway M1 G2 by Delta Energy Systems GmbH [9]	1 W @ 5 V	250
SMA Data Manager M Lite by SMA Solar Technology AG [10]	4 W	410
Solar-Log 2000 by Solare Datensysteme GmbH [11]	3 W	1483-1796
MaxWeb XPN by SolarMax [12]	24 W (maximum)	719-863

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 [13], 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 [14] 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 [15] 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 [13], 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 [16], 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 mW/cm2. 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 [17], 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 [18], 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 paper is structured as follows: Section 2 details the monitoring system architecture and introduces the data logger's components such as the ESP32, voltage sensor, current sensor, and microSD card. Section 3 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 Section 4, 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. Section 5 presents experimental results, including data storage on the SD card and real-time monitoring capabilities via a web portal. The paper concludes with a summary in Section 6.

2. Data Logger

2.1 *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 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.

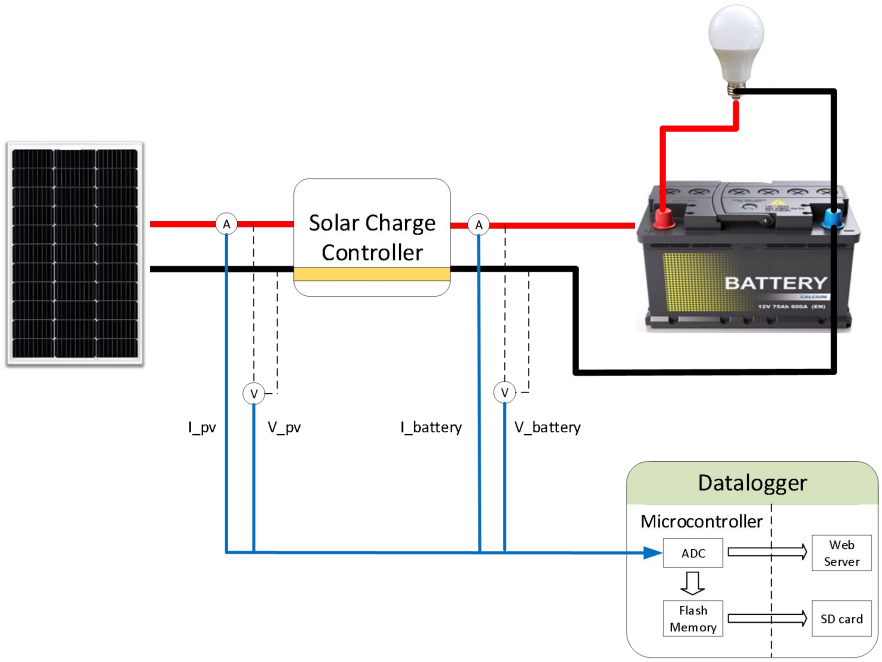


Figure 1. Connection diagram of PV system and developed data logger.

2.2 *Microcontroller FireBeetle 2 ESP32-E*

The microcontroller used in this paper 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 the ideal candidate for IoT applications. ESP32 microcontrollers made by different vendors may have different power consumptions mainly due to the choice of on-board voltage regulator. DFRobot FireBeetle ESP32 microcontroller is a low-power consumption microcontroller designed for IoT projects. 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 10μA electric current in deep-sleep mode and the main controller supports USB and 3.7V 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 paper partly due to its extremely low power consumption during power saving modes. In deep-sleep mode, consumed electric current is only 10μA according to the datasheet [19]. Arduino UNO is a classic microcontroller board based on the ATmega328P, and it is used in many research [13,17,20]. Table 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 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 with a knife 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 μA.

Reasons that FireBeetle 2 ESP32-E is selected as the microcontroller are justified below. From Table 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.

Table 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

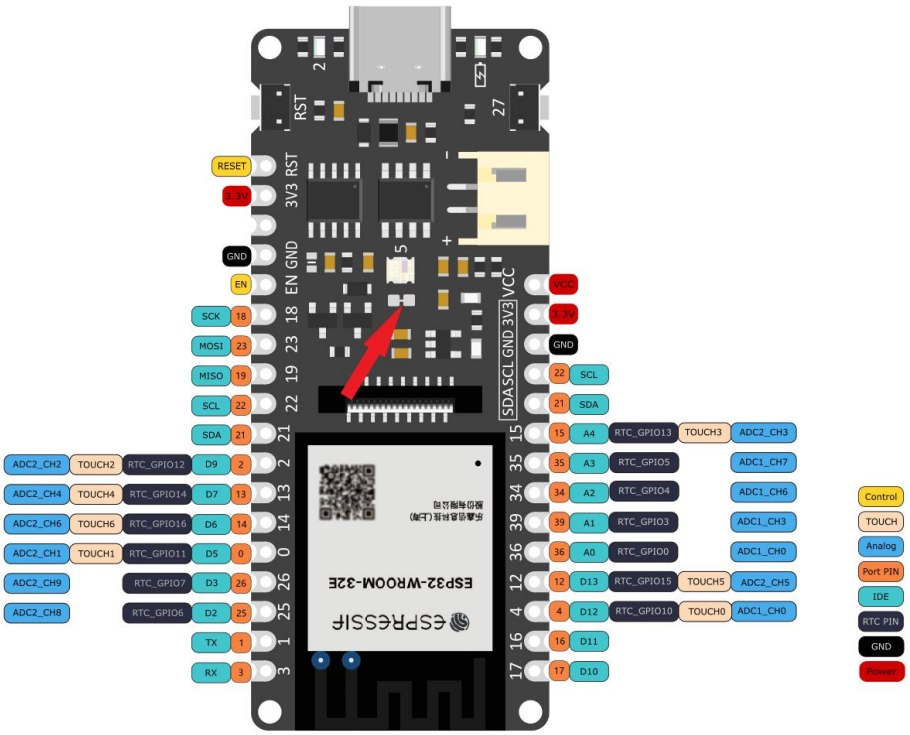


Figure 2. FireBeetle 2 ESP32-E pinout. Cut off the small line pointed by the red arrow to disable onboard RGB LED to reduce power consumption.

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 [21].

2.3 *Peripherals*

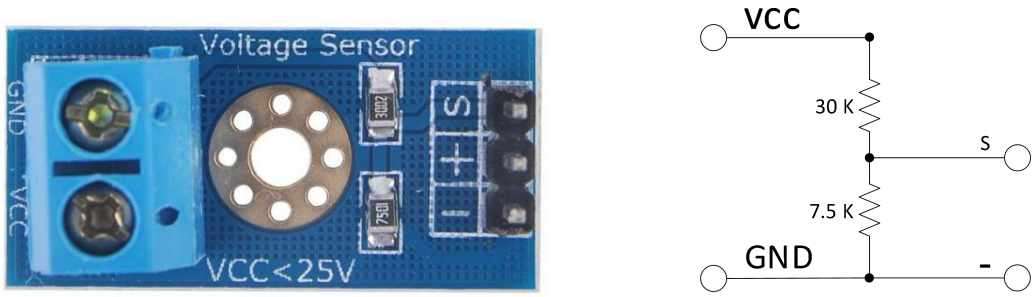
In this section the peripherals connected with the FireBeetle 2 ESP32-E are introduced. Voltage sensors and current sensors sample the readings regularly, and the SD card is responsible for storing these sensor readings.

2.3.1 Voltage Sensor

The HiLetgo voltage detection module DC 0-25V is based on resistive voltage divider to measure the voltage. Figure 3a is the real image of the voltage sensor, and Figure 3b refers to the working principle diagram of it. VCC and GND are ports of the voltage to be measured. Two resistors, 30K Ohms and 7.5K Ohms, 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 7.5K/ (7.5K+30K) = 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 KΩ 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 volts.

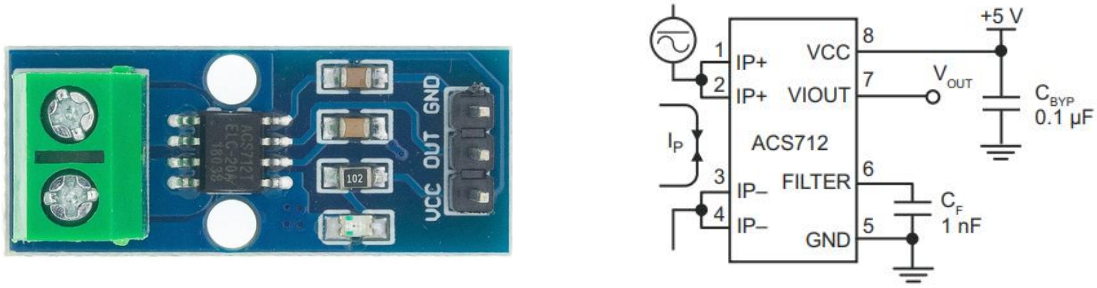


(a) (b)

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

2.3.2 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 4a is the image of ACS712 current sensor with ±20 A range. The connection schematic is illustrated in Figure 4b. 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 volts are connected with pin 8, and GND is attached to pin 5. Pin 7 stands for Vout signal and should be hooked up with analog pin of the microcontroller. For ESP32 of which the nominal voltage is 3.3 volts, the Vcc and the ground pin of ACS712 sensor should be connected with 5 volts 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 volts.



(a) (b)

Figure 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 ±5 A, ±20 A, and ±30 A. The supply voltage to ACS712 series is all 5 volts, with a fluctuation of ±0.5 volt. Different types of ACS712 current sensor has 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 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 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 SD Card Transflash Breakout

Figure 5 is the demonstration of 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 MISO pin (GPIO 19) of the microcontroller; DI is related to 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 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.

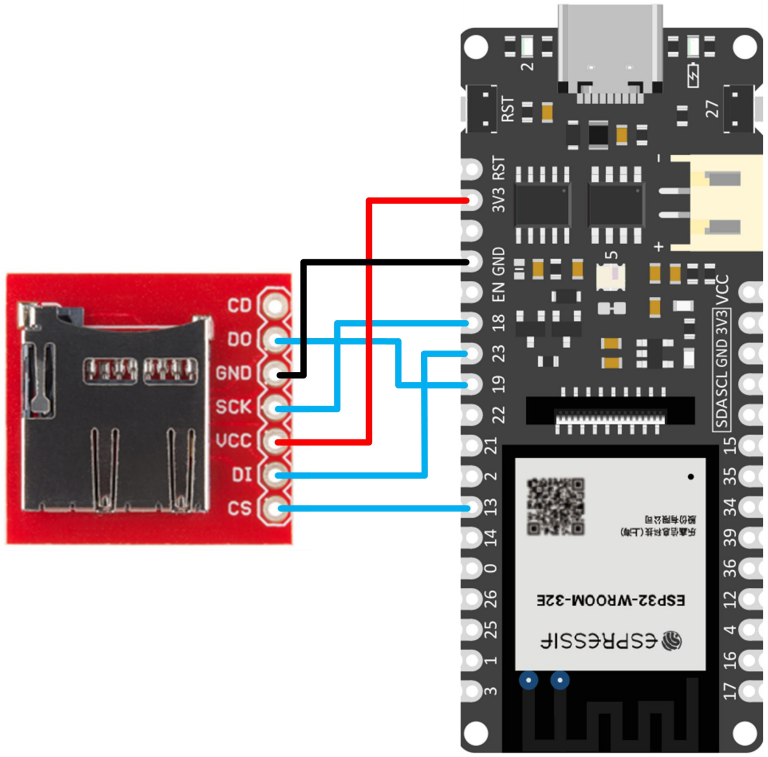


Figure 5. SD card transflash breakout connection diagram with FireBeetle 2 ESP32-E.

3. 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. 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 [22] it is expressed as

$$p = CV^2 f \tag{1}$$

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.

3.1 *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 2.2V to 3.3V to Firebeetle 2 ESP32-E. It is achieved by providing V+ and Ground from AB-3300 to 3.3V 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 6 demonstrates that power consumption does decrease with lower voltage, while it does not follow the relationship stated in Equation 1 but in a linear way. The reason is it is tested with a complete board including Tensilica LX6 dual-core processor and onboard peripherals. Besides, when supply voltage drops below 2.6V, 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 ostensible power saving since the system stability is compromised. For battery powered data logger, a suitable battery will extend the replacement period significantly.

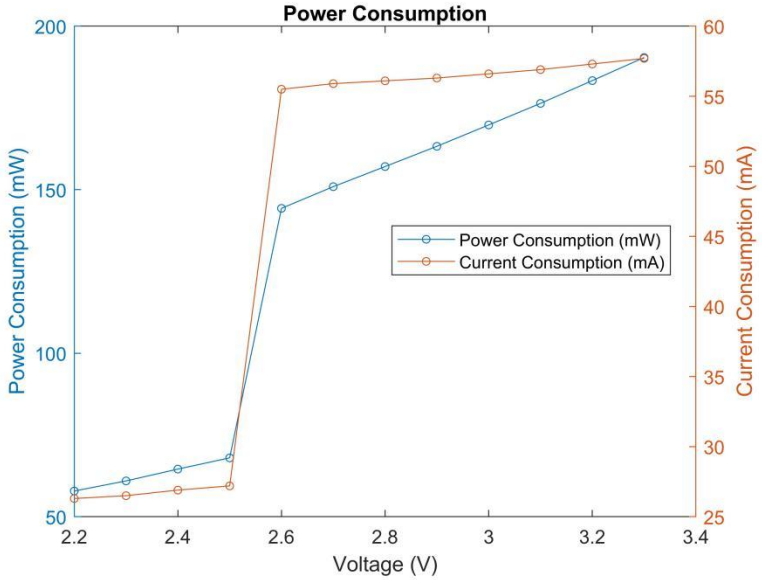


Figure 6. FireBeetle 2 ESP32-E current and power consumption at different levels of voltages, with a LED blink program running on the board.

3.2 *Impact of Frequency*

According to Equation 1, the CPU frequency is proportional to the 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 7. Adjusting CPU frequency can be achieved by calling setCpuFrequencyMhz function. The supply voltage is fixed at 5 volts.

From Figure 7, with the increasing of CPU frequency at fixed supply voltage, the current and power consumption piles up 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 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 [23]. 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.

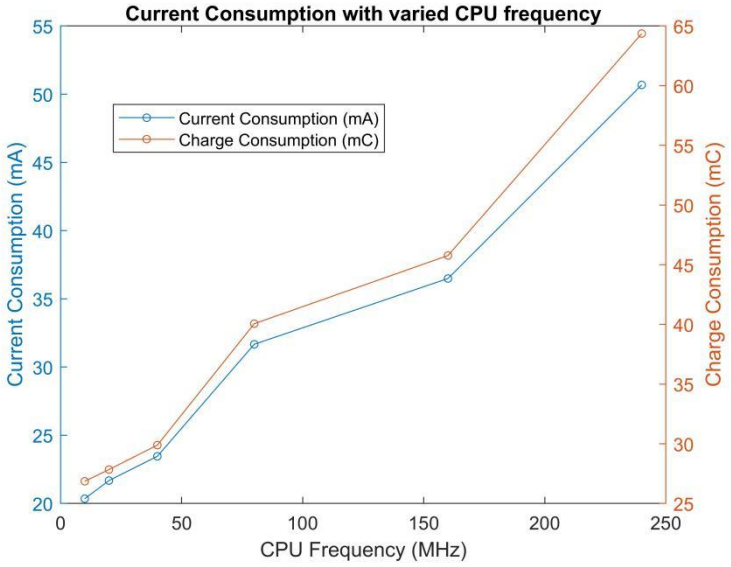


Figure 7. FireBeetle 2 ESP32-E current consumption at different levels of CPU frequencies, with a saving event program running on the board.

3.3 *Power Saving Modes of ESP32 and the Applications*

3.3.1 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. Details will be covered in Section 3.1.3. 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 paper. 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 4 illustrates the power modes with their active components and corresponding current consumptions.

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

Power Modes	Active Components	Current Consumption
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 µA 10 µA
Hibernation	RTC (timer, part of RTC GPIOs)	5 µA

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 1. 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 5.

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

Wake-up Sources	Wake-up Description	Parts to power on
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.

3.3.2. Local Data Storage

In several research papers [24–27], 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 [17] 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 [17] 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 paper, considering the relatively low current consumption and the non-volatile nature that ensures data integrity even without the supply power.

Table 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 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.

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

Memory	Operation	Current Consumption (mA)	Period (ms)	Discharge (mC)
SD card	Initialization	65.49	4620	302.4
	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

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 [28], 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 [15], 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.

3.3.3. 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 paper 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 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 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.

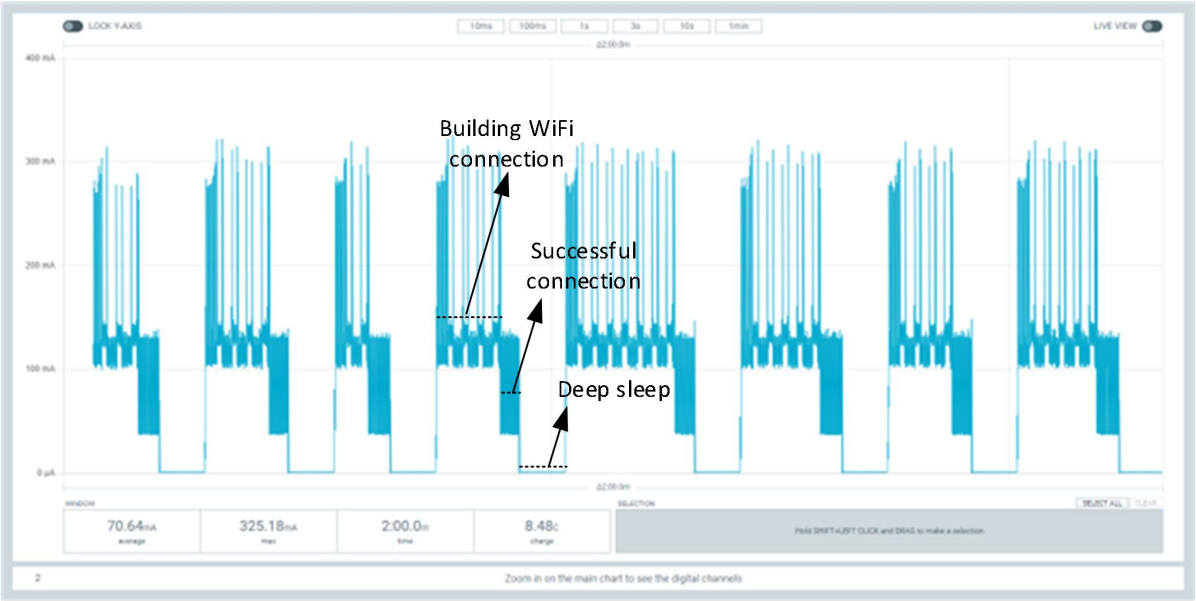


Figure 8. Current consumption during Wi-Fi connection.

Table 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 (s)	5.236	7.049	4.047	7.049	12.04	9.078	7.049	9.051	**7.575**
Current during building connection (mA)	117.26	119.65	114.92	119.50	119.85	117.57	119.58	117.79	**118.27**
Current after connection is built (mA)	61.46	60.56	60.83	61.94	60.53	61.26	61.17	64.27	**61.50**
Current during deep sleep (mA)	1.35	1.35	1.35	1.35	1.35	1.35	1.35	1.35	**1.35**

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.

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

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 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 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 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	**2.86**	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 paper 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.

4. System Design

4.1 *Algorithm Flowchart*

The entire code used in this paper is developed with C++ language in Arduino IDE 2.2.1 version. Figure 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 (30s) and remote monitoring (45s) 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 paper 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 are the customization of HTML web page that is displayed by entering local IP address.

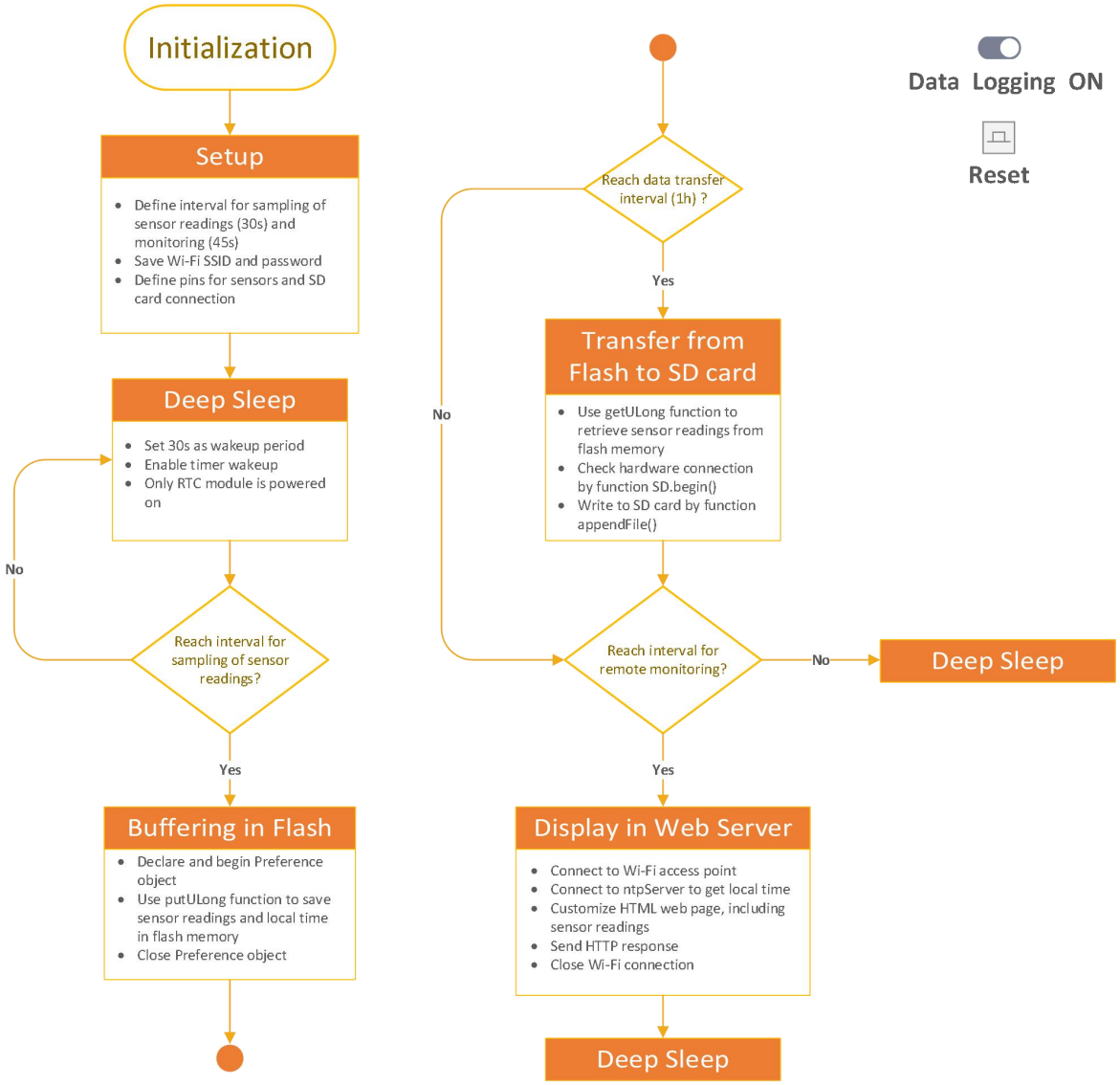


Figure 9. The flowchart of developed power saving algorithm.

4.2 *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, 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 suffice to power the sensors. By measuring more sets of PV panels and batteries, the unit cost can be reduced largely. In this paper 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 paper. 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 paper, 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 ±10 volts and two-channel digital inputs which can measure ±30 volts, DI-145 is capable of being the data logger for the PV system in this paper. 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 0.2 μA. 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 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.

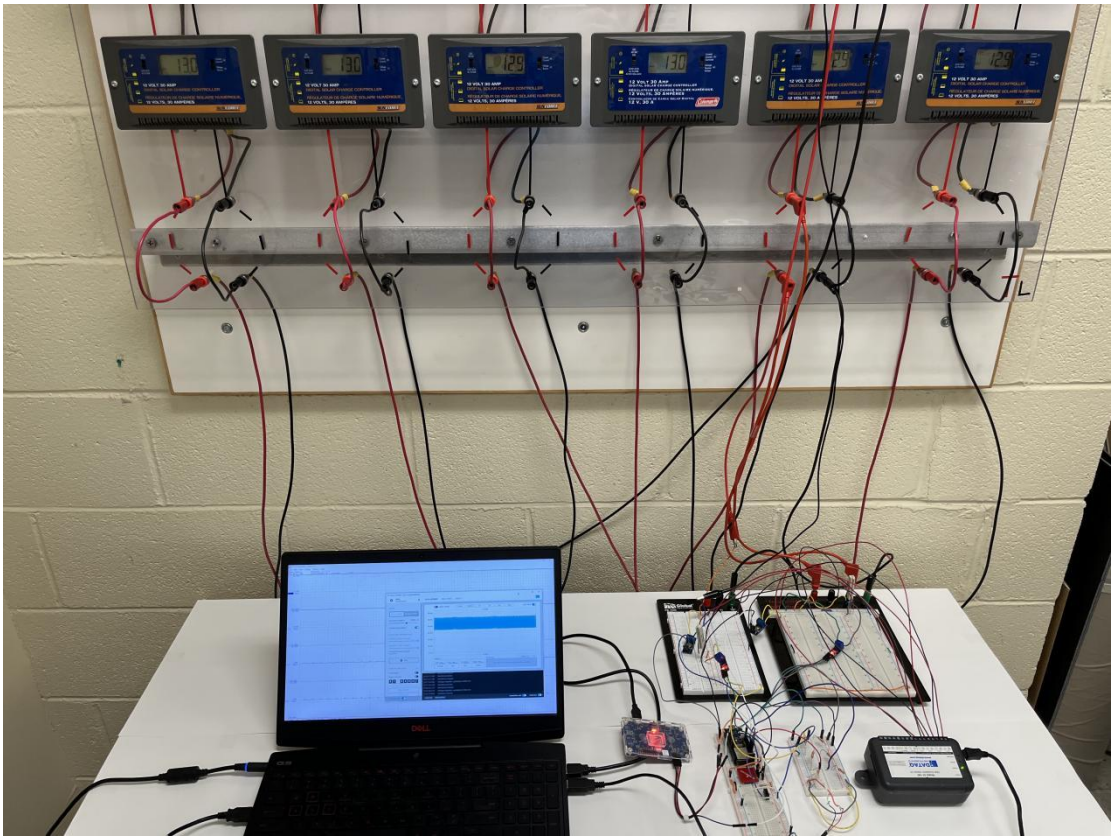


Figure 10. Experiment test setup.

5. Experimental Validation

5.1 *Datalogging Results Comparison with DATAQ DI-145*

Figures 11 and 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.

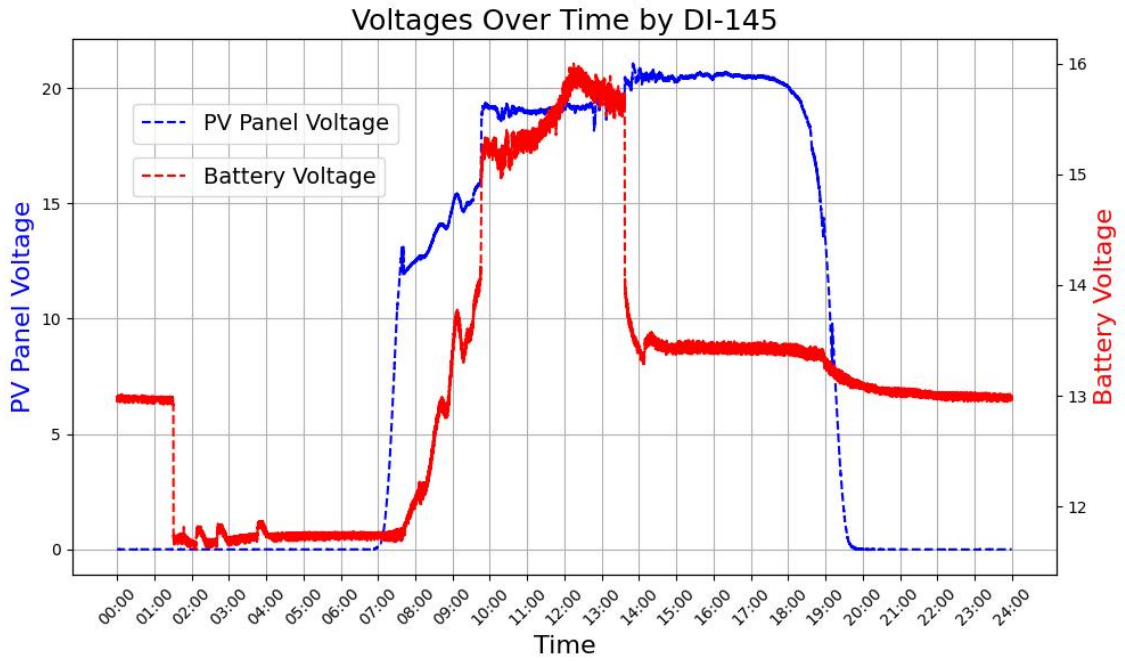


Figure 11. Voltage logging scatter plot over 24 hours by DI-145.

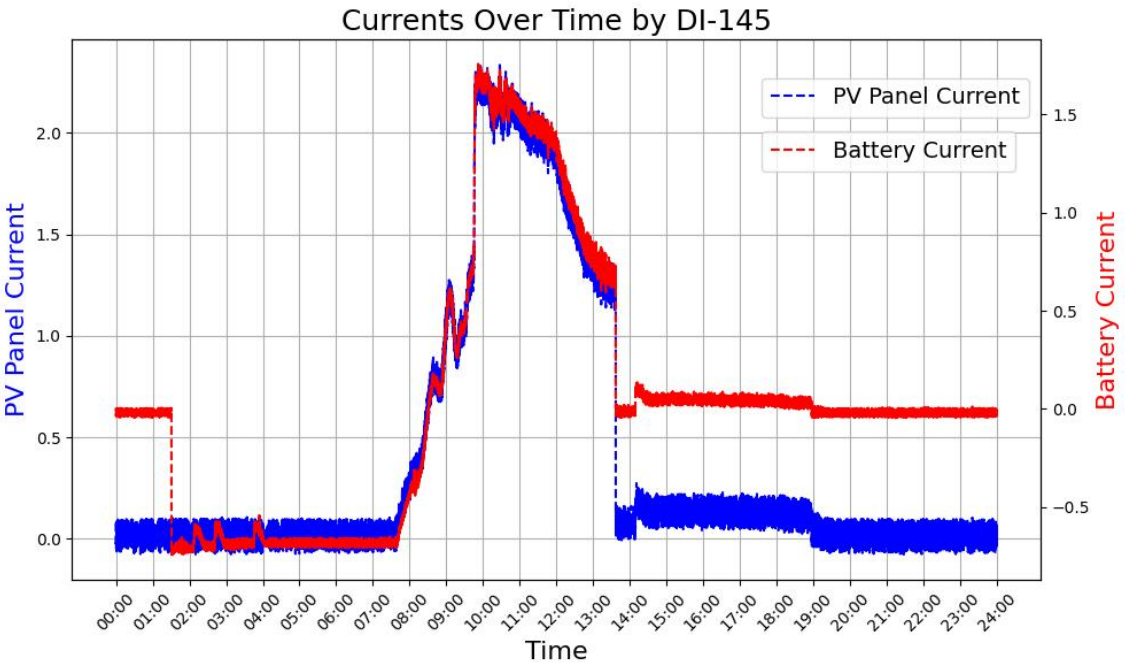


Figure 12. Current logging scatter plot over 24 hours by DI-145.

Recordings of voltage and current sensor readings by the low power data logger developed in this paper are revealed in Figures 13 and 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 paper is well-functioning during this experiment.

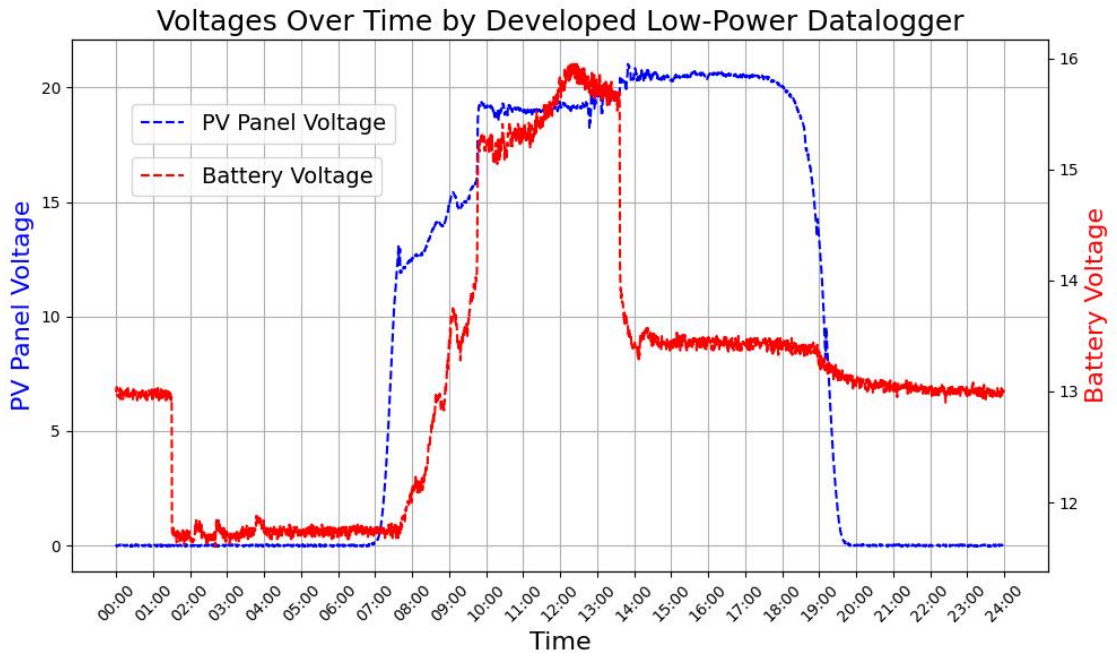


Figure 13. Voltage scatter plot over 24 hours using the developed low-power data logger.

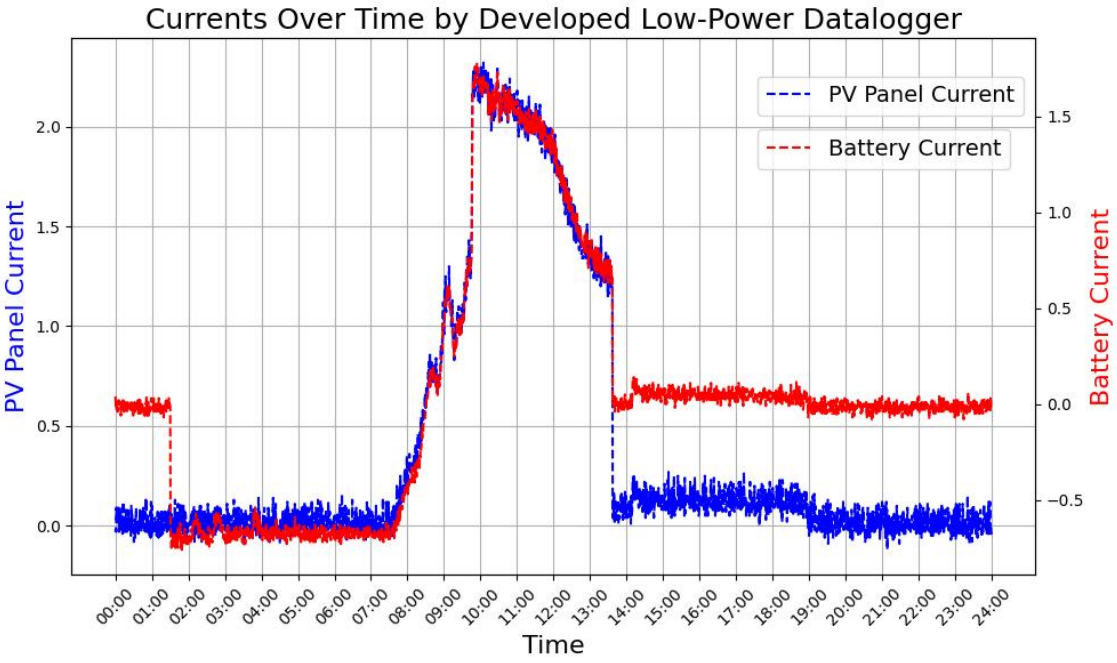


Figure 14. Current scatter plot over 24 hours using the developed low-power data logger.

5.2 *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 paper. 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 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 [29].

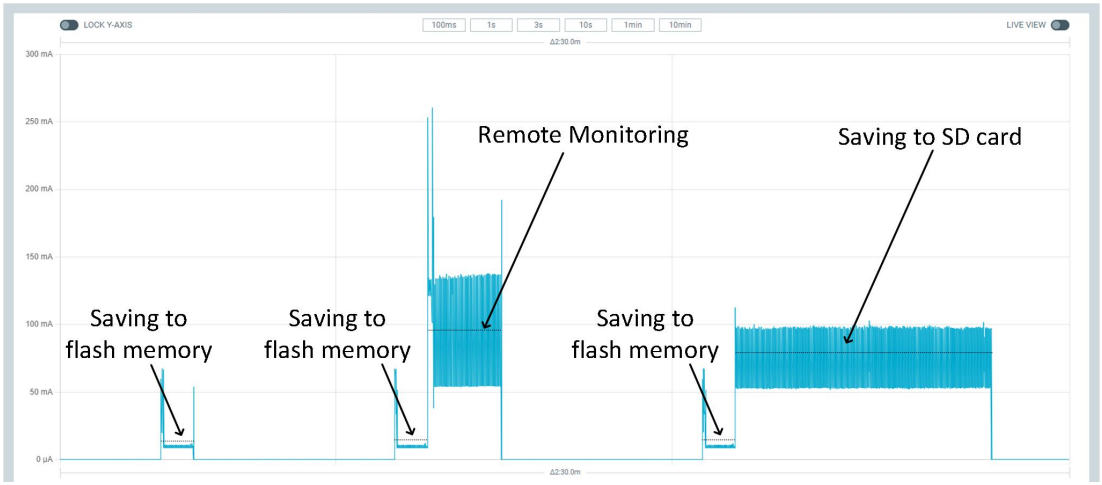


Figure 15. Current consumption including all scenarios of the data logger tasks.

5.3 *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 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 can be accessed by entering 192.168.137.245 in the browser.

MUN ECE PV SYSTEM DATA LOGGER: REAL-TIME DISPLAY

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 16. Web page showing electrical parameters of monitored PV system.

Low-cost is one of the principal features of the developed PV system data logger. Table 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.

Table 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
Total	**55.05**

6. Conclusion

This paper 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 paper is by assuming that the supply voltage keeps constant at 3500 mV, while the battery voltage decreases over time in practical applications.

Conflict of interest

There is no conflict of interest for this study.

References

[1] Yokwana, X.; Yusuff, A.; Mosetlhe, T. Faults in a Large-scale photovoltaic installation: A Review. In Proceedings of 2020 6th IEEE International Energy Conference (ENERGYCon), Gammarth, Tunisia, 28 September–1 October 2020, https://doi.org/10.1109/ENERGYCon48941.2020.9236456.

[2] 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 2023).

[3] Madeti, S.R.; Singh, S. Monitoring system for photovoltaic plants: A review. *Renew. Sustain. Energy Rev.* **2017**, *67*, 1180–1207, https://doi.org/10.1016/j.rser.2016.09.088.

[4] Gmbh, D. ADL-MXSpro Multifunctional solar data logger. Available online: https://www.meier-nt.de/en/photovoltaics/products/adl-mxspro-en (accessed on 15 September 2023).

[5] 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 15 September 2023).

[6] Q.reader 602 + Extension 01. Available online: https://www.gantner-environment.com/fileadmin/user_upload/q.reader_602_20190928.pdf (accessed on 15 September 2023).

[7] HUAWEI. SmartLogger3000A. Available online: https://solar.huawei.com/-/media/Solar/attachment/pdf/eu/datasheet/SmartLogger3000A.pdf (accessed on 15 September 2023).

[8] Powador-proLOG Operating instructions. Available online: https://kaco-newenergy.com/index.php?eID=dumpFile&t=f&f=6615&token=fda0ce4ca172c6c9356974075baa80235ae7af27 (accessed on 15 September 2023).

[9] 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 15 September 2023).

[10] SMA Data Manager M. Available online: https://files.sma.de/downloads/EDMM-10-DS-en-40.pdf (accessed on 15 September 2023).

[11] Solar-Log Manul V.3.6.0.G. Available online: https://www.europe-solarstore.com/download/solarlog/SolarLog_products_manual.pdf (accessed on 15 September 2023).

[12] MaxWeb XPN Installation instructions. Available online: https://docplayer.net/65483810-Maxweb-xpn-installation-instructions.html (accessed on 15 September 2023).

[13] 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 J.* **2019**, *19*, 4308–4320, https://doi.org/10.1109/jsen.2019.2898667.

[14] 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 2022 8th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 21–22 July 2022, https://doi.org/10.1109/ICWT55831.2022.9935404.

[15] 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 J.* **2018**, *19*, 679–691, https://doi.org/10.1109/jsen.2018.2876635.

[16] 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. *Appl. Energy* **2018**, *224*, 322-329, https://doi.org/https://doi.org/10.1016/j.apenergy.2018.04.091.

[17] 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, https://doi.org/10.1109/access.2020.3041373.

[18] Kalay, M..; Kılıç, B.; Mellit, A.; Oral, B.; Sağlam, S. IoT-Based Data Acquisition and Remote Monitoring System for Large-Scale Photovoltaic Power Plants. In Proceedings of International Conference on Electronic Engineering and Renewable Energy Systems, Saidia, Morocco, 20–22 May 2022, https://doi.org/10.1007/978-981-19-6223-3_65.

[19] FireBeetle ESP32 IOT Microcontroller (Supports Wi-Fi &Bluetooth) SKU: DFR0478. Available online: https://www.application-datasheet.com/pdf/dfrobot/dfr0478.pdf (accessed on 18 August 2023).

[20] Purwadi, A.; Haroen, Y.; Ali, F.Y.; 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 2011 International Conference on Electrical Engineering and Informatics, Bandung, Indonesia, 17-19 July 2011, https://doi.org/10.1109/ICEEI.2011.6021693.

[21] esp32-adc-calibrate. Available online: https://github.com/e-tinkers/esp32-adc-calibrate (accessed on 14 Au

gust 2023).
[22] Crowe, J.; Hayes-Gill, B. Choosing a means of implementation. In *Introduction to Digital Electronics*; Newnes: Oxford, UK, 1998; pp. 191–239.
[23] Broell, L.M.; Hanshans, C.; Kimmerle, D. IoT on an ESP32: Optimization Methods Regarding Battery Life and Write Speed to an SD-Card. IntechOpen: Rijeka, Croatia, 2023; Volume 1, Chapter 7. https://doi.org/10.5772/intechopen.110752.
[24] Bouzguenda, M.; Chtourou, S.; Alarfaj, M.; Sumsudeen, R.M.; Shwehdi, M. Arduino Uno Wi-Fi DeMilitarized Zone-based monitoring of solar photovoltaic systems. *Meas. Control.* **2022**, *55*, 136–145, https://doi.org/10.1177/00202940221090553.
[25] Khaleel, K.; Atyia, T.H.; Al-Naib, A.M.T.I. Design and Development of Real-Time Data Acquisition of P hotovoltaic Panel Parameters via IoT. *NTU J. Renew. Energy* **2022**, *3*, 1–8, https://doi.org/10.56286/ntujre. v3i1.276.
[26] İnner, B. Data monitoring system for solar panels with bluetooth. In Proceedings of 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 15–18 May 2017, https://doi.org/10.1109/10.1109/SIU.2017.7960529.
[27] Barka, C.; Messaoudi-Abid, H.; Setthom, H.B.A.; Abdelghani, A.B.-B.; Slama-Belkhodja, I.; Sammoud, H. A real time, wireless and low cost data acquisition system for residential PV modules. In Proceedings o f 2020 6th IEEE International Energy Conference (ENERGYCon), Gammarth, Tunisia, 28 September–1 O ctober 2020, https://doi.org/10.1109/ENERGYCon48941.2020.9236592.
[28] Beddows, P.A.; Mallon, E.K. Cave Pearl Data Logger: A Flexible Arduino-Based Logging Platform for Lo ng-Term Monitoring in Harsh Environments. *Sensors* **2018**, *18*, 530, https://doi.org/10.3390/s18020530.
[29] DI-145 User's Manual. Available online: https://www.dataq.com/resources/pdfs/manuals/145-manual.pdf (accessed on 2 September 2023).