Development of a Portable and Low-cost Remote Sensor for Measuring Solar and Related Data

Haoulomou, Pepe
九州大学総合理工学府環境エネルギー工学専攻

九州大学
KYUSHU UNIVERSITY

Development of a portable and low-cost remote sensor for measuring solar radiation and related data

A Thesis

By

Haoulomou Pepe

Submitted in Partiall Fulfillment for The Degree of Master of Science

Interdisciplinary of Graduate School of Engineering Science

Department of Energy and Environmental Engineering

Under the Supervision of:

Prof.
Takahiko Miyazaki

Assoc. Prof.
Kyaw Thu

August 16, 2021

ACKNOWLEDGEMENT

I would like to express my deepest sense of gratitude to my Professor, Professor Takahiko Miyazaki (Supervisor) and Associate Professor Kyaw Thu (Associate Professor) from Thermal Energy Conversion Systems Laboratory for their immense support and guidance throughout my entire research. I would personally like to say a big THANK YOU to Dr. Frantisek Mikisik (Tutor) without whom this work would not have been possible. It a is great honor for me to have joined a laboratory where everyone is always ready to help. I am grateful to all the students and faculty member of Thermal Energy Conversion Systems Laboratory for their time, suggestions and advice that made my research life comfortable. I would also like to thank Dr. Takada for assistance with the material and tools he provided us with. I wish to express my sincere appreciation to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) for the funding during my research. It is an amazing program with a helpful and friendly staff. I feel honored to be a part of MEXT Program. I would also like to express my sincere gratitude to my parents in Guinea for their support, caring and guidance during the whole research time. Finally I wish to thank my other family members, friends and well-wishers for their sacrifice, understanding and encouragement in my study and lab works.

Abstract

In recent decades, solar energy, a clean and renewable energy, has experienced a tremendous growth due to its energy potential and its non-polluting aspect for the environment. Given these factors, this renewable energy is now the subject of several studies ranging from its application, to the estimation of its photovoltaic potential in all corners of the globe. Solar energy, which is increasingly becoming the alternative for many countries of the world, is experiencing viscous progress in the African continent (solar and wind energy: acount for 3% of electricity generated in 2018). This is due to the fact that energy policy makers in the matter have little or no notion on how much electrical energy can be generated from photovoltaic systems in each country concerned. The quantification of solar energy for any location necessarily involves the measurement of surface radiation, a process today mainly dominated by satellite remote sensing. The data from these satellite remote sensing are still far from correctly explaining the spatial and temporal variability and also lack local verification and thus, should not be perceive as accurate. As a result, great efforts are always devoted to improving the modeling and recovery of solar radiation data. In this study, we propose a new device for measuring surface solar radiation at a lower cost with an accuracy of more than 95%. The remote sensor operates automatically and completely off grid and transmits the measurements data to an online database. Constantly online, it adjusts its operation according to the local weather conditions. With this remote sensor, ground-based solar radiation measurement data will be more accessible than ever and code used in this study is made public for the general public.

Keywords: Renewable energies, solar radiation, remote sensing, micro-controllers, photovoltaic potential, automatism, environment

Table of Contents

List of Figures

1 Introduction

1.1 Research Background

When it comes to renewable energies, solar energy has become unmissable due to its tremendous energy potential. The application of solar energy technologies has experienced an exponential growth over the past three decades to a point that it is considered as the best and most promising alternative to conventional fossil fuels still heavily used in modern day life for power generation.

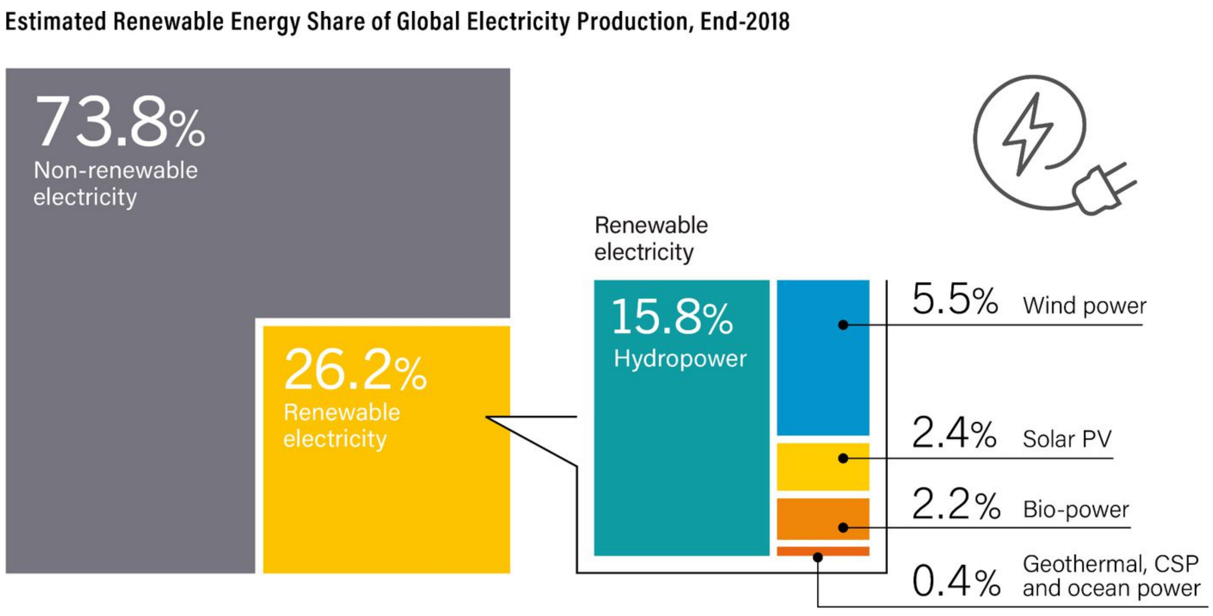


Figure 1.1: Share of renewable energies in the over all energy production of the world

Solar energy resources are massive and widespread, and they can be harnessed anywhere that receives sunlight. The amount of solar radiation, also known as insolation, reaching the Earth's surface every hour is more than all the energy currently consumed by all human activities each year. A number of factors, including geographic location, time of day, and weather conditions, all affect the amount of energy that can be harnessed for electricity production or heating purposes. Solar photovoltaic are the fastest growing electricity source. In 2018, around 100 GW of global capacity was added, bringing the total to about 505 GW and producing a bit more than 2 percent of the world's electricity. Solar energy can be captured for electricity production using: A solar or photovoltaic cell, which converts sunlight into electricity using the photoelectric effect. Typically, photovoltaic are found on the roofs of residential and commercial buildings. Additionally, utilities have constructed large (greater than 100 MW) photovoltaic facilities that require anywhere from 5 to 13 acres per MW, depending on the technologies used. Concentrating solar power, which uses lenses or mirrors to concentrate sunlight into a narrow beam that heats a fluid, producing steam to drive a turbine that generates electricity. Concentrating solar power projects are larger-scale than residential or commercial PV and are often owned and operated by electric utilities. Solar hot water heaters, typically found on the roofs of homes and apartments, provide residential hot water by using a solar collector, which absorbs solar energy, that in turn heats a conductive fluid, and transfers the heat to a water tank. Modern collectors are designed to be functional even in cold climates and on overcast days. Electricity generated from solar energy emits no greenhouse gases. The main environmental impacts of solar energy come from the use of some hazardous materials (arsenic and cadmium) in the manufacturing of PV and the large amount of land required, hundreds of acres, for a utility-scale solar project.

1.2 General Overview of Remote Sensing

Remote sensing, which is the use of electromagnetic energy to measure the physical properties of distant objects, is generally referred to as the use of sensors and cameras attached to aircraft or satellites to detect and classify objects on earth. while it is true in some cases, it is not always the case. Remote sensing is used in almost every area of our life: from automatic lighting systems to weather stations, to satellites in space. Cameras on satellites and airplanes take images of large areas on the Earth's surface, allowing us to see much more than we can see when standing on the ground. Sonar systems on ships can be used to create images of the ocean floor without needing to travel to the bottom of the ocean. Cameras on satellites can be used to make images of temperature changes in the oceans.

Some specific uses of remotely sensed images of the Earth include: Large forest fires can be mapped from space, allowing rangers to see a much larger area than from the ground. Tracking clouds to help predict the weather or watching erupting volcanoes, and help watching for dust storms. Tracking the growth of a city and changes in farmland or forests over several years or decades. Discovery and mapping of the rugged topography

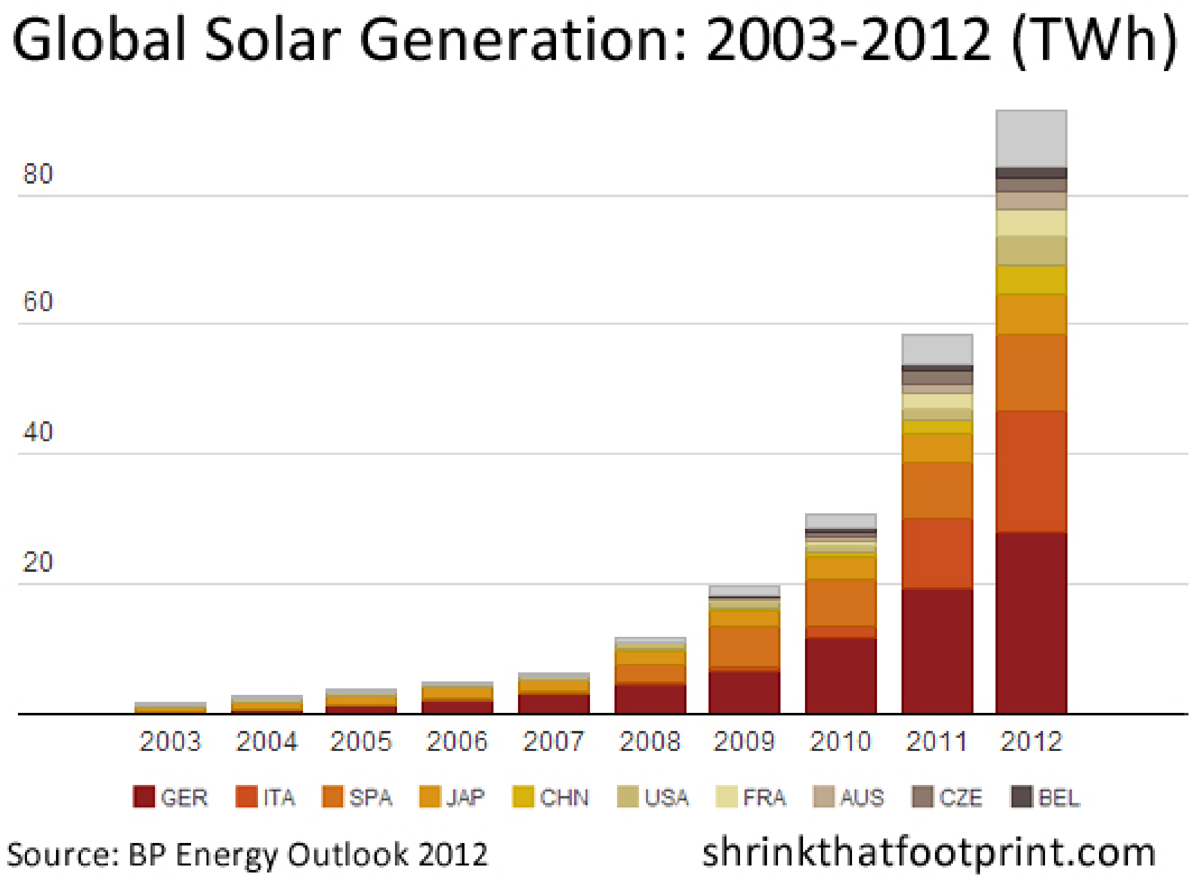


Figure 1.2: Growth of solar power generation

of the ocean floor (e.g., huge mountain ranges, deep canyons, and the "magnetic striping" on the ocean floor).

While this technology has experienced a tremendous growth in the monitoring, classification of distant objects both on earth and in space, it has not experienced any growth in the measurement of ground-based solar radiation measurement.

1.3 Motivation

Remote sensing can pretty much be used anywhere and depending on where you want to use it or what you want to use it for, the requirements would definitely be different. Building a remote sensor that needs access to internet service to work sounds like a daunting idea when we consider how hard it is to access the internet mostly in developing countries and in remote places of the globe with less or no humans activities of advanced nations.

But, luckily, that is about to change. Starlink which is a satellite internet constellation operated by SpaceX to provide satellite Internet access to most of the Earth, is a full

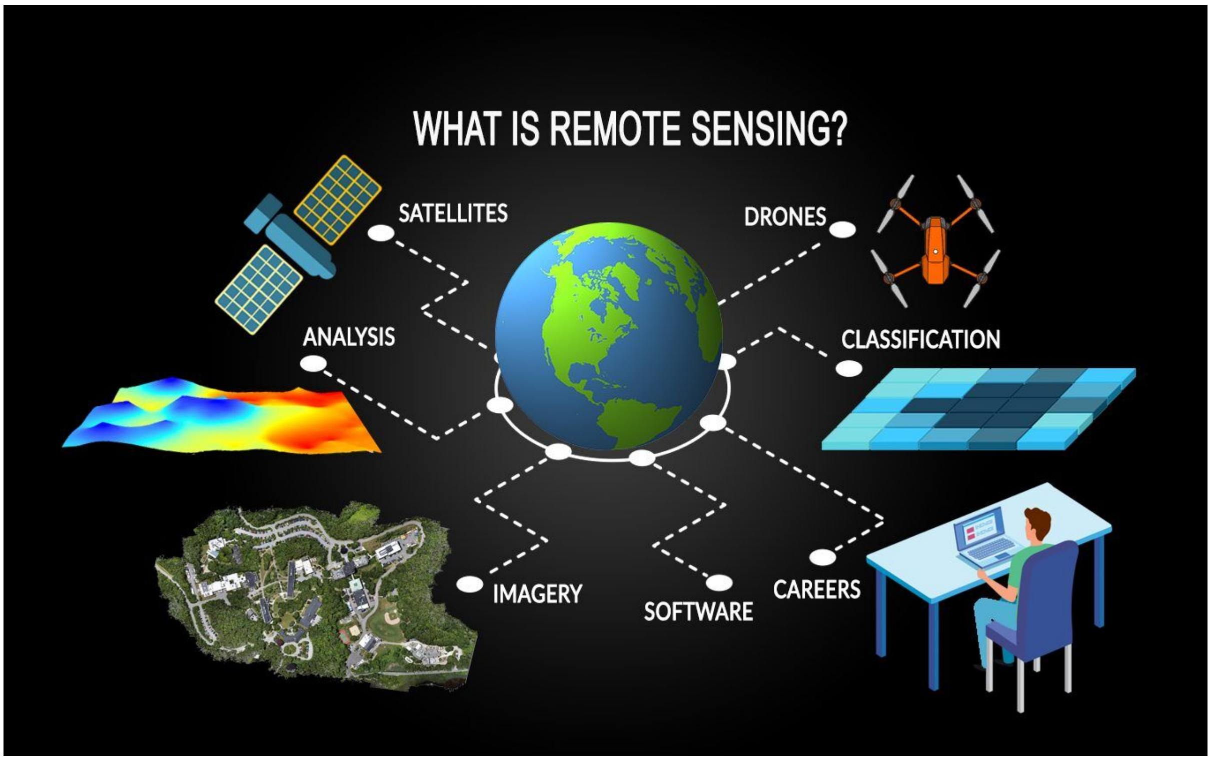


Figure 1.3: Remote Sensing

development and testing phase of this cluster of satellites will orbit planet earth at low altitude to provide internet services. The constellation consists of over 1600 satellites in mid-2021, and will eventually consist of many thousands of mass-produced small satellites in low Earth orbit (LEO), which communicate with designated ground transceivers. As of July 2021, the beta service offering is available in 11 countries. This is going to revolutionized the world. It means that ideas like building a remote sensor for measuring solar radiation and posting the recorded data online will be possible for any location of the globe thanks to this amazing technology. In this studies, our main priority was to use the concept of remote sensing at the ground level to measure the amount solar radiation reaching the earth surface and convert that electromagnetic energy into useful energy $[w/m^2]$ and be able to send the recorded data to a cloud-hosting database for further analysis.

1.4 Layout of the Thesis

This work is subdivided into six(6) chapters. In Chapter 1, a general introduction of the work is presented along with a general overview of remote sensing and its applications and the motivation for this Research study. Chapter 2 describes the current status of remote sensing and satellites data. Chapter 3 layouts a comprehensive description of the major equipment used to build our remote sensor and their electrical characteristics. Chapter 4 show the result of many months of building and design the remote sensor and the validation of the data recorded. Chapter 5 gives a general conclusion of the work done in this study while chapter 6 is all about the code used to make to control,read the sensors data, store them locally and collect and post these recorded data to google sheet.

2 Current State of the Art

Over the last three decades, solar energy has gained a considerate number of attentions for the great energy potential it holds both thermally and in term of photovoltaic. Thanks to its environment friendly aspect, this form of energy is considered by most as the best alternative for conventional energy sources like fossil fuels. For countries with the financial means and the technology, several weather stations have been built where solar radiation along with other weather related data like temperature, barometric pressure and relative humidity data are constantly being recorded and saved for future usage.

In other part of the globe where no such data exist, satellites approximation data or imageries are becoming the norm as pointed out by [Zarzalejo et al., 2009] that solar radiation derived from geostationary satellite images has become an advantageous technique for solar resource characterisation over large areas. But, we should be very careful when using satellite approximation data in scientific research as the accuracy of these satellites approximation data typically lack geographically localized verification and they should not be perceived universally accurate. This has been highlighted by several researchers. [Fu and Fukumori, 1996] investigated the uncertainty of the estimate resulting from errors in satellite altimetry is investigated, in particular, the effects of the orbit and tide errors. The radiance measurement errors associated with a sensor's polarization dependence could decrease the measurement accuracy below the requirements of reliable climate monitoring [Wielicki et al., 2013].

Equally too, the importance of ground-based solar radiation to validate satellite derived solar radiation has also been underlined by many other researchers and several algorithms and models have been developed during the last three decades for estimating the solar irradiance at the earth surface from satellite images [Pinker et al., 1995, Renne et al., 1999]. [Zarzalejo et al., 2009] underlined that the knowledge of the solar resource at the earth surface, with enough accuracy, is essential for planning any solar energy system at a given location. [Zarzalejo et al., 2009] used simultaneous data of satellite derived cloud index and hourly global irradiance on ground-based stations for model development and assessment for 28 locations in Spain covering geographical areas corresponding to

Mediterranean, Semi–Arid and Oceanic climate. [Sun et al., 2019] mentioned that measurements obtained from many satellite radiometric instruments have some dependence on the polarization state of the solar radiation reflected from the surface and scattered by atmospheric molecules and particles. And most importantly, the need to improve satellite data accuracy has been expressed in U.S. inter-agency reports [Ohring et al., 2005, Ohring et al., 2007] and international observing system plans and the Global Space-Based Intercalibration System [Mason and Simmons, 2011] [Christian, 2005] [Goldberg et al., 2011] [Goldberg, 2007].

Although a lot has been said about the errors and the inaccuracy of satellite approximation solar radiation data, there is no specific study that tackles this issue by offering an affordable solution. Hence the need to conduct this research that aims to build a remote sensor that can be basically replicated by anyone to help address the lack of ground-based solar irradiance data for any location.

3 Experimental Setup

To construct the current remote sensor, we used a bunch of electronic components, a plastic box, a piece of plywood and several other equipment. The major components are listed bellow with a full description of what each is and what it is used for.

3.1 Real-time clock (RTC)

With a world moving so fast toward automation, control over timing behavior is difficult to achieve, and timing behavior is neither predictable nor repeatable. A real-time clock (RTC) is an electronic device (most often in the form of an integrated circuit) that keeps time. It is integrated in most automatic programs or software to help keep the time and restore that time when called for. Although the term often refers to the devices in personal computers, servers and embedded systems, RTCs are present in almost any electronic device which needs to keep accurate time of day. The term real-time clock is used to avoid confusion with ordinary hardware clocks which are only signals that govern digital electronics, and do not count time in human units. RTC should not be confused with real-time computing, which shares its three-letter acronym but does not directly relate to time of day.

3.1.1 Features

Although keeping time can be done without an RTC, using one has benefits:

- Low power consumption (important when running from alternate power)
- Frees the main system for time-critical tasks
- Sometimes more accurate than other methods

3.1.2 Power Source

RTCs often have an alternate source of power, so they can continue to keep time while the primary source of power is off or unavailable. This alternate source of power is normally a lithium battery in older systems, but some newer systems use a super capacitor, because they are rechargeable and can be soldered. The alternate power source can also supply power to battery backed RAM.

3.1.3 Operation

It supports I2C interface. I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus invented in 1982 by Philips Semiconductors. It is widely used for attaching lower-speed peripheral ICs to processors and micro-controllers in short-distance, intra-board communication.

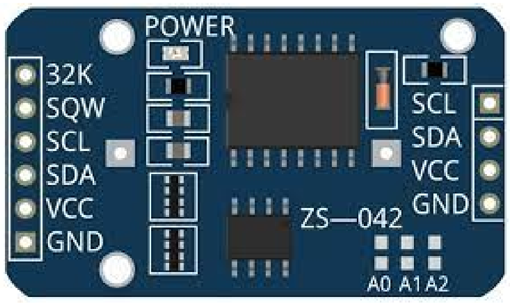


Figure 3.1: Real-time clock module (RTC)

3.2 BME280: Humidity, Temperature and Pressure Sensor

The BME280 is an integrated environmental sensor developed specifically for mobile applications where size and low power consumption are key design constraints. The unit combines individual high linearity, and high accuracy sensors for pressure, humidity and temperature in an 8-pin metal-lid 2.5 x 2.5 x 0.93 mm 3 LGA package. The BME280 is designed for low current consumption (3.6 μA @1Hz), long term stability, and high EMC robustness. The humidity sensor features an extremely fast response time which supports performance requirements for emerging applications such as: context awareness, and high accuracy over a wide temperature range. The pressure sensor is an absolute barometric pressure sensor which features exceptionally high accuracy and resolution at very low noise. The integrated temperature sensor has been optimized for very low noise and high resolution. It is primarily used for temperature compensation of the pressure and humidity sensors, and can also be used for estimating ambient temperature. The BME280 supports a full suite of operating modes which provides the flexibility to optimize the device for power consumption, resolution, and filter performance.This precision sensor can measure relative humidity from 0 to 100% with ±3% accuracy, barometric pressure from 300Pa to 1100 hPa with ±1 hPa absolute accuracy, and temperature from -40°C to 85°C with ±1.0°C accuracy.

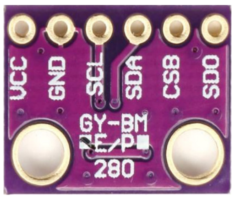


Figure 3.2: BME280 (Temperature, Humidity and Pressure sensor)

3.2.1 Features

- Operation range:
  - Pressure: 300 $\cdots$ 1100 hPa
  - Temperature: -40 $\cdots$ 85°C
- Interface: I2C and SPI
- Average current consumption in sleep mode: $0.1\mu A$
- Humidity sensor
  - Accuracy tolerance:±3% relative humidity
- Pressure sensor
  - Temperature coefficient offset : $\pm 1.5 Pa/K$ (equiv. to $\pm 12.6 cm$ at 1 °C temperature change)

3.2.2 BME280's Operation

The BME280 supports I2C and SPI (3-wire/4-wire) digital, serial interfaces. The sensor can be operated in three power modes: Sleep mode, normal mode and the forced mode. In normal mode the sensor automatically cycles between a measurement and a standby period. This mode is recommended when using BME280 built-in IIR filter when short-term disturbances (e.g. blowing into the sensor) need to be filtered. In forced mode, the sensor performs a single measurement on request and returns to sleep mode afterwards. This mode is recommended for applications that require a low sampling rate or host-based synchronization. In order to tailor data rate, noise, response time and current consumption to the needs of the user, a variety of oversampling modes, filter modes and data rates can be selected. In combination with several short term disturbance filter settings, the sensor can be programmed in a very flexible way in order to adapt to application and power management requirements. To simplify the design-in phase, default settings optimized for several example use-cases such as weather monitoring, elevator/stair case detection, drop detection or indoor-navigation are provided.

3.3 INA219 Current sensor

The INA219 is a current shunt and power monitor with an I 2 C- or SMBUS-compatible interface. The device monitors both shunt voltage drop and bus supply voltage, with programmable conversion times and filtering. A programmable calibration value, combined with an internal multiplier, enables direct readouts of current in amperes. An additional multiplying register calculates power in watts. The I 2 C- or SMBUS-compatible interface features 16 programmable addresses. The INA219 is available in two grades: A and B. The B grade version has higher accuracy and higher precision specifications. The INA219 senses across shunts on buses that can vary from 0 to 26 V. The device uses a single 3- to 5.5-V supply, drawing a maximum of 1 mA of supply current. The INA219 operates from –40°C to 125°C.

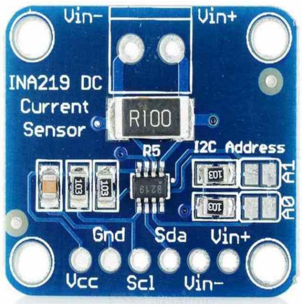


Figure 3.3: INA219 Current Sensor

3.3.1 Features

- Senses Bus Voltages from 0 to 26 V
- Reports Current, Voltage, and Power
- 16 Programmable Addresses
- Working Temperature Range (–40°C to 125°C.)
- Filtering Options

- Calibration Registers

3.4 Solar Panel 10 watt, 5V

One of the biggest challenges in developing a remote sensor device is how to keep the system operational. This can be an easy question to answer if the device application is within reach of the electrical grid; but it can also be daunting if the device needs to be deployed in remote location without any access to the grid. Depending on the location, solar panels are one the practical solutions achievable with less resources. In this project, two mini-solar panels (10W, 5V each) mounted in parallel were used to provide charge to the on board batteries. During the day, energy from the sun is converted into electrical energy and stored in one of the batteries. That electrical power is then restored to the second battery every two days at night, when the system is shutdown.

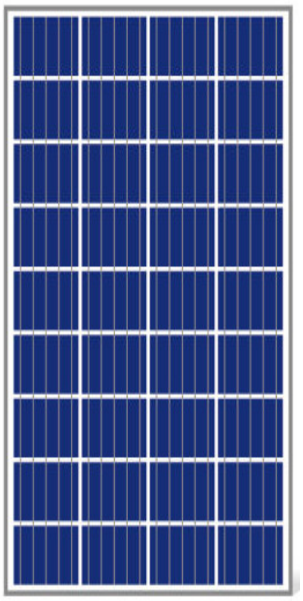


Figure 3.4: 10 Watt, 5V Mini-solar Panel

3.5 Raspberry Pi Zero

The Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. It is a low cost, fully capable computer with a low power consumption (around 80 mA when: Idle, HDMI disabled, LED disabled) and used in many electronic applications where a typical computer functionality is required. What is more important is that the Raspberry Pi has the ability to interact with the outside world with its many General Purpose Input

Output (GPIO) pins and has been used in a wide array of digital maker projects, from weather stations and electronic automation processes to surveillance-cameras systems. This device is used as the main board computer of the current remote sensor and therefore is responsible of communicating with all the sensors through its GPIO pins and storing the collected data on hard disk.

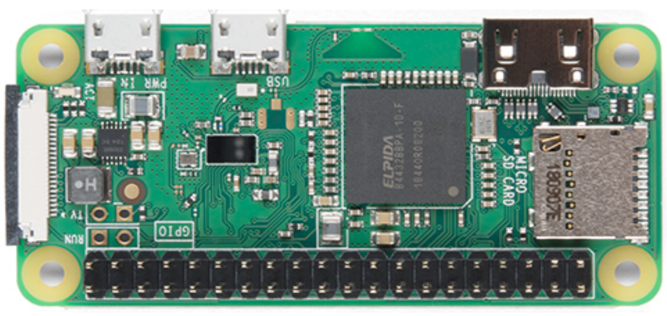


Figure 3.5: Raspberry Pi zero

3.6 ML-01 Si-Pyranometer

ML-01 Si-pyranometer is a global solar radiation measurement sensor by the Japanese company EKO. It complies to ISO 9060-2018 (Fast response class C) and is made for high quality irradiance measurements. The ML-01 is an industrial grade solar sensor specially made for irradiance measurement applications for the meteorological, agricultural and environmental studies.

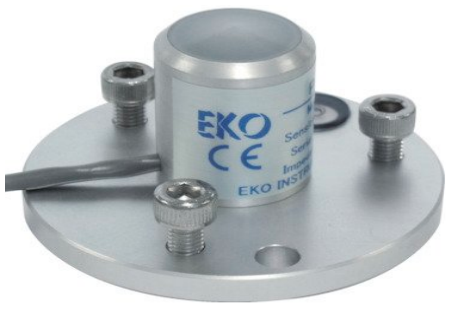
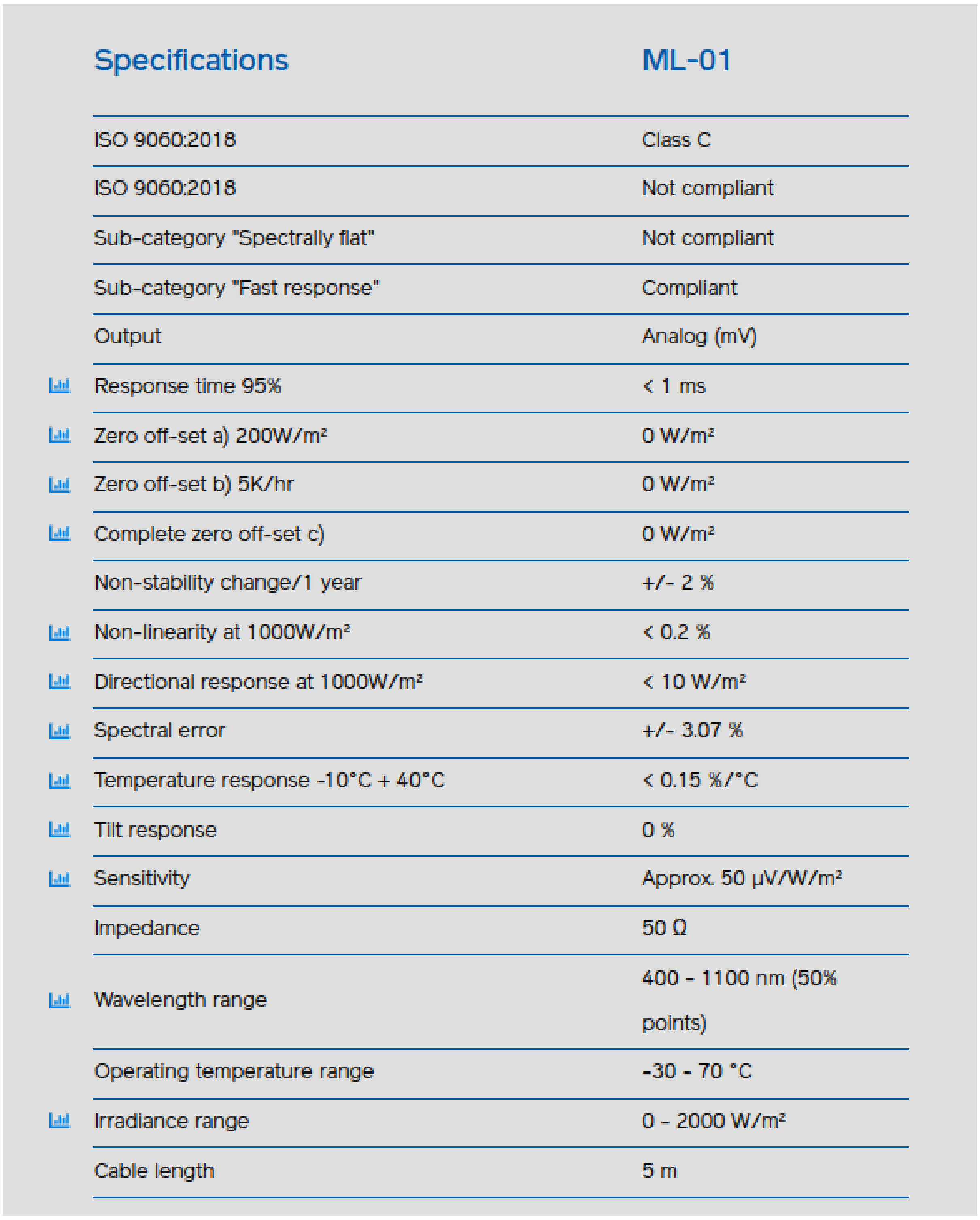


Figure 3.6: ML-01 Si-Pyranometer

The compact dimensions of the sensor body make it easy to integrate it within any application using it with or without a mounting plate. For global horizontal measurement applications, the sensor can be mounted in the horizontal position with a standard removable mounting plate with a spirit level and leveling feet. The Mono-Silicon detector with UV resistant diffuser gives a cosine response also at low solar elevation angles. Besides the effects of soiling or water deposition on top of the diffuser.



	Specifications	ML-01
	ISO 9060:2018	Class C
	ISO 9060:2018	Not compliant
	Sub-category "Spectrally flat"	Not compliant
	Sub-category "Fast response"	Compliant
	Output	Analog (mV)
📊	Response time 95%	< 1 ms
📊	Zero off-set a) 200W/m²	0 W/m²
📊	Zero off-set b) 5K/hr	0 W/m²
📊	Complete zero off-set c)	0 W/m²
	Non-stability change/1 year	+/- 2 %
📊	Non-linearity at 1000W/m²	< 0.2 %
📊	Directional response at 1000W/m²	< 10 W/m²
📊	Spectral error	+/- 3.07 %
📊	Temperature response -10°C + 40°C	< 0.15 %/°C
📊	Tilt response	0 %
📊	Sensitivity	Approx. 50 µV/W/m²
	Impedance	50 Ω
📊	Wavelength range	400 - 1100 nm (50% points)
	Operating temperature range	-30 - 70 °C
📊	Irradiance range	0 - 2000 W/m²
	Cable length	5 m

Figure 3.7: ML-01 Si-Pyranometer Specifications (datasheet)

3.7 Analog To Digital Converter (ADC)

In digital processing systems, analog to digital converter is a basal unit playing an imperative role as digital processing overweights the analog processing in most of the applications. ADC bridges the analog trend with digital reign.This device has a stated typical accuracy of 0.01% (but it has a maximum accuracy of 0.15%). This accuracy includes all sources of error (voltage reference, Gain error, offset and noise). This ADC is used to convert the input signal of the solar radiation sensor (ML-01 Si-Pyranometer) into a voltage which is then converted into solar irradiance w/m^2.

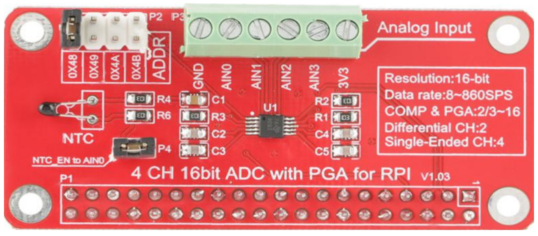


Figure 3.8: Analog to digital converter ADS11X15

3.7.1 Features

- Wide Supply Range: 2.0 V to 5.5 V
- Low Current Consumption: 150 μA (Continuous-Conversion Mode)
- Programmable Data Rate: 8 SPS to 860 SPS
- Single-Cycle Settling
- Internal Low-Drift Voltage Reference
- Internal Oscillator
- I2C Interface: Four Pin-Selectable Addresses
- Four Single-Ended or Two Differential Inputs (ADS1115)
- Programmable Comparator (ADS1114 and ADS1115)
- Operating Temperature Range: –40°C to +125°C
- Accuracy:

3.8 ESP32 nodemcu

ESP32 is a series of low-cost, low-power system on a chip micro-controllers with integrated Wi-Fi and dual-mode Bluetooth. It is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions. Used as the main controlling unit in this project, it responsible of turning off and on the the raspberry Pi Zero, controlling charge supply to the main battery.

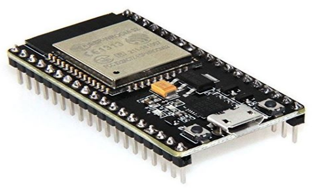


Figure 3.9: ESP32 nodemcu

3.9 Internet Dongle

With major telecommunication companies that provide Internet services to the public based in big cities (mostly in developing countries), building a remote sensor that needs access to the Internet to post recorded data to a cloud hosting database is one of the hardest tasks to achieve. This is where portable Internet dongles come in. These devices come in various forms with various requirements. A dongle is a small USB device that allows you to access the Internet. It can also be referred to as a Wi-Fi dongle, USB modem, Internet stick, USB network adapter or USB mobile broadband stick. Dongles are popular because they offer greater flexibility than fixed line connections and can be used on the go. A LTE 4G Wi-fi dongle by **Mugast** is used in this project to provide Internet service to the on board computer when needed.

Figure 3.10: Internet Modem

3.10 Relays

The ability to control a system of interconnected electrical components with a low-input current would be far more difficult the help of relays. A relay is basically an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact forms, such as make contacts, break contacts, or combinations thereof.

Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal. Relays were first used in long-distance telegraph circuits as signal repeaters: they refresh the signal coming in from one circuit by transmitting it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

The automatic control of the whole system (including supply and charge of the batteries) were achieved using a serie of 7 relays modules.

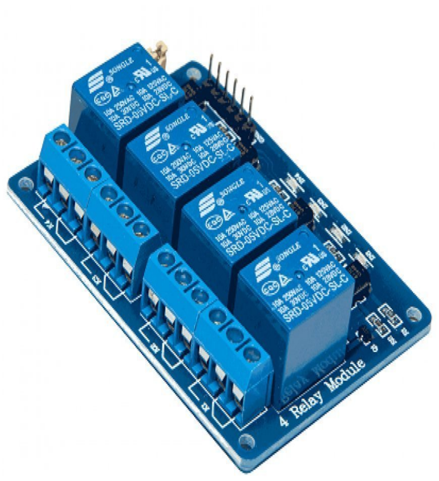


Figure 3.11: Four channels Relay

3.11 Arduino Leonardo Mini Pro

The Arduino Pro Mini is a microcontroller board based on the ATmega168 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. A six pin header can be connected to an FTDI cable to provide USB power and communication to the board.

The Arduino Pro Mini is intended for semi-permanent installation in objects or exhibitions. The board comes without pre-mounted headers, allowing the use of various types of connectors or direct soldering of wires. The pin layout is compatible with the Arduino Mini.

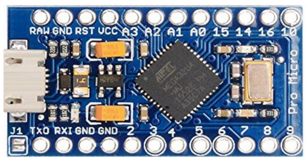

RAW GND RST VCC A3 A2 A1 A0 15 14 16 10
Pro Micro
J1 TXO RXI GND GND 2 3 4 5 6 7 8 9

Figure 3.12: Arduino Leonardo Mini Pro

There are two version of the Pro Mini. One runs at 3.3V and 8 MHz, the other at 5V and 16 MHz.

3.11.1 Features

- Operating Voltage: 3.3V or 5V (depending on model)
- Input Voltage: 3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- Flash Memory: 16 KB (of which 2 KB used by bootloader)
- SRAM: 1 KB
- EEPROM: 512 bytes
- Clock Speed: 8 MHz (3.3V model) or 16 MHz (5V model)

3.12 ACS712 Current Sensor

The ACS712 is a fully integrated, hall effect-based linear current sensor and an integrated low-resistance current conductor. Simply put, it is a current sensor that uses its conductor to calculate and measure the amount of current applied.

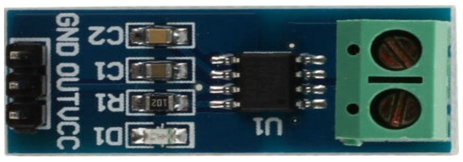


Figure 3.13: ACS712 current sensor

3.12.1 Features

- $80kHz$ bandwith
- 66 to 185 mV/A output sensitivity
- Low-noise analog signal path
- 1.2 $m\Omega$ internal conductor resistance
- Total output error of 1.5% at TA = 25°C
- Stable output offset voltage

3.13 ROMOSS Power Bank (30,000 mAh, 18 W)

A Lithium Polymer battery, used to store electrical energy provided by the on board solar panels for future use.

Figure 3.14: ROMOSS Power Bank

3.14 FS400-SHT3X (Temperature and Humidity Sensor)

A dust-proof digital temperature and humidity sensor with a stainless steel plastic housing, high precision and low consumption. The sensor has normal performance within the recommended operating range. Long-term exposure to conditions outside the normal range, especially at humidity >80%, may result in temporary signal drift (drift after 60 hours +3% RH) and again prolonged exposure to high concentrations of chemical smell will cause the sensor reading to drift.

3.14.1 Features

- **High Precision:** Digital and resolution up to 14 bit can be applied to computer room monitoring, warehouses, construction sites, concrete test block inspection, agricultural greenhouses, flower seedlings, etc
- **COMPACT:** Internal self-calibration, no calibration required, compact product structure work condition for field application and product integration: sensors ensure normal performance within the recommended range of motion.
- **Temperature and Humidity Sensor:** Close combination of temperature and humidity for convenient measuring.
- **Stability:** Anti-leaching, dust proof, anti-rust, high reliability, long-term stability. The relative humidity of air is a function of temperature. The temperature is greatly affected by relative humidity.
- **I2C Interface:** I2C serial interface can be easily integrated.

Figure 3.15: SHT31 Temperature and Humidity Sensor

3.15 Current Supply and Consumption of the major components

The electrical components used in this project have different requirements when it comes to the supply power and operation current. You will find the specification of each component in term of operation current and supply voltage range if any in Table 3.1.

Device Name	**Manufacturer**	**Supply Voltage**	**Current consumption**
ML-01 Si-Pyranometer	EKO	-	-
BME280	Socobeta	$1.71V \rightarrow 3.6V$	$0.16\mu A$
INA219	HiLetgo	$3V \rightarrow 5.5V$	$1mA$
ADS1115 ADC	LaDicha	$2V \rightarrow 5V$	$50\mu A$
4G LTE USB	Mugast	$5V$	$172 \cdots 200mA$
FS400-SHT3X	TOPINCN	$3.3V \rightarrow 5V$	$< 10mA$
ACS712 (5A)	Hiletgo	$3.3V$	$10mA$
Raspberry Pi Zero	Raspberry Pi	$5V$	$80mA$
ESP32 Nodemcu	KeeYees	$3.3V \rightarrow 5V$	$27mA \cdots 44mA$
Arduino Leonardo Mini Pro	Aideepen	$3.3 \rightarrow 5V$	$1.58mA$
Four Channels relay	ELEGOO	$5V$	$80 \cdots 100mA$

Table 3.1: Electrical Specifications of the major components

3.16 Sensors Communication Protocol

3.16.1 I2C Communication

With this communication protocol, various sensors (up to 128) can share the same Data and Clock lines without interupting one another. Such technique was used to talk to the RTC module, the bme280 sensor, the FS400-SHT3X sensor and the current sensor ina219 via the same Data and clock lines.

I2C which stands for Inter-Integrated Circuit, is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. I2C Communication Protocol uses 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.

- Serial Data (SDA): Transfers of data takes place through this pin.
- Serial Clock (SCL): Carries the clock signal.

I2C operates in 2 modes Master mode and Slave mode; and each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.

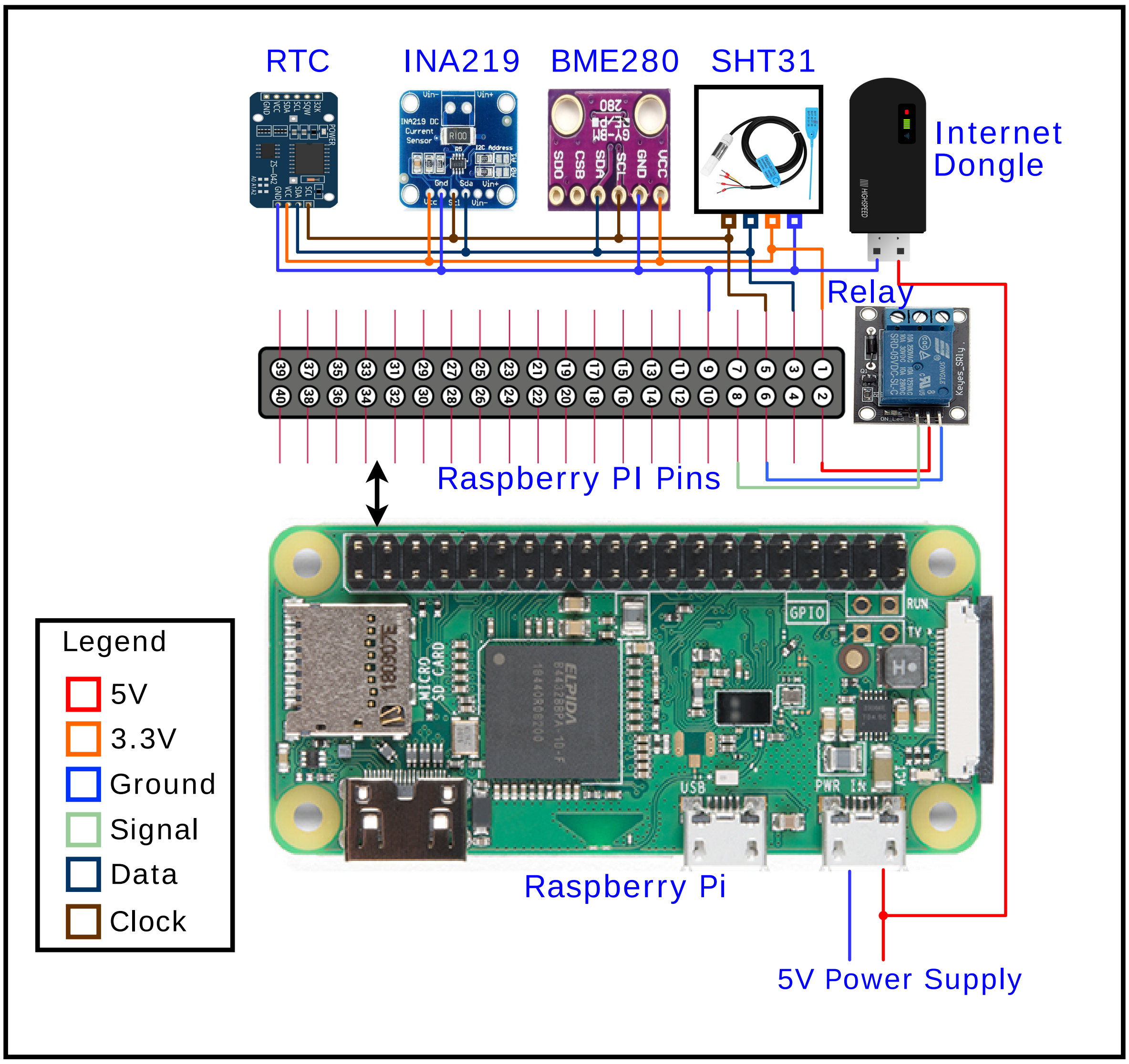


Figure 3.16: Sensors Connection to the raspbery Pi zero

According to I2C protocols, the data line can not change when the clock line is high, it can change only when the clock line is low. The 2 lines are open drain, hence a pull-up resistor is required so that the lines are high since the devices on the I2C bus are active low.

Read/Write Bit :

A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.

Advantages:

Can be configured in multi-master mode.
Complexity is reduced because it uses only 2 bi-directional lines

Limitations:

Slower speed.
Half-duplex communication is used in the I2C communication protocol.

3.17 Analog to Digital Communication Protocol

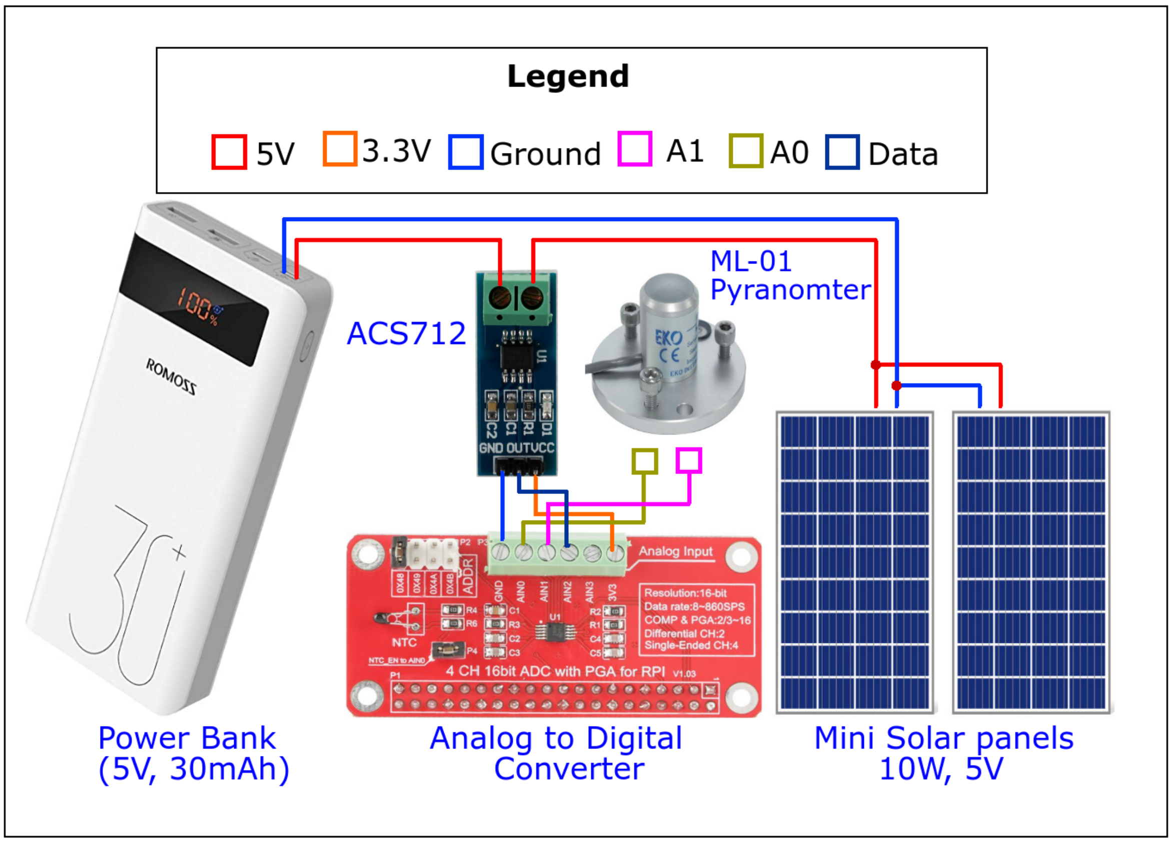


Figure 3.17: ADC connection

An ADC converts a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal. The conversion involves quantization of the input, so it necessarily introduces a small amount of error or noise. Furthermore, instead of continuously performing the conversion, an ADC does the conversion periodically, sampling the input, limiting the allowable bandwidth of the input signal.

The performance of an ADC is primarily characterized by its bandwidth and signal-to-noise ratio (SNR). The bandwidth of an ADC is characterized primarily by its sampling rate. The SNR of an ADC is influenced by many factors, including the resolution, linearity and accuracy (how well the quantization levels match the true analog signal), aliasing and jitter. The signal-to-noise ratio of an ADC is often summarized in terms of its effective number of bits (ENOB), the number of bits of each measure it returns that are on average not noise. An ideal ADC has an ENOB equal to its resolution. ADCs are chosen to match the bandwidth and required SNR of the signal to be digitized. If an ADC operates at a sampling rate greater than twice the bandwidth of the signal, then per the Nyquist–Shannon sampling theorem, perfect reconstruction is possible. The presence of quantization error limits the SNR of even an ideal ADC. However, if the SNR of the ADC exceeds that of the input signal, its effects may be neglected resulting in an essentially perfect digital representation of the analog input signal.

3.17.1 Resolution

The resolution of the converter indicates the number of different values it can produce over the allowed range of analog input values. Thus a particular resolution determines the magnitude of the quantization error and therefore determines the maximum possible signal-to-noise ratio for an ideal ADC without the use of oversampling. The input samples are usually stored electronically in binary form within the ADC, so the resolution is usually expressed as the audio bit depth. In consequence, the number of discrete values available is usually a power of two. For example, an ADC with a resolution of 8 bits can encode an analog input to one in 256 different levels (28 = 256). The values can represent the ranges from 0 to 255 or from 128 to 127 depending on the application.

Resolution can also be defined electrically, and expressed in volts. The change in voltage required to guarantee a change in the output code level is called the least significant bit (LSB) voltage. The resolution Q of the ADC is equal to the LSB voltage. The voltage resolution of an ADC is equal to its overall voltage measurement range divided by the number of intervals:

$$Q = \frac{E_{FSR}}{2^M} \tag{3.1}$$

where M is the ADC's resolution in bits and E_{FSR} is the full scale voltage range (also called 'span'). E_{FSR} is given by

$$E_{FRS} = V_{RefHi} - V_{RefLow}$$

where V_{RefHi} and V_{RefLow} are the upper and lower extremes, respectively, of the voltages that can be coded. Normally, the number of voltage intervals is given by

$$N = 2^M$$

where M is the ADC's resolution in bits.

In our case, the resolution of the ADC is 16 bits and maximum input voltage of $3.3V$.

In order to convert the input voltage from the solar radiation sensor in W/m^2, we use the following equation:

$$E = \frac{ADC_{Diff} \times M}{R \times SS} \tag{3.2}$$

where ADC_{Diff} is the ADC Differential reading, $M = 0.512$ is the multiplier value given by the Manufacturer, R is the resolution of the ADC which is $2^{16} = 65537$ and SS the sensor sensitivity which equals $53.7\mu V/Wm^{-2}$.

Similar approach was also used to measure the current consumed by the remote sensor as the ACS712 current also outputs analog data.

3.18 System power supply and consumption

Two current sensors (Figures : 3.13 and 3.3) are used to measure current supply to the onboard batteries by the two incorporated mini solar panels and current being consumed by the remote sensor. During the day, the sun energy converted into electrical energy is stored one of the batteries. That energy is restored to the other battery during the night. This helps remove fluctuation that could be caused by the variation of the sun energy availability during during the day which could eventually disturb our measurements.

3.18.1 Data Recording and Storage

Every minute, a python script is instantiated by the raspberry pi and all the sensors data are read. Once available, these data are then stored in a local MySQL database along with the corresponding timestamp at which they were recorded. This process repeats every day from 5am to 8pm. Operation is halted from 8pm to 4:59 am to restore electrical power from the storage battery to the main battery which runs the whole system.

3.19 Data Posting Online

Constantly online, the remote sensor makes an API call to google sheet for the timestamp of the last posted data, only after that, it selects from the database any data with a greater

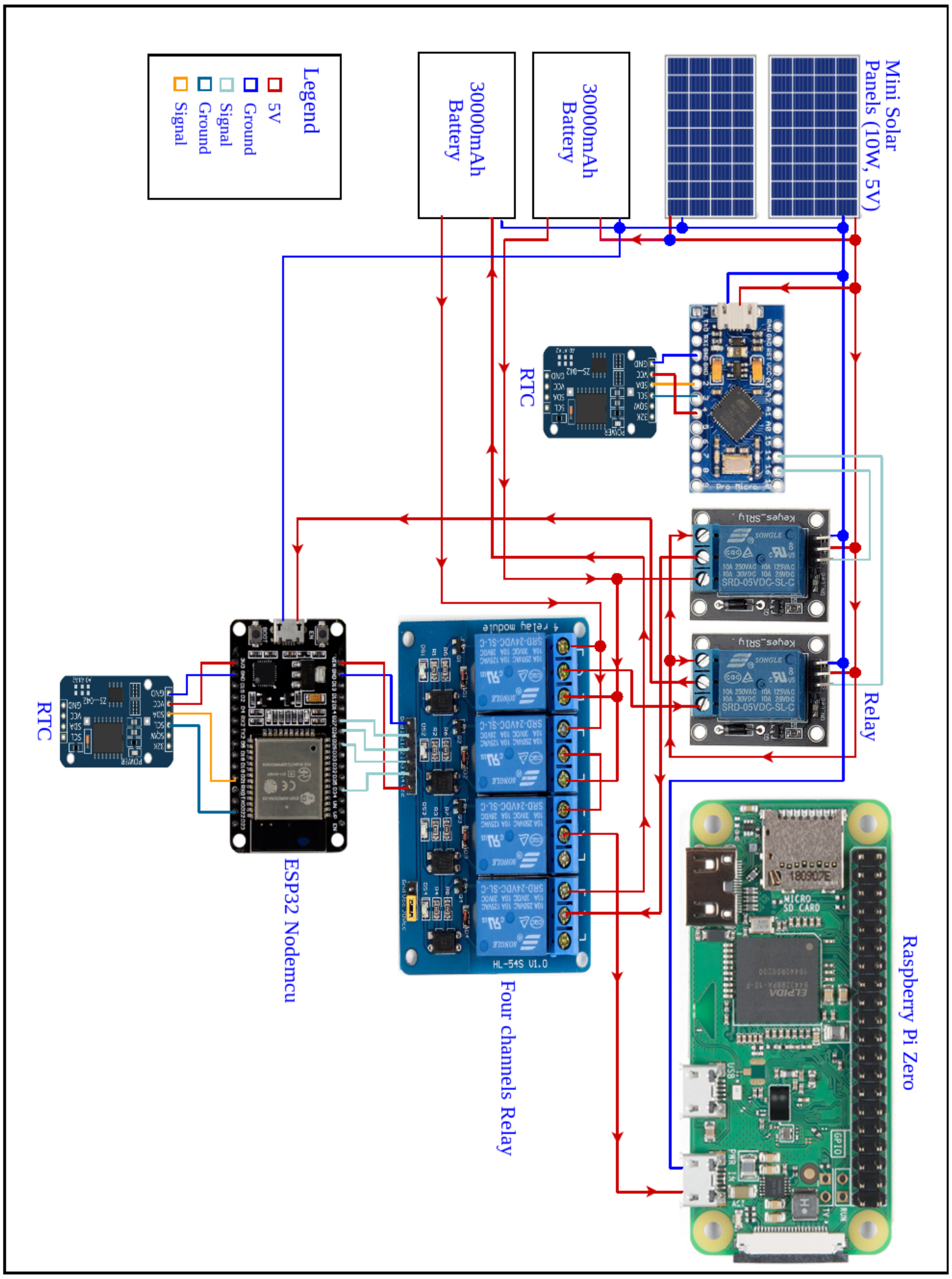


Figure 3.18: Charge and supply schematic design

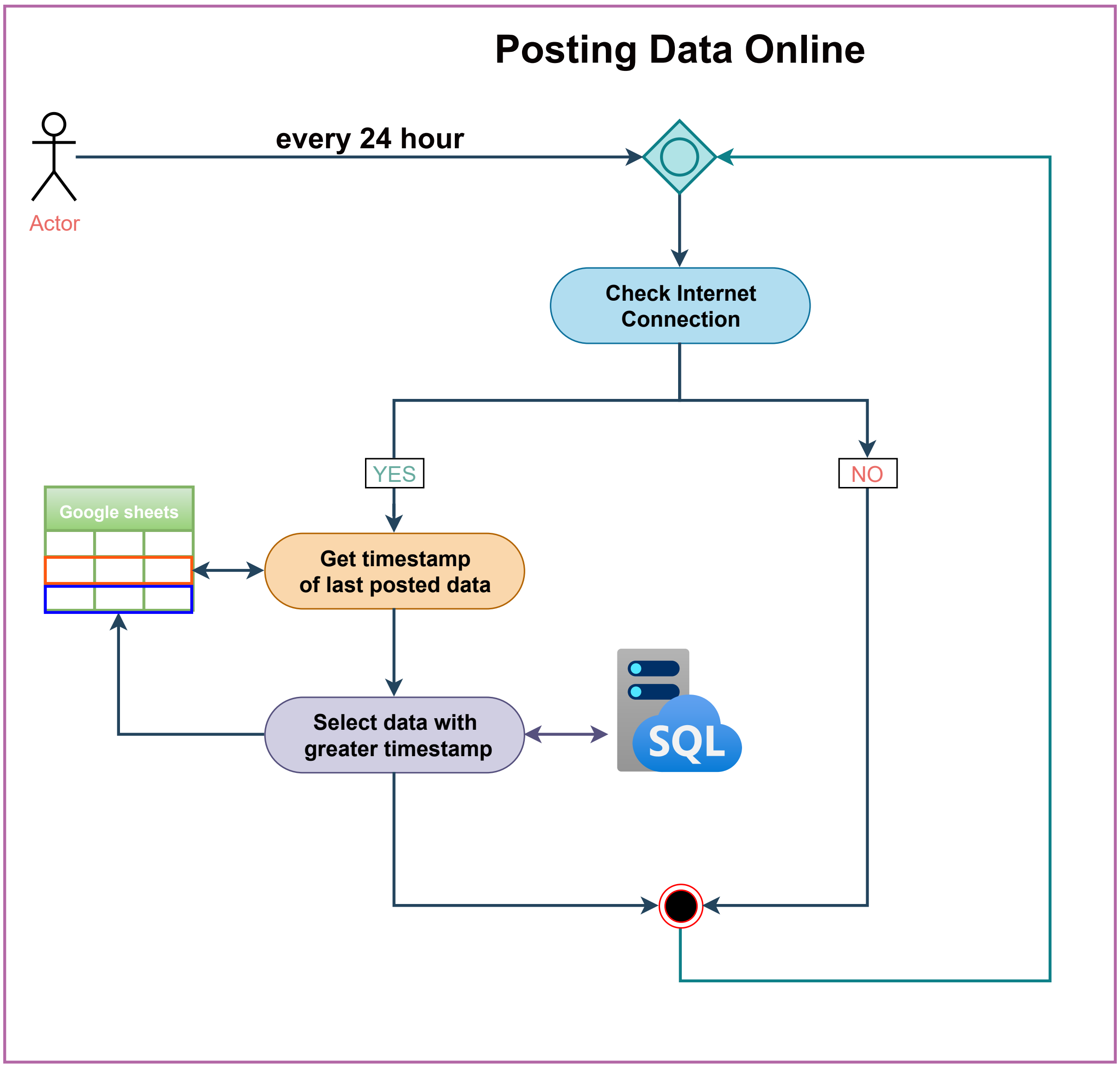

Posting Data Online
every 24 hour
Actor
Check Internet Connection
YES
NO
Google sheets
Get timestamp of last posted data
Select data with greater timestamp
SQL

Figure 3.19: Posting Schematic Diagram

timestamp and posts them back to google Sheet accordingly. This process is repeated every hour from 5am to 19pm.

4 Result and Discussion

A remote sensor for measuring global solar radiation, relative humidity, temperature and pressure data was developed using low-cost and accurate sensors along with a single board computer (Raspberry Pi Zero). Python Programming Language was used to communicate with each sensor. The remote sensor was deployed at the University Campus and the recorded data was primarily stored in a local MySQL database before being collected and posted to a cloud-hosted database every 24h. A measurement of current supply by the two mini-Solar panels to the onboard batteries and current consumption by the system was also carried out. Eihiro Seiki ML-01, a pre-calibrated solar radiation sensor by the Japanese manufacturer EKO Instruments Co., Ltd, was used to measure global solar radiation. BME280 sensor was used to measure temperature, humidity and relative pressure inside the box of the remote sensor, while FS400-SHT3X sensor was used to measure humidity and temperature at 1m above the remove sensor. Hence the deployment location was not ideal for measuring temperature and relative humidity because of the heat storage capacity of the surrounding walls, Japan Weather Station data was used to validate our measurements using Python non-linear curve fitting with the following equation:

$$f(x) = xe^{-a} + b \tag{4.1}$$

where a represents the coefficient, b the intercept, x the measured variable and $f(x)$ the corrected value of the corresponding variable. Table 1 shows a sample of an hourly mean data recorded.

As can be seen from the above table, the remote sensor proved to be fully operational off grid with an average power consumption of 0.280A and a total of 15 hours operation per day from 5 am to 8 pm, with an automatic power-on and power-off ability to help save batteries life during the night. Once fully charged, the two onboard batteries can power up the system for up to 14 days without interruption, which is critical in case of several consecutive days of overcast or rain, time during which, the two onboard mini-solar panels cannot generate enough electricity to provide charge. From space to earth, satellite-based remote sensing is widely used to study the physical properties of distant objects. But

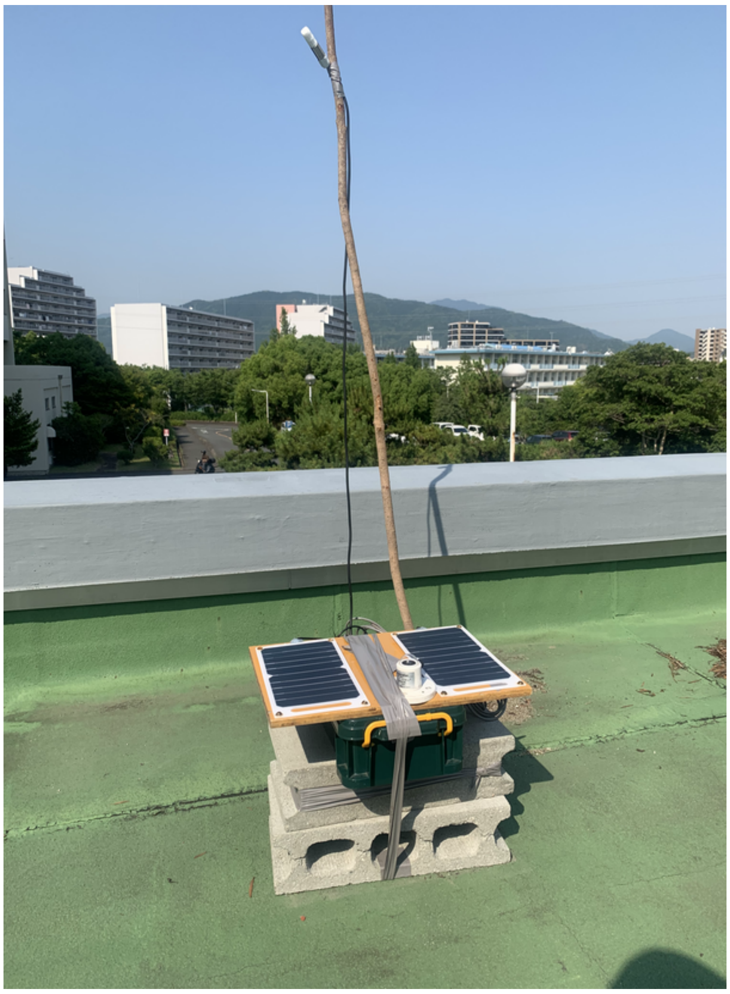


Figure 4.1: Remote Sensor on testing site

Datetime	Radiation $[W/m^2]$	Temperature $[°C]$	Humidity [%]	Pressure $[hPa]$
29/06/2021 08:00 am	439.6	25.6	71.6	1004.8
29/06/2021 09:00 am	578.8	26.6	66.1	1004.6
29/06/2021 10:00 am	598.3	27.2	63.7	1004.5
29/06/2021 11:00 am	435.5	27.2	63.2	1004.4

Table 4.1: Hourly mean of global solar radiation, temperature, relative humidity and pressure data recorded under clear sky condition

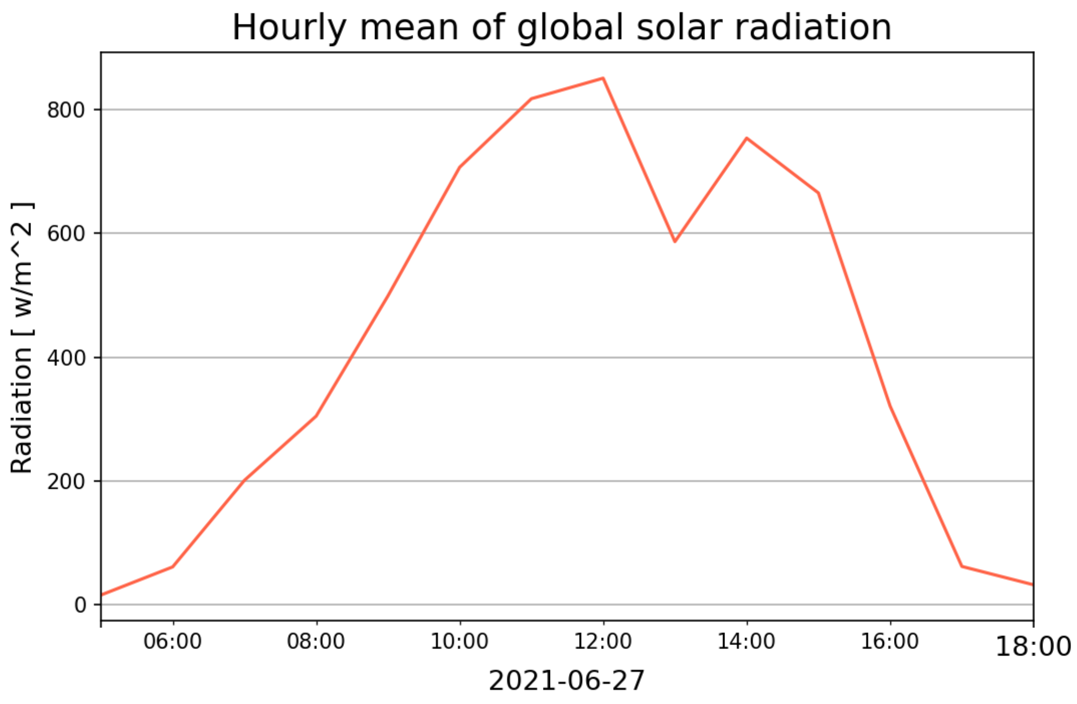


Figure 4.2: Hourly mean of global solar radiation

the accuracy of these data is questionable. Errors in satellite data products are known unknowns. However, quantifying the quality of these products by decomposing the inherent uncertainty components can be a very challenging task [**?**]. Which measurement is more accurate: taking Earth's surface temperature from the ground or from space? Since satellites technically measure neither temperature nor the surface (where people live), it's safe to say that ground thermometers are more accurate than satellite measurements [**?**]. Therefore, as can be seen from the above figures, ground-based remote sensing technology can be used to collect useful information like the amount of solar radiation reaching the earth surface for any location of the globe. It cost around 60,000 Yen to build this remote sensor.

4.0.1 Advantages of Remote Sensing

The ability to collect information and images of the Earth's land, atmosphere, and oceans from aircraft and satellites has been available for quite some time now. This technology was already used during world war II to take aerial photographs of enemies territories and map them as a means of military reconnaissance. The first U.S meteorological satellite was launched in 1960 and the first U.S. civil satellite to observe and monitor the land surface, Landsat, was launched in 1972. Over the last forty years, the U.S. Government has helped advance the state of civil space-based remote sensing.

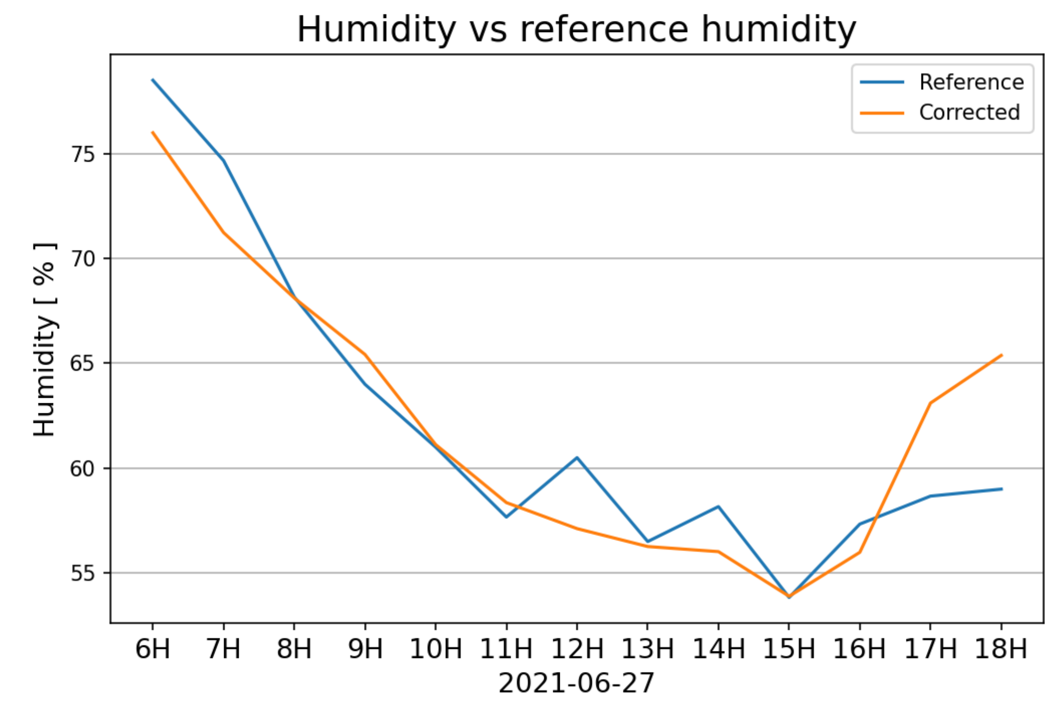


Figure 4.3: Hourly mean of global solar radiation

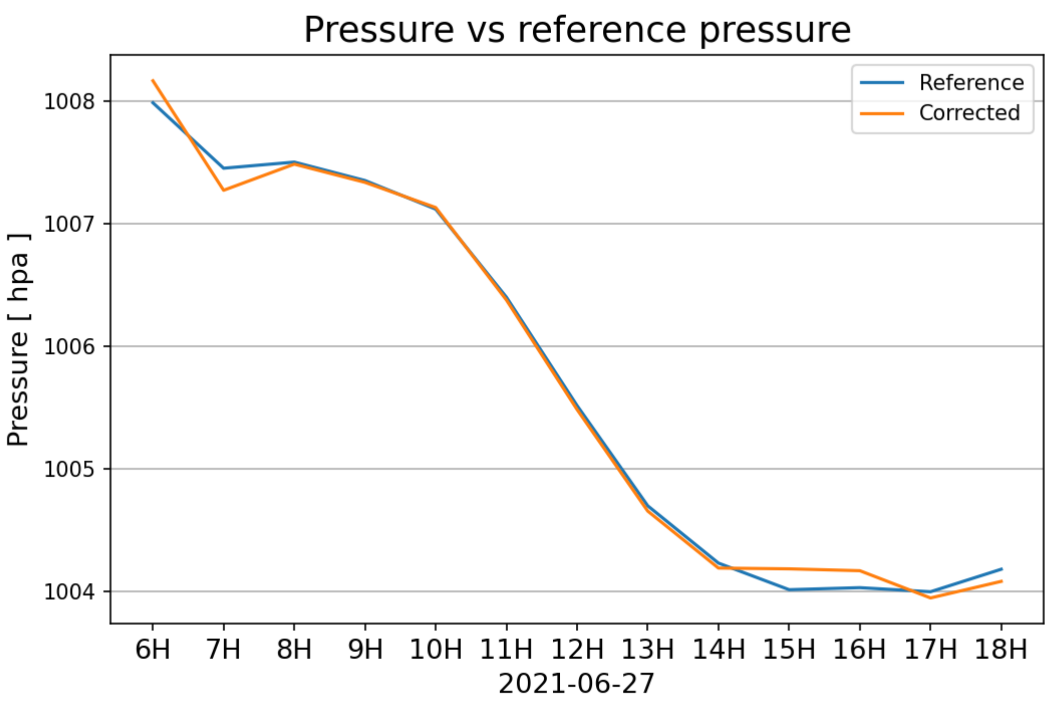


Figure 4.4: Hourly mean of global solar radiation

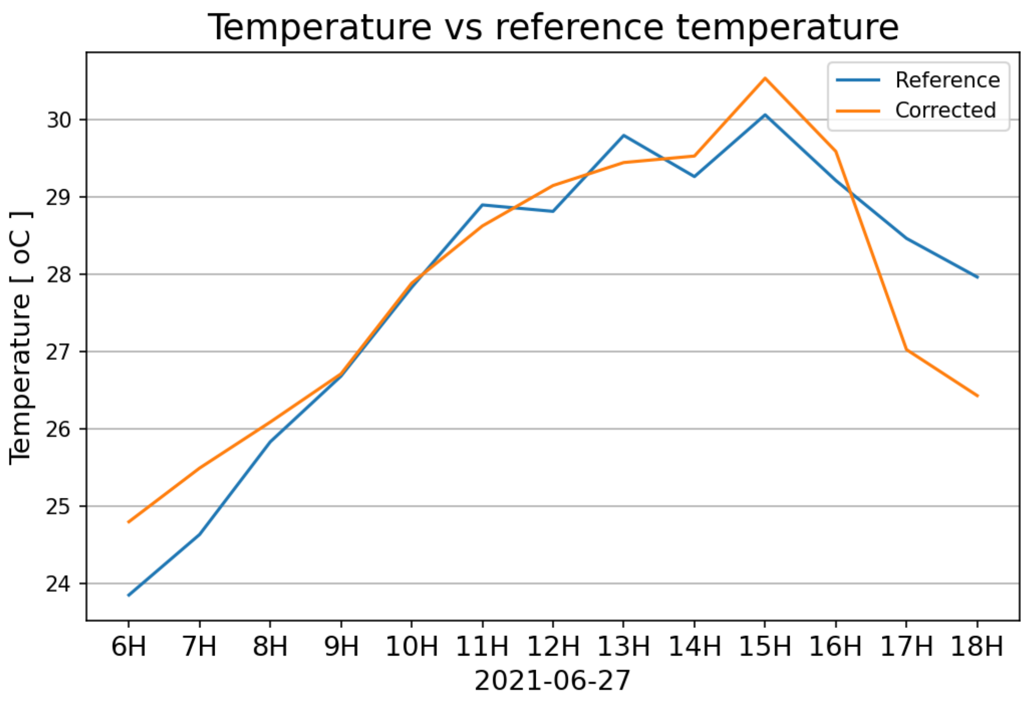


Figure 4.5: Hourly mean of global solar radiation

Through NASA and the National Oceanic and Atmospheric Administration (NOAA), the U.S. Government has launched an ongoing series of increasingly more capable Earth-observing satellites to support an operational weather monitoring service and to conduct research to better understand the Earth's land, ocean, atmosphere, and biosphere, their relationships, and how the Earth system changes over time. In addition, the U.S. Geological Survey has been responsible for archiving and managing civil land remote sensing data. The Land Remote Sensing Policy Act of 1992 set commercial land remote sensing as a U.S. policy goal and included a process to license private remote sensing satellite operators. In the early 1990s the first licenses were issued to private remote sensing operators and by 1999 the first commercial remote sensing satellite was launched. The advantages of remote sensing include the ability to collect information over large spatial areas; to characterize natural features or physical objects on the ground; to observe surface areas and objects on a systematic basis and monitor their changes over time; and the ability to integrate this data with other information to aid decision- making. Remote sensing from airplanes or satellites can be collected at various spatial resolutions [spatial resolution refers to the smallest feature that can be resolved in an image]. High resolution remote sensing images can resolve smaller features (often less than a meter in size) whereas moderate or lower resolution images can detect features in a size range of tens to hundreds of meters or larger. Remote sensing instruments may also acquire data in different spectral bands of the electromagnetic spectrum (e.g., infrared, near-infrared), which provides information, for example, to help classify and categorize vegetation. Data collected in the thermal infrared bands are especially useful for water management. Light detection and ranging (lidar) instruments provide topographic data that can form the basis of digital elevation models.

State and local governments can also benefit from remote sensing information to better monitor land use, assist in transportation planning, and deal with other infrastructure and public safety issues. In addition, commercial enterprises use the data to help support their businesses. For example, real estate companies use imagery to enhance the information provided on real estate property listings, and transportation companies may use remote sensing data to help route trucks.

5 Conclusion

This work is a work in progress. During PhD courses, this actual remote sensor will be used to record a time series ground-based solar irradiance in the Republic of Guinea, West Africa for a Research related to the estimation of its photovoltaic potential in various regions of the country. This analysis is a critical step in achieving energy autonomy for this country with huge natural resources but still lingering into complete darkness in term of electricity, which a crucial component for any reliable social and economic development. As seen from the graphs, remote sensing is a technology that be used to solve pertinent questions in science. This study demonstrates that, with less resources, such technology an be exploited and used in answering fundamental questions underlined by many other researchers in the past.
This technology, although largely misinterpreted and being classified as limited to space and earth observation from a far distant, it is basically a simple and yet technique that could be used in many other areas of our life to help address scientific questions.

References

[Christian, 2005] Christian, E. (2005). Planning for the global earth observation system of systems (geoss). *Space Policy*, 21(2):105–109.

[Fu and Fukumori, 1996] Fu, L.-L. and Fukumori, I. (1996). A case study of the effects of errors in satellite altimetry on data assimilation. In *Modern Approaches to Data Assimilation in Ocean Modeling*, pages 77–96. Elsevier.

[Goldberg et al., 2011] Goldberg, M., Ohring, G., Butler, J., Cao, C., Datla, R., Doelling, D., Gärtner, V., Hewison, T., Iacovazzi, B., Kim, D., et al. (2011). The global space-based inter-calibration system. *Bulletin of the American Meteorological Society*, 92(4):467–475.

[Goldberg, 2007] Goldberg, M. D. (2007). Global space-based inter-calibration system (gsics). In *Atmospheric and Environmental Remote Sensing Data Processing and Utilization III: Readiness for GEOSS*, volume 6684, page 668402. International Society for Optics and Photonics.

[Mason and Simmons, 2011] Mason, P. and Simmons, A. (2011). Systematic observation requirements for satellite-based data products for climate–2011 update.

[Ohring et al., 2007] Ohring, G., Tansock, J., Emery, W., Butler, J., Flynn, L., Weng, F., Germain, K. S., Wielicki, B., Cao, C., Goldberg, M., et al. (2007). Achieving satellite instrument calibration for climate change.

[Ohring et al., 2005] Ohring, G., Wielicki, B., Spencer, R., Emery, B., and Datla, R. (2005). Satellite instrument calibration for measuring global climate change: Report of a workshop. *Bulletin of the American Meteorological Society*, 86(9):1303–1314.

[Pinker et al., 1995] Pinker, R., Frouin, R., and Li, Z. (1995). A review of satellite methods to derive surface shortwave irradiance. *Remote Sensing of Environment*, 51(1):108–124.

[Renne et al., 1999] Renne, D. S., Perez, R., Zelenka, A., Whitlock, C., and DiPasquale, R. (1999). Use of weather and climate research satellites for estimating solar resources. *BIOPROCESS TECHNOLOGY*, 13:171–240.

[Sun et al., 2019] Sun, W., Wielicki, B. A., Baize, R. R., Lukashin, C., Hu, Y., Zubko, E., Videen, G., Kim, S. S., and Choi, Y.-J. (2019). Modeling polarized solar radiation from a snow surface for correction of polarization-induced error in satellite data. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 222-223:154–169.

[Wielicki et al., 2013] Wielicki, B. A., Young, D. F., Mlynczak, M. G., Thome, K. J., Leroy, S., Corliss, J., Anderson, J. G., Ao, C. O., Bantges, R., Best, F., Bowman, K., Brindley, H., Butler, J. J., Collins, W., Dykema, J. A., Doelling, D. R., Feldman, D. R., Fox, N., Huang, X., Holz, R., Huang, Y., Jin, Z., Jennings, D., Johnson, D. G., Jucks, K., Kato, S., Kirk-Davidoff, D. B., Knuteson, R., Kopp, G., Kratz, D. P., Liu, X., Lukashin, C., Mannucci, A. J., Phojanamongkolkij, N., Pilewskie, P., Ramaswamy, V., Revercomb, H., Rice, J., Roberts, Y., Roithmayr, C. M., Rose, F., Sandford, S., Shirley, E. L., Smith, W. L., Soden, B., Speth, P. W., Sun, W., Taylor, P. C., Tobin, D., and Xiong, X. (2013). Achieving climate change absolute accuracy in orbit. *Bulletin of the American Meteorological Society*, 94(10):1519–1539.

[Zarzalejo et al., 2009] Zarzalejo, L. F., Polo, J., Martín, L., Ramírez, L., and Espinar, B. (2009). A new statistical approach for deriving global solar radiation from satellite images. *Solar Energy*, 83(4):480–484.

6 Appendix

To upload the following C code to the Arduino Leonardo mini pro and the ESP32 microcontrollers, we used platform IO IDE embedded inside Atom editor.

Arduino Leonardo Solar Controller

```
#include <Arduino.h>
#define PIN_PI 8
#define PIN_CHARGE 6
#define MINUTE 0
#define wait_time 10000
#define wait_longer 60000

// Date and time functions using a DS3231 RTC connected via I2C and Wire lib
#include "RTClib.h"

RTC_DS3231 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"};

void setup () {
  Serial.begin(9600);
  pinMode(PIN_PI,OUTPUT);
  pinMode(PIN_CHARGE,OUTPUT);
  delay(5000);

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
  }

  if (rtc.lostPower()) {
    Serial.println("RTC lost power, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }

  // When time needs to be re-set on a previously configured device, the
  // following line sets the RTC to the date & time this sketch was compiled
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
}

void PowerPi(){
  digitalWrite(PIN_PI,HIGH);
  delay(wait_time);
  digitalWrite(PIN_PI,LOW);
  delay(wait_longer);
}

void loop () {
  DateTime now = rtc.now();
  int year=now.year();
  int month=now.month();
  int day=now.day();
  int hour=now.hour();
  int minute=now.minute();
  int second=now.second();

    Serial.print("Time: ");
    Serial.print(year);
    Serial.print("-");
    Serial.print(month);
    Serial.print("-");
    Serial.print(day);
    Serial.print(" ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.println(second);

    // if(day%2 != 0){
    int hours[]={8,10,12,14};

    int hour_length=sizeof(hours)/sizeof(hours[0]);
    for (int i=0; i<hour_length;i++){
      if(hour==hours[i] && minute==MINUTE){
        digitalWrite(PIN_CHARGE,HIGH);
        delay(10000);
        digitalWrite(PIN_CHARGE,LOW);
        delay(2000);
        PowerPi();
      }
    }
  // }
  delay(1000);

}
```

ESP32 Controller

```
// Date and time functions using a DS3231 RTC connected via I2C and Wire lib
#include "RTClib.h"
// #define LED_BUILTIN 2
#define PIN_PI 33 //33
#define PIN2 25 //25
#define PIN_ESP 32 // 32
#define PIN_CHARGE 26 // 26
#define wait_time 2000
#define wait_long 30000

RTC_DS3231 rtc;
```

```
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"};

void setup () {
  Serial.begin(115200);
  delay(2000);
  pinMode(PIN2, OUTPUT);
  pinMode(PIN_PI, OUTPUT);
  pinMode(PIN_CHARGE,OUTPUT);
  pinMode(PIN_ESP, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);

  digitalWrite(PIN2,HIGH);
  digitalWrite(PIN_CHARGE,HIGH);
  digitalWrite(PIN_ESP,HIGH);
  delay(5000);
  digitalWrite(PIN_PI,HIGH);
  delay(10000);

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    // Serial.flush();
    // abort();
  }

  if (rtc.lostPower()) {
    Serial.println("RTC lost power, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }

  // When time needs to be re-set on a previously configured device, the
  // following line sets the RTC to the date & time this sketch was compiled
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC wi digitalWrite(PIN2, HIGH);th an explicit date &
     time, for example to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2021, 8, 13, 18, 52, 0));

//
   #-------------------------------------------------------------------------------
DateTime now = rtc.now();
int year=now.year();
int month=now.month();
int day=now.day();
int hour=now.hour();
int minute=now.minute();
int second=now.second();
Serial.print("Time: ");
Serial.print(year);
Serial.print("-");
Serial.print(month);
Serial.print("-");
Serial.print(day);
Serial.print(" ");
Serial.print(hour);
Serial.print(":");
Serial.print(minute);
Serial.print(":");
Serial.println(second);

if(hour >= 20 || hour < 5){
  delay(5000);
  digitalWrite(PIN_PI, 0);
  delay(wait_time);
  digitalWrite(PIN_PI, 1);
  delay(wait_time);
  digitalWrite(PIN_ESP, 0);
  delay(wait_time);
  digitalWrite(PIN_CHARGE, 0);
  delay(wait_time);
  digitalWrite(PIN_ESP, 1);
  delay(wait_time);
  digitalWrite(PIN2, 0);
}else{
    digitalWrite(PIN_ESP, 0);
    delay(wait_time);
    digitalWrite(PIN_CHARGE,0);
    delay(wait_time);
    digitalWrite(PIN_PI, 1);
    delay(wait_time);
    digitalWrite(PIN_ESP, 1);
    delay(wait_time);
}
}

int getMinute(int m){
  return 60000*m;
}

void loop () {
    DateTime now = rtc.now();
    int year=now.year();
    int month=now.month();
    int day=now.day();
    int hour=now.hour();
    int minute=now.minute();
    int second=now.second();

    // digitalWrite(PIN_PI, LOW);
    if(hour >= 5 && hour < 20){
      Serial.print("Time: ");
      Serial.print(year);
      Serial.print("-");
      Serial.print(month);
      Serial.print("-");
      Serial.print(day);
      Serial.print(" ");
      Serial.print(hour);
      Serial.print(":");
      Serial.print(minute);
      Serial.print(":");
```

```
 Serial.println(second);
 ////// Cut Charge and turn on Pi
   if(hour == 5 && minute < 3){
     digitalWrite(PIN2, 1);
     delay(wait_time);
     digitalWrite(PIN_PI, 0);
     delay(4000);
     digitalWrite(PIN_PI, 1);
     delay(wait_time);

      digitalWrite(PIN_ESP, 0);
      delay(wait_time);
      digitalWrite(PIN_CHARGE,0);
      delay(wait_time);
      digitalWrite(PIN_PI, 1);
      delay(wait_time);
      digitalWrite(PIN_ESP, 1);
      delay(wait_time);
      digitalWrite(PIN_PI, 0);
      delay(wait_time);
      digitalWrite(PIN_PI, 1);
     delay(180000);
  }

}else if(day % 2 == 0){
  // Provide Charge to the other battery
  if(hour==19 && minute >= 57){
    delay(20000);
    digitalWrite(PIN_PI, 0);
    delay(4000);
    digitalWrite(PIN_PI, 1);
    delay(wait_time);
    digitalWrite(PIN_ESP, 0);
    delay(wait_time);
    digitalWrite(PIN_CHARGE,1);
    delay(wait_time);
    digitalWrite(PIN_ESP, 1);
    delay(wait_time);
    digitalWrite(PIN2, 0);
```

```
        delay(180000);
    }
  }else{
      // Just cut supply to the PI
     if(hour==19 && minute >= 57){
     digitalWrite(PIN2, 0);
     delay(wait_time);
     delay(180000);
     }
  }

    delay(1000);
}
```

6.1 Code used on the Raspberry Pi Zero

Assuming that the following modules are properly installed and configured:

System level MySQL-server

Python3 libraries

- Adafruit-ADS1x15
- adafruit-circuitpython-bme280
- RPi.GPIO
- smbus2
- mysql-connector-python
- serial
- pi-ina219
- Adafruit INA219
- Adafruit SHT31-D

6.1.1 Major functions that perform reading the sensors data

```
import Adafruit_ADS1x15
import time
import os
import adafruit_sht31d
import adafruit_bme280
import digitalio
import busio
import board
import bme280
import smbus2
from datetime import datetime
from time import sleep
import dht11
import RPi.GPIO as GPIO
from ina219 import INA219
from ina219 import DeviceRangeError
import Adafruit_DHT


# INA219
def consumption():
    SHUNT_OHMS = 0.1
    ina = INA219(SHUNT_OHMS)
    ina.configure()
    # print("Bus Voltage: %.3f V" % ina.voltage())
    # try:
    #     print("Bus Current: %.3f mA" % ina.current())
    #     print("Power: %.3f mW" % ina.power())
    #     print("Shunt voltage: %.3f mV" % ina.shunt_voltage())
    # except DeviceRangeError as e:
    #     # Current out of device range with specified shunt resistor
    #     print(e)
    return round(ina.current(), 2)


def BME280():
    port = 1
    address = 0x76
    bus = smbus2.SMBus(port)
    calibration_params = bme280.load_calibration_params(bus, address)
    data = bme280.sample(bus, address, calibration_params)
    main_data = [round(data.temperature, 2), round(
        data.humidity, 2), round(data.pressure, 2)]
    return main_data

# SHT31
# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()
sensor = adafruit_sht31d.SHT31D(i2c)

def sht_sensor():
    temp = round(sensor.temperature, 2)
    humidity = round(sensor.relative_humidity, 2)
    data = [temp, humidity]
    return data



# Radiation && supply
# Create an ADS1115 ADC (16-bit) instance.
adc = Adafruit_ADS1x15.ADS1115()



def get_radiation():
    GAIN = 16
    value = adc.read_adc_difference(0, gain=GAIN)
    value = value*0.512/65534
    #value = value*1.005236788+0.0000001218
    #value = value/0.00000731
    value = value/0.0000537
    radiation = round(value, 1)
    return radiation



# Supply
def power_supply():
    GAIN = 2
    supply = ((3.3/2)-(adc.read_adc(3, gain=GAIN)
                    * (4.096/65534))*10 # 1000mA = 100mV
    return round(supply, 2)
```

6.1.2 Reading and storing the sensors data

```
import mysql.connector as mysql
import os
import sys
from allSensors import *
from time import sleep

try:
    con = mysql.connect(host=os.environ.get("HOST"), user=os.environ.get(
    "USER"), password=os.environ.get("PASSWORD"), db=os.environ.get("DB"))
    c = con.cursor()
except Exception as e:
    print(e)
    # pass

# Create data storage table
c.execute('create table if not exists usbdata(_id int(11) auto_increment
    primary key, date DATETIME not null, radiation real not null, temperature
    real not null, humidity real not null, pressure real not null, sht_temp
    real not null, sht_hum real not null,cons real not null,spl real not
    null)')

# Data Summary table
sql = '''
CREATE VIEW if not exists hourlySummary
as select date(date) as date,
hour(date) as hours,
round(avg(radiation), 2) as radiation,
round(avg(temperature), 2) as bme_temp,
round(avg(humidity), 2) as bme_hum,
round(avg(pressure), 2) as bme_press,
round(avg(sht_temp), 2) as sht_temp,
round(avg(sht_hum),2) as sht_hum,
round(avg(cons),2) as cons,
round(avg(spl),2) as supl
from usbdata group by hours
'''
c.execute(sql)



def insert_records(rad, temp, hum, press, sht_temp, sht_hum, cons, spl):
    sql = f'INSERT INTO usbdata(date,radiation,
        temperature,humidity,pressure,sht_temp, sht_hum,cons,spl)
        values(now(),{rad},{temp},{hum},{press},{sht_temp},{sht_hum},{cons},{spl})'
    c.execute(sql)
    con.commit()



try:
    cons = consumption()
    temp, hum, press = BME280()
    sht_temp, sht_hum = sht_sensor()
    spl = power_supply()
    rad = get_radiation()
    comand=f'{rad},{temp},{hum},{press},{sht_temp},{sht_hum},{abs(cons)},{spl}'
    print(comand)
    # if cons and spl and rad and temp and hum and press and sht_temp and
        sht_hum:
    insert_records(rad, temp, hum, press, sht_temp, sht_hum, cons, spl)
except Exception as e:
    sleep(30)
    os.system("sudo reboot")
```

6.1.3 Providing internet service

```
 import RPi.GPIO as GPIO
from time import sleep
from datetime import datetime



# Set our GPIO numbering to BCM
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
# Define the GPIO pin that we have our digital output from our sensor
    connected to
pin = 21
GPIO.setup(pin, GPIO.OUT)

day = int(datetime.now().strftime("%d"))



GPIO.output(pin, GPIO.HIGH)
sleep(180)

import send

GPIO.output(pin, GPIO.LOW)
```

6.1.4 Posting recorded data to google sheet

```
#!/usr/bin/python3
import socket
import requests
import os
import sys
from time import sleep
import mysql.connector as mysql

# 15/05/2021 added links
ref_url =
    "https://script.google.com/macros/s/AKfycbzQ4dK-Gn0C74C_0wWHQR6f9qJI1Z8Qzd9mmhNLSqpYQ-PA
post_url =
    "https://script.google.com/macros/s/AKfycbwXk_2FKHatCgvFuNUNy0QFrU6BBnP7g2IZIURFeee6JymB



def check_conection():
    try:
        socket.create_connection(('google.com', 80))
        return True
    except Exception:
        return False


conn = check_conection()

try:
    con = mysql.connect(host=os.environ.get("HOST"), user=os.environ.get(
        "USER"), password=os.environ.get("PASSWORD"), db=os.environ.get("DB"))
    c = con.cursor()
except Exception as e:
    print(e)

if conn:
    r = requests.get(ref_url)
    if r.status_code == 200:
        date = r.json()['date']
        date_str=str(date)

        sql = f'select date,radiation, temperature,humidity,pressure,
            sht_temp,sht_hum, cons, spl from usbdata where date >
            str_to_date("{date}","%Y-%m-%d %H:%i:%s") limit 5000'
        c.execute(sql)
        r = c.fetchall()

        data = []
        if len(r) > 0:
            for i in r:
                date = str(i[0])
                gb_r = i[1]
                temperature = i[2]
                humidity = i[3]
                pressure = i[4]
                sht_temp = i[5]
                sht_hum = i[6]
                cons = i[7]
                spl = i[8]

                data.append({
                    'date': date,
                    'gb_r': gb_r,
                    'temperature': temperature,
                    'humidity': humidity,
                    'pressure': pressure,
                    'sht_temp': sht_temp,
                    'sht_hum': sht_hum,
                    'cons': cons,
                    'spl': spl
                })
            requests.post(post_url, json=data)
```