**IoT Farm Energy Management (Solar) System**

Bumjun Oh, Hyeongyu Lee, Jiacheng Wang, Hanbyeol Lee, Sangje Jeong, Minji Lee, and Eric T. Matson

Author’s affiliation (will be updated later)

Chung-Ang University & Purdue University

addresses (will be updated later)

wjc847867030@gmail.com, dhsys199@gmail.com,  yhb18340@gmail.com, dlgusrb09@gmail.com, jun915350@gmail.com, lee3450@purdue.edu  and  ematson@purdue.edu

***Abstract—* Amongst renewable energy sources, solar power energy is gaining popularity thanks to low carbon dioxide emission and increasing efficiency. Populations who are taking part in agricultural activities tend to rely on electricity generated from a long distance and solar power energy is not an exception. Photovoltaic(PV) solar panels are one of the ways to harness solar power into renewable energy with only a major initial cost but low maintenance. However, due to the tendency of solar energy systems being far away from the electricity users, energy health monitoring in real time and maintaining recovery from faults is the key goal. This report suggests a way to interact and maintain an off-grid solar energy system from many miles away.**



Fig. 2. Photovoltaic solar panel

***Index Terms—* CAT.5e, CH340, CLI, COM, DC, Ethernet cable, LoRa, LoRaWAN, Modbus protocol, MPPT, MQTT, Node-Red, PV, Register, RJ45, RS232, RS485, Serial communication protocol, USB,  database, server, solar charge controller, AWS, Node-RED, MongoDB**

1. INTRODUCTION

The purpose of this project is to build a module that monitors the real-time status of an off-grid solar energy system via communication with a solar charge controller and LoRa device to send data through a long distance. The main use of the application is rural areas with difficult access to solar energy due to distance.

1. BACKGROUND
2. ENVIRONMENT SETTING
3. *Raspberry Pi Setting*

The application is currently using Raspberry Pi 3 Model B running the 32-bit Raspberry Pi OS. It has Python 3 and Python 3 IDLE for Python script editing and running. It is required to have various python modules and libraries to execute communication with solar charge controller.

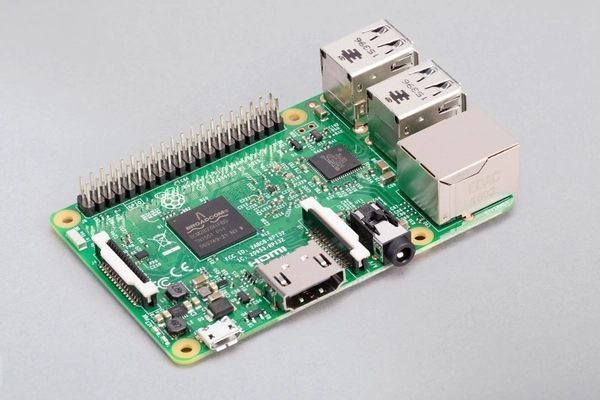


Fig. 1. Raspberry Pi 3 Model B

1. *Photovoltaic Solar Panel*

The photovoltaic solar panel (shortly PV panel ) used in this application generates 30W(watts) with 18V output. It generates electricity with the solar cells attached to it and sends electricity through direct current(DC) electric wires.

1. *MPPT solar charge controller*

The equipment that is used in this project is MPPT solar charge controller manufactured by a company named EPSOLAR, model named XTRA3120N. The controller receives electricity generated from the solar panels. Simultaneously, the controller either charges connected batteries or gives output load for immediate use of electricity. The controller has a micro computing unit(MCU) inside that monitors and saves real-time data that has been generated by the flow of electric current. The data are stored in 16-bit registers and single-bit coils. This equipment can have communications with other devices via RS485 serial communication with Modbus protocol. It uses a physical port that is widely used for internet connection, such as the RJ45 port for COM.



Fig. 3. EPEVER XTRA3210N

1. *12V Lead-acid Battery*

Solar charge controller needs to save power in a battery that has the compatible voltage input and ampere according to the controller. The controller can charge batteries with input voltage of 38v. However, since the solar panel’s optimal output voltage is 18v, a battery with lower input volt rate is needed.

Fig. 5. 12v 2a lead-acid battery



1. *Improvised USB to RS485 cable*

The EPSOLAR company produces and sells communication cables that can allow users to connect between PCs and controllers. However, the project uses an improvised and an economical way that can function identically to the official product. Here are the materials to make up the USB to RS485 cable:

* 1. *CAT.5e:* Category 5eethernet cable should be peeled off on one end to make connection with the USB to RS485 module which is described after.

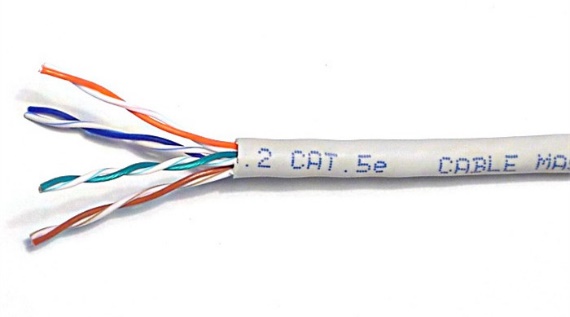


Fig. 6. CAT. 5e cable

* 1. *USB to RS485 module:* This module has several components that allow data flow from RS485 Modbus protocol to USB. The improvising process requires connecting CAT.5e wires number 4(RS-485-B) and number 6(RS-485-A)( green and blue respectively), to the screw terminal with ‘A’ and ‘B’ accordingly.

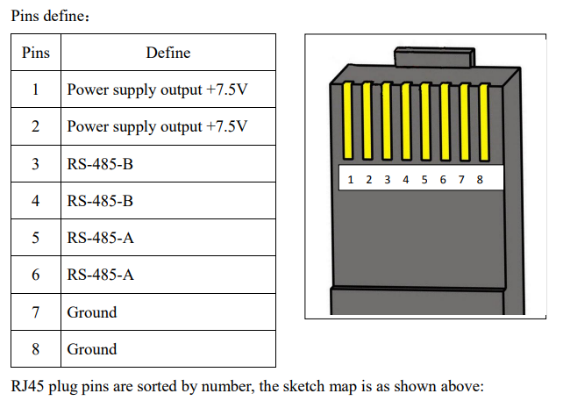


Fig. 7. Pin layout of CAT.5e cable

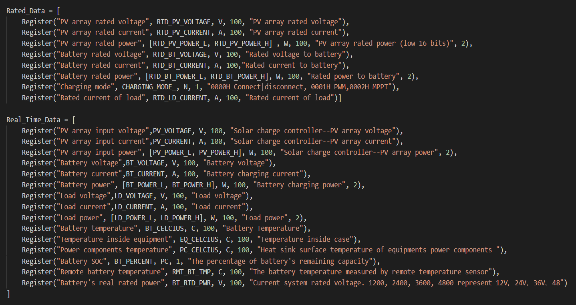


Fig. 11. Register arrays



Fig. 8. USB to RS485 module

1. *Python scripts for communication between controller and Pi*
   1. *Register addresses:* The solar charge controller uses 16-bit addresses and single-bit coils to as a Modbus protocol. There are read input register, read and write holding registers, read and write coils and read discrete inputs. The datasheet of these addresses was saved in the script as constants for easier use.

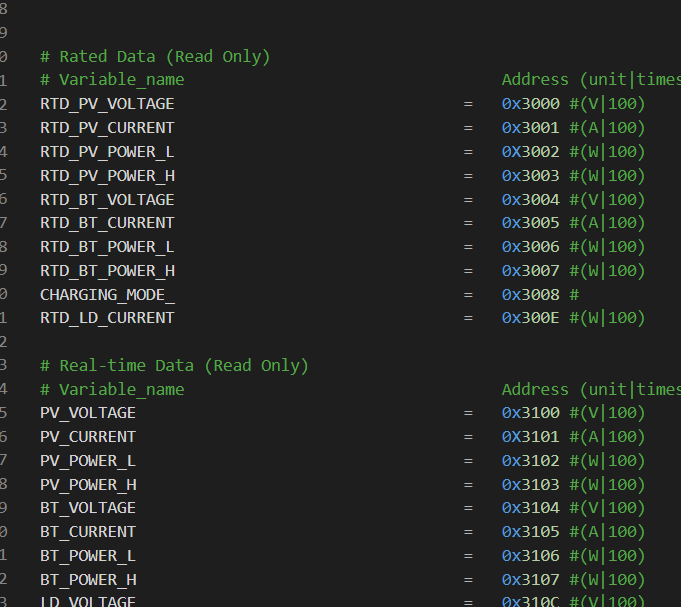


Fig. 9. Register address constants example

* 1. *Register object arrays:* Using the previous addresses, all related data such as variable name, unit, descriptions are saved to an instance of a class named ‘Register’. Collection of those instances form arrays by their categories.

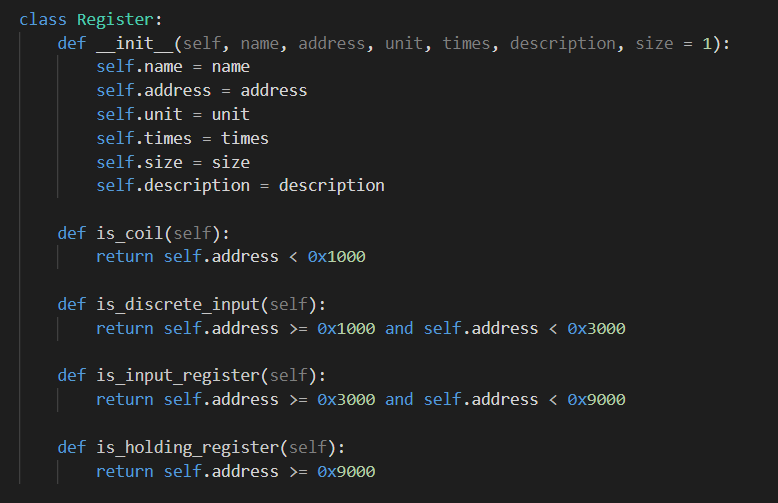


Fig. 10. Register class

* 1. *Main:* The main python script firstly checks the connection with the solar charge controller and if connection is made, then makes another connection to the database server (MongoDB) to upload generated data in real-time. When connection with the database server is finally done, the running script is ready to receive data from the controller and save individual data from each register to a dictionary. Iteration in the register arrays allow to refer to different data representing different things like the voltages and currents of PV panels and batteries. Once the dictionary is filled with data that were requested, the dictionary is finally sent to the database with respective time stamp and object ids.

1. *AWS EC2 Instance for Deployment*

AWS EC2 Ubuntu Instance was used as a remote server.

Amazon EC2 Instance is a virtual server in Amazon’s Elastic Compute Cloud for running applications on the AWS infrastructure.[1]

AWS is a comprehensive, evolving cloud computing platform. EC2 is a service that enables business subscribers to run application programs in the computing environment.

**

Fig. 12. AWS EC2 Instance

1. *Node-Red Application for Data Visualization*

Electricity, Temperature, Humidity and other infos had to be shown in detail by web Application so that User can easily see the current status of Solar panel system.

Node-Red Framework was used for Data Visualization which shows the data extracted from database in real-time.

Node-Red is a flow-based development tool for data visualization. It provides web-browser based flow editor.

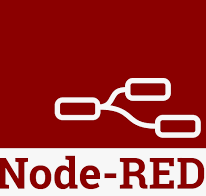
**

Fig. 13. Node-Red FrameWork

1. *MongoDB Database for DataBase*

MongoDB is an open source, nonrelational database management system that uses flexible documents instead of tables and rows to process and store various forms of data.[2]

Solar Panel infos coming from the solar controller is saved to MongoDB through Rasberry Pi.

Database saved in MongoDB is displayed by Node-Red Framework

**

Fig. 14. MongoDB Database

1. DATA AND RESULTS
2. *Data from solar charge controller*

Real-time data with Real-time status of the controllers are requested and sent to Raspberry Pi and again sent to the database.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. *AWS EC2 Server Running*

Virtual server in Amazon’s Elastic Compute Cloud runs applications on the AWS infrastructure.

<http://18.116.64.150:1880> is the link of the Solar Energy Management System.

텍스트이(가) 표시된 사진

자동 생성된 설명

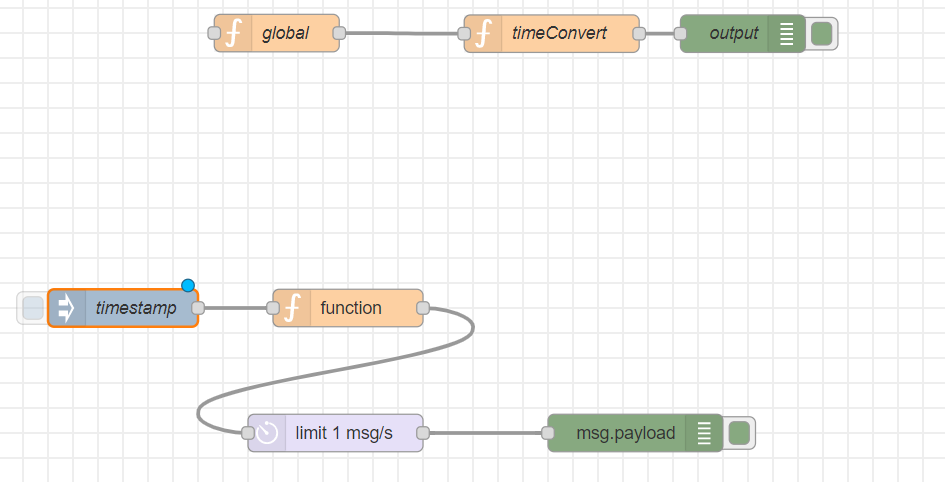
1. Node-RED

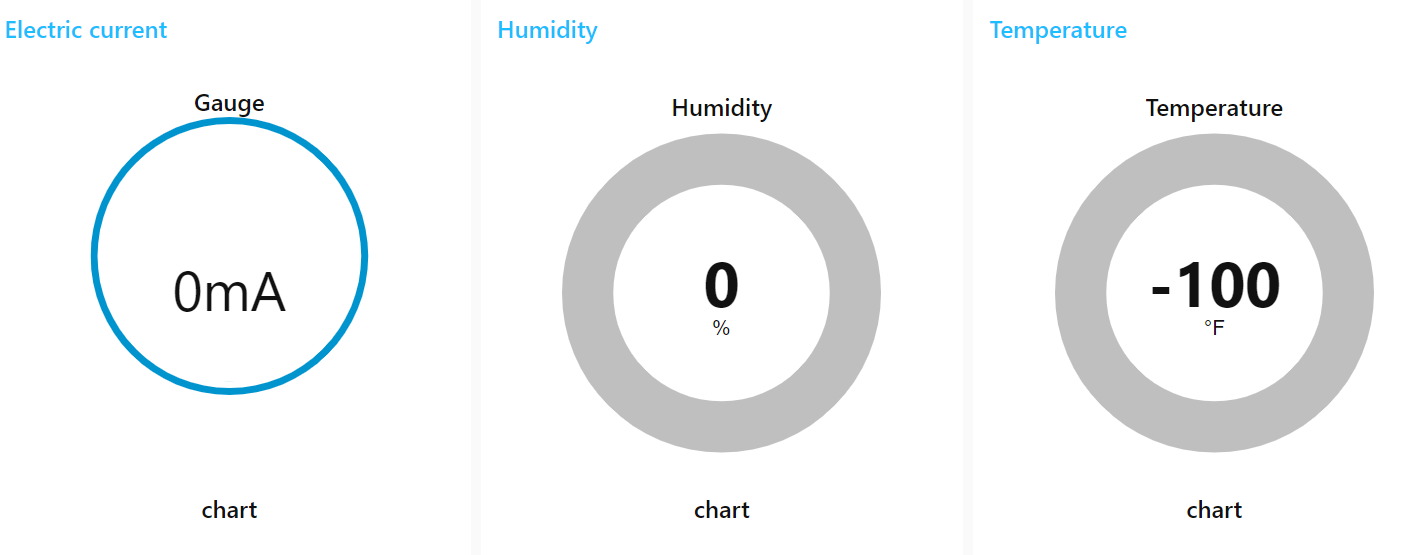
Interacting with MongoDB, Node-red shows the controller’s Real time data using AWS.

The Node-RED pages consist of main tracer and gauge. Main tracer shows the real time data from the MPPT controller. The data from the controller are sent to MongoDB which is a storage in this system.

Node-RED is connected with database and can show the real time status and real time data. Gauge shows humidity and temperature sensor’s data from the Raspberry pi.

Displayed data is updated per 30 seconds, according to the timestamp.





1. DISCUSSION
2. Solar Panel to Controller
3. *Photovoltaic(PV) Solar Panel*: The Solar Energy could be an electric energy to use eco-friendly in our life. Photovoltaic(PV) solar panels are chosen to receive and charge the energy.
4. *The MPPT solar charge controller*: This controller either charges connected batteries or gives output load for immediate use of electricity. The controller has a micro computing unit(MCU) inside that monitors and saves real-time data that has been generated by the flow of electric current. The data are stored in 16-bit registers and single-bit coils. It uses a physical port that is widely used for internet connection, such as the RJ45 port for COM.
5. Controller to Raspberry pi
6. Raspberry pi
7. *MPPT controller*: This equipment can have communications with other devices via RS485 serial communication with Modbus protocol.
8. *Connection*: RS485 module has several components that allow data flow from RS485 Modbus protocol to USB. The improvising process requires connecting CAT.5e wires number 4(RS-485-B) and number 6(RS-485-A)( green and blue respectively), to the screw terminal with ‘A’ and ‘B’ accordingly.
9. Setting up in LoRa Network
10. Raspberry pi to MongoDB
11. Raspberry pi: Raspberry Pi 3 Model B running the 32-bit Raspberry Pi OS has Python 3 and Python 3 IDLE for Python script editing and running.
12. MongoDB
13. *Python script*: The main python script firstly checks the connection with the solar charge controller and if connection is made, then makes another connection to the database server (MongoDB) to upload generated data in real-time. When database server is connected, the running script is ready to receive data from the controller and save individual data from each register to a dictionary.
14. MongoDB to Node-RED
15. Node-RED

Node-RED Framework was use for visual programming. Datas saved MongoDB is requested by Node-Red.

As a response, Node-RED receives the datas of solar panel which was saved by Rasberry pi.

There are 2 pages showing the Infos. 1st is Main Tracer and 2nd is Gauge.

Main Tracer shows the Solar, Battery, DC Load, Controller infos.

Additionally it shows data on amount of Energy generated and consumed.

Gauge page shows current temperaute and humidity info of solar panel.

All the infos are updated every 30 seconds as Node-RED request new info to the MongoDB per 30 seconds.

User is able to see the most recent data of solar panel, which is conducive to efficiently managing the solar panel system.

1. Connection
2. CONCLUSION
3. ACKNOWLEDGEMENTS
4. REFERENCES

[1] What is AWS EC2 Instance ? <https://searchaws.techtarget.com/definition/Amazon-EC2-instances>

[2] What is MongoDB Database? https://www.ibm.com/cloud/learn/mongodb