Grand Plan Outline

Author: William Kerr

# Introduction:

Goals:

My goal is to create a monitoring system for a solar-powered greenhouse. The requirements of the greenhouse are to store plants at a reasonable temperature while providing power for ancillary devices such as pumps and fans, provide means to monitor said system, and power said monitoring system with batteries recharged from rooftop solar panels providing power.

UCSC has 24 solar panels that provide 1.5kWhr of power per day. The power will be delivered to the system properly using MPPT Solar Charge Controller, provided by Morningstar. Each charge controller can only take 150V, and the total amount of solar panels we have sums up to 196V. Therefore, we will have two charge controllers working in tandem to charge a battery pack. These charge controllers can charge the same battery pack without interference, according to Morningstar. Both charge controllers will have a remote temperature sensor connected to it in order to monitor the amount of charge being delivered to the batteries. The charge controllers will deliver power to a battery pack, but these charge controllers cannot regulate all the batteries by themselves.

The battery pack consists of 8 batteries, each with a maximum capacity of 180Ah and nominal voltage of 3.2V, provided by CALB. The batteries will be connected in series in groups of four, summing up to 12.8V per group. The battery pack will be stabilized, its output current regulated, and its temperature regulated by a Battery Management System (BMS), provided by EMUS. The BMS consists of a central control module, two CAN Cell Group Modules, two top isolators, two bottom isolators, and 8 Cell Isolator boards. All components will be controlled by a CAN bus originating from the central control module, that EMUS has programmed for its end users.

The BMS will control the charging of the batteries using variables such as the voltage, current, temperature, and charge of each individual cell, and modify internal variables such as the battery balancing rate accordingly to prevent damage to the batteries. If an individual battery dies, the BMS system will alert you with its monitoring system. The monitoring system can monitor from each battery: current drawn, individual voltage, temperature, and more. The monitoring system will calculate how much charge the battery has based on its own internal logic system. If the charge of any battery gets too low, it will focus the incoming charge from the solar panels on that particular battery. If the maximum charge of any battery gets too low, it will let you know with an alert. The monitoring system will monitor the batteries’ temperature. The BMS will cool the batteries if they get too hot, and warm the batteries if they get too cold. The monitoring system will let you know with an alert. There are other variables the monitoring system monitors. Check the components document for details.

The battery pack will power the internal computer system inside the greenhouse, an AC outlet somewhere in the greenhouse, and the 12V distribution nodes scattered throughout the greenhouse. Battery power will be converted from DC to AC using a power converter called the Suresine inverter, provided by Morningstar, that will be located near the Charge Controller. The AC power outlet will have a maximum power output of 300W. The 12V distribution nodes will distribute power to various devices that one might need for a greenhouse experiment, such as an air pump, hanging lights, a heater, a water pump, etcetera.

The internal monitoring system consists of a Raspberry Pi main microcontroller, which will be located next to the Charge Controllers for convenience, a 2G GSM shield attached to the main microcontroller, a 7” touch screen for convenience, the faculty sensors, the student sensors, and the Raspberry Pi slave microcontroller(s).

The faculty sensors will need to monitor light, humidity, and temperature, as per the client request. This can be achieved with light sensors and humidity & temperature combination sensors. Since I have the liberty to choose which sensors I will use, I will choose sensors that utilize the I2C serial protocol. This way, a daughterboard can be designed that will enable the main microcontroller to monitor up to 8 I2C devices. The daughterboard will have detachable inputs, so it is possible to change the sensors utilized in the greenhouse.

The main Raspberry Pi microcontroller is powerful enough to suit our needs. It will be running a Linux system called Raspbian. I will install a touch screen display to the main microcontroller, so if a faculty member needs to debug it, they can see what they’re doing without needing to connect to the Raspberry Pi using an SSH connection. In case of an emergency, I will include a way to connect to the main Raspberry Pi using an SSH connection and the proper permissions. The main Raspberry Pi and the touch screen will have an enclosure to protect it from the elements. The main Raspberry Pi’s tasks will consist of reading the sensor values from the faculty sensors using the I2C serial protocol, receiving a student sensor value package from the slave computers located around the greenhouse over the Bluetooth protocol, reading and storing the values generated from the Charge Controller’s internal PLC, reading and storing the values generated from the Battery Management System’s internal logic system, compiling all of that data, and then sending it over a 2G TCP connection directly to a server somewhere at UCSC with a GSM/GPRS daughterboard attachment. The 2G connection will be provided by Ting, as per client request.

The student sensors can consist of almost any type of sensor, and it seems random at this point in time. Here are some possibilities I can envision: water temperature sensors, soil moisture sensors, light sensors, humidity & temperature combination sensors, pressure sensors. Since it is impossible to know what serial protocol the student sensors will utilize, I will incorporate the most popular serial protocols into a daughterboard I will design, including I2C, SPI, UART, and 1-wire. I will incorporate a lot of UART ports into my daughterboard design, so that way if their sensor uses a strange protocol not listed here, a faculty member or another engineering student can make a bridge from that protocol to UART, write some code to read their sensor, and the slave microcontroller will be able to include their sensor in the data packet it sends to the master microcontroller.

The slave Raspberry Pi microcontrollers will be powerful enough to send data packets to other microcontrollers using the Bluetooth protocol, read sensor values over a variety of serial protocols, and process that data and compile it into a data packet. The slave Raspberry Pi will have an enclosure to protect it from the elements, but it will not have a sensor. It will be programmable from the main Raspberry Pi. The data packets will contain this type of data: name of owner, name of sensor, type of sensor, protocol utilized, value itself, what unit it’s measured in, time and date of when the value was taken.

The main Raspberry Pi microcontroller will be sending a packet of data, utilizing the GSM daughterboard, to a UCSC server somewhere. Once the server receives the data, it will store it in a SQL-style database. That server will also be connected to the internet, so users can connect to it from their phones. When a user connects to this server, the server will serve a website back to the user. The user can then use that website to send a request to the server asking for student sensor data, faculty sensor data, Battery Management data, or Charge Controller data. The user can add a format to their request, such as a graph over time, instantaneous data, data from the last time the computer has polled, data from a specific point in time, etc.

If the user requests a graph over time, they will need to specify what type of data they are requesting, Y-Axis units (i.e. V, mA, etc.), X-axis data (i.e. units of time), and range and scale for both X and Y axis for each dataset they request. Optional: a caption, a title, names for the X and Y axis. The server will return a webpage containing their graph, and some graph metadata: how many data points are returned, what the scale and range are, and a legend if they requested multiple data sets.

# Sources:

The Charge controller can work in tandem with another charge controller of the same brand:

<https://www.morningstarcorp.com/parallel-charging-using-multiple-controllers-separate-pv-arrays/>