Greenhouse Monitoring Project

University of California, Santa Cruz



Sustainability Lab

Electrical Engineering EE195, 2019

William Kerr

Adapted from Last Year

Contents

**Part 1: Framing the Project**

1a) Problem Statement

1b) Background and Feasibility

1c) Bibliography

1d) Stakeholder Analysis

1e) Metrics and Criteria for Success

1f) Idea Generation

1g) Design Specifications and Trade-offs

**Part 2: Project Proposal**

2a) Motivation and Final Problem Statement

2b) Scope of Work

2c) Methodology: Proposed technical solution

2d) Methodology: Timeline and major milestones

2e) Impact Assessment Strategy

2A) Parts List and Budget

**Part 3: Appendices**

Resource Requirements

Supporting Documentation

Business Model Canvas

**To the Reader: I am not good at professional language yet. I based this off the previous team’s work, and I haven’t edited my work from a writer’s perspective yet.**

**Part 1: Framing the Project**

1a) Problem Statement

The University of Santa Cruz Sustainability-Lab has two greenhouses that are used to facilitate research, which a team of engineering students previously worked on to provide solar power and sensing capabilities. The previous team successfully provided power to the greenhouses, but the ability to monitor the system needs to be completed. Our team will continue where the last team left-off to provide the ability to extract the power data, connect a master microcontroller to the remote server, connect slave microcontrollers to the master microcontroller, connect new sensors easily, and read the data on a website. To achieve these goals, we must re-design and modify the current sensor node system to provide the intended readings, modularity, and system practicality for its intended use of research.

1b) Background and Feasibility

Based on our problem statement, we can start by identifying topics that are relevant. Topics include Wireless Sensor Networks, Bluetooth, Web Design, Microcontroller Connectivity to TCP/IP servers, and sensor design. There is a system that uses Wireless Sensor Networks (WSNs) to improve agriculture efficiency. It works very efficiently. However, it is in a tomato greenhouse in Italy. So, we would like to inform our designs for our sensor network at UCSC from Srbinovska’s WSN in Italy. We can further analyze how a website can transmit and receive information with a remote microcontroller. We also need to survey the possible switching technologies to interface with the power system and microcontroller with reliability and safety. It is possible to consider more efficient and different technologies to meet the client’s needs such as: a phone application, a different microcontroller, and a remote/screen interface on-site.

1c) Bibliography

1. We chose this source because it has a working Wireless Sensor Network already implemented somewhere in the world. We can use this source to our advantage to start building a network here.

Mare Srbinovska, Cvetan Gavrovski, Vladimir Dimcev, Aleksandra Krkoleva, Vesna Borozan, *Environmental parameters monitoring in precision agriculture using wireless sensor networks*, Journal of Cleaner Production, Volume 88, 2015, Pages 297-307, ISSN 0959-6526,

<https://doi.org/10.1016/j.jclepro.2014.04.036>. (<http://www.sciencedirect.com/science/article/pii/S0959652614003916>)

(To access it, use a UCSC VPN if not connecting from CruzNet or eduroam)

2.) Monnit is our competitor in United States. Their products work, but they are expensive. A single coin cell Thermocouple sensor costs 90 USD. An industrial grade thermocouple sensor costs 190 USD. Their customers include large corporations such as Little Caesars, Shutterstock, Nestle, The Boy Scouts of America, and others that can afford this type of technology.

MONNIT,inc. *Wireless Thermocouple Sensor*. Monnit, inc.

https://www.monnit.com/Products/Wireless-Sensors/Coin-Cell/Wireless-Thermocouple-Sensor

3.) The group did not know how to connect a microcontroller to a TCP/IP server before. We do now. Here is where we learned it from, along with some explanation from Isaac.

TCP/IP to Microcontroller connectivity tutorial:

C.L. Stephens, *TCP/IP - An introduction for 8 & 16 bit Microcontroller Engineers* Computer

Solutions Ltd. <http://www.computer-solutions.co.uk/download/tcp_tutorial.pdf>

4.) This is an existing technology that is used for a large-scale greenhouse. They offer specific sensors that are all-in-one. It is important to compare our design to this technology because it is easy-to-use and similar to the SlugSense technology that we are considering to use as part of our project.

<https://www.dataloggerinc.com/applications/greenhouse-monitoring/>

5.) This is a DIY Greenhouse project that uses an Arduino and has the functionality that is required by our client. It is organized and is a good reference our design.

<https://hackaday.com/2012/06/05/large-scale-arduino-controlled-greenhouse-does-some-serious-farming/>

6.) This is a NASA database of solar energy tracking. I got my solar power data from here.

<https://power.larc.nasa.gov/>

Methodology:

<https://power.larc.nasa.gov/documetns/POWER_Data_v9_methodology.pdf>

7.) This is a website to help you calculate the angle for solar panels for optimal efficiency.

<https://www.solarpoweristhefuture.com/how-to-figure-correct-angle-for-solar-panels.shtml>

1d) Stakeholder Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stakeholder | Descriptor | Interest(s) in project | Degree of importance | Potential impact |
| I.D.E.A.S.S.:  Impact Designs: Engineering and Sustainablility through Student Service | UCSC program | Facilitate more projects, | Medium | positive |
| Sue A. Carter | UCSC Physics Professor | Use of solar panels | High | Positive |
| Ronnie Lipschutz | UCSC Rachel Carson College Professor | Needs the hydroponics lab | High | Positive |
| S-Lab:  Carson 152 students + previous students | UCSC Students | Would like to find out the cheapest method to grow food | High | Positive |
| S.U.F.I.S.: Sustainable Urban Food Initiatives | Customer | Would like to find out the cheapest and energy-efficient method to grow food | Moderate | Positive |
| Tela Favaloro | Customer | Wants to get this project done | High | Positive |

1e) Metrics and Criteria for Success

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criteria | Indicator | Type | Units | Baseline | Result/Goal | Measurement Strategy |
| User Friendly Website | Easy to find sensors? | qualitative | Minutes spent per visit | No website | < 5 minutes per visit, > 30 seconds | Measure minutes spent on the website |
| Reliability | Uptime | quantitative | % (Uptime / total time) | No website | 99% uptime | Log downtime minutes |
| Sensors | Number of sensor ports | qualitative | # | 12 | 36 | Count # of microprocessor ports |
| User-friendly manual | Percent of students able to understand the manual | qualitative | %: | 0 | 95% | Survey the students for manual readability |

1f) Idea Generation

Problem: Interfacing with BMS and Charge Controllers

There are 3 devices we must be able to read, and 1 master microcontroller. Unfortunately, the Arduino only has 1 USB hub, and adding an external USB hub requires an extra power source and is slow. The Raspberry Pi Model 3B+ has 4 USB ports we can utilize. The Arduino supports Rs-232 communication, but only supports 1 USB port. We need 3 ports. The Raspberry Pi is capable of 3 ports. Changing the master microcontroller also means a modification of code. I will need to rewrite the code entirely. There’s also the problem of wires running across the greenhouse floor.

My solution is to have multiple microcontrollers installed at the greenhouse. There will be a master microcontroller. Its job will be to 1.) monitor the BMS system and the two Tristar MPPT Solar Chargers, 2.) Communicate with the slave microcontrollers that gather sensor data, 3.) compile data into a JSON format (best for internet-based data exchange) and send it to our server.

There will be two slave microcontrollers. Their job will be to 1.) monitor the voltage levels of each individual solar panel with a 12-channel ADC chip (or multiple 4-channel chips), 2.) read sensor data from student sensors (data protocol can be UART, I2C, 1-Wire, or SPI), 3.) send all that data through Bluetooth to the master microcontroller.

The BMS system comes equipped with temperature sensors for the batteries that are already installed. It also comes equipped with code that actuates a heater and/or cooler when the batteries get too hot or too cold. All that’s needed is to install the heater and the cooler. I do not have a 12V heater and/or cooler, though.

The Tristar MPPT can be monitored using an RS-232 cable. It uses a MODBUS protocol. Python has a library for MODBUS that I can use to monitor the Tristar MPPT.

I will write a detailed manual of how the system is characterized, what parts break the most and how to fix them, contact information, etc.

I have characterized the solar panels already. The expected power output is 5956 watt-hours/day on January 31st, 2019.

The batteries have also been characterized. The 4 batteries we have installed can hold up to 2304 watt-hours of energy altogether, and last for about 2000 life cycle. If we increase it to 16 batteries (4 rows of batteries), we can have up to 9216 watt-hours of storage.

Monitoring the greenhouse would work like this: the master microcontroller would request data from the BMS system and the two Tristar MPPT Solar charge controllers as fast as it can. Every 10 seconds, it sends the data to a database.

Data flow:

Request Data -> Send Data -> Set Timer for 10s -> Request Data -> if Timer hasn’t reached 10s, wait -> Send Data.

The data being harvested would be sent to a MySQL based database system using a 2G wireless FONA system. ALTERNATIVE IDEA: we could have a REALLY long ethernet cable, plug one end into the Raspberry Pi’s ethernet port, and plug the other end in to the nearest Ethernet based internet port, and send it that way. Kevin from the farm next door might have a port we can use for this, or we could ask the arboretum. I will have to investigate if this idea is even feasible.

If Talmor did his job, then the web server should be ready to go already. If not, I will have to work on this myself. I have tried to investigate, but the website is down, and I cannot SSH into the server. Therefore, I cannot know if Talmor has done his job with the web server, and I will simply need to know what database to store data in.

Problem: Interfacing with Sensors

The whole point of this greenhouse is to be able to harvest Sensor data. The problem is, the master microcontroller is at the front of the greenhouse, the greenhouse is 23 feet long, and the sensors need to be able to be placed anywhere. How are we going to fix this problem? If we connect a 23-foot I2C cable from the master microcontroller, there will certainly be errors in transmission due to electromagnetic noise, and it would be impractical to use the sensors in the greenhouse. My solution is to use slave microcontrollers (also Raspberry Pis) that harvest data for me. They will not send data on their own; they will send data to the master microcontroller so that the master can send data to the database. How are we going to communicate between the two microcontollers? I’ve got two designs to remedy this problem: either we use a 25-foot-long cable for intra-USB communication, or we can use Bluetooth.

What type of sensors will we be accepting? We could incorporate the SlugSense sensors, but that would be late-stage work. I must be able to work with standard, off-the-shelf sensors first. If SlugSense likes using the I2C protocol, programming for them would be easier. I have no clue what protocol they use, though. As for off-the-shelf sensors, we will be accepting I2C, 1-wire, UART, and SPI based sensors (the 4 most popular types of sensors). Each one of these protocols will have a manual that comes with it so you know how to plug them into the microcontroller.

Problem: Harvesting Data from Sensors

Every sensor has its own protocol for how to harvest data. How are we going to account for this? I propose that we have a set of standard sensors that you can select from, that already have their own harvesting script that I write during this project. But, what if someone brings in a pressure sensor (i.e. something that I won’t be here to program for)? I propose that if that situation arises, someone will have to write a data-harvesting script in python. This is one of the reasons I’m choosing a Raspberry Pi instead of something like an Arduino; if someone wants to use a pressure sensor, and knows how to program in Python, they can write their own script, have it approved by Tela, and upload it to the slave Raspberry Pi they want to use it on using the script.

Problem: Sensor Usage

Will we leave the sensors on all the time, or will we only turn them on when we want to use them? I propose that we leave them on all the time. What happens if someone wants data from when the sensor was turned off? Plus, how would we actuate (turn on and off) sensors remotely? This would be a challenge in and of itself.

Feature: Incorporating SlugSense Sensors

Rather than working on the sensor diagnostics and web development from scratch, we can utilize SlugSense product developed by UCSC affiliates. We would build our system around this product. This would provide an app for user-friendly real-time data. We can modify the code to display data-overtime to meet the client’s needs. We would still have the issue of interfacing with the battery management system and charge controllers, but this product would allow us to focus more heavily. We would still need to build a website, but the database is already built and we can extract data from it to create the website.

1e) Design Specifications and Trade-offs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Design** | **Ease of Use** | **Cost** | **Modular** | **Easy to Integrate** | **Accessibility** | **Total** |
| Weighting | 3x | 1x | 2x | 1x | 2x |  |
| Using only an Arduino for everything | --- | + | -- | --- | 0 | ------ |
| Using only a Raspberry Pi for everything | + | + | ++ | +++ | + |  |
| Using a Raspberry Pi and Slaves connected via RS-232 cable | ++ | --- | + | ---- | ++ |  |
| Using a Raspberry Pi and Slaves connected via Bluetooth | ++ | - | + | 0 | ++ |  |
| Use Arduino and Shield | - | + | -- | ++ | ++ |  |
| Use Arduino and Shield redesign | + | + | - | ++ | ++ |  |
| Use STM32F4 with New Shield | ++ | - | ++ | --- | ++ |  |

**Part 2: Project Proposal**

2a) Motivation and Final Problem Statement

The S-Lab at the University of California, Santa Cruz has two green houses that are used for facilitating projects for student innovation. The green houses use sustainable energy to power devices needed for their research. A group of engineering students designed and implemented a power system that regulate power storage and supply power to the greenhouse. They also began but did not complete the sensor interface for data analytics of the power system and greenhouse environmental metrics. After that, another team picked up from where they left off to work on the database. I will pick-up where the last team left-off to build and improve on their design.

As of now, the S-Lab’s Green House has a reliable and safe power source, but further improvements are needed to complete the desired functionalities by the S-Lab for advanced research. The S-Lab needs the core project to be completed. I will deliver the following: A user-friendly connection port for sensors, integration of the battery management system with the microcontroller for data harvesting, integration of the charge controllers with the microcontroller for data harvesting, and a website with modular data display.

2b) Scope of Work

The primary scope of my work will be microcontroller configuration and website design. Since the power system is fully operational to our knowledge, we will not be doing any work on the solar panels or on the batteries themselves. Instead, we will collect information about those items, analyze the data, and put the analysis of that data on the website. We also will add the functionality of adding actuating capabilities to the system to have a well-rounded system that can be monitored and controlled.

Primary Deliverables:

1. Ease of use for connecting sensors to the slave microcontroller. This means:

* I will write a manual for adding sensors to the system or redesign the shield to be user-friendly with few steps to connect.

2. Data analytics for the power system. Includes:

* Voltage of each individual battery using a Battery Management System (BMS),
* Voltage of each individual solar panel using a 12-channel ADC (gathered from the slave microcontrollers)
* Current drawn from solar panels (gathered from Tristar MPPT Solar Charge Controller)
* Current drawn from each individual battery
* Total instantaneous current being drawn
* Temperature of batteries. The BMS has a feature which actuates a cooling fan or a heating element (sold separately, we need to install these) if the batteries get too cold or too hot.

3. A website that:

* Real-time data and data over-time
* Successfully able to add and remove sensors with the guidance of my manual.

4. Thorough Documentation as a User Manual

Stretch Goals:

1. Ability to actuate devices from the website

2. Data from both the BMS and Tristar MPPT Solar Charge Controllers

3. User-friendly interface with Slave and Master Microcontrollers

2c) Methodology: Proposed Technical Solution

My solution is to have 3 microcontrollers do the work for me.

2d) Methodology: Timeline and major milestones

2e) Impact Assessment Strategy

The type of data needed to assess the strategy:

How we will obtain the data:

Over what time period will we collect our data?

Identify any negative outcomes:

2f) Parts List and Budget

**Part 3: Appendices**

3a) Resource Requirements

3b) Supporting Documentation

Source code for FONA connectivity to Arduino UNO:

<https://github.com/adafruit/Adafruit_FONA>

3c) Business Model Canvas