Solar Lighting System

Lyric Haylow

Jeb Malek

Josh George

**System Description and**

**Final Report**

Revision – 0

30 May 2024

# Table of Contents

[Table of Contents 2](#_Toc1790122669)

[Table of Figures 3](#_Toc920046057)

[Table of Tables 3](#_Toc844052160)

[1. Overview 5](#_Toc1460820374)

[2. Development Plan and Execution 5](#_Toc332733279)

[2.1 Design Plan 5](#_Toc986329560)

[2.2 Execution 6](#_Toc1973874375)

[2.3 Validation Plan 6](#_Toc1126804806)

[3. Critical System Report 7](#_Toc782813166)

[3.1 Charge Controller Report 14](#_Toc875452136)

[3.2 Power System Report 16](#_Toc2025071528)

[3.3 Microcontroller Report 18](#_Toc1824615603)

[3.4 Mobile Application Report **Error! Bookmark not defined.**](#_Toc1187308175)

[4. Conclusion 18](#_Toc1520085087)

[4.1 Key Decisions 19](#_Toc1047493997)

[4.2 Learnings 19](#_Toc501707545)

## **Table of Figures**

Figure 1. System Design Plan Diagram

Figure 2: The validation plan for the solar lighting system.

Figure 3: Mobile Application Home Page

Figure 4: Mobile Application Battery Levels

Figure 5: Mobile Application Lighting Preferences

Figure 6: Mobile Application Bluetooth Device List

Figure 7: Mobile Application 10-Hour History

Figure 8: Mobile Application User Settings

Figure 9: Mobile Application About Page (Dark Theme)

Figure 10 BQ24650 Typical Application schematic

Figure 11: BQ24650EVM schematic

Figure 12. Converter PCB

Figure 13. Inverter

Figure 14. System Structure

Figure 15. Microcontroller Connection\

Figure 16. Final Demostration

## **Table of Tables**

Table 1: Mobile application validation plan.

Table 2: Solar charge controller validation plan.

Table 3: Power inverter validation plan.

Table 4: Microcontroller validation plan

## **1. Overview**

The Solar Lighting System is a modular household-based application that uses renewable energy sources to cut down on utility expenses. The system uses a solar panel to power lights in both the patio and foyer and work autonomously with motion sensors controlling their activation. The system consists of several different subsystems: Charge Controller, Power System control, Microcontroller, and a Mobile Application. Proper integration of all subsystems was necessary to ensure a working product, as well as pass all deliverable requirements.

## **2. Development Plan and Execution**

### **2.1 Design Plan**

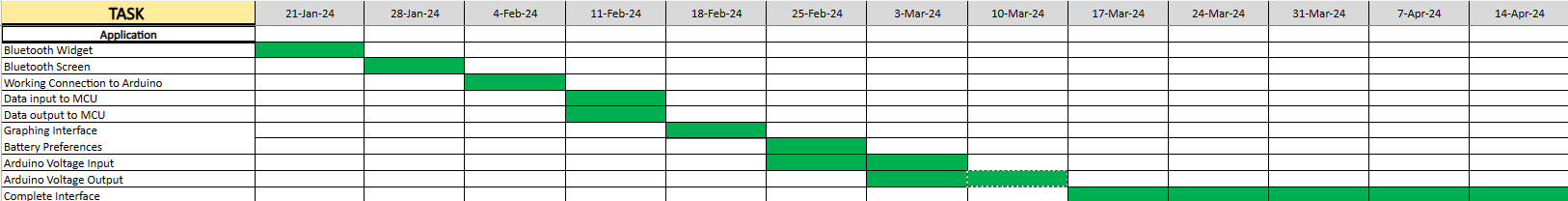
The design for the Solar Lighting System begins with the solar panel. A large panel absorbs sunlight and converts it to energy, powering a large battery through a solar charge controller. The 12V battery is charged up and connected to a power inverter, which inverts direct current into the alternating current necessary for the lights to be operated. Two LED lights are implemented into the system, one to represent an indoor foyer light and the other to represent an outdoor patio light. A microcontroller is also powered by the battery and connected to the sensors, enabling them, and disabling them on command. The battery is also connected to the microcontroller through a stepdown circuit, lowering the voltage to a value readable by the MCU. Finally, the microcontroller connects to a mobile application, allowing a user to turn lights on and off at will, switch the operational mode of the system from “Manual” to “Motion”, and read the battery percentage of the system.

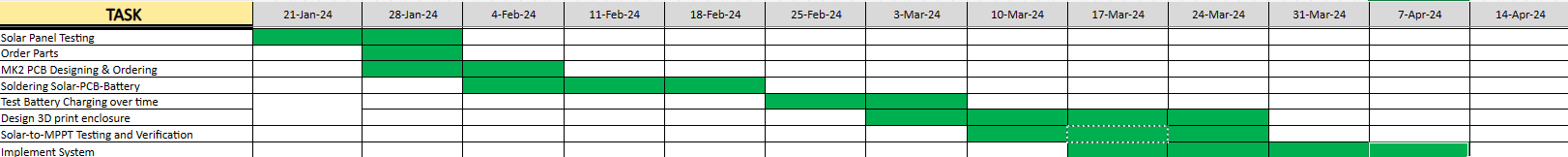
A diagram of electronic components

Description automatically generated

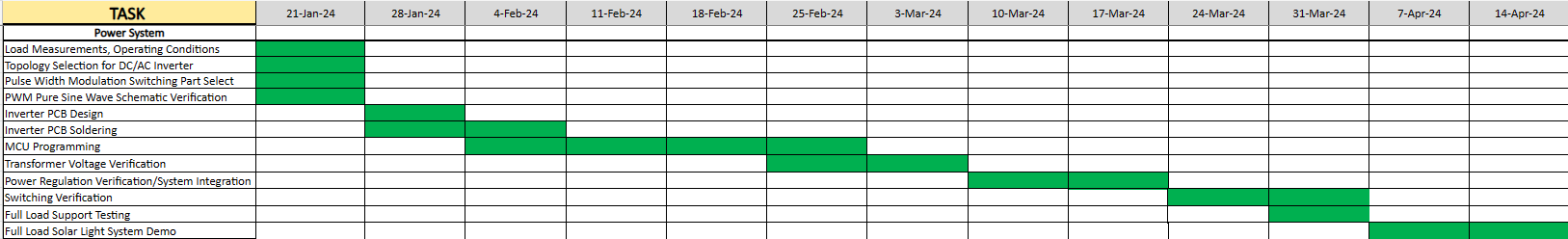
Figure 1. System Plan Diagram

### **2.2 Execution**

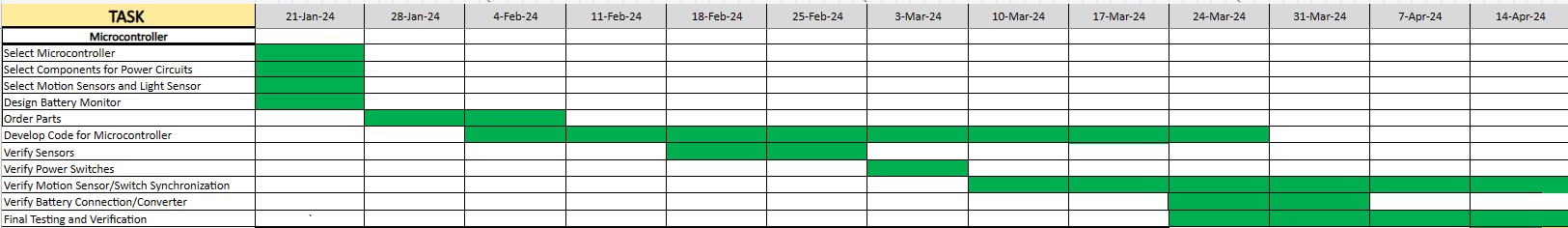
*Table 1: Mobile application validation plan.*



*Table 2: Solar charge controller validation plan.*



*Table 3: Power inverter validation plan.*

*Table 4: Microcontroller validation plan.*

### **2.3 Validation Plan**

A screenshot of a computer

Description automatically generated

*Figure 2: The validation plan for the solar lighting system.*

## **3. Critical System Report**

### **3.1 Mobile Application Report**

The mobile application for the Solar Lighting System works as intended with five unique screens in addition to the Home Screen: About, Battery Level, Lighting Preferences, 10-Hour History, and User Settings. The following screenshots are taken directly from the app, which was loaded on a Motorola Moto G Stylus.

A screenshot of a cell phone

Description automatically generated

*Figure 3: The Home screen, displaying four different buttons and a settings option.*

The home screen was redesigned to better portray the system to a customer as well as give the UI a cleaner feel. Clicking on “About” will successfully display a page which provides the user with a paragraph explaining the purpose of the app. Clicking on “Battery Level” will lead to the following screen.:

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

*Figure 4: The battery level screen when the system is low in charge (left) and the same screen when the system charge capacity is nearly full (right).*

The app is able to reliably indicate the battery percentage and updates live every time this screen is loaded. Both screenshots were taken on the same sunny day, in which the system was able to charge up from 40% to 90% within 2 hours.

From the home page, clicking on Lighting Preferences will lead to the following page:

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

*Figure 5: The Lighting Preferences screen before and after connection.*

The buttons on this screen will only activate once Bluetooth connection has been established with the Arduino Rev3 Uno board through the HC-05 Bluetooth module. To connect, first pair the discoverable module with the phone through the phone settings. Next, open the app and navigate to this screen. Clicking on “Connect” in the top right corner will open the following screen:

A black rectangle with white dots

Description automatically generated

*Figure 6: The list of paired Bluetooth devices in the area. HC-05 module is shown as discoverable.*

Click on the shown Bluetooth module to connect to the system. Once the connection has been established, the user will be able to interact with the system in two operational modes, “Manual” and “Motion”. Using the switch can toggle between the systems. In “Manual” mode, the motion detectors are off and the lights can be manually turned on or off using the displayed buttons. Setting the switch to “Motion” disables the two buttons and activates the motion sensors through the microcontroller.

Clicking on 10-Hour History from the home page will navigate the user to the following page:

A screenshot of a cell phone

Description automatically generated

*Figure 7: The 10-Hour History fragment of the application.*

A graph is displayed on this fragment, along with some text explaining this screen. The graph displays battery percentage data of the system throughout the day, more specifically from the past 10 hours of the system operating. The system successfully records the battery percentage on the Y-axis and displays them with their respective hour marking on the X-axis. The percentage is read by the microcontroller, sent to the app, and then sent from the app to a database hosted by Firebase. The data in the database is then read by the app and graphed in a comprehensive manner. The above graph was taken from a moderately sunny day which got darker in the evening.

Clicking on the “User Settings” icon navigates the user to the final page of the app:

A screenshot of a computer

Description automatically generated

*Figure 8: The User Settings segment of the application.*

The settings for the app include a Time Zone setting, a Language setting, and a dark theme. The time zone is implemented in the graph, which provides different hour markings depending on which time zone was selected. The language dropdown box allows the user to choose between English, Spanish, French, and Japanese as viable options to operate the app within. Lastly, the dark theme offers a different UI scheme for those who desire it. It can be seen below, demonstrated with the “About” screen:

A screen shot of a cell phone

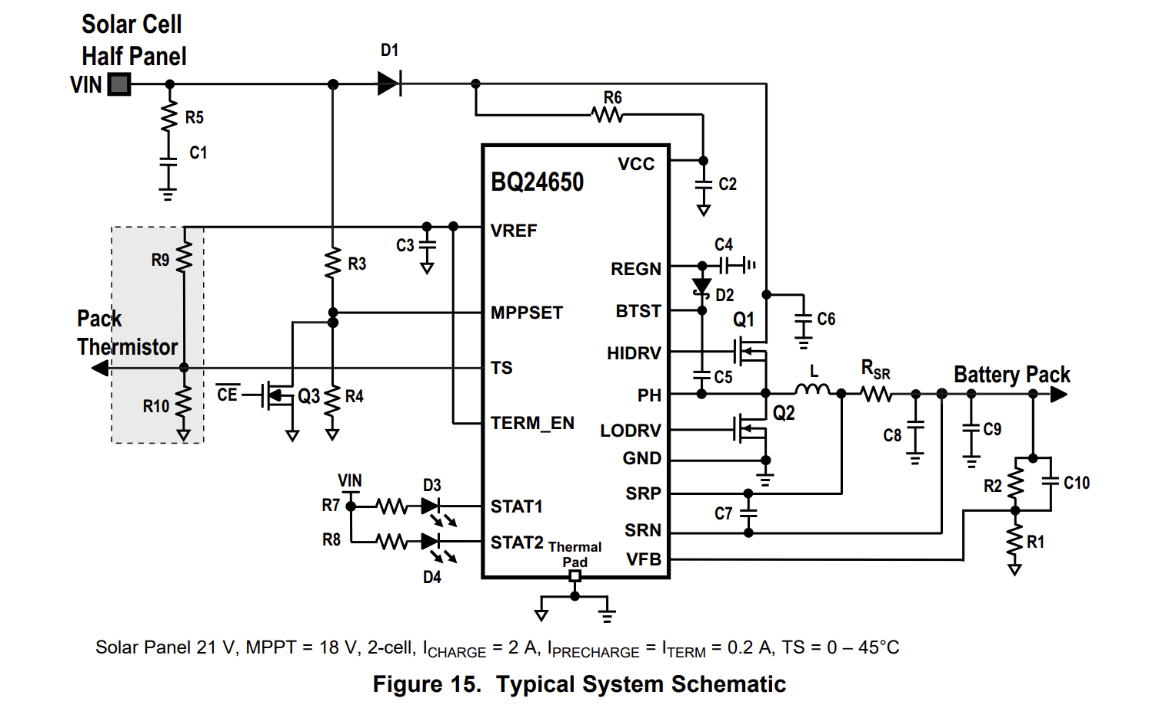
Description automatically generated

*Figure 9: About segment of the application, displayed when “Dark Theme” is switched on.*

In conclusion, the application successfully provides data to the system in the form of setting the operational and successfully processes data from the system in the form of the battery percentage and a graphical user interface. It accurately reads the battery percentage of the implemented 12V battery and is able to command the system in real time with the use of Bluetooth and the Arduino R3 Uno microcontroller. The app succeeded in its primary functionality; however, future adjustments to this subsystem would likely include more options for user customization. It would also be worthwhile to implement functionality for multiple Solar Lighting Systems, so that a user could control and obtain data from more than one system at once.

### **3.2 Charge Controller Report**

The Solar Charge Controller, which was to regulate power coming from the solar panel to the battery, was chosen to be the BQ24650 design from TI. During 403 and 404 efforts were made to get the design to work, and the design shown in the datasheet is shown below.

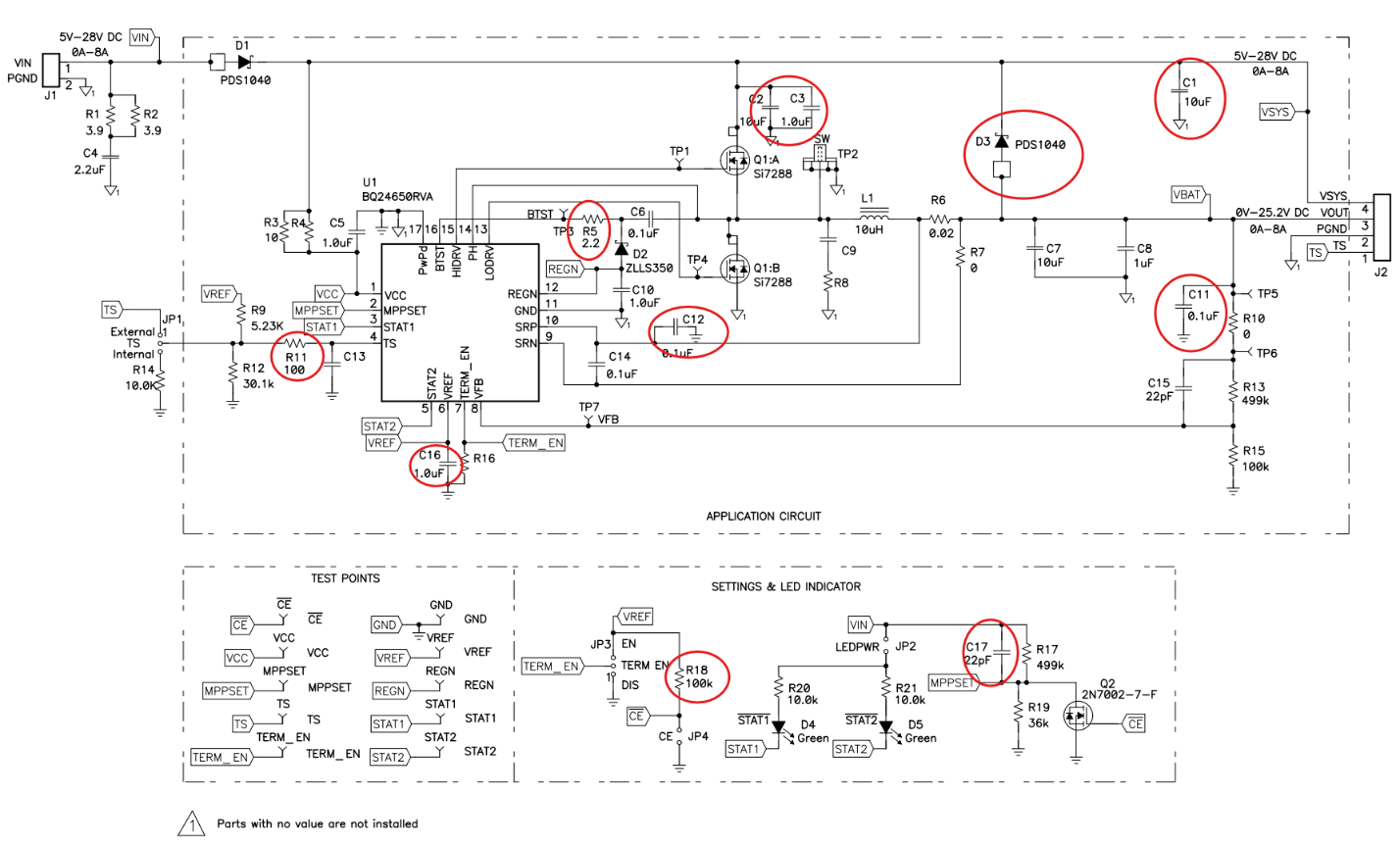


*Figure 10: BQ24650 Typical Application schematic*

Two separate boards were printed during this time, and several attempts to get them to draw current were inconclusive. After many attempts, much testing, and looking into their datasheet, the conclusion I’ve come to is that their typical application design is faulty. Listed below are verifications of what the second board was capable of, and worked, as well as tests done to try and correct functionality. All mentions to parts will be in reference to Figure 2.

* Connecting board to 21V 1A input from DC Power Supply resulted in Charging light turning on, and voltage carries across the board appropriately. Output voltage and Battery Pack high pin read at ~12V, so output also set correctly. When connected to 12V 18Ah Lead-Acid battery, charging light stays on, however charging current limited to around 4mA. The voltage at the VFB pin read around 2.1V, which is value expected for the board to be set to output.
* Connected fully charged battery to output with no input attached, and LED D4 turned on, showing it recognizes fully charged batteries. Confirmed for both Lead-Acid and Lithium-Ion batteries.
* After hooking up DC Power, attached the faulty battery, and all LEDs turned off, showing the board accurately reads faulty attachments.
* Attempted changing R2 and R1 voltage divider ratio, both for raising and lowering voltage division value. Raising it caused no real change, as still recognized battery needed charge but didn’t pull large current, and lowering it caused board to recognize battery as prematurely fully charged.
* Attempted to change R3 and R4 voltage divider ratio for proper reading from solar panel. No real change, but ultimately changed both values to match solar panel voltage input we chose.
* Resoldered both IC and Q1-Q2. No discernable change in voltage values.
* Attempted to remove Rth, thermistor. The board would not work without it, which is incorrect with what datasheet provides.
* Attempted using solely 12V 10Ah Lithium-Ion battery, rather than planned Lead-Acid batteries, since charger is built more for Lithium-Ion batteries. There was no discernable difference in voltage between the two, and the main functionalities worked the same. Returned to Lead-Acid battery since it had larger capacity.
* Soldered a second board completely, just in case first soldering had some issue with it. Same functionality as first board.

After much testing, I turned to the evaluation sheet for the BQ24650 and concluded that the typical application design in their main datasheet is faulty. Shown below is the evaluation board schematic, and all components circled in red are not represented in the typical design schematic that TI proposes works. There are additional components listed, but many of them are for the purposes of testing.

*Figure 11: BQ24650EVM schematic*

With these obvious differences between boards, as well as another associate who is using the same board and coming across the same issue, I strongly believe that the BQ24650 typical application design is faulty, or at the very least missing several components. Because of this, we opted for using the charge controller that came with the solar panel we had bought. Using this charge controller, I was able to meet integration requirements for my subsystem.

The other addition I had for the project was creating a quick connection between the battery and the Arduino, for the purpose of reading voltage from the battery and sending it to the app for reading the battery percentage. While estimating battery percentage using voltage isn’t as accurate, we were going for functionality first, and if we then had time we planned to add a current sensor. In the end we opted for reading through voltage. This was accomplished by a high resistor value voltage divider, which stepped the voltage value down from 12V to around 2.0V, using a ratio of . For testing, we applied the circuit and read the 12.6V from the battery as 2.23V on the Arduino programming. The slight change in value I’ve determined was likely due to a slight difference in resistance than expected.

### **3.2 Power System Report**

The power system was able to provide stable voltage for required loads for demonstration purposes. The nominal 5 V Output was regulated and supplied Arduino, switch, motion sensor, and Bluetooth Module sufficient voltage and current supply to maintain operation.

A green circuit board with small black round object

Description automatically generated

Figure 12. Converter PCB

The inverter subsystem required a factory-made solution for AC load support. After strenuous testing on the Microchip device which was unable to be targeted the most likely explanation is that through a design flaw with a grounding issue in the transformer. Converters needed for the microcontroller were not able to supply source voltage therefore the MCU was not powered up. After consideration to focus more on the converter subsystem the AC load was supported with the purchase of the AIMS Power DC To AC Power Inverter rated for 800W well over our intended demonstration load.

A close-up of a power inverter

Description automatically generated

*Figure 13. Inverter*

The final system structure is shown below.

*A machine with wires and wires

Description automatically generated with medium confidence*

*Figure 14. System Power Structure*

### **3.3 Microcontroller Report**

Using an Arduino Uno Rev 3 the motion sensor data, switch actuation, and mode of operation control were implemented in this subsystem of the project.

A close-up of a machine

Description automatically generated

*Figure 15. Microcontroller Connection*

## **4. Conclusion**

The Solar Lighting System demo we made functioned as needed, however numerous improvements could be made for better overall quality. Regardless, all subsystems worked as needed for integration purposes and led to a fully functional project.

*A solar panel on a pole

Description automatically generated*

*Figure 16. Final Demonstration*

### **4.1 Key Decisions**

All decisions regarding what pieces were to be used for individual subsystems were made by those in charge of that part of the project, however frequent communication between each other, as well as our sponsor for this project, led to a few aspects of our original project being changed.

One of the initial key decisions made was how to implement an Arduino microcontroller. With the loss of a teammate early on, we gained permission to use a bought microcontroller, rather than solder one ourselves. The software portion was delegated to both Jeb and Josh, seeing as they are most closely affected by the microcontroller and are most aware of how these subsystems integrated.

The other major decision we made was to remove the flood light from our system design. Ultimately, our sponsor decided that the power requirements of a flood light would likely cut down on the product's lifetime by an amount too detrimental. Our expected lifetime of the lighting system would be around 9 hours; however, a flood light would have cut that value down to around 2 hours.

### **4.2 Learnings**

Android application design. Arduino microcontroller coding. Printed power circuit board design, manufacturing, and assembly, and physical power electronic construction were key takeaways from this senior engineering design project. Problem solving and design choices lead to critical decision making and schedule forecasting considering costs. The final product demonstrated provided a functional representation of what our team's task required.