Under The Sun Drink Mixer

Laura Cano, Moises Dominguez, Mike Tyrlik, Stephen Zimmerman

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — An Original Drink Mixer controlled by a smart phone and powered by the sun. Living in the sunshine state is a huge motivation in wanting to build a drink mixer; Under the Sun Drink Mixer will be ideal for BBQs and tailgating to increase hydration. In the midst of summer, people want to be hydrated at all times and what better way than a solar powered drink mixer.**

***Index Terms* — Circuits, databases, microcontrollers, microprocessors, power MOSFET, solar energy**

I. Introduction

The idea of building a drink mixer is influenced by the American culture and lifestyle of having convenient things at hand and it is also influenced by the Floridian weather which makes it more appealing to have an automated drink mixer. However, it is known how hot Florida Summers are and with tailgating season this fall what a perfect way to enjoy the time than by having an automated solar powered drink mixer!

Another incentive to build a drink mixer is to avoid long queues when everyone is trying to serve a drink at the same time. By ordering through the phone, the client application will provide the user with a unique code. With this code, the user can obtain their customized drink with Under the Sun Drink Mixer.

Some of the goals during this project are to create a social environment among its users by minimizing the amount of time spent at serving drinks and increasing social interaction. If less time is spent at serving, one could interact more with the guests, family, and friends. Also, with tailgating season approaching, people usually put all beverages in a shared cooler and lose track of their drinks while socializing. Having all the beverages in the drink mixer apparatus, no one has to worry about other people grabbing their drinks.

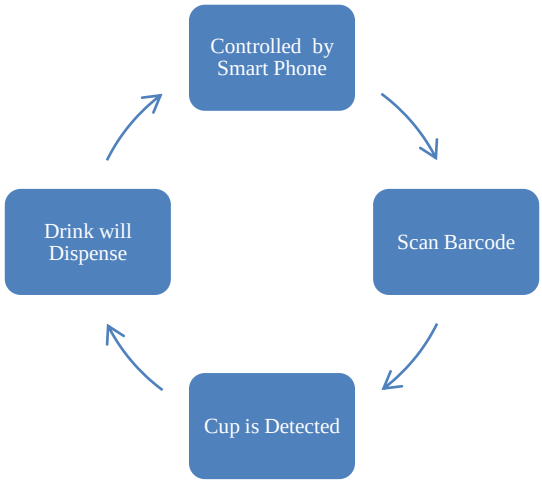

Fig. 1. Overview

II. System Platforms

A. *BeagleBone Black*

1. *Overview*

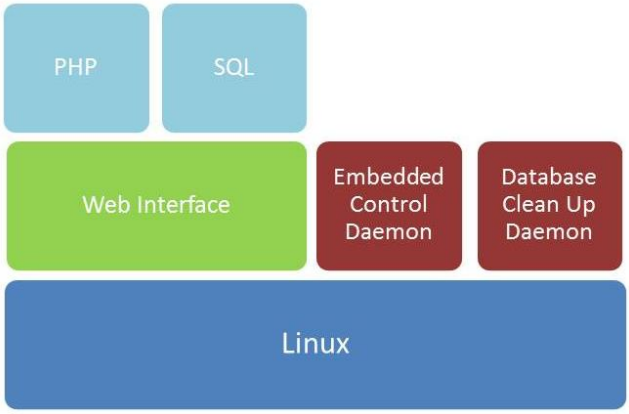

Fig. 2. Software Architecture

In the beginning, a custom PCB was designed that utilized the TI AM335x ARM processor and was to run for all of the necessary software. This processor provided enough power to meet the requirements of Linux along with all of the I/O pins that would be necessary to read from sensors and to control dispensing devices. This design, although efficient, was quite expensive to produce in orders of one so the initial design was quickly modified under the guidance and acceptance from Dr. Richie. The resulting design contained several aspects of the original design, with the exception that the use of the significantly more affordable development board, the BeagleBone Black, would be incorporated into the design. The Under the Sun Drink Mixer uses a BeagleBone Black

development board for various uses throughout the project. It utilizes a Texas Instruments AM3359 1.0 GHz ARM processor running Arch Linux 3.8.12. The Linux operating system is loaded from the 2GB of onboard flash. The primary role of the BeagleBone is to facilitate the software required to keep track of and manipulate the various drink orders from multiple users.

2. Web Interface

The web interface utilizes the common elements of Linux, Apache, MySQL and PHP (LAMP). These software components were chosen for their widespread support, open source software model, and for their ability to simplify several design aspects of our design and allow us to focus on higher level functionalities. The primary client iOS application communicates to the web interface for several components that include: ordering drinks, canceling drinks, checking container levels, checking drink orders, and refilling containers.

Ordering Drinks

When a drink is ordered from the iOS application, the iOS application sends the amount of ingredients in ounces to the "Order Drink" web page using a POST request. The server verifies that the variables that were received are valid and all appropriate. If the correct variables were sent and they are of the proper type and within acceptable values, the server will retrieve several settings from the SQL database. The first attribute that will be tested is the amount of available ingredients. If there are sufficient ingredients for the requested drink, the drink will be reserved for the user. The reservation process entails several steps. The first step that the server must undertake is to create a unique order ID for the order. The order ID employs the use of a Globally Unique Identifier (GUID). The use of a GUID was chosen because of its ability to greatly minimize the possibility of both duplicate order IDs as well as the ability to correctly guess an order ID. These two attributers are important aspects to consider when trying to maintain a robust and collision free design. Upon successful creation of the GUID, the server will generate a timestamp to track order time. The server will proceed to store all of the needed parts of information before sending a reply to the client iOS application. When all items are completed error free, the server will then return the order ID and the order's expiration time back to the iOS application encoded using JSON.

Cancelling drinks

Since only one drink can be ordered at a time, the user is allowed to cancel a drink if they change their mind and would like to opt for a different choice of drink. This is done using a POST request to the cancel drink page with the order ID of the drink that is to be canceled. When the server receives this request, it will first ensure that it was given valid information. If the barcode exists and is of the correct type, the server will then attempt to retrieve the order with the specified order ID. If the order has not yet expired or has not yet been picked up, this step will succeed. Upon the retrieval of the order information, the server will update the time in which the drink was ordered by the maximum amount of time the user has to pick up the drink. For instance, it the drink was ordered at 15:45 and the user cancels the drink before it expires at 15:55, the server will change the time of day the drink was ordered to 15:35. Here the server is only responsible for making the drink order "expire" by changing the order time from which it was ordered. After this occurs, the drink will be considered expired and the database clean up daemon (discussed later below) will handle the re-allocation process of ordered ingredients.

Checking Canister Levels

The iOS application can query the database to determine the amount of unreserved ingredients and also the actual amount of fluid remaining in the containers. When it comes to ingredient status, the only amount that is stored in the MySQL database is the unreserved ingredient levels. That is the amount of an ingredient that is not allocated toward making ordered drinks. Because only non-reserved status is maintained, to find out how much fluids are actually left in the bottles simple arithmetic needs to be performed. By summing the reserved ingredient amounts of the orders that have not yet expired and the orders that have not yet been picked up with the corresponding unreserved amounts stored in the database we can calculate the actual bottle fill levels.

Checking Drink Orders

In order for the client application to maintain consistency with the drink machine, there needs to be a facility to verify drink order validity. In order to verify a drink order, all that needs to be done is to send order ID to the webserver. The webserver will then query the SQL database and determine the validity of the order. If the order is still valid, the server will return the order ID that was received along with the expiration time back to the iOS application. The expiration time is calculated using the order time and the maximum reservation time. For instance, if the drink was ordered at 17:06, the expiration

time would be 17:16. If the order ID is no longer valid, the server will return failure messages and the iOS application will make the proper adjustments.

Refilling Canisters

Since the amount of ingredients that has been dispensed is tracked, there needs to be a way of updating the amount that is contained in the bottles. The iOS application allows for the user to notify the server that any of the ingredient bottles have been updated or refilled with ingredients. This is an important aspect to address because as shown in figure 3 there can be a large amount of the bottle empty that could be refilled to allow for more drink orders. For instance, if a bottle can hold 64oz. and there is currently 20oz left in the bottle and 6 of those 20oz are unreserved then that leaves 14oz that is reserved. In this instance, re-filling can be done in order to prevent depletion of ingredient amounts and to allow users to continue to order. After the container is refilled with ingredients and the application is notified of this change, the 64oz. container now has 50oz. unreserved and 14oz. of reserved ingredients. Both reserved and un reserved levels can be viewed in the iOS application as discussed later below.

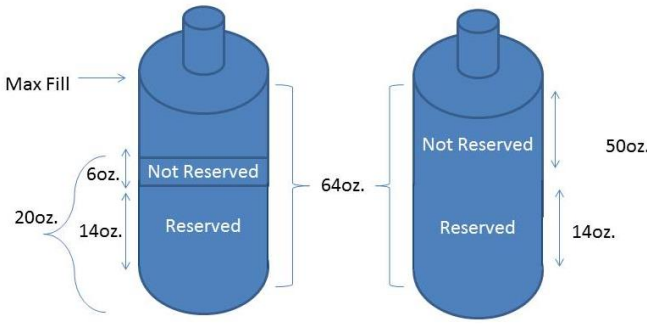

Fig. 3. Illustrated Example

3. Embedded Control Daemon

The Embedded Control Daemon is the control center of the dispensing operation. Where the Web Interface functionality of ordering a drink ends is where the Embedded Control Daemon begins to pick up the dispensing trail. The Embedded Control Daemon first starts by reading input from the barcode scanner. It will block the main thread until some input is received. Once the first character is read the ECD will begin using a timeout period between characters. If an entire order ID length of characters is received, it will store the order ID and move on. If there is too much latency between any of the characters after the first, any buffered input will be discarded and the ECD will begin blocking again. Once a valid order ID is received the orders ingredients are retrieved and are prepared to be dispensed. If the order has not yet expired and has not yet been picked up, then the order is considered valid. At this time the Embedded Control Daemon will then send dispense commands to the Control Board. If the Control Board successfully dispenses the user's drink it will signal back to the Embedded Control Daemon stating such a condition. At that point the Embedded Control Daemon will mark that drink order as being picked up.

If the Control Board responds back as failing to dispense correctly or if a timeout period is reached before hearing back from the Control Board, the Embedded Control Daemon will leave the order untouched. The embedded Control Daemon will repeat this process of blocking and dispensing continuously. If a second user scans a barcode before the first user's interaction with the machine has completed or when waiting for the Control Board to respond, the Embedded Control Daemon will completely ignore it by flushing its USB input buffer before reading more barcodes. This decision was made with impression that if someone is using the machine and it is busy interacting with someone else it will ignore the other background "noise".

In order to communicate with the Control Board the Embedded Control Daemon has a special protocol that is used to communicate over the serial communication lines between the Control Board and the Embedded Control Daemon. The various control codes and acknowledgements are shown below in table 1.

Table 1: Dispense Codes

Command	Meaning
D	Dispense with the following ingredients
F	Finished
Y	Yes/Ok/Continue
N	No/Stop
T	End Line
Z	Finished dispensing the current drink

For example, by using the dispense codes shown in Table 1, 3000oz. of all six ingredients can be dispensed by using the sample dispense code below. All the dispense codes must begin with a "D" in which the Control Board will respond with a "Y". For any ingredient amount that is to be sent to the Control Board must be terminated with the escape character "T". This is required to inform the Control Board when to expect the end of a line. After every command received, the Control Board is responsible for sending a continue command "Y". With the dispense

code below, the Embedded Control Daemon will receive eight "Y"s from the Control Board if all messages are received correctly. Upon successfully dispensing the drink ingredients, the Control Board will inform the Embedded Control Daemon with the control code "Z".

Sample Dispense Code

```
D3000T3000T3000T3000T3000T3000TF
```

Examples:

```
BBB: Tries to send a drink to dispense
CB: Available to dispnse
Result: CB succesfully dispenses

BB == BeagleBone Black
CB == Control Board (Atmega)

    Direction  Value Sent
    BB -> CB    D
    CB -> BB    Y
    BB -> CB    <ing0>T
    CB -> BB    Y
    BB -> CB    <ing1>T
    CB -> BB    Y
    .
    .
    .
    BB -> CB    <ingN>T
    CB -> BB    Y
    BB -> CB    F
    CB -> BB    Y
    [Dispense Drink Here]
    CB -> BB    Z
    BB -> CB    Y
```

Fig. 4. Sample Dispense Code

4. Database Clean Up Daemon

The Database Clean Up Daemon serves two distinct and important fundamental purposes. The first purpose of the Database Clean up Daemon is to look for drink orders that have not yet expired. This is important because the data in an SQL database is passive and will not react on its own. In the event that a drink order is found that has expired, the Database Cleanup Daemon will mark the order as expired, however the more important and second purpose of the Database Cleanup Daemon is to un-reserve the order's ingredients back to the bottle for other drink orders to utilize. Without this, the bottles ingredients would be "lost" because the ingredients could be reserved but fi never picked up by the user they would never be un-allocated.

5. Barcode Scanner

The barcode scanner that was implemented into the design was the Motorola DS9208-SR4NNU21Z Desktop Barcode Scanner. This barcode scanner offers everything that the Under the Sun Drink Mixer needs; it is a hands-free device, it can read off of mobile devices, it can scan in direct sunlight and in darkness, and it offers USB connectivity. The Barcode scanner utilizes the Human Interface Device (HID) protocol for communication, simplifying the interactions required between the scanner and the Embedded Control Daemon.

6. Database

The data management system that was chosen to be used was MySQL. The use of a relational database to store orders and ingredient levels drastically simplified the complications of storing and retrieving user data and also allows for future growth with minimal re-working. The use of a relational database also allows for the ability to easily store a large amount of records which allows the ability to add history, user statistics and other management tasks that were not able to be implemented in the current revision of this dispenser. The database schema that was employed consisted of the variables that were needed to store drink orders. The database schema used can be seen below in Figure 5.

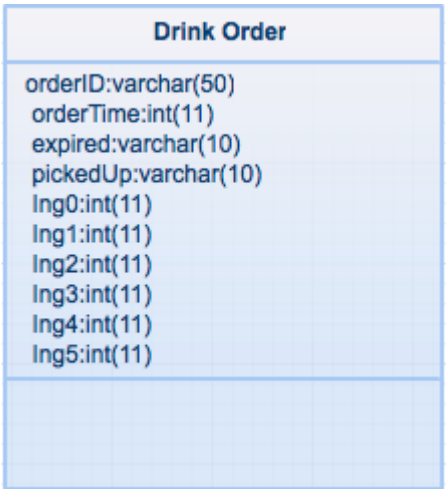

Fig. 5. Database Schema

B. Control Board

1. Overview

The Control Board is a component that is responsible for all of the manual dispensing tasks. It consists of an ATmega328p microcontroller that interacts and communicates with the Embedded Control Daemon that runs on the BeagleBone Black. This control board runs the commands sent to it using the communication protocol discussed above in Table 1 and Figure 4. The Control Board is responsible for translating the volumetric measurements of the drink order into time based values that will be used. The Control Board uses a series of interrupt based timers to dispense the various ingredients. Throughout the dispense process if the user is taking too long to complete any given task, the dispensing is cancelled. If the drink order had any part of it dispensed, the drink will be marked as dispensed. If the user did not get past the second step of placing the cup into the dispense notch, the dispensing will be canceled but they will be able to try again later as long as the order did not expire within that time.

2. LCD Screen

During the dispense process the user is given feedback and some instructions as to what they are required to do next. The LCD Display is the only feedback that is provided to the user of the status of during the dispensing process. The screen encompasses updates to the user of the four different stages of the dispensing process. These stages can also be viewed as a state diagram for the Control Board and are:

1) Scan order barcode
2) Place cup under dispenser
3) Wait for dispensing to complete
4) Customer appreciation

The LCD screen that is employed is a 1.8 inch TFT with true TFT color up to 18-bits per pixel, and a resolution of 160 X 128. This LCD is powered with an input voltage of 5V and uses the Sitronix ST7735R as the display interface driver. The LCD uses the Serial Peripheral Interface Bus (SPI) as a communication interface, consuming six I/O pins to the microcontroller.

Fig. 6. LCD Dispensing Stages

3. Dispensers

The entire vending unit includes six 12 volt electronic solenoid valves which dispense the ingredients, such as liquors or mixers. The six electronic valves control how much liquid comes out of the syphon tube and into the customer's cup. All the six bottles have around two liters each (64 ounces). The CO_2 tank holds enough volume to fully displace all the contents in the ingredients several times allowing for multiple ingredient bottle refills per CO_2 tank fill. An air regulator is used to reduce the approximately 1800 psi of the tank to approximately 5 psi that is fed into the bottles for dispensing. PVC tubing connects the various components of the dispensing system. The electronic solenoids are controlled by microprocessor on the Control Board. The I/O pins on the Control Board are isolated from the solenoids and higher voltage electronics by the use of opto-isolators. Figure 4 below shows the solenoid component from the overall circuit. It takes approximately .5 Amps at 12V to open the solenoid during dispense time. Due to this high current draw, the Control Board dispenses the ingredients serially one after another.

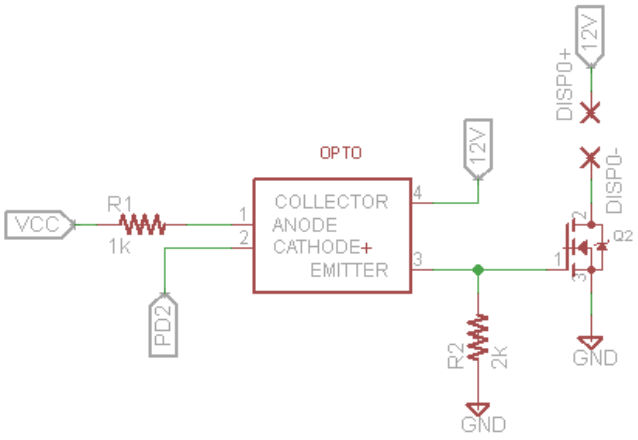

Fig. 7. Solenoid Component Schematic

4. Cup Sensor

In order to only dispense ingredients into the user's container, a short range infrared sensor is utilized to detect the presence of a cup. An infrared sensor is ideal since it has a fairly accurate distance measuring capability at the somewhat short ranges needed. The infrared sensor also is able to measure distances in both daylight and darker environments. Given that the space provided for the cup will be about 8 inches and the sensor should not detect the other side of the cup slot in the absence of the cup, the analog to digital converter on the microcontroller is biased to increase the measuring resolution within this range. The sensor that is his sensor has an analog output that varies from 3.1V at 3cm (1.18in) to 0.3V at 30cm (3.94in).

C. iOS APP

The primary software interface that the user encounters is the application written for iOS on Apple's iPhone. This application serves the purpose of allowing users to browse a menu of available drinks to create, order a drink from the menu, and generate a barcode to pick up the drink, cancel a drink order, and perform some basic maintenance tasks. The application is designed to be a fully functioning application that is simple to learn and efficient to use. The application utilizes TCP/IP to communicate with the Web Interface, allowing almost endless methods to communicate as well as an incredibly flexible security methods. If needed, communication encryption and various levels of authentication can be added to all operations.

1. Ordering

Users are able to order drinks with the use of the iOS application. Users are presented with a menu of predetermined drinks along with the ingredients used to create the drink. The application allows the user to scroll through the various drink choices. Upon finding the most desired drink, the user simply presses the order button conveniently located at the bottom of the screen. The reservation request is then sent to the server. If there are not enough of any ingredients to fulfill the drink order, the ordering will fail and the user will be alerted of the reason. The order is an asynchronous process, allowing the user to browse about the other areas of the application while the order is taking place. When the order is complete and the order ID is received, the drink is displayed in the "Pick Up" section of the application. The "Order Drink" button will be disabled if the user actively has a valid order ID.

Fig. 8. Ordering Drink

2. Picking up & Cancelling

The application gives the user the ability to pick up a drink that they have ordered from the machine. To do this, the user can navigate to the "Pick Up" section of the application to view the drink that is waiting to be picked up. While on the "Pick Up" screen, the user is presented with a barcode that is used when picking up the drink they ordered. The user will then be required to go to the machine and scan the barcode while no other drink order is actively being dispensed. In the event that the user does not pick up the drink before the order expires, the user will get a message saying that their drink order expired due to waiting too long to pick up their drink. Figure 9 shows the

overall view of the pick-up screen after the user pressed the "Order Drink" button shown in Figure 8.

From the “Pick Up” screen the user is also able to cancel their current drink order. If this is the case, the user presses “Cancel” and a cancel request is sent to the web interface. The order will only be removed from the “Pick Up” view if the order is successfully canceled, if the drink has expired, or if the drink has been picked up. The user also has the ability to verify that their drink order is still valid by pulling down on the menu. This will initiate a background operation so the user may continue to use the application while the refresh is occurring.

Fig. 9. Cancelling Drink

3. *Maintenance*

To aid in the operation and maintenance of the Under the Sun Drink Mixer, it was decided that the user would be able to update the fill level of the ingredient canisters directly from the iOS application. This concept is discussed in significantly greater detail above in section 2-A. Another feature that the user can also easily modify is the name of the machine in which they are communicating with. This can be modified through the use of a keyboard entry field within the application.

Fig. 10. Tracking

III. *Power Board*

To charge the system that will be used the battery will be the maximum power point tracking design. This includes both buck and boost circuitry in a full bridge configuration. A microcontroller will control the MOSFET’s that will implement the buck or boost circuits based on the input voltage from the solar panel and the battery voltage. Current sense modules will be used to sense current coming in from the solar panel and current going into the battery; when current into the battery is low, the battery is charged. Two IRF2104 is a high voltage, high speed power FET and IGBT driver with both high and low side output channels that will drive the buck and boost circuitry

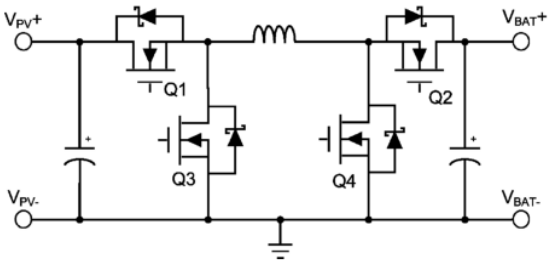

Fig. 11. Buck-Boost Converter

The Atmega328p will be used to implement the MPPY algorithm. A constant voltage method with the varying output of the solar panel and produce a constant 13.8 volts during the charge state. Once the battery is fully charge the controller will turn off charging and wait till the battery voltage drops to 13 volts, the controller will then go back into the charge state and charge the battery back up to complete charge.

1. Solar Panel

Two 10 watt solar panels from solar power tech, Inc. will be used to charge the battery. The current is .59A and the voltage is 17.3V; since both panels will be parallel, the current will be doubled.

2. Battery

A 12 volt 12 amp hour lead-acid battery will be used to power the project. If battery voltage is low, slow charge current is applied, if battery voltage is higher than 10 volts, high charge current is applied. When battery voltage reaches 14 volts, a floating charge voltage of 13.4 volts is applied. Lead-acid charge profile can be seen in figure 4.

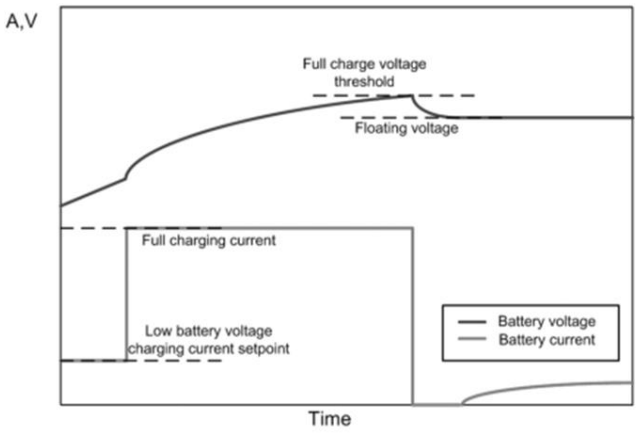

Fig. 12. Lead-Acid Charge Profile

3. Charge Controller

The ATmega microcontroller contains a half bridge IRS2104 driver which increases the voltage of Buck-Boost converter. There are currents sensing for input/output ASC712 module and it has 5V regulator to power the chip.

CONCLUSION

All of the aforementioned systems comprise a collection of components needed for the design of Under the Sun Drink Mixer. It encompasses multiple and complex components which when working together in harmony, permits for the functionality of the system for the pleasure and enjoyment of its users.

ACKNOWLEDGEMENTS

Stephen Zimmerman is currently a senior at the University of Central Florida. . He plans to graduate with a Bachelor's of Science in Computer Engineering in December 2013. After graduation, he will be serving in the United States Air Force as a Cyber Operations Officer. He is currently a cadet in the Air Force ROTC program.

Laura Cano is currently a senior at the University of Central Florida. She plans to graduate with a Bachelor's of Science in Computer Engineering and an International Engineering Minor in December of 2013. She is Currently working at Siemens Energy as a Project Manager.

Moises Dominguez is currently a senior at the University of Central Florida. . He plans to graduate with a Bachelor's of Science in Electrical Engineering in May of 2014. He has accepted an offer at Southern Manufacturing Earnest products here in Orlando as a manufacturing engineering.

Mike Tyrlik is currently a senior at the University of Central Florida. He plans to graduate with a Bachelor's of Science in Computer Engineering in December 2013. Currently an intern at Lockheed Martin Corporation as a Software Developer.

REFERENCES

[1] "*AN-2121 SolarMagic SM3320- BATT-EV Charge Controller Reference Design",* Texas Instruments, Dallas, Texas, December 2010.