THE BARTENDER BUTLER BOT (B3)

Yianni Babiolakis, Rachael Caskey, Edward Nichols, and Corey Scott

College of Engineering and Computer Science, University of Central Florida, Orlando, FL, 32816-2450, USA

***ABSTRACT* — The objective of this project is to design and implement an automated drink-dispensing and autonomous delivery system, at a minimized unit-cost, integrated by a front-end application layer. The aim of the device is to allow the user to physically select a beverage from a menu via touchscreen interface, whereupon the system automatically dispenses the individual ingredients of the drink into an empty glass and autonomously delivers the completed beverage back to the user. This prototype provides a consumer-level proof of concept for a light-hearted continuous-use appliance as a way to logistically support the hosting of private social gatherings - and serves as a precursor to assisting Food and Beverage staff in the long-run.**

***Index Terms* — F&B system automation, IR sensor, iRobot, line-following, MQTT, peristaltic pump, PID, Python, Raspberry Pi, ultrasonic sensor**

I. INTRODUCTION

Since the Industrial Revolution, the relationship between mankind and technology has been one of automation, of developing new devices which can simplify or entirely replace mundane and repetitive tasks to free ourselves and our time for bigger, better things. One form of personal automation that has remained popular since early speculative science fiction has been the concept of a robotic butler or assistant, able to perform small tasks around the house for the families of tomorrow.

The Bartender Butler Bot, or B3, aims to conquer one aspect of a home robotic-butler by automating the process of having drinks with a group of friends, a popular and timeless form of private social gathering. The device utilizes three distinct subsystems to take a user's drink order while they are seated at a table, transport their empty glass to a secondary location where the drink is automatically poured, and deliver the finished drink back to the user. This aims to prevent the frequent, disruptive breaks a diligent host must make in the middle of conversation with their guests to physically leave the table and restock drinks. The three subsystems which make up the device are the Bartender (tasked with processing the user's order into certain amounts of specific ingredients, and pouring them into the glass), the Butler (tasked with the physical delivery of the user's glass between the Bartender and the user's table), and the Application (tasked with the overall GUI, recipe storage, and taking the user's order).

The B3 is not the first drink-dispensing unit to exist. There are multiple other systems available to consumers, and henceforth, providing a greater service was one of the primary focuses of this project. Automating the process of creating a drink is appealing, but it would still take the user out of a social event, albeit only temporarily. In public venues the service of delivering drinks is provided by wait staff; however in a private setting this service is sacrificed in favor of a more intimate interaction. By creating a system that both dispenses and delivers a beverage, we hope to be able to allow the user to comfortably host uninterrupted social gatherings.

II. SUBSYSTEMS

The best way to explain the individual components of the B3 and their purpose to the greater system is within terms of it's three subsystems. The Bartender and Butler, while primarily operating on a physical level, can each interact with the Application layer via MQTT, allowing for cohesion of the overall device.

A. BARTENDER

The primary role of the B3's Bartender is to dispense the beverages requested by the user. It is concurrently responsible for key functions such as, confirming the presence and exact location and positioning of a valid drinking glass and, based on the beverage chosen by the user, dispensing the liquids from various bottles of available beverage ingredients into the awaiting drinking glass. As per our requirements, the design is set to allow at least three different ingredients for users to choose from

when selecting their desired beverage. This process will be done precisely, accurately, and at a prompt rate, in order to achieve a specified time requirement of less than 2 minutes.

Fig. 1: The Bartender with five (5) ingredients available

The physical structure of the Bartender, as shown in Figure 1, is designed to house the dispensing system. It consists of laser-cut sheets of medium-density fibreboard (MDF), off-the-shelf 2-oz liquor dispensers, peristaltic pumps, silicon tubing, and a Wi-Fi enabled microcontroller module on a custom designed PCB.

The silicon tubing extends out from each ingredient bottle to a dispensing nozzle located directly above the location of the awaiting drinking glass placed in the Butler. Once the Bartender receives the user's order, it awaits the arrival of the Butler. Upon arrival, the Bartender, by use of the Hall Effect sensors, verifies the Butler's position once it finishes the docking procedure. The Bartender also utilizes proximity sensors to confirm the presence and location of the drinking glass brought by the Butler. Once all checks are confirmed, the dispensing process begins.

For the Bartender to communicate with the other subsystems, it receives various notifications from the application via MQTT. The Bartender subscribes to MQTT to receive flags such as 'app/order' (that indicates the recipe of the order placed), the 'butler/overflow' (that prevent incidents such as an overflow that could cause unnecessary messes or even possible damage to system), and the 'app/userDeclaredAbort' (which can be sent at any time to immediately halt the system). The Bartender also publishes its own flags to MQTT once it has confirmed the accuracy of the Butler's docking process, verified the validity of the drinking glass, and the glasses exact positioning. Additionally, the Bartender publishes a flag, 'butler/start', to notify that the requested beverage has been correctly dispensed and ready for the Butler to deliver the final product to the user.

B. BUTLER

The primary purpose of the Butler is to transport the user's empty glass to the Bartender to be filled, and then transport the filled drink back to the user. Additionally, it holds the touchscreen where the user inputs their order and interacts with the GUI in general. Mounted on the Butler at a height comparable to the table which holds the Bartender is a cupholder chamber meant to secure the cup during delivery, whose base is made up of a load sensor, used for confirming the placement of an empty glass and preventing an overpour from the Bartender.

The base of the Butler was selected as an iRobot Create 2.0, since it provides sufficient balance and motor-driven transportation, can carry a significant load, has open programmability through serial-port connection, and has automatic docking features which help simplify alignment with the Bartender's pumps. Using laser-cut sheets of medium-density fibreboard and 80/20 aluminum supports, the frame was constructed to mount onto the iRobot Base, with the cupholder and user touchscreen at the very top, as shown below.

Fig. 2. Computer model of the Butler frame.

The Butler as a whole is controlled by a Raspberry Pi 4.0, which connects to the various sensors of the subsystem through it's built-in GPIO pins, and to the iRobot Create 2.0 via USB-to-Serial Port connection. This connection allows the Raspberry Pi to send preformatted opcode commands to the iRobot including direct control over its motorized wheels and a start command for its automatic docking procedure.

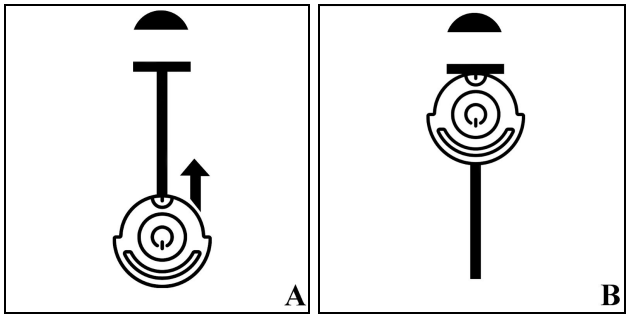


Fig. 3 PID Line Following

The additional sensors connected to the Raspberry Pi include an array of 5 simple IR sensors mounted to the front of the iRobot and connected to a multiplexer, as well as TAL221 load sensor connected to an HX711 amplifier chip. These IR sensors provide the input which the Pi uses to process through its PID line following program in order to control the two wheels of the iRobot and stay on track throughout its path, regardless of which direction it is travelling. The PID controller processes the error between the center of the IR array and the line itself, producing an output based on the error value (Proportional) the total error over time (Integral) and the rate of change in error (Derivative) in order to adjust the wheels and smoothly correct this error, whether due to drift or an intentional cure in the path. The load sensor determines whether or not an acceptable cup has been placed in the cupholder as a failsafe to prevent a trip to the Bartender with nothing to receive the drink.

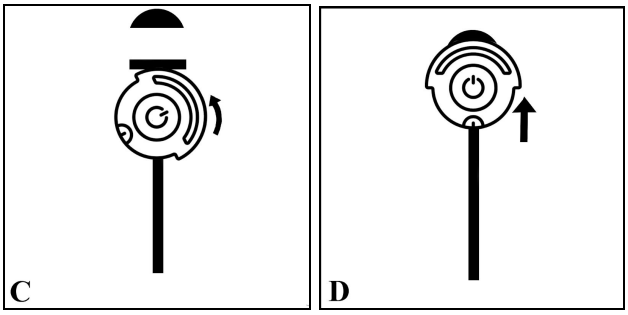


Fig. 4 Butler Docking

Both ends of the Butler's path are marked by a "tee" in the line of tape it follows, which is wide enough to simultaneously be picked up by all of the IR sensors. Since this is the only part of the path where this occurs, this specific input instance triggers the Butler's docking procedure. The docking procedure consists of the Pi commanding the iRobot to rotate in place for 180 degrees, then send the docking command. The initial rotation is due to the fact that the IR sensors are actually mounted to the back of the iRobot, a choice made in order to prevent them from being damaged while the iRobot docks. The iRobot's docking procedure is capable of smoothly and accurately aligning the Butler at the two docking stations kept at the user's table and under the Bartender's pumps.

Using MQTT communication protocols (elaborated upon in the *Application* section), the Butler is able to subscribe to flags from the Application and the Bartender, which it stands by for an order to know

when to begin its standard processes for travelling to or from the Bartender, respectively.

C. APPLICATION

The primary purpose of the application is to get user input and feedback, as well as mediate the communication and synchronization of the Bartender and Butler subsystems. To accomplish this, a seamless GUI was developed, a persistent database was connected to the GUI, and a method of communicating the information to the other subsystems was established. For developing such an application, a library that allows for machine to machine talk, a library that allows interaction with a persistent database, and a library to assist in GUI creation are required. Python provides lightweight libraries that are ideal for fulfilling these roles, such as paho-mqtt, sqlite3, PyQt5, respectively. As such the application was developed in Python 3.7, the most up to date version accessible to the Raspberry Pi 4.0 used to host the application.

The general layout of the Graphical User Interface, or GUI, is as follows. Upon application startup, the GUI will open on what is referred to as the Primary Window. From here, the user will be able to access most of the relevant information and actions available to them. The right panel will display mostly informational data, such as the status of the ingredients configured, battery charge of the Butler, and Bartender connection confirmation. The remainder of the window will consist of basic actions, such as a menu button, a quick order (immediately reorders the previous drink ordered), settings, and an exit button.

The Menu Window will regenerate every time the user goes to the page, to ensure any updates in the configuration or drinks added via the custom recipe creation are available immediately. The right panel from the Primary window will be displayed here as well, but in a thinner panel with simplified information indicators. A similarly sized panel for navigation options will appear on the left side.

The Configuration and Custom Recipe Windows are both simple windows that request user input, and then insert the user given information into their respective SQL database tables. The Custom Recipe Window can be accessed from the Menu Window and has the same left navigation panel and right information panel.

In the process of completing its assigned task, the application goes through 5 various states of functionality. In its Idle State, the application will simply be in a low power, low resource consuming mode designed to reduce average power consumption of the device. Awakening from there, it will go into its Primary State, where it displays the main window and monitors various states of the Butler and Bartender. Once the user selects the menu, custom drink, or configuration option, it will enter the Query State, which ceases monitoring of everything except for emergency interrupts from the other subsystems. It will then query the SQL database at this point to generate a list of options based on the aforementioned selection from the user. Once an option has been selected, it will move to the Primary State if the user was editing the configuration, or place an order and go to the Order State if the user was in the custom drink or menu window.

The Order State is the most delicate state of the Application, as it will constantly be monitoring the Butler and Bartender to ensure that the order is being completed successfully. Furthermore, it needs to be capable of detecting when something has gone awry and take proper steps to alert the Bartender and Butler on how to properly resolve the situation. It will need to flawlessly communicate between the two, ensuring that the Butler has successfully arrived at the Bartender, that alignment is proper, and that no emergency exit flags are raised. Such emergency flags include the cup not being present, the cup overflowing, user input emergency stop, and the Butler veering off track.

All flags that will be sent through the system will be interpreted by their own subsystem's programs. However, if a message were to be received multiple times, it would create, in the worst case scenario, a fatal crash that also results in extraneous data fed into the SQL database or Bartender subsystem. Because

this worst case scenario would need user input to manually undo the damage, this was to be avoided at all costs. On the other hand, if a message were to not be received at all, then the system would simply not work as the user told it to. Though this is a simple result, it still has a significant impact on the quality of the system as a whole. As such all MQTT messages are sent with a level 2 Quality of Service (QoS). This ensures that the MQTT messages are sent once and exactly one.

The database will consist of 3 tables that will assist the application in determining what to display to the user as viable menu options. The configuration table will be populated with data the user has input and configured with the Bartender. Upon traversing to the menu, the menu will auto generate by cross referencing the configuration table with the recipes table through the menu table. The recipes table will have a row for each ingredient in a recipe involved in a recipe, as well as its name. The menu table will then specify which rows in the recipes table contain a recipe with any given name. This will initially be populated with popular drinks commonly served in both public and private settings; however, if the user so choses, they can create a custom recipe that will be added to the end of these tables with the appropriate information. When the menu is next traversed to, it will be recreated with the new recipe (so long as the relevant ingredients have been configured.)

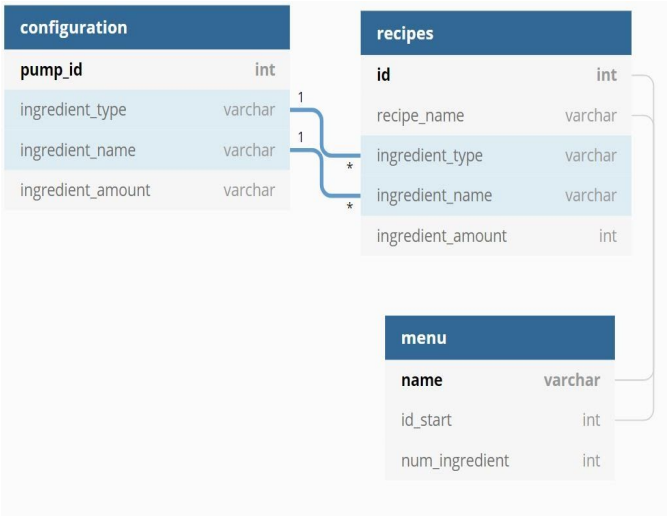


Fig. 5- Table layout for SQL database

III. COMPONENTS

A. BARTENDER

The dispensing system of the Bartender is made up of carefully chosen components that follow the applicable standards and meet the design specifications. A main standard that needed to be recognized for our project is that of the health and safety standards, due to the fact that our end product is one that will be consumed. This required the choice of dispensing components that come in contact with the liquids, to be FDA approved thus the silicon tubing was used. Additionally, digital standards such as the wireless communication standards were ones that needed to be accounted for during the component selection process, specifically that of the microcontroller. To successfully reach the goal for the specified mixing time of a beverage, as stated in our design specifications, the selection of peristaltic pumps with silicon tubing was made, along with the Feather HUZZAH with ESP8266 as the microcontroller. To complete the dispensing system the following were also chosen based on desired characteristics and ease of integration with the testing board and each of the other components; the L293D Motor Driver Breakout, the open - source 74HC4051 8-channel MUX breakout, and 2-Channel DC 5V Relay Module with optocouplers. These selections not only produced the required rate required for dispensing the liquid into the awaiting glass, they were also utilized because they successfully achieved the required accuracy needed during the testing process.

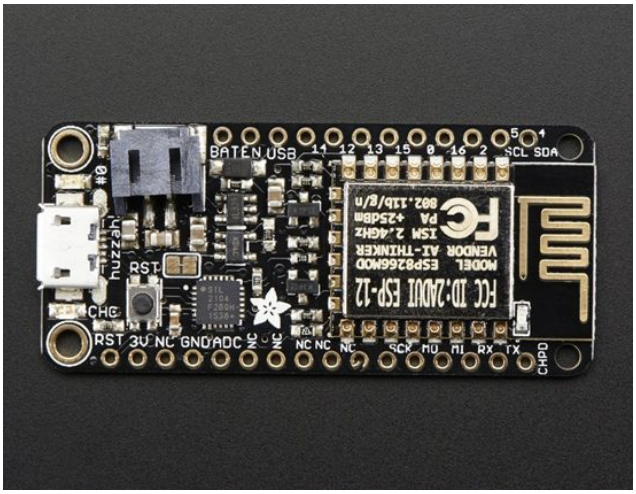


Fig 6- Feather HUZZAH with ESP826, selected microcontroller for the Bartender.

Sensors are also essential components for the Bartender in that they are needed for the purpose of validating the presence and exact location of the drinking glass brought by the Butler. Following the successful testing process of multiple sensors, the HC-SR04 UltraSonic Proximity sensors were chosen to accomplish this function. Additionally, Hall Effect sensors were added to verify the accuracy of the Butler's position once it finished the docking procedure.

B. BUTLER

The primary function of the Butler subsystem is to deliver drinks between the Bartender and the user. To accomplish this, some kind of robotic platform was needed for structural foundation and for transportation. Three viable options were identified: creating our own robotic platform from scratch, modifying a motorized base to receive control signals, or purchasing an existing platform with built-in communication protocols and preset commands. Given that creating a custom robotic platform or modifying an existing one would involve several mechanical elements, the natural choice for initial development was to use a platform already designed to accept serial commands. The iRobot Create 2.0 was by far the most flexible and development friendly robotic platform, with features such as serial command-line interface and automatic system docking.

To properly direct the robotic platform, a method of navigation was required. Given the requirement specs and environments that the system was likely to be in, the two most viable methods of navigation was a line following algorithm utilizing an array of infra-red (IR) sensors, or use an application of computer vision, via the OpenCV platform in conjunction with a camera. Though the Raspberry Pi that the Butler is hosted on has the resources and capacity to support OpenCV, and the idea of an overall system that doesn't require making a visible path in a user's home was tempting, the high difficulty and lengthy process of fine tuning such a system was intimidating given the short time frame allotted for this project. Thus, line following algorithms directed by IR sensors was selected for this prototype stage of development

A development environment is needed to receive information from the application, control multiple sensor inputs, and run a line following algorithm. Additionally, with the intent to house the Butler program and the Application within the same environment, ample computing power was a requirement. Though there are many self-contained development-oriented platforms with the computing capacity and memory space to handle these tasks, such as Intel's NUC, or Huawei's HiKey 4, none are as functionally accessible, or cost-effective as the Raspberry Pi 4.0 from the Raspberry Pi Foundation. Additionally, the Raspberry Pi 4.0 draws significantly less power than its competition, requiring a consistent 5 V at 2.5A to power on and stay actively running. This lends itself well to the requirement specification of having the Butler unit stay active on a single charge for a minimum of 90 minutes, a goal which was greatly surpassed in this project.

Since the external sensors all draw power through the Raspberry Pi's GPIO pins, the setup for powering the Pi in turn solves the need for powering the additional sensors. Originally, this was to be accomplished by tapping into the current of the iRobot's motor driver. This was a very intuitive solution in that the iRobot's docking stations constantly recharge its battery, meaning that the entire system would be charged through the docking stations at either destination for the majority of any given period of time, hypothetically giving a very efficient single-charge battery life for the system. However, this plan fell apart due to the docking command on the iRobot automatically overriding the motor driver settings, causing the power to be cut off at every instance of docking, in turn shutting off the Raspberry Pi. Instead, an external battery was purchased to constantly keep the Raspberry Pi powered on throughout the system functions, which luckily still provided an overall system charge life well above the initial requirement specifications.

Given that the Butler, whose main task is to transport liquid, would have an array of electronic equipment onboard, some level of precautionary measure had to be made to ensure any small error would not cause the Bartender to pour fluid into the Butler when it does not have a cup. There would need to be a method of checking to ensure that a container had been placed in the Butler's cup holder. A simple load sensor could be used to determine if weight had been placed, IR sensors could detect if an object was nearby when it blocked the environment's light, or utilizing OpenCV for computer vision software. For similar reasons stated before, computer vision was eliminated due to complexity and time. Between IR sensors and load cells, both were prone to human errors, such as using the system in a low-light environment or placing a weight that was not a cup in the cup holder, respectively. Ultimately, the potential of variable light levels, as well as glass cups, eventually led to the decision to use a load cell. The TAL220B was a highly rated load cell, cheap and consistent, ideal for use in the weight range of an empty cup to a full cup.

C. *SOFTWARE*

To host the programs for the Application and Butler on the Raspberry Pi 4.0 an operating system was needed. Though there are many available options, Raspbian is a Debian variant tailored to specifically providing ease of use for a Raspberry Pi. As such, we started with the baseline version of Raspbian. The default installation of Raspbian includes many general use utility services and programs; however, to further improve efficiency and to free up as many resources as possible, it was stripped down to its bare essentials.

With technology having such a large impact on modern day society, there are an innumerable amount of methods of machine to machine communication, as such there were many appealing options for the design to be able to utilize; however, MQTT stood out from the rest. Simple and free, MQTT offered the ability for a low resource machine to machine talk on a publish/subscribe basis. That is to say, when new information is published to a central broker, it is then redistributed only to machines that have subscribed to that topic. This allows a simple, centralized distribution system distinctly separated from each of the subsystems that are connected to it, ensuring that there is no irrelevant information given.

For the persistent database, there are a few standard options, such as MongoDB, SQLite3, and Oracle; however, SQLite3 offers a lightweight format of the SQL style database. In addition to this, SQL is a familiar database language, helping to reduce the amount of development time. Many existing databases utilize SQL, which allows for the potential of future development to be able to cross reference existing databases for additional recipes or other relational data, such as flavor or drink recommendations.

IV. CONCLUSION

Over the course of this two-semester project's development and implementation, this group has had the opportunity to develop on much of the knowledge gained throughout undergraduate studies, gaining an additional understanding of topics such as M2M industrial communication protocols (MQTT) closed-loop control systems (PID), and front-end GUI design. Ultimately, the process of developing these concepts as tangible parts of a greater project helped not only in gaining hands-on experience in development but also an understanding of collaborative engineering and the integration of electrical subsystems. The experience of identifying a function that seemed it could be improved upon, designing a system capable of accomplishing the tasks necessary to do so, and then engineering a realization of that design in the real world has been rewarding on many levels. Given the opportunity to continue development on this project, this group would incorporate a complete PCB for the Bartender and Butler units, provide them with aesthetic improvements, and expand the GUI to be more robust.

V. REFERENCES

[1] iRobot. "Create ® 2 Battery Power." 2015. https://www.irobotweb.com/-/media/MainSite/PDFs/About/STEM/Create/BatteryPower.pdf.

[2] van Heesch, Dimitri. "Quality of Service." Paho MQTT C Client Library, 13 Sept. 2018, 13:40:20, www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/qos.html.

[3] Poole, Ian. "IEEE 802.11b." Electronics Notes, Radio-Electronics.com, 2006, www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/802-11b.php

[4] Poole, Ian. "IEEE 802.11g Wi-Fi." Electronics Notes, Radio-Electronics.com, 2006, www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/80211g.php.

[5] Poole, Ian. "IEEE 802.11n Standard." Electronics Notes, Radio-Electronics.com, 2006, www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/80211b.php.

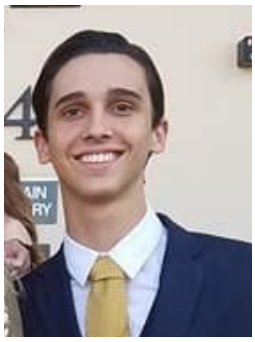
[6] Jiang, Nancy. "NSF/ANSI 25-2017: Vending Machines for Food and Beverages - ANSI Blog." *The ANSI Blog*, 3 July 2019, blog.ansi.org/2017/06/nsfansi-25-2017-vending-machines/#gref.

[7] Kelechava, Brad. "NSF/ANSI 61-2019 - Drinking Water Components Health Effects - ANSI Blog." *The ANSI Blog*, 21 Oct. 2019, blog.ansi.org/2016/10/nsf-ansi-61-2016-drinking-water-components/#gref.

[8] "NSF/ANSI/CAN 61." *Drupal*, www.nsf.org.cn/en/our-services/service-by-industry/water_and_wastewater/municipal-water-treatment/nsf-ansi-standard-61.

[9] iRobot. "Create ® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600." 2016. https://www.irobotweb.com/~/media/MainSite/PDFs/About/STEM/Create/iRobot_Roomba_600_Open_Interface_Spec.pdf?la=en.pdf

VI. ENGINEERS



Yianni Babiolakis is studying for a Bachelor's degree in Electrical Engineering at the University of Central Florida, with intent to graduate in May 2020. He plans to begin work at Burns & McDonnell after graduation, as a substation engineer in their Transmission and Distribution department. He passed his FE examination this year and hopes to earn PE Licensure in the future.

Rachael Caskey is studying for a Bachelor's degree in Electrical Engineering, along with a minor in Mathematics at the University of Central Florida, with intent to graduate in May 2020. Her goal is to pursue a career in engineering with a focus in power and renewable energy. She is currently studying to take her FE exam after graduation and is a student member of IEEE and PES.

Edward Nichols, is studying for dual Bachelor's degrees in Electrical Engineering and Economics, with intent to graduate in Fall of 2020. He transitioned directly from an internship that began in mid 2018 to a full-time role as a SCADA Engineer at Siemens Gamesa Renewable Energy since September of 2019, and actively endeavors to continue assembly of a successful career in SCADA and industrial control systems before venturing into entrepreneurship. He is currently studying for FE and CFA examinations.

Corey Scott is studying for his Bachelor's degree in Electrical Engineering, along with a minor in Computer Science at the University of Central Florida, with intent to graduate May 2020. Corey is looking to pursue a career in computer science, as well as independently developing applications to further solidify his programming foundations.