“Spicer” Automated Spice Dispenser

Adrian Garcia, Jacob Wood, Marcos Barros, and Nicholas Campbell

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

***Abstract* — A specialized Smart Spice Dispenser and Bluetooth communications device, the Spicer is a computer and electrical engineering project. Spicer is a solution to the problem of messy spice racks, and the difficulty of measuring precise quantities with a typical spice container for home use. It can select spices, and dispense them one teaspoon at a time for appropriate and precise measurements. It uses an Android OS Mobile Application as well as a Machine Learning-based gesture interface, to control its functionality and for ease of use. These features are implemented using an ESP32 for the Bluetooth communication with the Android App, a Raspberry Pi to process images, and a TI MSP-430 for control of rotation and dispensing, connected to both devices. This allows Spicer to be used to make a variety of recipes in different, convenient ways.**

***Index Terms* — Mobile applications, Food technology, Food industry,Microcontrollers, Bluetooth, and Mechanical systems.**

I. Introduction

Created to solve the problem of messy spice racks, the “Spicer” smart-spice dispenser allowed for a nicely organized, automated spice rack that can be used to accurately dispense teaspoons of various spices, designed as part of a computer engineering and electrical engineering project. Spicer is implemented on a Texas Instruments MSP-430FR6989 Microcontroller, an Espressif ESP32, a Raspberry Pi, and a custom designed 3D Printed chassis, all of which can reliably communicate with one another to allow for successful implementation of all features, among which there are Bluetooth controls, a Machine Learning Model for hand gesture detection, and a custom dispenser which measures spices. Spices are measured in one teaspoon increments and are dispensed as such, allowing for precise dispensing. Further, another benefit of Spicer is the ability to control it through multiple interfaces while allowing for use of the entire feature-set. It is possible to select the spice and dispense a desired amount using either Hand Gestures, in-app Voice Recognition, or the mobile application’s UI.

II. System components

Spicer can be split into multiple components, which will be introduced here. Below, a system components diagram can be seen.

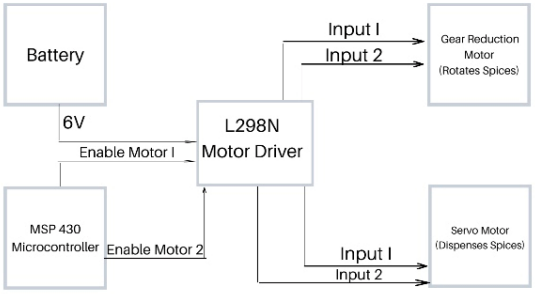


Fig 1. General Hardware Flowchart

A. MSP-430 Microcontroller

The TI MSP-430 is used as the central hub for all device operations. Indeed, although there are many devices within Spicer that contain a CPU, the MSP-430’s CPU acts as a communications hub. It receives commands from the ESP32 and Raspberry Pi through an I2C bus, and indeed it executes them after receiving them by sending other commands to the motors and to the dispenser mechanism. Additionally, the MSP-430 sends feedback of its instructions to the ESP32 and from there to the mobile application to keep all components in sync.

B. ESP32 Bluetooth Microcontroller

The Espressif ESP32 SoC Microcontroller acts as a middleman interface between our mobile application and Spicer’s hardware controls. It is what the app actually connects to when using it, and it is what pre-processes commands before sending them via I2C to the MSP-430. It also is the one responsible for keeping the act in sync with various other command sources, such as the Hand Gesture interface.

C. Machine Learning Model & Raspberry Pi

The Hand Gesture recognition system works using a TensorFlow Lite model that was trained using images of predefined hand gestures which were manually labeled. This model is run on a Raspberry Pi with an attached Google Edge Tensor Processing Unit Accelerator, which allows for high frame rates and efficient gesture recognition. The Raspberry Pi then sends commands determined from the Hand Gestures to the MSP-430, in parallel with the ESP32. To prevent the potential simultaneous use of the I2C Bus, there is a hardware semaphore built in between the Raspberry Pi and the ESP32, which ensures if one tries to send a command while the line is in use, it will not be send to avoid a crash. Instead, it will be delayed until the line is once again available. This is a very rare occurrence, and in practice it will most likely never occur, but the functionality is there to avoid such potential conflicts.

D. Dispenser System

The Dispenser for the selected spice works using a servo motor which is attached to a paddle. This paddle hooks into the selected spice container and pulls it forward, revealing a small hole which allows for one teaspoon of spice to fall at a time. This process can be repeated to allow for multiple teaspoons to be dispensed, or even multiple tablespoons.

E. Time-to-Flight Sensor

There is a Time-To-Flight Light Sensor controlling the rotation of the device, and being connected to the ESP32. This sensor looks out for the completed rotation of a single container, and then this triggers a subroutine in the ESP32 to send a special stop command to the MSP-430, which allows for exact rotations, ensuring the containers are always aligned with the dispenser mechanism.

F. Rotational Mechanism

There is a special adapter and gear system for a motor to turn a central axel attached to the spice containers, which allows for spice selection. This rotational mechanism has 3 states, turning clockwise, turning counterclockwise, and stopping. When turning in either direction, it will continue turning until it receives a stopping signal, which is triggered only by the aforementioned Time-To-Flight Light Sensor.

G. Mobile App

Finally, there is a custom mobile application which allows the device to be controlled from a mobile device that runs the Android Operating System. This application allows for touch and voice recognition interfaces to all of the other device components and thus allows for control of the entire feature-set.

III. SYSTEM CONCEPT

As a cohesive system, Spicer works by using the MSP-430 as a central control hub, to which all instructions are relayed and through which all instruction are executed. Everything else acts as a “peripheral”, which provides data to the MSP-430. However, in practice, the configuration of the MSP-430 is *of* the peripheral, with the Raspberry Pi and ESP32 acting as I2C masters while the MSP-430 is a slave. This configuration allows for multiple devices to send data simultaneously, without compromising on the MSP-430’s low power features, and allowing for the hardware to be stacked above the MSP-430 without causing potential thermal issues.

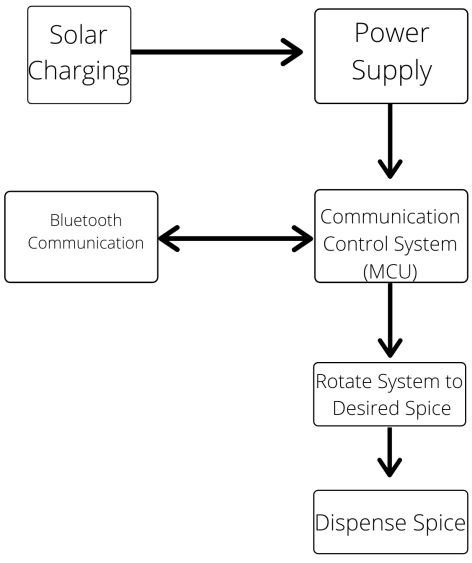


Fig 2. General System Flowchart

IV. HARDWARE DETAILS

The hardware of the device consists of the aforementioned microcontrollers, the Raspberry Pi, a camera, motor and gear system, and a custom dispenser system.

The chassis of the device was designed specially for this project, and was manufactured using a 3D printer. The chassis for the most part remains static, with an upper adapter for the containers which turns around an axel.

A. Dispenser Mechanism

The chassis contains a special section attached to where the camera is located. This special section is a specialized railing with a spring which can be pulled by a small Futaba S3004 Servo Motor in the upper section of the chassis. This servo is tied to a paddle which is aligned with the container, and upon being pulled by the servo's rotation, the paddle hits a second paddle on the container, and this moves the container into a funnel. The container itself has a hole aligned with a smaller, teaspoon-sized container, which contains the paddle. This smaller secondary container, is what is pulled into the funnel, which allows for the teaspoon of spice to be dispensed. Upon being dispensed, the servo pushes the secondary container back, and a spring on the railing aids in this process. When the secondary container is realigned with the main container, the secondary container is re-filled once again. This allows for dispensing to be performed quickly, effectively, and accurately.

B. Turning Mechanism

There is a custom adapter for a Machifit 25GA370 DC 6V motor to be held. This adapter uses a pair of gears to turn an axel. Attached to the motor, is an L298N motor driver, which receives an input from the microcontrollers, which determines the motor turning direction. Additionally, a pair of encoders attached to the motor allow for the determination of the current location of the motor. This, coupled with the Time to Flight sensor, allows for the precise turning of the containers so that they are always aligned with the dispenser. The layout of the containers includes a cross shaped support, which the time to flight sensor is aligned with. When one of the four supports off the cross block the sensor, the encoder is triggered, which moves the container by a small offset of approximately 10 degrees.

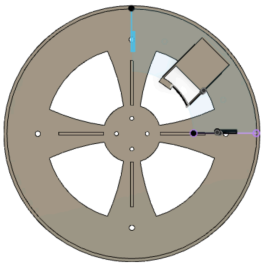


Fig 3. Rendering of the Spice Container Support

This is due to the fact that, otherwise, the containers would turn at different rates depending on direction, due to the rising edge vs falling edge readings of the time to flight sensor. This method allows us to use a the opposite edges while allowing for the container to remain aligned with the dispenser, eliminating the possibility of misalignment due to the direction.

C. L298N Motor Driver

The L298N motor driver is a 15-lead Multiwatt full-bridge that is designed for high voltage and high current use. While we didn't need the high voltage use of the L298N, we do need around 1 amp to run our rotating motor and dispensing motor. The L298N is rated for a max of 4 amps, which easily meets our requirement for needing 1 amp to run the two motors. The extra 3 amps gap adds for some realistic leeway in our design through different tolerances and if we max out the motors that the L298N is driving. Plus, the minimum voltage supply for the L298N is 4.8 volts (maximum voltage supply is 46 volts), which means we can run the 6-volt motors using this driver.

D. Gear Reduction Motor

The Machifit 25GA370 DC 6V micro gear reduction motor that operates at speeds between 9rpms to 12rpms. Since, the L298N motor driver allows for us to rotate the spice rack clockwise and counterclockwise, the maximum time for a spice to be in position to be dispensed is 3.33 seconds. While giving an output of 8.3 kg.cm amounts of torque, the Machifit will only consume a maximum of 3 Watts (0.5 amps).

E. Bluetooth Communication

Bluetooth is a standard for short range, low power, and low-cost wireless communication that uses radio technology (McDermott-Wells). With this knowledge Bluetooth is the solution to this problem. To avoid using too much power for the solar charging this is a great way to solve the problem of rationing the power we have too harshly. This also give the user the power to use the phone app to communicate with the spice holder. Since most modern-day phones give the ability to communicate with Bluetooth it makes this an easy implementation. For a wide variety of reasons Bluetooth is the best way to communicate with two devices from short range, this influenced the project to use this technology.

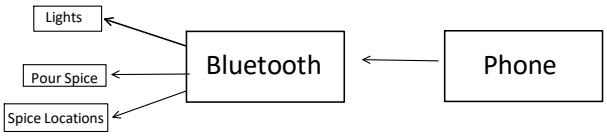


Fig 4. General Bluetooth Flowchart

The Bluetooth communication is performed using the Bluetooth Serial library provided by Arduino, which gave us the ability to set a host name for the mobile device to pair to. Additionally, the communication with the mobile app is bidirectional, when a command is sent, there is an expectation to receive a preset Bluetooth response to determine if the command was successfully received and executed. This is critical to the asynchronous behavior. If this was not implemented, then the app could send multiple requests at the same time and there would need to be a queue implemented for requests to be processed. Moreover, even with a queue structure there is a delay between when a message can be received, to avoid attempting to discover what that delay could be, since it is variable, only one process is implemented at a time.

The pairing and unpairing was tracked in hardware by flags, these flags are only accessible via callbacks. There is a looping function that checks the callback if it includes the pair or disconnected flag. After checking if an event was triggered, the pairing light would continue to blink or stay solid if it is not paired or paired, respectively. This pairing light would normally be attached to the chassis, to allow for quick and easy visual confirmation that the device and the mobile app are communicating correctly. However, due to manufacturing constraints due to the ongoing COVID-19 pandemic, the light is not currently on the chassis. Regardless, the functionality is present.

V. SOFTWARE DETAILS

The software of the device is composed of various components, ranging from the MSP-430 and ESP32's hardware control software, to the Android OS mobile application.

A. Mobile Application

The application will open and start on the main page. The first UI element on the main page will show the Bluetooth connection status with the turn table and have a button to establish a connection to the turn table if it is not already connected to the phone. Since the phone app is considered a client it must be set up as such. We set up the UUID which identifies the phone. For our implementation, we require the user to pair with the spice dispenser beforehand. Upon completion of this pairing the phone app will look for the name of the spice dispenser automatically. The client class creates a socket based on the id. This gives a way to communicate with the spice dispenser. If the spice dispenser is properly connected the user will be notified at the top of the App that this has been completed. After an action has been requested of the spice sensor, the handler changes the state to listening, which as discussed is an asynchronous behavior which because it is dependent on the response of the Bluetooth. There is no way to track how long a blocking behavior has to been done.

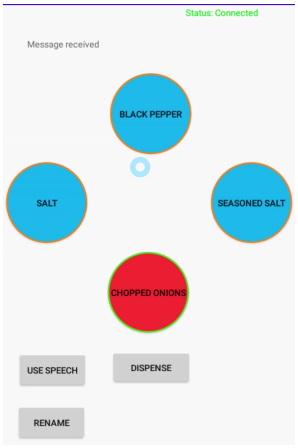


Fig 5. Main UI of Mobile Application

To help with this, whilst in the listening state all buttons are disabled. Upon message retrieval it is displayed to the user on the app. This also releases the user from the lock of one an action currently being implemented. Finally, Google Voice was implemented in this Bluetooth app. This integration allowed the user to request any spice they wanted and the quantity. This considered if the user wanted tablespoons or teaspoons, one tablespoon is three teaspoons. This function also considered getting multiple spices and spice requests. This was done by creating a queue of things to do and sending a request after each of the processes was complete.

B. MSP-430 Software

The MSP-430 software consists of a single script that initializes all its pins, and it triggers a process of resetting the servo to an acceptable state. Upon completion of this process, the MSP-430 enters a state of listening, where it will only trigger a function upon receiving an interrupt from either master device, the Raspberry Pi, or the ESP32. These commands are saved to a buffer, and are executed in order of arrival. The only exception to this is a "stop" command, which is executed immediately on arrival regardless of the buffer state. This is because "stop" commands only arrive once an instruction is completed, thus requiring that they be performed as soon as possible. Otherwise, an instruction will never be flagged as completed, and deleted from the buffer.

While the "stop" command might be simple to imagine as a simple cancellation of the current command, it is not the case. Due to the falling/rising edge problem discussed prior with the Time to Flight sensor, a stop command actually triggers a subroutine which performs a precise constant movement to cause a stop on the center of the selected container's cross section.

Additionally, aside from the "future instruction" buffer, there is a separate "past instructions" buffer, which is used for syncing purposes. In order to account for separate, instruction sources, the past instruction buffer logs all instructions received, and sends them, periodically and upon request, to the ESP32 to update the App state with. After being received by the ESP32, this buffer is cleared.

C. ESP32 Software

The ESP32 has an initialization process reliant on the MSP-430 being ready. Once it is, the ESP32 will command it to turn until a preset, marked container is in a preselected location. When this container arrives in place, the ESP32 will clear the MSP-430s instruction buffer of the turning commands, and will be ready for use. The microcontroller then acts as a Bluetooth listener, receiving commands from the mobile application, and sending them to the MSP-430. Additionally, roughly every 100 milliseconds, it polls for, and clears, the MSP-430's past instruction buffer. The ESP32 also is responsible for sending the "stop" command to the MSP-430 in any instruction case, since it has the Time To Flight sensor connected to its I2C bus. This is because, to lower power consumption, the MSP-430 is configured as a slave I2C device, and thus, the MSP-430 could not have the sensor in *its* buffer. The ESP32 was decided to be the best option under those circumstances.

D. Hardware Semaphore

It should be noted that the Raspberry Pi and ESP32 share the I2C bus to the MSP-430, which means a "collision" might occur, wherein they both send a command simultaneously. This would cause a breakdown in communication, since the message and the corresponding ACK would no longer arrive in the correct order. To prevent this, a hardware semaphore was implemented. Aside from the I2C bus, the ESP32 and Raspberry Pi maintain a direct connection to one another, and, while a communication is open to the MSP-430, the other will be blocked from sending commands forward to the TI microcontroller. When either device finishes its communication, the semaphore is set to allow new communication to occur, thus allowing the other device to send its command forward.

E. Raspberry Pi & Machine Learning Model

The Raspberry Pi relies upon a Tensorflow Lite model trained with a video feed of hand gestures performed on multiple angles, broken into frames. This model was then recompiled to use a Google Coral Edge TPU accelerator, for an increased frame-rate and thus faster response times, and was sent to the Raspberry Pi. The Pi is also connected to a camera, which it reads video feed from, processing frame by frame. The frame is then compared against the

model using the TPU, and if a match is found, the appropriate command is sent via I2C, if available. Additionally, to prevent too many commands to be sent from a single match over multiple frames, after a command is successfully sent and received, the Pi is prohibited from sending further commands after a span of 3 seconds has passed. Additionally, due to the camera placement, some preprocessing was necessary, such as the requirement to invert the image, as the camera had to be placed upside down to allow for the wiring to be in a safe spot.

TensorFlow Lite was selected over the alternative, YOLO, due to its compatibility with the small sized Edge TPU. YOLO produced similar accuracy and performance on the Raspberry Pi itself, but the increment in performance once TensorFlow Lite was compiled for the TPU was drastic, of nearly 200%.

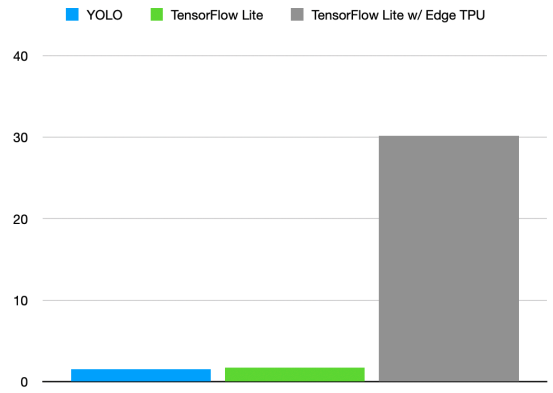


Fig 6. Comparison of Different ML Model Frame Rate in Hz.

VI. ADDITIONAL FEATURES

Spicer was built during the COVID-19 pandemic using 3D printing for the manufacturing process, and using manual processes for assembly. Because of these restrictions, several planned features, although designed and with working parts, could not be placed in the final prototype proper. These are listed here.

A. Rechargeable Battery

Designing this spice rack system, we needed a battery that could output 3.5A at 6V to make sure that our entire system will be self-sufficient. A 6-volt 12Ah battery will give us the 6-volt supply with an output current of 3.5A nominally without putting real stress on the battery. More stress on the battery will cause it to need more charging, which can cause a problem due to the size of the planned PV Panel.

B. PV Panel

The biggest factor when finding the right PV panel would be the time it takes for it to recharge the battery. While the spice rack would only be used during certain times of the day like dinner, lunch, or even breakfast and for maybe 30 minutes maximum, the PV panels won't have to be too large to recharge the battery to full capacity. By selecting an 18-volt 4.2-watt PV panel, the time to fully charge a 6V 12Ah battery would be roughly 34 hours (see equation below from "Estimating Solar Charge Time for Batteries" [4]), but that is to recharge the battery completely when it is drained. But like stated before the battery will never be completely drained since the user will only have the battery running for only 30 minutes at a time.

$$Time\,to\,Fully\,Charge = \frac{2*(Battery\,Capacity\,(Watt\,hours)}{PV\,Panel\,Power\,(Watts)}$$

Eqn 1: Charging Time for a Solar Panel

C. TPS62147 DC/DC Switching Regulator

The TPS62147 DC/DC Switching Regulator will be used to drop the battery voltage from 6V (VBAT) to 3.3V (VOUT) and maintain a maximum output current of 2 amps, which will to run both the MSP430 and the ESP32. Like previous components, the converter will need capacitors to dissipate the noise that is caused by the switching of the component. Then the resistors will need to have a value so that the DC drop will give an output of 3.3V. When making R1 equal to 560 KΩ and R2 equal to 180 KΩ, we will be able to get our 3.3V output (it is actually equivalent to 3.29V)[1].

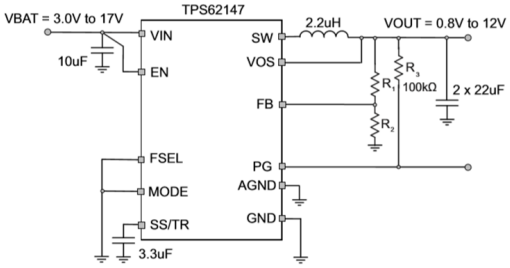


Fig 7. DC/DC Switching Regulator

D. LM317 Solar Panel Linear Regulator

In order to connect our 18-volt, 4.2-watt panel to our 6-volt battery we will need to add a circuit to drop the voltage down from the panel to the battery, but keep the amount current and power the same. So, by adding a LM317T voltage regulator to step down the voltage from 18-volts to 6-volts we will need to use the circuit below. The circuit shows how the resistors will set the output voltage to be equal to 6 volts and the capacitor is used to keep the noise coming from the solar panel low [2]. Then the transistor is to modulate the current that will be going to the battery from the battery so that the solar panel will not jeopardize the integrity of the battery.

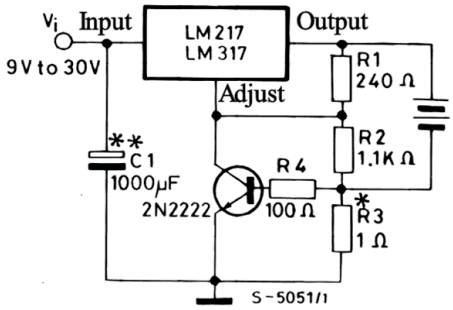


Fig 8. LM317 Solar Panel Linear Regulator

E. Online Recipe Search

The mobile application also has the ability to retrieve information from recipes from the Internet, which would let the user potentially request spices in the exact amount that is required for a recipe they may be interested in cooking. The online recipe search feature uses a separate activity. This activity would retrieve recipe information based on the user's search query, process this information retrieved from the Internet, and send any request the user may make to the main activity, which handles communication with the spice dispenser.

F. Edamam API

Edamam is a company which provides software developers with different application programming interfaces (APIs) to easily retrieve nutritional and recipe information from different websites in one place. This is done by sending an HTTPS request to the Edamam API with a string that contains a search URL with certain parameters that specify what the API should retrieve. The API then returns a JSON object (a universal data format created to allow for more convenient data interchange [5]) with certain key-value pairs, as specified by the Edamam API documentation [6]. There are different levels of access to the features of the API, with the free level restricting some of the parameters that the API returns to our application. However, none of those affect the functionality we intend to have for this project.

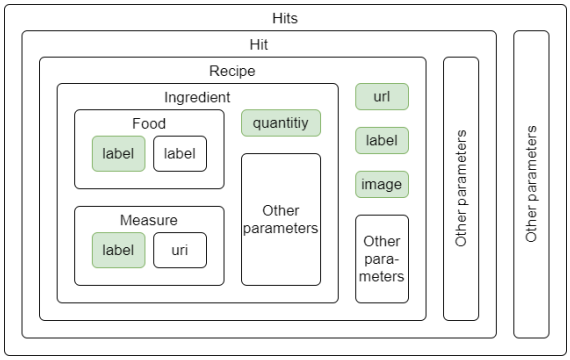


Fig 9. API Structure

The JSON object we retrieve as a result of sending an HTTPS request to the Edamam API is illustrated in Figure 1. The result comes in form of a Hits JSON object. We extract the Hit array from this Hits object. Each Hit object contains information for each individual recipe retrieved by the API. Each Hit object has a Recipe object which contains, among other information, the recipe title, recipe URL, recipe image and an array of Ingredient objects. We use the recipe title, URL and image URL to populate UI elements in the recipe search activity. The Ingredient object contains Food object, which has the name of each ingredient which contains the name of the ingredient, a Measure object which contains the name of unit the quantity of the ingredient is measured in. The measure amount of the ingredient is also used. After processing the

returned JSON, the UI is populated with data, and, based on the currently available spices on Spicer, appropriate selecting and dispensing commands will be sent to the Main Activity, which controls communication with Spicer.

G. Printed Circuit Board

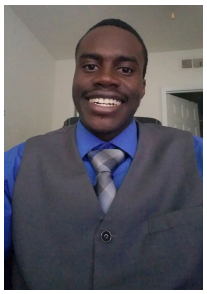
A printed circuit board was designed and manufactured for use with Spicer, which would integrate all the processing and microcontrollers into a single board. However, due to the COVID-19 pandemic, it was impossible to solder the components to this board, as no equipment to do so was available, and the businesses dedicated to this were not currently open and available for use by the public. Because of this, Spicer currently works with our prototype circuit composed of the microcontrollers with a specialized soldered adapter for ease of use. This circuit, and the PCB would have the same functionality, the difference being of course the reduced risk of failure with a PCB, as there would be less wiring, and the improved form factor of Spicer, as there would be more space available in the internal area of the device that could allow for bigger containers to be used.

VII. CONCLUSION

Spicer is a convenient and easy to use way to access spices while cooking. Its available interfaces will allow even the busiest of Chefs to enjoy its rich features, and, were we in more normal circumstances, outside of the ongoing pandemic, then the device would be even better. Spicer has a bright future, with potential growth and incremental improvements in the long term.

VIII. THE ENGINEERS

Adrian Garcia is a 21 year old Computer Engineering student who is currently looking for work in the Software development industry. He wants to focus his career in the development of software over hardware, while using his hardware studies for optimization of software, rather than development of new hardware.



Nicholas Campbell a 24-year old computer engineering student that loves seeing computers interact with the outside world. He hopes to have a software engineering career that is filled with robotics. He has accepted a software engineering position at MRSL Real-Time Systems.

Jacob Wood is a 22-year old graduating Electrical Engineering student who is taking a job with Lockheed Martin in Orlando Fl, as an Associate Electrical Engineer that will specialize in product support for military systems.

Marcos Antonio Barros is currently a senior at the University of Central Florida and will receive a Bachelor of Science in Computer Engineering in August of 2020. As of writing of this paper, he has accepted a job offer at Capco as an Associate. He plans to eventually continuing his studies by returning to the University of Central Florida to obtain a Master after obtaining practical work experience in software engineering.

IX. REFERENCES

[1] Texas Instrument, "TPS62147, TPS62148 High Accuracy 3-V to 17-V 2-A Step-Down Converter With DCS-Control™." https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=http%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftps62147

[2] "LM317 3-Terminal Adjustable Regulator." *Texas Instrument*, www.ti.com/lit/ds/slvs044x/slvs044x.pdf.

[3] STMicroelectronic,"DUAL FULL-BRIDGE DRIVER." www.st.com/content/ccc/resource/technical/document/datasheet/82/cc/3f/39/0a/29/4d/f0/CD00000240.pdf/files/CD00000240.pdf/jcr:content/translations/en.CD00000240.pdf.

[4] Voltaic, "Estimating Solar Charge Time for Batteries." https://blog.voltaicsystems.com/estimating-battery-charge-time-from-solar/#:~:text=%20How%20to%20Estimate%20Solar%20Charge%20Time%20,The%20solar%20charge%20times%20above%20assume...%20More%20

[5] https://www.json.org/json-en.html

[6] https://developer.edamam.com/edamam-docs-recipe-api