Barkaroo: The Collar Companion

Cody Khong, Vincent Martinez, Allan Nevalga, Jesse Ray

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

***Abstract* — The barkaroo: collar companion is a device that will help any new dog owner to understand the responsibility of taking care of their pet. The collar will sit on the dog's collar and track movement and keep a daily schedule via a provided phone app for easy use. The hope is to protect dogs from being forgotten after the initial excitement and show the daily task to the new owner.**

***Index Terms* — Bluetooth low energy, Global Positioning System, Accelerometer, Thermoresistivity, Speech processing, Human voice, Light Emitting diode**

I. Introduction of collar

This product is to help new pet owners to take responsibility for their new pet, as we all discussed our love of furry companions and saw how sad it is to see people forgetting to take care of their animals after the initial excitement goes away. We noted that this was especially true for small children as they do not comprehend the responsibilities associated with owning a pet. By making a collar/harness with automated reminders triggering LEDs to notify the child or forgetful owner of their dog needing their attention, whether this be the dog likely needing to; go to the bathroom, be fed, be walked, or be given affection. When looking up this idea we found there were plenty of activity trackers for pets (dogs especially); LINK AKC, PetPace, Whistle, and Fitbark to name a few of the biggest names in the market space at the time of our research. These competitive devices help to track the activity of your pet, alert you to when your dog might have got out and then help you to find them. They do this by monitoring your pet using a heart rate sensor and a GPS tracker. Although these products have great use cases, we believe that they are missing important features that do not allow them to be used for helping forgetful owners and teaching children the value of responsibility with their pets.

We want our product to be a lightweight, cost effective, low power and feature filled product that offers more for the same price as compared to the competitors mentioned earlier. Our collar will implement the notable features from leading competitors such as GPS tracking, real-time notifications, and customizable mobile applications. Together with this internal information will be from certified sources from general to specific size dog hygiene to each of their own necessary diet. This in return will allow our LED to toggle for when it is the right time for feeding or to go outside. We want our product to be usable and available to everyone, so we are taking steps to make the collar more accessible. So, we will be implanting a way to change LED color to accommodate those with colorblindness. We also had a hard time finding the best parts as of the writing and prototyping of this product there is a part shortage so finding the best and cheap parts will be hard but not impossible.

II. Hardware overview

A. Microcontroller

The ESP32 WROVER although not as power efficient as some of our other choices it does have features where it counted most. It also has a community unmatched by almost any other chip and therefore it checks off the box for ease of development while still giving our Electrical Engineers enough freedom to design around and with it, with up to 150Mbps speeds allowing for quick uploads and downloads of small sensor data used in our project. Has multiple kinds of sensor interfaces including ADCs for calculating values directly from an analog voltage reading such as for a Thermistor as well as I2C and SPI for other sensors such as our accelerometer and RGB LEDs and some DAC for our speaker although low bit count, we want this collar to be small. The major drawbacks of the ESP32 are the package size and the power draw of the complete unit. After much discussion the ESP32 WROVER was chosen for our project.

B. Accelerometer

The LIS2DW12 is a three-axis accelerometer that is ultra-low power (sub 1µA). The chip has "very low noise: down to 1.3mg RMS in low power mode". This chip offers a 32 level FIFO for storing measurements and 16-bit outputs for high resolution data. Has gesture recognition which opens to the possibility of being able to recognize when a dog might be eating or drinking or etc. The package is also the smallest package size out of all other

parts, just 2x2x1mm. Lastly it is readily available which is again a major reason to use this chip.

This sensor from STMicroelectronics uses I2C or SPI, I2C requiring a two-wire interface before accessing the sensors while one SPI interface however this is needed to properly access the sensors. The module also has single data conversion on demand meaning this will give us a huge advantage in processing data because MCU ultimately gets to decide when the sensor should measure it. Digital output interface provides high speed transmission on demand and can enter low power mode with given the right parameters. LISDW12 high configurability allows it to meet the demands for BarkaRoo's needs. This device has also an embedded temperature sensor that is stored as two's complement data and left justified.

$$X_{grav}(t) = \alpha(0) * [(x_{raw}(t) * \beta(0)) + (x_{raw}(t-1) * \beta$$
$$(x_{raw}(t-2) * \beta(2)) + (x_{grav}(t-1) * \alpha(1)) +$$
$$(x_{grav}(t-2) * \alpha(2)] \quad (1)$$

We will be using SPI to communicate to our microcontroller unit and will automatically pass the values of each direction's acceleration by request from the appropriate register but it does not natively have any sort of step counting functionality. Now that we are able to set and read, we can use this information to detect the direction of the gravitational acceleration and thus determine if a step is taken. To do this we are using a series of software implemented filters to separate gravitational acceleration from user acceleration and then get an output that is used for tracking the repeated vertical oscillations that all animals experience during a step.

We start by passing the data from the accelerometer directly into a low pass filter equation for a filter in Equation (1) with a cutoff frequency of 0.5Hz to separate gravitational acceleration from our data as we know gravitational acceleration should be very near constant. Once we have gravitational acceleration, we can subtract this value from the value read off of our sensor and this gets user acceleration. Once we have these separate values, we then dot product them in an effort to isolate the acceleration only in the direction of gravitational acceleration. Finally, we pass our calculated values through a high pass and then low pass filter to get rid of any peaks that are too fast to be steps as well as peaks too slow to be steps.

C. Temperature sensor

We found many IC chips that are low power and contactless with shutdown/ interrupt features, but these chips are hard to find and could cause more issues if say a chip went bad and we need a replacement. Seeing how little time we have and having an IC chip would be nice and could allow more flexibility, better accuracy and a way to communicate an interrupt but some of the chips require calibration and making sure that they can communicate with already existing parts like the processor. So, we made a choice to go with the thermistor NXRT15XV10. Its very light, small and accuracy is fair with a +1C but this product does not need to be extremely accurate to read the ambient temperature.

$$T = \frac{1}{(\frac{1}{T_0} + \frac{1}{B} ln(\frac{R_t}{R_0})} \quad (2)$$

Where: T is the temperature to be measured in Kelvin; To is the reference temperature in Kelvin for 25 degree Celsius; Ro is the thermistor resistance at To; B is Beta or B parameter, provided by the manufacturer in their specification.

Implementation of the thermistor won't be particularly difficult as ESP32 GPIO pins can be configured as ADC (Analog to Digital Converter). Luckily for us we know that ESP32-IDF has built in functions for ADC applications. Our team has also noted the non-linearity of ESP32 ADC where there are instances that an inaccurate temperature reading above a certain temperature will occur. ESP32's high CPU clock and small factor is also receptive to noise as we collect data from the thermistor for analysis and adjust by adding a filtering algorithm or adding a small capacitor. [5]

D. LED

We went with the CHINLY RGB LEDs because they have a good build with waterproof rated at IP67 that will allow us to comfortably place the led on the collar. With RGB we can make these lights accessible to those with color blindness using a setting in our phone app. The only downside to the LED inclusion is that we need another power line as these LEDs need a 5-volt line which can drain our battery. But we will not use it too often only when needed.

E. Speaker and amplifier

This speaker is waterproof rated at IP67 which can last for a while submerged in water. With a very nice small size of 12x6mm and a dB rating of 116 this little speaker has very good sound quality, although it as an impedance of 32Ω which is fine for what we want for a speaker and only a frequency range of 300 to 7kHz which is great as this sits perfectly in the human speech range. We also had to keep in mind that dog's ears (threshold of pain at 95dB) are more sensitive to noise then humans (threshold of pain at 130dB) so we will lower the rails of the amplifier and set up a high pass filter at 7khz so that its loud enough to be heard not loud enough to hurt the dog. We will use the ESP32 DAC to send out the audio signal. The only bad thing is that this DAC is only 8bits, with testing this is passable and is loud and clear enough to be heard. We will add a low pass filter to the amplifier as well to clear any noise that may come from the DAC. (equation (2) provide by lumissil microsystem)

$$f = \frac{1}{2\pi RC} \qquad (3)$$

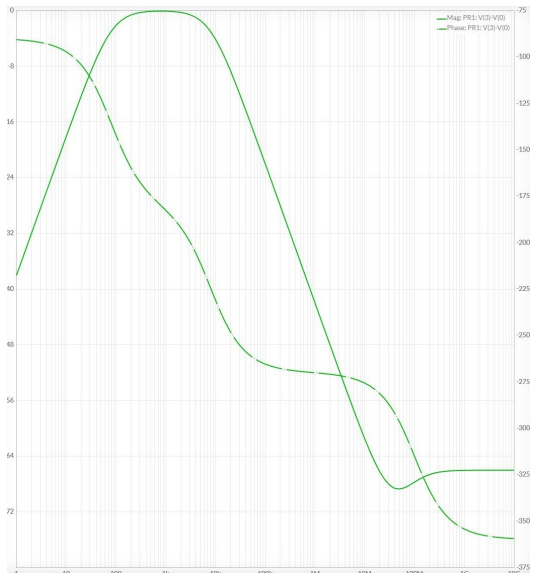

Fig. 1. The range of frequency with respect to dB that the speaker will go which correlates to human speech.

F. Lipo Charging Chip and battery

The LM3658 will be our choice, comes with safety features such as: multiple charging time monitoring systems, over-current and over temperature monitoring, and pre-conditioning for our battery if it is ever overly depleted. Other great features for our project include a 1A charging current if using a wall adapter and a USB limiting feature for when this product might be connected to a USB port such as one located on your laptop. The chip also comes with a sleep mode, low-quiescent current, and a package size of 3x3mm. The most useful feature of this charging chip is that it is designed for the purpose we intend to use it in (USB power interfaces) so it can communicate through USB to any device or just take its max allowed current. The only real drawback of this chip is we are limited to 1 cell batteries if we ever wanted to change the battery voltage, we would also need to change this charging IC, but it comes at the benefit of a much cheaper price for the features included

Part:	Max current (A):	Min Current (A):
Accelorometer	0.00009	0.000003
LIPO charger IC	0.0006	0.000005
Boost	0.000109	0.000005
Buck	0.000948	0.0000006
ESP32	1.1	0.0008
Speaker	0.01	0.006
RGB LEDs	0.1	0.02
Thermistor	0.001	0.001
GPRS/ GSM	0.5	0.0007
# of Hours In a week:	168	168
	Total worst case current draw:	Total best case current draw:
	1.712657	0.0285136
Amount of time a 1AH could run it for (Hours):	0.583888076	35.07098367
Amount of time a 1.5AH could run it for (Hours):	0.875832113	52.60647551
Amount of time a 2AH could run it for (Hours):	1.167776151	70.14196734
Amount of time a 3AH could run it for (Hours):	1.751664227	105.212951
Amount of time a 4AH could run it for (Hours):	2.335552303	140.2839347
Amount of time a 5AH could run it for (Hours):	2.919440378	175.3549184

Fig. 2. Estimated power usage over time with add parts

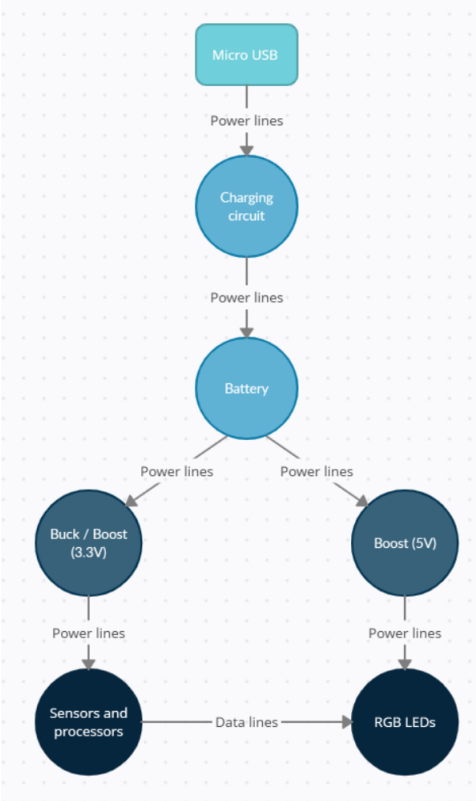

Fig. 3. Power supply layout

G. *Global positioning system*

We wanted a way to track our pet with our product. Originally, we planned to have LoRa as our GPS as it did not require a subscribing based only issue is that LoRa is a new technology and only one tower only exists in Florida where this product is being developed another reason we moved on from this technology is because LoRa uses very low frequency signals which is fine for some application as some of the receivers will set on high areas but our product will be on a pet which is low to the ground, so the fear of this not working efficiently.[1] We went on to work with the SIM800L as this one only needed a cell tower as it uses GPRS using only cellular towers, unfortunately we ran into issues with this module as this device uses a specific device tracking algorithm that the current company does not fully support anymore thus having us move forward to Ai-Thinker A9G module.[3]

III. Software overview

A. *Bluetooth Low Energy*

One mode of communication that we will be using in this project will be Bluetooth Low Energy (BLE) to communicate information such as if the dog has the collar on. what it is doing, and other activities we wish to track and convey to the user though their phones.

One area where we spent a majority of our time during this project was implementing a way to send RGB values over BLE to our dog collar and having the color properly displayed. In order to implement this, we first started by setting up all the proper permissions and server/client attributes on our mobile application. Our first step was ensuring that BLE was enabled in our mobile application. After this was coded successfully, we had to code a proper way to scan and connect to our ESP32. This was done utilizing the ESP32's Unique Universal ID (UUID) and address.

Once the connection was successful, we started modeling how we would send a RGB value to ESP32 in order for it to read the value and display the color on our LEDs. We started by making a color wheel that was able to cycle through all combinations of RGB values from (0,0,0) to (255,255,255). The color wheel was implemented in a way that allowed the user to visually choose a color and the wheel would return RGB hexadecimal string (#000000) to (#FFFFFF).

Once the user chose a color, the wheel would display a hex string with the values of the color that user chose. Our next step was to take this string and send it over BLE to our ESP32. This was done by sending a "send" type to our ESP32 that would let it recognize that it was about to receive three RGB values. Then, we had a function that would split the hex string into 3 values to send to the ESP32. The ESP32 was coded so it knew that the first value after the "send" type value was for the red LED ®, the second value was for the Green LED (G), and the last one sent was for the blue LED (B).

After receiving the values from our mobile application, the ESP32 would go on to convert these hex values to decimal values. These decimal values would then be the brightness value that the corresponding LEDs would change to after being processed.

B. ThingSpeak and Wi-Fi

ThingSpeak IoT platform was another suitable choice for BarkaRoo purpose and implementations. Having a free limited option to sending no more than 3 million messages each year earning this platform the highest rank in that category. ThingSpeak uses channels to send and retrieve data from the microcontroller unit. It has a social media plugin compatible with Twitter for automatic "tweet" notifications of triggered events as well as Twilio APIs and support most if not all mobile and web applications. ThingSpeak ability to transform our collected data from ESP32 to a visualize data representation on a cloud platform solidifies our components working status. This visualize data also give our team feedbacks on outputs that could be inaccurate, wrong or simply disabled components for debugging purposes. This rapid prototype testing also give a quick assurance that our sensors will have no issues in cooperation of our software application.

In regards to integrating ThingSpeak into our project, we used the platform's API keys and Channel ID in order to facilitate communication between our mobile application and the ESP32. ThingSpeak creates unique API keys for reading and writing that were integrated into our firmware for the ESP32. Similarly, these keys were also included in our mobile application. To test the read and write features needed for our project, our mobile application would seed a request number through to one of ThingSpeak's fields. The ESP32 would read this request number and write corresponding data according to the request number to a different field. Our mobile application would also have a "read" button that allows us to read the data just uploaded by the ESP32 of a particular field.

C. Global Positioning System

The Ai-Thinker A9G module is still as capable as the initial choice of SIM800L with the difference that A9G supports GPS tracking. Without being said, the software development platform for A9G with our initial prototyping of its capabilities was done over Ai-Thinker Serial Tool that allows for AT based commands for GSM/2G connectivity initiation, GPS tracking and SMS functionalities.

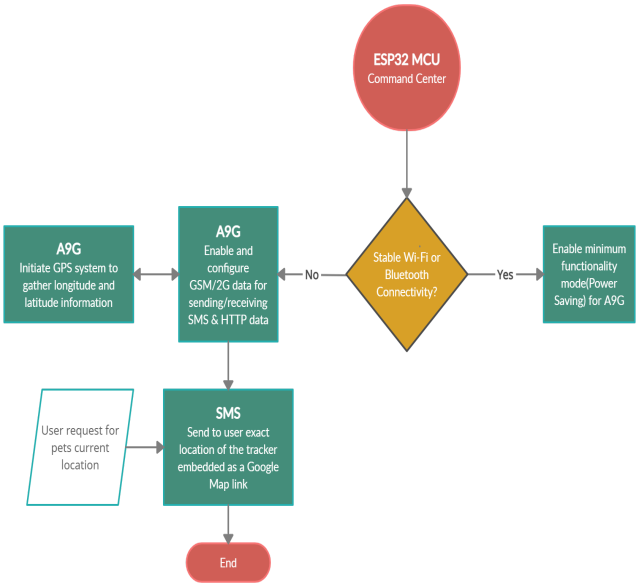

Fig. 4. A9G Functional Overview

After we have verified the AT command capabilities of the A9G we moved forward to completely integrate its functionalities through ESP-IDF software development environment of ESP-32. The primary language ESP-IDF will be done in Embedded C. The software design structure for A9G programming consists of individual function sets to 'Enable/Disable' GSM/2G connections and GPS systems when it is not needed, sending SMS with relevant information for location, GPS retrieval of location and parsing of NMEA(National Marine Electronics Association) GPS data into latitude and longitude data. This structure was purposely designed by our team to facilitate rapid testing, debugging and deployment of each function our wireless module will provide for our system.

D. Android Studio

The mobile application development aspect of this project will be a key component in addition to the hardware design. This application will be the main mode of interaction users will have to get the hardware we design to be usable for the features we plan on including. We chose to make a mobile application for this project since the collar we are designing would be best suited for interaction through the phone and on-the-go. As such, a stationary web application that requires users to sit at a

personal computer was ruled out. For the platform, the team decided the development on the Android platform would be most beneficial for us throughout the project. In large part, this is due to the open-sourced nature of Android. Android allows development, integration, testing, and deployment to be more manageable with ample documentation online that will help us with any issues that may arise during the process.

After narrowing down our options to an Android, mobile application, we looked at tools we believe would best suit this project. In order to develop an Android application, we decided to use a mix of both Android Studio and Visual Studio Code as our IDE for this project. Android Studio will allow us to see a glimpse of our final application as we code with its built-in Android Emulator. Based on IntelliJ, Android Studio offers a platform that allows us to code with included templates, integrated version control tools using GitHub for team development, Java/C/C++ support, and much more included. Since Android Studio is supported by Google and has been extensively used over the years for other application development, this established platform allows us to fully utilize the ample documentation and online resources while developing our mobile application. In tandem with Android Studio, we will also be incorporating Visual Studio Code into our workflow since most of our team members have experience with this IDE. With Visual Studio Code, we will be able to keep our options open and ease into the Android Studio environment.

The tools we have at our disposal seem best suited for how we want to proceed with our development process. At that time (6/24/2021), the mobile application development aspect of our project was constrained mostly by the lack of hardware to work on. As such, we are starting mobile application development with a barebone prototype. This prototype will be with Android Studio with a superficial, UI design that will act as a placeholder for the features we plan on including once the hardware is completed. This barebone app gave us a bit more perspective on how we want our finished product to operate with the functionalities we plan on having with the placeholders within the app. Additionally, we were able to simultaneously work on the application and the hardware while adding and removing aspects of our project we find achievable or otherwise. From our bare prototype with dummy buttons and tabs, we integrated working features as they are done to test on the Android application (blinking an LED, receiving a Bluetooth transmission, etc.) one-by-one to test functionality. After ensuring the feature we are testing works in a controlled, isolated environment, we brought the application with the feature we are testing to a "live" environment to simulate what the end user's experience will be like without the developers interfering with the code and hardware. From this process, we saw how well this testing methodology worked and repeated the process throughout development with each feature filling in the placeholder on the prototype until we have a stable, feature-filled, mobile application.

IV. Overview of the Collar

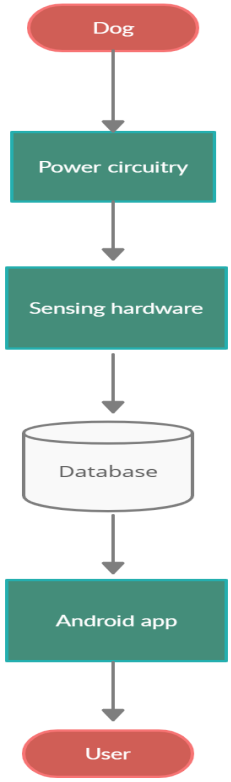

Fig. 5. The complete layout

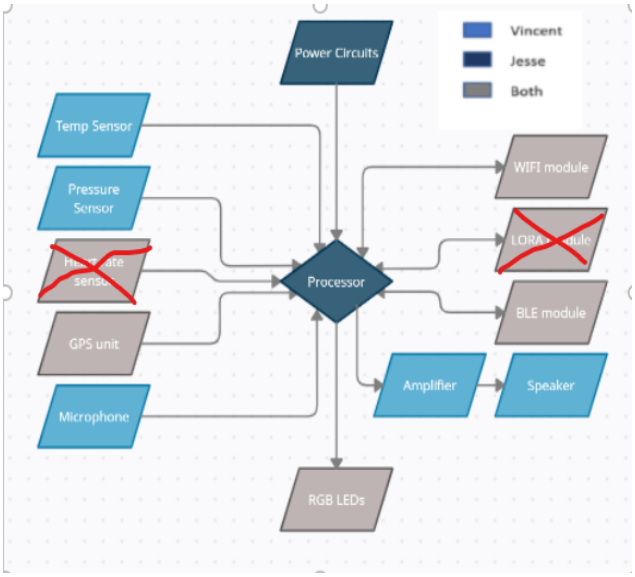

Fig. 6. hardware layout (due to some constraints/decisions we have cross out part we are no long working on)

The barkaroo collar will have many different parts that will work in tangent to one another. We will have a temperature sensor running with a pressure sensor to let the user know if the dog is wearing its collar. gps will text the user the dog's location if requested, speaker and LED will go off with a present of audio lines giving attention to the dog, Wi-Fi and BLE will be sending data from the esp32 to the android app. This device will be able to track your pet though many ways.

Communication is key, using the esp's different communication methods will be helpful A Universal Asynchronous Receiver/Transmitter (UART) is a hardware feature that handles communication (i.e., timing requirements and data framing) using widely-adopted asynchronous serial communication interfaces, such as RS232, RS422, RS485. A UART provides a widely adopted and cheap method to realize full-duplex or half-duplex data exchange among different devices. The ESP32 chip has three UART controllers (UART0, UART1, and UART2) that feature an identical set of registers for ease of programming and flexibility. Each UART controller is independently configurable with parameters such as baud rate, data bit length, bit ordering, number of stop bits, parity bit etc. All the controllers are compatible with UART-enabled devices from various manufacturers and can also support Infrared Data Association protocols (IrDA). [2]-[4]

I2C is a serial, synchronous, half-duplex communication protocol that allows co-existence of multiple masters and slaves on the same bus. The I2C bus consists of two lines: serial data line (SDA) and serial clock (SCL). Both lines require pull-up resistors. With such advantages as simplicity and low manufacturing cost, I2C is mostly used for communication of low-speed peripheral devices over short distances (within one foot). ESP32 has two I2C controllers (also referred to as ports) which are responsible for handling communications on the I2C bus. Each I2C controller can operate as master or slave. As an example, one controller can act as a master and the other as a slave at the same time. [2]

The ESP32 has four SPI peripheral devices, called SPI0, SPI1, HSPI and VSPI. SPI0 is entirely dedicated to the flash cache the ESP32 uses to map the SPI flash device it is connected to into memory. SPI1 is connected to the same hardware lines as SPI0 and is used to write to the flash chip. HSPI and VSPI are free to use. SPI1, HSPI and VSPI all have three chip select lines, allowing them to drive up to three SPI devices each as a master.[2]

V. THE CONSTRAINTS

The health and safety constraints will be discussed for our device. This dog collar will be placed on dogs and used by humans. Taking these two groups into consideration, necessary precautions and planning must take place to ensure that no one gets injured or harmed in any capacity. The device will be primarily around the dog. Therefore, we must ensure that no components or features will harm or disturb the dog. Additionally, the device will be used by humans. Any testing or use by humans must be carefully planned and taken into account to make sure that no one can be injured or harmed by the device. If the device is clearly not safe or healthy to use, then the final product will either not be released or even approved for release.

While developing and testing the product, we will ensure to the best of our abilities that our device does not trigger any health and safety concerns. Since this device will predominantly reside on the dog neck, we will have to ensure that the battery, wiring, and all components with electricity will not malfunction at any point and injure the dogs or their owners. Once we can ensure that device has a certain level of durability, we do not see the collar raising health and safety issues. Although the potential of our hardware malfunctions while on the dog is a concern, we do not foresee this as a huge hurdle for us to overcome since most of our hardware is low voltage. To

further ensure the safety of the dogs, we will be making the enclosure for our circuitry and wiring waterproof and dustproof to the best of our abilities.

Varying weather conditions is a prominent standard usually discussed within product developments. These conditions will allow operability of the device within allowed parameters specified by our team of engineers. As per our development procedure, water protection is the top priority in design of the enclosures. IP65 enclosure is the ideal bound of the development of BarkaRoo collar making sure splashes of water will prevent damage within the electrical unit. Humidity is a main concern as well in connection with water damage. High humidity can cause water condensation that eventually leads to corrosion if not correctly dried. Low humidity on the other hand is susceptible to electrostatic discharge that can result in frying of device components.

VI. Board Design

When designing this board for our group we needed a Li-ion/ LiPo charger/management IC as well as power ICs that could convert the variable voltage of the battery to three-point-three volts and five volts. To do this we decided to first use a boost converter to get the voltage of the battery to a constant five volts and then from that five-volt rail make our three-point-three-volt rail using a buck converter off of the five-volt rail. This design requires that we have a boost converter that can handle both the five-volt rail demand as well as the three-point-three-volt rail demand. This meant that the max current draw of the boost regulator should at least be one-point-five-amps. About one amp for the RGB LEDs and then another half an amp for the ESP32 and sensors.

Acknowledgement

The authors wish to acknowledge the assistance and support of the professors at the University of Central Florida, Siying Yu, Seif El Shafei, Anirudh Pise, Charlie Jordan and Samuel Richie

References

[1] "What Is LoRaWAN® Specification." *LoRa Alliance*®, 6 May 2021, lora-alliance.org/about-lorawan/.

[2] "UART ESP32." *ESP*, Espressif, docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/uart.html.

[3] "Global Positioning System." *Wikipedia*, Wikimedia Foundation, 2 July 2021, en.wikipedia.org/wiki/Global_Positioning_System#Regulatory_spectrum_issues_concerning_GPS_receivers.

[4] "ESP-IDF." *IoT Development Framework I Espressif Systems*, www.espressif.com/en/products/sdks/esp-idf.

[5] Cheung, Henry, et al. "Using a Thermistor with Arduino and Unexpected ESP32 ADC Non-Linearity." *ETinkers*, 29 Apr. 2020, www.e-tinkers.com/2019/10/using-a-thermistor-with-arduino-and-unexpected-esp32-adc-non-linearity/.

Cody Khong is a 22 year-old Computer Engineering student receiving his degree from the University of Central Florida. He hopes to pursue a career in database engineering and administration, application development, or biomedical technologies. He is currently interning at Phoenix Logistics LLC. where he hopes to start his career after graduation as a full-time Software Engineer.

Jesse Ray is a 24-year-old Electrical Engineering student. Jesse wishes to pursue a career in power electronics, specifically charging systems for electric vehicles and the electric vehicle's internal charging and battery management systems for companies like Tesla, Rivian, Lucid, GM, ABB, and Siemens. Jesse currently works at Advanced Charging Technologies in the research and development / test engineering department

Vincent Martinez is a 26 year-old Electrical engineering student who will be graduating from University of Central Florida. He wishes to increase his knowledge by becoming a full time electrical engineer in board design development in control systems, robotics and more. places of interest would be Lockheed Martin, Siemens.

Allan Nevalga is a 26yr old Computer Engineering student. Allan is pursuing a career in software engineering for a Financial Tech firm this coming Spring. Allan was a former Research and Development Engineer for a start-up company last year delving in Hardware Integrations.