Project Apricorn
Interactive Pokéball with Radio Frequency Identification

Rebecca Casimir, Kalindi Desai, Trey Sandefur, and Darren Yong

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

***Abstract* — Project Apricorn is defined by the interactive experience created between the user and a Pokéball that utilizes radio frequency identification (RFID) to communicate between the user and system. Orlando is known for being the home of many themed entertainment parks and having close ties with the University of Central Florida, Project Apricorn is designed to be an extension to the interactive experience. The Pokéball will feature an integrated printed circuit board with a RFID transceiver which will relay information to the base station which will be connected to a database to determine the user's identity to access interactive software selection.**

***Index Terms* — interactive, interface software, microcontroller, signal detection, wireless communication.**

I. Introduction

Project Apricorn explores how the guest experience can be improved in themed entertainment environments by increasing active immersion, and improving interactivity through the use of more advanced technology. The land that Apricorn lives within is based in the world of Pokémon [3], the highest-grossing media franchise of all time. The core of the guest experience lies in the use of handheld radio frequency identification assemblies, modeled after the iconic Pokéball from the world of Pokémon. In this land, you become a trainer, and get to gain your own Pokémon, and battle with them just as it's shown in the video games, movies, television programs, and other media. Your story begins at the Pokécenter (#1 on Fig. 1.), where you can gain your first Pokéball and your first Pokémon. In the Pokécenter, the guests are walked through an experience that is modeled after the source material.. As guests begin to be immersed within their own Pokémon story, the embedded RFID printed circuit board (PCB) begins an initialization protocol that connects to a system-wide MySQL database. Once this experience is done, the guests can travel throughout the land and begin battling with their own Pokémon (#2 on Fig. 1.), and write their own Pokémon story. Apricorn serves as a proof-of-concept project on how a system wide deployment could be approached. Described below is a systematic approach to designing, engineering, and implementing Project Apricorn's Pokéball PCB, base station, and MySQL database battle experience. .

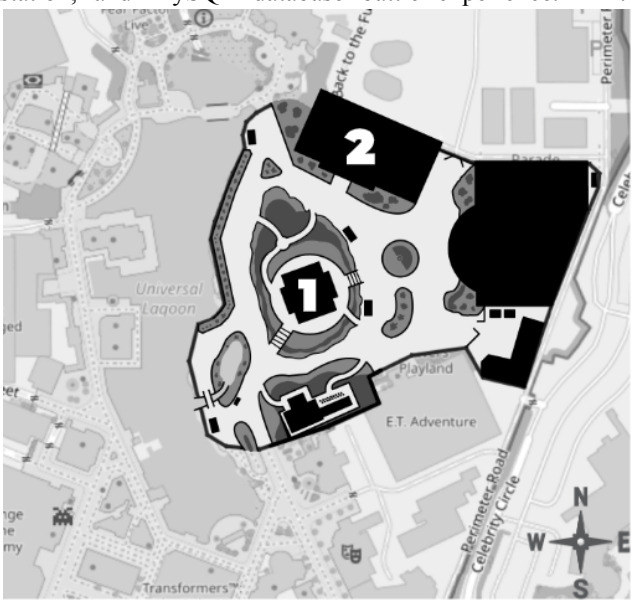

Fig. 1. Ideation of a Pokémon themed land at the Universal Studios Orlando theme park.

II. Hardware Components

The hardware components consist of two major entities, the handheld Pokéball assembly and base station. The base station is a home beacon for all handheld devices to communicate with throughout the land. The Pokéball's core comes in the form of an RFID transceiver to transmit and receive data to-and-from the base station. Fig. X. outlines the overall block diagram of Project Apricorn. The major hardware components are further explored and detailed in subsections below. .

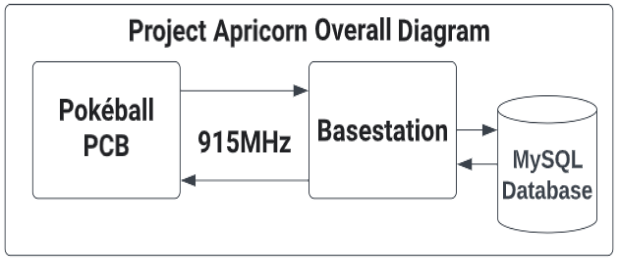

Fig. 2. Project Apricorn Overall Block Diagram

A. Pokéball Shell

This project's foremost focus is on the guest interaction occurring via the handheld RFID transceiver assembly. This required necessary housing for the assembly to reside within to provide adequate protection for the sensitive electronic components. The group elected to 3D print this housing utilizing Polycarbonate (PC) filament via printers provided by AOA at their Winter Park Headquarters. The housing itself is modeled after the famous standard Pokéball found in the world of Pokémon. The assembly was modeled within the AutoDesk Fusion360 modeling software. Fusion360 provided the robust modeling tools and testing environments that the group needed to ensure that the assembly was adequately designed prior to moving into the 3D printing phase.

The 3D printed assembly comes in the form of 6 (six) major articles:

TABLE I. Pokéball housing article descriptions

Article	Description
(i) Lower Interior Shell	Foundational piece of the assembly that directly connects the PCB to its enclosure and provides rigid support to the internal electronics
(ii) Upper Interior Shell	Upper connecting piece of the PCB enclosure that connects articles (ii) and (v). Provides total protection to the internal electronic components
(iii) Lower Exterior Shell	White colored exterior decorative piece that molds to article (ii) to resemble iconic shape of the standard Pokéball
(iv) Upper Exterior Shell	Red colored exterior decorative piece that molds to article (i) to resemble the iconic shape of the standard Pokéball
(v) Locking Rods	Cylindrical rods that connect articles (i) and (ii). Holds down PCB between articles (i) and (ii) via non-conductive padding material
(vi) Ornate Button	Non-functioning white decorative piece

All 6 (six) articles come together via the use of a JB Weld bonding agent to provide sufficient support and rigidity to the total assembly for the sake of the presentation demo. The articles were sanded, primed, painted, and coated to stay as true to the source material as possible.

Fig. 3. Pokéball Shell Designed and Rendered in Fusion360

B. Pokéball Printed Circuit Board

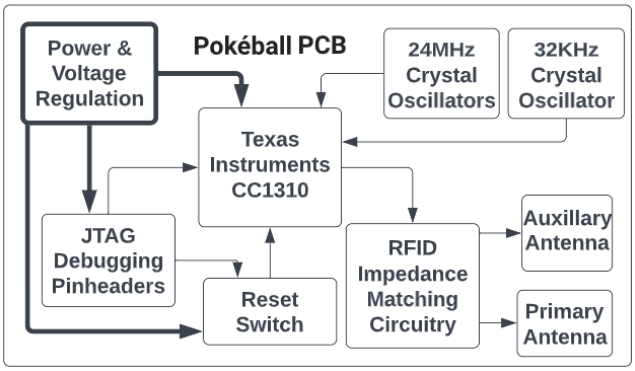

Fig. 4. Pokéball PCB Block Diagram

Our printed circuit board (PCB) features 7 (seven) major blocks:

(I) Power Circuitry

(II) Reset Switching Circuit

(III) JTAG Pin Header Circuitry

(IV) 24MHz Crystal Oscillator Circuitry

(V) 32KHz Crystal Oscillator Circuitry

(VI) MicroController Circuitry

(VII) 915MHz Antennas & Impedance Matching

I. Power & Voltage Regulation

The main purpose of the power circuitry is to provide voltage regulation, electrostatic discharge (ESD) protection, and filtering of unwanted noise in the system. The power is provided via a single 3.0V Lithium-Cobalt Oxide coin cell battery. This 3.0V is then boosted via a TPS61291 Low Iq Boost Converter manufactured by Texas Instruments. This boosts our voltage to 3.3V and provides power to all subsystems within our PCB. This component was selected due to its remarkable efficiency of both boosting and regulating the output voltage under extreme circumstances.

TABLE II. Voltage regulator efficiency

V_{in}	V_{out}	I_{out}	Efficiency
1.2V	3.3V	10mA	85%
1.8V	3.3V	10mA	90%
2.5V	3.3V	10mA	94%
3.0V	3.3V	10mA	96%
1.2V	3.3V	100mA	85%
1.8V	3.3V	100mA	91%
2.5V	3.3V	100mA	94%
3.0V	3.3V	100mA	95%

A 1SS315TPH3F RF Schottky Diode is utilized in series to the coin cell battery holder to provide ESD protection in the chance that our 3.0V LiCB battery is inserted with incorrect polarities. The orientation chosen will protect all other components of our PCB from otherwise harmful ESD. The 1SS315 is rated at 5V, and 30mA, which provides ample protection for this project. The final major component of the power circuitry is a BLM18HE Ferrite bead that suppresses high frequency noise from within our power circuitry prior to powering the remaining systems of our PCB.

II. Reset Switch

The reset switching circuit contains a 2200pF and 47.5kΩ resistor for debouncing of the parallel single-pole single-throw reset button. This circuitry features the same ESD protection Schottky Diode that is featured in the above *Power Circuitry* section.

III. JTAG Pin Headers

The JTAG Pin Header circuitry consists of 2x5 rows and columns pin headers that connect to the JTAG_TCKC and JTAG_TMSC pins of our CC1310 microcontroller. The JTAG specific pins feature two ESD diodes as featured in other parts of our PCB to protect our circuitry. One of the pin headers is utilized as a bypass of a 3V3 connection to the main circuitry of our PCB.

IV. 24MHz Crystal Oscillator

The CC1310 requires two external crystal oscillator circuits for use of all components within the microcontroller. The load capacitors for the 24MHz crystals was calculated by the following equation:

$$C_X = 2(C_L - C_{stray}) \qquad (1)$$

The above equation (1) provided us with the proper process to find the values needed for our load capacitance. The variables in the equation were provided in the specific data sheet of the 24MHz crystal. The calculated value returned was 9pF. The max clock operation of the microcontroller within the PCB is 48MHz, which is utilized by the 24MHz crystal and doubled internally.

V. 32KHz Crystal Oscillator

The same process and equation (1) were used to calculate the values of the 32KHz crystal oscillator circuitry for our PCB. After gathering variables from our 32KHz datasheet, the final calculated value of the load capacitance was 12pF for this circuitry.

VI. MicroController

The chosen microcontroller unit for this project was the Texas Instruments CC1310 - Sub-1GHz microcontroller unit [5]. The CC1310 is a remarkably robust chip from the Texas Instruments line of RF devices that serve as both a transceiver, as well as a microcontroller processing unit. The package that was chosen was the 48-pin F32RGZR. The major distinction with this package sizing is its FLASH memory (in kB) and pin count. The smallest flash size of 32kB was chosen due to the necessary memory sizing the project required. This choice also minimized cost, as the major price points for this line of controllers comes in its memory sizing.

The MCU is based off of a 16-bit ARM© Cortex©-M3 architecture, and is capable of operating in a frequency range of:

TABLE III: CC1310 operating frequency bands

MINIMUM	MAXIMUM	UNIT
287	351	MHz
359	439	
431	527	
718	878	
861	1054	

The CC1310 has a typical operating voltage range of 1.8-3.8V, with absolute maximum ratings in the range of -0.3-4.1V. The active mode average current consumption ranges from 2.0-6.0mA depending on the crystal oscillator used, transmitting or receiving active, and other peripheral components in use. The utilization in this project ranges from 3.3-3.6mA from readings taken. Added benefits of the CC1310 come in the form of its numerous modules embedded onto the chip. The above mentioned features can be found in many other MCUs, but few off-the-shelf MCUs provide an embedded RF core as robust as the CC1310. The RF core contains two (2) analog-to-digital converters, Digital PLL, DSP Modem, separate storage elements from the overall MCUs architecture, and its own ARM© Cortex©-M0 processor as well [5].

The remaining portions of the overall microcontroller circuitry comes in the form of various bypass capacitors near each power input of our CC1310. Each voltage supply pin was provided with a 0.1µF bypass capacitor, as well as a 1.0µF bypass capacitor in close proximity to the MCU at the output of the power circuitry described above in (*I*).

VII. 915MHz Antennas & Impedance Matching

The most important aspect of the Pokéball's PCB lies in the RF portion of our schematic. This portion, or "block", consists of the two antennas and the RF impedance matching circuit. This project utilizes only one antenna that is embedded directly into the PCBs two copper layers, but a secondary SMA connected antenna can be utilized in the emergency case of damage being caused to the embedded antenna. The primary embedded antenna is a helical 50-Ohm antenna that operates with highest efficiency in the 915-920MHz range. The nature of this antenna required that an impedance matching circuit be tied directly to the RF_N and RF_P pins of the CC1310 MCU to allow for proper communication to be achieved. Texas Instruments provides application information in matching circuit layouts in the CC1310 datasheet, of which we utilized the "Differential Operation" mode. The use of differential mode provides a higher noise immunity, and overall less-lossy signals in both transmission and reception.

The impedance matching circuit, also known as a matching network, is a combination of active components (capacitors and inductors) in specific configurations that "match" impedances between the pin of the MCU and the 915MHz antenna. Utilizing a matching network ensures that minimal power losses occur within the load. Texas Instruments specified that a matched impedance of Z = 44+j15Ω. This impedance value, combined with the Differential Operation mode guidelines given by Texas Instruments inside of the CC1310 datasheet provided enough information to create a successful RF Impedance matching network.

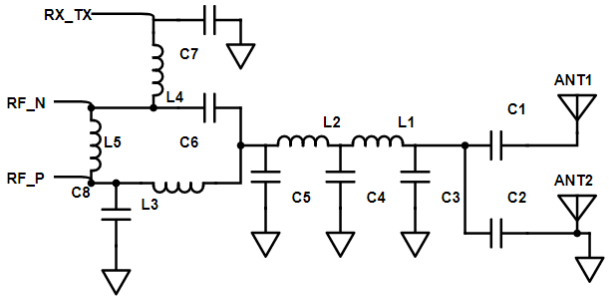

Fig. 5. Differential Mode Impedance Matching Network

In the above section *Power & Voltage Regulation* is discussed the operating voltage of 3.3V selected for this PCB. This was chosen not only due to required power within the overall PCB, but also in deciding the necessary transmitting power to properly communicate with the base station. The CC1310 at 915MHz reaches a maximum transmitting output power of 14.25dBm at a supply voltage (VDDS) of 3.3V. The use case of Project Apricorn utilizes 3.3V at approximately 17.4mA, providing 0.057W of power, or -12.41dBm of transmitting power. The sensitivity of the receiver in its tested state by Texas Instruments was -110dBm. It is expected that this value would be a few magnitudes higher due to the nature of the impedance matching circuit components, 3D printed housing, and other unaccounted for losses.

C. Base Station

The base station will consist of a printed circuit board that will host three main components. The first component is the RFID transceiver, RFM95W by HopeRF. The second component is the ATMEGA328P-AU microcontroller. The third component is the USB to TTL serial converter which allows the microcontroller to be connected using an USB connection to allow

programming to the microcontroller. The software used to program the microcontroller will be the Arduino IDE.

I. RFM95W RFID Transceiver

The RFM95W is a LoRa module that allows long range radio frequency communication which will be 915 MHz in this case. This module is able to achieve a sensitivity of -140 dBm and an integrated +20 dBMm power amplifier allows this module to be perfect for our low cost method of communication. The small size of this module allows to reduce the overall size of the base station. The RFM95W will feature a breakout board that allows a U.FL antenna to be connected to feature a communication range of over 100 meters.

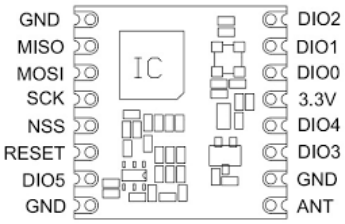

Fig 6: RFM95W Transceiver

II. ATMEGA328P-AU Microcontroller

The ATMEGA328P-AU is a surface mount AVR series microcontroller. This microcontroller features 8 MHz clock frequency and an onboard flash memory of 32 KB. The microcontroller will be powered by 3.3V and is programmed with a bootloader allowing the user to program the unit with the Arduino IDE. The board allows connection to a FTDI board which is described in the next section. This unit also features a small form factor which allows the size of the base station to be reduced in size.

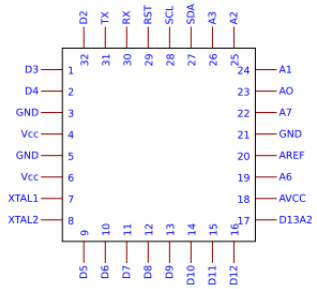

Fig 7: ATMEGA32P-AU Microcontroller

III. FT232RL USB to TTL Serial Converter

The FT232RL is a breakout board that allows the FTDI connection from the microcontroller to be converted from a USB to allow programming from a PC. The FR232RL features a small factor breakout board that converts the processes the signal from USB to FTDI for the ATMEGA328P-AU microcontroller to accept.

III. Software Detail

A. Application Stack Specifics

The system's application will be utilizing a Linux operating system, Apache HTTP server, MySQL database, and PHP programming for API creation, commonly known as a LAMP stack. The LAMP stack is preferable to other options as it is known for being reliable, fast, and secure. As the Pokeball was designed around the concept of being utilized in a Pokemon-themed amusement park, the LAMP stack would be most beneficial. It is highly scalable and requires low maintenance and will be able to maintain all of its core features necessary for the size of an amusement park. The application and framework for the LAMP stack is deployed by DigitalOcean.

B. Battle Sequence

One of the major requirements for this project's success is a successful battle sequence. The battle sequence will only commence after the user has successfully logged in and registered their Pokeball. As this project is based around the concept of a Pokemon-themed amusement park, one of the main features of the project is for the user to roam around the park and "catch" a Pokemon. After the Pokemon is "caught", it will be stored in the Pokeball ready for battle. With this concept in mind, the limitations of the current system do not allow for roaming and "catching" Pokemons therefore the Pokeball will demonstrate its capabilities of detecting a Pokemon and then storing it.

Once the user is ready to battle their Pokemon, they will go to the application and begin the battle sequence. The battle will show a graphic of the user's Pokemon and the Pokemon the program decided to be the best opponent based on the user's Pokemon's score and type. The program will call the four moves associated with the user's Pokemon and display each possible move on screen. The user will be allowed to choose from any of the four moves and then see how the opponent counter attacks. The four moves are distinguished by different strengths. Once the user or opponent has made their move, the program will automatically update their scores. After one of the Pokemons have lost all of their health, the application will designate a winner.

C. Database Specifics

In compliance with the LAMP stack, the system's vital data is stored in a MySQL database. This database will be

called “Apricorn”. The project requires three tables for the best result for the system. The system implements a Users table, Pokemon table, and Pokeball table.

The Users table stores seven fields: the ID, DateCreated, DateLastLoggedIn, FirstName, LastName, Login, and Password. This is the main table connected to the application as it verifies the user exists and will connect to the Pokeball.

In the figure below, an example of the Users table is shown. In this table, due to the size constraints, the field names have been simplified but still follows the fields previously mentioned. The SQL command to call all of the Users was called for the figure but for demonstration purposes, only two test users are shown below.

SELECT * FROM Users;

ID	DateC	DLLI	FN	LN	Login	Pass
**	22-11-08 02:29:20	22-11-09 23:34:24	Joh n	Smit h	JS10 1	*** ***
**	22-11-08 01:09:32	22-11-09 17:28:37	Ana	Smit h	AD1 5	*** ***

Fig. 8. Example of MySQL “Users” table in Apricorn Database.

The Pokemon table stores ten fields: PokemonID, PokemonName, PokemonType, BasePokemonScore, WeaknessType, ResistanceType, Move1, Move2, Move3, and Move4. The PokemonID and PokemonName are unique to each Pokemon. The PokemonType and BasePokemonScore standardize the Pokemon and set a basis of which the application can understand what type of Pokemon it is. All of the fields, except the PokemonID, stays true to the Pokemon industry and was found on their Pokedex, their database of all Pokemons [3]. The Pokedex characterizes each Pokemon and remains consistent in all of their products. As this project is modeled around a Pokemon-themed amusement park, the data for the Pokemons stays true to the franchise.

In the figure below, an example of the Pokemons table is shown. The original Pokemons table has been split into two separate tables for ease of understanding. Due to the separation, the ID field remains on both tables to understand which Pokemon the data correlates to. The SQL command to call all of the Pokemons was used for the figure but for demonstration purposes, only the first three Pokemons are shown below.

SELECT * FROM Pokemons;

ID	Name	Type	Score	WT	RT
1	Bulbasaur	Grass, Poison	318	Water, Electric, Grass, Fighting	Fire, Ice, Flying, Psychic
2	Charmander	Fire	309	Fire, Grass, Ice, Bug	Water, Ground, Rock
3	Squirtle	Water	314	Fire, Water, Ice, Steel	Electric, Grass

ID	Move 1	Move 2	Move 3	Move 4
1	Vine Whip	Poison Powder	Synthesis	Solar Beam
2	Ember	Dragon Breath	Flamethrower	Flare Blitz
3	Water Gun	Water Pulse	Rain Dance	Hydro Pump

Fig. 9. Example of MySQL “Pokemons” table in Apricorn Database.

The Pokeball table stores six fields: PokeballID, isEmpty, UsersID, PokeballScore, PokemonScore, and PokemonID. The UsersID, PokemonID, and PokemonScore fields are all foreign keys that are connected to the Users table and the Pokemon table. Each Pokeball is only allowed to hold one Pokemon at a time therefore the isEmpty field allows the program to check if the Pokeball is allowed to store a Pokemon. The PokeballScore is unique to each Pokeball and the user connected as it designates the user’s progress in the amusement park. This field was created with scalability in mind as the project is not currently concerned with competition between users.

In the figure below, an example of the Pokeballs table is shown below. The two entries shown below are of two options for the Pokeballs: empty and filled. One Pokeball is shown with a Pokemon stored within and the associated

PokemonID from the Pokemon table, which designates that the Pokemon, Bulbasaur, is stored. The second Pokeball shows that the isEmpty field has a value of 1 therefore the Pokeball is empty and has no Pokemon stored.

SELECT * FROM Pokeballs;

ID	isEmpty	Users ID	Pokeball Score	Pokemon Score	Pokemon ID
**	0x00	**	100	318	1
**	0x01	**	0	0	NULL

Fig. 10. Example of MySQL "Pokeballs" table in Apricorn Database.

The Users table and Pokeball table are both growing tables as more users use the product. The application will pull and post data to both tables. After the Pokemon table was created, it is purely used for pulling data from and does not require more data to be added unless more Pokemons are to be added manually.

D. Frontend Specifics

As mentioned earlier, due to the nature of the LAMP stack, the user interface has been coded in Javascript, HTML, and CSS.

From a UI perspective, the right combination of Javascript and CSS can allow for a seamless and intuitive program that allows moving objects without having to use too many API endpoints. Instead of each action of the Pokémon battle and each move within a battle being an endpoint, the javascript would pull the points associated from each move and tally the final score. This saves the programming from crashing. This would also mean that a great part of the program relies on Javascript to perform the necessary actions (as most ideal applications do).

In the case of CSS, modern improvements to the language have allowed the support of: PNG, SVG, JPEG, GIF, audio, and almost any commonly used media file that one may commonly see in a given platform. Any media movement seen within our application may just be the result of the right combination of CSS elements and variables. Smooth movement would just have to depend on the (1) Framework, (2) Network connectivity, and (3) other files and/or dependencies living within that existing framework.

E. Arduino to MySQL

As the base station has been previously discussed in section II.C, Arduino software and hardware has the ability to update directly into a user interface platform that supports a database. Originally, the idea was to stick with TI Instruments to have a uniform brand of technologies (as this team has operated with TI through most of the coursework), however; it came to show that Code Composer Studio (CCS) does not support any form of MySQL (or any other) database connectivity. If TI instruments wanted to be used, there would have to be countless files of code written in C++ to even attempt any form of connection. This scenario would not even guarantee a seamless transition between the base-station and the User Interface so this obstacle was avoided altogether.

Ultimately, connection from an Arduino code to a MySQL is a lot more simpler than what it sounds like. All that is needed is an Arduino integrated development environment, and an existing MySQL database (preferably one that is supported by HTTP, for security concerns). There is an option of connecting to a database ip address, similar to the free MySQL community database or the HTTP option where there exists a framework ip address in which a MySQL database lives such as the one in this project.

Code for connectivity may will look like:

```
int    HTTP_PORT   = 80;

String HTTP_METHOD = "GET";

char   HOST_NAME[] = "192.108.0.17";
String PATH_NAME   = "/PokeId.php";
```

Fig. 11. Example of Arduino code to Mysql connectivity in Apricorn Database.

Here it is shown that the location has been identified as well as the endpoint that will be performing the insertion of the RFID into the user interface.

Additionally, the API endpoint on the user interface side will look like:

```
<?
$rfid = $_GET['id'];
$other = $_GET['pokemon'];
?>
```

Fig. 12. Example of API (PHP) endpoint code.

This endpoint is responsible for selecting the Pokemon from the databases mentioned earlier that will then be assigned to said player.

F. Final Software Flow

Being that this is meant to support a commercial and entertainment environment, the program allows for any party of interest to register for an account. In order to have any interaction with a Pokémon, a user must be in possession of a Pokéball. That Pokéball contains the PokéID necessary to discover which Pokémon has been assigned to that player as well as enter battle. After any sort of playtime and battle (to where a score can officially be officially updated), it will then update the leaderboard, which is accessible for all players.

VII. Conclusion

All of the mentioned components and systems have been developed in order to ensure successful implementation of the Pokéball scheme. While different approaches and changes have been made, they have shown to provide a learning experience for the assembly of a design of this scale. Furthermore, if worked upon in the future, there are aspects that can be improved on to increase efficiency: connectivity, frequency emitting and receiving responses, as well as software playtime.

Biography

Rebecca Casimir is currently a senior attending the University of Central Florida. Her intended degree is a Bachelor's of Science in Computer Engineering. As of today she is currently interning at Walt Disney Studios in cyber security and she plans on continuing to transition to full time after graduating in Fall 2022.

Kalindi Desai is currently a senior attending the University of Central Florida. Her intended degree is a Bachelor's of Science in Computer Engineering. As of today, she is currently interning with Ximple Solutions, a company that designs and develops cloud-based ERP software. She plans on working with Ximple full time after graduating in Fall 2022.

Trey Sandefur is a senior at the University of Central Florida. He will be finishing his Bachelor's of Science in Electrical Engineering at the conclusion of the Fall 2022 semester. He will be transitioning from an Electrical Engineering Intern position at AOA into a full-time Associate Show Producer position. AOA serves as a themed entertainment production firm that has helped bring world-class attractions, resorts, and experiences to life.

Darren Yong is currently a senior at the University of Central Florida. He will receive his Bachelor's of Science in Electrical Engineering in December of 2022.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr.s Samuel Richie and Lei Wei; University of Central Florida.

A sincere gratitude is extended to all of the professors and mentors who have been able to guide the authors through coursework and projects. If not for learning these essential concepts, this design would not be possible.

Lastly, the authors would like to thank their family, peers, and close ones who have supported them throughout their college careers.

References

[1] W. H. Cantrell, "Tuning analysis for the high-*Q* class-E power amplifier," *IEEE Trans. Microwave Theory & Tech.,* vol. 48, no. 12, pp. 2397-2402, December 2000.

[2] Neaman, Donald A, "Microelectronics Circuit Analysis and Design" *2009 McGraw-Hill Education*.

[3] "The Official Pokémon Website." Pokemon.com, https://www.pokemon.com/us/pokedex/.

[4] "Texas Instruments" ti.com, *https://e2e.ti.com/support*

[5] Texas Instruments, "CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU", SWRS181D datasheet, Jul. 2018.