Audio Manipulation Box (A.M.B.)

Nathan Cherry, Elian Seda-Figueroa, Mark Fritz, and John Sinanis

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

***Abstract* — Audio manipulation in the form of stem splitting; facilitated in a box refined for space and cooling. The Audio Manipulation Box is designed around the functionality of adjusting the different noises within a song to either amplify or reduce its impact on the track - whether that be for learning, studying or teaching. The internal communicative components regard a BeagleBone AI64 - denoting a 64 bit hierarchy - and an ESP32 which work together to transfer an audio file (.mp3), split it into parts and reconstruct it for the presented purpose. Spleeter is the core separation process utilized. The design was meant to be spatially efficient, user friendly and conducive to an environment which promotes learning.**

***Index Terms* — Machine Learning, Spleeter, Audio Processing, Tensorflow, Voltage Regulator, 64-bit architecture, Bluetooth**

I. Introduction to AMB

One of our group members, Nathan Cherry, an aspiring engineer and passionate dance instructor, found himself facing a unique challenge during one of his dance classes. One of his students struggled to hear a particular part of a song, hindering their ability to learn the dance routine effectively. Determined to find a solution, Nathan was inspired to combine his love for music and his engineering abilities to create a product that could isolate specific stems of a song. Recognizing the potential to aid the music and dance education market, our group embarked on a journey to develop a device that would allow instructors to emphasize and isolate certain elements of a song, making it easier for students to follow along and enhance their learning experience. With our innovative mindset and dedication to improving music and dance education, our hopes to make a significant impact in the industry.

II. Current Market

Currently, the market for products that isolate specific stems of a song for dance instruction is relatively limited. One notable example is Kanye West's Donda Stem Player, which aimed to provide users with the ability to isolate individual elements of a song. However, the product faced criticism for its subpar user interface and limited supply, which hindered its widespread adoption.

Additionally, there are desktop computer programs available that offer similar functionality, particularly in the field of DJing. These programs allow DJs to manipulate and isolate stems of songs for mixing purposes. While these programs are effective, they lack the portability required for dance instructors who often need to teach in various locations.

In light of these limitations, AMB aims to fill a gap in the market by designing a portable device specifically tailored for dance instruction. By focusing on user-friendly features and ensuring a seamless user experience, the product intends to provide dance instructors and students with an intuitive and efficient tool for isolating and emphasizing specific song stems. The goal is to create a product that is highly portable, allowing instructors to carry it easily between different teaching environments. By addressing these key aspects, AMB hopes to offer a unique and valuable solution that caters specifically to the needs of the music and dance education market.

III. Core Component Selection

To best understand the system we should look at the current components that contribute to the grand scheme. This consists mostly of components purchased through a manufacturer due to the need of higher computational power and the time constraint of the project. However, there are a few components within the system that were designed and manufactured by our team members. This section will endeavor to explain the characteristics of each component and how it was realized within our circuit.

A. Main Processor

To introduce the most variable choice component in our system, our main processor has received at least 4 iterations due to the ultimate needs and obstacles we have faced. Ultimately we decided to go with the BeagleBone AI64, where - within BeagleBone - we had 2 alternative choices for which would trump as our main. The AI64 was the most attractive, despite the others being more than capable of functioning within our system. This specific board boasts a much needed 64 bit architecture. There were others that could have been applicable here, but due to the imposed constraints we had to choose the most feasible option. The reason the AI64 is our main choice here revolves around the architecture but also the accessibility of supported operating systems, which include Linux, QNX and RTOS - if that was the route we decided for this project. Of all our options, this is the most expensive one, but it came with all the accessibility necessary to achieve our goal, which concurrently necessitates AI and machine learning. All of this is capable with the TI Jacinto TDA4VM processor that comes on board.

B. Display

The LCD display incorporated in our design is the RC1602B-BIW-CSV. This part receives a 5V DC input to power its screen. We interact with the LCD by I2C communication. The display will allow the user to interact with the AMB through 2 lines by 16 characters displayed on the LCD. Furthermore, the backlight is adjustable. It's also notable that the display has a 16-pin interface, which has more accessibility than the previous options. Ultimately, this display choice was the most attractive option because LCD displays have higher resolution, lower readability, a high refresh rate and are less expensive.

C. Bluetooth

This system contains a bluetooth module to transfer data between the associated paths of engagement. It was more logical to go with bluetooth because the audio files we are sending are not meant to be egregiously large and a common goal within this practice was to maintain power efficiency, which is where bluetooth succeeds. The ESP32 offers BLE and Bluetooth classic. We are settling for less security, due to less security protocols in Bluetooth than WiFi. However, this being considered, the only pertinent information would be the audio file transfers and some basic app interface. The app interface is facilitated by Beaglebone and would not be directly affected by this, so our only problem would concern the consistency of data transfer. The ESP32 also offers several forms of communication, due to the package coming with an integrated microcontroller as opposed to its competitors. These include SDIO, UART, I2C and SPI - where we use UART for our information based transfer and SDIO to fully integrate the part itself.

D. Battery

As the needs for the system became more prevalent, the charging characteristics also shifted. We started with a Nickel Metal Hydride topology but had to reconstruct the circuit needs. The circuit became more complicated with the newfound obstacles. This necessitated several components that will be expanded upon more thoroughly in the sections that follow. Upon further research we settled on a Lithium Ion battery single cell 3.7V with a 9900mAh discharge capacity. This battery is called the Samsung INR21700-50E. Using Lithium Ion in this system is more beneficial as the surrounding parts are generally smaller, more thermally efficient and took up less space on our board. The only downside was the concurrent additions necessary to achieve the values needed to operate properly.

IV. Charging Circuit Components

The charging system was its own board at one point. Due to several changes and iterations, we combined the main board and the charging circuit into one board. Effectively, this would save space and add more portability. We also had to increase our circuit protection and further modified the circuit to achieve the types of values required to run the processes from the BeagleBone AI64 and the ESP32.

A. Charge Controller

The charge controller used to maintain an effective charge rate for our Lithium Ion battery is the TI BQ25895RTWR. For our purpose, the charge efficiency would sit at 91% given the output is 3A. If we look at the BoostMode operation with a 4.5V to 5.5V output, we would achieve a 3.1A output. There is an integrated control to use either of these properties. Due to the multiple interactive parts within this part of the system, the circuit protective elements specific to this charge controller are attractive. There is battery temperature sensing for charge and boost modes. This controller is also set for thermal regulation using a thermistor which would allow the controller to shutdown if necessary. This component utilizes communication through I2C or USB.

B. Overcharge/Battery Protection

Our battery protective element here is the TI BQ29700DSER. This battery protective element is of package size 6-WFDF. This BQ29700 is capable of an array of circuit protection which include overcharge detection, over-discharge detection, charge overcurrent detection, discharge overcurrent detection and load short-circuit detection. If our battery was depleted, we would have access to zero voltage charging. This portion of the circuit has two power MOSFETS which are meant to handle high charge or discharge currents.

C. Voltage/Current Boosting

For our BeagleBone AI64, we had to allocate a specific amount of power to run properly. This included an array of choices, but ultimately chose the TPS61235PRWLR. It has all of the qualities necessary to achieve the output called for by the BeagleBone AI 64. The input is from the battery, which is being boosted from 3.7V to 5.1V. The boasted amperage output is up to 3.5A but would realistically sit between 2.9A and 3.1A. The TPS offers an output current monitor. The only downside to using this component is it's considered a switching regulator, which has a frequency of 1MHz in the TPS. This will create more noise than the alternative voltage regulator. The other downside is the amount of internal parts necessary to allow this concept to happen.

V. System Concept

To understand the complete system, it would help to realize a short diagram. There are more intricate systems within but the processes can be displayed as follows:

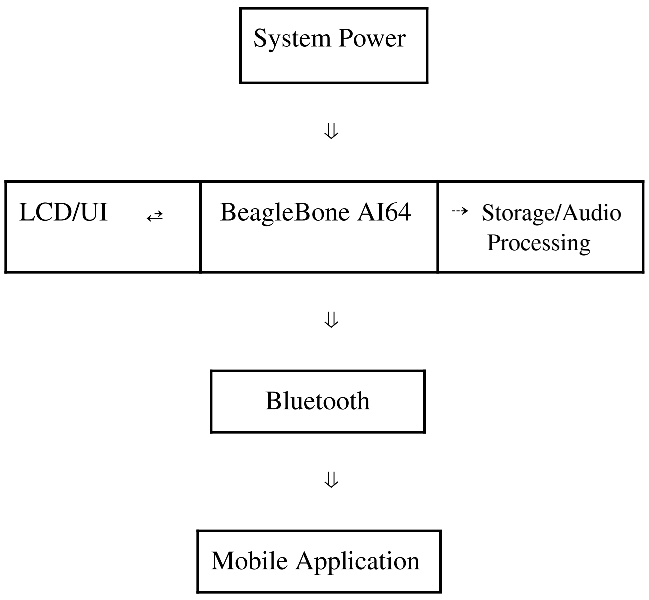

Fig. 1. Simplified system flowchart

The user turns on the device with a switch. This will enable the LCD screen, the ESP32, and the BeagleBone to turn on. After turning on the device, the user can connect to AMB from the application via bluetooth classic. In addition, the user can use the application to connect to a bluetooth speaker (must include BLE Audio). Navigating through the application or by using the LCD screen, the user can play their favorite songs with varying values of stem separation. This could mean keeping the vocal track completely in the song, removing the vocals completely, or keeping the vocals - but at a significantly lower volume relative to the rest of the track. After the user begins playing the song, the stem separation values lock. The user must pause the song before adjusting the stem separation values again. Lastly, if the user would like to add or remove songs from AMB they can use the application to do so or additionally remove the SD card and upload the songs directly onto the device. Uploading a song will require a delay so the device can compute the separation before being played. Therefore, the songs will be stored on the device to quickly play songs that have already had the stem separation completed.

A. Hardware System Concept

The hardware components of this project are represented in the following diagram:

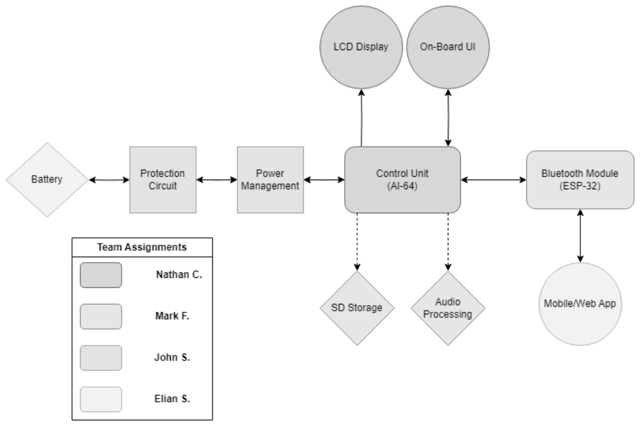

Fig. 2. Specified Hardware system flowchart

The main power source for AMB is the battery. AMB uses a 18650 Lithium-Ion battery that provides 3.7 VDC and is capable of safely providing a steady 7A. The particular 18650 battery we chose has a discharge capacity of 9900 mAh. To provide 5VDC to the BeagleBone and the ESP32 for power, a TPS61235PRWLR boost converter is utilized. This increases the voltage from the battery from 3.7 VDC to 5.1 VDC. In addition, there is a battery protection circuit. This helps protect the battery against reverse voltage and further prevents damaging the

BeagleBone AI64, which is expensive to source. Lastly, there is a charging circuit. The charging circuit pulls power from the USB-C connector on the LCD board. The USB-C connection can provide 5VDC and 3A to this circuit. This circuit will charge the rechargeable batteries to prevent the need of replacing the battery after it dies and offers a fast charging standard to allow for easier use. The LCD board also acts as our onboard user interface (UI) in case they do not wish to use our mobile app. There is an ON-OFF switch to the left of the LCD screen that will enable power to run to the BeagleBone and ESP32. As power is being applied to the board, a LED will turn on showing that AMB is on.

B. Software System Concept

The software components of this project are demonstrated in the following diagram:

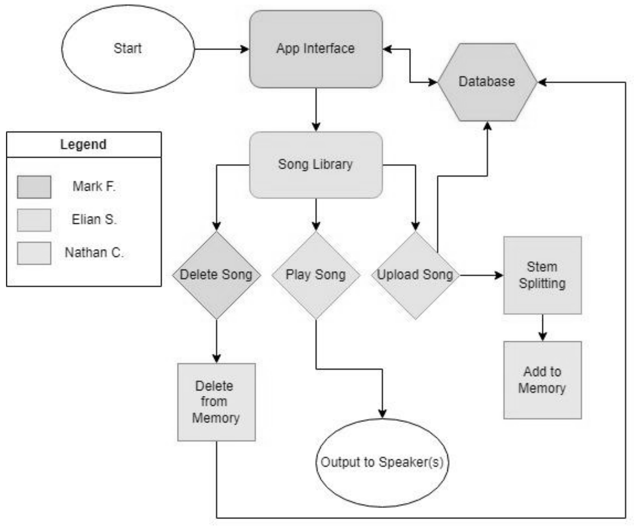

Fig. 3. Specified Software system flowchart; delete, play and upload featuring stem splitting

The user interface would be the start of this system. The mobile user interface will allow the user to start the processes concurrent to stem separation. Once the bluetooth from the AMB is connected to the chosen device, the application will allow you to add, delete or play a song that will be accessed from the library. When you upload a song to the app, it will be stored in the database located in the BeagleBone. The selected song will be separated and can be replayed through the speaker. The application will facilitate the operations involving stem splitting. In the app, we can choose individual portions that will be filtered and modulated. This will adjust the levels of the sound to either exclude or reduce that sound. We can go further and isolate that sound by reducing the surrounding sounds. This core functionality is ran by Spleeter, where the intensive processes will be handled by the BeagleBone AI-64s architecture.

VI. Hardware Detail

We can go into further detail of the main system components and how they interact with one another. The hierarchy, in conjunction with the assembly, is dependent on the figures laid out and ultimately the foundation for our circuits.

A. BeagleBone AI-64

The BeagleBone AI-64 is our main processor. All the design ultimately revolved around this platform. It is a powerful board with many avenues of functionality.

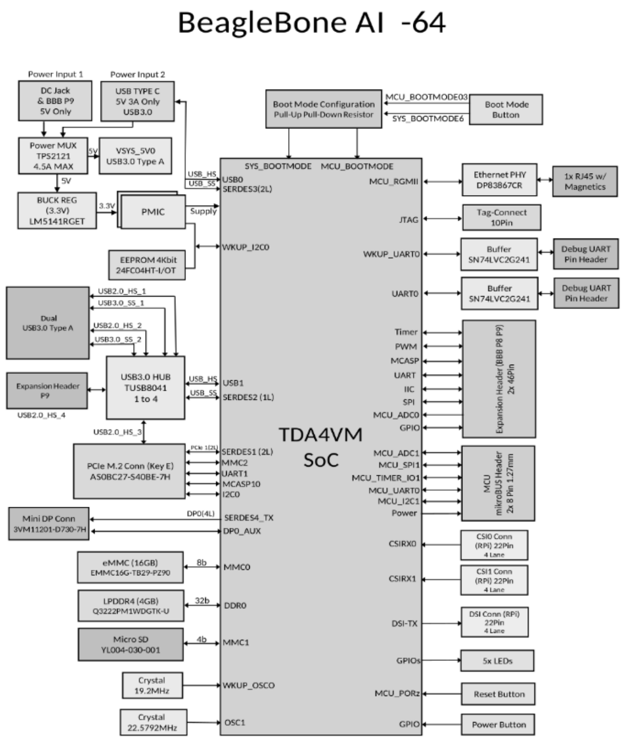

Fig. 4. BeagleBone AI64 Core Components

This board has several key characteristics that were specifically attractive for this application which include:

(1) TI TDA4VM with dual Arm Cortex-A72 (64-bit) arm CPU; not particularly common for these board types. Meant for machine learning.

(2) 4GB SDRAM which gives us plenty of clearance to do the core processes can be quite intensive while doing background operations such as playing while splitting a song.
(3) Two PMICs and one additional LDO to provide the different power rails; TPS65941213 and TPS65941111 for additional board safety
(4) Up to 2 Gb dual-core RTU and Industrial Communication Subsystems
(5) Deep-learning MMA at 1.0 GHz
(6) Dedicated hardware redundancy check blocks (MCRC)
(7) Secure Boot Management/storage support

These unique characteristics applied to audio storage and the operation of Spleeter to house these operations. It was absolutely necessary that a 64-bit architecture was used to work with TensorFlow. These processes coincide with UART_RX and UART_TX on the main board. The BeagleBone AI-64 has several access ports and we can use the power from the board to power the bluetooth

B. ESP-32

The bluetooth was an additional component we needed for two main reasons; the BeagleBone AI-64 does not provide one and the transfer of audio files was necessary through bluetooth as opposed to Wi-Fi. There are a few necessary details worth highlighting.

(1) Provides bluetooth classic and BLE
(2) UART, SPI, I2C and I2S
(3) PWM
(4) SDIO/SPI/UART host controllers
(5) 12-bit SAR ADC (Analog-to-Digital Converter)

These core features are explicit to the files we are sending, which are fairly small to facilitate. UART will have an issue sending or receiving files that are too big due to parsing. The solution for that scenario is to send the same file in smaller parts so the information could be clarified easier and is one of the methods actively employed to accurately send information via bluetooth. The limitations of the quality are reliant on the filter of the speaker receiving the audio and the rate at which the speaker is receiving data. Given there are 34 GPIO for this device, we were also given much freedom in how we were able to route our connections on the device. To benefit the power management of our system, the ESP32 is designed around low power consumption. In fact, there are multiple programmable power modes that can be used on the ESP32 for the processes available, due to the vast programmability of the module. We can also use some features specific to these processes that the ESP boasts such as clock gating and power scaling. The bluetooth module also boasts internal components centered around noise reduction and interference to get some of the cleanest audio signals, which is important considering the amount of interactivity between the boards. While the system is quite simple, it's easy to observe multiple areas where the system noise could become an active issue. On the other hand, the ESP32 dispels any security issues one may have with its integrated secure booting, flash encryption, hardware acceleration and up to 8KB of secured SRAM. It's also important to note that we are sending information in parts for clarity and consistency with no bit parity.

C. TI BQ25895RTWR

To keep the system working properly, the battery we chose for the system needs to both supply and be supplied with enough energy to not cut the BeagleBone AI-64 off, or concurrently any other subsystem that interacts with these circuits. This particular charge controller was adapted for use with the single-cell lithium-ion. Here are some key features that we needed to make this a feasible working component:

(1) Overvoltage/Overcurrent/Overtemp sensing protection
(2) Battery temperature sensing/fault indication
(3) Charge status indicators
(4) Programmable charge current

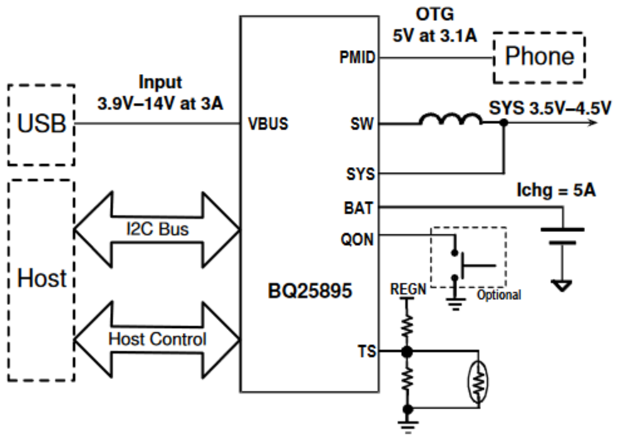

Fig. 5. BQ25895 Simplified Schematic

With the input voltage range from 3.9V to 14V, using our USB connector with the 5V worked perfectly for this application, to where we had the freedom of programming a charge current of up to 3A. The battery calls for a charging current of 2.5A. The operating efficiency sits at 94% which is a margin of reliability acceptable for a build as such. We could go further and configure/control this controller using I2C as a communication protocol. This PMIC also offers a Low Power Mode for anticipated conditions that would be lower than normal, while also accommodating the high voltage inputs. If it was necessary to do so, we could adjust this circuit to discharge up to 9A and it would be capable of doing so given the 11-mΩ battery discharge MOSFET. Furthermore, we can monitor the voltage, temperature and charge current which all need to be realized in the system to get the values needed for all the interactive components. These two equations were used to realize our maximum threshold. Our Iinmax is used to set the required values while the Iin is used to actually measure the value introduced.

$$I_{INMAX} = \frac{K_{ILIM}}{R_{ILIM}} \tag{1}$$

$$I_{IN} = \frac{K_{ILIM} \times V_{ILIM}}{R_{ILIM} \times 0.8\ V} \tag{2}$$

D. TI BQ29700DSER

Because our circuit needs a boost converter to achieve a 5V/3A, we also need a better form of circuit protection than what has been described in the previous sections. The choice here was primarily driven by its sophisticated methods of protection. A refined system will always have multiple layers as that is what's necessary to run continuously without major problems. Absolutely necessary characteristics to consider are as follows:

(1) Under-Voltage Lockout protects the battery from deep discharge. Similar protection for excess charge (OVP).
(2) Short Circuit Protection for the boosting circuit, charge controller, battery and the connectors on the main board.
(3) Access to indication signals/flags. Includes fault detection mechanisms in conjunction.
(4) Typical applications include small technology; minimal space on PCB due to WSON package.

We combined our main board and the charging board into one as the evolution of AMB took its course. The necessity for smaller parts and a more spatially efficient sequence of parts became apparent. The digital communication interface gives us more accessibility critical information via I2C or SPI. This is particularly beneficial for the newfound challenges with battery management. However, upgrading the system's parts in general called for more parts to assist in the adaptations. One of these is concurrent with our protective IC, being the next critical hardware choice - our boost converter.

E. TI TPS61235PRWLR

This is the central component of our boosting system and a concurrent central component for our BeagleBone AI-64. The TPS will be used interchangeably with 'boost converter' in the following section. The boost converter circuit specifically needs an inductor for the concept of the boost to work. The inductor will build a charge as the continuous switch opens and closes at a 1MHz frequency. As previously discussed, this will add noise to the system but is also the most refined solution despite this fact. The inductor was chosen in contrast to the equation below, where the inductor chosen would need a saturation current higher than the possible peak current through the inductor. Also, the RMS current value was noted to be higher than the average input current.

$$I_{L\_peak} = I_{IN\_avg} + \frac{\Delta I_L}{2} = \frac{I_{OUT}}{1-D} + \frac{V_{IN} \cdot D}{2 \cdot L \cdot f_{SW}} \tag{3}$$

$$\Delta I_L = \frac{V_{IN} \cdot D}{L \cdot f_{SW}} \qquad D = 1 - \frac{V_{IN} \cdot \eta}{V_{OUT}} \qquad (4)\ \&\ (5)$$

Using this method of power delivery is more beneficial because it is more consistent, in terms of amperage and voltage values. It facilitates one diode to protect our voltage source and inductor. We can expect high efficiency from this line of product which is always a necessary characteristic. Our application calls for a good threshold of amperage to work with, since power will be distributed to several areas and some additions may eat up amperage at a considerable rate. For our 3.7V input and a 5.1V output, the output amperage realized in this circuit is featured in the TPS datasheet as up to 3.5A but realistically reaches 3.7A. If necessary, the output could be calibrated to specificity or simply monitored. For circuit testing, there is a boost status indication accessible. Since this system is ran primarily through a battery and charged via USB, we can note the benefit of having a low quiescent current. The boost converter chosen touts a 10-μA which is easily applicable for a system that constantly could pull energy. In fact, the load status to the system could further aid in mitigating power consumption. While shutting down, the output is immediately disconnected from the input and the current consumption is reduced by 90% of this value. The only other core concept choices revolve around the safety of the overall system and the lifespan of the boost converter. These include:

(1) Soft start control
(2) Reverse current blocking
(3) Over voltage protection
(4) Thermal shutdown protection

VII. Software Detail

To endeavor to explain the processes occurring between the bluetooth module, the BeagleBone AI64 and the application, we should take a quick look at the operations the AMB will be capable of producing for the user. To quickly encapsulate the operation, the user will transport a file that will be processed by the BeagleBone AI64 and stem split the file back to the application, but continuously stored for bluetooth playback. The application receives the functionality where the following diagram represents this scenario well.

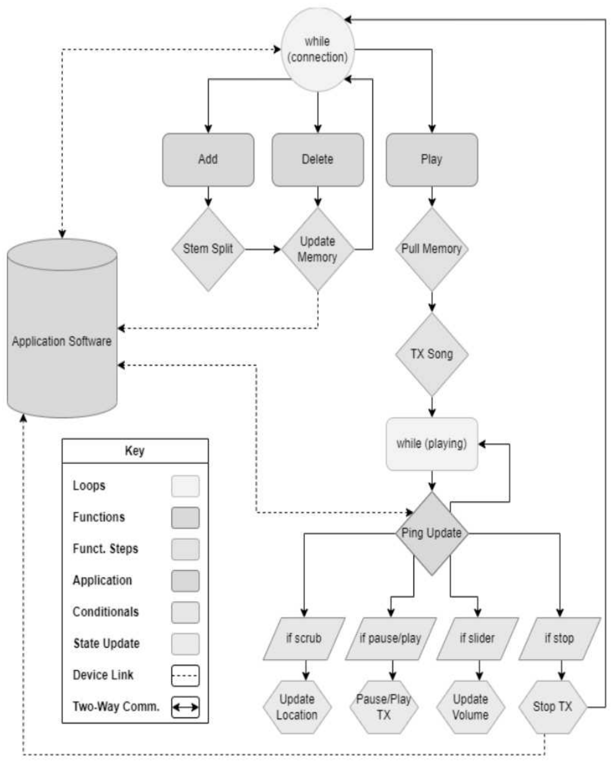

Fig. 6. Embedded software diagram

Following the given diagram, the operations start at the application launch and are controlled through a while loop. The user is capable of adding or deleting to the queue. With the songs at your disposal, choosing one will stem split and will update the system memory to have all the files along with functionalities. When the user wants to play, that file will be offered to the bluetooth and a different scenario for the song adjustability is introduced. Primarily, the box presented will be able to pause, play and stop. The sound parameters will be locked at this point. However, the user will be able to pause the song and adjust these as necessary. Our sliders will adjust the sound of the track and the individual sounds of the stems that were separated during the process. A core package transfer method we employ is JaySON, which is a portable and efficient data transfer tool. This makes sense with the limitations imposed by the bluetooth transfer through UART. This is used for structured data in the hierarchy - since we are sending multiple packages at a time.

VIII. System Evolution With Solutions

Throughout the development of the AMB, several iterations had to be considered with different obstacles per iteration. This section will endeavor to explain the different criterias considered to reach the current status of the AMB. A grand majority of the crucial system changes revolved around the platform for software development. Different considerations for the processing capabilities drove different solutions for software implementation. The main processing platform for stem separation is Spleeter, which has proven to tax the system and has proliferated a few issues. Per stem separation, the size of the individual stems would cost memory and storage to facilitate the processes. This was not the original path, as we chose several different platforms that seemed feasible until problems arose.

The first few boards we chose for embedded development had unique issues which drove us away from pursuing further action. We reached out to NXP and were able to get two sets of boards. The boards we received were similar to the i.mX8 platform we had looked at prior and seemed good on paper until we needed resources. We could not find a board layout for the 1170 boards received and could not get the 1060 boards to connect or interact with our devices. However, it did meet all of our criteria and even had its own platform for coding called MCUXpresso. This could have been feasible in a business setting. Having to find a realistic platform to accommodate the time constraints imposed, we decided to move onto the next major platform of choices including BeagleBone.

As mentioned previously, the power management system took a complete evolution and was tailored for a different style of charging altogether. The system was centered around a multiple cell Nickel Metal Hydride topology. The core charging controller was the BQ2002T and was assisted in regulating the amperage of the charge to the battery via a voltage regulator. The voltage regulator in choice was the LM317, which was regulating the three 1.2V batteries in parallel at 1.43A of charging. The maximum that this circuit could charge was 2A, however it was more important to reduce heat and ensure safety of the circuit. This method would have employed heat shrink and copper to reduce the heat produced by the voltage regulator. The only other core components include a thermistor and necessary circuit elements such as diodes, capacitors and resistors. However, the problem with this system is that when we needed to impose more constraints, the following circuit could not keep up with the demands. It was when the option towards machine learning and AI became feasible that we redesigned the charging circuit which added key components to achieve our goal. The key changes were the battery choice from Nickel Metal Hydride to Lithium-Ion which required a new charge controller. From here, we realized our board choice would need a specific input. The decision to add a boost converter and a circuit protective element would become one of the newest issues regarded here

To properly power the BeagleBone AI-64 we need consistent power to the board. If we do not receive a consistent ramp input upon start or a consistent voltage throughout, we risk the BeagleBone restarting or failing to boot properly. This is actually what we experienced upon further testing of our boost converting circuit. The boost converter provides up to 3.6A, but since we have other areas also drawing power this eats up a significant amount of power necessary for the main processor. The BeagleBone needs a minimum of 3A. This is a problem since the BeagleBone is now starving of current. With the need for fans as well, we noticed each fan will consume 0.25A. We found a solution and that is simply to add another voltage source to help overcome the obstacles presented. There were problems with the embedded but those have been ironed out, as communication with the bluetooth and the serial identification of the speakers were an issue.

IX. Conclusion

To conclude the progress and current status of the AMB, this project has been driven with careful consideration to our limitations and stretch goals laid out. As with any system evolution, our stretch goals were subject to change and the core values that we aimed to pursue have outweighed the issues faced throughout. In either sense, the AMB is a well refined learning process that could only have been realized through market influence and the availability of specific components to make these processes happen. We have found more gratification from solving the issues as they became apparent as a team.

Acknowledgement

The authors would like to thank and acknowledge the support of our sponsor Mr. Herbert Gingold and Dr. Chung Yong Chan.

Meet The Engineers

Nathan Cherry is an Electrical Engineering student with an interest in dance. Nathan plans to further pursue a masters degree at UCF in semiconductors and analog communication.

Elian Seda-Figueroa is a Computer Engineer candidate at UCF with a background in machine learning and hopes to bring this knowledge into the systems engineering domain.

Mark Fritz is a Computer Engineer and an aspiring singer. Mark hopes to use the skills that he acquired during this experience, to delve into the realms of semiconductors and embedded systems.

John Sinanis is an Electrical Engineer candidate at UCF who strives to incorporate his passions of problem solving, fabrication and music in a meaningful way. John hopes to transfer the skills acquired to avionics.

References

[1] *Beaglebone AI-64 high level specification¶ (2023) BeagleBone AI-64 High Level Specification - BeagleBoard Documentation. Available at: https://docs.beagleboard.org/latest/boards/beaglebone/ai-64/ch05.html (Acce ssed: 06 July 2023).*

[2] *"TI BQ25895." Texas Instruments, Oct. 2022, www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fbq25895. Accessed 6 July 2023*

[3] *Spleeter: A Fast and Efficient Music Source Separation Tool. GitHub Repository. Retrieved from https://github.com/deezer/spleeter*