Music Bike

Nick Baroody Ethan Diep Bennett Lahn Matthew Pham

ABSTRACT

Music Bike is an innovative project that transforms the cycling experience by integrating real time adaptive music with a rider's actions. Using sensors mounted on a bicycle, such as an IMU and Hall Effect sensor, the system collects data on speed, tilt, and G-force, which is transmitted using Bluetooth Low Energy to an Android application. Additional threshold algorithms were used to detect different tricks, like a wheelie, jump, and 180. The core objective was to create an intuitive connection between cycling and music, allowing riders to feel as though they are conducting the music with their movements. The mobile application utilizes the FMOD audio engine to dynamically alter the music's tempo, layers, and effects based on the sensor data, creating a unique listening experience for the rider. The system achieves a low-latency feedback loop and seamless sensor integration, ensuring that the music adapts in real time without disrupting the cycling experience. By providing an immersive, adaptive soundtrack that responds to the rider's physical motions, Music Bike reimagines cycling as a performance, turning each ride into a personal, musical journey.

Keywords

Adaptive audio, real-time music, Bluetooth Low Energy (BLE), sensor integration, ESP32, FMOD Studio, Android application, cycling, interactive music system, dynamic audio, embedded system.

1. INTRODUCTION

Music Bike is an innovative project that transforms the cycling experience by integrating real-time adaptive music with a rider's actions and tricks on the bike. Using sensors, such as an IMU and Hall Effect sensor mounted on the bike, the system collects data related to speed, tilt, and G-force, which is transmitted to a mobile application via Bluetooth Low Energy (BLE). The mobile app uses this sensor data to adjust the rider's music in real time, creating an interactive and dynamic experience for the rider. The Music Bike is designed to make the act of cycling feel like conducting an orchestra, where each motion influences the musical composition, enhancing the rider's engagement with both the physical and auditory elements of the activity.

Figure 1: The Music Bike

Cycling is a popular recreational and fitness activity, but its experience is often static, with little variation beyond the rider's surroundings. Music Bike addresses this by adding a layer of musical dynamism, transforming cycling into an immersive multi-sensory experience. Drawing inspiration from similar systems like the Mercedes-AMG MBUX SOUND DRIVE, which adapts music to a driver's behavior, and the Sonic Bike, which uses GPS to trigger music based on location, the Music Bike adapts the music based on the cyclist's performance and movements. Such as performing tricks or changing riding intensity. This project aims to achieve a low-latency real time connection between sensor data and audio feedback, ensuring that the music feels seamlessly responsive.

This report will describe the design and development of the Music Bike, detailing the hardware setup, sensor integration, mobile app functionality, adaptive audio logic using FMOD Studio, and performance metrics. It will also cover challenges faced during implementation and the solutions developed to ensure a smooth and engaging cycling experience. Finally, we will discuss the potential

future applications of this technology and opportunities for expanding its capabilities, as well as possible industry standards our design would be held to if it was made into a commercial product.

2. RELATED WORK

2.1 Mercedes-AMG MBUX SOUND DRIVE

We took some inspiration from Mercedes-AMG's MBUX SOUND DRIVE which is a feature in some Mercedes-AMG vehicles that allows the music in the car to respond to one's individual driving style. "MBUX SOUND DRIVE connects the vehicle's hardware with the music software using precise in-car signals. The system uses sensors to interpret driving dynamics such as acceleration, steering, braking and recuperation and transforms them into musical expressions" [1]. This allows drivers to treat their vehicle as an instrument and compose music in real time.

Our Music Bike takes inspiration from this project, as our goal was to make riding a bike feel like composing an orchestra. The most obvious difference between our Music Bike and the MBUX SOUND DRIVE is that ours is implemented on a bike, rather than a car. But this is an important distinction because the MBUX SOUND DRIVE uses more basic sensor inputs, such as acceleration and steering, while the Music Bike was programmed and trained to detect tricks, such as wheelies, jumps, and 180s on the bike. Another key difference is that the MBUX SOUND DRIVE adapts the driver's own music, while the Music Bike has preset tracks that respond to various sensor inputs.

2.2 Sonic Bike

Another similar product to ours is the Sonic Bike, "which was invented by Kaffe Matthews and has evolved over 16 years of international projects, and continues to be researched and developed to expand the compositional potentials and unique listening experiences it creates" [2]. The Sonic Bike plays sound and music that changes depending on where the cyclist goes and how fast they ride. It is a bicycle with speakers on the front and a GPS tracking audio system on the back. The Sonic Bike performs from a Raspberry Pi3 with a GPS receiver attached. The system is powered by a USB charged power pack and everything is stored in a box secured on the back of the bike frame. The Sonic Bike uses two types of software, the mapping software which enables the drawing of zones of sound on streets, paths, parks, anywhere creating a sound map score. The other software is the system software which runs on he Raspberry Pi2 and enables the bike to locate itself on this sound map and it triggers specific music fragments to play in certain ways at particular locations.

Our Music Bike is similar to the Sonic Bike as they both transform cycling into a dynamic musical experience. The key difference is that the Music Bike uses various sensors, such as an IMU and hall effect sensors to detect tricks on the bike and then have the music respond to those sensor inputs. On the other hand, the Sonic Bike uses a GPS system to adapt the music based on where the cyclist is on a certain sound map. Additionally, the Sonic Bike's kit with the microcontroller and GPS is placed in a box behind the seat, while our implementation is placed on the shaft below the seat. This difference makes sense, as our implementation needs to be placed here to more accurately detect the tricks that the cyclist performs.

3. TECHINAL DETAILS

3.1 Overview

The Music Bike system transforms cycling into an interactive, dynamic musical experience by integrating real time sensor data with adaptive audio. The system is composed of the hardware, a mobile application, and an audio engine that all work together to create a seamless experience.

The device is composed of a custom PCB board with connections to integrate a microcontroller, an IMU, and Hall Effect sensors. The sensors collect data and send it to our app via BLE. The data is interpreted and if a bike trick, such as a wheelie, jump, or 180, is detected, then the FMOD audio engine, which contains multiple audio banks that the rider can choose from and each bank varies in how it adjusts the music using a set parameter and the rider can hear this change in music in real time.

3.2 Theory of Operation

The architecture consists of three main components: the hardware, the mobile application, and the FMOD audio engine.

Hardware: The hardware consists of an ESP32-S3 microcontroller, which is responsible for reading sensor data from multiple sources, such as an MPU9250 IMU and a KY-003 Hall Effect sensor. The sensor data is transmitted to the mobile app via BLE. The ESP32-S3 is also powered by a lithium battery.

Mobile Application: The mobile app, developed for Android, connects to the hardware via BLE, receives sensor data, and processes it for music playback. The app communicates with the FMOD Studio engine to adapt the music based on sensor inputs. The user interface allows the rider to pick their track based on different FMOD Studio banks and view the sensor data.

FMOD Audio Engine: FMOD Studio is used for the adaptive audio system, which responds to changes in the sensor data. FMOD processes the incoming data to modify various aspects of the music, such as pitch, layering of instruments, and changing the song based on the rider's performance.

3.3 Implementation Details

3.3.1 Hardware

Most of our hardware is implemented in a small, compact sensor pack that is mounted just below the bicycle seat.

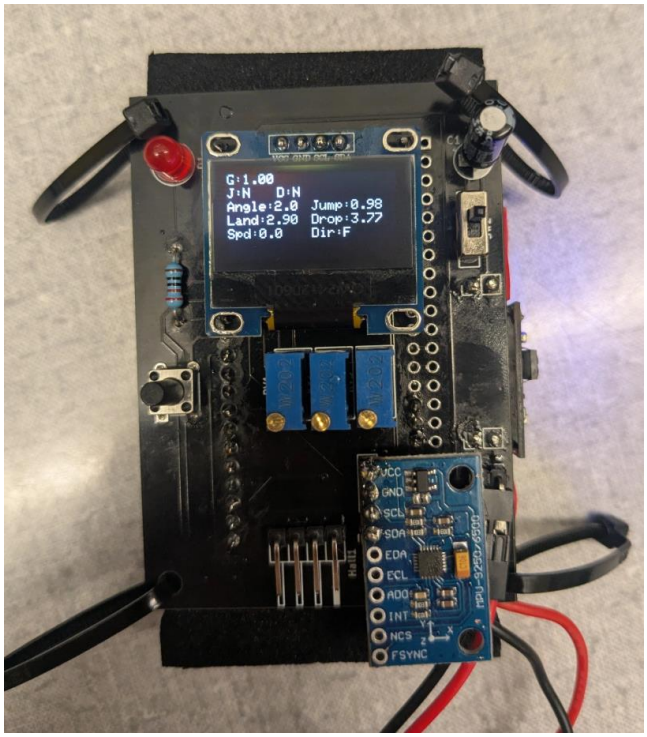

Figure 2: Front of Sensor Pack

Figure 2 shows the top of our sensor pack and it contains the display, accelerometer zero button, power switch, MPU9250 IMU, LED for confirming Bluetooth connection, and potentiometers for adjusting the jump/drop sensitivity.

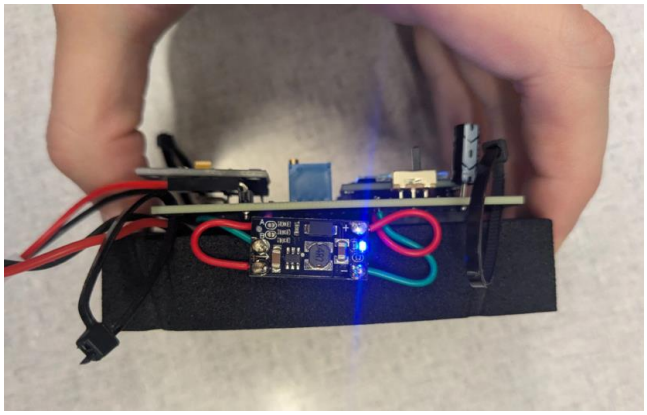

Figure 3: Side of Sensor Pack

Figure 3 shows the side of our sensor pack, which contains the power delivery board used to regulate the voltage of our attached lithium-ion battery.

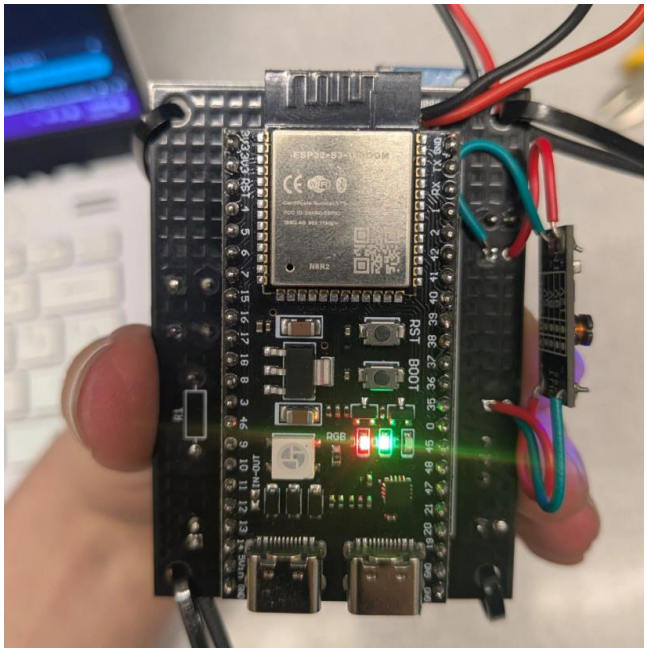

Figure 4: Back of Sensor Pack

Figure 4 shows the back of our sensor pack, which contains our ESP32-S3 microcontroller. When mounted on the bike, the back is covered by a piece of foam and the lithium-ion battery is placed in the void between the microcontroller and the PCB.

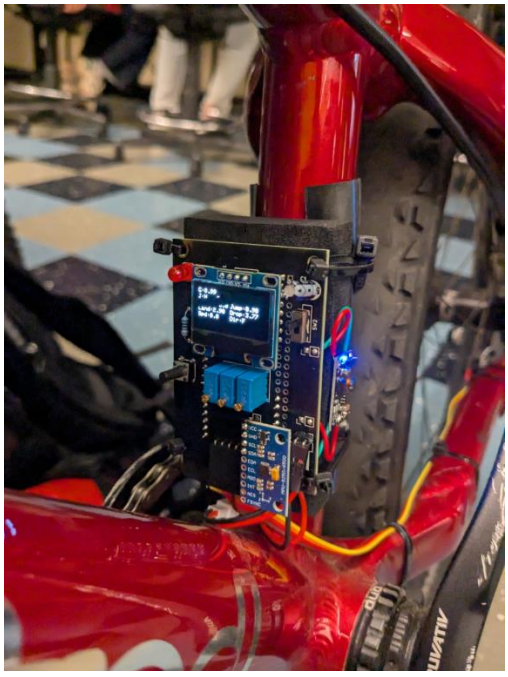

Figure 5: Sensor Pack Mounted on Bike

Figure 5 shows the sensor pack mounted to the bike using a piece of rubber to reduce vibrations and zip ties to keep the device steady. The sensor pack is mounted below the seat and in a sturdy place to ensure safety of the device and for accurate sensor data readings.

Figure 6: Hall Effect Sensors

Figure 6 shows our two KY-003 Hall Effect sensors mounted with zip ties near the rear wheel hub. The magnets attach to the spokes of the wheel and the two hall effect sensors are used to detect direction.

3.3.2 Software

Our mobile application was implemented using Android Studio and it connects to our sensor pack device to receive sensor data from it via BLE. Our app user interface is quite simple and has multiple sections. The first is the Devices section and this is where you can connect to one of the sensor pack devices via Bluetooth and the LED on the sensor pack will turn on to confirm successful Bluetooth connection. The Devices section of the app can be seen in Figure 7.

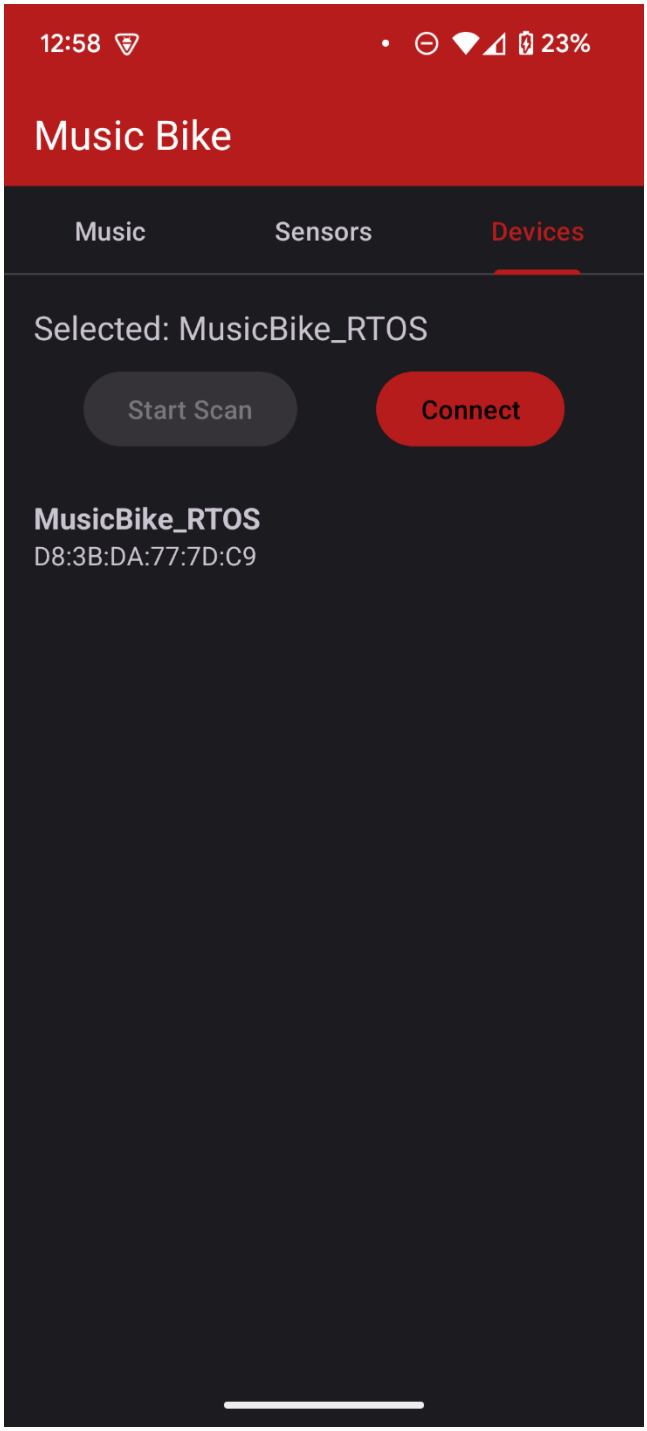

Figure 7: Devices Section on App

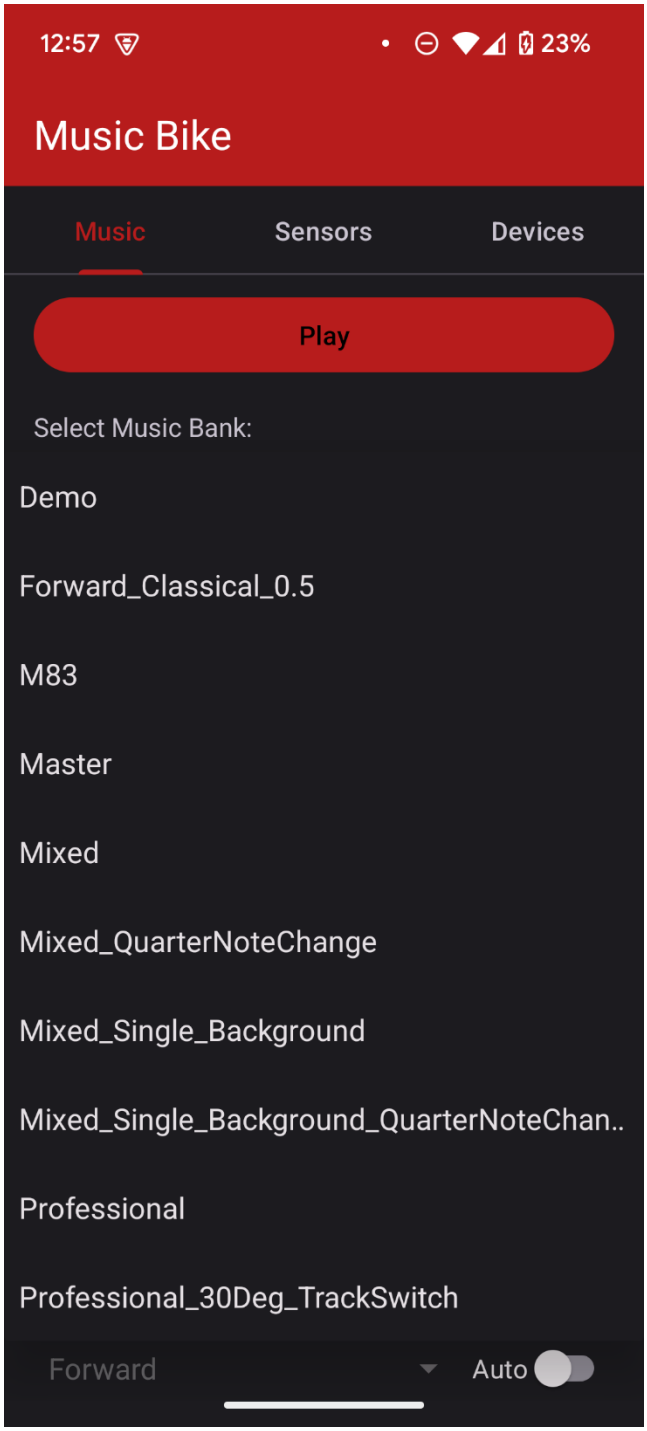

Figure 8: Select Music Bank in Music Section on App

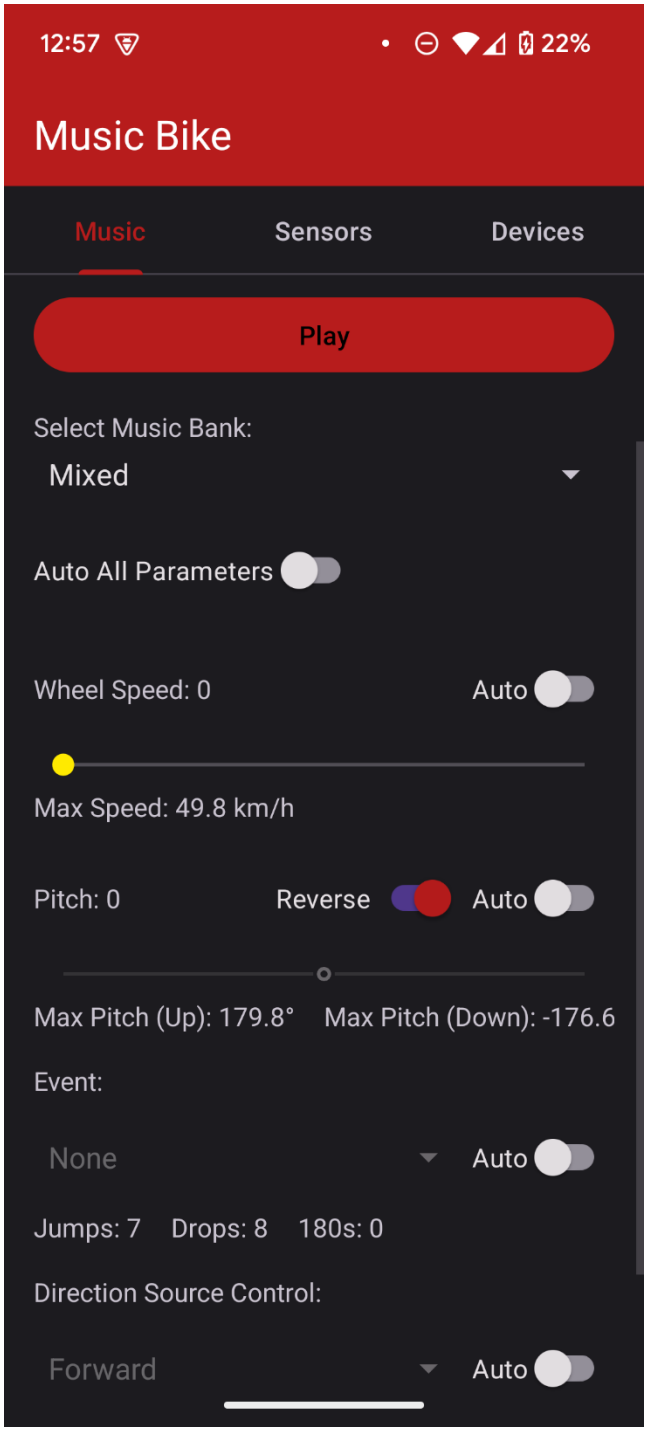

Figure 9: Parameters in Music Section on App

Another section on the app is the Music section and this is where the rider can select which music/audio bank they want to listen to. We used FMOD Studio to edit the music and create the various music banks, each of which contain unique audio adjustments based on parameters that correspond to different bike tricks. The FMOD audio engine library is connected to the Android app, so it can interact with the data sent from the sensor pack in real time. Figure 8 shows where the rider can choose from different music banks.

Figure 9 shows the parameters and events that happen in real time when the rider is riding the bike. This allows the rider to see the data being collected and sent from the microcontroller. The app works to receive sensor data from the sensor pack via BLE, and also connects with the FMOD audio engine to establish various bike tricks as parameters in FMOD studio, and when one of these parameters/tricks is detected, FMOD studio is configured to adjust the music, depending on the music bank that is selected by the rider.

3.4 Distinctions

3.4.1 PCB

Initially, our hardware design prototype was a perf board based setup with the parts soldered on. We developed a custom PCB to replace this, so that the components are in a more compact, durable, and professional design suitable for long term testing and potential deployment. The schematic was made to integrate the ESP32 microcontroller, sensor connectors, power regulation, and potentiometer inputs onto a single board. The PCB schematic and layout can be seen below.

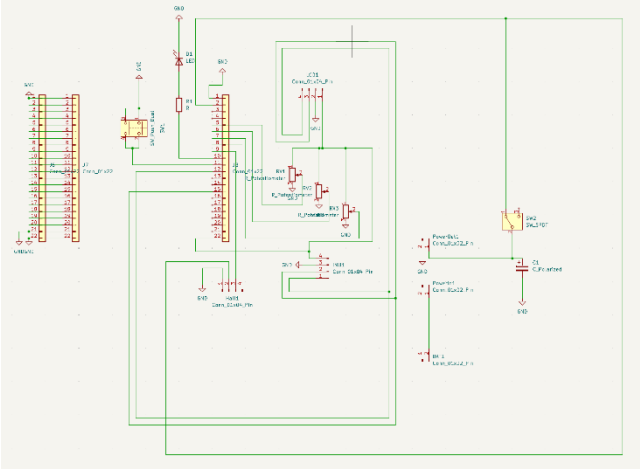

Figure 2: PCB Schematic

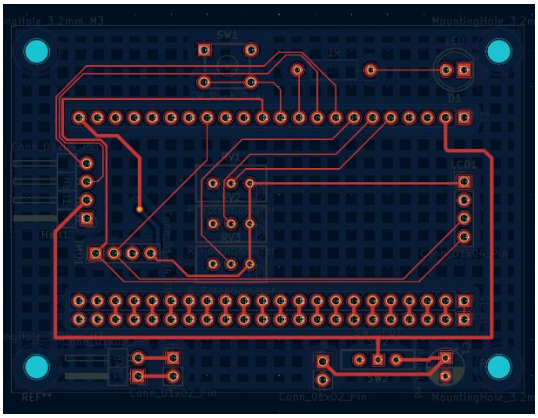

Figure 3: PCB Layout

3.4.2 "Machine Learning"

Our first attempt to accurately detect jumps, hops, and wheelies used a TensorFlow model running on our mobile app, trained using a custom dataset. We ran into accuracy and latency problems with this solution. Reducing the false positive rate of our model required extending the input to 10 seconds, but this limited the user to detecting a trick once every 10 seconds. Even with this modification, we were unable to accurately detect 180s. We attribute this to our dataset, which naively contained only time, current pitch/yaw/roll, and g-force. It did not contain angular velocity and similar raw metrics that define a movement like a 180. This was a critical oversight, but by the time we realized our mistake we didn't have time to collect new data.

So, we pivoted to an algorithmic threshold classifier. The classifier requires three separate criteria to be true to decide a 180 has occurred: a g-force anomaly (classified as a g-force over 2.37g or below 0.6g) within the last 1.5s, angular displacement must be over 90 degrees, and a "spin score" must be over 0.8. Both spin score and angular displacement decay with time if angular velocity is below a certain threshold. "Spin score" grows faster if the current angular velocity is higher. It was included as a metric of detection to increase the number of tunable parameters and reduce the likelihood of false positives by biasing detection towards large, high angular velocity movements.

With this algorithm, we were able to achieve detection accuracy above 80% with fewer false positives than our machine learning algorithm. Machine learning is still something we'd revisit in the future if we had the time to train a more robust, larger dataset.

3.4.3 Live Demo

To highlight our confidence in our design's reliability, we chose to include a live demo of all app and bike functionality as one of our distinctions. We demoed the Bluetooth, as well as adaptive music capable of detecting forward, reverse, pitch, angle, jumps, and 180 detection, where we achieved near perfect accuracy.

4. DISCUSSION

4.1 Reflection

After thorough testing and using machine learning to train our model to detect various bike tricks, such as a jump and a 180, we found that our design works very well in detecting these tricks and then sending a signal to the app, which sends a signal to FMOD to adjust the music the rider hears.

Next steps could include further training our model with machine learning and gathering more data to train and test to further improve the accuracy of detecting tricks.

Currently our technology is primarily used to make it seem like the cyclist is conducting an orchestra, by performing various bike tricks to adjust the music. However, another application of our technology could be simply detecting bike tricks. Our code can accurately detect various tricks, such as a wheelie, jump, 180, and riding backwards, so cyclists who like to perform tricks could use the music as cue or acknowledgement that they performed the trick correctly.

We learned a lot from this project, including learning how to use new software, such as FMOD Studio and Android Studio. Additionally, this project allowed us to learn more about embedded systems in general and how we can seamlessly integrate the hardware with the software.

If we were to do something differently, we would use machine learning to train the data from the actual device (ESP32 microcontroller), instead of from the app. This could make the machine learning training slightly more accurate because it would be training the raw data from the input sensors on the

microcontroller, instead of from that data that is sent to the app via Bluetooth.

4.2 Open Source

We have made our project open source so that any of us can be as involved as we want in the future, and so that the project continues to grow. Having an open-source environment would allow anyone to make edits or suggestions to our GitHub repository. To officially make our project open source, we obtained an MIT License for our GitHub repository, which is a popular open-source license that allows developers to use, copy, modify, merge, publish, and distribute software, provided the original copyright and license text are all included. Making our project open source will allows us and others to contribute their ideas and hopefully further develop the Music Bike.

4.3 Industry Standard

An industry standard that the Music Bike would be held to if it was developed into a commercial product would be the IEEE 802.15.1 standard, which defines the specifications for Bluetooth technology. This standard is relevant for the Music Bike, as the project utilizes Bluetooth connectivity between the ESP32 microcontroller and the Android app. Bluetooth, based on IEEE 802.15.1, enables the seamless transfer of sensor data from the bike's IMU and hall effect sensors to the mobile app, allowing for real time interaction and feedback during biking activities.

Key Aspects of IEEE 802.15.1 Relevant to the Music Bike:

1. *Wireless Communication and Power Consumption:*
   Bluetooth devices that comply with IEEE 802.15.1 are designed for short range communication and for the Music Bike, the short-range nature of Bluetooth is ideal for maintaining a stable connection between the microcontroller and the app. This standard also specifies low power operation, which is crucial for a battery-operated system like the Music Bike. By being held to this standard, the Music Bike can operate efficiently without draining the battery too quickly, ensuring longer usage periods.

2. *Security Features:*
   The IEEE 802.15.1 standard incorporates security protocols such as encryption, authentication, and authorization, which are crucial for protecting data transmitted between the bike and the mobile device. For the Music Bike, these security measures could prevent unauthorized access to user data, so complying with these security features ensures that the Music Bike is a trustworthy and secure user experience.

3. *Data Transfer Rates and Latency:*
   The Music Bike requires real time data transmission for activities like detecting bike tricks and triggering corresponding changes in music. Compliance with the data rate and latency specifications of IEEE 802.15.1 ensures that the Music Bike responds to user actions without noticeable delays. This real time performance would be critical for a seamless and immersive interaction between the rider and the music.

By adhering to the IEEE 802.15.1 standard, the Music Bike can ensure reliable, efficient, and secure Bluetooth communication between the microcontroller and the app. This standardization is important for maintaining device compatibility, minimizing connection issues, and ensuring smooth performance, all of which contribute to a positive user experience.

5. CONCLUSION

In this report, we have presented the development and implementation of the Music Bike system, which transforms the cycling experience by integrating real time adaptive music with the rider's actions. By leveraging sensors such as an IMU and Hall Effect sensor, the system captures motion data, processes it on an ESP32 microcontroller, and uses this data to adapt music playback in real time using the FMOD audio engine. The integration of BLE communication ensures low-latency interaction, creating an immersive and responsive musical experience for the rider.

The primary goal of Music Bike was to create a system where the rider feels like they are conducting the music through their movements. The system achieves this by dynamically adjusting the music based on sensor data. Through careful design and optimization, we have succeeded in building a low-latency system that responds intuitively to the rider's physical actions, providing a seamless connection between cycling and music.

Throughout the course of the project, we encountered and addressed several challenges, including sensor data accuracy, and achieving the desired level of music adaptation. Our solutions, such as implementing sensor filtering algorithms and optimizing BLE communication, helped mitigate these issues and ensured that the system performed reliably during testing.

While the project has been successful in demonstrating the core concept of Music Bike, there are several potential areas for future improvement. For instance, we could explore more advanced sensor types for greater accuracy or integrate more complex music adaptation to provide an even richer listening experience. Additionally, increasing the system's power efficiency or integrating additional environmental sensors could further enhance the experience.

Overall, Music Bike represents an exciting step forward in combining physical activity with adaptive audio, creating a unique form of interactive entertainment. Future iterations of this project could lead to applications in other areas, such as fitness training, gaming, or even music therapy, where real time sensory input can influence music in meaningful ways. The potential for innovation in this field is vast, and we look forward to continuing to develop and refine this technology.

6. REFERENCES

[1] Mercedes-AMG. "MERCEDES-AMG MBUX SOUND DRIVE." https://www.mercedes-amg.com/en/mercedes-amg-mbux-sound-drive (accessed June 4, 2025).

[2] B. R. Institute. "Sonic Bike." https://sonicbikes.net/sonic-bike/ (accessed June 4, 2025).