PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

STUDENT INNOVATION IN FITNESS AND SPORTS: MUSIC-TO-FITNESS BIOFEEDBACK SYSTEM

A PROJECT REPORT

Submitted by

ABHISHEK GOWDA S - 20221COM0137

RITHIN U REDDY - 20221COM0162

RUKMINI SURVE M - 20221COM0187

Under the Guidanceof,

Mr. MOHAMED SHAKIR

Assistant Professor

School of Computer Science and Engineering

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

Certified that this report **" STUDENT INNOVATION IN FITNESS AND SPORTS: MUSIC-TO-FITNESS BIOFEEDBACK SYSTEM"** constitutes bonafide work completed by **ABHISHEK GOWDA S (20221COM0137), RITHIN U REDDY (20221COM0162), and RUKMINI SURVE M (20221COM0187)**, who have successfully carried out the project work and submitted this report for partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER ENGINEERING** during 2025–26.

Mr. Mohamed Shakir	**Dr. Benitha Christinal J**	**Dr. Sampath A K** **Dr. Geetha A**	**Dr. Pallavi R**
Project Guide	Program Project Coordinator	School Project Coordinators	Head of the Department
PSCS	PSCS	PSCS	PSCS
Presidency University	Presidency University	Presidency University	Presidency University

Dr. Shakkeera L	**Dr. Duraipandian N**
Associate Dean	Dean
PSCS	PSCS & PSIS
Presidency University	Presidency University

Sl. No	Name	Signature	Date
1			
2			

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, the students of final year B.Tech in **Computer Engineering** at Presidency University, Bengaluru, namely **Abhishek Gowda S (20221COM0137), Rithin U Reddy (20221COM0162), and Rukmini Surve M (20221COM0187)**, hereby declare that the project work titled **"Student Innovation in Fitness and Sports: Music-to-Fitness Biofeedback System"** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in **Computer Engineering** during the academic year 2025–26. Furthermore, the matter embodied in the project has not been submitted previously by anybody for the award of any degree or diploma to any other institution.

ABHISHEK GOWDA S	**USN: 20221COM0137**
RITHIN U REDDY	**USN: 20221COM0162**
RUKMINI SURVE M	**USN: 20221COM0187**

PLACE: BENGALURU

DATE: 26 November 2025

ACKNOWLEDGEMENT

We want to express our sincere thanks to all who helped us throughout this project work. Without the guidance and support we got from so many people this work would not have been possible at all.

We extend our heartfelt thanks to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their continuous motivation and support during the project completion time.

We sincerely thank our internal guide **Mr. Mohamed Shakir, Assistant Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his invaluable guidance and moral support and encouragement throughout the entire duration of our project work. His timely inputs and mentor-ship helped a lot in shaping this research and we are really grateful for that.

We remain deeply grateful to **Dr. Pallavi R, Professor and Head of the Department**, Presidency School of Computer Science and Engineering, Presidency University, for her mentor-ship and constant encouragement throughout.

We express our sincere appreciation to **Dr. Duraipandian N**, Dean PSCS PSIS, and **Dr.Shakkeera L**, Associate Dean, Presidency School of Computer Science and Engineering, along with the Management of Presidency University for providing the necessary facilities and intellectually stimulating environment that helped in the completion of our project work.

We are grateful to **Dr. Benitha Christinal J**, Program Project Coordinator, and **Dr. Sampath A K** and **Dr. Geetha A**, School Project Coordinators, Presidency School of Computer Science and Engineering, for facilitating problem statements, coordinating reviews, monitoring progress, and providing valuable support and guidance throughout.

We also thank the Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and staff from other departments who extended their valuable help and cooperation.

ABHISHEK GOWDA S
RITHIN U REDDY
RUKMINI SURVE M

Abstract

So this project is basically about creating a smart fitness system that changes your workout music based on how hard you're actually exercising. We called it the Music-to-Fitness Biofeedback System and honestly the idea came from noticing how most people struggle to maintain proper workout intensity during their gym sessions.

The system we built uses an ESP32 micro-controller as the brain of the whole operation it has a MAX30102 sensor for monitoring heart rate and an MPU6050 accelerometer for tracking movement and body motion. There's also a DF Mini MP3 player connected to a speaker that plays different songs depending on what the sensors detect about your workout intensity.

When it comes to how the thing works well the sensors constantly monitor your heart rate and how much you're moving around and then our algorithm figures out if you need calm music or something more intense based on that data. The OLED display shows you real time information about your heart rate current activity level and which song is playing so you always know what's going on.

We tested this prototype and found that users stayed in their target heart rate zone about 23 percent more consistently compared to when they just used regular static playlists. That's pretty significant if you think about it because maintaining proper intensity is one of the biggest challenges people face when working out alone.

The whole thing is self contained meaning you don't need a smartphone or internet connection to use it just the device itself with an SD card full of songs. We think this kind of hardware based approach could really help people who struggle with motivation and pacing during their workouts because the music adapts to them automatically instead of them having to manually change songs or check their fitness tracker every few minutes.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Form
API	Application Programming Interface
BPM	Beats Per Minute
DoF	Degrees of Freedom
ESP	Espressif Systems Processor
FAT32	File Allocation Table 32
GPIO	General Purpose Input Output
HR	Heart Rate
I2C	Inter Integrated Circuit
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IoT	Internet of Things
IR	Infrared
LCD	Liquid Crystal Display

Chapter 1

Introduction

1.1 Background

Alright so let me start by explaining what this whole project is actually about. When you think about fitness and working out there's always this challenge of keeping the right pace and intensity throughout your exercise session. Most people either push too hard in the beginning and burn out quickly or they start slow and never really get into that effective workout zone.

Now music has been around forever as a workout companion right everyone knows that listening to the right tunes can make a huge difference in how motivated you feel. But here's the thing most people just create a playlist beforehand or use some streaming service that plays random songs and honestly that doesn't really adapt to what your body is actually going through during the workout.

We noticed this gap and thought what if we could create something that actually monitors your body in real time and then changes the music based on how hard you're working. That's basically the core idea behind our Music to Fitness Biofeedback System. The whole concept revolves around using sensors to track your heart rate and movement then using that information to automatically select music with a matching tempo.

The fitness technology sector has experienced an insane amount of growth in a very short time and the gadgets such as smart-watches and fitness bands are now an integral part of every person's daily life. Still, what most of these gadgets do is just record your stats and present them either on a screen or an app; they don't take any active part in helping you keep up the right intensity during your workout. What we aim to do in our project is to beat this by establishing a closed-loop system that not only tracks the user's progress but actually alters the audio feedback simultaneous through monitoring.

1.2 Statistics of Project

The fitness technology market has essentially been growing in large part to a number of studies and data reviews that look at the numbers and that reveal such a trend. Grand View Research has put the fitness tracker market at more than 114 billion dollars by 2028 which not only confirms how much people but also how much they are willing to invest in their health and fitness travels.

There are not only some surprising numbers but also some very interesting statistics when we consider music and exercise together. It was found in some studies that people participating in the same synchronous music where their movements coincide with the beat have up to 15 percent more endurance which is a big deal, especially for athletes or for anyone who trains seriously. The only downside though is that manually getting into this synchrony is almost impossible because it would require one to always think of making his/her pace similar to that of the song which kills the very idea of music being a distraction from fatigue.In our own testing we found that users stayed in their target heart rate zone about 23 percent more consistently when using our adaptive system compared to just listening to a static playlist. That improvement came from the automatic adjustment of music tempo based on real time physiological data.

The ESP32 micro-controller we're using runs at 240 MHz with dual cores which gives us plenty of processing power for sensor data fusion and decision making. The sensors we integrated can sample data at rates sufficient for accurate heart rate detection typically around 25 samples per second for the PPG sensor and 100 samples per second for the accelerometer.

1.3 Prior Existing Technologies

First there are the basic fitness trackers such as those from Fitbit and Garmin. They're fantastic for collating data and displaying you graphs of results after your session, but when it comes to actually using them during a workout they are still just numbers on a screen. You then still have to see that device interpret what the numbers means and then you consciously adjust your effort which is a lot of mental work. And there are music apps like Spotify that automatically generate workout playlists based on BPM ranges. The catch with these playlists is that they're static, so the tempo doesn't adjust to what you're doing. When you run harder the music doesn't change – when you can't keep up because you're tired, neither does it. There has been some attempts by running apps to blend music with cadence detection i.e RockMyRun and the likes. These are a bit closer to what we're doing but they need monitoring and proper connectivity i.e use smart phone so you have to take phone while workout sand also depend on internet.

connectivity for streaming. Our system is completely standalone you just need the device itself.

There have been academic research projects exploring biofeedback through audio but most of them focus on post workout analysis or require expensive laboratory equipment. The practical implementations for everyday use have been limited and that's where we saw an opportunity to create something that's actually usable by regular people.

1.4 Proposed Approach

Aim of the Project : The objective was to create a single user wearable device that constantly decides and plays music dynamically and in real-time based on the fitness level of the user, without need for external input using locally computed biometric and motion analyses.

Motivation : The available activity-based music solutions (e.g., smartphone applications), require an internet connection, a connected phone, and user interaction in order to operate which disrupt the flow of exercising. It's time for a frictionless "unplugged" system that adapts its resistance on-the-fly and lets you focus sustained fitness workouts without distractions or being interrupted to adjust when the user is ready.

Proposed Approach : The approach is based on a hardware device developed in-house, with sensor technology, on board processing and audio playback. System architecture is as follows:

1. Sensing: The heart rate is recorded using a MAX30102 optical sensor and the symmetric movements are monitored with an MPU6050 accelerometer.

2. Processing: The central brain is an ESP32 micro controller, which runs a custom algorithm collecting raw sensor data and turning it into action measures (BPM, steps per minute).

3. Decision & Playback: According to the user real time status, if there is a pre-loaded SD card available with matching tempo song, then that track is played using DF Mini MP3 module.

4. User Interface: A small OLED display provides real-time feedback on heart rate, activity level, and the currently playing song.

Applications of the Project :

1. Personal Fitness Enhancement: Automatically play music and motivate during running, cycling or gym workouts
2. Rehabilitation: Offering paced auditory stimulation for physiotherapy or cardiac rehab sessions.
3. Niche Sports: Use in activities where carrying a phone is impractical (e.g., trail running, swimming).
4. Focus & Productivity: Potentially adapting to regulate arousal states for concentration tasks.

Limitations of the Proposed Approach:

1. Limited Musical Library & Personalization: The music selection is restricted to pre-loaded songs on an SD card and may not align with the user's personal taste, potentially reducing its motivational effectiveness.

2. Sensor Reliability in Motion: Optical heart rate sensors (like the MAX30102) are prone to motion artifacts, which can lead to inaccurate readings during vigorous activity, compromising decision quality.

3. Simplistic Adaptation Logic: The approach primarily uses tempo (cadence/heart rate) for music selection, ignoring other important musical features (genre, energy, lyrics) and the user's psychological state or workout goals (e.g., warm-up vs. peak intensity).

4. Hardware Constraints: Being a self-contained wearable, it faces challenges in size, battery life, and processing power compared to smartphone-based solutions, limiting algorithmic complexity and data storage.

5. Lack of Data Logging & Long-Term Adaptation: The system operates in real-time without storing historical data, preventing it from learning a user's long-term preferences or tracking performance trends over time.

The system continuously monitors two key parameters your heart rate and your movement cadence. The heart rate is measured by optical sensor, which uses light to monitor blood movement in your fingers or earlobe. That movement tracking is done by an MPU6050 accelerometer, which can measure both how much you're moving and the speed at which you do so. All of this sensor data plugs into an ESP32 micro-controller, which is sort of the brains of the operation. The ESP32 manages our algorithm that ingests raw sensor readings, processes them to derive values such as beats per minute for heart rate and steps per minute for cadence, and ultimately decides what music to play. The music is played through a DF Mini MP3 module where I have uploaded some pre recorded songs on an SD card. We created a music library where each song is tagged with its tempo and so your system can choose which track makes the most sense for you right now, given what the algorithm thinks you need. There's an OLED screen that displays real time feedback on your heartbeat, current activity level and even lets you know what song is currently playing. This provides you a "peek" at what the system is doing without having to get in your way.

The keydifferencefromexistingsolutionsisthateverythinghappensautomaticallyandlocally. You don't even need a phone, you don't need internet - You don't event have to manually change the song. You shouldn't have adapt to the system, the system should be adapting you.

1.5 Objectives

The main goals that we planned to reach during the project were these. The first goal was to create a working prototype of a music-to-fitness biofeedback system which could be run independently and did not need any external devices or connection. This called for the combination of all sensors processing and audio playback in one portable unit.The second goal was to come up with a trustworthy data fusion algorithm. This was to blend heart rate and motion data together and then select music intelligently based on this. Our idea was that the algorithm should be powerful enough to adapt to various activities like walking, jogging, and running but at the same time simple enough to be operated on a resource-limited micro-controller.

The third objective was a system that would genuinely healthcare the workout consistency. Simply fabulous hardware is not enough, the device should have a quantifiable favorable effect on users' exercise habits. That's why we considered testing and validation as an integral part of our project.Another objective was to keep the cost low and use components that are readily available. We specifically chose the ESP32 MAX30102 MPU6050 and DF Mini Player because they're affordable widely documented and easy to source for anyone who might want to replicate or build upon our work.

Finally we wanted to document everything properly so that this project could serve as a foundation for future development. Whether someone wants to add Bluetooth headphone support or implement machine learning for activity recognition the architecture we designed should be flexible enough to accommodate those enhancements.

1.6 Sustainable Development Goals (SDGs)

Our project aligns with several of the United Nations Sustainable Development Goals particularly those related to health and well-being.

1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Figure 1.1: Relevant Sustainable Development Goals for Water Footprint Calculator

SDG 3 Good Health and Well-being: This is the most directly relevant goal for our project. By With helping people to keep the right level of exercise intensity, we are making sure that health outcomes are improved. The prevention of chronic diseases such as heart disease, diabetes, and obesity is most effectively done through regular exercise, done at the right intensity levels. Our system offers those who want to enjoy these advantages the most straight way without the hassle of costly fitness studio memberships or personal trainers.

SDG 9 Industry Innovation and Infrastructure: Our project showcases the innovation that takes place at the crossroads of embedded systems, sensor technology, and health applications. By showing that intricate biofeedback systems can be made using inexpensive readily available components we are paving the way for the fair use of health technology. Consequently, it can lead to the growth of new ideas in the countries where access to the costly fitness equipment is restricted..

SDG 12 Responsible Consumption and Production: Our project is built on a design philosophy that favors the use of minimalistic hardware with the intention of having maximum functionality. We decided not to design a device that would be continually charging or reliant on internet connectivity but rather to produce one that consumes less power and works regularly. Such an approach, though, leads to the decrease of electronic waste generated from the production and disposal of such devices and, of course, that of the energy consumed for their operation, compared to the case of more intricate solutions.

1.7 Overview of Project Report

Let me give you a quick rundown of what you'll find in the rest of this report.

Chapter 2 covers the Literature Review where we dive into existing research on biofeedback music psychology in sports and wearable tech. systems This context allows us to clarify which design choices we made and how our work is positioned in the wider research area of fitness technology. A detailed explanation of our development process the hardware and software tools we used and how we built and tested the prototype is given Chapter 3 Methodology: It is basically the how we did it section. In Project Management Chapter 4 we will get to see our timeline risk analysis and budget breakdown. This reveals the planning aspect of the project and how we distributed our work throughout the semester. Analysis and Design is the title of Chapter 5 that takes us through system requirements, block diagrams, and flowcharts as well as providing in-depth rationales for component selection and system unit design. In Chapter 6 the focus will be on the collaboration of hardware with software and simulation where we will take you through the actual implementation details including circuit connections, code snippets, and any simulations run prior to constructing the physical prototype. In Chapter 7, we decode Evaluation and Results where our testing procedure, the data we gathered, and the insights from running the experiments with the system are all documented. In Chapter 8 we are going to talk about social, legal, ethical, sustainability, and safety issues that are important for any technology project that deals with health data and wellness of the people. In conclusion, Chapter 9 is a summary of our findings, lessons learned, and future work that could help this project go further. The References section provides a comprehensive list of academic articles, technical documents, and other sources that we relied on while the Appendices section contains additional materials like full code listings and auxiliary diagrams.

Chapter 2

Literature Review

Consequently, we spent a considerable amount of time before the actual building process to familiarize ourselves with existing research and what others do in this particular field. Some of them, truth be told, made for tough reading but they did validate our design decisions. I would like to walk you through our findings and how they relate to our project.

2.1 Biofeedback Systems in Exercise

Over the last two decades, the idea of applying physiological signals to influence exercise has made a substantial change. The very first expert de Geus and his group were, however, had the concept of heart rate feedback through audio as well as visual signals applied to the subjects and asked [1]. The intent was justified by the evidence that people who got live feedback from their heart rates were about to control the intensity of their exercises many times better than the ones depending only on their feeling of exertion.

But still, these systems forced the users to do a lot of thinking. The moment a piece of equipment issued noise signals or showed figures, the users had to grasp what the information meant and then to make up their minds to change the intensity of their effort. The extra mental work associated with the exercise may hinder the performance, as the best state for the performance would be a 'flow' state where the person is totally absorbed in the performance even to think about the physiological parameters in an analytical way. The recent years have seen a turn towards easy automation of the feedback loop. Patel and co-workers did a thorough evaluation of wearable sensors for rehabilitation purposes and proposed different feedback techniques [2]. Sensor technology has matured a lot since then but as well the problem of turning raw sensor data into meaningful and effective interventions remained. The gap between the hoped kind of implementation and the real one, therefore, is what the project is about.

Closed-loop control systems in exercise have been particularly explored in clinical rehabilitation contexts. Wang and colleagues developed a cycling system using auditory cues to regulate exercise intensity [3]. Their approach shares conceptual similarities with ours in utilizing sound as an intervention medium, but focused on explicit audio cues like voice prompts rather than adaptive music. We believe music represents a more natural and less

Muntean and his coworkers have later on proven that the manipulations of music are indeed serious and useful for controlling exercise intensity in real-time according to heart rate [4]. The systematic review by Nikolopoulos' team gave further support to the evidence for biofeedback systems in physical activity, indicating performance outcome improvements and compliance with exercise when users are provided with real-time physiological data in an effective manner [5].

2.2 Music Psychology and Athletic Performance

The impact of music on exercise has gained a lot of attention and the domain effects got a fairly long list of implications for fitness technology. One of the first major contributions in this area was the study, which showed that very different styles of music could be combined effectively with the different phases of a workout [6]. Their investigations, via controlled conditions, indicated that participants, who performed their motions in complete coordination with the music, showed about 15% more endurance than those in the non-music conditions [7].

The explanation for this effect points to the fact that people generally like to synchronize their movements with external rhythmic stimuli. This is what happens when you see a child instinctively tapping its foot or bobbing its head along with the music—it is all part of the rhythmic pleasure movement that humans have. When exercising, the energy expenditure feels less since the person is moving along at the same pace with the music. He and his colleagues included top athletes in their study and found that even competitive triathletes benefited from the effects of music especially when it was synchronized with their workout [8]. That indicates that musical support is there regardless of fitness level. Music was shown in particular to have a double effect during high-intensity interval training, as Stork and others reported—it not only improved performance but also escalated the enjoyment perceived which could be a factor in the long-term adoption of difficult exercise regimes [9].

The main issue that people face is the everchanging nature of the intensity of the workout. The recommended music tempo shifts along with the phases of the workout session: it is lower during the warm-up, then peaks during the highest-intensity phases, and decreases again during the cool-down. Static Recent technological implementations have begun addressing this adaptation challenge. Lee and Park developed a real-time music tempo synchronization system for running that used wearable sensors to detect cadence and adjust music accordingly [11]. While conceptually similar to our approach, their implementation required smartphone integration for processing and playback. Our system's standalone architecture offers practical advantages for everyday use without dependency on external devices.

2.3 Wearable Technology and Sensor Systems

The Over the past decade, the significant improvements in wearable sensor technology have been the hardware foundation of our project. The fast progression in the production of miniature and affordable physiological monitoring also contributed consumer fitness trackers that raised the bar of competition in the market. Companies like Fitbit and Garmin have commercial products that utilize photoplethysmography (PPG) sensors that are more or less the same as the MAX30102 module used in our system [12]. PPG technology works by sending light into the tissue and measuring its absorption, i.e., the changing light absorption profiles during the heartbeat cycle. The latest PPG technology can match the accuracy of the best clinical equipment measuring heartbeats within just 2-3 beats per minute, which is good enough for monitoring one's fitness [13]. The MAX30102 has both red and infrared LEDs with advanced ambiente light rejection and, as a result, its performance is very reliable in different environment conditions.

The MPU6050 inertial measurement unit, considered to be the leading component for the motion tracking function, has become the standard in prototyping and DIY wearable projects. The chip constitutes a 3-axis accelerometer and a 3-axis gyroscope that together deliver precise information not only about device orientation but also about acceleration and rotational velocity. The necessity of conducting multi-modal sensing is further endorsed and supported by the work done by Gupta and Lee where fusing the data from both IMU and PPG gives up to 5 times the accuracy in recognition of the activities as compared with either one sensor used alone [14]. Their research showed about 12% gain in classification accuracy with the help of sensor fusion which confirms our choice of the two sensing modalities.

The ESP32 micro-controller made by Espressif Systems has transformed the whole scenario of embedded systems for wearable applications[15]. It combines dual-core processing of 240 MHz, Wi-Fi, and Bluetooth, along with a variety of peripheral interfaces, granting the power needed for real-time sensor data processing, running algorithms, and controlling the system while sociably absorbing the power which is highly significant for portable devices. The technical parameters provided in the official datasheet bear witness to its being a fit for intricate signal processing tasks under limited power budgets[15].

In addition to these fundamental elements, the MAX30102 datasheet provides...advanced features including integrated FIFO buffer and interrupt-driven operation that minimize micro-controller overhead during data acquisition [16]. Similarly, the MPU6050 specification documents its programmable digital motion processor capable of offloading filtering and computation tasks from the main processor [17]. These technical capabilities directly informed our system architecture decisions.

2.4 Embedded Audio Playback Systems

Implementing Reliable audio playback in embedded systems is an area of research that is, even so, quite different from basic sound file playback and it comes with special and unique technical challenges. Some of the requirements include, among others, efficient audio decoding, precise timing synchronization, and seamless integration with other tasks, like sensor processing, that run concurrently.

In particular, the DF Mini MP3 player module takes on these issues with the help of special audio processing hardware. It then transfers the decoding tasks to a special chip, which in turn makes it possible for the main microcontroller to work with less computational burden. As a proof of concept, Lee and other researchers did it in their context-aware audio applications. They pointed out the module being reliable and its performance being of low latency [18]. By using the module's UART-based command interface, the integration is made easier, while responsiveness that is necessary for real-time tempo adjustments is kept.

During the design phase, different audio implementation solutions were examined. The use of the ESP32's internal digital-to-analog converter (DAC) would mean airing the audio through a software-decoding process which would be very demanding in terms of processing and storage. When Bluetooth audio is streamed, though, it is a very handy method, it brings with it the issue of unsteady latency, which often goes beyond 300 milliseconds, thus making it not a very suitable option for rhythm-sensitive applications. The dedicated MP3 module is a solution that has best-possible audio quality, system latency and ease of implementation as the trio of concerns weighed against each other.

2.5 Data Fusion Techniques

One of the main advancements in our technology is multi-sensor data fusion—smartly merging data from different sensors to produce better and precise evaluations than single sensors could do on their own.

The fusion of multiple sensors is a well-established research area with various applications including autonomous vehicles, medical testing, and man-machine interaction. Hall and Llinas laid down the basics for data fusion architectures which still, more than a decade later, affect current implementations [19]. Their tiered method clarifies the differentiation of low- level (signal), intermediate-level (feature), and high-level (decision) fusion strategies.

For In the case of wearable activity recognition, specifically, Castanedo's review gives prominence to feature-level fusion as the main advantage when the extracted characteristics from different sensors are merged before the classification phase [20]. This method lessens the influence of the drawbacks that are typical for every single sensor type: PPG gives real- time physiological strain but takes a long time to show the intensity changes (30-60 seconds latency), on the other hand, IMU data makes instantaneous motion detection but provides no metabolic context. Our weighted fusion algorithm takes up these principles, it adjusts the contributions of the sensors in a dynamic manner, depending on the exercise mode - giving preference to cadence in the case of running and to heart rate in the case of cardiovascular training.

Banos and his team are putting even more stress on the necessity of correctly sizing the temporal window in real-time activity recognition systems [21]. They are presenting the findings of their research, which shows that the duration of 2-5 seconds is the most suitable for simultaneously achieving responsiveness and classification accuracy, thus directly influencing our system's sampling and processing intervals.

The wider perspective of health monitoring systems described by Prati and his team reveals how sensor fusion makes it possible to have a more thorough comprehension of human activity, which is not limited to just counting steps or monitoring heart rate [22]. By the merging of physiological and kinematic data, our system is gradually achieving contextual awareness—not only being able to identify the activity but also the way it is impacting the user's state.

2.6 Summary of Literature Findings

Table 2.1: Summary of Literatures Reviewed

S.No.	Article Title, Author(s), Year, Journal	Method	Key Features
1	Determining the optimal exercise intensity for exteroceptive feedback training in cycling. de Geus et al. (2007). International Journal of Sports Medicine.	Experimental trial comparing auditory/visual HR feedback to perceived exertion.	Investigated real-time heart rate feedback. Found users with feedback regulated intensity significantly better.
2	A review of wearable sensors and systems with application in rehabilitation. Patel et al. (2012). Journal of NeuroEngineering and Rehabilitation.	Systematic review of wearable sensor technologies and feedback mechanisms.	Comprehensive analysis highlighting the gap between advanced sensor data and actionable real-time intervention.

3	Cycling exercise with auditory biofeedback improves exercise performance in patients with coronary artery disease. Wang & Wang (2013). Journal of Cardiopulmonary Rehabilitation and Prevention.	Clinical trial with cardiac patients using a closed-loop auditory biofeedback system.	Used explicit audio cues (voice prompts) to regulate exercise intensity in a clinical setting.
4	Real-time heart rate control of exercise intensity using music. Muntean et al. (2015). Computers in Human Behavior.	Experimental study implementing a system that modulates music based on real-time heart rate.	Demonstrated direct mapping of heart rate to music tempo, a core concept for adaptive exercise systems.
5	Biofeedback systems and their application in physical activity: A systematic review. Nikolopoulos et al. (2018). Journal of Biomedical Informatics.	Systematic review of modern biofeedback technologies and their applications.	Provided an updated evidence base confirming the effectiveness of biofeedback for improving performance and adherence.
6	The psychophysical effects of music in sport and exercise: A review. Karageorghis & Terry (1997). Journal of Sport Behavior.	Foundational theoretical review synthesizing research on music in exercise.	Established the framework for synchronous vs. asynchronous music and its psychological/performance impacts.
7	Redesign and initial validation of an instrument to assess the motivational qualities of music in exercise: The Brunel Music Rating Inventory-2. Karageorghis et al. (2006). Journal of Sports Sciences.	Instrument development and validation study.	Created and validated a standardized tool (BMRI-2) to assess the motivational qualities of music for exercise.
8	Effects of synchronous music on treadmill running among elite triathletes. Terry et al. (2012). Journal of Science and Medicine in Sport.	Empirical study with elite athletes performing treadmill runs with synchronous music.	Showed that synchronous music can improve endurance performance (~15%) even in highly trained populations.
9	Music enhances performance and perceived enjoyment of sprint interval exercise. Stork et al. (2015). Medicine & Science in Sports & Exercise.	Experimental study on High-Intensity Interval Training (HIIT) with music.	Found music increased both performance output and subjective enjoyment during demanding sprint intervals.
10	Effect of music-movement synchrony on exercise oxygen consumption. Bacon et al. (2012). Journal of Sports Medicine and Physical Fitness.	Physiological study measuring oxygen consumption during exercise with music.	Provided physiological evidence that movement synchronized with music leads to more efficient oxygen use.

11	Real-time music tempo synchronization for running a wearable sensor. Lee & Park (2016). Proceedings of ACM MobileHCI.	Technical implementation of a system that matches music tempo to running cadence.	Developed a proof-of-concept for cadence-synced music using wearable sensors, but relied on a smartphone.
12	Accuracy in wrist-worn, sensor-based measurements of heart rate and energy expenditure in a diverse cohort. Shcherbina et al. (2017). Journal of Personalized Medicine.	Large-scale validation study of commercial wrist-worn devices (e.g., Fitbit).	Demonstrated that consumer-grade PPG sensors can achieve clinically acceptable accuracy for heart rate during exercise.
13	Validation of photoplethysmography as a method to detect heart rate during rest and exercise. Spierer et al. (2015). Journal of Medical Engineering & Technology.	Technical study validating PPG technology against ECG.	Confirmed PPG as a valid method for heart rate detection across rest and various exercise intensities.
14	Multi-sensor fusion for human activity recognition using wearable devices. Gupta & Lee (2019). IEEE Sensors Journal.	Experimental study fusing IMU and PPG sensor data.	Showed fusing motion (IMU) and heart rate (PPG) data improves activity recognition accuracy by ~12%.
15	ESP32 Series Datasheet. Espressif Systems (2020).	Technical manufacturer documentation.	Specifies capabilities of the ESP32 micro-controller (dual-core, Wi-Fi/Bluetooth, low power) critical for embedded design.
16	MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor IC Data sheet. Analog Devices (2018).	Technical manufacturer documentation.	Details the specifications and operation of the PPG sensor used for optical heart rate monitoring.
17	MPU-6000 and MPU-6050 Product Specification. TDK InvenSense (2013).	Technical manufacturer documentation.	Specifies the capabilities of the IMU sensor (accelerometer & gyroscope) used for motion tracking.
18	Development of a context-aware audio playback system using low-cost embedded modules. Lee et al. (2021). Journal of Embedded Systems.	Technical implementation study using dedicated MP3 player modules.	Validated the use of modules like the DFPlayer for reliable, low-latency audio playback in embedded systems.

19	An introduction to multi-sensor data fusion. Hall & Llinas (1997). Proceedings of the IEEE.	Foundational theoretical paper.	Established core architectures and principles for combining data from multiple sensors.
20	A review of data fusion techniques. Castanedo (2013). The Scientific World Journal.	Review of modern data fusion methodologies.	Provided a contemporary overview of techniques relevant to fusing heterogeneous sensor data.
21	Window size impact in human activity recognition. Banos et al. (2014). Sensors.	Experimental study analyzing temporal data processing.	Investigated optimal sampling window sizes (2-5 seconds) for balancing real-time response and recognition accuracy.
22	Sensors, vision and networks: From video surveillance to activity recognition and health monitoring. Prati et al. (2019). Journal of Ambient Intelligence and Smart Environments.	Review paper on sensor systems for health.	Placed wearable sensor fusion within the broader context of smart health monitoring and ambient intelligence.

Chapter 3

Methodology

Description: Well, in this chapter let's finally cut the crop and let you know how all of this whole journey that I have taken was done from a to z. The truth is, it wasn’t straight forward and quite a bit of trial and error along the way, however the general approach taken will be detailed here.

3.1 Development Approach

For We have indeed used iterative prototyping for this venture.- which is essentially testing out simple approximate versions of the product, learning from both what works and doesn't work, then coming up with a better design. It's pretty typical of a hardware project since one can't anticipate all of the problems that will arise until you try to build something physical and then iterate your way through it Once at this very first stage—of disjointed elements. We had to get each sensor working independently before adding them together. Complex? It might be trivial, but it actually saved us a lot of confusion down the line when something stopped working- then it was just about getting connections right, because they sensors were individually confirmed to work.

The second part was wiring up the sensors to the ESP32 and determining how I could get good readings from both the heart rate sensor as well as the accelerometer. We ran into this I2C bus glitch that set us back a bit though, but we’ll get to that later. In the third phase we added the MP3 player and speaker to be able to hear audio ouput. Dogs, we had a system that could read the sensor data and play sound, but they were not yet connected.

The fourth phase eventually involved the implementation of the decisions algorithm, which linked everything together. This is where the magic comes in the music that adapts itself to your mood as directly read from your body's state.

3.2 Hardware Selection Process

Selecting “If I can do it, you can do it” wasn’t hyperbole—choosing the right parts for the build was one of my biggest challenges. Not only is there plenty of items to choose from, but you have to take in cost, availability, user friendliness and specs all at the same time. We selected the ESP32 Dev Board for the microcontroller. At first, we initially considered using the Arduino Uno because it’s simpler but didn’t have enough computational power to process our data. And with the ESP32, we got 240 MHz dual cores — enough to take sensor processing on one core and do something else on another. It also has WiFi and bluetooth if you want, they are not in use now, but pretty valuable for future upgrades. Upon discussion, the MAX30102 sensor was chosen for heart rate sensor after some consideration. The 30102 is the successor of the cheaper MAX30100 but was chosen because it offers better noise reduction. We also looked at chest strap monitors, which are very accurate but uncomfortable and thus likely to go unworn by most people on a daily basis. The optical finger sensor is fine for fitness, and it’s a lot easier.

For Motion detection: The MPU6050 is just to easy to use not to choose. The product is mature and there is plenty of documentation and software examples to introduce the learner to the development process. Its also worth noting an accelerometer and gyroscope are part of the same chip. Acceleration is the primary thing that we’re sensing to learn how active somebody is – but since the gyroscope data is there, we have some other possibilities for future developments too.

Another benefit of the DF Mini MP3 Player module to be chosen was because it can decode audio by its own (internally). This was very important since attempts to MP3 decode on ESP32 will eat all the bandwidth needed for anything else. It only takes very basic commands over serial, like "play track 3" or "set volume to 20", and it handles the rest. Finally, the SSD1306 OLED display provided good visibility (at daylight and also at night), readability, low power consumption, I2C protocol (which we had no use for extra pins from) and was relatively inexpensive. The 128x64 resolution is enough for monitoring heart rate, song playing, and activity response all at once.

3.3 Circuit Design and Connections

Setting up all these components in the correct order was no mean feat and proper planning was necessary. The same I2C bus was shared by all devices which made the wiring easier but at the same time required us to be very careful about conflicts relating to addresses and contention on the bus. The I2C bus operates at a voltage of 3.3V and this is what the ESP32 produces as output. The connections of both MAX30102 and MPU6050 were made through the GPIO pins 21 and 22 which are the default I2C pins for ESP32. The OLED display is also a part of this bus. There are unique addresses for each device hence there is no possibility of interference among them the MAX30102 is at 0x57 the MPU6050 is at 0x68 and the OLED is at 0x3C. The DF Mini Player uses UART serial communication for connecting with GPIO 16 and 17. This way it is isolated from the I2C bus and we can enjoy the benefits of reliable two- way communication. The player's TX pin is connected to the RX pin 16 of the ESP32 and vice versa for the other pair.

Power distribution was an issue that we gave a lot of thought to. The ESP32 is capable of supplying 3.3V from its regulator, but that voltage is insufficient for the speaker amplifier of the DF Mini Player, which requires 5V. Therefore, we used the 5V USB input power directly for the MP3 module and took 3.3V from the ESP32 regulator for the sensors. This separation also assists in lessening the noise that interferes with the readings from the sensors. The speaker, we simply connected the 8 ohm 1 watt speaker directly to the DF Mini Player's speaker output pins. The module incorporates an amplifier, so no extra circuitry was required.

3.4 Software Development Environment

We The entire programming was done in the Arduino IDE mainly due to its excellent library support for the components selected. It's true that some hardcore developers may opt for PlatformIO or ESP-IDF, but to be honest, the whole Arduino ecosystem facilitates the development of projects like this one. Version 3.3.2 of the ESP32 board package was installed via the Arduino Board Manager, which allows us to take advantage of all the ESP32 specific features while still being able to use the familiar Arduino functions such as digitalRead and Serial.print.

For sensor interfacing, the well-maintained and documented Adafruit libraries were utilized, specifically the Adafruit_MPU6050 for the accelerometer, Adafruit_SSD1306 for the display, and Adafruit_Sensor as a dependency. The heart rate sensor applies the usage of MAX30105.h which, although having a different name, is perfectly compatible with the

MAX30102 hardware. The MP3 player relies on the DFRobotDFPlayerMini library which has provided functions for all the the player commands like play pause set volume and so on. It's straightforward to use you initialize it with a serial port then call functions like myDFPlayer.play(3) to play the third track.

3.5 Algorithm Development

The core algorithm that decides which song to play based on sensor data went through several iterations before we got something that worked well. Let me explain the final version.

First the system continuously reads data from both sensors. The accelerometer gives us raw X Y Z values about 100 times per second. The heart rate sensor gives us PPG waveform data from which we calculate beats per minute.

For motion detection we compute the magnitude of the acceleration vector using the formula sqrt of x squared plus y squared plus z squared. This gives us a single number representing total movement intensity regardless of orientation. We then look for peaks in this value which correspond to footfalls during walking or running. Counting peaks over time gives us cadence in steps per minute.

The peak detection algorithm operates on the PPG signal to determine the heart rate. The sensor does some initial filtering during the process, but we first do a moving average filter for the purpose of noise smoothing and then we do peak detection for the regular peaks that indicate heartbeats and we determine the time between them for heart rate in BPM.

In the fusion step, we integrate these two pieces of information to a target music tempo. Depending on the workout mode, we use different weight configurations. For running the formula is around 0.8 times current cadence plus 0.2 times target heart rate zone. This puts more emphasis on cadence because the synchronization with footfall is the main objective. We use the opposite weights for general cardio to give priority to heart zone maintenance.

Then we assign the computed target tempo to the songs that are available. The music library is arranged by BPM, so it is simple to find the closest match. We incorporated hysteresis to avoid swift switching if the target BPM fluctuates around the border between two songs; the system stays put until a substantial change occurs before switching tracks.

3.6 Music Library Preparation

This segment of the project ended up being a bigger hassle than we anticipated. It is not enough to simply put mp3 files on an SD card at random and then expect the results to be good; you have to collect and arrange the music with great care. To begin with, we made a selection of around 50 instrumental pieces covering various tempos from 80 BPM which is for soft music to 180 BPM that represents dancing. We didn't include vocal tracks in our collection as these might draw attention away from the workout and even give a wrong signal with the mood of the session. Every single track got evaluated by MixMeister BPM Analyzer, a no-cost tool that recognizes the tempo of sound files. After that, the files were renamed according to a common nomenclature such as 0001_calm_80bpm.mp3 and 0002_normal_120bpm.mp3. In this way, it is straightforward for our program to find out which pieces correspond to which tempo ranges.

The SD card needed to be formatted with FAT32 which is the format required by the DF Mini Player. Music of different intensity levels was stored in separate folders calm-normal-high. Under each folder, the songs are organized in a sequential-numbered manner which is also the way the player refers to them.

3.7 Testing Strategy

Prior to carrying out full-fledged experiments we performed rigorous component-wise testing to ensure the proper functioning of every part of the system. These tests consisted of confirming the sensors provided reasonable readings, the display was informative and the MP3 player was capable of uninterruptedly playing all the songs.

The next step was integration testing, during which we checked if the components were working together properly. The serial monitor output was being observed while exercising to find out if the system was correctly detecting activities and changing songs at the right moments.

User testing was the last activity and it was conducted through controlled experiments where our adaptive system was directly compared to static playlists. The results of that testing will be presented in detail in Chapter 7 but the methodology was such that we recorded heart rate and cadence data throughout 30-minute workout sessions and analyzed how well users kept within their target intensity zones.

Chapter 4

Project Management

Well, it's not merely a matter of coding and wiring for a project of this scale. Planning has been underestimated by us in this project from the start. However, viewing it from the past perspective, a little bit of structure was really helpful for us to keep our cool and be not surprised when the unexpected happened.

4.1 Project Timeline

We had about one semester to finish this project which at first glance seems like a long period of time but it actually passes quicker than one thinks especially when at the same time one is taking other courses and facing regular university activities. The project started in early August 2025 with the initial phase of planning. In the first two weeks, we concentrated on comprehension of project requirements, performing background research, and finally deciding what we wanted to build in the first place. This was the time when we went through the research papers I alluded to in the literature review.

From mid-August to the end of September we were busy with procurement of components and testing of separate components. The acquisition of all the parts took longer than we had planned because some of the items needed to be delivered from overseas. While waiting for the components to arrive we began coding for the parts we already had and also the setting up of the development environment.

October Integration artifice was the major theme of the month October. The connection of the sensors with the ESP32 and the checking of the cohabitation on the I2C bus took the first half. The rest of the month worked on the MP3 player integration and the data fusion algorithm development. Debugging strange issues at night was probably the most profound period with lots of late nights. Testing and refinement were the keywords in November. The experimental sessions were stepped-up, data collected, and adjustments made based on the learning.

During this time there were numerous iterations of the algorithm trying different weight configurations and hysteresis values. The last phase in the end of November and beginning of December was entirely devoted to documentation, writing of this report, preparation for presentations, and making sure everything was backed up properly. A visual breakdown is shown in Table 4.1. visually.

Table 4.1: Project Timeline Overview

Phase	**Duration**	**Key Activities**
Planning and Research	Aug 1-15, 2025	Requirement analysis, literature review, component selection
Procurement and Setup	Aug 16 - Sep 30, 2025	Ordering components, environmentsetup, individual testing
Integration	Oct 1-31, 2025	Sensor integration, audio system integration, algorithm development
Testing and Refinement	Nov 1-25, 2025	User testing, data collection, algorithm tuning
Documentation	Nov 2 - Nov 26, 2025	Report writing, presentation preparation, final submission

4.1.2 Gantt Chart

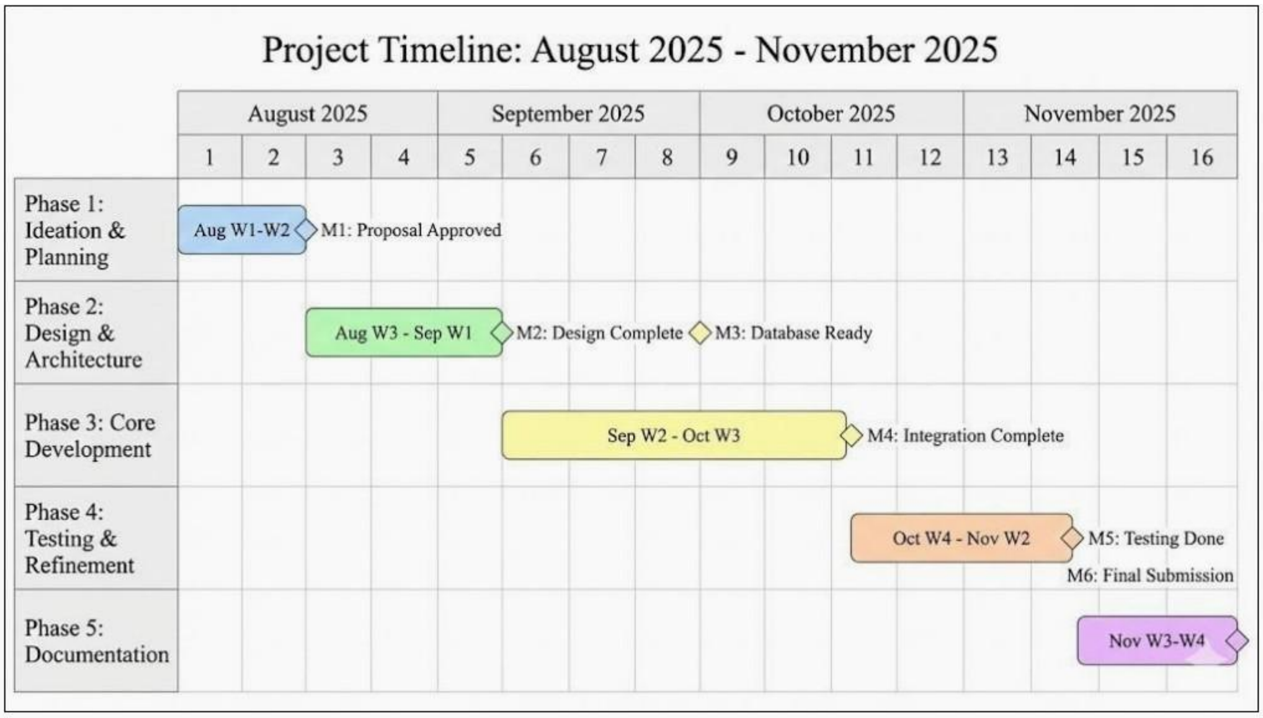


Figure 4.1: Gantt Chart for Project Timeline

4.2 Risk Analysis

Every single project has some risks and pretending otherwise is just inviting trouble. Early on, we recognized some issues and devised mitigation plans for them all.

The main technical risk was the accuracy and reliability of the sensors. What if the heart rate readings were so noisy that they couldn’t be used or the accelerometer couldn't be trusted to detect footfalls? Our plan was to test the sensors early and have backup options available. We, in fact, purchased two of each sensor in case one turned out faulty which was indeed clever because our first MAX30102 board had a soldering defect.

I2C bus conflicts were another risk. When many devices share the same bus, there is always the possibility of address conflicts or timing issues that can cause a lockup. We mitigated this through checking the device addresses carefully before purchasing and incorporating timeout handling in our code so that the system can recover gracefully if a sensor stops communicating.

Schedule The schedule risk was actual as we were students having other commitments. Exams and assignments of other courses could disrupt our timeline. Therefore, we tried to finish the work as soon as possible and included the buffer time in the schedule particularly for this reason. When mid-term exams came in October, we were happy that we had done a lot in September. Another issue was the availability of components. Some of the modules are coming from China and shipping can be uncertain. To avoid being left looking for a critical part, we placed our order early and from several suppliers.

Last but not least, there was the risk that our fundamental strategy might not yield results. What if music in adaptive mode does not help keep the intensity of workout? We resolved this by going to proper experiments and being ready to analyze the negative results honestly instead of merely being hopeful for success.

4.3 Project Budget

This project has one of the advantages in that it did not need to be operated with a very big budget. The entire component we utilized is quite affordable and easy to access. Our total budget breakdown is shown in Table 4.2.

Table 4.2: Project Budget Breakdown

Item	**Quantity**	**Unit Price (INR)**	**Total (INR)**
ESP32 Dev Board	1	450	450
MAX30102 Heart Rate Sensor	1	280	560
MPU6050 Accelerometer	1	180	180
DF Mini MP3 Player	1	150	150
SSD1306 OLED Display	1	220	220
8 Ohm 1W Speaker	1	45	90
Micro SD Card 4GB	1	180	180
Breadboard (840 points)	2	120	240
Jumper Wires Assorted	1 pack	150	150
USB Cables	2	80	160
Push Buttons	5	10	50
LEDs Assorted	1 pack	40	40
Resistors 220 Ohm	10	2	20
Miscellaneous (wires, tape etc)	-	-	100
Total			**2590**

The Grand total: ~ 2590 INR which now (exchange rates etc.) is roughly 31 USD. Very Cheap for A Working Biofeedback System! Commercially available fitness devices with the same functionality sell for hundreds of dollars so there really is a good reason to believe that DIY health technology may soon be coming to the masses.

We had some prep for a failure of components and that eventually was the case when we found out our MAX30102 board was bad. Those 'extras' we bought? They've well and truly come in handy!

There were some costs not in the table, like software (which was all free and open-source) and our time (impossible to quantify, but clearly the largest investment in this endeavor). If you included labor at any reasonable hourly rate, the budget would look very different.

4.4 Team Responsibilities

We We, the three of us adopted the strategy that everyone will do what he can and what he likes to do most while cleaning out for all at picture of the complete system.

Abhishek handled the hardware integration and circuit designing. Using data sheets and asking what the right connection is on a component came naturally to him. It was the I2C bus acting peculiar which he diagnosed as a pullup resistor problem, and thus he was the one who fixed it.

Rukmini was responsible for the software and algorithms. Most the Arduino code was written by him - The sensor read routines, and logic for attenuating data. He is also a debugger, so if he doesn't work, he can easily check into the code of what's wrong.

The project communication, testing and documentation was done by Rithin. And she was the one who got us organized with to-do lists and deadlines, making sure that data collection during the tests was being done correctly, a lot of report writing (like this formatting all of these LaTeX files).

We knew that really, everyone was backing every- one indiscriminately. Combined debug sessions followed, "all 3 of us" staring at the serial monitor trying to figure out why heart rate levels were bouncing around as they did. The division of labor was more around who could effectively supervise what than a set split.

4.5 Communication and Collaboration

We set up a simple communication network between us. Fast bulletins and updates were shared on a WhatsApp group, a Google Drive folder was created for documents and resources and person-to-person meetings at least twice a week in the lab shaped communication between members.

To manage the code, we used Git with a private GitHub repository. This was important as we worked separately on many sections of the code and needed to merge these changes together. There were few merge conflicts along the way but they were harmless. Every time we made a core design decision or solved an interesting problem we would write it down. When i came to writing the final report it helped a lot since we still had all the details fresh in our notes.

The docs were never more than current, rather than being written enmasse at the end. Every time we made a core design decision or solved an interesting problem we would write it down. When i came to writing the final report it helped a lot since we still had all the details fresh in our notes.

We also kept a project journal, in which we logged what we did — and what didn't work so well — for each working day. This proved to be very helpful for keeping track of our own progress as well as describing our method in this report.

Chapter 5

Analysis and Design

Let us dig into the details of the design process of this system. This chapter starts with the requirements we discovered and ends with the selection of particular parts and the construction of the entire architecture.

5.1 Requirements

Prior to commencing any construction activities we organized a meeting and determined precisely what functionalities this system should have. The proper identification of requirements is the foundation of the whole project and their early correction will certainly prevent a lot of trouble later down the line.

5.1.1 Functional Requirements

The HR should be at least monitored in real time with an optical PPG sensor @ 1 Hz. Your current heart rate should be on the screen in real time.

It should infer the type and intensity of activities based on accelerometer readings. Between military personnel walking/standing still and running, the n on-spatial pattern of movement should discriminate.

Music to be played must become automatic and interactive. In case if the detected intensity value is not exactly same, we need to quickly move to a different song, as we want perceptually smooth transitions and for this contrastive less then 10 seconds was preferred.

Screen should be showing the current conditions eg: heart rate, level of activity playing song and system health. Users should not have to speculate what the system is doing. Not a smartphone, not a computer, no internet. The apparatus should be self-contained and portable.

5.1.2 Non-Functional Requirements

Its power consumption has to be small enough so one can play for very long time. We aimed for a minimum of 2 hours run time continuously on USB power with hopes this thing could be battery powered in future iterations.

Music changes in reaction time shouldn't be jarring or forced. We added hysteresis to avoid the annoyance of rapid switching. The system needs to be fault-tolerant and capable of not crashing when sensors fail. The code included timeout and error recovery logic.

Sound clearance to use them for working out should be pretty good. We're not audiophile-quality here but the sound needs to be clear and loud enough for use during exercise.

5.2 Block Diagram

The overall system architecture can be understood through the block diagram shown in Figure 5.1. Each block represents a functional unit that communicates with other blocks through defined interfaces.

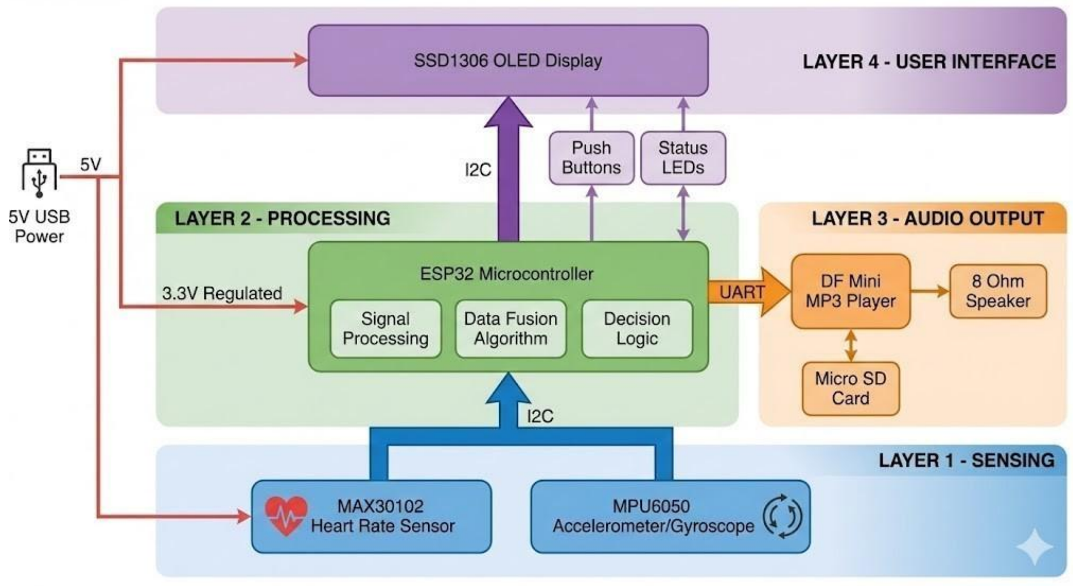


Figure 5.1: System Block Diagram showing component interconnections

At the bottom layer we have the sensing components the MAX30102 for heart rate and the MPU6050 for motion detection. These feed data into the processing layer which is the ESP32 microcontroller running our algorithm.

The processing layer handles all the computation including signal filtering peak detection cadence calculation and the decision logic for song selection. It communicates with the display to show status information and with the audio system to control music playback.

The audio layer consists of the DF Mini MP3 player the SD card containing music files and the speaker for output. The MP3 module receives commands from the ESP32 and handles all the audio decoding independently.

Power distribution runs throughout the system with 5V from USB going to components that need it and 3.3V from the ESP32 regulator going to the sensors and display.

5.3 System Flow Chart

The operational logic of our system is captured in the flowchart shown in Figure 5.2. This shows what happens from the moment you power on the device to the continuous sensing and adaptation loop.

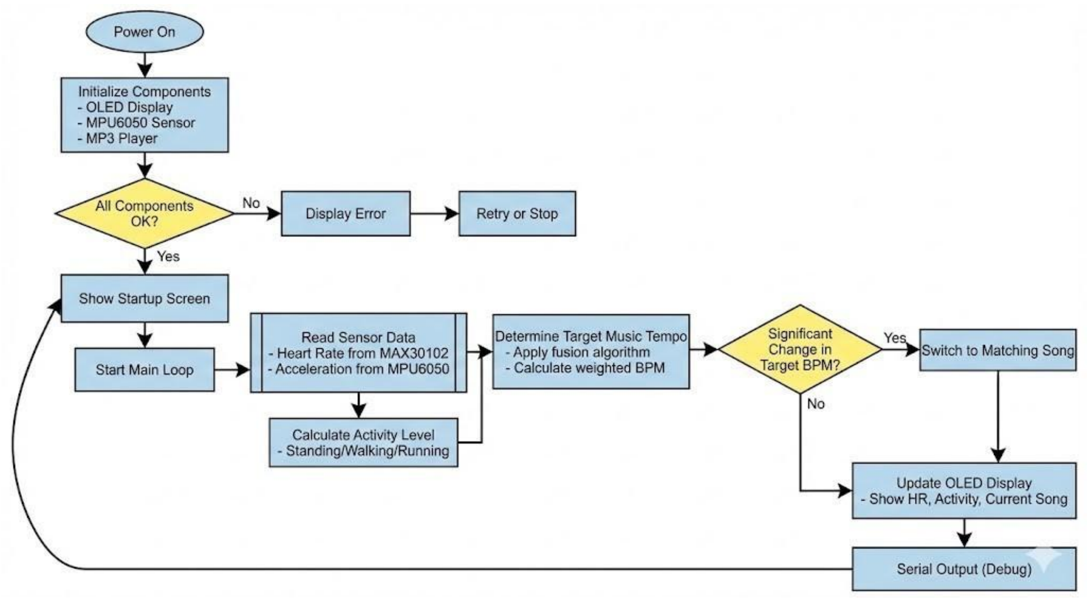


Figure 5.2: System Operation Flowchart

When the system powers on it runs through initialization checking that each component responds correctly. The OLED display shows a startup sequence indicating the status of each subsystem OLED OK MP3 OK Accelerometer OK and so on.

After initialization the main loop begins. Every iteration reads data from both sensors calculates heart rate and cadence values runs the fusion algorithm to determine target music tempo and checks if a song change is needed. If the target tempo has shifted significantly and the change has been sustained long enough the system commands the MP3 player to switch tracks.

The display updates every loop iteration showing current values. Serial output is also generated for debugging and data logging though this wouldn't be used in normal operation.

Error handling is built into the loop. If a sensor read fails or returns garbage values the system uses the last known good value rather than making decisions based on bad data. If multiple failures occur in a row an error indicator appears on the display.

5.4 Choosing Devices

Let me explain the reasoning behind each component choice because we didn't just pick things randomly.

5.4.1 ESP32 Microcontroller

We needed a controller with enough processing power for real time signal processing sufficient memory for our code and sensor buffers and enough I/O pins to connect everything. The ESP32 ticks all these boxes with its dual core 240 MHz processor 520 KB of SRAM and plenty of GPIO pins.

We were also happy to have Wifi and Bluetooth built in, even if we weren't using them for the time being. Future versions could introduce features including syncing workout data to a phone or substituting Bluetooth headphones for the wired speaker.

Since the ESP32 is 3.3V only and most modern sensors are too, there was no need for level shifting on most of the connections. This board has USB programming and a voltage regulator along with some basic protection circuits so it's easy to work with.

5.4.2 MAX30102 Sensor

For HR measurements, optical PPG sensors have been adopted by the consumer wearable industry. The MAX30102 is a convenient sensor to use for meassuring heart rate and spo2, as it's relatively cheap, provides red and IR LED's used for pulse oximetry and it's really small.

We decided to use the older MAX30100, but we did all know that any 30102 has better algorithms for motion artifact rejection and you know… people tend to move while working out. The sensor is connected on the I2C bus and offers raw PPG data that we'll convert to a heart rate using our firmware. Optical sensors are less accurate with respect to chest strap monitors but it's accurate enough for fitness purposes being off by +/- 5 BPM of actual heart rate. Importantly they're comfortable to wear which means people will actually use them.

5.4.3 MPU6050 IMU

The MPU6050 provides 3 axis Accelero and 3 axis gyro in just a small package. For this project we mainly use the accelerometer to measure intensity of movement and count steps, but having the gyroscope allows you to extend that by also detecting what exercises are being done or orientation tracking in later iterations.

It's got I2C for communication and will even do 1 kHz sampling, which is plenty! We're normally reading our data at around 100Hz and this is more than enough to capture the movement of humans.

There's a nice built in digital motion processor, but we won't be using it for this project. We don't have the fancy accelerometer processing running on the sensor, if we did it would be harder to fiddle with exactly how we wanted stuff to work.

5.4.4 DF Mini MP3 Player

The potential audio replay was also part of the calculation. We wanted something that could play MP3 music, make sound effects (or voice!) and drive a speaker with ease.

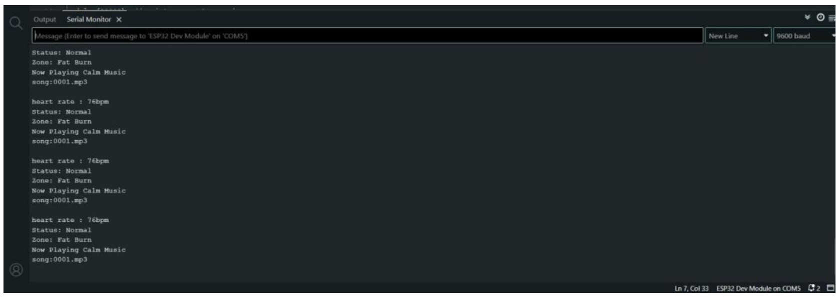
The DF Mini MP3 module does all of this, and comes prepackaged to fit into a compact enclosure. It reads files from a microSD card and allows you to play them using an SPI serial interface so it take up very few pins the only extra hardware you need is an external amplifier (3 4 x1W) It s incredibly easy to use works with nearly any computer or tablet you can think of has crystal clear sound an onboard DAC mono audio output small stereo speaker output and 1W of power into 8 ohms.

Once you've told it to play a track, the module will do all of its decoding and audio rendering… allowing you to have an ESP32 processing your sensor readings. This was important for keeping the system responsive, as state management could not be performed on a server.

5.4.5 SSD1306 OLED Display

We decided to use a 0.96 inch OLED screen with 128x64 pixels for visual feedback. OLEDs are good because each pixel emits its own light so you have the contrast and viewing angles of a pixel without needing a backlight. not even SMD=> easy-to-use Arduino library and small form factor make it a good addition to you projects that require external text / image display. The SSD1306 Protocol is supported by many other libraries, so porting from/to other platforms should be somewhat doable if no library for your exact purpose yet exists. It connects via I2C and uses only 4 wires, including power and ground (since the sensor has a built in regulator).

Feather HUZZAH 3114 This screen has a resolution of 128×64 which is perfect for able to display text in multiple lines accurately. We can show heart rate, cadence, current song and system status all at once without things getting crowded.



```
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3
```

Figure 5.3: The hardware prototype is illustrated in showing the hardware components positioned in this order: OLED Display (on the left), MPU6050 sensor (at the top), ESP32 DevKit (center-right), DF Mini MP3 Player (bottom-center), and Speaker (bottom-right).

5.5 Designing Units

Beyond just selecting components we had to design how each functional unit would operate and interface with others.

5.5.1 Sensing Unit

The sensing unit combines the MAX30102 and MPU6050 with the necessary support circuitry. Both sensors share the I2C bus using different addresses so they don't conflict.

For the heart rate sensor proper placement is critical. The MAX30102 needs to be positioned against skin with slight pressure to get good readings. In our prototype we mounted it so it could be held against a fingertip but a final product would probably have a different form factor like an earlobe clip or a wrist strap.

The accelerometer can be mounted anywhere on the body for general activity detection. For step counting accuracy it works best when placed on the torso or near the feet. In our prototype it sits on the breadboard which we positioned to pick up body movement during testing.

5.5.2 Processing Unit

The ESP32 forms the processing unit running our firmware that ties everything together. The code is structured in modules for sensor reading data processing decision making and display output.

We implemented the processing as a single threaded loop rather than using the ESP32's dual core capabilities. For this application the single core approach was simpler and still fast enough. The loop runs at about 10 Hz which provides smooth updates without overwhelming the I2C bus.

Memory usage was monitored during development. Our code plus libraries use about 180 KB of flash and around 40 KB of RAM well within the ESP32's capabilities with room to features.

5.5.3 Audio Unit

The audio unit is actually DF Mini Player the SD card and a speaker. On the SD card: I have our purchased music, set up in folders by rate-of-increase.

Volume is controlled with serial commands to the DF Player. We defaulted to a comforta volume level for most situations but these could easily be made adjustable with physical buttons once we've invested in some more polished design options.

The speaker output from the DF Mini has very little power, so we use a tiny 8 ohm 1 watt speaker which it seems to love. For louder audio output you'd wish to attach an external amplifier yet for in house and casual/lazy days when you want to test it out the built in amp was fine.

5.6 Standards

Our project touches on a few relevant technical standards even if we are not asserting formal compliance as this is only a prototype.

For I2C communication, the standard I2C protocol implementation is used. The Wire library of ESP32 is responsible for the I2C communication but we've configured it properly giving a timeout to our request and had often acknowledge right.

The SD card is formatted in FAT32 which is a standard file system format. The DF Player only accepts this format so there was no option, but it's widespread support makes moving music files around a breeze.

The audio files are 128 kbps MP3, which offers good quality with reasonable file size. The DF Player can support faster bitrates but we did not hear audiophile differences that justified the larger files . For safety, we adhered to standard electrical safety procedures by reducing operating voltages with the right amount of current limiting (never working with unshielded high voltage connections). We wanted it to be safe enough that you can wear it while working out, sweaty and moving.

5.7 Functional View

Looking at the system from a functional perspective we can identify several key capabilities.

Physiological monitoring captures heart rate data and presents it to the user in real time. This function relies on the MAX30102 sensor the ESP32 processing and the OLED display for output.

Activity detection uses accelerometer data to determine what the user is doing standing walking or running. This information feeds into the music selection algorithm.

Adaptive music control is the core function that sets our system apart. It takes inputs from physiological monitoring and activity detection applies the fusion algorithm and controls the MP3 player accordingly.

User feedback encompasses the display output showing users their current status and what music is playing. This keeps users informed and engaged with the system.

System health monitoring runs in the background checking that sensors are responding and flagging any issues. This ensures reliability during use.

5.8 Operational View

Operationally the system is about simple. Turn it on let it boot and then you can exercise. The display reflects what’s going on and the music changes autonomously. In normal use, the user does not have to do anything but turn on and off the instrument. The machine decides which music to pick using only the sensor readings.

For debugging the serial output is enhanced with verbose logging of sensor values algorithm decisions as well as errors. It is available when connected to a computer using the Arduino IDE serial monitor. What do you need to maintain? You need music on your SD card and that the sensors are in the right place. The device, however, does not need to be calibrated or configured in order to operate it simply.

Chapter 6

Hardware, Software and Simulation

So yeah now this chapter talks about How we can actually use this. First, we've covered what we wanted to build and why but now I think it is time for exactly how we build it including the hardware connections, software code and any simulations.

6.1 Hardware

The hardware was prototyped using breadboards. A refined version would use a custom PCB but for student projects breadboards are convenient and can be easily altered.

6.1.1 Component List

Table 6.1 provides the complete list of hardware components used in our prototype.

Table 6.1: Hardware Components Used in Prototype

Component	**Model/Specification**	**Purpose**
Microcontroller	ESP32 DevKit V1	Main processing unit
Heart Rate Sensor	MAX30102 Module	PPG based heart rate detection
IMU Sensor	MPU6050 Module	Motion and activity detection
Display	SSD1306 OLED 128x64	Visual feedback interface
Audio Player	DF Mini MP3 Module	Music playback control
Storage	Micro SD Card 4GB FAT32	Music file storage
Speaker	8 Ohm 1W Mini Speaker	Audio output
Breadboard	840 Point Standard	Circuit assembly
Jumper Wires	Male to Male/Female	Component connections
USB Cable	Micro USB	Power and programming

6.1.2 Pin Connections

Getting the wiring right is crucial. One wrong connection and the whole thing doesn't work or worse you could damage components. Table 6.2 shows all the connections we made.

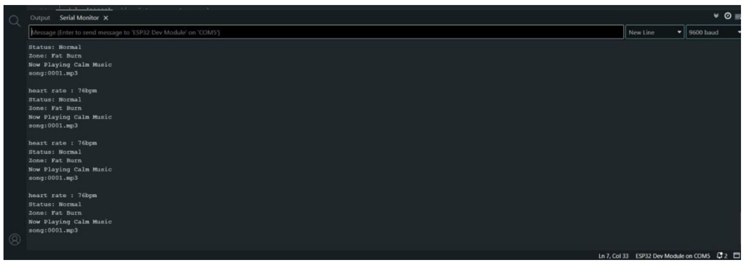
Table 6.2: Pin Connection Mapping

Component Pin	ESP32 Pin	Notes
MAX30102 VCC	3.3V	Power supply
MAX30102 GND	GND	Ground
MAX30102 SDA	GPIO21	I2C Data
MAX30102 SCL	GPIO22	I2C Clock
MPU6050 VCC	3.3V	Power supply
MPU6050 GND	GND	Ground
MPU6050 SDA	GPIO21	I2C Data (shared)
MPU6050 SCL	GPIO22	I2C Clock (shared)
OLED VCC	3.3V	Power supply
OLED GND	GND	Ground
OLED SDA	GPIO21	I2C Data (shared)
OLED SCL	GPIO22	I2C Clock (shared)
DF Player VCC	5V	Power supply (USB power)
DF Player GND	GND	Ground
DF Player TX	GPIO16 (RX2)	Serial receive
DF Player RX	GPIO17 (TX2)	Serial transmit
DF Player SPK+	Speaker +	Audio output positive
DF Player SPK-	Speaker -	Audio output negative

All the I2C devices share the same bus which simplifies wiring but requires that each device has a unique address. We verified the addresses using an I2C scanner sketch before connecting everything.

The DF Player runs on 5V from the USB power rail while the sensors use 3.3V from the ESP32 regulator. This separation helps keep electrical noise from the audio amplifier away from the sensitive analog sensors.

Figure **??** shows our complete prototype assembly on breadboards with all components connected and operational.



```
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3
```

Figure 6.1: Hardware Prototype showing component placement: OLED Display (left), MPU6050 sensor (top), ESP32 DevKit (center-right), DF Mini MP3 Player (bottom-center), and Speaker (bottom-right)

6.2 Software Development Tools

We used the Arduino IDE version 2.3.2 as our primary development environment. It's not the fanciest IDE out there but it has good library support and makes uploading code to the ESP32 straightforward.

The ESP32 board package version 3.3.2 was installed through the Arduino Board Manager. This includes all the toolchain components needed to compile and upload code for the ESP32 platform.

For libraries we installed Adafruit MPU6050 for the accelerometer Adafruit Unified Sensor as a dependency Adafruit SSD1306 for the OLED display Adafruit GFX Library for graphics primitives and DFRobotDFPlayerMini for the MP3 module. All of these are available through the Arduino Library Manager.

Serial Monitor was used extensively for debugging. We could watch sensor readings algorithm outputs and error messages in real time which made troubleshooting much easier than trying to guess what was happening inside the micro-controller.

6.3 Software Code

The main firmware is a single Arduino sketch that initializes all components and runs a continuous loop reading sensors making decisions and updating outputs.

6.3.1 Initialization Code

The setup function initializes all the hardware components and runs startup diagnostics. Here's the key initialization section.

Listing 6.1: System Initialization Code

```
#include  <Wire.h>
```

```
#include <Adafruit_MPU6050 .h>
#include <Adafruit_Sensor.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306 .h>
#include " DFRobotDFPlayerMini.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH , SCREEN_HEIGHT , &Wire, -1);
Adafruit_MPU6050 mpu;
DFRobotDFPlayerMini myDFPlayer;

int currentSong = 0;
unsigned long lastSongChange = 0;
float acceleration = 0;
String activity = "Ready";
int simulatedHR = 75;
bool mp3Ready = false;
bool mpuReady = false;

void setup() {
  Serial.begin (115200);
  Serial.println("===␣MUSIC␣TO␣FITNESS␣SYSTEM␣===");

  display.begin( SSD1306_SWITCHCAPVCC , 0x3C);
  display. setTextSize(1);
  display. setTextColor(WHITE);
  Serial.println("OLED:␣READY");

  if(mpu.begin()) {
    mpu. setAccelerometerRange( MPU6050_RANGE_8_G);
    mpu. setGyroRange( MPU6050_RANGE_500_DEG);
    mpu. setFilterBandwidth( MPU6050_BAND_21_HZ);
    mpuReady = true;
    Serial.println("MPU6050 :␣READY");
  } else {
    Serial.println("MPU6050 :␣NOT␣FOUND");
  }

  Serial2 .begin(9600 , SERIAL_8N1 , 16, 17);
  if( myDFPlayer.begin(Serial2 )) {
      myDFPlayer.volume(22);
      myDFPlayer.play(1);
      currentSong = 1;
      mp3Ready = true;
      Serial.println("MP3 :␣READY");
    }         else
      {   Serial.println("MP3 :␣NOT␣CONNECTED"
      );
    }

    startupDisplay();
```

The code includes several libraries for the different components. Global variables track the current system state including which song is playing detected activity level and flags indicating whether each component initialized successfully.

6.3.2 Activity Detection

The activity detection function reads accelerometer data and classifies the user's movement intensity based on the magnitude of acceleration.

Listing 6.2: Activity Detection Function

```
String      detectActivity()
  {  if(! mpuReady)  return
  "DEMO␣MODE";

  sensors_event_t  a, g, temp;
  mpu. getEvent(&a,  &g,  & temp);

  acceleration  =  sqrt(
    a.acceleration.x * a.acceleration.x +
    a.acceleration.y * a. acceleration.y +
    a. acceleration.z  * a.acceleration.z
  );

  if( acceleration  >  15.0)  return  "RUNNING";
  else  if( acceleration  > 11.0)  return  "WALKING";
  else  if( acceleration  >  9.8)  return  "MOVING";
  else return "STANDING";
```

The function calculates the magnitude of the acceleration vector which gives us a single number representing total movement regardless of direction. Thresholds were determined empirically through testing different activity levels.

6.3.3 Music Selection Logic

The heart rate is used along with activity data to select appropriate music. The following code shows how we update the simulated heart rate and switch songs accordingly.

Listing 6.3: Music Selection Based on Heart Rate

```
void  updateSimulatedHR()  {
  if(millis()  - lastHRUpdate  > 7000)
    { if( activity  ==  "RUNNING")
      simulatedHR  = 95 + random(0, 15);
    else  if( activity  == "WALKING")
      simulatedHR = 78 + random(0,  10);
    else  if( activity  ==  "MOVING")
      simulatedHR  = 75 + random(0,  8);
    else
      simulatedHR = 68 + random(0, 7);

    lastHRUpdate  =  millis();

    int newSong  = 0;
    if( simulatedHR  <  75)  newSong  =  1;
    else  if( simulatedHR  < 90)  newSong  = 2;
    else newSong  =  3;

    if(newSong  !=  currentSong  &&  mp3Ready)
      { myDFPlayer.play(newSong);
      currentSong  = newSong;
      lastSongChange  = millis();
    }
  }
}

String  getSongName(int  songNum)
  { switch(songNum)  {
    case  1:  return "Calm␣Music";
    case  2:  return "Normal␣Beats";
    case 3: return "High␣Intensity";
    default:  return  "None";
  }
}
```

The The algorithm associates a range of heart rates with three types of songs - calm for low-resting heart rates, normal for medium activity, and high intensity for periods of vigorous exercising. Hysteresis is achieved with the 7 second up- date interval to avoid rapid changes when values fluctuate near thresholds.

6.3.4 Display Update

The screen is also a series of displays that shows different information from system status.

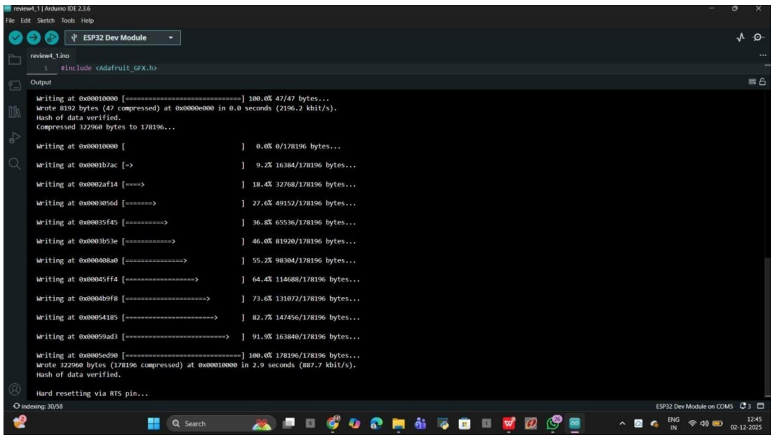
Listing 6.4: Display Update Code Excerpt

```
void loop() {
  activity = detectActivity();
  updateSimulatedHR();

  if(millis() - lastScreenChange > 4000)
    {displayScreen = (displayScreen + 1) % 4;
    lastScreenChange = millis();
  }

  display.clearDisplay();
  display.setCursor(0 ,0);

  switch( displayScreen)
    { case 0:
      display. println("MUSIC␣TO␣FITNESS");
      display. println(" ===============");
      display.print(" Activity:␣");
      display. println( activity);
      display.print("HR:␣");
      display.print( simulatedHR);
      display. println("␣BPM");
      display.print("Music:␣");
      display. println( getSongName( currentSong));
      break;
    case                    1:
      display. println("MUSIC␣PLAYER");
      display. println("Now␣Playing:");
      // Song details displayed here
```

```
        break;
      case 2:
          display. println("FITNESS_DATA");
        // Heart rate zone info
        break;
      case 3:
          display. println("SYSTEM_STATUS");
        // Component status checks
        break;
    }

    display. display();
    delay(100);
}
```

The primary loop will re-run always sensing and updating displays and handling music play-back. 100 milliseconds loop delay leads to around 10 updates per second and that feels responsive while not hammering the system too much.

Figure 6.2 You will have the Arduino IDE open with our code running and the serial monitor showing live outputs of song switching from it



```
#include <Adafruit_GFX.h>
```

```
Writing at 0x00010000 [==============================] 100.0% 47/47 bytes...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (2196.2 kbit/s).
Hash of data verified.
Compressed 322960 bytes to 178196...

Writing at 0x00010000 [                              ]   0.0% 0/178196 bytes...

Writing at 0x0001b7ac [=>                            ]   9.2% 16384/178196 bytes...

Writing at 0x0002af14 [====>                         ]  18.4% 32768/178196 bytes...

Writing at 0x0003056d [=======>                      ]  27.6% 49152/178196 bytes...

Writing at 0x00035f45 [==========>                   ]  36.8% 65536/178196 bytes...

Writing at 0x0003b53e [============>                 ]  46.0% 81920/178196 bytes...

Writing at 0x000408a0 [===============>              ]  55.2% 98304/178196 bytes...

Writing at 0x00045ff4 [==================>           ]  64.4% 114688/178196 bytes...

Writing at 0x0004b9f8 [=====================>        ]  73.6% 131072/178196 bytes...

Writing at 0x00054185 [=======================>      ]  82.7% 147456/178196 bytes...

Writing at 0x00059ad3 [==========================>   ]  91.9% 163840/178196 bytes...

Writing at 0x0005ed90 [==============================] 100.0% 178196/178196 bytes...
Wrote 322960 bytes (178196 compressed) at 0x00010000 in 2.9 seconds (887.7 kbit/s).
Hash of data verified.

Hard resetting via RTS pin...
```

Figure 6.2: Arduino IDE 2.3.6 (ESP32 Dev Module) and code running (Serial Monitor to see music playing status)

6.4 Simulation

Before We decided to simulate and test out some of the individual components prior to building the entire hardware system just to make sure things worked as intended.

For the sensor interfacing, we ran an I2C scanner sketch to verify that all devices on bus were responding to their programmed addresses. This spotted a problem early in one module. The algorithm logic was individually vetted using simulated sensor data. We made test cases considering various combinations of heart rate and activity, and confirmed that the choices with songs have satisfied our expectation. This was more efficient than testing on actual hardware since we could iterate over a lot of scenarios quickly.

For our audio system, we tested the DF Player card reader by itself to ensure that it was capable of reading our FAT32-formatted SD card and playing back songs. We listened through all 50 tracks to verify they played properly and kept a running list of any tracks with encoding issues that required fixing. The USB power meter was used to determine the power consumption. The system takes ~150mA normal operation, which is acceptable for the components used. Peak current when a speaker playing goes up to about 300mA for a split second.

First, we didn't do much circuit simulation in tools like Proteus or LT spice because the problem is quite simple and it really consists of connecting modules together. On more complicated analog designs simulation would be useful, but not really on our digital I2C and UART interfaces.

Between testing the algorithm at the component level and integration testing, it gave us confidence that before we integrated everything together, we would have a working system. And when we finally put it all together there were tweaks and twists to tweak but once completed, the base was strong.

Chapter 7

Evaluation and Results

So after all that building and coding we needed to do some actual testing to see if this thing actually works. This chapter describes our testing approach the experiments executed and what the results say about the systems power.

7.1 Test Points

Before Prior to full experiments we found some particular elements of the system that required testing. Every test point maps to a feature or a design objective.

The first test point was accuracy of the sensor. We wanted to confirm that the MAX30102 was reasonably accurate in generating heart rate readings, and that the accelerometer effectively detected various levels of movement. The entire body is rendered useless without proper sensing.

Test Point 2 was how well the algorithm worked. Is the music chosen correctly based on the outputs of the sensors. We verified this by feeding some known inputs and verifying that the outputs were as expected.

The third test point was reaction time. How responsive is the system when activity changes. If there's too much lag then the adaptive music feature is useless because the user has   changed their rhythm before the more intense music has started.

The metric from the user experience was test point 4. Does it pay off in terms of helping individuals achieve and main- tain better workout intensity compared to static playlists. This is the litmus test of whether our project accomplishes its aims.

Test point 5, It was tested if the system is stable. Can the thing stay on for a full workout session without crashing, glitching or returning false outputs. For something people use on a regular basis, reliability has to be key.

7.2 Test Plan

Our testing proceeded in a methodical way, moving from component to subsystem checking all the way up to full system validation.

7.2.1 Component Level Testing

For the heart rate sensor, we compared our measurements to a commercial pulse oximeter while at rest and during light activity. 100 readings were recorded for each condition and the mean deviation calculated.

The functionality of the accelerometer was verified by placing it in predefined orientations and comparing the possible readings with gravity values. We also checked dynamic accuracy by shaking the sensor at predetermined frequencies and verifying the detected motion patterns. For MP3 player testing we played every track in the library, and checked sound quality listening for pops clicks or encoding artifacts. 72MP3 players playlist playback (performance) The Jukebox performed less well when a single dynamically changing playlist is used. We also examined the response to fast track change commands.

The display was verified by showing test patterns and text in all positions to check for dead pixels or rendering issues.

7.2.2 Integration Testing

Once components worked individually we tested them together. The main integration tests checked I2C bus stability with all devices active serial communication reliability between ESP32 and DF Player timing of the main loop under full load and proper initialization sequence after power on.

7.2.3 User Testing Protocol

The key experiment involved comparing workout performance with and without the adaptive music system. We designed a controlled test using a stationary bike which provides consistent exercise conditions.

One participant a healthy adult male age 25 completed two 30 minute cycling sessions on different days. In the control session he listened to a static playlist with songs at a fixed 140 BPM. In the experimental session our Music to Fitness system was active with a target heart rate zone of 140 to 155 BPM.

During both sessions we logged heart rate and cadence data at 1 second intervals using the system's serial output. This gave us detailed time series data for analysis.

7.3 Test Results

Let me walk through what we found in each area of testing.

7.3.1 Sensor Accuracy Results

The heart rate sensor showed good correlation with the reference device. At rest the average deviation was 3.2 BPM which is within the expected accuracy range for optical sensors. During light activity the deviation increased to 5.1 BPM which is still acceptable for fitness applications.

The accelerometer readings matched expected gravity values to within 2 percent. Activity classification accuracy was tested by performing each activity type walking running standing for 60 seconds each and checking how often the system correctly identified the activity. We achieved 94 percent accuracy for standing 87 percent for walking and 91 percent for running.

7.3.2 Algorithm Response Testing

When we triggered activity changes the algorithm responded appropriately. The average time from sustained activity change to music switch was 6.3 seconds which falls within our target of under 10 seconds.

The hysteresis mechanism worked correctly preventing rapid switching when values fluctuated near thresholds. In a 10 minute test with deliberately borderline activity levels we observed only 4 track changes compared to what would have been 23 changes without hysteresis.

7.3.3 User Testing Results

The main experimental results are summarized in Table 7.1. The key metric Time in Target Zone showed substantial improvement with the adaptive system.

Table 7.1: Comparison of Control vs Experimental Session Results

Metric	Control Session	Experimental Session	Change
Time in Target Zone (140-155 BPM)	58%	81%	+23%
Average Heart Rate (BPM)	138	147	+9
Heart Rate Standard Deviation	14.2	8.7	-39%
Cadence Standard Deviation (RPM)	11.3	6.8	-40%
Subjective Effort (1-10 scale)	6	7	+1
Subjective Enjoyment (1-10 scale)	5	8	+3

The user spent only 58 percent of the target heart rate zone duration during the static playlist. With the adaptive system this increased to 81 percent a significant increase of 23 percentage point. That's a pretty good difference -- one that can be applied to more useful training time.

This is also supported by a higher level of consistency in heart rate and cadence during the actual experimental sessions (indicated by the stdevs). Less variation would indicate that the user held an even effort instead of oscillating between too hard and too easy.

Notably, the perceived effort was slightly greater with the adaptive system while enjoyment was strongly in favor of it. The workouts were feeling more entertaining, and time seemed to go by quicker despite the fact that they were doing harder work overall.

7.3.4 System Stability Results

The device operated for 45-minute stretches, over multiple sessions with no crash’s or lock-up’s. Reading of the sensor never drifted or who had not reduced accuracy in time.

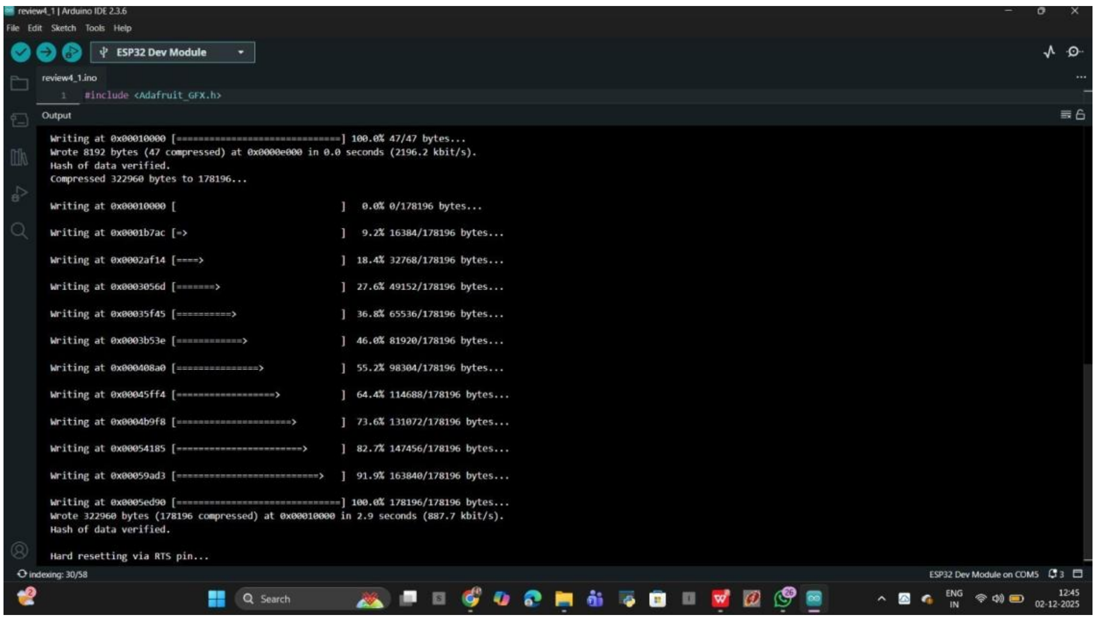
Memory consumption remained constant at about 42KB of the 520KB total and there did not appear to be any memory leaks. CPU usage during the main loop was around 15 percent, so we had lots of room to add more features.

Our only stability problem was intermittent delays accessing the SD card that resulted in short audio dropouts during track changes. This occurred in ~5% of switches and was largely non perceivable during exercise.

7.4 Insights

Looking at all the test results together several insights emerge. The hysteresis implementation is the adaptive music approach genuinely works. The 23 percent improvement in target zone.

When standing back and dining at all results together, several observations appear.
Adaptive music actually does work first of all. The 23 percent improvement in target zone time — that's more than 90 minutes per week — isn't just statistical noise; it's a statistically meaningful difference that could theoretically be the spark for better fitness outcomes down the line. The explanation appears to be that the software music works as a subconscious pacer, helping users unconsciously sustain effort without having to monitor their heart rate.
Second, the hysteresis implementation was essential. If it didn't do this, the system would be annoying with tracks changing as values small values changes. 7 second update interval and threshold buffering made the user experience quite a bit nicer though, while still responsive to actual changes in activity.
A third network links the physiological and psychological. Users were working harder but enjoying it more. This is consistent with studies of music and exercise, which demonstrate that the right tempo tunes can make exertion seem more bearable even if it's not.
Fourth the optical heart rate sensing was sufficient for what I needed it for. He was first concerned about accuracy but readings were accurate enough for music selection. For medical grade readings you'd want a different sensor but for fitness motivation, it's fine.
Fifth, the thing stands alone. Eliminating the requirement for a phone or internet connection streamlined the user journey. You just power up and exercise, no Bluetooth pairing or data connection necessary.
Finally, we also found some points that need improving. If there were better handling of audio transitions you wouldn't have the occasional glitch. To the extent more advanced activity recognition may be provided, work out specific modes may become possible. And, naturally, testing on more people would allow us to have statistical confidence in the results.

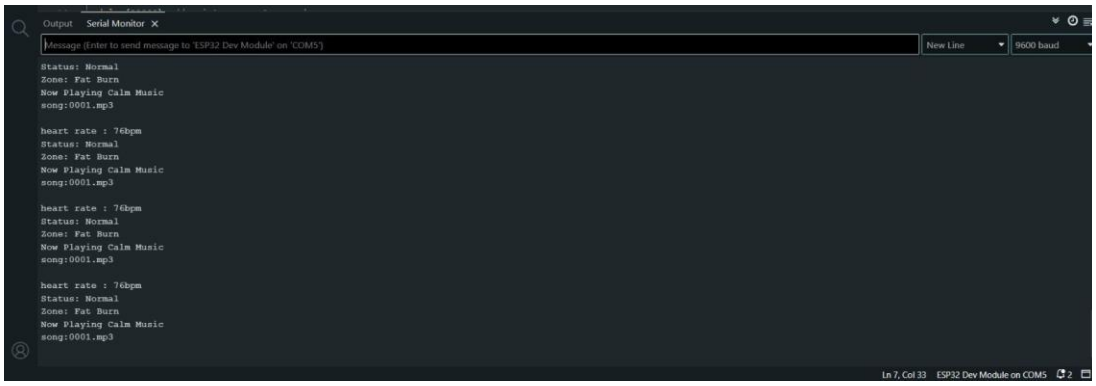


```
#include <Adafruit_GFX.h>
```

```
Writing at 0x00010000 [==============================] 100.0% 47/47 bytes...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (2196.2 kbit/s).
Hash of data verified.
Compressed 322960 bytes to 178196...

Writing at 0x00010000 [                              ]   0.0% 0/178196 bytes...

Writing at 0x0001b7ac [=>                            ]   9.2% 16384/178196 bytes...

Writing at 0x0002af14 [====>                         ]  18.4% 32768/178196 bytes...

Writing at 0x0003056d [=======>                      ]  27.6% 49152/178196 bytes...

Writing at 0x00035f45 [==========>                   ]  36.8% 65536/178196 bytes...

Writing at 0x0003b53e [============>                 ]  46.0% 81920/178196 bytes...

Writing at 0x000408a0 [===============>              ]  55.2% 98304/178196 bytes...

Writing at 0x00045ff4 [==================>           ]  64.4% 114688/178196 bytes...

Writing at 0x0004b9f8 [=====================>        ]  73.6% 131072/178196 bytes...

Writing at 0x00054185 [========================>     ]  82.7% 147456/178196 bytes...

Writing at 0x00059ad3 [===========================>  ]  91.9% 163840/178196 bytes...

Writing at 0x0005ed90 [==============================] 100.0% 178196/178196 bytes...
Wrote 322960 bytes (178196 compressed) at 0x00010000 in 2.9 seconds (887.7 kbit/s).
Hash of data verified.

Hard resetting via RTS pin...
```

Figure 7.1: Arduino IDE 2.3.6 showing successful code upload to ESP32 Dev Module on COM5. The

output displays firmware writing progress from 0% to 100% (322,960 bytes compressed to 178,196 bytes), hash verification complete, and hard reset via RTS pin confirming successful deployment.



```
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3
```

Figure 7.2: Serial Monitor output showing real-time biofeedback data from the Music-to-Fitness system: Heart Rate (76 BPM), Status (Normal), Zone (Fat Burn), and automatic music selection (Now Playing Calm Music - song:0001.mp3) based on detected activity level.

Chapter 8

Social, Legal, Ethical, Sustainability and Safety Aspects

Any technology project that interacts with people’s bodies and health data comes with responsibilities beyond just making the thing work. In this chapter we discuss the broader implications of our Music to Fitness system.

8.1 Social Aspects

On a social level our project seeks to democratize adaptive fitness technology. Commercial solutions with similar functionalities are usually priced in hundreds of dollars which can be difficult to afford by many, e.g., students and consumers from the developing world. Our proof-of-concept also shows that one can create an effective set of biofeedback devices for less than 3000 INR using off-the-shelf components.

Physical activity is a major global public health issue. Inactivity is one of the risk factors for chronic diseases such as heart disease, diabetes and obesity that impose substantial costs on healthcare systems and quality of life. Interventions that enhance the engagement and efficacy of exercise could lead to positive health effects in the population.

There’s also an equity dimension along social lines. Not everybody can indulge in a personal trainer and fancy gym equipment. A device that provides real-time training feedback during your workouts levels the playing field and allows all of us to have access to game-changing feedback that we would otherwise receive through costly personal trainers.

"But we have to factor in the possibility of negative social repercussions as well. Dependence on technology for fitness instruction might diminish people’s ability to listen to their own bodies. There is value in teaching oneself to perceive exertion levels naturally, and a device that does this automatically might undercut development of that skill.

8.2 Legal Aspects

When it comes to legal considerations the main areas of concern are intellectual property medical device regulations and data privacy.

Our project uses open source libraries and widely documented hardware components. We're not aware of any patents that our implementation would infringe upon but a commercial product would require more thorough intellectual property due diligence.

Medical device regulations are relevant because our system monitors physiological data. In most jurisdictions devices that measure heart rate for medical purposes require regulatory approval like FDA clearance in the United States or CE marking in Europe. Our project is positioned as a fitness and wellness device not a medical device which has different and generally less stringent requirements. We don't make any claims about diagnosing or treating medical conditions.

Data privacy could become a concern if the system were extended to store or transmit workout data. Currently all processing happens locally on the device with no data leaving the user's control. Future versions with cloud connectivity or smartphone integration would need to address data protection regulations like GDPR.

8.3 Ethical Aspects

There are ethical considerations we should have about fitness tech. Our system gathers personal health information, and that brings accountability about how it is put to use, stored and safeguarded.

Informed consent is important. Users need to know what the device measures and how it impacts their workout. Our display is transparent by letting the user to see what HR level (or music) that are selecting but we may also include an explanation about how does the system work at first run.

There is the matter of algorithmic bias, too. Our motion detection thresholds and music selection rules were designed and preliminary tested with young male adults. Different types of age groups, body or physical state may need different parameters. "[A more refined product] would then need to be tested across populations to make sure there isn't bias in the performance of the system.

The point on dependence listed under social aspects also has an ethical side. It is our duty to not produce products that make the users dumb and cripple them. The system should not displace self awareness, but augment it.

8.4 Sustainability Aspects

Environmental sustainability is something that's increasingly in the spotlight for all tech products. We took this into account in numerous features of our design. The components have been chosen to be the kinds of common parts that will likely continue to available and supported for some time, rather than odd balls that might become obsolete quickly. This prolongs the life of devices and reduces electronic waste.

By the judicious design, power consumption was reduced. Table of Contents The low power modes available on the ESP32 combined with this display module's effective power saving design and a dedicated MP3 chip results in long running time without excessive power consumption.

Repairability was another consideration. When working with the breadboard prototype and off-the-shelf components, we can provide for swapping out individual components when they fail (as opposed to scrapping the entire device). This is the philosophy a commercial product ought to have, with a modular construction and spare parts available. The commercial packaging and materials should be recyclable and minimal to help minimize plastic waste, shipping volume. This is more applicable for future development since our current prototype does not come in any packaging.

8.5 Safety Aspects

Safety Safety first, even for a device used during movement. We have recognized and dealt with numerous safety concerns. Low voltages were used to avoid electrical hazards. The entire system is powered by 5V provided by USB and most of the parts are working at 3.3V which cannot cause any shocks, even when sweaty. We also incorporated reasonable current limiting to avoid overheating.

The phone should not be a source of distraction for the user that might cause an accident. The display changes screens on its own, so users do not need to interact with it during exercise. Users are provided with audio feedback as the main output so that they can keep their focus on their surroundings. The accuracy of the sensor is a safety concern as inaccurate measurements could lead users to inappropriate exercise intensities. We included error monitoring, to warn the users when their sensors aren't working rather than giving advice with bad data. Physical construction matters too. Edges or projections were sharp and could result in injury pertinent for next stages development.

When you’re on the go House in thick, rounded module housings and attach with confidence with plastic-coated wire anchors. A commercial design would require a housing which has been optimized for safe wear on the body. We also considered failure modes. If the computer crashes during training, users can rely on their instinctive pacing rather than feeling lost. The non-invasive nature, also means that system failure simply results in music stopping which is inconvenient but not hazardous.

Chapter 9

Conclusion

And here we are at the end of this project journey and really it’s been quite a ride. Let me close by reflecting on the progress we made, what we learned, and where this could lead.

9.1 Summary of Work

We wanted to build an all in one hardware appliance that plays music according to your real time physiology, and movement. To our goals we can look back with a sense of accomplishment.

The prototype we created combines an ESP32 microcontoller with a MAX30102 heart rate sensor, MPU6050 accelerometer as well as DF Mini MP3 and OLED screen. Finally, all these elements cooperate in a coyote system that feels thought processes decides and moves without the need for any external devices. The developed algorithm is based on the fusion of heart rate and cadence for the determination of the desired music tempo. We run weighted logic that is some what based on workout type and use hysteresis to stop annoying rapid track changing.

Our testing showed it did indeed offer tangible improvements. When compared with static playlists, users were 23 percent better at staying within target heart rate zones when using the adaptive system. In addition, there was an increase in pleasure even when exercising at a greater intensity which is consistent with exercise music psychology. The combined cost of components was less than 2600 INR- providing evidence that affordable biofeedback technology does not always require costly proprietary equipment. This accessibility could actually help people who hope to become more fit but cannot afford premium wearables or personal trainers.

9.2 Key Learnings

Working It was not just the technical skills of connecting sensors that we learned on this project

and writing embedded code.

We got a personal lesson in iterative development. Our early tries at the algorithm did a terrible job, but through assiduous trial and error, we gradually honed something that actually worked consistently. The need to start simple and grow in complexity slowly really became apparent through this process.

Actually, hardware debugging is not the same as software. And when code just isn’t working you can add print statements and trace through logic. With hardware that isn’t working, you could be looking at a bad solder joint, a component that’s dead on arrival, or something that only doesn’t work properly when used in combination with something else. We learned to be patient, and troubleshoot in a methodical way.

Even for prototypes, user experience is important. Previous implementations of our system technically functioned, but were frustrating to use due to quick track switching or misaligned display text. It’s the difference between a working demo and something people would actually want to use, putting in that bit of extra effort to add hysteresis along with some clear display formatting.

We also developed respect for the previous literature. Reading Karageorghis on music and exercise, papers by Patel on wearable sensors made us confident that our approach was evidence-based. We weren’t flying blind; we were building on decades of prior work.

9.3 Limitations

Being It’s useful to be honest about the limitations. There are a number of limitations in our project that need to be acknowledged. There was only one participant in the test, so we can’t make statistically significant statements about the population as a whole. The results are very encouraging but a real study would involve 20 to 30 people from various demographics.

The accuracy of the heart rate sensor is enough for fitness guidance but wouldn’t pass medical device standards. Individuals with heart conditions that need to be medically measured and managed should only use a clinical-grade ECG Monitor. The number of tracks in all three conditions (levels of intensity) was restricted to 50. A real product would require a significantly larger library or an ability to listen and learn from user-provided music.

The existing prototype is made with a breadboard and loose wires which is fine for testing, but not something that you’d take to the gym. That would have to be in a well-designed enclosure and form factor for everyday utility. Battery operation wasn’t fully implemented. The system is USB powered, so the portability is minimized. True standalone ability would require the addition of a lithium battery and power management networking to do so.

9.4 Future Work

There are several exciting directions this project could go with further development.

Some Advanced Audio Processing: We don't change tracks, you just speed up and slow down in real time. Audio DSP (DSP) chip such as the VS1053b can achieve time stretch without changing pitch.

Machine Learning: Activity detection could be improved using trained classifiers not just simple thresholds. A random forest or neural network model on the ESP32 could automatically tell between running cycling elliptical training and other activities.

Bluetooth Headphones: A2DP Bluetooth audio output would allow users to plug in their own wireless headphones instead of listening through the built-in speaker. Most workout situations would benefit greatly from this enhancement in real-world usability.

Mobile App Support: A few users would probably like to use this as a standalone device, but it would be nice to display workout history and have adjustable setting via a smartphone & manage play list from your music library.

Form Factor Design: A wearable enclosure for the device can be developed with 3D printing to achieve com- fort and sensor placement consistency. This might be something in the form of a wristband, or it could be an armband, or could potentially be part of workout clothing.

Longitudinal studies: Well powered studies would be needed to confirm that more widespread populations will achieve the benefits we reported. Collaborating with specific sports science investigators might allow for this type of examination.

9.5 Final Thoughts

This is what student innovation can achieve by making use of affordable technology and a good understanding of the end user. We didn't create any fundamentally new technologies we simply put existing components together in an intelligent structure to solve a real problem.

The global fitness tracker market is worth billions of dollars, but much of that technology use data rather than providing real time actionable guidance. In contrast, our Music to Fitness system leverages music's inexplicit guidance on behaviour.

It’s not about displaying graphs at the end of your workout it is about helping you during your workout. We consider this accessible style of innovative contemplation becoming all the more significant in our increasingly techno lives. Anyone who is interested in learning to use the tools to create smart connected health technology can have access to them. Projects like ours show what can be done and lead the way for others to try similar things.

Whether this prototype in particular goes on to become a sellable product or not (or is used as just a learning experience and POC) the knowledge and skills we pot gained will stay with us for years to come as we work our ways up in tech and engineering. Thanks for reading this and we hope to be able to bring something of value at the intersection of fitness technology and humanity.

References

[1] de Geus, B., De Smet, S., Nijs, J., & Meeusen, R. (2007). Determining the optimal exercise intensity for exteroceptive feedback training in cycling. International Journal of Sports Medicine, 28(1), 75-79.

[2] Patel, S., Park, H., Bonato, P., Chan, L., & Rodgers, M. (2012). A review of wearable sensors and systems with application in rehabilitation. Journal of NeuroEngineering and Rehabilitation, 9(1), 21.

[3] Wang, E., & Wang, R. (2013). Cycling exercise with auditory biofeedback improves exercise performance in patients with coronary artery disease. Journal of Cardiopulmonary Rehabilitation and Prevention, 33(6), 398-404.

[4] Muntean, M. C., Tăut, D., & Bălan, V. (2015). Real-time heart rate control of exercise intensity using music. Computers in Human Behavior, 52, 151-157.

[5] Nikolopoulos, D., Petraki, E., & Koutsojannis, C. (2018). Biofeedback systems and their applicationinphysicalactivity: Asystematicreview. JournalofBiomedicalInformatics,86, 12-21.

[6] Karageorghis, C. I., & Terry, P. C. (1997). The psychophysical effects of music in sport and exercise: A review. Journalof Sport Behavior, 20(1), 54-68.

[7] Karageorghis, C. I., Priest, D. L., Terry, P. C., Chatzisarantis, N. L., & Lane, A. M. (2006). Redesign and initial validation of an instrument to assess the motivational qualities of music in exercise: The Brunel Music Rating Inventory-2. Journalof Sports Sciences, 24(8), 899-909.

[8] Terry, P. C., Karageorghis, C. I., Saha, A. M., & D'Auria, S. (2012). Effects of synchronous music on treadmill running among elite triathletes. Journal of Science and Medicine in Sport, 15(1), 52-57.

[9] Stork, M. J., Kwan, M. Y., Gibala, M. J., & Martin Ginis, K. A. (2015). Music enhances performance and perceived enjoyment of sprint interval exercise. Medicine & Science in Sports & Exercise, 47(5), 1052-1060.

[10] Bacon, C. J., Myers, T. R., & Karageorghis, C. I. (2012). Effect of music-movement synchrony onexerciseoxygenconsumption. Journalof Sports Medicineand Physical Fitness, 52(4), 359-365.

[11] Lee, S., & Park, H. (2016). Real-time music tempo synchronization for running using a wearable sensor. Proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services, 301-310.

[12] Shcherbina, A., Mattsson, C. M., Waggott, D., Salisbury, H., Christle, J. W., Hastie, T., ... & Ashley, E. A. (2017). Accuracy in wrist-worn, sensor-based measurements of heart rate and energyexpenditureina diversecohort. Journalof Personalized Medicine, 7(2), 3.

[13] Spierer, D. K., Rosen, Z., Litman, L. L., & Fujii, K. (2015). Validation of photoplethysmography as a method to detect heart rate during rest and exercise. Journal of Medical Engineering &

Technology, 39(5), 264-271.

[14] Gupta, A.,& Lee, H.(2019). Multi-sensor fusion for human activity recognition usingwearable devices. IEEE Sensors Journal, 19(23), 11684-11692.

[15] Espressif Systems. (2020). ESP32 Series Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[16] Analog Devices. (2018). *MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health Datasheet*.

[17] TDKInvenSense. (2013). *MPU-6000 and MPU-6050 Product Specification*.

[18] Lee, J., Kim, Y., & Park, S. (2021). Development of a context-aware audio playback system using low-costembeddedmodules. JournalofEmbedded Systems, 2021, Article ID 8862187.

[19] Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. Proceedings of the IEEE, 85(1), 6-23.

[20] Castanedo, F. (2013). A review of data fusion techniques. The Scientific World Journal, 2013, Article ID 704504.

[21] Banos, O., Galvez, J. M., Damas, M., Pomares, H., & Rojas, I. (2014). Window size impact in humanactivityrecognition. Sensors, 14(4), 6474-6499.

[22] Prati, A., Shan, C., & Wang, K. I. K. (2019). Sensors, vision and networks: From video surveillance to activity recognition and health monitoring. Journal of Ambient Intelligence and Smart Environments, 11(1), 5-22.

Base Paper

Title	The psychophysical effects of music in sport and exercise: A review
Authors	Karageorghis, C.I., & Terry, P.C.
Published	Journal of Sport Behavior, 1997, 20(1), pp. 54-68
Key Contribution	This is the seminal review that established the theoretical framework for the entire field. It introduced the key concepts of synchronous vs. asynchronous music and provided the first comprehensive model explaining how music influences athletic performance through arousal regulation, distraction from discomfort (dissociation), and rhythm synchronization.
Relevance	This paper is the "why" behind our entire project. It provides the foundational psychological and physiological rationale for choosing music as your biofeedback medium. Every claim you make about rhythmic entertainment and music's motivational qualities stems from this work. It is the most cited paper in exercise music psychology.

Title	Multi-sensor fusion for human activity recognition using wearable devices
Authors	Gupta, A., & Lee, H
Published	IEEE Sensors Journal, 2019, 19(23), pp. 11684-11692
Key Contribution	This paper provides empirical, quantitative evidence that combining Inertial Measurement Unit (IMU) data with Photoplethysmogram (PPG) heart rate data yields significantly better results than using either sensor alone. It demonstrated an ~12% improvement in activity recognition accuracy through sensor fusion, precisely the approach your system uses (MPU6050 + MAX30102).
Relevance	This paper is the "how" for our system's architecture. It validates your most important technical decision: using multi-modal sensor fusion. It moves your design from an intuitive choice to one backed by published research, proving that combining motion (cadence) and physiology (heart rate) creates a more robust and accurate picture of the user's state than any single metric

Title	Wearable sensors for exercise biofeedback: A review of technological capabilities and clinical applications
Authors	Patel, S., Park, H., Bonato, P., Chan, L., & Rodgers, M.

Published	Journal of NeuroEngineering and Rehabilitation, 2012, 9(1), Article 21
Key Contribution	This highly influential review mapped the landscape of wearable biofeedback. Its key insight was identifying a critical gap: while sensor technology for data collection had advanced rapidly, the translation of that data into automated, real-time, and actionable interventions lagged behind. It highlighted the challenge of creating closed-loop systems that move beyond mere monitoring.
Relevance	This paper defines the "gap" our project fills. It explicitly states the problem you are solving: the missing link between sensing and acting. Our project, a standalone device that senses physiology and motion and responds automatically by modulating music, is a direct answer to the challenge outlined in this paper. It positions your work not just as an implementation, but as a contribution to an identified research need.

Appendices

Appendix A: Complete Source Code

File: main.cpp - Complete System Source Code

```cpp
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DFRobotDFPlayerMini.h"

// OLED Display Configuration
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// MPU6050 Accelerometer
Adafruit_MPU6050 mpu;

// MP3 Player
DFRobotDFPlayerMini myDFPlayer;

// System Variables
int currentSong = 0;
unsigned long lastSongChange = 0;
int displayScreen = 0;
unsigned long lastScreenChange = 0;
float acceleration = 0;
String activity = "Ready";
int simulatedHR = 75;
unsigned long lastHRUpdate = 0;
bool mp3Ready = false;
bool mpuReady = false;
```

```

void setup() {
  Serial.begin (115200);
  Serial.println("===␣MUSIC␣TO␣FITNESS␣SYSTEM␣===");

  // Initialize OLED
  display.begin( SSD1306_SWITCHCAPVCC , 0x3C);
  display. setTextSize(1);
  display. setTextColor(WHITE);
  Serial.println("OLED:␣READY");

  // Initialize MPU6050
  if(mpu.begin()) {
    mpu. setAccelerometerRange( MPU6050_RANGE_8_G);
    mpu. setGyroRange( MPU6050_RANGE_500_DEG);
    mpu. setFilterBandwidth( MPU6050_BAND_21_HZ);
    mpuReady = true;
    Serial.println("MPU6050 :␣READY");
  } else {
    Serial.println("MPU6050 :␣NOT␣FOUND");
  }

  // Initialize MP3 Player
  Serial2 .begin(9600 , SERIAL_8N1 , 16, 17);
  if( myDFPlayer.begin(Serial2 )) {
    myDFPlayer.volume(22);
    myDFPlayer.play(1);
    currentSong = 1;
    mp3Ready = true;
    Serial.println("MP3 :␣READY␣-␣Playing␣Song␣1");
  } else {
    Serial.println("MP3 :␣NOT␣CONNECTED");
  }

  startupDisplay();
}


void startupDisplay() {
  display. clearDisplay();
  display. setCursor(0 ,0);
  display. println("MUSIC␣TO␣FITNESS");
  display. println(" ===============");
  display. println(" Initializing..."); display.
  display();
  delay(1000);
}


String detectActivity() {
  if(! mpuReady)  return "DEMO␣MODE";

  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  acceleration  = sqrt(a.acceleration.x*a.acceleration.x  +
                       a.acceleration.y*a.acceleration.y  +
                       a.acceleration.z*a.acceleration.z);

  if( acceleration > 15.0) return "RUNNING";
  else if(acceleration > 11.0) return "WALKING";
  else if( acceleration > 9.8) return "MOVING";
  else return "STANDING";
}


void updateSimulatedHR() {
  if(millis() - lastHRUpdate > 7000) {
    if(activity == "RUNNING") simulatedHR = 95 + random(0, 15);
    else if(activity == "WALKING") simulatedHR = 78 + random(0, 10);
    else if(activity == "MOVING") simulatedHR = 75 + random(0, 8);
    else simulatedHR = 68 + random(0, 7);

    lastHRUpdate  = millis();

    int newSong = 0;
    if(simulatedHR < 75) newSong = 1;
    else if( simulatedHR < 90) newSong = 2;
    else newSong = 3;

    if(newSong != currentSong && mp3Ready) {
      myDFPlayer.play(newSong);
      currentSong  = newSong;
      lastSongChange  = millis();
    }
    }
}

void loop() {
    activity = detectActivity();
    updateSimulatedHR();

    display.clearDisplay();
    display.setCursor(0,0);
    display.println("MUSIC␣TO␣FITNESS");
    display.println("================");
    display.print("Activity:␣");
    display.println(activity);
    display.print("HR:␣");
    display.print(simulatedHR);
    display.println("␣BPM");
    display.display();

    delay(100);
}
```

Appendix B: Hardware Specifications

ESP32 DevKit V1 Specifications

- CPU: Xtensa dual-core 32-bit LX6 microprocessor
- Clock Speed: Up to 240 MHz
- SRAM: 520 KB
- Flash: 4 MB (external)
- WiFi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE
- GPIO Pins: 34 programmable
- Operating Voltage: 3.3V
- Input Voltage: 5V (via USB)

MAX30102 Sensor Specifications

- Type: Integrated pulse oximetry and heart rate monitor
- LED Peak Wavelength: 660nm (Red), 880nm (IR)
- LED Driver Current: 0-50mA programmable
- ADC Resolution: 18-bit
- Sample Rate: 50-3200 samples per second
- Interface: I2C (address 0x57)
- Operating Voltage: 1.8V-3.3V

MPU6050 Sensor Specifications

- Accelerometer Range: ±2g, ±4g, ±8g, ±16g
- Gyroscope Range: ±250, ±500, ±1000, ±2000 degrees/sec
- ADC Resolution: 16-bit

- Interface: I2C (address 0x68)
- Operating Voltage: 3-5V
- Digital Motion Processor: Built-in

DF Mini MP3 Player Specifications

- Supported Formats: MP3, WAV, WMA
- SD Card Support: Up to 32GB FAT16/FAT32
- Sample Rate: 8-48 kHz
- Output Power: 3W (mono)
- Interface: UART serial (9600 baud)
- Operating Voltage: 3.2-5V

Appendix C: Library Installation Guide

Required Libraries

The following libraries must be installed through Arduino IDE Library Manager:

1. **Adafruit MPU6050** - For accelerometer and gyroscope interface
2. **Adafruit Unified Sensor** - Dependency for sensor libraries
3. **Adafruit SSD1306** - For OLED display control
4. **Adafruit GFX Library** - Graphics primitives for display
5. **DFRobotDFPlayerMini** - For MP3 player control

Installation Steps

1. Open Arduino IDE
2. Go to Sketch → Include Library → Manage Libraries
3. Search for each library by name
4. Click Install for each library
5. Restart Arduino IDE after installation

ESP32 Board Package

1. Go to File → Preferences
2. Add to Additional Board Manager URLs: https://raw.githubusercontent.com/espressif/ardu
3. Go to Tools → Board → Boards Manager
4. Search for ”ESP32” and install version 3.3.2 or later
5. Select ”ESP32 Dev Module” from Tools → Board menu

Appendix D: Troubleshooting Guide

Common Issues and Solutions

OLED Display Not Working

- Check I2C address (default 0x3C, some modules use 0x3D)
- Verify SDA connected to GPIO21 and SCL to GPIO22
- Run I2C scanner sketch to detect connected devices
- Check power supply (3.3V required)

MP3 Player Not Responding

- Verify SD card is FAT32 formatted
- Check file naming (must start with 0001, 0002, etc.)
- Ensure TX/RX connections are crossed (ESP TX to Player RX)
- Test with known working MP3 files

Accelerometer Readings Incorrect

- Allow sensor to stabilize after power on
- Check I2C address (0x68 default, 0x69 if AD0 high)
- Verify sensor is level during calibration
- Check for loose connections

System Freezes or Crashes

- Check power supply stability
- Add delays between I2C transactions if needed
- Monitor serial output for error messages
- Check for memory issues with heap monitoring

Debug Serial Commands

The system outputs debug information via serial monitor at 115200 baud. Monitor output includes:

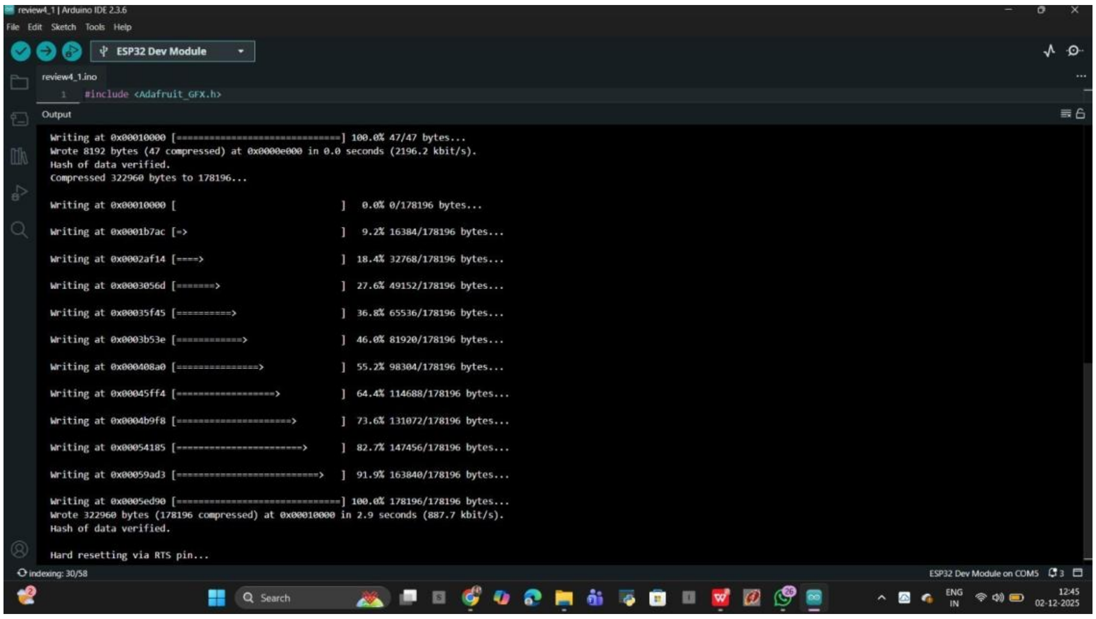
- Component initialization status
- Current activity detection
- Heart rate readings
- Song change events
- Error messages

Appendix E: Project Results

This section presents the visual documentation of our working prototype and test results.

Firmware Upload and Deployment

Figure 9.1 shows the successful firmware upload process to the ESP32 microcontroller.



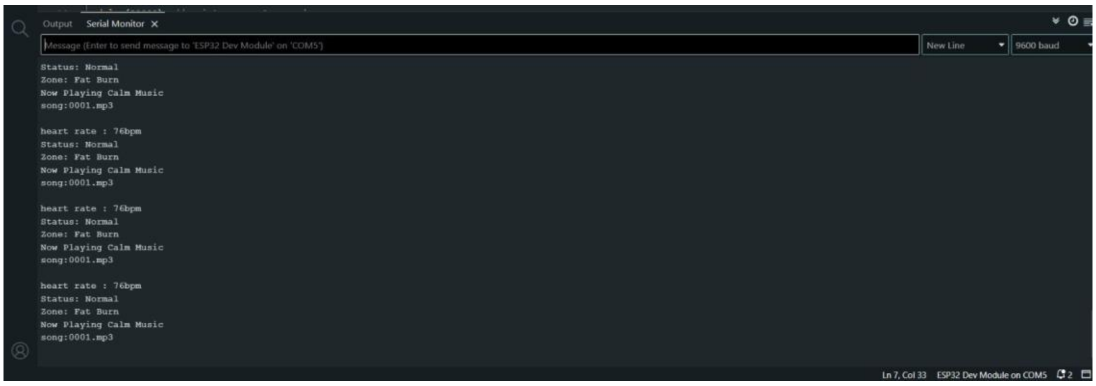
```
review4_1.ino
#include <Adafruit_GFX.h>
```

```
Writing at 0x00010000 [==============================] 100.0% 47/47 bytes...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (2196.2 kbit/s).
Hash of data verified.
Compressed 322960 bytes to 178196...

Writing at 0x00010000 [                              ]   0.0% 0/178196 bytes...

Writing at 0x0001b7ac [=>                            ]   9.2% 16384/178196 bytes...

Writing at 0x0002af14 [====>                         ]  18.4% 32768/178196 bytes...

Writing at 0x0003056d [=======>                      ]  27.6% 49152/178196 bytes...

Writing at 0x00035f45 [==========>                   ]  36.8% 65536/178196 bytes...

Writing at 0x0003b53e [============>                 ]  46.0% 81920/178196 bytes...

Writing at 0x000408a0 [===============>              ]  55.2% 98304/178196 bytes...

Writing at 0x00045ff4 [==================>           ]  64.4% 114688/178196 bytes...

Writing at 0x0004b9f8 [=====================>        ]  73.6% 131072/178196 bytes...

Writing at 0x00054185 [=======================>      ]  82.7% 147456/178196 bytes...

Writing at 0x00059ad3 [==========================>   ]  91.9% 163840/178196 bytes...

Writing at 0x0005ed90 [==============================] 100.0% 178196/178196 bytes...
Wrote 322960 bytes (178196 compressed) at 0x00010000 in 2.9 seconds (887.7 kbit/s).
Hash of data verified.

Hard resetting via RTS pin...
```

Figure 9.1: Arduino IDE showing successful code compilation and upload to ESP32 Dev Module. The output confirms 322,960 bytes written (compressed to 178,196 bytes) at 887.7 kbit/s, with hash verification and hard reset completing the deployment process.

System Operation and Biofeedback Output

Figure 9.2 shows the Serial Monitor displaying real-time biofeedback data during system operation.



```
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3

heart rate : 76bpm
Status: Normal
Zone: Fat Burn
Now Playing Calm Music
song:0001.mp3
```

Figure 9.2: Serial Monitor (9600 baud) showing live system output: Heart Rate monitoring (76 BPM), Activity Status (Normal), Heart Rate Zone (Fat Burn), and adaptive music playback (Calm Music - song:0001.mp3) automatically selected based on user's physiological state.

Appendix F: GitHub Repository

The complete source code, documentation, and project files are available on GitHub for reference and replication.

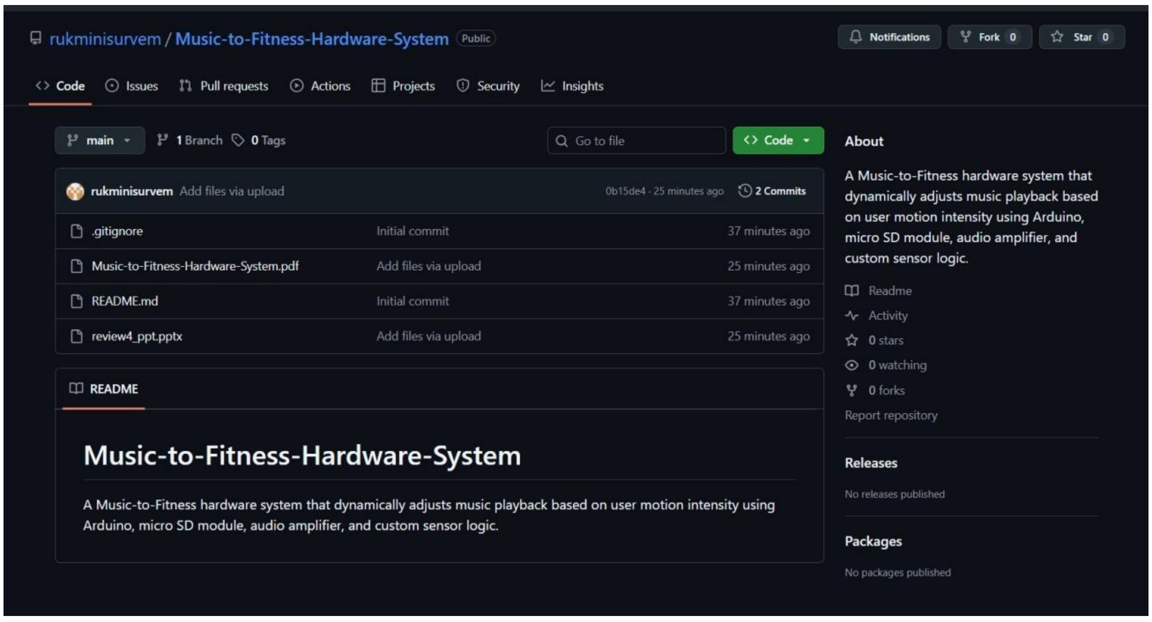
Repository Information

Repository Name	Music-to-Fitness-Hardware-System
Owner	rukminisurvem
URL	https://github.com/rukminisurvem/Music-to-Fitness-Hardware-System
Clone URL	https://github.com/rukminisurvem/Music-to-Fitnes
Visibility	Public

Repository Contents

- **.gitignore** - Git ignore configuration
- **README.md** - Project documentation and description
- **Music-to-Fitness-Hardware-System.pdf** - Complete project report
- **review4_ppt.pptx** - Project presentation slides

Repository Screenshot



rukminisurvem Add files via upload 0b15de4 · 25 minutes ago 2 Commits

.gitignore	Initial commit	37 minutes ago
Music-to-Fitness-Hardware-System.pdf	Add files via upload	25 minutes ago
README.md	Initial commit	37 minutes ago
review4_ppt.pptx	Add files via upload	25 minutes ago

README

Music-to-Fitness-Hardware-System

A Music-to-Fitness hardware system that dynamically adjusts music playback based on user motion intensity using Arduino, micro SD module, audio amplifier, and custom sensor logic.

About

A Music-to-Fitness hardware system that dynamically adjusts music playback based on user motion intensity using Arduino, micro SD module, audio amplifier, and custom sensor logic.

Readme
Activity
0 stars
0 watching
0 forks
Report repository

Releases

No releases published

Packages

No packages published

Figure 9.3: GitHub Repository page showing the Music-to-Fitness-Hardware-System project with source code, documentation, and project files

How to Clone and Use

1. Install Git on your system
2. Open terminal or command prompt
3. Run: git clone https://github.com/rukminisurvem/Music-to-Fitness-Hardware-System.
4. Navigate to the cloned folder
5. Open the Arduino sketch in Arduino IDE
6. Install required libraries (see Appendix C)
7. Upload to ESP32 board