

StepQuest: A Gamified Fitness Tracker

Mohamed Al Mahruqi, Jacey Klose, Spenser Tacinelli, William Wimberly

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

***Abstract* - In an era where sedentary lifestyles are increasingly common, finding effective and engaging ways to promote physical activity presents a significant challenge. Many people struggle to maintain interest in traditional fitness routines due to their repetitive nature and lack of engagement. StepQuest addresses this challenge by integrating physical exercise with the compelling and interactive elements of role-playing games (RPGs). Using the MPU-6050 inertial measurement unit (IMU) to convert real-life movements into progress within a virtual world, players can explore, battle, and quest as they exercise. The device features a power circuit with Li-ion battery charging capabilities so that users can wear it as they exercise, much like a standard fitness tracker. Using a user-friendly touchscreen interface ensures that users can easily interact with the user-interface and easily participate in the gaming elements in a modern form-factor.**

I. Introduction

The StepQuest project uses the MPU-6050 inertial measurement unit (IMU) to track the player's physical movements and translate them into actions within a virtual RPG environment. This functionality relies on algorithms designed to recognize a variety of exercises, enabling these movements to directly affect gameplay. The system includes a power circuit designed for use with Li-ion batteries, facilitating its use in tracking exercises and movement, while still being able to be worn on the user's wrist. This portable form-factor is expected in modern fitness trackers. Interaction with the game is made possible through a touchscreen interface, allowing players to explore the game environment, accept quests, and manage their inventory directly. This configuration records and

applies physical activity to the game, creating a tangible connection between the user's real-life actions and their virtual progress.

II. Engineering Specifications

A concise overview of the device specifications selected by our group for further development can be seen in the table below. Our three chosen specifications for demonstration are counting steps, screen response time, and quest completion. Though these details are open to revision, they provide an initial understanding of the device's capabilities and features.

A. Testing Step Counting

Step counting is tested by manually counting the number of steps taken by the tester and comparing the number to the number of steps counted by the device. A tally counter is sometimes used to reduce human error.

As our engineering specification is specifically for step counting with a normal, steady gait the user avoids using the arm to which the device is attached for anything else such as using their phone or touching their face. The required accuracy is 80% or greater.

B. Testing Response Time

Testing the screen response time cannot be done by the naked eye as the time requirement is short at only 500ms. In order to test the response time a slow-motion video is recorded of the device screen and a digital stopwatch as the tester switches screens or clicks buttons on the screen.

The tester then reviews the footage and takes their best estimate of the time the contact was made with the device and the time the screen has fully switched and calculates the response time. Some human error is expected in this process.

C. Testing Quest Completion

Quests are tested by generating only fixed length quests of each type and testing how many of that activity (squats, jumping jacks, or steps) are actually required before the quest is completed.

	Engineering Specifications	Details	Value
1	Device Size	The size of the device is convenient.	<45mm
2	Battery Life	On full charge battery life is sufficient	>=8hr
3	Counting Steps	Using an accelerometer, we can count the steps of the user with a normal and steady gait accurately	>=80%
4	Timekeeping	Through a Real-Time clock module the device is able to keep time with accuracy.	+/-1 min per month
5	Response Time	Loading different screens in the game should be quick.	<500ms
6	Quest Completion	Accepted quests complete after the requirement is completed by the user.	>=90%
7	Screen Wake	The screen should wake up from idle in a reasonable amount of time.	<1s

Table 1. Engineering Specifications

III. System Components

StepQuest consists of a number of components in order to achieve the aforementioned engineering specifications. This section provides a short technical explanation of each major component.

A. Inertial Measurement Unit

An IMU or Inertial Measurement Unit is a sensor that records the device's velocity and orientation. Typically, this is accomplished using a combination of an accelerometer and a gyroscope. StepQuest utilizes an MPU6050 which was chosen due to its thorough documentation. The IMU is a vital component due to its use in step counting and quest completion, which are necessary specifications for our project.

B. Battery

StepQuest would not be complete without a proper battery charging the MCU and the rest of the components. It was decided to go with the Li-ion battery DTP603443 with 850 mAh. The battery is equipped with a protection IC preventing any damage. The 850 mAh capacity gives around 6-8 hours of use. The battery will be recharged with 500mA rate which makes the total recharge time is 01:42 minutes.

C. Vibration Motor

The notification system is aided by a vibration system. Certain events need the attention of the user and a vibration system helps notify and alert the user. Among the three major types of vibration motors, Eccentric Rotating Mass (ERM), Linear Resonant Actuator (LRA), and Coin vibration, we chose coin vibration for its shape and practicality in the PCB design. Our design uses FIT0774 module which has 11000 RPM and 3V voltage rating.

D. MCU

To run the project, a micro-controller unit had to be selected that met our many demands and goals. The MCU would have to have enough power to push pixels to our 240x240 pixel screen, while also maintaining a step/exercise algorithm, while also keeping track of the time and many states, as well as having enough program storage to host the graphics for the game. A stretch goal of ours was to have Wi-Fi, so this would also be a plus. Another huge plus would be documentation. We would also be using the Arduino IDE and libraries to program, so this narrowed our choices a bit. Ultimately it came down to: the ATmega328P, ESP32, or the RP2040.

Straight away it could be seen that the ATmega was the weakest choice of the group, not having dual cores,

being much slower than the other choices, and not offering Wi-Fi.

Being nearly the same in performance, size, cost, and availability, our choice of the ESP32 essentially came down to community support, and support in the Arduino IDE. Essentially the features that we were looking for, like Wi-Fi were only well documented for the C language, but not yet available in the Arduino IDE for the RP2040. Ultimately, this and great community support is what led us to the ESP32.

E. Boost Converter

As the battery gets drained, the MCU may be subject to shortage in the voltage supply. The boost converter boosts the voltage that is supplied by the battery to 5V. The boost converter feeds the linear regulator with 5V and the power regulator reduces that voltage to 3.3V. TPS61099 is the boost converter version we chose where the voltage is expected to be boosted to 5V with the use of two resistors forming a voltage divider.

F. Screen

Since the screen is how the user interacts with the game, and it is the hardware that they will physically see the most, the choice of screen is highly important. Aside from the goal of trying to solve fitness motivation, the screen was essentially the driver for this project. Right away when we saw the Waveshare 1.28 round touch screen [1], we knew that we were going to make a fitness tracker as this is exactly what the display looks like it was made for. Not only is the styling familiar to users, but at 240x 240 and 65k colors, it displays beautifully as well.

The screen hardware uses a GC9A01 display driver, and the CST816S as a touch driver, both of which have libraries available in the Arduino framework. The display uses SPI communication for display, and although it is advertised as 4—wire SPI, it uses 7 wires to communicate and 1 for backlight. The capacitive touch screen technology uses I2C. This was initially an issue as the accelerometer also used I2C, and performance suffered when on a daisy chain, Arduino and the ESP32 ended up supporting software I2C so any pins could be used to talk over this protocol. Ultimately that screen and the accelerometer ended up talking on their own networks, rather than sharing one via daisy chain, and performance improved.

IV. Hardware Details

Hardware details for each of the nine subsystems and their purpose, not including the microcontroller described in section three, are described here.

A. Data

The data subsystem includes the USB port, UART bridge and the data download circuit. The USB input provides data input to upload code to the microcontroller and power to charge the on-board battery. The USB used is a micro-B USB utilizing USB serial bus 2.0 protocol. The primary usage for the UART bridge is to allow for serial communication between the microcontroller and the user's computer. Serial communication is used for debugging purposes. Without the UART bridge, feedback cannot be received on the user's computer, and debugging would be difficult. The data download circuit facilitates a proper digital data stream to the microcontroller.

B. Power

The power subsystem includes the charging circuit, boost converter, and linear voltage regulator. The charging circuit is used to charge the battery when 5V input is connected via the USB port. If no USB input is connected, then the output of the battery goes to the input of the boost converter circuit described below. The boost converter circuit is used to increase the voltage of the battery, allowing for proper voltage input value to the linear voltage regulator described below. The battery used in our project outputs 3.7V, but the linear voltage regulator requires a 5V input. The boost converter circuit facilitates this necessary voltage increase. The linear voltage regulator takes a 5V input, supplied from the boost converter circuit, and outputs a steady 3.3V output. This steady 3.3V output is used to power the microcontroller and the peripherals. The next page shows these circuits.

C. Peripherals

The peripherals include the accelerometer, vibration motor, buttons, and touchscreen display module. The accelerometer provides acceleration deviations in three dimensions. This data is then used to approximate a user's movement and is used in various calculations. The vibration motor is used as a haptic feedback mechanism for the user. The two peripheral

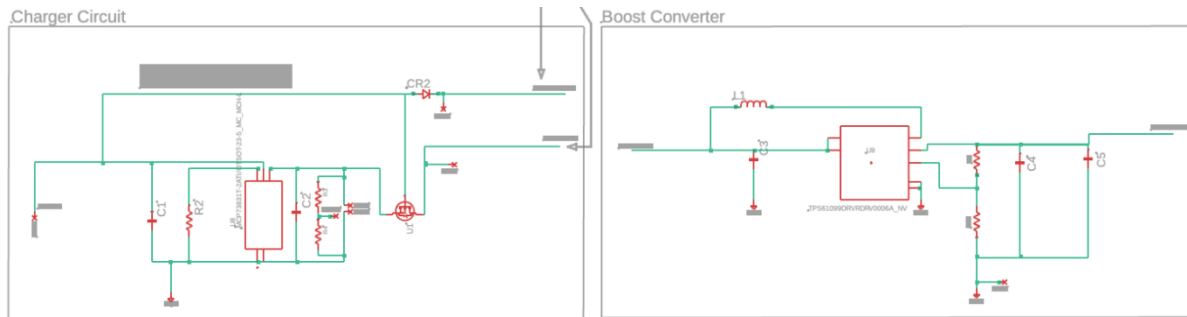


Figure 1. The Charger and Boost Converter Circuits

buttons allow the user to provide direct feedback to the device. The touchscreen display is used as both a way for the user to provide feedback to the device through the touchscreen functionality, and to provide a display to the user.

V. Board Design

Important design stages and considerations pertaining to the PCB's design are described in this section.

A. PCB Design Stages

The initial prototype PCB was not fully functional. The power and accelerometer subsystems were not working. Due to the small dimensions of the PCB (2-inch diameter), the usage of 0402 components, and 6 mil trace widths, troubleshooting and modifications were difficult.

Two additional PCBs were then designed. A PCB containing only the charging and boost converter circuit, and a PCB containing the remaining subsystems. These designs were larger in dimensions, 0805 components were utilized, and larger trace widths were used. This allowed for proper troubleshooting and any necessary modifications. Ports on the side of each board were placed to allow for connections between the PCBs.

B. Design Considerations

The USB micro-B port used is both a through-hole and surface mount package. The through-hole packaging allows for stability and a more robust design. The original prototype used on a surface mount USB port, and the port easily broke off. The USB port is also placed on the edge of the PCB as it is directly

accessed by the user, and being on the edge allows for convenient and intuitive usage.

All the subsystems the user will interact with are placed on the PCB's edges. While the PCB is larger than a final production model, this components placement simulates an intuitive experience in the general usage of such a device. These subsystems include the USB port, buttons, and display module.

VI. Software Details

StepQuest has a large software component, the implementation of which is described below.

A. Game Design

As a large portion of StepQuest is the game component, game design is a vital part of the project. The game runs off a state-machine based on the player. If the player is in a town, they can access town situations like the Quest Board and the Item Shop. If the player is traveling, they do not have access to those items, and if they are in a Dungeon they have access to Dungeon quests. There are also sub-states where the player may be completing a specific type of Quest, such as jumping-jacks (in-progress), and a state for if the player is walking. Most of these states can run concurrently thanks to the ESP32s multiple cores.

a. Map

On the map screen players can touch a location that they wish to travel to in order to begin the travel system. The logic for the travel system is as follows:

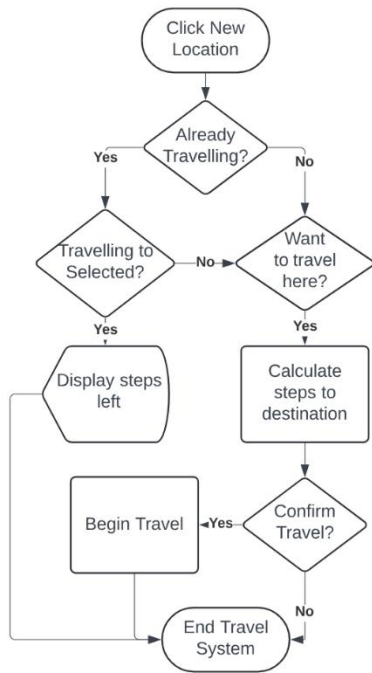


Figure 2. Travel logic used to travel between the in-world locations of StepQuest

In order to accurately calculate the steps between locations, regardless of where the player is, the game keeps track of which path the player is on and how many steps it will take to enter the next path as well as the total number of remaining steps. Once a player has reached their destination, they will receive a pop-up notification explaining that they have arrived. While traveling, users can see their progress on the home screen as well as on the status screen, as far as how many steps they have currently taken towards their destination.

b. Towns

The town screen provides access to a shop and quest board when the player is in a town.

Shops provide items based on the current player's location with later locations having better items. These items can improve either the player's armor stat or power stat with both stats mainly being used to increase the player's odds of winning in combat during dungeons and travel.

Quest boards provide quests that the player can complete in order to obtain experience and gold. Quest difficulty is randomly generated based on the player

level and location and can be one of three types: squats, jumping jacks, or steps. Quests can also be trashed to reroll a new quest or activated to begin checking for the requirement of the quest.

The simplified logic for both the shop and quest board is shown below:

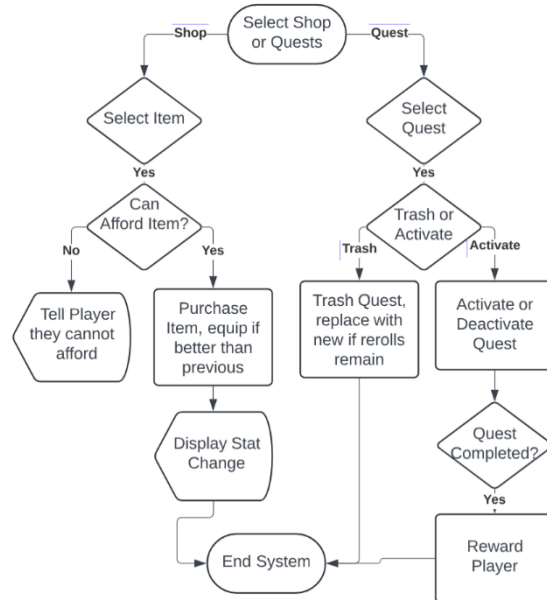


Figure 3. Town logic showing the basics of how the shop and quest board works in StepQuest

c. Dungeons

At first, the idea was that dungeons would be a timed version of the quests. As progress marched forward, more ideas flourished, and the ideas arose that more variety could be added to the game component. The final implementation of dungeons added two more types of quests: Items, and Gold. Since items are rare in the game these styles of quests only appear every 5 levels and will reward a random item based on the player's level. Gold quests appear 1/3 of the time when not on the 5th level. Regular quests are rolled 2/3 times when not on the 5th level. There is currently no timed component to the dungeon quests, however only a certain number of floors are rolled for the dungeon, and once those have been defeated the player must wait until noon or midnight for the dungeon quests to re-roll.

d. Status Screen

Players can see their status in the game through the Status screen. This screen always displays the player's level, a picture of the player, and their current status, such as "In Town", "Traveling," or "In Dungeon." Users can also click on the player on the screen to be taken to a separate status screen which will tell them more of the player's stats like how much money they have, their armor, and weapon level, etc. While in a town, the screen will show a backdrop of the town, display the player's level, and their status. Different backdrops are shown for different statuses.

B. Step Algorithm

A simplified version of the step algorithm is shown below:

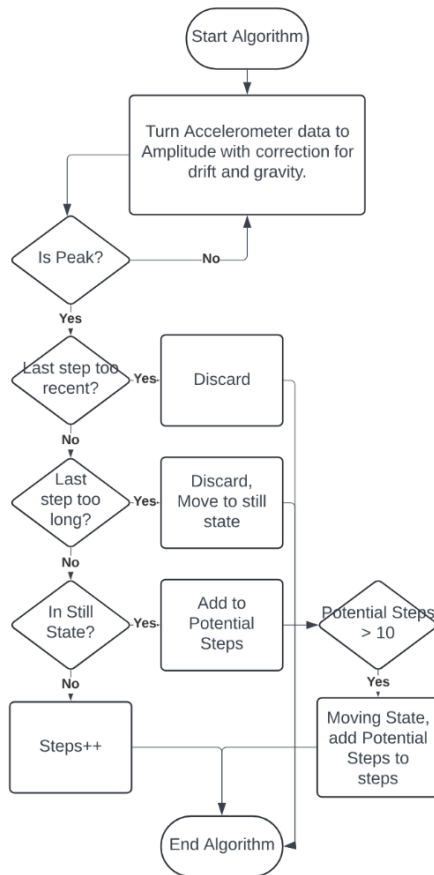


Figure 4. The step algorithm used to count steps in StepQuest

The step algorithm runs in the background of StepQuest at almost all times with the exception of during active squat and jumping jack quests. A modified improved peak detection algorithm was used which counts steps if an acceleration peak meets a number of requirements.

Due to the way step counting for both travel and quests works the step algorithm is only run on one set of accelerometer data at a time, returning a number of steps to be added to the total steps. This algorithm runs every 20 milliseconds, a time reference it uses to determine if a recent step was too recent or old.

The algorithm accounts for extraneous motion in a small sense by only counting continuous steps and requiring 10 counted steps to move to the moving state. However, consistent arm motion still thwarts the algorithm and is falsely counted as steps which could certainly be improved upon in the future.

C. Quest Completion

Quest completion is a vital component of StepQuest as it is one of the best ways for the player to increase their gold and experience. There are three types of quests and each has its requirement tracked in different ways.

a. Squat Quest

While the squat task is activated steps are not counted as the acceleration data is instead sent to the squat algorithm. Like the step algorithm, the squat algorithm converts the accelerometer data to a single amplitude with a correction for drift and gravity, however the similarities end there.

As seen below, the user must first reach below the squat threshold minimum in order to set the squat flag. If the squat flag is set then the player has 2 seconds to reach the squat threshold maximum in order to count a squat. As a result, if a player squats and then rests before rising, a squat will not be counted. Testing has shown that this algorithm is fairly accurate, especially if the user holds their arms out in front of them as is typical when doing squats.

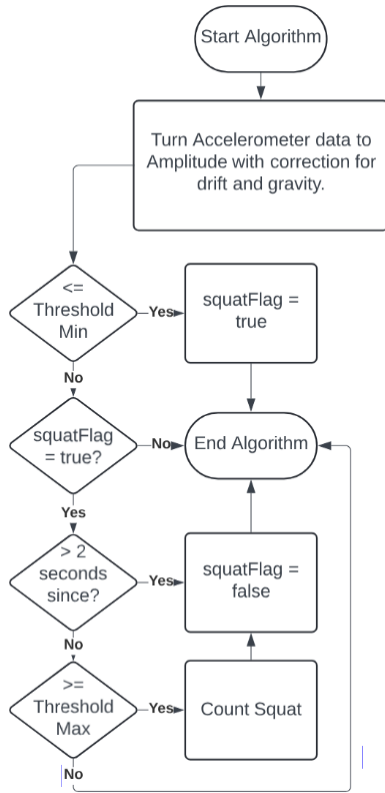


Figure 5. The algorithm used to count squats.

b. Jumping Jack Quest

The jumping jack quest is fairly straightforward compared to the other two algorithms. The algorithm is shown below:

Like the squat algorithm, when the jumping jack task is activated steps are not counted since the

acceleration data is sent to the jumping jack algorithm. It does the same first step as the squat algorithm. The algorithm is simple, only requiring that the player reaches above a fairly high jumping jack threshold minimum in order to count a jumping jack. Jumping jacks also cannot be counted unless the last one counted occurred more than 0.5 seconds prior to the new one, preventing miscounting.

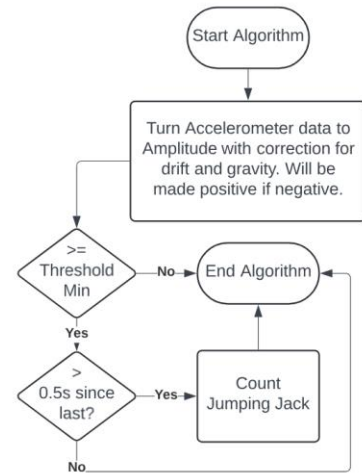


Figure 6. The algorithm used to count jumping jacks.

In practice, this algorithm is the least accurate out of the three and could certainly use some more tweaking in the future.

c. Step Quest

The step quest is the easiest quest to implement as it simply uses the aforementioned step algorithm. As such, while this quest is active the step algorithm

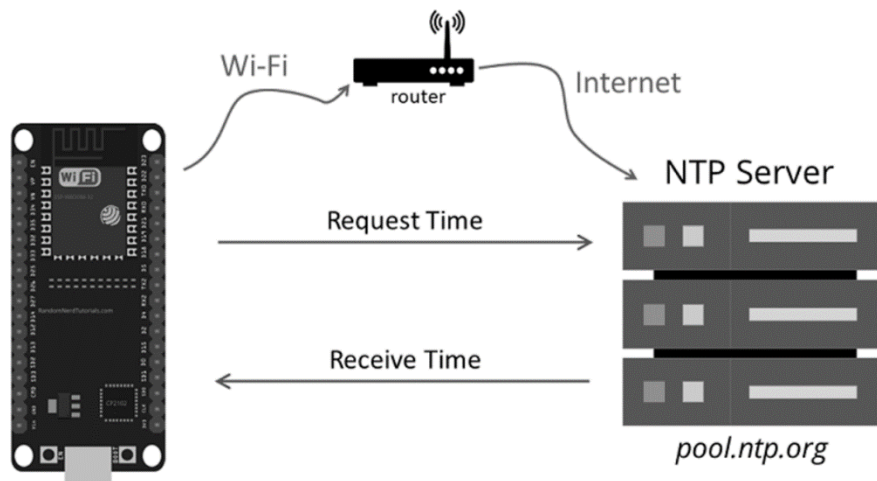


Figure 7. How StepQuest updates time when connected to WiFi

continues to add to overall player steps as well.

D. Time Keeping

Since the project portrays the shape of a watch and is worn on the wrist, the user expects the device to be able to tell time accurately. To accomplish this, we use a combination of accumulating the values from Arduino's "millis()" when offline and pinging an NTP time server when the device is connected to Wi-Fi. The following two items are particularly of note.

a. Arduino's millis()

Arduino's "millis()" is incremented (for 16 MHz AVR chips and some others) every 1.024 milliseconds, then incrementing by 2 (rather than 1) every 41 or 42 ticks, to pull it back into synch; thus some millis() values are skipped. It is not documented how it works with non AVR chips like the ESP32, however it is at least accurate to the second, and that's all that really matters for our time-keeping.

b. NTP Server

This method works by sending a request to a server on the internet (in our case, "pool.ntp.org") to ask for the current time. The server then sends a response with its time, and the ESP32 uses this information to adjust our timekeeping struct accurately. It's like asking a precise clock for the time and then correcting your watch based on the answer and how long it took to hear back.

VII. Conclusion

StepQuest is a project aiming to improve the lives of people by bettering their relationship with fitness. As it stands, StepQuest provides a user-friendly experience while providing accurate step counting and task completion in order to make fitness fun again.

THE ENGINEERS



Mohamed Al Mahruqi is an Electrical Engineer student specialized in power systems and renewable energy. Mohamed is offered a job by Entrust Solutions located in Orlando, Florida.



Jacey Klose is a 23-year-old graduating Computer Engineering Student. Jacey is taking a job with Google in Austin, TX as a Software Engineer post-graduation.



Spenser Tacinelli is an Electrical Engineering student, who previously interned at EMA, Inc. in Orlando, Florida.



William Wimberly is an 8 – year robotics teacher turned CPE major, graduating in Spring '24 from UCF in Orlando, FL. He loves hacking electronics and making all the coolest projects, with all the coolest people.

ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of the professors of the Department of Electrical and Computer Engineering. We especially thank Dr. Arthur Weeks, Dr. Lei Wei, and Dr. Chung Yung Chan.

REFERENCES

- [1] "1.28inch Round LCD Display Module with Touch panel, 240×240 Resolution, IPS, SPI And I2C Communication | 1.28inch Touch LCD," www.waveshare.com.
<https://www.waveshare.com/1.28inch-touch-lcd.htm>