TipsyTracker

By

Akash Patel
Eshrit Tiwary
Sumedh Vemugantı

Final Report for ECE 445, Senior Design, Spring 2023

TA: Dushyant Singh

03 May 2023

Project No. 17

Abstract

This paper delves into the design process and outcomes of TipsyTracker, a sophisticated system developed to encourage responsible alcohol consumption at social gatherings by precisely monitoring and managing individuals' blood alcohol content (BAC) levels. The paper commences with an analysis of the challenges that TipsyTracker seeks to resolve, followed by a comprehensive overview of the overall design. Design alternatives are subsequently examined, accompanied by a thorough exploration of the requirements and verification processes for the major subsystems. Additionally, the paper discusses the project's parts and labor aspects, ultimately concluding with a summary of the project's accomplishments and potential future improvements.

Contents

1. Introduction

1.1 Problem

Reckless alcohol consumption is an all-too-common issue, leading to a staggering 140,000 fatalities each year. To combat this crisis, it is crucial to cultivate responsible drinking habits and empower individuals to take responsibility for their actions. Establishing a system for partygoers to monitor their Blood Alcohol Content (BAC) levels and ensuring an informed host is present to address over-consumption can make a significant difference to combatting this problem.

1.2 Solution

In response to this issue, we have designed TipsyTracker, a system that promotes responsible drinking by helping individuals monitor their blood alcohol content (BAC) levels. Upon arrival, guests register at the designated station, typically the host's computer, and receive an RFID-enabled wristband or card linked to their name and phone number. After a set interval, guests will be prompted via text message to check their BAC levels using the TipsyTracker device. The device, powered by an ESP32 microcontroller, features an RFID sensor, a breathalyzer sensor, and an LED. Guests simply scan their RFID tag and, once the LED illuminates, breathe into the breathalyzer module to determine their BAC levels. This data is then transmitted to an off-PCB Raspberry Pi.

The Raspberry Pi manages the necessary software and databases, facilitates communication between the device and registration station, and dispatches notifications. If a guest's BAC surpasses a pre-established threshold, both the host and guest will be alerted, which supports responsible drinking. By fostering awareness and accountability, TipsyTracker serves as an effective solution for addressing the problem of excessive alcohol consumption.

1.3 Visual Aid

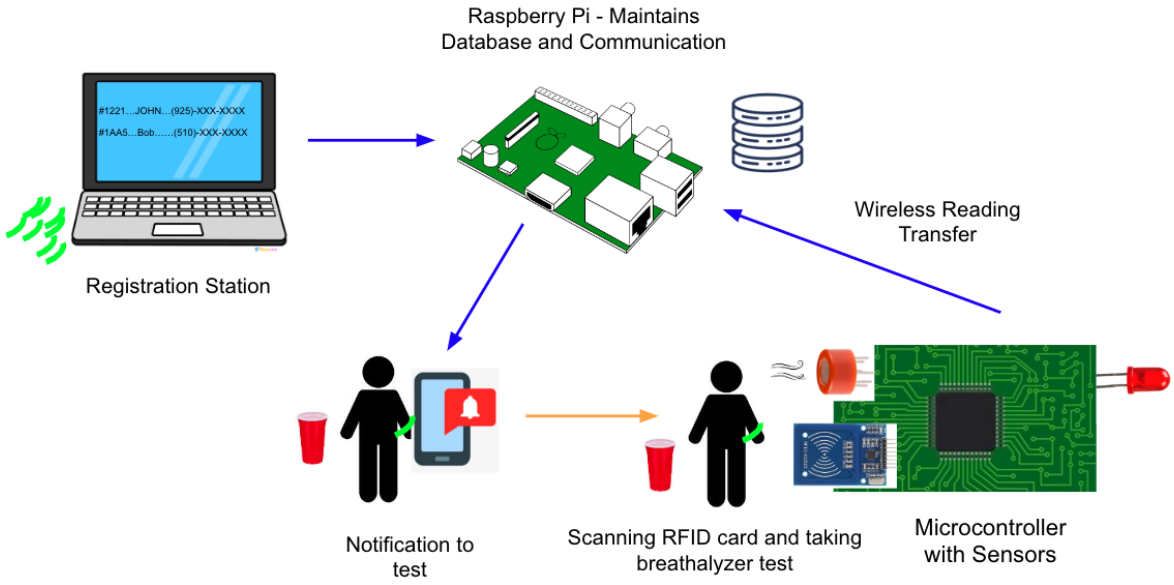

Figure 1. Visual Aid of the TipsyTracker system

1.4 High Level Requirements

1. **Precision in Alcohol Monitoring**: The device should offer exact and dependable alcohol monitoring, guaranteeing that Blood Alcohol Content (BAC) measurements remain consistent and precise within a tolerance of +/- 10% from standard values.
2. **Swift Outcomes**: The device should facilitate rapid and easy testing, providing immediate results that can be accessed on the web portal within 10 seconds or less.
3. **Capacity for Users**: The device must have the capability to store and manage BAC test results for a minimum of 500 attendees.

1.5 High Level Block Diagram

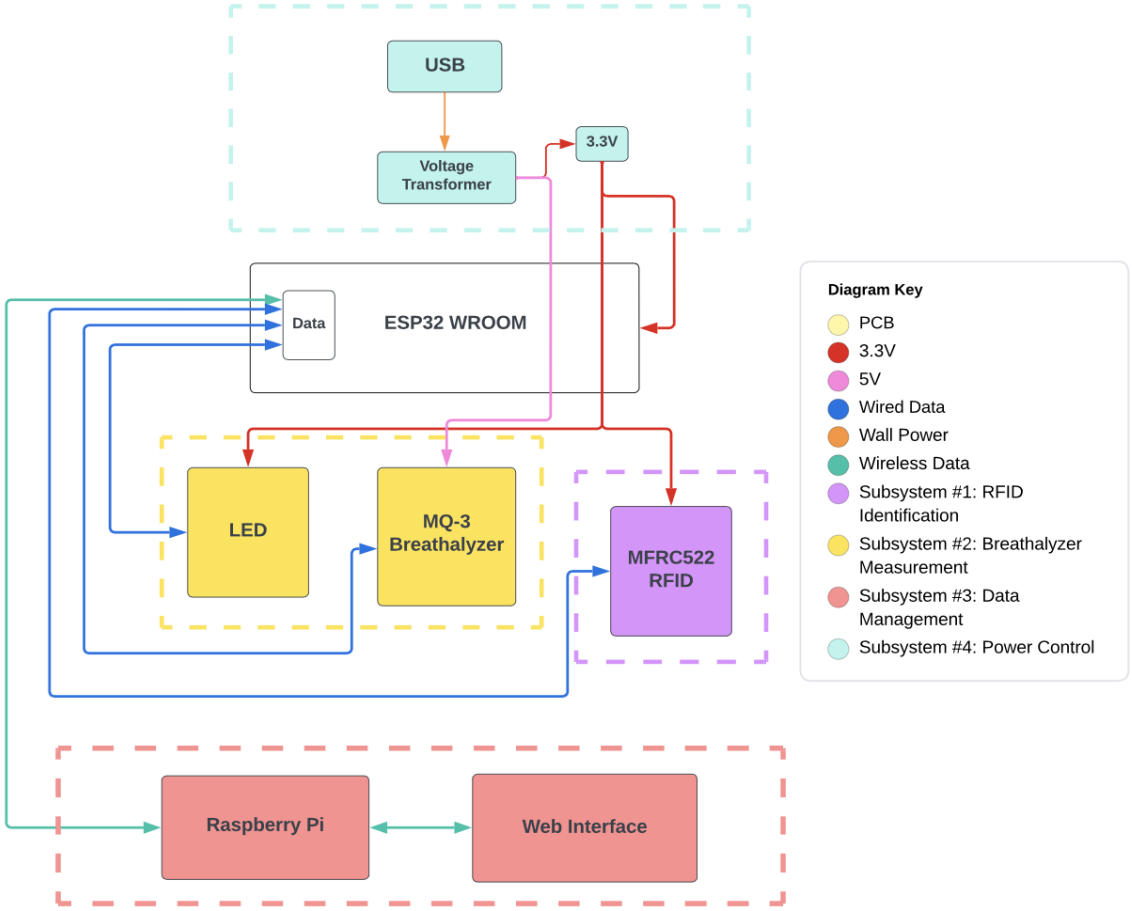

Figure 2. High Level Block Diagram of the TipsyTracker System

The TipsyTracker system is composed of four vital subsystems that guarantee proper functionality. The ESP32 microcontroller serves as the central processing unit, gathering and analyzing data from various sensors. The RFID subsystem employs an RFID reader to recognize attendees and link their BAC levels to their accounts. The breathalyzer subsystem utilizes the MQ-3 alcohol sensor to determine attendees' BAC levels and employs an LED light to assist in testing. The power subsystem, consisting of a Micro USB 5V and USB-UART and voltage transformer, delivers a stable and consistent energy supply to the system. The data management subsystem collects information from the ESP32, oversees communication

between the device and registration station, and dispatches notifications to attendees and the host. Each of these subsystems is essential to the system's overall performance.

2. Design

2.1 Breathalyzer Subsystem

The MQ-3 breathalyzer sensor measures the BAC levels of partygoers. This subsystem is linked to the ESP32 microcontroller, which works together with the RFID identification subsystem to ensure that the test results are attributed to the correct person.

When a partygoer initiates a breath test, the ESP32 sends a signal to the breathalyzer subsystem to start reading the BAC levels. The subsystem then analyzes the sample and transmits the results back to the ESP32. Using the RFID data received from the identification subsystem, the ESP32 associates the BAC reading with the correct partygoer and wirelessly sends this information to the Raspberry Pi for further analysis. A light indicates when the breathalyzer subsystem is ready for a partygoer to take a test, and this is when the ESP32 begins to read data from the MQ-3 sensor.

IO4 is used to connect the MQ-3 sensor to the ESP32 microcontroller. The 5V output of the voltage transformer powers the sensor, while the testing LED is connected to the ESP32 microcontroller via IO32 and powered by the 3.3V output of the voltage transformer.

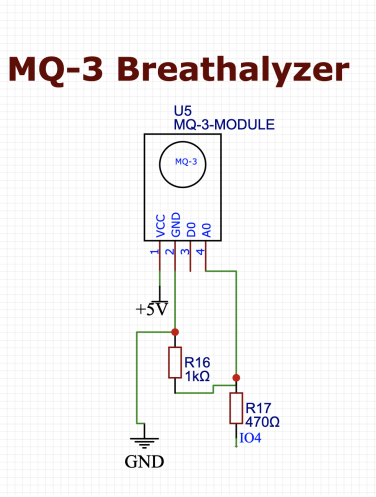

Figure 3. Breathalyzer Schematic

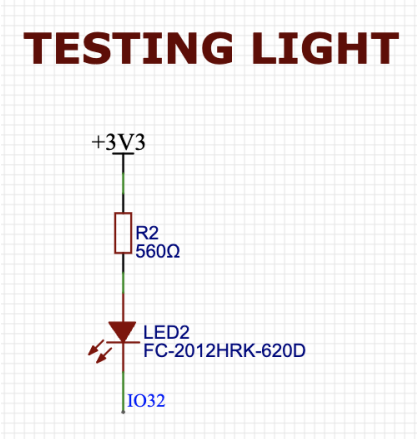

Figure 4. Testing LED schematic.

2.2 RFID Subsystem

The RFID subsystem, featuring an MFRC522 RFID sensor, serves as an essential component in offering a unique identification system for partygoers while seamlessly integrating with the Breathalyzer and Data Management Subsystem. The host assigns an RFID tag to each attendee, associating it with their name and phone number. When partygoers initiate a breath test by scanning their RFID card, the ESP-32 microcontroller captures the relevant data through the MFRC522 sensor and wirelessly transfers it to the off-PCB Raspberry Pi-powered Data Management Subsystem responsible for storing each individual's information.

The Data Management Subsystem works in harmony with the RFID subsystem, allowing for consistent notification delivery regarding the partygoers' BAC levels after each breath test. The RFID scanning process is also synchronized with the Breathalyzer subsystem. Before users begin their breath test, they must wait for a short period during which an LED blinks. After completing the test and a subsequent cooldown period, the next user receives a notification to take their test and repeats the cycle by scanning their RFID card. This streamlined process established by the RFID subsystem ensures a user-friendly experience with clear and efficient data management.

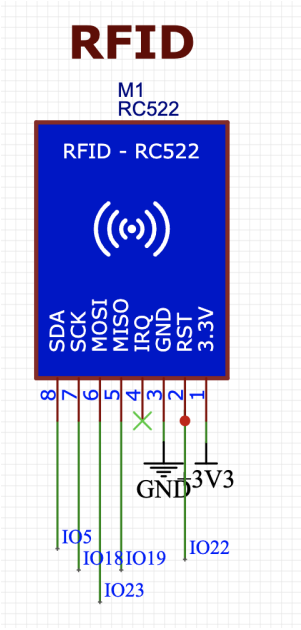

Figure 5. RFID Sensor Schematic

2.3 Data Management Subsystem

The Data Management subsystem is designed to ensure management of user's RFID data, phone numbers, and BAC readings and handle communication between the Arduino and Raspberry PI. Featuring a Raspberry Pi as the central hub for user database management, web interface hosting, and notification dispatching, the subsystem effectively streamlines data flow and communication between all elements, including the ESP32, Raspberry Pi, and host's computer. By utilizing HTTP for data transmission, the subsystem ensures a reliable and standardized method of exchanging information between the ESP32 and the Raspberry Pi.

This configuration enables real-time updates and monitoring of BAC levels, providing both attendees and the host with accurate information in a timely manner. By connecting all components to a user-established hotspot, the subsystem guarantees effective communication and a consistent data flow. The host is also promptly informed if a guest fails to comply with testing or exhibits a dangerously high BAC level, allowing them to take necessary action. Overall, this subsystem configuration delivers a user-friendly experience, prompt notifications, and a secure, organized database while effectively monitoring responsible alcohol consumption at social events.

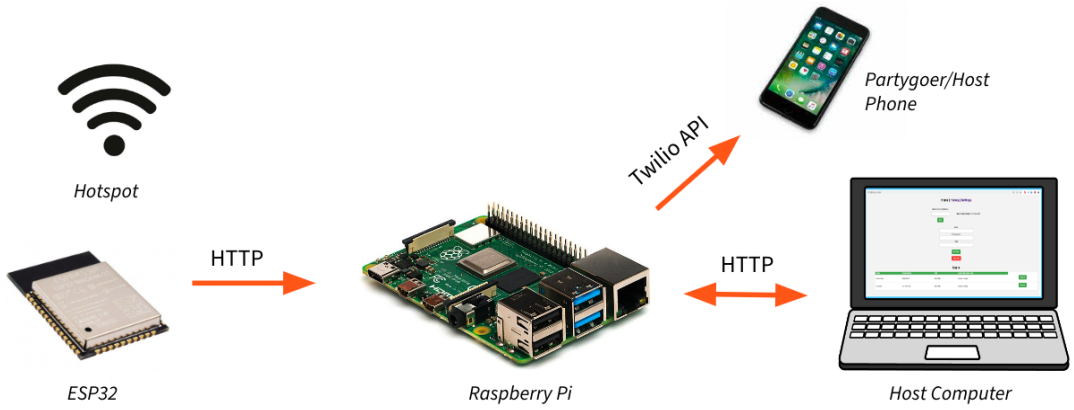

Figure 6. Visual Outline of the Data Management Subsystem

2.4 Power Subsystem

The Power Subsystem delivers and sustains power to the appropriate sensors and subsystem components. The Micro USB 5V serves as the primary power source, supplying the required voltage to establish a stable connection with the ESP32. This 5V voltage is then converted through a voltage transformer to a consistent 3.3V, meeting the operational needs of the ESP32, LED, and RFID sensor. However, the MQ-3 peripheral necessitates direct power from the 5V output. This subsystem plays an essential role in powering the components on the PCB.

Additionally, it is worth mentioning that the final design employed the ESP32 development board, which effectively managed the power subsystem, ensuring that all components were adequately powered and functioning optimally.

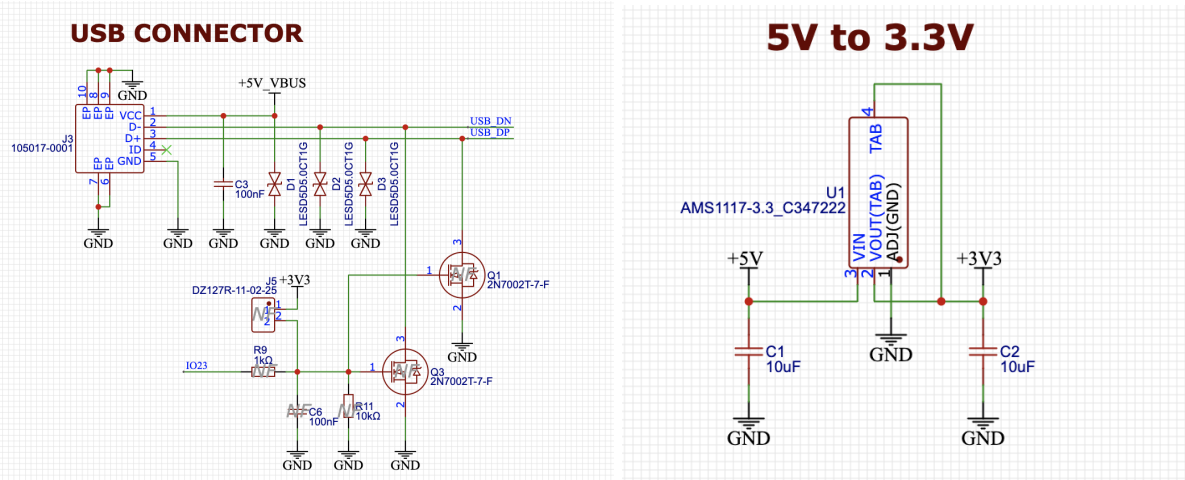

Figure 7. Schematics for the MicroUSB 5V Connector and the 5V to 3.3V AMS1117-3 Linear Voltage Regulator

2.5 PCB Design Procedure

Throughout the development process, we underwent multiple PCB iterations to refine and enhance our design. In the first iteration, we inadvertently used the ADC2 pins on the ESP32, which conflicted with the WiFi functionality, rendering the design unusable. Additionally, we employed 0402 resistors and capacitors on the PCB, which proved to be too small for easy soldering *(see Appendix B, Figure B.1)*.

In the second PCB iteration, we addressed the MQ-3 pin issue by switching to a different pin and opted for through-hole resistors and sensors to simplify the soldering process *(see Appendix B, Figure B.2)*. However, we encountered challenges in soldering the U2 component (USB to UART), which necessitated further revisions.

Faced with time constraints, we proceeded with a third PCB design that utilized the ESP32 development board. This approach allowed us to bypass the soldering issues we had experienced with the U2 component. Ultimately, we were successful in soldering and fully operationalizing the ESP32 development board PCB, achieving a reliable and functional design *(see Appendix B, Figure B.3)*.

3. Verifications

3.1 Breathalyzer Subsystem Verification

The design of the breathalyzer subsystem was guided by two key requirements: first, the ability to measure BAC levels with a high degree of accuracy, within 10% of standardized values, and second, the ability to seamlessly synchronize with the RFID and data management subsystems through an LED. To achieve these requirements, the MQ-3 sensor was programmed to output BAC levels with a deviation of less than 22% of standardized values.

To validate the accuracy of the system, we conducted a study with 10 volunteers who consumed alcohol over a period of three hours. The output of the MQ-3 sensor was compared to the results obtained from an official breathalyzer, with the comparison facilitated by serially debugging the sensor using the Arduino IDE. The resulting data, *including Figures A6 and A7 in Appendix A*, was used to extrapolate a linear model, as shown in Equation 1.

In order to ensure precise logging of BAC levels, the Breathalyzer Measurement subsystem had to be synchronized with the LED timing. During our demonstration, we showed that when the user tapped their RFID tag, the LED started blinking. Once the LED turned solid, the user initiated a breath test. Upon completion, the LED dimmed down, and the data was transmitted to the portal and the user's personal portal via the Data Management Subsystem. This process helped to ensure that BAC levels were accurately measured and recorded, providing users with a reliable tool for monitoring their alcohol consumption. *Please refer to Appendix A, which contains the detailed Requirements and Verification Table.*

3.2 RFID Subsystem Verification

We designed the RFID subsystem to ensure seamless integration with other components in the TipsyTracker device. Our team developed use cases and proactively addressed potential issues to eliminate interference between users' BAC data. To achieve this, we assigned a unique ID to each user's information (phone number, name, and BAC levels) through an RFID number and designed a user-friendly testing method.

The subsystem enables users to scan their RFID tags at a distance of three centimeters from the sensor, after which they can prepare for the test. We ensured that each RFID tag was recognized as a unique user attribute, preventing any overlap of BAC data. During the integration of the subsystem, we also successfully incorporated a cooldown period to maintain test accuracy and minimize interference between users. *To gain a deeper understanding of these requirements and their verification, please refer to Appendix A, which contains the detailed Requirements and Verification Table as well as clarifying diagrams.*

3.3 Data Management Subsystem Verification

In order to create an efficient and user-friendly system, several key requirements were established and successfully met in the final design. The Raspberry Pi was designed to receive data from the ESP32 via HTTP, ensuring a reliable and standardized method of exchanging information. Additionally, the Raspberry Pi was tasked with sending notifications to both partygoers and the host, keeping everyone informed and aware of their respective BAC levels in a timely manner. Lastly, the device was engineered to accommodate and store BAC test results for a minimum of 500 partygoers, making it an ideal solution for larger social gatherings while promoting responsible alcohol consumption. These requirements were essential in crafting a system that effectively addresses the issue of excessive alcohol consumption at social events. *To gain a deeper understanding of these requirements and their verification, please refer to Appendix A, which contains the detailed Requirements and Verification Table as well as clarifying diagrams.*

3.4 Power Subsystem Verification

For the Power Subsystem, several critical requirements were set and achieved to ensure the seamless functioning of the entire system. First, the subsystem supplied the MQ-3 sensor with a 5V power source, enabling accurate and reliable alcohol sensing. Second, it converted the 5V input to a consistent 3.3V output in order to power the ESP32, LED, and RFID sensors effectively. These carefully designed requirements were essential in establishing an efficient power management system that supported the operational needs of each component, contributing to the overall effectiveness and reliability of the device. *To gain a deeper understanding of these requirements and their verification, please refer to Appendix A, which contains the detailed Requirements and Verification Table as well as clarifying diagrams.*

4. Costs

4.1 Final Parts Breakdown

Description	Quantity	Manufacturer	Price
1kΩ resistor	1	BOJACK	$0.0005
560Ω resistor	1	BOJACK	$0.0005
470Ω resistor	1	BOJACK	$0.0005
ESP 32 WROOM Development Board	1	Flutesan	$5.996
MQ-3 Alcohol Sensor	1	Sparkfun	$2.31
Mifare RC522	1	HiLetgo	$5.69

Parts Cost: **$13.9975/unit**

4.2 Labor

Assuming a reasonable salary for an ECE graduate in Illinois of $35 per hour, and a total of 150 hours of work for three partners, the total labor cost is:

*$35/hr/person * 150 hrs * 3 people = $15,750*

Total Labor Cost: **$15,750**

Total Cost (including parts): **$15,763.9975**

5. Conclusion

5.1 Accomplishments

The development of Tipsy Tracker involved the implementation of Power, Data Management, RFID and Breathalyzer subsystems, which were crucial in achieving accurate BAC measurements. After successfully designing and integrating all subsystems, a final demo was conducted to showcase the interaction of the subsystems. The demo yielded positive results, with all

subsystems performing seamlessly and interacting well with one another, and synchronizing well with one another.

BAC data generated by Tipsy Tracker was compared to that of a commercial breathalyzer, which served as a benchmark for accuracy. The results were promising, as there was a strong correlation between Tipsy Tracker's BAC data and that of the commercial breathalyzer. This demonstrated that Tipsy Tracker was capable of accurately measuring BAC levels with a high level of precision. The strong correlation between Tipsy Tracker's BAC data and that of the commercial breathalyzer further validated the effectiveness of the system, making it a reliable tool for monitoring and maintaining safe levels of alcohol consumption.

5.2 Uncertainties

In the development of TipsyTracker, there are several uncertainties that could be addressed to improve the system's performance and reliability. One such uncertainty is the accuracy of the BAC measurements, which could be further validated through additional testing utilizing a more professional grade breathalyzer. This would provide a benchmark for accuracy and help to ensure that TipsyTracker's measurements are consistent and reliable.

Another area of uncertainty is the use of a Raspberry Pi for data storage and management. While the current setup is effective, implementing cloud solutions could provide greater scalability and flexibility, allowing users to access their data from anywhere and ensuring that it is stored securely.

Finally, the final packaging of the PCB is also an area that could benefit from further attention. A stronger, more durable packaging solution would help to protect the system from damage and ensure that it can be used safely and reliably in a variety of environments.

5.3 Ethical Considerations

Ethics and safety were essential considerations during the development of TipsyTracker. One such ethical feature of the system is the "Party Over" button, which is designed to delete all user data at the end of the party. This ensures that users' personal data is not stored unnecessarily and helps to maintain their privacy. In addition to privacy concerns, safety is also a top priority for the system. The TipsyTracker features a water-resistant casing that helps to mitigate shock and electric safety hazards, reducing the risk of accidents and ensuring that the system can be used safely in various environments.

Finally, it is important to ensure that users are aware of the system's status, and any potential issues that may arise. To this end, the TipsyTracker includes a notification feature that alerts users when the system is down or experiencing technical difficulties. This ensures that users are not relying on inaccurate or unreliable BAC data and can make informed decisions about their alcohol consumption.

5.4 Future Work

There are several areas that can be improved to enhance the system's functionality and user experience. One potential avenue for future work is the implementation of a centralized control mechanism for multiple TipsyTracker devices. This would allow users to monitor and manage multiple devices preventing large queues at a single location, making it easier to keep track of BAC levels for groups of people or at events.

In addition to centralized control, further work can be done to improve the accuracy of the system's BAC measurements. While the current relationship between the MQ-3 sensor and actual BAC percentage is effective, there may be more fitting options that could be explored to enhance the system's accuracy even further.

Another potential improvement is the addition of NFC technology for mobile identification. This would allow users to quickly and easily identify themselves to the system, eliminating the need for users to wear a wristband or carry around an RFID card. This could streamline the user experience and further improve the accuracy and reliability of the system.

Appendix A: Requirement and Verifications

RFID Subsystem R and Vs:

Requirements	Verification
• The RFID sensor must be able to read tags within a range of at least 3 cm.	• Use a known RFID tag and place it at varying distances from the RFID reader antenna within the required range. Confirm that the tag is successfully read each time via the board microcontroller serial debugging, by printing the outputs with the Arduino IDE.

• The RFID sensor must be able to distinguish between different RFID tags.	• Use multiple known RFID tags and place them within the required range of the RFID reader antenna. Confirm that the board microcontroller identifies each tag and displays its unique identification number via serial debugging by printing the outputs with the Arduino IDE.
• The RFID subsystem must be reliably polled by the microcontroller to ensure no missed scans and accurate monitoring of tag readings.	• Set up a test scenario where multiple RFID tags are present within range of the reader. • Configure the microcontroller to poll the RFID subsystem at a specific interval, and record the tag readings received by the microcontroller. • Increase the speed of the RFID tags passing through the reader and repeat the test. • Confirm that the microcontroller continues to poll the RFID subsystem at the configured interval and all tag readings are accurately recorded by serial debugging (through the Arduino IDE).

RFID	Distance	Detected
A	1 cm	Yes
A	2 cm	Yes
A	3 cm	Yes

Figure A.1. RFID table demonstrating successful detection of RFID tags at varying minimum distances

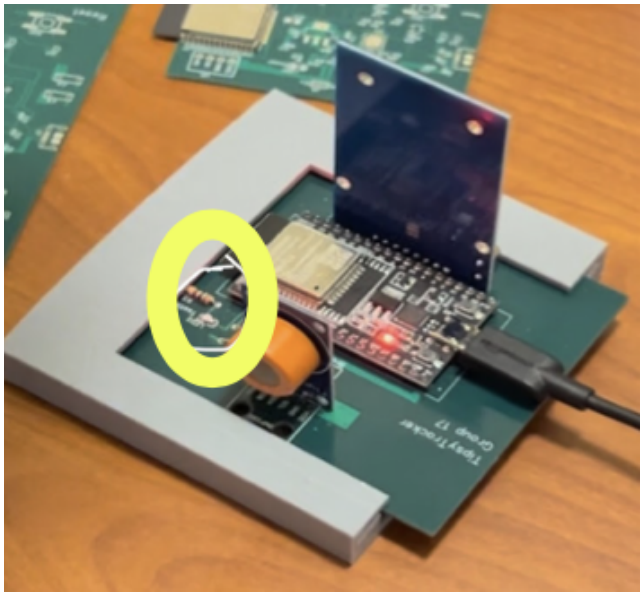

Figure A.2. Unlit LED showing cooldown period before the beginning of the next breathalyzer test

Data Management Subsystem R and Vs:

Requirements	Verification
• The Raspberry Pi must receive and store data from the ESP32 through HTTP.	• Configure the ESP32 and Raspberry Pi to communicate through HTTP. • Transmit test data from the ESP32 to the Raspberry Pi and ensure it is received and stored correctly • Monitor the data stored on the Raspberry Pi and verify that it matches the data transmitted from the ESP32 • In case of communication errors, use debugging tools to identify and resolve issues.
• The Raspberry Pi must send notifications to partygoers and the host.	• Set up a test notification schedule and ensure that notifications are sent to partygoers and the host at the designated times.

	• Monitor the notifications received by partygoers and the host and verify that they match the notification schedule • In case of notification failures, use debug the notification sending program on the Raspberry Pi
• The Raspberry Pi must be able to accommodate the data of 500 partygoers	• Create 500 dummy users to demonstrate accommodation of this size in the database

Test Case	RFID Tag	Bac Level	Expected Outcome	Actual Outcome
1	b449a889	0.03	Data received and stored	Data received and stored
2	c23a9111	0.04	Data received and stored	Data received and stored
...	...	...	...	...
300	j34k1501	0.1	Data received and stored	Data received and stored

Figure A.3. Table demonstrating successful verification of data sending via HTTP

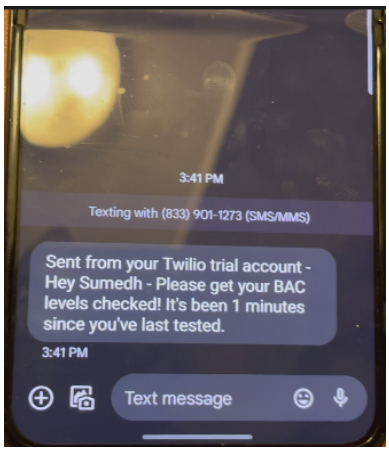

Figure A.4. Screenshot demonstrating successful verification of notifications

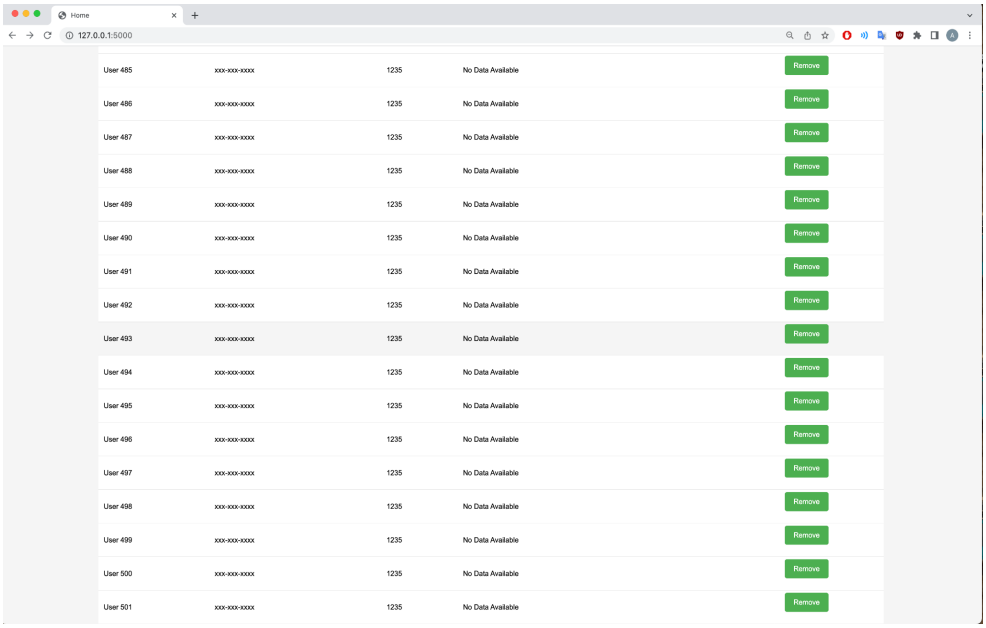

Figure A.5. Screenshot demonstrating verification of accommodating 500 partygoers

Breathalyzer Subsystem R and Vs

Requirements	Verification
• The MQ-3 sensor shall be able to accurately measure BAC levels with a deviation of less than 10% of standardized values.	• See Test 1 and Test 2 plots and equations. False Pos + False Neg = 22%
• The Breathalyzer Measurement subsystem must synchronize with the LED timing, ensuring that the ESP32 starts logging BAC levels precisely when the LED illuminates	• Verified visually & through discussion of LED behavior as Sumedh and Eshrit test.

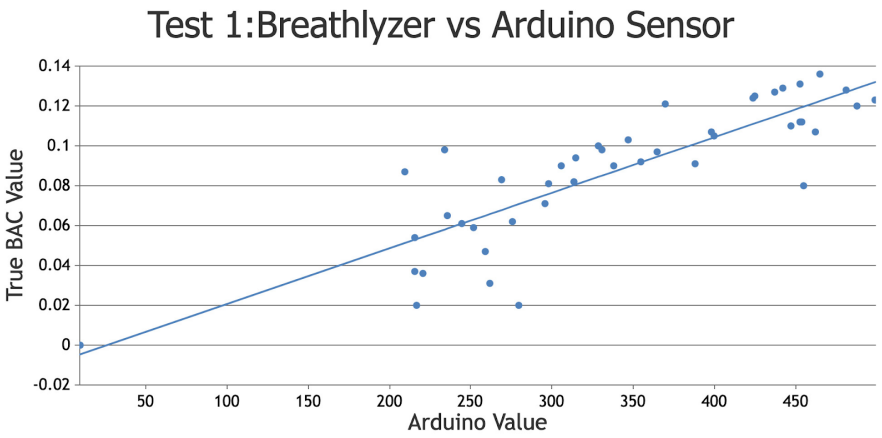

Figure A.6. Plot of Breathalyzer BAC levels vs Arduino Analog Values

Predicted BAC = 0.00028 * Arduino Value - 0.00739

Equation 1. BAC Levels vs Arduino Analog Values

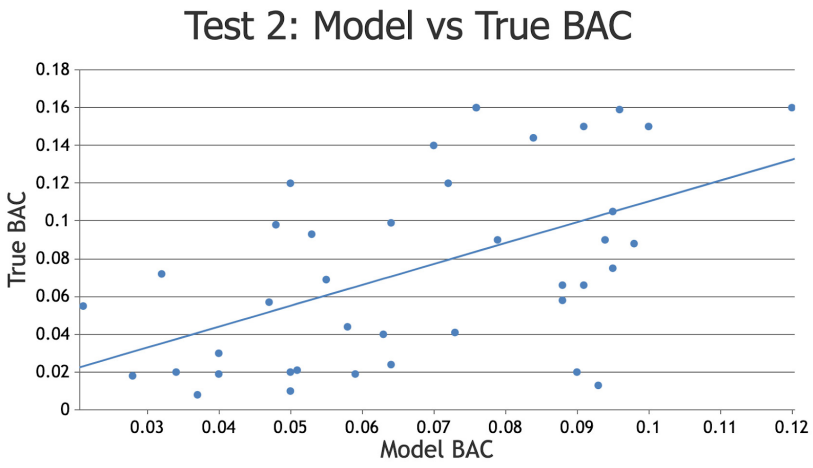

Figure A.7. Plot of Breathalyzer BAC levels vs MQ3's BAC

Power Subsystem R and Vs:

Requirements	Verification
• The Power Subsystem must be able to supply the MQ-3 sensor with a steady 5V	• Measure the current and voltage output of the Power Subsystem (when drawing from USB power) using a multimeter during MQ-3 sensor operation to ensure that it meets the requirement of 5V.
• Must convert 5V to 3.3V to power the ESP32, LED, and RFID sensors	• Measure the current and voltage output of the Power Subsystem using a multimeter during ESP32, LED, and RFID sensor operation to ensure that it meets this correct output voltage. We can probe different areas in the PCB (exposed copper areas) to see if it's receiving the correct output.

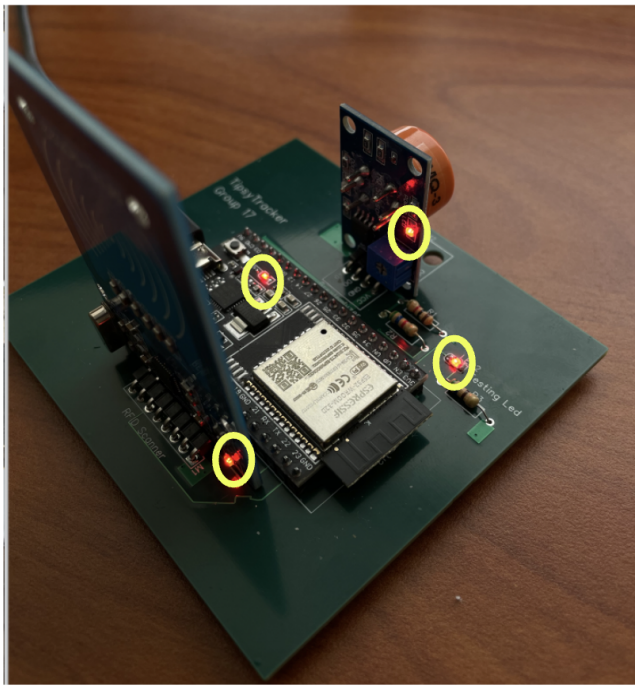

Figure A.8. Verification of power subsystem showing all sensors lit up

Appendix B: PCB Designs

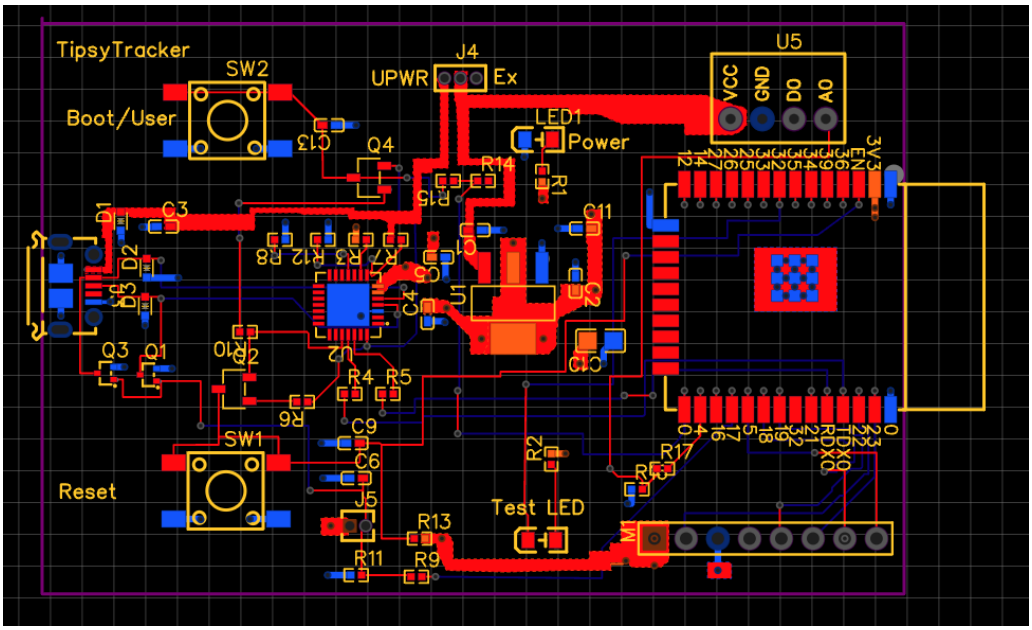

Figure B.1. PCB Design 1 Layout

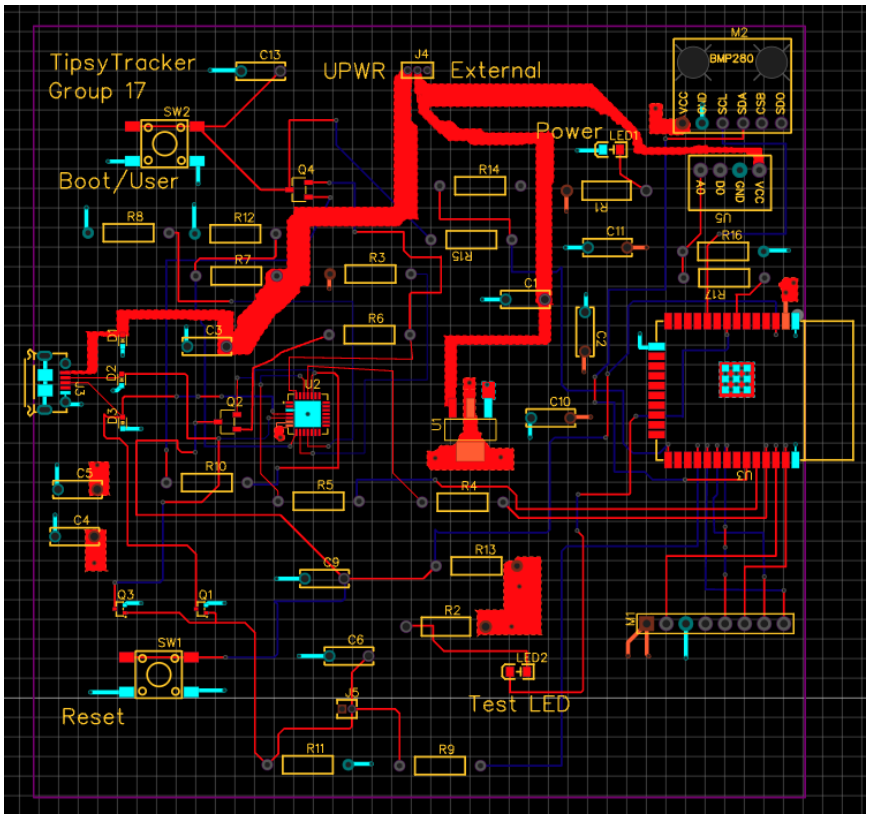

Figure B.2. PCB Design 2 Layout

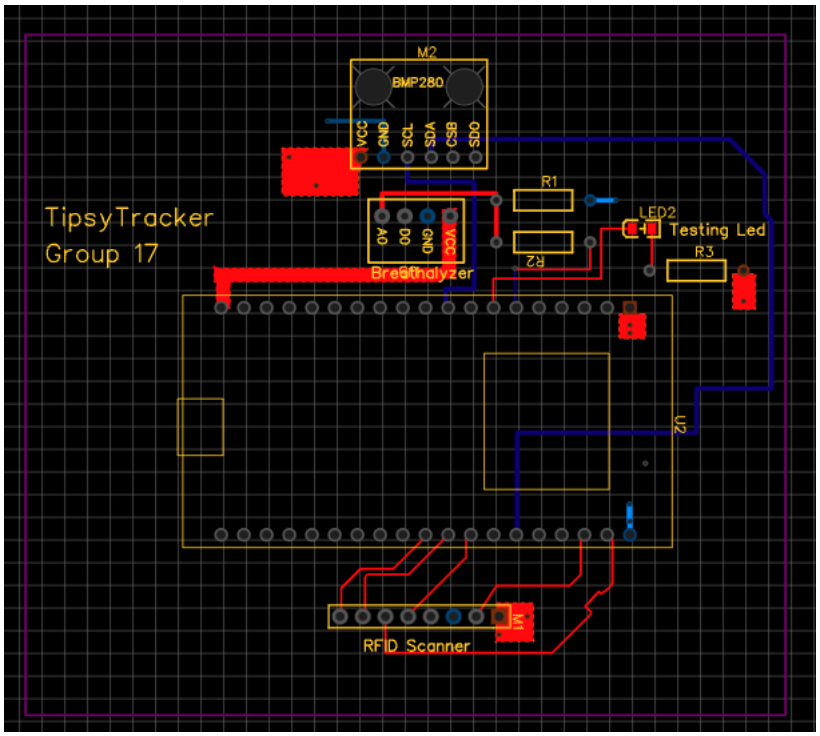

Figure B.3. PCB Design 3 Layout

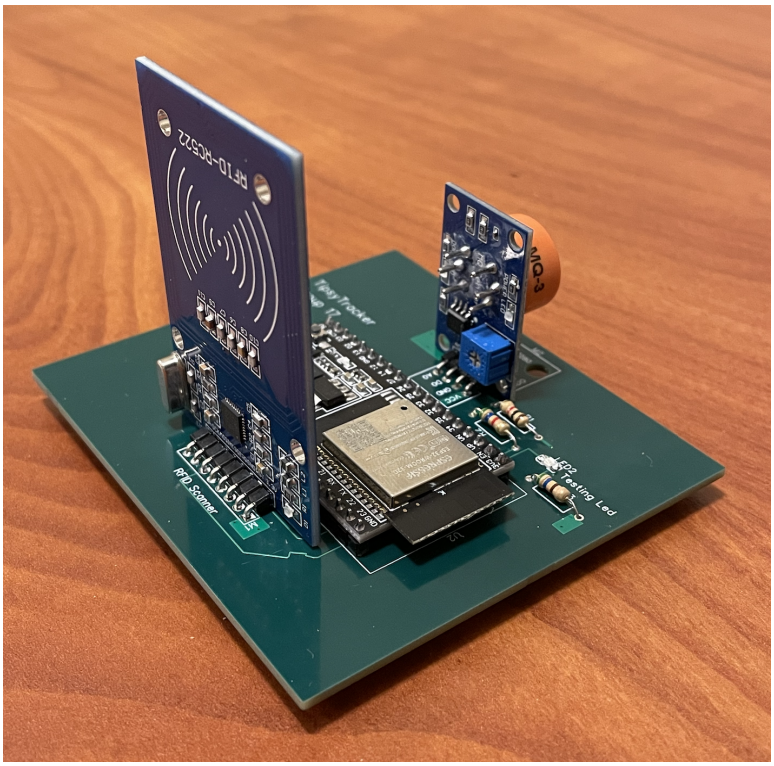

Figure B.4. PCB Design 3 Final Solder

References

[1] “Development boards | ESPRESSIF systems.” [Online]. Available: https://www.espressif.com/en/products/devkits. [Accessed: 26-Mar-2023].

[2] “IEEE code of Ethics.” [Online]. Available: https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-code-of-ethics.pdf. [Accessed: 26-Mar-2023].

[3] “Power supply ESP32 module - Espressif.” [Online]. Available: https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf. [Accessed: 26-Mar-2023].

[4] “RF Wireless World,” *ESP32 Arduino Interfacing with Gas sensor diagram, working, code*. [Online]. Available: https://www.rfwireless-world.com/ApplicationNotes/ESP32-interfacing-with-Gas-sensor.html. [Accessed: 26-Mar-2023].

[5] Xukyo, “Using an RFID module with an ESP32 • aranacorp,” *AranaCorp*, 03-May-2021. [Online]. Available: https://www.aranacorp.com/en/using-an-rfid-module-with-an-esp32/. [Accessed: 26-Mar-2023].