TispyTracker

Ece 445 Design Document - Spring 2023

Akash Patel

Sumedh Vemuganti

Eshrit Tiwary

Contents

1. Introduction

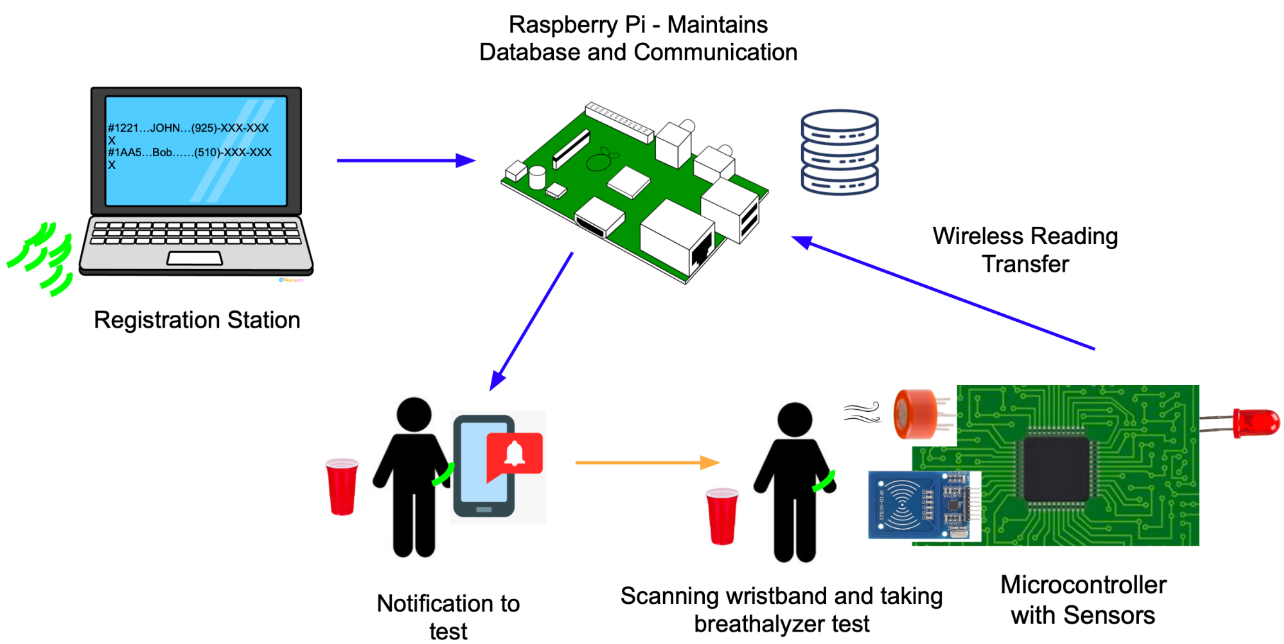
1.1. Problem

Irresponsible drinking is a widespread problem, especially among university students. Unfortunately, many people often lose control of their alcohol consumption simply due to a lack of awareness of how much they have consumed. The statistics are clear: according to the Centers for Disease Control and Prevention (CDC), excessive alcohol consumption leads to the deaths of more than 140,000 individuals each year (NPR 2022). By promoting responsible drinking habits and encouraging individuals to take control of their own actions, we can work towards reducing the harm caused by this preventable issue. Our mission is to proactively tackle this challenge head-on, particularly during social gatherings and parties, by developing a cutting-edge system that monitors alcohol intake.

1.2. Solution

To address this, we have developed TipsyTracker, a system that encourages responsible drinking and keeps individuals informed about their blood alcohol content (BAC) levels. Upon arrival, guests will sign-in to the registration station (the host's computer) and will be given an RFID-enabled wristband/card that is attributed to their name and phone number. After a certain time period, the guest will be notified (via text) to scan their BAC levels at the TipsyTracker device. The device is controlled by an ESP32 microcontroller and consists of an RFID sensor, breathalyzer sensor, and an LED. The guest will scan their RFID tag, and once the LED turns on, breathe into the breathalyzer module that calculates their BAC levels. This information is then sent to an off-PCB Raspberry Pi. The Pi maintains the necessary software and databases, handles communication between the device and registration station, and sends notifications. If a guest's BAC level exceeds a predetermined limit, the host and guest will be notified, which encourages safe drinking. By promoting awareness and responsibility, TipsyTracker is an effective solution to the issue of excessive alcohol consumption.

1.3. Visual Aid

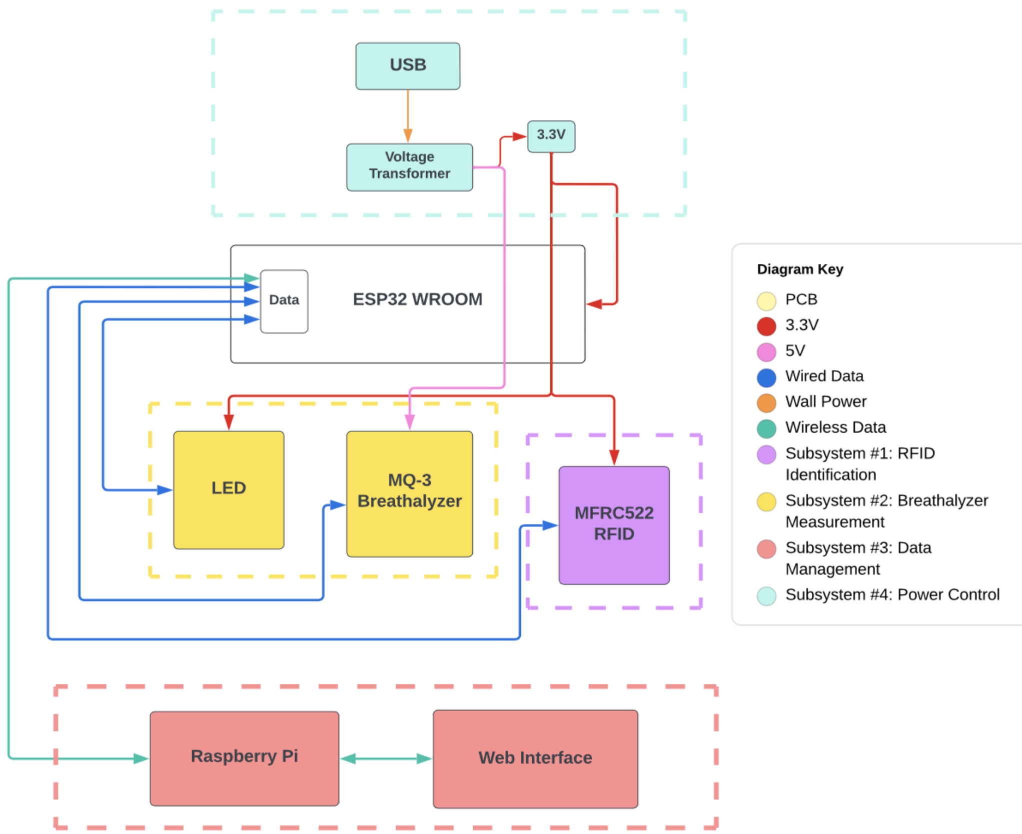


1.4. High-Level Requirements

1. Accurate Alcohol Monitoring: The device must provide precise and reliable alcohol monitoring, ensuring that Blood Alcohol Content (BAC) readings are consistent and accurate within a +/- 10% tolerance of standard values.
2. Instant Results: The device must enable quick and convenient testing, delivering instant results that are available on the web-portal in less than 10 seconds.
3. User capacity: The device must be able to accommodate and store BAC test results for at least 500 partygoers.

2. Design

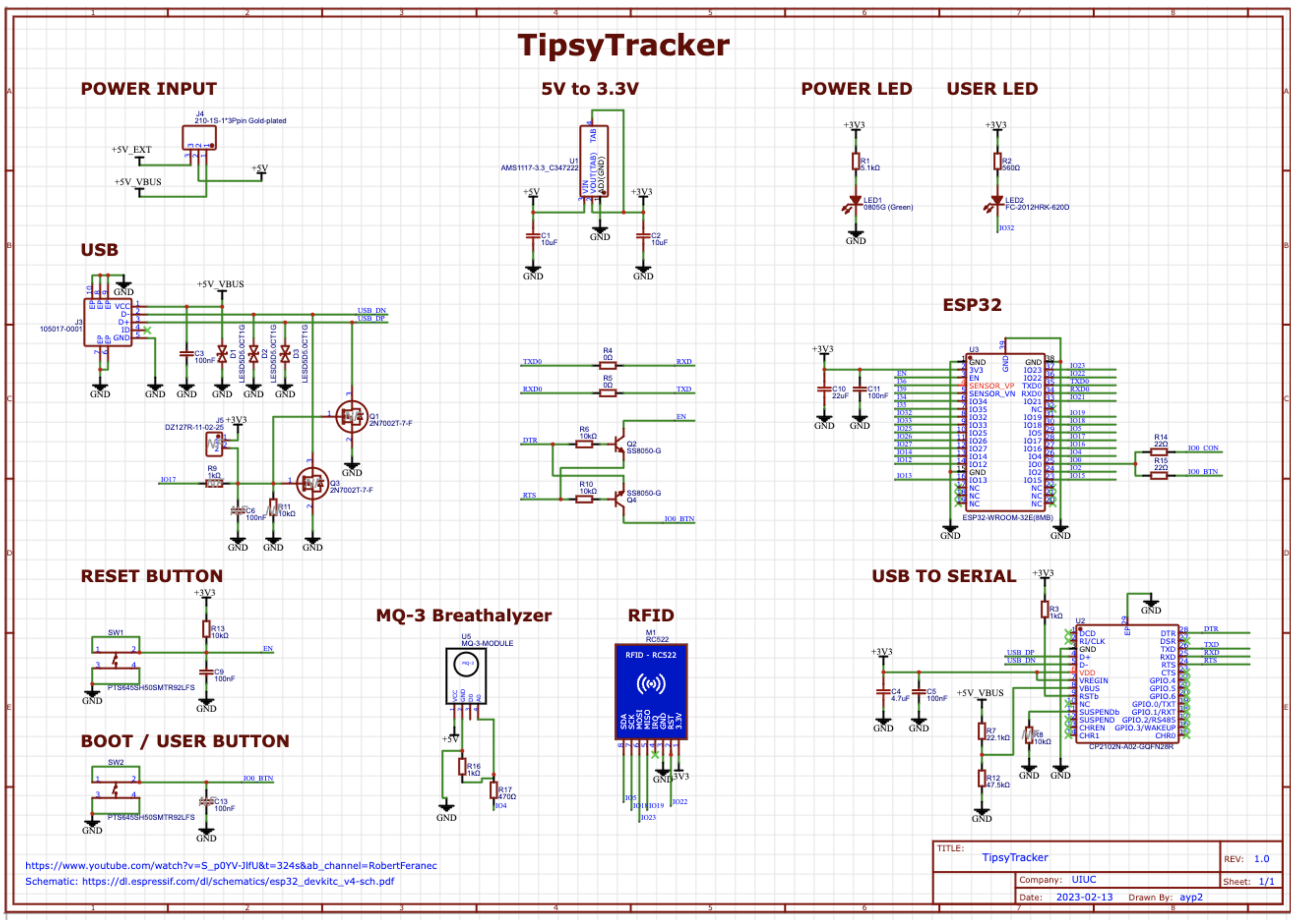
2.1. Block Diagram



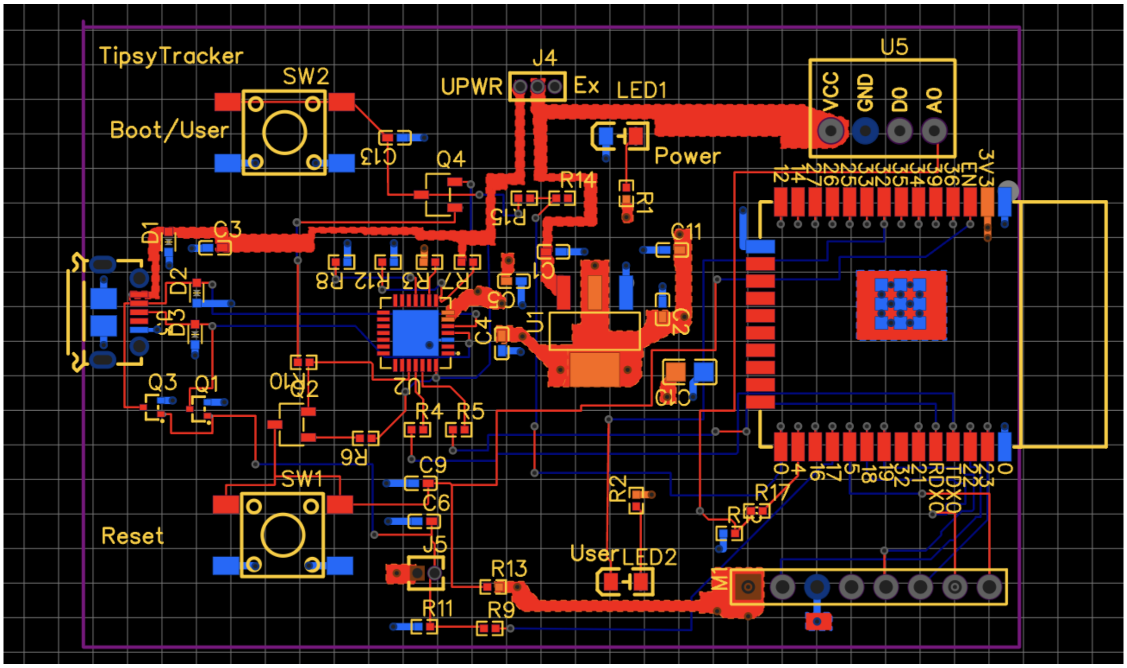
The system comprises several critical subsystems that work together to ensure seamless operation. The ESP32 microcontroller acts as the central processing unit, collecting and processing data from various sensors. The RFID subsystem uses an RFID reader to identify partygoers and associate their BAC levels with their respective accounts. The breathalyzer subsystem leverages the MQ-3 alcohol sensor to measure partygoers' BAC levels and uses an LED light to indicate the readings. The power subsystem, which consists of a Micro USB 5V&USB-UART and voltage transformer, provides a stable and consistent power supply to the system. The data management subsystem collects data from the ESP32, manages the communication between the device and the registration station, and sends notifications to partygoers and the host. Each of these subsystems is crucial to the system's overall function, and any malfunction or deviation from requirements can result in adverse outcomes.

2.2. Schematic and PCB Layout

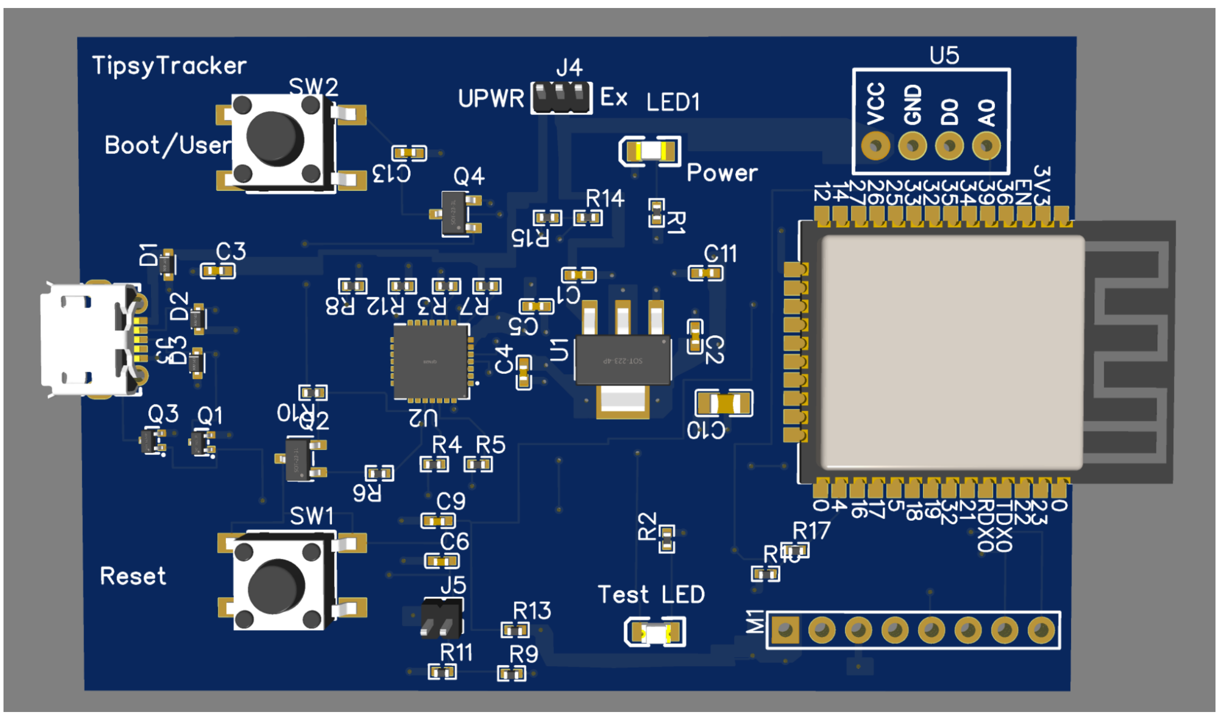
Schematic



PCB Layout



PCB 3d View



2.3. Subsystem Overview and Requirements

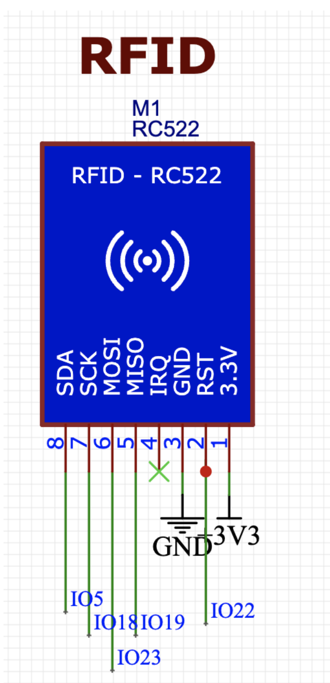
2.3.1. Subsystem 1: RFID Identification

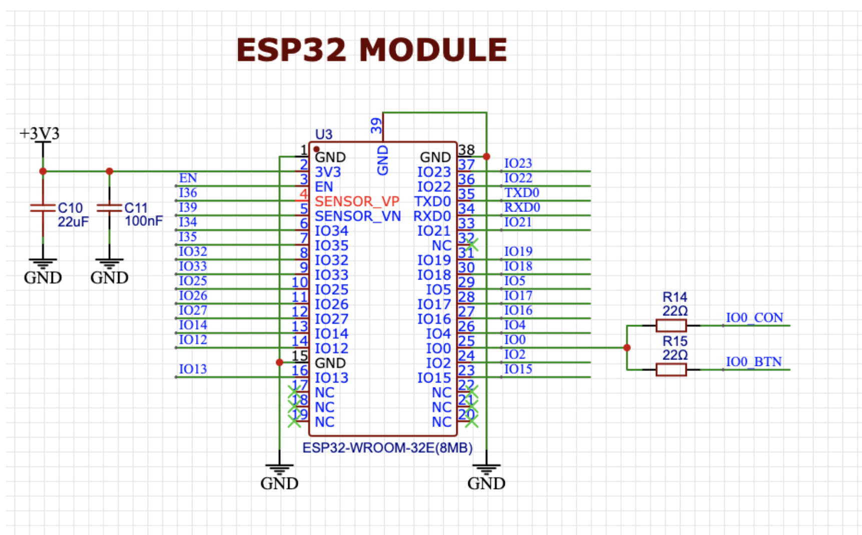
Overview:

This subsystem will be responsible for identifying each partygoer by reading their RFID-enabled wristband/card when they initiate a breath test, utilizing the MFRC522 RFID sensor. Ultimately, the sensor's readings will attribute the Breathalyzer Measurement subsystem's readings to an individual partygoer. This happens because the ESP32 will wirelessly transmit the RFID data to the data management subsystem (off-pcb and on the Raspberry Pi), which will correlate the RFID to the user's name and phone number for further analysis.

The subsystem will be connected to the ESP32 microcontroller via IO5, IO18, IO23, IO19, and IO22. It will be powered by the 3.3V output of the voltage transformer.

Schematics:





Requirements and Verification:

Requirements	Verification
• The RFID subsystem must be able to read RFID tags within a range of at least 10 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 subsystem 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 IDE).

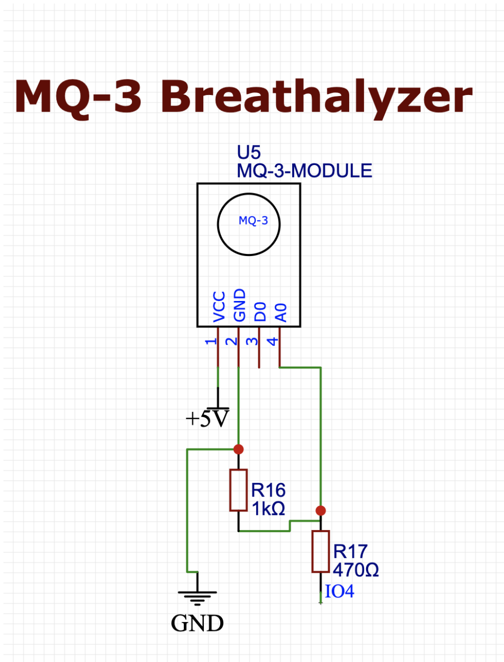
2.3.2. Subsystem 2: Breathalyzer Measurement Subsystem

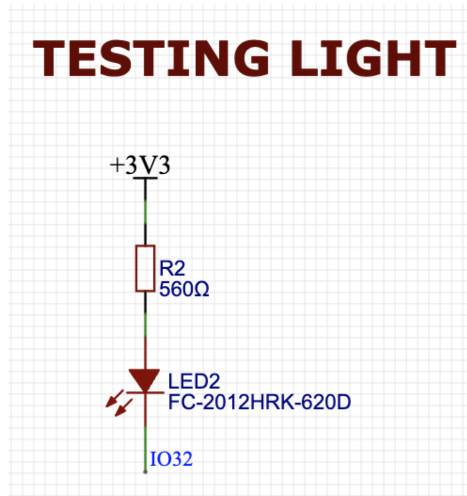
Overview:

The breathalyzer subsystem is responsible for measuring the BAC levels of the partygoers using the MQ-3 breathalyzer sensor. It is connected to the ESP32 microcontroller, which communicates with the RFID identification subsystem to ensure that the test results are associated with the correct partygoer. When a partygoer initiates a breath test, the ESP32 sends a signal to the breathalyzer subsystem to begin reading the BAC levels. The subsystem then analyzes the sample and sends the result back to the ESP32. The ESP32 uses the RFID data received from the identification subsystem to associate the BAC reading with the correct partygoer, and wirelessly transmits this data to the Raspberry Pi for further analysis. A light will turn on to indicate when the breathalyzer subsystem is ready for a partygoer to take a test, and is when the ESP32 will begin reading data from the MQ-3 sensor.

The MQ-3 sensor will be connected to the ESP32 microcontroller via IO4. It will be powered by the 5V output of the voltage transformer. The testing LED will be connected to the ESP32 microcontroller via IO32. It will be powered by the 3.3V output of the voltage transformer.

Schematics:





Requirements and Verification:

Requirements	Verification
• The MQ-3 sensor shall be able to accurately measure BAC levels with a deviation of less than 10% of standardized values.	• The MQ-3 sensor shall be calibrated using a known alcohol concentration and verified for accuracy within a tolerance of ±10% of the intended value. We will test this by utilizing an official breathalyzer and comparing results. We will view the output of the MQ-3 sensor by serially debugging it with the Arduino IDE.
• The Breathalyzer Measurement subsystem must synchronize with the LED timing, ensuring that the ESP32 starts logging BAC levels precisely when the	• Connect the Breathalyzer Measurement subsystem and LED subsystem to the ESP32 microcontroller. • Upload a test program to the microcontroller that turns on the LED for a specified duration and logs BAC levels

LED illuminates.	during that time. ● Observe the LED to ensure that it turns on at the correct time and for the correct duration. ● Observe the BAC logging data to ensure that it corresponds with the timing of the LED turning on. ● If the timing does not match, use the Arduino IDE serial monitor to debug the code and adjust the timing until the LED and BAC logging are synchronized.

2.3.3. Subsystem 3: Data Measurement Subsystem

Overview:

This subsystem will be responsible for handling the communication between the device and the registration station, as well as sending notifications to partygoers and the host. It will be powered by a Raspberry Pi, which will handle data storage, analysis, and management of the entire system. The ESP32 microcontroller will interface with the Raspberry Pi via MQTT, sending data collected from the breathalyzer and RFID sensors. The data management subsystem will then process the data and generate notifications to partygoers at set intervals to remind them to test their BAC levels, as well as notify the host of any concerning readings. This seamless integration between the subsystems is crucial for the overall accuracy and efficiency of the system.

Requirements and Verification:

Requirements	Verification
● The Raspberry Pi must receive and store data from the ESP32 through MQTT.	● Configure the ESP32 and Raspberry Pi to communicate through MQTT. ● Transmit test data from the ESP32 to the Raspberry Pi and ensure it is received and stored correctly ● Monitor the data stored on the

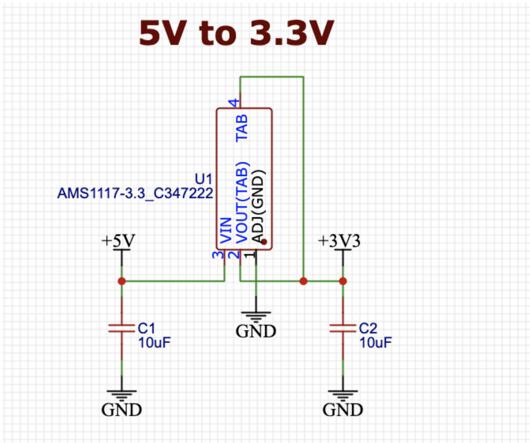
	Raspberry Pi and verify that it matches the data transmitted from the ESP32 ● In case of communication errors, use debugging tools such as MQTT Spy to identify and resolve issues.
● The Raspberry Pi must send notifications to partygoers and the host.	● Set up a test notification schedule and ensure that 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

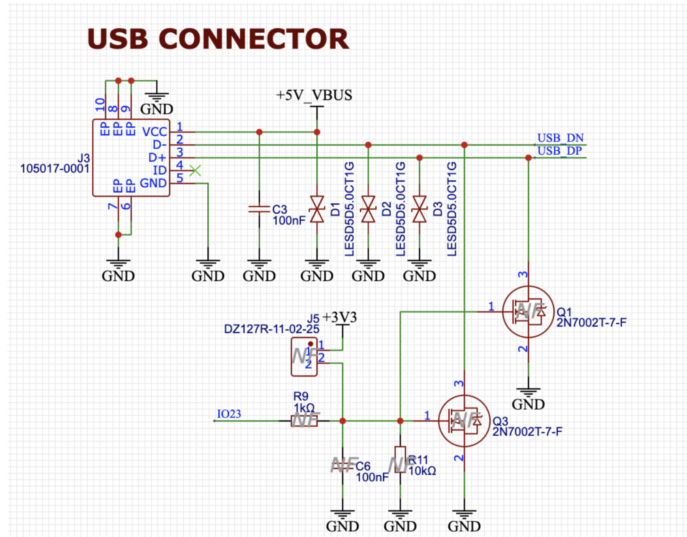
2.3.4. Subsystem 4: Power Control

Overview:

The Power Subsystem will be a crucial component in ensuring the seamless functioning of the entire system. The Micro USB 5V & USB-UART will serve as the primary source of power, providing the necessary voltage to establish a stable connection with the ESP32. The 5V voltage from the USB-UART will be then transformed through a voltage transformer to a consistent 3.3V, which will sustain the operational needs of the ESP32, LED, and RFID sensor. However, the MQ-3 peripheral will need to be directly powered by the 5V output. This subsystem will play a vital role in ensuring the components on the PCB are powered.

Schematics:



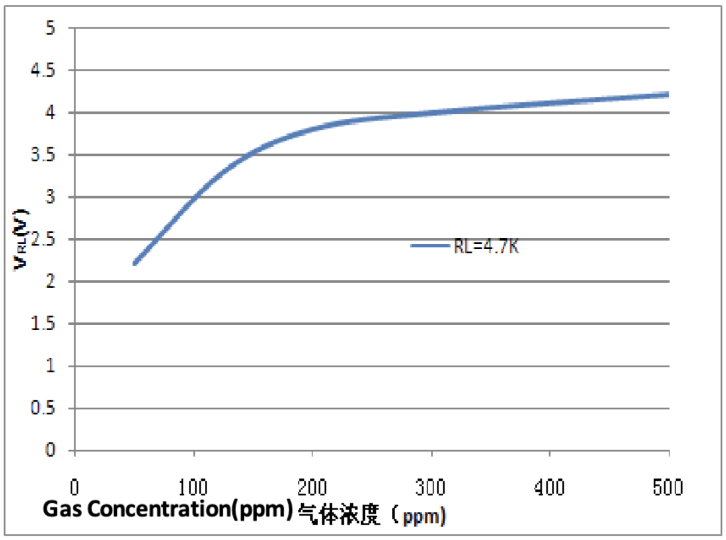


Requirements and Verification:

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.

2.4. Tolerance Analysis

Datasheet Conversion from MC3 Voltage to PPM



Let **X'** and **Y** be the analog readings from the MQ-3 sensor and a standardized breathalyzer tool, respectively. X will be an analog value 1023. We will map X' to a number between 0V and 5V linearly. Using the data sheet above, we will convert from Volts to ppm, the inverse of which will give us **X** in mg/dL (BAC units). We will then perform the following analysis between **X** and **Y.**

The Pearson correlation coefficient, denoted by r, is calculated as:

r = cov(X, Y) / (std(X) * std(Y)) [Eq1]

where **cov(X, Y)** is the covariance between **X** and **Y** and **std(X)** and **std(Y)** are the standard deviations of **X** and **Y,** respectively. The covariance is calculated as:

cov(X, Y) = E[(X - E[X])(Y - E[Y])

where **E[X]** and **E[Y]** are the expectations of **X** and **Y**, respectively. The standard deviation is calculated as:

std(X) = sqrt(var(X))

where **var(X)** is the variance of **X**.

The Pearson correlation coefficient is a measure of the linear relationship between X and Y. If the correlation coefficient is close to 1, it indicates a strong positive relationship between the two measurements, which is necessary for the TipsyTracker to work reliably. On the other hand, a correlation coefficient close to -1 indicates a strong negative relationship, while a correlation coefficient close to zero indicates a weak relationship between the two measurements. The correlation coefficient should be calculated and analyzed to assess the accuracy of the MQ-3 sensor and ensure that the TipsyTracker works reliably.

3. Cost and Schedule

Parts:

Description	Quantity	Manufacturer	Extended Price
ESP32-WROOM-32E	1	Espressif	$5.99
Mifare RC522	1	HiLetgo	$5.69
MQ-3 Alcohol Sensor	1	Sparkfun	$2.31
C0402 10uF Capacitor	2	Samsung Electro-Mechanics	$0.014
C0402 100nF Capacitor	9	Samsung Electro-Mechanics	$0.0613
C0402 4.7uF Capacitor	1	Samsung Electro-Mechanics	$0.067
C0805 22uF Capacitor	1	Samsung Electro-Mechanics	$0.067
USB Connector: MICRO-USB-SMD_1050 17-0001	1	Molex	$0.24
5.1kΩ R0402	1	UniOhm	$0.0005

560Ω R0402	1	UniOhm	$0.0005
1kΩ R0402	2	UniOhm	$0.001
10kΩ R0402	5	UniOhm	$0.0025
22.1kΩ R0402	1	UniOhm	$0.0005
47.5kΩ R0402	1	UniOhm	$0.0005
PTS64 Switch	2	C&K	$0.258
AMS Voltage Regulator	1	Youtai Semiconductor Co., Ltd.	$0.0383
CP2102N USB-IC	1	Silicon Labs	$3.63

Parts Cost: $**18.37 / unit**

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: **$15,768.37**

Schedule:

Week	Task	Person
February 20th - February 27th	Review and order necessary parts and components	Everyone

	Finalize the system architecture	Everyone
	Develop initial schematics for the printed circuit board (PCB)	Akash
February 27th - March 3rd	Layout the PCB design	Everyone
	Develop and test the embedded software for the system using ordered parts	Sumedh
	Begin research of the enclosure for the system	Eshrit
March 6th - March 13th	Order the PCB from a manufacturer	Akash
	Finalize the enclosure design and begin construction	Eshrit
	Test and debug the embedded software for the system	Sumedh
March 13th - March 20th	Receive the PCB from the manufacturer	Everyone
	Begin soldering and assembling the components onto the PCB	Eshrit
	Continue construction of the enclosure for the system	Eshrit
March 20th - March 27th	Complete the PCB assembly and testing	Everyone
	Begin integrating the PCB and enclosure	Akash
	Begin testing the system as a whole	Everyone
March 27th - April 3rd	Continue testing and debugging the system	Sumedh
	Finalize the user interface design and implementation	Sumedh
	Develop any necessary documentation for the project	Everyone
April 3rd - April 10th	Complete testing and debugging of the system	Sumedh
	Finalize the documentation for the project	Eshrit

	Prepare for any necessary demos or presentations	Everyone
April 10th - April 17th	Conduct final testing and quality control checks	Everyone
	Address any final issues or bugs that arise	Everyone
	Prepare the system for shipping and/or deployment	Everyone
April 17th - May 2nd	Conduct any necessary follow-up work or troubleshooting	Everyone
	Complete any necessary paperwork or administrative tasks	Everyone
	Wrap up the project and submit for grading or review	Everyone
	Final checks before submission. Time to account for unexpected complications	Everyone

4. Ethics and Safety

The ethics and safety of our system, TipsyTracker, are of utmost importance. There are potential ethical and privacy concerns surrounding the collection and use of data. To ensure that privacy is maintained, we will temporarily collect and store the minimum necessary data and promptly remove it after every party. Furthermore, notifications will only be sent to guests and the host in accordance with the set intervals and BAC thresholds, and will not be shared with third parties or easily exported by the host. To ensure transparency, we will have clear and open policies regarding data collection and storage, providing users access to all their data. This aligns with the IEEE Code of Ethics, which states: "to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to promptly disclose factors that might endanger the public or the environment" (IEEE, 1).

It is our duty to ensure that our users are fully informed and kept up-to-date in the event of any outages. In situations where individuals may have relied on our system to monitor and regulate their alcohol intake, it is of utmost importance that we promptly communicate any system failures. This not only helps to protect our users' health and well-being, but it also aligns with our moral obligations to provide accurate and timely information. In the event of a system shutdown, it is our responsibility to immediately notify users and advise them to closely monitor their behavior until the system is fully restored.

To combat misuse of TipsyTracker, we will make a clear emphasis on the system's proper utilization. The ACM Code of Ethics and Professional Conduct highlights the importance of responsible behavior, stating: "Maintain high standards of professional competence, conduct, and ethical practice" (ACM, 1). The instructions for using TipsyTracker will emphasize its intended purpose of promoting responsible drinking and avoiding any malicious use, such as deliberately not informing guests of their BAC levels, which could put them in danger by misleading them into thinking they are drinking responsibly.

Overall, TipsyTracker is a highly safe product. TipsyTracker will not include batteries, high-powered components or circuits, or any dangerous chemicals apart from the alcohol from users' breath. The only close human interface with the device will be the users blowing into the breathalyzer.

Apart from the following, there are no physical safety concerns and no other risk factors to users or developers that require special consideration:

1. Liquid damage occurring from drinks being spilled on the device
2. Users consuming excessive alcohol in the event of an outage

The potential for electrical equipment being near liquids is a concern. To mitigate this risk, we will enclose our system in a casing, and also protect the power source. The

RFID, Breathalyzer, and Power Control Subsystems are the only ones that have a high risk of being exposed to liquid. The data management subsystem does not require a special enclosing as it will be kept at the registration station, far away from the other three subsystems.

In the event of a system outage, if people are relying on TipsyTracker to monitor their consumption, they may dangerously over consume. To prevent this, the moment an outage is detected on the TipsyTracker device, a notification will be sent out to all registered users to take extra caution and monitor their drinking.

When developing our system, we will make sure to adhere to safety standards and regulations. By prioritizing ethics and safety, we aim to promote responsible drinking and reduce harm caused by excessive alcohol consumption, while upholding the standards set by the IEEE and ACM Codes of Ethics.