GYMHIVE TRACKER

By

Aryan Shah

(aryans5@illinois.edu)

Kushal Chava

(kchav5@illinois.edu)

Final Report for ECE 445, Senior Design, Spring 2025

TA: Aishee Mondal

07 May 2025

Project No. 28

Abstract

The GymHive Tracker is a pressure-sensor and RFID-enabled PCB that mounts to individual gym machines to automatically log occupancy and coordinate user queues. A GHF-10 force sensor linked to an ESP32 microcontroller detects loads above 50 lb, identifying equipment use with 95 % accuracy. We relay RFID scans processed over SPI via Wi-Fi/MQTT to AWS IoT Core, where a cloud service updates machine status, stores user check-ins, and notifies the next user. End-to-end latency from load change to app display is under 4 seconds; RFID identification completes within 1 second; and predicted wait times stay within 15 % of actual session lengths. The fully soldered prototype operated continuously for 12 hours without fault, demonstrating a low-cost, scalable solution for real-time gym-equipment management.

Contents

1. Introduction

Commercial gyms frequently face the problem of unpredictable equipment availability, leading to long waiting times and disrupted workout routines. Gym members often follow structured exercise plans that depend on a specified order of machines, and unexpected delays reduce workout efficiency and member satisfaction. To address this, we developed the GymHive Tracker, an end-to-end system that monitors machine occupancy in real time and manages a digital queue via a smartphone app. By embedding a GHF-10 pressure sensor under machine contact points and integrating an MFRC522 RFID reader with an ESP32 microcontroller, our device detects when a machine is in use, associates each session with a user's gym key fob, and provides estimated wait times to those waiting in line.

1.1 Purpose & Problem Statement

Problem: During peak gym hours, equipment availability is both limited and unpredictable, leading to:

- **Long wait times** that interrupt members' workout flow and force them to wait rather than train.
- **Routine changes** or skipped exercises as users scramble for open machines, reducing overall workout effectiveness and satisfaction.

Solution Overview: We built GymHive Tracker, an integrated IoT-and-cloud system that improves how members reserve and use equipment:

- **Real-time occupancy detection:** We embed pressure-sensor PCBs under machine pads to immediately detect when equipment is in use.
- **RFID check-in & adaptive queueing:** Members tap their gym fob to join a virtual queue, enter planned sets/reps, and receive personalized wait-time estimates based on historical usage data.
- **Cloud-enabled web app notifications:** A web interface shows live machine status and pushes turn-ready alerts, so members can optimize their workout and never miss their exercise.

High-Level Functionality: The GymHive Tracker aims to eliminate uncertainty through its high-level requirements:

1. **Real-Time Equipment Monitoring:** The system must accurately detect equipment occupancy with at least 95 % accuracy (± 5 lb threshold for acceptance), filtering out random weight fluctuations.
2. **Efficient Data Transmission and Display:** The user must be able to transmit PII (personally identifiable information) via RFID within 3 seconds of scanning a key fob. The microcontroller must process occupancy data and user check-ins within 3 seconds and transmit updated availability to the central AWS server within 1 minute of an occupancy change.
3. **Queue Management and User Notifications:** The system must estimate wait times with a maximum error margin of 20 % ± 3 % by analyzing user-entered reps and sets (high error margin due to tendency of set and rep times to vary across users).

2 Design

2.1 Visual Aid

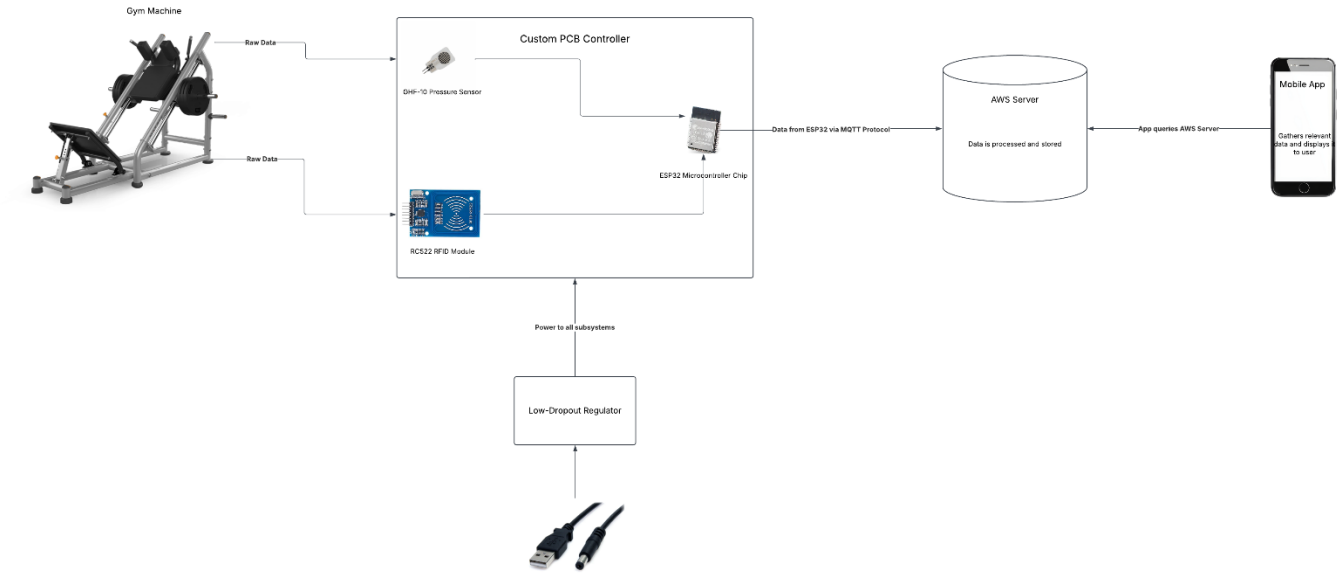


Figure 1 – High-Level Overview of GymHive Tracker

The diagram shown above details our end-to-end system for real-time gym equipment tracking:

1. **Gym Machine → Sensors**
    a. We embed a GHF-10 pressure sensor in key contact pads to detect usage.
    b. We mount an MFRC522 RFID module on the machine's side for easy gym-fob access.
2. **Sensors → Custom PCB Controller**
    a. We route raw pressure readings and RFID read/write data to our PCB.
    b. An ESP32 on the PCB samples the sensor via ADC and communicates with the RFID module over SPI.
3. **PCB Controller → AWS Server**
    a. ESP32 publishes each occupancy and user-ID event over MQTT via on-board Wi-Fi.
    b. AWS IoT Core receives, processes, and stores the incoming data streams.
4. **AWS Server → Web App**
    a. Our HTTP-based web application queries the AWS backend for current machine status and queue information.
    b. We display live availability and estimated wait times to gym members.
5. **Power Supply**
    a. A 5 V USB input feeds a low-dropout regulator (LDO).
    b. The LDO steps down to a stable 3.3 V rail powering the ESP32, pressure sensor, and RFID module.

2.2 Block Diagram

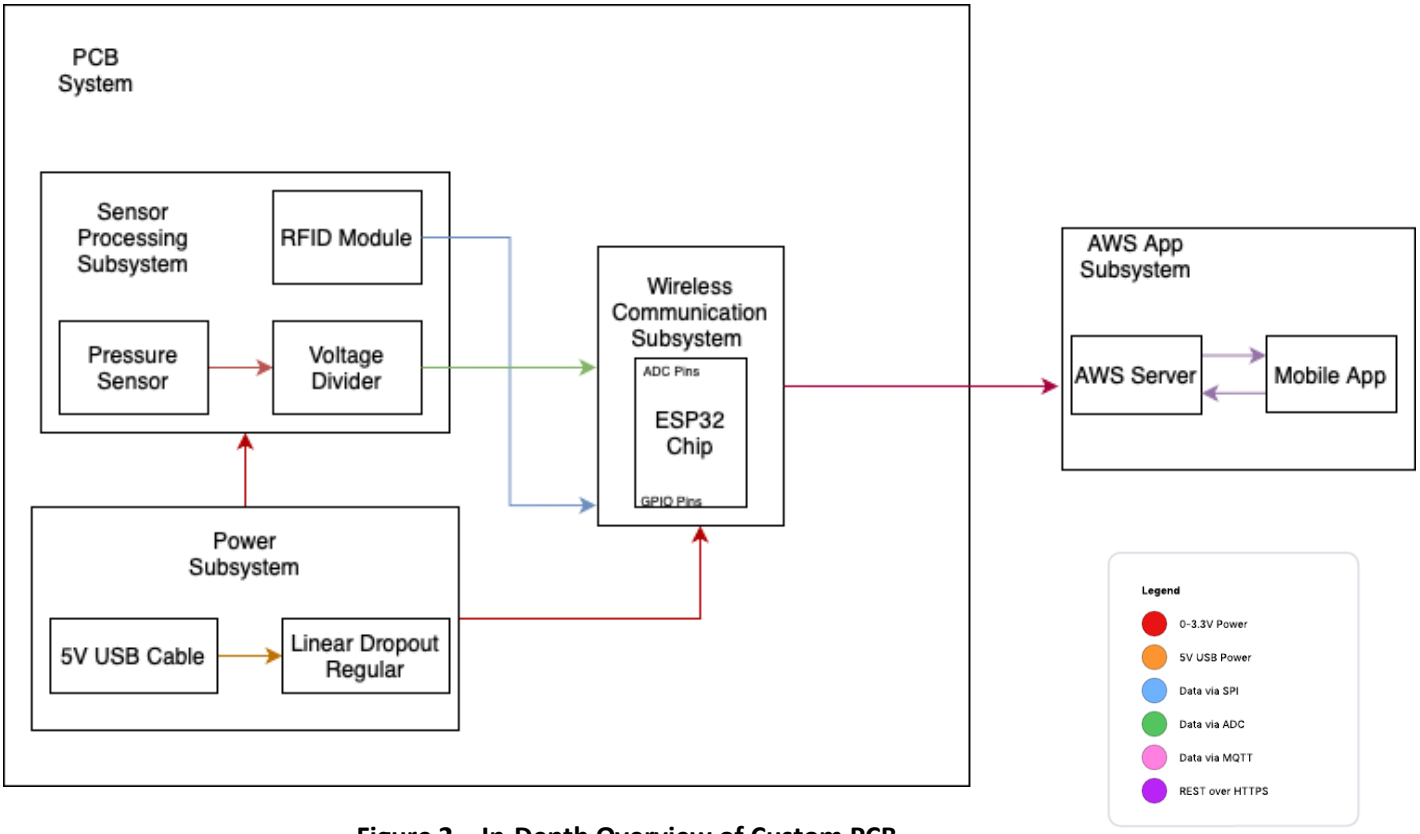


Figure 2 – In-Depth Overview of Custom PCB

Below is an expanded walkthrough of our block diagram, tracing every signal path, interface and subsystem responsibility:

1. **Power Subsystem**
    a. 5V USB Cable
        i. Supplies raw power from any standard USB port plugged into a wall adapter.
    b. Linear Dropout Regulator (LDO)
        i. Steps 5 V down to a clean 3.3 V rail with < 50 mV dropout.
        ii. Feeds the pressure sensor, voltage divider, RFID module, and all ESP32 power pins.
2. **Sensor Processing Subsystem**
    a. **Pressure Sensor Chain**
        i. GHF-10 Pressure Sensor
            1. Mounted under the machine's contact pad to detect load changes.
        ii. Voltage Divider
            1. Scales the sensor's output to match necessary force readings into the ESP32's 0 – 3.3 V ADC input range.
        iii. ADC Line

1. One of the ESP32's ADC pins continuously collects the conditioned pressure voltage.

b. **RFID Reader Interface**
   i. MFRC522 RFID Module
      1. Reads/writes 13.56 MHz gym-fob tags.
   ii. SPI Bus
      1. SCK, MOSI, MISO, SS, and RST lines wired to the ESP32's SPI-capable GPIOs.
      2. Firmware polls the module at ~50 Hz and debounces tag reads in software.

3. **Wireless Communication Subsystem**
   a. ESP32 Microcontroller
      i. Sensor Fusion: Timestamp and package each pressure reading + tag ID.
      ii. Logic & Queue Management: Decide "occupied" vs. "unoccupied" events, track per-user session durations.
      iii. MQTT Client: Publishes occupancy events and user-ID messages over Wi-Fi to AWS IoT Core.
      iv. HTTPS Client: Can poll configuration endpoints or report firmware health to AWS API Gateway.
4. **AWS App Subsystem**
   a. **AWS IoT Core**
      i. Receives MQTT topics, enforces TLS, and triggers downstream Lambdas.
   b. **AWS Lambda & DynamoDB**
      i. Transforms incoming messages into machine-state records and queue entries.
   c. **API Gateway & Mobile/Web App**
      i. Exposes endpoints for status, historical logs, and wait-time estimates.
      ii. The HTTP-based web app queries these endpoints to render live machine availability and manage check-in/queue operations.

2.3 Physical Design

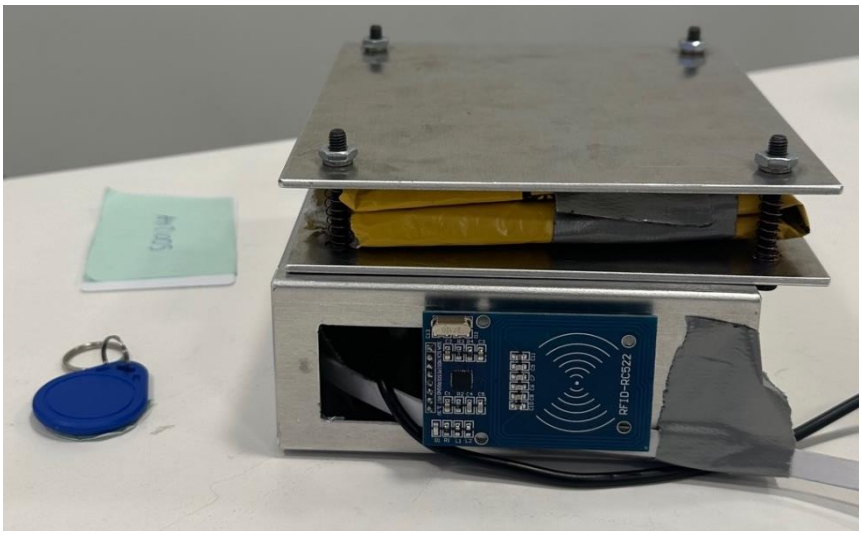


Figure 3 – Physical Design of GymHive Tracker

The figure above shows our physical prototype for the GymHive Tracker:

1. **Top Assembly (Load Sensing)**
    a. Two aluminum plates act as the "pad" surfaces. Between them sits the GHF-10 pressure sensor, surrounded by a foam layer to mimic the feel of a gym pad. Four compression springs ensure even force distribution and return the top plate to its unloaded position.
2. **RFID Module (User Check-In)**
    a. We mount the MFRC522 board on the side of the enclosure to give users easy access for tapping their gym key fob.
3. **Gym Key Fob Example**
    a. Shown at left are typical 13.56 MHz RFID tags (key-fob and card) that members scan against the reader to register themselves on the machine's digital queue.

2.4 Subsystem Overview

The system is organized into six subsystems. The Pressure Sensing Subsystem uses a GHF-10 piezoresistive force sensor whose variable resistance is converted to a conditioned voltage by a voltage divider and then processed by the ESP32's 12-bit ADC. The Microcontroller Subsystem centers on an ESP32-WROOM-2D module, which samples both sensor and RFID data and facilitates any other PCB communication. The RFID Read/Write Subsystem employs an MFRC522 with a 27.12 MHz crystal oscillator and matching network to read and write gym-fob tags over SPI. The Power Subsystem – built around a USB-C connector and an AMS1117-3.3 linear regulator – delivers a stable 3.3 V rail to every component. The USB-to-UART Programming Subsystem uses a CP2102 bridge to enable firmware flashing and serial debugging. Finally, the Software Subsystem implements ESP32 firmware that publishes occupancy and check-in events via MQTT to AWS IoT Core, exposes HTTP endpoints through API Gateway and Lambda, and drives a web-app frontend for real-time status and wait-time estimates. Our final custom PCB layout is provided in Appendix B.

2.4.1 Pressure Sensing Subsystem

2.4.1.1 Function & Key Component(s)

Primary Function: Detect when a user is occupying a machine pad ($\geq 50\ lb$) with $\geq 95\ \%$ accuracy and zero false negatives.

Key Component: GHF-10 flexible piezoresistive polymer (0 – 500N / 0 – 110 lb) [1]

- We mount the GHF-10 pressure sensor via a dedicated PCB connector for easy assembly.
- Interfaced in a simple voltage-divider: GHF-10 as Rs and a fixed 100 kΩ resistor (R1) to ground.
- Output V2 feeds the ESP32's 12-bit ADC (0 – 4095 translates to 0 – 3.3 V).
- Threshold: ADC count ≥ 1500 (~2.5 V) flags "in use."
- Resistant to random weight fluctuations (water bottle, wallet, phone, keys, etc.).

2.4.1.2 Equations & Simulations

Voltage Divider

2. Using a fixed resistor R1 in a voltage divider configuration for an output V2 that increases with respect to added force.

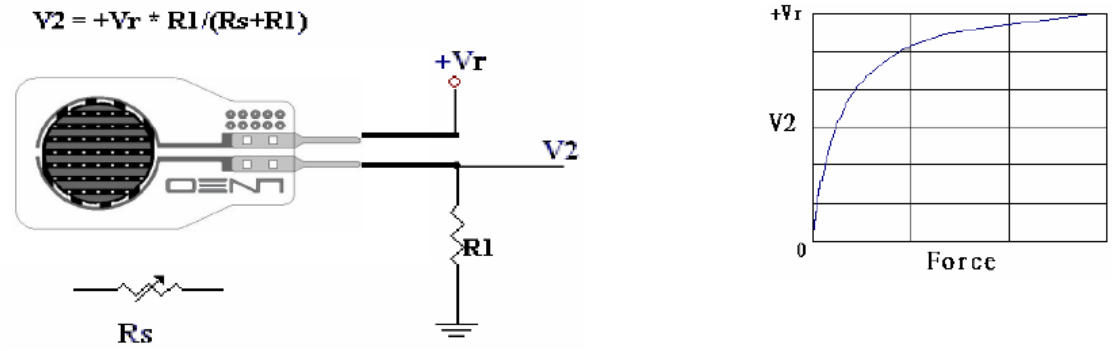


Figure 4 – Datasheet Schematic of GHF-10's Voltage Divider Configuration

According to the datasheet [1] of the GHF-10, the output V2 increases non-linearly with respect to added force and a chosen resistor value, R1.

- **Divider Equation:** $\boldsymbol{V_2(F) = V_{ref} * \frac{R_1}{R_1 + R_s(F)}}$, where $\boldsymbol{V_{ref} = 3.3\ V}$, $\boldsymbol{R_1 = 100\ k\Omega}$, and $\boldsymbol{R_s(F)}$ is the sensor resistance at force F.
- **Calibration Curve:** We digitized the nonlinear $\boldsymbol{R_s(F)}$ data from the GHF-10 datasheet and simulated $\boldsymbol{V_2(F)}$ across increments of 0 – 100 lb.
- **Simulation Results:**
  - At 50 lb: $\boldsymbol{V_2} \approx 2.5V$ → ADC ≈ 1500 counts.
  - At 0 lb: $\boldsymbol{V_2} \approx 0V$ → ADC ≈ 0 counts.
  - Curve steepness yields > 10 counts/lb sensitivity around our threshold region.
- **Interpretation:** These simulations guided our choice of R1 to maximize dynamic range in the 50 – 80 lb zone where most users apply force.

ADC Conversion

Due to the ESP32's 12-bit resolution on its ADC pins [2], we must convert raw values in the [0,4095] range using the following code in our data processing:

$$ADC_{counts} = 4095 * \frac{V_{ADC}}{3.3\ V}$$

2.4.1.3 Design Alternatives

Op-Amp Linearization: Rather than a simple voltage divider, place the GHF-10 in an op-amp amplifier circuit [1]. This yields a more linear voltage-to-force relationship and could perhaps boost the signal around our 50 lb detection point, improving ADC resolution. The tradeoff is adding one small amplifier IC and a couple of resistors, slightly increasing cost and board area.

2.4.1.4 Schematic

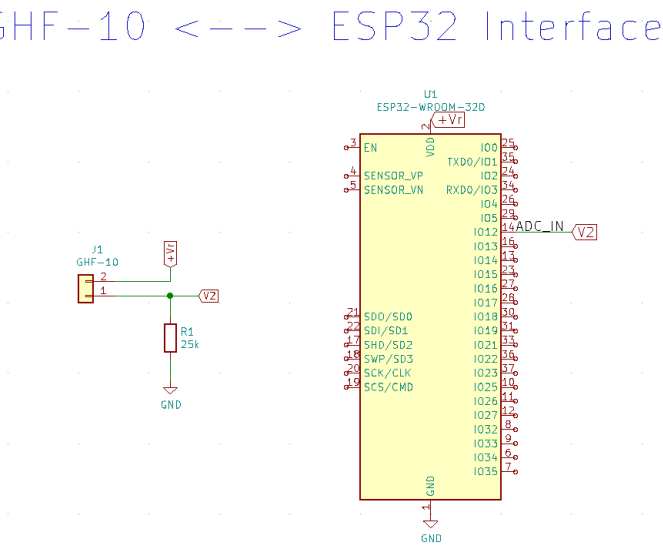


Figure 5 – Designed Schematic of Pressure Sensing Subsystem

2.4.2 Microcontroller Subsystem

2.4.2.1 Function & Key Component(s)

Primary Function: Facilitate sensor reads, RFID check-ins, queue logic, and cloud communication.

Key Component: ESP32-WROOM-32D Module [2]

- Built-in Wi-Fi & Bluetooth for MQTT connectivity to AWS IoT Core.
- 12-bit ADC for reading the pressure-sensor voltage divider.
- SPI Interface driving the MFRC522 RFID reader.
- Sufficient GPIO pins for push-button, status LEDs, and power control.

2.4.2.2 Design Alternatives

Espressif labeled the ESP32-WROOM-32D Module as “Not Recommended for New Designs” (NRND). Its successor, the ESP32-WROOM-32E uses an upgraded Espressif’s Eco V3 silicon design, while retaining similar functionality and footprint design. This allows for straight pin-for-pin compatibility, full Wi-Fi + Bluetooth Classic support, while enabling an upgraded design that retains our existing PCB layout yet leverages the latest silicon improvements – delivering better power efficiency, higher throughput, and enhanced security features.

2.4.2.3 Schematic

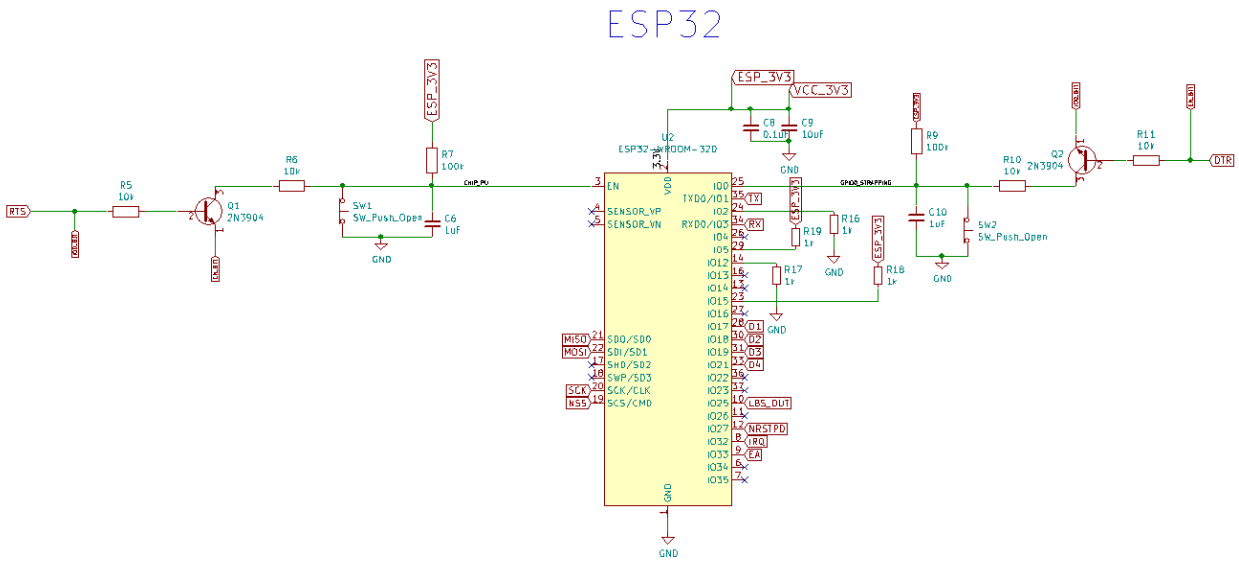


Figure 6 – Designed Schematic of Microcontroller Subsystem [3]

2.4.3 RFID Read/Write Subsystem

2.4.3.1 Function & Key Component(s)

Primary Function: Wirelessly read a member’s gym-fob tag at 13.56 MHz and deliver a unique ID to the microcontroller in under 100 ms.

Key Component: MFRC522 RFID Read/Write IC [4]

- On-chip HF front-end compliant with ISO/IEC 14443-A, handling both carrier generation and demodulation.

- Requires a 27.12 MHz crystal oscillator and a small LC matching network (coil + capacitors) tuned for resonance.
- Connects to the ESP32 via SPI (SCK, MOSI, MISO, NSS) plus IRQ and RESET lines [5].

2.4.3.2 Equations & Simulations

- **Resonant Frequency:** $f_{res} = \frac{1}{2\pi\sqrt{LC_{eq}}}$, where L is the antenna coil inductance and C_{eq} is the total series capacitance.
- **Design Targets:** L ≈ 700 nH, C_{eq} ≈ 200 pF → f_{res} ≈ 13.56 MHz.
- **Simulation:** Impedance sweep analysis confirmed a < -10 dB return loss at 13.56 MHz, ensuring optimal energy transfer at a 50 mm read range.

2.4.3.3 Design Alternatives

- **Multi-Protocol RFID Front-End**
  - A single IC that supports both low-frequency (125 kHz) and high-frequency (13.56 MHz) tags.
  - Allows for a universal reader – one chip that can handle any gym-fob standard.
  - Example ICs include ST25R3916 or NXP PN 5180.
- **Smartphone-based NFC/BT pairing**
  - Replace hardware reader with a companion mobile app that taps built-in NFC or Bluetooth-LE iBeacon tags on enabled devices.
  - This design was initially shut down due to concerns regarding gaining access to necessary data via popular smartphone models.

2.4.3.4 Schematic

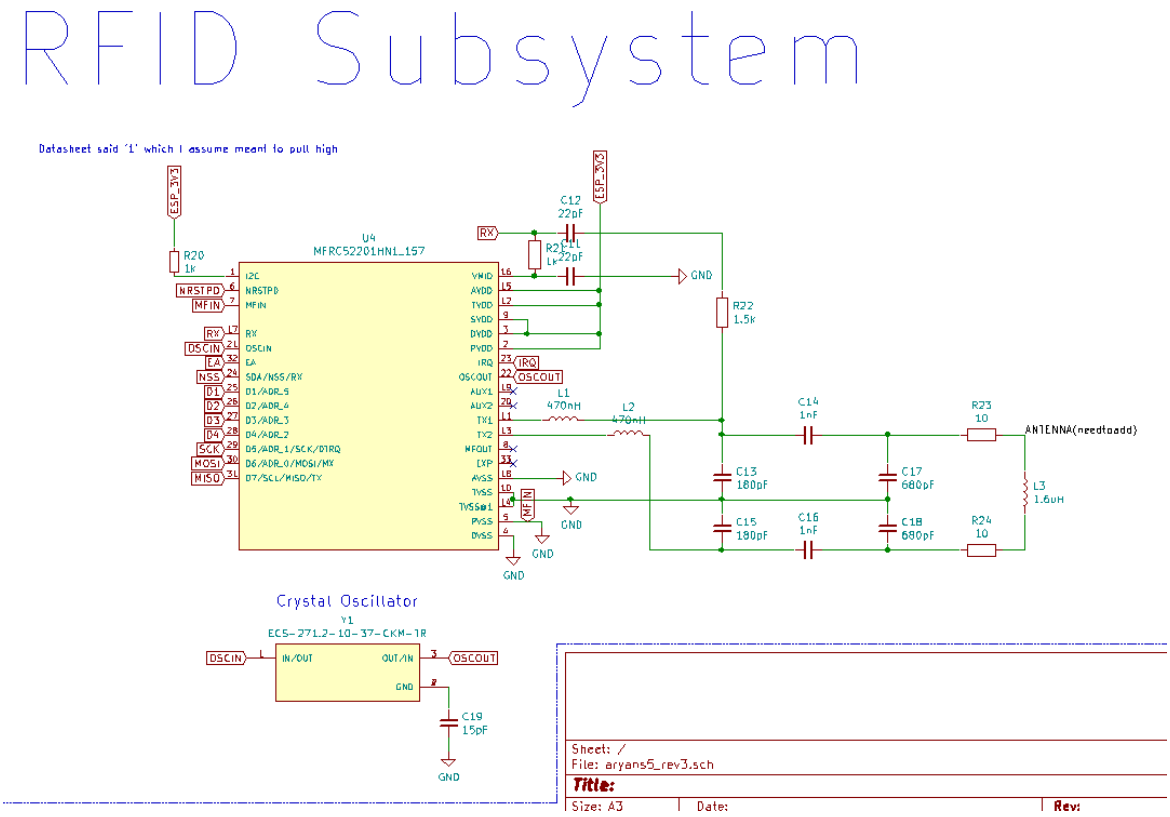


Figure 7 – Designed Schematic of RFID Read/Write Subsystem

2.4.4 Power Subsystem

2.4.4.1 Function & Key Component(s)

Primary Function: Deliver a stable 3.3 V rail (± 0.1 V) capable of up to 500 mA to power the ESP32, MFRC522, and sensors from a readily-available 5 V source [6].

Key Components: TYPE-C-31-M-12 USB-C Connector [14] & AMS1117-3.3 LDO [15]

- USB-C connector provides 5 V inlet (up to 3 A capability).
- Linear regulator steps 5 V → 3.3 V.
- Input/output decoupling capacitors.
- Reverse-current protection diode on the 5 V line.

2.4.4.2 Equations & Simulations

- **Power dissipation in LDO:** $\boldsymbol{P_{loss} = (V_{in} - V_{out}) * I_{out} = (5.0V - 3.3V) * 0.5A = 0.85W}$
- **Thermal considerations:** With a thermal resistance of ~50 °C/W, a 0.85W dissipation yields a temperature change of ~42 °C – acceptable with our small heatsink area and airflow near gym equipment.

- **Dropout margin:** AMS1117 requires ~1.1 V dropout; $V_{in} - V_{out}$ = 1.7 V leaves a 0.6 V margin, ensuring regulation even if USB-C drops to 4.7 V effective input.

2.4.4.3 Design Alternatives

- **Larger-Footprint USB-C Connector**
  - Swap the compact TYPE-C-31-M-12 for a robust, panel-mound or board-lock USB-C jack with enlarged pads and mechanical posts.
  - The existing footprint was delicate and fragile, breaking off easily during integration and testing. This required us to swap to a 9 V battery as a last-ditch effort, not ideal as it provides a much higher peak current draw and is incompatible with our linear regulator.
- **Switch-Mode Buck Converter**
  - Switching to a switch-mode buck converter for our step-down logic would increase our input-voltage range drastically. It would allow for higher current capability, and an increased efficiency as opposed to a linear LDO design.

2.4.4.4 Schematic

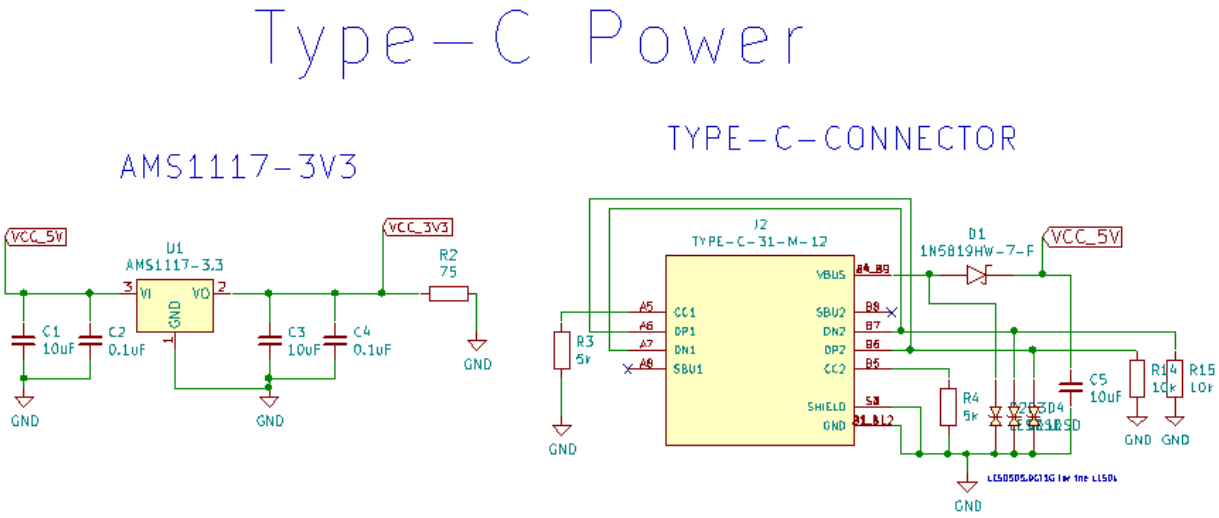


Figure 8 – Designed Schematic of Power Subsystem [6]

2.4.5 USB-to-UART Programming Subsystem

2.4.5.1 Function & Key Component(s)

Primary Function: Provide a reliable PC ↔ ESP32 interface for flashing firmware and real-time serial debugging.

Key Component: SiLabs CP2102 USB-to-TTL bridge [7]

- Enumerates as a virtual COM port over USB-2.0.
- Outputs 3.3 V-level TX/RX signals compatible with the ESP32's UART.

- Exposes RTS and DTR lines used to automatically toggle the ESP32's EN and IO0 pins for easy programming [3].
- Auto-reset sequence:
    - DTR → IO0 pulled low during rest to enter bootloader mode.
    - RTS → EN toggled to reset the chip, initiating the flash sequence without manual button presses.

2.4.5.2 Design Alternatives

- **On-Board CP2102 Integration**
    - Instead of using an external USB-to-TTL dongle, we can place the CP2102 (and its supporting passive components) directly on our PCB. This eliminates a loose adapter, reduces cable clutter, and streamlines assembly.

2.4.5.3 Schematic

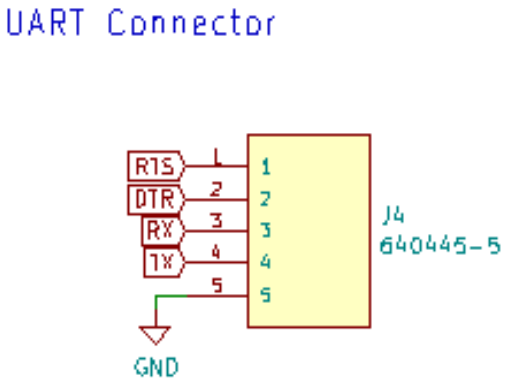


Figure 9 – Designed Schematic of Microcontroller Subsystem [3] (see Figure 6 for expansion)

2.4.6 Software Subsystem

2.4.6.1 Function & Architecture

Primary Function: Connect sensor/RFID inputs, queue logic, and user interaction via a cloud-backend web interface.

High-Level Modules:

1. **IoT data ingestion** (ESP32 → AWS IoT Core → Lambda → DynamoDB).
2. **API Layer** (API Gateway → Lambda) outputting live machine + queue state.
3. **Frontend Client** (HTML/JS) for Home, Queue, and Account views.
4. **Local Fallback Logic** (localStorage) for queue persistence and push-notification preferences.

2.4.6.2 AWS Backend Integration Flow [8]

1. **DynamoDB Table: Machine State**
    a. Stores each machine's latest occupancy flag, current RFID binding, and timestamp.
2. **Lambda Functions**
    a. **RouteRFIDtoDynamo**
        i. Trigger: AWS IoT rule on ESP32 publish
        ii. Action: PutItem to MachineState
        iii. Role: RouteRFIDtoDynamo-role (DynamoDB:PutItem)
    b. **GetLatestRFIDState**
        i. Trigger: API Gateway GET /latest
        ii. Action: GetItem from MachineState
        iii. Role: GetLatestRFIDState-role (DynamoDB:GetItem)
3. **API Gateway (HTTP API)**
    a. Route: GET /latest → GetLatestRFIDState
    b. Deploys to $default stage
4. **Frontend Integration**
    a. 2 separate API gateways for pressure sensor and RFID module
    b. Polls every 500 ms or on state change to update machine cards and queue table

2.4.6.3 Features

- **Secure ESP32 firmware**
    - Connects over WPA2-Enterprise to campus Wi-Fi
    - Establishes mutual-TLS with AWS IoT Core using a Root CA, device certificate, and private key.
    - Reads the GHF-10 pressure sensor (ADC) and MFRC522 RFID (SPI), then publishes JSON messages via MQTT.
- **Web frontend (HTML & JavaScript)**
    - Home Page
        - Live machine cards showing "Occupied" or "Unoccupied" based on pressure sensor outputs.
        - Current User as well as checked-in users in the queue.
    - Account Page
        - Login/bind flow for unregistered tags
        - Secure login
        - Input for sets/reps with suggested defaults based on user history
    - Queue Page
        - Real-time table of queued users
        - Own-duration and wait-until-start estimates
        - Auto-kick on inactivity
        - Browser push-notification control

2.4.6.4 Frontend UI

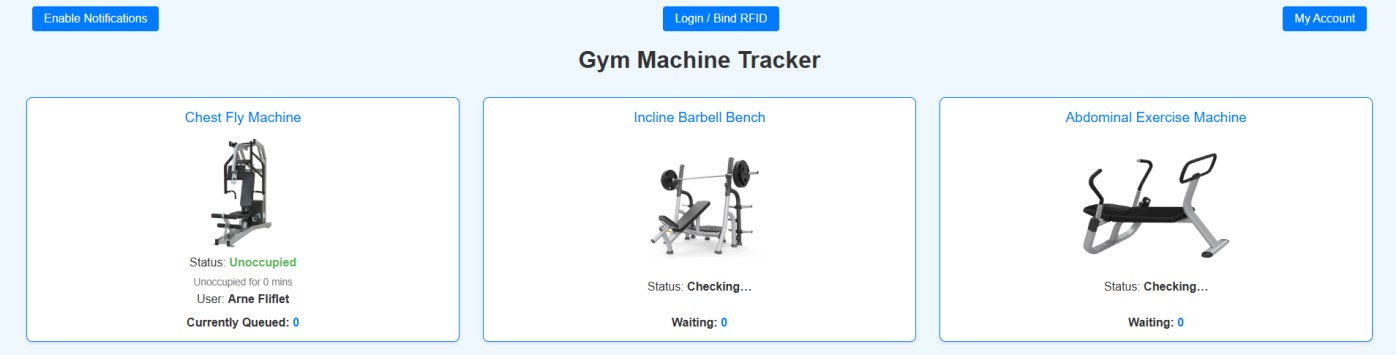
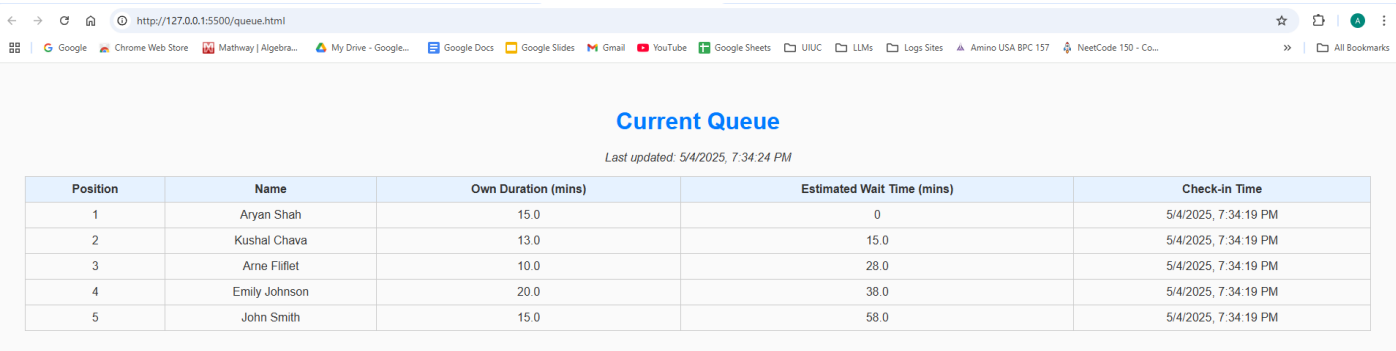
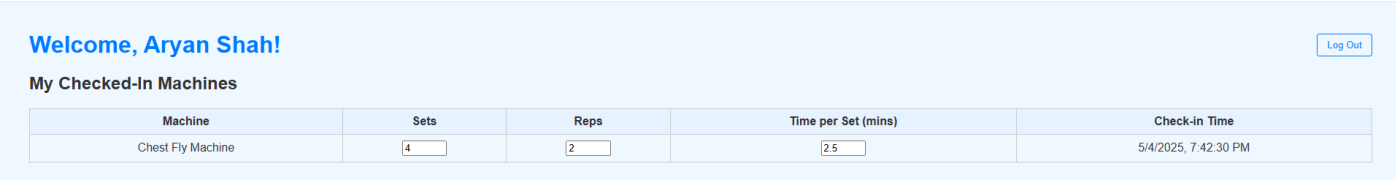


Figure 10 – GymHive Tracker Home Page



Position	Name	Own Duration (mins)	Estimated Wait Time (mins)	Check-in Time
1	Aryan Shah	15.0	0	5/4/2025, 7:34:19 PM
2	Kushal Chava	13.0	15.0	5/4/2025, 7:34:19 PM
3	Arne Fliflet	10.0	28.0	5/4/2025, 7:34:19 PM
4	Emily Johnson	20.0	38.0	5/4/2025, 7:34:19 PM
5	John Smith	15.0	58.0	5/4/2025, 7:34:19 PM

Figure 11 – GymHive Tracker Queue Page



Machine	Sets	Reps	Time per Set (mins)	Check-in Time
Chest Fly Machine	4	2	2.5	5/4/2025, 7:42:30 PM

Figure 12 – GymHive Tracker Account Page

3. Design Verification

We outlined requirements and verification procedures across each subsystem to not only confirm the functionality of every peripheral but also ensure overall reliability and user safety. An expanded table of all requirements and verifications can be found in Appendix A. Below we detail those verifications that were not met, organized by subsystem, along with the root causes.

3.1 Pressure Sensing Subsystem

- **Requirement:** ≥ 95 % detection accuracy with 0 % false negatives at ≥ 50 lb.
- **Measured:** We detected 19 out of 20 weight placements ≥ 50 lb, yielding a 5 % false-negative rate.
- **Root Cause:** ADC threshold set at 1500 counts sits too close to the sensor's noise floor; occasional dips below threshold occur under borderline loads.

3.2 Microcontroller Subsystem

- **All checks passed:** ADC sampling rate, SPI timing for RFID, Wi-Fi latency (<300 ms), and power-mode transitions met their specifications.

3.3 RFID Read/Write Subsystem

- **All checks passed:** 13.56 MHz tag reads at 50 mm in < 50 ms, and SPI transactions completed reliably.

3.4 Power Subsystem

- **Requirement (Implicit):** Maintain mechanical connection under repeated use.
- **Measured:** We observed that the connector detached after a few handling trials.
- **Root Cause:** The small-footprint TYPE-C-31-M-12 lacks board-lock posts; mechanical stress concentrates on solder joints.

3.5 USB-to-UART Programming Subsystem

- **All checks passed:** CP2102 auto-reset (RTS/DTR toggling) worked in 100 % of trials when correctly receiving power; 115200 baud links sustained error-free.

3.6 Software Subsystem

- **Requirement:** Prediction error ≤ 20 % ± 3 % in all cases.
- **Measured:** Historical-average fallback produced 23 % average error, with only 4 out of 5 trials inside margin.
- **Root Cause:** Simple rolling-average model fails to capture per-user variability under low-data conditions.

4. Costs

In this chapter, we present a comprehensive overview of all expenditures incurred in the development of the GymHive Tracker, encompassing both hardware components and the labor required to design, assemble, and test the system. Section 1 breaks down each component by manufacturer, retail price, bulk purchase price, and the actual unit cost realized in our build. Section 2 then summarizes the person-hours and associated rates for activities such as system design, firmware/software development, hardware integration, and validation testing. Based on our estimates, total parts costs totaled to $82.28, total labor costs amounted to $39,000, bringing our final cost to $39,082.28.

4.1 Parts

Note: Various resistors, capacitors, and other passive components obtained at no cost from the EShop have been omitted. We calculated bulk purchase costs using unit prices from the Manufacturer's Standard Package.

Table 1 – Parts Costs

Part	Description	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
0022232021	CONN HEADER VERT 2POS 2.54MM	Molex	0.16	0.09	0.32
AMS1117-3.3	IC REG LINEAR 3.3V 1A SOT-223-3L	EVVO	0.27	0.11	1.35
1N5819HW-7-F	DIODE SCHOTTKY 40V 1A SOD123	Diodes Incorporated	0.25	0.06	1.25
LESD5D5.0CT1G	9.4A 18.6V 5.6V Bidirectional 5V SOD-523 ESD and Surge Protection (TVS/ESD) ROHS	LRC	0.01	0.01	0.01
TYPE-C-31-M-12	5A 1 16P Female Type-C SMD USB Connectors ROHS	Korean Hroparts Elec.	0.17	0.11	0.86
TS04-66-70-BK-260-SMT	SWITCH TACTILE SPST-NO 0.05A 12V	Same Sky	0.18	0.11	0.90
640445-5	CONN HEADER VERT 5POS 3.96MM	TE Connectivity AMP Connectors	0.41	0.24	0.82

410-212	PMODUSBUART USB TO UART MODULE	Digilent, Inc.	14.99	14.99	14.99
MFRC52201HN1,157	IC RFID READER 13.56MHZ 32HVQFN	NXP USA, Inc.	8.00	5.24	16.00
ECS-271.2-10-37-CKM-TR	CRYSTAL 27.1200MHZ 10PF SMD	ECS Inc.	0.47	0.30	1.88
AL-77P-01	AL-77P Diecast Aluminum Enclosure	Polycase	19.04	14.56	19.04
ESP32-WROOM-32D-N4	RF TXRX MOD BT WIFI TH SMD	Espressif Systems	3.80	3.80	19.00
GHF-10	10mm Dia Touch / Pressure Sensor	UNEO Inc.	5.86	4.44	5.86
Total			**53.61**	**44.06**	**82.28**

4.2 Labor

To provide a realistic estimate, we use the reported average starting salary for Computer Engineering graduates from ECE Illinois AY 21-22 [9]: $109,176 per year, which corresponds to approximately $52 per hour. We then applied an overhead multiplier of 2.5 to account for benefits, equipment, and indirect costs. Assuming each team member contributes roughly 150 hours of work, the calculation is as follows:

$$\$52/hr * 2.5 * 150hr = \$19{,}500\ per\ member$$

For our two-member team, this yields a total labor cost of:

$$\$19{,}500 * 2 = \$39{,}000$$

5. Schedule

The table below presents a rough, week-by-week timeline of the semester's work, detailing each team member's most important assigned tasks and milestones to clearly reflect how the project progressed and who was responsible for each activity.

Table 2 – Weekly Schedule

Week of	Task(s)	Team Member(s)
Jan 27, 2025	RFA Work	Aryan and Kushal
Feb 10, 2025	1st TA Meeting Project Proposal Visual Aid Block Diagram	Aryan and Kushal Aryan and Kushal Aryan Aryan

	Tolerance Analysis	Kushal
Feb 17, 2025	Incorporate proposal review feedback	Aryan and Kushal
Feb 24, 2025	1st round PCB Design 2nd TA Meeting PCB Review	Schematic (Aryan) Aryan and Kushal Aryan and Kushal
Mar 3, 2025	PCB Design Design Document Physical Design Sketch Design Expansion (Design Review) 3rd TA Meeting	Aryan (Schematic) Aryan and Kushal Aryan Aryan and Kushal Aryan and Kushal
Mar 10, 2025	Breadboard demonstration build and software	Wiring (Kushal), Software (Aryan)
Mar 17, 2025	PCB Round 2 layout Component Ordering	Schematic (Aryan), Board (Kushal) Aryan and Kushal
Mar 24, 2025	TA Meeting PCB #2 Assembly	Aryan and Kushal Soldering (Kushal) with Aryan's help
Mar 31, 2025	PCB Round #3 Schematic Work Board Revision	Schematic (Aryan), Board (Aryan and Kushal)
Apr 7, 2025	TA Meeting Individual Progress Report	Aryan and Kushal Aryan and Kushal
Apr 14, 2025	TA Meeting PCB Round #2 Assembly, Testing, and Verification	Aryan and Kushal Assembly (Kushal), R&V (Aryan)
Apr 21, 2025	TA Meeting Further PCB Round 2 Debugging	Aryan and Kushal Aryan and Kushal
Apr 28, 2025	Mock-Demo TA Meeting PCB Round 3 soldering PCB Round 3 Testing & Verification Finalize physical design Final software setup	Aryan and Kushal Kushal Aryan and Kushal Aryan and Kushal Aryan (RFID), Kushal (GHF-10)
May 5, 2025	Final Demo Mock Presentation Prep Final Presentation Prep Final Paper Write-Up	Aryan and Kushal Aryan and Kushal Aryan and Kushal Aryan and Kushal

6. Conclusion

6.1 Accomplishments

- **End-to-End Prototype Delivered**

    - Designed and fabricated a custom ESP32-based PCB integrating a GHF-10 pressure sensor and MFRC522 RFID reader.
    - Packaged all electronics in a compact enclosure with spring-loaded mounts to ensure consistent weight distribution and reliable sensor readings.
    - Developed firmware to read sensor and RFID data, then publish occupancy and check-in events over MQTT to AWS IoT Core.
- **Cloud Backend**
    - Implemented an AWS Lambda pipeline that receives IoT messages, processes queue logic, and writes state to DynamoDB.
    - Exposed HTTP API endpoints via API Gateway for fetching live machine status and queue positions.
- **Interactive Web-App Front End with Real-Time Notifications**
    - Built a dashboard that visualizes each machine's occupancy, displays virtual queues, and lets users enter planned sets/reps.
    - Integrated browser push notifications to alert members the moment their turn arrives – eliminating the need to constantly monitor the screen.
- **Subsystems Validated against High-Level Requirements**
    - **Pressure detection:** GHF-10 reliably flagged "in use" vs. "vacant" under varying loads.
    - **RFID check-in:** Gym fobs and RFID cards were read consistently at normal tap distances.
    - **Adaptive queueing:** The system computed and updated personalized wait-times based on user-entered reps/sets and live occupancy data.

6.2 Uncertainties

Despite meeting our core requirements, several areas exhibited quantitative shortcomings that may impact robustness and user experience:

- **Mechanical Durability of Power Connector**
    - The USB-C footprint on our PCB proved too small: in handling tests, the connector detached during trials, necessitating repeated reflow-soldering and raised concerns about long-term stability.
- **GPIO Availability for RFID Integration**
    - We were unable to solder wires directly on the ESP32 pads off the MFRC522 dev-kit module, increasing our volume enclosure and complicating routing.
- **Pressure-Sensor Input Reliability**
    - On the final PCB, the GHF-10's original ADC line failed to register any readings because that pin was required for Wi-Fi. Only after remapping did we observe 100 % read success, but this initial failure underscores pin-assignment constraints in future revisions.
- **Fallback Wait-Time Estimation Error**
    - When users omit sets/reps, the system defaults to historical averages; this mode yielded an average prediction error of 23 %, with only 4 of 5 trials falling inside our ±3 % threshold (target: 20 % ± 3 %). While our primary mode (with input) achieved only 10 %

error with 5/5 accuracy, the fallback must be refined to avoid occasional out-of-bound estimates.

6.3 Future Work / Alternatives

To extend the GymHive Tracker's functionality and improve its robustness, we recommend the following improvements/alternatives:

- **IMU Integration [10]:** Add an inertial measurement unit (IMU) to each machine to eliminate manual input and enable fully automatic rep counting.
- **Predictive Analytics:** Develop machine-learning models trained on historical session durations to provide personalized wait-time forecasts for users.
- **Iterative Field Testing:** Conduct systematic trials across diverse gym equipment and user populations to build a comprehensive performance database and uncover further edge-case behaviors.
- **Physical Design Miniaturization:** Redesign and condense the hardware enclosure so it can be more seamlessly embedded into existing gym machines without obstructing normal operation.
- **Kalman Filtering [11]:** Implement a real-time Kalman filter on sensor readings to reject outliers and noise, thereby improving pressure sensing measurement accuracy.

6.4 Ethical Considerations

6.4.1 User Data Privacy and Security

According to IEEE Code of Ethics Section 1.1 [12], the intent of engineers should be "to hold paramount the safety, health, and welfare of the public". By collecting and transmitting user data between the ESP32 microcontroller, AWS server, and mobile app, we put personally identifiable information at risk. To combat this, we allowed for a secure account login mechanism on our front-end client. For an added layer of security, any data relayed from the ESP32 to AWS IoT Core uses TLS (Transport Layer Security) to encrypt data using root of trust, device, and private key certificates.

6.4.2 Physical Safety of Equipment and Users

According to the ACM Code 1.2 [13], the goal of an engineer should be to "avoid harm… negative consequences, especially when those consequences are significant and unjust… include unjustified physical or mental injury, unjustified destruction…". Our embedded PCB design within gym equipment can cause unintended hazards. To pose little risk to the user, we ensure that each electrical component is securely enclosed to mitigate exposure to electrical or flammable hazards. In addition, our PCB operates at a low, stable voltage from 0 – 3.3 V which reduces risks of any electric shocks or fire hazards.

References

[1] “GHF10-500N ENG,” UneoTech, [Online]. Available: https://www.uneotech.com/uploads/product_download/tw/GHF10-500N%20ENG.pdf.

[2] Espressif Systems, “ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet,” Espressif, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

[3] Grainger Engineering, Univ. Illinois, “ESP32 Example,” ECE 445 Wiki, [Online]. Available: https://courses.grainger.illinois.edu/ece445/wiki/#/esp32_example/index.

[4] NXP Semiconductors, “MFRC522 Low-Voltage NFC Reader IC Datasheet,” [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[5] ElectronicWings, “RFID-RC522 Interfacing with ESP32,” [Online]. Available: https://www.electronicwings.com/esp32/rfid-rc522-interfacing-with-esp32.

[6] Instructables, “Build Custom ESP32 Boards From Scratch – the Complete Guide,” [Online]. Available: https://www.instructables.com/Build-Custom-ESP32-Boards-From-Scratch-the-Complete/.

[7] Silicon Labs, “CP2102/CP2102N USB-to-UART Bridge Datasheet,” Rev. 1.2, [Online]. Available: https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf.

[8] Amazon Web Services, “Setting Up Amazon DynamoDB,” AWS Developer Guide, [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SettingUp.html.

[9] Univ. Illinois at Urbana-Champaign, “Why ECE? Salary Averages,” [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.

[10] National Science Foundation, “PURL 10189179,” [Online]. Available: https://par.nsf.gov/servlets/purl/10189179.

[11] R. E. Kirtley, “Kalman Filter,” MIT, [Online]. Available: https://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf.

[12] IEEE, “IEEE Policies 7–8: Corporate Governance,” [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

[13] ACM, “ACM Code of Ethics and Professional Conduct,” [Online]. Available: https://www.acm.org/code-of-ethics.

[14] Korean Hroparts Elec, “TYPE-C-31-M-12 USB Type-C Connector Datasheet,” LCSC, [Online]. Available: https://www.lcsc.com/datasheet/lcsc_datasheet_2410010003_Korean-Hroparts-Elec-TYPE-C-31-M-12_C165948.pdf.

[15] Advanced Monolithic, “AMS1117 3.3 V LDO Regulator Datasheet,” [Online]. Available: http://www.advanced-monolithic.com/pdf/ds1117.pdf.

Appendix A Requirement and Verification Table

This table lists each high-level design requirement, the corresponding verification procedure, and its pass/fail status. Any requirement is marked “Y” for verified and “N” for not verified, with reasoning for its failure.

Table 3 – Requirements and Verification

GHF-10

Requirement	Verification	Verification Status
The GHF-10 system must detect a short circuit within 10 ms	• Power to the affected circuit is cut-off and reduced to 0 – 0.3 V within 50 ms • Time of cut-off is measured and entered stored log files based on ADC output values • The system must resume normal operation within 5 seconds • During testing, no permanent damage to relevant subsystem components after 10 trials	**N** We wanted to test this if we had more time, but this seemed too risky to attempt as we might damage our circuit and/or our components.
The GHF-10 force pressure sensor must detect occupancy weight of 50 lb or more	• Must detect applied weight within +/- 5 lb of accuracy • Must maintain 95 % accuracy in distinguishing actual occupancy from fluctuations in sensor readings • Output voltage to the ADC pin must correlate to approximately 2.3 – 2.6 V to ensure reliable transmission	**Y**
The GHF-10 system interfaces properly with the ADC pins of the ESP32	• Known weights of 10, 50, 100 lb (more can be added if time allows) will be added onto the pressure sensor • Each weight range must properly detect the correct occupancy weight • A reasonable output voltage within +/- 0.3 V must be applied and read from the ADC pins of the ESP32 • Output voltage must not exceed	**Y**

	3.3 V or output below 0.0 V	
The GHF-10 system will interface properly within a simulated gym environment	• Temperatures between 65-, 70-, and 75-degrees Fahrenheit will be simulated to ensure reliable operation across different gym environments • The foam pad will likely act as an added insulator, which may increase the temperature • Since the GHF-10 system is properly insulated, on the other hand, any rain/snowy conditions will not need to be tested	**Y**
The GHF-10 system reliably updates to the ESP32 and AWS Server accordingly	• The microcontroller must process occupancy data (output from Arduino IDE code) within 3 seconds • Updated availability must be transmitted to the central AWS server within 1 second of an occupancy change, leading to a total response time from the GHF-10 → ESP32 → application of 4 seconds +/- 2 seconds	**Y**
Software for interfacing between the GHF-10 and ESP32 is accurate	• The system must correctly determine the user occupancy based on the 50 lb threshold and signify a successful occupancy status accordingly • Forces below 50 lb will ensure that occupancy is not detected • The voltage output must be correctly read according to 12-bit resolution with an error margin of +/- 0.3 V	**Y**

ESP32-WROOM-32D

Requirements	**Verification**	**Verification Status**

● The ESP32 must require proper voltage regulation	● Must receive a stable 3.3 V power supply, stepped down from a 5V USB source by the AMS1117-3.3 ● Voltage must remain within the ESP32's safe operation range, which is from 3.0 - 3.6 V. Since none of our peripherals will exceed 3.3 V, we will put this value at 3.4 V to be safe ● Output of voltage regulator must be confirmed with a multimeter to be 3.3 +/- 0.1 V to be safe ● Voltage fluctuations under load will be measured using an oscilloscope ● High computation loads will be placed to ensure voltage remains stable ● In the event of a short circuit, power must be cut off immediately to 0.0 V to prevent damage to other components	**Y**
● The ESP32 must be able to communicate securely with the AWS server	● The ESP32 must connect and maintain a stable connection to a designated Wi-Fi network, and print statements using its built-in WiFi.status() function will be tested in a loop of 500 ms intervals ● RSSI signal strength and packet loss data will be monitored ● In the event of a network disruption (which can be simulated by manually shutting on/off a personal hotspot), it will be tested 5 times, and each time, automatic reconnection must be met ● Using AWS built-in services, compare sent and received hash values for data integrity ● Measure round-trip latency using timestamps to ensure 1 second communication	**Y**

• The ESP32 must interact with the GHF-10 using ADC	• Measure voltage output across varying levels of 0 to 110 lb and ensure that output is within 0 - 3.3V • To validate, there will be no tolerance accepted above 3.3V for higher pounds of force. If we have ~2.5 V of output at 50 lb, any tolerance below 3.3V for 110 lb of max force is fine (as we just detect occupancy, not exact weight) • Voltage readings using a multimeter should match the ESP32's analogRead() within +/- 0.3 V	**Y**
• The ESP32 must correctly exchange data with the MFRC522 via SPI protocol	• The ESP32 should accurately read RFID tags within 25 +/- 10 mm (rated for a max of 50 mm). • Testing RFID tags at different distances from 5 mm, 10 mm, 25 mm, 50 mm, and a success rate of ~25 scans at each interval • Attempt a failed read with an unsupported key fob type (for example, my apartment key fob) at 5 times, and must re-attempt scan each time	**Y**

MFRC522

Requirements	**Verification**	**Verification Status**
• Ensure that the MFRC522 is correctly supplied with power	• Ensure power supply is off at the beginning • Ensure that DVDD, AVDD, TVDD, PVDD pins are physically connected to the positive voltage rail of the power supply • Verify that DVSS, AVSS, PVSS, TVSS are physically connected to a ground rail • Using a multimeter, check for	**Y**

	short circuits between the power rails and ground (there should be 0 shorts) ● Turn on the power supply and set it to the correct voltage for the MFRC522. Measure voltage at DVDD, AVDD, TVDD, PVDD. Ensure voltage is within 2.5 V – 3.3 V range for these positive rails. For the ground pins, they should measure 0 V.	
● The crystal oscillator must be correctly connected and oscillating	● Ensure power supply off ● Verify that the 27.12 MHz crystal is physically connected to OSCIN and OSCOUT. Ensure capacitors are grounded ● Use an oscilloscope to probe the OSCIN and OSCOUT pins ● Verify that a stable oscillating signal at 27.12 +/- 0.5 MHz is present. This can be measured using the frequency of the signal.	**Y**
● The antenna circuit must be properly connected to the PCB	● Ensure power supply off ● Verify all relevant components are physically connected to the pins in the correct configuration ● If available at the ECE 445, use a network analyzer to measure impedance and resonant frequency of the antenna circuit ● If not available, connect the MFRC522 to the microcontroller, and attempt to read an RFID tag. If the tag can be read, the antenna circuit is functioning as intended. Repeat this 5 times in a row with a 100 % success rate.	**Y**

TYPE-C-31-M-12 & AMS1117-3.3

Requirements	Verification	Verification Status

● The AMS1117-3.3 regulator shall provide a stable 3.3 V +/- 0.1 V output to the ESP32 at an input of 5 V +/- 0.25 V from the USB-C	● Apply a voltage of 5 V +/- 0.25 V via the USB-C connector ● Measure the voltage at the VCC_3V3 pin of the AMS1117-3.3 using a multimeter ● Using a multimeter and variable electronic load, increase load to 500mA and verify output voltage is stable within 3.2 V and 3.4 V	**Y**
● The output voltage at the ESP32's VDD pin must be 3.3 V +/- 0.1 V	● Using a multimeter, ensure that the output voltage at the VDD pin of the ESP32 (ESP32_3V3) must be within the specified range ● Ensure that the measured voltage is between 3.2 V to 3.4 V	**Y**
● The power subsystem must be able to withstand continuous input of 5 V without damage	● Connect USB-C power source to supply continuous 5 V ● Under max load at 500 mA, monitor all components to ensure no overheating (namely, the linear regulator) ● Ensure no components exceed 80 degrees Celsius under operation	**Y**

Appendix B Final PCB Design

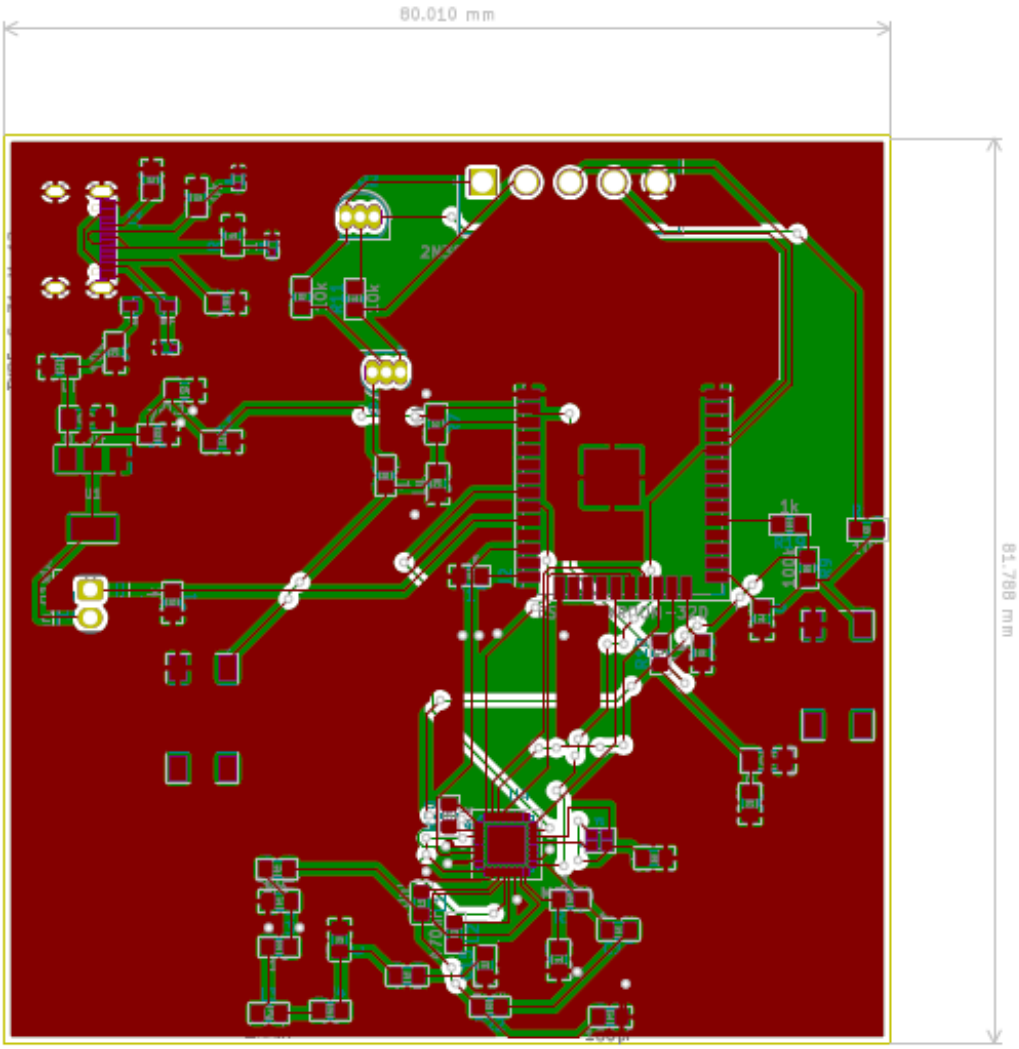


Figure 13 – Final PCB Design