Final Project Report

Climber Timer

Team 16

Supervisor - Dr. Giordano Scarciotti

Alexander Prescott - 01191642
Archit Sharma - 01199766
Benjamin Biggs - 01064601
Bonne van Oordt - 01242762
Tamara Elmani - 01081131
Thomas Poskitt - 01059987
Tomasz Bialas - 01205145

Contents

1 Abstract

There is currently no commercially available automated system to log what problems a climber completes or how quickly they are completed. The project attempts to solve this problem by creating a network of RFID tags that a climber can scan using a rechargeable, active RFID wristband at the start and end of each problem. The wristband sends data to the database server via Wi-Fi, which can then be accessed by the climber via a website.

The final product can easily be retrofitted to an existing climbing wall, with minimal changes to the infrastructure of the centre. The wristband has a battery life of approximately 20 hours and is rechargeable via a micro-USB port. It is also simple to use and implement, making it an attractive option for climbing centres looking to improve the user experience.

The prototype was tested in a variety of scenarios to observe its performance. It was found that it functioned appropriately given the goals of the project. From these tests, it was concluded that a project of this scope could be successful if it was released within the indoor climbing market.

Further improvements to optimise performance and functionality are possible but the prototype has proved the projects potential for success.

2 Introduction

Using technology to advance sport achievement has rapidly become embedded in today's society. Data analytics have been applied to a variety of sports to allow athletes to track and improve their performance. The market for wearable devices and applications used for developing and coaching athletes is fast-growing, with the sports coaching platform technology market predicted to grow to $864 million by 2021, a steep increase from its value of $49 million in 2014.[1]

Bouldering is a form of rock climbing performed on small rock formations or artificial rock walls in a climbing centre without the assistance of ropes or harnesses. Within one of these climbing centres, there are a multitude of unique problems on each wall that vary in difficulty and in the skills required to complete the route. There is currently no automated way to keep track of what problems have been completed or the time taken for a climber to ascend. The aim of this project is to create a wearable device that logs activities in a climbing centre with the extension of providing a community platform for users to discuss how they approached the problem and compete for ascent times.

There are a few other devices or applications currently on the market that serve a similar purpose. In terms of tracking activities in a climbing centre, there are solutions such as the Climbax[2] wristband, the Whipper tag[3] and a couple of basic tracking apps that do not log any activities automatically. The unique selling point of the Climber Timer stems from its ability to automatically log activities whilst differentiating between individual problems on a climbing wall, and providing problem-specific data to users.

This report will revisit the rationale for the final design of the product, outline any changes made since the Preliminary Concepts Report, demonstrate the steps the group has taken in order to develop a working prototype, testing procedures carried out and how this report can be followed for the benefit of any future work.

3 Project Management

The team was split into two broad subgroups: software and hardware. The hardware subgroup focused on the physical wristband circuitry, connectivity via RFID and Wi-Fi and accurate timing with NTP. The software subgroup focused on the server and how to receive and store data from the wristband into the database, as well as facilitating autonomous data transfer from the database to a website that users can access.

A Gantt chart (appendix B) was drawn up at the start of the project to give milestones to aim for as the project progressed.

As shown in the Gantt Chart, work on both hardware and software was completed in parallel, which is reflected by the groups meeting schedule. Prior to the division of the group, short meetings were held on a weekly basis in a lab environment. After the division of the group, every second meeting was replaced with a subgroup meeting. General meetings served to keep both subgroups working cohesively to ensure that future interfacing would not be an issue. All minutes can be found in appendix C. Group communication was done via Slack and file sharing was done via a shared Google Drive.

4 Design Specification

The interim report specified the important design criteria from the product design specification, but these have since been tweaked and appropriately justified after further deliberation and early prototyping findings. The product had to satisfy the following criteria:

Performance: the product should be able to record: whether a problem has been completed, number of attempts taken to complete a problem, and time taken for a successful ascent. The product should also provide further information on the problems: grade, setter, set date and estimated strip date. This data would have to be input manually as they cannot be determined automatically. Since this information is provided by the centre, the product would still be convenient and automated for the user. Data that has been updated should be immediately available to the user via an appropriate, user-friendly interface.

Environment: the product is to be used within indoor climbing centres and should therefore be able to function in high humidity and chalk concentration. Components of the device should be appropriately encased if they are susceptible to these conditions. The product must be able to withstand scrapes and knocks as a result of climbing and falling. The functional temperature would generally be around room temperature, but depending on the location of the climbing centre and the time of year, an appropriate temperature range would be -10°C to 40°C.

Operating time: in the case of battery powered components, the minimum battery life is the maximum duration of a climbing session which is approximately 4 hours. A more realistic goal is a lifespan that takes into account the possibility of consecutive climbing sessions, without the opportunity to recharge. Therefore an operating time of 16 hours would be more practical, ensuring the availability of the devices during busy days.

Target product cost: the closest comparable products/technology have a retail price of approximately £140[3] per device. However, the product that is being created will not be a stand alone product, it is a system that will be installed in climbing centres. The wristbands would then be rented out to customers. The price per wristband will likely be cheap, but the system cost will be higher. An appropriate price for a single wristband would be £20, with a fee of £200 for the rest of the system.

Size, weight and ergonomics: an important criterion is that the product does not interfere with the climbing experience and capabilities of the user. It is also vital that the system is not inconvenient for the climbing centre to fit to a problem. This means that wearables should be lightweight and small. A decent reference is a large wristwatch, with dimensions and weight limited to approximately 55 x 55 x 20 mm and 120 g. For this product, lighter and smaller is better. The wall-mounted tags should also be small and in particular, not protrude from the wall, to avoid accidental damage or interference. With a depth of no more than 15 mm, the height and width are less important but should be limited to the size of a small hold: 50 x 60 mm. Considering ergonomics, wearables must: be comfortable and unobtrusive, not limit the movement of hands or feet and be secure yet easy to remove. Notably, the product must be constructed from skin-friendly and shock-resistant material where relevant with no sharp edges. Wearables and other parts should not contain user-operable controls (switches, buttons, etc), but rather work automatically using an intuitive method. The system should have audio or haptic feedback to inform the user of a successful connection (to a tag).

Safety: batteries must be suitably encased and current-limited to prevent burns to skin in the case of a malfunction. The structural integrity of the holds and wall must not be compromised. No parts may rest on the crash mats as this would compromise the safety of a climber should they fall. For the same reason, any potentially harmful components worn by the user must be encased in resistant material.

5 Development Process

Once the problem the group was trying to solve was identified and characterised, five solutions were drawn up, as detailed in the Preliminary Concepts report. A brief outline of each of these solutions and the rationale behind the final design choice can be found below.

In order to fully understand the design specifications and important features of a device that could log activities in a climbing centre, the group conducted practical market research at two climbing centres, Westway and Vauxwall. This research included speaking to the operations managers about the potential of the product from a business point of view.

> *"The indoor climbing market is currently growing at an ever increasing rate. With the number of climbing centres in London alone exceeding 15, there is definitely room for a product that performs these functions."*
>
> – Jonny White, Westway Operations Manager

Active wristband, passive wall tags: RFID passive tags attached to climbing wall next to start and finishing holds for climbers to tap upon start and completion. This was the chosen solution.

Passive wristband, active wall tags: RFID active wall-mounted tags next to start and finishing holds for climber to tap upon start and completion. Implementing an active wall tag in a climbing centre may prove to be problematic as it needs to be connected to constant power supply. This would make it difficult to rearrange the tags when the problems are changed.

Camera: real time tracking of climbers during their ascent using a device with a camera. This solution would have been infeasible given the time frame for this project and the complexity of processing required. It also wouldn't have served as a practical solution for a climbing centre due to space issues and high added expense.

Wearable smart tag: a device that uses various sensors to track the movement and position of the climber as the problem is completed. Upon further research into the metrics required for this solution, it was clear that it would be very difficult to measure problem specific data, especially given that there would be no way to know when the problem set was complete. Very accurate sensors and comprehensive algorithms would be required to estimate the user position to a sufficient degree of accuracy.

Smart holds: all necessary components are contained within holds and attached at the top and bottom of each problem. Data would be logged with respect to the time between the climber scanning their finger at the bottom and the top of the problem. This cannot be retrofitted onto an existing wall as the climbing centre would need to buy specific holds that could only function as either first or the last hold of a problem. This would be inconvenient for climbing centres as well as expensive due to the cost of producing these smart holds and number of holds required. Fingerprint scanning would also be unreliable due to the nature of the sport itself, as climbers' hands often retain frequent 'wear and tear', get very sweaty and covered in chalk.

In order to further assist in deciding which concept best fulfils the main criteria outlined in previous reports, a concept selection matrix was created, as shown in figure 1. The main deciding factors for choosing the 'active wristband, passive wall tags' solution were the feasibility given the time available for the project and the flexibility it would give to climbing centres when updating problems on the wall.

	Active Wristband	***Passive Wristband***	***Camera***	***Smart Tag***	***Smart Holds***
Accurately record problem specific data	+	+	s	-	+
Ease of information transfer and display	s	s	s	s	s
Low latency updates of data	s	s	-	s	s
Feasibility of design implementation	+	s	-	+	-
Environment	s	s	-	s	s
Power consumption	+	-	s	+	-
Battery life	s	s	s	s	+
Life in service	+	+	s	s	s
Overall solution cost	+	s	-	+	-
Size	+	+	-	+	-
Weight	+	+	s	+	-
Minimal user controls	s	s	+	s	+
Shockproof	+	+	-	+	s
Safety	s	s	s	s	s
Total	8	4	-5	5	-2

Figure 1: Concept selection matrix

6 Technical Design

6.1 High Level Design

The project was broken down into two main systems: hardware and software. The hardware refers to the wristband and RFID tags, while the software system encompasses the database and the website. The software and hardware systems can be broken down into several subsystems as shown in figure 2 and 3.

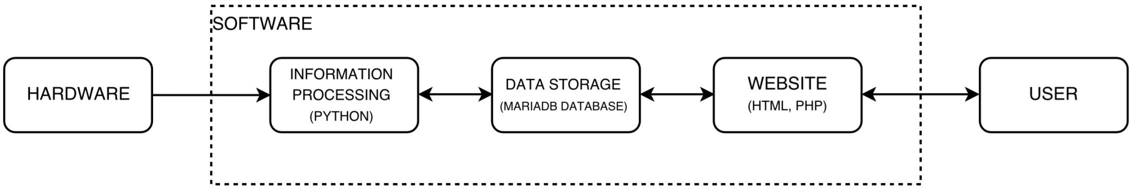

SOFTWARE
HARDWARE
INFORMATION PROCESSING (PYTHON)
DATA STORAGE (MARIADB DATABASE)
WEBSITE (HTML, PHP)
USER

Figure 2: High level design - software

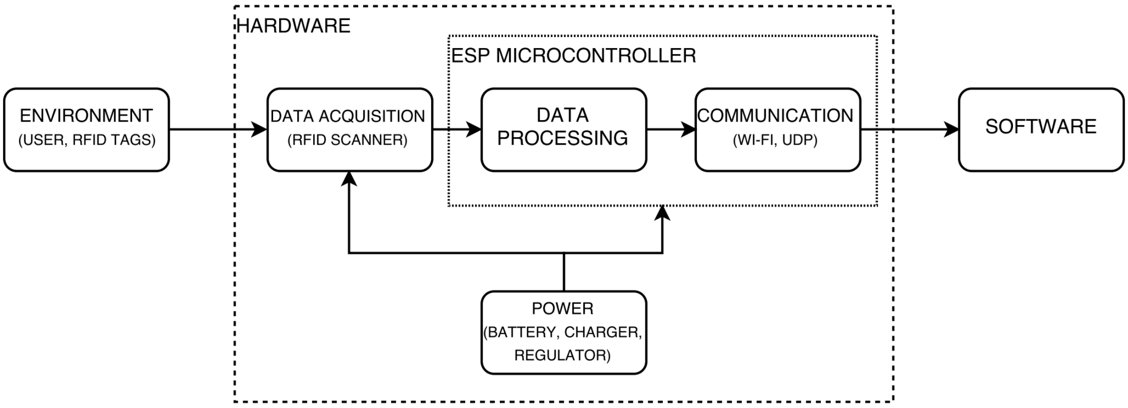


Figure 3: High level design - hardware

6.2 Hardware

6.2.1 Overview

A prototype wristband needed to be manufactured to follow the specifications outlined in the PDS. This meant that it had to record the ascent time to the nearest 100 ms and then upload this information as soon as possible. It was also desirable that the wristband would notify the climber of a successful scan with haptic or auditory feedback. For the convenience of the climber, the wristband had to last for at least 4 hours (approximate time of a long climbing session) on a single charge. Ideally the wristband could last 16 hours on a single charge so it would only need one charge daily which would be convenient for the climbing centre. For added convenience, the device needed to be rechargeable from a standard micro USB cable. Most importantly, the case had to be unobtrusive to the climber and not provide any significant handicap.

6.2.2 RFID Tag

To identify a passive tag attached to the climbing wall, the wristband requires a low-power RFID module. The MFRC522 development board was chosen to implement this because of its low cost, availability and its open source libraries. Once work had started on this board, it was discovered that the development board was hard wired to use SPI, not I^2C despite the MFRC522 being able to do both. Using I^2C would have been more convenient but SPI worked adequately.

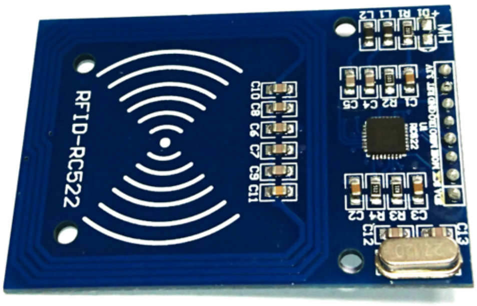


Figure 4: MFRC522 development board

Figure 5: ESP-12

As Wi-Fi has been chosen as the mode of communication of the wristband to the rest of the data processing infrastructure, a microcontroller with integrated Wi-Fi functionality was chosen for simplicity of implementation and small physical size. The ESP-12 (figure 5), an ESP8266 microcontroller-based module, was selected for: its small form factor, integrated Wi-Fi antenna, software library compatibility with the Arduino ecosystem, community support due to its popularity and relatively low cost.

The MFRC522 is advertised as having "flexible interrupt modes"[4]. However, it seemed impossible for the module to send an interrupt to the ESP-12 to wake it from sleep mode. This was disappointing because the initial design relied on using the ESP-12 microcontroller in deep sleep mode which uses very little power. It would only awaken when a tag was scanned. This however, was not possible.

6.2.3 Sleep Mode

As the wristband is a wearable, battery life is a critical characteristic of the system. Therefore power saving methods need implementing. The ESP-12 offers several power saving modes[5]:

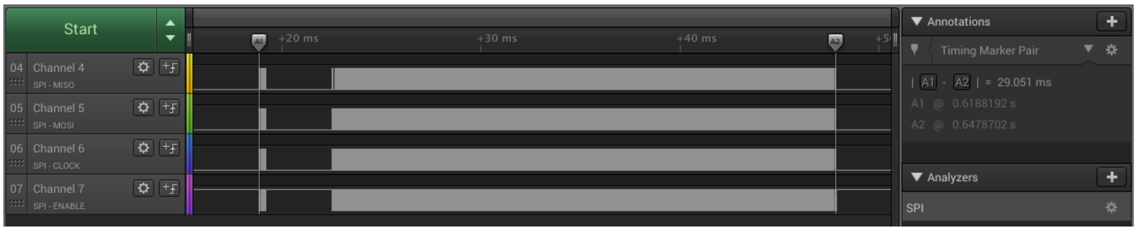
Deep sleep: the microcontroller core, the system clock and the RF modem are powered down but the RTC clock remains active. This mode can only be exited through a reset, which can be triggered through an RTC timer. The current draw in this mode is approximately 1 µA.

Light sleep: the microcontroller core, the system clock and the RF modem are powered down. However, any wakeup events (including RF baseband/Wi-Fi media access, the RTC timer or external interrupts on GPIO) can wake up the microcontroller. Current draw in this mode is approximately 1 mA.

Modem sleep: only the RF modem is powered down. Current draw in this mode is approximately 15 mA.

Initially, the RFID reader was expected to issue an interrupt request when reading an RFID tag, waking up the microcontroller to read the data. However, further testing showed that the RFID IC does not issue IRQ signals when the main communication interface is not activated, which meant modem sleep could not be implemented.

Instead, a polling method was established by putting the system to sleep and waking it up periodically. A standard card polling sequence was timed to last 30 ms as seen in figure 6. A sleep frequency of 10 Hz was then chosen to match the 0.1 s timing accuracy requirement of the PDS, giving a wakeup cycle of 30 ms and a sleep cycle of 70 ms.



| A1 - A2 | = 29.051 ms
A1 @ 0.6188192 s
A2 @ 0.6478702 s

Figure 6: SPI transaction

Finally, in order to maximise the amount of power saved, light sleep with the modem disabled was chosen as it offered the lowest power consumption while still giving a negligible wakeup time. On tag detection, the modem is powered on, a connection to the Wi-Fi network established, the data is sent and the core and modem are powered back off, allowing for the cycle to continue.

Finally, the RFID chip is also powered down when the system enters sleep, and re-enabled on system wakeup. The power LEDs of the RFID board and the ESP-12 were also disabled.

Testing

The current draw of the system was measured using the voltage drop across a 1 Ω resistor.

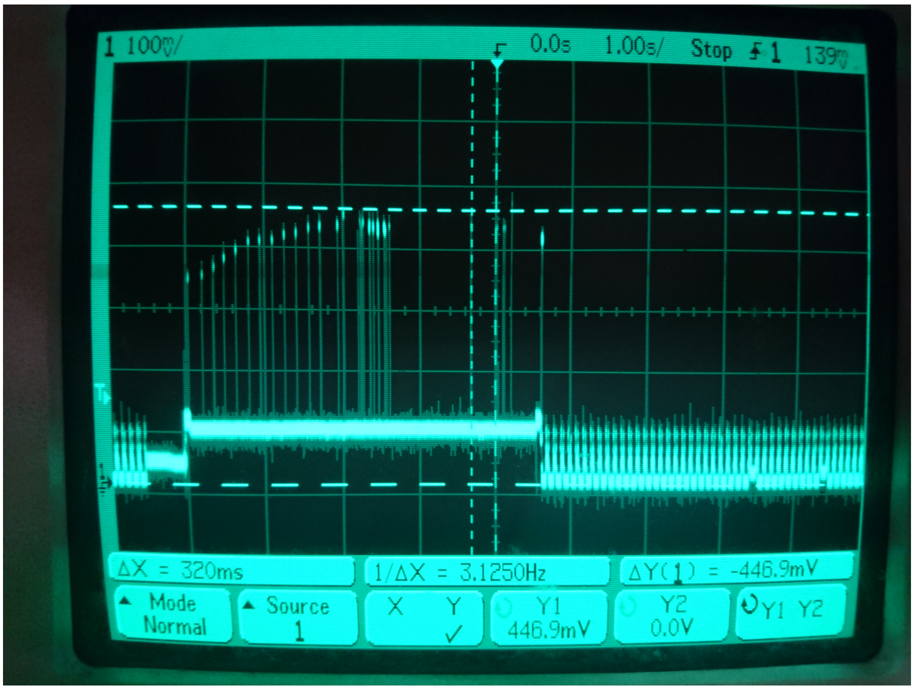


Figure 7: System current draw after card detection

As shown on figure 7, the Wi-Fi communication window draws approximately 70mA with peaks reaching approximately 450mA. This data will be used later for power regulation design.

The average idle system current draw was measured to be approximately 15mA, down from approximately 85mA (70mA for the ESP-12 with active Wi-Fi and 15mA for the RFID reader). See appendix F.3.

The power saving software optimization can therefore be considered a success, as the power consumption has been decreased by a factor of five; Wi-Fi communication windows are not included in this metric as they constitute an insignificant amount of time when compared with the time spent in an idle state.

6.2.4 UDP Packet Transmission

After researching the Arduino libraries associated with the ESP-12 module, the UDP library[6] was found. The User Datagram Protocol (UDP) is a simple method of sending packets of information via Wi-Fi.

A second method of packet transmission called TCP was then considered. This system was more robust as the sender requires a handshake to confirm that the correct data packet has been accurately received.

UDP was chosen to: reduce the software complexity with regards to data transmission, reduce complexity of the receiver, and reduce the time required to implement a working communication method from the ESP-12 to the server.

The packet composition was chosen to minimise the amount of processing and communication time. The main block of code that is responsible for communication is outlined in appendix F.1 and was derived from online examples[7]. The block initialises the UDP data stream with the IP address of the server and the port number on which the server is listening. These are arbitrary values that can be changed to fit different server configurations.

The code writes the following information to the UDP packet:

- RFID UID (card identifier)

- Wristband reader ID
- UNIX timestamp in seconds
- 100s of milliseconds calculated by the internal clock for additional accuracy

The information is separated by a delimiter (\r\n) to make decoding the packet easier for the receiver.

6.2.5 Timing

The first method proposed for calculating the time taken for a climber to complete a problem was to have the ESP-12 stay out of sleep mode and count the time between the scanning of a start tag and a finish tag.

The second method proposed was to utilise the Network Time Protocol library[8] to synchronise the system clock on the ESP-12 and provide time stamps associated with every scan. NTP is an internet protocol that coordinates the clocks of computing systems to a given reference time[9].

Using the first method would require the nature of the tag (start, finish, login, etc) to be determined and different attempts at a problem to be distinguishable. These extra logic functions would complicate the code but the benefit would be a higher degree of timing precision, to the nearest 30ms as opposed to 100ms.

The lower power consumption and reduced logic complexity was deemed preferable to the increased accuracy. Therefore, NTP was the chosen method. The full code is in appendix F.1 and was implemented using the NTPClient library.

The ESP-12 documentation[5] states that the Real Time Clock may be prone to a drift in the order of milliseconds. In future versions of the code the ESP-12 will reconnect with the NTP pool and synchronise more often so that the effect of the drift is minimised.

6.2.6 PCB

Whilst the initial development of the wristband was done on breadboard, a PCB was made for the final prototype to reduce the size and weight of the wristband, and to add supporting power management components. The two main aspects to this were power and PCB design.

Power Development

The goal was to design a rechargeable battery-based power source for the system to run off, with the specification that it must be rechargeable via micro-USB and have a battery life greater than 4 hours.

A battery needed to picked first as the charging and voltage subsystems were subject to the battery specifications. The lithium-polymer (Li-Po) battery LP-402933-1S-3[10] was chosen for its high energy density, small physical size, 300 mAh capacity (offering approximately 20h of runtime) and 600 mA maximum current output (above the 450 mA Wi-Fi transmission peak current draw). The Li-Po chemistry also allows for nominal voltages higher than 3.3 V, meaning a step-down voltage regulator could be used.

To charge the battery, the MCP73833 charging controller[11] was chosen because it supports the Li-Po chemistry, 4.2 V charging voltage, 5 V input voltage (USB power rail nominal voltage) and has a small MSOP-10 package suitable for manual soldering. The -833 variant was picked for its battery cell thermal monitoring in addition to its standard thermal regulation features, making it safer to use.

Because battery output voltages are often unstable, a voltage regulator was required to control it. A switching mode step-down regulator was chosen for its high efficiency compared to a linear voltage regulator. The power rail voltage was fixed at 3.3 V as both the microcontroller[5] and the RFID chip[4] on the board support it and it was closer to the lower limit of the battery voltage. 3.3 V was chosen over 3 V to prevent potential voltage drops during RF activity from causing either of the ICs to enter brownout. The TI LM3671-3.3 voltage regulator [12] was chosen because it supports 3.3 V regulated output voltage, 600 mA continuous current and input voltages of 3.3 V to 4.2 V. It is also available in a compact SOT-23 package with few external passive components and fully accessible pins.

PCB Design

A schematic was made to imitate the breadboard prototype, as well as contain the core passives needed for the ESP-12 to start-up (given by the Adafruit Huzzah breakout schematic[13]). Power IC schematics were also drawn up based on their respective datasheets[12][11].

The PCB outline was drawn to fit under the RFID PCB next to the battery in order the keep the wristband as small as possible. Components needing a given position (connectors, buttons etc.) were placed first, followed by the ESP-12 module and the power ICs.

The PCB was then routed following the design rules for class 6C manufacture at Eurocircuits[14] to keep the expenditure down. Traces (power traces in particular) were made thicker to prevent any voltage drops across the board. The full PCB schematic and layouts can be found in appendix G.1, appendix G.2 and appendix G.3.

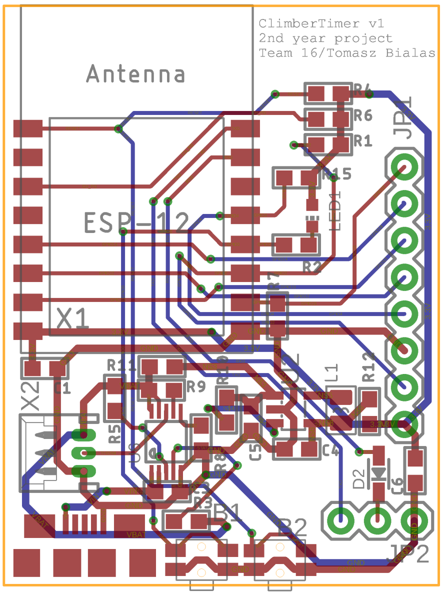


Figure 8: PCB layout

A modified version had to be drawn to match the manufacturing specification of Newbury Electronics (PCBTrain Express) as the PCBs needed to be reordered. This was due to the original courier failing to deliver parcels from Eurocircuits on time, with a delivery delay of over a week on a next-day delivery.

6.2.7 Enclosure

Initial research into current wrist mounted wearables meant we investigated the designs of products like the Apple Watch[15] and the Fitbit[16]. Both these designs opt for a streamline, lightweight casing. Taking this into consideration a minimalist, futuristic aesthetic was chosen for the wristband enclosure. As previously mentioned, the wristband should not restrict the climber in any way so a curve was added to the back plate.

The PCB, RFID reader and battery were then modelled in Solidworks so that the enclosure size could be adjusted and the correct position for the micro-USB charging port could be added. A clearance of 1 mm was given between individual parts so that the internal components could be fitted with additional

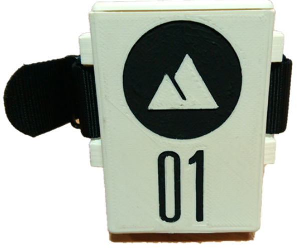


Figure 9: Completed Enclosure

foam padding to absorb impacts from climbing. This will minimise damage to the sensitive parts of the wristband like the RFID antenna, in addition to protecting the potentially hazardous Lithium Polymer battery.

As part of the development process, the first iteration of the enclosure was 3-D printed and then exposed to drops and general wear and tear. The first print (appendix H.1) was too weak to allow for a generic wrist strap to be attached. The second iteration (appendix H.2) was designed to have more material along the strap holders to retain their strength. This design was subsequently printed, tested and deemed to be effective so work was started on the model of the enclosure lid (figure 9).

In keeping with modern products it was decided that the logo should be displayed without the product name. The wristband identification number is also printed underneath this so that staff can quickly identify malfunctioning products and send them out for maintenance. An appropriate font was sourced to fit with the aesthetic.[17]

6.3 Testing

Hardware tests

A set of successful, simple tests were performed to evaluate the functionality of the final product:

- The battery charging circuit worked, it charges the battery when connected to a 5V micro USB supply.
- The power regulator output was recorded as 3.3V with no load.
- The ESP-12 module initialised when injected with 3.3V.
- The ESP-12 connected to Wi-Fi and was able to send data. It was programmable through the programming header on the board.
- The SPI connection from the ESP-12 module to the MFRC522 board was successful, and data was transmitted correctly.
- The buzzer was audible on card scan.

Load Test

The load tests performed on the final wrist band product showed a large voltage discrepancy down to 2.5V, as well as audible coil whine on the inductor.

The PCB was designed following the TI specification but there was an issue with a mismatched capacitor. This has since been confirmed by a TI engineer[18].

The capacitor was chosen for the use of X5R as a dielectric material, the low Equivalent Series Resistance (ESR) and the capacitance value. Unfortunately the capacitance under 3.3V bias was too low, causing oscillation and instability.

Since the circuit works under no load, this issue would be solved with capacitor replacement and a closer ground connection (already implemented in the Eurocircuits board version).

Timing Test

The timing test was performed using a stop watch and an altered version of the code that would constantly print timestamps every second. The aim of the test was to obtain the amount of drift for a given duration and compare it to the range specified by the manufacturers.

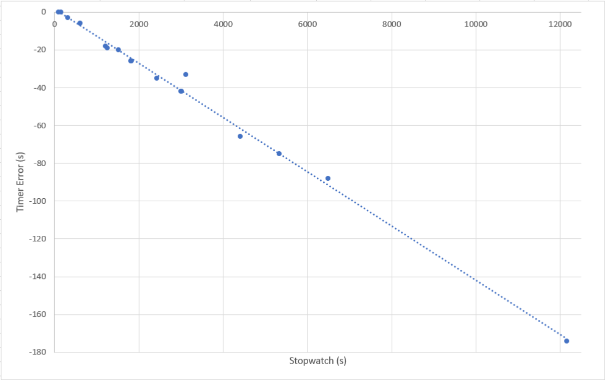


Figure 10: ESP-12 Timer Lag from Correct Time

The drift lag measured after 12159s (approximately 20 minutes) was 174s. A full record of the testing can be found in appendix F.2. This was worse than stated in the ESP-12 Documentation. It was discovered that the light sleep function was causing the drift to increase. As a result it was decided that the sleep function should be used infrequently and for short periods of time.

6.4 Software

6.4.1 Overview

The aim of this system is to receive packets from the hardware system, interpret these packets, store the relevant data in the database and display the contents of the database on a web interface. The web interface must also allow a user with admin privileges to alter specific data in the database.

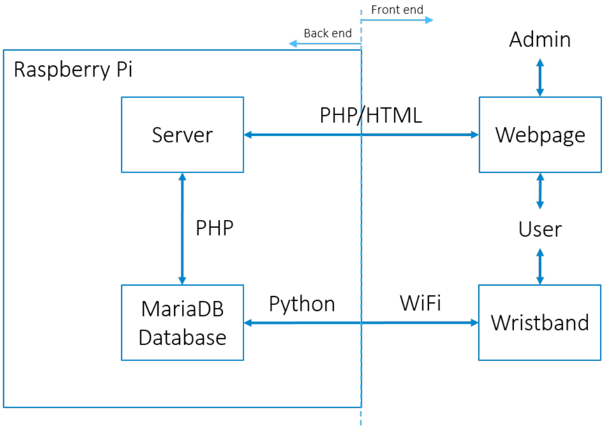


Figure 11: Network block diagram

A Raspberry Pi will be used to host the database server along with the port listening software. This will be the hub for the product as shown above. The user will interact only with the wristband and the website. The website will use PHP to communicate with the MariaDB server while the wristband will sent UDP packets to the port listening software (UDP Receiver).

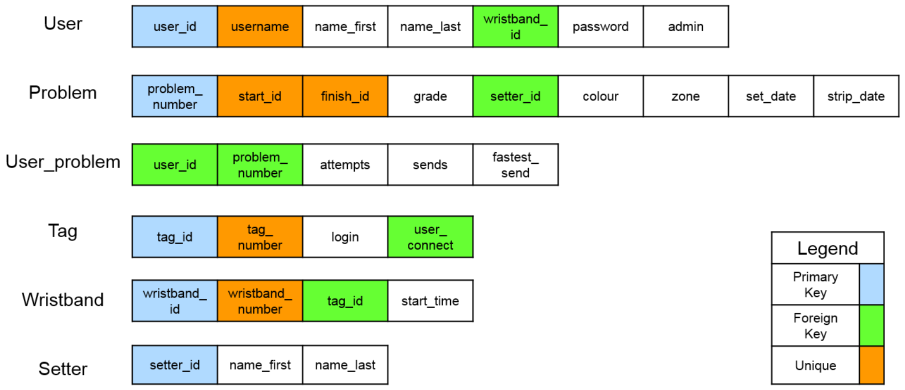
6.4.2 Database

Overview

To store all the user and problem data, a MariaDB database server was used. MariaDB is an open-source offshoot of MySQL, a relational database system. Using relational database, means tables within a database can reference one another using foreign keys. This maintains integrity within the database as data cannot be input into foreign key columns if the data doesn't exist in the column the key is referencing.

System Concept

The basic idea of the database structure was to have a `user` table, a `problem` table and a `user\_problem` table which creates a record for each unique user-problem pair. To save on data storage, setters were stored in a `setter` table so they could be referred to by number in `problem` instead of by name. It also allows for future data manipulation only concerning setters to be much more easily implemented. This was then also done for `tag` and `wristband`. All tables except `user\_problem` have a primary key, a column that must be filled and must be unique for all rows, preventing table entries with duplicate IDs. This maintains data integrity by preventing identical wristbands and the like accidentally being recorded. A unique column is like a primary key, but can be empty. The final database structure is shown in figure 12, and the database was created using MariaDB's tutorials on MySQL[19] as a guide.



User	🟦 user_id	🟧 username	name_first	name_last	🟩 wristband_id	password	admin		
Problem	🟦 problem_number	🟧 start_id	🟧 finish_id	grade	🟩 setter_id	colour	zone	set_date	strip_date
User_problem	🟩 user_id	🟩 problem_number	attempts	sends	fastest_send				
Tag	🟦 tag_id	🟧 tag_number	login	🟩 user_connect					
Wristband	🟦 wristband_id	🟧 wristband_number	🟩 tag_id	start_time					
Setter	🟦 setter_id	name_first	name_last						

Legend	
Primary Key	🟦
Foreign Key	🟩
Unique	🟧

Figure 12: Database structure

Most of the table columns are self-explanatory, but a few are of note. `tag\_number` and `wristband\_number` exist despite them having ID columns so the numbers can be printed on the items for easy identification. `login` is a boolean value that allows any tag to be used to assign a wristband to a user, as well as adding new wristbands to the database. `admin` is a boolean value that determines if a user is an admin for the website. `user\_connect` contains the ID of a user who is being assigned a wristband. If a wristband then scans the tag that has he user ID within 5 seconds, the wristband will then be assigned to the user.

Testing

There were few tests that could be performed on the database without the website or the UDP receiver, so all that could be tested was whether data that was nonexistent in referenced columns could be input into columns with the relevant foreign keys, and if it could then be updated or deleted afterwards. This was tried in all foreign key columns, and the database performed as expected.

Modifications

The main modification made was when the MariaDB server was moved from a laptop to a Raspberry Pi. Instead of running the server as a root user, it was run as an ordinary, unprivileged user to increase security as otherwise a malicious attacker could create files posing as the root user who has no permission restrictions.

6.4.3 UDP Receiver

After the wristband sends out the packet over Wi-Fi, a server-side program needs to receive the packet, process it and input it into the database. It was decided a Python script would be suitable as they are simple to run on the Raspberry Pi.

For the server to receive the UDP packet sent by the wristband, a UDP port needed to be opened on the Raspberry Pi. The Python program (UDP receiver) used port 5000 as it is freely available and then listens for any data sent from IP address “0.0.0.0”, which means it listens to any device transmitting to that port. Once the packet is received, the receiver parses the packet into its components and then uses a logic flowchart to process the data for the database correctly.

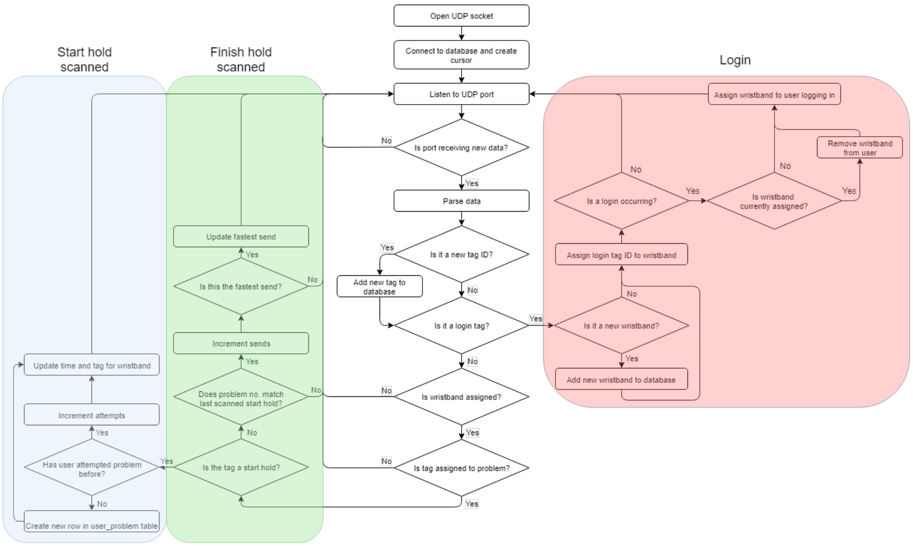

Open UDP socket
Connect to database and create cursor
Start hold scanned
Finish hold scanned
Login
Listen to UDP port
Assign wristband to user logging in
No
Is port receiving new data?
Remove wristband from user
No
No
Yes
Is a login occurring?
Is wristband currently assigned?
Yes
Yes
Parse data
Update fastest send
Yes
Is it a new tag ID?
Assign login tag ID to wristband
Yes
Is this the fastest send?
No
Add new tag to database
No
No
Yes
Is it a login tag?
Is it a new wristband?
Increment sends
No
Yes
Update time and tag for wristband
Yes
Add new wristband to database
Does problem no. match last scanned start hold?
No
No
Is wristband assigned?
Increment attempts
Yes
No
Yes
Yes
Has user attempted problem before?
Yes
Is the tag a start hold?
No
Is tag assigned to problem?
No
Yes
Create new row in user_problem table

Figure 13: Final logic flowchart of the UDP receiver

To interact with the MariaDB server, a `cursor` is initialised which allows Python to write to the database and MySQL and receive the results back. The full code can be found in appendix I.

Problems

A few problems were encountered when initially developing the receiver, one of which was accessing the database via Python. The standard Python library cannot connect to a MySQL server so the library `mysql-connector` was installed. However the receiver still wouldn’t connect because the user the receiver was acting as to access the database didn’t have the relevant permissions to read and write to it. This was solved by using the following command in the MariaDB client: Another issue was that the results the `cursor` returned from database queries came in Python tuplets. This meant `cursor` results could not be directly compared to the packet strings due to the mismatch in data types. This was solved by accessing the result via an index.

Testing

To test the system, mock scenarios were set up within the database and a string structured like the packet the wristband would send was passed through the receiver. The results were found by showing the content changes from the tables within the database. The initial tests undertaken are shown in appendix J. The results of the tests were that the system only crashed when a new tag was scanned and when the finish tag scaned was from a different problem to the previously scanned start tag. It should be noted that at first all tests caused the system to crash as the receiver would not commit the changes made to the database. This was solved by using `mariadb_connection.commit()` at the end of the packet processing loop.

One issue that became apparent while testing was that not all the data packets sent were picked up by the receiver. To investigate this, wristband was modified to send 10 packets and the receiver modified to count how many were received. The time delay between sending the packets was varied to see the effect on how many were received. Results are shown in figure 14

	Number of packets received				
Time delay between packets (ms)	***Attempt 1***	***Attempt 2***	***Attempt 3***	***Attempt 4***	***Average (%)***
10	3	2	3	0	20
20	3	1	2	3	22.5
30	6	2	3	2	32.5
40	5	6	2	2	37.5
50	3	8	6	4	52.5
60	6	4	7	5	55
70	4	4	7	7	55

Figure 14: Results from time delay test

At 50 ms 52.5% of packets are received on average and increasing the time delay does not markedly improve the percentage received. Because increasing the time delay further lengthens the transmission time for little gain, 50 ms was chosen to be the new time delay, as well as increasing the number of packets sent to 10. The system can be modelled as a binomial distribution of $X \sim B(10, 0.525)$ where X is the number of packets received. From this, it can be found that the probability at least one packet is received is 99.9%.

Modifications

After some investigation, the cause of the system crashes was found to be when the `cursor` returns `NULL` after a query. This is not a Python tuple, and thus cannot be accessed via indexing the cursor, which the receiver was doing in some `if` statements when comparing the result to `None`. The `cursor` was changed to just return the full string in these instances, and the tests then performed as expected.

Another modification made was to include the variable `old_data`, which would contain the last packet processed by the receiver. This is because the wristband sends multiple packets because the lack of handshaking with UDP means that packets may not be delivered. If more than one of the same packet is received, the receiver will now process it once and ignore all subsequent packets.

6.4.4 Website

The website must have several features that can be split into user features and admin only features. If a user is an admin, they will have access to all the user features plus the specific admin only features. It must also force the user to login so that the web pages are secure. The table below shows these features.

User Features	***Admin Only Features***
Login to a user account	Login to an admin account
Link a wristband to the user account that is signed in	Delete problems from the problem database
View completion data for the logged in user	Add wristbands to the system
View further data on problems that the user has attempted	Alter wristbands in the system
View the full problem database	Add tags to the system
Search the problem database using any characteristic	Alter tags in the system
Logout	Add setters to the database

Figure 15: Website feature breakdown

The website breaks down into seven different pages, of which only six are visible to an admin, and only five to a regular user.

For the full PHP code, see appendix M.

Login Page: the first page a user is greeted with (see appendix K.1). The user can login or sign up.

If the user attempts to login, the website queries the database table `user` to check if the login attempt is valid. If it is, the user is directed to the main user page (User Data) and the input data is stored in a global variable for login verification on different pages. If it is incorrect, the user is notified that the input is invalid.

If the user clicks 'Sign Up', then they are redirected to the sign up page.

Sign Up: the user is presented with a form to fill with basic info (see appendix K.2). If there is a blank field on submission, the page rejects the input and prompts the user to fill out all the fields. When this condition is met, the page creates an entry in the `user` table containing the provided details. A user ID is assigned automatically. The user can then click a login button that returns them to the login page.

User Data: the first page that the global variable containing the login data is called and checked against the database. If it is found to be incorrect, the user is returned to the login page. This check prevents the user from accessing the page just from the URL (known as URL bypassing).

This is the main user page, as it shows the user all of the data that they have recorded thus far (see appendix K.3). It outputs the content of the database table `user_problem` in a visually appropriate table. An extra column called 'More Info' that leads the user to the Problem Data page. When this is clicked, the problem number of the corresponding problem is passed to the problem page by the URL.

Below the table is a set of navigational hyperlinks that lead to different parts of the website:

- **Problem List:** directs the user the Problem Data page
- **Settings:** directs the user to the Settings page (only visible to admin users)
- **Connect Wristband:** directs the user to the Connect page
- **Logout:** returns the user to the home page and clears the global variable containing the login data

Connect: after the normal login credentials check, the user is presented with an input box called 'Wristband Number' (see appendix K.4). The user will then enter the number of wristband they are going to use, and press enter. The page will then begin a five second countdown during which the user will scan a login tag with the wristband. If this is successful, the wristband will now be linked to that user.

The five second countdown is in place to prevent confusion, as it is possible for the login tag to be scanned by another user for a different purpose. By limiting the time the user has to scan, the potential for this error has been reduced.

The page works be setting temp data in the database that is removed after five seconds. A column called `user_connect` in the database table `wristband` is set to the user ID of the current user. At the end of the countdown, the page is temporarily directed the Connect Result page.

This page is a very small page with one basic function. It has no visual representation as the user is immediately redirected back to the Connect page once the required function has been performed. The page looks at the `user_connect` column of the `wristband` table and checks this against the entered wristband number.

By the time this page runs, the database should have been updated by the UDP receiver accordingly. If the user scanned correctly within the five second window, the UDP receiver should have read the user ID from the wristband table, and used this to put the wristband ID into the `user` column that corresponds to the user ID, therefore linking the wristband to the current user.

Before redirecting back to the Connect page, the page clears the data stored in `user_connect` so that the user has to scan within the five seconds.

Problem Data: this page is the largest page functionally. For a standard user, the page will display a list of all the problems in the `problem` table of the database including all the data attached to each problem (see appendix K.5). This includes:

- Problem Number
- Grade
- Setter
- Zone (problem area)
- Set Date
- Strip Date

Above this table, the user will see a 'Search Problem' function which allows the user to query the database using any of the characteristics shown in the table.

Also, if the user arrives at this page having pressed more info on a problem, the table will only show information about that problem. This is done by reading the problem number from the URL.

Below the table is another set of navigational hyperlinks leading to other sections of the website.

As an admin, a couple of extra things are visible: An 'Add Problem' function above the 'Problem Search function', a 'Delete' function and the usual extra navigational hyperlink, 'Settings'.

The 'Search Problem' function works by making dynamic queries. All input fields are checked to ensure the input is valid and then the query is made. The result of which is printed in the table.

The 'Add Problem' function forces the user to input all fields. It will also check certain entries, such as 'Setter' to ensure the setter exists within the `Setter` table. If all is correct, the page performs an `INSERT INTO` query.

Finally the 'Delete' function simply uses the corresponding problem number (as it is a unique field) to perform a `DELETE FROM` query.

Settings: this page is an admin only page that will redirect the user to the login page if they are not logged in as an admin.

A correctly logged in admin has access to several functions which are as follows: 'Add Wristband', 'Add/Update Tag' and 'Add Setter' (see appendix K.6).

For 'Add Wristband', the admin should scan the wristband on a login tag and then press the 'Add' button on the website. The page first queries to check if the selected wristband number is already in use. If not, the page queries the Tag table to get the tag ID of the login tag. It then queries the the Wristband table, to see which wristband last scanned the login tag (as the tag ID of the last scanned tag is stored temporarily in the Wristband table). It then sets the wristband number of the scanned wristband to the value entered in the box.

For the 'Add/Update Tag'. the admin scans the tag they want to reassign with a wristband. Then they enter the wristband number of the wristband they used and the new tag number they want to assign to the tag they just scanned. The page first queries the Tag table to ensure that the tag number is not already in use. If not, it queries the Wristband table and uses the entered wristband number to find the tag ID of the scanned tag. It then assigns the tag ID to the entered tag number in the Tag table.

Finally for the 'Add Setter' it simply performs an INSERT INTO query on the Setter table. A query is made prior to this to check the setter does not already exist.

As usual, the page includes the same set of navigational hyperlinks leading to all other pages of the site.

The functionality described in the PDS has been compared with the actual functionality below.

PDS Feature		***Actual Performance***
The application shall have instant data updating		Data is not updated real time due to the simplicity of the site, however it will update on refresh
The application will also allow the user to view information about a given problem including:	Setter grade (Difficulty assigned by the setter)	The website allows the user to do this
	Setter (person who created the problem)	The website allows the user to do this
	User grade (Voted grade by the community)	The concept was not implemented in the final design
	Date of creation	The website allows the user to do this
	Estimated date of removal	The website allows the user to do this

Figure 16: PDS comparison - website

Problems

When writing the PHP needed for these web pages, several problems arose.

Firstly, when a user input was being used as part of a query, it was found that the user could enter a query and delete data from the database. This was solved by using a function called preg_match which checked the user input against a list of symbols and rejected it if it contained any of them.

After this, it was also found that a non-admin user could access the 'Settings' page just by typing in the correct URL. This was solved by using a previously mentioned global variable $_SESSION to store the users login information. This was checked for validity on every page to ensure this URL bypass was not possible.

The final issue that arose was to do with the admin only features. For a regular user, the visuals that were supposed to be invisible to a non-admin user, were visible. They didn't perform the functions they were supposed to, but they were still visible. This was solved simply by including the HTML code inside conditional PHP statements.

Testing

Testing was carried out using a test database to check that the website was performing as required (appendix L).

It was found that the website fulfilled the PDS requirements. Further improvements are possible but for the purposes of a prototype, it functions sufficiently.

6.5 Development Costs

The budget for the total project including prototyping was £200. The final cost came to £204.54. The purchase of some development parts such as an ESP-12 breakout board and an FTDI cable could have been omitted but they were purchased to allow the code to be written faster as two people could work on them in parallel. The complete breakdown of everything purchased is in appendix D.

The breakdown costs of the wristband itself are in appendix E.

Excluding all the development costs, the total cost of the final wearable device is £66.83 which is not within reasonable limits as multiple wristbands would be required per centre. However, should these be produced in a batch size of 50 or larger the PCBs drop to approximately £2 per board instead of £25. The Li-Po battery would also decrease from £18 to £10. Extra savings can be made by methods such as removing the MFRC522 development board and replacing with the MFRC522 chip with a separate antenna.

7 Evaluation

The Climber Timer aimed to solve the issue of the lack of an automated system for indoor climbing centres to log their customers' climbs. It successfully does this with the RFID tag system combined with the database and website. By having the RFID tags on each problem passive, it allows for easy installation on walls when new problems are set and doesn't require a huge infrastructure overhaul of the centre. The relatively inexpensive RFID tags and wristband allow for the Climber Timer to easily be scaled up for climbing centres of any size at a reasonable cost. The RFID wristbands are effective due to its long battery life and its size not hindering the user's climbing ability. Improvements can be made but on the whole the Climber Timer solves the initial problem.

8 Business Plan

The system will be marketed to climbing centres as a package. They will select how many wristbands and how many tags they need, and our company will install the system. How they use the system will be decided by the centre, but our company will recommend renting out the wristbands to users the same way they rent out shoes and chalk. This way they will make regular returns on the system. It will be possible to implement sale plans like a monthly rental system or a 'Buy 8 wristband hires for the price of 10' deal.

As a target audience, climbing centres in the UK is a small group. To ensure that the business can function in the long term, there will be a yearly software subscription fee. The cost of each product is shown below.

Item	***Unit Price (£)***
Wristband	20
Tag (per 50)	40
Yearly Software Subscription	100
Installation Fee	200

Figure 17: Product price breakdown

9 Future Development

9.1 Hardware

9.1.1 Power Consumption

The greatest challenge for the hardware is to increase the amount of time the wristband is usable for the climber. The simplest method would be to increase the battery capacity. This is inadvisable as it would increase the size of the wristband.

In addition to the light sleep command, the ESP-12 has a deep sleep command. This would draw current in the order of microamps when disconnected from the internet. The main issue with this is that external circuitry would be required to wake the ESP-12. In an ideal situation, the interrupt function of the MFRC522 reader would be used but this was not possible with the board available.

There is potential for transitioning to a different Wi-Fi enabled microcontroller such as the ESP32[20]. These have a lower power consumption and a faster processing speed.

Bluetooth could be used as an alternative to Wi-Fi as the power consumption is significantly lower. The nRF51822[21] is a low energy Bluetooth module that has an idle current consumption of 100µA with a peak of 10mA. The range is significantly lower than Wi-Fi so this may not be a viable option.

9.1.2 Size & Enclosure

The main component limiting the size of the system is the MFRC522 development board. The board was initially chosen to reduce the prototyping time due to its pre-assembled nature. After working with the board it would now be possible to alter the PCB design to include the relevant components. This would allow the antenna to be constructed off-board, potentially embedded in the enclosure.

Other simpler methods to reduce the size of the device include:

- Optimise the PCB design.
- Use thinner PCB material
- Choose smaller components
- Implement a removable programming header

If the product were taken to be mass manufactured it would be cost effective to produce a mould for injection moulding and have the enclosure be constructed from ABS as recommended by the British Plastics Federation[22]. ABS impact resistant so the walls of the enclosure could be made thinner. Another material option would be a flexible rubber. This would increase the comfort of the user as it would flex to fit their wrist as well as compress on impact to reduce shocks to the internals.

9.1.3 Timing Improvements

Adding an external Real Time Clock chip would decrease the timing drift as this would not be affected by the ESP-12 sleep function. The DS1337 chip[23] is an I^{2}C controlled Real Time Clock which completely manages all timekeeping functions. It can count seconds, minutes, hours and can keep track of the Day, Date, Month, and Year with Leap-Year Compensation valid up to 2100.

It has surface-mount package with an Integrated Crystal (DC1337C). This will save space on the future PCB. The chip can function over a temperature range of -40°C to 85°C meaning that it will work in the climbing environment.

9.2 Software

9.2.1 Website Security

While some security measures were implemented in this project, they would not be adequate for the Climber Timer to be used commercially. Currently the passwords of the users are stored in cleartext within then database, which is very vulnerable should a hacker gain access to it. This can be solved by using a hash function on the passwords, so the original passwords are not recoverable from the result stored. This method still has vulnerabilities as common passwords could be identified as they would have the same hash result. For good password security, the password should be passed through the hash function in combination with the username. Due to the uniqueness of usernames, common passwords could not be identified solely through the hash result as the result returned would now be different.

9.2.2 Website Aesthetics

While the current state of the website is functional, it is not visually appealing. To improve this, CSS files can be used in conjunction with the PHP files already developed. An example of a potential CSS-styled login page is shown below.

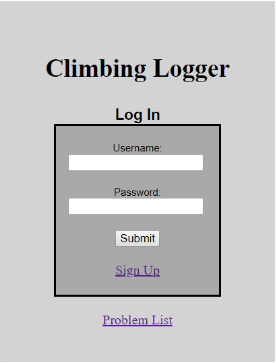


Figure 18: Current login page

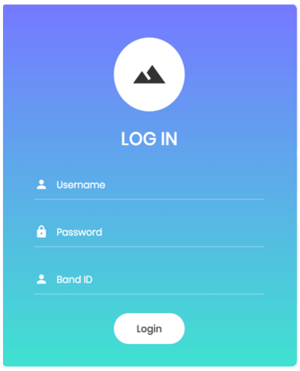


Figure 19: CSS login page

There is a clear visual improvement in the website when CSS is used, and this can enable the website to be more user-friendly with an intuitive user interface.

10 References

[1] ReportsnReports. *Sports Coaching Platform Technology Market Worth $864M by 2021.* URL: https://www.prnewswire.com/news-releases/sports-coaching-platform-technology-market-worth-864m-by-2021-561558701.html (visited on 15/03/2018).

[2] Climbax. *Climbax home page.* URL: http://www.climbax.co.uk/ (visited on 17/12/2017).

[3] Indiegogo. *Whipper Indiegogo campaign page.* URL: https://www.indiegogo.com/projects/whipper-world-s-1st-climbing-performance-tracker-sports-fitness (visited on 17/12/2017).

[4] NXP. *MFRC522 datasheet.* URL: www.nxp.com/docs/en/data-sheet/MFRC522.pdf (visited on 07/03/2018).

[5] AI-Thinker. *ESP-12 datasheet.* URL: http://www.kloppenborg.net/images/blog/esp8266/esp8266-esp12e-specs.pdf (visited on 13/02/2018).

[6] Arduino. *UDP Library.* URL: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiUdp.cpp (visited on 20/02/2018).

[7] Arduino. *UDP Examples.* URL: http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-class.html (visited on 20/02/2018).

[8] Testato. *NTP Library.* URL: https://github.com/arduino-libraries/NTPClient (visited on 10/03/2018).

[9] NTP. *What is NTP?* URL: http://www.ntp.org/ntpfaq/NTP-s-def.htm (visited on 13/03/2018).

[10] Dubilier. *LP-402933-1S-3 datasheet.* URL: http://www.farnell.com/datasheets/1666647.pdf?_ga=2.80958292.219595593.1521824954-874997243.1521824954 (visited on 15/02/2018).

[11] Microchip. *MCP73833 datasheet.* URL: http://ww1.microchip.com/downloads/en/DeviceDoc/22005a.pdf (visited on 21/02/2018).

[12] Texas Instruments. *LM3617 datasheet.* URL: http://www.ti.com/lit/ds/symlink/lm3671.pdf (visited on 22/02/2018).

[13] Adafruit Industries. *Adafruit Huzzah ESP8266 breakout PCB schematics.* URL: https://github.com/adafruit/Adafruit-Huzzah-ESP8266-Basic-Breakout-PCB (visited on 02/03/2018).

[14] Eurocircuits. *Eurocircuits PCB design rules.* URL: https://www.eurocircuits.com/pcb-design-guidelines/ (visited on 10/03/2018).

[15] Apple Inc. *Apple Watch.* URL: https://www.apple.com/uk/watch/ (visited on 07/03/2018).

[16] Fitbit. *Fitbit.* URL: https://www.fitbit.com/uk/home (visited on 07/03/2018).

[17] Valve. *Enclosure font.* URL: https://developer.valvesoftware.com/wiki/Custom_Testchamber_Signs (visited on 08/03/2018).

[18] TI forum users Texas Instruments. *TI E2E forum.* URL: https://e2e.ti.com/support/power_management/non-isolated_dcdc/f/196/p/673583/2480113 (visited on 22/03/2018).

[19] MariaDB. *MariaDB Training & Tutorials.* URL: https://mariadb.com/kb/en/library/training-tutorials/ (visited on 28/02/2018).

[20] Espressif. *ESP32.* URL: https://www.espressif.com/en/products/hardware/esp32/overview (visited on 15/03/2018).

[21] Nordic Semiconductor. *Low Energy Bluetooth Module.* URL: https://www.mouser.co.uk/new/nordicsemiconductor/nRF51822-multiprotocol-SoC/ (visited on 15/03/2018).

[22] British Plastics. *Injection moulding.*
URL: http://www.bpf.co.uk/plastipedia/processes/injection_moulding.aspx (visited on 12/03/2018).

[23] Maxim Instruments. *External Real Time Clock.*
URL: https://www.maximintegrated.com/en/products/digital/real-time-clocks/DS1337.html (visited on 13/03/2018).

[24] Amazon. *Wristband purchase link.* URL: https://www.amazon.co.uk/dp/B00B23NIOY/ref=cm_sw_r_cp_apap_oy5wwYUd8rGzu (visited on 23/02/2018).

Appendices

A Glossary

- **Bouldering** - short climbing above crash pads without ropes. This project will only deal with bouldering in an indoor environment.

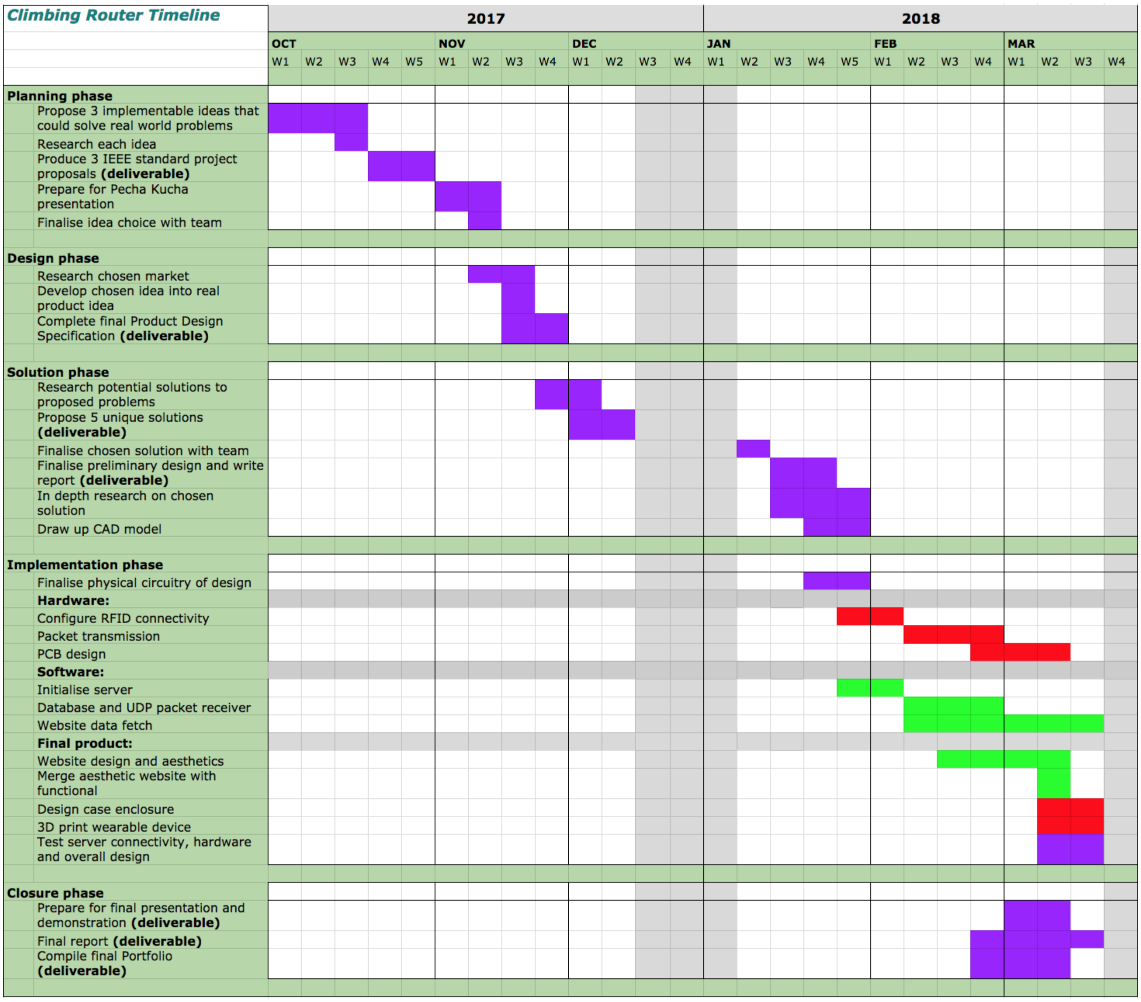
- **Problem** - a specific route to the top of a bouldering wall normally indicated by a specific colour of hold.

- **Setter** - a person who creates problems within a climbing centre, by drilling holds to the wall.

- **Set Date** - the date on which the problem was put on the wall.

- **Strip Date** - the date on which the problem was removed from the wall.

B Gantt Chart



Climbing Router Timeline	2017													2018												
	OCT					NOV				DEC				JAN					FEB				MAR			
	W1	W2	W3	W4	W5	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W5	W1	W2	W3	W4	W1	W2	W3	W4
Planning phase																										
Propose 3 implementable ideas that could solve real world problems	🟪	🟪	🟪																							
Research each idea			🟪																							
Produce 3 IEEE standard project proposals **(deliverable)**				🟪	🟪																					
Prepare for Pecha Kucha presentation						🟪	🟪																			
Finalise idea choice with team							🟪																			
Design phase																										
Research chosen market							🟪	🟪																		
Develop chosen idea into real product idea								🟪																		
Complete final Product Design Specification **(deliverable)**								🟪	🟪																	
Solution phase																										
Research potential solutions to proposed problems									🟪	🟪																
Propose 5 unique solutions **(deliverable)**										🟪	🟪															
Finalise chosen solution with team															🟪											
Finalise preliminary design and write report **(deliverable)**																🟪	🟪									
In depth research on chosen solution																🟪	🟪	🟪								
Draw up CAD model																	🟪	🟪								
Implementation phase																										
Finalise physical circuitry of design																	🟪	🟪								
Hardware:																										
Configure RFID connectivity																		🟥	🟥							
Packet transmission																				🟥	🟥	🟥				
PCB design																						🟥	🟥	🟥		
Software:																										
Initialise server																		🟩	🟩							
Database and UDP packet receiver																				🟩	🟩	🟩				
Website data fetch																				🟩	🟩	🟩	🟩	🟩	🟩	
Final product:																										
Website design and aesthetics																					🟩	🟩	🟩	🟩		
Merge aesthetic website with functional																								🟩		
Design case enclosure																								🟥	🟥	
3D print wearable device																								🟥	🟥	
Test server connectivity, hardware and overall design																								🟪	🟪	
Closure phase																										
Prepare for final presentation and demonstration **(deliverable)**																							🟪	🟪		
Final report **(deliverable)**																						🟪	🟪	🟪	🟪	
Compile final Portfolio **(deliverable)**																						🟪	🟪	🟪		

	Work distribution key
🟪	Whole group
🟥	Hardware subgroup
🟩	Software subgroup

Figure 20: Project Gantt chart

C Meeting Minutes

C.1 Meeting: 13/10/2017

Time 13:11 - 13:52

Present: AS, TP, AP, TB, BB, BO

- Climbing Wall Router: - Alex & Bonne
  - Displays video
  - Take picture of wall
  - App
  - Mainly software
  - Maybe ranking system
  - Logs time
  - Drill tag to stay by hold
  - THROUGH TO PROPOSAL ROUND
- Bike:
  - Automatic gearbox
  - DENIED
  - Auto-balancing bike
  - Beyond technical ability
  - DENIED
- Fingerprint Lock:
  - Maybe buy a fingerprint module
  - Resulting in mainly software
  - Not rewarding challenge
  - DENIED

C.2 Meeting: 20/10/2017

Time 13:15 - 13:59

Present: AS, TP, AP, TB, BB, BO

- Dyslexia Pen: - Ben & Tomasz
  - Mount camera to highlighter sized device
  - Heavily biased toward software
  - THROUGH TO PROPOSAL ROUND
- OAPill Dispenser:
  - too derivative
  - DENIED
- Auto-Jar:

  - Over engineering, solves no problem
  - DENIED
- Emergency Mesh Network:
  - emergency button for people in disaster areas
  - backup network
  - NEED TO CHECK IF NEEDED: Tomasz
  - NEEDS GOVERNMENT PERMISSION
  - DENIED
- Guitar auto-tuner: - Archit & Tom
  - for one type of guitar (acoustic)
  - maybe for violin or ukelele
  - THROUGH TO PROPOSAL ROUND

C.3 Meeting: 03/11/2017

Time 12:10 - 12:30

Present: AS, TP, AP, TB, BB, BO

- Contacted climbing centre to gauge viability of router system
  - Operations manager said it was a good idea
- Spoke to Learning Support assistant about viability of dyslexia pen
  - Current products are £200+
  - If it can be made cheaper it is viable
- Spoke to a number of guitar owners
  - Potentially viable if it doesn't damage guitar
  - Cheap
  - Works fast as tuning can be quite quick

C.4 Meeting: 10/11/2017

Time 12:15 - 12:43

Present: AS, TP, AP, TB, TE, BB, BO

- Make slides for Pecha Kucha
- Prepare for presentation
- Work divided between subgroup
  - Router - Alex & Bonne
  - Dyslexia pen - Tomasz & Ben
  - Guitar tuner - Archit & Tom

C.5 Meeting: 24/11/2017

Time 12:15 - 12:52

Present: AS, TP, AP, TB, TE, BB, BO

- Final vote after presentation to hackspace
  - Climbing router chosen
- Task for all members is to research potential methods of creating router
- Alex will return to climbing centre to get more info about requirements

C.6 Meeting: 01/12/2017

Time 12:15 - 12:52

Present: AS, TP, AP, TB, TE, BB, BO

- 3 out of 5 solutions decided
  - Active wrist, passive wall
  - Active wall, passive wrist
  - Wearable smart tag
- Active wrist, passive wall - Tom and Alex
  - Can be expanded to all holds easily, not just start and end
  - Means don't have to worry about battery life
  - RFID
  - Centre would hire out and charge wristbands
  - Would have to pay attention to detection range, especially if applying to all holds as they can be close together huge
  - Wristband may end up being very chunky
  - End hold might need more thought due to positioning the hand holds
  - Screw tags into wall
- Active wall, passive wrist
  - Would require far too many wristbands
  - Wristbands can be tiny and flexible
  - Battery life of wall would have to be long to match when problems are stripped
  - Screw tags into wall
  - Cheap
  - More idiot proof/durable
  - Costs under £5
- Wearable smart tag - Bonne
  - https://www.indiegogo.com/projects/whipper-world-s-1st-climbing-performance-tracker-sports-fitness#/
  - Works for route cimbing

  - May have issues with bouldering, doesn't know how you finish
  - Has various sensors to do useful stuff (maybe too many sensors required for accurate tracking?)

C.7 Meeting: 08/12/2017

Time 12:35 - 13:30

Present: AS, TP, AP, TB, TE, BB, BO

- Final 2 solutions
  - Cameras/kinect
  - Smart holds
- Cameras - Tomasz
  - Have a QR code on your back
  - Body contortions may mess things up
  - Need an identifying marker on the person somehow
  - Technical software, expense
  - Identifying when a person succeeds and fails could be an issue
  - Use a kinect?
- Smart Holds - Ben
  - Pressure sensitive holds
  - Would have to replace ALL holds in a centre (expense)
  - Finger print scanning
  - Holds only solution
- Gantt Chart - Tamara
  - Gantt chart
  - Next steps
  - Miscellaneous write up
- Design Criteria - Archit
  - Performance
  - Life in service
  - Environment
  - Safety
  - Ergonomics
  - Installation
  - Target Product Cost
  - Size
  - Weight
- Deadline: Wed 13th 6pm

C.8 Meeting: 19/01/2018

Time 12:35 - 13:30

Present: AS, AP, TB, BB, BO

- Bonne is made 2nd editor
- Need to meet with supervisor
- A decision needs to be made about the final design
- Find out about website that needs to exist
- Decided on passive tag, active wrist
- Need to decide on communication methods (Bluetooth, Wi-Fi etc.)
- NFC tags needed
- Decided on Wi-Fi
- Not a custom PCB
- Deadline: 02/02/18 5pm - designs and component lists

C.9 Meeting: 26/01/2018

Time 12:28 - 13:20

Present: AS, TP, AP, TB, TE, BB, BO

- Division into subgroups
- Database - Tamara (might become webmaster), Alex, Bonne, Archit
  - App/website (app probably not necessary)
  - Data lists
  - Server (necessary for this project)
- Hardware - Tomasz, Tom, Ben
  - Send identifier from tag to band and timestamp, unique identifier for band and tag for from band to server
  - Will require power
  - NFC
  - Wi-Fi modules
  - Clock (from processor)
  - Set of passive tags for wristband

C.10 Meeting: 09/02/2018

Time 11:18 - 11:44

Present: AS, TP, AP, TB, TE, BB, BO

- Hardware
  - Initial test board working
- Software

  - Look into wordpress
  - Check out jangle
  - Listen to ports/socket for information (write a program)
  - Python, MariaDB, API
  - Set up databases - consider storage

C.11 Meeting: 23/02/2018

Time 12:15 - 12:52

Present: TP, AP, TB, BO

- Hardware
  - Haptic feedback from device?
  - Fix deep sleep issue with fixed interrupt modes on MFRC522
  - Begin to design enclosure
- Software
  - System concept complete
  - Implement UDP receiver

C.12 Meeting: 09/03/2018

Time 12:30 - 13:05

Present: AS, TP, AP, TB, TE, BB, BO

- Begin preparing for final presentation
- Points to be covered in presentation
  - Problem overview
  - Overview of Wi-Fi, RFID, ESP
  - Hardware subgroup overview and problems
  - Sleep, UDP, NTP, PCB highlights, CAD model industrial design highlight
  - Business Plan
- Hardware
  - Where is our PCB?
- Software
  - Improve website aesthetics - Tamara

C.13 Meeting: 14/03/2018

Time 11:43 - 13:05

Present: AS, TP, AP, TB, TE, BB, BO

- Run through for final presentation

- Set up tags and test overall server connectivity for demo
- Further hardware testing - make sure timing issue is solved
- Connect aesthetic website to server and pull data (merge PHP codes)
- Final report delegation

D Project Development Costs

Part Name	***Part No.***	***Supplier***	***Unit Price***	***Quantity***	***Total Price***
4.7μF 6.3V X5R capacitor	GRM188R60J475KE19D	Farnell	£0.123	20	£2.40
4.7μF 10V X5R capacitor	GRM188R61A475KE15D	Farnell	£0.127	10	£1.27
1μF X5R capacitor	GRM188R60J105KA01D	Farnell	£0.0787	1	£0.79
1.2V signal diode	1N4148WS-E3-08	Farnell	£0.138	5	£0.69
8 way SIL 2.54mm header	2211S-08G	Farnell	£0.04	5	£0.20
3 way 90° 2.54mm header	MC34747	Farnell	£0.024	5	£0.12
2.2μ H Inductor 1.9A	DFE201610E-2R2M=P2	Farnell	£0.194	5	£0.97
Green LED 2.4V	LGQ396-PS-35	Farnell	£0.191	5	£0.96
Blue LED 2.9V	LNJ926W8CRA	Farnell	£0.378	5	£1.89
10kΩ resistor	MCWR06X1002FTL	Farnell	£0.0031	50	£0.16
1kΩ resistor	MCWR06X1001FTL	Farnell	£0.0031	10	£0.03
4.7kΩ resistor	MCWR06X4701FTL	Farnell	£0.0003	10	£0.03
0Ω resistor	MC0603SAF0000T5E	Farnell	£0.0073	20	£0.15
Li-Po charging controller	MCP73833-FCI/UN	Farnell	£0.719	5	£3.60
90° Li-Po connector	53048-0310	Farnell	£0.206	5	£1.03
300mAh Li-Po Battery	LP-402933-IS-3	Farnell	£18.45	1	£18.45
SMD tactile switch	SKSCLDE010	Mouser	£0.428	10	£4.28
Micro USB connector	ZX62-AB-5PA(31)	Mouser	£0.561	5	£2.81
3v3 SMPS regulator	LM3671MF-3.3/NOPB	Mouser	£1.03	5	£5.15
Piezo buzzer	AT-2310-T-LW100-R	Mouser	£1.80	3	£5.40
HUZZAH ESP-12 breakout	ADAFRUIT 2471	Rapid	£10.21	2	£20.41
ESP-12 SMD	ADAFRUIT 2471	Rapid	£7.25	2	£14.50
FTDI Cable	USBSERIALTTL	Rapid	£14.28	2	£28.56
3 off custom PCB	Not Applicable	PCB Train	£75.50	1	£75.50
RFID reader/writer	MFRC522	eBay	£3.85	2	£7.70
Wristband	Not Applicable	Amazon	£7.49	1	£7.49
Total					£204.54

E Prototype Costs

Part Name	*Type*	*Unit Price*	*Quantity per unit*	*Total Price*
4.7μF 6.3V X5R capacitor	PCB component	£0.123	3	£0.37
4.7μF 10V X5R capacitor	PCB component	£0.127	1	£0.13
1μF X5R capacitor	PCB component	£0.0787	1	£0.08
1.2V signal diode	PCB component	£0.138	1	£0.14
8 way SIL 2.54mm header	PCB component	£0.04	1	£0.04
2.2μ H Inductor 1.9A	PCB component	£0.194	1	£0.19
Green LED 2.4V	PCB component	£0.191	1	£0.19
10kΩ resistor	PCB component	£0.0031	9	£0.03
1kΩ resistor	PCB component	£0.0031	1	£0.01
4.7kΩ resistor	PCB component	£0.0003	1	£0.00
0Ω resistor	PCB component	£0.0073	2	£0.01
Li-Po charging controller	PCB component	£0.719	1	£0.72
90° Li-Po connector	PCB component	£0.206	1	£0.21
300mAh Li-Po Battery	Battery	£18.45	1	£18.45
SMD tactile switch	PCB component	£0.428	2	£0.86
Micro USB connector	PCB component	£0.561	1	£0.56
3v3 SMPS regulator	PCB component	£1.03	1	£1.03
Piezo buzzer	PCB component	£1.80	1	£1.80
ESP-12 SMD	External Module	£7.25	1	£7.25
Custom PCB	PCB	£25.17	1	£25.17
RFID reader/writer	External Module	£2.12	1	£2.12
Wristband	Wristband	£7.49[24]	1	£7.49
				£66.83

F Hardware Code Resources

F.1 ESP-12 Code

```
#include <SPI.h>
#include <MFRC522.h>
#include <Wi-FiUdp.h>
#include <ESP-12Wi-Fi.h>
#include <NTPClient.h>

extern "C" {
#include "gpio.h"
#include "ets_sys.h"
#include "user_interface.h"
}

#define RST_PIN 2
#define SS_PIN 15
#define IRQ_PIN 4

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
MFRC522::MIFARE_Key key;

// Initialise array that will store new NUID
byte nuidPICC[4];

//Network name and password
const char ssid[32] = "Climber_Network";
const char password[64] = "iamthesenate66";

struct station_config stationConf;
Wi-FiUDP Udp;

//Delimiter used to split up message, ID of the wristband
const char  DELIMITER[] = "\r\n";
const char READER_ID[] = "01";

//The address of the NTP pool used to get the address of an NTP server
#define NTP_OFFSET 0
#define NTP_ADDRESS "europe.pool.ntp.org"

//initialises timing parts of the code
NTPClient timeClient(Udp, NTP_ADDRESS,NTP_OFFSET);

unsigned long currentmillis = 0;
unsigned long previousmillis = 0;
unsigned long millidifference = 0;
unsigned long currentepochtime = 0;
unsigned long previousepochtime = 0;

void setup() {
  //initialise outputs and esp8266 sleep type
  gpio_init();
  wifi_fpm_set_sleep_type(LIGHT_SLEEP_T);

  pinMode(0, OUTPUT);
  digitalWrite(0, HIGH);

  pinMode(5, OUTPUT);
  //set buzzer frequency
  analogWriteFreq(4000);

  Serial.begin(115200);
  //set up communication with the RFID reader board
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }

  Serial.println(F("Starting␣networking"));
  Serial.print(F("Connecting␣to␣SSID:␣"));
  Serial.println(ssid);
  //Connect to the Access Point
  wifi_set_opmode(STATION_MODE);
  stationConf.bssid_set = 0;
  os_memcpy(&stationConf.ssid, ssid, 32);
  os_memcpy(&stationConf.password, password, 64);
  wifi_station_set_config(&stationConf);
  wifi_station_connect();

  Serial.println(F("Waiting␣for␣network..."));
  while (wifi_station_get_connect_status() != STATION_GOT_IP)
  {
    Serial.print(wifi_station_get_connect_status());
    delay(500);
  }

  Serial.println();
  //update the onbaord real time clock to the correct UNIX time according to NTP
  timeClient.begin();
  timeClient.forceUpdate();
  delay(20);
  Serial.println(timeClient.getEpochTime());
  //disconnect for power saving
  Serial.println(F("Network␣setup␣OK,␣disconnecting"));
  wifi_station_disconnect();
  wifi_set_opmode(NULL_MODE);

  Serial.println(F("Setup␣complete"));
}

void loop() {
  currentmillis = millis();
  //get current unix timestamp from the RTC
  currentepochtime = timeClient.getEpochTime();

  if (currentepochtime > previousepochtime )
  {
    previousmillis = currentmillis;
  }

  previousepochtime = currentepochtime;
  //get the milli second difference for accuracy of 10ths of a second
  millidifference = ((currentmillis-previousmillis)+50)/100;

  //poll the RFID reader to check for card upon waking up, takes approx 30 ms
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial())
  {

    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);

    //beeps the buzzer
    analogWrite(5, 511);
    delay(500);
    analogWrite(5, 0);

    // Check is the PICC of Classic MIFARE type
    if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
        piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
        piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
      Serial.println(F("Your␣tag␣is␣not␣of␣type␣MIFARE␣Classic."));
      return;
    }

    if (rfid.uid.uidByte[0] != nuidPICC[0] ||
        rfid.uid.uidByte[1] != nuidPICC[1] ||
        rfid.uid.uidByte[2] != nuidPICC[2] ||
        rfid.uid.uidByte[3] != nuidPICC[3] )
    {
      Serial.println(F("A␣new␣card␣has␣been␣detected."));

      // Store NUID into nuidPICC array
      for (byte i = 0; i < 4; i++)
      {
        nuidPICC[i] = rfid.uid.uidByte[i];
      }

      Serial.print(F("Card␣NUID:␣"));
      printHex(rfid.uid.uidByte, rfid.uid.size);
      Serial.println();
    }
    else
    {
      Serial.println(F("Card␣read␣previously."));
    }

    Serial.print("Timestamp:␣");
    Serial.print(timeClient.getFormattedTime());
    Serial.print(".");
    Serial.println(millidifference);

    // Halt PICC
    rfid.PICC_HaltA();

    // Stop encryption on PCD
    rfid.PCD_StopCrypto1();

   Serial.println(F("Connecting␣to␣network␣for␣data␣upload..."));
    wifi_set_opmode(STATION_MODE);
    wifi_station_connect();

    Serial.println(F("Waiting␣for␣network..."));
    while (wifi_station_get_connect_status() != STATION_GOT_IP)
    {
      Serial.print(wifi_station_get_connect_status());
      delay(500);
      system_soft_wdt_feed(); // nice doggo
    }
    Serial.println();


    system_soft_wdt_feed(); // nice doggo

    //connects to server at correct UDP port
    Udp.beginPacket("192.168.50.1", 5000);
    Serial.println(F("Sending␣packet..."));
    Udp.print(DELIMITER);
    //sends the UID of the RFID tag
    for (byte i = 0; i < rfid.uid.size; i++)
    {
      Udp.print(rfid.uid.uidByte[i] < 0x10 ? "␣0" : "");
      Udp.print(rfid.uid.uidByte[i], HEX);
    }
    Udp.print(DELIMITER);
    Udp.print(READER_ID);
    Udp.print(DELIMITER);
    Udp.print(currentepochtime);
    Udp.print(DELIMITER);
    Udp.print(millidifference);
    delay(10);
    Udp.endPacket();

    system_soft_wdt_feed(); // nice doggo
    //packet is not always heard so copies of the packet are sent
    //The database can handle copies
    Udp.beginPacket("192.168.50.1", 5000);
    Serial.println(F("Sending␣packet..."));
    Udp.print(DELIMITER);
    for (byte i = 0; i < rfid.uid.size; i++)
    {
      Udp.print(rfid.uid.uidByte[i] < 0x10 ? "␣0" : ".");
      Udp.print(rfid.uid.uidByte[i], HEX);
    }
    Udp.print(DELIMITER);
    Udp.print(READER_ID);
    Udp.print(DELIMITER);
    Udp.print(currentepochtime);
    Udp.print(DELIMITER);
    Udp.print(millidifference);
    delay(10);
    Udp.endPacket();

    system_soft_wdt_feed(); // nice doggo
    Serial.println(F("Disconnecting␣network..."));
    //transmission complete, disconnecting for power saving
    wifi_station_disconnect();
    wifi_set_opmode(NULL_MODE);
  }
  //maximal power savings by shutting down the modem and the CPU
  light_sleep();
  delay(71); //needs to be 1 ms longer than the light_sleep time
}

//call back funciton required on wakeup
void wakeupnormal()
{
  wifi_fpm_close();

  Wi-Fi.forceSleepBegin();
  Serial.flush(); // CERTAIN CODE LINES MAY BE VITAL TO YOUR SUCCESS. DO NOT DESTROY VITAL CODE
  rfid.PCD_SoftPowerUp();
}

//engages light sleep using ESP specific API
void light_sleep()
{
  rfid.PCD_SoftPowerDown();
  wifi_station_disconnect();
  wifi_set_opmode(NULL_MODE);
  wifi_fpm_open();
  wifi_fpm_set_sleep_type(LIGHT_SLEEP_T);

  wifi_fpm_set_wakeup_cb(wakeupnormal);
  wifi_fpm_do_sleep(70*1000);
}

//Helper routine to dump a byte array as hex values to Serial.
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? "␣0" : "␣");
    Serial.print(buffer[i], HEX);
  }
}

//Helper routine to dump a byte array as dec values to Serial.
void printDec(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? "␣0" : "␣");
    Serial.print(buffer[i], DEC);
  }
}
```

F.2 ESP Timer Testing

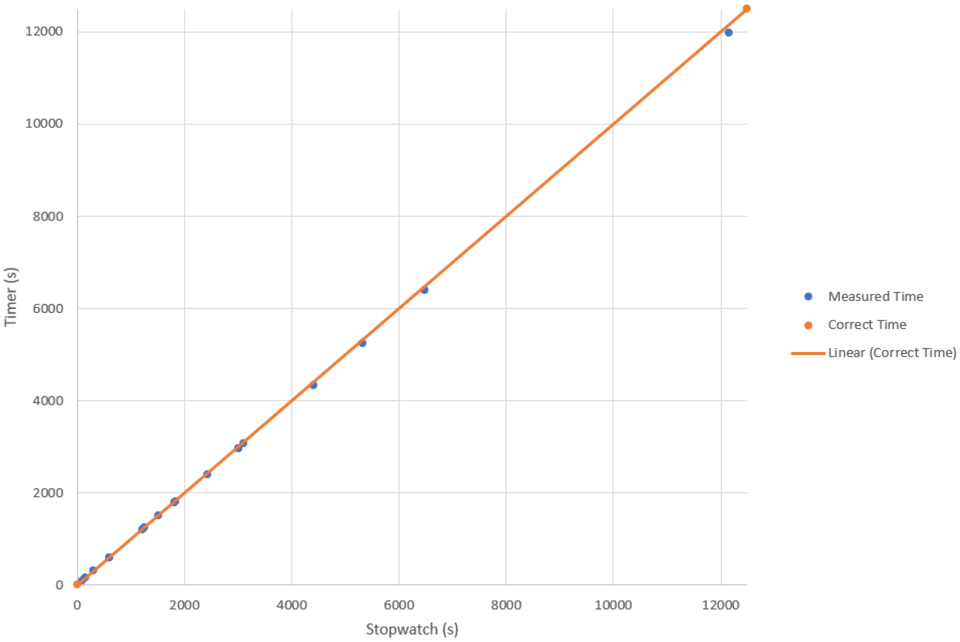


Figure 21: Comparison of Stopwatch time to ESP-12 Timer

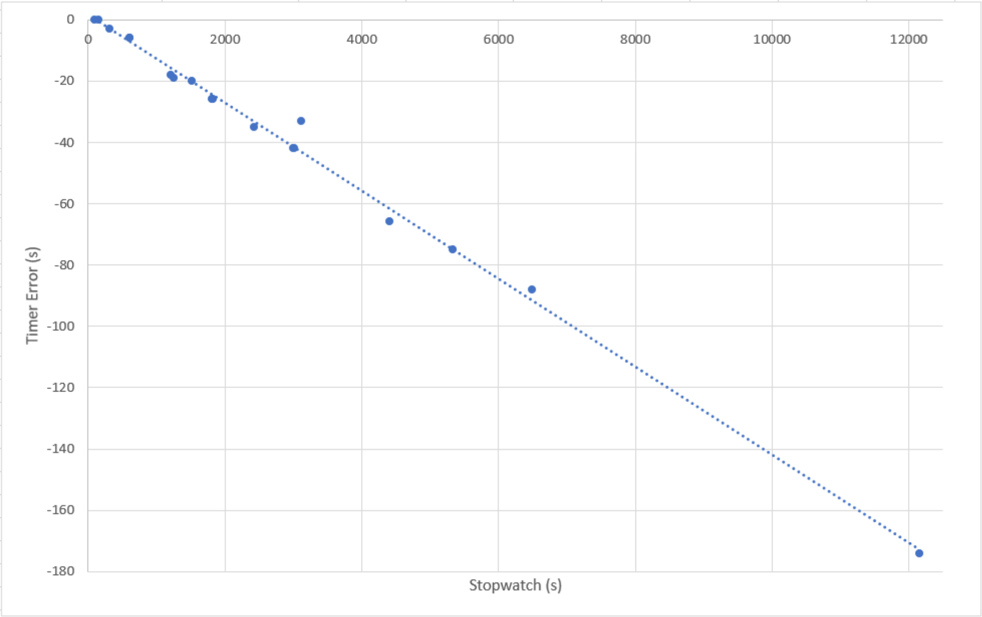


Figure 22: ESP-12 Timer Lag from Correct Time

F.3 Oscilloscope Power Consumption Data

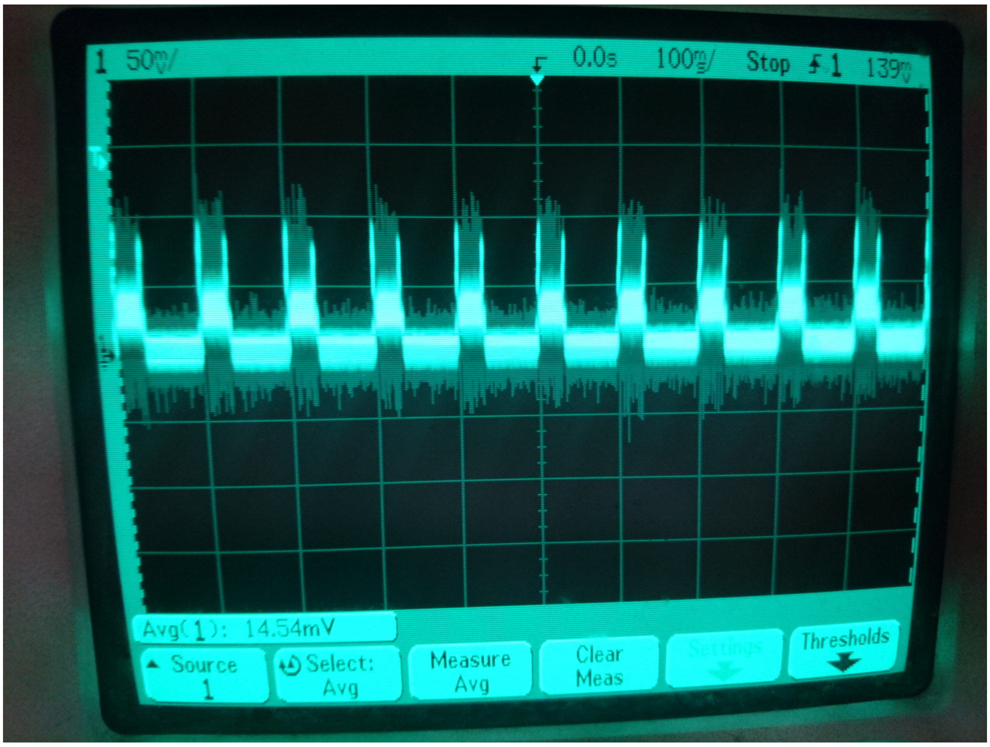


Figure 23: Idle system current draw

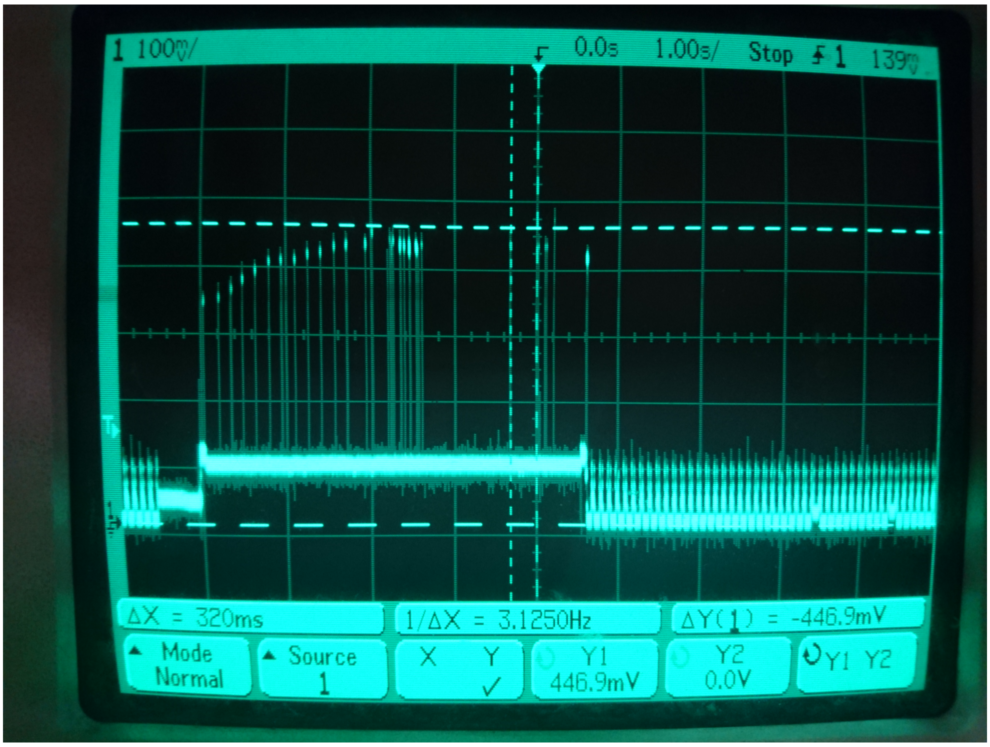
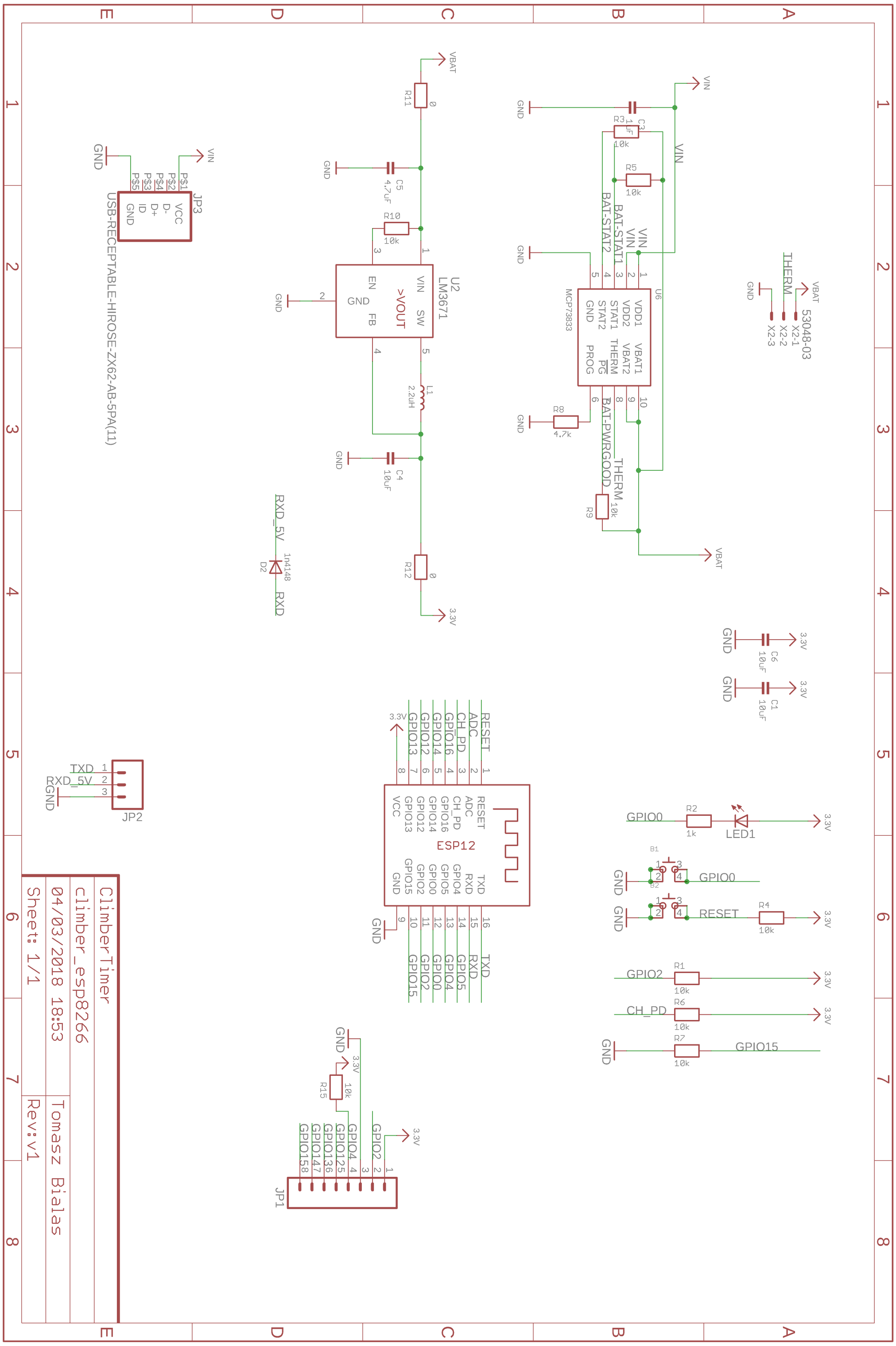


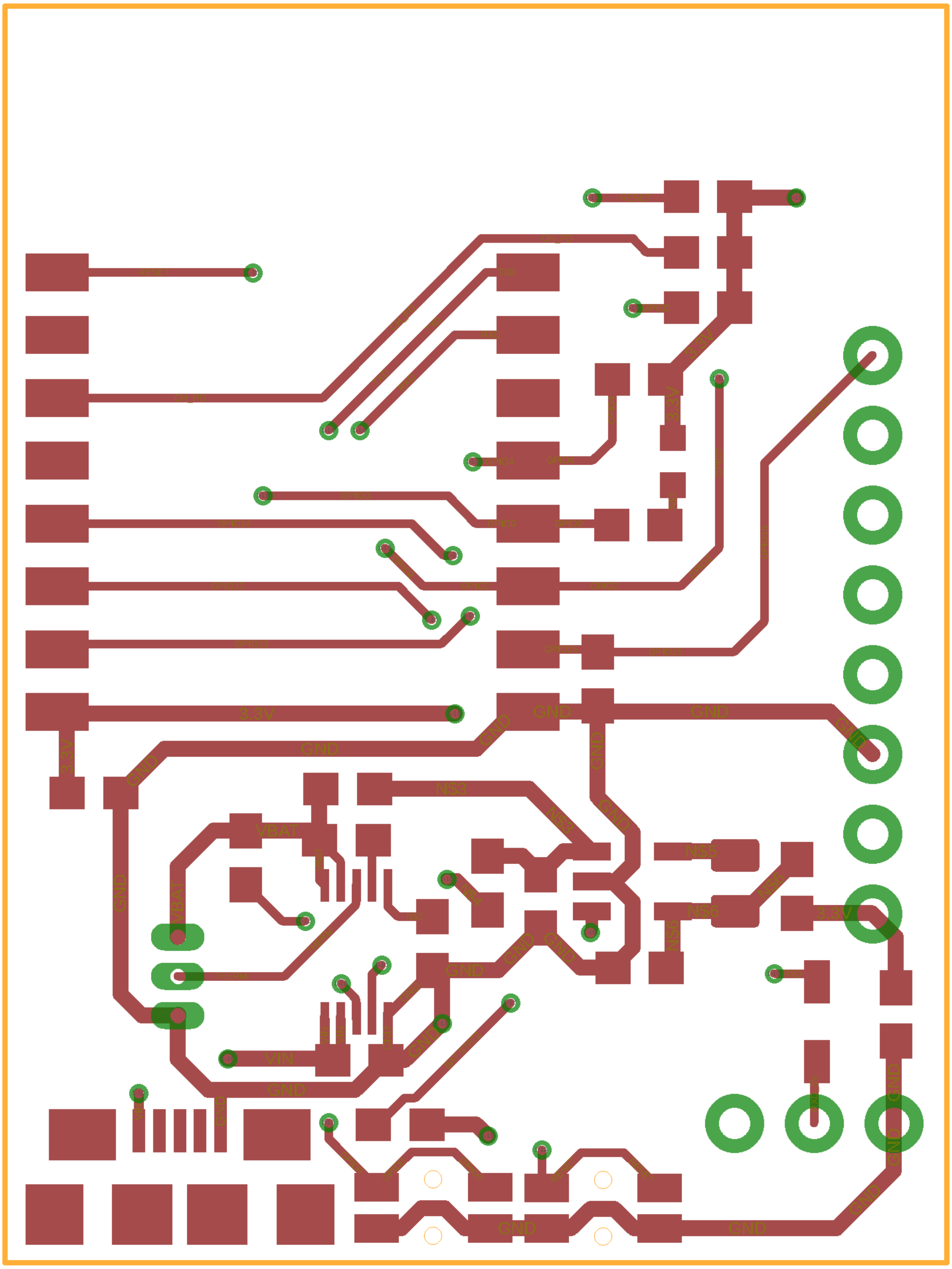
Figure 24: System current draw

G PCB Development

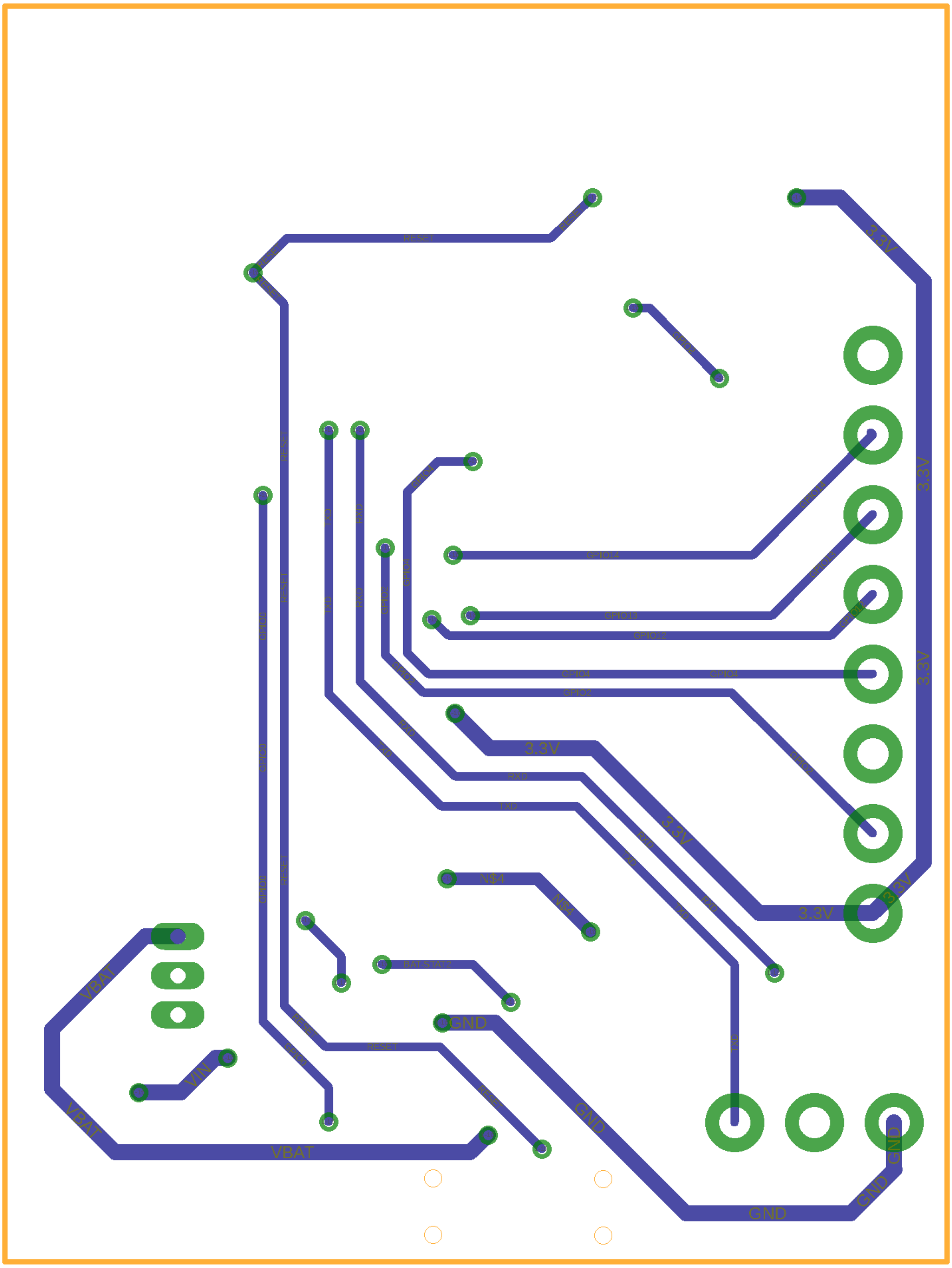
G.1 Schematic



G.2 Top Layer



G.3 Bottom Layer



G.4 Physical PCB

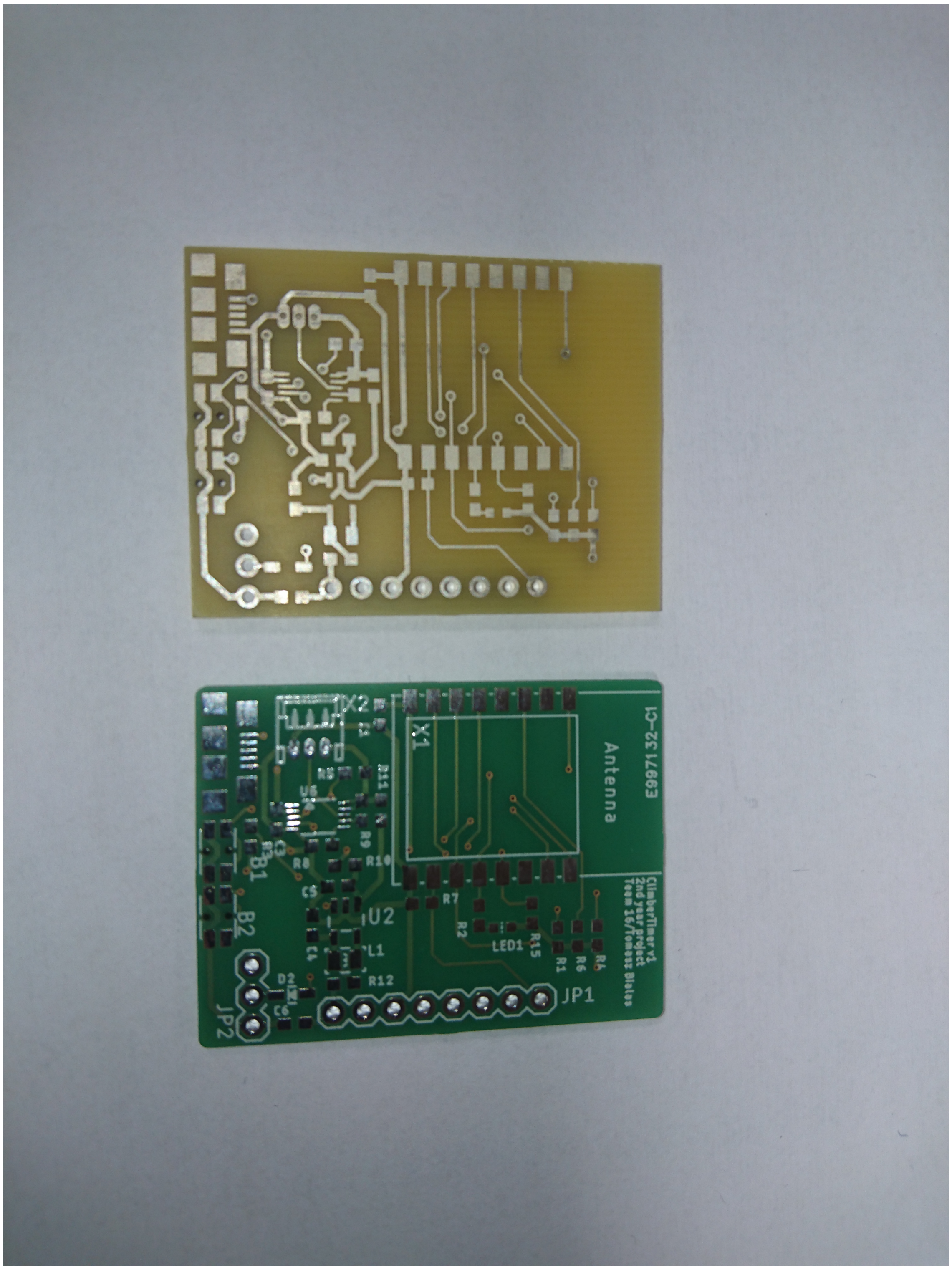
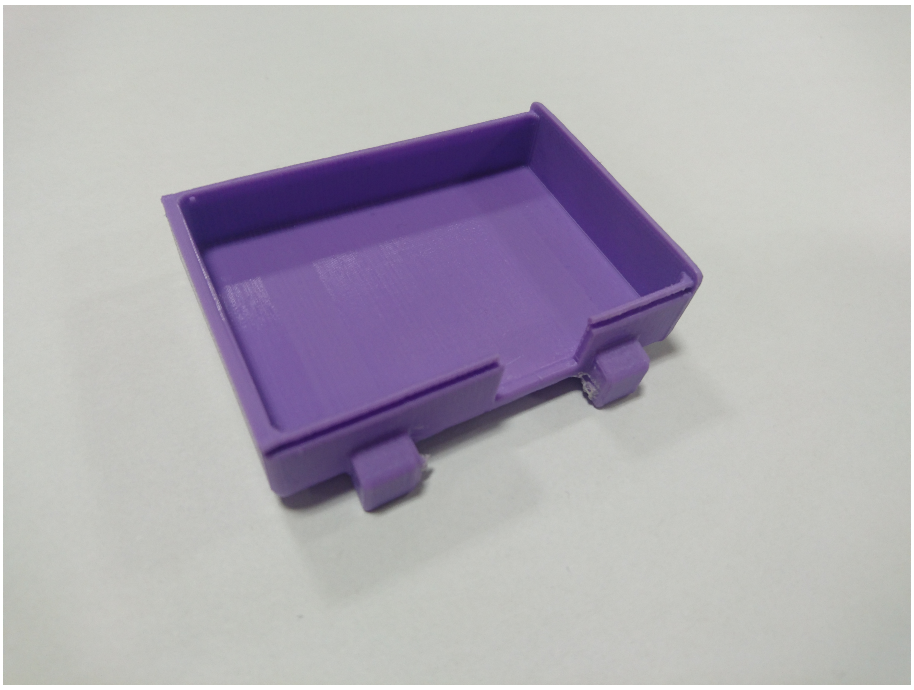
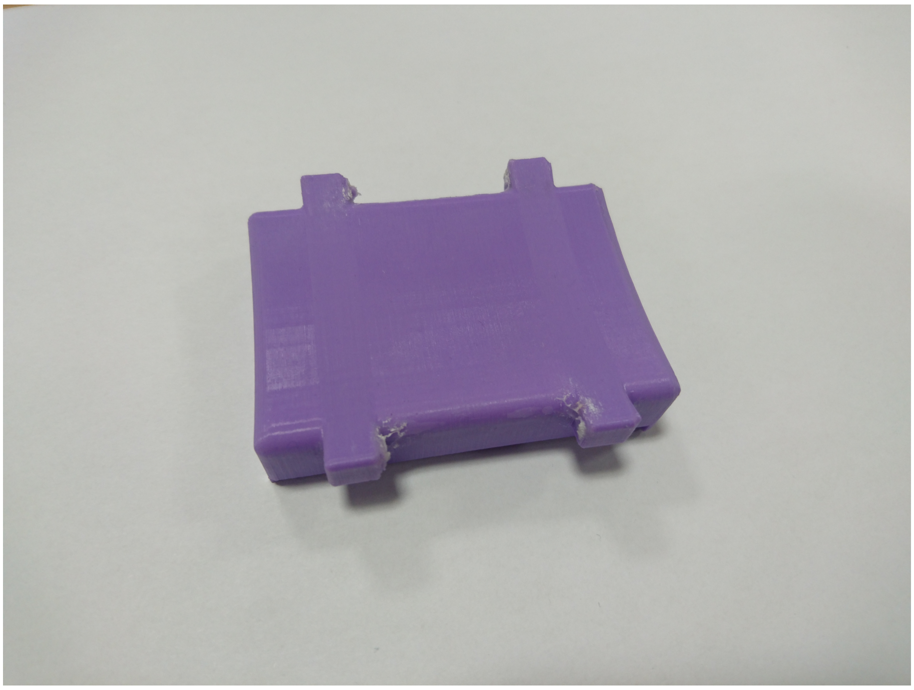


Figure 25: Top:PCB Train Rushed Job, Bottom: PCB Express Delayed Shipping

H Enclosure Development

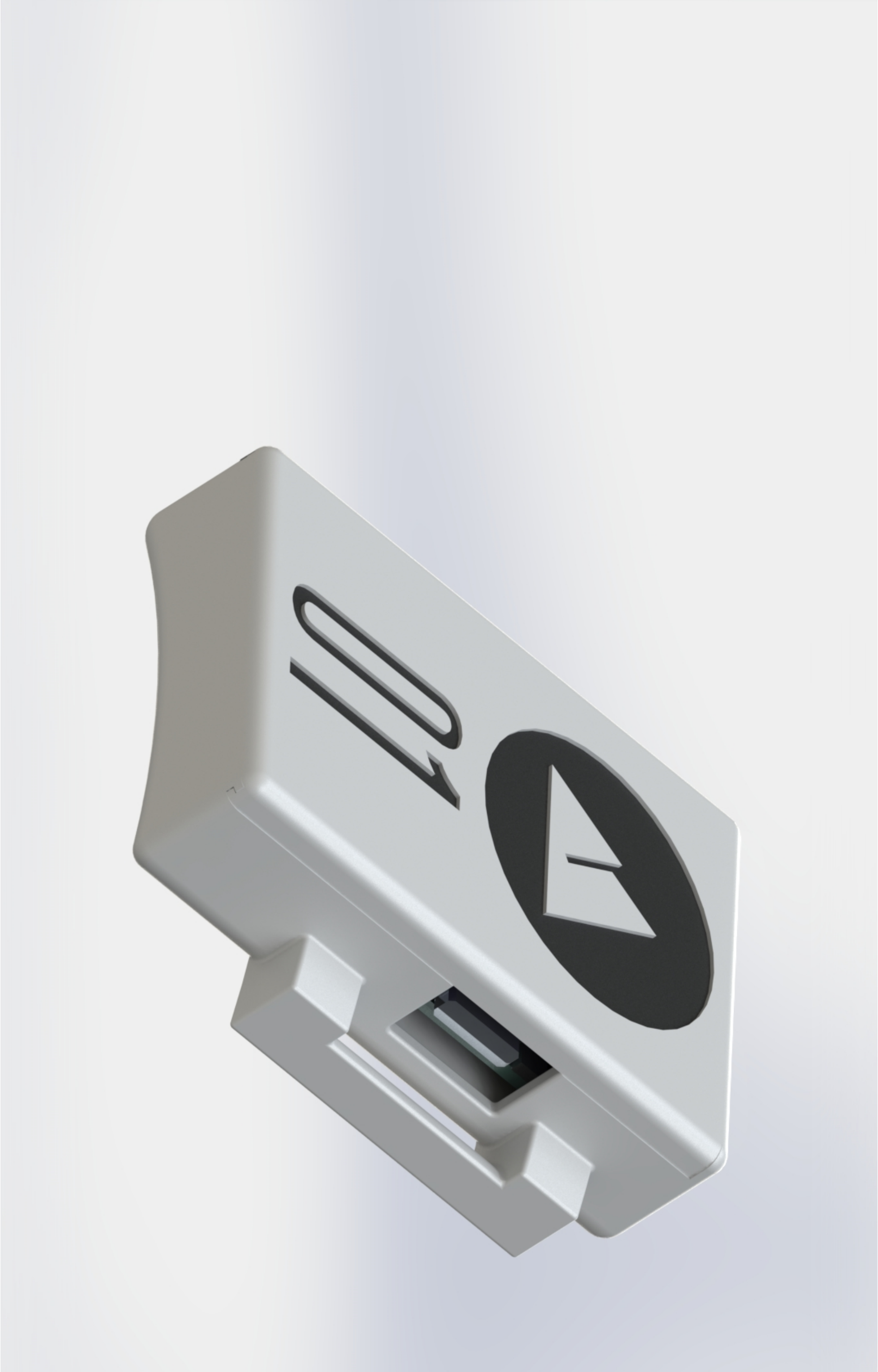
H.1 1st Iteration



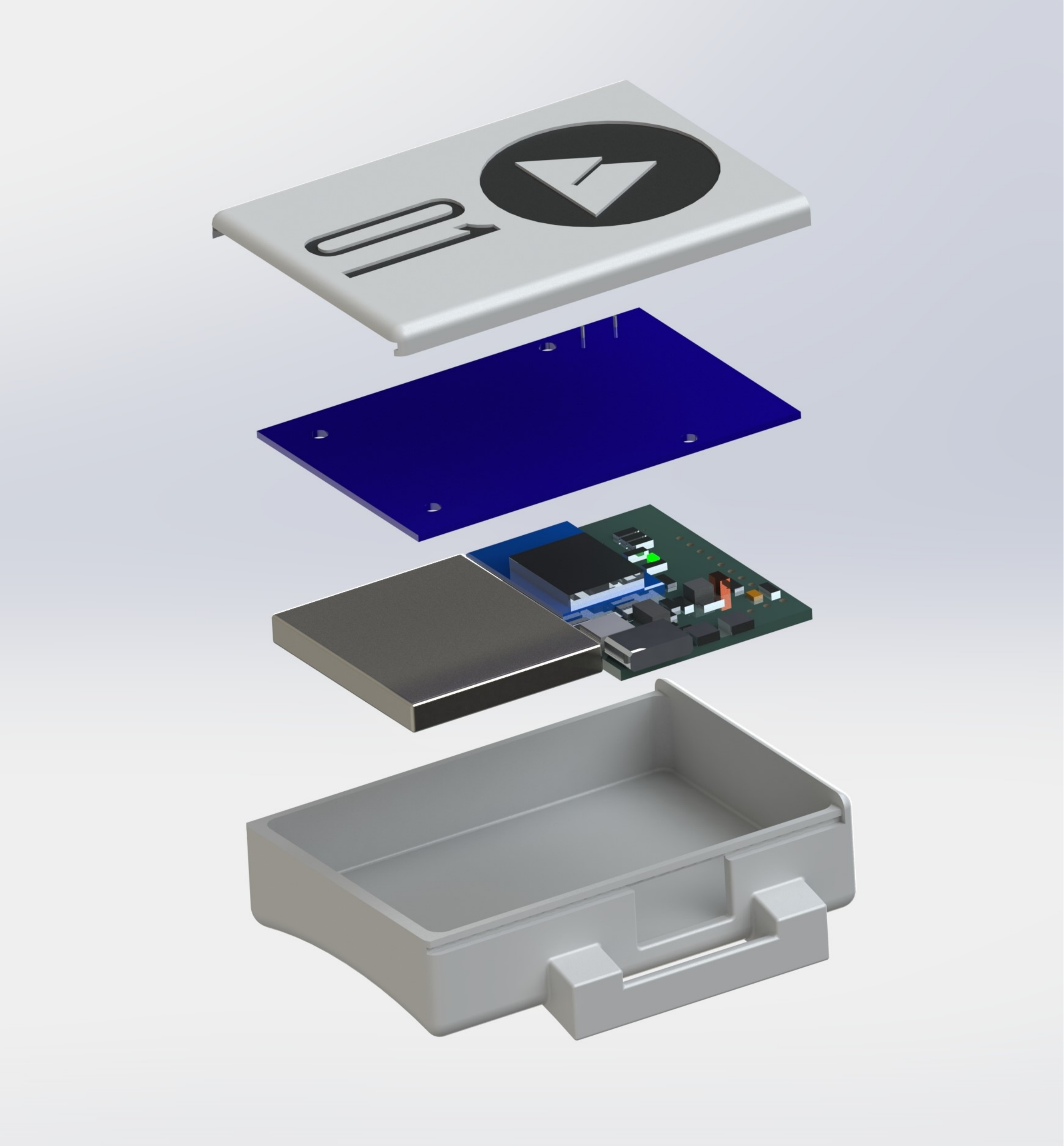


H.2 2nd Iteration

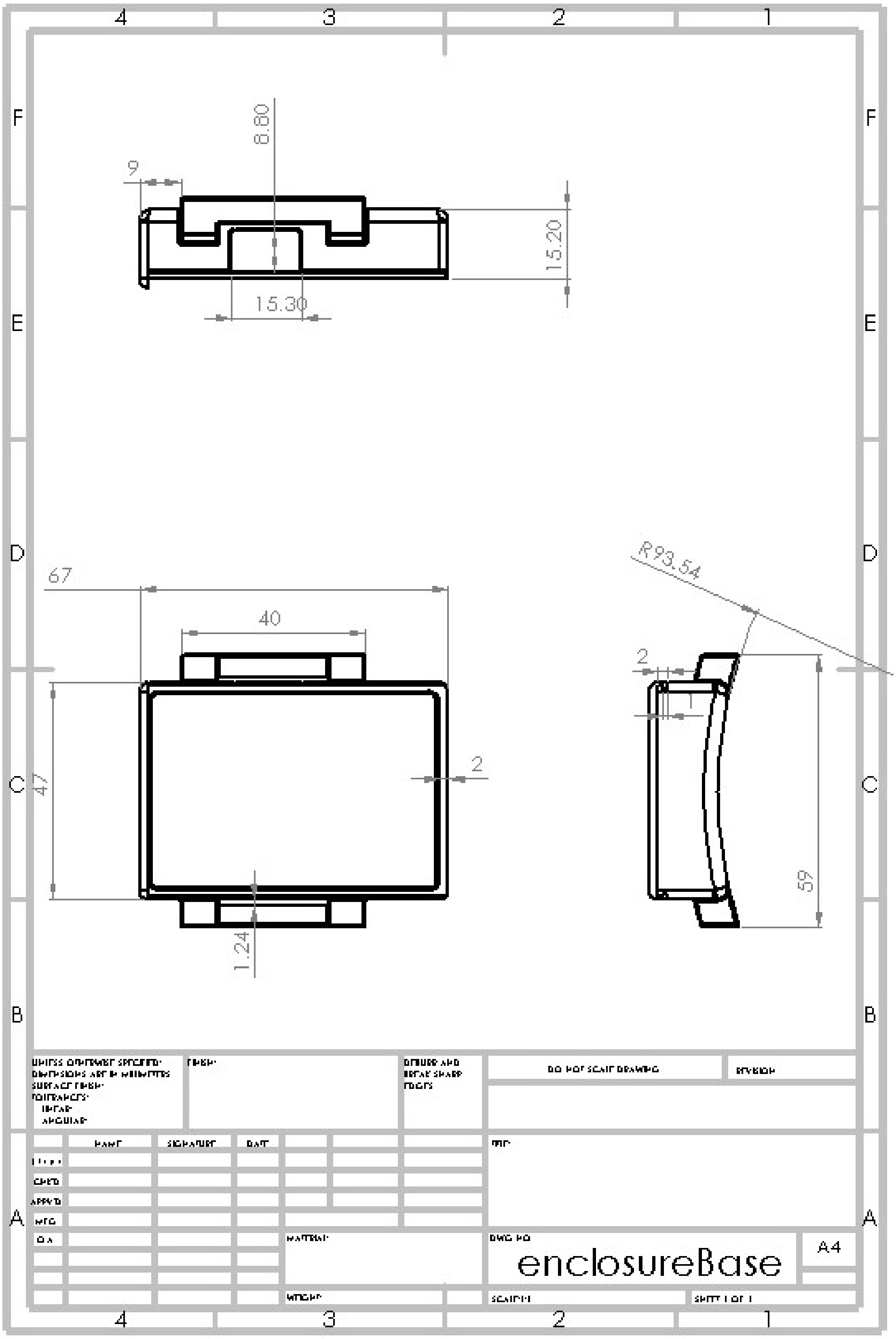
H.3 Closed Render



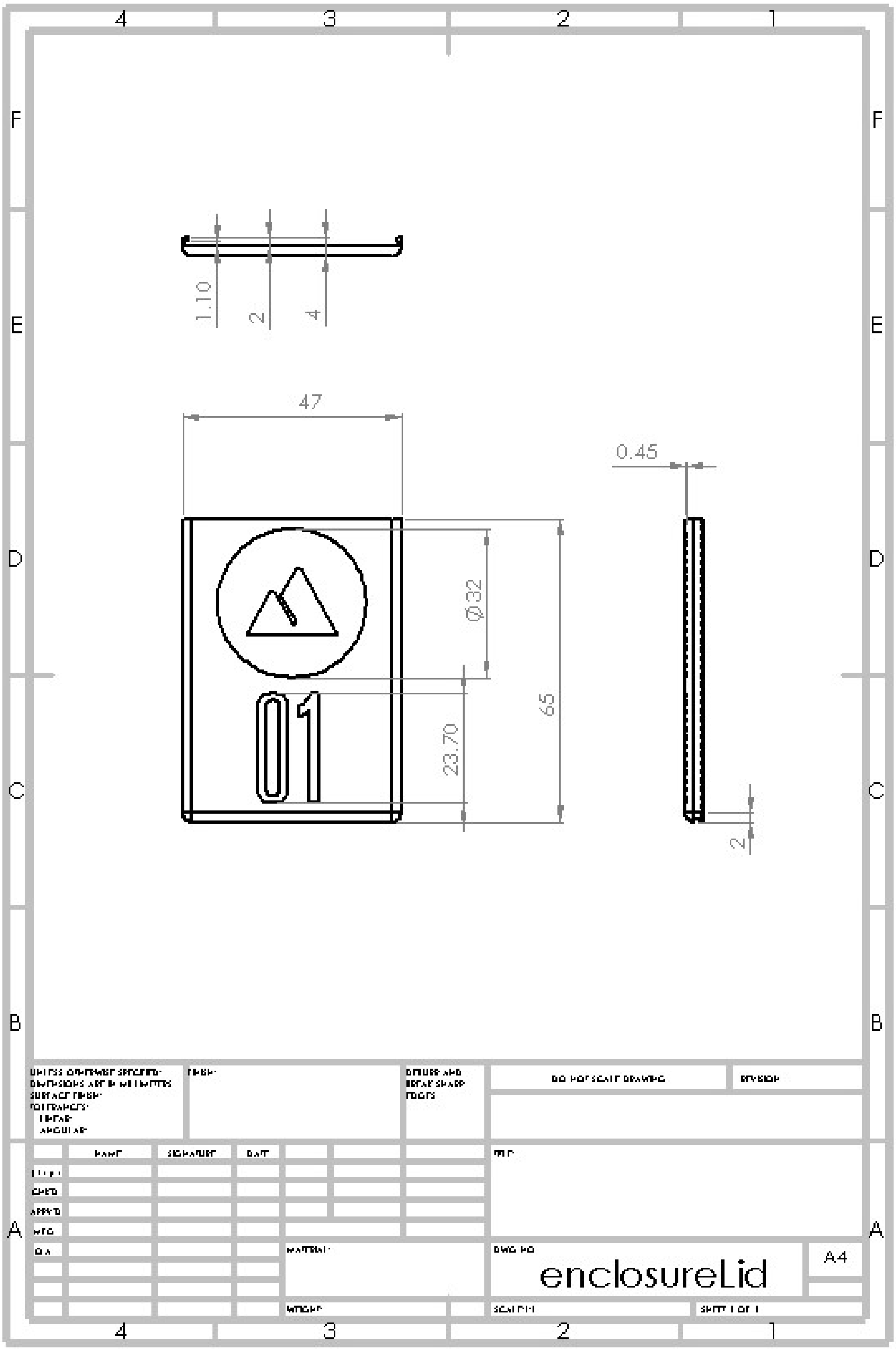
H.4 Exploded Render



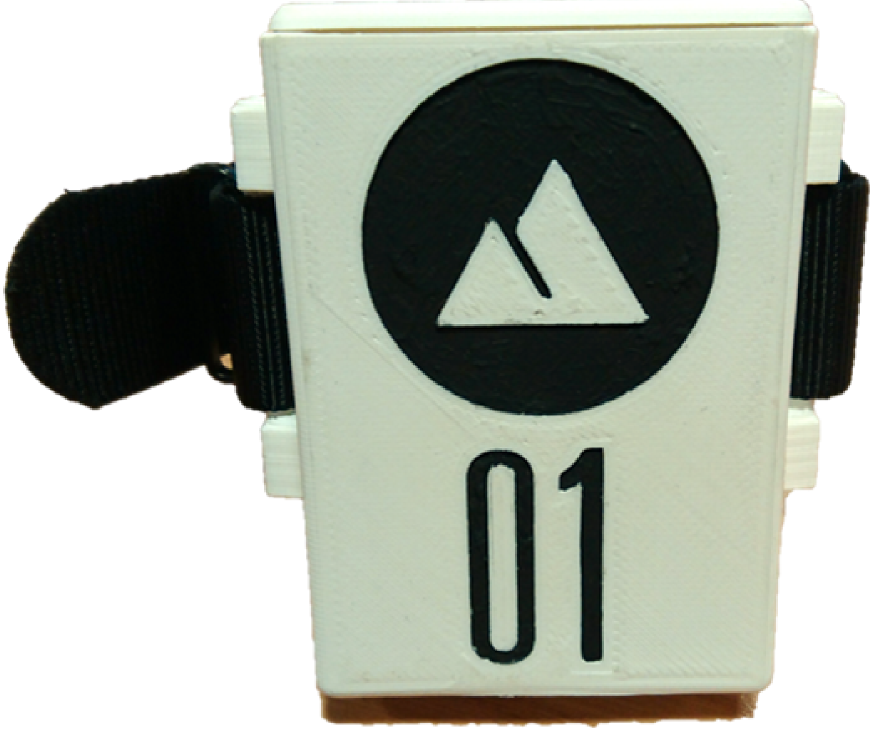
H.5 Enclosure Base Drawing



H.6 Enclosure Lid Drawing



H.7 Final Form



I UDP Receiver Code

```
#!/usr/bin/python
import mysql.connector as mariadb
import socket

#set up udp socket
UDP_IP = "0.0.0.0"
UDP_PORT = 5000

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP

sock.bind((UDP_IP, UDP_PORT))
sock.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 1)

#connect with mysql server and create cursor
mariadb_connection = mariadb.connect(user='pi', password='Team16ProjectClimber',
    ↪ host='localhost', database='climbing')
cursor = mariadb_connection.cursor(buffered=True)
print("connected to MariaDB server")

old_data = ""
i=0
while True:
  data, addr = sock.recvfrom(1024)
  #see if there is recieved data that isn't a repeated packet
  if data != old_data:
    print(data)
        print("duplicate number: " + str(i))
        i=0
        #parse packet
        tag_id = data.splitlines()[1]
        wristband_id = data.splitlines()[2]
        timestamp = int(data.splitlines()[3] + data.splitlines()[4])

        #add new tag to database
        cursor.execute("SELECT * FROM tag WHERE tag_id=%s", (tag_id,))
        if cursor.fetchone() == None:
          cursor.execute("INSERT INTO tag (tag_id, login) VALUES(%s, 0)",
    ↪ (tag_id,))
          print('new tag added')
          mariadb_connection.commit()

        #check if login tag and assign wristband
        cursor.execute("SELECT login FROM tag WHERE tag_id=%s", (tag_id,))
        if cursor.fetchone()[0] == 1:
          #add new wristband to database
          cursor.execute("SELECT * FROM wristband WHERE wristband_id=%s",
    ↪ (wristband_id,))
          if cursor.fetchone() == None:
                cursor.execute("INSERT INTO wristband (wristband_id, tag_id)
    ↪ VALUES(%s, %s)", (wristband_id, tag_id,))
                print('new wristband added')
                mariadb_connection.commit()

          else:
                cursor.execute("UPDATE wristband SET tag_id=%s WHERE
    ↪ wristband_id=%s", (tag_id, wristband_id,))
                print('updated wristband')
                mariadb_connection.commit()

          #check if login attempt is occurring
          cursor.execute("SELECT user_connect FROM tag WHERE tag_id=%s", (tag_id,))
          user_connect = cursor.fetchone()[0]
          if user_connect != None:
                cursor.execute("SELECT wristband_id FROM user WHERE wristband_id=%s", (wristband_id,))
                if cursor.fetchone() != None:
                  cursor.execute("UPDATE user SET wristband_id=NULL WHERE wristband_id=%s", (wristband_id,))
                  print('removed wristband')
                  mariadb_connection.commit()
                cursor.execute("UPDATE user SET wristband_id=%s WHERE user_id=%s", (wristband_id, user_connect,))
                mariadb_connection.commit()
                print("wristband assigned")
          else:
                print("login failed")

        #standard use
        else:
          cursor.execute("SELECT * FROM wristband WHERE wristband_id=%s", (wristband_id,))
          if cursor.fetchone() != None:
                cursor.execute("SELECT user_id FROM user WHERE wristband_id=%s", (wristband_id,))
                if cursor.fetchone() != None:
                  cursor.execute("SELECT problem_number FROM problem WHERE start_id=%s OR finish_id=%s", (tag_id, tag_id,))
                  if cursor.fetchone() != None:

                        #get user id
                        cursor.execute("SELECT user_id FROM user WHERE wristband_id=%s", (wristband_id,))
                        user_id = cursor.fetchone()[0]

                        #get problem id if finish hold or end hold
                        cursor.execute("SELECT problem_number FROM problem WHERE start_id=%s OR finish_id=%s", (tag_id, tag_id,))
                        problem_number = cursor.fetchone()[0]

                        #work out if it's start hold or finish hold
                        cursor.execute("SELECT start_id FROM problem WHERE problem_number=%s", (problem_number,))

                        #check if start hold
                        if cursor.fetchone()[0] == tag_id:

                          #check if user has tried problem before
                          cursor.execute("SELECT attempts FROM user_problem WHERE user_id=%s AND problem_number=%s", (user_id, problem_number,))
                          if cursor.fetchone() == None:

                                #create new row for user and problem
                            cursor.execute("INSERT INTO user_problem (user_id, problem_number, attempts, sends, fastest_send) VALUES(%s, %s, 1, 0, 0)", (user_id, problem_number,))
                                mariadb_connection.commit()
                            print('user attempting new problem')

```

```
              #increment attempt counter
                      else:
                            cursor.execute("UPDATE user_problem SET attempts = attempts + 1 WHERE user_id=%s AND problem_number=%s", (user_id, problem_number,))
                            mariadb_connection.commit()
                            print('attempts incremented')

                      #update time and tag id in wristband
                      cursor.execute("UPDATE wristband SET tag_id=%s, start_time=%s WHERE wristband_id=%s", (tag_id, timestamp, wristband_id,))
                      mariadb_connection.commit()

                    #increment sends (i.e. finish hold)
                    else:
                      #check wristband to see if it matches with the same problem as start hold, if true then increment succeses
                      cursor.execute("SELECT finish_id FROM problem WHERE start_id=(SELECT tag_id FROM wristband WHERE wristband_id=%s)", (wristband_id,))
                      query = cursor.fetchone()
                      if query == None:
                            print("tag assignment error")

                      elif query[0] == tag_id:
                            cursor.execute("UPDATE user_problem SET sends = sends + 1 WHERE user_id=%s AND problem_number=%s", (user_id, problem_number,))
                            mariadb_connection.commit()
                            print("incremented sends")
                            cursor.execute("UPDATE wristband SET tag_id = NULL WHERE wristband_id=%s", (wristband_id,))
                            mariadb_connection.commit()

                            #check if this is fastest clear
                            cursor.execute("SELECT start_time FROM wristband WHERE wristband_id=%s", (wristband_id,))
                            clear_time = timestamp - cursor.fetchone()[0]
                            cursor.execute("SELECT fastest_send FROM user_problem WHERE user_id=%s AND problem_number=%s", (user_id, problem_number,))
                            best_time = cursor.fetchone()[0]
                            print(str(timestamp))
                            cursor.execute("SELECT start_time FROM wristband WHERE wristband_id=%s", (wristband_id,))
                            print(cursor.fetchone()[0])
                            print("clear time: " + str(clear_time) + " best_time: " + str(best_time))
                            if best_time > clear_time or best_time == 0:
                              cursor.execute("UPDATE user_problem SET fastest_send=%s WHERE user_id=%s AND problem_number=%s", (clear_time, user_id, problem_number,))
                              mariadb_connection.commit()
                              print("new best time!")
          else:
                            print("Finish hold doesn't match start hold")

              else:
                    cursor.execute("UPDATE wristband SET tag_id=%s WHERE wristband_id=%s", (tag_id, wristband_id,))
                    mariadb_connection.commit()
```

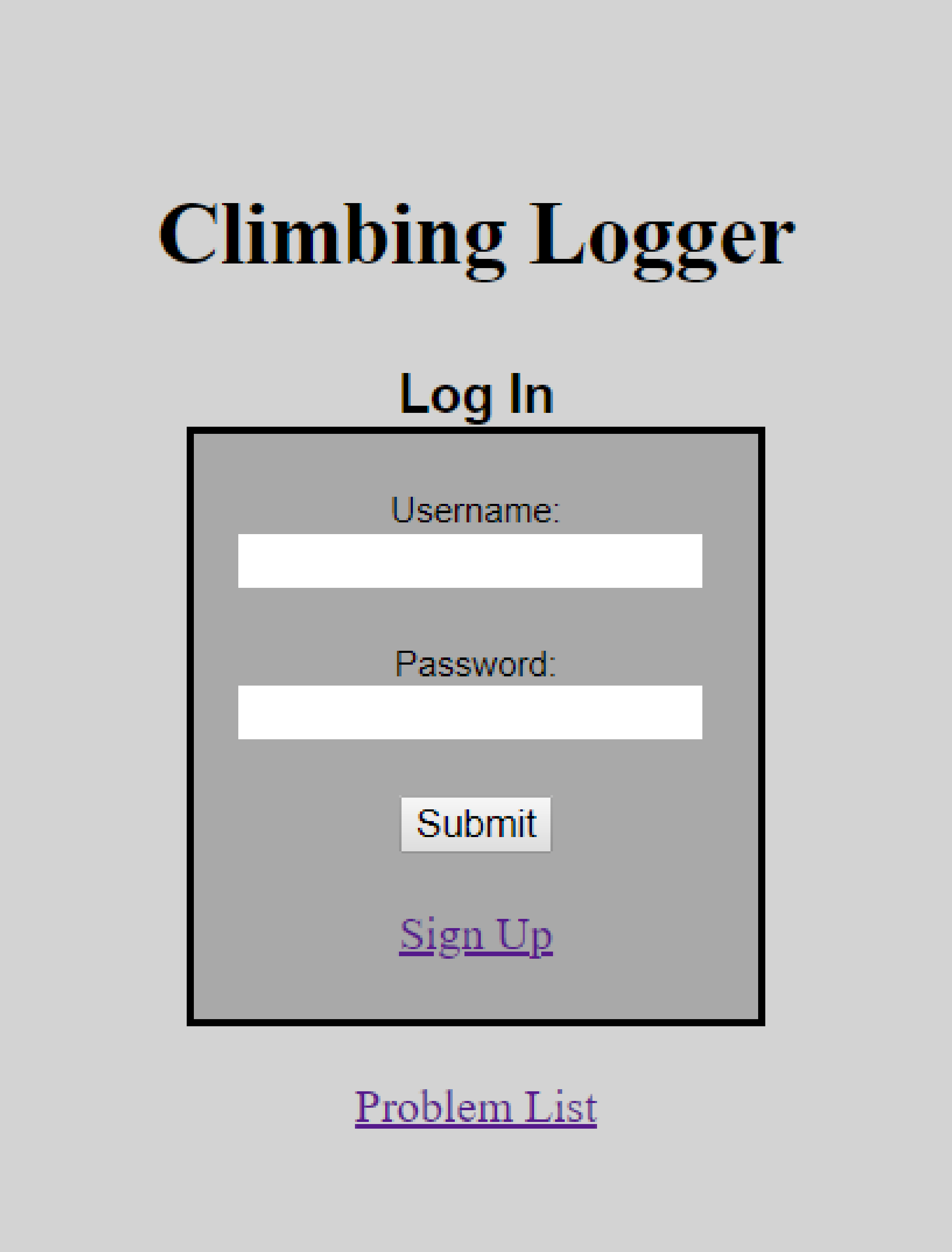
```
                        print("tag not assigned to problem")

                else:
                  cursor.execute("UPDATE wristband SET tag_id=%s WHERE wristband_id=%s", (tag_id, wristband_id,))
                  mariadb_connection.commit()
                  print("wristband not assigned")

          else:
                print("wristband not registered in system")

        old_data = data
        mariadb_connection.commit()
  else:
        i=i+1
mariadb_connection.close()
```

J UDP Receiver Tests

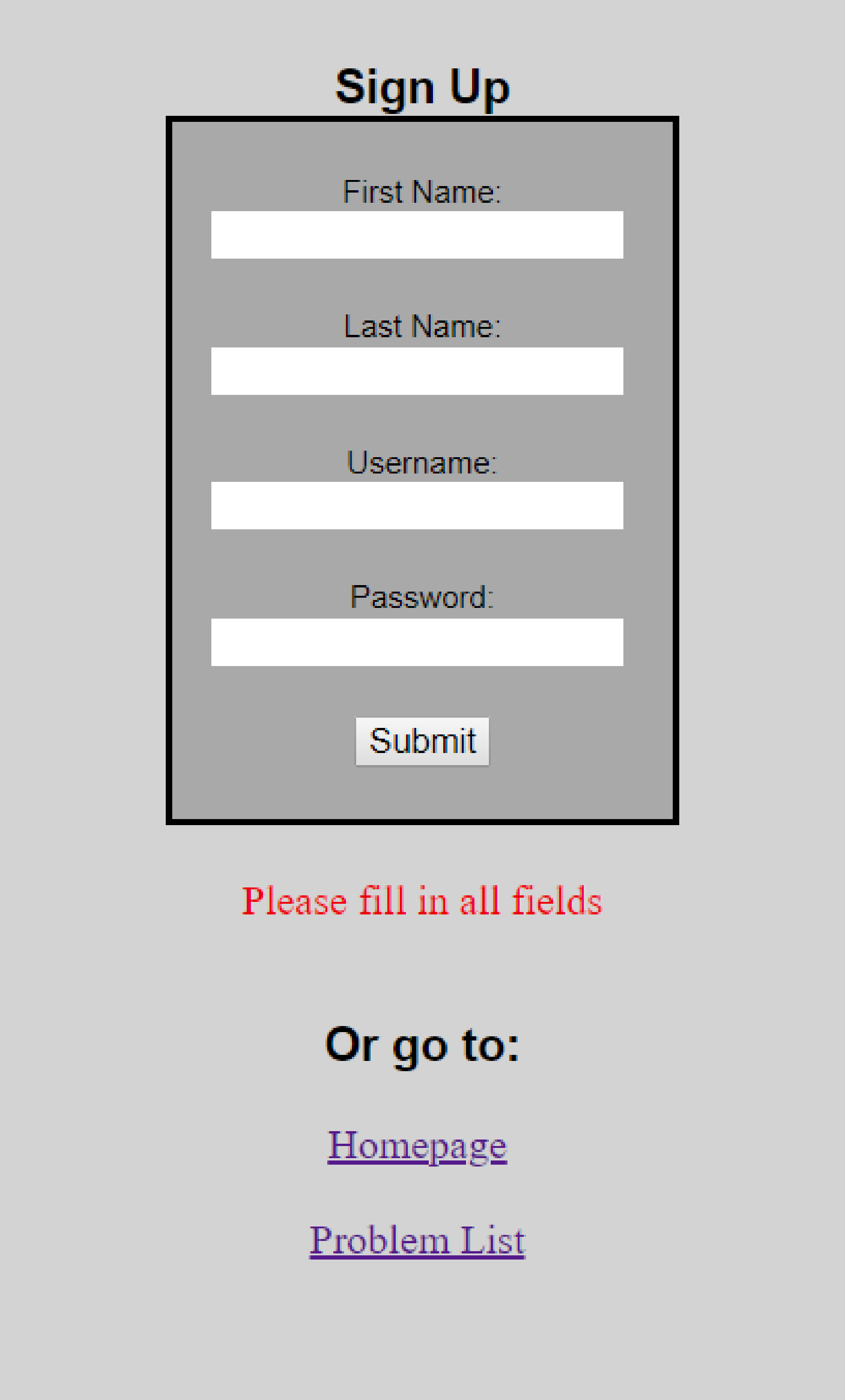
Test	***Database Setup***	***String Input***	***Expected Result***	***Result***
User login	User ID in *user_connect* for a login tag	Login ID Wristband ID timestamp	Wristband ID in *user*	As expected
New tag	Tag ID not in *tag*	Tag ID Wristband ID timestamp	Tag ID added to *tag*	System crash
New wristband	Wristband ID not in *wristband*	Login ID Wristband ID timestamp	Wristband ID added to *wristband*	As expected
User starts new problem	User-problem pair doesn't exist in *user_problem*	Start ID Wristband ID timestamp	User_problem pair added to *user_problem* with *attempts* and *fastest_send* set to 1 and 0 respectively. *start_time* updates with timestamp	As expected
User restarts problem	Start ID in *tag_id* for wristband	Start ID Wristband ID timestamp	*attempts* for user-problem pair increments and *start_time* updates with current timestamp	As expected
User finishes problem slower than best time	*fastest_send* for user-problem pair set to -1. *tag_id* for wristband set to start ID of problem	Finish ID Wristband ID timestamp	*sends* increments	As expected
User finishes problem faster than best time	*fastest_send* for user-problem pair set to 1. *tag_id* for wristband set to start ID of problem	Finish ID Wristband ID timestamp	*sends* increments and *fastest_send* set to 0	As expected
User scans finish tag different from different problem to previous start tag	*tag_id* set to 0 in *wristband*	Finish ID Wristband ID timestamp	Nothing	System crash
User scans unassigned tag	Tag ID removed from *problem*	Tag ID Wristband ID timestamp	Nothing	As expected
User uses unassigned wristband	Wristband ID removed from *user*	Tag ID Wristband ID timestamp	Nothing	As expected

K Website Images

K.1 Login Page



K.2 Sign Up Page



Sign Up

First Name:

Last Name:

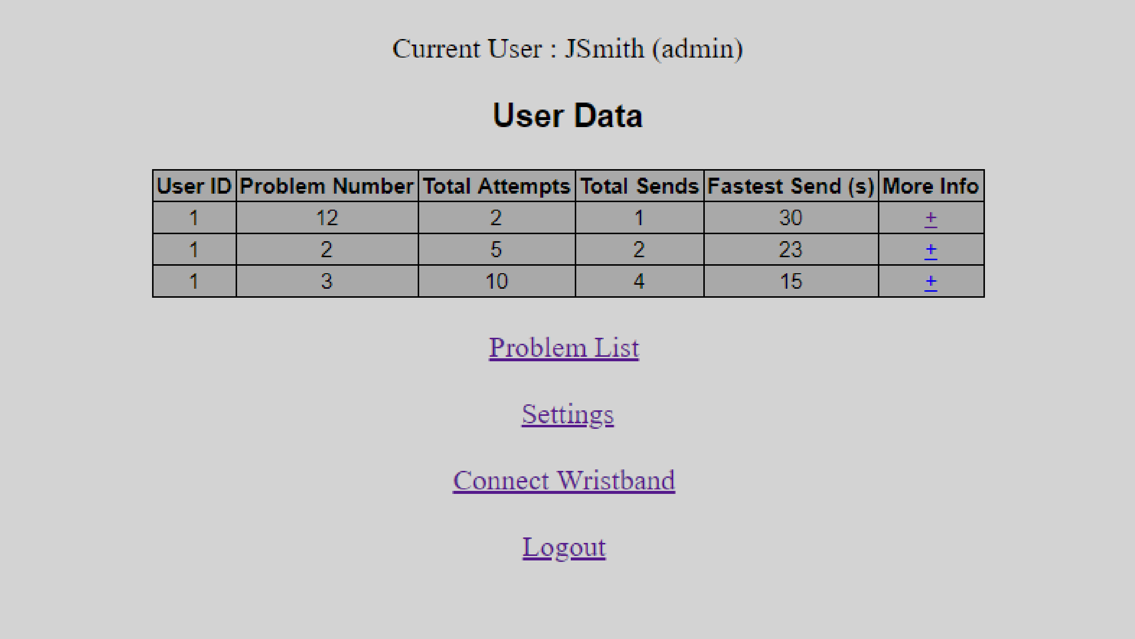
Username:

Password:

Please fill in all fields

Or go to:

K.3 User Data Page

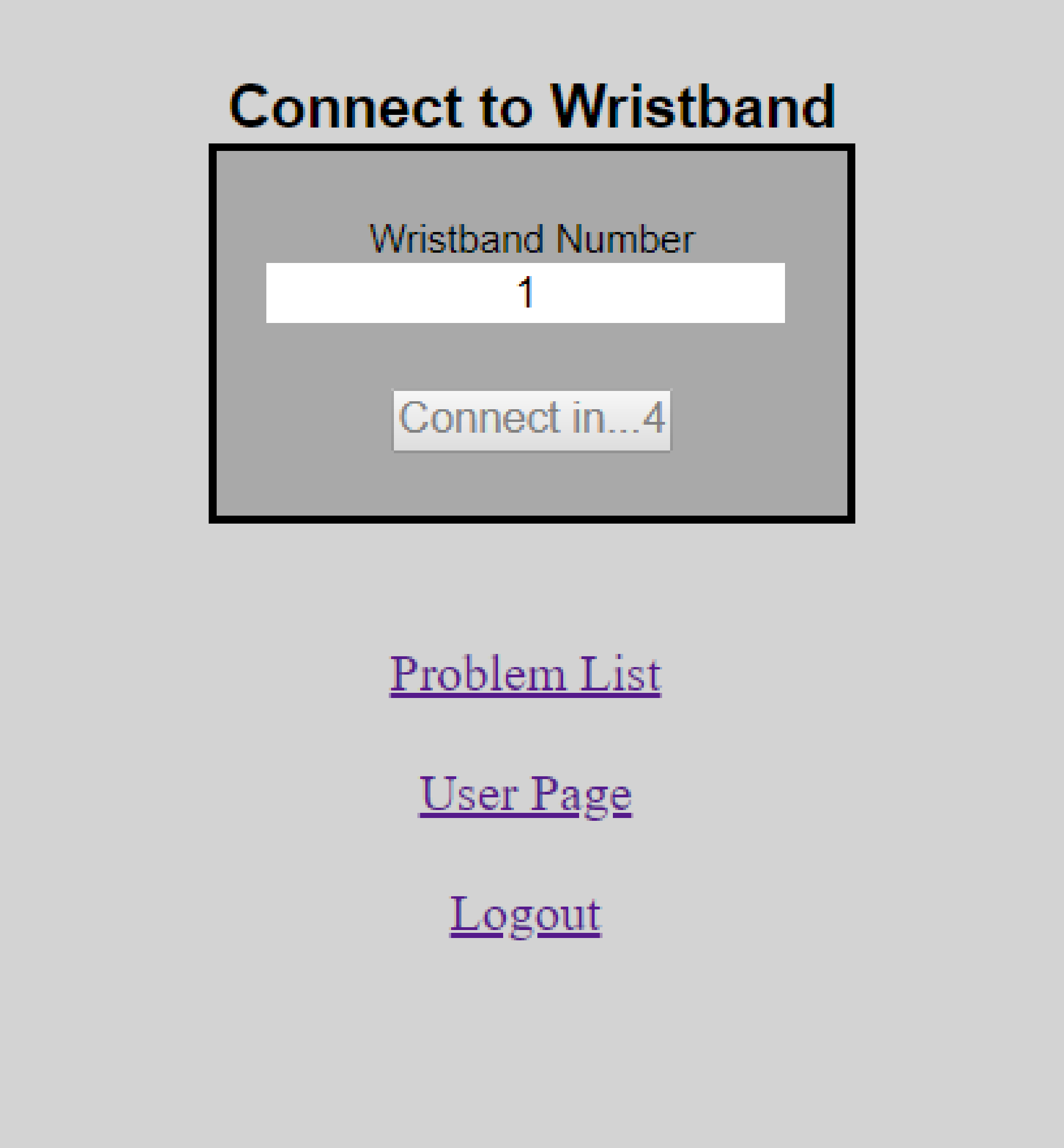


Current User : JSmith (admin)

User Data

User ID	Problem Number	Total Attempts	Total Sends	Fastest Send (s)	More Info
1	12	2	1	30	+
1	2	5	2	23	+
1	3	10	4	15	+

K.4 Connect Wristband

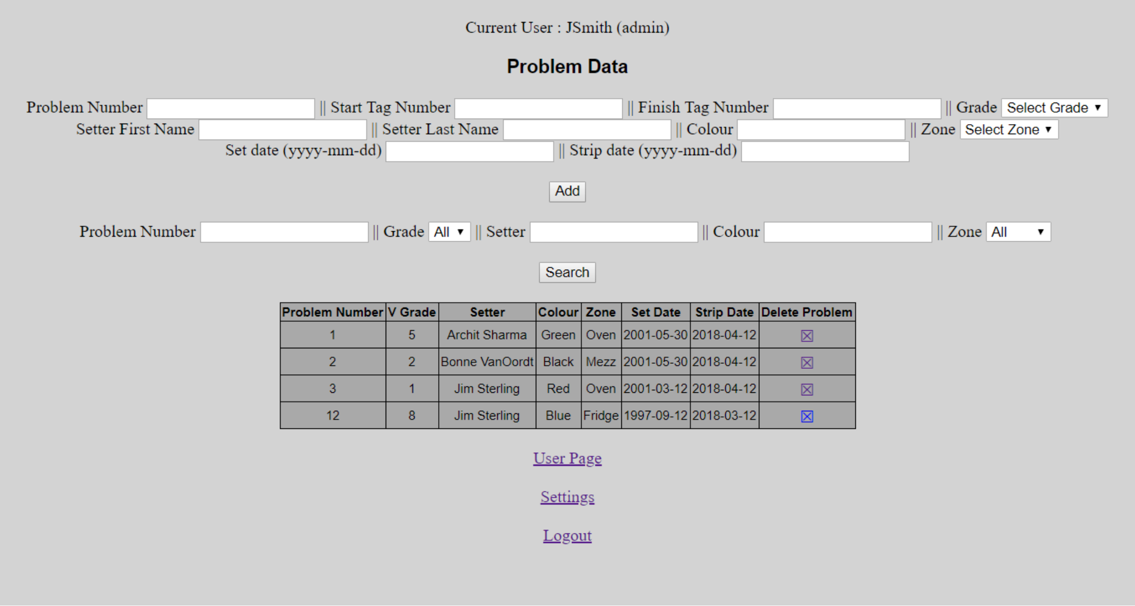


Connect to Wristband

Wristband Number

1

K.5 Problem Data Page



Current User : JSmith (admin)

Problem Data

Problem Number || Start Tag Number || Finish Tag Number || Grade Select Grade ▾
Setter First Name || Setter Last Name || Colour || Zone Select Zone ▾
Set date (yyyy-mm-dd) || Strip date (yyyy-mm-dd)

Add

Problem Number || Grade All ▾ || Setter || Colour || Zone All ▾

Search

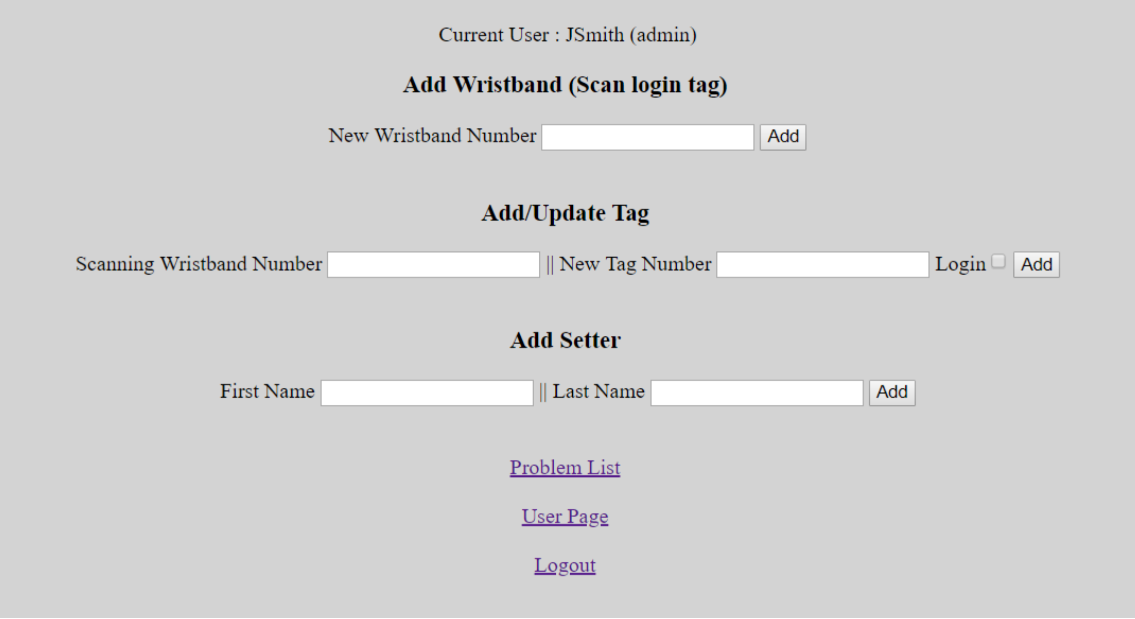
Problem Number	V Grade	Setter	Colour	Zone	Set Date	Strip Date	Delete Problem
1	5	Archit Sharma	Green	Oven	2001-05-30	2018-04-12	☒
2	2	Bonne VanOordt	Black	Mezz	2001-05-30	2018-04-12	☒
3	1	Jim Sterling	Red	Oven	2001-03-12	2018-04-12	☒
12	8	Jim Sterling	Blue	Fridge	1997-09-12	2018-03-12	☒

User Page

Settings

Logout

K.6 Settings Page



Current User : JSmith (admin)

Add Wristband (Scan login tag)

New Wristband Number Add

Add/Update Tag

Scanning Wristband Number || New Tag Number Login □ Add

Add Setter

First Name || Last Name Add

Problem List

User Page

Logout

L Website Test Table

Test	***Action***	***Field Input(s)***	***Expected Result***	***Result***
Login	Entered correct credentials	Username: JSmith Password: testing123	Redirect to User Data page and show JSmith as the current user (admin)	As expected
Sign Up	Enter some information and submit	First Name: Bert Second Name: Jenkins Username: BertieBoy Password: bBoy123	Update `User` table with new information	As expected
More Info	Press more info next to a problem	NA	Redirect to the Problem Data page with only the selected problem visible	As expected
Connect wristband	Scan login tag with wristband and enter appropriate data on site	Wristband Number: 1	Update `User` table with wristband number	As expected
Search	Enter a colour and click search	Colour: Blue	Show a list of problems that all have blue as the colour	As expected
Add problem	Enter valid credentials and press add	14 1 2 V4 Jim Sterling Red Fridge 2018-03-16 2018-04-16	Add problem to `Problem` table	As expected
Add wristband	Scan a login tag and enter a new wristband number	New Wristband Number: 7	Update `Wristband` table with new wristband number	As expected
Add/Update Tag	Scan a tag and enter the new tag number	Scanning Wristband Number: 1 New Tag Number: 4	Update `Tag` table with new tag number	As expected
Add setter	Enter new setter names and click add	First Name: Bert Second Name: Jenkins	Update `Setter` table with new setter	As expected

M PHP Code

M.1 homepage.php

```php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Log In</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div id="pagetitle"><h1>Climbing Logger</h1></div>
<div id="climbingform">
<h2>Log In</h2>
<div id="forminputs">
<form action="#" method="post">
<p>Username:</p>
<?php  //populate input box with previous value if it exists
if (array_key_exists("username", $_POST)) {
        echo "<input type=\"text\" name=\"username\" value=\"{$_POST["username"]}\">";
} else {
        echo "<input type=\"text\" name=\"username\" value=\"\">";
}
?>
<br><br>
<p>Password:</p>
<input type="password" name="password" value="">
<br><br>
<input type="submit" value=" Submit ">
</form>
<br>
<a href="signup.php">Sign Up</a>
</div>
<?php
session_start(); //open session variable
$_SESSION = array(); //initialise session to an empty array
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
}
if (!array_key_exists ("username", $_POST)) { //checks if user has attempted log in
} else {
        if ($_POST["username"] == null) { //if username field is empty
                $_POST = array();
                echo "Incorrect username or password<br>";
        } else {
                $query = mysqli_query($dbconnect, "SELECT * FROM user where username = '".$_POST["username"]."'") or die (mysqli_error($dbconnect)); //query database to get password
                $info = mysqli_fetch_array($query);

                if ($info["password"] == $_POST["password"] && $info["username"] == $_POST["username"]) { //redirects if login was correct
```

```
                        $_SESSION = $_POST;
                        header("Location: userdata.php");
                        exit();
                } else { //outputs login error if login was incorrect
                        $_POST = array();
                        echo "Incorrect username or password<br>";
                }
        }
}
?>
<br><a href="problemdata.php">Problem List</a>
</body>
</html>
```

M.2 signup.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Sign Up</title>
<link rel="stylesheet" href="style.css" />
</head>
<body>
<div id="climbingform">
<br><h2>Sign Up</h2><br>
<div id="forminputs">
<form action="#" method="post">
<p>First Name:</p>
<?php //populate input box with previous value if it exists
if (array_key_exists("name_first", $_POST)) {
        echo "<input type=\"text\" name=\"name_first\"
    ↪ value=\"{$_POST["name_first"]}\">";
} else {
        echo "<input type=\"text\" name=\"name_first\" value=\"\">";
}
?>
<br><br>
<p>Last Name:</p>
<?php //populate input box with previous value if it exists
if (array_key_exists("name_first", $_POST)) {
        echo "<input type=\"text\" name=\"name_last\"
    ↪ value=\"{$_POST["name_last"]}\">";
} else {
        echo "<input type=\"text\" name=\"name_last\" value=\"\">";
}
?>
<br><br>
<p>Username:</p>
<?php //populate input box with previous value if it exists
if (array_key_exists("username", $_POST)) {
        echo "<input type=\"text\" name=\"username\"
    ↪ value=\"{$_POST["username"]}\">";
} else {
        echo "<input type=\"text\" name=\"username\"value=\"\">";
}
?>
<br><br>
<p>Password:</p>
<input type="password" name="password" value="">
<br><br>
<input type="submit" value=" Submit ">
</form>
</div>
</div>
<?php
session_start(); //open session variable
$_SESSION = array(); //initialise session to an empty array
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect=mysqli_connect($hostname,$username,$password,$db); //initiate connection
if ($dbconnect->connect_error) { //output connection error if connection fails
```

```
        die("Database connection failed: " . $dbconnect->connect_error);
}
if (array_key_exists ("username", $_POST) && array_key_exists ("name_first",
    ↪ $_POST) && array_key_exists ("name_last", $_POST) && array_key_exists
    ↪ ("password", $_POST)) { //check user logged in correctly
        if ($_POST["username"] == null || $_POST["name_first"] == null ||
    ↪ $_POST["name_last"] == null || $_POST["password"] == null) { //check fields
    ↪ aren’t empty
                //NOTE this prevents URL login bypass
                $_POST = array();
                echo "<div style=\"color:#db204e\">Please fill in all
    ↪ fields</div><br><br>";
        } else {
                $query = mysqli_query($dbconnect, "SELECT * FROM user where
    ↪ username =  ’{$_POST["username"]}’;") or die (mysqli_error($dbconnect));
    ↪ //get user data from database
                $user_info = mysqli_fetch_array($query);

                if(is_array($user_info)) { //if username is already in use
                        $_POST["username"] = "";
                        echo "<div style=\"color:#db204e\">Username already
    ↪ exists</div><br><br>";
                } else {
                        $query = mysqli_query($dbconnect, "INSERT INTO user
    ↪ (username, name_first, name_last, password) values(’{$_POST["username"]}’,
    ↪ ’{$_POST["name_first"]}’, ’{$_POST["name_last"]}’, ’{$_POST["password"]}’)")
    ↪ or die (mysqli_error($dbconnect));
                        echo "<div style=\"color:#00932c\">Success! <br><a
    ↪ href=\"/homepage.php\">Login</a></div><br>"; //creates user
                }
        }
} else {
        echo "<div style=\"color:#ed0007\">Please fill in all
    ↪ fields</div><br><br>";
}
?>
<h2>Or go to:</h2><br>
<a href="homepage.php">Homepage</a>
<br><br>
<a href="problemdata.php">Problem List</a>
<br>
</body>
</html>
```

M.3 userdata.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - User Data</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<br>
<?php
session_start(); //open session variable
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
} //check user is logged in
if (!array_key_exists("username", $_SESSION)) { //is user logged in
        header("Location: homepage.php");
        exit();
} else { //check user is logged in correctly
        $query = mysqli_query($dbconnect, "SELECT * FROM user where username = '".$_SESSION["username"]."';") or die (mysqli_error($dbconnect));
        $user_data = mysqli_fetch_array($query);
        if ($user_data["username"] != $_SESSION["username"] || $user_data["password"] != $_SESSION["password"]) {
                header("Location: homepage.php");
                exit();
        } else {
                $admin = "";
                $admin_in = false; //set admin status to default (no)
                if ($user_data["admin"] == 1) { //if user is an admin
                        $admin_in = true;
                        $admin = " (admin)";
                }
                echo "<center>Current User : " . $user_data["username"] . $admin . "<br><br>"; //current user printout
                $query = mysqli_query($dbconnect, "SELECT * FROM user_problem where user_id = '".$user_data["user_id"]."';") or die (mysqli_error($dbconnect));
                //printout all user_problem data
                ?>

                <h2>User Data</h2><br>
                <table>
                <tr>
                <td><strong>User ID</strong></td>
                <td><strong>Problem Number</strong></td>
                <td><strong>Total Attempts</strong></td>
                <td><strong>Total Sends</strong></td>
                <td><strong>Fastest Send (s)</strong></td>
                <td><strong>More Info</strong></td>
                </tr>
                <?php //populate table with database data
                while ($row = mysqli_fetch_array($query)) {
                        echo
                        "<tr>
                        <td>{$row['user_id']}</td>
                        <td>{$row['problem_number']}</td>
                        <td>{$row['attempts']}</td>
                        <td>{$row['sends']}</td>
                        <td>{$row['fastest_send']}</td>
                        <td><a href='problemdata.php?pn={$row['problem_number']}'><center>+</a></td>
                </tr>\n";
                }
        }
}?>
</table>
<br>
<a href="problemdata.php">Problem List</a>
<br><br>
<?php
if ($admin_in == true) { //settings tab for admins only
        echo "<a href=\"settings.php\">Settings</a>";
        echo "<br><br>";
}?>
<a href="connect.php">Connect Wristband</a>
<br><br>
<a href="homepage.php">Logout</a>
<br>
</body>
</html>
```

M.4 connect.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Connect</title>
<link rel="stylesheet" href="style.css"</link>
</head>
<body>
<div id="climbingform">
<h2>Connect to Wristband</h2>
<div id="forminputs">
<form action="/connect.php" method="post" id="connect_form">
<p>Wristband Number</p>
<?php //populate input box with previous value if it exists
if (array_key_exists("num", $_POST)) {
        echo "<input type=\"text\" name=\"num\" value=\"{$_POST["num"]}\">";
} else {
        echo "<input type=\"text\" name=\"num\" value=\"\">";
}?>
<br><br>
<button type="submit" id="button" form="connect_form"><div id="connect"> Connect
    ↪ </div></button>
</form>
</div>
<script> //javascript to display countdown for wristband connect
var time = 5;
var counter = time;
function countdown(){
        document.getElementById("button").disabled = true;
        document.getElementById("connect").innerHTML = "Connect in..." + counter;
        var timer = setInterval(function(){
                if(counter > 1){
                        counter--;
                        document.getElementById("connect").innerHTML = "Connect
    ↪ in..." + counter;
                }
                else{
                        counter = time;
                        clearInterval(timer);
                        document.getElementById("connect").innerHTML = " Connect ";
                        document.getElementById("button").disabled = false;
                        window.location.replace("/connectresult.php");
                }
        }, 1000);
}</script>
<?php
session_start(); //open session variable
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate
    ↪ connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
}//perform correct user check (prevent URL bypass)
if (array_key_exists ("username", $_SESSION)){
        if ($_SESSION["username"] == null) {
```

```
        header("Location: homepage.php");
        exit();
    } else {
        $query = mysqli_query($dbconnect, "SELECT * FROM user where
  ↪ username =  '{$_SESSION["username"]}'") or die (mysqli_error($dbconnect));
        $user_info = mysqli_fetch_array($query);
        if ($user_info["password"] == $_SESSION["password"] &&
  ↪ $user_info["username"] == $_SESSION["username"]) {
            if (array_key_exists("num", $_POST)) { //if user has input
  ↪ a wristband number to connect to
                $query = mysqli_query($dbconnect, "UPDATE tag SET
  ↪ user_connect = '{$user_info["user_id"]}' WHERE login = 1") or die
  ↪ (mysqli_error($dbconnect)); //set temp data
                echo "<script> countdown(); </script>"; //start
  ↪ countdown script then redirects to connectresult
                $query = mysqli_query($dbconnect, "SELECT * FROM
  ↪ wristband WHERE wristband_number = '{$_POST["num"]}'") or die
  ↪ (mysqli_error($dbconnect));
                $wristband_info = mysqli_fetch_array($query);
  ↪ //get wristband data
                if(!is_array($wristband_info)) { //if wristband
  ↪ exists
                    $query = mysqli_query($dbconnect, "UPDATE
  ↪ tag SET user_connect = NULL WHERE login = 1") or die
  ↪ (mysqli_error($dbconnect));
                    header("Location:
  ↪ connect.php?status=no_w");
                    exit(); //remove temp data
                }
                $_SESSION["wristband_id"] =
  ↪ $wristband_info["wristband_id"];
            }
        } else { //redirect i user isn't logged in correctly
            header("Location: homepage.php");
            exit();
        }
    }
} else { //redirect if user isn't logged in
    header("Location: homepage.php");
    exit();
}if(array_key_exists("status", $_GET)){
    $status = $_GET["status"];
    if($status == "success"){ //if connection was successful
        echo "<h2> Success! </h2>";
    }
    else if($status == "no_w"){ //if connection failed
        echo "<h2> Wristband is not in database! </h2>";
    }
    else if($status == "fail"){ //other error
        echo "<h2> Oops...Something went wrong. Please try again! </h2>";
    } }?>
<br><br>
<a href="problemdata.php">Problem List</a>
<br><br>
<a href="userdata.php">User Page</a>
<br><br>
<a href="homepage.php">Logout</a>
<br>
</body>
</html>
```

M.5 connectresult.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Connect Result</title>
<link rel="stylesheet" href="style.css"</link>
</head>
<body>
<?php
session_start(); //open session variable
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate
    ↪ connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
}if(empty($_SESSION)){ //if session is not populated
        header("Location: homepage.php"); //eject user
        exit();
} else {
        $query = mysqli_query($dbconnect, "UPDATE tag SET user_connect = NULL
    ↪ WHERE login = 1;") or die (mysqli_error($dbconnect)); //remove temporary data
        $query = mysqli_query($dbconnect, "SELECT * FROM user WHERE username =
    ↪ '{$_SESSION["username"]}';") or die (mysqli_error($dbconnect));
        $user_info = mysqli_fetch_array($query); //get user data
        if ($user_info["wristband_id"] == $_SESSION["wristband_id"]){ //if
    ↪ wristband was scanned
                header("Location: connect.php?status=success");
        } else {
                header("Location: connect.php?status=fail");
        }
}?>
</body>
</html>
```

M.6 problemdata.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Problem Data</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<?php
session_start(); //open session variable
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate
    ↪ connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
}$user_in = false; //set default value showing user is not logged in
$admin_in = false; //set default value showing admin is not logged in
if (array_key_exists("username", $_SESSION)) { //check if user is logged in
        $query = mysqli_query($dbconnect, "SELECT * FROM user where username =
    ↪ '".$_SESSION["username"]."';") or die (mysqli_error($dbconnect));
        $user_data = mysqli_fetch_array($query);
        if ($user_data["username"] == $_SESSION["username"] &&
    ↪ $user_data["password"] == $_SESSION["password"]) {
                $user_in = true; //set user is logged in to true
                $admin = "";
                if ($user_data["admin"] == 1) {
                        $admin_column  = "<td><strong>Delete
    ↪ Problem</strong></td>";
                        $admin_in = true; //set admin is logged in to true
                        $admin = " (admin)";
                } else {
                        $admin_column = "";
                }
                echo "<br><center>Current User : " . $user_data["username"] .
    ↪ $admin . "<br>"; //admin variable adds "(admin)" to the current user printout
        } else {
                $admin_column = "";
        }
} else {
        $admin_column = "";
}echo "<br><h2>Problem Data</h2>";
if ($admin_in == true) { //print out add problem form for admins only
        echo "<form action=\"problemdata.php\" method=\"post\"><br>";
        echo "Problem Number ";
        if (array_key_exists("add_problem_number", $_POST)) { //populate input box
    ↪ with previous value if it exists
                echo "<input type=\"number\" name=\"add_problem_number\"
    ↪ value=\"{$_POST["add_problem_number"]}\">";
        } else {
                echo "<input type=\"number\" name=\"add_problem_number\"
    ↪ value=\"\">";
        }
        echo " || Start Tag Number ";
        if (array_key_exists("start_tag", $_POST)) { //populate input box with
    ↪ previous value if it exists
```

```
                echo "<input type=\"number\" name=\"start_tag\" value=\"{$_POST["start_tag"]}\">";
        } else {
                echo "<input type=\"number\" name=\"start_tag\" value=\"\">";
        }
        echo " || Finish Tag Number ";
        if (array_key_exists("finish_tag", $_POST)) { //populate input box with previous value if it exists
                echo "<input type=\"number\" name=\"finish_tag\" value=\"{$_POST["finish_tag"]}\">";
        } else {
                echo "<input type=\"number\" name=\"finish_tag\" value=\"\">";
        }
        echo " || Grade ";
        echo "<select name=\"add_grade\">
                <option value=\"empty\">Select Grade</option>
                <option value=0>V0</option>
                <option value=1>V1</option>
                <option value=2>V2</option>
                <option value=3>V3</option>
                <option value=4>V4</option>
                <option value=5>V5</option>
                <option value=6>V6</option>
                <option value=7>V7</option>
                <option value=8>V8</option>
        </select><br>";
        echo "Setter First Name ";
        if (array_key_exists("add_setter_first", $_POST)) { //populate input box with previous value if it exists
                echo "<input type=\"text\" name=\"add_setter_first\" value=\"{$_POST["add_setter_first"]}\">";
        } else {
                echo "<input type=\"text\" name=\"add_setter_first\" value=\"\">";
        }
        echo " || Setter Last Name ";
        if (array_key_exists("add_setter_last", $_POST)) { //populate input box with previous value if it exists
                echo "<input type=\"text\" name=\"add_setter_last\" value=\"{$_POST["add_setter_last"]}\">";
        } else {
                echo "<input type=\"text\" name=\"add_setter_last\" value=\"\">";
        }
        echo " || Colour ";
        if (array_key_exists("add_colour", $_POST)) { //populate input box with previous value if it exists
                echo "<input type=\"text\" name=\"add_colour\" value=\"{$_POST["add_colour"]}\">";
        } else {
                echo "<input type=\"text\" name=\"add_colour\" value=\"\">";
        }
        echo " || Zone
        <select name=\"add_zone\">
                <option value=\"empty\">Select Zone</option>
                <option value=\"Oven\">Oven</option>
                <option value=\"Fridge\">Fridge</option>
                <option value=\"Mezz\">Mezz</option>
        </select>";
        echo "<br>Set date (yyyy-mm-dd) ";
        if (array_key_exists("set_date", $_POST)) { //populate input box with previous value if it exists
```

```
                echo "<input type=\"text\" name=\"set_date\" value=\"{$_POST["set_date"]}\">";
        } else {
                echo "<input type=\"text\" name=\"set_date\" value=\"\">";
        }
        echo " || Strip date (yyyy-mm-dd) ";
        if (array_key_exists("strip_date", $_POST)) { //populate input box with previous value if it exists
                echo "<input type=\"text\" name=\"strip_date\" value=\"{$_POST["strip_date"]}\">";
        } else {
                echo "<input type=\"text\" name=\"strip_date\" value=\"\">";
        }
        echo "<br><br><input type=\"submit\" value=\" Add \">";
        echo "</form>";
}?>
<form action="problemdata.php" method="post">
<br>
Problem Number
<input type="text" name="problem_number" value="">
|| Grade
<select name="grade">
<option value="all">All</option>
<option value=0>V0</option>
<option value=1>V1</option>
<option value=2>V2</option>
<option value=3>V3</option>
<option value=4>V4</option>
<option value=5>V5</option>
<option value=6>V6</option>
<option value=7>V7</option>
<option value=8>V8</option>
</select>
|| Setter
<input type="text" name="setter" value="">
|| Colour
<input type="text" name="colour" value="">
|| Zone
<select name="zone">
<option value="all">All</option>
<option value="oven">Oven</option>
<option value="fridge">Fridge</option>
<option value="mezz">Mezz</option>
</select>
<br><br>
<input type="submit" value=" Search ">
</form>
<br>
<table>
<tr>
<td><strong>Problem Number</strong></td>
<td><strong>V Grade</strong></td>
<td><strong>Setter</strong></td>
<td><strong>Colour</strong></td>
<td><strong>Zone</strong></td>
<td><strong>Set Date</strong></td>
<td><strong>Strip Date</strong></td>
<?php echo $admin_column ?>
</tr>
<?php //following 2 functions are used to obtain the URL extension
function url_origin( $s, $use_forwarded_host = false ) {
        $ssl      = ( ! empty( $s['HTTPS'] ) && $s['HTTPS'] == 'on' );
        $sp       = strtolower( $s['SERVER_PROTOCOL'] );
        $protocol = substr( $sp, 0, strpos( $sp, '/' ) ) . ( ( $ssl ) ? 's' : '' );
        $port     = $s['SERVER_PORT'];
        $port     = ( ( ! $ssl && $port=='80' ) || ( $ssl && $port=='443' ) ) ? '' : ':'.$port;
        $host     = ( $use_forwarded_host && isset( $s['HTTP_X_FORWARDED_HOST'] ) ) ? $s['HTTP_X_FORWARDED_HOST'] : ( isset( $s['HTTP_HOST'] ) ? $s['HTTP_HOST'] : null );
        $host     = isset( $host ) ? $host : $s['SERVER_NAME'] . $port;
        return $protocol . '://' . $host;
}function full_url( $s, $use_forwarded_host = false ) {
        return url_origin( $s, $use_forwarded_host ) . $s['REQUEST_URI'];
}function isRealDate($date) { //function checks date is in the correct format and valid
        if (false === strtotime($date)) {
                return false;
        }
        list($year, $month, $day) = explode('-', $date);
        return checkdate($month, $day, $year);
}$absolute_url = full_url( $_SERVER ); //gets full URL
//the following nested if mountain checks in the right order that every input has a valid input and outputs the appropriate error message if not
//it also checks with database values to ensure the input data is allowed
if (array_key_exists("add_grade", $_POST)) {
        if ($_POST["add_problem_number"] != "") {
                $query = mysqli_query($dbconnect, "SELECT * from problem where problem_number = {$_POST["add_problem_number"]};") or die (mysqli_error($dbconnect));
                $problem_number_query = mysqli_fetch_array($query);
                if (!is_array($problem_number_query)) {
                        if ($_POST["start_tag"] != "" && $_POST["finish_tag"] != "" && $_POST["start_tag"] != $_POST["finish_tag"]) {
                                $query = mysqli_query($dbconnect, "SELECT * from tag where tag_number = {$_POST["start_tag"]};") or die (mysqli_error($dbconnect));
                                $start_id = mysqli_fetch_array($query);
                                $query = mysqli_query($dbconnect, "SELECT * from tag where tag_number = {$_POST["finish_tag"]};") or die (mysqli_error($dbconnect));
                                $finish_id = mysqli_fetch_array($query);
                                if (is_array($start_id) && is_array($finish_id)) {
                                        $query = mysqli_query($dbconnect, "SELECT * from problem where start_id = '".$start_id[0]."' or finish_id = '".$start_id[0]."'
                                        or start_id = '".$finish_id[0]."' or finish_id = '".$finish_id[0]."';") or die (mysqli_error($dbconnect));
                                        $check_if_tag_in_use = mysqli_fetch_array($query);
                                        if (!is_array($check_if_tag_in_use)) {
                                                if ($_POST["add_grade"] != "empty") {
                                                        if (!preg_match('[\W]', $_POST["add_setter_first"])) {
                                                                if (!preg_match('[\W]', $_POST["add_setter_last"])) {
                                                                        $query = mysqli_query($dbconnect, "SELECT setter_id FROM setter where name_first = '".$_POST["add_setter_first"]."' and name_last = '".$_POST["add_setter_last"]."';") or die (mysqli_error($dbconnect));
```

```
                                    $setter_id_for_add = mysqli_fetch_array($query);
                                    if ($setter_id_for_add != "") {
                                        if (!preg_match('/[^a-z\s-]/i',$_POST["add_colour"])) {
                                            if ($_POST["add_zone"] != "empty") {
                                                if (isRealDate($_POST["set_date"]) == true) {
                                                    if (isRealDate($_POST["strip_date"]) == true) {
                                                        if ($_POST["strip_date"] > date("Y-m-d")) {
                                                            $query = mysqli_query($dbconnect, "insert into problem (problem_number, start_id, finish_id, grade, setter_id, colour, zone, set_date, strip_date) values({$_POST["problem_number"]}, \"{$start_id[0]}\", \"{$finish_id[0]}\", {$_POST["add_grade"]}, {$setter_id_for_add[0]}, \"{$_POST["add_colour"]}\", \"{$_POST["add_zone"]}\", \"{$_POST["set_date"]}\", \"{$_POST["strip_date"]}\");") or die (mysqli_error($dbconnect));
                                                        } else {
                                                            echo "Strip date is in the past";
                                                        }
                                                    } else {
                                                        echo "Strip date doesn't exist";
                                                    }
                                                } else {
                                                    echo "Set date doesn't exist";
                                                }
                                            } else {
                                                echo "No zone selected";
                                            }
                                        } else {
                                            echo "Invalid colour";
                                        }
                                    } else {
                                        echo "Setter doesn't exist";
                                    }
                                } else {
                                    echo "Invalid setter second name";
                                }
                            } else {
                                echo "Invalid setter first name";
                            }
                        } else {
                            echo "No grade selected";
                        }
```

```
                                        } else {
                                                echo "Tag number(s) in use";
                                        }
                                } else {
                                        echo "Tag values(s) don't exist";
                                }
                        } else {
                                echo "Invalid tag values";
                        }
                } else {
                        echo "Problem number in use";
                }
        } else {
                echo "Invalid problem_number";
        }
}if (array_key_exists("grade", $_POST)) { //sets statement concatonators to empty strings
        $and1 = "";
        $and2 = "";
        $and3 = "";
        $and4 = "";
        //following block contructs a query string based on the input parameters
        //this means that if a user only inputs one search parameter the query can still happen
        //the string concatonators are updated accordingly to append new parameters to the total string
        if ($_POST["problem_number"] == "") {
                $search_problem_number = "";
        } else {
                $search_problem_number = "problem_number = {$_POST["problem_number"]}";
        }
        if ($_POST["grade"] == "all") {
                $search_grade = "";
        } else {
                $search_grade = "grade = {$_POST["grade"]}";
                if ($_POST["problem_number"] != "") {
                        $and1 = "and";
                }
        }
        if ($_POST["setter"] == "") {
                $search_setter = "";
        } else {
                if ($_POST["grade"] != "all") {
                        $and2 = "and";
                }
                $names = explode(' ', $_POST["setter"]);
                $name1 = $names[0];
                $name2 = "";
                if (array_key_exists(1, $names)) { //splits name into first and last
                        $name2 = $names[1];
                        $two_names1 = "and name_last = '".$names[1]."'";
                        $two_names2 = "and name_first = '".$names[1]."'";
                        $name2 = $names[1];
                } else {
                        $two_names1 = "";
                        $two_names2 = "";
                }
                $query = mysqli_query($dbconnect, "SELECT setter_id FROM setter where name_first = '".$names[0]."' {$two_names1} or name_last =
'".$names[0]."' {$two_names2};") or die (mysqli_error($dbconnect));
                $search_setter_id = mysqli_fetch_array($query); //gets the setter id for future query
                $search_setter = "setter_id = '".$search_setter_id[0]."'";
        }
        if ($_POST["colour"] == "") {
                $search_colour = "";
        } else {
                if ($_POST["setter"] != "") {
                        $and3 = "and";
                }
                $search_colour = "colour = \"{$_POST["colour"]}\"";
        }
        if ($_POST["zone"] == "all") {
                $search_zone = "";
        } else {
                if ($_POST["colour"] != "") {
                        $and4 = "and";
                }
                $search_zone = "zone = \"{$_POST["zone"]}\"";
        }
        if ($_POST["problem_number"] != "" || $_POST["grade"] != "all" || $_POST["setter"] != "" || $_POST["colour"] != "" || $_POST["zone"] != "all") { //check all input fields aren't blank
                $search = "where {$search_problem_number} {$and1} {$search_grade} {$and2} {$search_setter} {$and3} {$search_colour} {$and4} {$search_zone}";
        } else {
                $search = "";
        }
} else {
        $search = "";
}if ($absolute_url == "problemdata.php") {
        $query = mysqli_query($dbconnect, "SELECT * FROM problem {$search};") or die (mysqli_error($dbconnect));
} else {
        //following block obtains URL extension to determine search or delete
        $p_d_split = Explode('?', $absolute_url);
        $p_d_all = $p_d_split[count($p_d_split) - 1];
        $p_d = substr($p_d_all, -strlen($p_d_all), 1);
        $parts = Explode('=', $absolute_url);
        $id = $parts[count($parts) - 1];
        if ($p_d == 'd' && $admin_in == true) { //if user is admin and delete is true
                $del = mysqli_query($dbconnect, "delete from problem where problem_number = {$id};") or die (mysqli_error($dbconnect));
                $query = mysqli_query($dbconnect, "SELECT * FROM problem {$search};") or die (mysqli_error($dbconnect));
        } else {
                if ($p_d == 'p_d') {
                        $query = mysqli_query($dbconnect, "SELECT * FROM problem where problem_number = {$id};") or die (mysqli_error($dbconnect)); //perform search by problem_number
                } else { //if URL is neither search or delete
                        $query = mysqli_query($dbconnect, "SELECT * FROM problem {$search};") or die (mysqli_error($dbconnect));
                }
        }
}while ($row = mysqli_fetch_array($query)) { //keep outputting until full table has been outputted
        $query1 = mysqli_query($dbconnect, "SELECT * FROM setter where setter_id = {$row["setter_id"]};") or die (mysqli_error($dbconnect));
```

```php
        $setter_name = mysqli_fetch_array($query1);
        if ($admin_in == True) {
                $admin_del = "<td><a
    ↪ href='problemdata.php?did={$row['problem_number']}'
    ↪ style=\"text-decoration:none\"><center><font
    ↪ size=\"3\">&#9746;</font></a></td>";
        } else {
                $admin_del = "";
        }
        echo
        "<tr>
        <td>{$row['problem_number']}</td>
        <td>{$row['grade']}</td>
        <td>{$setter_name['name_first']} {$setter_name['name_last']}</td>
        <td>{$row['colour']}</td>
        <td>{$row['zone']}</td>
        <td>{$row['set_date']}</td>
        <td>{$row['strip_date']}</td>
                {$admin_del}
</tr>\n";
}//show site navigation conditionally
echo "</table><br>";
$in_or_out = "in";
if ($user_in == true) {
        echo "<a href=\"userdata.php\">User Page</a><br><br>";
        if ($admin_in == true) {
                echo "<a href=\"settings.php\">Settings</a><br><br>";
        }
        $in_or_out = "out";
}echo "<a href=\"homepage.php\">Log{$in_or_out}</a><br>";
?>
</body>
</html>
```

M.7 settings.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Climbing Logger - Settings</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<?php
session_start();
//connect to mysql server on pi
$hostname = "localhost";
$username = "pi";
$password = "Team16ProjectClimber";
$db = "climbing";
$dbconnect = mysqli_connect($hostname,$username,$password,$db); //initiate connection
if ($dbconnect->connect_error) { //output connection error if connection fails
        die("Database connection failed: " . $dbconnect->connect_error);
}
if (!array_key_exists("username", $_SESSION)) { //if session variable is populated
        header("Location: homepage.php"); //eject user
        exit();
} else {
        $query = mysqli_query($dbconnect, "SELECT * FROM user where username = '".$_SESSION["username"]."';") or die (mysqli_error($dbconnect));
        $user_data = mysqli_fetch_array($query); //get user data

        if ($user_data["username"] != $_SESSION["username"] || $user_data["password"] != $_SESSION["password"]) { //if user is not logged in
                header("Location: homepage.php");
                exit();
        } else {
                if ($user_data["admin"] != 1) { //if user is not an admin
                        header("Location: homepage.php");
                        exit();
                } else {
                        echo "<br><center>Current User : " . $user_data["username"] . " (admin)<br><br>";
                }
        }
}?>
<font size="4"><b>Add Wristband (Scan login tag)</b></font>
<br><br>
<form action="settings.php" method="post">
New Wristband Number
<?php //populate input box with previous value if it exists
if (array_key_exists("new_wristband_number", $_POST)) {
        echo "<input type=\"text\" name=\"new_wristband_number\" value=\"{$_POST["new_wristband_number"]}\"> ";
} else {
        echo "<input type=\"text\" name=\"new_wristband_number\" value=\"\"> ";
}?>
<input type="submit" value=" Add ">
</form>
<?php
if (array_key_exists("new_wristband_number", $_POST)) { //if add wristband was submitted
        if ($_POST["new_wristband_number"] == "") { //if input is not blank
```

```
                        echo "Incomplete data<br><br>";
        } else {
                $query = mysqli_query($dbconnect, "SELECT * FROM tag where login = 1;") or die (mysqli_error($dbconnect));
                $login_tag = mysqli_fetch_array($query); //get tag data
                $query = mysqli_query($dbconnect, "SELECT * FROM wristband where tag_id = '".$login_tag["tag_id"]."';") or die (mysqli_error($dbconnect));
                $wristband_data = mysqli_fetch_array($query); //get wristband data
                if (!is_array($wristband_data)) { //if wristband was not scanned
                        echo "Login tag not scanned<br><br>";
                } else {
                        $query = mysqli_query($dbconnect, "update wristband set wristband_number = '".$_POST["new_wristband_number"]."' where tag_id = '".$login_tag_id."';") or die (mysqli_error($dbconnect)); //update wristband number
                        $query = mysqli_query($dbconnect, "update wristband set tag_id = null where wristband_number = '".$_POST["new_wristband_number"]."';") or die (mysqli_error($dbconnect)); //delete temp tag data
                        echo "Wristband added<br><br>";
                }
        }
} else {
        echo "<br><br>";
}?>
<font size="4"><b>Add/Update Tag</b></font>
<br><br>
<form action="settings.php" method="post">
Scanning Wristband Number
<?php //populate input box with previous value if it exists
if (array_key_exists("scan_wristband_number", $_POST)) {
        echo "<input type=\"text\" name=\"scan_wristband_number\" value=\"{$_POST["scan_wristband_number"]}\"> ";
} else {
        echo "<input type=\"text\" name=\"scan_wristband_number\" value=\"\"> ";
}?>
|| New Tag Number
<?php //populate input box with previous value if it exists
if (array_key_exists("new_tag_number", $_POST)) {
        echo "<input type=\"text\" name=\"new_tag_number\" value=\"{$_POST["new_tag_number"]}\"> ";
} else {
        echo "<input type=\"text\" name=\"new_tag_number\" value=\"\"> ";
}?>
Login <input type="checkbox" name="login" value=1>
<input type="submit" value=" Add ">
</form>
<?php
if (array_key_exists("new_tag_number", $_POST)) { //if add tag was submitted
        if ($_POST["scan_wristband_number"] == "" || $_POST["new_tag_number"] == "") { //if input is not blank
                echo "Incomplete data<br><br>";
        } else {
                $query = mysqli_query($dbconnect, "SELECT * FROM wristband where wristband_number = '".$_POST["scan_wristband_number"]."';") or die (mysqli_error($dbconnect));
                $get_tag_id = mysqli_fetch_array($query); //get wristband data
                if (!is_array($get_tag_id)) { //if data doesn't exist
                        echo "Wristband doesn't exist<br><br>";
                } else {
                        if ($get_tag_id["tag_id"] == "") { //if temp data is empty
```

```
                                echo "No tag scanned<br><br>";
                        } else {
                                $query = mysqli_query($dbconnect, "select * from tag where tag_number = '".$_POST["new_tag_number"]."';") or die (mysqli_error($dbconnect));
                                $tag_number_query = mysqli_fetch_array($query); //get tag data
                                if (is_array($tag_number_query)) { //if tag value is taken
                                        echo "Tag number is already in use<br><br>";
                                } else {
                                        $query = mysqli_query($dbconnect, "select * from tag where tag_id = '".$get_tag_id["tag_id"]."';") or die (mysqli_error($dbconnect));
                                        $tag_data = mysqli_fetch_array($query);
                                        if (!is_array($tag_data)) { //if tag number is not in use
                                                $query = mysqli_query($dbconnect, "insert into tag (tag_id, tag_number) values('".$get_tag_id["tag_id"]."', '".$_POST["new_tag_number"]."');") or die (mysqli_error($dbconnect));
                                                echo "Tag number added";
                                        } else { //if tag number is in use (reallocate
                                                $query = mysqli_query($dbconnect, "update tag set tag_number = '".$_POST["new_tag_number"]."' where tag_id = '".$get_tag_id["tag_id"]."';") or die (mysqli_error($dbconnect));
                                                echo "Tag number updated";
                                        }
                                        if (isset($_POST["login"])) { //if user wants a login tag
                                                $query = mysqli_query($dbconnect, "select * from tag where login = 1;") or die (mysqli_error($dbconnect));
                                                $login_tag_query = mysqli_fetch_array($query);
                                                $query = mysqli_query($dbconnect, "update tag set login = 0;") or die (mysqli_error($dbconnect));
                                                $query = mysqli_query($dbconnect, "update tag set login = 1 where tag_id = '".$get_tag_id["tag_id"]."';") or die (mysqli_error($dbconnect));
                                                echo "<br><br>";
                                        }
                                        $query = mysqli_query($dbconnect, "update wristband set tag_id = null where tag_id = '".$get_tag_id["tag_id"]."';") or die (mysqli_error($dbconnect)); //remove temp data
                                }
                        }
                }
        }
} else {
        echo "<br><br>";
}?>
<font size="4"><b>Add Setter</b></font>
<br><br>
<form action="settings.php" method="post">
First Name
<?php //populate input box with previous value if it exists
if (array_key_exists("add_setter_first_name", $_POST)) {
        echo "<input type=\"text\" name=\"add_setter_first_name\" value=\"{$_POST["add_setter_first_name"]}\"> ";
} else {
        echo "<input type=\"text\" name=\"add_setter_first_name\" value=\"\"> ";
}?>
|| Last Name
<?php //populate input box with previous value if it exists
if (array_key_exists("add_setter_last_name", $_POST)) {
        echo "<input type=\"text\" name=\"add_setter_last_name\" value=\"{$_POST["add_setter_last_name"]}\"> ";
} else {
        echo "<input type=\"text\" name=\"add_setter_last_name\" value=\"\"> ";
}?>
<input type="submit" value=" Add ">
</form>
<?php
if (array_key_exists("add_setter_first_name", $_POST)) { //if add setter was submitted
        if ($_POST["add_setter_first_name"] == "" || $_POST["add_setter_last_name"] == "") { //if input is blank
                echo "Incomplete data<br><br>";
        } else {
                $query = mysqli_query($dbconnect, "SELECT * FROM setter where name_first = '".$_POST["add_setter_first_name"]."' and name_last = '".$_POST["add_setter_last_name"]."';") or die (mysqli_error($dbconnect));
                $add_setter_query = mysqli_fetch_array($query);
                if (is_array($add_setter_query)) { //if setter already exists
                        echo "Setter already exists<br><br>";
                } else { //add setter
                        $query = mysqli_query($dbconnect, "insert into setter (name_first, name_last) values('".$_POST["add_setter_first_name"]."', '".$_POST["add_setter_last_name"]."');") or die (mysqli_error($dbconnect));
                        echo "Setter added<br><br>";
                }
        }
} else {
        echo "<br><br>";
}?>
<a href="problemdata.php">Problem List</a>
<br><br>
<a href="userdata.php">User Page</a>
<br><br>
<a href="homepage.php">Logout</a>
<br>
</body>
</html>
```