UNIVESITY OF REGINA: ELECTORNIC SYSTEMS ENGINEERING

Gymtron

The Automatic Weight Training Tracker

Mark Rumpel & Kyle Shaw

April 1, 2015

Table of Contents

1.0 Introduction

In today's world, technology is integrated into almost every area of one's life, whether it be a smart phone, robots used in assembly lines, or other forms of everyday automation, there's no denying that technology is here to stay. However, one area significantly lacking automation is the health and fitness industry, specifically in workout tracking. That being said, there are many fitness apps and devices available that are able to track one's progress, but they all require some sort of user input to gather the necessary data. This is still rather cumbersome and time consuming so we saw a need for a way to automatically track ones exercises and make the results of those workouts viewable later. This need was the driver behind our initial goal for our design project, which was to automatically track a user's exercise with as little input from the user as possible. Through further research, we found there were more applications for a seamless tracking device than just personal fitness. When we spoke to Barclay Dahlstrom from the Kinesiology department, he recommended that such a device could be used to track rehabilitation patients as well as track individual participants in large research studies (personal communication, 2015). Thus, we found that there was indeed a need for a way to automatically track exercises.

To accomplish our goal of automatically tracking a user's exercise, there were several requirements we needed to meet. Firstly, we had to track the movement of the user which can be done by measuring the position of the weight stack on an exercise machine. Next, to ensure that each user is able to view their personal data, we needed to have a way for them to log into the machine using a unique identifier. Additionally, we also wanted to provide some visual feedback in the form of lights to guide the user's form as well as ensure that only reps completed using proper form are logged. Finally, we need to store all of the logged data and present it to the user via a personalized webpage.

2.0 Methods and Materials

2.1 Design Process

With the project defined, we started on the process of designing a physical system that could meet our requirements. First, we had to define the exercise that we were going to track. There are many methods of weight training so in the interest of time we chose a specific method and exercise to focus on. We chose to use a weight stack style machine for performing tricep pushdowns. These machines are designed for the user to complete a specific exercise by lifting a weight stack using a system of pulleys. The movement of the weight stack always follows the same vertical path, and therefore, is easier to track than other weight training methods such as free weights.

Next we broke the project down into several independent functions that could be integrated together to form the desired weight training tracking system. This broke the problem up into smaller pieces, making the project easier to manage. We then

brainstormed different methods to implement each function. With a number of options for implementing each function, we then evaluated each option based on its difficulty to implement, cost, ability to perform the function, and the ease in which it could be integrated with the rest of the system. During this process, some options had to be further researched to make an informed decision. After evaluating each alternative, we put the best solutions together into an integrated system. The following sections will further detail the alternative selection process for each function in the system. A block diagram of the components that we chose can be found in Section C.

2.1.1 Microcontroller and Webserver

In order for our system to operate, we needed both a microcontroller to house the logic that collected our data and provide feedback to the user as well as a webserver to distribute the data to users via a webpage. Instead of implementing these needs using two separate devices we looked into a single product called the Raspberry Pi to do both. The Raspberry Pi is an inexpensive computer the size of a credit card that runs a Linux operating system. It has both networking and GPIO capabilities which made it an ideal choice for our project. Combining these GPIO and networking capabilities in the same box was both cost effective and less complex than trying to integrate two separate devices together. As mentioned, the Raspberry Pi runs a Linux operating system. This is both an advantage and a drawback. Linux is a common operating system therefore there are a variety of open source libraries available, which can be used use to save time while writing code. The drawback to a Linux operating system compared to a traditional microcontroller with no operating system is that it introduces a lot of overhead processes, complicating the execution of time sensitive programs. In the end, we decided that the benefits of using the Raspberry Pi in our project outweighed potential scheduling problems we may encounter because of the operating system, especially given the limited scope of our project.

2.1.2 Rep Tracking

In order to track the reps that a user completes on the machine we required a sensor capable of tracking the position of the weight stack. There are a variety of sensors available that can provide position information. The sensors that we considered were accelerometers, ultrasonic distance sensors, infrared distance sensors, and rotary encoders.

Early in the design process, we were most interested in using an accelerometer. We had no previous experience in using accelerometers but noticed that they have been used in many similar applications that are currently available on the market. To learn more about accelerometers we met with Dr. John Barden, a member of the Kinesiology Department that currently uses accelerometers in his research. In our meeting with Dr. Barden, we discussed how he uses accelerometers to analyze the performance of competitive swimmers. This discussion included the placement of the accelerometer, data sampling techniques, and his procedure for processing acceleration data to analyze the athlete's movements. To supplement this discussion

we conducted further research into accelerometer data analysis techniques and experimented tracking the weight stack movement using a smartphone app that logs the accelerometer data of the phone. This data is shown in a graph in Section E. This test suggested that the periodic motion of the weight stack can be determined directly by using an algorithm that searches for patterns in the acceleration data. We also read a research paper that showed position data could also be accurately derived from the acceleration data by integrating the acceleration data (Slifka, 2003).

After exploring the use of accelerometers, we looked into the possibility of incorporating a rotary encoder into the machine. Rotary encoders make use of a rotating shaft to create a pulse every time the shaft rotates through a given angle. By counting the pulses, you can determine the distanced the shaft has rotated. The best way to use this kind of sensor in our system would be to integrate it into the pulley system of the machine. After trying to source a rotary encoder, it became apparent that to integrate it into the weight stack machine we would have to build a customized mechanical apparatus, further complicating our design.

The final two sensors that we looked into were ultrasonic and infrared distance sensors. The ultrasonic sensor emits high frequency sound waves that bounce off the target object and return to the sensor. The time it takes for the echo to return to the sensor can then be used to calculate the distance of the object. The infrared sensor works in similar fashion, reflecting infrared light off of the target object and then determining the distance to that object by sensing how the infrared light reflects off of the object. Infrared sensors are a lot more sensitive to interference as the colour of the target object and sunlight can affect the sensor. Therefore, we eliminated the infrared sensor from our considerations. Ultrasonic sensor accuracy also depends on the target object material, but varies with the hardness of the target object. So a soft object such as cloth would not reflect as well as a hard object such as metal or wood. The weight stack is metal so we did not think this would present a problem. It must also be noted that the distance calculated using the ultrasonic sensor depends on the speed of sound, which depends on the temperature of the air the sound passes through. Therefore, for the best results a temperature sensor should be coupled with the ultrasonic sensor. We had an ultrasonic sensor available, so we conducted a quick test using it to measure the distance of the weight stack while completing a workout. The results of the test are shown in Section E. With this quick test, we achieved accurate measurements of the full range of movement of the weight stack.

After exploring each sensor, we chose the ultrasonic sensor. This sensor was easy to implement and could easily interface with any microcontroller, as it only requires the generation of a trigger pulse, the measurement of a return pulse, and a simple calculation based on the length of the return pulse. This is much easier to implement than the analysis required to interpret accelerometer data. The data that rotary encoders provide has a similar complexity to the ultrasonic sensor, but was much harder to integrate into the existing weight stack machine than the ultrasonic sensor. In

addition, the ultrasonic sensor cost less than both the accelerometer and any of the rotary encoders that we sourced.

2.1.3 Weight Measurement

One of the key metrics of weight training is the amount of weight that you lift. Therefore, we needed a method to track the weight that the user selects from the weight stack. Again, we considered a number of options including a traditional scale, load cell, and a keypad for manual entry.

A traditional scale is difficult to interface to a microcontroller and is likely to sustain damage if the weights are allowed to drop onto the scale. Mark's previous experience with load cells was not favourable. He found they were difficult to mount in a position that provided reliable results. Using a keypad is simple but requires additional input from the user and is subject to user error. As one of the driving forces in the project was to simplify the process of tracking your workout, we felt a keypad did not fit well with this objective. Therefore, we looked to our classmates for ideas on how we could indirectly measure the weight a user has selected.

This brainstorming yielded a number of results the most promising of which was assigning a unique resistance to each weight in the stack and then somehow measuring this resistance when the weight selection pin is inserted into the weight stack. As the selection pin moves with the stack, we decided to use a microcontroller to measure the resistance and use wireless communication to send this information back to the main microcontroller which would remain stationary to allow for other wired connections. We used wireless communication to avoid any reliability and safety issues related to having a cable wired to the moving stack. Potential problems could include cable failure due to mechanical stress, or cables interfering with the pulley system.

With a strategy to measure the weight, we had to spec a microcontroller that would meet our needs. Before diving into the details of designing a microcontroller system, we looked to see what was commercially available. The search yielded a Bluetooth Low Energy radio and microcontroller combo called the LightBlue Bean. At $30 USD, it was cheaper than anything we could build ourselves and was ideal for our requirements. It includes a microcontroller with a built in analog to digital converter, a library to access the Bluetooth properties of the device, as well as a temperature sensor. For these reasons we chose to use the LightBlue Bean instead of building something ourselves.

2.1.4 User Feedback

Communicating the state of our machine to the user was important as it helps them understand what it is doing and provides confirmation that it is working. We wanted to accomplish this using a very simple interface that was easy to learn and purely visual. We avoided adding any audio feedback to the user as gyms are often loud environments and it is common for people to listen to personal music players while working out.

The simplest feedback system we came up with was to use a strip of RGB LED lights. We could use different colours and brightness to communicate to the user what state the machine is in and guide them to complete proper form. This approach easily grabs the user's attention, inviting them to make use of the machine. Other technologies such as LCD displays are smaller, require the user to take time to read the display, and may include additional complexity if the user has to navigate through a GUI on the display to view relevant information. All of which do not agree with our objective of creating the simplest possible tracking system.

Once we had decided that the RGB LED light strip was the best option, we defined how it would function. An analysis of the exercise we chose revealed that we only needed to communicate five different states to the user. We then assigned a colour and brightness to each state. These states were Resting (red, constant brightness), Moving Up (blue, increasing brightness), Moving Down (blue decreasing brightness), Rep Complete (green, constant brightness), and Idle (all lights off).

2.1.5 User Login

The weight machine could be used by any number of people; therefore, we needed a method to differentiate between users. We decided to create a logon system that required a user to log into the machine before starting their workout. As this is an added step to a normal workout routine, we wanted to allow the user to log in as quickly as possible to maintain a streamlined user experience.

Finding inspiration with pay pass credit card systems we decided to determine how feasible it would be to add Near Field Communication (NFC) to our system for user login. We found that there were a number of modules that allowed for easy implementation of an NFC sign in system, all of which were priced low enough to make it a better choice to buy than to build one ourselves. We evaluated three different modules based on the amount of documentation available for the device and price. The three modules we evaluated were from ITEAD Studios, Adafruit, and DFRobot. All three modules used the same NFC hardware and differed in the supporting hardware that was included for integration with a microcontroller. We decided upon the ITEAD module, as it was not only the cheapest but ITEAD also provided documentation on connecting the module to a Raspberry Pi.

The near field communication (NFC) module was able to read the unique identifier (UID) of a MiFare card through the lid of our outer case and then store the id in a variable. In the code, we wrote it such that the NFC device would poll for a card by entering the login function when the program is in the idle state and waiting for another user to login to Gymtron. The code implemented to read the UID from the card was based on the code provided by libnfc, which is an open source library for NFC devices, this code can be seen in Section C. This code was then modified to fit our specific needs. The login function works such that once a card is detected during polling, the data from the card is read and the UID is stored in a variable and passed back to the main function as the current user. This allows for the current user's ID to be used in the

query when submitting to the database to ensure that data is logged to the right user and can be viewed by that user at a later date. We also considered a screen that would allow users to manually enter their login info but felt it took away from the simplicity of the machine.

2.1.6 User Interface Panel

Another necessary component of any commercial device is the ability to easily interact with and convey information to the user. Gymtron utilizes a simple user interface that consists of two buttons and two LED's. The first button allows the user to increment their current set, and the second button allows the user to logout of Gymtron. We also used a green and a red LED indicate to the user whether a user is currently logged into Gymtron.

2.2 System Integration

After choosing all of the individual components, we need to come up with a plan to integrate them. This involved two main efforts: connecting the hardware together and writing the software to control the hardware. Throughout this process, we built each section of the system and tested it to ensure it worked as expected. Then we started piecing each portion of the system together by adding small sections to the system, testing the system to ensure it still operated correctly, fixing any problems that arose, and then adding in another portion of the system until it was complete. Following this process ensured that any problems that we ran into were easier to solve as we dealt with integration problems one by one.

2.2.1 Hardware Integration

With all of the components selected for our system, we had to develop a strategy to physically connect them all together. The main component in the system is the Raspberry Pi so this process consisted of developing the necessary hardware to connect external devices to the Raspberry Pi. This process was complicated by varying operating voltages of our sensors, and the output current capabilities of the Raspberry Pi GPIO pins. Our general strategy for connecting these external devices to the Raspberry Pi was to design a circuit that would allow the Raspberry Pi to communicate/control the external element and then house all of these circuits on a single protoboard that served as an external interface to all sensors. This board was then connected to the Raspberry Pi via a ribbon cable to simplify the connection process. Before building any circuits on the protoboard we tested them in the lab using a breadboard to make sure they worked as expected. We chose to use a protoboard over designing a printed circuit board, as all of the circuits we required were simple to build. Based on our previous experience we felt confident that we could build them reliably on a protoboard. In addition, choosing to go with a protoboard simplified our schedule for the project, as we did not have to account for the time it takes for manufacturers to receive, build, and ship printed circuit boards.

The following sections detail the integration hardware that was required to connect our system together. For wiring diagrams please refer to Section B.

2.2.1.1 Power

The Raspberry Pi, NFC module, and ultrasonic sensor all required an input voltage of 5V. The Raspberry Pi does provide a 5V output pins that would allow us to power both the NFC device and the ultrasonic sensor if we chose to bring all of our power through the Raspberry Pi using an AC/DC wall adapter that can plug into the Raspberry Pi's micro USB power port. However, the RGB LED light strip is rated for 12V and through testing we determined that we would need to power the light strip with at most 8.5V to achieve a desirable brightness. To avoid having to require two separate power connections, we decided to bring a single 9V power source to the interconnect board to power the lights, and then use a switching voltage regulator to provide a 5V source for the Raspberry Pi, NFC module and ultrasonic sensor. We chose a switching regulator as it is more efficient than a linear regulator, without adding any complexity to the work that we had to complete.

2.2.1.2 HC-SR04 Ultrasonic Distance Sensor

As mentioned in the previous section the ultrasonic distance sensor required a 5V power source. It also operated at a 5V logic level which was not compatible with the Raspberry Pi's strict 3.3V logic level. Therefore, we had to implement a circuit that could convert a 3.3V signal sent from the Raspberry Pi to a 5V signal that could be interpreted by the ultrasonic sensor and more importantly convert 5V signals from the ultrasonic sensor to a 3.3V signal compatible with the Raspberry Pi.

In previous courses, we accomplished this using a BJT as a high side switch of sorts. The circuit works by using the signal from the Raspberry Pi to control the base of the BJT, whose other leads are connected to a 5V source and ground (through a resistor). This solution works but at the time of development, we were deciding between two different ultrasonic sensors, one of which used a single pin for both input and output, driving the need for a circuit capable of bidirectional logic level shifting, something the BJT circuit cannot accomplish easily. Therefore, we went looking for another solution to our level shifting problem. We found our solution from the online hobbyist electronics store Sparkfun. They sell a product that achieves bidirectional level shifting using a single N-Channel MOSFET and two 10K resistors (Sparkfun, 2015). Sparkfun also provided a link to a Phillips application note that detailed the use of this particular solution for use with I2C and included a specification for suitable MOSFETs (Phillips, 1997).

The bidirectional level shifting circuit described by the Phillips application note was ideal for our needs so we decided use it. Sparkfun sells a product that includes four level shifting circuits, but we did not buy it because we only needed a maximum of two circuits and could easily build the circuit described in the Phillips application note ourselves. Therefore, we found a transistor that met the Phillips specifications for ideal

operation and used it in our design. The best transistor that we found was the same transistor that Sparkfun used in their design.

In the end, we did not choose the ultrasonic sensor that used a combined input and output pin. Despite this, we stuck with the bidirectional level shifter design as it offered a more elegant solution to our problem at no extra cost or complexity.

2.2.1.3 LED Light Strip

As the length of the LED light strip was variable and no datasheet was included with the strip when we purchased it, we had to experimentally determine the maximum amount of current the lights would draw. To accomplish this we simply connected the light strip to a lab power supply and measured the current for a variety of voltages. We found that the maximum brightness that was comfortable to look at was with 10V applied to the strip. At 10V, the lights consumed a maximum of 303mA.

The Raspberry Pi can supply a maximum of 16mA from any one of its GPIO pins so we decided to implement a high side switch to control the lights using an N-Channel MOSFET for each the red blue and green LEDs on the strip. We used an N-Channel MOSFET as it has a very low on resistance. A low on resistance translates to almost no power dissipated in the switch itself and a very small voltage drop over the switch which allowed us to use a lower input voltage to the system. This is useful because AC to DC wall adapters are cheaper and easier to find at lower voltages. Another advantage of the N-Channel MOSFET is that it requires almost no current at its gate to bias the transistor to its “on” state. This minimized the current drawn from the Raspberry Pi.

The LED strip design also required brightness control, which was accomplished using PWM. The Raspberry Pi only has two PWM channels and the LED strip has three colours. We decided to use one PWM channel to control the blue LEDs and then use two AND gates to allow the second PWM channel to control the brightness of both the red and green LEDs. The use of PWM also served to provide voltage regulation for the LED strip as it was powered directly from the AC/DC wall adapter input.

2.2.1.4 Bluetooth Communication

As we decided to use the LightBlue Bean for weight measurement, we needed a method to enable Bluetooth Low Energy communication on the Raspberry Pi so that it could communicate to the LightBlue Bean. This turned out to be quite simple to achieve. The only hardware required was a CSR 4.0 USB Bluetooth radio that we plugged into one of the Raspberry Pi’s USB ports. This radio was then easily controlled by installing an open source Bluetooth library on the Raspberry Pi called BlueZ. This library handled all of the low level details required to use the USB Bluetooth radio by providing command line tools. These tools allowed us to complete tasks such as a searching for nearby Bluetooth LE devices, connecting to Bluetooth LE devices, and reading information provided by these devices. The particular USB Bluetooth radio that we

used was recommended by the Raspberry Pi community, as other Raspberry Pi users had success using it in their own projects.

2.2.1.5 LightBlue Bean Resistance Measurement

In order to measure the unique resistance of each of the weights in the weight stack with the LightBlue Bean, we built a simple voltage divider circuit. Pictured in figure 1, one can see that we used a digital output to apply a voltage to the circuit. This gave us the control to power the circuit only when we need to take a measurement from the analog input which is connected between the two series resistances. The variable resistance (R2) is supplied through an audio cable that connects the circuit to one of the unique resistance boxes on the weight stack of the machine. This resistance changes depending on which box the audio cable is plugged into. R1 ensures that a direct short to ground does not occur if the audio cable contacts are shorted together. A direct short to ground is likely to create enough current to destroy to the digital output.

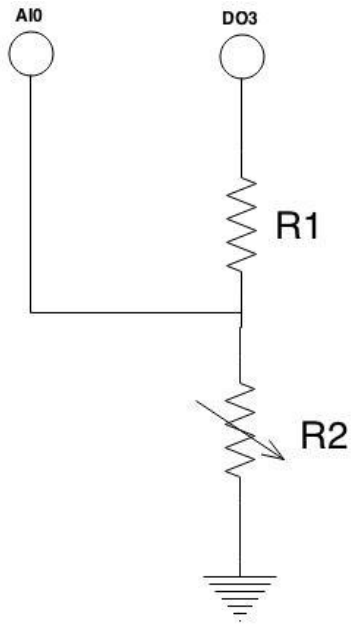


Figure 1

The weight stack contains 20 different weights so we had to pick 20 different resistance values to identify each weight. The analog input is converted into a 10 bit digital value by the microcontroller so the range of possible measurements ranges from 0 to 1023. As each resistance should create a unique voltage, we split this range into 20 different sections, and then converted the middle value of each range into an equivalent resistance. Using the middle value of each range ensured that there would be as small of chance as possible to confuse two different weights. The supply voltage is used as a reference when the analog input is measured so all that is needed is to ensure the voltage over R2 will provide the desired digital value is the desired ratio between R1

and R2 which is the same as the ratio of the desired digital value and the maximum digital value (1023). In addition, the analog to digital converter in the LightBlue Bean microcontroller requires an input resistance of less than 10K Ohms to ensure that the capacitance of its internal circuitry is charged fast enough to provide reliable results. Therefore, we also ensured that the total resistance of our voltage divider circuit did not exceed 10K Ohm. To calculate the resistance needed for each weight we used a spreadsheet, which is included in Section E.

As our system is a prototype, we only built resistance blocks for three weights to save on costs, but used the same resistances as required for the full design.

2.2.1.6 Networking

We implemented a simple Local Area Network (LAN) in order to demonstrate the operation of our webserver. The main component in the LAN was a router that acted as a switch to connect the Raspberry Pi to a laptop. The router also acted as a DHCP server, automatically assigning a unique IP address to each device that connected to the network. Both the Raspberry Pi and the Laptop were connected to the router via Ethernet cables.

2.2.2 Software Integration

We chose to use the C programming language to implement a program that could collect and store data into a database. We chose C because it is the same language that the external libraries we planned to use were written in, and it was a language that we both had previous experience using. To allow both of us to work on the code simultaneously and easily integrate it once complete we decided to write functions for each external device that we had to interface to, and then call these functions from a main control program. We defined the required inputs and outputs of each function before starting so we could test our respective portions of the software without using our partner's complete code.

After the data was collected and submitted to the database, it was made accessible to the user via a variety of web technologies. Both the data collection (c program), and data distribution (webserver) code is discussed in the sections below.

2.2.2.1 Data Collection and User Feedback Program

As mentioned above we implemented a single C program for the control logic of our system. The following flow chart describes the program flow.

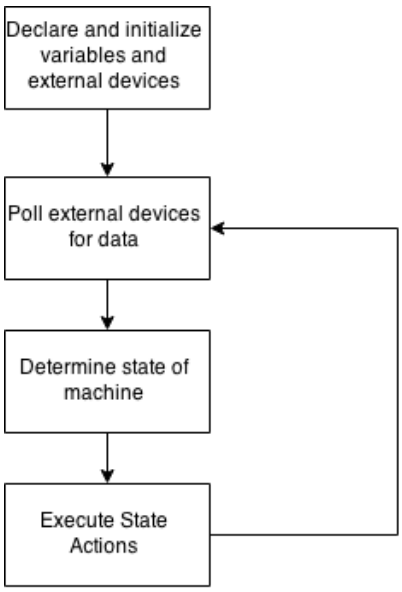
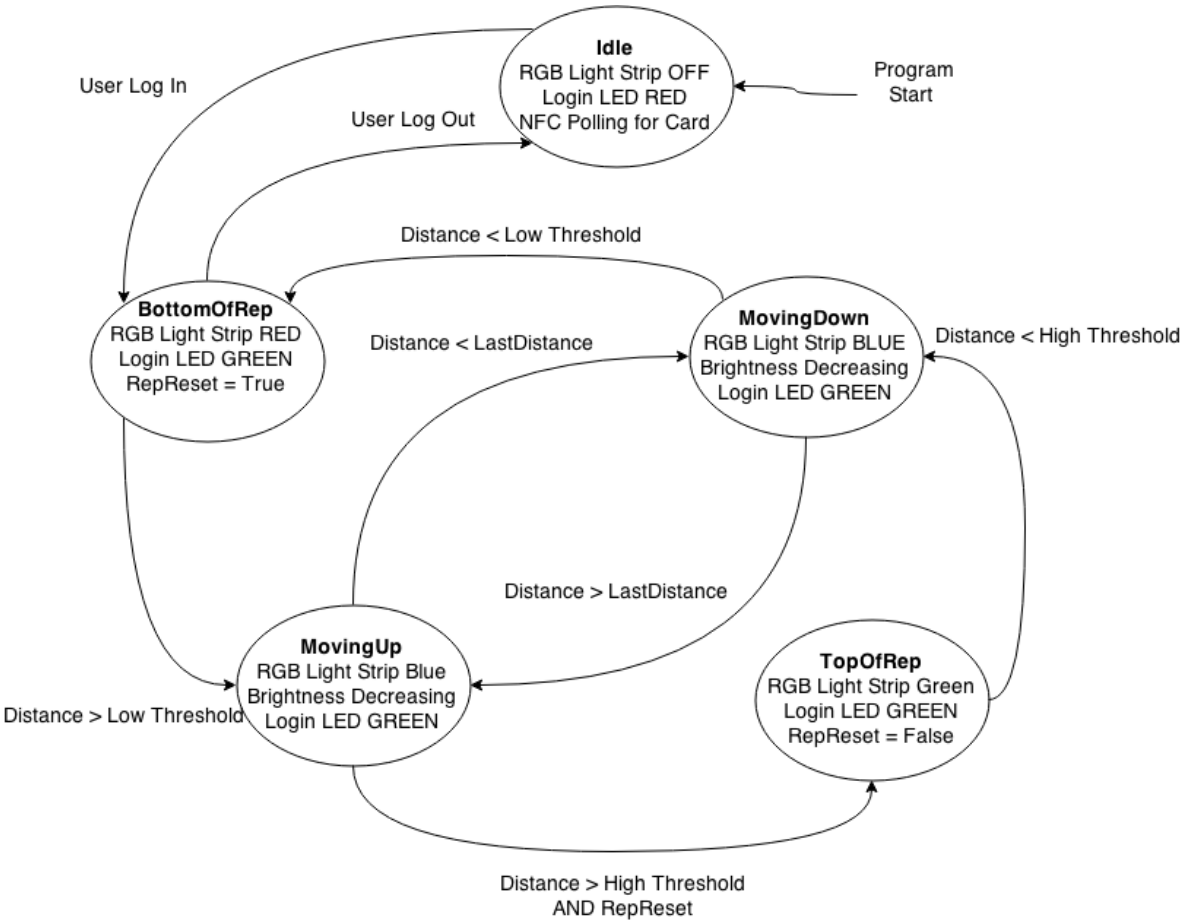


Figure 2

The program begins by declaring the necessary variables as well as initializing any external devices. This includes turning on the Raspberry Pi USB Bluetooth radio, and configuring the Raspberry Pi GPIO. Next, the external devices are polled so that the state of the machine can be determined. If there is no user logged into the machine, the NFC device will be polled until a user card is detected, and then that user will be logged in. If a user is currently logged in, the distance of the stack will be measured. Together the user log in status and the distance to the stack define the state of the machine. The following diagram illustrates how each state of the machine can be achieved.



Inputs	Outputs
Distance	RGB Light Strip (OFF, RED, GREEN, BLUE)
Last Distance	Login LED (RED, GREEN)
User Log In	
User Log Out	

Figure 3

In addition to setting outputs like the LEDs, various data collection operations were also performed during some of the states. The TopOfRep state counted new reps and incremented the current set when necessary. The BottomOfRep state logged out the user when a timeout had occurred or the manual flag had been set through the logout button.

2.2.2.2 Website

For the webpage, our desired result was to allow the user to log in to their account and view all their previous workouts and then choose a workout and view the specific details for each set in that workout. To do this, two things needed to be done, the data needed to be stored in an easily accessible way, and then accessed and displayed to the user via a web interface. To store the data we implemented an SQL database that would store each set as an entry with the following attributes: user id, workout number, set number, number of reps, weight, time, and date. These records

and the data in each record can then be accessed via MySQL queries through php and then displayed through HTML on the webpage. The code we wrote to interface between php and MySQL was loosely based on the example code provided by W3Schools in their tutorials, that code was then heavily modified to perform as we needed it to for our application. The Raspberry Pi was used as the webserver in this case and Apache HTTP Server was installed to allow it to serve html and php files over a network. The website was structured so that the user is first directed to a login page and prompted for credentials, and cannot move past the login page until valid credentials (which are verified by cross referencing with users in the database) are input, the credentials are then stored in a cookie via an intermediate php page. The user is then directed to the workouts page and is presented with all of their previous workouts which can be retrieved by querying the database and selecting all records with a matching user id. The user can then enter the workout they wish to view details for into the form and submit. Once again an intermediate php page is used to store the result of the form in

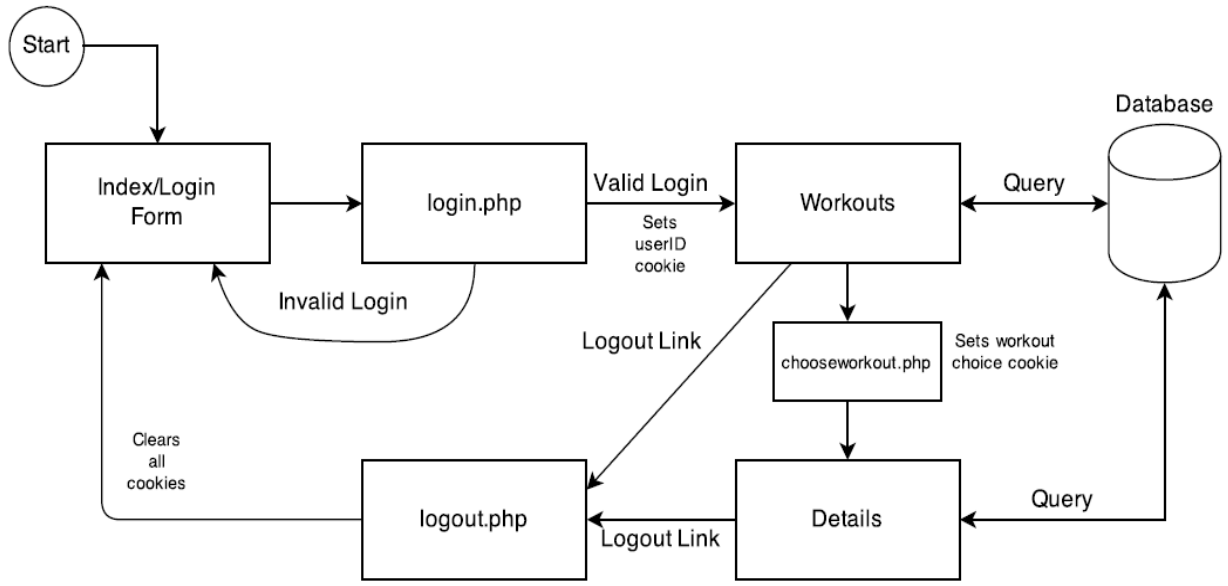


Figure 4

cookie and the user is directed to the details page where the details for that workout are viewable. This is done by querying the database with the data stored in the cookies and .displaying the results returned. The user can then either select a different workout or logout, upon which all cookies are cleared. This is illustrated by the flowchart in figure 4 above. Cookies were used to store the users input over sessions because the data being stored isn't particularly sensitive so security isn't a concern. In addition, since the user is storing login data in the cookie, it is convenient to be able to store the data for a longer period of time to allow for remembering of credentials as cookies allow you to set the expiry time whereas sessions expire when the application is closed. We chose to have the user view the summary of all past workouts and view the details of a single workout rather than viewing all exercises done at once to reduce the clutter on each page and give the user a more succinct and simple viewing experience.

3.0 Results

Overall, our design for the Gymtron did end up performing as expected and meeting all requirements specified. The following sections will elaborate on the specific results of each device and how they performed versus our expectations and overall in our system.

3.1 Card Login

The NFC device was extremely reliable over the course of this project and never gave an erroneous reading. However, there were two occasions in which the NFC device was unable to be found and output an error message. This was a very rare occurrence though and was easily fixed be restarting the executable so we didn't consider it to be a significant problem. As far as outputs, the results of the NFC device were binary, in that it either read the correct UID or it read an incorrect UID so the measure was essentially the ability to read the correct UID. Thus, since it read the correct UID almost every iteration, our results were as expected and excellent overall in terms of reliability.

3.2 Webpage

For the webpage, it was required that the users were able to login, view their past workouts, and then select a workout and view the details for that workout. For the scope of this project it was decided that a very basic webpage would be built. Knowing this the visual effects and design of the page were left very basic and more focus was put on the actual function of the page. The output of the website is again binary in that it will either work or not. In this case, we can measure the website by whether or not it performed all the necessary specifications; which were to allow the user to login, then view basic details for all workouts and then view details for any particular workout. We did this by creating a standard test case that can be performed and if all steps perform as expected, the original design requirements have been met. By this measure, the webpage performed as expected as met all requirements including some extra features such as only being able to access the login page if one is not logged in. The test cases used to confirm the operation of the webpage can be found in Section E.

3.3 User Interface

Originally, we did not have any user interface and relied on timeout periods to increment the set and log the user out. However, we found that relying on each user to have perfect timing was unreliable given that each user's workout can differ in tempo. Additionally, utilizing timeouts left the user with little control over the progress of their workout and resulted in user's blindly trusting the device's set timeout periods. Adding the buttons and LED's may have sacrificed some speed from the overall process but it greatly increased the accuracy of the results and made the experience much more intuitive for the user. The performance of the interface is once again measured in the

standard test case, and both the buttons and the LED's performed as expected in testing. Furthermore, the timeout functionality of both the logout and increment set buttons performed as expected as a button event was simulated once the time limit was reached.

3.4 User Feedback

The LED strip worked as expected, and we received positive feedback from visitors during project day as to its ability to clearly communicate the status of the machine to the user. We also received a lot of feedback from interested parties that they would like to see some sort of more detailed display to communicate information such as your current set/rep or the weight that you used last time. These comments suggest that we may have over simplified our communication with the user, and also highlighted that as engineers we have to ensure we take the needs of our clients into consideration instead of just what we think is best. In conclusion, the RGB light strip was effective in providing a simple feedback system to the user, but could have benefited from a supplementary display for more detailed information.

3.4 Rep Measurement

Our strategy for rep measurement met the requirements we set out to meet, and allowed us to successfully complete our standard test. We had some problems with the quality of the data that we received from the ultrasonic sensor, but after filtering the data, we were able to get reliable results. The particulars of the problems we experienced with the ultrasonic sensor are described in section 4.2. With the filter in place, we were able to measure every successful rep that we completed and only measured a rep after the user had completed the necessary actions (as per the standard test case).

3.5 Weight Measurement

The weight measurement system we designed worked better than expected. The only communication errors occurred when the LightBlue Bean ran out of battery, and our method of calculating the weight from a resistance worked every time. The resistors that we used had a tolerance of 10 percent so our method of resistance selection and weight calculation did not depend on exact resistance values, making it more robust and easy to manufacture. During battery failure, we implemented code that threw an error message, and then used a default temperature and set the weight to negative one. This allowed the rest of the machine to continue operating normally. We also programed the LightBlue Bean in such a way as to limit battery consumption. This was successful as in four months of testing we only had to replace the battery once. To conclude the weight measurement system was robust and supplied accurate information as required by our standard test case.

4.0 Discussion

Overall, we can see from the results that we were indeed successful in meeting the requirements of our design. We were able to login into the machine and track the user's workout accurately while providing feedback to the user. We were also able to store the data in a database and then display each user's data back to them in a personalized fashion via the Gymtron webpage.

4.1 What Worked Well

One of the strengths of using the Raspberry Pi and C for our platform is the abundance of libraries and support available for devices. Having pre-built libraries for things such as GPIO, Bluetooth communication, MySQL interaction, and the NFC device were a great help during the design process and saved us a lot of time over the course of the project. This was especially true for the NFC device which was able to utilize libnfc, which is an open source library for various NFC applications. Libnfc provides basic start up code to help users get their device running initially; we used this code as a base for the login function we implemented for Gymtron. This helped the NFC device function very reliably as the base code being used has already been tested by other users and optimized. Overall, the libraries were very helpful in that they let us focus on our design rather than having to program all of the back end functions we would need to implement our design.

4.2 What Did not work well?

The portion of our design that we had the most problems with was the quality of the ultrasonic sensor data. The sensor provided a lot of erroneous data that had the effect of causing the RGB LED light strip to unpredictably change colour as well as produce the occasional phantom rep. In order to ensure reliable operation of the machine we had to remove this data using a filter. We tried both and averaging filter and a median filter. We found that he median filter worked best the best. These results are illustrated in a series of graphs that are included in Section E. We had to limit the filter size to 3 samples for both the averaging and median filters, as larger sample sizes caused the program to unpredictably freeze. This occurred with a variety of delay times between samples. The programmed froze because the Raspberry Pi would suddenly use 100 percent of the system resources. We observed this using the task manager provided by the Raspberry Pi operating system. We did not have enough time to dig deeper into this problem but suspect it had something to do with our management of the processor's time and the nature the delay function in a multitasking system.

5.0 Conclusions

Overall, this project has been a great learning experience for us in both engineering design as well working in team environment. We have learned how important planning and communication are in major projects as well the importance of the testing process. Even though we were successful in meeting our specifications for

this project, there are still some things we would do differently, as well as a few things we would like to implement in the future if we had more time.

5.1 What would we do differently?

An aspect we would have changed for this design project is the formality of our testing process. Originally, we tested individual components as they were implemented and we really had no formal process for testing each part other than it performed as we were expecting in normal conditions, this disregarded special operation cases and left us exposed to bugs in uncommon cases. To test the entire system, we essentially just tested the machine as if we were an average user, this again left us exposed to bugs in the corner cases. We found this during our demonstration to the faculty, as we encountered a couple bugs we had never seen before. Afterwards, we decided to implement a formal testing case that would cover all cases, and allow us to document performance in each case. Doing this allowed us to find and fix several bugs to improve Gymtron's performance overall. In hindsight, we would have done this much earlier in the design process, which likely would have saved us a lot of time in the end.

5.2 What would we do if we continue?

Lastly, we will discuss a few things that we would like to add to our project if we were to continue working on Gymtron after graduation. Firstly, we would implement a dedicated offsite server to store the data rather than using the Raspberry Pi as the webserver for each device. Doing this would be a much safer and scalable option as all the data could be stored in a single place, and backed up easily in addition to being housed in a safer environment for electronics (i.e. in a datacentre rather than a gym). Furthermore, we would also switch the method of data transfer to the database to Wi-Fi simply to reduce the amount of wiring needed. Next, we would start to implement different types of exercises and machines to make our product more marketable to a wider audience, rather than just having a single exercise available. In addition to more exercises, we would also add more analysis to the webpage. Examples of deeper analysis could be anything from using graphs and charts to display the user's data more visually, to benchmarking users to the population, or adding personalized machine settings to each profile to ensure the most accurate data possible is being collected by accounting for physical differences between users. Finally, we would also change the way we track the movement of the weight stack either by using a rotary encoder, which offers more accuracy than an ultrasonic sensor, or by using an accelerometer, which allows for more complex parameters to be measured but is also much more complex than the ultrasonic sensor. In conclusion, there are many future improvements that could be made to Gymtron, and we would have more than enough work ahead of us if we decide to continue working on Gymtron.