# POWER RACK MANAGER

By

Bartu Alp
Derek Niess
John Quinn

Final Report for ECE 445, Senior Design, Spring 2020
TA: Madison Hedlund

May 2020
Project No. 47

## Abstract

The Power Rack Manager is a device that gym operators can easily attach to a power rack or squat rack to provide an interface for users to schedule time to use the rack. They can put their names in a queue using the mobile app for this product, and will get a notification when the rack is available for them to use.

## Contents

# 1. Power Rack Manager Motivation

Power racks at gyms have a high degree of traffic and would benefit greatly from advances in IOT technology. With that in mind, the original team came up with a solution that technically fulfills this improvement, but there are issues and inefficiencies in their design that left their product in need of significant modification, as we will go into detail about in the coming chapters.

## 1.1 Updated Problem Statement

Power racks are a crucial component to a person's workout as you are able to partake in a multitude of weightlifting exercises. Every gym consists of usually multiple power racks that come equipped with a bench, a barbell and weights. Some of the exercises that you can accomplish with a power rack include squatting, benching, deadlifts, and cleans. Although these are the most common exercises that you are able to do utilizing a power rack, many more exist as well. Oftentimes when people show up at the gym, most, if not all, of the power racks are in use due to their highly versatile nature. People will then have to wait an unknown amount of time for the people that are using them to complete their exercises. People also often do not know whether or not there is a line to wait in to use a particular power rack because they are off doing other exercises. College students suffer through this problem the most as college gyms are often packed with people and lines form quickly to use the power racks. Although this problem has a significant effect on college students and their busy lives, it is not unique to them. Users of any gym have no doubt run into this issue at one time or another. There currently does not exist an orderly way in which gym users are able to know how many people are in line for a particular power rack. This can cause workouts to run extremely long if gym users need to wait extended periods of time to be able to use a power rack and complete their exercises.

## 1.2 Updated Solution

Our solution is a power rack managing device that consists of a system in which you are able to put your name in a queue to use a particular power rack. An LCD display screen as well as a mobile application will display the current user of each power rack as well as the next three people in line waiting to use the power rack. It will also display a color to indicate whether the power rack is being used or not. A red color will indicate that the power rack is currently in use while a green color will indicate that no one is currently using the power rack. There will be a button attached to the power rack that a user will press when they begin using the power rack or when they have completed their exercise(s) on that particular power rack. When the button is pressed, signifying that a user has finished with a power rack, the queue will update on the LCD screen as well as the app. An app notification will also be sent whenever the queue changes for the particular power rack that you are waiting for. There will also be an ultrasonic motion sensor that will sense whenever no one is using the power rack and if no one has been using it for two minutes, it will update the queue accordingly. This is to safeguard against people forgetting to press the button that signifies that they are no longer using the power rack.

For most weightlifters and people who exercise in general, power racks are critical for a successful workout. The versatility that they offer is unmatched by any other weightlifting equipment. Because power racks allow users to perform a wide variety of exercises, they are in extremely high demand. The issue is that gyms only have a limited number of power racks and during busier hours, lines form with people waiting to use the power racks. Sometimes, people have to wait for an absurd amount of time to be able to use a power rack. Personally, we have all had to wait 20 minutes or longer to be able to use a power rack. Waiting an extended period of time can disrupt the routine and rhythm of a person's workout and negatively affect the overall workout. Also, when longer lines form, confusion ensues regarding who is next in line to use a power rack. This occurs because rarely does someone sit around and wait to use a power rack. People usually tell the person(s) currently using the power rack that they are next in line before moving on to do another exercise while waiting their turn. This causes confusion on who is next in line. The Power Rack Manager can solve all of these issues by showing the queue for each power rack on a mobile application as well as an LCD display. It can also allow you to add your name to the queue and signify when you are done with the power rack while updating the queue accordingly.

The main differences of our solution compared to the original group's solution is that we are using ultrasonic sensors and our interface includes an LCD display and a mobile application. Our primary method of detection uses a user button that is pressed whenever someone begins using a power rack or finishes. The original solution uses a large mat with load cells. Our primary usage detection method is much simpler and less expensive as load cells can be easily damaged by large amounts of force. Our secondary form of occupancy detection is ultrasonic sensors while they used infrared sensors. Ultrasonic sensors are better at human movement detection as infrared sensors run into issues in darker settings. With that being said, infrared sensors are generally cheaper, in terms of cost, than ultrasonic sensors. Our user interface is two-fold with a mobile application and an LCD display. Their user interface uses a website and does not include a queue option like our design does. Also, mobile applications are more accessible and easier to use than a website. Our power supply comes from a standard AC wall outlet while they utilize a battery that needs to be recharged often. Lastly, our communication network uses a microcontroller on each power rack which is more reliable, faster and more robust than their centralized, hierarchical structure.

## 1.3 Updated High-Level Requirements

- The mobile application must display the current queue list of every weight rack in the user's gym and the display must be refreshed within five seconds in the event of an update to any queue list.
- Once an occupied weight rack becomes vacant, the queue list of that weight rack should update accordingly (if there is a line), whether the occupancy button is pressed or once the ultrasonic motion sensor detects no movement within a two-minute interval.
- The mobile application must notify the users waiting in the queue within 15 seconds when their chosen rack becomes available for them.

## 1.4 Updated Visual Aid

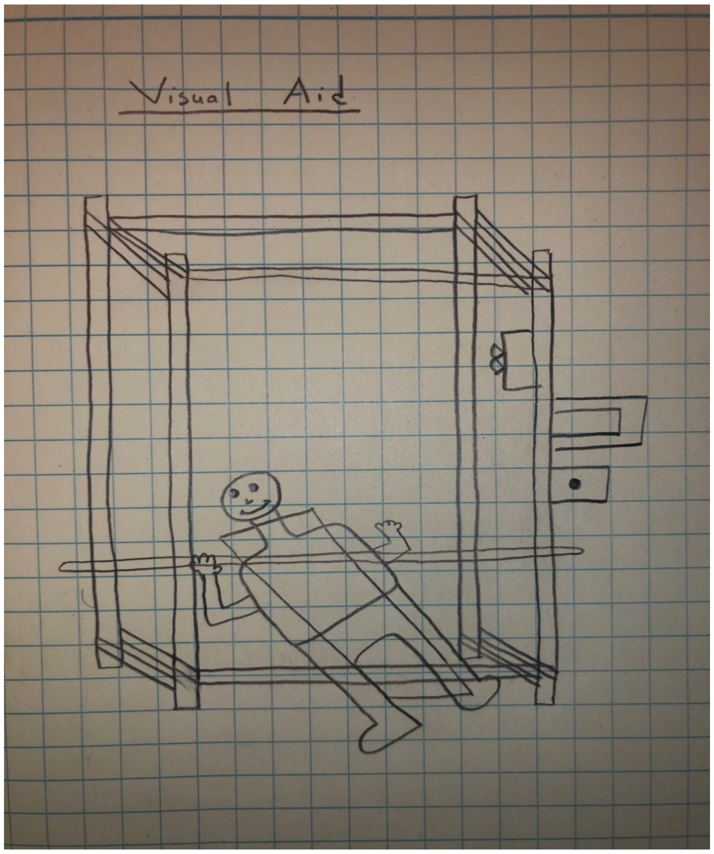


Figure 1. Power Rack Manager visual aid

The above picture depicts the image of a user who is currently doing the bench press movement. The sensors mounted on the right-hand side detects the movement and classifies this rack as occupied. The LCD display shows the name of the current user as well as put a red light to accentuate that it is occupied. Below the LCD display there is the user button which is already pressed, notifying the system that the rack is in use.

## 1.5 Updated Block Diagram

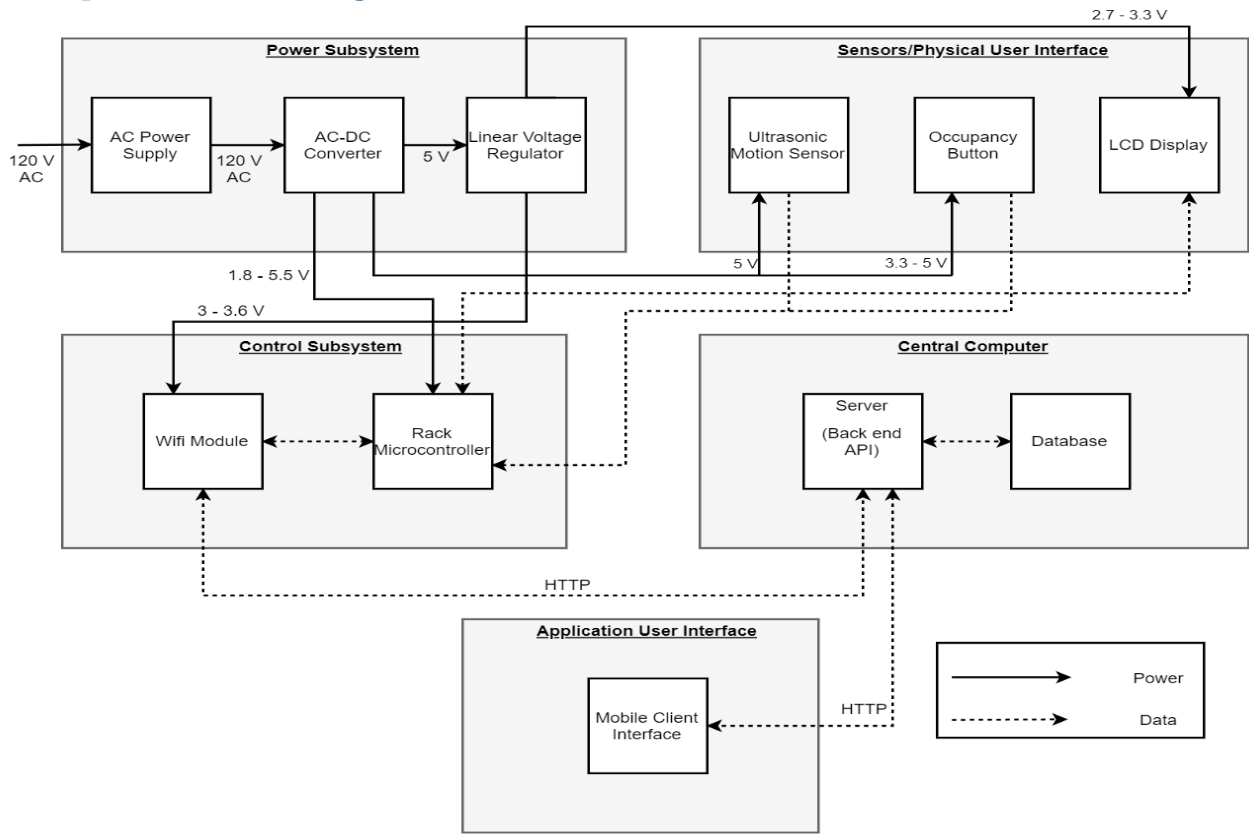


Figure 2: Power Rack Manager Block Diagram

The overall block diagram for the Power Rack Manager is shown in figure 2. The design consists of five subsystems. The power subsystem is responsible for providing adequate power to the components within the sensors/physical user interface subsystem and the control subsystem. The sensors/physical user interface subsystem is responsible for gathering data to be sent to the control subsystem for processing as well as receiving data from the rack microcontroller to be displayed on the LCD display. The central computer interface is responsible for handling and storing the processed data from the control subsystem and transmitting this data to the mobile client interface within the application user interface subsystem. The application user interface subsystem is responsible for receiving and displaying the correct data on user's mobile devices as well as communicating the displayed information to the server within the central computer subsystem.

## 2 Power Rack Manager Implementation

### 2.1 Power Supply Simulations

We performed simulations to ensure that we will be able to adequately supply our Power Rack Manager given the standard 120 V AC wall outlet supply. We were unable to recreate the exact model of the AC/DC converter that we will be using (IRM-15-5) in the LTSpice software due to lack of information from the datasheet. However, we were able to find an LTSpice circuit that functions the same as our model with minor adjustments. This model is an isolated AC/DC converter utilizing a transformer and feedback to maintain a 5 V DC output. This 5 V output is exactly the same DC output that the IRM-15-5 model would output as well. To simulate the current that we require for the components of our Power Rack Manager, a constant current source was added to the output of the converter at 269 mA. This is the maximum current that will be drawn by the ultrasonic sensors, occupancy button, LCD display, microcontroller and wifi module. With changing the inductance of the third output winding of the transformer within the converter, the simulation circuit is shown in Appendix B - Figure 3 [5].

As shown in Appendix B - Figure 3 [5], our output load is represented by a current source that is denoted as Iout. Output capacitors are attached as well to reduce the voltage ripple of the output voltage. Because the output voltage was originally 5 V DC at an output current of 2.2 A, the inductance of the third winding of the transformer needed to be increased to 4 µH from 3.9 µH to achieve the 5 V DC output that we require at a smaller current of 269 mA. The results of the output voltage can be seen in figure 4 as we ran a transient simulation from 0 to 15 ms.

In Appendix B - Figure 4, we are able to see that the output DC voltage takes approximately 10 ms to achieve stability. The output voltage is originally at -0.6 V before beginning to increase at 6.5 ms. This negative voltage will not be damaging to any of the components as they require a certain level of positive voltage to turn on and conduct. The voltage waveform spikes just before 7 ms but this voltage will also not be damaging to any components as the components that require 5 V DC have an adequate threshold to be able to sustain this voltage. The wifi module and LCD display require a lower voltage than 5 V DC to operate and will be protected from this higher voltage by a linear voltage regulator. To see where the output voltage is after stability is achieved, we need to zoom in on this waveform at anypoint after 10 ms. This can be seen in Appendix B - Figure 5.

In Appendix B - Figure 5, the output voltage is at 5 V DC and will be able to adequately power our components given a current of 269 mA. Voltage ripple is always a concern when converting from AC voltage to DC voltage as this fluctuation can cause issues when DC voltage is required as a power supply. The output capacitors are used to reduce this ripple and we see that the ripple is sufficiently small at about 2 mV. This ripple is sufficiently small enough to not cause any issues when powering our components that require 5 V DC. However, because the LCD display requires an operating voltage of 2.7 to 3.3 V DC and the wifi module requires an operating voltage of 3.0 to 3.6 V DC, we need to also attach a voltage regulator to the output of the AC/DC converter. Once again, the model of the voltage regulator that we will be using is unable to be converted into LTSpice so an appropriate replacement was selected. The linear voltage regulator used in this simulation outputs a DC voltage of 3.3 V while being able to sustain 200 mA of current. This output voltage is the same as the actual voltage regulator that we would be using while being able to sustain the 200-mA current that the LCD display and wifi module require. The complete power supply simulation circuit with the voltage regulator added is shown in Appendix B - Figure 6 [5].

In Appendix B - Figure 6 [5], the current needed to supply the occupancy button, ultrasonic sensors and microcontroller is signified as Iout1 at 69 mA. To represent the current that the LCD display and the wifi module require, we denote a current source as Iout2 at 200 mA as the output of the linear voltage regulator. A transient simulation of the output voltage of the voltage regulator is shown in Appendix B - Figure 7.

In Appendix B - Figure 7, the output voltage of the regulator is negative at approximately -2.5 V DC before increasing at approximately 6.5 ms. This negative voltage output is not an issue for the LCD display and wifi module that are being supplied by the regulator due to internal protection of these components as well as the components not turning on until a positive threshold voltage is reached. The output voltage stabilizes just before 7 ms and we are able to zoom in on this waveform to see where this voltage level is actually at. This is shown in Appendix B - Figure 8.

In Appendix B - Figure 8, we see that the voltage level is just under 3.3 V DC which is sufficient to supply both the LCD display and wifi module while sustaining a DC current of 200 mA. We need to ensure that the voltage remains at 5 V for the other components when the voltage regulator is added. This AC/DC output voltage waveform is shown in Appendix B - Figure 9.

In Appendix B - Figure 9, we see that the output of the AC/DC converter is approximately 5 V DC with a small ripple of 3 mV. This voltage will be able to supply the microcontroller, occupancy button and ultrasonic sensors while being able to sustain the 69 mA of DC current that these components require. These simulations show that our AC/DC converter will be able to adequately supply all of our components and that the voltage regulator will be able to drop the voltage down effectively to supply lower rated voltage components.

## 2.2 Tolerance Analysis

The reliability of our design depends primarily upon the accuracy of the ultrasonic sensors. More specifically, the fusion of sensor data must sufficiently explain the state of the system's environment. We must start by describing what the environment is and the assumptions of this environment. The environment our system is designed to function properly in is any section of a gym with barbell power cages or power racks. We are not considering smith machines. That said, our design would likely function properly if applied to a smith machine. Given this detail, we note that an Olympic barbell, the most popular type of barbell, is 52 inches between sleeves, and we can assume that racks compatible with this barbell are anywhere from 48 inches to 50 inches in width, 40 inches to 50 inches in depth, and always at least 80 inches in height [4]. Our last assumption is that a user is deemed to be using a rack if they are within a radius of half of the width of the rack centered at the midpoint of where the barbell is mounted. We reason that this is practical because the user will be performing reps close to where the rack arms are, and the user will be re-racking the barbell after each set anyway.

In order to determine how many ultrasonic sensors to use to cover the circular area shown in Appendix B - Figure 10, we must calculate the effective surface area measureable from a single ultrasonic sensor. We start with the assumption the sensor packet will be at a height of 80 inches, which any rack should be able to accommodate. If we ignore the added surface area caused by generating a conic beam at an angle and instead assume that the surface area forms a perfect circle--and we can make this assumption to no negative effect because an angled conic beam is always larger than a perfectly perpendicular beam--then we can calculate the radius of the coverage of a single ultrasonic sensor using the following equation:

$$r = h \tan(\text{beam}) \qquad (1)$$

Using equation 1, we input a height h of 80 inches and a beam angle of 15 degrees, as mentioned in the ultrasonic sensor datasheet, to find that a single ultrasonic sensor generates a minimum effective coverage with a radius of 21.44 inches [8].

With the knowledge of how much area a single ultrasonic sensor can cover, we can now determine how many ultrasonic sensors are needed to detect movement within the above radius around the barbell. This problem can be reduced and compared to a set of special covering problems that find the largest coverable square given an input number of unit circles [14]. To do so, we must first find the precise factor needed to relate the side of the covered square to the radius of the circle to account for the fact that the covering problems we will be comparing our own problem to are normalized:

$$s = f*r \qquad (2)$$

Plugging in a radius of 21.44 inches and side length of 50 inches, we get a factor of 2.33. With this known, we can compare it to the following chart to determine which covering problem, defined by the input number of unit circles, will solve our own problem:

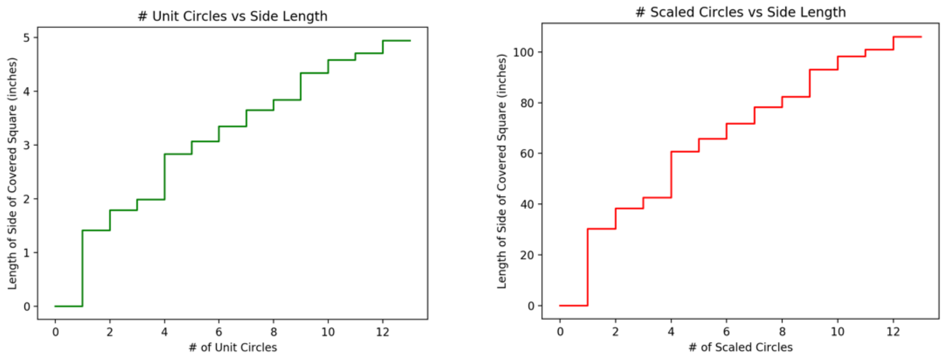


Figure 11: The plot on the left presents the relationship between the number of unit circles and the maximal side length of a square coverable by those circles. The plot on the right presents the same relationship, but in the case where the radius of the circle is not 1 inch, but 21.44 inches. Note that the two plots are step functions that are right continuous.

As one can see, we need a minimum of 4 circles to cover our target square. Given this knowledge and upon further investigation into the solution to the covering problem where the input is 4 circles, we can determine the coordinate distance of the where the center of the beam should be aimed at on the ground relative to the horizontal position of the sensor packet:

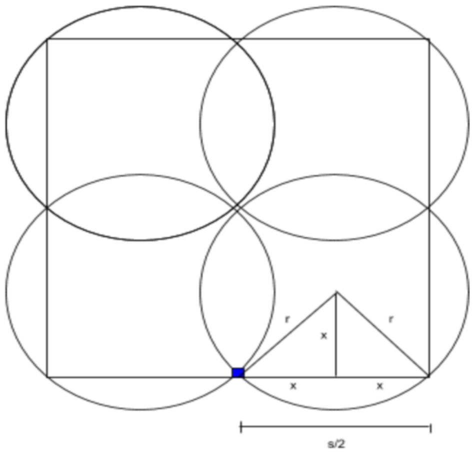


Figure 12: The square is just large enough to inscribe within it the critical circular area in which movement must be detected, so if the square is covered, then the critical circular area is also covered. Each circle has a radius of r inches. Each circle overlaps a corner of the square, and all of them join at the center of the square. The blue dot symbolizes where the sensor packet is located.

To find the (x, y) coordinates of the center of each circle, we must solve for x in figure 3. Using the Pythagorean theorem, find that x is 15.16 inches. We can then plug this value into the below equations to get the coordinates:

$$(x, y)_{\text{top left}}=(-x, s-x) \qquad (x, y)_{\text{top right}}=(x, s-x) \qquad (3)$$
$$(x, y)_{\text{bottom left}}=(-x, x) \qquad (x, y)_{\text{bottom right}}=(x, x)$$

The last step is to determine the angle to point each sensor at along the horizontal and vertical plane. For the horizontal plane, we calculate H such that 0 degrees is perpendicular with the surface holding the sensors:

$$H, \text{bottom left}= \arctan(x /-x) \quad H, \text{bottom right}= -\arctan(x / x) \qquad (4)$$
$$H, \text{top left}= \arctan((s-x) /-x) \quad H, \text{top right}= -\arctan((s-x) / x)$$

We determine V for the vertical plane such that 0 degrees is horizontal with the ground. Since the height of the sensors will be 80 inches, we can calculate the vertical angles using the following equations:

$$V, \text{bottom left}= \arctan(x / 80) - 90 \quad V, \text{bottom right}= \arctan(x / 80) - 90 \qquad (5)$$
$$V, \text{top left}= \arctan((s-x) / 80) -90 \quad V, \text{top right}= \arctan((s-x) / 80) -90$$

The two issues to be addressed in our analysis are:

1. Ensure that the inaccuracy of the sensors do not throw off the decision that movement is detected.
2. Ensure that the sensors cover the critical area of the power rack.

Issue one is trivial. Since we do not care about the distance measurement itself but the detection of any small change in this distance measurement, we can account for the inaccuracy of +/- 3mm of each sensor by filtering out any change in distance less than 6mm [8]. Regardless of the exercise, lifting a barbell requires a movement that far exceeds 6mm. Taking the average female upper arm length of 13.62 inches, or 345.9 mm, as a reference for what can reasonably be considered the minimal displacement an exercise would likely require (leg barbell exercises almost always require more movement than upper body exercises), the sensors should theoretically detect movement 57 times over.

For issue two, we can see that the problem boils down to properly calculating the orientation of the sensors and physically placing those sensors such that this angle is met. Let's go back to how we solved this when we reduced our problem to the previously mentioned covering problem with 4 circles. If our sensors are perfectly oriented in the way we calculated it, we'd have a square coverage with a side length of 2.828 * 21.44 = 60.63 inches. This creates a 60.63 - 50 = 10.63-inch buffer in the event the sensors were not properly oriented.

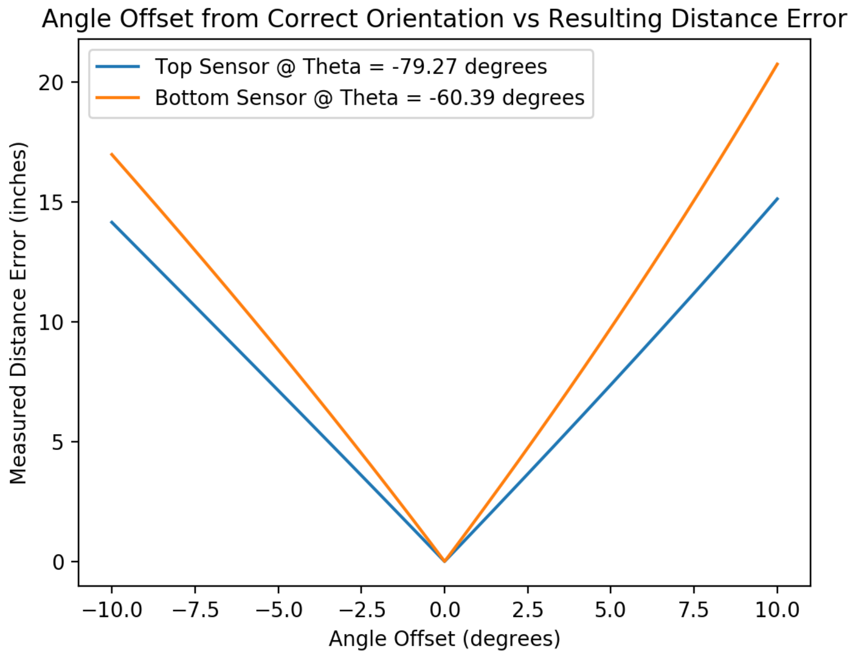


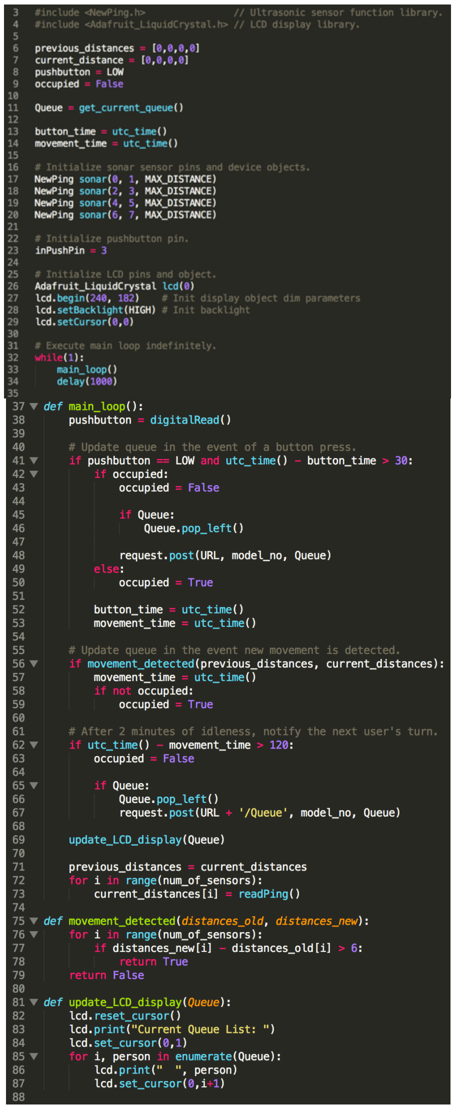
Figure 13: This is a plot showing the effect of an applied angle offset when physically attempting to orient a sensor correctly based on whether the sensor is meant to point at the center of a top circle or bottom circle in the previously mentioned covering problem. Note that an error from the theoretical distance between the sensor and targeted position on the ground is greater than or equal to 10.63 inches at an angle of roughly 5 degrees or higher.

In order for issue two to fail, a sensor would have to be misoriented by an angle of at least 5 degrees from its intended angle, regardless of which circle area the sensor is meant to cover. Given that even the crudest inclinometer has an accuracy of at most 2 degrees, misorienting the sensors should not be a problem [13].

## 2.3 Software Implementation

Since we had no means of acquiring the hardware components of our design within 10 days, the entirety of our implementation of this project is strictly software-based and . There are a few functions that needed to be implemented in order for this project to operate:

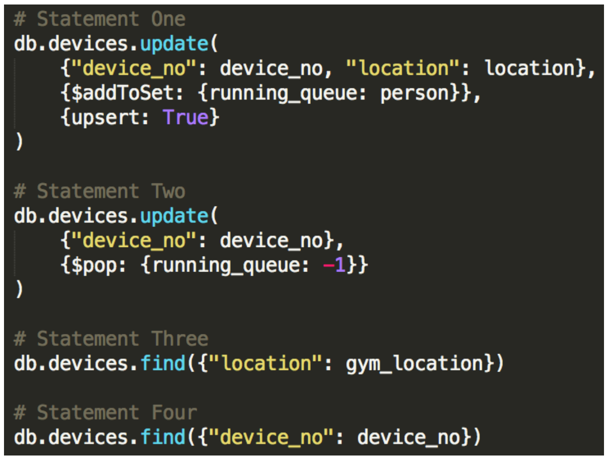
1. A process to update the LCD display with new queue modifications.
2. A process to poll and analyze the push button and ultrasonic sensors.
3. An initialization process to instantiate all peripheral components.
4. A set of NoSQL commands for handling data needed across the mobile app UI, backend API and microcontroller.



```
#include <NewPing.h>                // Ultrasonic sensor function library.
#include <Adafruit_LiquidCrystal.h> // LCD display library.

previous_distances = [0,0,0,0]
current_distance = [0,0,0,0]
pushbutton = LOW
occupied = False

Queue = get_current_queue()

button_time = utc_time()
movement_time = utc_time()

# Initialize sonar sensor pins and device objects.
NewPing sonar(0, 1, MAX_DISTANCE)
NewPing sonar(2, 3, MAX_DISTANCE)
NewPing sonar(4, 5, MAX_DISTANCE)
NewPing sonar(6, 7, MAX_DISTANCE)

# Initialize pushbutton pin.
inPushPin = 3

# Initialize LCD pins and object.
Adafruit_LiquidCrystal lcd(0)
lcd.begin(240, 182)    # Init display object dim parameters
lcd.setBacklight(HIGH) # Init backlight
lcd.setCursor(0,0)

# Execute main loop indefinitely.
while(1):
    main_loop()
    delay(1000)

def main_loop():
    pushbutton = digitalRead()

    # Update queue in the event of a button press.
    if pushbutton == LOW and utc_time() - button_time > 30:
        if occupied:
            occupied = False

            if Queue:
                Queue.pop_left()

            request.post(URL, model_no, Queue)
        else:
            occupied = True

        button_time = utc_time()
        movement_time = utc_time()

    # Update queue in the event new movement is detected.
    if movement_detected(previous_distances, current_distances):
        movement_time = utc_time()
        if not occupied:
            occupied = True

    # After 2 minutes of idleness, notify the next user's turn.
    if utc_time() - movement_time > 120:
        occupied = False

        if Queue:
            Queue.pop_left()
            request.post(URL + '/Queue', model_no, Queue)

    update_LCD_display(Queue)

    previous_distances = current_distances
    for i in range(num_of_sensors):
        current_distances[i] = readPing()

def movement_detected(distances_old, distances_new):
    for i in range(num_of_sensors):
        if distances_new[i] - distances_old[i] > 6:
            return True
    return False

def update_LCD_display(Queue):
    lcd.reset_cursor()
    lcd.print("Current Queue List: ")
    lcd.set_cursor(0,1)
    for i, person in enumerate(Queue):
        lcd.print("  ", person)
        lcd.set_cursor(0,i+1)
```

Figure 14: Initialization code block and main program procedure

Functions 1 and 2 are carried out consecutively and repeatedly within the main loop of the microcontroller. Text can be output and displayed to each line in the LCD screen using a simple function provided in the Adafruit LiquidCrystal library. This makes it easy for us to loop through each row and print the next person in the queue list. We created a function that prints the current queue to the display using this library at the end of every iteration of the main loop shown in figure 14, which easily executes within 5 seconds. As such, the part of the implementation helps fulfill requirement one to update the screen within 5 seconds. The push button and ultrasonic sensors are polled once every iteration in the main loop of our system. If we find that the button was pressed while occupied, we update the queue by removing the current user and send an update to the server. If we find that no movement has been detected within 2 minutes, the queue will be updated following the same logic. These two sub-tasks fulfill requirement two.



```
# Statement One
db.devices.update(
    {"device_no": device_no, "location": location},
    {$addToSet: {running_queue: person}},
    {upsert: True}
)

# Statement Two
db.devices.update(
    {"device_no": device_no},
    {$pop: {running_queue: -1}}
)

# Statement Three
db.devices.find({"location": gym_location})

# Statement Four
db.devices.find({"device_no": device_no})
```

Figure 15: All NoSQL commands needed for the mobile UI and backend API

For function 3, the initial setup of peripheral objects, timers, and global variables can be found in figure 15. The initialization process primarily involves specifying in the software which microcontroller pins are related to a peripheral in our design. In doing so, the software knows where to poll for signals. We also set timers for the pushbutton and ultrasonic sensors. We want to make sure that the pushbutton is not spammed, so once it is pressed, we must wait 30 seconds before processing future presses. We also want to keep track of idleness in the power rack. If 2 minutes have passed, we want to consider the power rack vacant and update the queue accordingly.

For function 4, the bulk of the code for the mobile app and backend API is already available as easily installable frameworks. Therefore, we simply had to focus on building the database and figuring out the query and modification statements that will be needed for the mobile app and backend. Since we are using a NoSQL database, the schema of our database is implemented on the fly when we make an insert command. The set of needed statements are shown in figure 15. Our database only needs one collection, "devices", which has keys for the unique device number, the gym location, and running queue list. Statement one is used to either initialize a new microcontroller in the database, or add a person to the queue. This is called by the mobile UI when a user requests to be put on the list. Statement two is used by the microcontroller to remove a user from the list when it has determined the user is no longer using the power rack. Statement three is used by the mobile UI to fetch the queue lists of all power racks at the user's gym. This statement is particularly important because SwiftUI will use this statement to repeatedly refresh its display of the running queues of each power rack at a gym, thus helping to fulfill requirements one and three. Statement four is used by the microcontroller to grab the running list of the power rack in the event of a reboot.

## 2.4 Component Costs

In table 1, the component costs are shown for the implementation of the Power Rack Manager for one power rack. This total cost comes out to be $139.84 which is relatively inexpensive. Of course, installing the Power Rack Manager for multiple power racks will cost more money but it is likely still manageable for gyms to install and operate. The original design's total component cost is $457.78 per power rack so our design is substantially more cost effective as it is essentially a third of the cost with more features and functionality.

Table 1. Component Costs

| **Description** | **Manufacturer** | **Part #** | **Quantity** | **Cost** |
|---|---|---|---|---|
| 5 V AC/DC Converter | Mean Well USA Inc. | IRM-15-5 [6] | 1 | $10.22 |
| 3 V Voltage Regulator | Analog Devices Inc. | ADP123AUJZ [7] | 1 | $1.11 |
| Ultrasonic Motion Sensor | Adafruit Industries LLC. | HC-SR04 [8] | 4 | $3.95 |
| Occupancy Push Button | Keyestudio | KS0029 [9] | 1 | $3.50 |
| Rack Microcontroller | Microchip Technology | ATMEGA328 [11] | 1 | $1.34 |
| LCD Display | Display Visions | EADOGXL240N-7 [10] | 1 | $50.92 |
| Wifi Module | Sparkfun | ESP8266 [12] | 1 | $6.95 |
| PCB | N/A | N/A | 1 | ~$40.00 |
| Mounting Costs | ECE Department | N/A | N/A | ~$10.00 |
| **Total** | | | | **$139.84** |

# 3. Power Rack Manager Conclusions

## 3.1 Implementation Summary

Even though we weren't able to actually build this product, we did our best to prove that our design will have a very high probability of real-life implementation. In order to prove this, we first started off with tolerance analysis. For tolerance analysis, Derek found the necessary qualifications the ultrasonic sensors must have in order to work within a practical margin of error. He also came up with the calculations proving if and how our design's main functionality can be implemented. Once the calculations were worked out, Bartu researched suitable components for the whole design that would fulfill the necessary functionalities of the product as well as the constraints found during tolerance analysis. After choosing all of the components, John was able to put all of the components into LTspice where he simulated our overall design to test how well our design functions. For the software-side of our product, Derek then implemented the database interface and wrote the pseudocode for the MCU to determine if there would be any glaring issues in the systemic execution of our design once it is translated into physical hardware.

We did all of these in order to prove that, in a real-life implementation, this design would work. To reiterate, since we did not have the chance to physically build this design, we need to bear in mind the uncertainties mentioned in 3.2. After performing a lengthy series of tests, simulations and theoretical implementations, it is clear that this design would effectively and seamlessly translate into a physical product.

## 3.2 Unknown, Uncertainties, Testing Needed

We were unable to complete a great deal of the building phase of this project due to a vast amount of limitations. First, we were not able to put in an order to receive all of the components necessary to build the Power Rack Manager. Due to this, none of the hardware was able to be assembled and tested. Also, if we were able to obtain the necessary components, we would still require the senior design lab to assemble these components as well as test them to ensure proper operation. We would need access to the soldering stations to solder the required components onto the PCB. To properly test the power subsystem, we would need to plug the 120 V AC wall outlet into the input voltage pin of the AC/DC converter and probe the output with a wattmeter to determine the converted DC voltage and ensure that it is 5 V DC. Going one step further, we would need to use an electronic load attached to the output of the converter and have it set to 269 mA. This electronic load serves to represent the maximum current that the components will draw and will allow us to unit test the converter to meet our needs. We would then need to probe the output of the converter to ensure that the output remains at approximately 5 V DC. Then, we would add in the voltage regulator to the output of the converter and use two electronic loads to represent the 69 mA that are supplied at 5 V DC and the 200 mA that are supplied at 3.3 V DC or the output of the regulator. We would probe the output of the converter and ensure that it remains at 5 V DC and we would also probe the output of the regulator and ensure that it is approximately 3.3 V DC. We would also need to calibrate the ultrasonic sensors and ensure that they are able to detect small movements. To do this, we would use a DC power supply in the lab as a power source so that we are able to unit test the sensors.

There are also a number of uncertainties that we are unable to resolve without access to the lab. The first uncertainty involves our microcontroller and the actual refresh rate that it can sustain regarding certain tasks. Given that the microcontroller has a single core processor, each task will be competing for processing power and they will have to be executing consecutively in one way or another. Because of this, the refresh rate of some tasks are bounded by the speed at which other tasks execute. For instance, the cumulative time spent checking the ultrasonic sensor signals and occupancy push button signal, updating the queue, and communicating with the server via API calls directly affects the best-case refresh rate of the LCD display because the function that updates the LCD display must be called after all of these tasks have finished. Another constraint that we need to verify is that the main loop is executing to completion quickly enough to pick up any flaggable changes in input signals. For example, we read the current value of the I/O pin of the microcontroller that is connected to the occupancy push button every iteration of the main loop. If this button is pressed too quickly or the loop is executed too slowly, the pulse of the button press will not be logged into a variable, so button press will not be registered. This also applies to the ultrasonic sensors when registering deviations in distance measurements that are used to detect motion. If the main loop is executing too slowly, there is a possibility that quick changes in distance detected by the ultrasonic sensor will not be read and logged by the microcontroller. While these unknowns are likely to be a non-issue, the only way to confirm this is if we wired up our components, uploaded our code, and tested them in the lab.

## 3.3 Ethics and Safety

Even though our project is not considered to be highly ethical at first glance, we believe that we are tackling a common problem for many sports enthusiasts. Our product intends to minimize the time lost in a gym as well as providing a fair distribution of equipment for all the gym members in the world. So, in other words, we are trying to create a product that would maximize the efficiency of our most important asset, time. We believe that a product/design that intends to create a fair environment for its users while minimizing the time they are losing is an ethical product.

To minimize the time lost for gym members, we are implementing an app that would notify its users about the vacancy of the racks in their gyms. This way users will be able to learn from the app whether or not there is an empty rack for them to workout. We will be totally honest and realistic while processing our sensor data to be as accurate as possible for each rack's vacancy. We believe this fulfills the 3rd clause of IEEE Code of Ethics which states "to be honest and realistic in stating claims or estimates based on available data" [1].

In the case when all the racks are occupied, we will have a queueing system that would allow the users to get in line for a specific rack. This way users won't lose time waiting for a rack to open up as they will be able to estimate when their rack will be vacant and schedule their time accordingly. In our experience, we have seen many individuals who waited for a rack to open up and when it was finally vacant, someone else just showed up and acted fast to steal their queue. Since our queuing system will provide an honest and fair way of equipment allocation, we fulfill the 8th clause of IEEE Code of Ethics which states "to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression” [1].

For a project like this, we, as a group, considered all the possible safety hazards that could happen in the desired setting and took the necessary precautions and measures to make our design/product as safe as possible. This approach aligns with the 1st clause of the IEEE Code of Ethics which states “to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment” [1]. For safety precautions we focused on two possible scenarios; our sensors sending out wrong signals to the control subsystem and having liquid spill over our sensor and power subsystems.

In the first scenario when our ultrasonic sensor sends out the wrong signal, our product won’t be able to determine the vacancy of the rack properly, resulting in the lost time of our customers. For this case, we are considering the sensor itself to fully function. Since the sensor is fully working, the only scenario where it can send a wrong signal is if the height of the rack makes the ultrasonic sensor out of the rack’s reach. To tackle this possible safety hazard, we chose an ultrasonic sensor that can detect movement within 6.5 meters. We believe this is more than enough since there are not any gym racks that are higher than 6.5 meters.

For the second case, where a liquid spills over either the sensor or the power subsystem, there are possibilities of fire and getting shocked. Since gym is a place where the users sweat a lot while also consuming a lot of liquids whether it is water or something else, there is always the possibility of spillage. To minimize the possibility of this incident, the first thing we did was to place all the sensor subsystems on top of the rack. As we can see from the physical design drawing in figure 1, we placed the sensors away from most people’s reach. However, since our product needs to be connected to a power outlet providing 120V, we also needed some protection close to the ground. This is why all of our wires/cables going to the power outlet will be covered with “Medium Capacity Cord Covers” [2].

Looking at the previous group’s design, we also found a possible safety hazard. Since they implemented load cells below the rack, there was a possibility of the rack becoming unbalanced. Changing this with a button that is placed next to the sensor subsystem, cleared away the possibility of this incident.

We, as a team, believe that there is always room for improvement and will enhance our product/design by being open to any feedback and comment from our users. We believe this approach fulfills the 7th clause of the IEEE Code of Ethics which implies “to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others” [1].

## 3.4 Project Improvements

During the idea creation process, we came up with a lot of different features for the Power Rack Manager, however, chose not to implement some of those because of the realistic timeline we had. However, looking back, if we had a year to complete this project rather than just a few weeks, we would have added some more features to serve a better product to our customers.

The first improvement that we can add to our existing project would be expanding the user interface to the LCD board. By expanding the user interface, we mean being able to queue yourself into a specific rack by just tapping on the screen. The LCD display would effectively be set up as a touch screen in this instance and allow you to enter your name to the queue. This would be a very effective improvement especially to those customers who do not like to carry their phones around while working out.

Another improvement we thought of was also a part of enhancing the user experience. If we were to collect the usage data of our specific customers, we would be able to provide them essential data to improve their workout experience. Collecting and analyzing the usage data can provide the customers with information like:

- The time duration between the workout sets
- How long the individual actually worked out
- How long they occupied the power rack

We believe that this information would be very useful since the customer would optimize themselves and their workouts accordingly.

One last idea to further improve our product came to our minds thinking about the current COVID-19 crisis. We know that gyms are not considered to be the cleanest environments and there are countless germs lingering on gym equipment. Bearing this in mind, in crisis times like this no one would actually want to touch anywhere unless it is absolutely necessary. So, we thought of ways to use our product without the need of touching or pressing any buttons. In our design, users would need to touch the button to show that the rack is occupied. Instead of touching, the use of NLP, in other words, a voice recognition feature can be a huge improvement in the path to making the product “touchless”. With this added “touchless” feature, our product can significantly limit the spread of germs, bacteria and viruses. We would still have the occupancy button and touch screen LCD display in the event that the voice recognition system malfunctions. The voice recognition system would allow users to queue themselves verbally as well as verbally signify when they begin using or finish using a particular power rack.

## 4. Progress Made on First Project

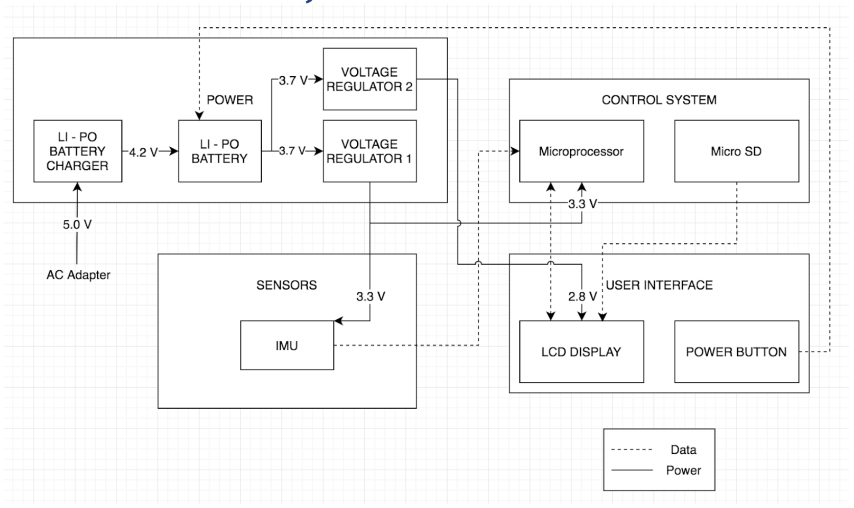


Figure 16: Updated Block Diagram for the Survival Wristband

After our design review, our instructor advised us to take out the GPS sensor from the design, which gave us the room to improve the refresh rate as well as simplify the overall design. To further optimize the computation output of our overall design, we considered ways to minimize the number of calculations needed to reach the resulting values necessary to choose which arrow to display. This led us to find an IMU that had 9 degrees of freedom and has outputs for the absolute orientation of the device with respect to a compass on Earth. Using this device instead, we would no longer need the magnetometers and inclinometer. Furthermore, we would no longer need to perform synchronized calculations using those sensors to get an output, which would reduce software complexity. So, we took out both the GPS, magnetometer, and inclinometer from our sensor subsystem and replaced our IMU with the new one that was able to accomplish better results than is mentioned on our original tolerance analysis [3]. Simplifying the sensor subsystem while achieving the same result minimized the downside of this adjustment and provided the system to work at a faster refresh rate.

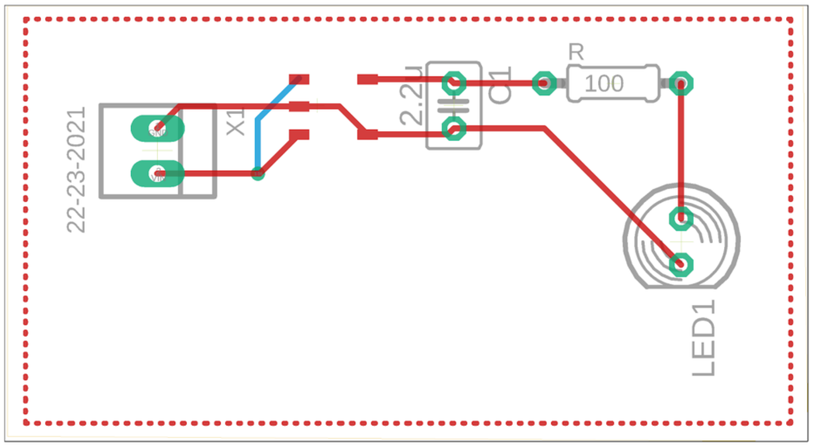


Figure 17: PCB schematic for the power subsystem of the Survival Wristband

## References

[1] "IEEE Code of Ethics," IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 03-Apr-2020].

[2] CableOrganizer.com, "Home," Cord Cover Protection - Conceal Power & Data Cables. [Online]. Available: https://www.cableorganizer.com/cord-covers/. [Accessed: 04-Apr-2020].

[3] "Design Document for Power Rack Manager." [Online]. Available: https://courses.engr.illinois.edu/ece445/getfile.asp?id=17111.

[4] E. Porter, "How big is a power rack? (Dimensions & 13 examples)," Trusty Spotter, 06-Mar-2019. [Online]. Available: https://trustyspotter.com/blog/power-rack-size/. [Accessed: 17-Apr-2020].

[5] "LTspice® Demo Circuits," *LTspice® Demo Circuits | Design Center | Analog Devices*. [Online]. Available: https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator/lt-spice-demo-circuits.html. [Accessed: 09-May-2020].

[6] "Datasheet for IRM-15," Mean Well. [Online]. Available: https://www.meanwell.com/Upload/PDF/IRM-15/IRM-15-SPEC.PDF.

[7] "Datasheet for ADP 122/123," Analog Devices. [Online]. Available: https://www.mouser.com/datasheet/2/609/ADP122_123-1503483.pdf.

[8] "3942," DigiKey. [Online]. Available: https://www.digikey.com/product-detail/en/adafruit-industries-llc/3942/1528-2711- ND/9658069. [Accessed: 17-Apr-2020].

[9] "Ks0029 keyestudio Digital Push Button," Ks0029 keyestudio Digital Push Button - Keyestudio Wiki. [Online]. Available: https://wiki.keyestudio.com/Ks0029_keyestudio_Digital_Push_Button. [Accessed: 17-Apr-2020].

[10] "Datasheet for DOGXL240-7," Electronic Assembly. [Online]. Available: https://www.lcd-module.com/fileadmin/eng/pdf/grafik/dogxl240-7e.pdf. 34

[11] "Datasheet for ATmega328PB," Microchip. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/40001906C.pdf.

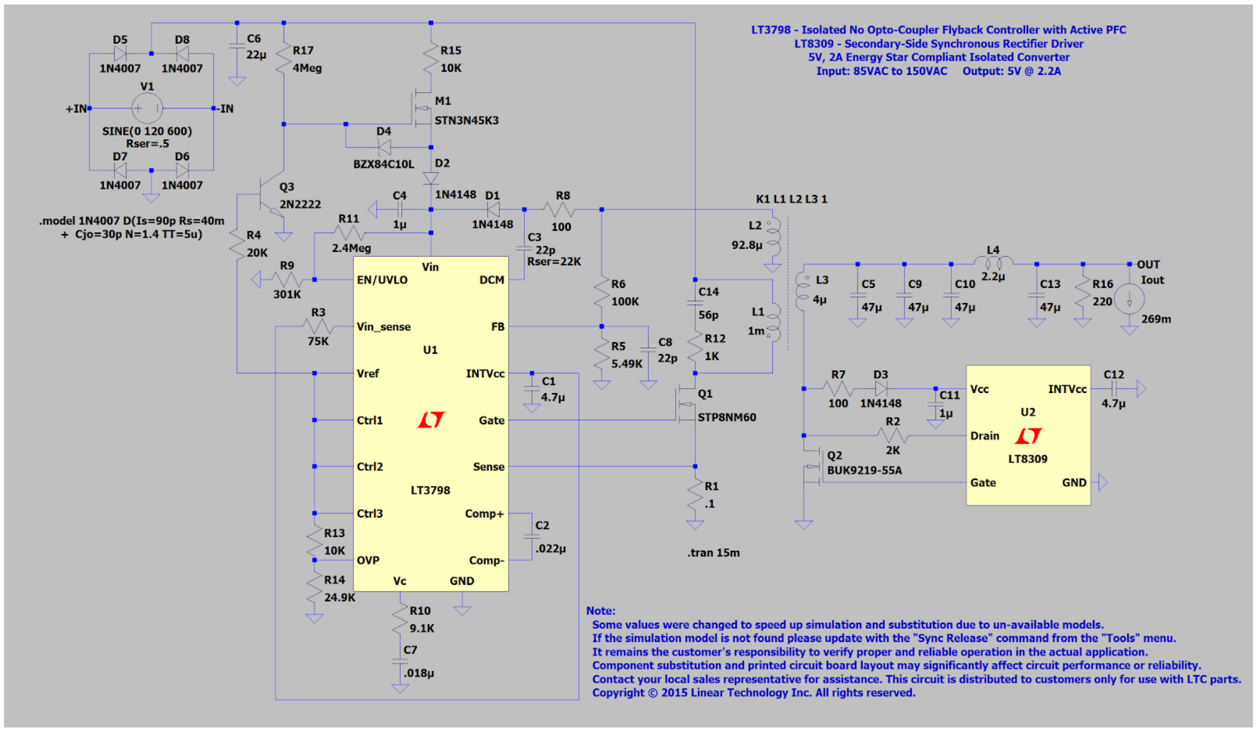
[12] "Datasheet for ESP8266," SparkFun Electronics. [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP8266ModuleV1.pdf.

[13] "Inclinometer," Wikipedia, 10-Apr-2020. [Online]. Available: https://en.wikipedia.org/wiki/Inclinometer#Accuracy. [Accessed: 17-Apr-2020].

[14] Circles Covering Squares. [Online]. Available: https://www2.stetson.edu/~efriedma/circovsqu/. [Accessed: 17-Apr-2020].

# Appendix B Checklist for ECE 445 Final Report Authors

## Figures



LT3798 - Isolated No Opto-Coupler Flyback Controller with Active PFC
LT8309 - Secondary-Side Synchronous Rectifier Driver
5V, 2A Energy Star Compliant Isolated Converter
Input: 85VAC to 150VAC     Output: 5V @ 2.2A

Note:
Some values were changed to speed up simulation and substitution due to un-available models.
If the simulation model is not found please update with the "Sync Release" command from the "Tools" menu.
It remains the customer's responsibility to verify proper and reliable operation in the actual application.
Component substitution and printed circuit board layout may significantly affect circuit performance or reliability.
Contact your local sales representative for assistance. This circuit is distributed to customers only for use with LTC parts.
Copyright © 2015 Linear Technology Inc. All rights reserved.

Figure 3. Power Supply Simulation Circuit [5]

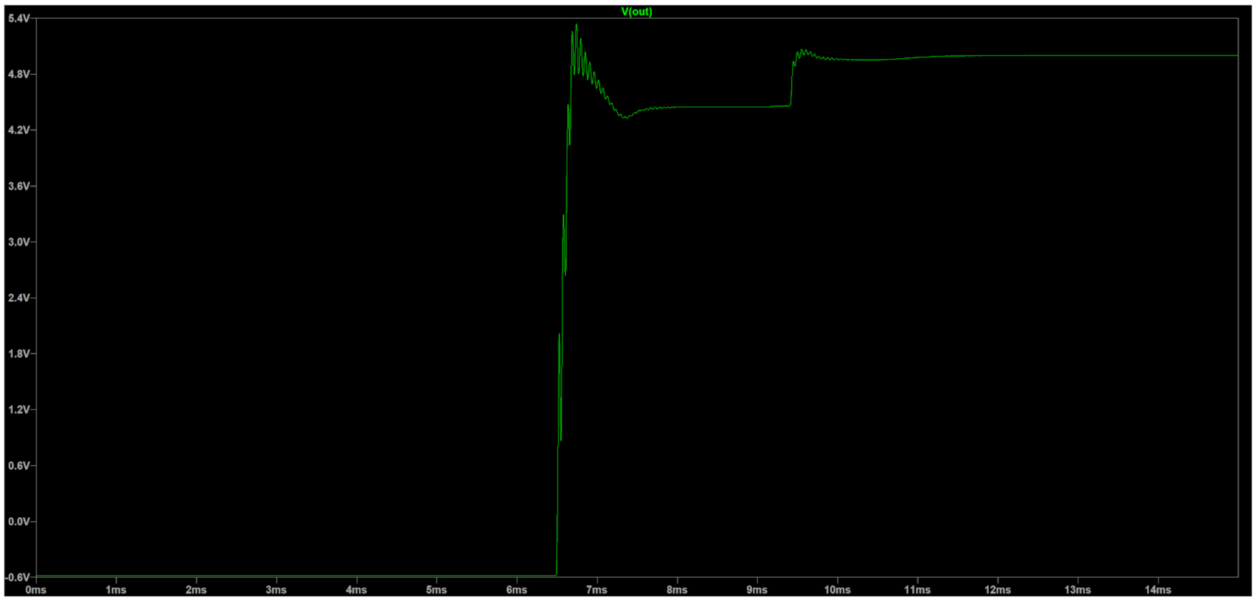
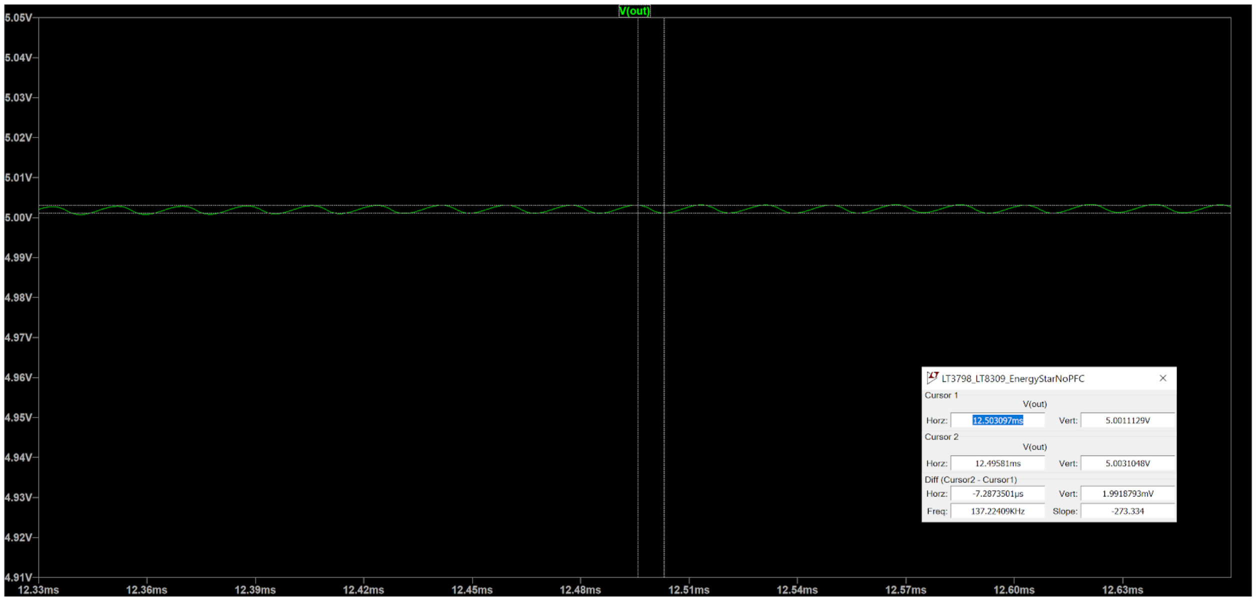
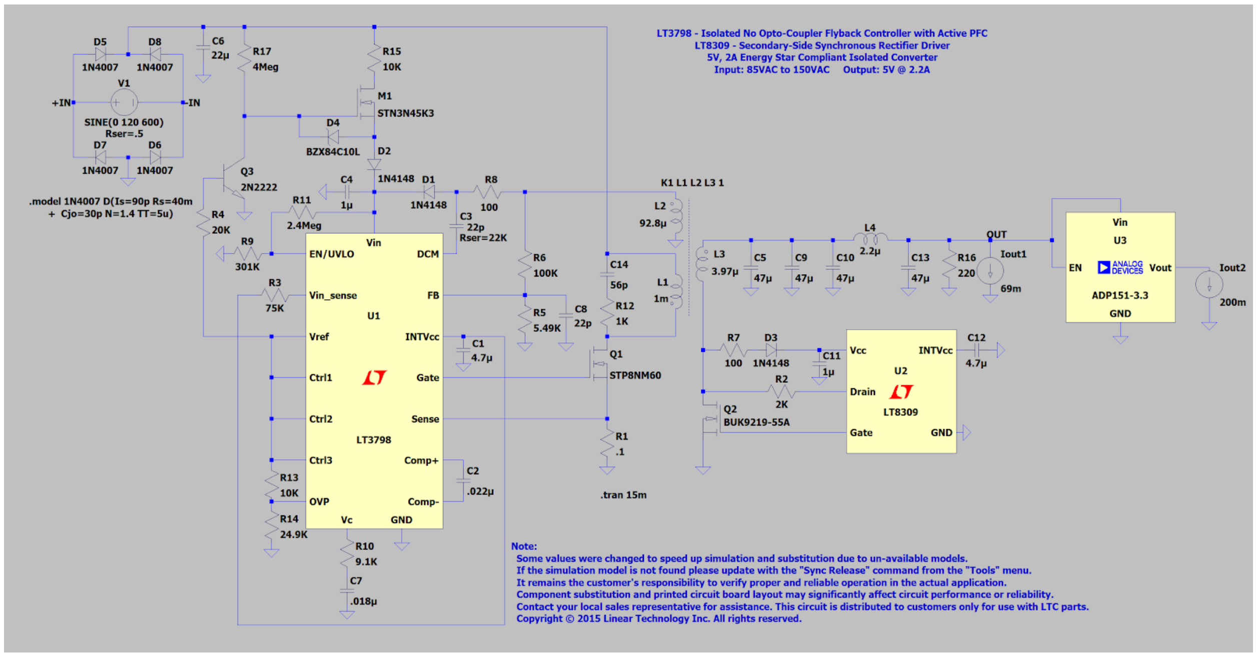


Figure 4. AC/DC Output Voltage Response



```
LT3798_LT8309_EnergyStarNoPFC
Cursor 1
                V(out)
Horz:  12.503097ms        Vert:  5.0011129V
Cursor 2
                V(out)
Horz:  12.49581ms         Vert:  5.0031048V
Diff (Cursor2 - Cursor1)
Horz:  -7.2873501µs       Vert:  1.9918793mV
Freq:  137.22409KHz       Slope: -273.334
```

Figure 5. Zoomed In View of Converter Output Voltage



.model 1N4007 D(Is=90p Rs=40m
+ Cjo=30p N=1.4 TT=5u)

LT3798 - Isolated No Opto-Coupler Flyback Controller with Active PFC
LT8309 - Secondary-Side Synchronous Rectifier Driver
5V, 2A Energy Star Compliant Isolated Converter
Input: 85VAC to 150VAC     Output: 5V @ 2.2A

Note:
Some values were changed to speed up simulation and substitution due to un-available models.
If the simulation model is not found please update with the "Sync Release" command from the "Tools" menu.
It remains the customer's responsibility to verify proper and reliable operation in the actual application.
Component substitution and printed circuit board layout may significantly affect circuit performance or reliability.
Contact your local sales representative for assistance. This circuit is distributed to customers only for use with LTC parts.
Copyright © 2015 Linear Technology Inc. All rights reserved.

Figure 6. Complete Power Supply Simulation Schematic [5]

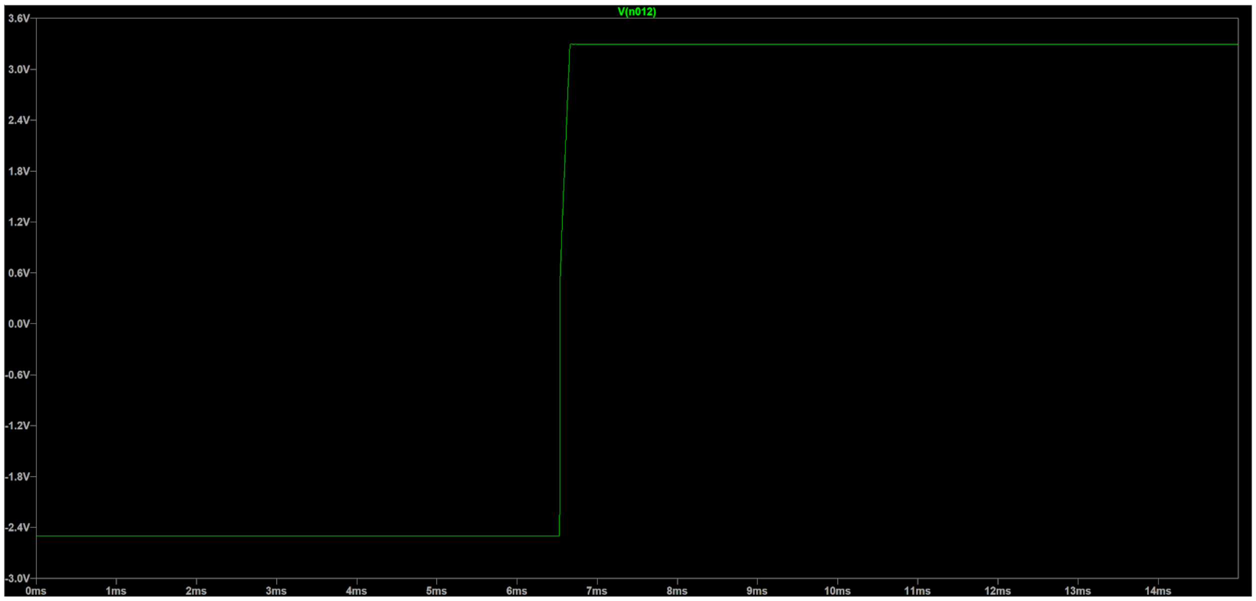


Figure 7. Voltage Regulator Output Waveform

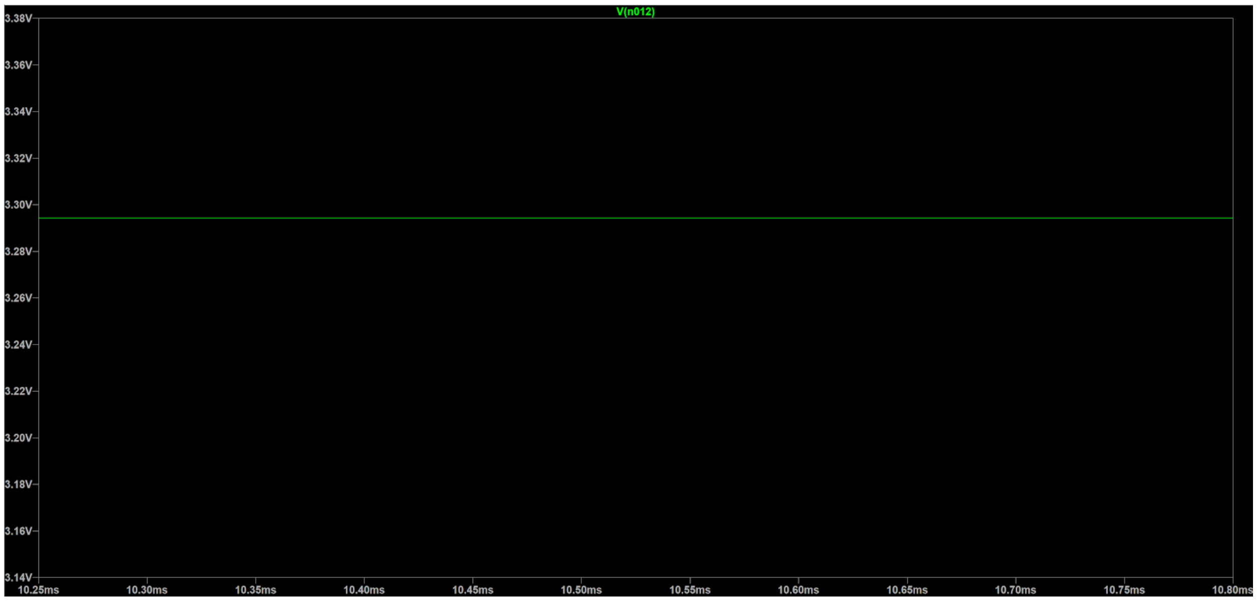
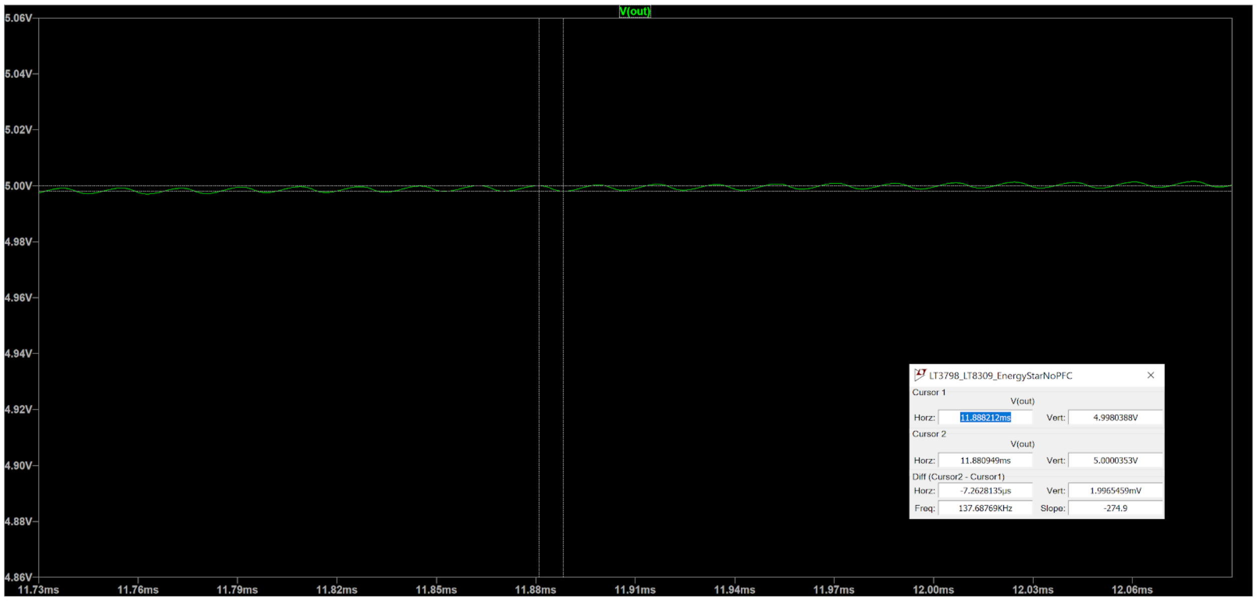


Figure 8. Zoomed In View of Voltage Regulator Output



```
LT3798_LT8309_EnergyStarNoPFC
Cursor 1
                    V(out)
Horz:  11.888212ms        Vert:  4.9980388V
Cursor 2
                    V(out)
Horz:  11.880949ms        Vert:  5.0000353V
Diff (Cursor2 - Cursor1)
Horz:  -7.2628135µs       Vert:  1.9965459mV
Freq:  137.68769KHz       Slope: -274.9
```

Figure 9. AC/DC Voltage Output With Voltage Regulator

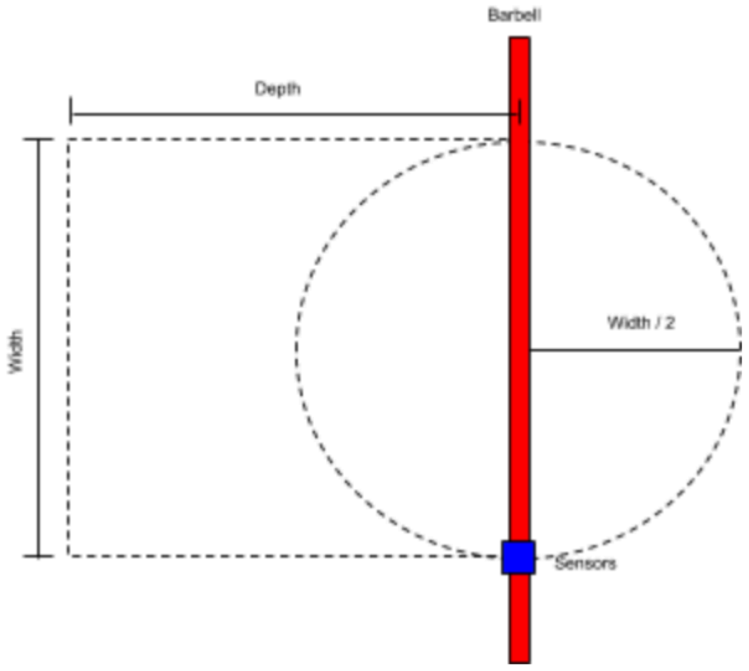


Figure 10: The dotted circle indicates the necessary area to detect movement within relative to the rack to determine occupancy.