Electronic Replacement for COVID-19 Building Monitors at UIUC

By

Patrick McBrayer

Zewen Rao

Yijie Zhang

Final Report for ECE 445, Senior Design, Spring 2021

TA: Yichi Zhang

4 May 2021

Project No. 13

Abstract

Our team designed a physical two-factor authentication system, as a replacement for the COVID-19 human building monitors on the University of Illinois campus. This proof of concept focuses on the entry mechanism based on SaferIllinois's QR generation, as well as a method of limiting entry to one per active user to maintain the safety standards a human monitor sets.

Contents

1 Introduction

1.1 Objective

Human building monitors around the UIUC campus pose an unnecessary risk to students and staff, when a technological solution could be implemented. These monitors are in place to make sure that the right people are being let into the right places, using the Safer Illinois App developed by the university. This app provides access to testing results for the students, and faculty, and creates a randomized, unique building access pass, so it cannot be replicated. Even with this amazing software, we still put these human monitors, and those who have to interact with them, at risk.

Our goal is to create an automated replacement system for these building monitors that still adequately protects the residents of the building. We will do this by creating a physical two-factor authentication network, that uses a central micro-controller to monitor the door for suspicious activity. As we do not have access to the back-end of the Safer Illinois application, or the ability to use campus buildings as a work space for our project, we will be designing a proof of concept 2FA system for UIUC building access. Our solution would be composed of two main subsystems, one that allows initial entry into the "airlock" portion of the building using a scannable QR code, and the other that detects the number of people that entered the space, or a form of people counter, to determine whether or not the user will be granted access to the interior of the building.

1.2 Background

COVID-19 has put everyone across the world in dire straits. Every system or service that is a part of life has been put through one of the most extreme stress tests we have seen in over a hundred years. However, places like the University of Illinois have been trend setters for solutions to the many problems that COVID has placed upon these systems. They have developed new forms of testing, state of the art tracking applications, and many other procedures and systems. However, even with these huge strides in innovation, some approaches are archaic in comparison.

Studies show that appropriate social distancing and mask wearing results in an approximately 3% transmission rate of SARS-CoV-2 [1]. While this seems positive in the grand scheme of things, these monitors are still at risk, and taking away the human element could potentially save these people from infection. On average, humans make about 4-6 errors per hour [2]. Let's say in the case of a building monitor, this error includes misuse of masks or improper social distancing, this creates great risk for all involved, and is not isolated to the monitor either. Since the students/staff being checked are interacting with those monitors, they make errors as well, further complicating the issue. People are not perfect, and human error will always find a way to create problems. With a fully automated system, we can remove that human element, and create a system that can monitor all buildings on campus around the clock.

1.3 Block Diagram & Physical Diagram

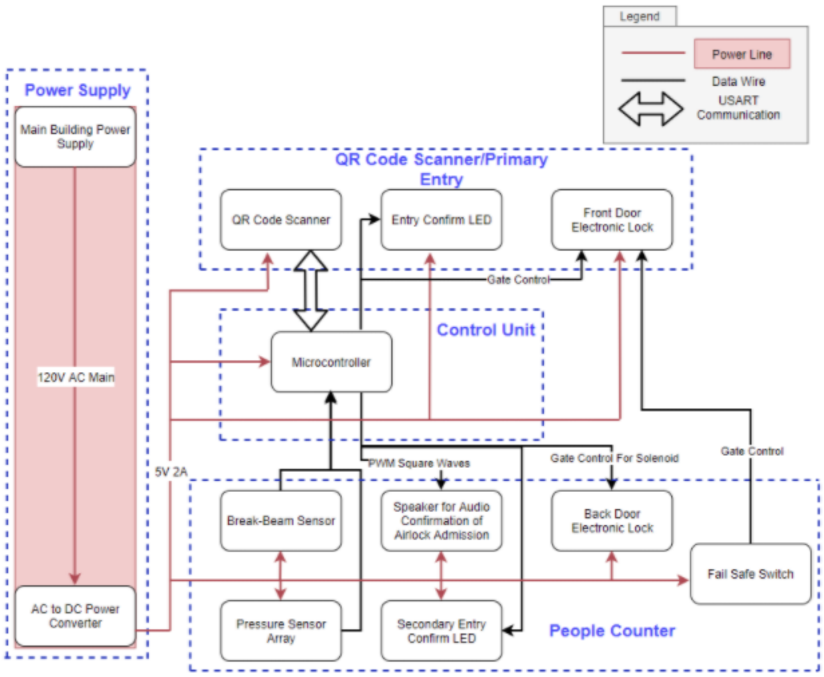


Figure 1: Electronic Building Monitor Block Diagram

The high-level block diagram in Figure 1 shows an overview of our proposed solution. Each sub-system of the design is separated and labelled accordingly, and all data lines are labelled for a quick understanding of how each subsystem passes information to another. The system is split into 4 main sub-systems, composing of the power supply, QR Code Detection System, Control Unit, and People Counter. All information is passed through the micro-controller located within the control unit to generate our desired output. A physical representation of our final project can be seen in figure 2.

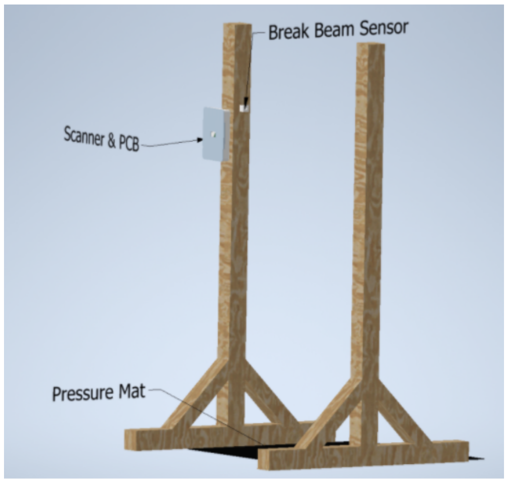


Figure 2: Physical Diagram of Device

1.4 High-Level Requirements

- QR Code Scanner should be able to determine if the user is allowed primary entry using the encoded information from the generated QR code, and refer back to that user when the next scan is performed to prevent repeat access being granted.
- Break-beam sensor and pressure mat within the airlock must work synchronously to determine the presence of a single student/faculty member, as well as accurately keep track of entries with a ¡5% error rate.
- Must be able to process each person within a 30 second time frame, to prevent congestion at the entryways of buildings across campus. The 30 seconds will begin with the QR code scan, and end when the second door is unlocked.

2 Design

2.1 Power Supply

The power supply for our system is quite simple. In the final iteration, we used a 5V 2A micro-USB AC/DC power converter to supply power to our entire system. With a 125uF capacitor bank in parallel to the supply to mitigate any transient noise in the system. Initially however, we planned to use a linear voltage regulator to help regulate that noise. We did not intially understand the operation of the linear voltage regulator, leading us to believe that if we supplied a 5V input voltage, we could output the same voltage level with a decreased noise profile. This turned out to not be the case, as the linear voltage regulator can only regulate down from the given input voltage. This is the reason we scrapped this idea, and decided to use a capacitor bank instead, as the signal from the converter had an already low noise profile, and we really just needed to mitigate feedback noise from our own system.

2.1.1 Micro-USB Breakout Board

To distribute the power from our wall converter, we had to use an after market breakout board provided by Adafruit, that could split the different signal lines of the micro-usb header, into their constituent parts. From this, we pulled the V+ and GND signals, and grounded the data ports, as we only are interested in the power.

2.2 QR Code Module

The QR Code module is what generates and collects the QR code with the information pertinent to the system. This includes identifying information such as name and UIN, as well as the COVID-19 testing status of the user.

2.2.1 Back-end Database

Due to the fact that we do not have the access to the database of the Safer Illinois App, we needed to implement a QR code generator that tells the system the COVID-19 testing status of the user. We needed a QR code generator that works for all types of cellphone. Developing applications that would function on different operating systems was not a good option for a short-term project, so we decided to use a website to generate the code that can be accessed from any device with an internet connection.

The server is deployed remotely. On the server, we need an OS, a server application (Apache), and a database (MySQL). The server is used to store user information and generate the QR code. The PHP code is running on the server to retrieve user information from the SQL database and generate the corresponding QR code. The workflow for the back-end database can be seen in figure 16.

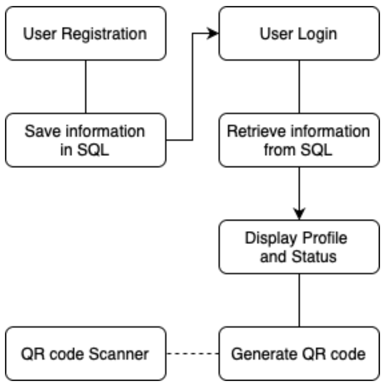


Figure 3: Back-end Database Workflow

2.2.2 Scanner

We are used a Raspberry Pi 4 and the camera module for the device to build a QR Code Scanner, as all after market products were either out of stock or out of our price range. With the help of the OpenCV library, we implemented a program that continuously checks for a QR code and sends any and all information to the microcontroller through the UART communication protocol. The workflow for the scanner can be seen in figure 4.

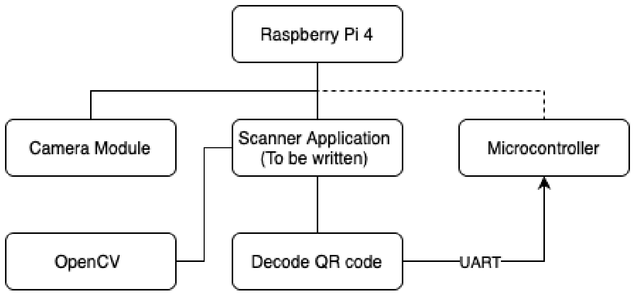


Figure 4: QR Code Scanner Workflow

2.3 Input Module/People Counter

This module is designed to determine the number of people crossing the threshold of the initial entry door. This was done through two sensing methods. One, a set of two TX/RX IR Break-Beam sensors, and two, an array of force sensitive resistors to act as a location determining pressure mat.

The breakbeam component of this system was quite simple. The only thing we wanted to detect from these was someone crossing the threshold of the door. This was done with the break-beam, as every time the IR beam is broken, a signal is sent from the receiver to the microcontroller to notify that this has occurred. The pressure pad component was much more complicated however.

2.3.1 FSR Array Based Pressure Sensor

Our initial thoughts were to use an aftermarket affordable option as suggested by many TA's on our initial RFA. However this quickly proved not to be a good solution, as all after market products with the sensing resolution that we initially planned on having, were up to 300 dollars on the absolute lowest end. Because of this, we quickly transitioned into the idea of building our own sensor, out of an array of force sensitive resistors.

The plan with this was to create a flush array of small sensors that acted as switches, and then mapping those sensors to determine the number of unique pressure areas. The way this was set up was by connecting each FSR to a voltage comparator/OP Amp, that cut off any transient noise on the system, and saturated to high when adequate pressure was applied, but still allowed for the MCU to see this as a high or low signal due to that saturation. The voltage comparator circuit schematic can be seen in figure 5.

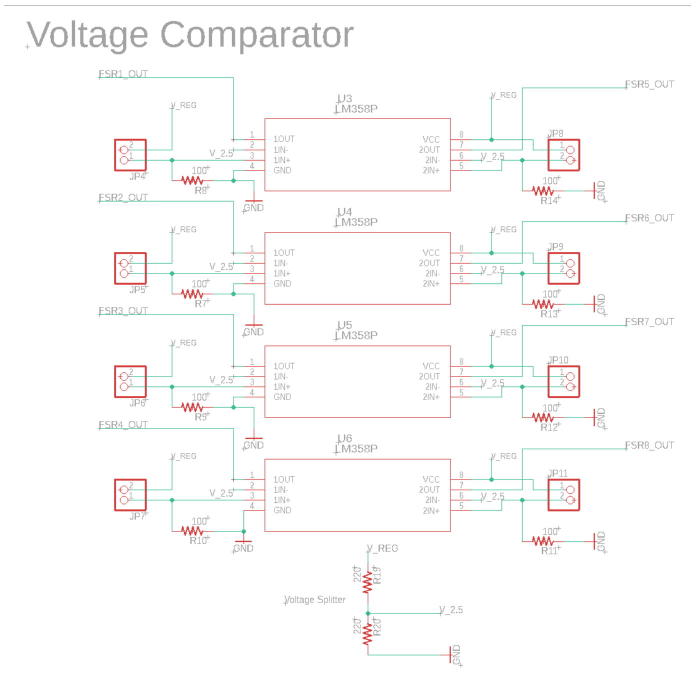


Figure 5: Voltage Comparator Circuit Schematic

Again however, this proved to be extremely expensive, as each square FSR cost about $5. So again we reworked the idea and landed on a forced user positioning to detect one or more users. We still used the voltage comparator setup for each FSR, but cut down on the number of them significantly. This was done by forcing the user to stand on a set of two long FSR's, to verify their position, and then all other FSR's were treated as traps, so that any other user attempting to enter the space would be immediately recognized as an error, and the system would kick them out. The final layout design can be seen in figure 6.

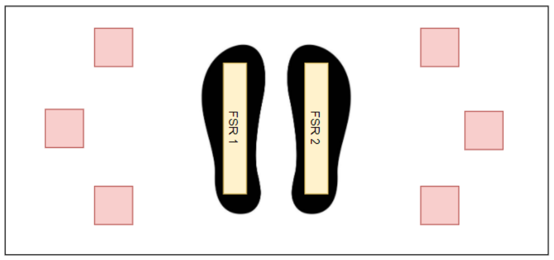


Figure 6: Pressure Plate Layout

2.4 Output Module

The output module consists of two main features, the solenoid actuators which represent our automatic locking mechanism, as well as our user feedback system which consists of both an LED, as well as an audio feedback system.

2.4.1 Locking Mechanism

The locking mechanism is the primary output of the system, as it is what ultimately lets the user into the building if all the tests are passed. We designed this system using two sets of solenoid valve circuits built from low-side transistor switches. We also implemented an emergency release button that would allow a user out of the airlock if they were to end up stuck there due to a reset of the system. Many challenges came as a part of designing this component.

We initially planned on using a low-side transistor switch built from a BJT, however when we built this onto a breadboard using the recommended BJT from the website we bought the solenoids from, it did not function. Due to this, we planned on swapping to a MOSFET version of the switch, as there were available MOSFETs in the lab. Again this posed a problem as the solenoid would heat up and only fire occasionally when the control signal was sent to the valve. After a bit of research, we determined that the issue was due to not truly understanding how the solenoid valve circuit worked. The issue was that the MOSFET we were using was not being driven into saturation with our 5V gate-source and drain-source voltage. We were in fact within the active region, which could contribute to a lot of the strange behaviors. Once we realized this, we ordered a new set of MOSFETs with transfer characteristics that fit our application. The circuit schematic for the locking mechanism can be seen in figure 7.

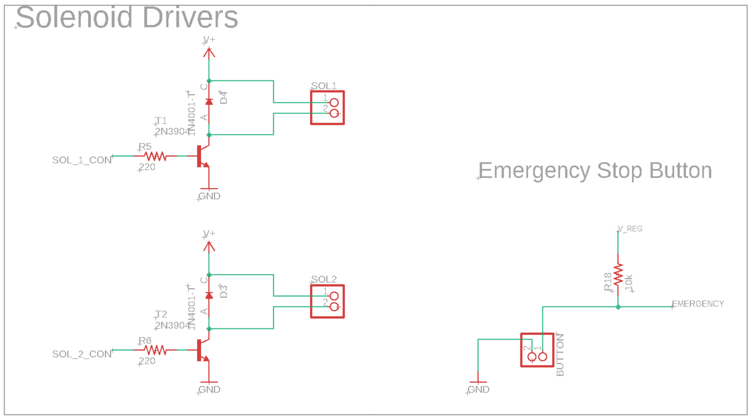


Figure 7: Locking Valve Schematic

2.4.2 Feedback System

The feedback system of our device is meant to inform the user of their success or failure at each verification stage. When the QR code is initially scanned, the user will be notified of it's success or failure through an LED indicator located on the scanner itself. As with most LED indicators, green shows that the users QR code was valid, and that their testing status was negative, meaning they have gained entry into the building, while red shows that either the QR code is not valid, the user has an expired test status, or they have tested positive. The LED also interacts with the people counter system, to inform the user whether they have passed or failed given the same color scheme.

The second component of our feedback system is an audio portion consisting of an audio amplifier as well as a 4Ω speaker. This was done as when the user is within the airlock, the LED would be less obvious as a source of feedback, so we determined that an audio cue would work nicely. While this portion of the system was laid out on the PCB, as well as having functional code, we ran out of time to solder on the components. The schematic for the amplifier can be seen in figure 8.

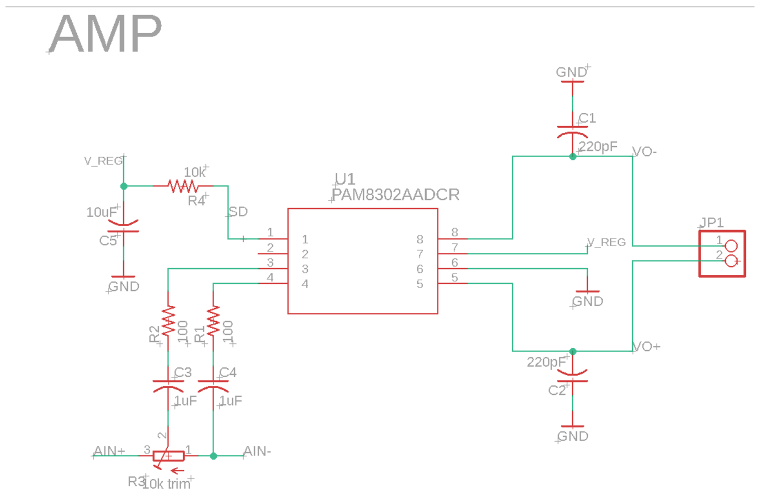


Figure 8: Audio Amplifier Schematic

2.5 Control Unit

The Control Unit is able to control the whole system as well as the computational logic. It controls and decides the status of the secondary locking mechanism based on the data collected from the various sensors, as well as uses the information scanned from the QR code to determine initial access into the people counting subsystem. With the data it receives, it can control both the LED and the solenoids. It supports the UART communication protocol to communicate with the Raspberry Pi which acts as a QR Code Scanner. The workflow of the micro-controller can be seen in figure 9.

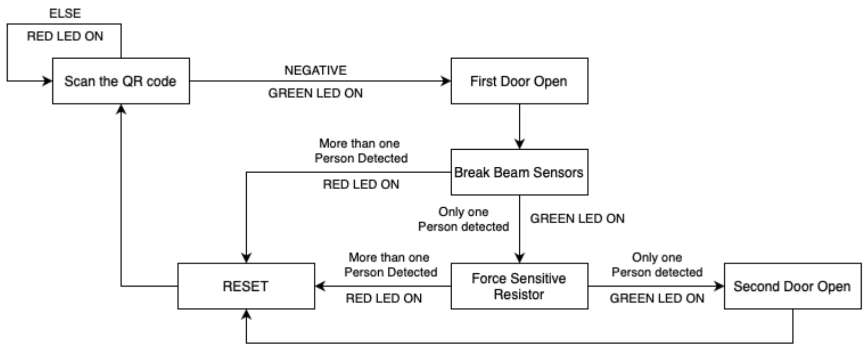


Figure 9: Control Unit Workflow

2.5.1 Micro-controller

The Micro-controller is the core of the Control Unit. It controls the whole system and determines the locking status both doors through signals to the solenoid valves. We use the recommended ATMEGA32-16PU to run our system. To program the micro-controller, we firstly used the Arduino as AVR ISP to burn the boot-loader into the micro-controller. After that, with the basic boot-loader running in the micro-controller, we used the FTDI USB to Serial Adapter to program it using UART communication. The signal layout for the microcontroller can be seen in figure 10.

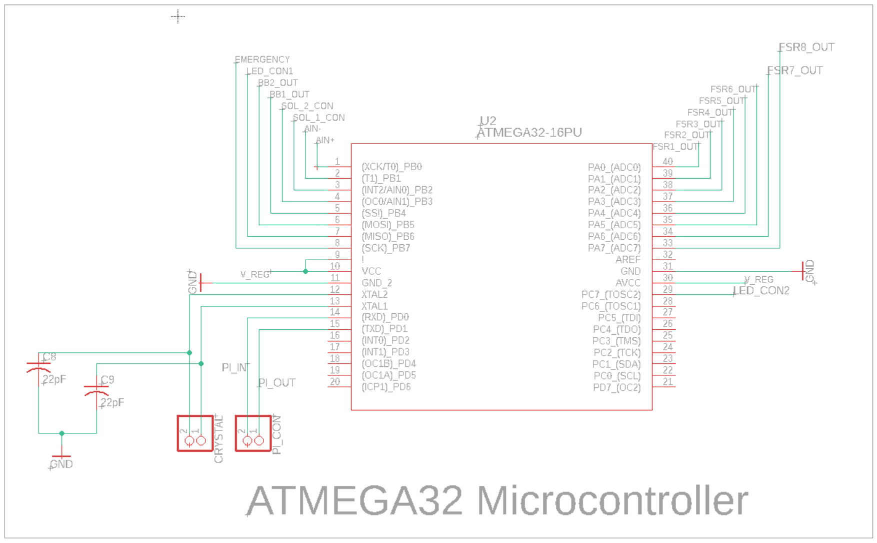


Figure 10: Control Unit Signal Layout

2.6 PCB

The PCB Layout of our design has a few contingencies implemented into it that ended up working in our favour. I incorporated two different ways of driving our signaling LEDs depending on the final implementation of our design. One used an inverter so that the light would be constantly red unless driven by a control signal, the other used two separate pins to drive red and green respectively. We ended up using the two pin method, as it made signaling the user much easier, as a constant red may initially be confusing.

The strange traces are due to multiple reworks as our design evolved over the semester, which probably should have been done from scratch, however I just moved things around and hoped it would work. Which, thankfully, it did.

Another component of our project not discussed in this presentation was the plan to incorporate an audio feedback system so the user was more aware of the results of the verification. This was incorporated onto the PCB, however we ran out of time to implement it. The layout can be seen in figure 11.

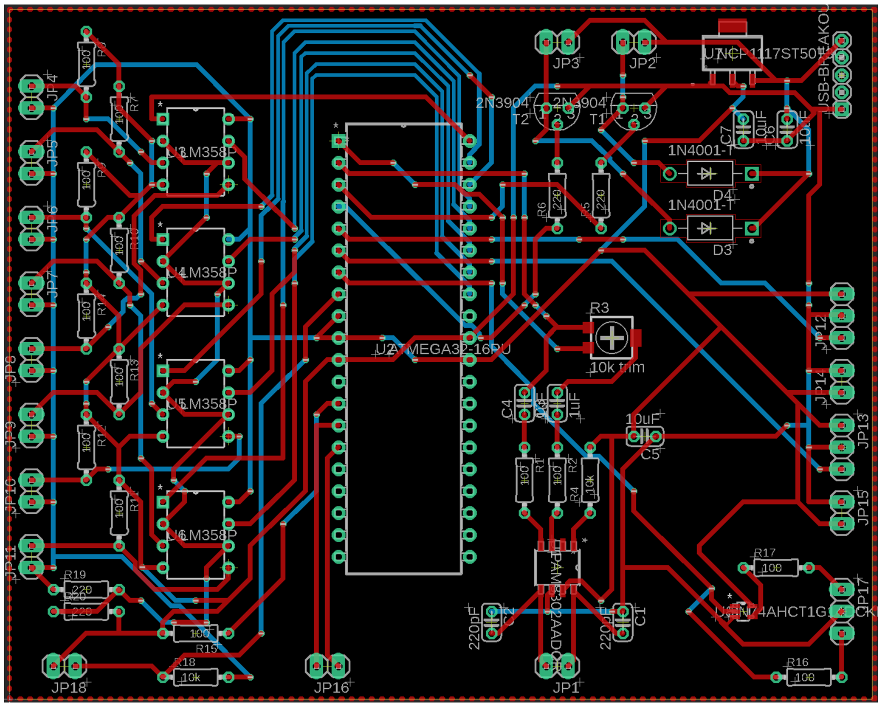


Figure 11: PCB Layout

3 Design Verification

For a comprehensive list of requirements, see Appendix 1 for each sub-component

3.1 Power Module

Each solenoid has a current draw of 1A, while the MCU has a rated current of 200mA. This means that if both solenoids were to be in the active state at the same time, then power would be lost to the MCU, causing the entire system to reset. This doesn't take into account the other power hungry devices of our circuit either. As each sensor, and the amplifier, will require some amount of additional power/current to run properly. To counteract this, we will fire the solenoids with the intention of all other components of the circuit resetting. This means that there must be a sequence of events, where this reset of sensors and other ICU's will not affect the overall integrity of the system. As the firing of the solenoid is the final step at the end of each of our sub-systems, this should be relatively simple. However we must make sure that at all costs the MCU is never reset, as this could cause issues with people being left locking within the "airlock" of our system, requiring usage of our fail-safe switch to safely exit.

We will be using a wallwart 5V 2A AC/DC Converter to power our system, meaning the maximum power that can be delivered is 10W of power. This can be seen using the power equation:

$$P = V * I \tag{1}$$

With this we can determine whether the current system can be supported with this maximum power supply. The QR Code Scanner is being powered separately using a raspberry pi, and will not be included in this calculation. That verification can be seen in table 1

Table 1: Power Draw Tests

Component	**Max Rated Voltage**	**Max Rated Current**	Power
ATMEGA32 MCU	4.5-5.5V	20mA	0.11W
Break-Beam Sensor	5V	20mA	0.11W
Solenoid	5V	1A	5W
LED	5V	20mA	0.11W
Op-Amp x8	5V	40mA (5mA Each)	0.2W

This shows that the device is easily powered by the given power supply, given that only one solenoid is drawing current at a time. This will be dealt with on the software side as specified above.

The testing in the power module required analyzing the voltage fluctuations we were getting due to triggering of the solenoids. After some research we determined that the inductive nature of the solenoids was pulling down the voltage level on the falling edge of the trigger. To mitigate this we determined that a capacitor bank in parallel with the power supply would work well. To determine the level of capacitance required we tested the voltage drop across a wide range of capacitors, and ended up using 125uF as our bank value. The testing data can be seen in table 2.

3.2 QR Code Module

Table 2: Capacitance Tests

Capacitance	Voltage at MCU when Solenoid Triggered
25uF	4.5V
50uF	4.56V
75uF	4.63V
100uF	4.8V
125uF	4.91V

3.2.1 QR Code Generator

After we developed the QR code generator, we needed to make sure it actually stored the tuples of users' information into the database. To test, we input some user information by registering multiple different names and passwords to simulate multiple users using the system. Then we ran SQL queries at the back-end to see if all tuples we input to the system are being stored correctly in the database. Also to make sure the code is correct, we use multiple different types of cellphones to scan the QR code, and decode the BASE64 string to make sure the information stored in the string is decoding properly on the other end.

3.2.2 QR Code Scanner

Another qualitative test we did to verify the QR code scanner is, we use various QR codes, not only the codes generated by our generator, but also some codes generated by some apps, which contain some URL. We connect the Raspberry Pi to a monitor and use it to scan the codes to see if our scanner will retrieve the right URL. Also this allowed us to make sure the PI wasn't sending any faulty signals to the micro-controller if an invalid QR code was scanned.

To make sure the information was being properly transferred to the control unit through UART, we separated the decoding code block from our firmware, and incorporated it with a simply LED algorithm, where the LED would indicate to us the validity of the code received by the control unit.

3.3 People Counter System

3.3.1 Break-Beam IR Sensors

After initial attempts of setting up the bream-beam sensor across our threshold, we ran into an issue where the signal was not consistent at the 30in. Distance that the datasheet indicated was it's max range. Due to this, we ran a series of tests to determine the actual range that we could operate the device at. Our testing method involved taking a 24in. Plank with 1in. Increments marked onto it and moving the break beam to these increments and taking a reading of the current output of the receiver. The results of those tests are shown in figure 12. We realized we could go slightly beyond this 24in. Distance, and settled on a 28in. Threshold width.

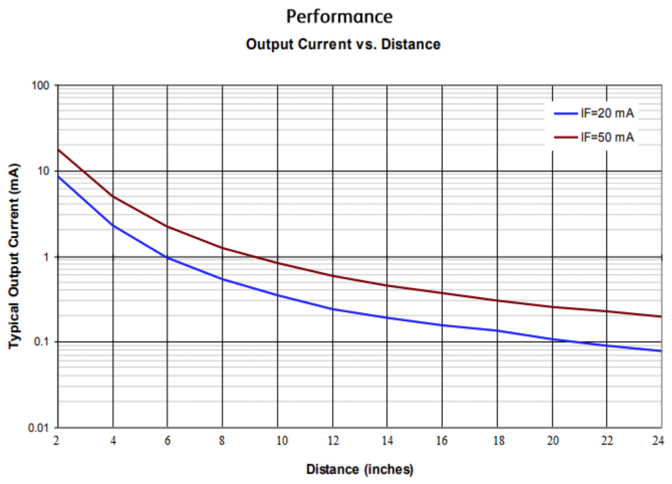


Figure 12: Current vs Distance Measurements

3.3.2 FSR Pressure Array

Testing on the FSR portion of this subsystem was much more qualitative. We just needed to make sure that the output of the voltage comparator was consistent so that the microcontroller could verify the difference between pressure and no pressure. The one problem we ran into during our testing was that the output of our voltage comparator was about 1V less that the power it was receiving. We know now that this is due to losses within the op-amp and ended up routing the signal to the analog pins of the MCU to account for this mistake. This meant we could set the threshold of high vs. low as opposed to using the 5V standard on the digital pins.

3.4 Output Module

3.4.1 Locking Mechanism

In the beginning, much of the troubleshooting on this portion of our project relied on a lot of trial and error with different transistor types, as we believed that given a proper voltage to the gate/base, we would have no problem creating a transistor switch, however, this ended up being a large flaw in our testing and research.

The locking mechanism testing ended up relying mostly on the transfer characteristics of the device. As we were having trouble deciding on the type of transistor switch to use, as described above, we needed to find transfer characteristics that fit the desired outcome. As our Solenoid draws 5V 1A, the drain source voltage needed to match this with a given 5V gate source voltage. The two MOSFETs available in the lab had the transfer characteristics shown in figure 13.

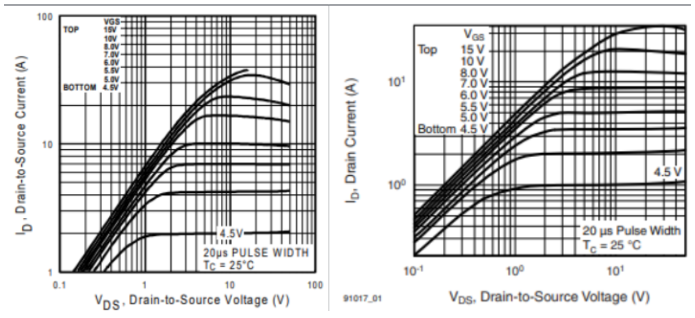


Figure 13: MOSFET Transfer Characteristics

As such, we ended up going with the IPR520N Power Transistor which has the characteristics shown on the left. In this graph you can see that the device is adequately driven into saturation, and will act like a switch as desired.

3.4.2 Feedback System

Testing on the feedback system was, once again, quite straight forward. As the only component of this system that we had functioning was the LED, we just had to verify that when certain steps of our verification process were reached, the LED was indicating that. To do this, we broke the firmware into its constituent components based on the subsystem, and tested that each step registered a red or green indicator from the LED, and that the pin of the microcontroller was outputting the desired 5V.

3.5 Micro-controller Firmware

Testing the firmware on the micro-controller was simple and straight forward. At the very beginning, after the initial version of the firmware was implemented, we tested it on an Arduino Uno, to make sure it had no logical problem in the code. After that, we tried to burn it into the micro-controller instead. For testing purposes, everything that needed to be triggered by the sensors was replaced by a time counter to verify whether the loop function was running continuously or being blocked. Also, we changed every output function including the solenoid driver and the LED driver to the Serial.Print() function to display every status on the monitor. After the logic of the code had been verified, we changed everything back to the digitalWrite and digitalRead, and connected the micro-controller to these external sensors and LEDs. All written and verified firmware can be found in Appendix B.

4 Cost

4.1 Parts

A breakdown of our components can be seen in table 3, and the total cost of our prototype comes out to be roughly $190.41. This is with an estimated cost of smaller components such as resistors/capacitors and small IC's.

Table 3: Part Costs

Part #	**Description**	Manufacturer	Quantity	Cost ($)
PAM8302AADCR	1 Channel Amp	Diodes Incorporated	1	0.65
1528-2526-ND	Break-Beam IR Sensor	Adafruit Industries	2	13.00
ATMEGA32-16U	Microcontroller	Microchip Technology	1	6.61
5823782011	5V 2A AC/DC Converter	Ronghua	1	5.95
BOB-12035	Micro-USB Breakout Board	Adafruit Industries	1	2.75
LTL1BEKVJNN	3 Pronged BiColor LED	Optoelectronics	2	0.74
Model 4B	Raspberry Pi + Camera	Raspberry Pi	1	72.21
N/A	Assorted Misc. Components	N/A	N/A	7.00
ROB-11015	Solenoid Actuators	Sparkfun Electronics	2	9.90
Interlink 406	Force Sens Resistors	Interlink	8	71.6
Total				190.41

4.2 Labor

Our fixed development costs are estimated to be $35/hour, 10 hours/week based on the average salary of entry level ECE graduates [6]. Operating on a 16 week semester results in a total cost of:

$$2.5 * \frac{\$35}{hr} * \frac{\$10}{hr} * 16wks * 3 = \$42,000 \tag{2}$$

Additionally we expect the machine shop to take around one hour to build our wooden threshold, so an extra labor cost of around $30 is accrued there.

4.3 Schedule

Table 4: Schedule

Week	Patrick	Zewen	Yijie
3/1	Hardware search, schematic creation, Cost Analysis, schedule creation and, MCU Research.	RV Table creation and QR Code Sensor development.	Hardware Search. Server Database research. QR Code Sensor development
3/8	Prepare for Design Review/Finalize Subsystem 1 Schematic/Begin Subsystem 2 Schematic	Create QR code scanner from raspberry PI	Research into UART data transmission for encoded QR code information
3/15	Finalize Subsystem 2 Schematic/Begin PCB Design. Construction of Pressure Plates.	Start research into MCU programming, and I/O port usage.	Create backend for subsystem 1 though Server creation on PI.
3/22	If first round PCB orders were not met, revise issues and make sure round 2 is met. Begin Testing subsystem 1 for edge cases/ Meeting	Continue research on MCU programming, and implementing the final code base onto the chip.	Extensive range testing on Scanner and Breakbeam sensors, evaluating the accuracy of our sensors.
3/29	Sanity Check, are we meeting deadlines, and if we are behind, where do we need to catch up?	Sanity Check, are we meeting deadlines, and if we are behind, where do we need to catch up?	Sanity Check, are we meeting deadlines, and if we are behind, where do we need to catch up?
4/5	Begin construction of our proof of concept work space (Small doors, detection field.)	Begin working on 3D printed Case	Begin Testing of Subsystem 2 for edge cases of people counting/Meeting Verifications
4/12	Conduct environmental testing on Subsystem 2	Conduct environmental testing on the QR Code Scanner + Case	Conduct environmental testing on QR Code Scanner + Case
4/19	Mock Demo Preparation + Final Checklist of R/V Tables	Mock Demo Preparation + Final Checklist of R/V Tables	Mock Demo Preparation + Final Checklist of R/V Tables
4/26	Prepare Final Presentation/Paper	Prepare Final Presentation/Paper	Prepare Final Presentation/Paper

5 Conclusion

5.1 Accomplishments

Overall, we are very proud of the work that we did on the project this semester. After a rough start with a harsh design review, we were very nervous as the success of our project. However, in the end we accomplished 95% of the project goals that we set ourselves. Even with many delays in part and PCB delivery, I never felt like we were behind once we reached the design stage. While I would say time management was a plus, in reality, it was more of a join determinism between our group, as we had a lab reservation almost every day, and attempted to always be iterating and improving on our designs. While the semester started in a very stressful way, by the end, we feel like this was one of the most rewarding classes, and projects, we have ever worked on.

5.2 Uncertainties

Due to the budget problem, the break-beam sensors that we are using are cheap but can only work stably at a max distance of around 20", which leads to the constraint of using a narrower door. To deploy the project on the general doors, we may switch to IR sensors which are more expensive but have a longer working distance. This sould be relatively simple as it would only require upgrading the TX portion of the device, as the the power of the transceiver is what determines the range of the device.

We are planning to implement the audio system to help the feedback system, which can provide different tones when different situations occur. We have successfully implemented the code and tested it separately on an Arduino, but have not enough time to migrate it to the microcontroller on the PCB board.

5.3 Ethical considerations

According to ACM's ethical guidelines [3], we should use personal data legally and collect only the necessary personal information. Given that this project cannot use data from safer Illinois, this project will simulate a two-factor authentication system. Therefore, this project will not collect data from users and will not use biometric information. If this project does go into realization, the system will not collect biometric information from students and will only use publicly available identifying information, such as students' name, and their NetId. And they will not be made public by this system.

To avoid the system adding any other new safety problems and concerns to the campus society, and according to IEEE Code of Ethics [9], the system is hand-free and adds no steps to the original people-checking process. The system just uses a QR code scanner to avoid the possibility of human processing errors as well as not decreasing the efficiency of entering the building. Also, since there can only be one person in the people detecting area, it will not violate the Social Distancing regulations in Illinois and UIUC campus [8].

Since the system involves a closed space that we are nicknaming the "airlock", there is a chance of a malfunction that could result in a user being locked within this environment. The main reason this could occur is that both solenoids fired at the same time, resulting in a system reset, leaving the user within. The workaround for this issue will come in the form of a fail switch within the "airlock" that will be hardwired into the control circuit of the entrance solenoid, allowing the door to be unlocked, and letting the user leave the space.

Some of the devices in this project will work outdoors, such as the QR code scanner. Given that it is difficult to guarantee a dry environment outdoors all the time, all devices outdoors must meet the IP55 [7] standard to guarantee safety.

Another safety problem is "hijacking" the system. Since People who haven't got a negative COVID-19 test result recently cannot generate a QR code with permission of entering the building, they may force or "borrow" the QR code generated by another person with the negative test result. This problem can't be solved since we are lacking a biometrics ID system, but adding a webpage for "reporting" may partially solve the problem. The University or the other administrations may use the information reported to start an investigation, and it can reduce the possibilities of people "hijacking" the system.

5.4 Future work

There are a great number of improvements that we would like to add to our system, given we have more time to work on it. Some of the most pressing issues involve the modularity of the device. If we were to deploy this product across campus, it would need to account for a wide range of entryways that could cause issues with our system. These issues come in the form of range issues, traffic concerns, etc. If we could make the project more modular then it could counteract this issue. One way we see this being possible is with the addition of wireless modules that could allow for placement anywhere in range of the system, without concern for wire management. A battery operated control unit would also allow for more flexibility when it comes to integration. Also as mentioned before, we would need to upgrade our IR Break-Beams as well.

We also would like to incorporate a higher resolution pressure detection method, that would limit the amount of error currently present in our system. This could be done with a larger number of FSR's in our array, or using an after market product with more sophisticated technology.

Finally, integration with Safer Illinois would make the data much more trustworthy as well as protected from attackers. If this could not be done, then we would need to upgrade our currently existing back-end database to allow for these things.

6 References

[1] The Lancet, ‘Physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: a systematic review and meta-analysis’, 2020. [Online]. Available: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)31142-9/fulltext. [Accessed: 15-Feb-2021]

[2] The Pulse, ‘Human Factors, Human Error and the role of bad luck in Incident Investigations’, 2016. [Online]. Available: https://www.linkedin.com/pulse/human-factors-error-role-bad-luck-incident-graham-edkins/ [Accessed: 18-Feb-2021]

[3] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” Association for Computing Machinery, 2018[Online]. Available: https://www.acm.org/code-of-ethics [Accessed: Fed. 16, 2021]

[4] United States Department of Labor, “Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices” Occupational Safety and Health Administration, 2019[Online]. https://www.osha.gov/dts/shib/shib011819.html [Accessed: Feb. 17, 2021]

[5] TT Electronics ‘Optical Emitter and Sensor Pair OPB100Z’, 2016. [Online]. Available:
https://www.ttelectronics.com/TTElectronics/media/ProductFiles/Optoelectronics/Datasheets/OPB100.pdf
[Accessed: 27-Feb-2021]

[6] Salary.com ‘Hourly Wage for Electrical Engineer I in the United States’, 2021. [Online] Available: https://www.salary.com/research/salary/benchmark/electrical-engineer-i-hourly-wages [Accessed: 1-March-2021]

[7] sourceiex.com ‘Degrees of Protection’, 2021. [Online]
Available: https://www.sourceiex.com/Catalogs/IP[Accessed: 1-April-2021]

[8] University of Illinois Covid-19 ‘Basic Person Actions’, 2021. [Online]
Available: https://covid19.illinois.edu/health-and-support/basic-personal-actions [Accessed: 1-April-2021]

[9] IEEE ‘IEEE Code of Ethics’, 2020. [Online] Available:
https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 1-April-2021]

Appendix A Requirement and Verification Table

Table 5: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Power Supply (a) The external power source should provide stable power of 100V-240V, which can fulfill the input requirement of the AC to DC converter. In the United States, since the supply voltage is 120V, we may assume the external power source will be 120V (+/- 15 (b) The AC to DC converter must be able to convert 100V-240V AC to 5V (+/- 5%) 2A(+/- 5%) DC, which is needed to power the system. Since the microcontroller can handle 5.5V at most, setting the voltage up to 5*1.05 = 5.25V is safe for the device.	1. A (a) To verify an AC power, set the multimeter to AC mode. Use one hand to hold the red and black probes to ensure safety. Do not let the metal parts of the probes touch each other to avoid short circuit. Insert the red probe into the smaller slot and the black probe into the larger slot. If it gives a reading of 110v-120v, the outlet is qualified. 2. B (a) To verify the DC power converted by the converter, a multimeter will be used to check if the output voltage of the converter is equal to 5V. If it gives a reading of 4.75 - 5.25V, the outlet is qualified. (b) To make sure the power supply is enough for both the components on the PCB board and the solenoid, we need to verify the converter can provide constant 2A current. To verify it, we can still use the multimeter under the current mode. If it gives a reading of 1.9A - 2.1A, the outlet is qualified.	Y
2. Breakout Connector (a) This connector should break out the micro USB's pins to VCC, GND, ID, D+, and D- . With the power output remaining consistent to the 5V 2A output of the Converter.	1. (a) Use a wall outlet adapter to connect to a micro USB to USB A cable. Plug the micro USB part into the Breakout Connector. Use the multimeter to measure the voltage. If the measurement is correct, then the Breakout connector is qualified.	Y

Continued on next page

Table 5 – continued from previous page

Requirement	Verification	Verification status (Y or N)
3. QR Code Scanner (a) Since the system must be able to process each person within a 30 second time frame, it's important to have the scanner have a high accuracy ($> 95\%$) to detect the QR code. (b) The scanner will be used by many people with different models of phone, so it must be able to handle them. As the scanner will be installed outdoors, it must be able to work properly under the sunlight. (c) The scanner will send the information retrieved from the QR code to the microcontroller within 3 seconds.	1. (a) Connect the scanner to a monitor directly. Instead of sending information to the microcontroller, we may display the information on the monitor. Make sure the QR code scanner can output the pre-set string correctly. 2. (a) Use different models of phones to display the QR code, and use the scanner to scan them. The expected strings included in the QR codes should be displayed on the monitor. (b) Repeat the previous step in different environments, and make sure it works well under the sunlights. 3. (a) Connect the QR code scanner with an Arduino using UART for the test purpose. Connect the Arduino to the computer, and open the Serial Monitor to check whether the Arduino can receive the information decoded by the scanner.	Y
Continued on next page		

Table 5 – continued from previous page

Requirement	Verification	Verification status (Y or N)
4. Output Module (a) The lock will work under 5V, and when receiving the signal from the microcontroller, it must unlock/lock the door and keep that state until the power status changes. (b) The LED runs well under 5V (+/- 5%) and 30mA (+/- 5%) and be clearly visible from 0.5m away.	1. (a) Use the Arduino as the power source to test the solenoid. (b) Build a small circuit on a breadboard, connect the Arduino's 5V output to a button, as well as connect the button to the positive side of the solenoid. (c) Connect the negative side of the solenoid directly to the GND pin on the Arduino. (d) Verify when the button is pushed, which means the 5V voltage power supply is turned on, the lock is immediately unlocked. (e) Verify when the power supply is cut, the lock is locked. 2. (a) Build a simple circuit containing only LED, power supply and resistor. The power supply will be 5V, and the resistor will be 100. (b) Keep the previous test, check the LED from 0.5 meters away.	Y

Continued on next page

Table 5 – continued from previous page

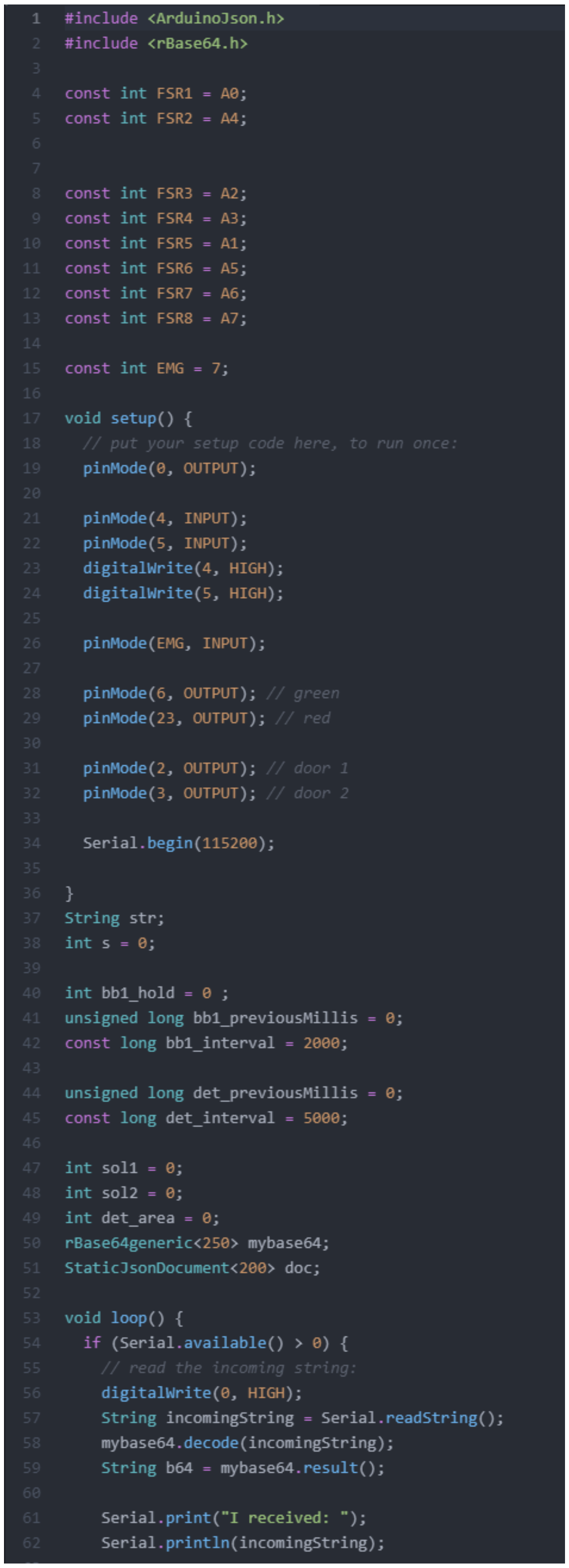
Requirement	Verification	Verification status (Y or N)
5. Break-Beam Sensors (a) The Break-Beam Sensor can work under 5V DC and transfer the information to the microcontroller immediately. (b) The sensor must have the longest detection distance ¿= 50cm, and provide enough accuracy to avoid the false alarm.	1. (a) Use the Arduino as a prototype test. Connect the sensor to the 5V and GND pin on the Arduino. To read the feedback of the sensor, connect the data line to a digital pin of the Arduino. (b) Connect a LED bulb to another digital pin on the Arduino. Write a short code, let the Arduino send a high signal to the LED pin when it receives the high signal from the sensor pin. 2. (a) Put the sensor in the distance of 50cm. When an obstacle appears between the receiver and the sender, the LED will be turned on. Perform multiple tests and calculate the accuracy rate to see if it is within the range we need.	Y
6. FSR Array Pressure Pad (a) Across a wide range of unique pressure profiles, the MCU must be able to make a rough estimate as to how many feet are preset on the mat, as well as consider outliers such as crutches or wheelchairs.	1. (a) Use the Arduino to test the pressure sensor. Connect the sensor to one of the digital pins of the Arduino. (b) We will construct 6 unique tests, to determine the accuracy of our system: 1 person walking in normally as a baseline, 1 person walking in sideways, 2 people walking in back to back, 2 people walking in sideways, Wheelchair entry, Person walking in on crutches.	Y

Continued on next page

Table 5 – continued from previous page

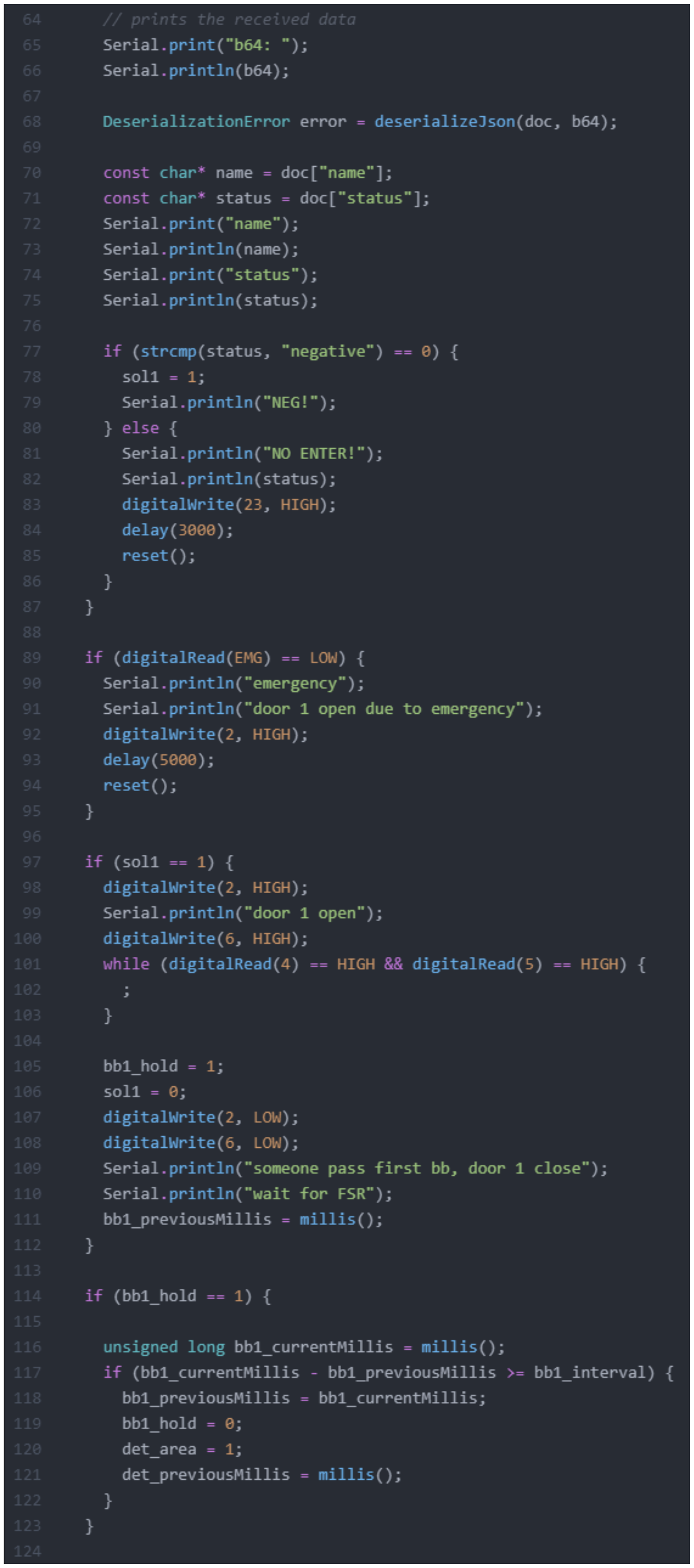
Requirement	Verification	Verification status (Y or N)
7. Speaker & Amplifier (a) The speaker must be clearly heard when people stand in the counter area. (b) The system must be able to communicate with the I2S interface on the MCU. (c) The Amplifier can drive the speaker of 2.5W at 4Ω.	1. (a) Attach the amplifier to a speaker with 4 and 2.5W. Turn on the amplifier and the speaker to see if the amplifier works properly. (b) Repeat this step in the counter area to see if they can be heard clearly.	N
8. Microcontroller (a) The microcontroller needs a bootloader to write the program into it successfully using the Arduino IDE. (b) The microcontroller can control the digital output pins correctly (c) The microcontroller can transfer data over Serial UART at the baud rate of 9600 and 115200.	1. (a) Burn the bootloader into the microcontroller using Arduino IDE. (b) Make sure the IDE doesn't show any error logs. 2. (a) Write a short LED blink code, and write it into the microcontroller using the Arduino as the ISP. (b) Connect a LED bulb to one of the digital pins, and check whether it can flash. 3. (a) On the basis of the previous code, write a new part of the code that every time the LED blinks, the microcontroller will send the Serial message. Connect the RX/TX pins on the ATMEGA32 to the Arduino. (b) Push the reset button on the Arduino. Open a serial console. Make sure every time the LED blinks, there will be a message sent by the microcontroller displayed in the monitor.	Y

Appendix B Firmware



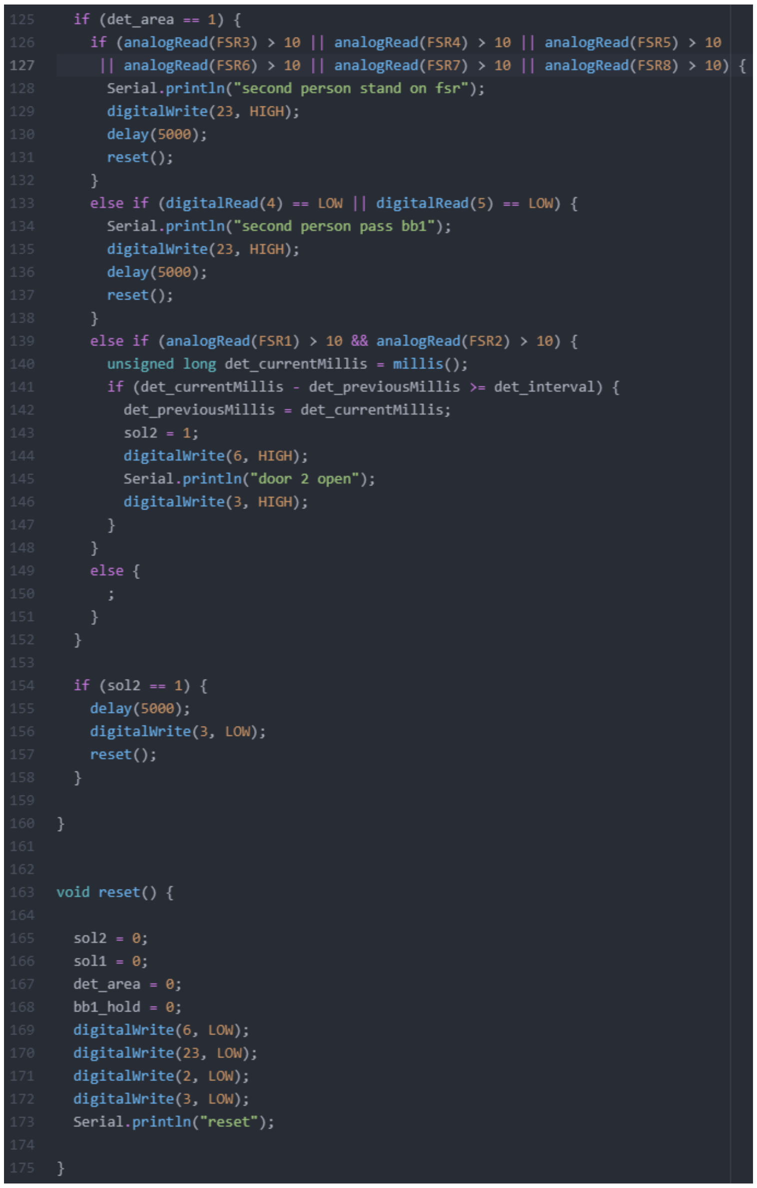
```
#include <ArduinoJson.h>
#include <rBase64.h>

const int FSR1 = A0;
const int FSR2 = A4;


const int FSR3 = A2;
const int FSR4 = A3;
const int FSR5 = A1;
const int FSR6 = A5;
const int FSR7 = A6;
const int FSR8 = A7;

const int EMG = 7;

void setup() {
  // put your setup code here, to run once:
  pinMode(0, OUTPUT);

  pinMode(4, INPUT);
  pinMode(5, INPUT);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);

  pinMode(EMG, INPUT);

  pinMode(6, OUTPUT); // green
  pinMode(23, OUTPUT); // red

  pinMode(2, OUTPUT); // door 1
  pinMode(3, OUTPUT); // door 2

  Serial.begin(115200);

}
String str;
int s = 0;

int bb1_hold = 0 ;
unsigned long bb1_previousMillis = 0;
const long bb1_interval = 2000;

unsigned long det_previousMillis = 0;
const long det_interval = 5000;

int sol1 = 0;
int sol2 = 0;
int det_area = 0;
rBase64generic<250> mybase64;
StaticJsonDocument<200> doc;

void loop() {
  if (Serial.available() > 0) {
    // read the incoming string:
    digitalWrite(0, HIGH);
    String incomingString = Serial.readString();
    mybase64.decode(incomingString);
    String b64 = mybase64.result();

    Serial.print("I received: ");
    Serial.println(incomingString);
```

Figure 14: Firmware Part 1



```
    // prints the received data
    Serial.print("b64: ");
    Serial.println(b64);

    DeserializationError error = deserializeJson(doc, b64);

    const char* name = doc["name"];
    const char* status = doc["status"];
    Serial.print("name");
    Serial.println(name);
    Serial.print("status");
    Serial.println(status);

    if (strcmp(status, "negative") == 0) {
      sol1 = 1;
      Serial.println("NEG!");
    } else {
      Serial.println("NO ENTER!");
      Serial.println(status);
      digitalWrite(23, HIGH);
      delay(3000);
      reset();
    }
  }

  if (digitalRead(EMG) == LOW) {
    Serial.println("emergency");
    Serial.println("door 1 open due to emergency");
    digitalWrite(2, HIGH);
    delay(5000);
    reset();
  }

  if (sol1 == 1) {
    digitalWrite(2, HIGH);
    Serial.println("door 1 open");
    digitalWrite(6, HIGH);
    while (digitalRead(4) == HIGH && digitalRead(5) == HIGH) {
      ;
    }

    bb1_hold = 1;
    sol1 = 0;
    digitalWrite(2, LOW);
    digitalWrite(6, LOW);
    Serial.println("someone pass first bb, door 1 close");
    Serial.println("wait for FSR");
    bb1_previousMillis = millis();
  }

  if (bb1_hold == 1) {

    unsigned long bb1_currentMillis = millis();
    if (bb1_currentMillis - bb1_previousMillis >= bb1_interval) {
      bb1_previousMillis = bb1_currentMillis;
      bb1_hold = 0;
      det_area = 1;
      det_previousMillis = millis();
    }
  }

```

Figure 15: Firmware Part 2



```
  if (det_area == 1) {
    if (analogRead(FSR3) > 10 || analogRead(FSR4) > 10 || analogRead(FSR5) > 10
     || analogRead(FSR6) > 10 || analogRead(FSR7) > 10 || analogRead(FSR8) > 10) {
      Serial.println("second person stand on fsr");
      digitalWrite(23, HIGH);
      delay(5000);
      reset();
    }
    else if (digitalRead(4) == LOW || digitalRead(5) == LOW) {
      Serial.println("second person pass bb1");
      digitalWrite(23, HIGH);
      delay(5000);
      reset();
    }
    else if (analogRead(FSR1) > 10 && analogRead(FSR2) > 10) {
      unsigned long det_currentMillis = millis();
      if (det_currentMillis - det_previousMillis >= det_interval) {
        det_previousMillis = det_currentMillis;
        sol2 = 1;
        digitalWrite(6, HIGH);
        Serial.println("door 2 open");
        digitalWrite(3, HIGH);
      }
    }
    else {
      ;
    }
  }

  if (sol2 == 1) {
    delay(5000);
    digitalWrite(3, LOW);
    reset();
  }

}


void reset() {

  sol2 = 0;
  sol1 = 0;
  det_area = 0;
  bb1_hold = 0;
  digitalWrite(6, LOW);
  digitalWrite(23, LOW);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  Serial.println("reset");

}
```

Figure 16: Firmware Part 3