Door ID

Authors: Emily Wong, Joe Zhao, Jason Huang Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**This project aims to create a door locking and unlocking mechanism that allows users to get in and out of their home using their face as their key. DoorID builds on currently existing ideas of using facial recognition for more secure access to important aspects of users lives, aiming to thwart thieves and unwanted intruders from gaining access to user's homes. It is robust to many different scenarios, including low lighting and power outages, and boasts a 95% accuracy rate to ensure safe and secure homes. Inductive charging also allows users to easily install DoorID without having to make any permanent changes to their home.

***Index Terms*—**Bluetooth, Battery, Depth Sensing, Facial Recognition, Inductive Charging

I. INTRODUCTION

We live in a society where technology surrounds us, both at work and at home. There are smartphones, smart TV's, smart ovens, and so much more. However, there is one part of our homes that hasn't made the transition to tech quite as fast: our doors. Our front doors are the entrance to our safe spaces, where we can relax after a long day at school or work. However, that expectation and trust can be violated when unwanted individuals gain access to our homes without notice. DoorID aims to solve that problem by using facial recognition to control who has access to your home. It is akin to a security guard, who checks the face of every individual who wishes to enter your home, allowing access to only known subjects. To ensure safety, DoorID must have at least a 97% accuracy rate with false positives, and a 93% accuracy rate with false negatives. The accuracy is slightly lower for false negatives because while it is important to allow authorized people in, it is always safer to err on the side of caution, especially when it comes to the safety of our homes. It also must be able to unlock the door, or give a deny decision, within 6 seconds of starting the facial recognition process, since the average amount of time it takes for an individual to open a door with a key is around 6 seconds. To power the device, DoorID needs about 3 Watts sustained, and 6 Watts peak power during camera usage. DoorID must also be able to continue normal function for up to 24 hours, for cases such as a blackout, or even user error such as accidentally unplugging DoorID.

There are a few competitors in this space already, namely Nest and Ring. However, Nest fails to provide such long lasting backup power (only 12 hours), and needs its outside component to be plugged in to a power source, meaning users usually must do some extra wiring work to get the system up and running. Ring is fully battery powered, so while it does better on battery life, it also requires constant battery replacement, unlike DoorID's use of inductive charging to provide power and rare usage of the backup battery. They are also Internet of Things (IoT) devices, making them susceptible to cyber-security vulnerabilities from both developers (bugs and vulnerabilities) and users (psychology). With DoorID, the only threat users really have to worry about are evil twins and dedicated criminals who are willing to break down your door.

II. DESIGN REQUIREMENTS

A. Facial Recognition

Using biometric information to unlock a door should ideally be more convenient and faster than unlocking with a key. Because of this, we took a look at how long it took each of us to unlock our respective doors without keys. On average, it took us around 6 seconds to get our keys from our pocket and unlock the door. That time is longer if we keep our keys in our backpack or bag. This is why we imposed a design requirement that the time from when the user wakes the device to begin facial recognition to the time its unlocked should be 6 seconds or less, on average.

With a lock, the impact of a false positive is far greater than a false negative. The paper that we have chosen to use as a guide achieves 98% accuracy. The paper did not detail the breakdown of false positives vs. false negatives. Because of this, we chose to sacrifice the rate of false negatives in order to achieve lower false positives, as we believe it is a better decision for security reasons to have a fail safe default. That is why we have chosen 3% as the rate of false positives, and 7% as the rate of false negatives. This allows us to have lower false positives, and 7% false negatives means that the expected time it takes to unlock DoorID is still lower than the average key time we found previously.

B. Power

Our power requirements stem from what we believe as essential guarantees on device operation, and convenience. Other products either rely on batteries that must be manually swapped out or must be plugged in on the outside. The first is not convenient, as it requires user intervention, and if the user does not replace the battery, they can be stuck outside. The second requires an AC power cord nearby, and any person could easily unplug it. To solve these issues, our requirement is outside of the case of long-term power outage, the user should not have to replace any batteries, and the user should not have to rely on a plug on the outside of the door.

Since users may be relying on the system to unlock the door, we want to ensure that moderate power outages should not affect the successful operation of the lock. Since the typical power outage in the U.S. lasts around 2-3 hours, we require our system remain operational for 24 hours, meaning that lock will be resistant to most outages. Even if the outage extends beyond 24 hours, as long as the user is able to enter their house in the first 24 hours, they have the opportunity to get a secondary form of access like a key.

C. Servos

Since we are working on actuating a known deadbolt, there is no reason that this process should fail. That is why we have a design requirement that given the face is recognized, the servos should successfully actuate the deadbolt 99.9% of the time.

D. User Feedback

User feedback on what the system is doing is very important. Therefore, we have three LEDs to indicate what state the system is in: red to indicate a failure (deny decision due to an unrecognized face, a failure of the 6 second unlock requirement, or a failure to register a new face into the facial database), yellow to indicate that is in a processing state (running facial recognition algorithm, or taking/saving pictures for registering), green to indicate success (door unlocks or registers user's face successfully). We require these LED signals to always indicate the correct state with 100% accuracy that the system is in, under any conditions.

III. ARCHITECTURE

The overall system architecture of the device is shown in the figure below

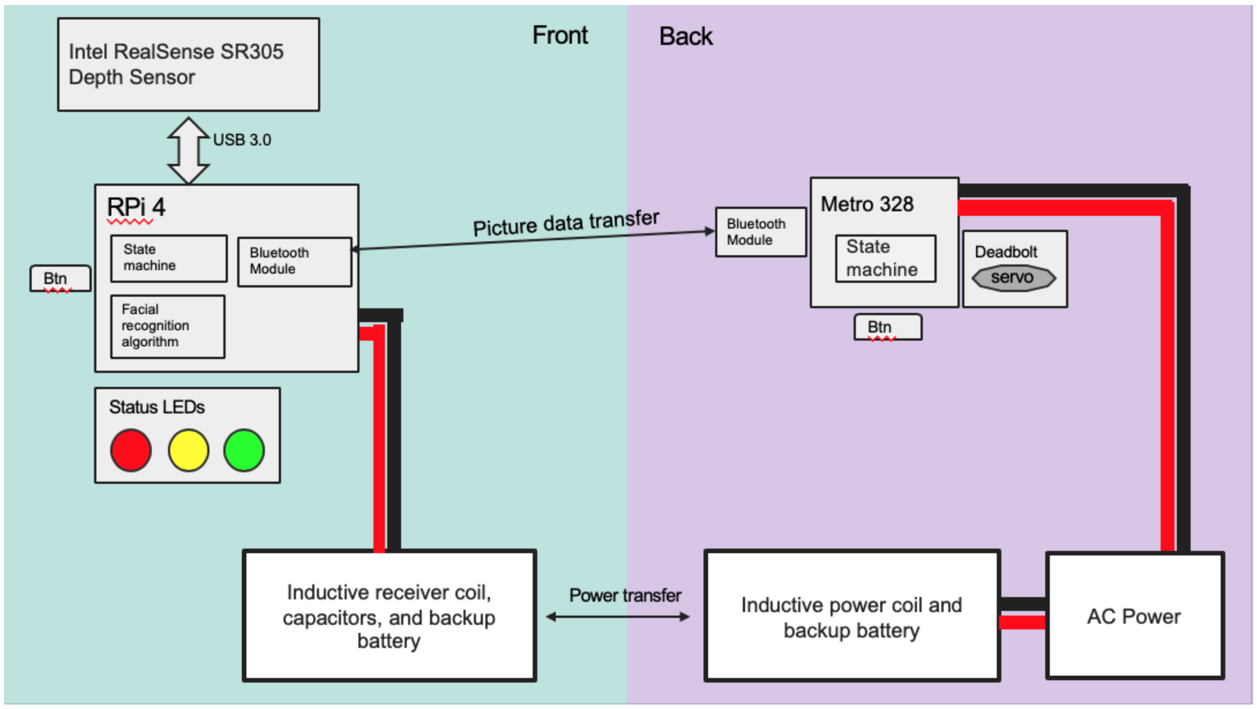


Fig. 1: System architecture diagram

The device is split into two parts. There is a portion mounted to the front of the door, and another to the back. The front of the door contains the Intel RealSense SR305 Depth Sensor, Raspberry Pi 4 (4GB RAM), and status LEDs, and is powered inductively through the door. A Bluetooth transmitter is attached to the RPi to allow it to communicate with the back of the door. The facial recognition algorithm is running on the RPi on the front. A backup battery is also attached to the front to provide power in the case the inductive power is not enough to maintain stable operation (ex: in the event of a blackout).

The second part of the device contains the microcontroller (Metro 328) that drives servos to actuate the deadbolt. In addition, there is a backup battery and appropriate circuitry (including a Class D switching circuit) needed to drive the inductive power coil.

IV. DESIGN TRADE STUDIES

A. Facial Recognition - Choosing a way to get data

For facial recognition, we had two main goals: a low cost system, and fast processing. With some research, and previous experience, we figured out that we could get depth data from one of three ways: time-of-flight cameras, infrared, or using pictures from a few different angles and calculating homographies to get points in a 3D space. After a bit of fiddling around with homographies using 2D rgb pictures from two different camera angles, since that would be the cheapest way to get data, we realized that the face did not have enough distinct points to get accurate enough depth data to make good authorization decisions. It was also very sensitive to changes in lighting and pose, so that was no longer an option.

More research on the time-of-flight sensors uncovered a few things: they are very accurate (down to ~3mm), very fast (since they require only one laser pulse to scan a whole scene), and very expensive. There were a few cheap time-of-flight sensors on Adafruit, but they only use one pulse to find the distance to a single surface. Time-of-flight cameras on the other hand were a bit out of budget, costing over 200 dollars.

Therefore we settled on infrared to do depth sensing. With the camera that we found, the Intel RealSense SR305, we were able to get two cameras in unit (infrared and RGB). With camera, we got 1.5 of our goals, since it was only $79, and it does depth calculations in house.

B. Facial Recognition - Accuracy

After implementing the 3D facial recognition algorithm presented by the paper, we found the accuracy to be lower than expected, at around 72%. We decided to pivot our project slightly and implement a 2D facial recognition algorithm as well, and incorporate the two algorithms together, since the 2D facial recognition algorithm had a much higher accuracy rate of 96%.

Our algorithm calls the 2D facial recognition algorithm first to see if a face is detected and if it is a registered face, and then use the 3D scan to make sure that an actual human being is in front of the camera. If someone were to try to “trick the camera” with a phone image or a printed-out photo of a registered face, the 3D recognition algorithm would detect a flat surface instead of a head-like 3D object.

Our integration of both the 2D and 3D recognition algorithm involves checking a 2D facial match first, and short circuits if the tested face is unregistered. The 3D algorithm is called if the face passes the 2D facial match to do another face check and make sure no one is trying to falsely unlock the door with a picture of a registered user’s face. With this integration, our accuracy went up to 97% false positive accuracy, which reaches our intended accuracy requirements.

Algorithm	**Average Accuracy**	**False Positive Accuracy**	**False Negative Accuracy**
2D	96%	98%	94%
3D	72%	77%	67%
2D + 3D	94%	97%	90%

Table 1: Algorithm accuracies (taken over 50 trials)

C. Facial Recognition - Timing

The average time that our algorithm takes to compute whether a user is registered or not is at around 5.9 seconds, which is less than the average time it takes to open the door manually, which is slightly greater than 6 seconds. We were able to significantly speed up our 2D facial recognition algorithm by storing the encoded facial points of the eyes, eyebrows, lips, nose, and chin instead of the actual photo. Therefore, our code does not have to recompute the encodings of photos over and over again, and instead just reads a .csv file to compare facial points.

D. Communications

Much of the design trade-offs for this subsystem came from the different communication protocols used by the pieces of hardware we wanted to use. We had originally planned to use the data coming from the data lines in the USB 3.0 cable that our camera used using a breakout board, and send that data to the microcontroller on the back of the lock for facial recognition processing. This original plan had two main benefits: not having the main processing unit exposed to the outside world, where it is at risk of being broken/stolen by bad actors, and having only one microcontroller controlling the entire system. However, with some advice from our mentors, we decided not to go with that plan, because we found that raw camera data was also intertwined with USB messaging protocol data, which “corrupted” the pictures that we had wanted to send.

With this change, we also had to change what microcontrollers we were using because we found that our camera was not backwards compatible with USB 2.0, even though in theory it should be. Therefore instead of being able to use the Metro 328 to take in camera data, we put a Raspberry Pi 4 on the front, which has USB 3.0 ports. With this, we decided to also do all the facial recognition processing on the front, as there’s enough processing power in the Pi to do all the matrix calculations. To not waste money, we decided to have the Metro on the back to receive permit/deny instructions and power servos to move the door’s deadbolt instead.

We decided to use bluetooth for our data transfer because it is fast and reliable. It is also able to send data through walls, which we thought was better than drilling a hole through a door and defeating the purpose of having inductive charging to avoid that same problem. Through testing in conjunction with the facial recognition tests, we found that sending data and actuating LEDs/servos took less than 0.1 seconds on average.

E. Power

The lock should remain operable as long as possible, without sacrificing the overall performance. The first component that we considered was the computer that we would use. Looking at other facial recognition techniques on the market, the vast majority leverage machine learning to train a model of the user’s face. This allows them to have extremely accurate detection even as the users face change. Given that machine learning seemed to promise the best results, we looked for boards that could provide the necessary computing power. Of these, the Jetson Nano, Intel Joule, and a few others came to mind.

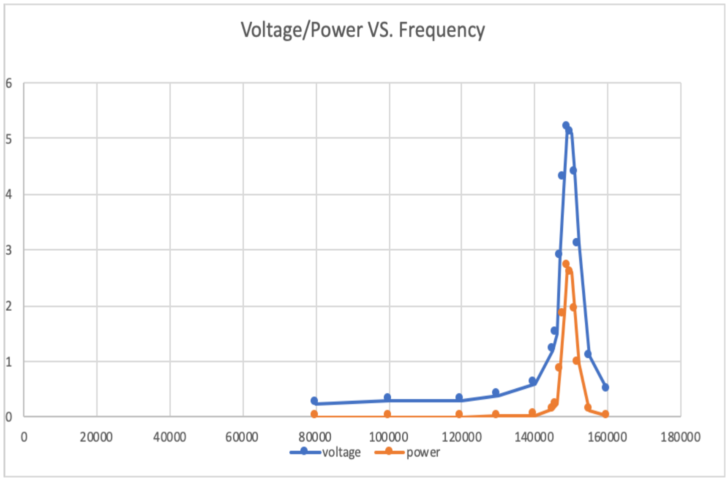
Analyzing the power consumption, a typical board of this capability could draw a maximum of 3-4 amps at 12 volts. This would produce a peak power requirement of almost 50 W. A preliminary analysis allowed us to conclude that it would be impractical to have such a high-power load for 24 hours. Additionally, their idle power draws were also low enough to meet our backup battery requirement. Following this, we decided to trade performance for a lower power draw. The Raspberry Pi was a good middle ground, as it was capable of enough to interface with high data rate cameras and run Linux but has less than half the peak and idle power draw of the previously mentioned devices.

The previous distinction between peak power and idle power is another space that we looked for tradeoffs. Most people are not constantly locking and unlocking their doors.

The usage pattern is most likely once or twice in quick succession, with long periods of time in between uses. If we were to send enough power to sustain peak computation 100% of the time, most of that power would be wasted. However, this would also mean that in the rare occasion that the user does request to unlock the door repeatedly, there would be sufficient power to allow no downtime between uses. Alternatively, we could attempt to adjust the power sent depending on the current power needs, resulting in no wasted power. This would require a way to dynamically change the tuning of the inductive coils, which would likely be very complex. The last option would be to find a middle ground, where the inductive coil would provide slightly more power than the required idle power, and store excess power in a series of capacitors. This would reduce power wasted, while still being able to provide enough power during peak computation.

Ultimately, we believe that the third option would be best for this project. Although it doesn't result in the most optimal power usage, it does significantly reduce the amount of power needed. There is a slight delay when recharging the capacitors, which can be resolved by using larger capacitors that can power several unlock attempts. This option also results in significantly less engineering than the second option of adaptive power.

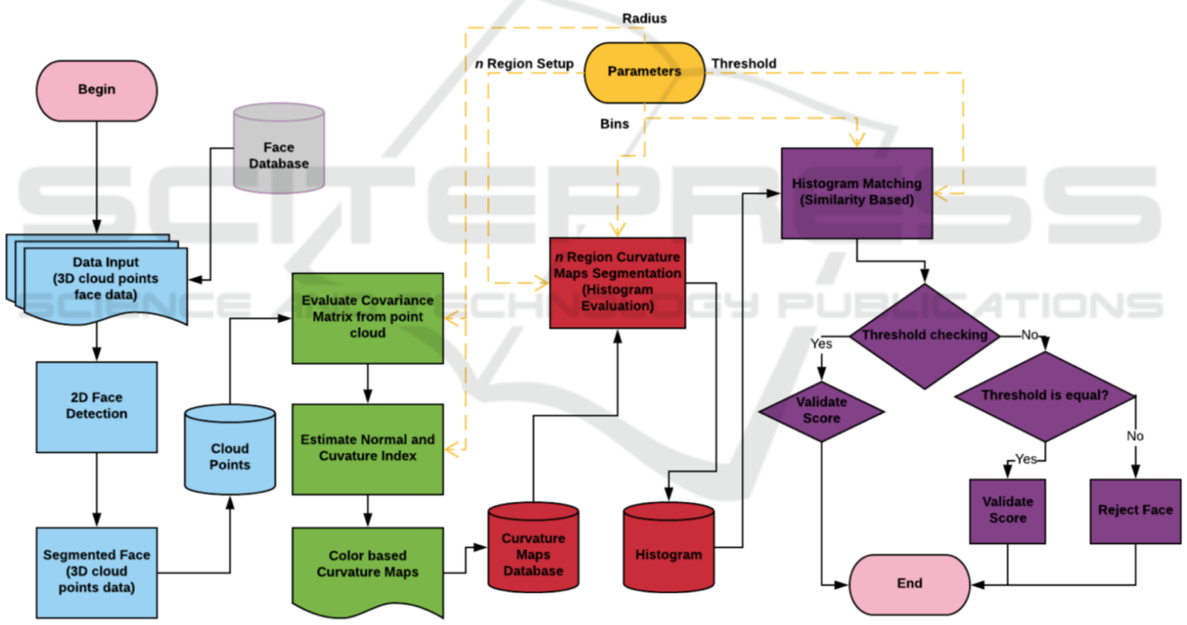
Another tradeoff was on the location of the backup power batteries. It would be considerably less complicated to have only one backup battery, and it would also make replacement easier. However, due to having two separate modules, one backup battery would require us to send backup power through the coils, of which would waste a huge amount of power. From our results for our power transfer circuit, we were only able to achieve approximately 6% efficiency. A battery capable of powering both sides would be enormous, and likely cost too much. The other option, having a backup battery on both sides, would increase complexity, but would enable us to lower the costs and buy smaller batteries for both sides. In the end, using 2 9V batteries in parallel on the back side along with a 26,000 mAh cell phone power bank, we were able to achieve at least 24 hours standby for both sides.



Lastly, there is a large frequency space that the coils could have been operated at. In terms of raw theoretical efficiency, the higher the frequency, the better the efficiency of power transfer. However, there are several factors that prevent this from being the optimal practical setup. First, when sending current through copper wire at high frequency, there is a phenomenon known as the skin effect. Charge begins to accumulate solely in a certain depth from the surface of the wire. This reduces the amount of area that charge can flow through, and results in an increased resistance. High frequency current results in lower ampacity, which would limit the magnitude of the magnetic field the inductor can generate. With a larger budget, Litz wire can be used to circumvent these issues, but unfortunately a large quantity of litz wire would be prohibitively expensive. Second, the amount of time it takes to charge and discharge the gates of the MOSFETs is not affected by the frequency of the coil. With a higher frequency, the charging of the MOSFET gates will represent a larger proportion of time. This will result in more power being wasted by the MOSFETs in linear operation. During the implementation, the charging speed of the MOSFETS became a limiting factor for increasing the frequency. Through better handling of the circuit topology, this was increased from 32khz to 150khz.

V. SYSTEM DESCRIPTION

A. Facial Recognition


Radius
n Region Setup
Parameters
Threshold
Bins
Begin
Face Database
Histogram Matching (Similarity Based)
Data Input (3D cloud points face data)
Evaluate Covariance Matrix from point cloud
n Region Curvature Maps Segmentation (Histogram Evaluation)
Threshold checking
Yes
No
Threshold is equal?
2D Face Detection
Cloud Points
Estimate Normal and Cuvature Index
Validate Score
No
Yes
Validate Score
Reject Face
Segmented Face (3D cloud points data)
Color based Curvature Maps
Curvature Maps Database
Histogram
End

For the facial recognition feature, two algorithms were implemented, a 2D and a 3D version. The 2D version uses dlib to detect 128 facial points that summarize all the main facial features, which are the eyebrows, chin, nose, lips, and eyes. To compare faces, the algorithm computes the “distance” of the facial encodings, which is just the sum of squares of the difference between each of the 128 points.

The 3D version consists of four overall steps. The picture below is a visualization of the algorithm.

Fig. 3: Flowchart for 3D facial recognition algorithm

The first step is the 3D cloud point preprocessing, where we extract the 3D facial data points via our RGB camera. Here’s an example of the results from this step:

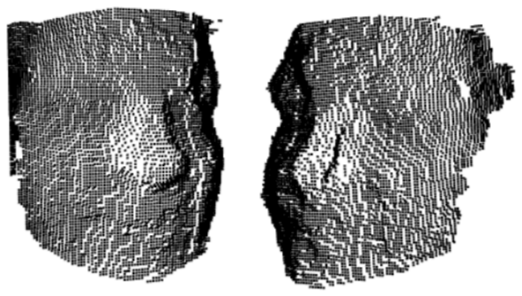


Fig. 4: 3D facial data pointcloud

The second step is to create facial curvature maps. We do this by evaluating the covariance matrix from point cloud data, and we can compute normal curvature indexes by the eigenvalues of the covariance matrix. Here are a few pictures showing the extracted normal indexes and curvature color map:

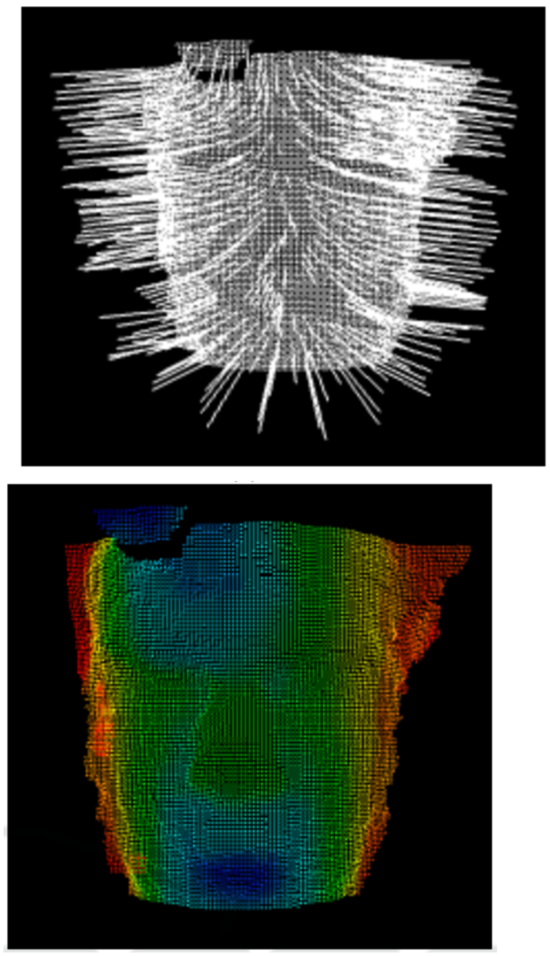


Fig. 5: Facial curvature maps

The next step involves us defining features using a curvature indexes from the previous step, and is used for similarity matching, which is the next step.

The last step is the similarity matching schema. To determine whether the detected face is registered, given input vector of curvature index features (C_V) extracted from previous step and alleged identity vector (C_I), a vector representing the biometric features of registered face, we determine a similarity score of C_V and C_I is greater than or equal to a predefined threshold *t*. If it is greater than or equal to *t*, then the detected face is a registered face, and the door unlocks. Otherwise, our algorithm rejects the detected face, and the door stays locked.

B. Communications

There are two main components to communications so that all the parts from the facial recognition to the power are tied together. On the front (see Fig. 83838), there is the Intel Realsense SR305 camera (used for taking both 2D and 3D pictures of users), a Raspberry Pi 4 (4GB RAM, 16GB SD card), 3 LED’s (red, yellow, green), and a pushbutton. The camera is connected to the Pi using a USB 3.0 to micro b cable, and the status LEDs and push button are connected to the Pi’s GPIO pins with female to male jumper wires.

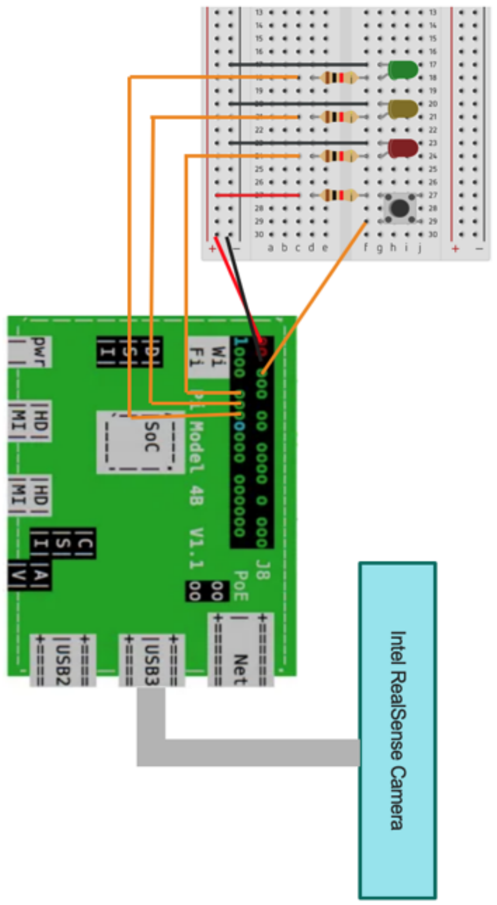
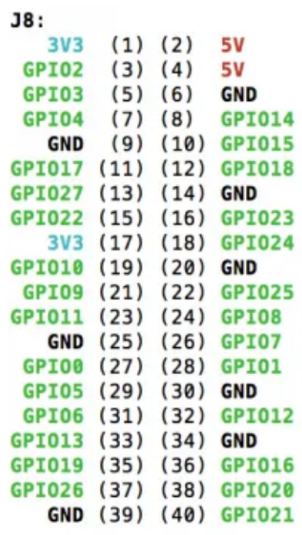


Fig. 6: Wiring for front



```
J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21
```

Fig. 7: Raspberry Pi GPIO pinout

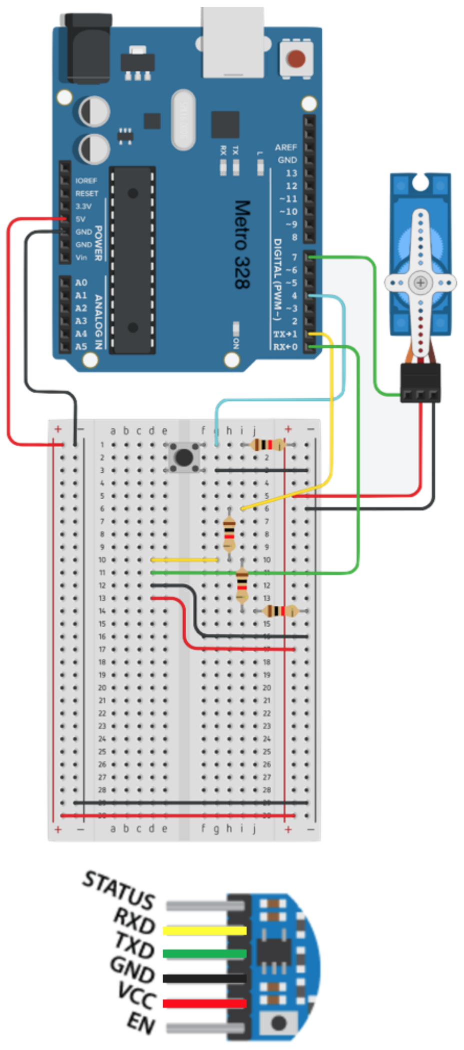


Fig. 8: Wiring for back

First is the communication between the Pi and the camera to take pictures for facial recognition. When the button is pressed, it sends a signal to the Pi to send instructions through the cable to the camera to take pictures using both the RGB and infrared depth camera using the Intel RealSense SDK 2.0 and send that data back to the Pi. From there, we turn on the yellow LED to signal that the Pi is currently processing, hand the raw camera data over to the facial recognition subsystem, and await for a permit or deny decision. In the background, there is a 6 second countdown, where if that limit is reached, all processing stops, and entry is immediately denied, as per our project requirements above. If the facial recognition returns a decision (permit/deny), then using the Pi's on-board bluetooth, the decision is sent to the bluetooth module (HiLetgo HC-05) on the back. The module is connected to a microprocessor (Adafruit Metro 328), which in turn powers the servos to move the deadbolt and unlock the door in the case of a permit. Once the door is unlocked, the Metro sends a success signal to the Pi, again using bluetooth, and the green LED lights up, signifying that the user can now enter. In the case the Metro receives a deny or servos are unable to move, the red LED lights up and a fail signal is sent to the Pi instead.

The second component is communications between the back and the front for registering an authorized face into the system's memory. Again, using bluetooth to serially communicate, when a button is pressed on the back, the Metro sends a signal to the Pi to wake the camera and take a picture after 5 seconds, giving the user enough time to position themselves. Their facial data (in the form of a 2D .png and a 3D .ply) is then processed into and stored in the Pi's SD card, and is now an authorized face for entry.

In terms of software, both the Pi and the Metro have scripts (in Python and Arduino respectively) that keeps track of state, facilitates the sending of serial bluetooth communications, and handles button presses and lighting of status LEDs. The Pi also has a startup bash script to automatically connect the bluetooth to the HC-05 and start running the Python script described above.

C. Power

The power system has two key circuits (See Fig. 9 and Fig. 10). The first is the circuit on the back hooked up to an AC wall adapter, that is responsible for driving an inductive power coil to send sufficient watts to the front as well as hold a backup battery in case of an outage.

The switching circuit used to oscillate current through the inductor is a Class-D switching circuit. It is comprised of a 50% duty cycle PWM signal connected to the gates of a CMOS. The 50% duty cycle PWM is generated using a 555 timer, and by tying the threshold and trigger pins together. The resultant output is then used as input to the gate driver that is used to drive the power MOSFETs. Gate charge time was reduced by using two gate drivers in parallel, and tying both outputs together to reduce noise.

To power the microcontroller on the back, the output of the AC adapter is selected via CMOS against a backup battery, then regulated with a switching voltage regulator to maintain a 5V supply.

The second circuit is on the front of the door. This circuit is responsible for rectifying the output from the receiving coil, charging the capacitors are used to handle peak power, and finally powering the RPi and camera. This is accomplished with a full bridge rectifier, and a voltage regulator between the capacitors and the RPi. Here again, there is a battery to provide backup power.

VI. PROJECT MANAGEMENT

A. Schedule & Team Member Responsibilities

In green, we have the different steps for the facial recognition algorithm we have chosen to implement, which will be written by Jason. In purple, we have Joe's responsibilities, which span the inductive charging, and backup power. In blue, we have Emily's responsibilities, which include setting up communication lines for data flowing between the front and the back of the lock, integrating the camera with the micro controllers, and deadbolt movement. Orange denotes time for all 3 members to work together on integration, final presentation/demo things, and also some extra slack. Our proposed schedule and the actual timeline diverged greatly due to many problems with getting the correct parts, especially with the camera, and parts breaking.

B. Risk Management

The biggest risk that we faced was integrating the power circuits with the circuits/hardware used for communications, mainly because none of us had ever made our own power circuits. To mitigate this risk, we did extensive testing on different test loads before moving on to Raspberry Pi 3s and Arduino Unos (owned previously) to emulate the Pi 4 and Metro. Only after we were able to power those boards were we confident enough to plug in our actual hardware, and thankfully nothing shorted or blew up.

We also ran the risk of running out of our allotted budget because of having to buy multiple cameras due to oddities with Intel's payment system, and not realizing how much hardware we needed for the power circuits. We mitigated this by trying to use as many parts as we could find in the ECE labs and using parts that we already had at home from other projects.

In terms of time management, we were able to get everything done on time because we were able to work on other parts of the project while we were stuck on others, such as being able to work on 2D facial recognition when we began having some camera interfacing issues with the Pi.

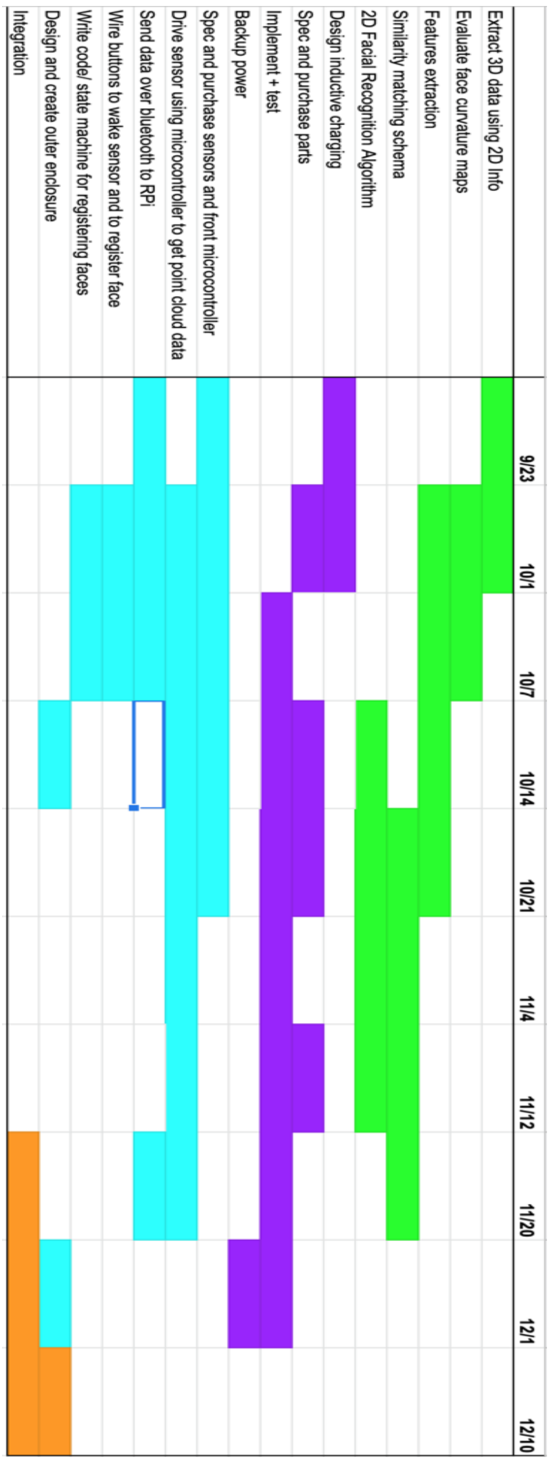


Fig. 11: Timeline of responsibilities & time taken to complete

C. Budget/Bill of Materials

Part	**Subsystem**	**Quantity**	**Total Price**	**Used?**
Door	Aesthetics	1	9.00	Yes
Door handle	Aesthetics	1	5.00	Yes
Intel RealSense SR 300	Facial Recognition	1	91.89	No - bought because Intel didn't let us buy the 305 until later (unknown reason)
Intel RealSense SR 305	Facial Recognition	1	91.89	Yes
Adafruit Metro	Communications	1	25.31	Yes
HiLetgo HC-05 Wireless Bluetooth Modules	Communications	2	15.98	Yes - used 1 because decided to use bluetooth on RPi instead
Pushbuttons	Communications	2	0.00	Yes
LEDs	Communications	3	0.00	Yes
Raspberry Pi 4B (2GB RAM)	Communications, Facial Recognition	1	52.06	No – not enough RAM, also got bricked
Raspberry Pi 4B (4GB RAM)	Communications, Facial Recognition	1	62.06	Yes
USB 3.0 to Micro B cable	Communications	1	5.99	Yes
Deadbolt	Communications	1	13.00	Yes
NMOS	Power	2	2.94	Yes
PMOS	Power	2	2.94	Yes
Clock	Power	1	0.73	Yes
Magnet Wire	Power	1	16.94	Yes
USB C breakout board	Power	1	0	Yes
Voltage regulator	Power	2	0	Yes

Diode	Power	4	0	Yes
Capacitor	Power	6	0	Yes
Battery	Power	2	0	Yes

Table 2: Summary of costs

Total amount spent: 395.73
Total amount remaining: 204.27

D. Tools

Tool	**Subsystem**	**Description**
OpenCV	Facial Recognition	Find bounds of a face to get rid of irrelevant background data
Numpy	Facial Recognition	Make matrix math much faster
dlib	Facial Recognition	Work with 2D facial recognition
plyfile	Facial Recognition	Work with .ply files for 3D facial recognition
Intel RealSense SDK 2.0	Communications	Communicate with camera to get it to take pictures

Table 4: Tools/libraries used in software development

VII. RELATED WORK

Our project is similar to a portion of the Google Home in the sense that Google Home also integrates a facial recognition algorithm into a home setting. Not only can it open doors for familiar faces, it can also display on screen personal data such as calendar appointments and messages. We focused on the facial recognition and door opening aspects of it.

VIII. SUMMARY

Overall, our project was able to meet most of the design specifications. In terms of facial recognition, we had to diverge from our original plan to just rely on 3D data, but overall, our new algorithm still displays accuracy matching the original 3D algorithm. The system is able to operate for more than 24 hours without being plugged in.

We learned a lot over the course of the semester while working on this project. The biggest lesson that we learned was that things never go as expected, and you should try to make as many parts of the project non-dependent on each other. This way you are able to work on other things while you're mitigating the problems from another part of the project. We also learned the importance of taking a break when you've been working on a problem for hours and getting nowhere! Sometimes a well deserved break was all we needed to get things up and running again.

IX. REFERENCES

[1] *3D Face Recognition on Point Cloud Data*, Luis Felipe de Melo Nunes, Caue Zaghetto, and Flavio de Barros Vidal

[2] https://github.com/ageitgey/face_recognition

[3]https://medium.com/@mahesh_joshi/raspberry-pi-3-and-arduino-communication-via-bluetooth-hc-05--5d7b6f162ab3

[4] IntelRealSense SDK 2.0 (https://github.com/IntelRealSense/librealsense)

[5] pyrealsense2 (https://intelrealsense.github.io/librealsense/python_docs/_generated/pyrealsense2.html#module-pyrealsense2)

X. Figures and Charts

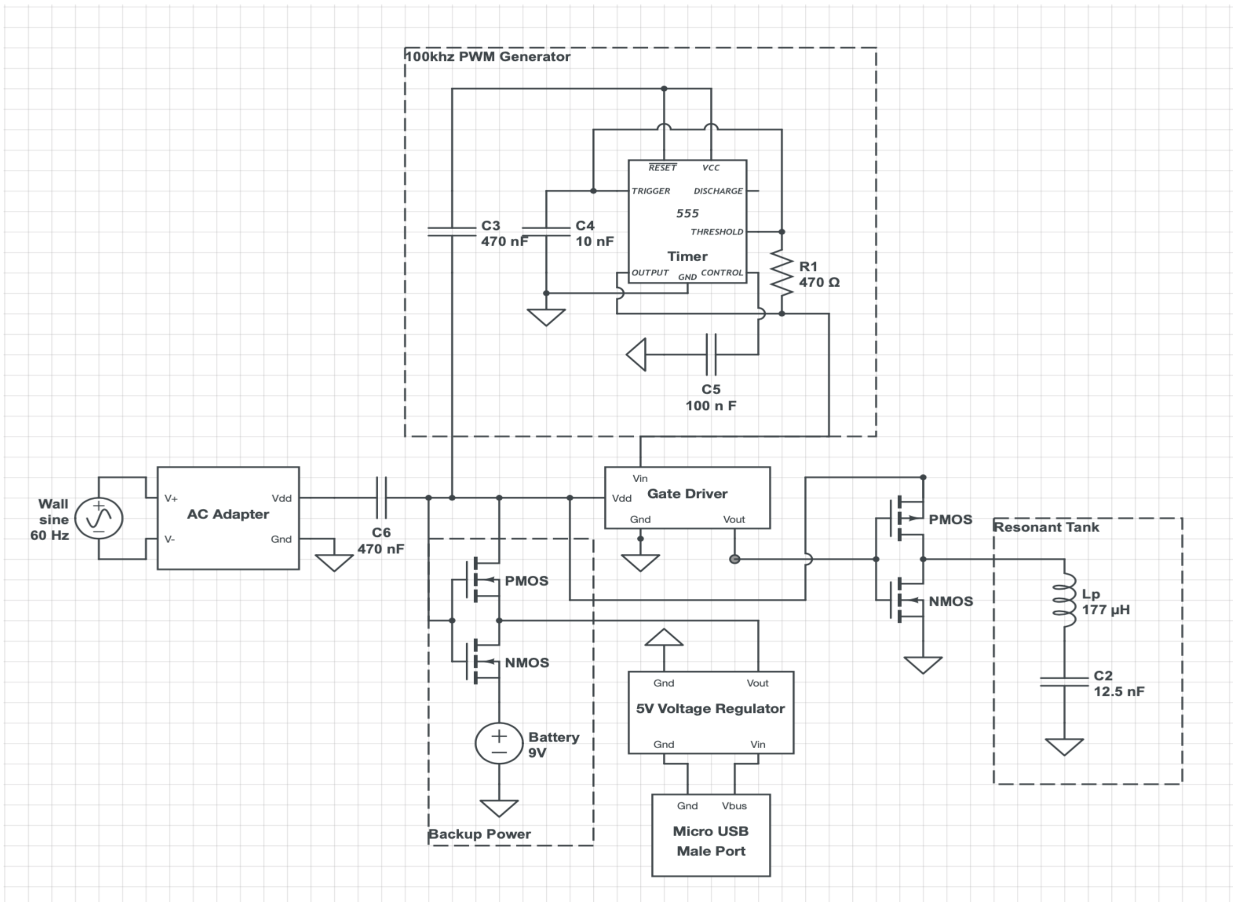


Fig. 9: Back power circuit

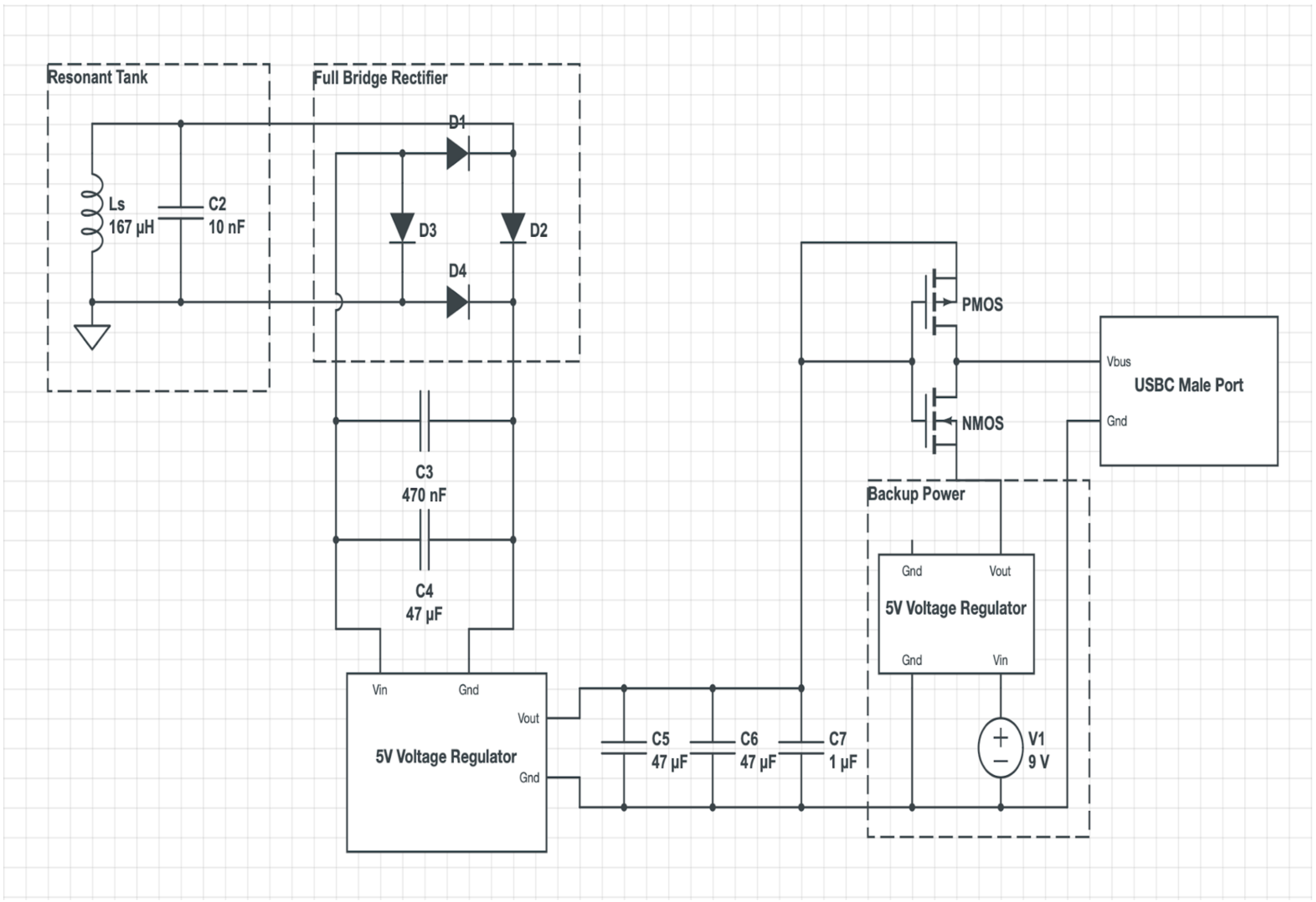


Fig. 10: Front power circuit