2014-2015

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

Senior Design Project:
Smart Door Lock

Project Report

Team: Wilson Mach, Julio Sanchez, Christina Sio, Wai Yin Wong
Advisors: Pat Mantey, Ethan Papp, Jeff Bertalotto
June 12th, 2015

Table of Contents

MOTIVATION 4

SYSTEM OVERVIEW 5

I. MECHANICAL 7

- Door 7
- Generator 9
- Enclosure 13
- Door handle 15

II. ELECTRICAL 17

Energy Considerations 17
- Power Budget 17
- Energy Budget 17
- Power and Energy Analysis 19

Component Selection 20
- Microcontroller 20
- Solar Panels 20
- Rechargeable Batteries 25
- DC/DC Converters 27
- Ideal Diodes 28
- Passive Components 30
- DC Motor for Deadbolt Actuation 31
- DC Motor for Generator 31

Budget 31

System Design 32
- First iteration 32
- Second iteration 33
- Third iteration 34
- Fourth Iteration 36
- Testing 36
- Prototyping 41
- PCB Design 43
- Generator Design 46

III. EMBEDDED SOFTWARE 49

- Overview 49
- Methodology 49
- Results 58
- Summary 61

IV. WEB FRAMEWORK 63

Server 63
Web Server Structure 64
Front-End 66
Web Application 67
Local 67
User Interface Program 67
Website 68
Server Program 71
Front End Program 72

REFERENCES 74

MOTIVATION

Our smart door lock is designed to be installed on the front doors of households to provide security and convenience through self-powered electronics and fingerprint authentication. Many smart products today sacrifice security for convenience, which puts both the owners and other household members at risk. We hope to mitigate the risks currently associated with smart home security systems while providing a convenient alternative to carrying traditional keys.

Security

The advantages of using a fingerprint reader is that although keys can easily be lost or duplicated, fingerprints are significantly less likely to be lost or duplicated. The art of lock picking is also more easily accessible knowledge than lifting fingerprints, simply due to the fact that keys have existed since the 18th century. Thus, we believe that fingerprint readers are a comparably better method of authentication.

Unlike many other fingerprint technologies, we choose to keep the fingerprint profiles on the door lock itself because the perpetrator must be physically present at the front door whereas servers can be hacked remotely. Also, when a server is breached, everyone who owns the door lock can be at risk. This is not true for fingerprints stored in the front door because only the people who are registered with the door will be at risk. Even then, it requires taking apart the physical barriers, choosing the correct wires, and reprogramming it with a hacked microcontroller.

Convenience

If the users forget their key, smart phone, or keyless fob that most other smart locks require to unlock the door, they can still get locked out. In addition, many smart locks employ a wall adapter as the primary source of power. These locks are inoperable during power shortages or battery depletion. Our solution to this problem is powering the system with a generator and solar panel.

SYSTEM OVERVIEW

The system in its entirety is a self-powered smart door lock that provides security and convenience using fingerprint authentication. A block diagram of the system is shown below in Figure 1.

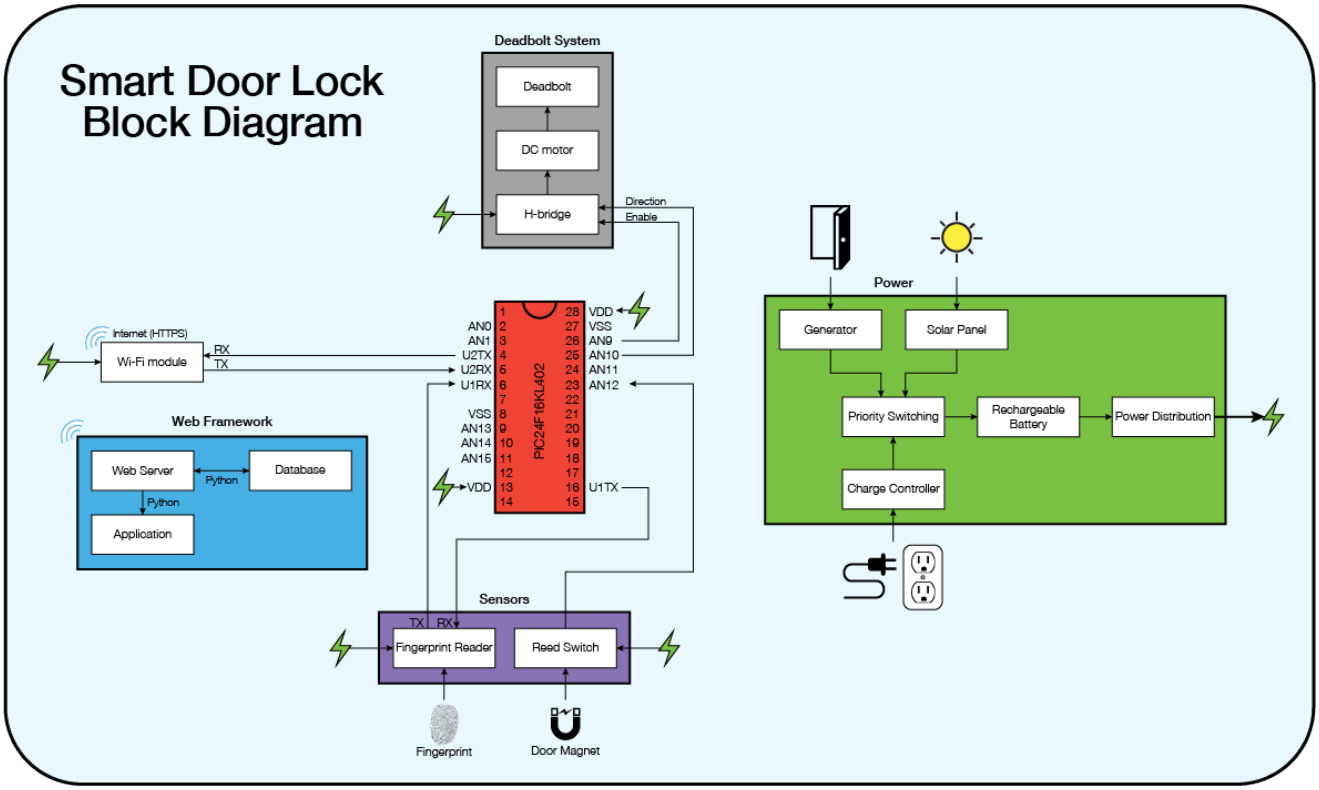


Figure 1. Block Diagram

Power

In addition to the power generated by the solar panel, we also have a generator mounted to the hinge of the door to provide back-up power to prolong the use time of the system even further. The solar panel provides a continuous source of power during the day and the generator provides a burst of power that takes priority for the duration of opening the door. The rechargeable battery provides a voltage rail to power the rest of the system. Once the battery becomes depleted after a year of use, the user can plug the system into an outlet using a wall wart, which will take priority for the duration of charging the battery back to full.

Control

The PIC24 microcontroller is what controls the entire system. It receives input from the fingerprint reader and Wi-Fi chip through UART, and input port feedback from the reed switch and a button which is placed in the front of the door.

Sensors

The fingerprint reader is used to authenticate the user in order to unlock the door. The magnetic contact switch allows us to determine whether the door is open or closed. The momentary push button allows us to identify and scan for a fingerprint.

Deadbolt System

To physically unlock the door, the PIC24 sends a digital and Pulse-Width Modulated (PWM) signal to an H-bridge to control a DC motor that turns the deadbolt.

Web Framework

To authenticate, the fingerprint reader communicates with the PIC24 via UART, which communicates via Wi-Fi to a web server and database of users. This determines if a fingerprint matches with one in the database. If the user is authenticated, the door unlocks and a message is sent over Wi-Fi indicating that a user has entered the home, along with a timestamp of when that user entered.

I. Mechanical

Our system is composed of a significant amount of mechanical design. This began with only the generator, but has since involved designing an enclosure for our electrical components as well as an ergonomic design for integrating the fingerprint reader into an existing door handle. It has also involved building a frame for a full-sized door to fully test our system in a well-simulated environment that allows greater interaction with the system as though it were installed on a real door.

Door

We built the door frame out of 2 by 4's. We bought our supplies from The Home Depot and from San Lorenzo Lumberyard. We used a circular saw to cut out the parts that we needed. This included two planks that were 7 ft long that stood vertically, two planks that were 3.5 ft long that went horizontally across the top and bottom. We also had a 3.25 ft horizontal support that helped prevent lateral movement. We also had three legs that extended out at 45 degrees to provide stability and allowed the door frame to stand upright. We had two of these that were 1.5 ft long on one side. On the opposite side we had a longer 4 ft one. This longer one was placed on the side towards where the door opens. These three legs were bolted down with 5/16 inch diameter, 3 inch long bolts. These were used to secure the legs down and prevent any movement. We had a problem with movement when we just had wood screws on them. We used corner braces/brackets, and die connectors to keep the corners sturdy and to prevent movement.

Figure 2. Door and frame

Figure 3. Frame support and corner braces

Figure 4. Frame support

We bought our door off of craigslist. We attached it to the door frame, making sure that enough space was left around it to allow for movement of opening and closing the door. We had to cut out the holes for where the deadbolt went. We aligned this hole with the preexisting hole for where the door handle fits. We also used a router attachment on the dremel rotary tool to remove material from the door to allow for the deadbolt to properly sit in place.

Generator

The design of our generator system evolved considerably throughout the quarter. We began with the idea that we would use the door handle itself as a cranking mechanism. However, we quickly found out through multiple trials that this would not generate enough energy to be significant. Our next idea is based on a suggestion that the energy used to open the door would be much greater than the small amount from a turn of the handle. Thus, our next idea was born. We designed a system that would be placed below the three preexisting hinges of a door. It would be made of a hinge placed such that it closes in on itself as the default hinges open. We used spacers to place this hinge along the same rotational axis as the other three hinges. We attached a quarter-cut driving gear to the left side of the hinge and a motor with a small driven gear to the right side of the hinge. The idea is that this driving gear will turn the small driven gear as the door opens, and therefore outputting a current and voltage from the generator. Our trials and calculations indicated that this would generate enough power to serve as a backup power source for our system. The electrical characteristics of this system are discussed in further detail in the electrical section of this report.

Figure 5. First iteration of generator attached to mock door

Figure 6. First iteration of generator up-close

The generator system was modeled in its entirety in SolidWorks, and went through several revisions before resulting in a design that worked. The first challenge we faced was to model the gears properly so that we maintain the desired gear ratio of 5:1 while ensuring a proper gear mesh. Shown in the figure below is the first set of printed parts modeled in SolidWorks. One of the issues we ran into when we actually printed our parts for the first time was with the strength of the parts and accuracy of the printer. With IEEE's Ultimaker 1, the accuracy was off by 1mm, which resulted in parts that did not fit properly and we needed to use a Dremel rotary tool to even assemble the entire system. In addition, the fill was set below what we requested and parts broke quickly.

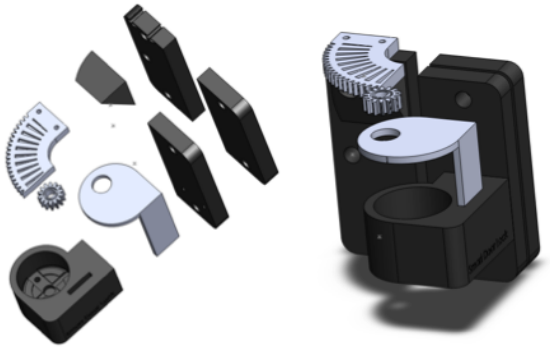


Figure 7. First set of 3D printed parts

After a few more iterations with CITRIS' Ultimaker 2, we also strengthened the parts themselves by removing unnecessary cutouts that were originally added to reduce material weight, in addition to increasing the thickness of parts wherever possible. We realized that weight reduction was a frivolous factor in our application and strength was much more important. We also opted to print the driving gear and spacer that it attached to as one piece rather than using screw holes and t-slot cutouts for that particular pair of parts. Shown in the figure below is the second iteration of 3D printed parts.

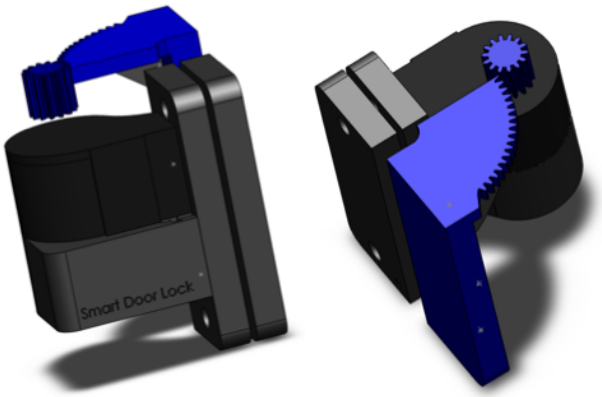


Figure 8. Second iteration of 3D printed parts

Although this new set of 3D printed parts improved the strength of the system, the gears being made of plastic filament was simply inadequate. Thus, our next iteration of our system involves metal gears that we had cut at the machine shop on campus. These new metal gears proved to be a much more durable solution that carried through to the final iteration of our project in the second quarter of senior design. Shown below is an exploded view of the generator system with metal gears, along with aesthetic changes to the overall design.

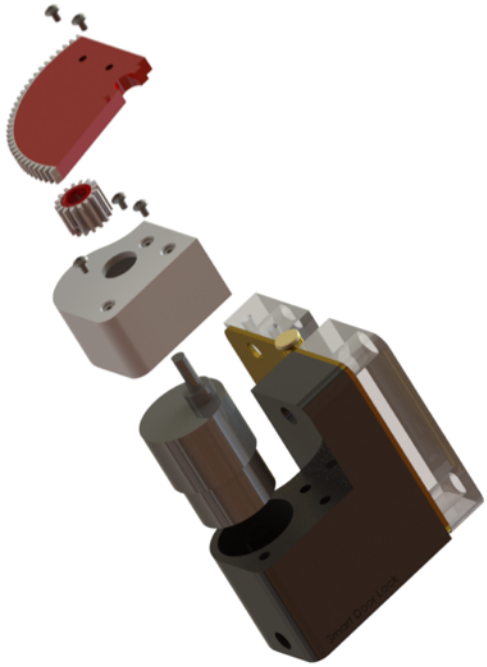


Figure 9. Exploded view of generator system

The figure below shows a photo of the final generator design after we printed all of the 3D parts and mounted them on our full-size door. We were finally able to generator consistent power from the opening of the door while meshing the gears favorably and keeping the shaft of the generator motor steady.

Figure 10. Actual system mounted on full-size door

Enclosure

In addition to the generator, we also have a 3D printed enclosure that houses the solar panel, deadbolt actuation motor, system reset/activate button, PCB, Wi-Fi module, and batteries. This is comparably the most complex 3D printed part in our system due to the size constraint and number of parts that need to fit precisely into the enclosure.

The design of this enclosure began with the angle of the solar panel, which we study in the electrical section of this report. Since the optimal angle we found to harvest the maximum amount of power from the sun is 60º, we designed for the tilt to be 60º in reference to the door. Then, we fit the deadbolt actuation motor into the enclosure under the solar panel, and add a compartment below it to house the PCB. Thus, the deadbolt motor essentially sits on a shelf within the enclosure. To route the wires within the enclosure, a slit is made on the shelf to allow the wires from the solar panel and deadbolt motor to attach to the PCB. To account for the wires under the solar panel, we add a U-shaped cutout to the enclosure so that the solar panel can slide in and sit flush against the enclosure. In order to hold the solar panel in place, we add a cover with U-shaped pegs that slot into U-shaped holes on the enclosure. The pegs hold the cover in place, and the cover holds the solar panel in place. To mount the enclosure to the wall, four screw holes are added – two above the enclosure and two below. To allow for the wall charger to be plugged into the circuit, a round cut-out is added to the cover. These ideas can be illustrated in the figures that follow.

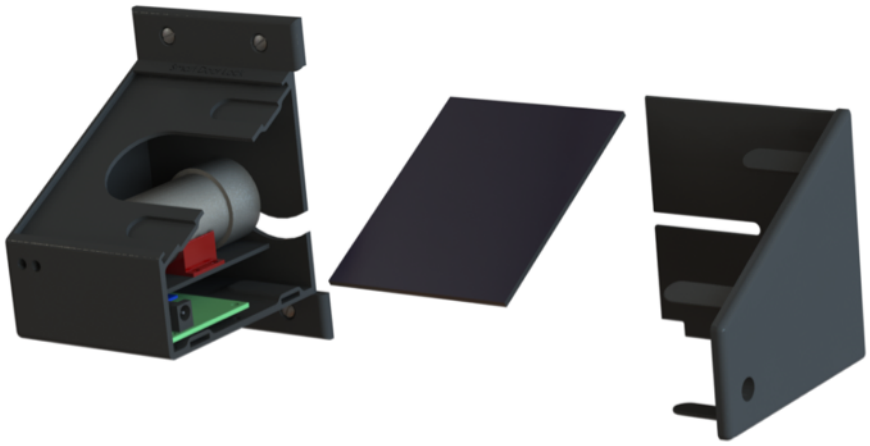


Figure 11. SolidWorks rendering of all enclosure components

To allow the motor to be inserted into the enclosure easily, an arc is cut out so that the shaft of the motor can swing into position. To keep the shaft from moving after insertion of the motor into the enclosure, the cut-out has a long U-shaped cut-out that holds the motor in place. This is shown in the figure below.

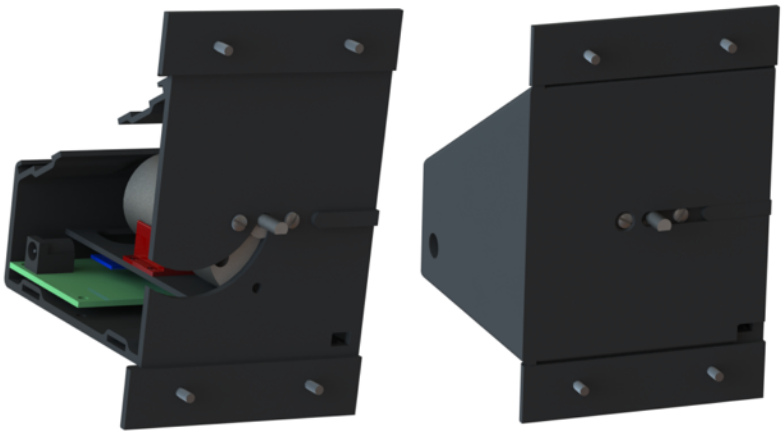


Figure 12. Back of the enclosure, without cover (left) and with cover (right)

Once the design was completed, we printed the enclosure and added our components to it. Shown below is a photo of the final completed enclosure design mounted on our full-sized door.

Figure 13. Final printed design of enclosure

Door handle

To integrate the fingerprint reader ergonomically into an existing door handle that we obtained from Home Depot, we began by modeling the door handle itself in SolidWorks, as well as the fingerprint reader that we had been testing with. Upon doing so, we noticed that it was going to be quite a tight fit. It proved to be a challenge fitting the fingerprint reader into the door handle, and in order to do so we made compromises with our design. This included allowing the wires to be visible from the back of the handle, as well as cutting away more metal than anticipated. Although this was not a dealbreaker in our design, it is one area that has much room for improvement in future iterations of the door handle. Shown in the figure below is the model that we began with.

Figure 14. 3D Model of door handle with ergonomically integrated fingerprint reader

One of the other challenges that we came across when designing this handle is evident in the following figure, which shows what the back of the fingerprint reader looks like:

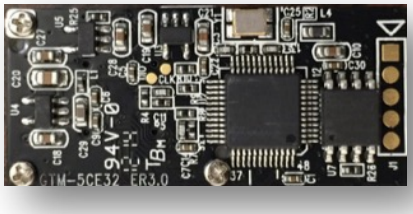


Figure 15. Back of fingerprint reader plate showing exposed circuitry

Since the door handle itself is a large piece of metal, we noticed an immediate problem. The exposed circuitry on the back of the fingerprint reader could potentially short if contact is made between the connections on the back of the board and the door handle. Thus, we came up with the following solution:

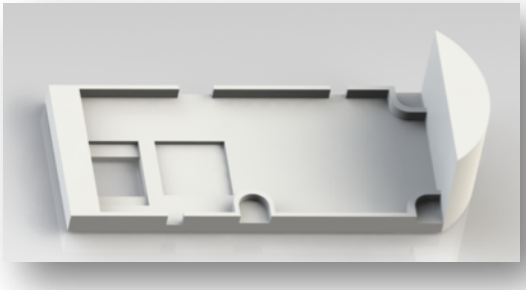


Figure 16. Rendering of fingerprint reader plate

The figure shown above is a plate that fits between the fingerprint reader and door handle, preventing any and all metal-on-metal contact between the two components. The plate is designed to account for varying heights of components on the back of the fingerprint reader, and a semicircle is added to the edge such that the area cut out of the door handle in order to slot the fingerprint reader in, is now filled by plastic filament, as can be seen in the final design shown in the figure below.

Figure 17. Final design of door handle with fingerprint reader integrated

II. Electrical

Energy Considerations

The first thing that we considered when designing our system was its energy requirements. This determines what components will be needed to power the system. In other words, we need to ensure that the energy generated must exceed the energy consumed to prevent a deficit.

Power Budget

We began by tabulating the power consumption of the system components with the nominal current and voltage found in the data sheet for each component, and calculated power from this, as shown in Table 1.

Table 1. Power Budget

Power consumed by each component						
Description	**Current (nA)**	**Current (uA)**	**Current (mA)**	**Current (A)**	**Voltage (V)**	**Power(W)**
Pic24 (active)		154		0.000154	1.8	0.0002772
Pic24 (HLVD)		5.24		0.00000524	1.8	0.000009432
Pic24 (sleep)	30			0.00000003	1.8	0.000000054
(ESP8266) Wifi (active)			135	0.135	3.3	0.4455
(ESP8266) Wifi (inactive)			1.2	0.0012	3.3	0.00396
(SP8266) Wifi (shutdown)						
H-bridge			280	0.28	3.3	0.924
(GT - 511C3) FR			130	0.13	3.3	0.429
Reed switch			1	0.001000	1.8	0.0018

Energy Budget

Once we obtained the power budget, we tabulated the energy consumption, from which we determined the daily energy requirements of our system components. This is based on the amount of time that each component is active per day. These calculations are then used to

determine how much energy we will need to generate with the solar panel and generator. Using the power budget shown above, we obtained the energy budget shown below in Table 3.

Table 2. Calculations for Energy Budget

Activity Frequency	
Activity	Frequency
Bursts per day	11
Time (s) per burst	0.5
Fingerprint enrollments per day	3
Time (s) per enrollment	120
Deletes per day	3
Time (s) per delete	2

Time spent in different modes		
Modes	Time (s)	Time (Hr)
Burst	5.5	0.001527777778
UI	120	0.03333333333
Idle		23.85736111
Identification	22	0.006111111111
Enroll	360	0.1
Delete	6	0.001666666667

Current used in different modes	
	current (A)
Burst	0.410154
UI	0.265154
Idle	0.000154
Identification	0.130154
Enroll	0.131354
Delete	0.131354
Total	1.068324

We first determined the frequency of each activity in the system, and the time spent in various modes of our system. This allowed us to determine how much energy each component used per day based on the amount of time it was used per day.

Table 3. Energy budget

Energy used in different modes (Wh)							
Mode	**Pic24 (active)**	**(ESP8266) Wifi (active)**	**(ESP8266) Wifi (inactive)**	**H-bridge**	**(GT - 511C3) FR**	**Reed Switch**	**Total (Wh)**
Burst	0.0000004			0.00141167	0.000655417		0.00206750
UI	0.000009	0.01485			0.0143		0.02915924
Idle	0.006613						0.00661326
Identify	0.0000017				0.002621667		0.00262336
Enroll	0.000027		0.000396		0.0429	0.000277	0.04360092
Delete	0.0000005		0.0000066		0.000715	0.000277	0.00099926
						Total	**0.08506355**

Our calculations indicate that we consume 85 mWh per day. Thus, we need to size a solar panel appropriately to sustain the system for a sufficient amount of time. This was also when

we looked for ways to reduce the energy usage of the system. We asked ourselves the question of whether each component must be active for the amount of time that they are. Using “sleep” modes and “enable” inputs on components when they are not truly needed saved a tremendous amount of unnecessary energy consumption.

Power and Energy Analysis

Upon the completion of our system with software and hardware fully integrated, we tested the actual energy consumption of the system to compare with the values we obtained theoretically. Shown in the table below are the results of this analysis.

Theoretical		Measured		Difference	
Current used in each mode		**Current used in each mode**		**Current used in each mode**	
Mode	**current (mA)**	**Mode**	**Current (mA)**	**Mode**	**Current (mA)**
Burst	410.124	Burst	473.124	Burst	63.000
UI	275.124	UI	169.124	UI	-106.000
Idle	0.124	Idle	0.0001	Idle	-0.124
Identification	130.124	Identification	125.124	Identification	-5.000
Enroll	205.124	Enroll	200.124	Enroll	-5.000
Delete	205.124	Delete	170.124	Delete	-35.000
Total	**1225.744**	**Total**	**1137.620**	**Total**	**-88.124**
Energy used in each mode		**Energy used in each mode**		**Energy used in each mode**	
Mode	**Energy (Wh)**	**Mode**	**Energy (Wh)**	**Mode**	**Energy (Wh)**
Burst	0.000689	Burst	0.002385	Burst	0.001696
UI	0.010088	UI	0.018604	UI	0.008516
Idle	0.008928	Idle	0.000008	Idle	-0.008920
Identification	0.002624	Identification	0.002523	Identification	-0.000101
Enroll	0.067691	Enroll	0.066041	Enroll	-0.001650
Delete	0.000564	Delete	0.000936	Delete	0.000372
Total	**0.09058**	**Total**	**0.09050**	**Total**	**-0.00009**

Theoretical		Measured		Difference	
Power (W)	**Energy (Wh)**	**Power (W)**	**Energy (Wh)**	**Power (W)**	**Energy (Wh)**
0.935000	0.066854	0.058240	0.061152	-0.876760	-0.051063
Power (W)	**Energy (Wh)**	**Power (W)**	**Energy (Wh)**	**Power (W)**	**Energy (Wh)**
0.008252	0.000075647	0.1308	0.001199	0.122548	0.001123353
Total	0.06693	**Total**	0.06235	**Total**	-0.04994

As indicated by the table above, we had a theoretical power consumption of 90.6 mW, actual power consumption of 90.5 mW, theoretical energy generation of 66.9 mWh, and actual energy generation of 62.4 mWh. This resulted in a theoretical lifetime of 376 days, and actual system life of 313 days.

Component Selection

Microcontroller

This is the “brain” of the entire system that connects all the components together, receives input from the two sensors and sends a control signal to unlock or lock the door. Our method of selection included choosing a microcontroller that had a sufficient number of input and output ports for all of our components, making sure that we had at least one Pulse Width Modulation output to control the H-bridge, and making sure that we had two UARTs to communicate with the fingerprint reader and Wi-Fi module. We also needed an analog-to-digital converter to read in analog signals. Finally, an internal comparator was considered to compare analog inputs for hysteresis. However, this final requirement is optional, because we may also choose to use software hysteresis.

Solar Panels

As our primary source of power, this was a critical decision in ensuring that the system has enough power to sustain itself without manually recharging it.

Size vs. power output vs. cost

We begin with a comparison of size, power output, and cost. In general, the size of the solar panel determines the amount of power that it can output, and as a result, cost. However, at very fractional differences, this may not be the case. This is especially true at smaller panel sizes. In our application, we sought out a 1 W panel, with the assumption that much of this nominal power would be lost in non-idealities and lack of sun in certain circumstances. A few of the panels and their specifications are tabulated as follows, with the 80 mm x 100 mm being our top choice.

Table 4. Solar Panel Area vs. Power

Area	Power
80 mm x 100 mm	.935 W
90 mm x 90 mm	1.015 W
112 mm x 84 mm	1.098 W
130 mm x 150 mm	2.5 W
150 mm x 86 mm	1.62 W
136 mm x 110 mm	1.98 W
165 mm x 135 mm	3.498 W

From the above comparisons, there is only a significant increase in power if the size of the panel is increased by more than 300 mm^2 in area. For a small increase in size of (112 mm - 100 mm) x (84 mm - 80 mm) = 48 mm^2, there is only an increase in power of (1.098 W - .935 W) = .163 W. This additional 163 mW could be significant enough for a minuscule power deficit, but without a deficit there is no need to make changes to the physical design of the system for the small gain in power. For the same area but different dimensions (square 90 mm x 90 mm), there is a (1.015 W - .935 W) = .08 W or 80 mW increase in power. Again, this would be an alternative panel to consider if we needed to overcome a small power deficit but a rectangular shape allows the other components (motor and batteries) to fit better while positioning the panel at an optimum angle. It should be noted that the square panel also has high voltage and less current (5.5 V, 170 mA vs. 7 V, 145 mA). This means that we could add more batteries in series for greater capacity but slower charging speed. This is a trade-off that may be important to consider in certain applications.

Measuring light intensity

To determine the feasibility of using solar panels, we measured the light intensity in various conditions using a lux meter. This included days with high cloud coverage and generally bad weather conditions, as well as the most ideal case of abundant sunlight. We found that in the highest intensity of light, our lux meter read around 90,000 lux, while in moderate light we obtained about 8000 lux. Finally, in low sun we were able to see 2000 lux. This corresponds to a best, average, and worst case scenario. This is taken into consideration when we revisit our energy budget in terms of energy generation.

Optimum Angle

The next factor we considered was the angle of the solar panels with respect to the sun for maximum efficiency. This is an important consideration because the position of the sun changes with the seasons. In winter, the sun is lower in the sky, while the opposite is true for the summer, as shown below. The optimum angle also varies with geographic location. We considered implementing an adjustable tilt solar panel mount on our system, but decided against this to avoid unnecessary complexity. Instead, we opted for the angle that would provide the most efficiency year-round. That angle is 60º.

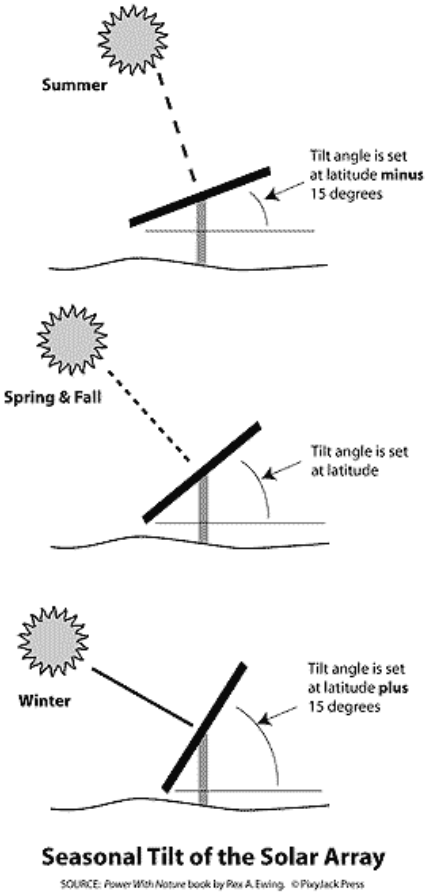


Figure 18. Angle of tilt for solar panels

Plotting the I-V characteristic curve

To find out the actual power output of the solar panels that we chose, we conducted an experiment around campus in various light intensities using two multimeters, a lux meter, a small solar panel, and a very simple circuit. The schematic is shown below.

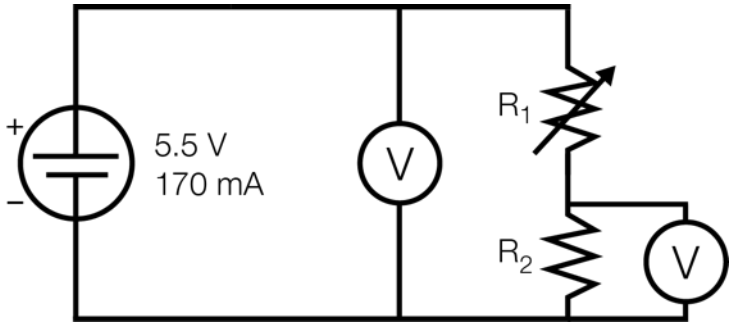


Figure 19. Experiment schematic

Figure 20. Experiment set-up

We used one multimeter to measure the voltage across the solar panel and another to measure the voltage across a resistor to calculate current. This allows us to graph the solar panel's I-V characteristic curve. A typical curve is shown in Figure 9, and the actual curve we obtained is shown in Figure 10. From this data, we may also determine the fill factor and efficiency of the solar panel at several light intensities, using the following formulae:

$$\eta = \frac{(ff * I_{sc} * V_{oc})}{(E * A)}$$

$$ff = \frac{(V_{max} * I_{max})}{(V_{oc} * I_{sc})}$$

The efficiency is a measure of the power output of the solar panel with respect to light intensity and area of the panel. The fill factor is a measure of the actual power output compared to the maximum rated output of the panel.

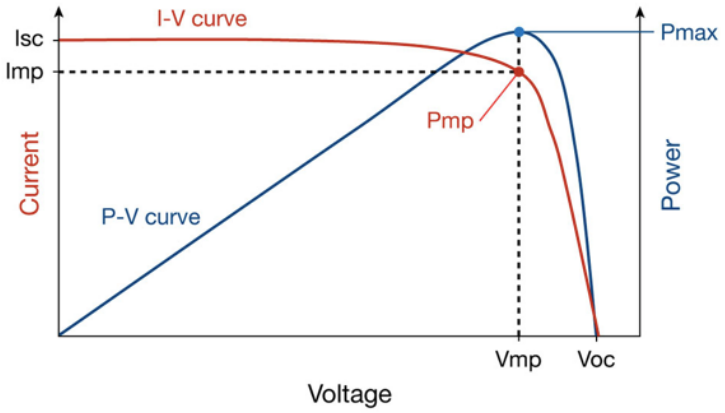


Figure 21. Typical solar I-V characteristic curve

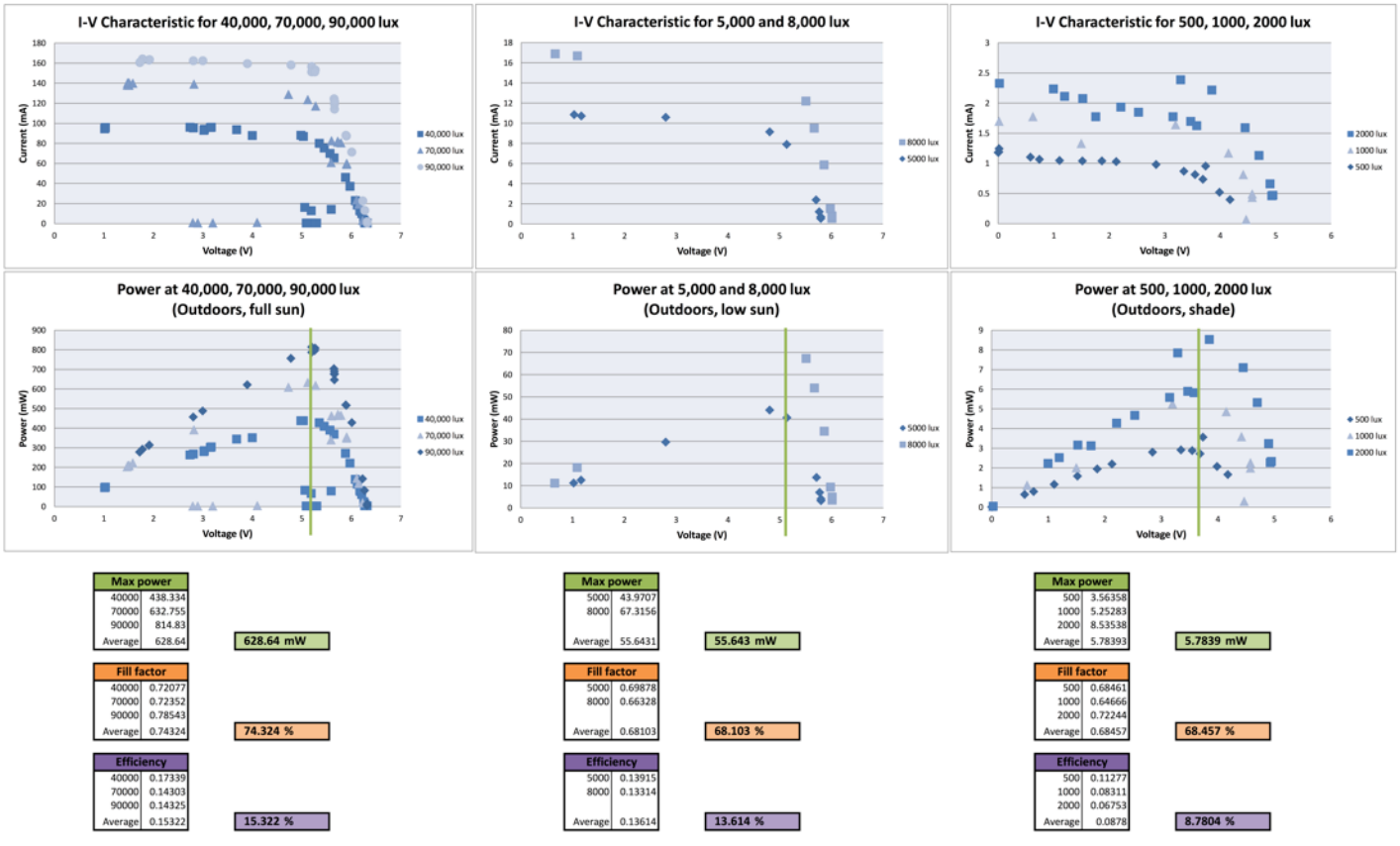


Figure 22. Experiment results

The line indicates where the maximum power point is for each case. This is the point at which power is maximized. There are integrated circuits (IC's) that implement maximum power point tracking (MPPT) to ensure that the maximum power is harvested in any light intensity.

From our results we obtained a worst case, average case, and best case, as shown in Table 5 below.

Table 5. Energy generated by solar panels

Energy generated by solar panels							
	Voltage (V)	Current (mA)	Power (W)	Light Intensity (lux)	Hours of light	Efficiency	Energy (Wh)
Solar panel (Best Case)	5.2	156.6981	0.81483	90000	10	0.14325	1.167243975
Solar panel (Average Case)	5.51	12.21698	0.0673156	8000	7	0.13314	0.06273679289
Solar Panel (Worst Case)	3.85	2.216981	0.00853538	2000	5	0.06753	0.002881971057
Solar Panel (Absolute Worst Case)	3.74	0.95283	0.00356358	500	3	0.11277	0.00120559475

Rechargeable Batteries

Battery Chemistry

We chose NiMH for our system due to its high energy density and inexpensiveness. In addition, they are readily available in standard sizes. Thus, although Li-Ion outperforms NiMH in every other way, it was not commonly available in standard sizes. The Li-Ion batteries we obtained were found to be unstable and dangerous. To avoid subjecting our users to this danger we opted for NiMH, which are much less dangerous should charging fail unexpectedly.

Battery Management

It's important to consider the charging characteristics of different battery chemistries. Some are more tolerable of certain charging techniques while others are not. Rechargeable NiMH batteries are generally not very tolerant of trickle charging indefinitely, and can be overcharged during fast charging, thereby requiring a charge controller for fast charge. The general rate of charge recommended for fast charging is 1C, while for slow or trickle charging is between C/10 and C/40. The charging characteristic curve is shown below.

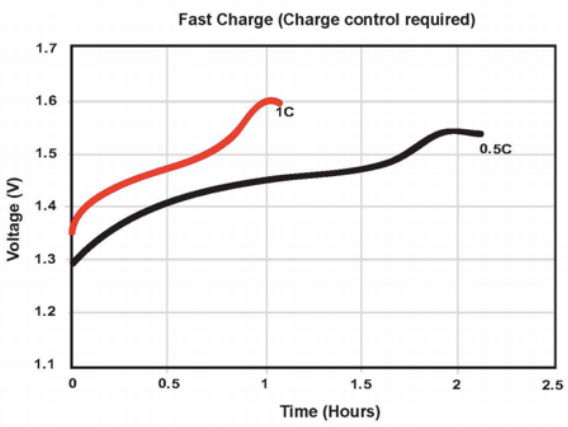


Figure 23. Charging characteristic curve of NiMH batteries

From the charging characteristic curve, we see that the voltage rises sharply as the battery approaches full capacity. There are a number of methods for charge termination of the battery, which is especially critical when fast charging the batteries.

If we consider the change in voltage as the battery approaches full charge, we see that it rises sharply, is constant for a moment, and immediately falls. This gives us three methods of charge termination methods using voltage:

- $-\Delta V$ (negative voltage slope)
- $0\Delta V$ (zero voltage slope)
- $\Delta V/\Delta t$ (decreasing positive voltage slope)

Next, we may consider the change in temperature as the battery approaches full charge, and see that it rises very sharply, comparably more than the voltage curve. We can monitor the temperature of the battery in relation to the ambient temperature, or the change in temperature of the cell itself:

- ΔT (cell temperature vs. ambient temperature)
- $\Delta T/\Delta t$ (cell temperature over time)

One issue to consider when monitoring voltage is that we may not have a stable reference voltage if we are measuring the voltage of the same battery the microcontroller is powered by. This is why we use a separate voltage reference in the charge controller chip.

However, a charge controller is not needed when trickle charging due to the inherent characteristic of using solar panels. A charge controller implements current limiting, along with a maximum charge timer. The experimental results in Figure 10 indicate that the current from the solar panel will never exceed 170 mA and the sun is only out for up to 14 hours a day – and not even at full lux every hour. A charge controller would ensure that charge doesn't exceed 245 mA and for no longer than 12 hours.

It is equally important to monitor the discharging characteristic curve of the battery. This shows how quickly the battery can be discharged at different rates of current pull, and allows us to determine when the battery becomes depleted.

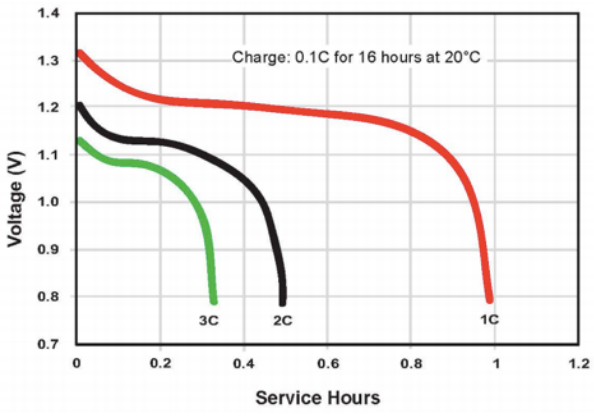


Figure 24. Discharge characteristic curve of NiMH batteries

DC/DC Converters

Linear regulators vs. Switching regulators

Although we originally considered only using linear regulators in our circuit, we analyzed the difference in power efficiency and decided that the trade-off between power efficiency and complexity of the circuit was well worth it. Linear regulators typically have an efficiency between 50-80%, while switching regulators typically have an efficiency over 90%. Switching regulators

achieve this high degree of efficiency because rather than continuously dissipating extra power as heat, they "switch" small amounts of power to the output from the input as needed, using a PWM from a microcontroller for example. The power analysis of the power dissipated by a low dropout regulator is shown below in Table 6.

Table 6. Power dissipation by a low dropout voltage regulator

	Power dissipated by low dropout voltage regulator Max output voltage from battery = 3.6 V						
	Voltage (V)	Desired Voltage (V)	Difference (V)	Current consumed (A)	Time on (hours)	Power dissipated (W)	Energy dissipated (Wh)
Pic24 (active)	3.3	1.8	1.5	0.000154	24	0.000231	0.005544
Pic24 (HLVD)	3.6	1.8	1.8	0.00000524	0	0.000009432	0
Pic24 (sleep)	3.6	1.8	1.8	0.00000003	0	0.000000054	0
(ESP8266) Wifi (active)	3.6	3.3	0.3	0.135	0.03333333333	0.0405	0.00135
(ESP8266) Wifi (inactive)	3.6	3.3	0.3	0.0012	0.1016666667	0.00036	0.0000366
(SP8266) Wifi (total shutdown	3.6	0	3.6		0	0	0
H-bridge	3.6	3.3	0.3	0.28	0.002083333333	0.084	0.000175
(GT - 511C3) FR	3.6	3.3	0.3	0.13	0.1454166667	0.039	0.00567125
Reed switch	3.3	1.8	1.5	0.001000	0.1016666667	0.0015	0.0001525
Total						**0.165600486**	**0.01292935**

Legend
1.8 V components
3.3 V components
unused

	V_in	V_out		I_load		P_waste	
	Voltage (V)	Desired Voltage (V)	Difference (V)	Load Current (A)	Power 1 (W)	Power 2 (W)	Energy (Wh)
1.8 V components	3.3	1.8	1.5	**0.001154**	0.001731	0.001731	0.0056965
3.3 V components	3.6	3.3	0.3	**0.5462**	0.16386	0.16386	0.00723285
Total				0.547354	0.165591	**0.165591**	**0.01292935**

Efficiency (%)
54.54545455
91.66666667

From the calculation above we see that for the 1.8 V components, the efficiency is only 54%. This can be increased to 80% by using a switching regulator instead, otherwise known as a DC/DC converter.

More specifically, we need one buck-boost converter to buck or boost 2.7 V - 4.2 V to 3.3 V and one buck converter to buck 3.3 V down to 1.8 V.

Ideal Diodes

Standard diodes typically exhibit a high voltage drop of .7 V while Schottky diodes have a lower voltage drop of .45 V. Ideal diodes however have a significantly lower voltage drop of .015 V. This is because ideal diodes are actually composed of a diode and MOSFET. Shown in Figure 13 is a comparison of the voltage drop between an ideal diode vs. Schottky diode with respect to load current.

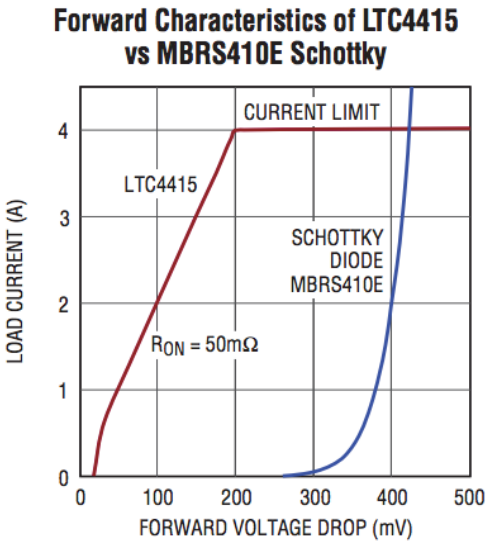


Figure 25. Comparison of forward voltage

Ideal diodes come in IC's that allow a special form of "Diode-ORing" to prioritize between two power sources. For example, if our primary source of power is from the solar panel and generator but we have a backup source such as a wall wart, then we can "program" this IC to allow the wall wart priority when its voltage exceeds the solar panel's using external resistors. The LTC4415 shown in Figure 14 is what we chose for this purpose. The resistors between IN1, EN1, and EN2 determine the proportion of input voltages needed to prioritize the primary source over the secondary source.

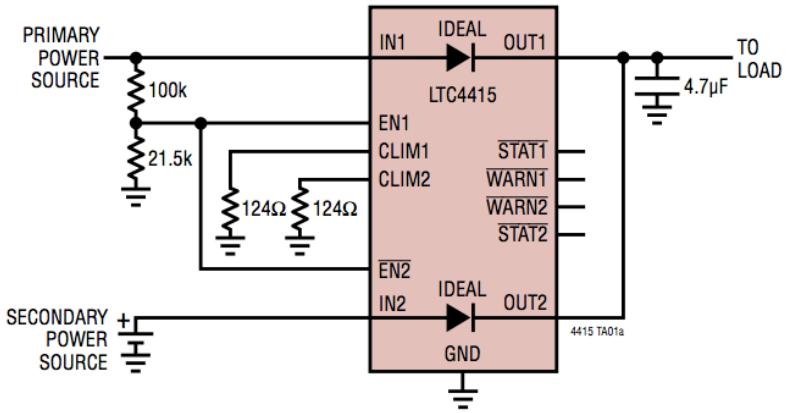


Figure 26. LTC 4415 – Dual 4A Ideal Diodes with Adjustable Current Limit

Passive Components

We used surface mount components that came in the 0805 package. We bought our surface mount resistors and capacitors from Digikey and we sampled all of our surface mount inductors from CoilCraft.

Capacitors

We chose ceramic capacitors because they are best suited for DC/DC converters that deal with less than 500 W. Ceramic capacitors have a capacity range between 1 pF and 100 μF, with a voltage rating between 6.3 V and 4000 V. The minimum capacitance needed can be determined using the formula below.

$$C_{min} = \frac{I_{out} * D * (1 - D) * 1000}{f * V_{p_{max}}}, D = \frac{V_{out}}{V_{in} * \eta}$$

Inductors

Inductors determine the ripple current sourced to the output capacitor, thereby affecting the ripple voltage. Ferrite core inductors with high frequency reduce power losses. Inductors must be low ESR and support peak current without saturating. Shielded inductors also minimize radiated noise. The minimum inductance needed can be calculated using the formulae below.

$$L_{min,buck} = \frac{V_{out}(V_{in} - V_{out})}{f * V_{in} * I_p}$$

$$L_{min,boost} = \frac{V_{out}(V_{out} - V_{in})}{f * V_{out} * I_p}$$

In summary, the values we took into consideration to calculate the minimum capacitor and inductor values in a DC/DC converter were:

- maximum input voltage
- maximum output voltage
- switching frequency
- maximum ripple voltage

- maximum ripple current
- duty cycle

DC Motor for Deadbolt Actuation

We used a 13 Volt DC motor, rated at 700 rpm. For the deadbolt application we found that we need to run the motor at 3.3V at 278 mA for about half a second to be able to turn the deadbolt from fully locked, to fully retract. This motor was purchased from MPJA.

DC Motor for Generator

We used a 12 Volt DC motor, rated for 30 RPM. This is a high torque motor that has a mini gear box. The 30 RPM rating was what we needed based on the 5 to 1 gear ratio. With the door being opened at 6 RPM, it should turn this generator at its rated 30 RPM for max efficiency.

Budget

Our total costs came out to be $973.49. We received funding from some of our colleges, and subtracting that gives us an out of pocket expenses of $398.49. This cost included the two trips to the machine shop, the PCB run, parts for the door frame and the door itself, electronic components, and other miscellaneous items. In production, the price of this would reduce significantly because the machine shop cost would lower as we increase our quantity, and this is true also for the PCB run, and for the 3D printing. For our product we have 3D printed parts which we were able to make at no cost. We have bolts, washers, and nuts which we purchased for $5. We have the two gears which we also got for free, but getting them cut by the machine shop cost $84. The motor that we use as a generator was $14, and the motor that we use for the deadbolt was $10. All the solder, flux, and wire were provided at no cost. The PCB board was $30, and all the ICs were sampled for free. The passive components were $20. The connectors were $5. The solar panel was $10. The door handle was $20, and the machine shop cost for it was $210. The wifi chip was $7, and the fingerprint sensor was $56. The deadbolt was $16. Subtracting the cost of the machine shop, and PCB run, we end up with a total price for parts of $143.

System Design

First iteration

The first iteration of the system consisted of three simple concepts:

1. Switching the power source off and on using power FETs controlled by the PIC24
2. Regulating the power from the battery to the components using linear regulators
3. Monitoring the charge of the battery using the internal Analog-to-Digital converter (ADC)

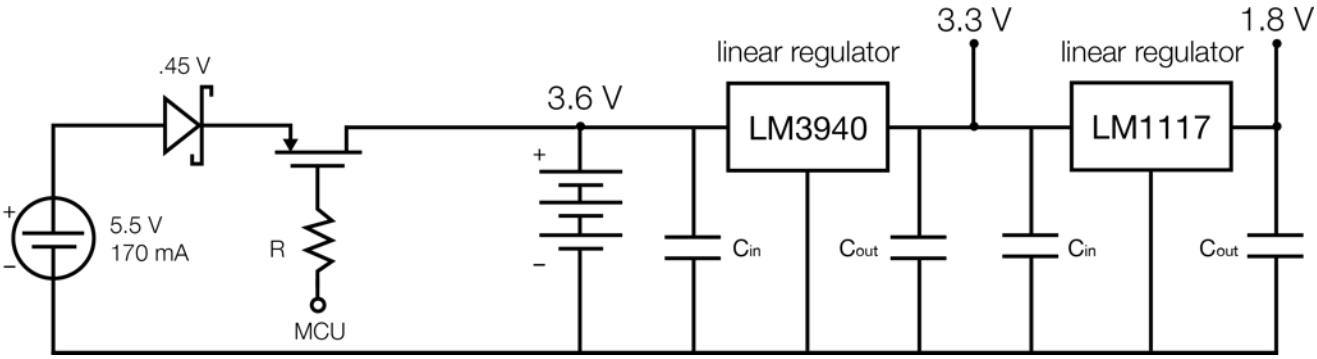


Figure 27. Circuit schematic of first iteration

This iteration of the system had a very low level of complexity, at the expense of higher power consumption. The switching circuit was a simple FET controlled by the PIC24, which received feedback from the voltage of the battery, which was monitored by the internal ADC in the PIC24. If the battery exceeded a certain level, the PIC24 would pull the FET low to turn it off and cease charging of the battery.

The power from the battery was then regulated to 3.3 V and 1.8 V using low drop-out linear regulators (LDO's). As discussion in the previous section of this paper, linear regulators are extremely inefficient, and in this case was as low as 54%.

Another issue with this iteration is the method of battery charge monitoring. Since the PIC24 is powered by the battery, the internal voltage reference in the ADC of the PIC24 is unstable. This may result in over or under-charging the battery due to inaccurate voltage measurements.

Second iteration

The second iteration of the system consisted of two primary parts:

- Battery Management
  - Charge Control and Monitoring using one magical integrated circuit chip
- Power Regulation
  - DC/DC conversion to two separate rails
    - Buck-boost to 3.3 V
    - Buck to 1.8 V

This iteration of the system fixes a few issues from the first iteration, particularly with regards to battery management and charge control. It also rectifies the issue of high power consumption when regulating the voltage from the battery to the components.

The battery management is done using one IC that is capable of both fast charging and slow charging, as well as monitoring the voltage and temperature of the battery using its own internal circuitry. The MAX712 is designed specifically for NiMH/NiCd batteries and is able to cease fast-charging when the battery is determined to be full and begin trickle-charging. It can detect this change using both voltage and temperature for greater accuracy of charge termination. The graphs showing the operation of this IC are shown below in Figure 17, from which we can observe the changes in current with respect to the status of the battery.

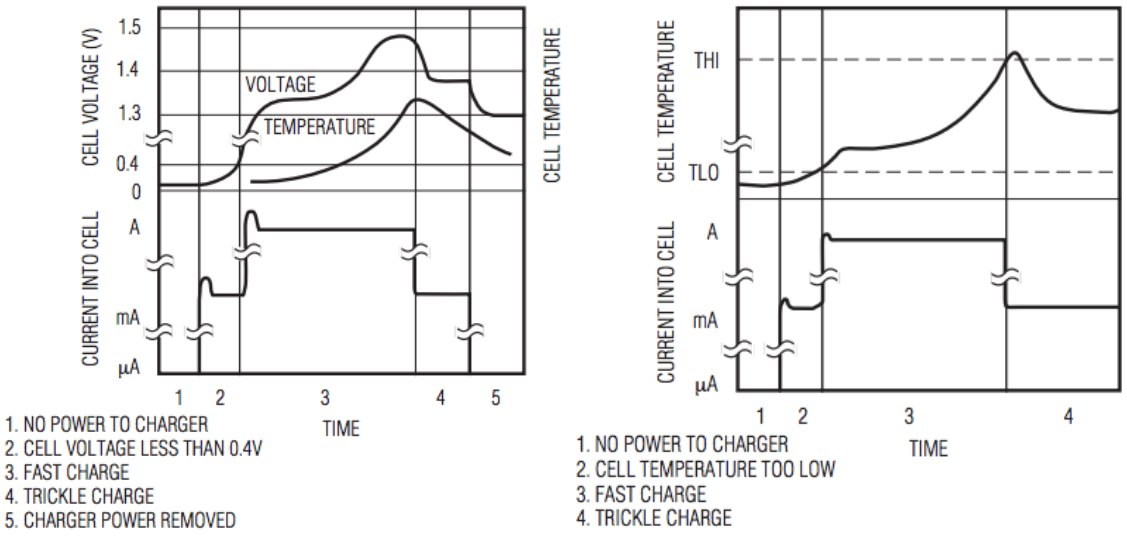


Figure 28. Typical Charging Using Voltage Slope and Temperature

There is no switching from power sources, but rather, a Schottky diode is placed after each source to prevent reverse current from flowing into each source from another. The source that produces the most voltage at any point in time will be the one that charges the battery.

The linear regulators have been swapped out for DC/DC converters, one buck-boost and one buck. This results in reduced power consumption and heat dissipation.

Third iteration

The third iteration involves a tripartite system:

1. Switching between three power sources using ideal diodes
2. Charge control between fast and slow charge
3. Regulating to two rails, as in the second iteration

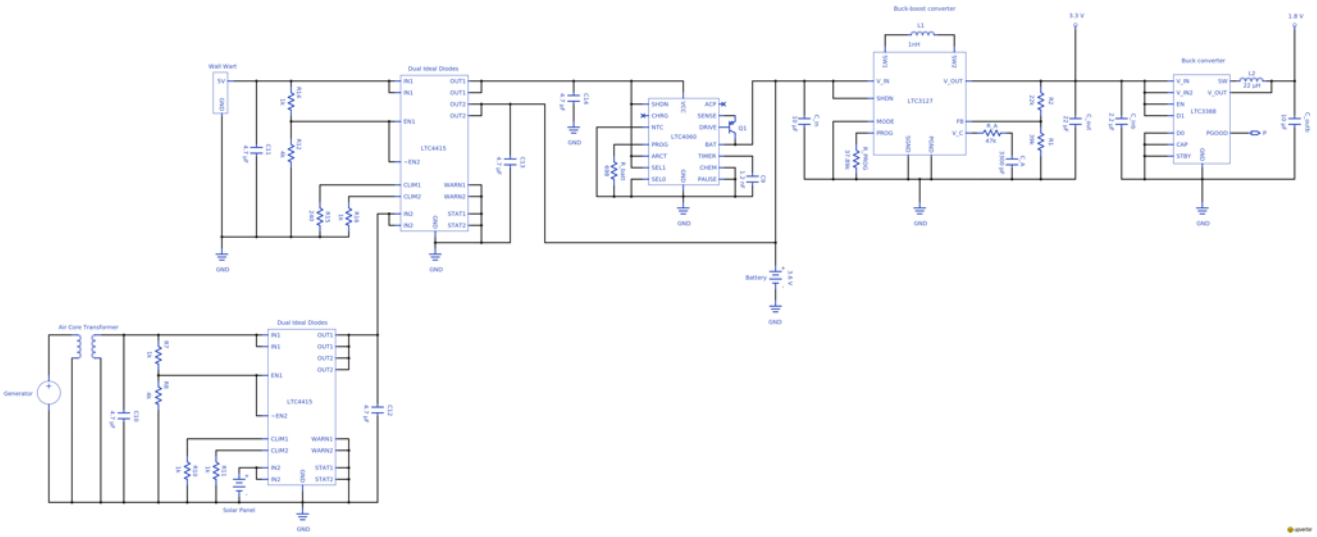


Figure 29. Full Charging Circuit

This final iteration further improves upon the changes made in the second iteration. There is now a formal switching circuit that prioritizes between three power sources. This is done using two cascaded dual ideal diode ICs. Each dual ideal diode IC essentially behaves as a logical OR gate. As explained in the component selection section however, they also allow implementation of special prioritizing of power sources using proportional voltage levels.

The solar panel is the primary source of power, but if the voltage of the generator exceeds that of the solar panel, only the generator is passed through to the battery. Similarly, if the voltage of the wall wart exceeds that of the generator or solar panel, then the wall wart takes priority. In addition, if the wall wart is prioritized, it is put through a fast charging IC (LTC4060) to fast charge the battery.

If the solar panel or generator is prioritized, then there is no charge controller needed for trickle charging. Trickle charging IC's typically perform their duty by charging the battery at a rate between C/10 and C/40 for a length of time between 10 - 14 hours. The solar panel automatically does this already, because we know from the experiment performed in Figure 10 that the current never exceeds 245 mA (C/10). In fact, it never even exceeds 170 mA. Furthermore, the sun is only out for 14 hours a day at the most, and not even at full intensity. Thus, no charge controller is needed since the solar panels already behave like one.

For completeness, the components and their respective circuits are shown below in Figure 19.

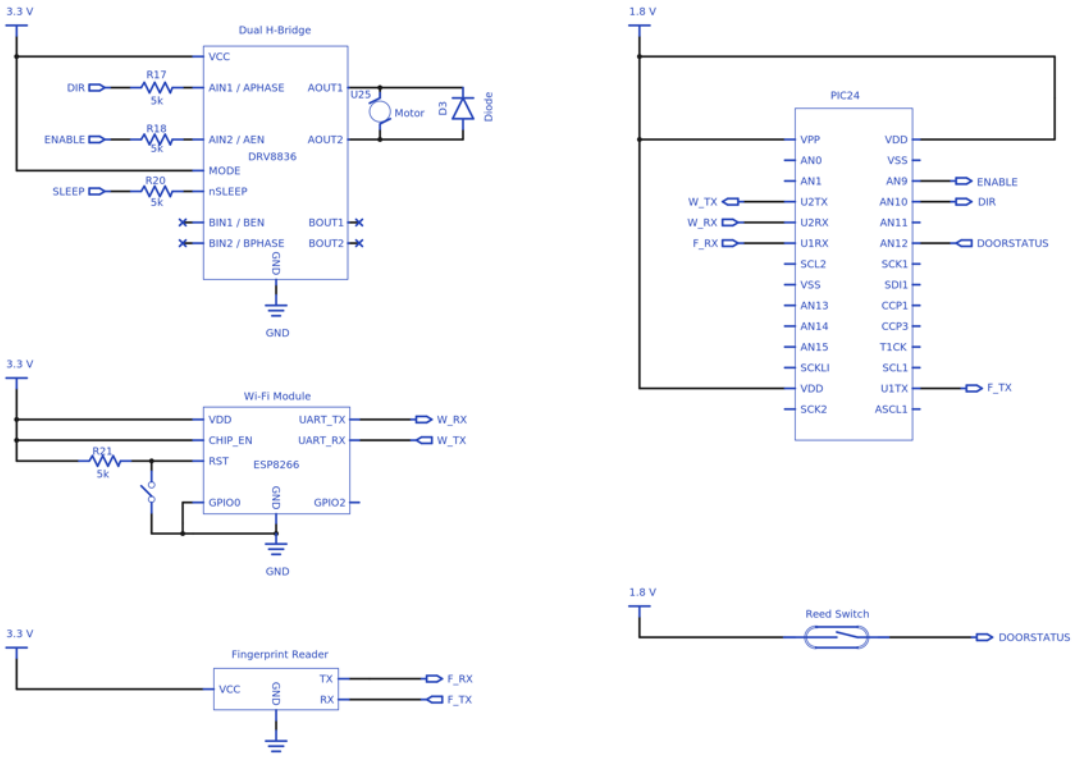


Figure 30. Component Circuits

Fourth Iteration

The fourth and final iteration of our system was completed in the second quarter of Senior Design.

One of the changes that was made accounted for when we realized upon further testing that our microcontroller is unable to function properly at 1.8 V. Thus, we removed our 1.8 V rail and powered all of our components from 3.3 V. This eliminates the need for the LTC3388 buck converter in the converter stage of our circuit. We instead used another version of this converter IC to step down the voltage from the generator to be below the maximum rating of 5.5 V for the ideal diode chips.

Another change that was made involves conserving more energy in the system by adding MOSFETs to drive the fingerprint reader and Wi-Fi module, which are two of the most power consuming components in our system. The FETs are controlled by the microcontroller to only power these components when they are needed. Otherwise, the components are kept off. This results in a significant reduction in power consumption, as detailed in the power analysis section of this report.

One small but useful addition to the circuit is the inclusion of two status indicator lights. These consist of one red LED and one green LED. The red LED indicates that the system is busy if it is blinking, or that an incorrect fingerprint has been detected if it is solid. The green LED indicates that the correct fingerprint has successfully been identified.

Testing

Once we designed our rechargeable battery circuit, we built it on breadboard and perfboard to do testing.

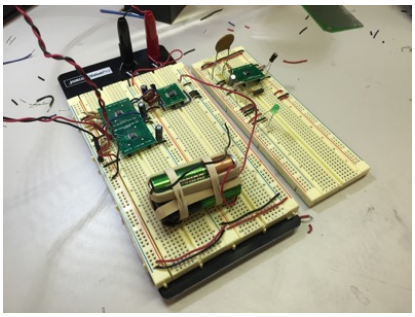


Figure 31. Initial breadboarding phase

To do this we required that we use breakout boards. These were needed because none of our ICs were available in through-hole packages. The breakout boards allowed us to test these chips on a breadboard and perfboard.

We used perfboard for the switching regulators because they weren't operating correctly when tested on a breadboard, and gave us the correct output when soldered onto a perfboard. We first tested the voltage regulator for the generator. The generator outputs a voltage that swings from 12V – 18V depending on how fast the door is being opened. We used an LTC3388-3 to efficiently drop down this voltage to 5V so that we could use it with the ideal diodes chip.

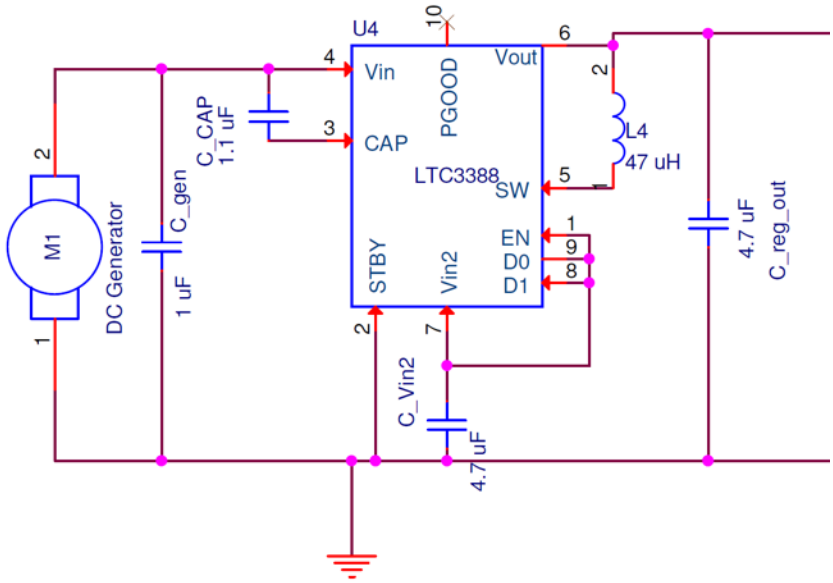


Figure 32. Generator and buck converter

We have bypass capacitors on the output of the generator and on the output of the regulator to help reduce any unwanted noise. We connected a capacitor between the Vin pin and the CAP pin to set an internal voltage rail that is used by the internal circuitry. We also connect a capacitor from the Vin2 pin to ground to set another internal voltage rail. This voltage rail is used by the enable input, and it is also used to set D0 and D1 high. The 47 µH inductor connected from the output pin to the SW pin is used to optimize the performance of the regulator.

We then tested the priority switching part of the circuit. To simulate the generator and the wall charger we used the power supply, and we were able to test directly with the solar panel.

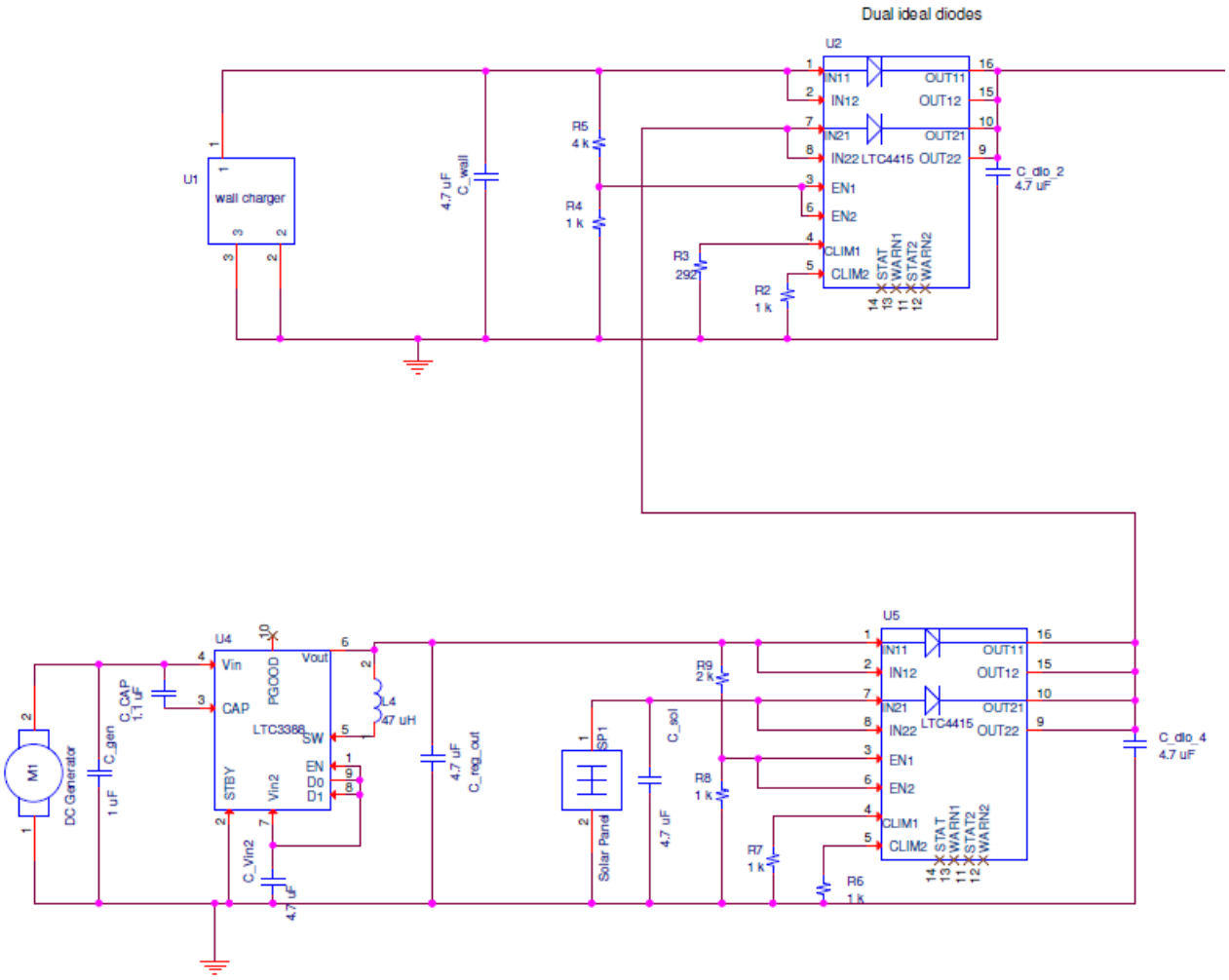


Figure 33. Priority switching stage

The wall charger gets the top priority and this is accomplished through cascading of the ideal diodes chips. The voltage divider seen at the enable inputs on the two ideal diode chips is used to determine which diode gets turned on.

Once we had the priority switching working properly, we moved on to test the fast battery charger. We connected the output of the ideal diodes chip to the input of the fast charger chip, the LTC4060.

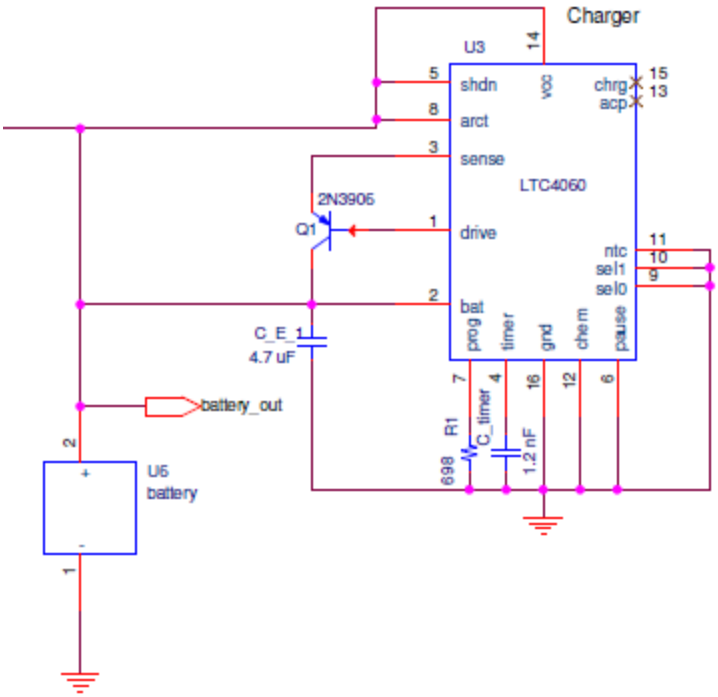


Figure 34. Charging stage

We saw that we were able to quickly charge up the battery up to 4.2V with this charger. Current from Vcc goes through internal circuitry and is outputted at the Sense pin. This supplies our external PNP transistor current at the emitter. The collector of the PNP transistor provides current to the battery.

After this, we tested the converter stage. We used the LTC3127 to efficiently drop down the voltage from the battery to create a 3.3V rail to be used by our modules.

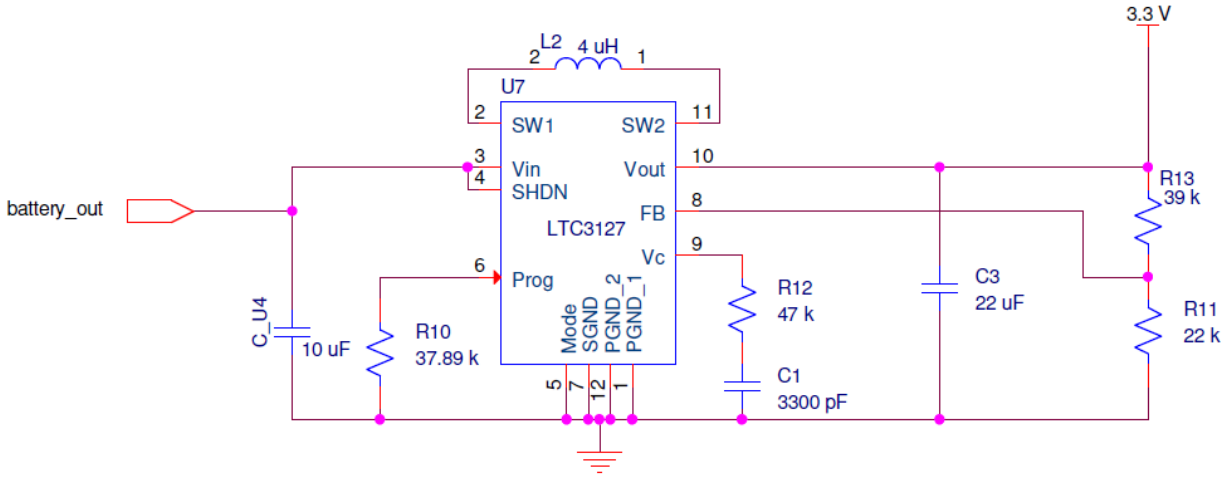


Figure 35. Converter stage

The output voltage is set by the equation:

$$Vout = 1.195 * \left(1 + \frac{R1}{R2}\right) V$$

The resistive divider is connected to the FB pin, and the output voltage can be adjusted from 1.8V to 5.25V.

Once we had this working properly to create the 3.3V rail with the priority switching and charging of the batter, we powered up the module stage with this rail. We made sure all our connections were the same as on the schematic, and we tested to make sure that the Wi-Fi chip was able to be powered, that the fingerprint reader was able to be turned on, and that the h-bridge was able to run the motor such that it could turn the deadbolt.

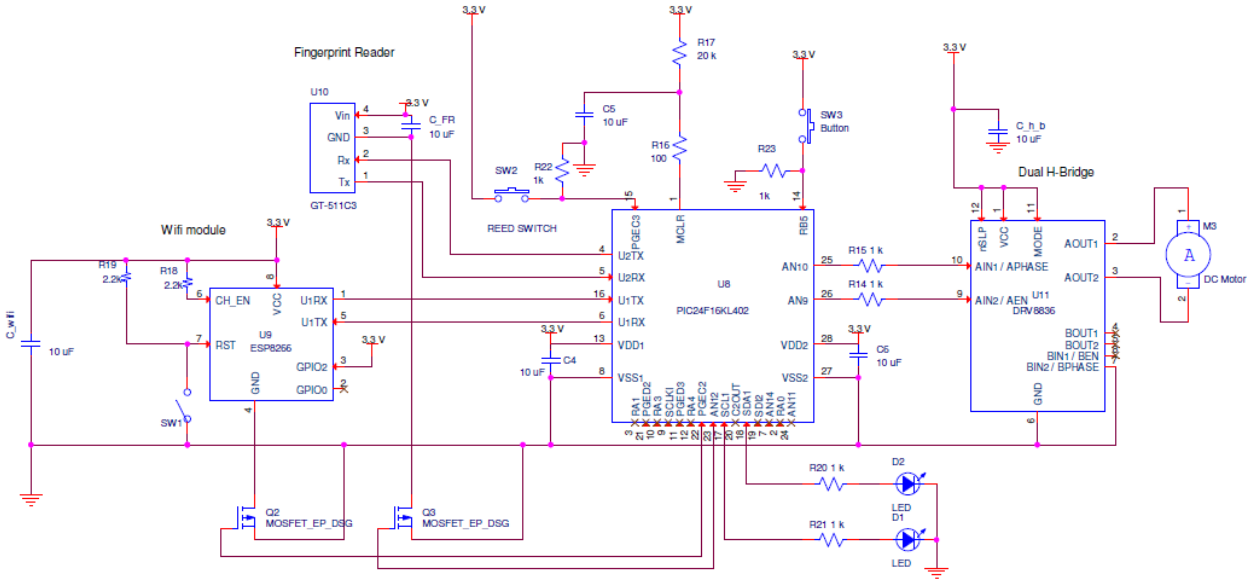


Figure 36. Module stage

We created our circuit schematic on OrCAD Capture before we did our testing on breadboard and perfboard. This allowed us to make changes along the way as we tested parts. And it also made the transition from OrCAD to Allegro very smooth.

Prototyping

Once we had our rechargeable battery circuit working properly on breadboard and perfboard we decided to prototype our circuit on a milled out board. To do this we had to import the netlist from OrCAD Capture once all the correct footprints were assigned to each part. This allowed us to route our traces in Allegro. We used Allegro to create the necessary gerber files to export into CircuitCam. In CircuitCam we insulated the toolpath and created the necessary layers so that the board router could regognize them. We used the LPKF board router that we have available in Jack Baskin. With a combination of 6 mil, 10 mil, and 16 mil end mills we were able to route our traces. We used 0.5 oz copper and calculated that we needed 12 mil trace width for signal traces and 25 mil trace width for power traces.

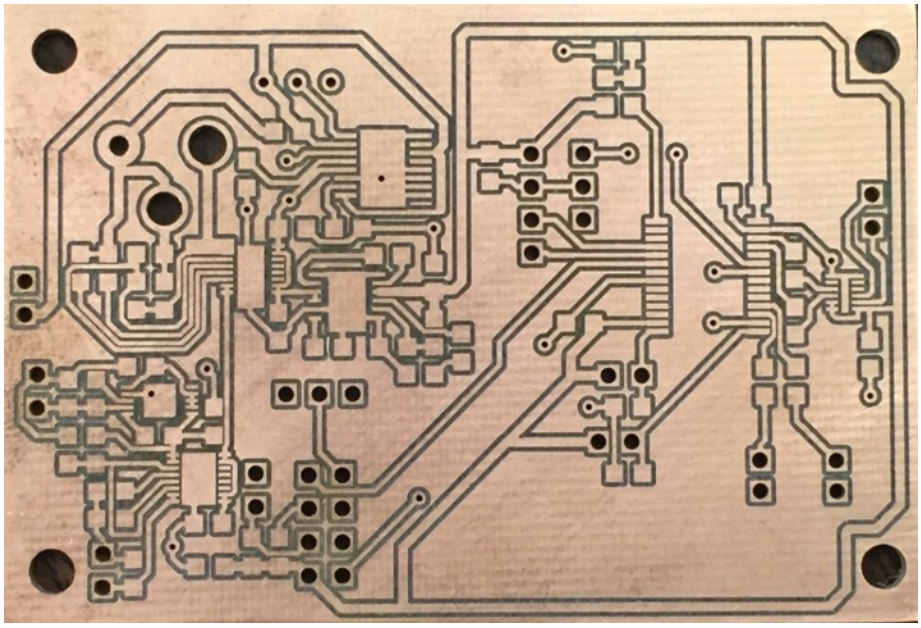


Figure 37. Milled copper board

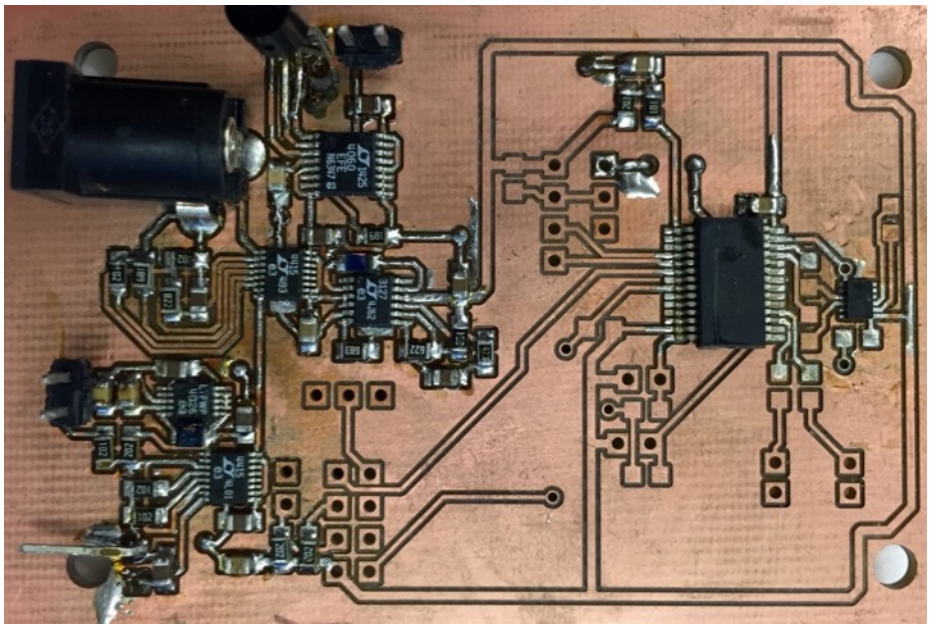


Figure 38. Milled copper board with components

We did three revisions on this board to make sure that we had all the drill holes the correct size and to make sure that footprints were correct for the ICs.

PCB Design

Since we are designing for the system to be mass-produced in the future for consumer use, we require that the system be put on a printed circuit board (PCB). The layout of the entire circuit was done in OrCAD Capture and the traces were routed in Allegro. We also created the gerber files in Allegro and we verified them using Gerbtool. This circuit includes the PIC24, H-bridge, fingerprint reader, Wi-Fi module, the two converters, the fast charging IC, and dual ideal diodes chip. There were already footprints in the preexisting libraries for the dual ideal diodes chip, fast charger, and all capacitors and resistors. For the rest of the parts we had to create our own footprints by using the program Package Designer. The following is an image of the footprint that we created for the PIC24.

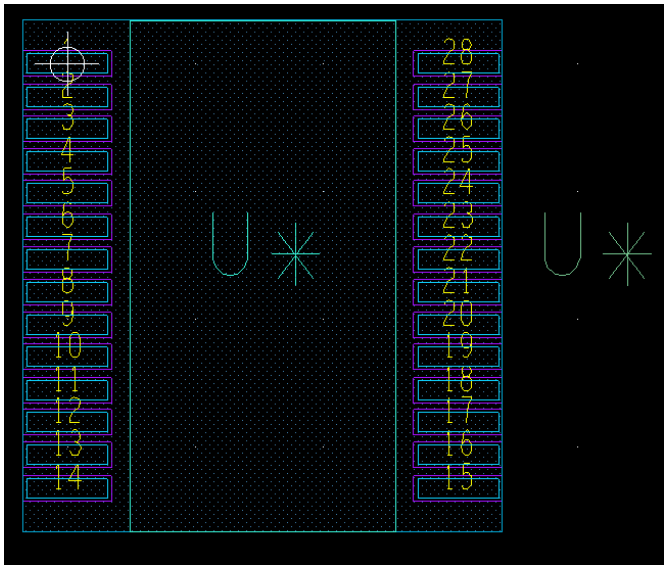


Figure 39. Footprint

This program makes it easy to create a package footprint by using the package symbol wizard. With it, we can enter all the package information that is given on a datasheet, such package type, number of pins, lead pitch, terminal row spacing, package width, and package length. This program also lets us choose what padstacks we want to use for our part. The measurements given in the datasheets were double checked by manually checking with calipers.

We ordered our PCBs from Bay Area Circuits. With their student special we were able to tile 16 of our boards onto a larger 50 square inch area, have 2 layers, include soldermask and

silkscreen layers, and only for $30. Each board was 2.75 in x 2 in, and the layout is shown below.

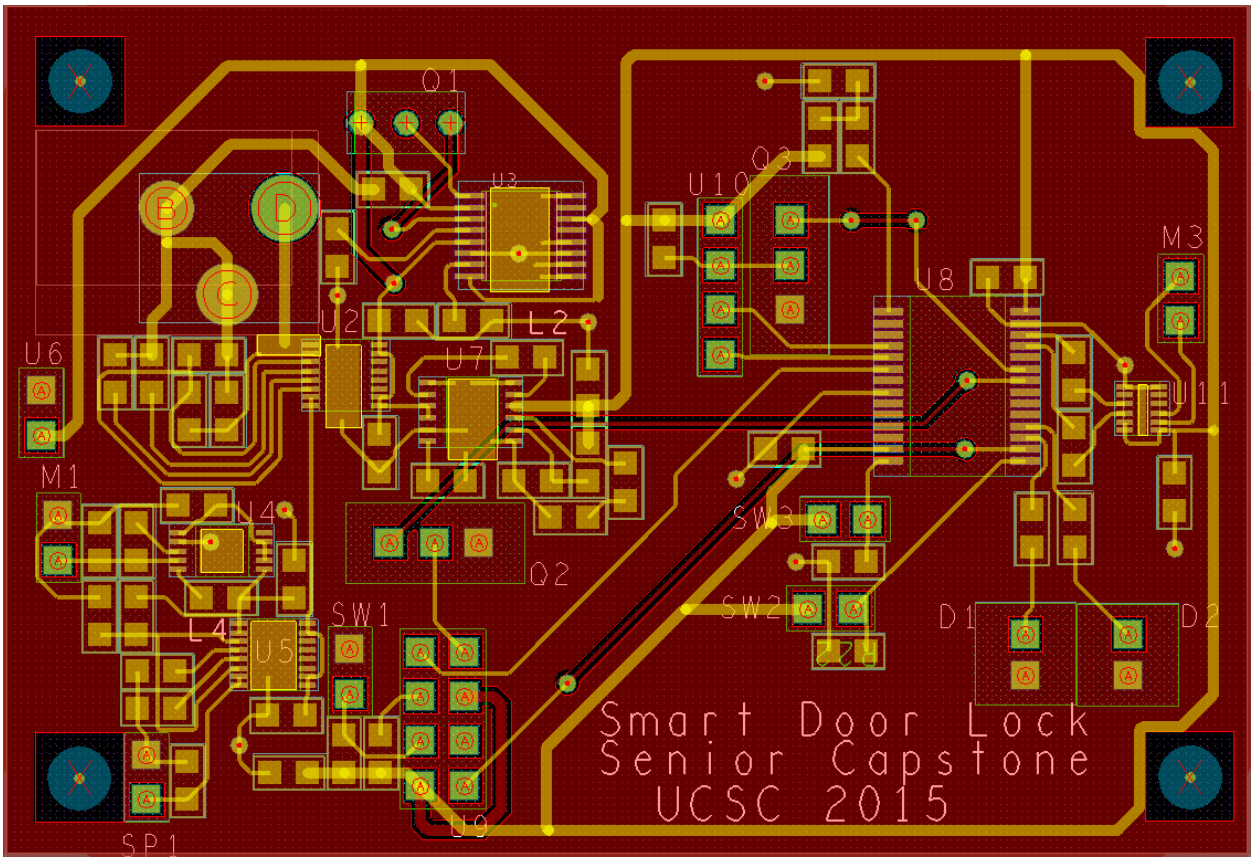


Figure 40. Final PCB Layout

Part Reference	PCB Footprint
C10	SMC0805
C12	SMC0805
C_gen_bypass	SMC0805
C_U4_bypass	SMC0805
C_U5_bypass	SMC0805
M1	SMC0805
Q1	TO92
R7	SMC0805
R8	SMC0805
R9	SMC0805
R10	SMC0805
R11	SMC0805
R12	SMC0805
R13	SMC0805
R14	SMC0805
R17	SMC0805
SP1	SMC0805
T1	dip4_3
U4	ltc4415
U5	ltc4415
U6	SMC0805
U18	tssop16
U19	SMC0805

Part Reference	PCB Footprint
C9	SMC0805
C13	SMC0805
C15	SMC0805
C16	SMC0805
C19	SMC0805
L3	SMC0805
L4	SMC0805
R3	SMC0805
R21	SMC0805
R22	SMC0805
R23	SMC0805
U2	ltc3127
U13	ltc3388

Part Reference	PCB Footprint
C17	SMC0805
C18	SMC0805
C20	SMC0805
C21	SMC0805
C22	SMC0805
C23	SMC0805
D2	diode
M3	SMC0805
R24	SMC0805
R25	SMC0805
R26	SMC0805
R27	SMC0805
R28	SMC0805
SW2	SMC0805
SW3	SMC0805
U14	wifi
U15	dip4_3
U16	PIC24ssop
U17	DRV8836

Figure 41. Component footprints

The image above shows the name of all the footprints that we used next to their corresponding part. We used the 0805 footprint for all the capacitors and resistors but we still have to ensure that all the values that we use are available in this package.

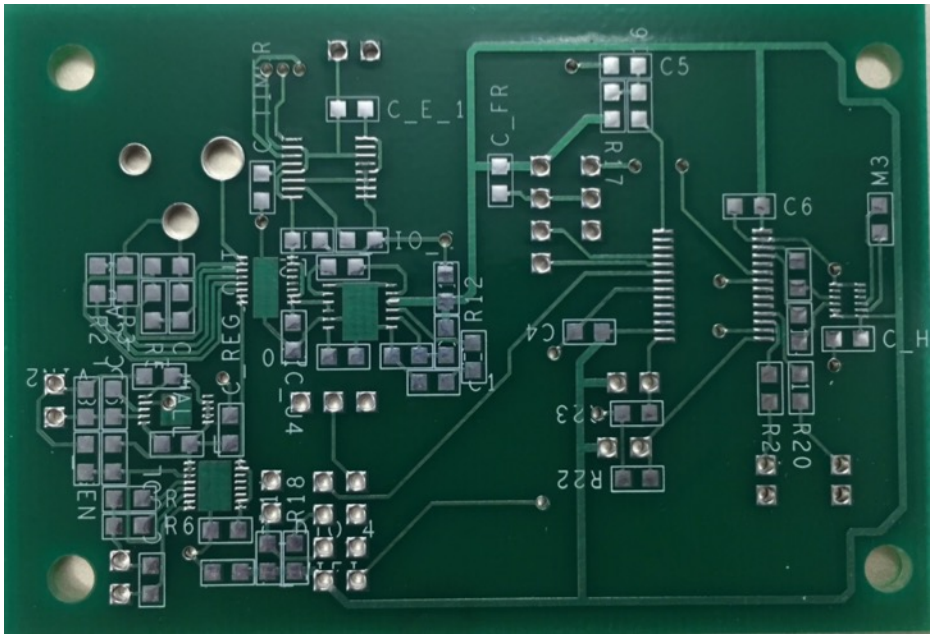


Figure 42. PCB

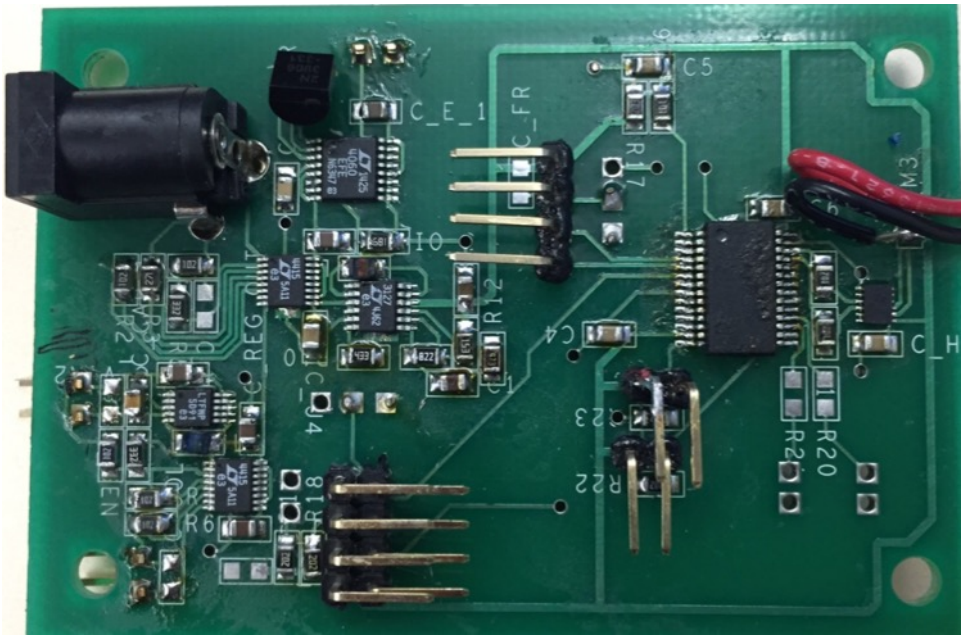


Figure 43. PCB with components soldered

Generator Design

We used a gear ratio of 5:1 for the generator. On average, doors open are opened at about 6 RPM. With the 5:1 gear ratio, the driving gear attached to the door would drive the gear attached to the generator at 30 RPM. From our experiments we calculated that if the generator turns at 30 rpm then on average it should be able to produce about 4 mWh of energy per day. This is with an assumed average of 11 door openings per day, and a door swing that lasts 2 seconds, and swings over a 90º range. To get the maximum power out of the generator we will have to attach an impedance matching network. The following graph shows that as the attached load impedance decreases the power output from the generator tends to spike. This peaks when the load impedance matches the internal resistance of the generator.

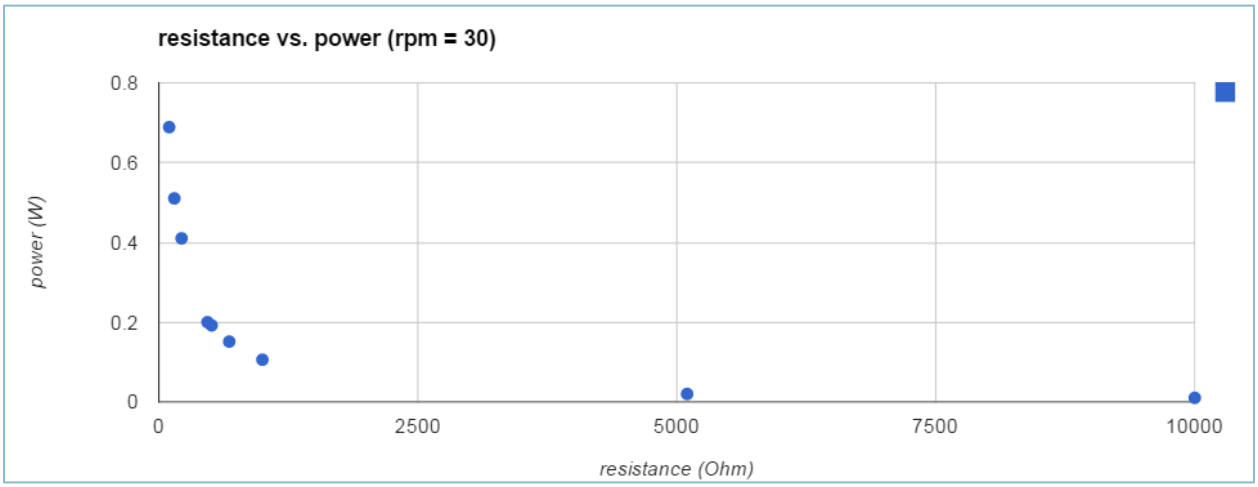


Figure 44. Load impedance vs. output power for the generator

In the graph above we can see that the output power is peaking as the load resistance drops. Experimentally I was not able to use less than 100 Ω but theoretically what we should have seen was that the output power peaks at the matching impedance, and then it drops back down to zero.

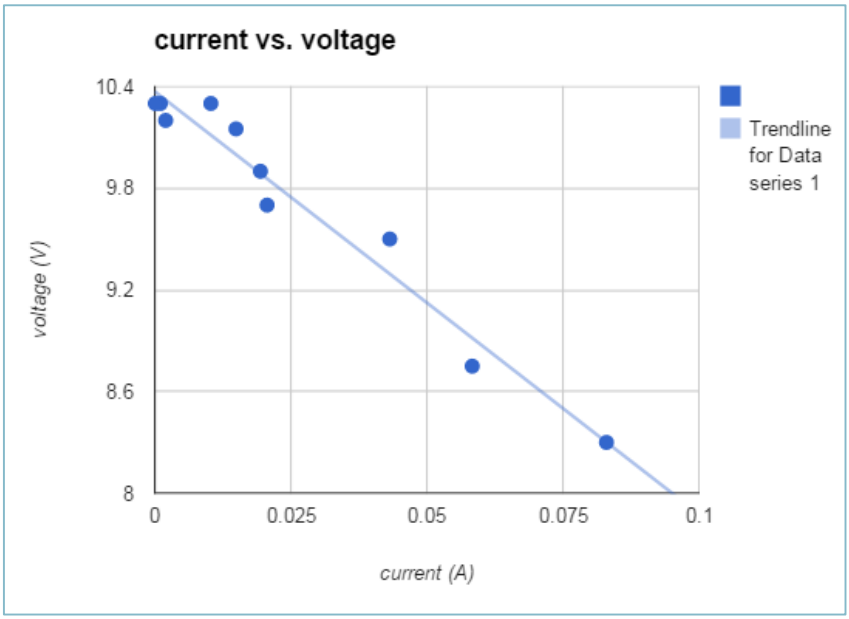


Figure 45. I-V plot for the generator output

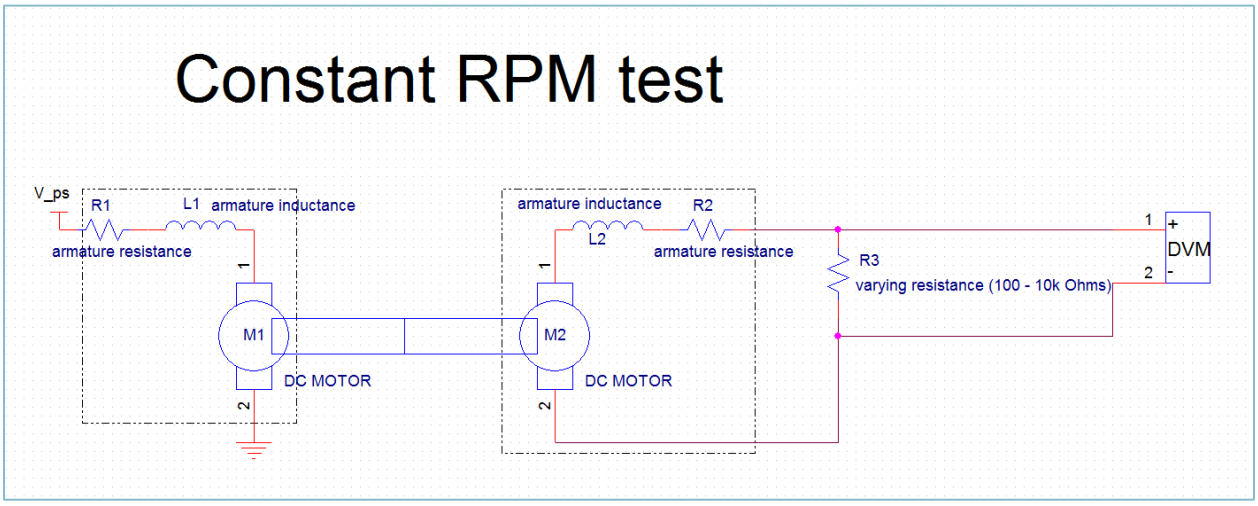


Figure 46. Circuit used for the I-V characteristics of the DC generator

With the circuit shown above we were able to take measurements of current and voltage across the load on the output of the generator. By varying the resistance and keeping the RPM constant, we were able to take measurements for current and voltage, through and across the load. This allowed us to obtain the I-V curve shown in Figure 28.

III. Embedded Software

Overview

The embedded programming side of the Smart Door Lock project was meant to control the locking and unlocking of the door, the functionality of the fingerprint reader, and the communication between the fingerprint reader and the user through an interface and a web server. The fingerprint reader that will be used is the GT511C3 purchased from Spark Fun. It takes 3.3 Volts to power and it works by sending UART commands. The PIC24F16KL402 microcontroller will be used to send and receive information packets from the fingerprint reader. In order to communicate with the webserver and be able to store information on the database, we integrated the esp8266 Wi-Fi chip so the microcontroller can send and receive information from the user or the web server.

Methodology

The first thing we did was to solidify what our requirements were and why we need the parts we picked (GT511C3 fingerprint reader, ESp8266 Wi-Fi chip, PIC24 microcontroller, and SPST NC reed switch). For the PIC24, we needed to have two UARTs that can communicate to both the ESP Wi-Fi chip and the fingerprint reader. Since project will also need to be low powered so we needed to see what the lowest power this microcontroller can run is. The fingerprint reader could store up to 200 fingerprints and had enough resources online to make this reader simple to implement. The ESP8266 Wi-Fi chip was cheap and also a lot of resources online to make this project more affordable and easier to implement.

After we figured out what the requirements were, we needed to know how everything was going to work together. First we made a state machine for the PIC24 that would dictate how everything is run and what the PIC is doing at each moment. Since there was a server side state machine, that was created and combined with the PIC state machine to make an overall flow chart of the entire system. The flow chart demonstrated the use of every part and how it adds to the overall functionality of the smart door lock.

Once the flow chart was done, it became the guide to remind us how everything worked. The next step was to do some more research on the PIC24 microcontroller to justify that it really is

the microcontroller for us. We needed to make sure the PIC24 can use 9600 baud rate at its lowest operating voltage.

After selecting and justifying the microcontroller, the UART needed to be set up. We start the code with a simply Hello World. Since there is no console to printf Hello world, a simple flash of the LED will be this PIC24s version of Hello World. To help with debugging, we set up more LED lights to flash and verify numbers, as shown below. Within the datasheet, there is an example code there. The TX interrupts every time it sends a byte and the RX polls for whatever it receives. Once it was confirmed that a byte could be sent and received, as verified by the oscilloscope, we were certain the UART indeed works.

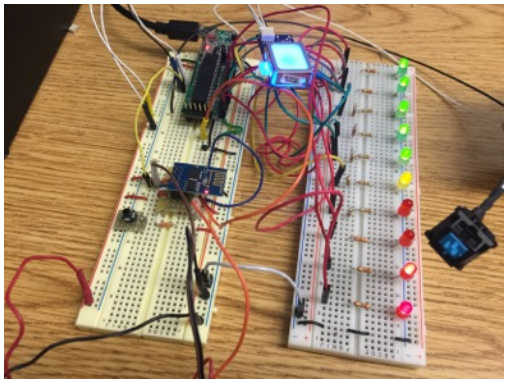


Figure 47. Debugging circuit

The next step is to communicate with the ESP8266 Wi-Fi chip from the PIC24. First we had to make sure the Wi-Fi chip works. We start with a very basic AT send command from the PIC24 and see if we get an “OK” response. If that didn’t work, then we shift our attention to the Wi-Fi chip to make sure it works. We start by studying the pinouts to know exactly what each pin does.

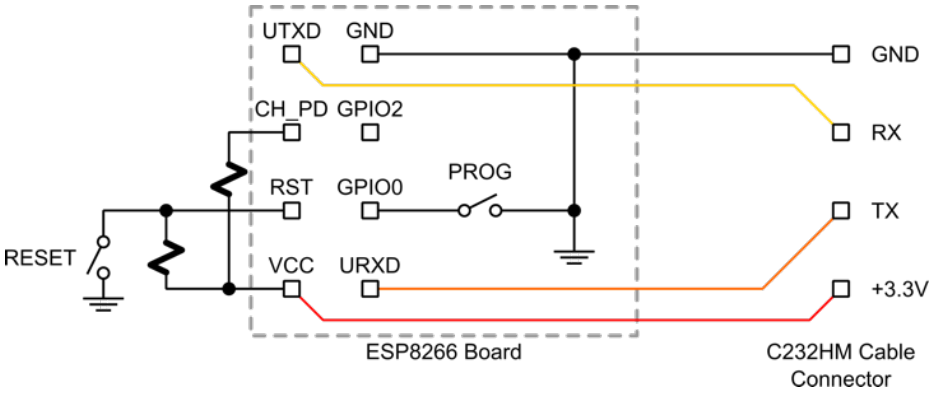


Figure 48. Pinout of the ESP8266 board

In order for the RST pin to work, we need a pull up resistor paralleled with a switch to ground to run. The CH_pd will always need to be high to work so a pull up resistor of 2.2 kΩ will work. GPIO0 needs to be connected to GND if the framework is being updated but has to be floating or high for normal use. This board is powered externally by 3.3 Volts.

Since this Wi-Fi chip is brand new, it needs to be flashed with the latest firmware. We download a flasher software and upload the latest firmware.

Figure 49. Flasher for the ESP8266

The flasher loads binary files to the appropriate addresses on the ESP8266 to configure its firmware and settings. All we needed to do was make sure GPIO0 is grounded and click "Flash(F)".

Once the ESP8266 is flashed, we needed to test it to make sure it's alive. We download and run ESPlorer which is an interface specifically designed to run the ESP8266 Wi-Fi chip.

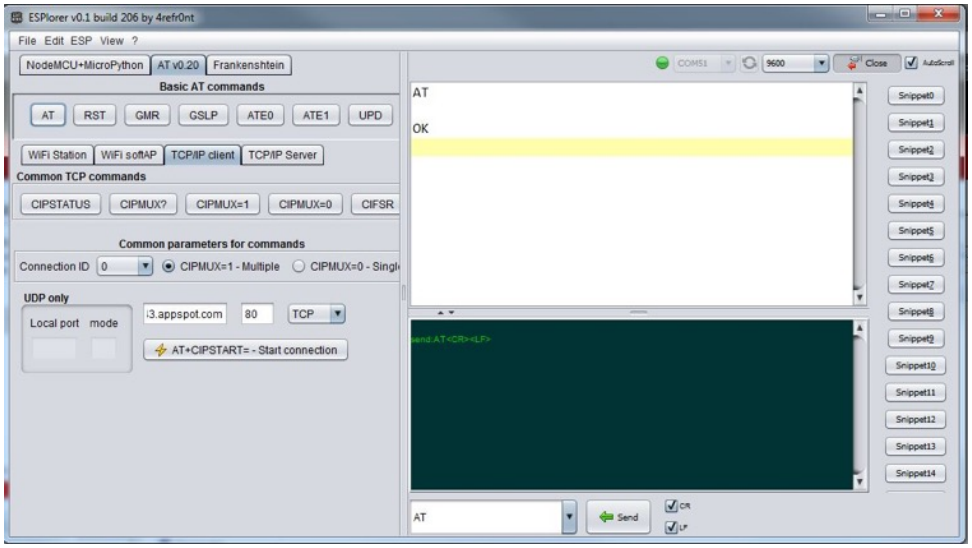


Figure 50. Interface of the ESPlorer

The ESPlorer had basic AT commands that can be sent to the Wi-Fi chip and the chip will respond with “OK”. This can also be used to printf from the PIC24.

We sent “AT+GMR” to check the firmware, making sure to always include return carriage and new line after every command. When we received garbled text, we checked the baud rate (default of 9600) or made sure to use a power supply to power the chip. We found that using the 3.3V from the PIC24 doesn’t work.

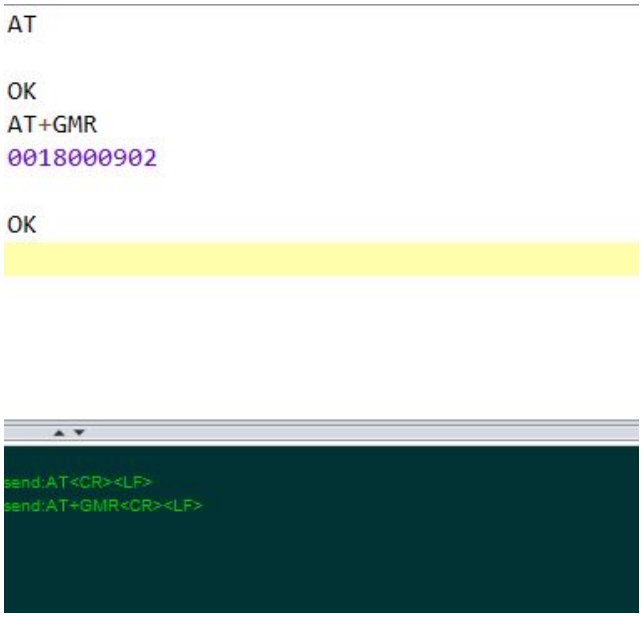


Figure 51. When sending AT+GMR, it returns the firmware number (0018000902 in this case) and OK.

Once we knew the chip was alive and working, we could test the server mode and client mode functionality. Server mode is the default mode since it's a local server that the ESP8266 broadcast. It can be found by checking the Wi-Fi networks on your smart phone or laptop. The default server mode network has the name "ESP" on it. It varies from different firmware. The name can be changed on ESPlorer with the AT+CWSAP command. For example, I used "AT+CWSAP="SDL_ESP","password",1,2" to change the network name to be "SDL_ESP" with the password as "password."

We proceed to test server mode by connecting a smart phone or laptop to the ESP local network. We made sure to use "AT+CIPMUX=1" to allow multiple connections and "AT+CIPSERVER+1,80" to start the server. Once a phone or laptop is connected, we go to a browser and enter the local IP address of the Wi-Fi chip. This can found by entering the command, "AT+CIFSR." Once we are connected we obtained a message similar to this one:

+IPD,0,372:GET / HTTP/1.1
Host: 192.168.4.1
DNT: 1
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 8_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12D508 Safari/600.1.4
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cache-Control: max-age=0

OK

This indicates that we successfully created a link between the phone/laptop and the Wi-Fi chip in server mode.

The next step is to test client mode. In client mode, the Wi-Fi chip is connected to the internet and is able to read information off the web. To connect to the internet, use the command “AT+CWJAP=” followed by the SSID and network password in quotes. It should take at least 10 seconds to connect and after that, disconnect of server mode by sending “AT+CIPSERVER=0.” After that it will prompt you to restart. We can do this by sending the command “AT+RST” or by hardware restarting. Use AT+CIPMUX=1 once more and use the AT+CIPSTART command to connect to a website. Once the chip is connected the internet, it needs to link with the Smart Door Lock website with the command, AT+CIPSTART=0,"TCP","elegant-moment-843.appspot.com",80.A successful connection is made with the response “Linked”. After any reset, the UART of the microcontroller will need a restart otherwise, the microcontroller can’t read anything from the wifi chip.

Once Client mode works when we receive a response, it’s time to send commands from the PIC24. If we check commands being send by the ESPlorer on the oscilloscope, we are able to read from left to right, “AT carriage return newline” in ASCII, so we need to send those exact bits in that exact order over the PIC24. We make sure the baud rate on the PIC24 is set to 9600. From reading the datasheet, there’s a formula for calculating the baud rate based on the frequency clock of the PIC24. The value of the baud rate needs to be stored in the U1BRG register. The value is calculated by dividing the clock frequency by the desired baud rate then divided by 16 and subtracted by 1. The baud rate or clock frequency of the PIC is correct when we send “AT” and receive an “OK”.

We set up a web page that can be sent from the PIC24 through the ESP8266, to the phone. We use a simple Hello World to get started. Once we have that working send an interface that allows the user to input their SSID and network password and a submit button. That will send the information back to the PIC. Once the information has been received, parse the information and connect the Wi-Fi Chip to Client mode to verify you can receive the correct information. Basically we do everything from the ESPlorer side but now in the PIC24 side. We could use the ESPlorer as a debugger.

The next thing is to send commands to the fingerprint reader to make sure that is alive. The GT511C3 is powered by 3.3 volts so a power supply will suffice. Commands send to the

fingerprint reader has to be in a specific format. There needs to be two command start codes, followed by the device ID, the input parameter, the command code, and then the check sum of all the offsets. A good way to test if the code works is to send the CmosLed command to turn the fingerprint reader off or on. We need to set the RX to receive the proper response packets. It's a lot more important from the fingerprint reader than the Wi-Fi chip because you need to get a response every time you send a command. Once the RX terminal reads the commands start codes, it can save the values that it gets from the RX to an array. That array can be then parsed to figure out error parameters and whether the response was successful or an error.

The first real test of receiving response packets is using the IsPressFinger command. When a finger is pressed the parameter is zero else it is nonzero. Once that works, receiving response packets is fully understood.

To check if isPressFinger is working, the command is sent repeatedly with a 200 ms delay in-between sends to not overwhelm the fingerprint reader and then checking the response parameter from the fingerprint reader to see if it's zero or not. Once a finger can be detected, the next step is to capture a fingerprint.

Capturing a fingerprint is used for the identification of the fingerprint as well as enrolling it to the fingerprint reader. The captureFinger command on the fingerprint reader has two settings; high and low. The high setting is for higher resolution, slower capture speed, and is used for enrolling. The low setting is for lower resolution but faster capture speed which is used for identifying. To test the captureFinger function, the identify function can also be tested.

Identify works by waiting for a finger to be pressed on the fingerprint reader and then once a finger is detected, the fingerprint is captured and temporary saved to a template. When the identify function is called, it compares the template that was saved to a template that is stored in the device to see if there is a match. If there is a match, it will return an acknowledgment along with the fingerprint id in the parameter else it will return an error. To test the identify function, a fingerprint must already be enrolled. Since the enrolling has not been tested and is a lot more complicated than testing identify, a demo-software can be found online to enroll a fingerprint.

Once identify works, and the microcontroller can distinguish between different fingerprints and non-enrolled fingerprints, it's time to move on to enrolling. Enrolling is a lot more complicated since it requires being able to wait for a finger to be placed on the device as well as waiting for a fingerprint to be removed from the device. How enrolling is done is by capturing three high resolution fingerprints and combining them into one template that will be saved into the fingerprint reader. To capture three fingerprints, the user will have to place and then remove their fingerprint when prompted to. The instructions on when to place or remove a finger will be sent from the microcontroller to the local user interface.

The microcontroller stores the user information of the fingerprints such as their access time. A struct array is built to represent each user. The struct will contain a parameter of whether or not they are an admin, a enrolled bit that checks if this user is enrolled, and a char array that contains the access time schedule for the user. Since the size of the struct is quite large and the microcontroller can only carry 16K of flash, we allowed only 10 users to be able to enroll on this fingerprint reader. The space is mainly taken up by the access time. When a fingerprint is enrolled, the user struct array, whose index is the value of the ID that was just enrolled, will be updated with a slot being filled and an access time schedule of all day and all week. Only the admin can change the schedule of the users.

To update the information of the user's access schedule, the wifi chip needs to connect to the Smart Door Lock website to get information on which user's access time needs to be changed. Since the microcontroller will need to parse a lot of information from the wifi chip, the RX ISR of the wifi chip will need to be structured in a way that can be build upon to taken in additional information if needed. The structure that was set up was a state machine using switch case statement. There will be a state for reading whether the command being send is a GET or a POST command. If the command is a POST, check whether the user wants to enroll, delete, or set the wifi settings. For the updating of the user information from the microcontroller, the product id, a list of who was deleted and who was added will be sent to the web server and the web server will respond with the current time and day of the week, the users that need to be deleted, and the schedule changes of any users.

With the addition of access times, the unlocking of the door is not simply just identifying a fingerprint. To unlock the door, the door needs to be closed and locked. Once a button is

pressed, the door will run the identify function that will check for a identifiable finger. Once a fingerprint has been verified, the microcontroller will check the access time of the user of that fingerprint ID. The current time is saved as minutes (a value from 0, meaning Saturday 12:00am, to 10080, which is Sunday at 11:59pm a whole week in minutes).
The access time array is 48 8-bit chars that each represents a day of the week. The reason there is 48 slots is because a day is divided into 30 minute blocks. This “Access Table” will be stored into the User’s access time

Ex. 0b10011100

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	1	1	1	0	0
Don’ care	Saturday	Friday	Thursday	Wednesday	Tuesday	Monday	Sunday

In order to check if the user has access at the current time, the minutes determine, what day it is and what 30 minute time it is throughout the day. The equation used to find the current day and current thirty minute block of the day is shown below.

```
currTime; // the minutes value from (0-10080) incremented from the timer interrupts.
                    // There is 10080 minutes in a week
day = (int) ((currTime * 7)/10080);   //get a value from (0-6)
thirtyMinBlock = (int) ( ( [currTime – (1440 * day) ]*48)/1440) //get a value from (0-47)
```

That time frame is a value from 0-47 which is the index of the access time array. The accessTime[thirty_minute_time_frame] will be masked with a bit that depends on what day it is. If that bit is 1 then they have access at that time.

The microcontroller will update in two ways; first is by sending the setWifi command from the user interface and second is by the periodic 30 minute updates. For demonstrating purposes, it much easier to set the SSID and network password to update.

For power saving features, the microcontroller will need to turn off and on the fingerprint reader and the wifi chip using MOSFETs. The fingerprint reader and wifi chip will be on when the door is open and is enrolling. Only the fingerprint reader will be on when a button is pressed for identifying.

Lastly, the top level state machine needs to be coded. The system will have three main states: Open, Closed, and Locked. The microcontroller turns on the wifi chip and can do enrolling, deleting and setting wifi information all in the open state. The closed state is where reed switch will tell the microcontroller whether or not the door is open else after 3-5 seconds; the door will lock and go into Locked state. In Locked State, the 30 minute interval updates will happen and once the button is pressed, the fingerprint reader will turn on to search for an identifiable fingerprint.

Results

The main top level state machine is shown below.

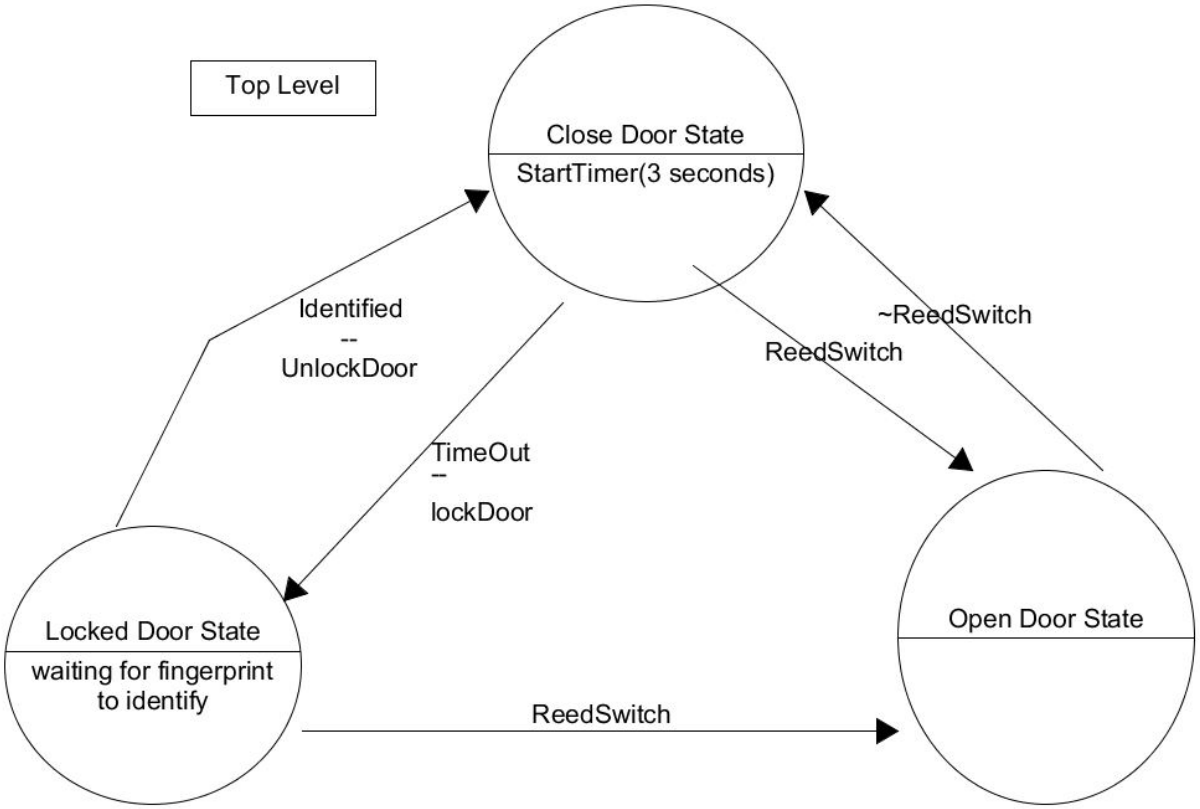


Figure 52. State machine showing three high level states: open, closed, and locked.

Considering an ideal case, when a door is locked, it is closed. Once a fingerprint gets authenticated, the door unlocks and the user either opens it and then closes it or keeps it

closed. In the first case, the door goes into the open case where it waits to be in the closed state. Once it's in the closed state, a timer will start for 3 seconds. When the timer times out, the door will lock. In the second case, when the user keeps the door closed after unlocking it, the door will still have a timer set to 3 seconds. If the user doesn't open the door in 3 seconds, the door will unlock. If for some reason the door is in a locked state but the door is opened, the door will automatically unlock.

The PIC24 can send commands correctly. Before the TX would send so fast, the TX buffer would get full so the RX on the devices would not be able to read in time. To solve this, I added a delay of 1 ms after every character send so the TX buffer would never get full. This will add a delay to the overall program as sending websites that take thousands of characters will take a couple of seconds to send everything.
For the ESP8266, server mode and client mode was a success. The PIC24 could send an HTML page through the ESP to the smart phone and display window where I could input an SSID and Network password.

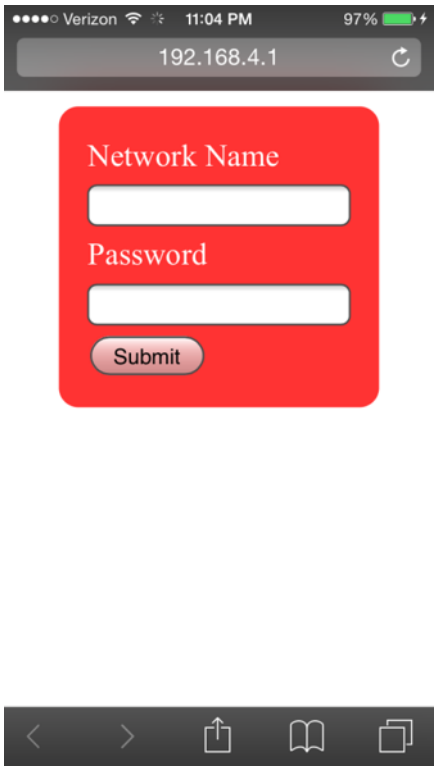


Figure 53. Output of the PIC24 to the iPhone to input SSID and network password

Once we click submit, the PIC24 will receive the information and will be used to connect to the internet. We used an iPhone hotspot to connect to the Internet since Cruznet doesn't seem to work very well. Once we could connect to the Internet, we sent a TCP request to the web server with the following results.

```
AT+CIPSTART=0,"TCP","elegant-moment-843.appspot.com",80

OK
Linked
AT+CIPSEND=0,62
> GET /admin HTTP/1.1HOST: elegant-moment-843.appspot.com
busy s...

SEND OK

+IPD,0,227:HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
Date: Tue, 24 Mar 2015 04:14:23 GMT
Server: Google Frontend
Content-Length: 26
Alternate-Protocol: 80:quic,p=0.5

hello ESP, from Web Server
```

Figure 54. Results of our attempt to connect to a website. The website displayed "hello ESP, from Web Server." We sent the AT+CIPSTART command to port 80 on the website to receive the text shown.

For the fingerprint reader, sending commands to set the LED on is successful. It's very easy to flip the light off and on. Receiving response packets was a real challenge. I can verify that IsPressFinger works since I programmed LEDS to come on if a finger is sensed and turn off otherwise.

At this point, the whole flow works. The microcontroller is intended to stay on for as long as possible so when it starts, its initial commands are to delete all the fingerprints on the fingerprint reader to have a clean slate. Two LEDs will light up to show the microcontroller has begun starting up. Having the door open, the user will connect to the wifi chip's IP address and pull up the user interface that is sent from the microcontroller. By pressing the "+" button, this will start the enrolling process. Following the instructions to place finger and remove finger, after three

presses the finger will be enrolled. You can verify this by closing the door which will start a 5 second timer which will lock the door after it times out. Press the button to start the identify function. When you place the correct finger, the door will unlock and you have 5 seconds to open the door before the door locks itself again. To update the system, have the door open, and press the Wifi button on the user interface to input the SSID and network password. Once you press update, the microcontroller will attempt to connect to the internet and website with the given information. Once that succeeds, the microcontroller will have the current time, 30 minute updates have been enabled and the web server will have a user to make a profile for. After the user profile has been created and the schedule has been changed, the update will change the user's access time on the microcontroller to match that on the web server.

Summary

The state machine hasn't been programmed yet but it will be for the Spring Quarter. The Wi-Fi chip works pretty well and there isn't much issue with it. We spent a lot of time learning how it works by reading and watching a lot of tutorials online. There are a lot of resources for it but we realized our issue was the ESP was not getting enough power. We were powering it from the PIC24, which was a really bad idea. Signs that warned us it was not receiving enough power was the error output, which meant there was not enough current running through the chip, and the red light that signals the chip is on was not on. Sometimes the red light was on but we couldn't tell if the chip was receiving. Once we attached a power supply, all our problems were solved. The ESPlorer is a great tool for getting the Wi-Fi chip to work and a good makeshift debugger for the PIC24. We couldn't get printf Hello World before but now we could with the ESplorer.

The UART was big problem for us since we weren't too confident with UARTs. We figured out our main errors, which was the fact that our TX and RX wasn't working 100%. The TX buffer would get full often so we added a delay to fix that. The RX would receive from the ESP correctly either but that was a baud rate problem. Since the formula was taken from the datasheet, the code wasn't entirely correct for my situation. The datasheet had the frequency clock set at 40 MHz and after doing research on the PIC we found that the PIC can't go any faster than 32 MHz so we knew it was a problem with that equation. We found the right clock

at 8 MHz which solved the UART problem and we were finally able to send and receive correctly from the Wi-Fi chip.

The Fingerprint reader was easier to implement since we got the UART to work. The real challenge was parsing all the information we received from the RX into arrays that can be used for the code. For instance, the Wi-Fi chip needed to receive the SSID and network password from the user. We had to look for the right patterns of chars before we store the characters into an array. It was a little tricky to do but once we got the pattern down, we were able to continue my program. The fingerprint readers IsPressfinger works but it seems to have a memory problem. We made the code where it would call IsPressFinger every other time we press a button. When the code starts, supposedly, a finger is pressed. After that, it seems to work normally. But once a finger is pressed and it detects that is true, after the next iteration, it still says it's true regardless if there is a finger there or not. But other than that it works.

A lot of time was spent cleaning up circuits to make sure parts worked and cleaning up code to make sure we understand how everything works. Next quarter, the fingerprint library will be written, and we will be able to enroll, delete, and identify fingerprints. A state machine will then be fully integrated into the system.

Most of the time was spent cleaning up code and re-doing and testing old functions such as WaitPress (checking to see if a finger is on the fingerprint reader) because as the code gets more complicated, the need for everything to work stacked. When things didn't work, we would have to go back to the root of the problem and dissect to see what's wrong.

The code had to be redone at some point. There were errors in the code and it seem best to have two programmers for this code so a reboot was done. The new code included more state machines for the ISR and the main code. It's still the same functionality but it's much more robust. The new code involved a lot more delimiting and comparing parsed strings. More emphasizes on saving memory was considered. Whenever we had the opportunity, we would use 8-bit unsigned integers. If needed, we would go up to 16-bits.

At the end of it all, it's a miracle this code worked as planned because a lot was factored in to make everything work. Everything needed to be powered correctly. The wifi chip needed to be send the right commands with the right length of the information being sent otherwise there will be a bricked chip or lost information. The fingerprint reader needed to be sent the right commands in the right format and received in the correct format to be read. There needs to be timer interrupts to keep track of time and perform timeouts. When a fingerprint is being enrolled, the wifi chip needs to maintain a connection to the user interface otherwise the instructions won't be sent. The instructions needs to be sent with the correct values of the length otherwise the chip will brick or it won't show up. Sometimes the chip bricks when things are working normally. Once a user has been enroll, the struct has to be updated and that updates a list that will need to be sent to the web server. Once we tell the server who has been enroll, we need to change the access time schedule and that requires precise state machines on the ISR. To check the access time for identifying you need to already have the current time, the current day, and find the thirty minute blocks that the current time is on. Masking a bit that is off set by the day, and Anding that bit with the access time to make sure if the user has access or not. It's really a miracle everything held together.

IV. Web Framework

Server

The purpose of the server is to keep track of the smart door lock's product ID and store the access time of each attempted entry. This product key is not used for authentication but rather the identification. The owner of the lock will have a password protected online account to access the user interface.

UCSC VM

There is not clear documentation or support for using UCSC's VM service. Once the firewall settings have been changed, log in is no longer permissible which prevented progress.

Google Cloud Platform

Beyond matching the requirements of server, database and front end hosting, Google Cloud Platform has a lot of powerful APIs which makes the future implementations of usability possible.

- Version Control & Deployment – There are logs for version control and it allows the user to roll back by typing in a simple command. Deployment is also just a simply click of a button and logging in to the Google account.
- Queried Database – It uses GQL, a dialect of SQL, and includes properties which are Python objects. This allow very easily integration and quick results from being built-in to the platform
- Emails (SMTP) – It is implemented through SMTP using a Google App Engine API which links the account registered with the platform and allows the administrator to easily email any user or groups with the help of querying
- Data Encryption – There exists Python libraries which can be imported and used for security. While this option exist, an online source claim that there is auto-encryption embedded into the Google Cloud Platform. Given that, the consideration of time, and the order of importance, more research is needed.
- Communication Security – SSL is only possible through channels which is a way to divide a single port for listening to requests. However, the scope of the project requires only HTTPS. Certificates are automatically granted to clients who choose the Google Cloud Platform.
- Language – Python is one of the few options supported by Google. This language is easy to learn, portable and scalable.

Web Server Structure

Pages

The web server is organized in many pages. In terms of implementation of the flow diagram for the smart door lock system, the structure of the web pages is irrelevant. What matters is the functionality, features and integration. Since the integration revolves around HTTP requests, the web is structured such that all functionality is triggered by HTTP requests, and should the Wi-Fi module want to access any pages, it can.

Additionally, the web server attempts to handle all possible error from typing in incorrect inputs, using the back button, and typing into the URL to trigger requests not otherwise possible by the point-and-click method of navigation. These considerations address the vulnerability of the web server and of the system.

- HomePage – It allows the user to input registry info, login, and receive email for forgotten passwords. The front end interface also checks for correct inputs before sending the request to the database to prevent faulty attempts.
- Account – It dumps the database information the user has once logged in and provides the interface to all the other functionalities including: adding users, deleting account, setting Wi-Fi connection, resetting password, and logging out.
- Expired – It is used for expired logins as the user attempts to re-authenticate from typing in the URL, back button, or clicking on the link in the forget password email
- Register – This registers an account with the product. Please do not confuse this with "enroll" which uses the admin rights of the account associated with the product to authorize the access of other users who do not own this product.
- Delete Account – It removes all users associated with the product registered with the account that is currently logged in.
- Enroll – As mentioned, this action is restricted to owners of the product and they are the only people who can add users to the system.
- Delete – This deletes a single user from the door lock system and can also only be deleted by the owner. Additionally, the owner is not allowed to delete his/her account and can only do so using "Delete Account"
- Schedule – It saves the parsed schedule string into the database
- Rename – It saves the new first and last names of the user modified
- Admin – This is where the Wi-Fi chip sends a get and post request to the server which is responded with a query string attached to the URL for communication.
- ForgotPassword – This page simply gathers the email.
- EmailPassword – This request sends a link to the email the user specified in "ForgotPassword" along with a generated token which expires in 5 minutes.

- SetPassword – It allows the user to reset password if they have already logged in. Even so, the identity is verified via prompting for the old password.
- ResetPassword – This allows the user to reset password by clicking on the link only. The token is verified for equivalency and expiry.
- SavePassword – This actually loads the new password into the database.
- SetWifi – This saves the new SSID and password information into the database.

Front-End

The main structure of this is simply input checking, data parsing for visualization, event checking, and CSS styling.

- account.js – This is for the main account page which contains all the database information. It is able to handle: reset password, delete account, delete user, add user, rename, and log out. Bootstrap.js modal has been applied to the functionalities to lessen the amount of page transitions. The schedule has to be parsed from a string consisted of bit-maps. Through bit-wise operations, this is convert into jQuery objects which translates to HTML elements on the page. The reverse is done to convert a compact string to be stored in the database. Again, a lot of input checking and post requests are required and done indirectly through the front end.
    - home.js – This is less involved than account.js because it only needs to check user input and issue three possible post requests: forgot password, login and register. Most of the work associated with this page is implemented on the server.
    - web.css - CSS is applied to all elements on each page and classes are used for the selection mechanism visualized on the schedule table.

Web Application

Local

Overview

The door lock system uses the on-board Wi-Fi chip as a server to obtain the web site. The website is simple and contains three possible options: enroll user, delete all users and configure Wi-Fi connection. When the physical door is opened, the authorized personnel can connect to the on-board Wi-Fi chip through her internet-enabled devices and gain access to this interface. When the authorized personnel decides to enroll a new fingerprint to the system by pressing the button beside the solar panel, instructions will show on the web page as loaded from the micro-controller via the Wi-Fi chip.

Features

- Enroll - When the user clicks on the plus sign, it indicates that the user wants to enroll a new fingerprint. This is only possible when the door is opened and that the Wi-Fi is in server mode. The interface will display instructions telling the user to place or remove her finger at appropriate times until the final message showing that the process is complete and that the users should now be updated on the web server. Each new users will be added to the queue and sent to the web server every thirty minutes such that the users cannot forcibly add user outside of the thirty minute block.
- Delete - When the user clicks on the minus sign, it indicates that the user wants to delete all users from the fingerprint reader. The microcontroller simply removes all user from the system along with the fingerprints saved on the fingerprint reader.
- Set Wi-Fi - When the user clicks on "Set Wi-Fi", it indicates that the user wants to edit the network connection. The user can simply enter a new network SSID and its corresponding password. Next time the thirty minute update occurs, the Wi-Fi will attempt to connect to the new network to reach the internet.

User Interface Program

This contains an HTML script with consists of CSS and JavaScript. The CSS simply adds color and resizes the element to fit the screen of the internet enabled-device. The JavaScript code uses XHR to receive responses and display them on the screen in the appropriate spots. The

code is triggered when the webpage has a state change in the browser, in this case, receiving an HTTP response. This is handled internally by the browser itself such that the state is equal to 4 when the data is ready to be read. The same code is triggered recursively as long as the received message does not end with a dollar sign. The dollar sign indicates that the process of enrolling a fingerprint, deleting a user, etc has been interrupted or is over. That way, the web page will cease the continuous requests which is used to step through the process of enrolling, deleting the user and setting the Wi-Fi. The script fires an HTTP request to the server on the Wi-Fi chip asking for the corresponding page. Set Wi-Fi requests the update synchronization which contains extra information including time, schedules and deleted users. The protocol between the embedded system and the internet enabled-device is a query string variables delimited by the ampersand symbol which abides to the standard HTTP.

Website

Overview

The website serves as an interface for the smart door lock as a mean to provide remote access to the system. It has the basic profile registration such as email and password authorization. In order to register, the user must provide the product ID included in the package which would come along with the door lock.

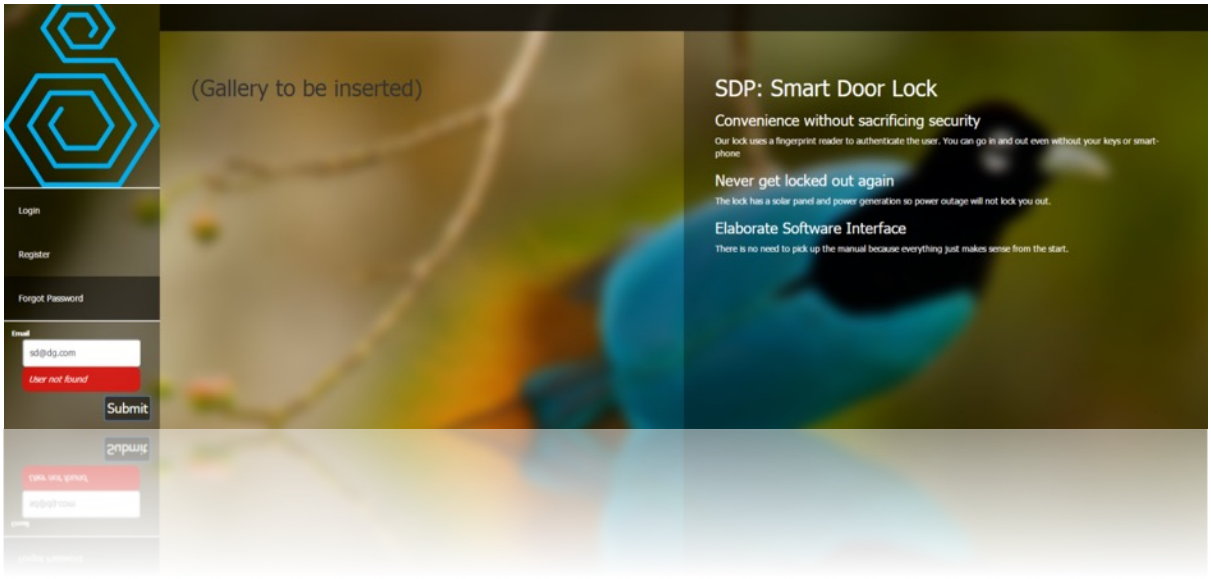
After registration, the users can enter the website and access features like add, delete users and accounts. For usability, we have also included a mechanism for password reset and forget password option. For security measures, there are no direct ways to modify the authentication on the physical system. Any action taken will be synchronized with the physical system to ensure stability and consistency. The users can choose to modify access schedule of each users, delete or rename them if they are admin of those certain users - permitting them to do so.

The user information is visualized using styled and animated HTML elements, located in the two main panels at the right hand side of the page. There is a menu on the left showing options to add users, delete accounts and more. When these options are selected, the main panels will display the information relevant to the options.

Features

Login Page

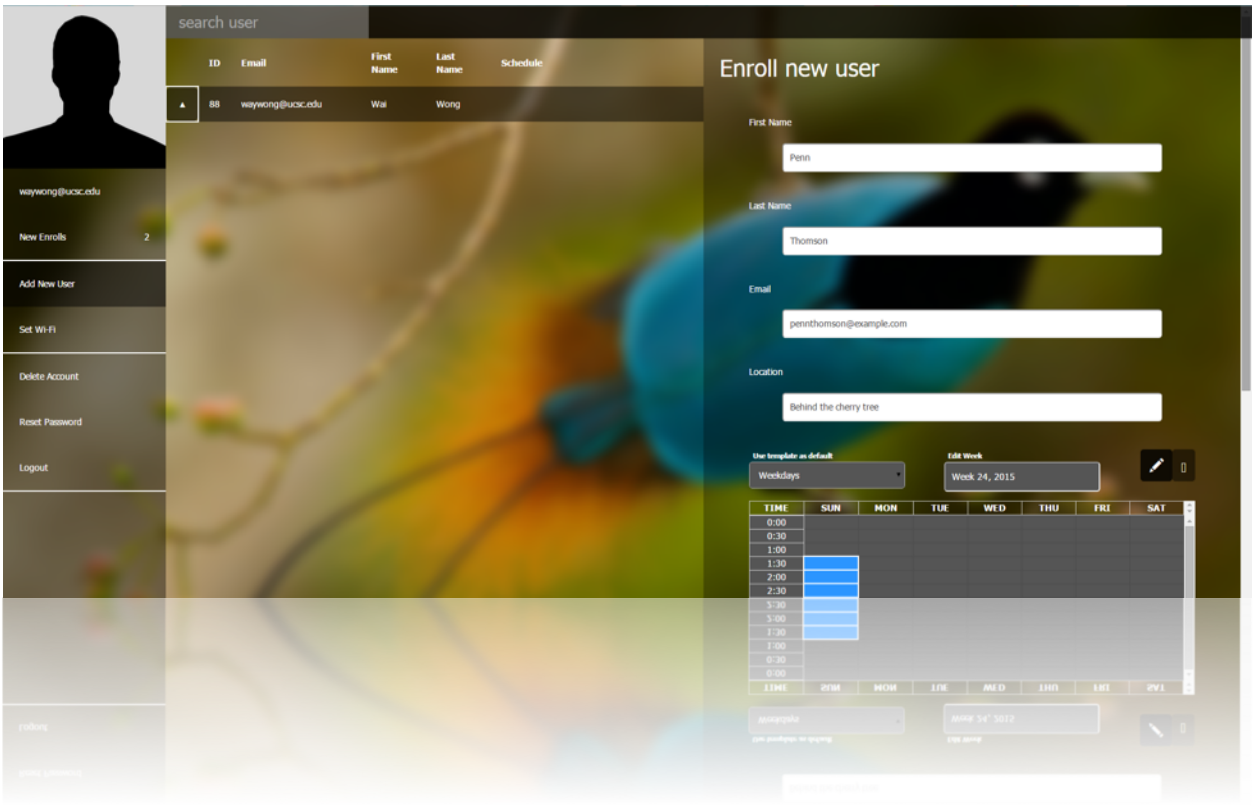
- User Registration - This is a standard part where the user enter in personal information including name, email, product ID as indicated in the package that will come with the door lock and password. Once this information is confirmed, the user will be redirected to the account page of this web interface.
- User Login - Using a registered account with just the email and password, the user will continue to the account page
- Forgot Password - When the user enters a registered email, an email will be sent or an error will be displayed otherwise as shown below.



Account Page

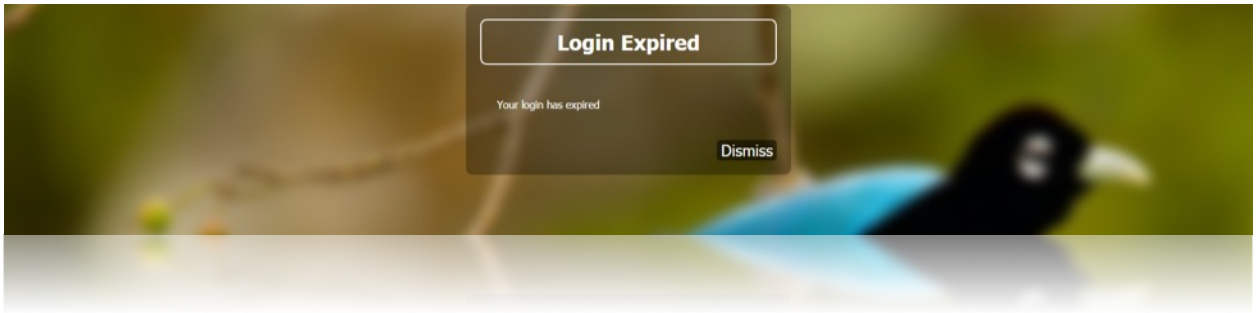
- Push Notifications - When the smart door lock system have new users enrolled on the local interface, notifications will be sent to the internet every thirty minutes. These new additions will be shown on the left hand menu and notify the user to enter names, schedule and email to invite each newly enrolled user to set their password. The physical door lock system has a thirty minute update to synchronize the information between the web server and the physical system. New notifications will be sent via an HTTP post request to the web server during the updates. Setting the Wi-Fi information on the location will trigger this update as mentioned previously.

- Add / Edit User - When adding a new user, the current user must provide the name, email, location of the door lock, and the access schedule of this user. After submitting this information, the new user will receive an email invitation with a link to the local interface and under the supervision of an authorized personnel, for the door to be opened, in order to enroll her fingerprint. Editing a user is very similar in that the name, schedule can be changed and even the account can be deleted if the current user has the authorization to do so.



- Set Wi-Fi - This information is set in the physical door system such that the system will use the specified network connection to connect to the internet. This includes the correct SSID and password for the connection to work, commonly through a home router. When this information is submitted, the next thirty minute update will
- Delete Account - This option is for deleting all information, accounts, and users associated with the door lock including the fingerprint profiles and users on the micro-controller. This will signal the physical system to remove all information during the next thirty minute update.
- Reset Password - Completing this form with the correct old user password will reset the user's password.

- Login Expired - This page shows that the login session has expired after thirty minutes of no action. For security measures, this also works against the back page button and hacking by manually entering a web URL. In any case, the user is redirected to the login page.



Server Program

Database

The database uses Google's Datastore API and is separated into two parts: the user and the product. The user object is used primarily to store information including first and last names, fingerprint ID, passwords, emails, belonging admin, access schedule, schedule template, SSID, network password, and a generated token which is used for password reset and enrollment invitations. The product object is associated to each door lock which has been manufactured and is used for synchronizing the information for the thirty minute interval update. It includes product ID, changed schedules, deleted users and new users who enrolled through the local door system but has not yet set-up an online account.

None of the functions are very exciting - they simply pull the objects from the database accordingly, set new values and put it back. They also assert the authorization before proceeding and response with error in the case of failure, errors, or denial of access.

Memcache

This is a feature in the Google's cloud server which works the same way cookies do on web browser, except this is for web servers. It stores temporary information and associates each piece of info with a key so the server can use the key to get the information back. This is different than SQL (GQL in this case) in Google's Datastore in that it is a hash table and has no filtering options. The reason for using this is because it is much faster and puts off the load that

would otherwise be put on the more expensive SQL manipulations used in the Datastore. Also, memcache has expiry which is perfect for sessions, enrollment expiry and more.

In our case, we used this for forget password in verifying the token which is randomly generated to make sure the user with the link gets to reset the password without knowing her password. The link expires quickly such that even if a curious person looks into her email and click on the link, this person cannot change the password again. The same idea applies to when a person receives email invitation to enroll or receive push notifications. This scheme keeps information more private while not putting any situation into a deadlock.

SMTP

This is Google's emailing API which allows the server to generate and send HTML encoded emails to any recipient from the accounts authorized in the Google accounts which own the server. Put it in a loop and out goes the email to everyone in the list. This is used for forget password, enrollment invitation, and account set-up invitation as discussed previously. To recap: forget password emails a link to reset the password; enrollment invitation directs the person with the certain email to enroll at the physical door system; account set-up invitation links the physically enrolled person with the certain finger ID to set up her online account.

Front End Program

Home Page

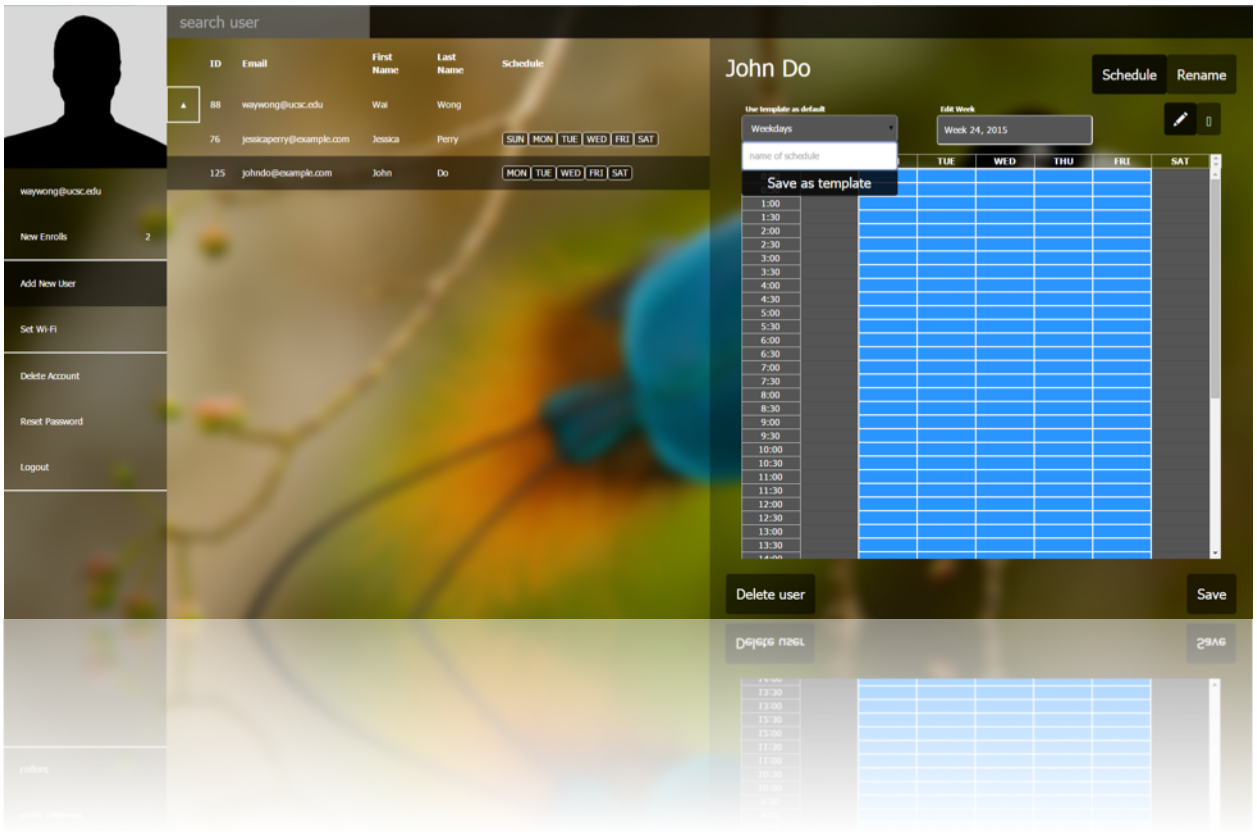
As with all other websites, the program uses JQuery to handle mouse events, bind these events to elements and Ajax submit forms to the server. This page checks for errors before sending it off to the web server to minimize the false request attempts in the server. Also, each button is linked to a particular form and depending on the button, the page does not necessarily need to communicate with the server, such as switching between displaying the registration form and the login form.

Account Page

This page is organized the exact same way but handles cookies for expiry as well because this is the page where the user is logged in. Also, this is a single page application such that all server responses will be rendered without refreshing the page. This is done by using JQuery's

Ajax method and simply one line to prevent the page from redirecting even after receiving information from the server which is what all browsers do by default.

For visuals such as schedules and list of users, a combination of CSS, JQuery, Glyph Icons is used for resizing, colors, keeping things relative to each other, transparency, fade in / out, hovering, icons, images and texts. The schedule is parsed back and forth to be sent to and from the server and be rendered as HTML elements. The following figure shows most of the visuals.



References

[1] Microchip, “Low-Power, Low-Cost, General Purpose 16-Bit Flash Microcontrollers with XLP Technology,” PIC24F16KL402 datasheet, 2013.

[2] Duracell, “Rechargeable NiMH Battery,” DC1500 datasheet, 2012.

[3] Linear Technology, “Dual 4A Ideal Diodes with Adjustable current Limit,” LTC4415 datasheet, 2012.

[4] Linear Technology, “20V High Efficiency Nanopower Step-Down Regulator,” LTC3388 datasheet, 2012.

[5] Linear Technology, “1A Buck-Boost DC/DC Converter with Programmable Input Current Limit,” LTC3127 datasheet, 2011.

[6] Texas Instruments, “Battery Charging,” Literature No. SNVA557.

[7] Texas Instruments, “Linear and Switching Voltage Regulator Fundamentals,” Literature No. SNVA558.

[8] Linear Technology, “Basic Concepts of Linear Regulator and Switching Mode Power Supplies,” Application Note 140, October 2013.

[9] Texas Instruments, “A solar-powered buck/boost battery charger,” High-Performance Analog Products, *Analog Applications Journal*, 2012.

[10] D. Schelle and J. Castorena, “Buck-Converter Design Demystified,” *Power Electronics Technology*, June 2006.