CMPE123 Senior Design Project: Lock Management System

Bowen Brooks, Samuel Wu *

Contact: bojbrook@ucsc.edu, sazwu@ucsc.edu

Abstract

This project is for the Computer Engineering Senior Design Capstone, at University of California, Santa Cruz, CMPE123. The project is meant to create an Internet of Things (Iot) capable smartlock to be used in a building. Each room will have a cloud-connected smartlock that is unlocked with Near-Field Communication (NFC) technology. The project emphasizes cloud connectivity, power management, and cost.

*Thanks to Professor Anujan Varma and teaching assistants Sargis Yonan and Sam Mansfield for their guidance during this project

Contents

1 Summary of Project(Bowen)

This goal of this project is to create a lock management system that could potentially replace the BSOE current system. Our system will have real time access to the rooms through Google cloud. Each lock will have a microcontroller connected to an NFC sensor. A user will unlock the room by tapping their android phone to the NFC sensor. The locks will have access to the Internet through WIFI. They will access Google cloud compute using TCP/SSL to get user access to the lab. Their will also be an administrative website that can view real time logs for each room and grant/revoke access at any given time.

2 Block Diagram (Sam)

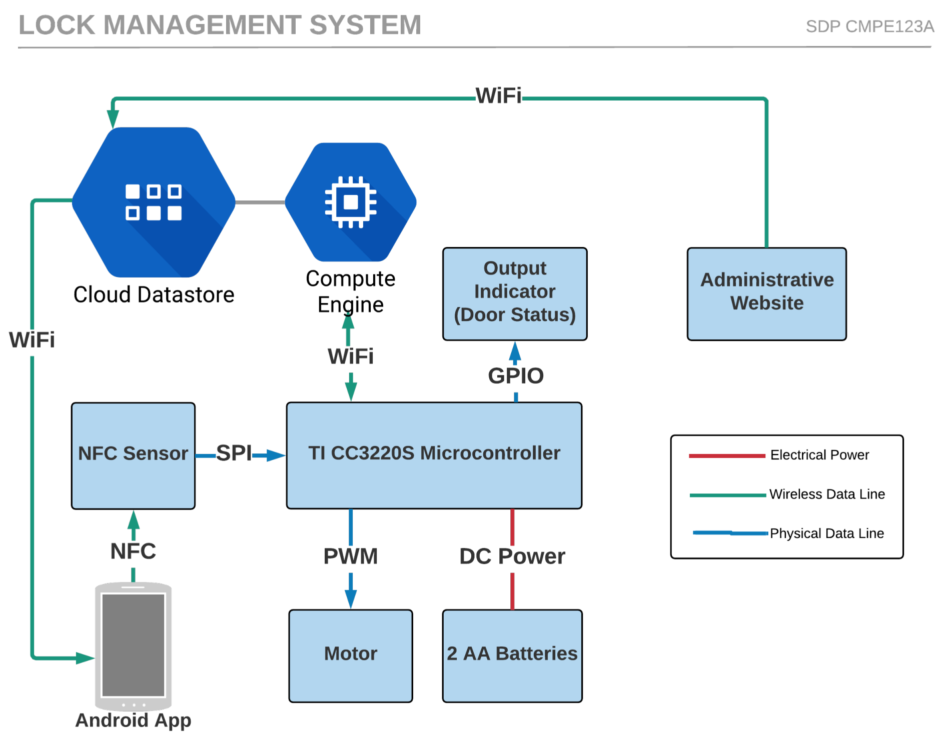

Figure 1: Block Diagram of System

3 Software Flow Chart (Sam)

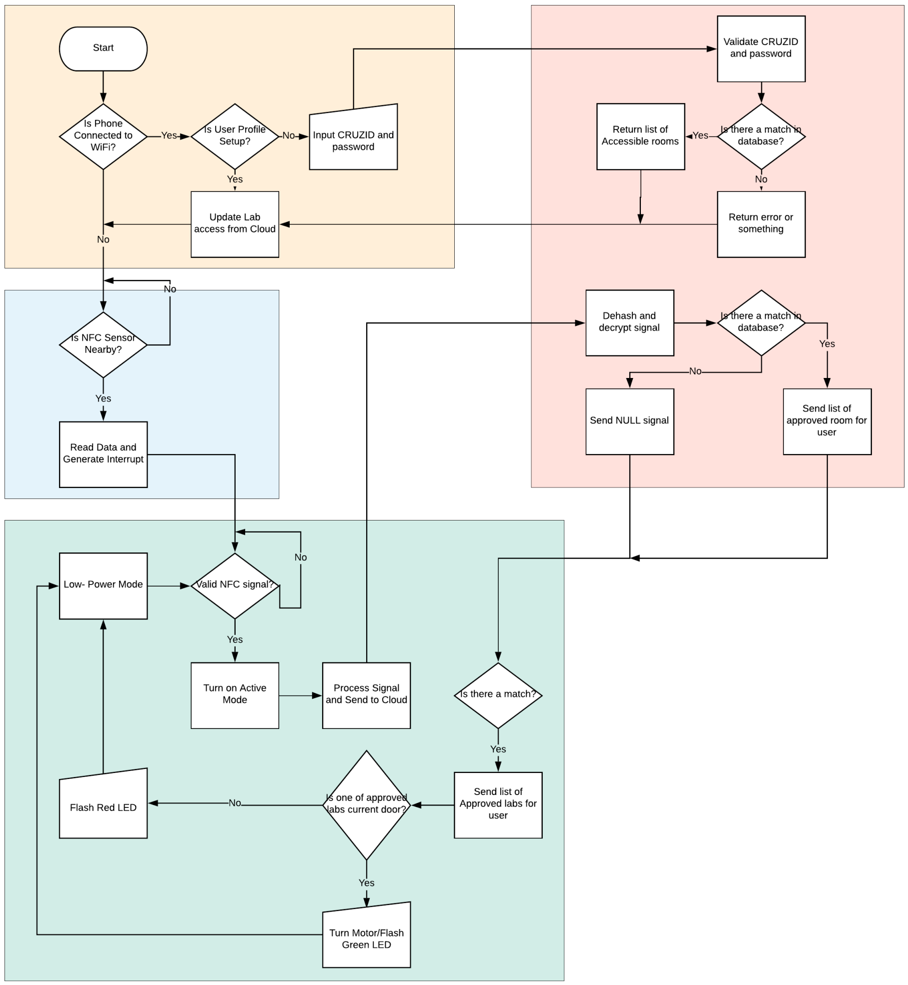

Figure 2: Yellow: Android app; Blue: NFC sensor; Green: microcontroller; Red: Google Cloud

4 Circuit Schematic (Sam)

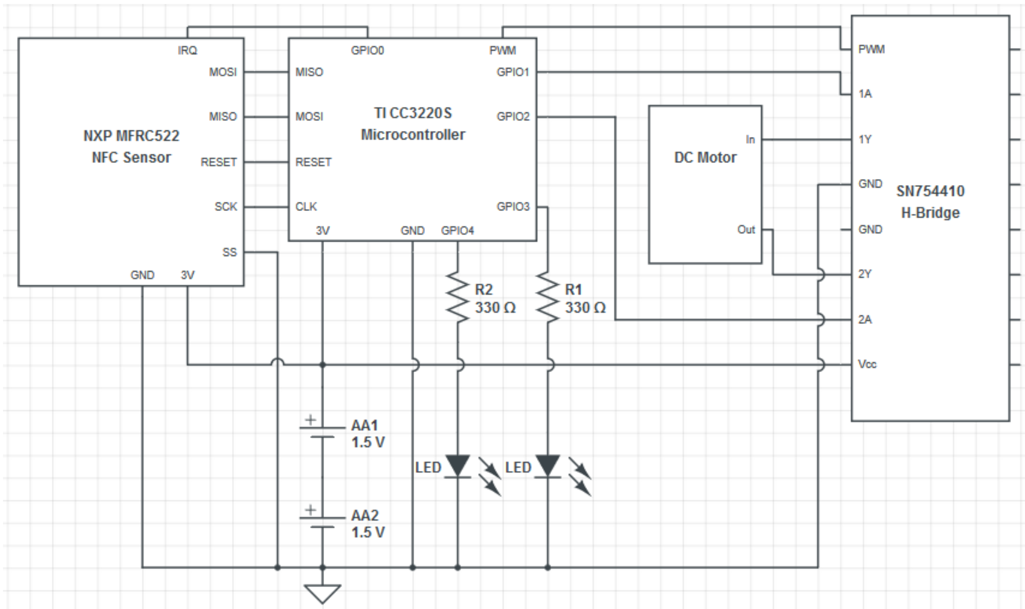

Figure 3: The circuit schematic of the project.

5 Major Components(Bowen)

5.1 TI 3220S Microcontroller

The microcontroller is hooked up to an NFC sensor which reads in a CruzID. This be sent to Google Compute Engine which returns a 1 or a 0 depending on if they have access.If a one is returned then the motor will turn to unlock the door. The microcontroller will be cycling through low power mode in order to preserve the battery life from the AA batteries.

5.2 TI 3220s API

5.2.1 is_allowed()

```
int is_allowed(char * cruzId, char * roomNumber);
```

Function sends the data to the gateway and returns a 1 if the student is allowed and a 0 otherwise.

5.3 Google Compute Engine

This a virtual machine running in the cloud. IT has a static IP address and will act as the server for the door locks. The connection will be using TCP and SSL for security. The micro will send its room number and the cruzID of the user. The VM will run a query and send back either a 1 or 0. Once it sends the results it kills the connection. It then logs the results into the Datastore for the given lab.

5.4 Google Compute Engine API

5.4.1 get_student_classes()

```
def get_student_labs(cruz\_id)
```

This function gets all of the classes thats the student is currently allowed access to.

5.4.2 get_classes_for_room()

```
def get_student_labs(room_number)
```

Function returns a query from the google cloud which list all of the classes that currently have access to the given room.

5.4.3 LogInRoom()

```
def LoginRoom(room_number, cruzID):
```

This function is called when a user enters a room. The first argument is the room number of which the user enters and the second argument is the cruzID of the user that enters the room. The function logs the enter time and stores it in the cloud. The leave time and cumulative time spent in the room is null.

5.4.4 LogoutRoom()

```
def LogoutRoom(room_number, cruzID):
```

This function is called when a user leaves a room. The first argument is the room number of which the user leaves and the second argument is the cruzID of the user that leaves the room. The function logs the leave time and stores it in the cloud. The cumulative time is calculated based on the leave time and the last time the student entered the room.

5.5 Google Cloud

Students, Faculty and rooms are all stored in Google Cloud. The cloud will respond from results from the microcontroller which will determine if a user has access to the rooms. The cloud storage can only be updated and from the administrative website. Students and rooms can be updated at anytime giving real-time access to the rooms. The cloud also logs all the login information for each room and which can be view on the administration website.

5.6 Administration Website

The administration website is where the faculty can add and revoke student access to the rooms. They can also view analytics from Google Cloud such as peak usage time and current room capacity.

5.7 Administration API

5.7.1 AddNewUser()

```
def AddNewUser(index, name, cruzID, priority, class_list = []):
```

This function adds to the "User" entity. The first argument is the index in the database, which acts as the key in case it needs to be accessed later. The second argument is the name of the student, with first and last name separated by a space. The third argument is the cruzID of the student without the ucsc.edu following it, i.e. sazwu. The fourth argument is the user priority. Highest priority is given to the lowest number. For now, admin priority = 0, faculty priority = 1, grad priority = 2, and undergrad priority = 3. The last argument is a Python list of classes the student is in i.e. CMPE167, AMS147.

5.7.2 AddNewRoom()

```
def AddNewRoom(index, room_number, class_list = []):
```

This function adds to the "Room" entity. The first argument is the index in the database, which acts as the key in case it needs to be accessed later. The second argument is the room number of a room that has an a lock on it, i.e. BE301, BE340A, E2-599. The third argument is a Python list of classes that have access to the room i.e. CMPE123A, CMPE121, CMPE167L.

5.7.3 ModifyRoomEntity()

```
def ModifyRoomEntity(lab_index, room_number, class_list = []):
```

This function modifies an existing "Room" entity. The entries in the entity must all be specified in the arguments. The first argument is the index in the database. The second argument is the room number in a room with an accessible lock. The third argument is a Python list of classes that have access to the room.

5.7.4 ModifyUserEntity()

```
def ModifyUserEntity(user_index, name, cruzID, class_list = []):
```

This function modifies an existing "User" entity. The entries in the entity must all be specified in the arguments. The first argument is the index in the database. The second argument is the first and last name of the student separated by a space. The third argument is just the cruzID of the student, and the fourth argument is a Python list of classes the student is in.

5.7.5 QueryAllRoomsEntities()

```python
def QueryAllRoomsEntities():
```

This function simply prints all "Room" entities and the information in each.

5.7.6 QueryAllUserEntities()

```python
def QueryAllUserEntities():
```

This function simply prints all "User" entities and the information in each.

5.7.7 DeleteRoomEntity()

```python
def DeleteRoomEntity(lab_index):
```

This function deletes an entire "Room" entity specified by the index passed in as the first argument.

5.7.8 DeleteUserEntity()

```python
def DeleteUserEntity(lab_index):
```

This function deletes an entire "User" entity specified by the index passed in as the first argument.

5.7.9 LogInRoom()

```python
def LoginRoom(room_number, cruzID):
```

This function is called when a user enters a room. The first argument is the room number of which the user enters and the second argument is the cruzID of the user that enters the room. The function logs the enter time and stores it in the cloud. The leave time and cumulative time spent in the room is null.

5.7.10 LogoutRoom()

```
def LogoutRoom(room_number, cruzID):
```

This function is called when a user leaves a room. The first argument is the room number of which the user leaves and the second argument is the cruzID of the user that leaves the room. The function logs the leave time and stores it in the cloud. The cumulative time is calculated based on the leave time and the last time the student entered the room.

5.7.11 CountRoomPopulation()

```
def CountRoomPopulation(room_number):
```

This function counts the number of people in the room specified by the first argument. It counts the number of people by counting the number of indexes in the room with a null value in the exit time.

5.7.12 ClearRoomLog()

```
def ClearRoomLog(room_number):
```

This function clears the entity and log of the room number specified in the first argument. This function could probably be used at the end of the day or end of the quarter.

5.7.13 HistogramEnterTime():

```
def HistogramEnterTime(room_number):
```

This function gives a Python list of size 24 with the frequency of time entered. Each hour interval corresponds to an index in the list. The function requires the first argument to be the room number of the room of which the histogram is desired.

5.7.14 HistogramExitTime():

```
def HistogramExitTime(room_number):
```

This function gives a Python list of size 24 with the frequency of time exited. Each hour interval corresponds to an index in the list. The function requires the first argument to be the room number of the room of which the histogram is desired. Any user in the room will have a "null" value for exit time, and thus will not be counted.

5.7.15 AddClassToRoom()

```
def AddClassToRoom(room_number, classes = [])
```

This is a helper function for parsing the CSV file when using function ParseCSVRoomEntity(). It takes in the room number and the classes to add. For example, if room "BE-340" can now be accessed by classes "CMPE167 and AMS114", the function would take in "BE-340" as the first argument and the list "CMPE167 and AMS114" as the second argument.

5.7.16 ParseCSVRoomEntity()

```
def ParseCSVRoomEntity(file)
```

This function parses a CSV file to create and add to a room entity. The first row of the .csv file is not read, as it is meant to label the columns. The first column of the file should be the room number and the second column of the file should be a list of classes that can access the room, separated by spaces.

5.7.17 CSVToCloudRoom()

```
def CSVToCloudRoom(file)
```

This function calls ParseCSVRoomEntity() and adds the data from the .csv file to the "Room" entity. The first row of the .csv file is not read, as it is meant to label the columns. The first column of the file should be the room number and the second column of the file should be a list of classes that can access the room, separated by spaces. The only argument should be the .csv file name.

5.7.18 AddClassToUser()

```
def AddClassToUser(user, info_list = [])
```

This function adds a list of classes to the user, and is used as a helper function for ParseCSVUserEntity() and CSVToCloudUser().

5.7.19 ParseCSVUserEntity()

```
def ParseCSVUserEntity(file)
```

This is a helper function used by CSVToCloudUser(). Refer to section CSVToCloudUser().

5.7.20 CSVToCloudUser()

```
def CSVToCloudUser(file)
```

This function takes in the data from a .csv file and puts it directly in the "User" entity. The .csv file should have the information name, cruzID, priority, and classes in that order. If there are multiple classes, the classes can be separated by spaces while staying in that column.

5.8 Android App

The android application allows for a user to sign in using their CruzID and password. When the phone is tapped against an NFC sensor the application will transmit the CruzID to the microcontroller. The application will have access to the Google cloud in order to view which rooms they have access to.

5.8.1 GetListOfLabs()

```
ArrayList<Integer> GetListOfLabs(String cruzId):
```

This function asks the google datastore what labs the user has access to with the cruz id. It return a list of all the labs they can get into for the quarter.

6 Battery (Sam)

The battery consumption is theoretical at this point, and is based off of a weighted sum of the manufacturer's specification in the datasheet for the TI CC3220S microcontroller and the NXP MFRC522 sensor. $amps_{total}$ is the weighted sum for amps used in a day.

$$\begin{aligned} &amps_{total}(\#logs) \\ &= \frac{(\#logs)(time_{log})}{60*60*24s}(amps_{log}) + \frac{60*60*24s - (\#logs)(time_{log})}{60*60*24s}(amps_{sleep}) \end{aligned} \tag{1}$$

$time_{log}$ is the total time it takes to do one complete log, including the time to turn sensor on, transmit over WiFi, and then receive over WiFi.

$$time_{log} = time_{sensor} + time_{tx} + time_{rx}$$

$$\begin{aligned} &amps_{total}(\#logs) \\ &= \frac{(\#logs)[(WiFi_{tx})(time_{tx}) + (WiFi_{rx})(time_{rx}) + (sensor_{on})(time_{sensor})]}{60*60*24s} \\ &\quad + \frac{60*60*24s - (\#logs)(time_{log})}{60*60*24s}(sensor_{sleep} + micro_{sleep}) \end{aligned} \tag{2}$$

$$\begin{aligned} amps_{total}(\#logs) = &\frac{(\#logs)[(266mA)(1s) + (53mA)(3s) + (100mA)(1s)]}{60*60*24s} \\ &+ \frac{60*60*24s - (\#logs)(5s)}{60*60*24s}(10\mu A + 1\mu A) \end{aligned} \tag{3}$$

$$\begin{aligned} amps_{total}(\#logs) &= 4.884*10^{-6}(\#log) + 11*10^{-6} - 6.366*10^{-10}(\#log) \\ &\approx 11*10^{-6} + 4.884*10^{-6}(\#log) \end{aligned} \tag{4}$$

AA batteries are typically anywhere from $2000 - 3000mAh$, so taking the lower bound, our 2 AA batteries in series will have approximately $4000mAh$ worth of charge.

$$time_{months}(\#logs) = \frac{1}{24*30} * \frac{amphour}{amps_{total}} \tag{5}$$

$$time_{month}(\#logs) = \frac{5.555*10^{-3}}{11*10^{-6} + 4.884*10^{-6}(\#logs)} \approx \frac{1137.39}{\#logs + 2.252}$$

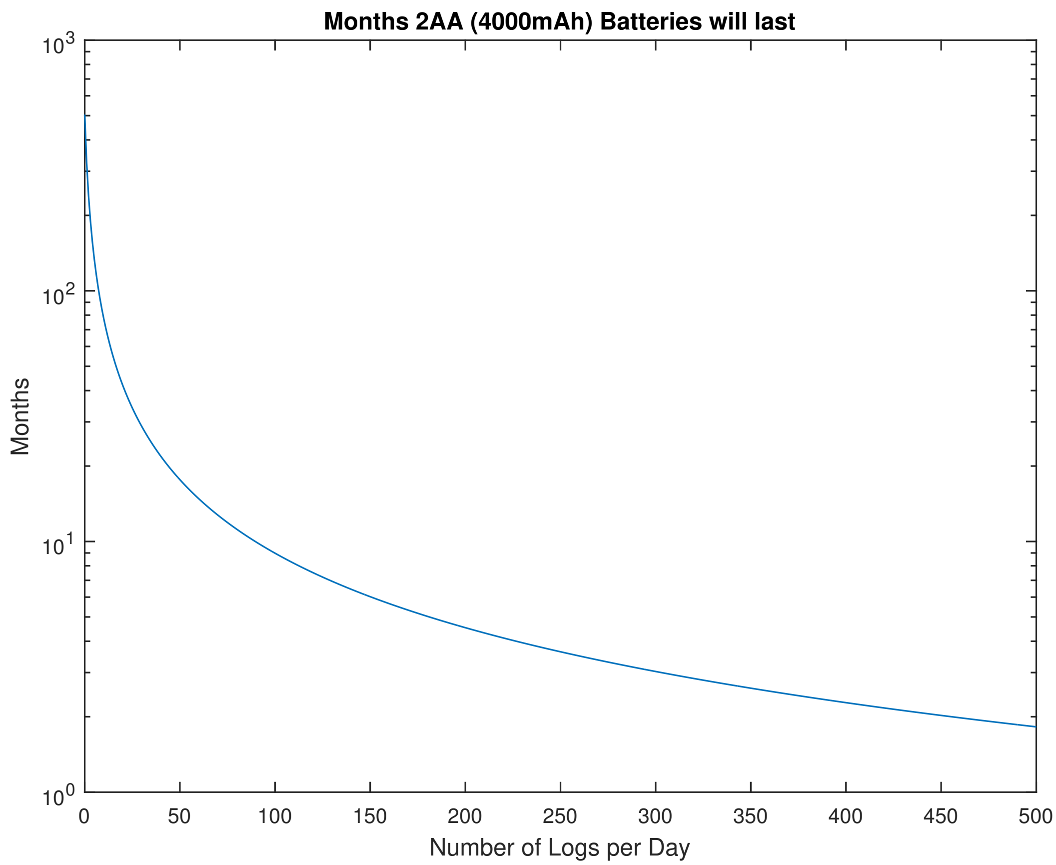

Figure 4: Theoretical Battery life using 2 AA batteries (4000mAh) and the NXP MFRC522 Sensor

7 Entities(Sam)

Room Entity		
ID	Room #	Groups
name=index0	E2-399	["BELLS","AMS147","CMPS101"]
name=index1	BE340A	["CMPE123A","CMPE123B", "CMPE129B"]

Student Entity			
ID	Name	cruzID	Groups
name=index0	Samuel Wu	sazwu	["AMS147","CMPS12B"]
name=index1	Bowen Brooks	bojbrook	["CMPE123A"]

BE340 Log Entity				
ID	cruzID	Enter Time	Exit Time	Cumulative
name=sazwu0	sazwu	02-07 17:26:55	02-07 17:27:02	0:00:07
name=bojbrook0	bojbrook	02-07 17:26:55	02-08 11:30:22	18:03:26
name=sazwu1	sazwu	02-08 11:27:19	02-08 11:27:57	0:00:44
name=hello0	hello	02-08 11:31:02	null	null

8 Schedule(Bowen & Sam)

8.1 Winter Quarter

8.1.1 Quarter Goals

We want to be able to have the microcontroller send and receive information to and from the cloud. Additionally, the NFC sensor should be able to read unique *RFID* tags.

8.1.2 Tasks for the Quarter

1. RFID/MCU communication
2. Cloud/MCU communication
3. Database
4. LEDs
5. Power management

Bowen	
Week 4	Design database, Design cloud API, Populate database
Week 5	Design cloud API
Week 6	Cloud AUTH/access, Design cloud API
Week 7	MCU Push/pull database, MCU Log interaction
Week 8	Cloud push results to MCU
Week 9	Finish MCU, Cloud clean up
Week 10	Start API calls for website

Sam	
Week 4	Learn basic Google Cloud, Design database, Design cloud API
Week 5	Populate the database
Week 6	MCU internet access, SPI interface for sensor
Week 7	MCU/RFID communication (MRFC522)
Week 8	MCU/RFID communication (PN522)
Week 9	Differentiate unique RFID tags, power management
Week 10	RFID wakeup MCU, sleep/wake modes

8.2 Spring Quarter(Bowen & Sam)

8.2.1 Quarter Goals

Everything should be completely finished. This includes full functionality of the Android App for communicating with the cloud and the NFC sensor with a *NFC* signal. Additionally, there will be an administrative website that pulls data from the cloud. The website will have administrative functionalities such as: adding, removing, or modifying user privileges. The administrator will also be able to navigate a clean UI to view analytics and data.

8.2.2 Tasks for the Quarter

1. NFC/MCU communication
2. NFC/App communication
3. Website
4. DC Motor/H-Bridge
5. Web/Cloud communication
6. App/Cloud communication

Bowen	
Week 1	Familiarize with Android API, Design UI
Week 2	Clean up anything from Winter, NFC/App comm
Week 3	NFC/App comm, App/Cloud comm
Week 4	App/Cloud comm, start design for website
Week 5	Finish app, cleanup app UI, website UI
Week 6	Website/cloud comm, website UI
Week 7	Finish website, debug all comm
Week 8	Debug all comm, debug any small things
Week 9	Buffer week, start report/presentation
Week 10	Finish everything

Sam	
Week 1	MCU/NFC comm (M6E Nano), App/NFC comm
Week 2	MCU/NFC debug, App/NFC comm
Week 3	DC Motor, clean up MCU code
Week 4	App/Cloud comm, Website
Week 5	Website UI, Web/Cloud comm
Week 6	Fix any small bugs, DC motor/H-Bridge
Week 7	Cleanup App UI, website UI
Week 8	Debug everything, buffer week
Week 9	Buffer week, start report/presentation
Week 10	Finish everything

9 List of hardware components(Bowen)

Component	Cost	Quantity	Total
TI CC3200S	$39.99	2	$79.98
Battery Case	$1.50	2	$3.00
Op Amp	$0.95	4	$3.80
Resistor Kit	$7.95	1	$7.95
NFC Sensor MRFC522	$9.99	0	$0.00
AA battery 20 pack	$8.54	1	$8.54
NFC Sensor PN532	$12.99	2	$25.98
Motor	$1.95	2	$3.90
LEDs 5 Pack	$2.95	1	$2.95
H-Bridge	$2.35	2	$4.70
Total Cost			$140.80