Northeastern University

College of Engineering

Department of Electrical and Computer Engineering

Boston, MA

Capstone 2 Design Proposal April 24, 2016

Wireless Parking Detection System

Yuecan Fan- Weining Zou – Xi Zhang – Mengfei Zhang – Sining Liu

TABLE OF CONTENTS

FIGURES

ABSTRACT

Almost every driver has experienced that spend half-hour or even more to find a street parking space. Or sometimes every garage around the mall is full and don't allow you to go inside even there might still have several available spaces. Sometimes you visit to a strange city or scenic spots and don't know where to park or which garage is the cheapest. During the weekend, you always spend more time on find a parking space than on the way.

Wireless Parking Detection System is a system that help you find a valid street parking space around you. All the data shows on the App are real time collected by our developed detect devices. Also, we provide the information of the garages around you include the parking fees, overnight availability, parking spaces availability and locations. All the functions of our system are designed to solve the problem of find a parking space.

Right now, our system can only detect the street parking spaces that have our devices installed. We hope that this system continued so that it can also shows the map of garages and their information. And if possible, we hope we can have contract with Boston government so that they do give us permit and funds to install our device in every public parking space in Boston, which means if you download our App, you will know all the parking spaces in Boston.

INTRODUCTION

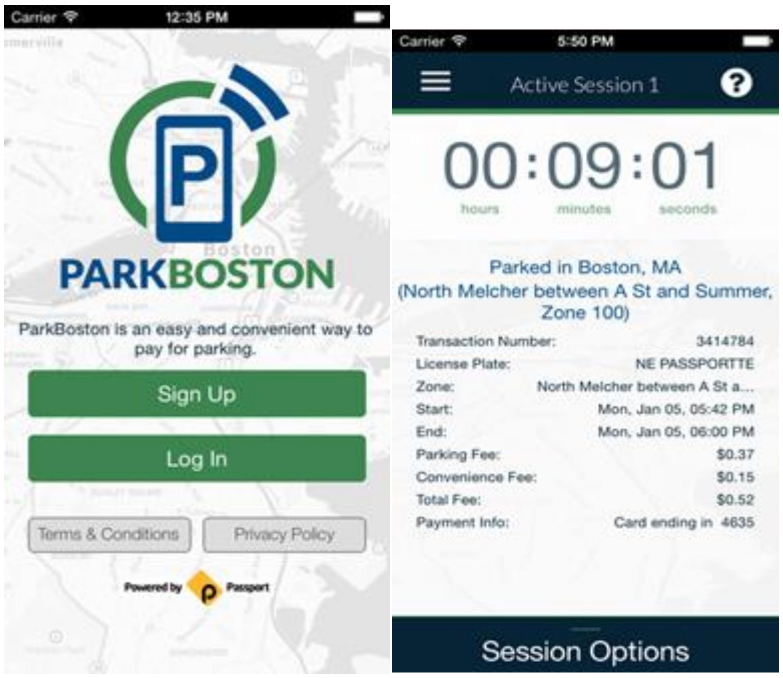
Detail Description of Wireless Parking Detection System

The Wireless Parking Detection System is actually composed by two parts: Hardware devices part and software App part. Our device can detect if there is a car in the corresponding parking space by the electromagnetic sensor. Then the sensor sends this information wirelessly to our transfer station which is connected with internet. Then the transfer station uploads the data to Cloud to be download by our App. Our App use the data from Cloud to provide the real time parking information to our users. The other functions of garage are pre-build in the App and collected by ourselves with the garage. If we contract with those private garage successfully, we may also have rewards system which can give a discount to those drivers who pay the garage fee by our App. And for those devices used in the garage, they are not wirelessly because they can use the internal network of the garage which will reduce the cost significantly.

Similar Projects

Park Boston

There is an existed App called “ParkBoston”. This App can let you pay your meter by phone and extend your parking time by phone [1]. Boston government had assigned a zone number to each street. So if you enter your zone number and your plate number, you can pay your meter by this App. However, you are not able to find where is an empty meter parking by this App which means this App just add a credit card function to those old style meter. (There is a batch of new meters that accept credit cards.)



Parked in Boston, MA
(North Melcher between A St and Summer, Zone 100)

Transaction Number:	3414784
License Plate:	NE PASSPORTTE
Zone:	North Melcher between A St a...
Start:	Mon, Jan 05, 05:42 PM
End:	Mon, Jan 05, 06:00 PM
Parking Fee:	$0.37
Convenience Fee:	$0.15
Total Fee:	$0.52
Payment Info:	Card ending in 4635

Figure 1: Park Boston App

BMW Parking Assistant System

As an automobile industry magnate, BMW also want to develop a new system which can help you with parking. Their method is using real time traffic information to do an analysis, then they will give you an estimate result that where might have a parking space. So their problem is parking space is not something can be calculated. So the veracity of this system will be a big challenge.

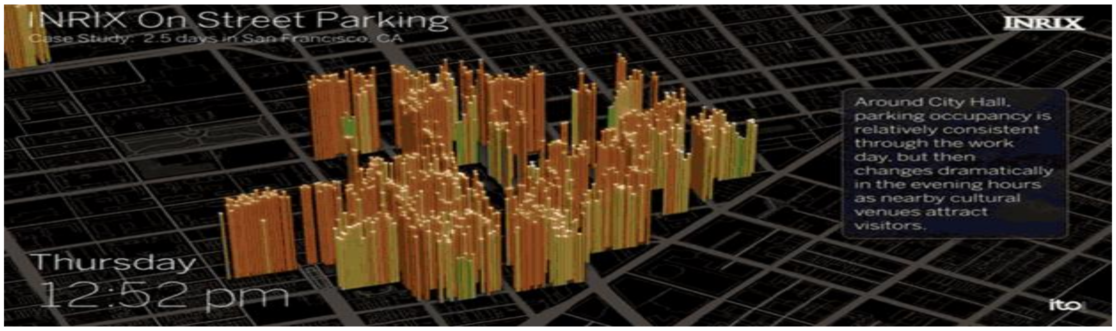

INRIX On Street Parking
Case Study: 2.5 days in San Francisco, CA
INRIX
Around City Hall, parking occupancy is relatively consistent through the work day, but then changes dramatically in the evening hours as nearby cultural venues attract visitors.
Thursday
12:52 pm
ito

Figure 2：BMW Parking Assistant System Website [2]

Parking sharing app - “Sweetch”

“Sweetch” is an app for sharing parking slots: assume A is looking for a slot while B is about to leave, and using the parking slot need $5. B upload his current slot onto Sweetch, then once get the slot, B can get $4 refund.

However, this app is not able to make things easy. It seems people who are willing to pay will be easier to get a parking place. Also, it may encourage people to wait for those anxious drivers pay for the slot, but not make the slot for other drivers on their own.

Some similar apps like MoneyParking, even breaks the law. The users of MoneyParking violate the Police Code section 63(c) that sell or rent public parking slots will be charging $300 fine. This business pattern also against the “Unequal competition law” in CA, which the company will be faced $2500 fine.

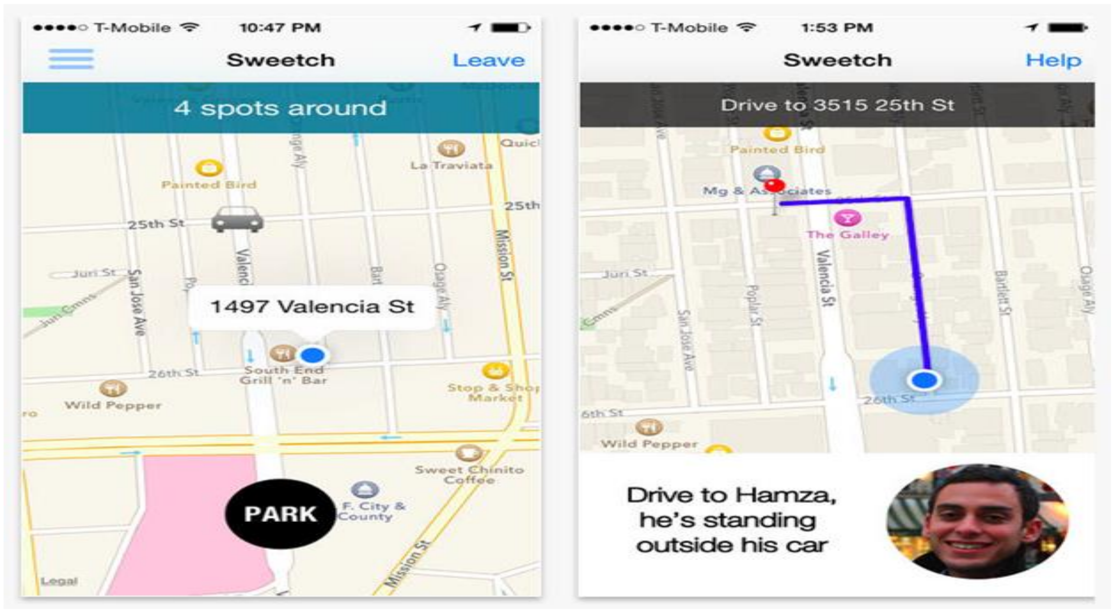


Figure 3: Sweetch App [3]

Market Analysis

Our App is designed to solve the parking problems which means our client base is all the drives. We do believe that if our project keeps to be developed and finally come out as an integrated App, our uses will be as many as Facebook.

Our main cooperation has two targets: governments and private garage company. For the governments, we will try to get the permit and support funds from them because our project helps them increase the citizens' QOL (quality of life), which is a main job of government. For the garage company, we do charge them for the device onetime and for the information advertisement by contract. Because our project can increase the popularity of garage and also increase the utilization rate of the garage.

PROBLEM FORMULATION

Our Solution

The Park Your Car system was designed to detect and display an available street parking spot synchronously and wirelessly.

Current parking websites only include some garage parking information and some apps are designed only for more convenient payment. We chose to make the device detect available street parking spots for several reasons; Drivers always faced problems in finding an available street parking position, they spent a lots of time to wait or hang around, our device can show the available spot on an app once the place is empty, they can know whether there is a spot immediately. Also the parking fee in a garage is too expensive for drivers especially for student, our device can help them save money.

We decided use sensors to detect the available parking spot on the street, wirelessly upload parking data to Cloud Server, and then visualize real-time street parking data on app. Consider we may not able to cover all street parking information, we build a function on app to let user do self-report.

Design Process

Our design process began with researching what is out there today and finding what weakness do those exist products have. This brainstorm session leads us to thinking about if we can improve those products. We want to come out a device that can solve a problem that the majority of people's face. We found people are always struggling to find a place to park. So, if we can build a system that let people know where a parking place is available, it will be a great system. There are some products out there can detect if there is a vehicle on a parking spot. However, most of them are wired/immobile. Besides, none of them can connect to an app to let user know immediately. By collecting data and transmitting it to users at the same time wirelessly, available parking space can be easily find by users.

Once we had formulated what kind of product we wanted to build, we had to figure out how exactly we built it. The Wireless Parking System need a sensor put on the parking place to detect whether there is a vehicle or not. This kind of sensor should not be influenced by any other environment noise like people or animals. The sensors are converted from analog to digital. This now digital output need to be put into a microcontroller which can do analysis and calculation base on those digital data. The output of microcontroller is ready to go wireless. A radio is needed to let transmit output information wirelessly. A base station-like device then collect all the data and upload it to cloud which can be reached from the app we built. At this point, we need to find a cloud we can use to store the data. The sensor, microcontroller and radio can be wired together as a sensor node. Each sensor node is powered by an alkaline battery that have the right voltage for the microcontroller. Converters may be needed for offering the right voltage to sensor or radio.

The sensor should face directly to vehicles so there is a hole on our sensor node to let sensor uncovered. The sensor node need to be hard enough in case hitting by vehicles.

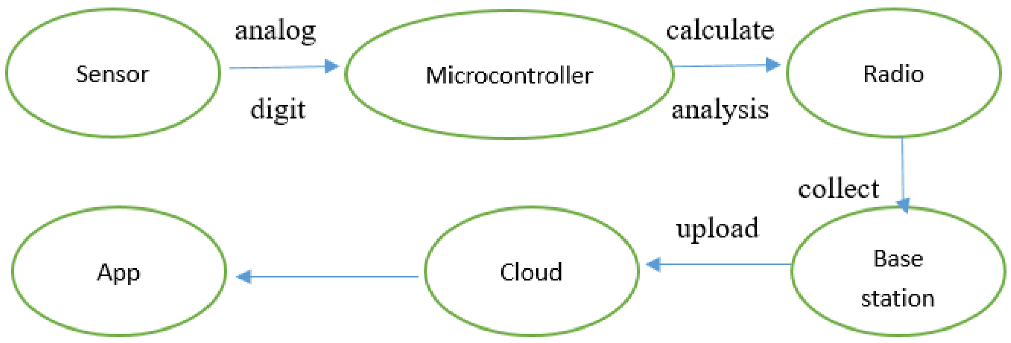


Figure 4: Design process

Measurement Analysis and component choice

Once our design was complete, we began with researching what kind of information sensors should take. Vehicle all contain lots of irons. So, magnetic sensor is the best choice for our Sensor Detection System. Compared with other sensors, magnetic sensors are smaller and won't be affected by climate change and noise in the environment.

The magnetic field not only can detect a car is parking, but also entering and leaving. We will assume there are four phases: Vacant, Entering, Parking Stop, Leaving.

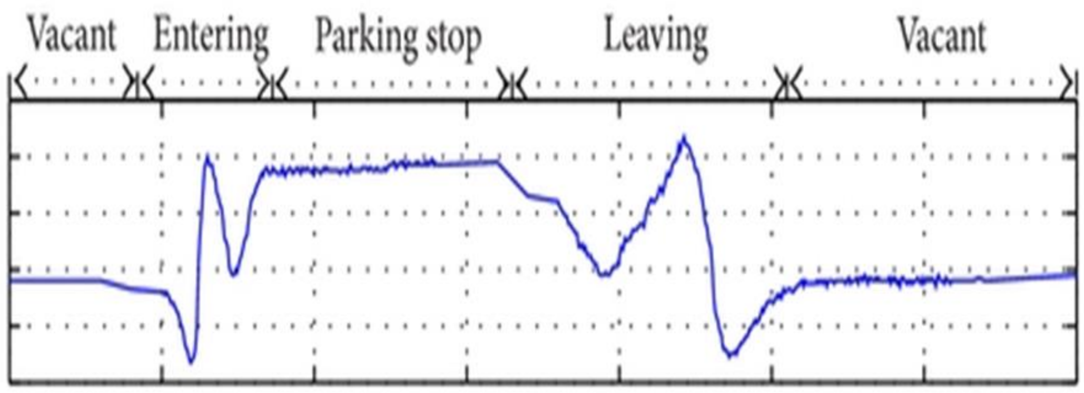


Figure 5: Four phases

When it is vacant phase, the magnetic field of a parking spot is the natural environment magnetic field because there is no car. When the parking spot is occupied, the magnetic field will be different. It will rise up to a higher level as the figure 5 shows. When a car is entering or leaving, the car will create a fluctuation of the magnetic field. So, we will know the exact time when a car is entering or leaving a parking spot.

Subsystem Analysis

Sensor Detection System

The Sensor detection system for park your car uses a magnetic sensor which are processed and input to the Arduino Uno Board. We notice the z-axis of senor detection is change dramatically when a car moves in or out. Thus, we set a range based on experiments and test on the road to let sensor show two statuses. The Arduino Uno Board is used to print out the values.

XBee Wireless Communication

When searching for a wireless solution to transmit sensor data between the Arduino board and IOS app, we decided on building a XBee network connection. The XBee router can be set up in the circuits with Arduino Board. It will simultaneous get data from sensor and transmit to the XBee gateway which acts as a coordinator. Once the gateway receive data, we write a python code to let it wirelessly upload to a device cloud by connecting with a hotspot. The IOS app has a synchronous function which access this data immediately.

User Interaction

Park Your car is designed for simple, easy-use and obvious user interaction. The Sensor Detection system is powered by battery and placed on the street near northeastern university campus. Once the system detects the change of parking position, the data will be uploaded. And user can login into the app to see data which are visualized as markers on the map view of app.

Hardware Components

Magnetic Sensor

After research and some comparison of different magnetic sensor, we decide to choose HMC5883L [4] magnetic sensor. HMC5883L magnetic sensor is used to make compass, so HMC5883L can detect magnetic field change in three directions. X, Y, Z directions are parallel to the surface of sensor while Z direction perpendicular to the surface of sensor. Due to sensors will always face to vehicles, Z direction data will the only data be used.

Figure 6: HMC5883L magnetic sensor

The HMC5883L utilize Anisotropic Magnetoresistive (AMR) of Honeywell [5]. AMR sensor can not only detect strength but also direction of a field if the field is dc static.

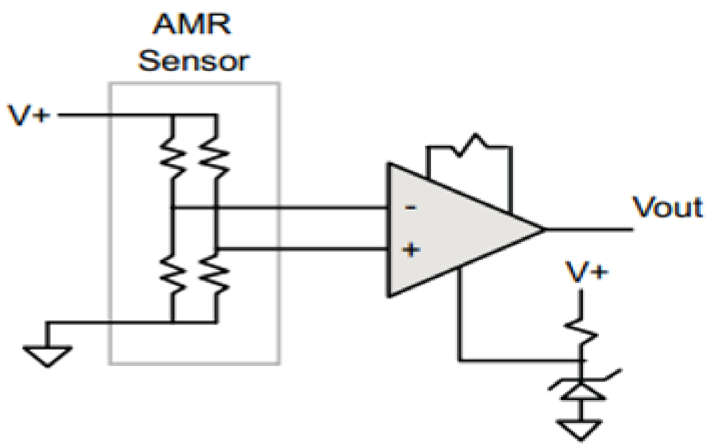


Figure 7: 2-AMR sensor circuit

As the figure 2 showed, AMR contains 4 resistors which are connected in a Wheatstone bridge configuration. The resistors are made of nickel-iron thin films deposited on silicon wafers, which cause resistance change by 2-3% in the magnetic field. The Wheatstone bridge makes it possible to measure both magnitude and direction of a field among a single axis. The AMR technique also allows HMC5883L combine with other circuit easily.

Arduino Uno

The Arduino Uno is used to control our magnetic sensor. As a most useful board, Arduino Uno has everything we need to achieve our goal. It is based on the Atmel ATmega328 8-bit Microcontroller, which is a low-powered and low-cost [6]. The analog pins and power pins let it can easily connect to our magnetic sensor and battery. Furthermore, the USB connection let us put code into microcontroller easily and can monitor data directly through Arduino IDE on the computer. By using Arduino IDE, we can write code to Arduino Uno. At first, we downloaded a code that can let Arduino Uno shows magnetic field data in all X, Y, Z three directions [7]. We built an array to store data of Z directions and let monitor screen of Arduino IDE show “0” for available and “1” for occupied by comparing data stored in the array.

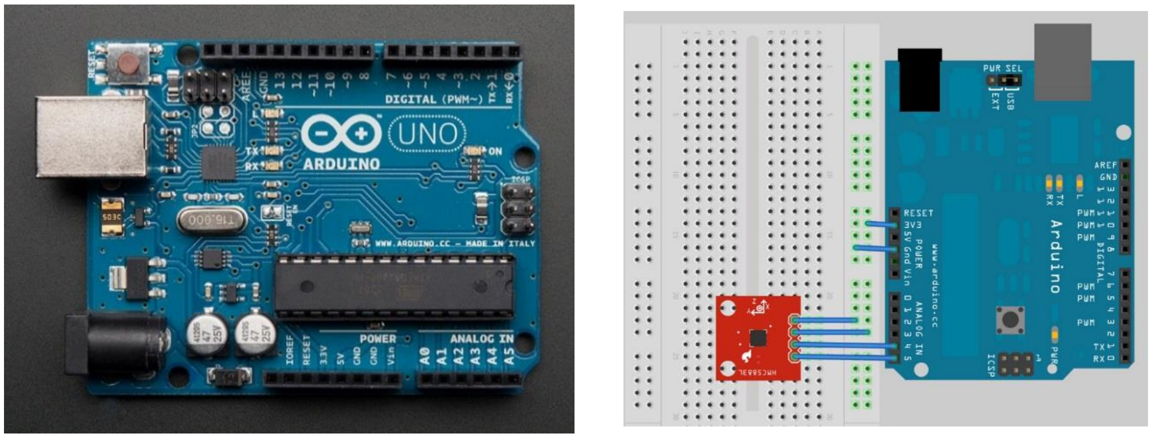


Figure 8: Arduino and Sensor Circuits

XBee-Local Data Transfer

The Arduino board has already collected data from the sensor now, we need a device to transmit data from the Arduino to online cloud. In our project, XBee is our choice.

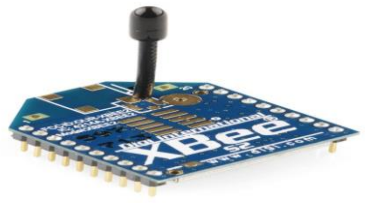


Figure 9: XBee

The XBee needs to be connected to a microcontroller so that the transmission can be done through it. So the Arduino board is the microcontroller, we wrote firmware code into the board and then control the XBee to transmit.

The XBee contains 20 pins, we need only four pins in this design. Pin 1 is used to connect to power supply, pin 2 is used to connect to converter to lower the voltage received from battery from 9V to 3.3V, pin 3 or DIN is the most important pin, because data enters the module from XBee through pin 3 as an asynchronous signal, and finally, pin 10 is connected to ground.

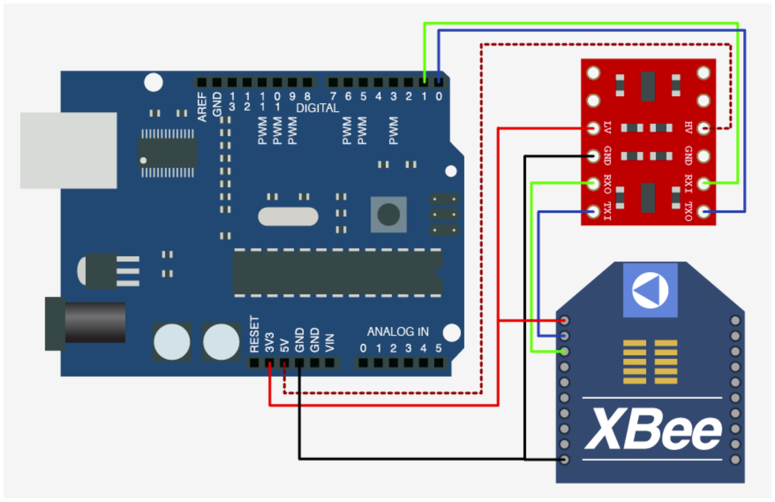


Figure 10: Connection Between Arduino, Converter (Red) and XBee

This is the connection we used in our design, the connecting wires on the converter can be soldered to make the connection stable and reliable. During the whole semester design, we tested a lot of times on this connection and we found that soldering is a must use technique because when the connections from converter to XBee unstable, the voltage from battery would directly go into the XBee, and XBee will be burned immediately. As the price of an XBee is about $20, being burned is obviously wasting of money. We saved lots of money after using soldering.

Gateway-Online Data Transfer

Now, the data should be transmitted to online cloud for our APP. We used Digi Gateway in this part of design.

Figure 11: Digi Gateway

This is a device that can automatically search all the XBees within its range once it is connected to WIFI. All you need to do is connect the gateway to a power supply, and search the signal of it on the computer, once you found the WIFI signal of the gateway, connect your computer to it and type in the configuration IP address in the web browser (Pretty much like configuring WIFI Router), then you will see all the available XBees listed in the webpage, we used Python to controlled the data type (whether we want a string or integer).

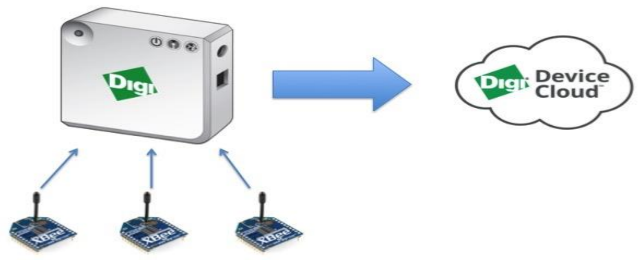


Figure 12: Working Principle

As shown in the figure, the digi gateway collects all the data from XBees and transmits the data to device cloud. Remember this process can only be done under WIFI environments. As mentioned, the device cloud receives all the data from gateway and stored the data for future uses. Digi gateway and Digi device cloud are from same company, and it’s very easy to connect them, just simply register at the device cloud and it will automatically receive data from the nearby gateways, because we only use one gateway, so our cloud only receives the data from one gateway.

The data we received is exactly the same as what we got from Arduino board, we just need to use Python to process the data and get the information we need. This is the end of the hardware design.

Power supply system

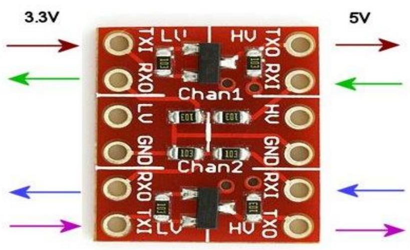


Figure 13: 9v Battery

Figure 14: Converter

The operating voltage for HMC5883L is 2.16 to 3.6V, while the XBee's voltage operation is 3.3V. So, the output pin on Arduino Uno we use is 3.3V. However, just HMC5883L works well for this. Some part of the circuit is still at 5V even if we use 3.3V. So, in order to let the XBee in a circuit work well, a logic level converter is needed [8].

By connecting the logic level converter between Arduino Uno and XBee, the voltage given to the XBee can be dropped from 5V to 3.3V. To power an Arduino Uno, for which the operating voltage is 7V to 12V, we simply use Duracell 9V batteries for our first generation. However, these kind of alkaline batteries just have about 600mah capacity, which cannot support our circuits for more than a day. The first solution we come out is using 6 AA batteries in series instead of Duracell 9V battery. One AA batteries have about 2000 mah capacity, which can support our circuit for 20 times longer. This is not a good way to solve the problem. First, 6 AA batteries force us to make the sensor node a lot bigger. Second, about 15 days is still not a long enough time range. XBee is the most important reason why our circuit consume so much power. The operating current for XBee is 45mA, which is very high. However, when XBee is in sleep mode, the operating current is less than 10uA, which means much less energy consumption. So, we need to use sleep mode of XBee to make sure our batteries can support our sensor nodes long enough. There is a sleep control pin on XBee just near the GND pin. XBees can be put into sleep mode after connect sleep control pin to Arduino UNOs. A code can be put into Arduino Uno to let XBees sleep when data stay the same (stay in "0" or stay in "1") and wake XBees up when data changes. By doing this, the system will still work well. At the meantime, circuits consume less energy and there will be less delay because much less data will be in the queue when go through Gateway to Device Cloud.

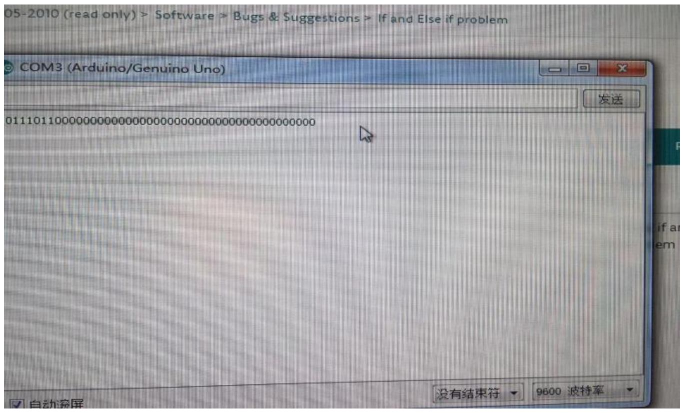
Software Components

Arduino Uno Code

The Arduino Uno code is broken into two segments, an initialization and a loop phase. In the initialization segment, the serial and I2C communication is be initialized [9]. The sensor HMC5883 IC is put into the correct operating mode and the communication between sensor and Arduino UNO board is open.

Then select mode register and continuous measurement mode to make sure the board is ready to listen data.

After the initialization finishes, the loop phase recursively tells the sensor to begin reading data from each axis, 2 registers per axis. The initial status of sensor is set as 0 which means there is an available spot. Then in next step, the new z-axis will be compared with the previous one. Once its change is greater than 30, it will converter the status from 0 to 1. It will continuously get data and change status if change exceeds the range. The delay we set is 3 seconds to be reasonable and operable. We examined many times and found that only the delay exceeds 3 seconds, the range can be significant, we also want to make sure the delay time is not too large.



05-2010 (read only) > Software > Bugs & Suggestions > If and Else if problem

COM3 (Arduino/Genuino Uno)

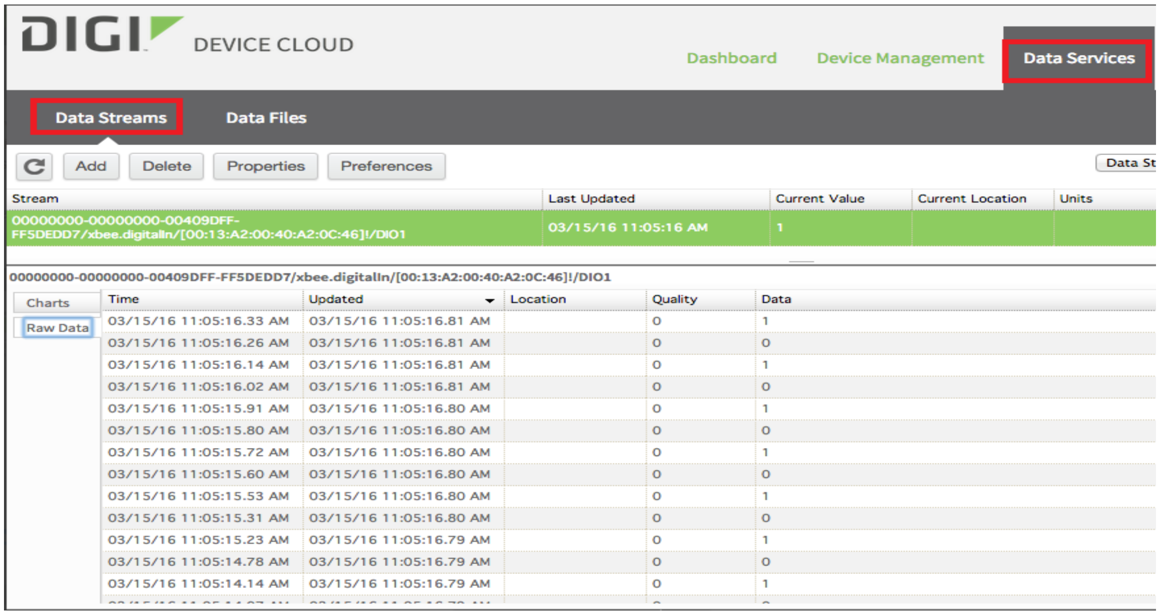
```
01110110000000000000000000000000000000000000000
```

Figure 15: Two status of Sensor

XBee Wireless Communication Code

The python code we wrote on gateway gathers all data in the interface of Gateway and then pushed it up to the Digi Device Cloud.

First, we define the data points which are the individual values stored at specific times. Then we use code to check platform of gateway, determine the XBee modem and building connection between gateway and device cloud. Then start a loop phase to listen the XBee data and upload to the cloud [10]. The data stream API in the device cloud allows data to be stored in a time series for long rime periods of time. The figure below shows the uploaded data which is the status of sensor.



Stream	Last Updated	Current Value	Current Location	Units
00000000-00000000-00409DFF-FF5DEDD7/xbee.digitalIn/[00:13:A2:00:40:A2:0C:46]!/DIO1	03/15/16 11:05:16 AM	1		

00000000-00000000-00409DFF-FF5DEDD7/xbee.digitalIn/[00:13:A2:00:40:A2:0C:46]!/DIO1

Time	Updated	Location	Quality	Data
03/15/16 11:05:16.33 AM	03/15/16 11:05:16.81 AM		0	1
03/15/16 11:05:16.26 AM	03/15/16 11:05:16.81 AM		0	0
03/15/16 11:05:16.14 AM	03/15/16 11:05:16.81 AM		0	1
03/15/16 11:05:16.02 AM	03/15/16 11:05:16.81 AM		0	0
03/15/16 11:05:15.91 AM	03/15/16 11:05:16.80 AM		0	1
03/15/16 11:05:15.80 AM	03/15/16 11:05:16.80 AM		0	0
03/15/16 11:05:15.72 AM	03/15/16 11:05:16.80 AM		0	1
03/15/16 11:05:15.60 AM	03/15/16 11:05:16.80 AM		0	0
03/15/16 11:05:15.53 AM	03/15/16 11:05:16.80 AM		0	1
03/15/16 11:05:15.31 AM	03/15/16 11:05:16.80 AM		0	0
03/15/16 11:05:15.23 AM	03/15/16 11:05:16.79 AM		0	1
03/15/16 11:05:14.78 AM	03/15/16 11:05:16.79 AM		0	0
03/15/16 11:05:14.14 AM	03/15/16 11:05:16.79 AM		0	1

Figure 16: Data Stream on Device Cloud

Application Code and functions

The user-facing segment of Park Your Car is an IOS application able to run on IPhone 6 or IPhone 6 plus. We use Swift to build functions and Story Board in Xcode to simulate the app.

The application is setup with our own designed registration and sign in system. User can create their own account, by typing their information.

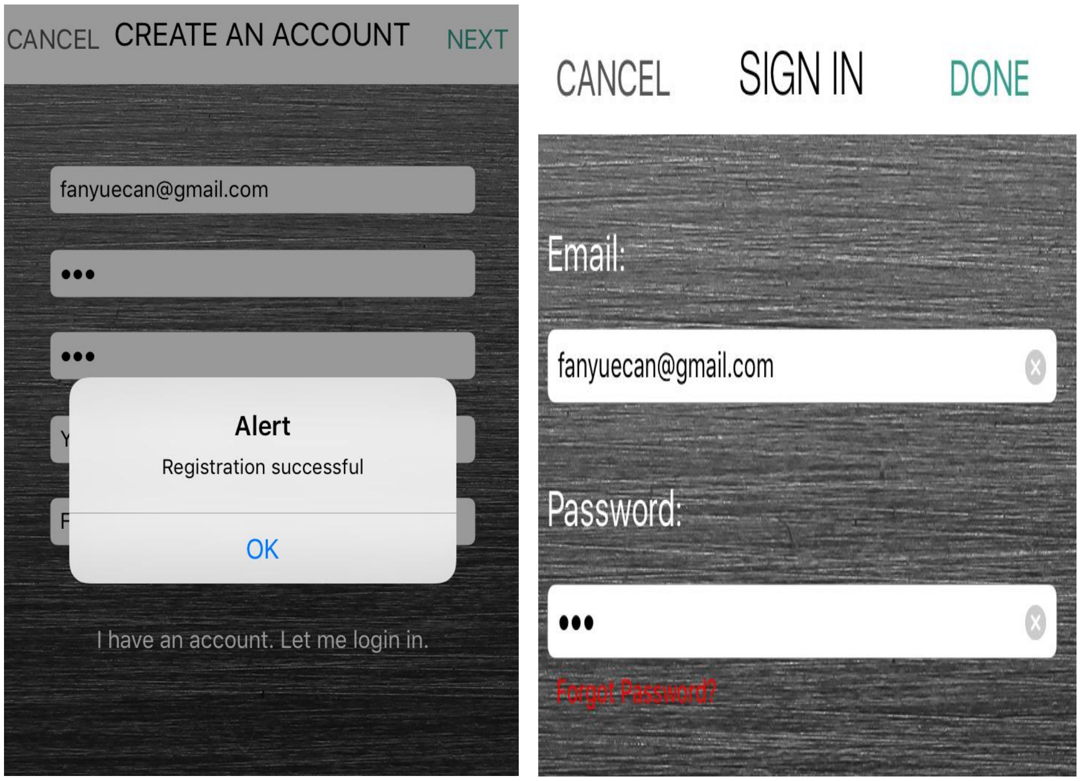
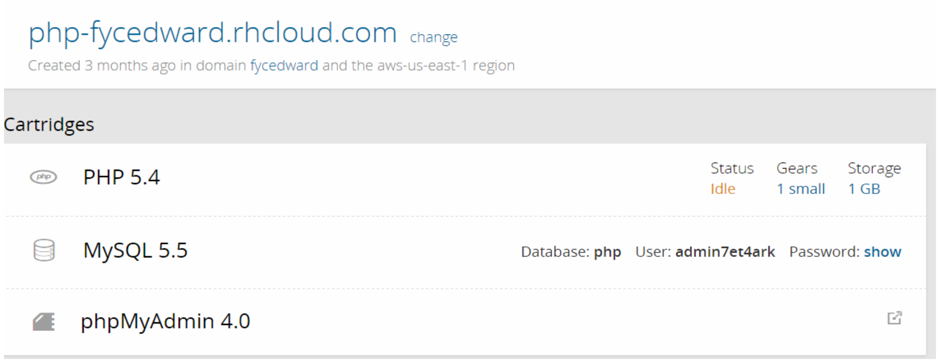


Figure 17: Registration System of App

Once they complete registration, they will get an email confirmation from our server. We use PHP to query the data from app and save to our online server-OPENSHIFT. The reason we chose OPENSHIFT is because it contains many database applications which can help us easily do data management.


php-fycedward.rhcloud.com change

Created 3 months ago in domain fycedward and the aws-us-east-1 region

Cartridges

	Status	Gears	Storage
PHP 5.4	Idle	1 small	1 GB

MySQL 5.5 — Database: php User: admin7et4ark Password: show

phpMyAdmin 4.0

Figure 18: Online Sever OPENSHIFT

The main function we build is use google map API to get user current user location and make real time data showing on the map. We use different color marker to illustrate the available spot. If there is a car move in, the marker will immediately disappear.

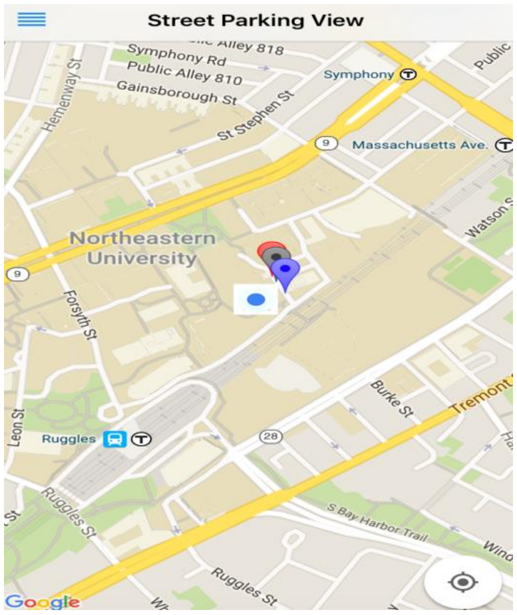
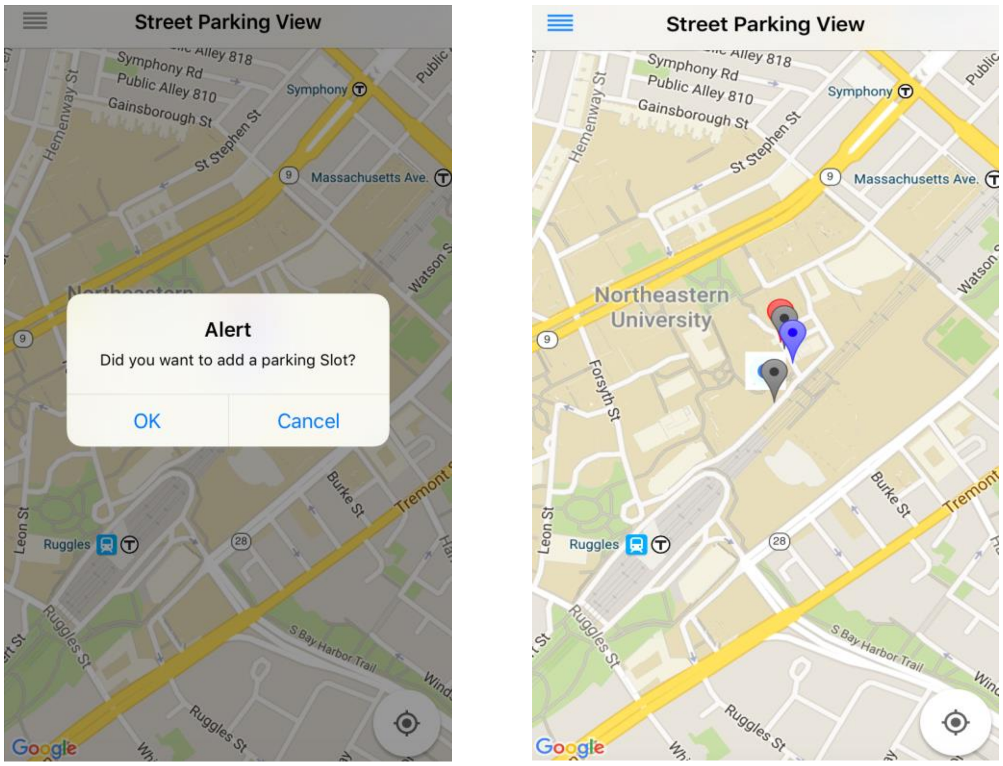


Figure 19: Map view of App [11]

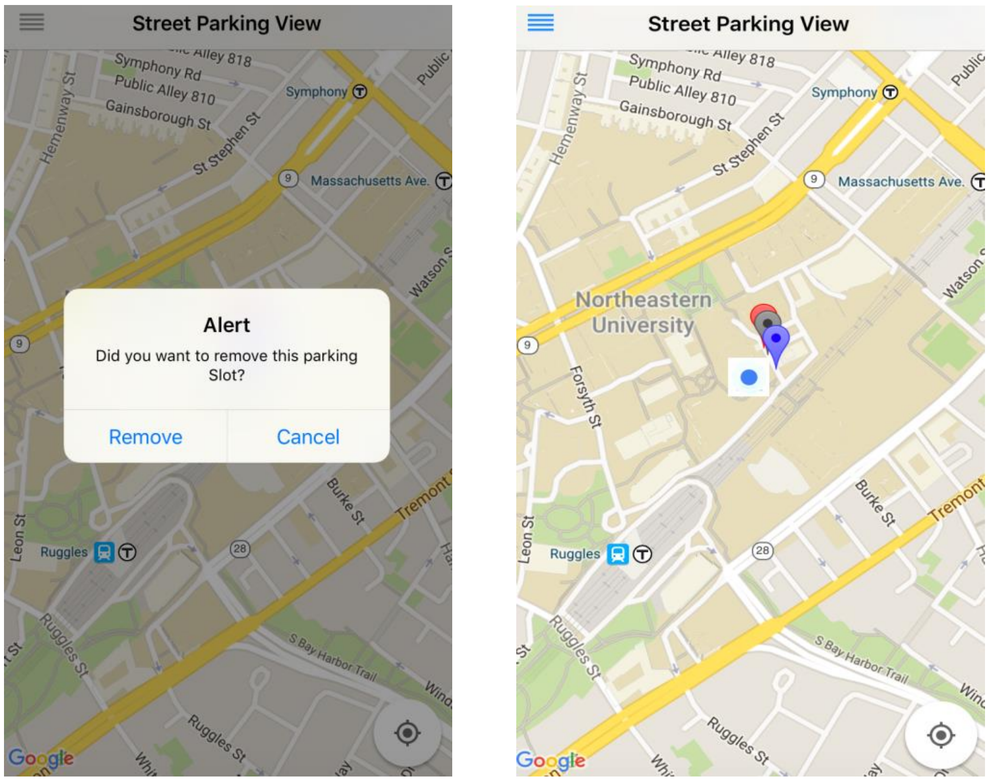
We also add a user self-report function to let user to save or delete an available spot. All data will be saved into our database.



Alert

Did you want to add a parking Slot?

Figure 20: Add a Parking Slot



Alert

Did you want to remove this parking Slot?

Figure 21: Delete a parking slot

To make our app more user friendly, we design our own icon and pictures in Photoshop, change the default background and adjust the button location.

Cost Analysis

Component	**Cost**
Arduino Unos	$17.97
XBees – Series 2	$68.85
Logic Level Converters	$8.85
Batteries	$17.97
Battery Connectors	$5.97
HMC5883L Magnetic sensors	$44.85
Wires	$23.97
Gateway	$120.00
Mini Boards	$11.99
Laser-cutting Boxes	$34.50
Total cost of products	$357.92

Figure 22: Cost Analysis

The total cost of our project came to about $360.00. Due to we build three sensor node, so actually each one is only about $80. And the Gateway which cost $120 can serve for many sensor nodes simultaneously. We ended up spending about all of our $1200 budget on everything we bought. Here are the reasons. We first tried to build our own PCB board so we bought some chip like HMC5883L-TR, CC1000 and ATmega128-8AU. However, we found Arduino Uno is very cheap and easy to use, so we simply choose to use Arduino Uno. We were using XBee – series 1 at the beginning. However, when came to the step for uploading data to Device Cloud, we found only XBee – series 2 have this function. So, we bought some XBee – series 2 instead of XBee – series 1. We also destroyed some XBees, sensors during testing and soldering because of the high voltage and bad soldering skills. The shipping fee also cost us a lot, the majority of our component needed to be bought online. Therefore, in order to let our product finished on time, we often need to choose faster and more expensive shipping method.

The cost of one sensor node can be reduced if we build our own PCB board. According to the research, we can spend no more than $15 to buy a HMC5883L-TR and a CC1000 to instead HMC5883L and XBee, which cost about $40 in total. So, we are confident that our product can be sold at a reasonable price.

Final Product

Our final product is a plastic glass box contains all hardware components. Each box can be installed on the road and use XBee to transmit collected data to a gateway nearby.

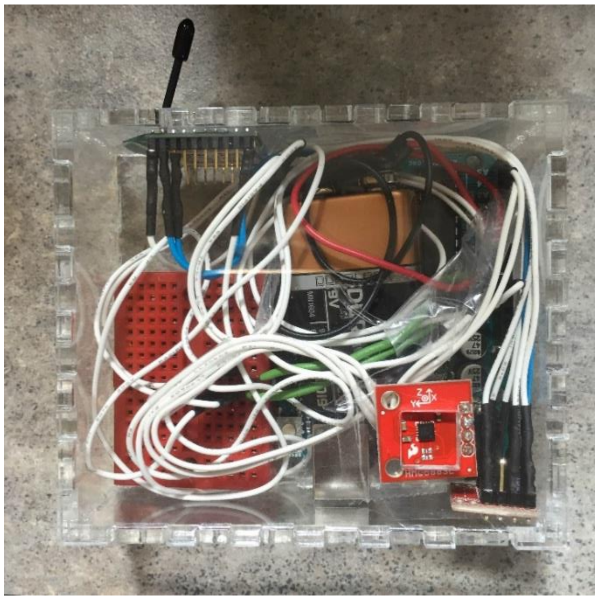


Figure 23: Final Product

Conclusion

This Capstone project taught us a lot about how to implement every possible method to develop our final product step by step and overcome all incoming problems. Each group member has their own strength and we learn from each other during the collaboration. The timeline is an important thing for us, team leader makes sure each group member completes the assignment and discuss to solve together if the problem if too hard for him.

During fall semester, we tried to learn new programming languages and purchase the materials to get known of each components.

In the spring semester, we spent a lot of time trying to build the hardware circuits and develop each functions in the app. The most problem we faced before is how to solve the wireless uploaded problem and make app get immediately. We search everything on the website, figure out the gateway can help and use Swift code which we never learn before. We debug many times and looked through some instructions or some videos recorded by other people. We figure out the solution on time and make our final product showing on the competition day. We are very proud of making brand new product which can help people solve real time problems.

Sources:

[1] http://park.boston.gov/
[2] http://inrix.com/products/parking/
[3] http://readwrite.com/2014/06/21/sweetch-parking-mission-san-francisco

[4] http://www.farnell.com/datasheets/1769350.pdf

[5]http://107.23.57.149//~/media/Images/Plymouth%20Website%20PDFs/Magnetic%20Sensors/Technical%20Articles/Vehicle_Detection_and_Compass_Applications_using_AMR_Magnetic_Sensors.ashx

[6] https://www.arduino.cc/en/Main/ArduinoBoardUno

[7] https://www.sparkfun.com/tutorials/301

[8] http://bildr.org/2011/04/arduino-XBee-wireless/

[9] https://www.sparkfun.com/tutorials/301

[10] https://www.digi.com/wiki/developer/index.php/XBee_to_Device_Cloud_-_DataPoint_Creation

[11] https://developers.google.com/maps/documentation/ios-sdk/