Parking Reservation System

ECE 445 Design Document - Spring 2019
Team 59
Ojus Deshmukh, Manjesh Mogallapalli, Vivek Calambur
TA: Kyle Michal
1. Introduction  
   1.1 Objective  
   1.2 Background  
   1.3 High Level Requirements  
2. Design  
   2.1 Block Diagram  
   2.2 Physical Design  
   2.3 Block Design  
      2.3.1 Power Supply  
      2.3.2 Parking Meter  
      2.3.3 WiFi Hub  
      2.3.4 Software  
   2.4 Schematic  
   2.5 Tolerance Analysis  
      2.5.1 Risk Analysis Overview  
      2.5.2 Ultrasonic Sensor Analysis  
3. Cost and Schedule  
   3.1 Cost Analysis  
      3.1.1 Labor  
      3.1.2 Parts  
   3.2 Schedule  
4. Safety and Ethics  
   4.1 Physical Safety  
   4.2 Data Privacy  
5. References
1. Introduction

1.1 Objective

We have all been in situations where we find ourselves driving aimlessly looking for public parking or having to park out of the way because no spots are available. This problem gets worse as cities grow—drivers in London spend an average of almost eight minutes looking for parking every day[1]. In the US, after accounting for time, fuel, and emissions, drivers lose over $70 billion[2] annually while looking for parking. Additionally, users often overpay for their parking spot. The average driver in the US spends “eight times more a year overpaying for parking than they do in parking tickets”[2]. These issues stem from not having a comprehensive system for parking that allows a user to effectively find available spots or to know that a spot will be saved for them on arrival.

Our solution is a comprehensive parking system that allows users to easily gauge the parking availability in a certain area before they start driving or simply reserve a spot ahead of time for a set window. Instead of having to deal with coin payment, garage systems, or blocked-access parking lots, the system will rely entirely on a parking meter and a companion app. Users will also have a choice to receive an RFID tag they can use to streamline their parking process.

Users will have two choices:

1) They can reserve a location ahead of time for a set window. On the day of their reservation, they can arrive at the lot, knowing they will have prepaid parking for the duration that they have selected.

2) They can also use the mobile app to find open parking spots. This functionality preserves the use of open, pay as you go parking locations, except it adds two additional features. Users can find parking spots quickly and easily on the map, thereby reducing time they spend driving while looking for a spot. Additionally, users will only be charged when they leave, preventing users from either overpaying for the spot or having to return to the spot to feed more time into the machine.

1.2 Background

Some parking reservation solutions exist, such as SpotHero[3], where a user can prepay to park at a parking garage of their choice. Users, however, don’t know exactly which parking spot they will be parking in and sometimes drive through multiple floors before finding an open spot. Users might also accidentally park in a spot that is reserved for other purposes, as is the case with garages that are shared with businesses, and end up getting towed.
In addition, there are some public garages have signs that display parking availability. Again, in this case, users do not know exactly where the open parking spots are, so users may waste time searching for them. Public parking garages generally have central payment points where users pay before they return to their cars, or users pay as they exit. Both of these methods result in time wasted and an overall inefficient payment process.

Our solution is efficient, as we assign spots. This results in less confusion and wasted time and effort. It also gives users the peace of mind knowing that they are parking in a legal spot from which they won't get towed. Finally, our reservation and meter system allow users to make payments painlessly, thereby improving the overall parking process significantly.

1.3 High Level Requirements

1. The meter will be able to verify users via RFID tags.
2. The hub unit will be able to support at least 10 active meters
3. The system will identify empty parking spaces with 90% accuracy
2. Design

2.1 Block Diagram

![Block Diagram Image]

Figure 1: Block Diagram
2.2 Physical Design

![Physical Diagram](image)

Figure 2 - Physical Diagram

2.3 Block Design

2.3.1 Power Supply

This subsystem will power all of our components with the appropriate voltage to function properly.

**Mains Supply**
For the scope of this class, our main power source will be a wall outlet. In real world deployment, this system will likely be given electrical wiring from power lines.

**Voltage Regulator**
The purpose of the regulators is to alter the voltage from the main supply to the desired voltage for the individual components.

**Parts:**
- Wall Adapter, 5V 2A DC: [https://www.sparkfun.com/products/12889](https://www.sparkfun.com/products/12889)
- Voltage Regulator, 3.3V 800mA: [https://www.sparkfun.com/products/526](https://www.sparkfun.com/products/526)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Output 5V(+/- 5% to maintain accuracy), 10A</td>
<td>1. Use a multimeter to measure the open circuit voltage</td>
</tr>
<tr>
<td>2. Output 3.3V(+/- 5% to maintain accuracy), 10A</td>
<td>1. Use a multimeter to measure the open circuit voltage</td>
</tr>
</tbody>
</table>
2.3.2 Parking Meter

The parking meter contains a microcontroller, WiFi module, RFID module, proximity sensor, and LCD screen, and is the main hardware component in this system.

**Controller**

We will be using an ATMEGA328 microcontroller to control all of the components onboard the meter, along with a 16MHz crystal resonator to improve clock speed.

**Parts:**


16MHz Resonator: [https://www.sparkfun.com/products/9420](https://www.sparkfun.com/products/9420)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The microcontroller can send/receive signals through UART at a speed of 115.2 kbps</td>
<td>1. Connect microcontroller to USB UART bridge, such as FT4222 or FT232, and to a terminal such as Putty 2. Set up terminal at 115.2kbaud 3. Send and echo back 100 characters 4. Ensure that all characters match those sent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. The resonator must be functioning at 16MHz</td>
<td>1. Connect an oscilloscope to measure the frequency of the resonator</td>
</tr>
</tbody>
</table>

**WiFi Module**

We will use a ESP8266 Wifi module for our project. With it, we can connect the meter to the same network as the Hub unit to transmit/receive data.

**Parts:**

ESP8266 WiFi Module: [https://www.sparkfun.com/products/13678](https://www.sparkfun.com/products/13678)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Can connect to same network as Hub</td>
<td>1. Have specific meters turn on and connect WiFi 2. Check active devices on Hub unit to ensure that the correct meters are connected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Can send/receive signals through WiFi</td>
<td>1. Connect specific meter to WiFi 2. Send command through backend to display “TEST” and green color on LCD display 3. Confirm that “TEST” is printed in correct color on LCD display.</td>
</tr>
</tbody>
</table>
RFID Module

We will be using an RFID Reader accompanied with RFID tags that come pre-programmed with a unique 32-bit ID for user verification.

Parts:
- RFID Reader(ID-12LA) : [https://www.sparkfun.com/products/11827](https://www.sparkfun.com/products/11827)
- RFID Button(32-bit, 125 kHz) : [https://www.sparkfun.com/products/9417](https://www.sparkfun.com/products/9417)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Can read 125 kHz RFID signal and output to processor</td>
<td>1. Scan multiple 125 kHz RFID buttons with reader 2. Ensure correct RFID code is read by reader by having test program print the code that was read for each button 3. Scan buttons again to ensure same code is repeated</td>
</tr>
</tbody>
</table>

Proximity Sensor

We will be using an Ultrasound Proximity Sensor(HC-SR04) on each parking meter to know whether a car is currently parked in its corresponding parking space based on the sensor’s output.


<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Accurately output whether the proximity sensor can detect a large object in the bounds of a parking spot.</td>
<td>1. Use a large cardboard box to simulate a vehicle 2. Measure out a “standard” parking spot 3. Test that proximity sensor detects box at the extremes of the spot (near, far, left, right) 4. Test that proximity sensor does not detect box outside bounds 5. Repeat 10 times to ensure accuracy &gt; 90%</td>
</tr>
</tbody>
</table>
**LCD Screen**

The LCD screen’s purpose to display pertinent information to the user regarding the reservation such as time left and whether the user has been verified.

**Parts:**

RGB Character LCD(16x2):


<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. System is powered with +5V(+/− 5%) and can draw enough current to power RGB backlight and change text</td>
<td>1. Switch screen color to Red  2. Switch screen color to Green  3. Switch screen color to Blue</td>
</tr>
<tr>
<td>2. Display relevant information to the user through color and text</td>
<td>1. Instruct screen to display specific text and set to specific backlight color  2. Ensure text/color is correct</td>
</tr>
</tbody>
</table>

**2.3.3 WiFi Hub**

The WiFi Hub acts as a wireless access point for the system, allowing the meters in its range to connect to the backend.

**Raspberry Pi**

The raspberry Pi has its own built-in power regulator, so it will be connected directly to mains supply.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Allow up to 10 devices up to 100 feet away to connect to the RPi access point</td>
<td>1. Connect 10 devices to the access point and ensure that connection does not drop  2. Test connection strength at multiple points along a 100 foot radius from the RPi  3. Ensure that signal strength is &gt; -60dBm</td>
</tr>
<tr>
<td>2. Connect to the backend through Wi-Fi or ethernet</td>
<td>1. Run a simple test script that reads and writes data to/from the database  2. Verify that correct data is sent/received</td>
</tr>
</tbody>
</table>
### 2.3.4 Software

The software subsystem is composed of two main components, the mobile application and the backend. The mobile app is the primary user interface for this system. With this, users can interact with the system and get their parking information anywhere. The backend is the location of all parking, reservation, and account data and the source of all commands to the parking meters. From here, the entire system can be controlled remotely.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1. Users will be able to make/view reservations | 1. Check backend database to ensure that reservation has been made.  
2. Check to see that application has the ability to poll database upon request |
| 2. Users will be able to check in/out to meters  | 1. Check database to see if RFID check-out/check-in is reflected in system  |
| 3. Take in and store reservations for the next day | 1. Send in reservations from mobile app  
2. Check backend database to ensure that reservation is stored correctly  
3. Stress the system by making reservations from multiple accounts in varying times |
| 4. Combine reservations to efficiently use as few spots as possible | 1. Write algorithm that takes in windows of time(reservations) and combines efficiently into fewer spots  
2. Manually arrange reservations to ensure that algorithm is finding most efficient solution |
| 5. Remotely authorize and charge users           | 1. Scan incorrect RFID and ensure that system reacts appropriately  
2. Scan correct RFID and ensure that system acts appropriately  
3. Ensure that system can track total spot usage time correctly |
| 6. Remotely send commands to the parking meters | 1. Send text and backlight info to meters and ensure that it’s set correctly |
Upon new data being encountered by either the RFID or Ultrasonic sensor the WiFi chip activates and sends the new data to the Raspberry Pi hub unit. The hub unit will then send and store the data in the database. Data can be accessed from the database for the spot assignment algorithm, data population for the mobile application, or to verify users. Upon any modifications/interactions from either the back end or by the user through the mobile application the process is reversed. And the appropriate data is sent to the LED display.

The receiving and sending of data from the sensors will all be done using Arduino Code (C++). The data transfer between the raspberry PI to AWS Database and data transfer between AWS Database to backend processes and mobile application will be done via Python/Flask. We believe that an AWS Database is the best option to house our data because it is the industry standard. The mobile application will be built for Android phones using Java.
Users can reserve parking in a garage/lot the day before at the earliest and up to 1 hour before a reservation at the latest. Upon the reservation option being active a user can pick a garage/lot they would like to park at and specify a time range from which they would like to park the car. Here the algorithm adds a 15 minute buffer to start and end times to give enough time for users to exit parking spaces. Then the “buffered” reservation is added to a spot with a non-conflicting reservation. Essentially, packing all reservations into one spot, to ensure maximum lot availability. If the reservation cannot be added to any existing reserved spots it is assigned a new spot. The algorithm will move reservations around spots to get an optimal assignment. The user will be notified one hour before their arrival of their specific spot details.

2.4 Schematic

2.5 Tolerance Analysis

2.5.1 Risk Analysis Overview
The biggest risk to getting our project to meet our high level requirements is the Parking Meter Subsystem. We will have to design a PCB that contains the Proximity Sensor, RFID Module, WiFi Module, LCD Screen, and Microcontroller. The WiFi module will need to communicate with each of the other
components to send data to and communicate data from the backend. The fact that we have never worked with these parts before will present a challenge within itself before we work on modifying them to meet our requirements. We have to be able to work with the RFID Module to be able to accurately verify user parking. This means getting the module to look for the appropriate user RFID code and processing it correctly to ensure a smooth reservation process. The RFID module is an essential component to our project and without it working we cannot verify reservations for our system.

In addition, we have to calibrate the Proximity Sensor such that it gives us an accurate reading as to whether a car is in present in the respective parking space. Given that the Proximity Sensor can read up to range of 4m we will have to make sure that the sensor is not picking up the presence of cars in neighboring parking spaces as well as ensuring the sensor is not reading any passing cars, humans, etc.

2.5.2 Ultrasonic Sensor Analysis

For the success of this project, it is imperative that our ultrasonic sensor works to the level of precision and accuracy that we have projected. An ultrasonic sensor that generates too many false positives will mark open spots as taken and not accurately display spot availability to the user. On the other hand, a false negative will also impact accuracy of spot availability and could allow a user to park without paying the required amount for the spot.

Specifications of a “standard” parking spot in the United States [8]

1) Width: 7ft - 9ft = 213cm - 274cm
2) Length: 16ft - 20ft = 488cm - 610cm

First, we need to analyze whether the measurement angle of the ultrasonic sensor will be restricted to a single parking spot at the lower width bound of a parking spot (in this case, 213cm). From the datasheet for the HC-SR04, we see that the effectual measuring angle is 15 degrees. With a few simple calculations, we can calculate the maximum depth that we can measure vehicle presence at:
370cm = ~12ft, which is much greater than the depth necessary to detect the presence of a vehicle. This confirms that we would still accurately be able to pick up parked cars if they are parked relatively far away from the curb.

The ultrasonic sensor works by sending out an ultrasound wave, and measuring how long it takes for the wave to return to the sensor. Sound travels at approximately 340 m/s. This corresponds to about 29.412\(\mu\)s per centimeter. To measure the distance the sound has travelled we use the formula:

\[
\text{Distance} = \frac{(\text{Time} \times 340 \text{ m/s})}{2}. 
\]

As seen from figure 4, the measuring error is inversely proportional to the size of the object. In this scenario, we are only interested in measurements of large objects (i.e. cars, motorcycles etc.). In the figure below, we can follow the distance error of the light blue line as a large object is moved from 20cm away to 180cm away. Here we see that object presence is still strongly detected as far away as 180cm (6ft), which is more than enough depth for us to work with.
On the other end of the spectrum, we are not concerned about the sensor handling objects that are too close to it. According to the HC-SR04 datasheet, it can measure objects at distances as close as 2cm away, at which point we would probably be more worried about a collision.

### 3. Cost and Schedule

#### 3.1 Cost Analysis

##### 3.1.1 Labor

Source: ECE Illinois website [6]

<table>
<thead>
<tr>
<th>Name</th>
<th>Hourly Rate</th>
<th>Hours</th>
<th>Cost * 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manjesh Mogallapalli</td>
<td>40.50</td>
<td>180</td>
<td>18,225</td>
</tr>
<tr>
<td>Ojus Deshmukh</td>
<td>40.50</td>
<td>180</td>
<td>18,225</td>
</tr>
<tr>
<td>Vivek Calambur</td>
<td>40.50</td>
<td>180</td>
<td>18,225</td>
</tr>
</tbody>
</table>

Total Labor = $54,675

##### 3.1.2 Parts

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATmega328 microcontroller (x3)</td>
<td>$14.49</td>
</tr>
<tr>
<td>WiFi Module (x3)</td>
<td>$20.85</td>
</tr>
<tr>
<td>RFID Reader (x3)</td>
<td>$89.85</td>
</tr>
<tr>
<td>RFID Buttons (x6)</td>
<td>$23.70</td>
</tr>
<tr>
<td>Ultrasonic Sensor (x3)</td>
<td>$11.85</td>
</tr>
<tr>
<td>LCD Display (x3)</td>
<td>$60.15</td>
</tr>
<tr>
<td>Raspberry Pi (x1)</td>
<td>$38.30</td>
</tr>
<tr>
<td>Voltage Regulator(3.3V)</td>
<td>$1.95</td>
</tr>
<tr>
<td>Wall Adapter(5V, AC DC Converter)</td>
<td>$5.95</td>
</tr>
</tbody>
</table>
### 3.2 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Manjesh</th>
<th>Vivek</th>
<th>Ojus</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/4/19</td>
<td>Begin the framework for the mobile application.</td>
<td>Set up AWS environment and write the framework for DB connection</td>
<td>Begin testing and verification for all components.</td>
</tr>
<tr>
<td>3/11/19</td>
<td>Finish writing mobile application. Help Ojus with PCB Design</td>
<td>Begin writing firmware code for data transmission to/from PCB.</td>
<td>Complete verification of all components. Design and order PCB.</td>
</tr>
<tr>
<td>3/18/19 (Spring Break)</td>
<td>Perform initial testing on mobile application. Comment and clean up existing code.</td>
<td>Perform initial testing on firmware and backend connection. Comment and clean up existing code.</td>
<td>Collaborate with others to ensure software is ready for integration.</td>
</tr>
<tr>
<td>3/25/19</td>
<td>Work with Vivek on assembling physical and electrical components. Help Ojus with PCB testing.</td>
<td>Assemble all physical components and electrical components.</td>
<td>Ensure ordered PCBs meet functionality requirements and order additional PCB if necessary.</td>
</tr>
<tr>
<td>4/1/19</td>
<td>Help Ojus with physical and electrical component integration. Integrate software with all hardware components.</td>
<td>Complete the final firmware code for the final PCB design.</td>
<td>Complete physical and electrical components integration.</td>
</tr>
<tr>
<td>4/8/19</td>
<td>Begin final testing of the complete system. Focus on issues regarding mobile application.</td>
<td>Begin final testing of the complete system. Focus on issues regarding Firmware.</td>
<td>Begin final testing of the complete system. Focus on issues regarding PCB performance.</td>
</tr>
<tr>
<td>4/15/19</td>
<td>Fix all mobile application issues.</td>
<td>Fix all Firmware issues. Ensure that all</td>
<td>Fix all PCB issues. Ensure that all physical</td>
</tr>
</tbody>
</table>
4. Safety and Ethics

4.1 Physical Safety

As stated in the IEEE Code of Ethics[3] #1, our first goal is to ensure the health and welfare of the public. First and foremost, we will need to address physical safety concerns for our system. As our meters and hub unit will be placed in outdoor locations, they will need to be resistant to weather elements. We will place all components in a waterproof case that adheres to IP67 standards[4]. This ensures that our meter and hub units are protected from both dust and water in outdoor use cases. Additionally, another concern that we will need to address is the possibility of mobile phone usage while driving. To combat this, we will have a few different safety features. First, a user who has reserved a spot will be assigned a spot number well ahead of their arrival time, allowing users to ensure that they have all the information they need before they get behind the wheel of a vehicle. Second, much like what navigation apps like Waze do, we will display a safety warning to drivers. While this doesn’t fully prevent drivers from using the app when on the road, it is as much as can be reasonably done to deter users from distracted driving.

4.2 Data Privacy

Another issue that is paramount in today’s environment is data privacy and security. As stated in the ACM Code of Ethics[7] 1.6, we aim to respect privacy. When a user provides us with their information, they trust that we make good decisions. This trust is not easily given, but very easily broken. In order for this system to be successful, we will need to collect a significant amount of user data. This may include some personally identifiable information (PII) and sensitive payment information. All user data will be encrypted before storage in our databases. We will be using a service such as AWS or GCP to store and
process our data, and trust that these services will adequately protect our data from external attacks. Additionally, we will utilize retention limits, and fully anonymize parking location and transaction data once a certain time period has passed.

In addition to the issues addressed above, we pledge to honor and abide by the standards set by the IEEE and ACM. We will follow all laws and safety regulations set by the course as well as the campus as a whole.

5. References


