# Campus Parking Availability System: Final Report

Patrick Elam, Jacob Lambert, David Lu, Ben Mills, Nathan Peck, Andrew Stubbs
University of Tennessee, Knoxville
COSC 402 – Senior Design
Team 27

April 24, 2015

Customer: Dr. Mark Dean

# Table of Contents

# I. Executive Summary

Commuter parking at the University of Tennessee, Knoxville can be a frustrating experience, due especially to the imbalance between commuter students and commuter parking spots. There are considerably more commuter passes than there are commuter spots, thus requiring most students to arrive on campus much earlier and causing large amounts of traffic during class transitions. Currently, there are very few resources to help alleviate this problem, thus a solution would be highly desirable by UT students. The Campus Parking Availability System (CPAS) aims to relay data to students, showing the occupancy of each parking lot. The system provides a website interface for the student, who can make decisions for parking based on the available data.

The design of the system consists of three different sections: on-site, off-site, and the website interface. The on-site system consists of a pair of ultrasonic systems at each entrance/exit of a parking lot. These sensors are connected to a Raspberry Pi, a small linux-capable single board computer, with Wi-Fi capabilities. The sensors monitor entering and exiting vehicles to maintain a current count, which the Raspberry Pi will send to a central server every minute. The off-site system is composed of the central server and the database. The central server receives data from every Raspberry Pi in the system and stores the information in the database. The server is also responsible for establishing and maintaining a connection with each on-site system. The website interface accesses the database to display the most current information from the system to the user. The website is configured to be convenient for desktop, mobile, or tablet device usage. Graphs and analytics are then performed on the historic data to provide trendlines and predictive analytics.

The biggest challenge of the project was discovering and fixing the physical limitations of our ultrasonic sensors. Our initial set of sensors functioned differently than what was expected, which required some research to find suitable replacements. After installing these sensors in the prototype location, the readings revealed interference between the two ultrasonic pulses. This problem was removed by controlling when each sensor took a reading, and thus synchronizing the readings such that neither sensor fired while the other was reading. Overall, CPAS has met a majority of the proposed requirements and goals that were created at the start of the Fall 2014 semester.

# II. Requirements

## 1. On-Site Systems

**1.1** Each tracked parking lot will have an on-site system.

**1.2** Each system will monitor the entrance/exits of the parking garage.

- **1.2.1** Systems will be capable of detecting 4-wheeled motor vehicles.
- **1.2.2** Systems can identify whether vehicles are entering or exiting the garage.
- **1.2.3** Systems will operate in conditions from 0 - 120° Fahrenheit.
- **1.2.4** Systems will use a single control unit located with the vehicle detection sensors, forming the "on-site system."
- **1.2.5** Systems will be weatherproof.

**1.3** The systems will communicate garage occupancy data to the central system.

- **1.3.1** The systems will record data received from vehicle-detecting apparatus.
- **1.3.2** Data will be timestamped before it is transmitted to the server.

**1.4** The systems will be capable of connecting to a UTK Wi-Fi network.

**1.5** The systems will require an external 110 Vac power source.

**1.6** The systems will have the ability to store data locally when there is no Wi-Fi connection.

**1.7** The systems will run data processing software.

- **1.7.1** Each system will mark its data, allowing the server to identify a unique garage.
- **1.7.2** The data will be paired with a time stamp.

## 2. Central System

**2.1** The central system consists of:

- **2.1.1** a main server.
- **2.1.2** a web server.
- **2.1.3** a database.
- **2.1.4** analytics software.

**2.2** The main server will:

- **2.2.1** receive formatted data from each on-site system.
- **2.2.2** link each component of the central system.

**2.3** The system will host a website.

- **2.3.1** The website will display the following data:
  - **2.3.1.1** total number of parking spaces
  - **2.3.1.2** number of available parking spaces
  - **2.3.1.3** percentage of parking spaces available
  - **2.3.1.4** rate of change in available parking spaces (spaces/hour) for each parking lot
- **2.3.2** A user will be able to access parking data for CPAS-equipped lots within three interactions after opening the website.

**2.3.3** The website will be viewable from any device with a web browser.

**2.4** The system will maintain a database for the collected data.

**2.4.1** The data will consist of a timestamp and an integer representing the occupancy of the garage.

**2.5** The system will perform analytics on the collected data.

**2.5.1** The data will be analyzed to create graphs for the website, including:

**2.5.1.1** Time series graph of occupancy

**2.5.1.2** Basic forecasting

**2.5.2** The data will not be skewed by outlier scenarios (Sporting events, on-campus concerts, etc.)

**2.5.3** The data will be analyzed in Monday-Wednesday-Friday, Tuesday-Thursday, and weekend groups, corresponding with standard class schedules.

## 3. General Requirements

**3.1** The system will relay parking space occupancy information to the servers within 90 seconds.

**3.2** The system will accurately measure the occupancy of each parking lot with a margin of error of 10% of maximum lot capacity.

**3.3** The system, in all phases of implementation, will obey all parking regulations established by Parking and Transit Services of the University of Tennessee, stated at http://parking.utk.edu/regulations/.

## 4. Report Requirements

**4.1** The report must include a section that describes how to run the on site system off of an alternative power source.

**4.1.2** Must provide calculations for how long the system will run off of the alternative system that we choose.

**4.1.3** Alternative power source must include battery support.

**4.2** The report will address the issue of users accessing the website while simultaneously driving a motor vehicle.

## III. Change Log

- Removal of Android/iPhone mobile application requirement
- Relaxation of accuracy requirements for lot occupancy measurements
- Addition of necessary power supply requirement
- Simplification of analytics requirement

# IV. Documentation of Design Process

### 1. Senior Design Team

The final design of the CPAS system is a culmination of two semesters' worth of investigation, testing, and modification. In order to design a system that fulfills the standards set by the team's customer, Dr. Mark Dean, and the Requirements document, it was essential to divide project tasks evenly amongst each team member. From the beginning of ECE/COSC 402, each member of Team 27 was assigned both a technical and an administrative role:

Patrick Elam – Team Leader/Lead Tester
Nathan Peck – Librarian/Designer
Benjamin Mills – Lead Presenter/Designer
David Lu – Lead Report Writer/Tester
Jacob Lambert – Solutions Architect/Tester
Andrew Stubbs - Website Designer/Tester

Despite establishing these roles, all team members consistently had some contribution in every phase of the design process, and this can first be seen in the initial research performed to define project solutions and implementation alternatives.

### 2. Research of Solutions

#### 2a. Sensor Research

Because the CPAS project was a continuation from the Fall 2014 semester, much of the research regarding potential solutions had already taken place and been documented in papers from COSC/ECE 401. A significant amount of the research leading up to the Spring 2015 semester was devoted to finding and selecting a sensor type that would enable our system to provide accurate, reliable vehicle tracking. On top of providing suitable tracking capability, our selection of sensor type would hinge on whether or not that sensor family could feasibly operate in both indoor and outdoor environments. As well, the sensor we would ultimately commit to would be relatively cheap to purchase and low in operation complexity. Such considerations were made with a mindset of designing a system that is highly versatile and scalable in its implementation.

In the Fall 2014 semester, the following sensor types that were sufficiently researched and debated: video image processing, infrared, ultrasonic, pneumatic tube, induction loops, and pressure pads. In the end, ultrasonic sensors were selected due to their relative inexpensiveness, low complexity, and ability to successfully operate in inclement weather conditions. As well, our research led to the discovery that ultrasonic sensors were a popular choice for parking space monitoring purposes throughout much of Japan. However, many of these existing schemes have very high price tags and

require individual sensors to be embedded into the surface of every parking spot. We hoped to cut down the costs of developing similar systems by using fewer sensors to accomplish the same task.

Beginning in January, we investigated ultrasonic sensor models for the prototyping phase of the project. An attractive candidate early on was the HC-SR04 sensor, which is by far the most popular model for amateur electronics projects as evidenced by user anecdotes found on Raspberry Pi and Arduino community forums.

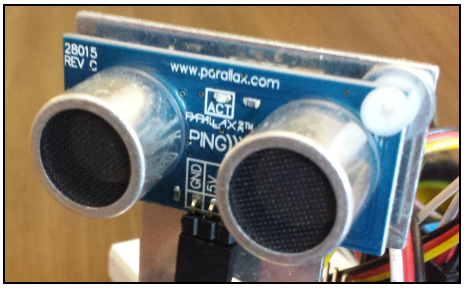


Figure 1. HC-SR04 Ultrasonic Sensor

The model itself is composed of separate sonar emitter and receiver transmitters fixed atop a small printed circuit board foundation. The board includes four digital I/O pins rated for 5V data output magnitudes which could interface without voltage regulation with the digital GPIO's of an Arduino circuit board. However, our early mindset was that we would be using the GPIO pins on the Raspberry Pi microcontroller which are rated at 3.3V. As a result, voltage regulators would be required if we were to bring distance measurement data in from one of these sensors to our Raspberry Pi. Moreover, we were able to find out through discussion with Dr. Dean that these sensors are designed with a strong preference for indoor environments, and, as a result, they wouldn't necessarily provide reliable operation in the outdoor settings of parking lots.

Having dismissed the idea of using the HC-SR04 model, our attention was shifted to locating an ultrasonic sensor that would provide high quality measurements in an outdoor setting. Under further advising by Dr. Dean, we were directed to the catalogs of the ultrasonic sensor manufacturer MAXBOTIX. MAXBOTIX provides a line of weather-proof ultrasonic sensors XL-MaxSonar-WR/WRC series at a noticeably higher price than the HC-SR04's yet a still relatively cheap $100. Not only are these sensors weather resistant, but they provide a number of other desirable features, including:

- Real-time auto calibration and noise rejection
- 3V to 5.5V supply with very low average current draw
- Protective cone shape with a long bore
- Precision narrow beam measurement capability

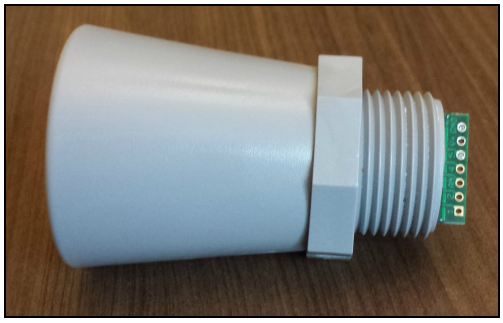

Figure 2. MAXBOTIX MB-7092 Weather Resistant Ultrasonic Sensor

Following initial investigation of MAXBOTIX models, Dr. Dean was able to supply us with two MB-7092 sensors that he had in his possession, and these are the sensors we selected for prototyping. The sensor itself operates by taking distance measurements via sonar emissions and bases data readings on the amount of time it takes for given pulses to be reflected back. It also uses the magnitudes of these reflections to discern object size. In order to read the data from the sensor, we intended to utilize the RS-232 TX/RX I/O pins located on the sensor tail end. We felt that serial RS-232 data would be less prone to noise and transmission losses that have a tendency to get the best of analog signal transmissions.

**2b. Microcontroller Research**

Research into microcontroller solutions was performed concurrently with sensor model research. The hardware component of the CPAS system requires a microcontroller to be wired to a given sensor network such that it can handle collection and processing of sensor data. As a result, it was essential for us to select a controller that we would not only feel comfortable working with, but also one that could provide the following services:

- RS-232 data communication
- 3+ USB ports for sensor and external USB hub accommodation
- Efficient power draw

Initially, debate was centered around whether we would proceed with the Arduino UNO or Raspberry Pi microcontrollers. Internet research led us to discovering that projects involving sensor communication frequently utilize Arduino boards. This is understandable given the following advantages of Arduino over competitors:

- GPIO ports are rated at 5V (the most common voltage seen with sensor data output)
- Arduino has Analog GPIO bus
- Arduino has significantly smaller power draw

Despite these advantages, some drawbacks became present based on further research. For one, it didn't take long for our group to decide on foregoing the use of GPIO pin communication in favor of USB data connection. In addition, Arduino provides an extensive shield library, which, while

inexpensive, tends to increase the cumulative footprint of the board quickly. This would be necessary given the fact that the UNO has only one onboard USB port, and additional ports are available in the form of shield extensions. Another drawback to the Arduino was the team's lack of confidence in its ability to connect reliably to WPA2 encrypted wireless networks, even with a Wifi shield.

One of our major goals with the microcontroller was to find a solution that is as cheap and compact as possible, and, as a result, our preference swayed towards the Raspberry Pi line of controllers. With the Raspberry Pi B+ model, we would have immediate access to 4 USB hubs right on the board. This would give us the capacity to collect serial data from two MAXBOTIX sensors using one board in addition to a few USB-RS-232 adaptors. We would be able to use another USB port to communicate with a USB hub. The USB hub provides connection service for Wi-Fi antenna as well as a Raspberry Pi USB-MicroUSB power cable. Finally, the board itself only costs $35.

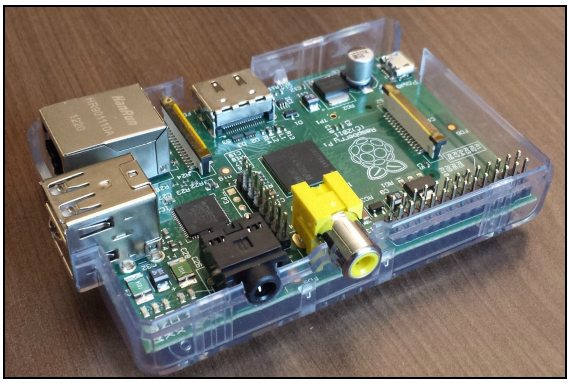

Figure 3. Raspberry Pi

## 3. Prototyping

### 3a. Prototype Location

Following meetings with UT Facilities Services, the Office of Information Technology (OIT), and the Finance and Administration Division, we had a number of options in terms of which parking lot or garage would be the test bed for the initial controller-sensor network. To have the best shot at meeting semester goals and actually implementing our system, the parking lot that we would use would require the following accommodations:

- Access to a 110 VAC wall outlet
- Reliably strong Wi-Fi signal strength
- Only one entrance/exit point (to minimize complexity of implementation)

After extensive debate and professional Wi-Fi signal survey work (courtesy of OIT), we made the final decision to proceed with Parking Garage P2/C5 Commuter Level. This garage has one entrance that doubles as an exit, and, with cooperation from UT Facilities Services, now has a 110 VAC wall outlet conveniently installed at the upper lip of the entrance overhang for use by our system.


Figure 4. Entrance/Exit to P2/C5 Commuter Parking

### 3b. Prototyping of Solutions

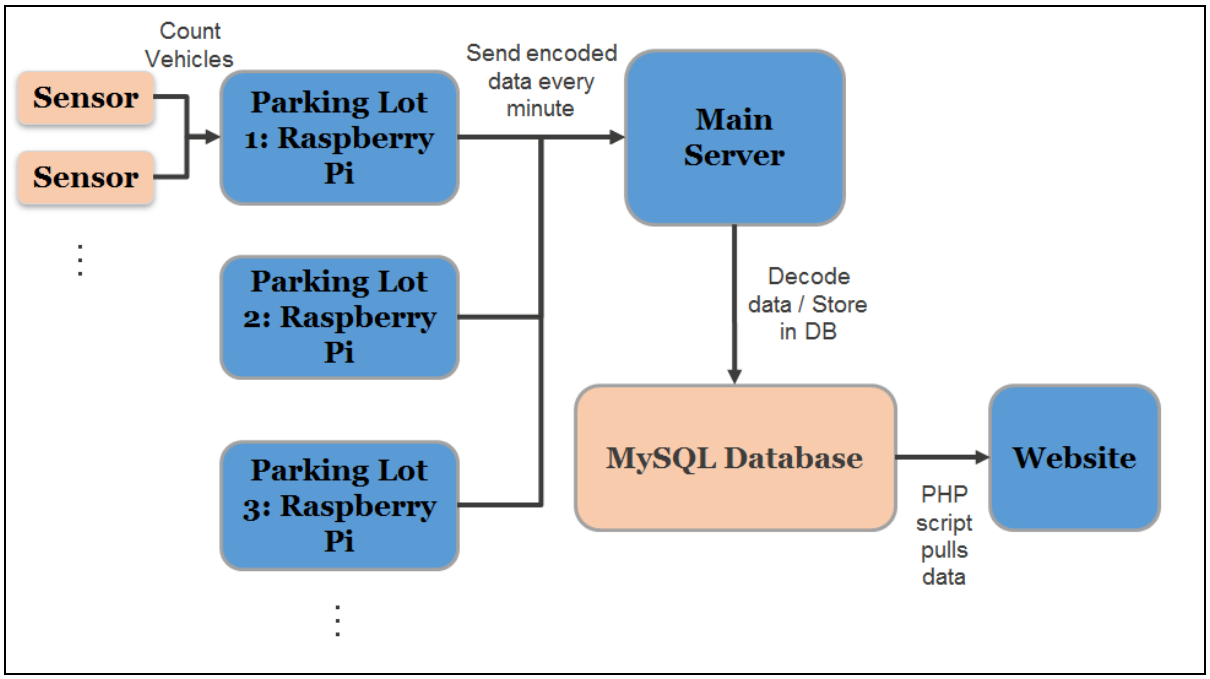


Figure 5. Solution Outline

### 3c. Sensor & Microcontroller Setup

CPAS required the installation of the selected sensors at each parking entrance/exit, in order to accurately measure the quantity of incoming/outgoing vehicles. The configuration of these sensors is set based on on-going testing efforts. These sensors are connected to a Raspberry Pi, which is connected to a continuous 110 VAC wall outlet. Future implementations of this system will attempt to utilize more mobile power sources, such as batteries, solar panels, etc. The Pi interfaces with the sensors to translate signals into the relevant data. More specifically, the Pi needs to be able to determine the difference between an incoming and an outgoing vehicle. Once this is determined, the current total number of vehicles is updated accordingly. The Pi is then responsible for encoding this data and sending it to a central server every minute. For the purposes of our prototype, the initial data would include a running total number of vehicles, a unique ID for the parking lot, a unique ID for the microcontroller, and a time stamp. The communication path between the Pi and the server is accomplished by opening a socket connection through the WPA2 Wi-Fi network.

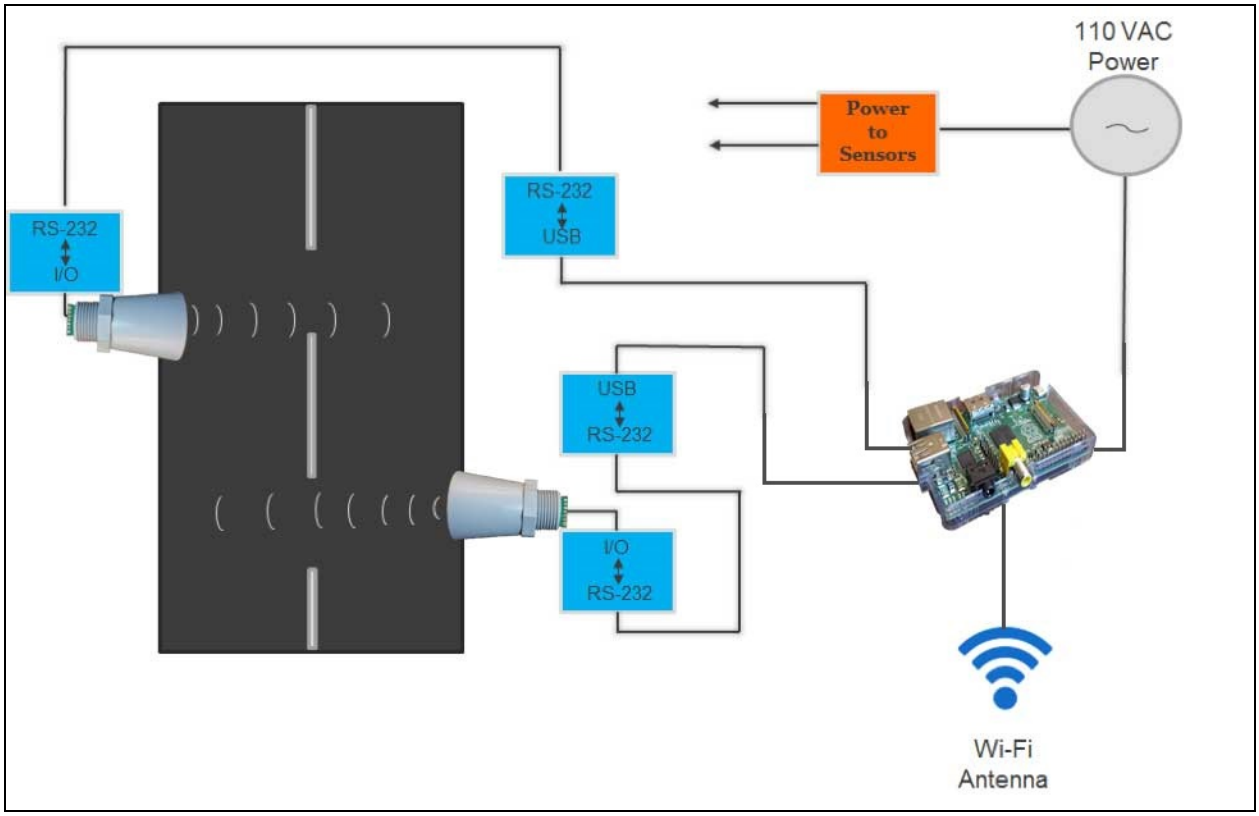


Figure 6. On-Site System Design

### 3d. Central Server & Database

OIT provided our team with a server running on a virtual machine (VM). The VM we were given is running Red Hat Enterprise Linux. We were given root access and installed an Apache webserver, mySQL database software, and Git, along with other development software. For security, the server itself has a firewall built into the operating system, and OIT also has a firewall protecting its VMs. We configured the server to accept connections coming from within UT's WPA2 encrypted network, as well as UT's wired network. We opened up a port for communication with the Raspberry Pi on the server's firewall, and also had OIT open the same port in their firewall. The central server will eventually be responsible for handling incoming data from every Raspberry Pi in the CPAS network, but our prototype will initially only handle a single Pi. The server receives encoded strings from the Pi, every minute, through a socket connection. Each string is decoded, and the data is stored in a connected database.

The server needs to ensure that every Raspberry Pi in the network is functioning. The Raspberry Pi will push data every minute, creating a heartbeat system. If a Raspberry Pi does not push data often enough, this might be an indication of a malfunction in the system. This fact can be reflected on the website, indicating that the occupancy shown may or may not be accurate. Once a problem is identified, it can be located and administrators notified such that steps can be taken to fix the issue.

### 3e. Website Implementation

After looking at different solutions we decided to implement a website using an Apache web server. We decided to use Apache as it is what the team was the most familiar with. The website would retrieve relevant parking information from the database using PHP scripts. PHP is a server side scripting language commonly used in combination with HTML, to send and receive information from a database. We decided to use the Bootstrap framework in order to make sure that the website would work on both mobile devices and computers. Bootstrap is an HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. [1] This configuration attempts to create the most optimal and simple interface for our system. With the Bootstrap framework, our website would be viewable on any device with a web browser. The framework automatically scales page information to be viewable at any resolution. This meant that development of the user interface would be universal and apply to all platforms and operating systems (Windows, OSX, Linux, Android, iOS, etc).

We installed the Apache web server on our system, which allows us to host our web page on the given server. In coordination with Bootstrap and CSS, we have created a basic HTML home page for our system. The website will access our database in set intervals in order to maintain the most updated version of our data set. Initial contents of our website consist of a total count of the vehicles in the parking lot. Future work on the website will create more efficient tools, such as maps, graphs, and statistics, to better display our data to the end-users. Additionally, we will need to take into account the safety issues of using a mobile device in a vehicle. This problem could be resolved by implementing a GPS-like voice navigation system, which would eliminate the need for the user to physically interact with the website. These future changes are further elaborated upon in Section 5.2.

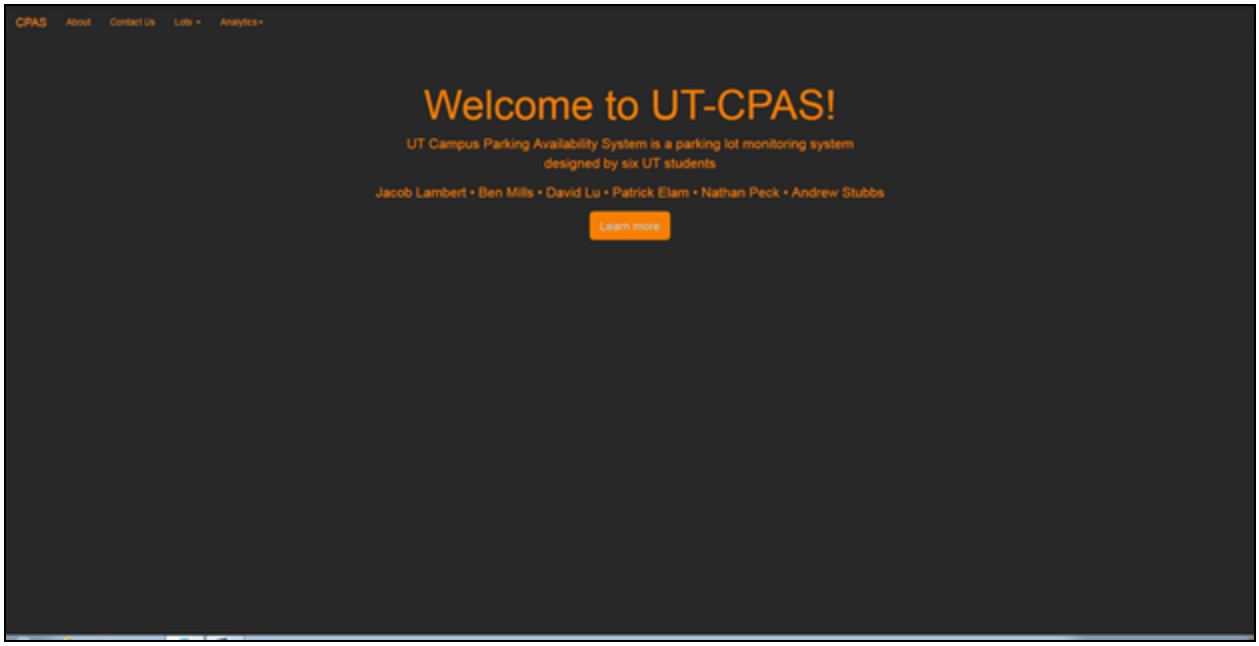


Figure 7. Website Homepage at [cpas.eecs.utk.edu](cpas.eecs.utk.edu)

## 4. Final List of Materials

### 4a. On-Site System

- 2 MAXBOTIX MB-7060 Ultrasonic Sensors
- 2 25’ RS-232 Extension Cables (1 per sensor)
- 2 25’ 5V DC Power Extension Cables with Barrel Connectors (1 per sensor)
- 2 DB9 Female Breakout Boards to Screw Terminals (1 per sensor)
- 1 120V AC -> 5V DC Wall Adapter Power Supply
- 1 Barrel Connector 2-Way Splitter
- 2 Barrel Connector DC Power Jack Sockets (1 per sensor)
- 2 RS-232 to USB Converter Cables (1 per sensor)
- 1 Raspberry Pi B+
- 1 4-port Powered USB Hub
- 1 USB Wi-Fi Antenna
- 1 USB to micro-USB Cable for Pi Power
- 2 (3” x 2” x 4”) Plastic Boxes for Sensors and DB9 Breakouts
- 1 (5” x 7” x 9”) Plastic Box for Raspberry Pi, Cable Slack, USB Hub, and Wi-Fi Antenna

Figure 8. Main box holding the Raspberry Pi, Wi-Fi Antenna, USB Hub, and cable slack. Stowed in the upper lip of garage entrance next AC to wall outlet.


Figure 9. Box housing Sensor #2. RS-232 extension cable and power cable are being fed in from above.

# V. Lessons Learned

### 1. Sensor Interference

Our initial tests in the Senior Design lab showed no sign of signal interference between the two sensors. Once the sensors were installed in the parking garage we began to notice that we were getting errant readings. However, these readings did not occur until the day after the sensors were installed ( about 24 hours later). After numerous tests, we determined that the sensors were beginning to conduct range readings with slightly different frequencies, and this frequency drift caused interference between the two sensors. Each sensor was operating at a different timing interval. For example, our first sensor conducts a range reading every 48.0mS, while the second sensor conducts a range reading every 48.2mS. Eventually, the timing of the two sensors would reach a point where one is in transmitting mode and the other is in the receiving mode. This drift was the cause of the errant readings.

In order to solve this problem, we needed a way to synchronously activate the sensors independently to take a range reading instead of letting them run autonomously. The sensors have a pin that can be pulled high to command a range reading, but we had no way of accessing that pin via our USB -> RS232 adapters. In order to access the pin on the sensor, we decided to use the previously unused TX (transmit) pin in the DB9 serial cable. In order to gain access to this pin, between the DB9 cable and the USB->RS232 adapter, we installed a modified extension cable. We cut this extension cable open to gain access to the pin that we needed, and we soldered an extension to the single wire within the cable.

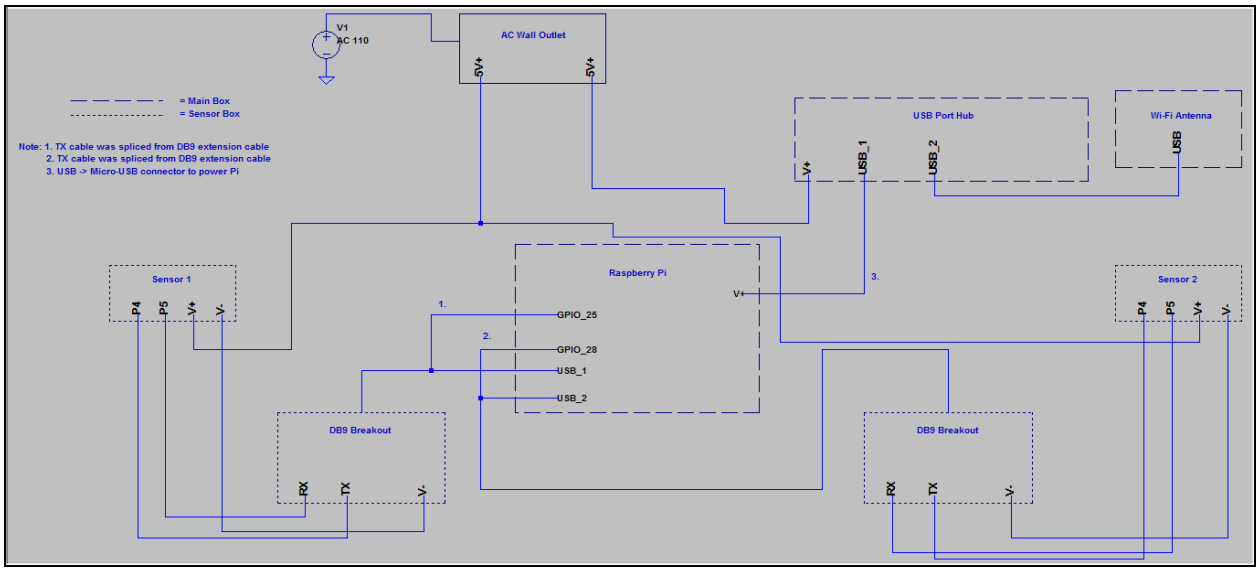


Figure 10. Schematic of the on-site system

We then connected this extension to the GPIO pins on the Raspberry Pi, allowing us to pull the signal high in the software and activating a range reading

After gaining the ability to command a range reading, we tried simply resetting the sensors at the same time, but this configuration still showed signs of interference. We decided that the next best option was to activate the sensors sequentially.

Sequential readings would reduce the chances of interference, since only one sensor would be reading at a time. For each read, one sensor's command pin was held high, and after 20µs, the sensor would send a pulse. This pulse took 40ms to return to the sensor. Then, the second sensor's command pin was held high to command a reading. The timing diagram below shows how the pin voltages and sound pings interacted. This solution addressed our initial issue and allows us to accurately track the flow of cars with no signal interference between the two sensors. At the same time, this solution allows us to still obtain 10 readings from each sensor per second.

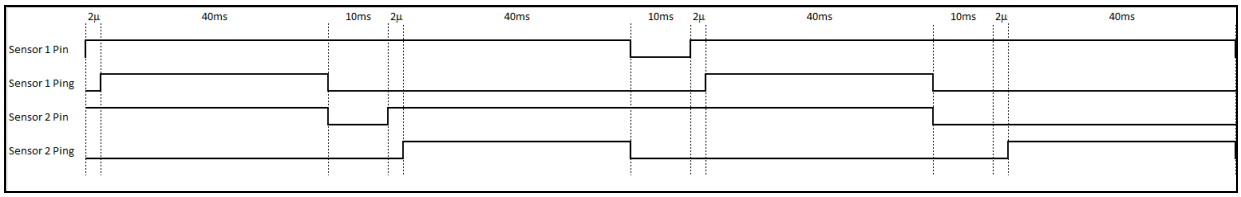


Figure 11. Timing diagram for sensors

## 2. Future Changes

As the ECE/COSC 402 course comes to a close, and we fulfill our final requirements for the CPAS project, there are still several features that can be added for the project's future development and refinement. The proposed modifications aim to make the system more accurate and improve the scalability, innovation, and marketability of the end product. These changes would allow the CPAS to encompass every parking lot on campus, providing the most complete parking information system to the user.

### 2a. Battery Power

Undoubtedly, the development that would most exponentiate the system's physical scalability would be to adapt the on-site system to run off of alternative power sources outside of traditional main AC power. Many of these alternative power sources involve renewable energy generation working in tandem with a constant, rechargeable battery supply. Each ultrasonic sensor accounts for 50 mAh of current draw, while the Raspberry Pi B+'s draw can range anywhere from 500 - 1000 mAh at a given time. This number takes into account the draw of 200 mA associated with each USB port in use. USB Wi-Fi dongles, such as the one used in this project, have a current draw from anywhere between 100 - 500 mA. In the ideal case, purchasing items on the lower current draw range across the board would bring the minimum hourly current draw of the on-site system to about 600 mAh.

Ideally, the CPAS system would be capable of 24/7 monitoring even under battery-powered conditions. On the optimistic side, a standard 12V car battery has a 40 Ah ~ 60 Ah capacity, which could uninterruptedly supply the system drawing 600 mAh for 66 ~ 100 hours. However, adding a car battery to the system adds the disadvantage of higher project cost, voltage regulation equipment

integration to step down battery voltages to usable levels, and making the system more environmentally intrusive given the battery size.

Examining the battery power capacities in a near best case, the other obvious concern with adapting to battery power is integrating a renewable energy source to maintain battery charge. Solar power first comes to mind. Harnessing solar energy effectively requires investing in solar cells with good conversion efficiency, which, realistically, would be somewhere around 15%. These cells would also have to be suitably consistent in their ability to charge the system battery. Another concern would be the placement of these solar cells to provide optimum exposure to sunlight. Alas, investigating a battery-powered CPAS solution would surely provide curious minds with a difficult, but rewarding project.

**2b. Software Enhancements**

On the software end, there is significant room for growth and innovation in CPAS. One such innovation would be to integrate voice-interactivity for mobile platforms that access the website. There are obvious concerns with regards to users simultaneously driving and accessing their phones while attempting to navigate the website. Ideally, having full website voice navigation would provide users with a reliable, hands-free method for locating available parking options while they are focused on driving.

Another noteworthy software enhancement would be Google Maps integration. The Google Maps application can already use a mobile phone's location to provide GPS directions. Working with the previous voice navigation feature, the system could utilize this GPS feature to determine appropriate parking options based on proximity. For example, Google Maps could determine your current location on campus and reference the CPAS database to let the user know the closest parking lot with available parking spots.

On the security side, website notifications and warnings can be implemented to alert users of losses in data communication. This would allow the system to determine that a particular parking lot's on-site system is malfunctioning and possibly relaying unreliable data.

As an alternative to adding the above features to the website, future iterations of this project might feature the development of dedicated mobile apps for each major hardware platform. A dedicated mobile app would allow users to store their favorite lots, and would allow users to be notified of any number number of events, such as a loss of data or the filling up of a garage that they were headed to. Users could also be notified of new garages that were added to the system.

From an administrative standpoint, many other software features are possible. Real time and historical data on parking availability and average garage occupancy could prove invaluable to parking services. Ticketing officers could know whether there were cars in a lot without having to go and check. Parking could be tracked specifically during sporting events.

## VI. References

Datasheet for Maxbotix MB7060 Ultrasonic Sensor:
http://www.maxbotix.com/documents/XL-MaxSonar-WR_Datasheet.pdf

Information on the Raspberry Pi B+ Model:
https://learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-model-b-plus-plus-differences-vs-model-b.pdf