BlueGrid: Parking Availability Sensors

Steven Lang • Dr. Scott Feister • COMP 499

Introduction

This project has been designed from the ground up to enable users of accessible parking spaces to better plan their commutes through live parking availability and location data. This proof-of-concept created would allow for businesses and cities to be more effective at providing accessible solutions to the public/customers.

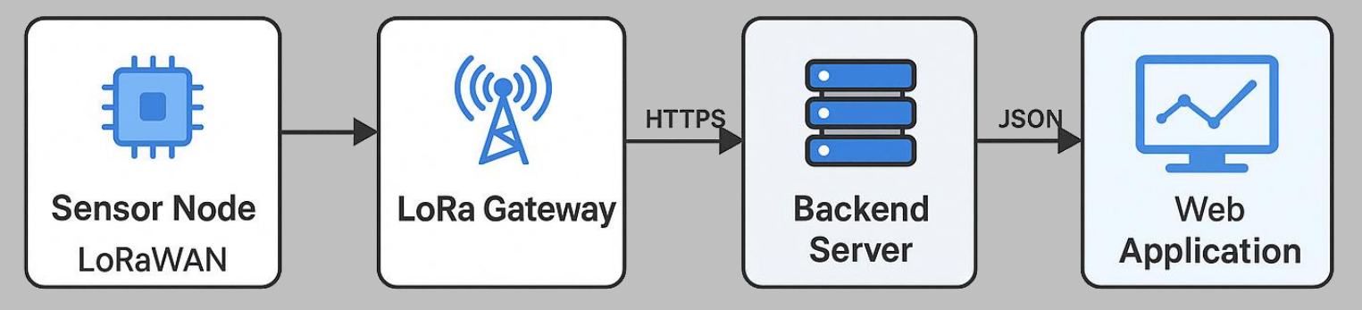
By leveraging a network of low-cost, LoRaWAN-enabled sensor nodes installed at each accessible parking spot, the system continuously monitors and transmits real-time availability data. This information is served through a modern, user-friendly web application that displays both live status and precise GPS coordinates of each monitored space. Users may navigate to the web application and see availability for accessible parking spaces based on location. In addition, each space has GPS-based location data that users may access through the map feature.

It lacks navigational capabilities at this time, however it was designed around the idea to allow for the closest available spot nearest the individual to be found at a moment's notice. In addition, this platform demonstrates the capability for institutions to adopt and provide support through smarter infrastructure.

Implementation

The goal for this project was to have a product that is stable, accurate, and close to real time while staying under budget.

I knew from the start that I wanted to use LoRa-WAN based technology due to the potential difficulty of providing a WiFi-enabled solution. In addition, it would allow for a future transition to microcontroller support.



The full system was developed via the following methods:

Embedded Systems
Sensors and relays both stem from a Raspberry Pi Zero 2 W, enabling the potential for Wi-Fi connections, file storage, and the ability to program with Circuit Python. Attached to both is a LoRa-WAN breakout board serving the USA license-free frequency range of 902-928 MHz for communication. Sensors are equipped with an HC-SR04 ultrasonic sonar sensor for simple spatial distance detection.

Total Cost: ~$55

Backend
The sensor data, when it reaches a network capable relay or home node, is provided to an exposed Python FastAPI. The data then updates a MariaDB database with the updated status of the sensor. Both are independent, modular, and fully containerized for quick deployment and error resolution.

Frontend
The main web page is served via an ASP.NET Razor Page which pulls from the FastAPI to get the sensor data. It then groups the data based on location (parking lot) and live updates with each space's availability status.

The map page utilizes the Mapbox API through classic HTML, CSS, and JS, allowing for GPS Longitude and Latitude data to be used as request parameters.

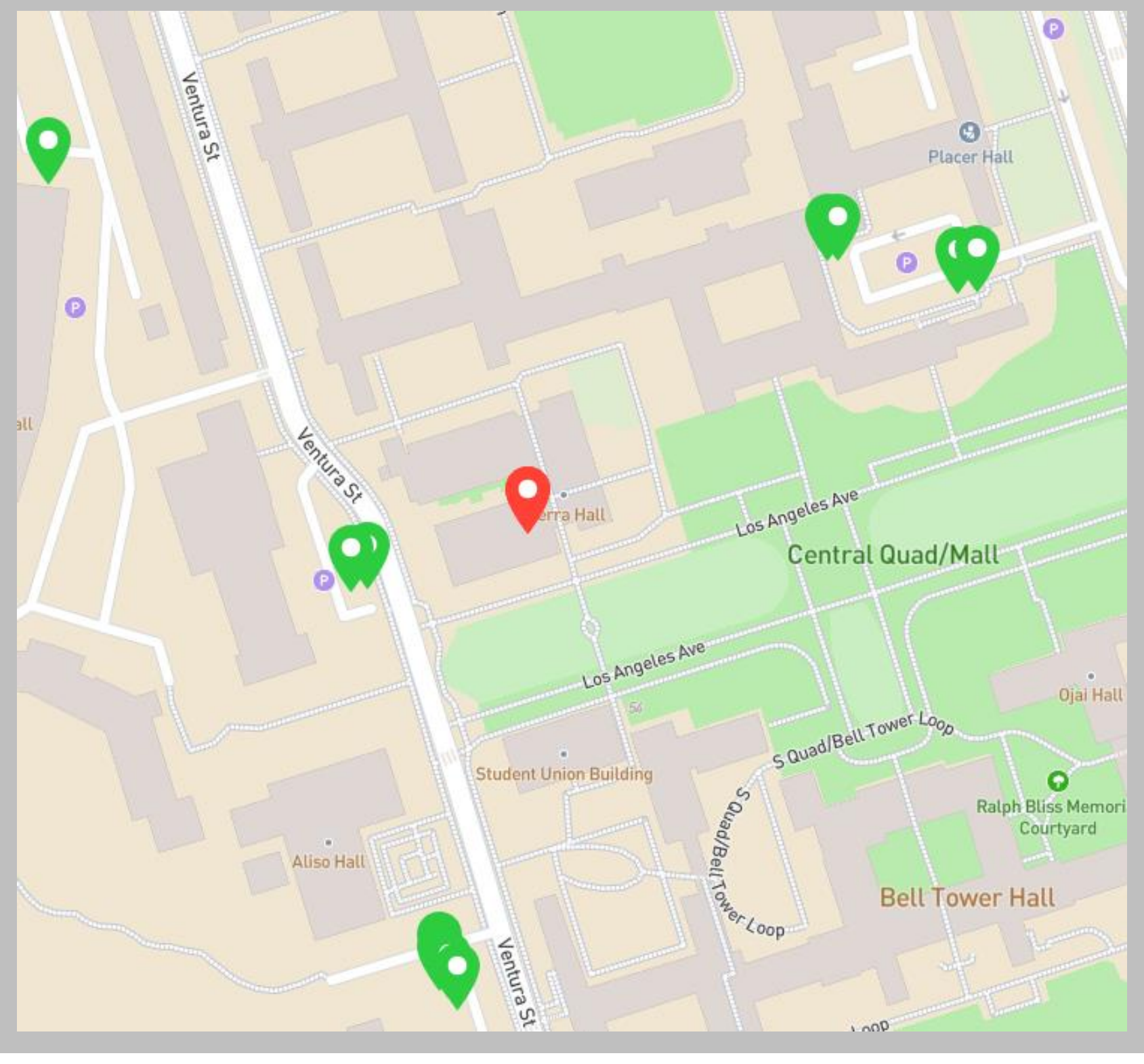
Both are served on my personal domain using Apache.

Results

The final results from my proof-of-concept determined that the project is a viable solution. Testing two sensors and a relay/gateway node setup in a controlled environment, I was able to have success at each step of the process. The sensors would read appropriately, the gateway would receive, and the database and web application would update appropriately. Having my own car parked in my controlled space was reflected in seconds within the application, just as with removing it.

Below, you can see a mockup of the targeted parking spaces with color-coded availabilities on our own campus compared to Sierra Hall. Each logically would live update the same, however for the sake of this project it is set to only provide demo data.

While the university does have accessible parking locations available, the project successfully proves that a cost-friendly live data solution can be done to enhance the experience and meet user needs.



Challenges

Each step of this project presented individual difficulties throughout, from diagnosing hardware issues to securing API exposure. Early iterations involved finding the best data storage method. Before settling on the FastAPI and .NET page, I attempted to use Graylog, Prometheus with Grafana Loki, PHP, and even simple text files to store and serve the data. None of them were as robust as I had hoped and came with their own issues of needing/leaking database credentials or excessive storage space.

The difficulty with the sensors came from the logic behind sending the data. At first, I sent data, always sending false (vehicle not there) once a second until it sensed something. This was a much larger power draw than first anticipated, as well as the cause of timing issues. If two sensors sent at the same time potentially only one or neither would be read by the relay node. After some trial and error, I finally realized it only needed to send when the state changed, and it could be logically assumed that a vehicle was not there until the sensor went off in reasonable succession, i.e. a person walking past would not send as opposed to a vehicle remaining stationary from a specific distance away for several seconds, finally sending the occupied flag to the relay node.

Power had been another issue to deal with. The embedded system requires a steady 5V at an average of 20 mA. Several iterations of battery packs and solar were used with two variations completed for the final design.

Another challenge that I needed to solve was a sensor heartbeat. There was no way to know if a sensor has completely failed by having its program crash (and not reboot) or the power source failing. One solution I attempted in a simulated test environment was a system daemon running that would consistently check if any messages had been received between a certain time by a node. If there were none, it was assumed as having failed. This however clashes some with the sensor logic and needs to be workshopped further.

Finally, I learned some time into my project that there are potentially other viable solutions that I could have implemented such as Meshtastic or MQTT.

Future Development

There are several ways I could have taken this project, and while I am happy with the end-result, there are some improvements or alternatives I would have liked to take. If I were to develop this further or suggest direct improvements, I would consider the following:

- **Different Hardware**: Microcontroller use instead of the Pi Zero to save power and use alternative communication methods. The Pi and its current power source use too much idle to be sustainable, lasting an average of 8 hours during my test trials. Another effect from utilizing the microcontroller is that it may also save even further on costs.
- **Create a mobile App**: I had started the project with having Waze style navigation be at the heart. This implementation however was too costly both technically due to my personal skill level as well as financially due to API usage. It would be nice to see it working where users may get directions to the nearest available spot to their desired destination.
- **Queueing**: Similar to some restaurants or EV charging apps, I had considered allowing users to save their place in line and be alerted once a spot became available. This would have been implemented with an HTTP form, a separate database table, and some form of email or other messaging service had I decided to fully commit to it. In addition, this may have resulted in issues with users not respecting the non-mandated line.
- **Advanced Sensing**: The current solution utilizes cheap and simple acoustic sensors, however I played around with the idea of using the Raspberry Pi Camera while running object detection/computer vision software using PyTorch. This would have increased accuracy tremendously at the cost of different hardware needs and higher power draw. This would have also made the maintenance and cost increase too much to be considered a viable product.
- **Real-World Trials**: My results come from a controlled environments that would mimic and actual parking space and relay node distance. The primary difficulty in fully testing the solution came from requiring permission from parking lot owners to trial the hardware. In addition, it would have required a large sum to have enough sensors available to be able to test a real parking lot that might average 4 applicable spaces.

Acknowledgements

I would like to thank Dr. Feister for his guidance and expertise throughout the project.

I would like to thank Professor Rios for his much-valued mentorship throughout my time at the University.

I would also like to thank my parents for providing continuous motivation to achieve and never forgetting how capable I can be.

Finally, I could not have achieved all that I have done without the support from my partner Audrey, who has been there every step of the way.

Links

LinkedIn	Capstone	Capstone - Map