Smart Parking Application with Ultrasonic Range Sensor

Andrew Dircks and Bennett McCombe

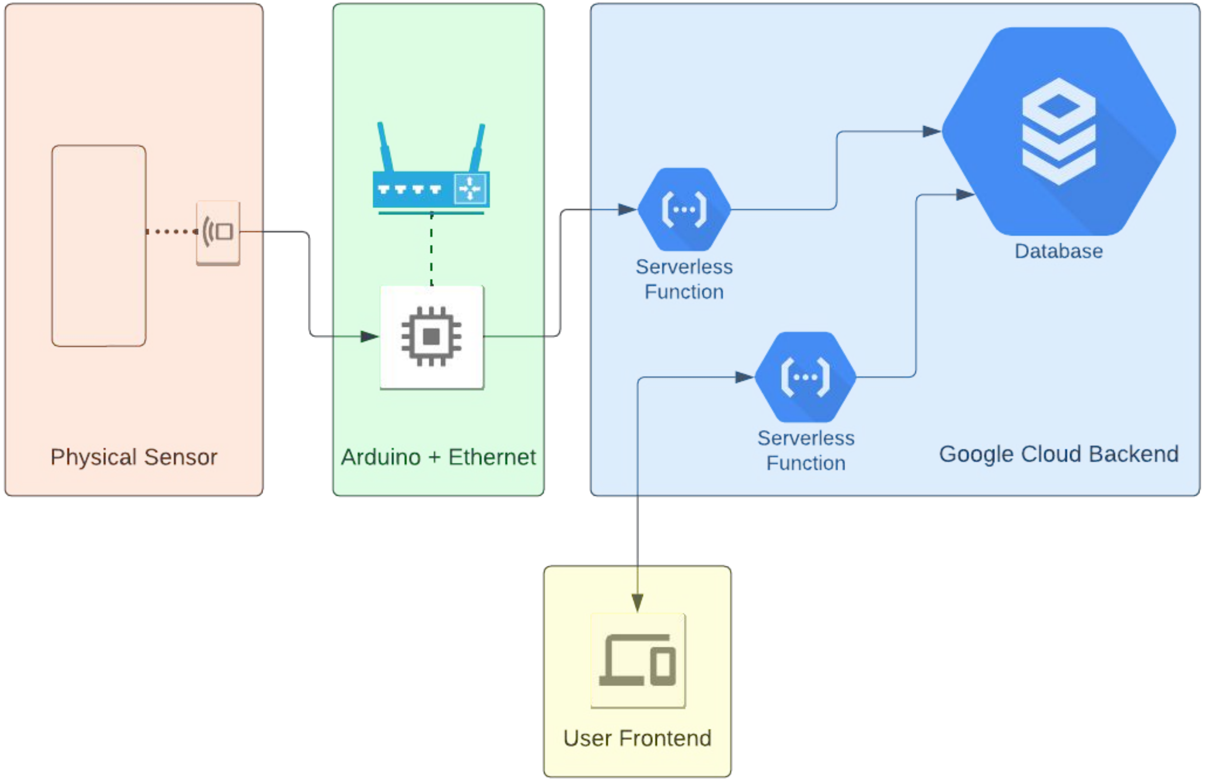
Motivation

- We share a 3-car lot with 8 of our housemates …
- A lot of time spent checking spot availability, then looking for street parking
- It would be incredibly useful to access a website showing spot availability in the lot

We aim to build a “smart parking” full-stack application to determine spot availability in real-time.

- Calibrate, connect, and mount an ultrasonic distance sensor
- Connect an Arduino to the internet to upload live measurements
- Build a cloud data pipeline and simple website

System Schematic

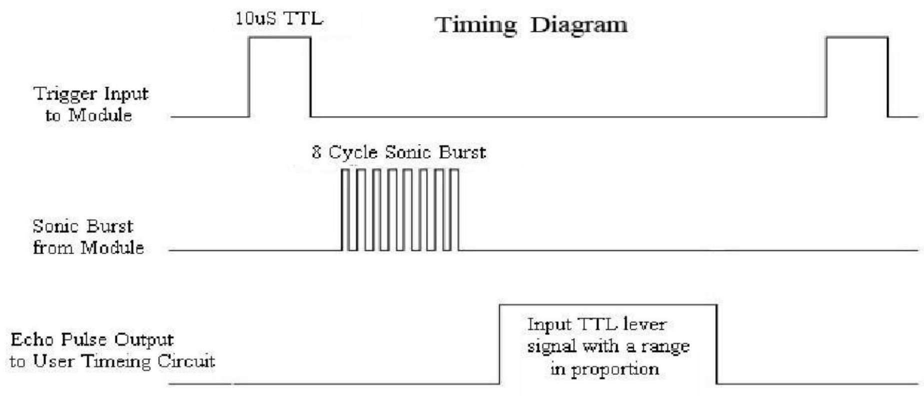


Ultrasonic Range Finder Specifications

- Sends 40 kHz signal out to determine if there is a pulse back
  - DC 5 V working voltage
  - 15 mA working current
- Costs $3.95
- Distance range: 2 cm - 1 m
- Analog output
- Great for use with translucent objects such as windows or liquids where infrared and optical sensors struggle

How it Works

- The transducer of the sensor 40 kHz cycles out following a 10 uS trigger input, then waits for the time that the pulse takes to reverberate off an object and bounce back to determine range



How it Works

- This sensor calculates range by using the formula:

$$\frac{\mu S \textit{ between trigger signal and receiving signal}}{58} = \textit{distance to object } (cm)$$

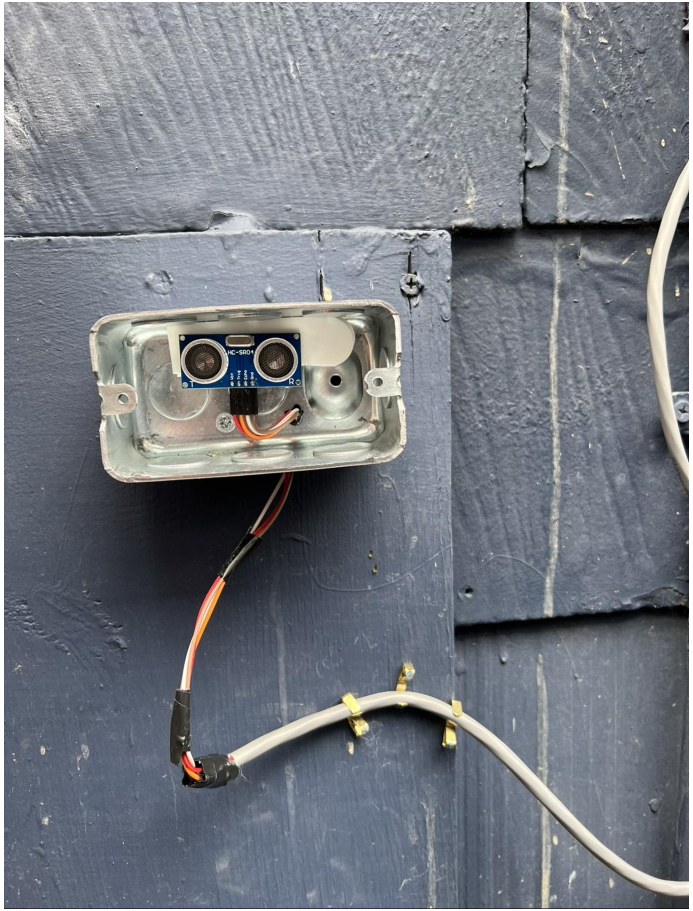
    - The seemingly random “58” factor is double the speed of sound (μS / cm)
- Numerous factors affect the accuracy of the sensor such as rain, snow, or sharp angles of the object it is detecting

MaxBotix

Installation

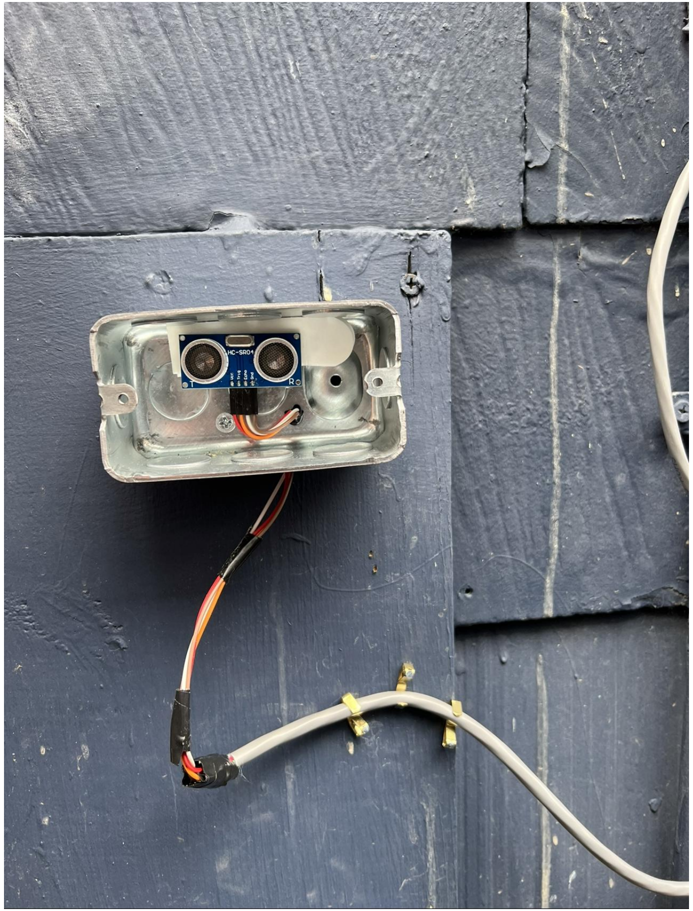
There were three objectives in installation

- Accuracy of detecting a car in the spot
- Weatherproofing
- Reliability and hardiness of mount and wiring



Installation

- Our initial design involved a metal casing for the sensor, with the sensor attached by velcro to be able to remove easily for troubleshooting
- Our wires were attached securely but exposed to the weather



Installation

- We then added an additional rain cover over the top to stop water from getting into the casing
- Wires were further insulated by electrical tape, and further secured by more duct tape

Installation

- Our shielded wire was then fed behind other wires attached to the house, and into the window
- This uses about 10 feet of shielded cable

Installation

- Once inside the house, the output from the sensor are plugged into the Arduino and Ethernet Shield
- The Ethernet Shield is wired to a router
- The powersource of the Arduino is via USB, plugged into a surge protector

Arduino + Internet

- The “Ethernet Shield” for the Arduino allows connection to the internet via ethernet cables, a WiFi router, and a simple code library

```
#include <Ethernet.h>

// ethernet init
byte mac[] = { 0xA8, 0x61, 0x0A, 0xAE, 0x6F, 0xA0 };
byte ip[] = { 192, 168, 1, 224 };
char fn_host[] = "us-east4-parkcath.cloudfunctions.net";
String fn_name = "parked";
EthernetClient client;

void setup() {
  // ...
  Ethernet.begin(mac, ip);
  // ...
}

void write(int spot_id, int distance) {
    if (client.connect(fn_host, 80)) {
        client.println("GET /parked?dist=" + String(distance) + " HTTP/1.1");
        client.println();
    } else {
        Serial.println("connection failed");
    }
}

void loop() {
  // calculate distance
  // ...
  write(distance);
  // ...
  // wait
}
```

Backend API

How can we upload and store our data efficiently and inexpensively for real-time access and post-analysis?

https://github.com/andrewdircks/smart-park

Google Cloud Platform

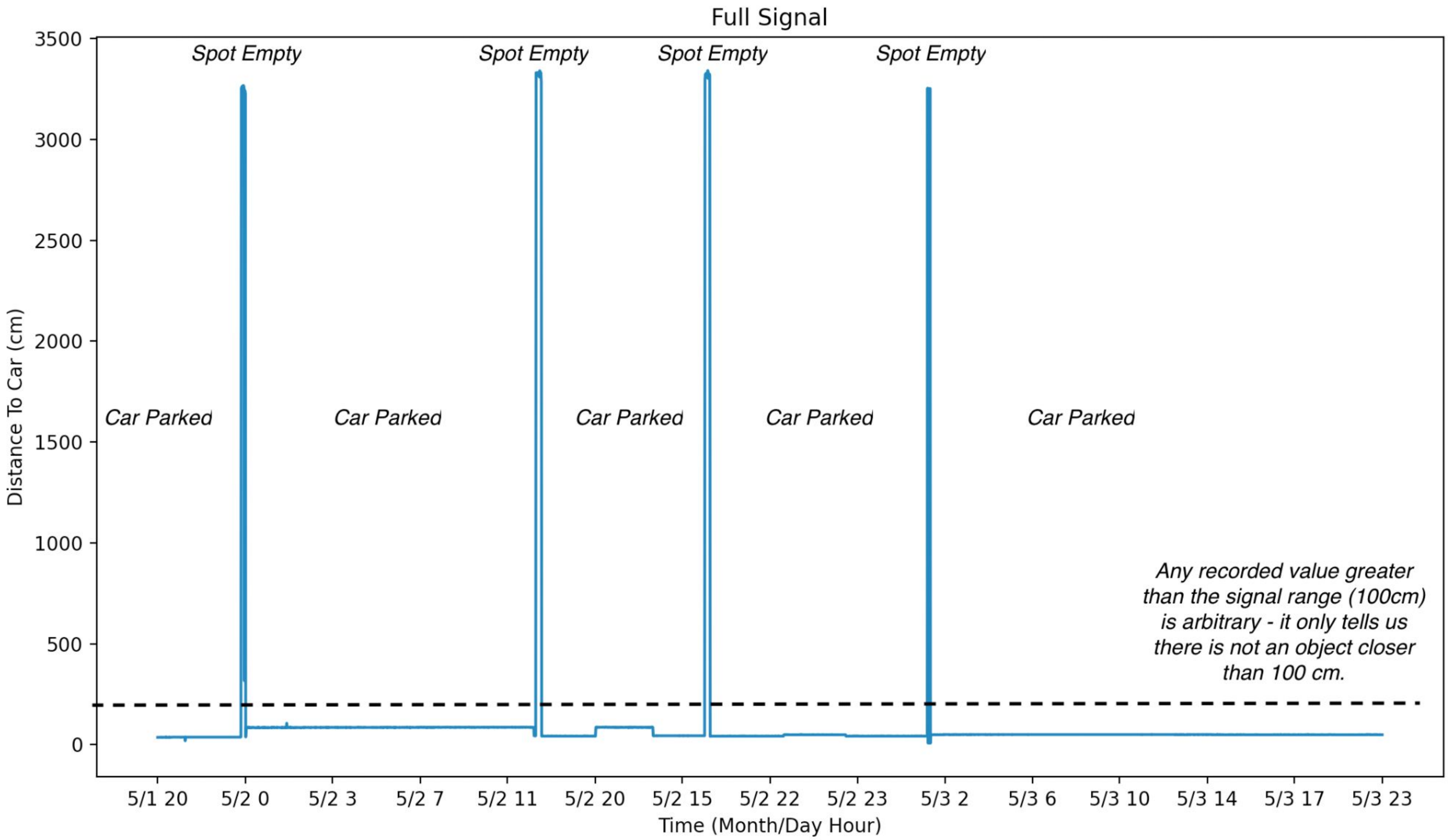
- PostgreSQL Database
- "Cloud Functions" (pay by execution-time compute)

Every Minute…

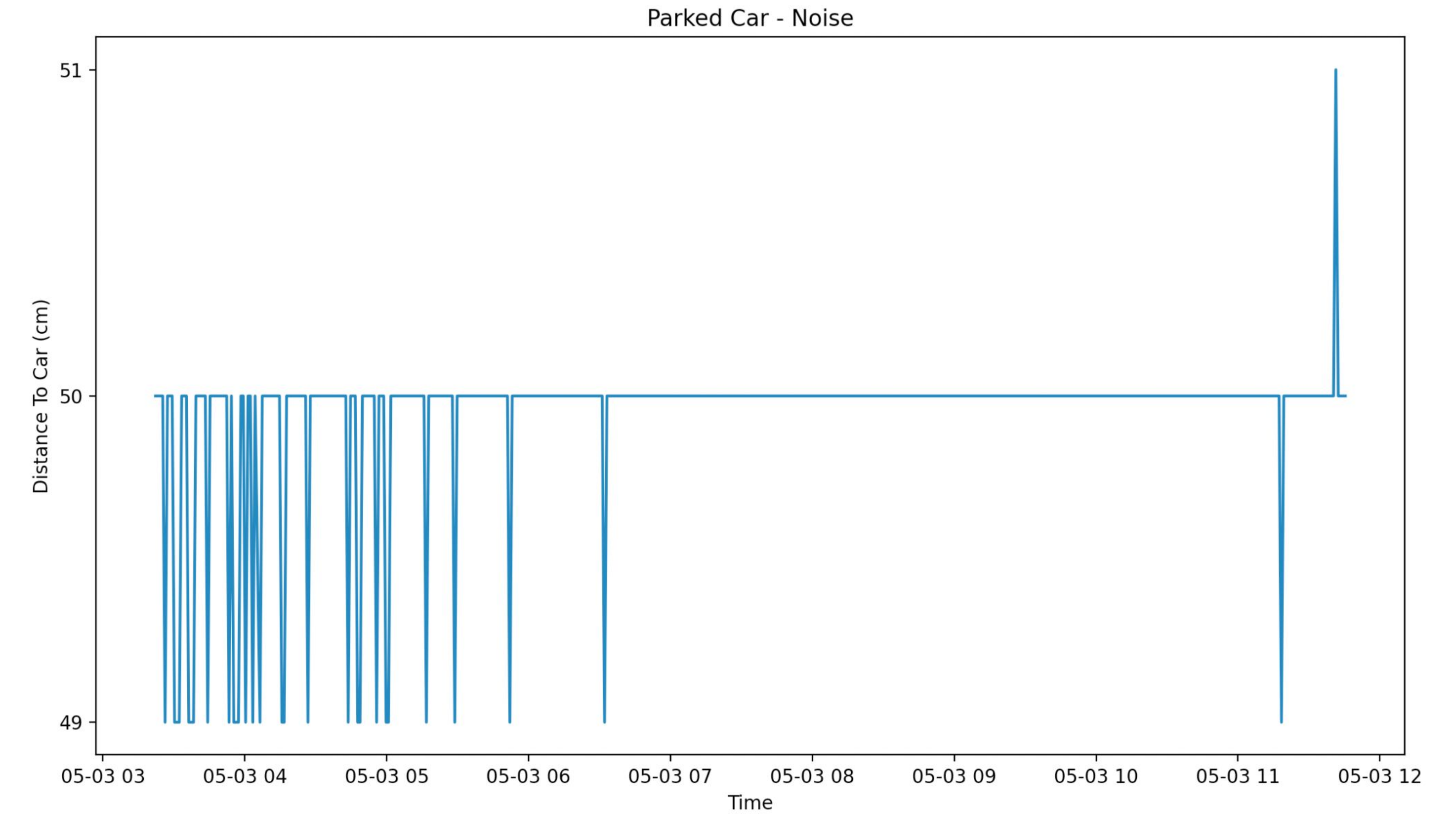
- Sensor makes a distance measurement
- Arduino calls a Cloud Function with that distance (via *Ethernet* library)
- Cloud Function writes the data to our database, where it is stored for later use

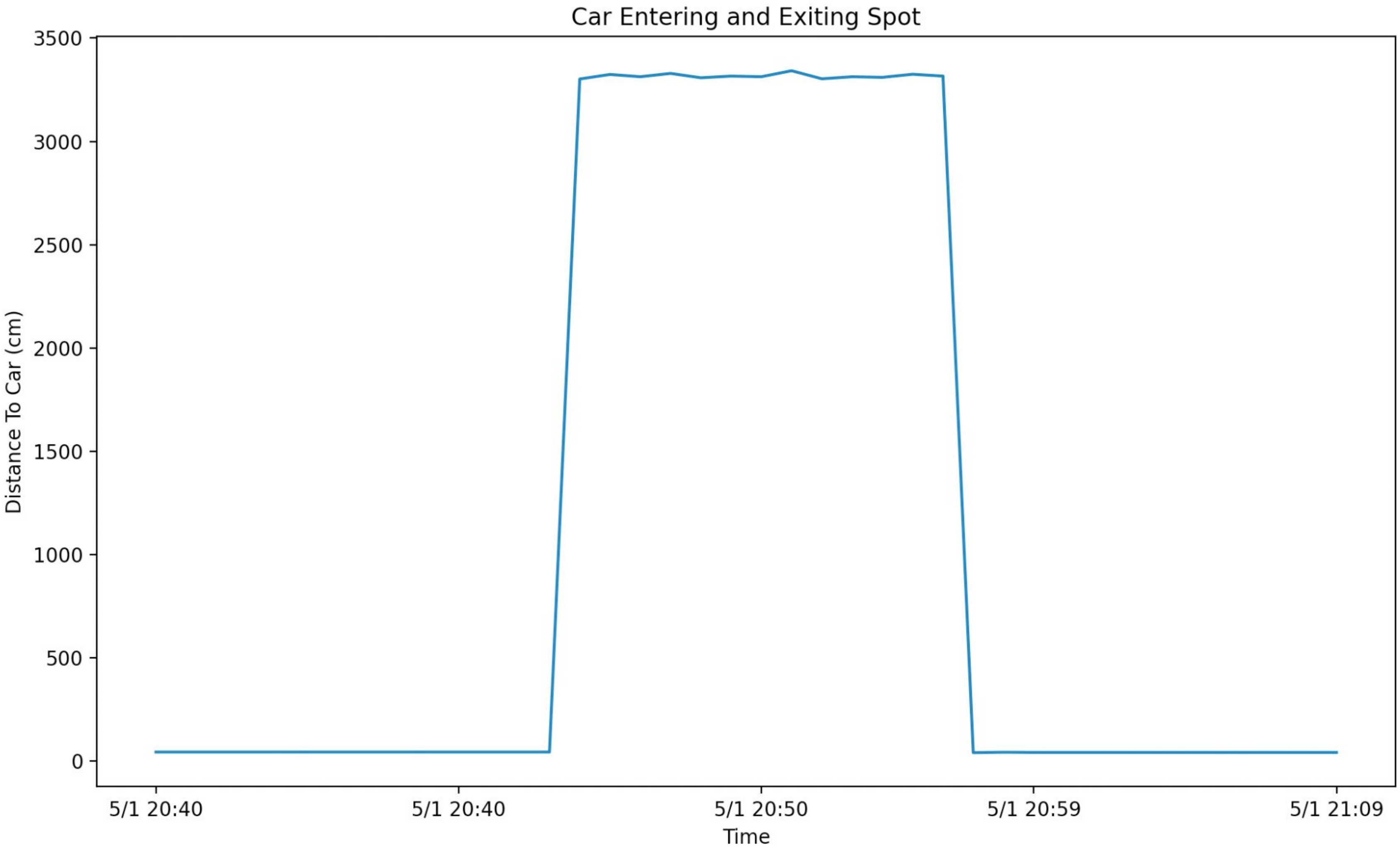
User Frontend

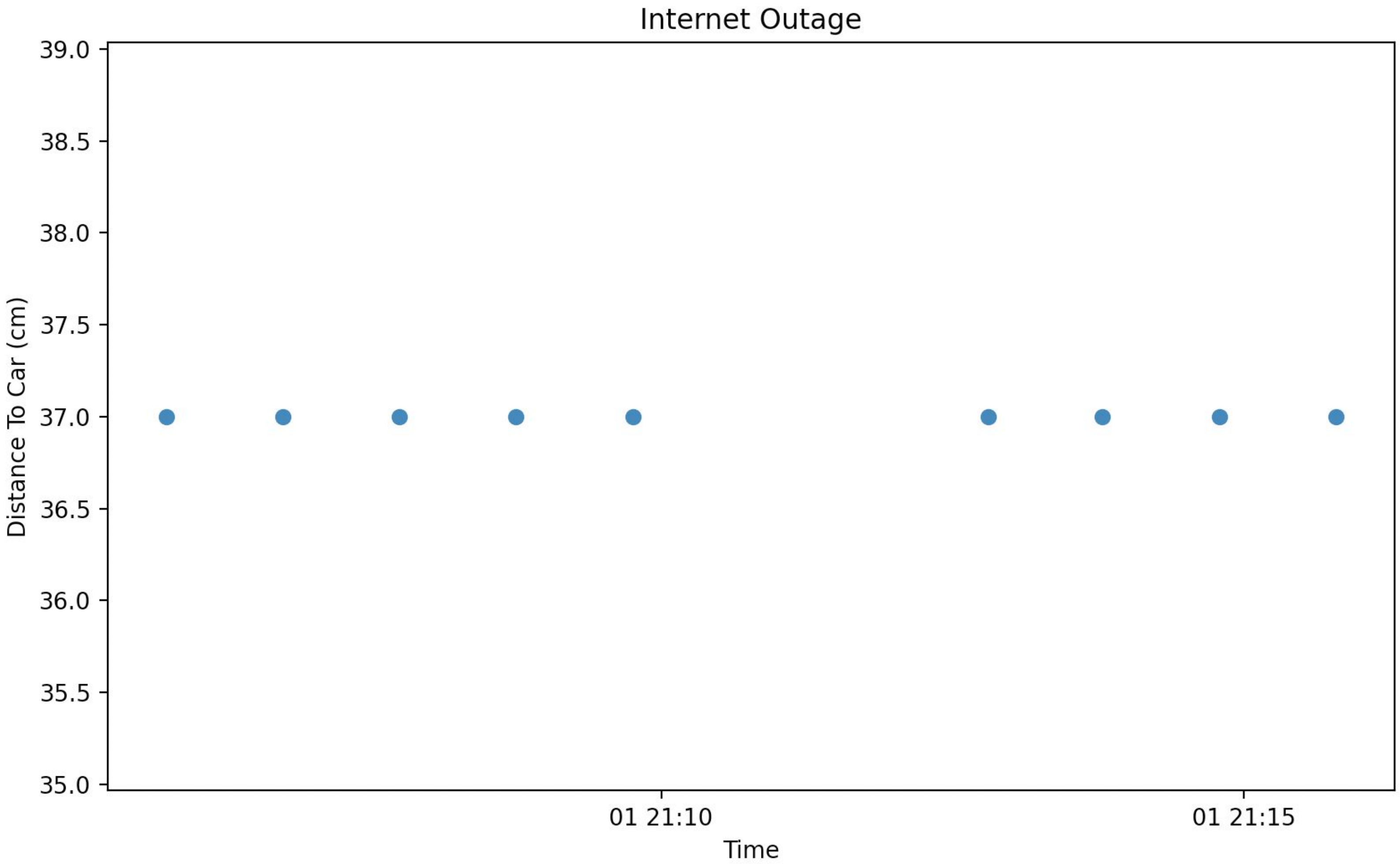
- We use another Cloud Function (accessed as a url) to
  - Fetch the most recent parking distance data from the cloud database
  - Determine spot availability
  - Serve a simple HTML webpage displaying availability
- Potential improvements to user interface



Full Signal
Spot Empty
Spot Empty
Spot Empty
Spot Empty
Car Parked
Car Parked
Car Parked
Car Parked
Car Parked
Any recorded value greater than the signal range (100cm) is arbitrary - it only tells us there is not an object closer than 100 cm.
Distance To Car (cm)
3500
3000
2500
2000
1500
1000
500
0
5/1 20
5/2 0
5/2 3
5/2 7
5/2 11
5/2 20
5/2 15
5/2 22
5/2 23
5/3 2
5/3 6
5/3 10
5/3 14
5/3 17
5/3 23
Time (Month/Day Hour)

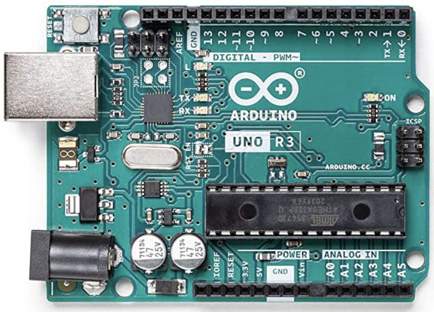

Parked Car - Noise
Distance To Car (cm)
51
50
49
Time
05-03 03
05-03 04
05-03 05
05-03 06
05-03 07
05-03 08
05-03 09
05-03 10
05-03 11
05-03 12

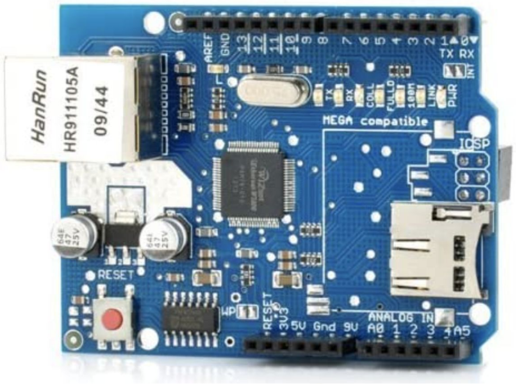

Car Entering and Exiting Spot
3500
3000
2500
2000
1500
1000
500
0
Distance To Car (cm)
5/1 20:40
5/1 20:40
5/1 20:50
5/1 20:59
5/1 21:09
Time

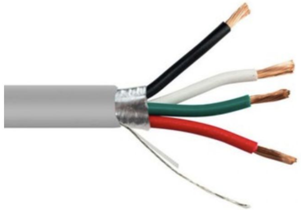

Internet Outage
39.0
38.5
38.0
37.5
37.0
36.5
36.0
35.5
35.0
Distance To Car (cm)
01 21:10
01 21:15
Time

Costs

- Because we had most things, our total costs came to around $40.00 for the ethernet shield, and supplies such as tape and the metal case
- Total costs:
  - Arduino - $22.77
  - Ethernet shield - $20.99
  - Sensor - $3.95
  - Wiring - ~$.60
  - Router - $69.99
  - Misc. - ~ 12.00
- **Total = $130.30**





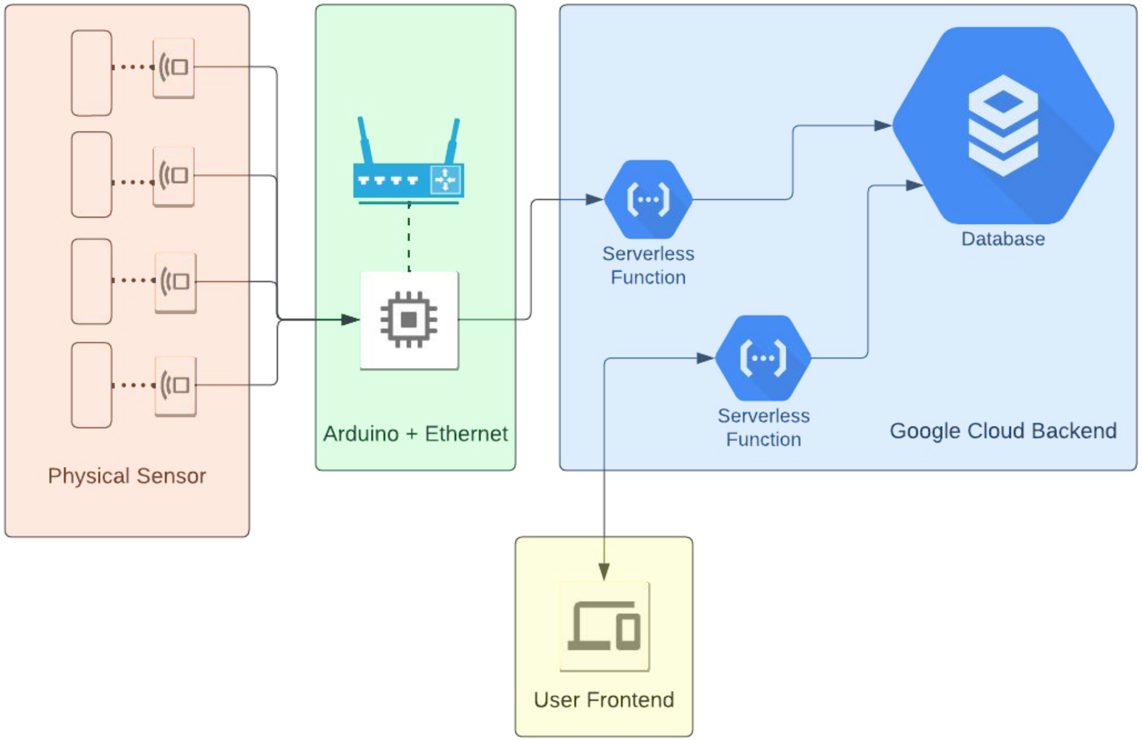


System Scalability

Our smart-parking application was designed for flexibility and scalability.

- Software
  - The code is entirely reusable
  - Agnostic to all hardware and number of spots (unique *spot_id* required for each)
  - To improve the user interface for multiple spots, more code could be written (not necessary)
- Cloud Infrastructure
  - Compute is executed by “serverless” functions, in which Google Cloud handles scaling of infrastructure with increased traffic/activity
  - Database is designed to handle variable number of spots
- Hardware
  - Each spot requires one new ultrasonic sensor, a new (custom) mount/weatherproof, and new wiring to the Arduino
  - The most time and money of the three categories for adding a new spot

Proposed System Schematic - Entire Lot (4 Spots)



Estimated Cost - Entire Lot (4 Spots)

- New hardware additions
  - 3 ultrasonic distance sensors
  - 3x the wiring
- Assume cloud infrastructure costs still negligible (amount of data still trivially small)
- Total costs:
  - Arduino - $22.77
  - Ethernet shield - $20.99
  - 4xSensor - $15.80
  - 4xWiring - ~$2.40
  - Router - $69.99
  - Misc. - ~ 12.00
- **Total = $143.95**

Camera Implementation

- Ultrasonic sensors used for fun + experimentation
  - Sensors class, after all!
  - By no means the cheapest or most efficient option
- Camera could provide and display the same information
  - Spot(s) taken or not taken?

For a real-world application, would this be a cheaper implementation?

Camera Implementation #1 - Ring

- Existing companies provide “smart home” technologies for **cheap**
- *Ring* provides seamless installation and integration with their online platform
- With about 15 minutes of setup, a *Ring* security camera could be mounted and configured to live stream the parking lot and monitor spots
- Cost:
  - Camera: $100
  - Subscription: $3/month

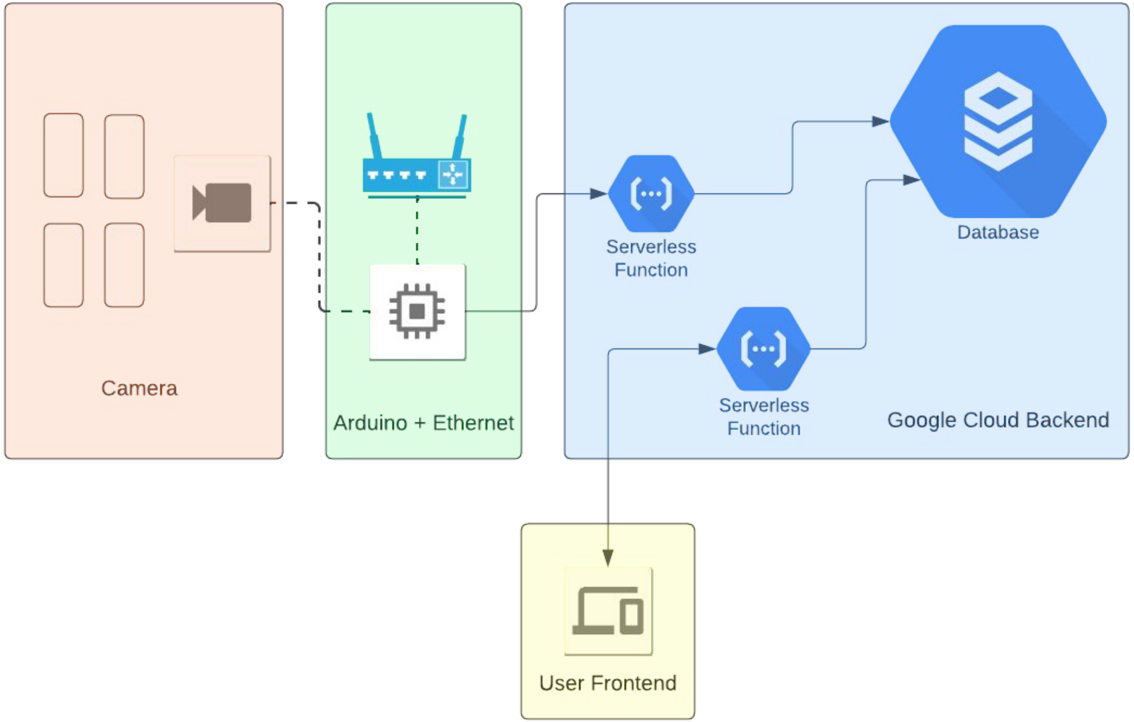
https://ring.com/products/stick-up-security-camera-battery

Camera Implementation #2 - Custom Integration

- Arduino “Camera Shield” takes images controlled by the microprocessor
  - 2 megapixel images
- In use with the “Ethernet Shield”, these images could be uploaded to the internet, similarly to how ultrasonic sensor distances are uploaded now
- The price of data storage, however, will increase
  - Image data much larger than numerical data

https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/

Camera Implementation #2 - System Schematic



Camera Implementation #2 - Cost

- Fixed Costs
  - Arduino - $22.77
  - Ethernet shield - $20.99
  - Arducam Camera Shield 2: $26
  - Router - $69.99
  - Total: **$139.75**
- Data Storage (assume one image a minute, each image is stored):
  - 2 megapixels = 2,000,000 pixels
  - 3 colors, 8-bit (0-255) depth per pixel
  - 2,000,000 * 3 * 8 = 48,000,000 bits = 6 Mb per image
  - 30 days/month * 24 hours/day * 60 minutes/hour * 1 image/hour = 43,200 images/month
  - 6 Mb/image * 43,200 images/month = 259,200 Mb/month = 259.2 Gb/month
  - GCP storage: $0.023/Gb (https://cloud.google.com/storage/pricing, *us-east4*)
  - 259.2 Gb/month * $0.023/Gb = **$5.96/month**

Smart Parking Implementations - Comparison (4 Spots)

	Fixed Cost	**Monthly Cost**	**Estimated Integration Time**	**Cloud Infrastructure Required**
Ultrasonic Distance Sensor	$143.95	$0	4 hours	Yes
Camera #1 - Ring	$100	$3	15 minutes	No
Camera #2 - Custom	$139.75	$5.96	1 hour	Yes

Retrospective (1 Spot)

- Application live for ***2 weeks***
- ***20,000 data points*** collected and stored
- Robust and successful physical integration
  - Mounting
  - Wiring
  - Weatherproofing
- Efficient, inexpensive, and reliable software implementation
  - $20 of Google Cloud Platform credits used
  - Zero system downtime
- Useful product for our housemates
  - “I used this site every day - it saved me easily 20 minutes each week”

Full Signal (May 1 - May 15)