IoT Smart Parking System

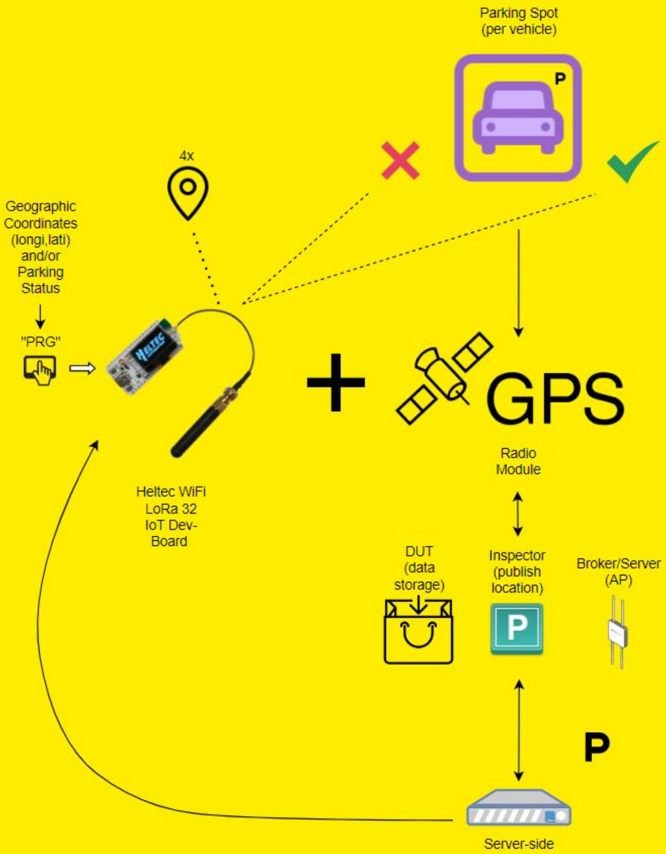
~ Project Demo ~

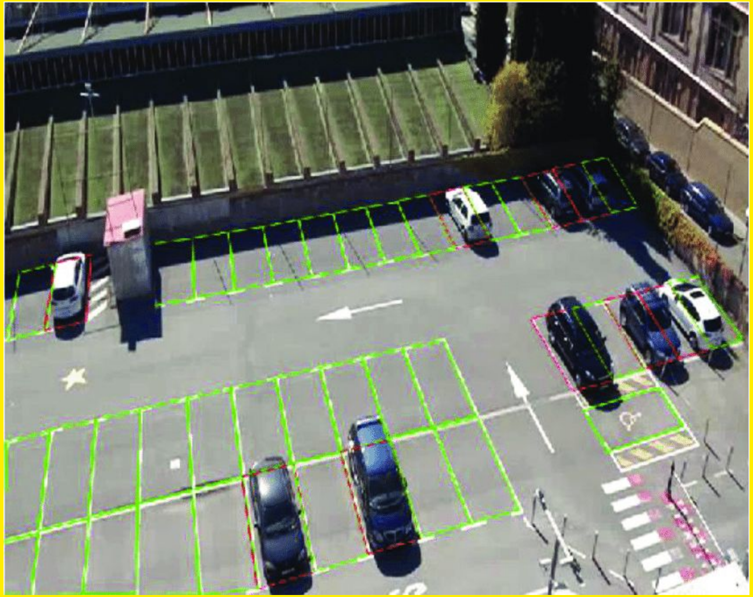
By: James Zhu, Abel Ressom,
and Josh Lee

Story/Motive

- Efficient for cities to find spots within a lot instead of looking around (especially when it’s packed)
- Account for a fifth of all car accidents which are parking-related, this method would bring organization[4]
- Interest for the Product:
    - Parking management companies (like Impark)
    - City government agencies in DC, NoVA, Richmond, Cville, etc.
    - Landowners
    - IoT Consulting Service Groups
- Improves managing of parking resources, solves the issue of find parking in busy/congested cities, provides real-time parking spot availability data, etc.

Concept Diagram

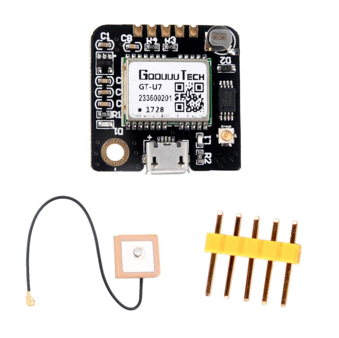

Parking Spot
(per vehicle)
P
4x
Geographic Coordinates (longi,lati) and/or Parking Status
"PRG"
+
GPS
Radio Module
Heltec WiFi LoRa 32 IoT Dev-Board
DUT (data storage)
Inspector (publish location)
Broker/Server (AP)
P
P
Server-side



Materials/Tools

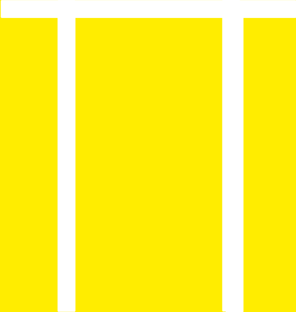
Equipment

- Laptops
- 3x Heltec LoRa V3 Boards (ESP32)
- Goouuu GT-U7 GPS Module
- Car
- Parking spots

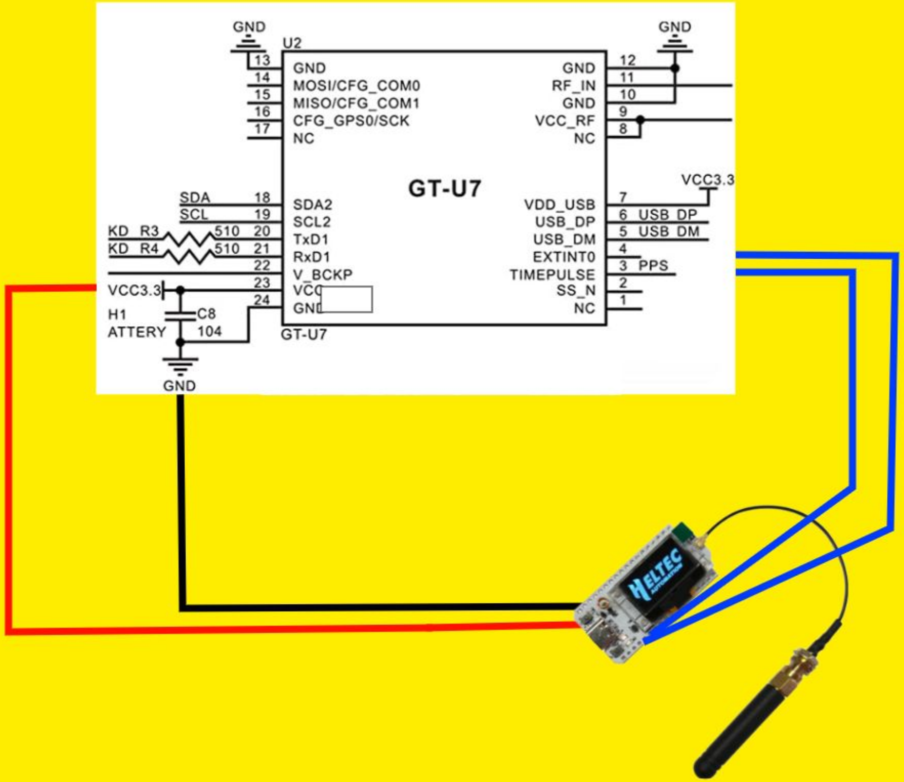


Software/Libraries

- VS Code
- TinyGPSPlus
- Software Serial
- Arduino



Wiring Diagram

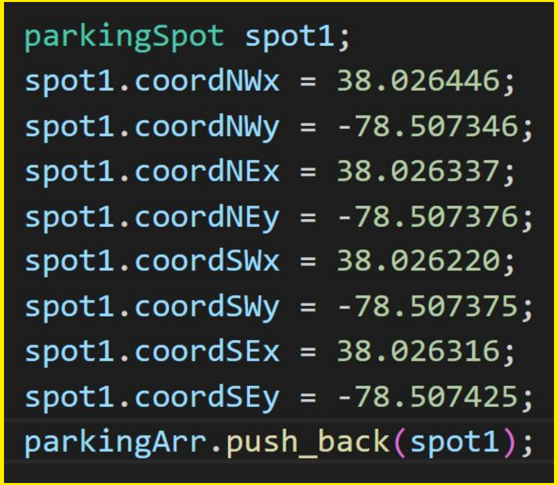


Code

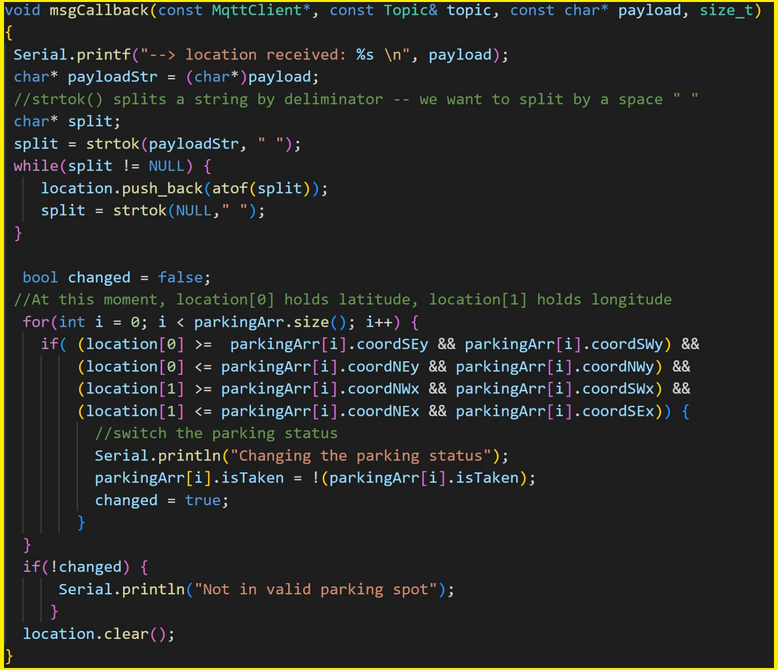
Snippet(s)

Data-storing (DUT)

Dimensioning



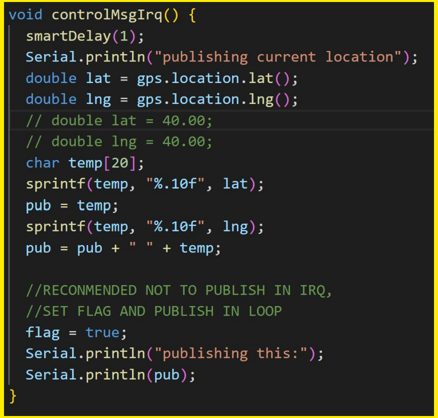
```
parkingSpot spot1;
spot1.coordNWx = 38.026446;
spot1.coordNWy = -78.507346;
spot1.coordNEx = 38.026337;
spot1.coordNEy = -78.507376;
spot1.coordSWx = 38.026220;
spot1.coordSWy = -78.507375;
spot1.coordSEx = 38.026316;
spot1.coordSEy = -78.507425;
parkingArr.push_back(spot1);
```



```
void msgCallback(const MqttClient*, const Topic& topic, const char* payload, size_t)
{
 Serial.printf("--> location received: %s \n", payload);
 char* payloadStr = (char*)payload;
 //strtok() splits a string by deliminator -- we want to split by a space " "
 char* split;
 split = strtok(payloadStr, " ");
 while(split != NULL) {
    location.push_back(atof(split));
    split = strtok(NULL," ");
 }

  bool changed = false;
 //At this moment, location[0] holds latitude, location[1] holds longitude
  for(int i = 0; i < parkingArr.size(); i++) {
    if( (location[0] >=  parkingArr[i].coordSEy && parkingArr[i].coordSWy) &&
        (location[0] <= parkingArr[i].coordNEy && parkingArr[i].coordNWy) &&
        (location[1] >= parkingArr[i].coordNWx && parkingArr[i].coordSWx) &&
        (location[1] <= parkingArr[i].coordNEx && parkingArr[i].coordSEx)) {
          //switch the parking status
          Serial.println("Changing the parking status");
          parkingArr[i].isTaken = !(parkingArr[i].isTaken);
          changed = true;
        }
  }
  if(!changed) {
      Serial.println("Not in valid parking spot");
    }
  location.clear();
}
```

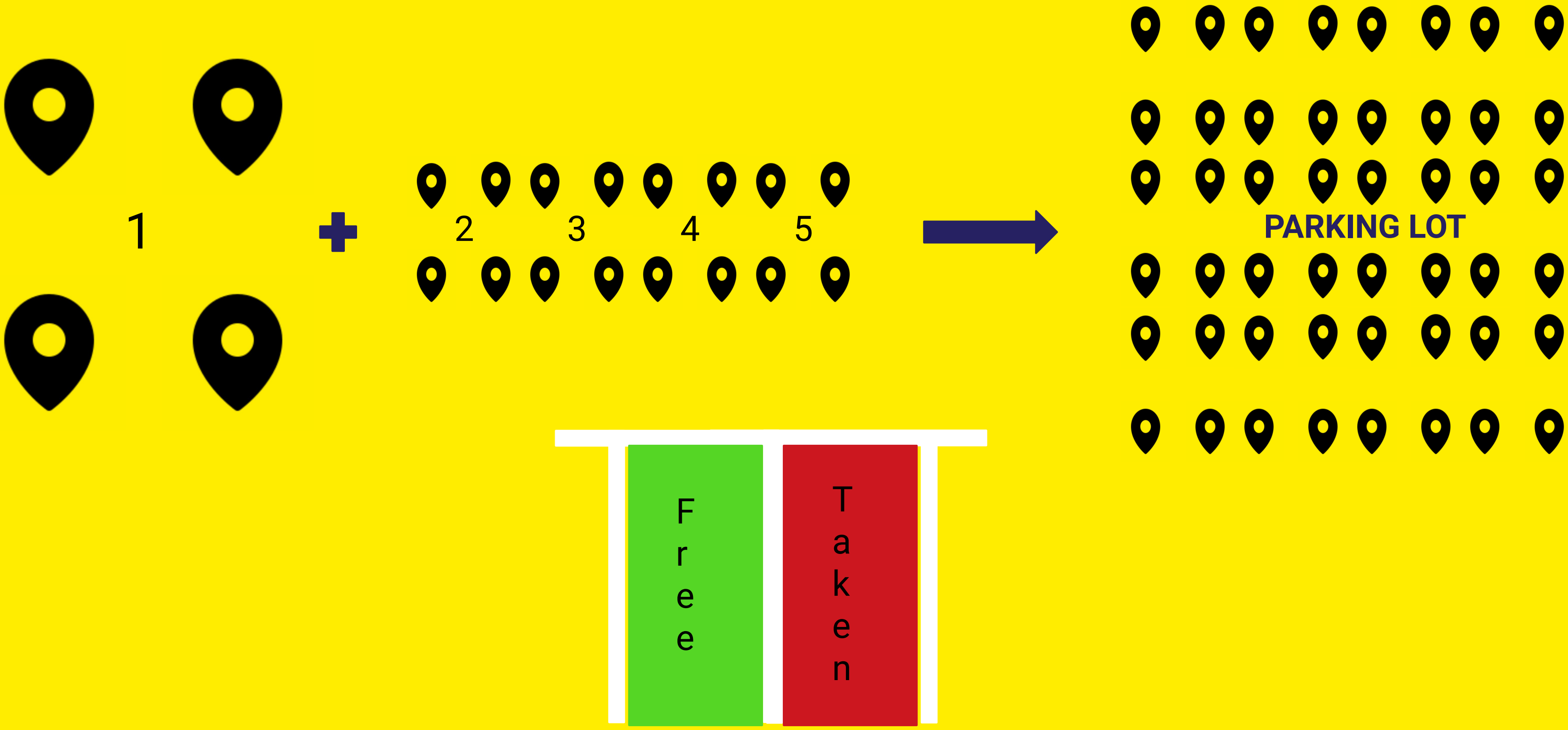
Publishing (Inspector)



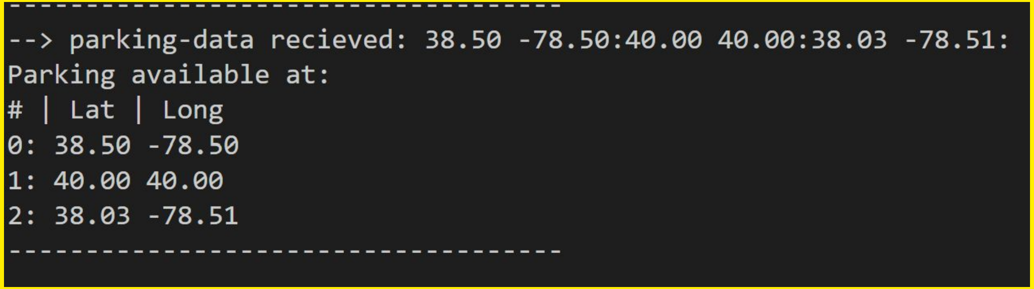
```
void controlMsgIrq() {
  smartDelay(1);
  Serial.println("publishing current location");
  double lat = gps.location.lat();
  double lng = gps.location.lng();
  // double lat = 40.00;
  // double lng = 40.00;
  char temp[20];
  sprintf(temp, "%.10f", lat);
  pub = temp;
  sprintf(temp, "%.10f", lng);
  pub = pub + " " + temp;

  //RECONMENDED NOT TO PUBLISH IN IRQ,
  //SET FLAG AND PUBLISH IN LOOP
  flag = true;
  Serial.println("publishing this:");
  Serial.println(pub);
}
```

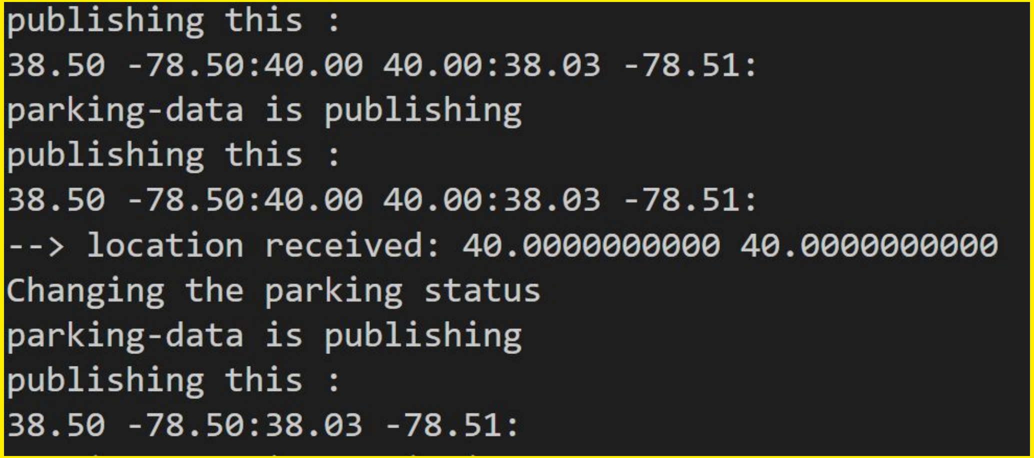
Implementation



Output



```
------------------------------------
--> parking-data recieved: 38.50 -78.50:40.00 40.00:38.03 -78.51:
Parking available at:
# | Lat | Long
0: 38.50 -78.50
1: 40.00 40.00
2: 38.03 -78.51
------------------------------------
```



```
publishing this :
38.50 -78.50:40.00 40.00:38.03 -78.51:
parking-data is publishing
publishing this :
38.50 -78.50:40.00 40.00:38.03 -78.51:
--> location received: 40.0000000000 40.0000000000
Changing the parking status
parking-data is publishing
publishing this :
38.50 -78.50:38.03 -78.51:
```

Work Division

- One person worked on the implementing TinyGPS-related gadget and getting the coordinates on the development board
- Another person worked on the DUT/broker
- Finally, the last person worked on implementing the button press so that the coordinates are sent (Inspector)

In terms of general project vision/ideas, we worked collaboratively

Future Works/Ideas

Some future improvements/implementations include:

- Rather than requiring every car to have a LoRa board, make the project compatible with WiFi through phones rather than just the MQTT protocol
- Incorporate some sort of Map APIs such as Google Maps, MapBox, or OpenStreetMap
- Use more embedded sensory data, e.g. image/force sensors for better recognition and even aiding parking instead of having to use a physical button
- Account for more parking shapes/geometry

References

1. *Car Parking Business Plan | Google Slides & PowerPoint*. (n.d.). Retrieved May 3, 2023, from https://slidesgo.com/theme/car-parking-business-plan#search-parking&position-1&results-23&rs=search
2. *Guide to NEO-6M GPS Module Arduino | Random Nerd Tutorials*. (2018, January 4). https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/
3. Hart, M. (2023). *TinyGPSPlus* [C++]. https://github.com/mikalhart/TinyGPSPlus/blob/ca29434514a5c5172bd807af0608df7f296582a2/examples/FullExample/FullExample.ino (Original work published 2013)
4. *Why hundreds are killed in crashes in parking lots and garages every year*. (2016, November 21). https://www.cbsnews.com/news/parking-lot-accidents-distracted-drivers-national-safety-council/