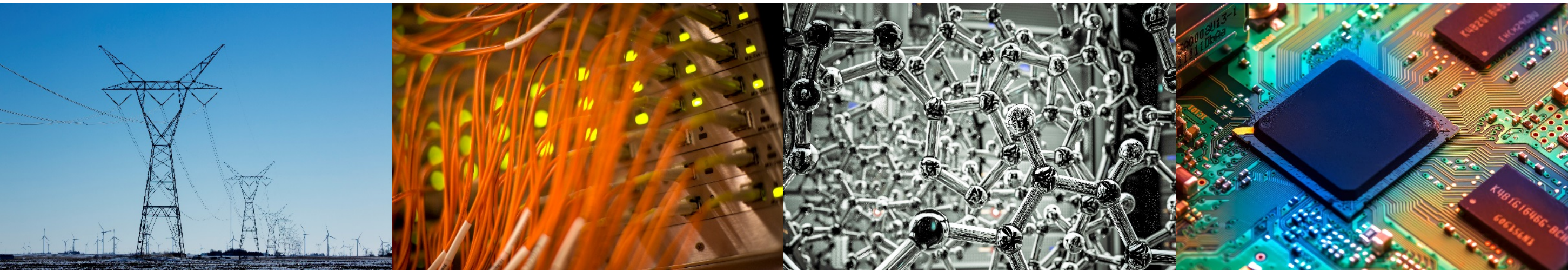


VACANT PARKING DETECTOR 2.0

Team #21: Jiahe Liu, Zeyu Zhang, Qingtao Hu

TA: Zhen Qin

ECE 445 Senior Design, April 30, 2018



I ILLINOIS

Electrical & Computer Engineering

COLLEGE OF ENGINEERING

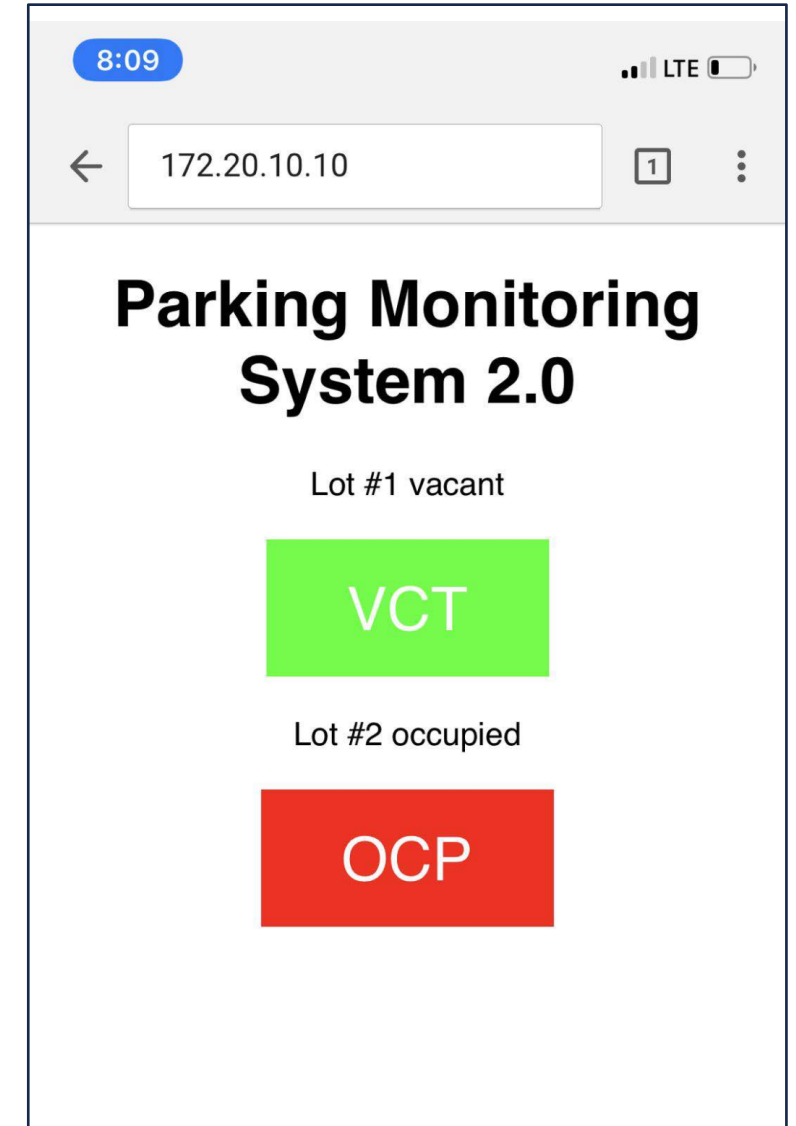
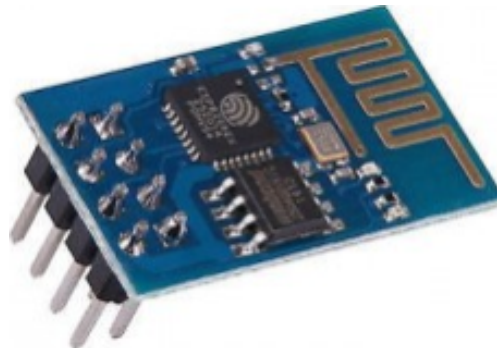
Project Motivation & Objective

- (INRIX) 17 hours and \$345 on average searching for parking in US
- A system detecting available parking spaces helps parking efficiently



Project Introduction

- Using ultrasonic sensor to detect the status of parking space (distance > 2m)
- Transferring the status data to the server via ESP8266 Wi-Fi module
- Check the parking space status using mobile device

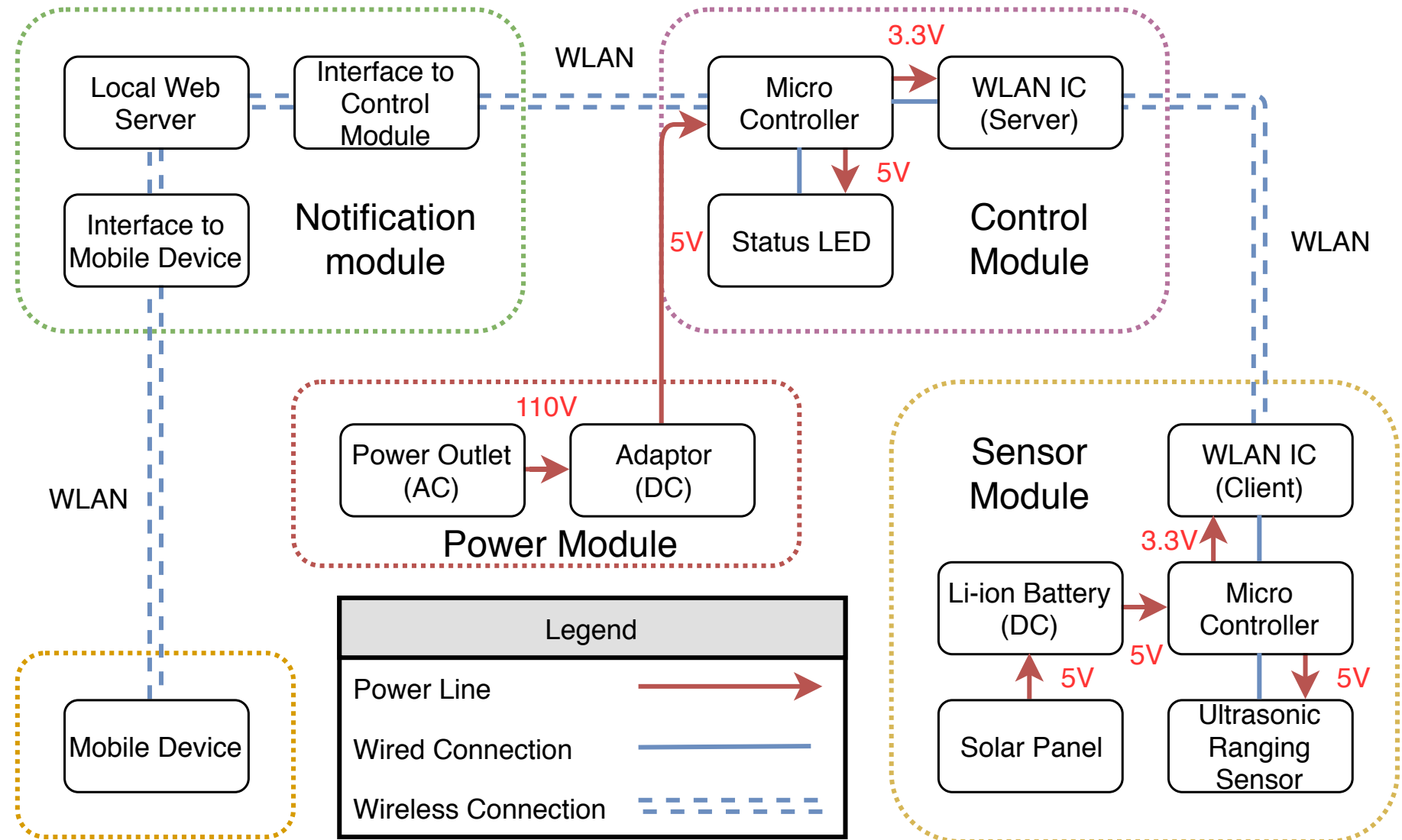


Project Comparison

	Our Project	Team #39 FA17	Improvement
Data Transmission	• Wi-Fi	• Radio Frequency	• Less interference • Higher speed
Proximity Sensor	• Ultrasound	• Infrared	• Wider range • More reliable • Cheaper
Parking Space Status Information	• LED • Webpage	• LED	• Use mobile device to access parking info

Design Overview

- 3 Hardware Parts:
 - Sensor Module
 - Control Module
 - Power Module
- 2 Software Parts:
 - Notification Module
 - Mobile Device
- 1 Goal:
 - Make parking easier



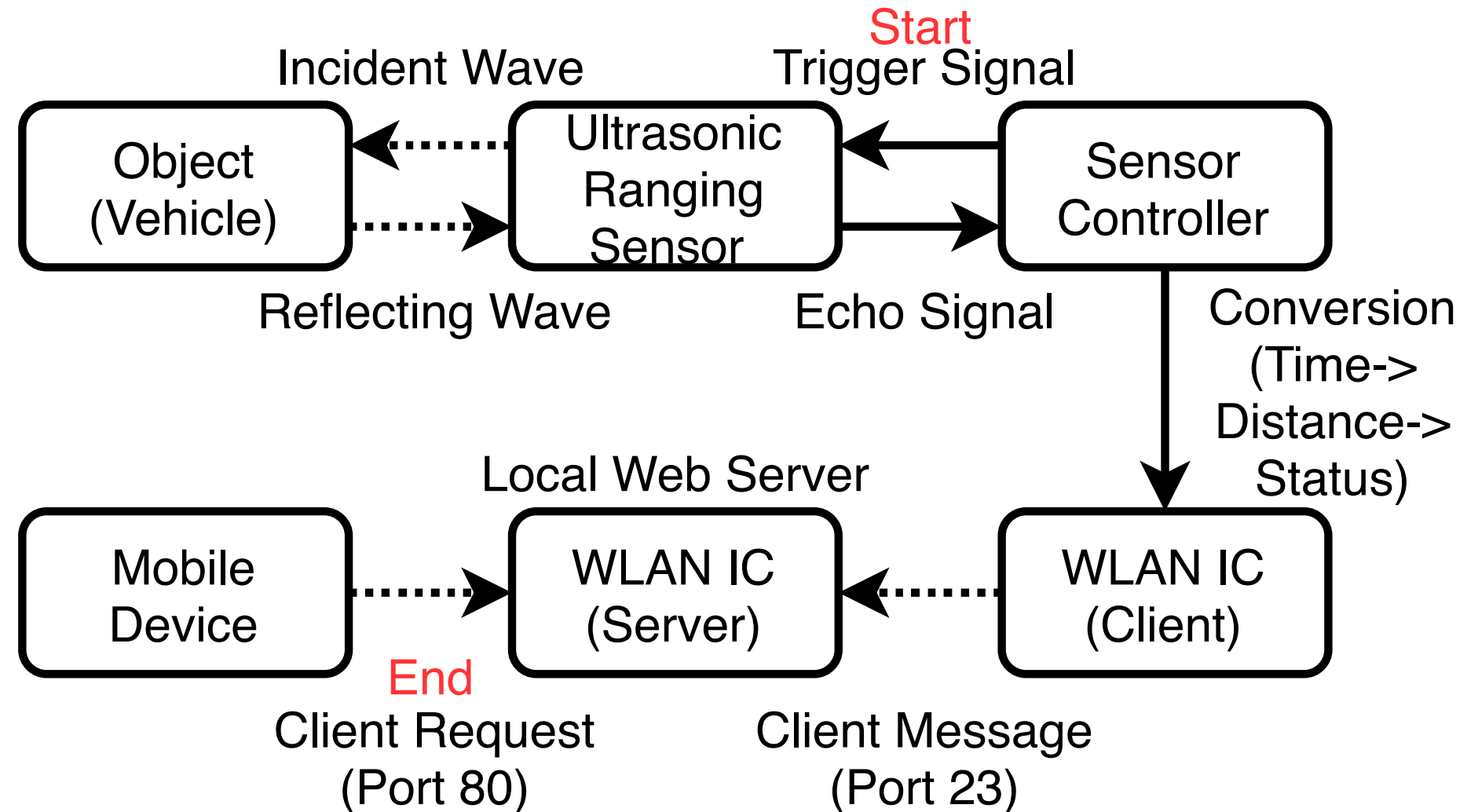
Design Overview

■ Data Flowchart:

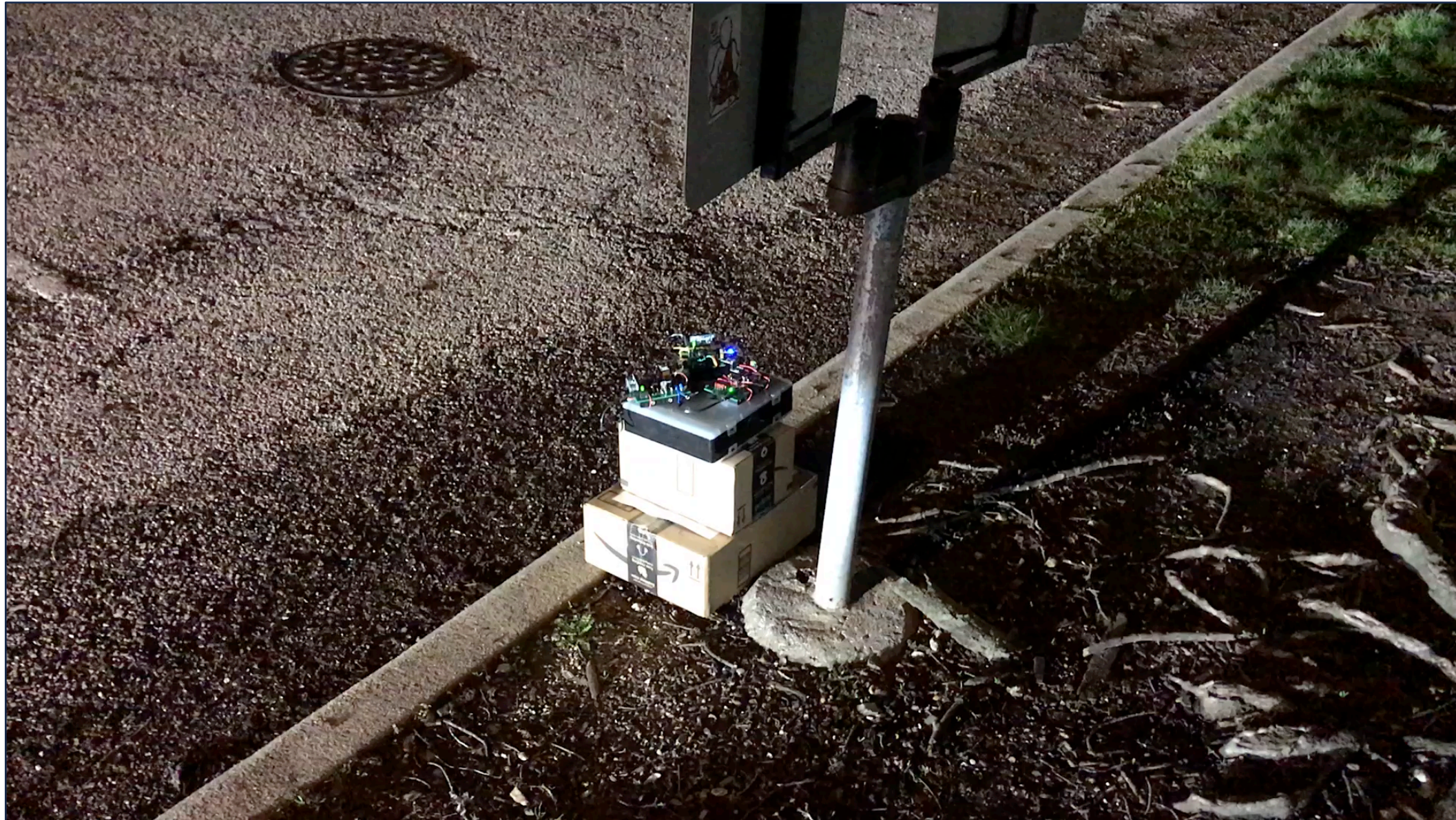
- Start: Sensor MCU initiates detection sequence
- End: Mobile device browses local webpage for parking info

■ Data Format:

- Space Index + Space status

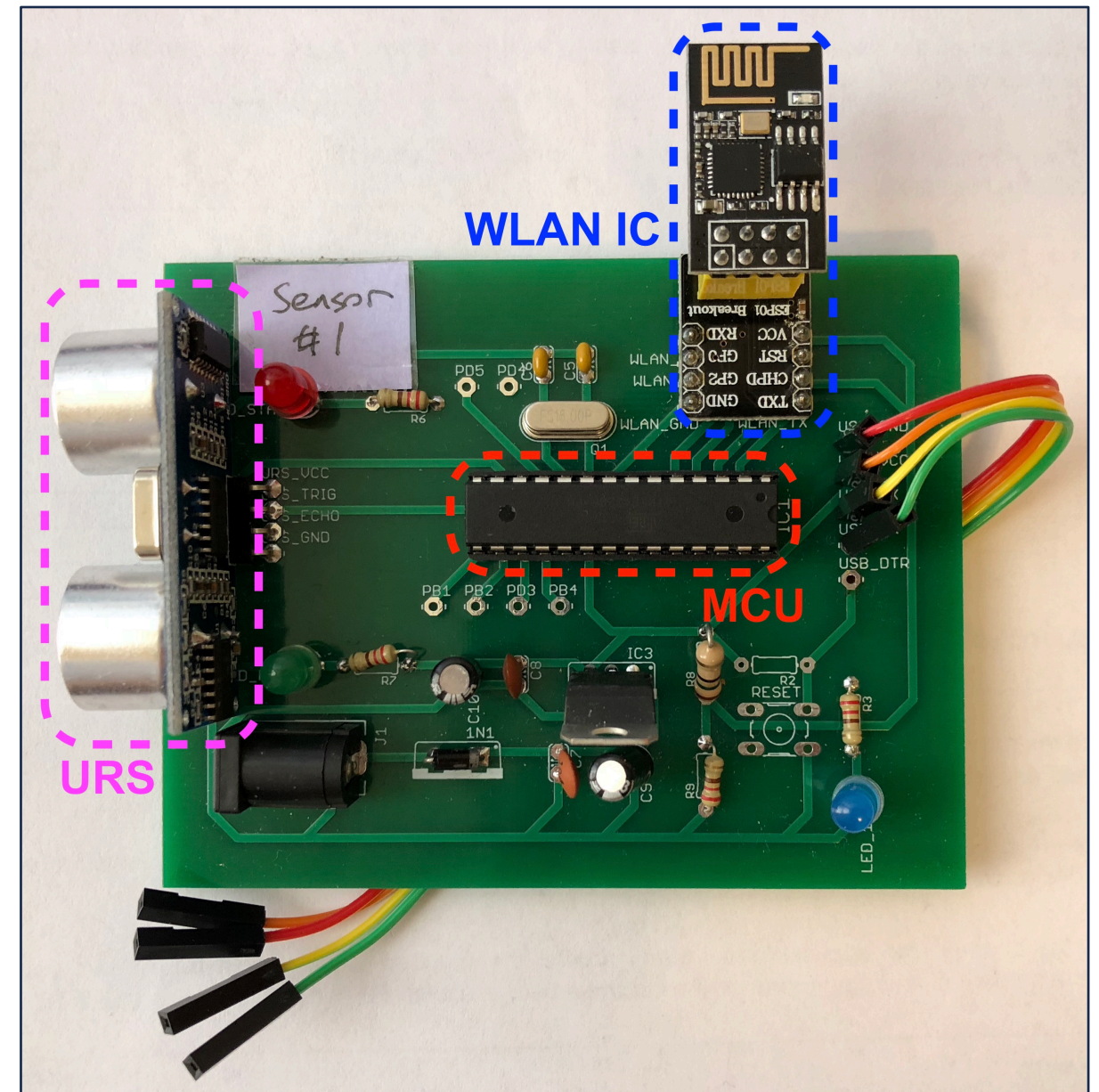


Project Demonstration

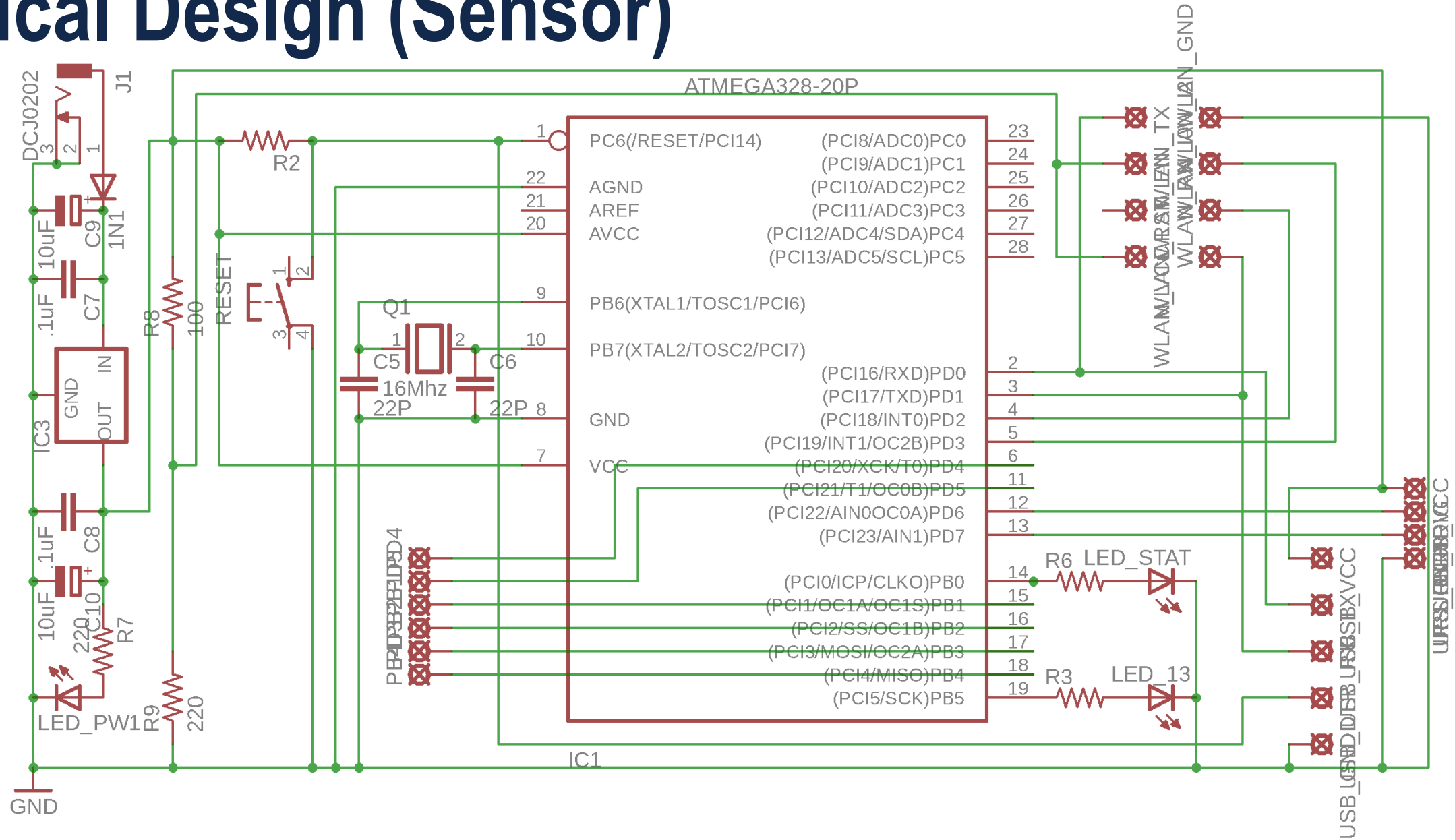


Physical Design (Sensor)

- Sensor Module:
 - Ultrasonic Ranging Sensor: **HC-SR04**
 - Micro Controlling Unit: **ATmega328P-PU**
 - WLAN IC: **ESP-01S**
 - Circuit Layout: **Arduino Clone Board**
- Functionalities:
 - Detect stationary vehicle within duty range
 - Report space status to Control Module via WLAN

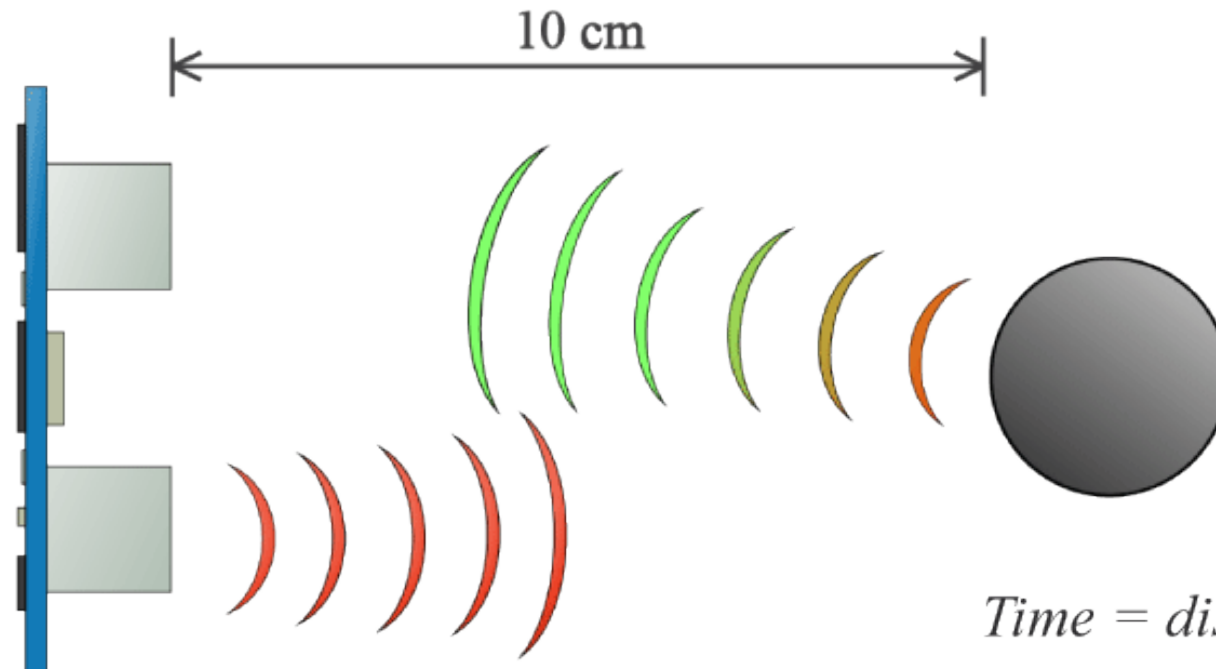


Physical Design (Sensor)



Physical Design (Sensor)

- URS Detection Schematic:
- Time Difference:
 - Round trip
- Object Distance:
 - 2m threshold
- Parking Status:
 - $d < 2\text{m}$: occupied
 - $d \geq 2\text{m}$: vacant



speed of sound:

$$v = 340 \text{ m/s}$$

$$v = 0,034 \text{ cm}/\mu\text{s}$$

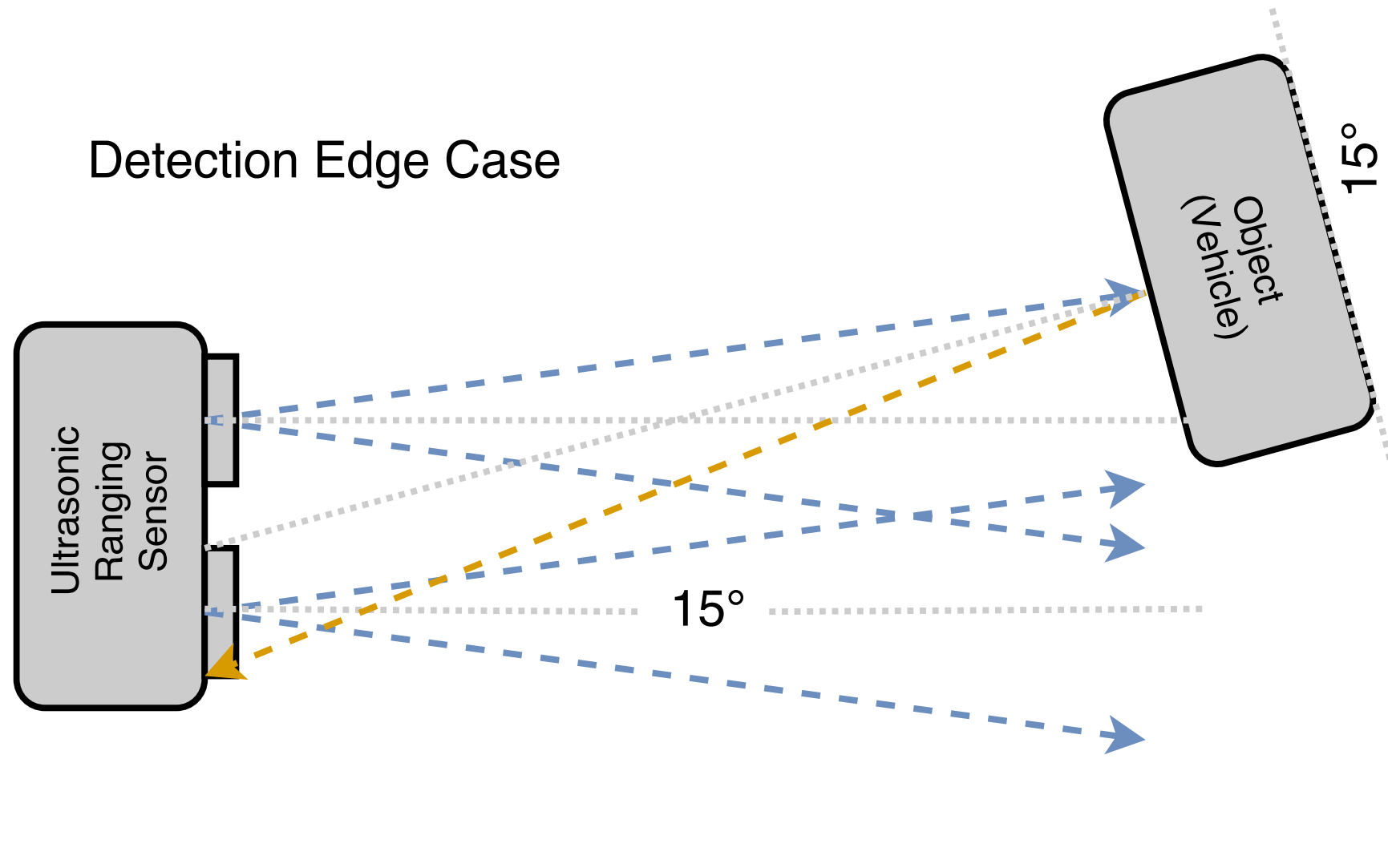
Time = distance / speed:

$$t = s / v = 10 / 0,034 = 294 \mu\text{s}$$

Distance:

$$s = t \cdot 0,034 / 2$$

Physical Design (Sensor Tolerance)

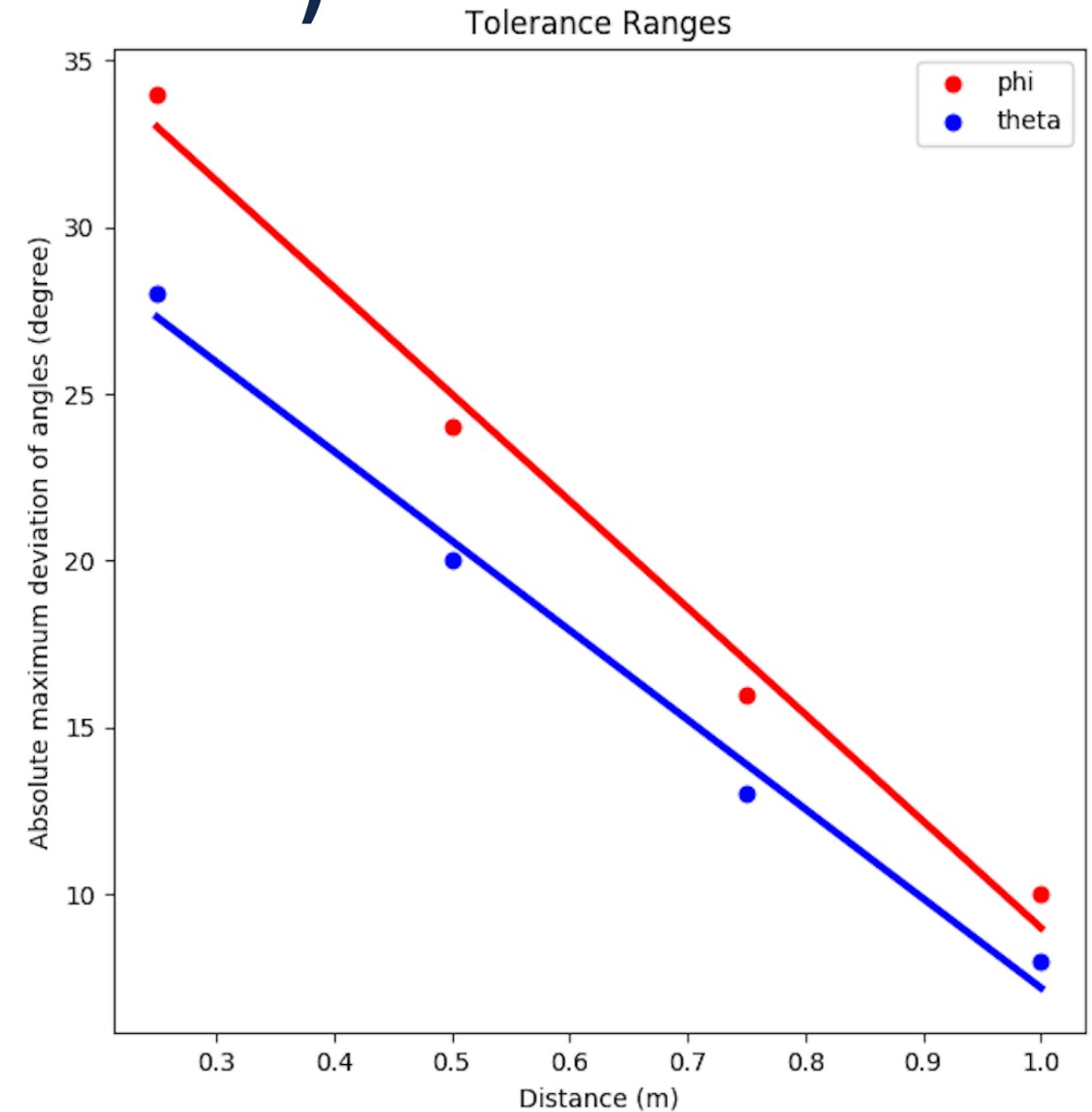


Physical Design (Sensor Tolerance)

Distance (d)	Horizontal angle (ϕ)	Vertical angle (θ)
0.25m	+/-34°	+/-28°
0.5m	+/-24°	+/-20°
0.75m	+/-16°	+/-13°
1m	+/-10°	+/-8°

$$|\phi| = -32 * d + 42$$

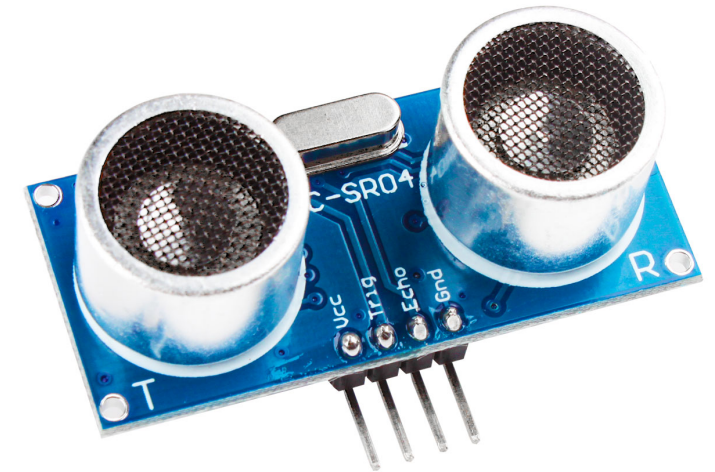
$$|\theta| = -26.67 * d + 34.67$$



Physical Design (URS R&V)

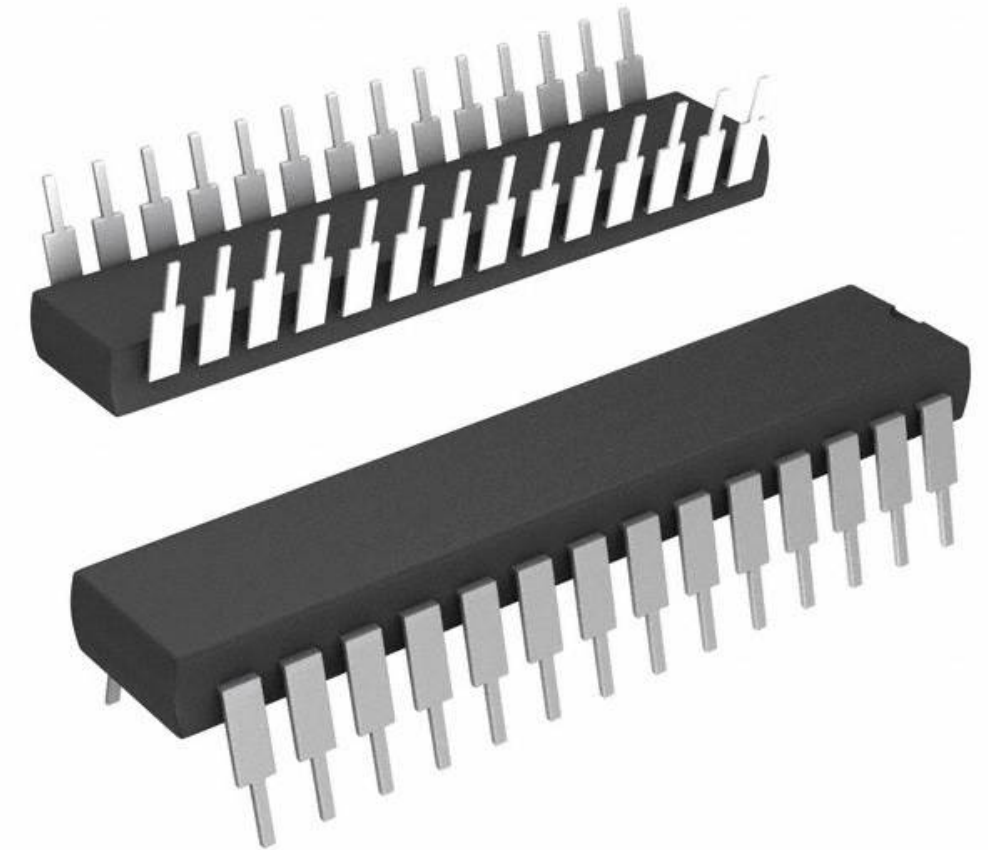
- Requirement:
 - *Detect the vehicle within 2m range*
- Verification:
 - Move a plate between 0.5m and 2.5m away from the URS
 - Compare the distance reading from URS to physical measurements

			
0.3cm	<15	<2mA	2~450cm
Resolution	Effectual Angle	Quiescent Current	Detection Distance



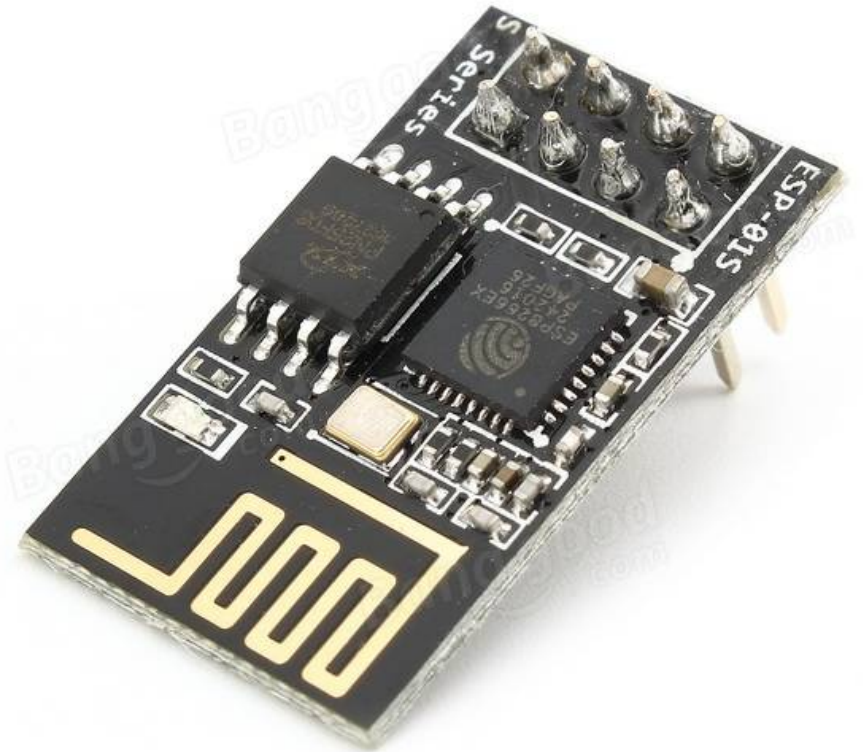
Physical Design (Sensor MCU R&V)

- Requirement:
 - *Convert time difference into space status with $\sim 0.5\text{Hz}$ refresh rate*
- Verification:
 - Access both signals from URS and MCU
 - Perform manual conversions and compare results to MCU output
 - Inject a 0.5Hz time-varying signal to the MCU and see if output updates timely



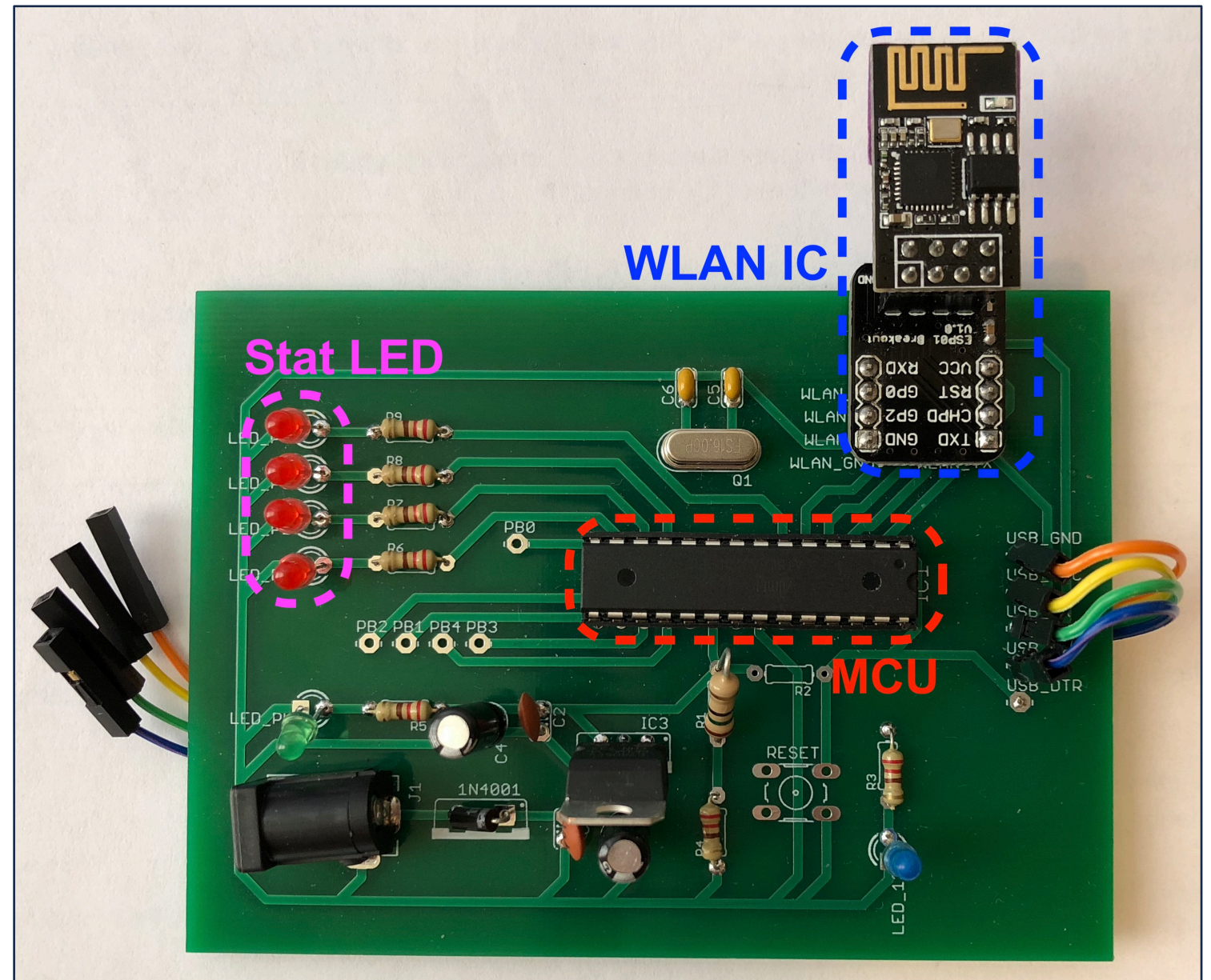
Physical Design (Client WLAN IC R&V)

- Requirement:
 - *Send message to Server WLAN IC with $\sim 0.5\text{Hz}$ refresh rate*
- Verification:
 - Inject a 0.5Hz time-varying signal to the Client WLAN IC
 - Check client message in Server WLAN IC to see if it updates timely

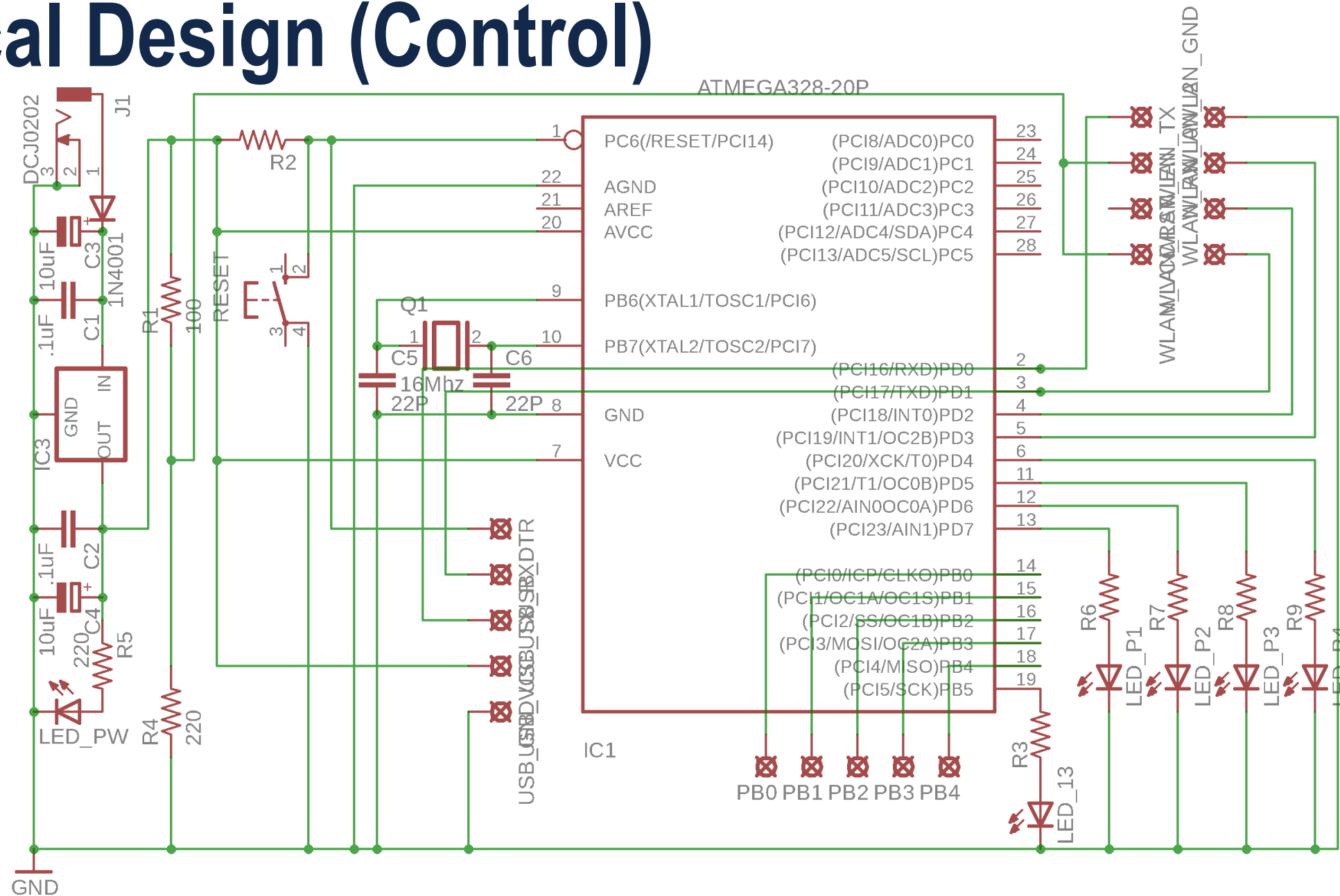


Physical Design (Ctrl)

- Control Module:
 - Micro Controlling Unit: **ATmega328P-PU**
 - WLAN IC: **ESP-01S**
 - Circuit layout: **Arduino Clone Board**
- Functionalities:
 - Gather space status from multiple Sensor Modules
 - Display status on a local webserver



ECE ILLINOIS

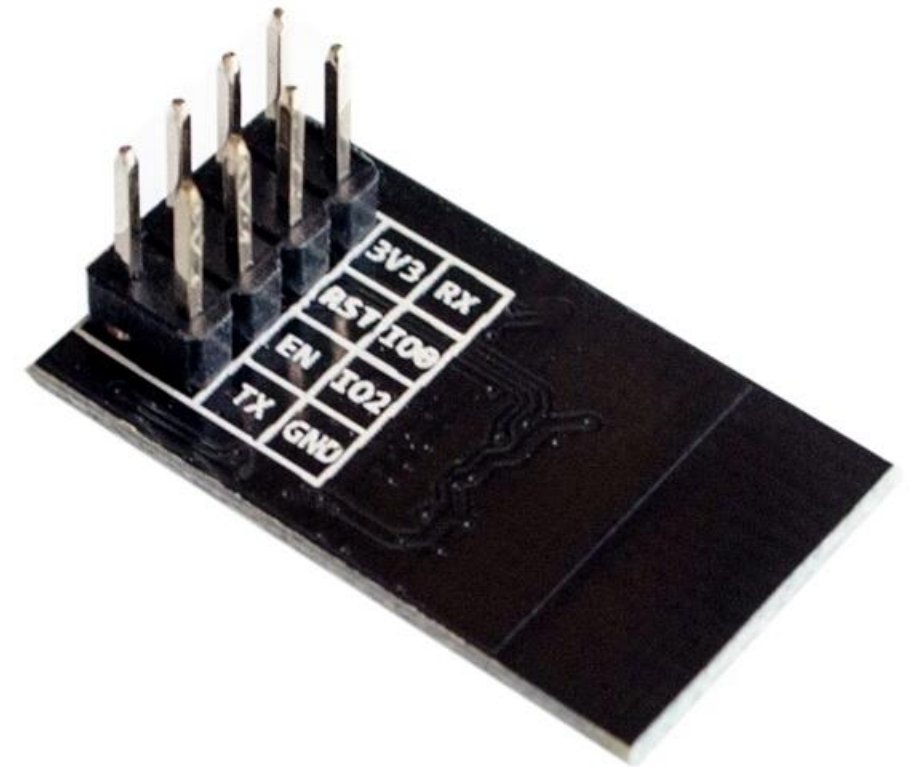


Physical Design (Control R&V)

- Requirement:
 - *Connect to all Sensor Module clients within range $\leq 25\text{m}$ via WLAN*
- Verification:
 - After configuration, place two WLAN ICs 25m apart
 - Confirm two-way wireless transmissions using predefined data packet

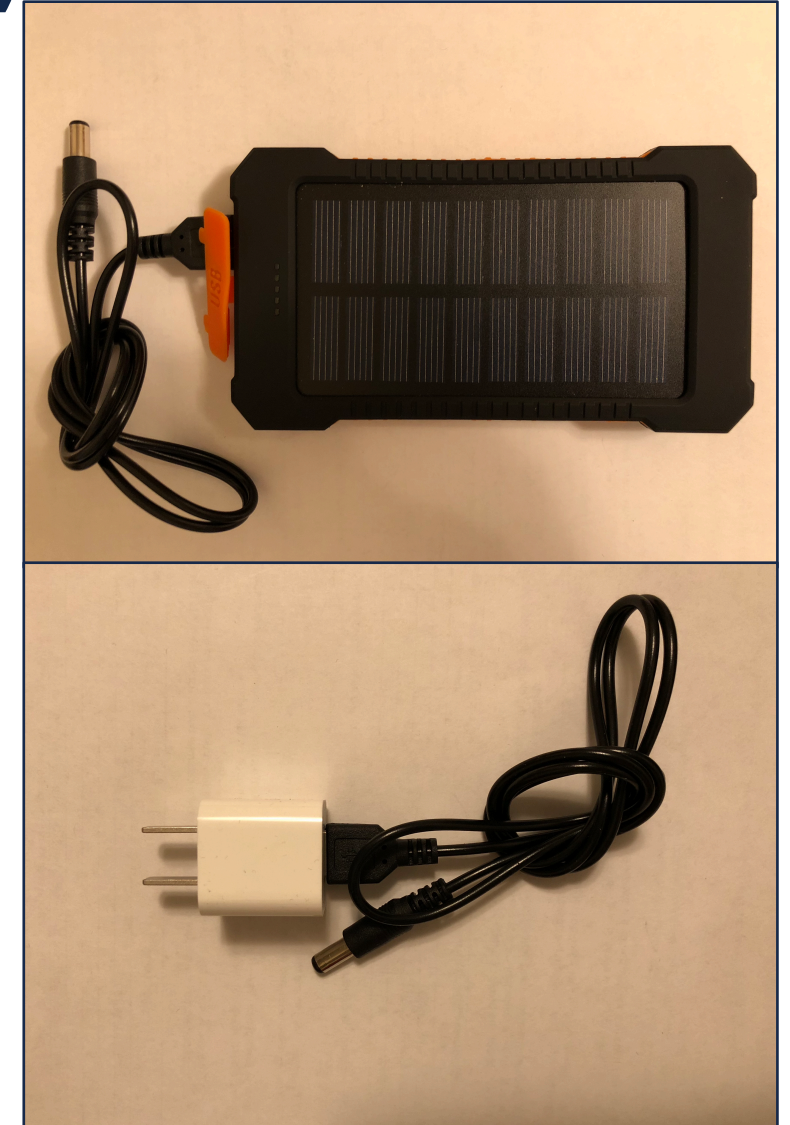
Physical Design (Server WLAN IC R&V)

- Requirement:
 - *Process data from different client WLAN ICs with $\sim 0.5\text{Hz}$ refresh rate*
- Verification:
 - Send a 0.5Hz time-varying signal from another WLAN IC to it
 - Monitor its serial ports to see if output updates timely



Physical Design (Peripheral and R&V)

- Off-the-shelf Hardware:
 - Solar Panel + Li-Po Battery
 - 5V-1A Power Adaptor
- Requirement:
 - *They must supply $5V \pm 0.1V$ and $<1A$ to support module normal operation*
- Verification:
 - Connect the power supply to a 10k Ω resistor
 - Measure resistor voltage and current



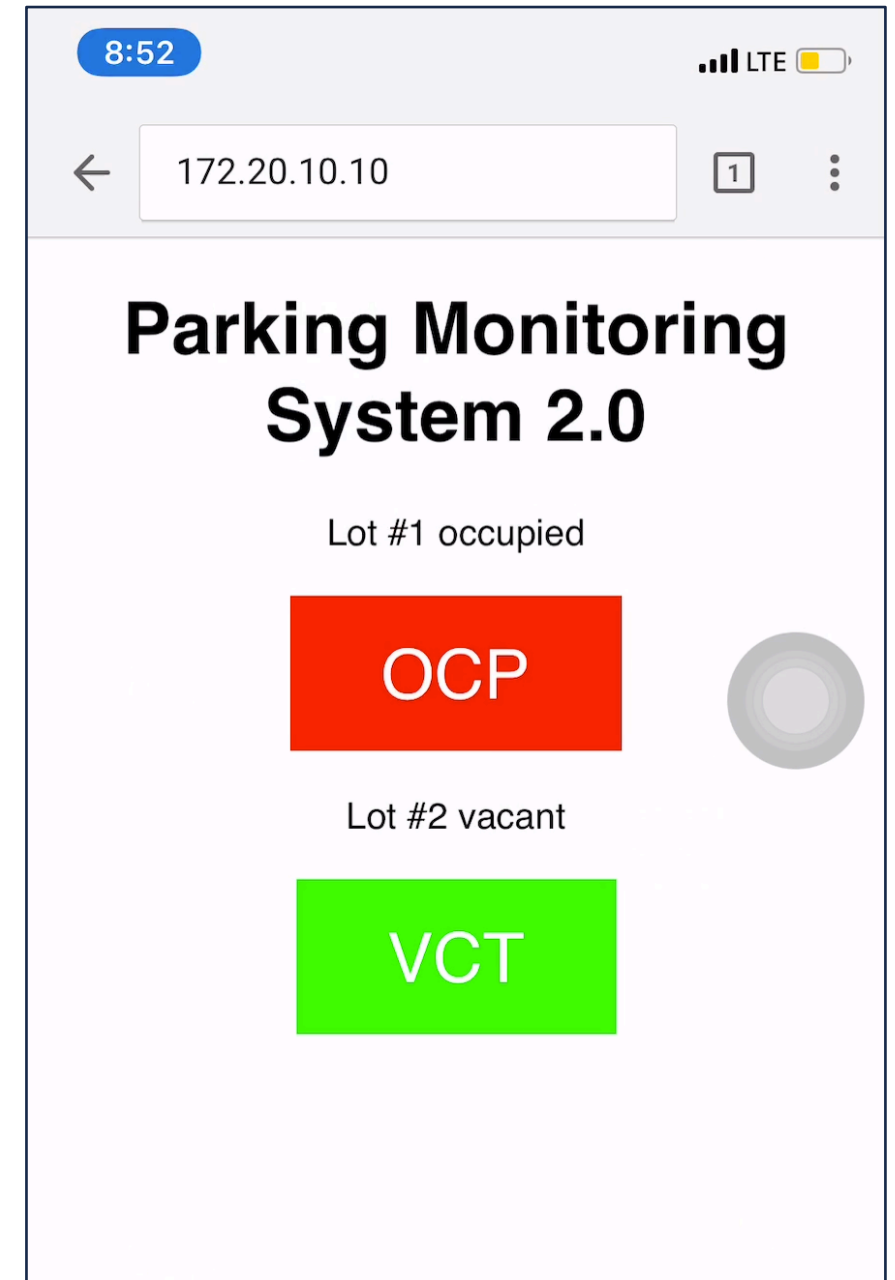
Software Design (Notification)

- Local Web Server:
 - Ambient Network Dependency: Require outside Wi-Fi router
 - Static IP Address: Easier access
 - Dual Ports Setup: 23 for WLAN IC, 80 for mobile device
 - Local Webpage: Display parking status message in graphics

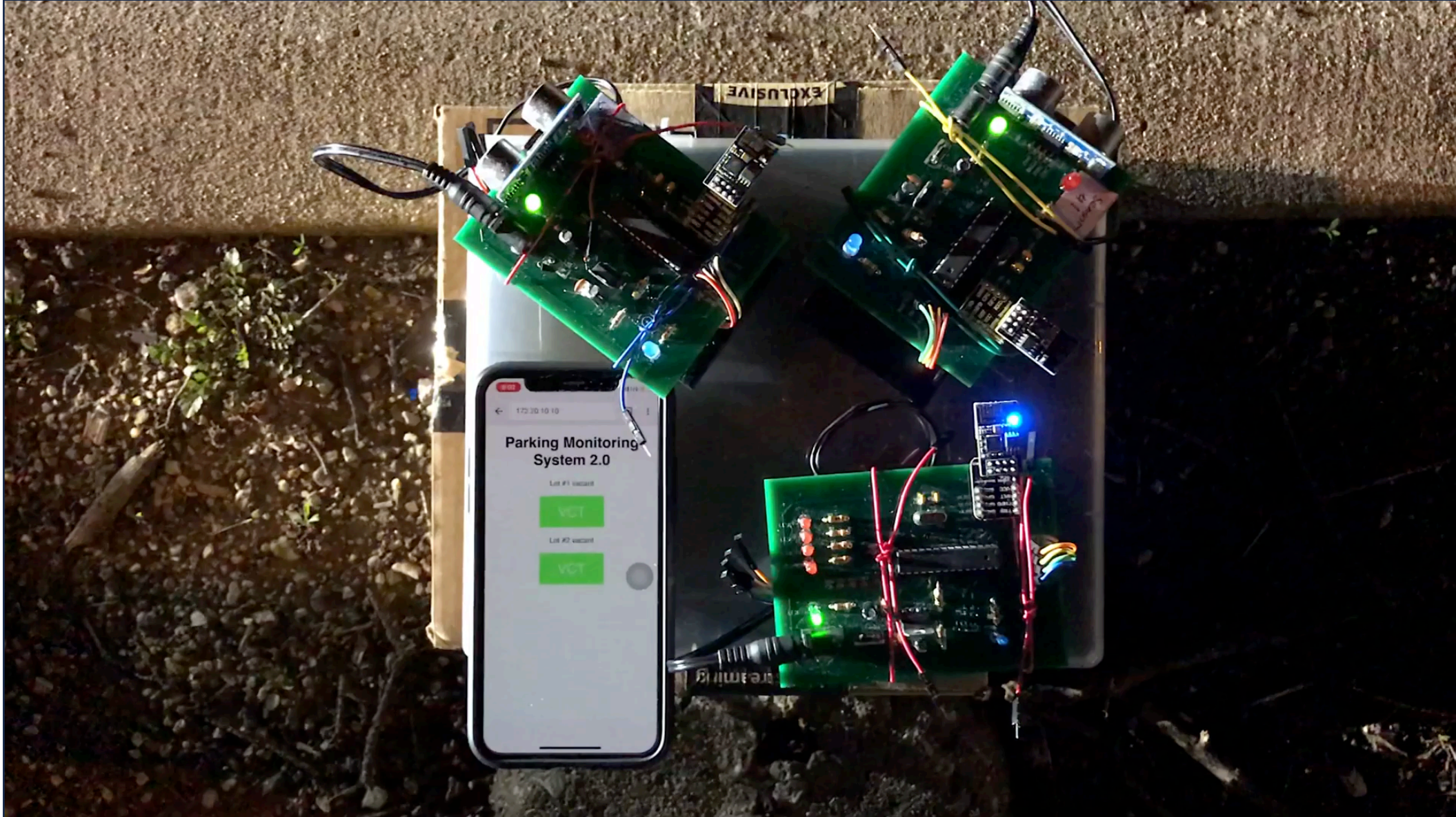
```
10 byte ledPin = 2;
11 char ssid[] = "Jiahe";
12 char pass[] = "zhangzeyu";
13 // esp server
14 WiFiServer server(23);
15 // web server
16 WiFiServer server_(80);
17
18 IPAddress ip(172,20,10,10);
19 IPAddress gateway(172,20,10,1);
20 IPAddress subnet(255,255,255,0);
21
22 String stat_1 = "vacant";
23 String stat_2 = "vacant";
24 String stat_3 = "vacant";
25 // Variable to store the HTTP re
```

Software Design (Mobile Device)

- Access Local Webpage:
 - Use the static webserver IP address
 - Parking status displayed in both text and graphic
 - Refresh the webpage to get updated parking info

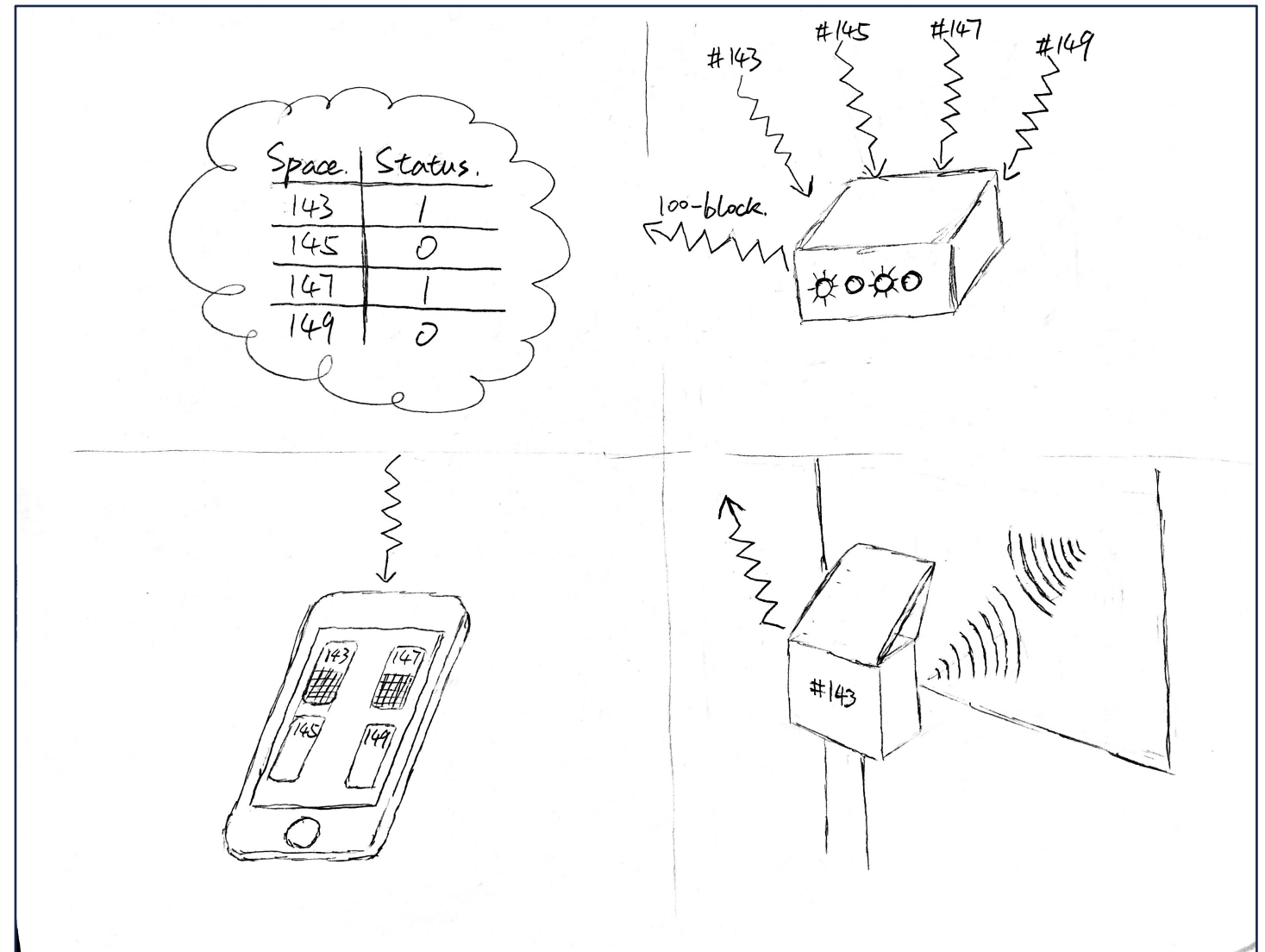


Software Demonstration



Project Development

- Success:
 - Unit Tests, Modular Integrations, and of course the Final Demo
- Challenge:
 - Transmits Data From Control Module To User



Project Development

- Method 1: Run python scripts to manage data locally and send them to user via Android
 - Pro: Better UI
 - Con: Static information update, Heavy workload, Permission issue

```
1  import serial
2  import re
3
4  parking = {}
5  client = 0
6  status = 0
7
8  # search for serial port
9  ser = serial.Serial('/dev/cu.usbserial-A9M9DV3R', 115200)
10 while(1):
11     print(ser.readline())
12     info = str(ser.readline())
13     # extract parking slot id
14     c = re.search('Client{(.+?)}', info)
15     if c:
16         client = c.group(1)
17         # extract parking status
18         s = re.search('Status{(.+?)}', info)
19         if s:
20             status = s.group(1)
21             parking[client] = status
22
23     output_f = open('parking.txt', 'w')
24     # update all the parking information for every loop
25     for cli in parking.keys():
26         if parking[cli] == '0':
27             output_f.write('Slot: '+cli+' '+Unoccupied'+'\r\n')
28         else:
29             output_f.write('Slot: '+cli+' '+Occupied'+'\r\n')
30     output_f.close()
```

Project Development (Method 1 Demo)

ECE445Dev - [C:\Users\hqt95\Desktop\ECE445Dev] - [app] - ...\app\src\main\java\com\example\hqt95\ece445dev\MainActivity.java - Android Studio 3.0.1

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ECE445Dev app src main java com example hqt95 ece445dev MainActivity
activity_main.xml ExampleInstrumentedTest.java AndroidManifest.xml MainActivity.java app
Main
  manifests
    AndroidManifest.xml
  java
    com.example.hqt95.ece445dev
      MainActivity
    com.example.hqt95.ece445dev (androidTest)
    com.example.hqt95.ece445dev (test)
  assets
  res
  Gradle Scripts
    build.gradle (Project: ECE445Dev)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
Captures
Build Variants
Favorites
Run TODO Android Profiler Logcat Terminal Messages
Emulator: Process finished with exit code 0 (a minute ago)
42:43 CRLF+ UTF-8+ Context: <no context>
8:06 PM 4/29/2018
```

```
16 TextView tv_text;
17
18
19
20
21
22
23 @Override
24 protected void onCreate(Bundle savedInstanceState) {
25     super.onCreate(savedInstanceState);
26     setContentView(R.layout.activity_main);
27
28     b_read = (Button) findViewById(R.id.b_read);
29
30     tv_text = (TextView) findViewById(R.id.tv_text);
31
32     b_read.setOnClickListener(new View.OnClickListener() {
33         @Override
34         public void onClick(View view) {
35             String text = "";
36             try {
37                 InputStream is = getAssets().open("file.txt");
38                 int size = is.available();
39                 byte[] buffer = new byte[size];
40                 is.read(buffer);
41                 is.close();
42                 text = new String(buffer);
43             } catch (IOException ex) {
44                 ex.printStackTrace();
45             }
46             tv_text.setText(text);
47         }
48     });
49
50 }
```


Project Development

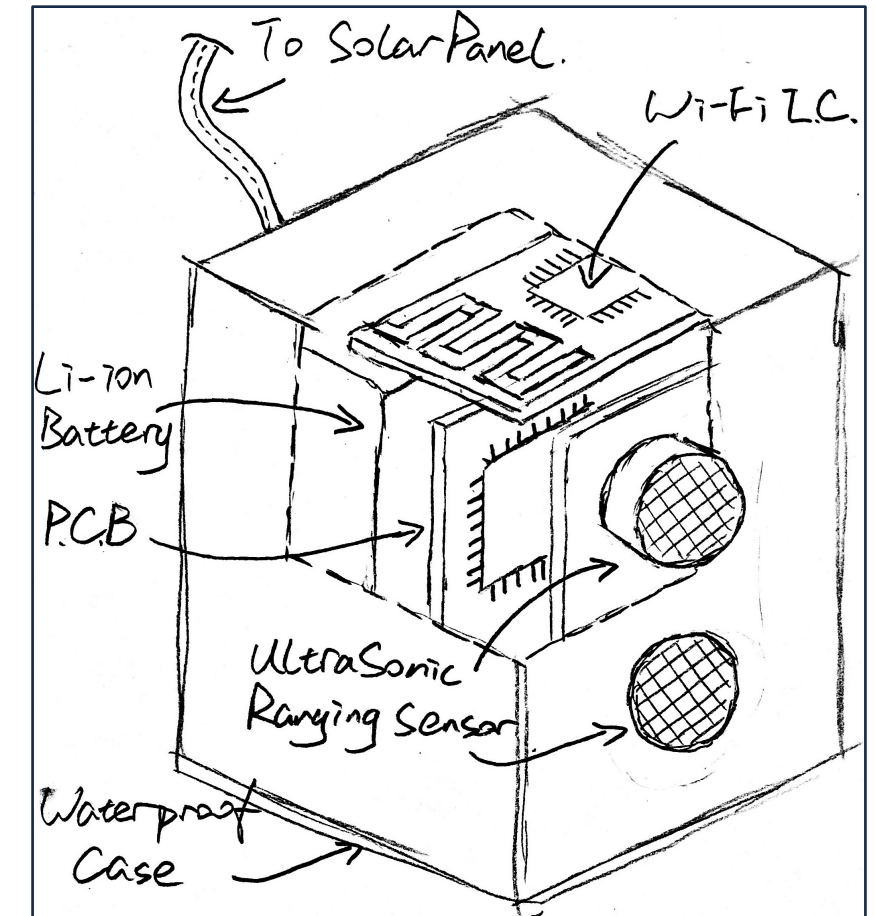
- Transmits Data From Control Module To User
- Method 2: Forward ports so that certain applications can access my computer
 - Pro: Easy to implement
 - Con: Dynamic IP address
 - Solution: Setup static IP address, or Dynamic Host Configuration Protocol (DHCP) reservation

Conclusion

- Ultrasonic sensor detects vehicle's occupancy status
 - Wi-Fi IC unit transmits data to control unit
 - Server reflects updated information to user
-
- Parking lot owners or law enforcers remotely monitor parking status
 - Drivers make better parking decisions

Future Work

- Water Resistance
- Plate Recognition
- Vehicle Type Recognition
- Distributed System:
 - Cheaper
 - Avoid database single-point failure
 - Easier to manage many parking lots concurrently



Question?

Thank you!

Supplement Materials

Physical Design (Control MCU R&V)

- Requirement:
 - *MCU Must be able to identify detection data source, i.e. decode the space index, and must share data with other components with $\geq 0.5\text{Hz}$ refresh rate*
- Verification:
 - Inject known dummy data with defined format into it
 - Access the output to check its decoding result
 - Inject a $\geq 0.5\text{Hz}$ time-varying signal to it and monitor its output using the oscilloscope

Full Data Flowchart

