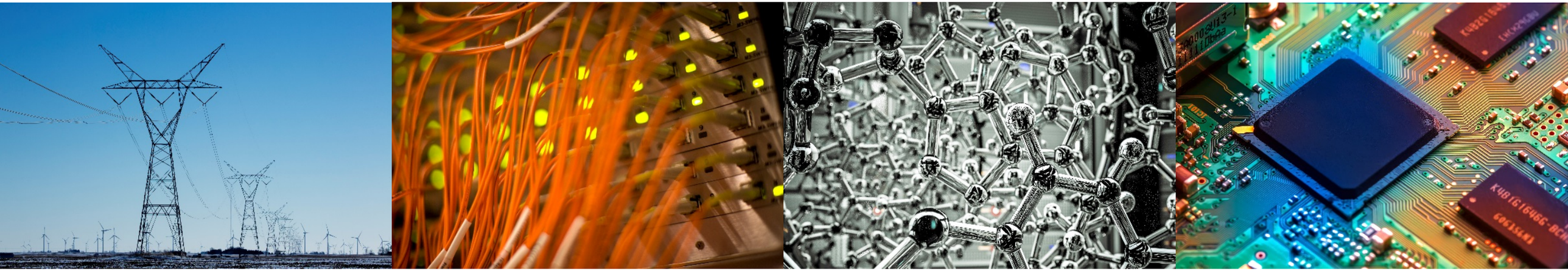
VACANT PARKING DETECTOR 2.0

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

TA: Zhen Qin

ECE 445 Senior Design, April 30, 2018



ILLINOIS

Electrical & Computer Engineering

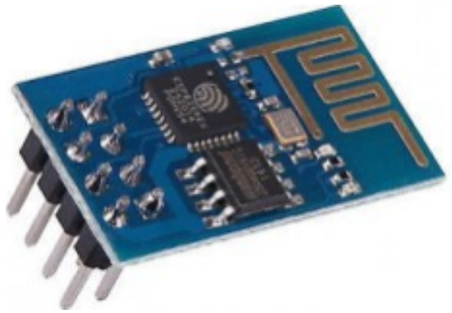
COLLEGE OF ENGINEERING

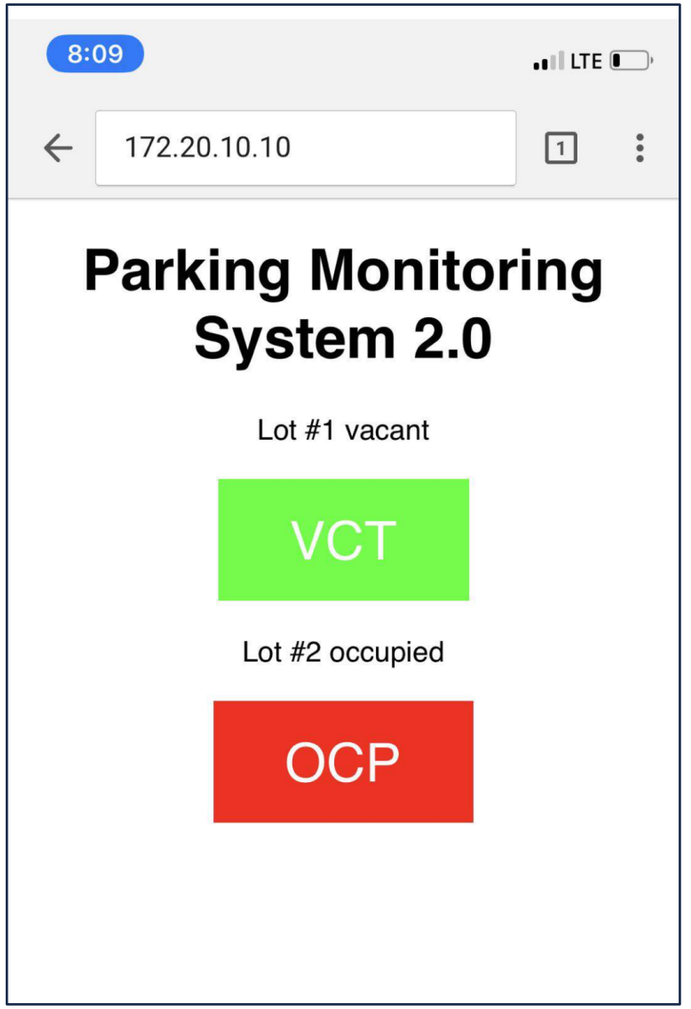
Project Motivation & Objective

- (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



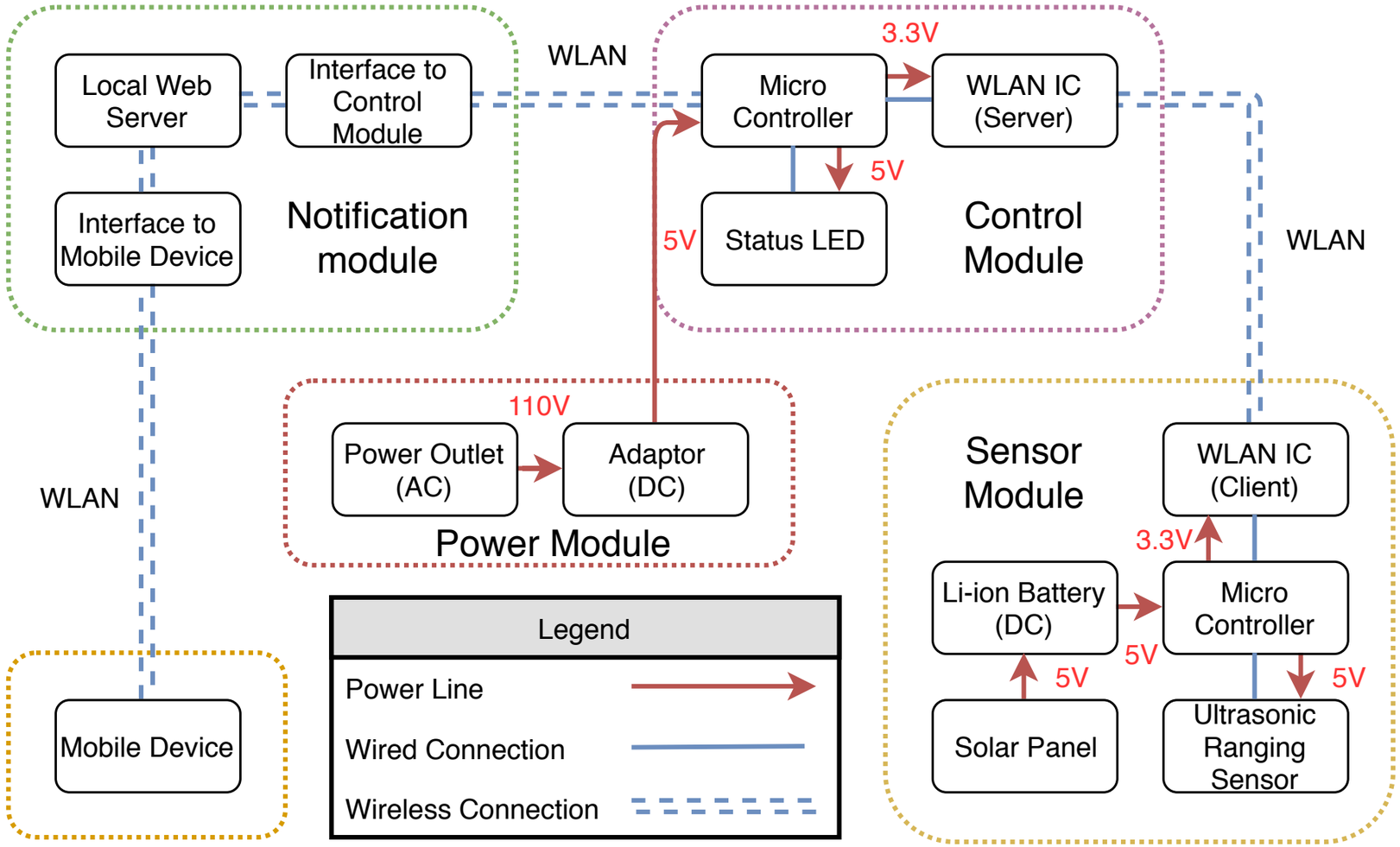


Project Comparison

	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

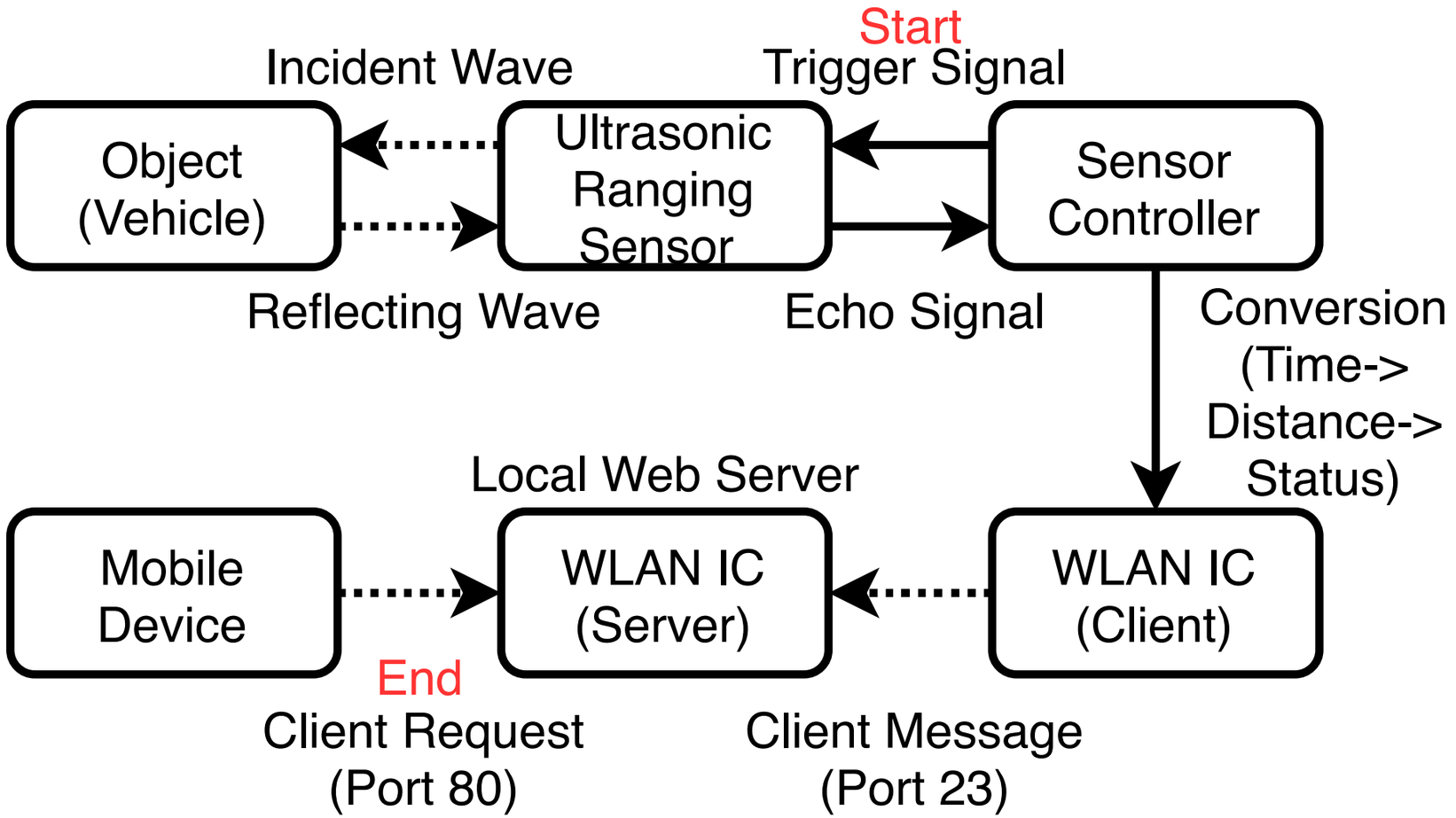
Design Overview

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

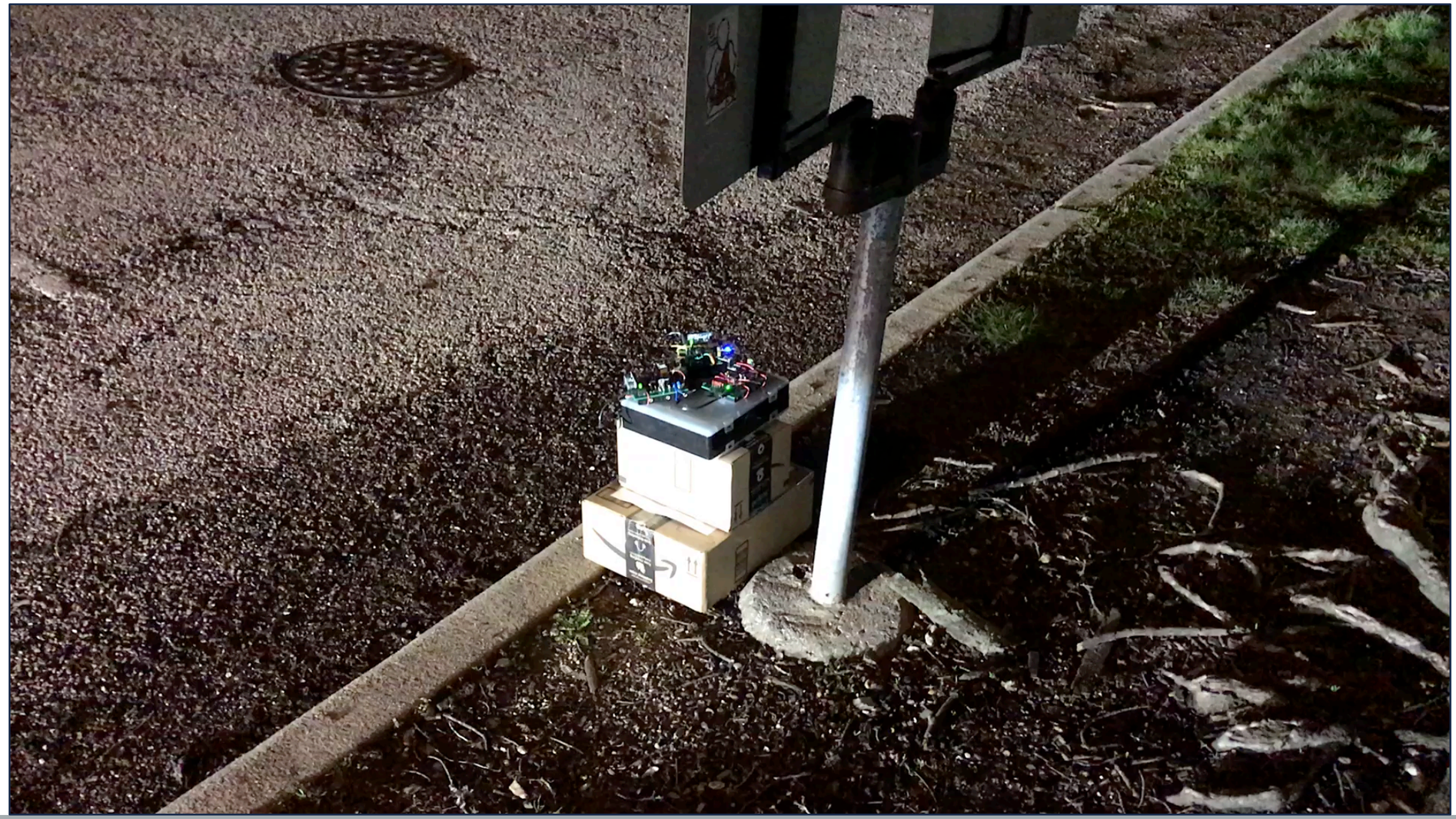


Design Overview

- Data Flowchart:
  - Start: Sensor MCU initiates detection sequence
  - End: Mobile device browses local webpage for parking info
- Data Format:
  - Space Index + Space status

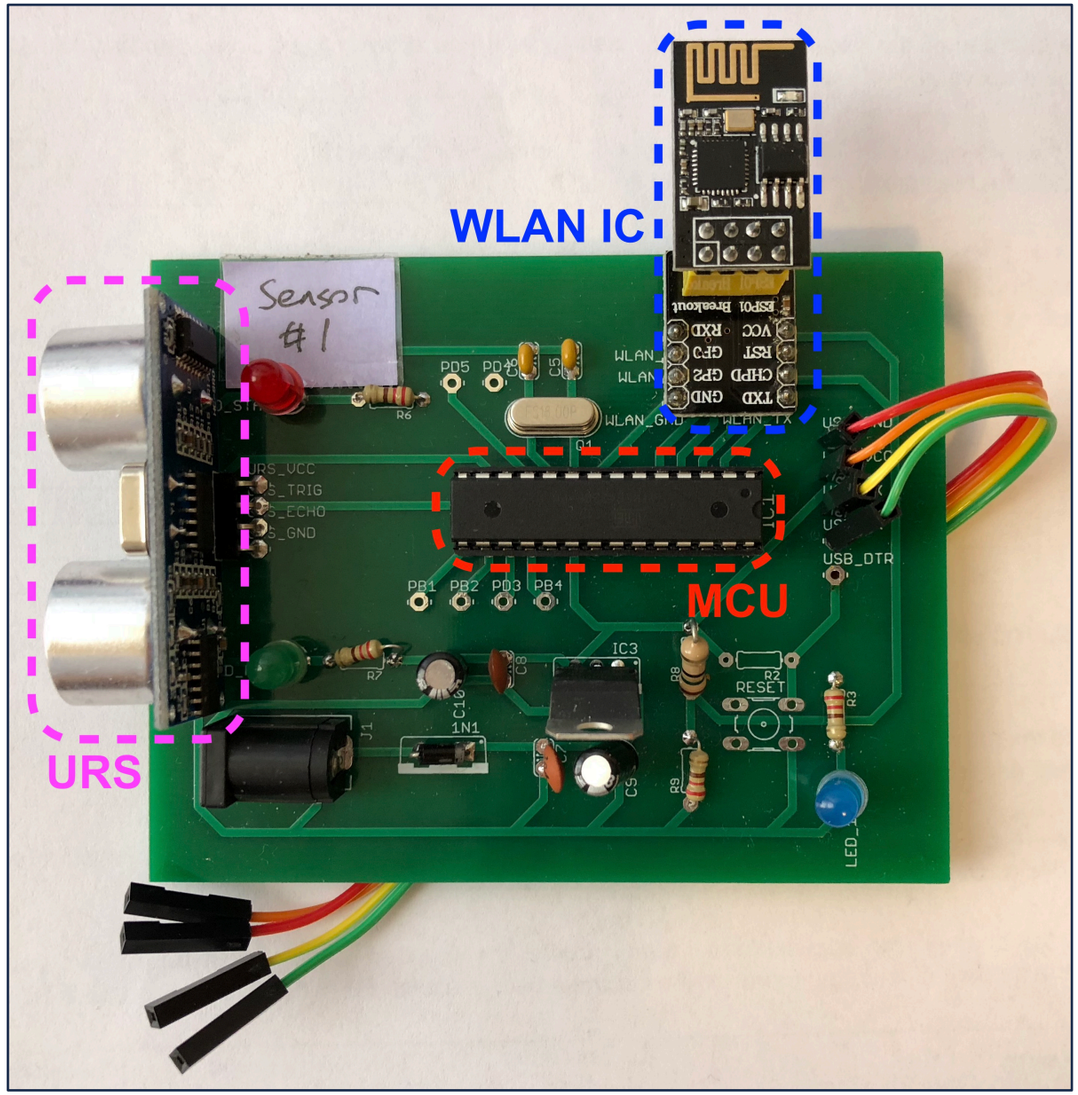


Project Demonstration

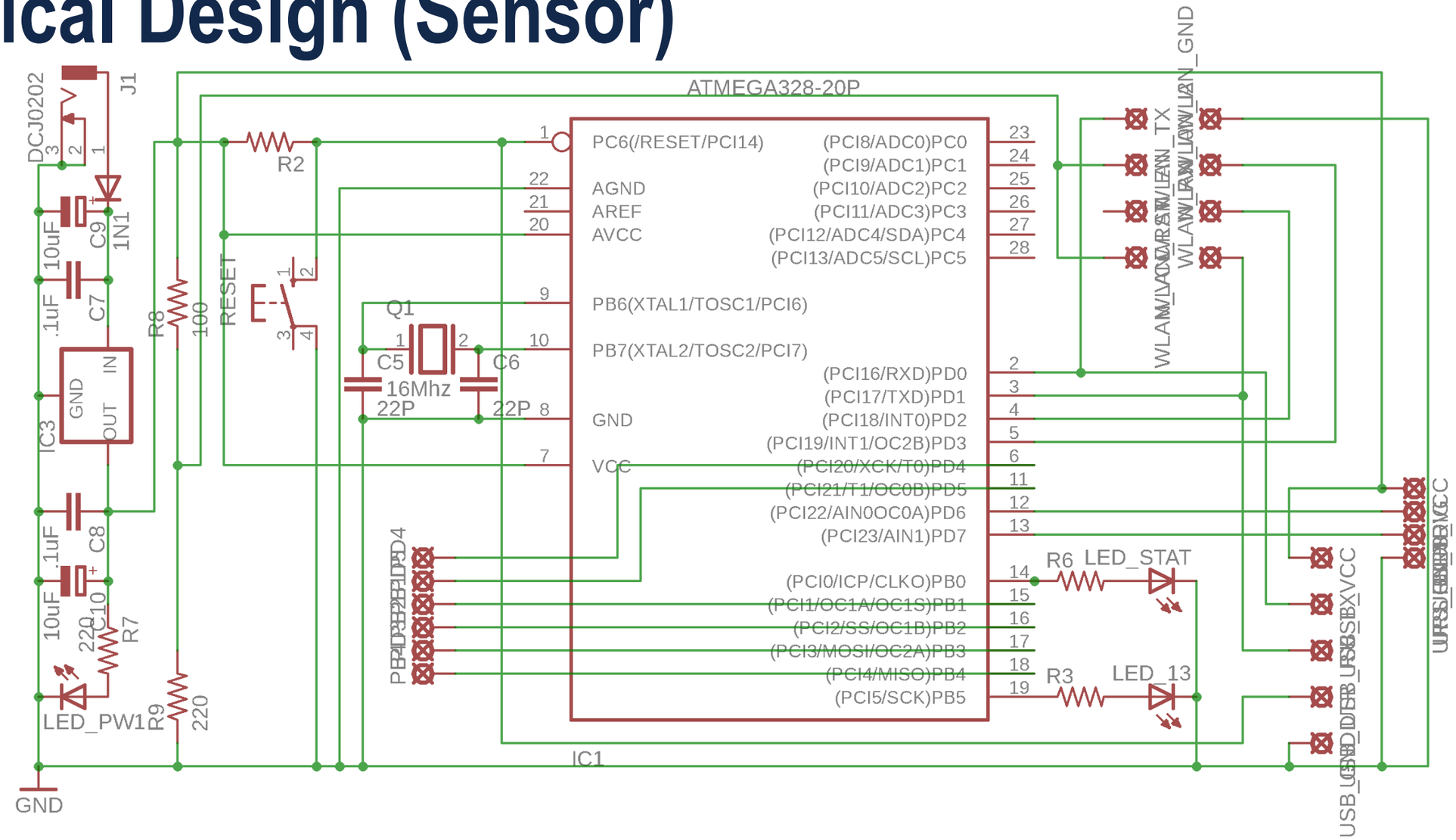


Physical Design (Sensor)

- 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

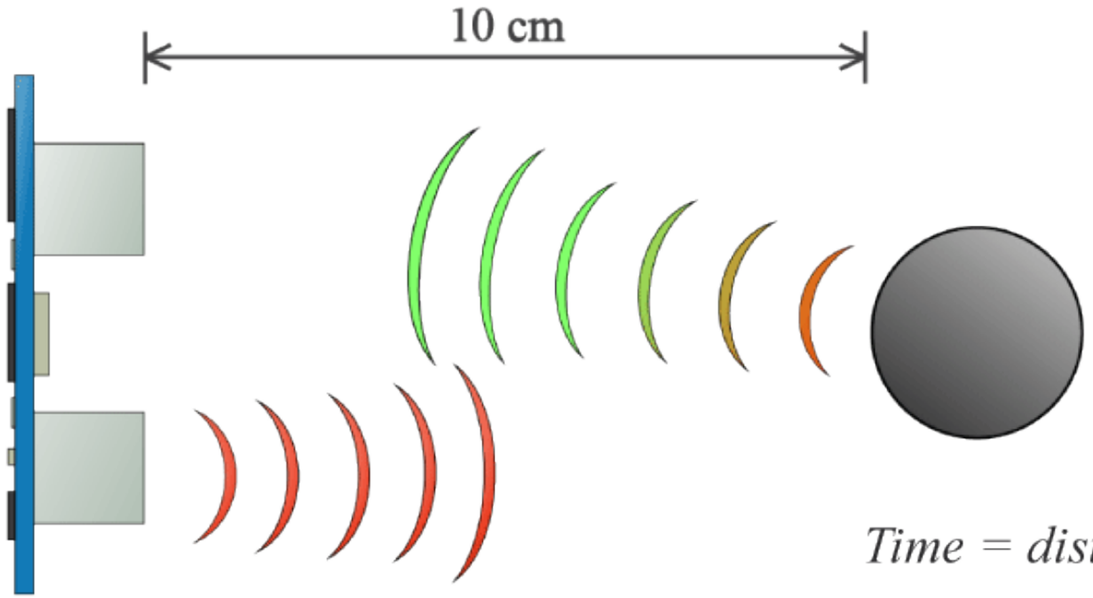


Physical Design (Sensor)



Physical Design (Sensor)

- URS Detection Schematic:
- Time Difference:
  - Round trip
- Object Distance:
  - 2m threshold
- Parking Status:
  - d < 2m: occupied
  - d >= 2m: vacant



speed of sound:

$v = 340\ m/s$

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

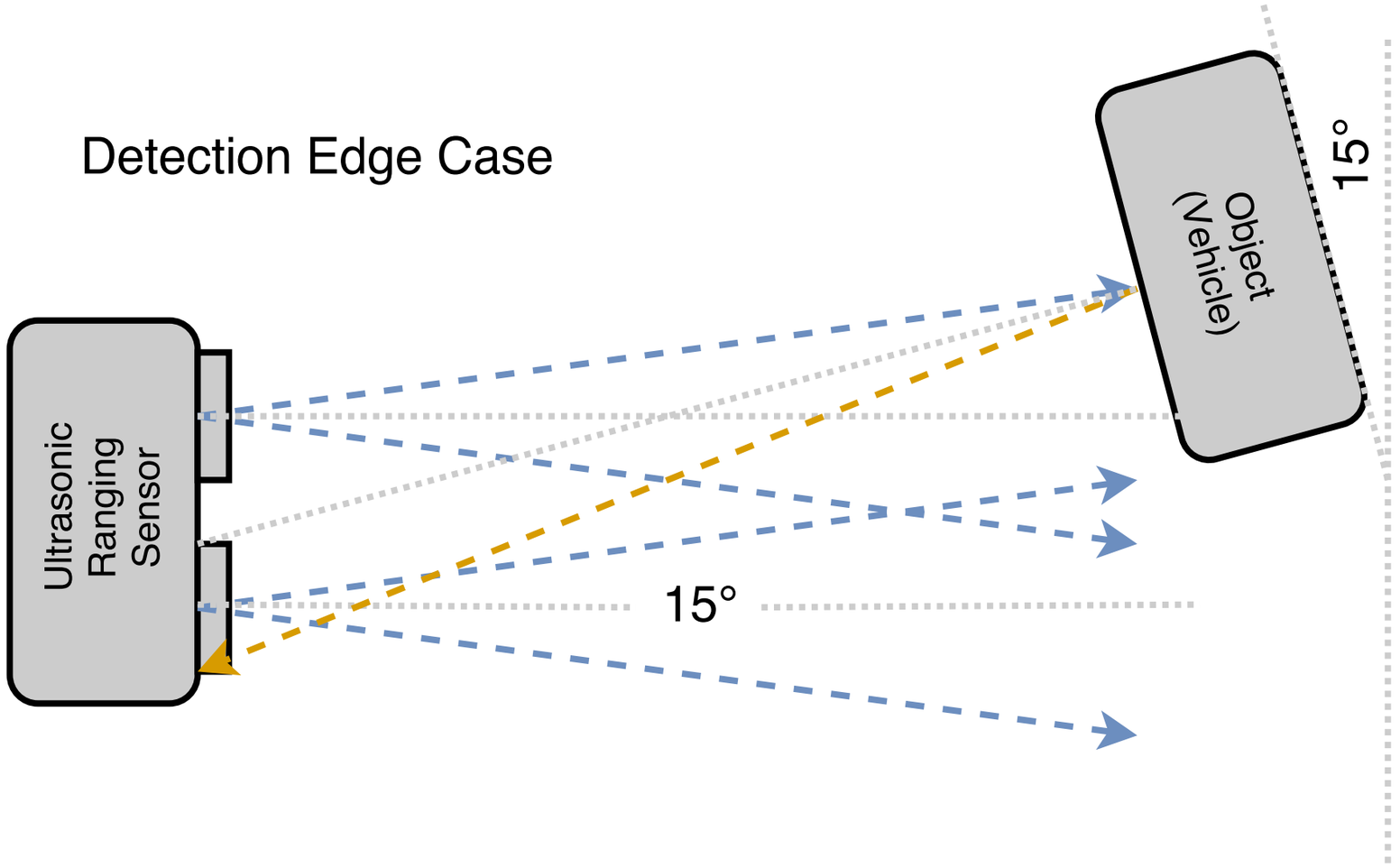
Time = distance / speed:

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

Distance:

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

Physical Design (Sensor Tolerance)

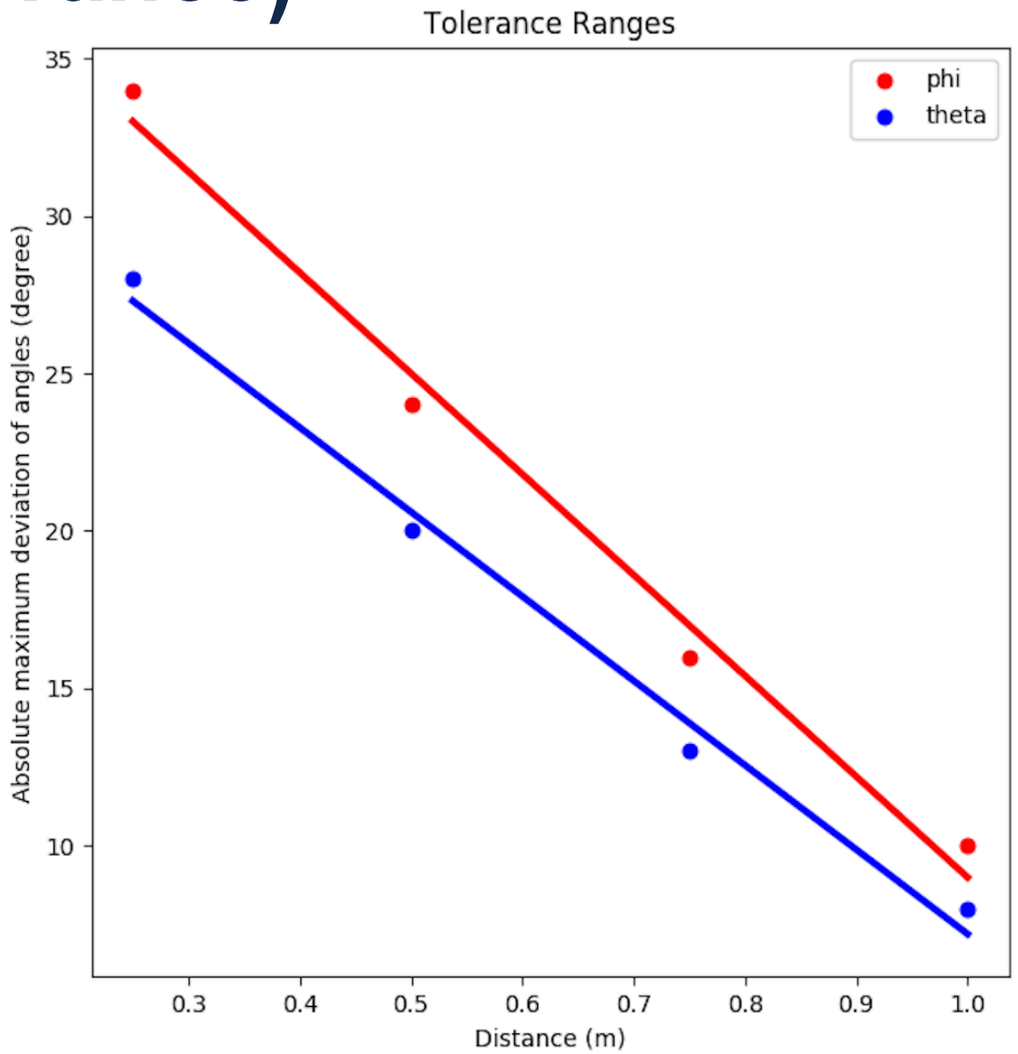


Physical Design (Sensor Tolerance)

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

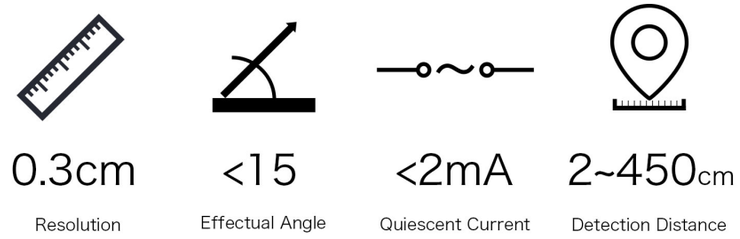
$$|\phi| = -32 * d + 42$$

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



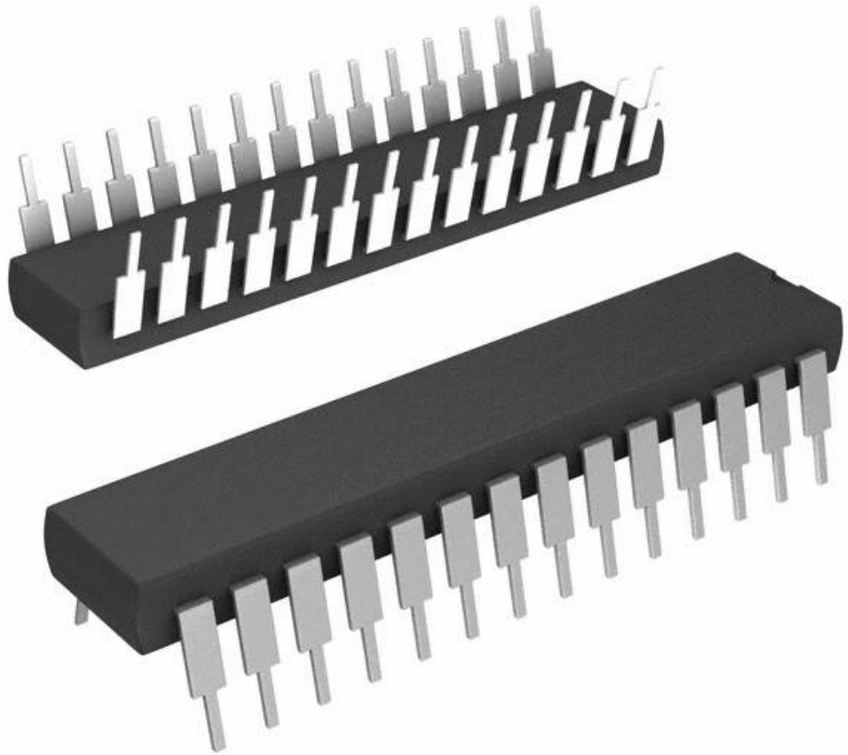
Physical Design (URS R&V)

- 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



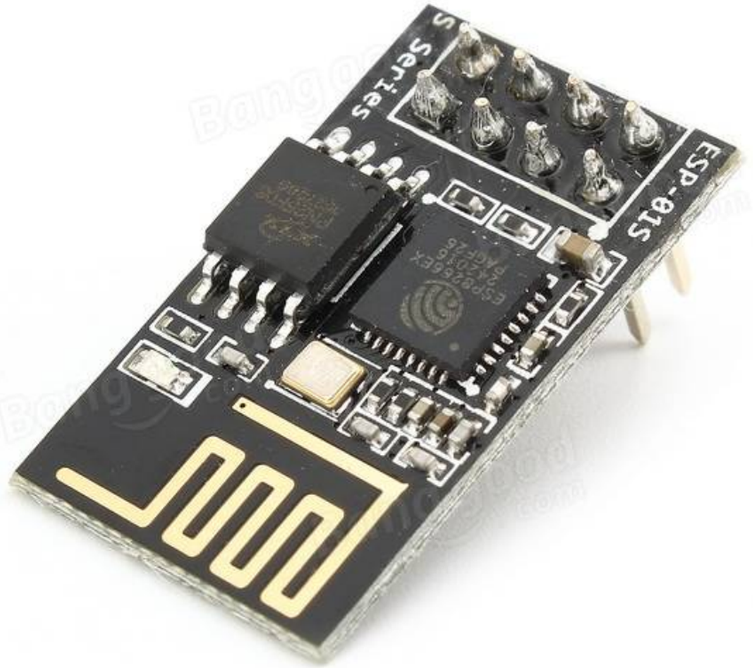
Physical Design (Sensor MCU R&V)

- Requirement:
  - *Convert time difference into space status with ~0.5Hz 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



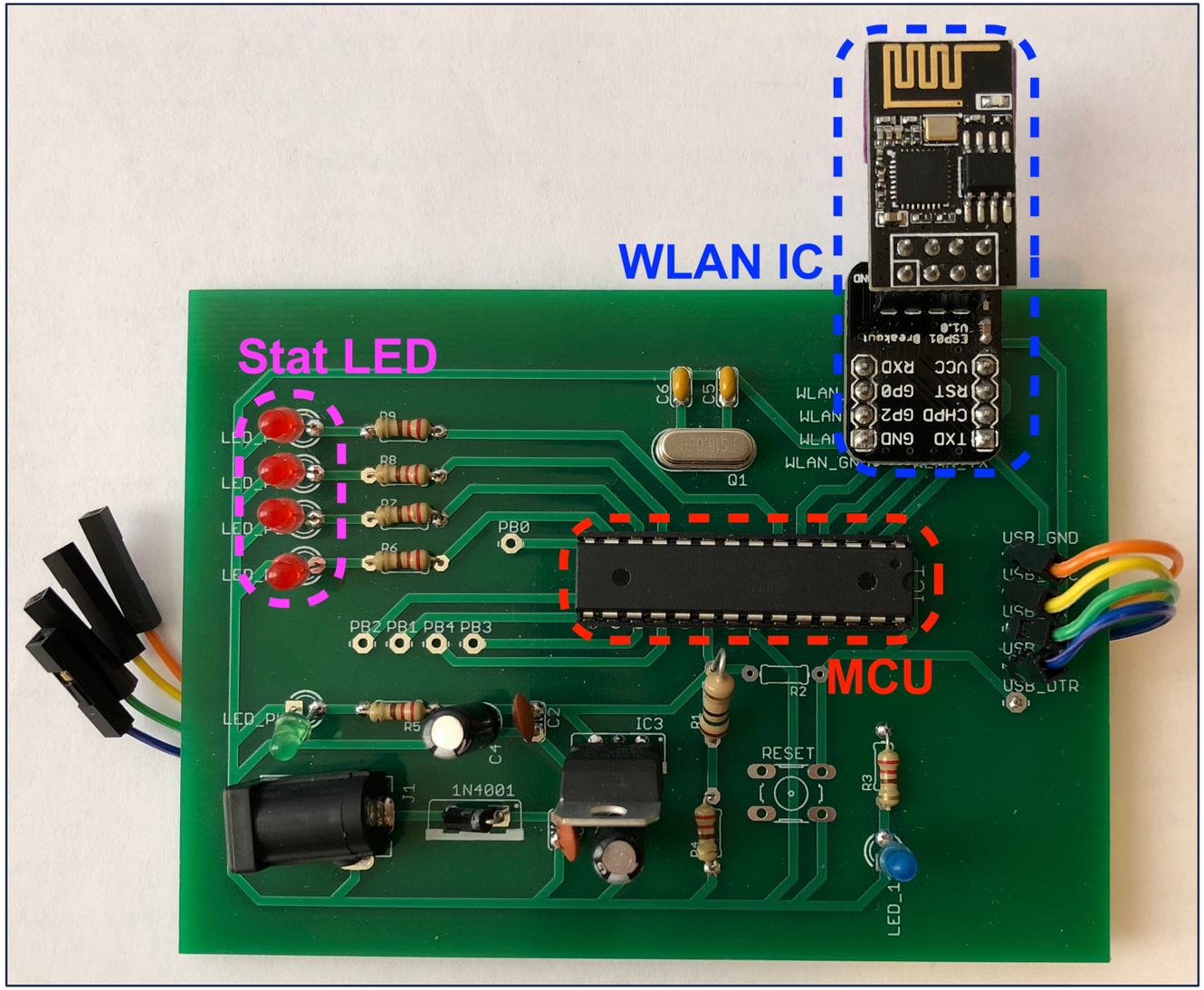
Physical Design (Client WLAN IC R&V)

- Requirement:
  - *Send message to Server WLAN IC with ~0.5Hz 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

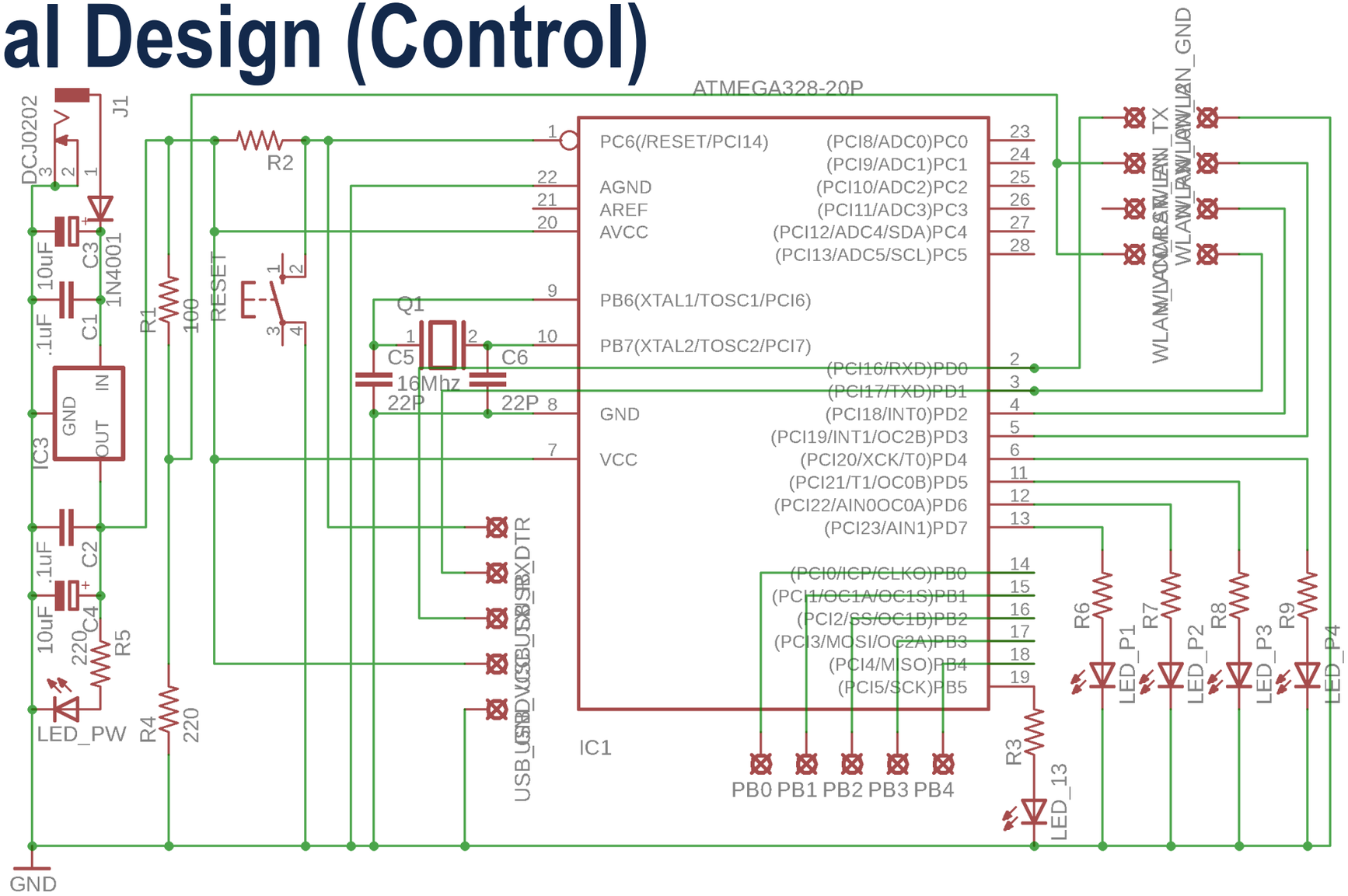


Physical Design (Ctrl)

- 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



Physical Design (Control)

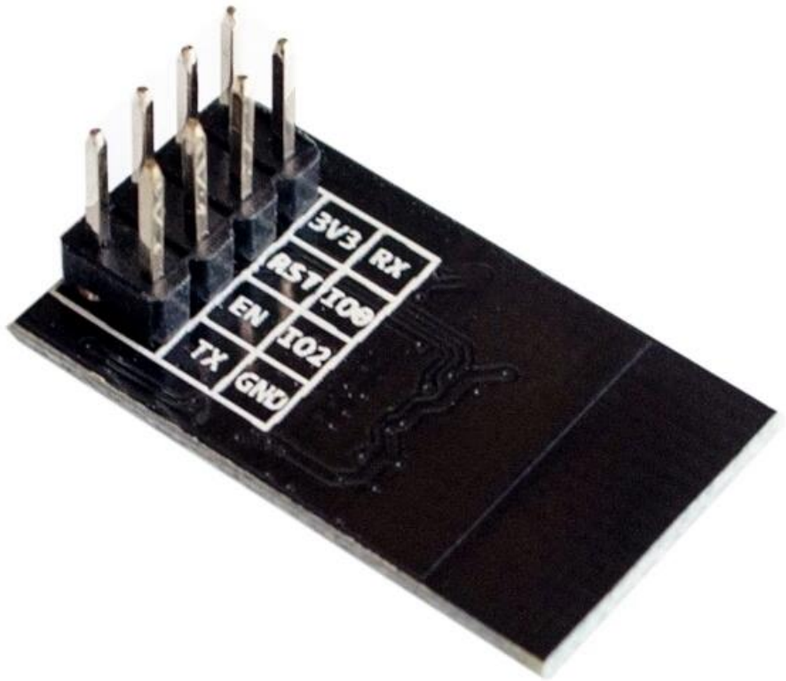


Physical Design (Control R&V)

- Requirement:
  - *Connect to all Sensor Module clients within range <= 25m 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 ~0.5Hz 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±0.1V and <1A to support module normal operation*
- Verification:
  - Connect the power supply to a 10kΩ resistor
  - Measure resistor voltage and current

Software Design (Notification)

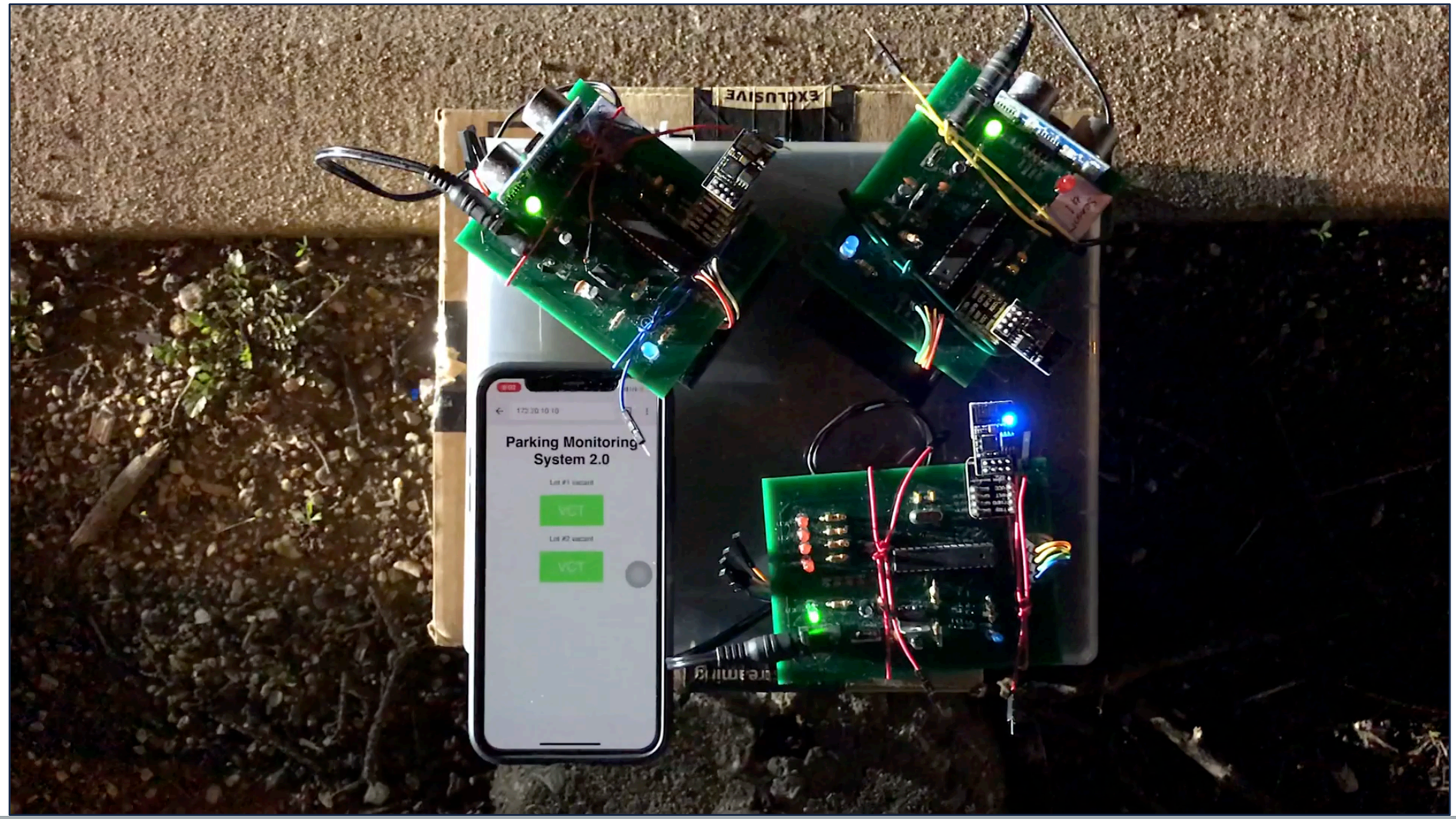
- Local Web Server:
  - Ambient Network Dependency: Require outside Wi-Fi router
  - Static IP Address: Easier access
  - Dual Ports Setup: 23 for WLAN IC, 80 for mobile device
  - Local Webpage: Display parking status message in graphics

```
byte ledPin = 2;
char ssid[] = "Jiahe";
char pass[] = "zhangzeyu";
// esp server
WiFiServer server(23);
// web server
WiFiServer server_(80);

IPAddress ip(172,20,10,10);
IPAddress gateway(172,20,10,1);
IPAddress subnet(255,255,255,0);

String stat_1 = "vacant";
String stat_2 = "vacant";
String stat_3 = "vacant";
// Variable to store the HTTP re
```

Software Design (Mobile Device)

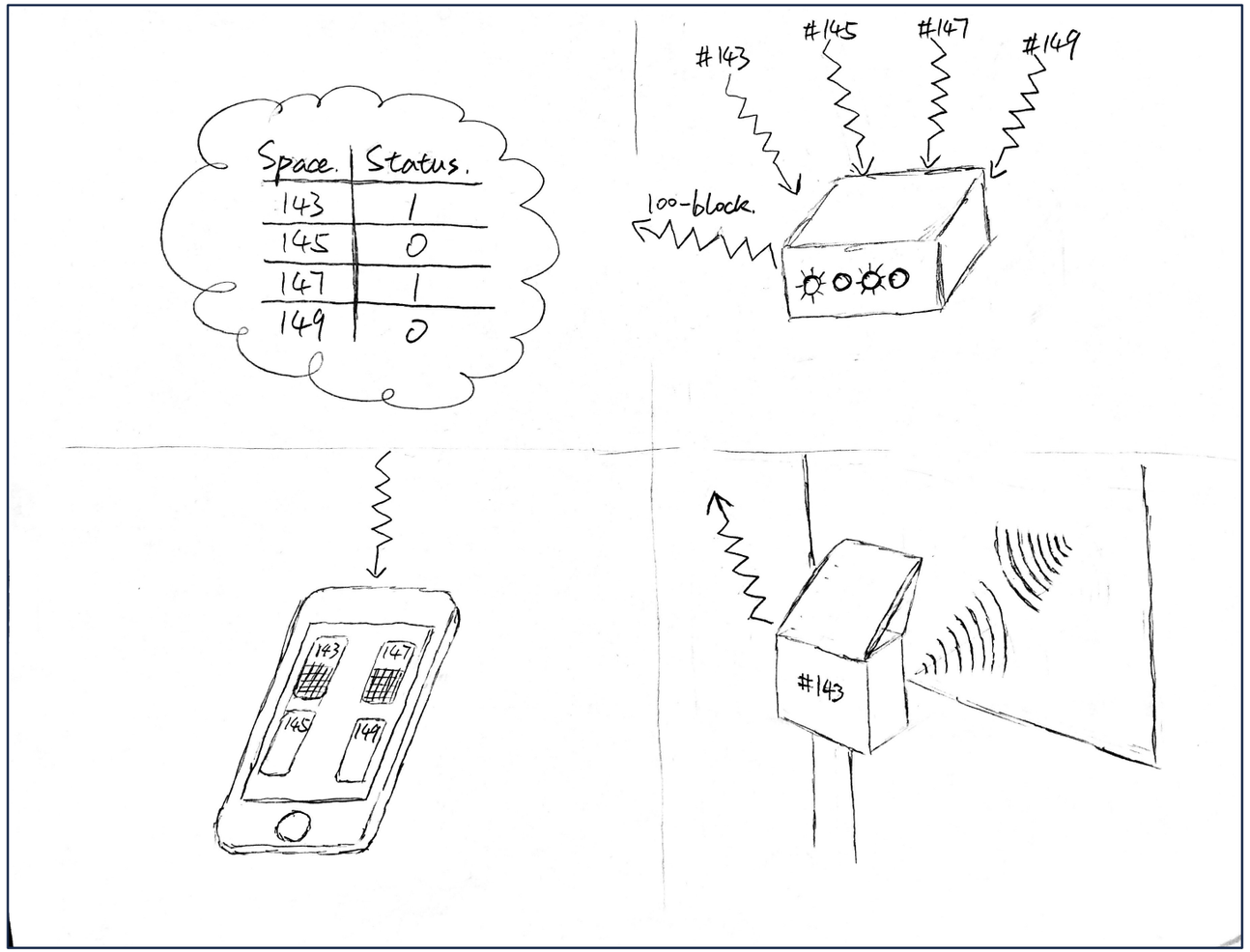
- Access Local Webpage:
  - Use the static webserver IP address
  - Parking status displayed in both text and graphic
  - Refresh the webpage to get updated parking info

Software Demonstration



Project Development

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

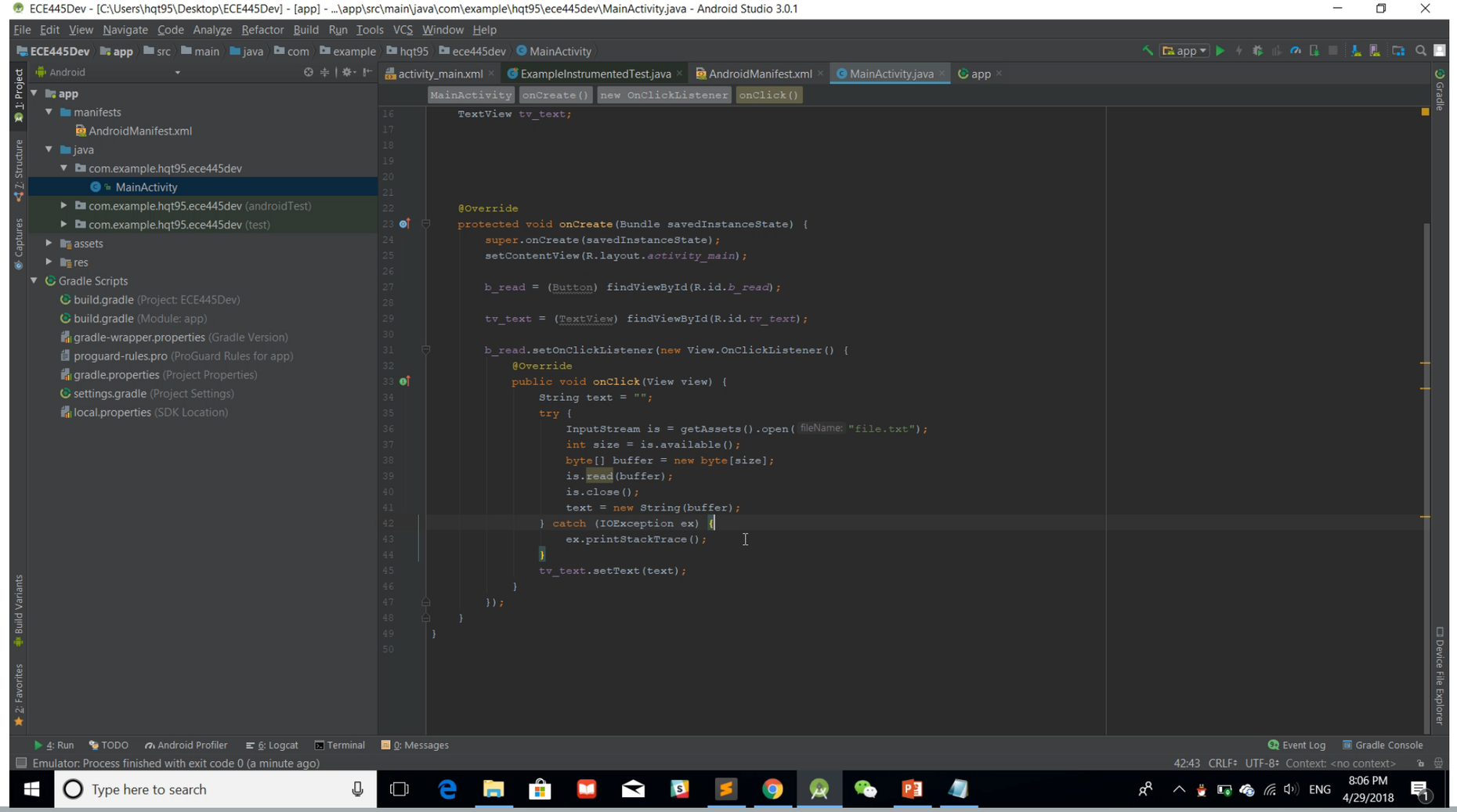


Project Development

- 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

```python
import serial
import re

parking = {}
client = 0
status = 0

# search for serial port
ser = serial.Serial('/dev/cu.usbserial-A9M9DV3R', 115200)
while(1):
    print(ser.readline())
    info = str(ser.readline())
    # extract parking slot id
    c = re.search('Client{(.+?)}', info)
    if c:
        client = c.group(1)
        # extract parking status
        s = re.search('Status{(.+?)}', info)
        if s:
            status = s.group(1)
            parking[client] = status

    output_f = open('parking.txt', 'w')
    # update all the parking information for every loop
    for cli in parking.keys():
        if parking[cli] == '0':
            output_f.write('Slot: '+cli+' '+'Unoccupied'+'\r\n')
        else:
            output_f.write('Slot: '+cli+' '+'Occupied'+'\r\n')
    output_f.close()
```

Project Development (Method 1 Demo)



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

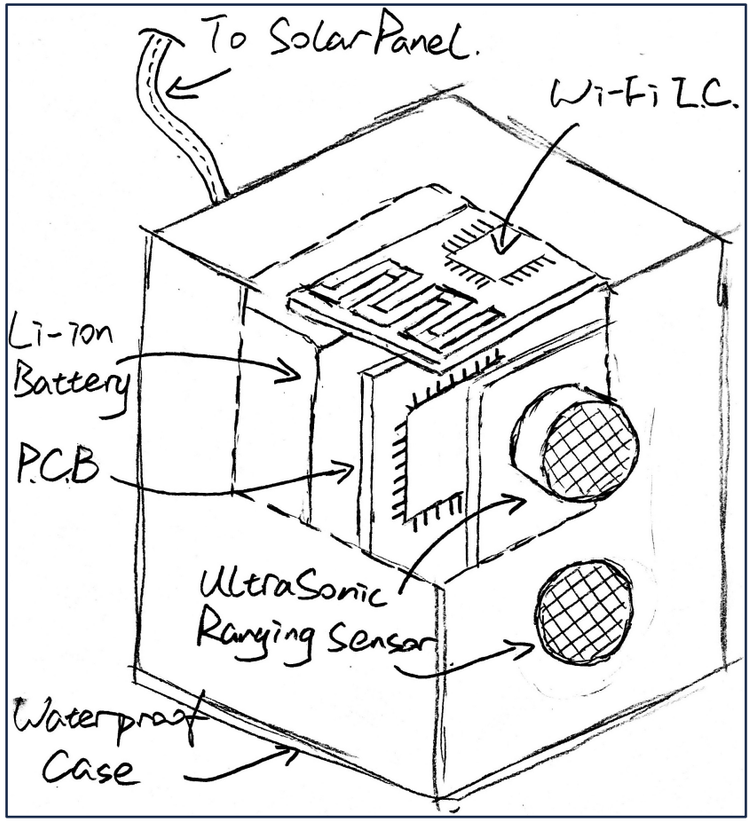
Conclusion

- 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 >=0.5Hz refresh rate*
- Verification:
  - Inject known dummy data with defined format into it
  - Access the output to check its decoding result
  - Inject a >=0.5Hz time-varying signal to it and monitor its output using the oscilloscope

Full Data Flowchart