Learn. Rise. Excel

Ganesh College of Engineering

Smart parking

Submitted by

Sandhiya.k
Sandhiya.s
Kowsalya.m
Sathya.r

Smart parking

development

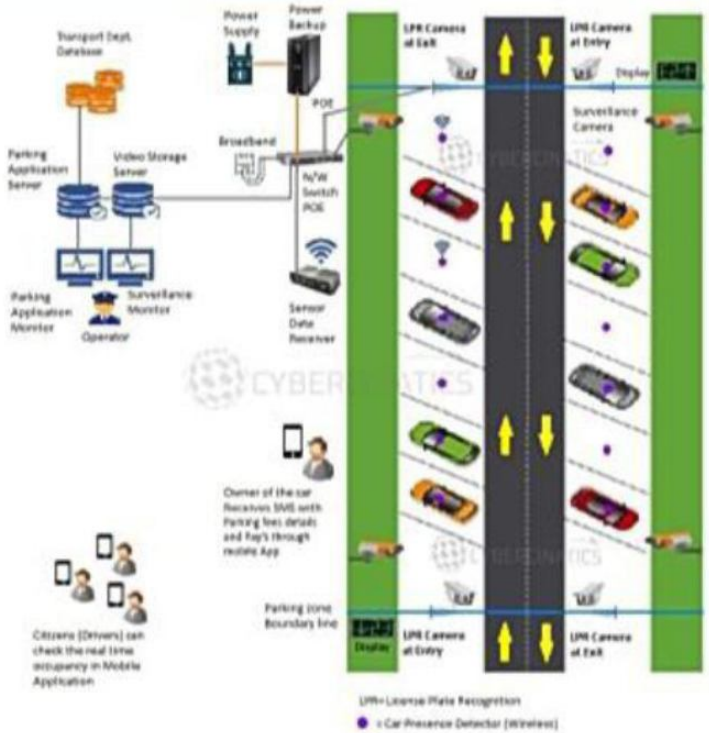
Smart Parking Management System

- Proposing a Customized Automated Vehicle Parking System.
- Suitable for On street, Off street and Multi-level parking Lots.
- Public can know the availability of parking vacancies in Real-time through mobile application.
- Governance bodies can monitor the system from their Remote command and Control center.
- Parking fees will be charged as per parking time.
- Surveillance to provide security.
- Recognize violation and charge them with penalty.
- Entire System is according to the new parking guidelines for smart cities.
- Readily Deployable System.

System Architecture

Architecture of Smart Parking System

System Architecture



Architecture of On-Street Parking System

Smart Parking Zone Features

- Access the real time occupancy of individual parking Lots.
- Parking Management Software, Mobile App for users, Web-Portal for resource management.
- Real time location based view to citizens about availability of parking space.
- one-to-one mapping with parking sensors
- Captures & Stores the number plate images of vehicles entering and exiting parking lot.
- Generation of parking receipts at the parking Lot.
- Hardware failure can be reported & tracked through portal
- Various reports like payment status, check-in vehicle list, check-out vehicle list, hardware active status, payment pending list etc.,
- Provision for Manual Entry incase the cameras do not recognize the vehicle.
- Surveillance cameras at the Parking lots provide secured monitoring.

Parking Zone Status Displays

- Parking Zone should be allocated either On-street or Off-street.
- The parking zone is identified by Sign boards.
- The On-Street Parking Zone will have no obstructions on road.
- LED Signboards will be installed at the entry point of the parking Zone.
- Real time vacancy availability will be displayed on LED Signboards at the entry.

Creating a smart parking project for ESP32 on the wokwi platform involves microcontroller to detect and manage parking spaces,and then visualizing the data on a virtual interface provided by wokwi.

Components needed

1.EPS32 development board

2. ultrasonic distance sensors

3.breadboard and jumper wires

4.wokwi virtual simulator

Project steps

1.hardware setup

a)connect the HC-SR04 ultrasonic sensors to ESP32 board.will need one sensor per parking space

b)wire the HC-SR04sensors as follows:

*VCC to 5V on ESP32

* GND to GND on ESP32

* Trig to a digital GPIO pin on ESP32

*Echo to another digital GPIO pin on ESP32

c) connect all the sensors in the same way , one for each parking space you want to monitor.

2.programming

a)write an Arduino sketch for the ESP32 that reads the distance data from the ultrasonic sensors.

‘“cpp

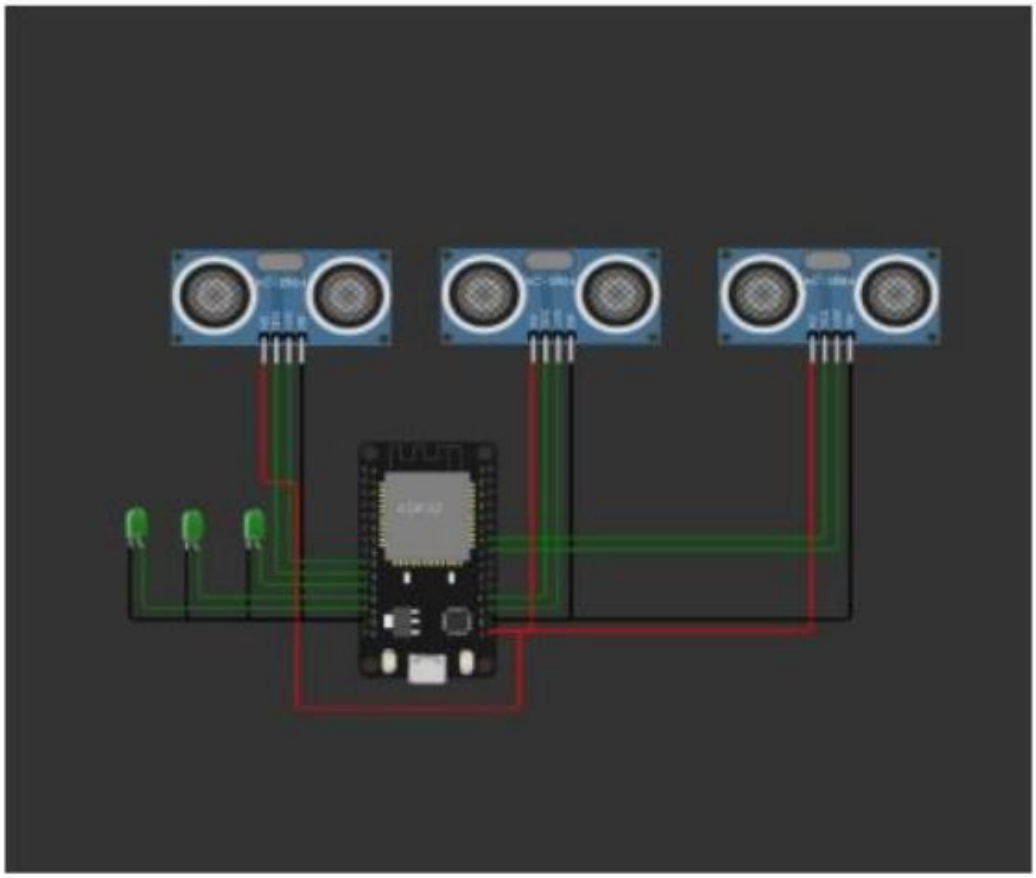
#include <ultrasonic.h>

```
Ultrasonic
sensor1(GPIO_TRIGGER1,GPIO_ECHO1);
Ultrasonic
sensor2(GPIO_TRIGGER2,GPIO_ECHO2);
//Add more sensors if needed

void setup() {
   serial .begin(115200);

viod loop() {
 long distance1 = sensor1 . read();
 lond distance2 = sensor2. read();
 // Read distances from more sensors if needed

// process distance data and manage parking
spaces here
delay(10000);// Delay for better readabilty
```

```
}
"'
```

Visuvalation:

Simulation:



code

```
#define ECHO_PIN1 15  //pins for sensor 1
#define TRIG_ PIN1  2    //pins for sensor 1

#define ECHO_PIN2  5    //pins for sensor
#define TRIG_PIN2  18   //pins for sensor

#define ECHO_PIN3  26 //pins for sensor
#define TRIG_PIN3   27 //pins for sensor

int LEDPIN1  = 13;
int LEDPIN2  = 12;
int LEDPIN3  =14;

Void setup() {
    Serial.begin(115200);
    pinmode(LEDPIN1, OUTPUT);
    pinmode(TRIG_PIN1, OUTPUT);
    pinmode(ECHO_PIN1, INPUT);

     pinmode(LEDPIN2, OUTPUT);
```

```
    pinmode(TRIG_PIN2, OUTPUT);
     pinmode(ECHO_PIN2,INPUT);

      pinmode(LEDPIN3, OUTPUT);
     pinmode(TRIG_PIN3, OUTPUT);
     Pinmode(ECHO_PIN3,INPUT);
}
float readDistance1CM() {
     digitalwrite(TRIG_PIN1, LOW);
     delaymicroseconds(2);
     digitalwrite(TRIG_PIN1, HIGH);
     delaymicroseconds(10);
     digitalwrite(TRIG_PIN1,LOW);
     int duration = pulseIn(ECHO_PIN1, HIGH);
     return duration* 0.034 /2 ;
}
float readDistance2CM() {
     digitalwrite(TRIG_PIN2, LOW);
     delaymicroseconds(2);
     digitalwrite(TRIG_PIN2,HIGH);
```

```
        delaymicroseconds(10);
        digitalwrite(TRIG _PIN2, HIGH);
         int duration = pulseIn(ECHO_PIN2, HIGH);
         Return duration *0.034 / 2;
}
Float readDistance3CM() {
        digitalwrite(TRIC_PIN3, LOW);
        delaymicroseconds(2);
        digitalwrite(TRIG_PIN3, HIGH);
        delaymicroseconds(10);
        digitalwrite(TRIG_PIN3 , LOW);
        int duration = pulseIn(ECHO_PIN3, HIGH);
        return duration * 0.034/ 2;
}

void loop() {
        float distance1 = readDistance1CM();
        float distance2 = readDistance2CM();
        float distance3 = readDistance3CM();
```

```
    bool isNearby1 = distance1 > 200;
    digitalwrite(LEDPIN1, isNearby1);

    bool isNearby2 = distance2 > 200;
    digitalwrite(LEDPIN2, isNearby2);

    Bool isNearby3 = disstance3 > 200;
    digitalwrite(LEDPIN3, isNearby3);

    serial.print("Measured distance: ");
    serial.print(readDistance1CM());
   serial.print(readDistance2CM());
    serial.print(readDistance3CM());
   delay(100);
}
```

a) Node-RED

Node-RED is low-code programming for event-driven applications tool that is used to wiring together hardware devices, APIs and online services in new and many interesting ways. Nowadays, Node-RED is basically an open-source visual editor that is used by many developers for the internet of things applications. The Node-RED system contains what we called "Nodes" which we can drag and drop on the visual editor and wire together. There are many varieties of Nodes that offer different functionality that ranges from a simple debug node through a Raspberry Pi node that allows the developers to read and write to the GPIO pins on the Raspberry Pi.

Fig. 12. Node-RED

Creating a smart parking IoT project using Node-RED involves several steps. Here's an outline along with some example code snippets:

1. Setting up Node-RED:

- Install Node-RED on your system or IoT device.
- Install necessary packages and dependencies.

2. Hardware Setup:

- Set up the necessary sensors (like ultrasonic sensors) in the parking spots.
- Connect these sensors to your IoT device (e.g., Raspberry Pi).

3. Writing the Flow:

In Node-RED, create a flow to handle the sensor data and control the parking status. Here's an example flow:

```json
[{"id":"dab5a248.2644a8","type":"ui_gauge","z":"dbed8b49.aef898","name":"Parking Status","group":"9c2a7697.dbb33","order":0,"width":0,"height":0,"gtype":"gage","title":"Parking Status","label":"","format":"{{value}}","min":0,"max":"10","colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","x":600,"y":320,"wires":[]},{"id":"f17aa1f9.82d34","type":"function","z":"dbed8b49.aef898","name":"Generate Random Data","func":"msg.payload = Math.floor(Math.random() * 11);\nreturn msg;","outputs":1,"noerr":0,"x":380,"y":320,"wires":[["dab5a248.2644a8"]]},{"id":"d26236ac.35f7d","type":"inject","z":"dbed8b49.aef898","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"5","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":200,"y":320,"wires":[["f17aa1f9.82d34"]]},{"id":"9c2a7697.dbb33","type":"ui_group","name":"Parking","tab":"14bb5c32.f36d82","order":1,"disp":true,"width":"6","collapse":false},{"id":"14bb5c32.f36d82","type":"ui_tab","name":"Home","icon":"dashboard","order":1,"disabled":false,"hidden":false}]
```