SMART PARKING PHASE-3

PROCESS AND DATA:

Defining Project Requirements:

- *Begin by clearly defining the problems of your Smart Parking project.*
- *What problems are you trying to solve? Understanding your project's goals is essential.*

IoT Device Deployment:

- *Choose the appropriate IoT devices for your project. These devices should be capable of collecting, processing, and transmitting data to a central server or cloud platform.*
- *Deploy these IoT devices strategically in the areas where parking areas are available or not? Ensure they are properly powered and connected to the internet.*

Developing Python Script:

- *Develop a Python script to interface with the sensors and IoT devices. This script should be able to collect data from the sensors, process it, and transmit it to a central database or cloud platform for analysis.*
- *Ensure the script is robust, capable of handling data from multiple sensors, and includes error-handling mechanisms.*

Data Analysis and Visualization:

- *Set up a data analysis platform that can receive data from your IoT devices and perform relevant analyses.*

Documentation and Assessment:

- *Create comprehensive documentation of your project, including details about the deployed IoT devices, sensors, the Python script, data analysis, and any findings or insights.*

USED SENSOR:

- *In this project we are use the ultrasonic sensor, LED's and ESP32.*
- ➢ *ULTRASONIC SENSOR:*
- *Ultrasonic sensor is used for measuring the distance from the object.*
- *In this place, we are using an ultrasonic sensor for parking space is available or not.*

- *LED:*

- *The red colour led indicates the parking place aren't available.*
- *The green colour led indicates the parking place is available.*

PYTHON SCRIPT:

```
int led_red_1   = 2;
int led_green_1 = 4;
int led_red_2   = 5;
int led_green_2 = 18;
int led_red_3   = 19;
int led_green_3 = 21;
int led_red_4   = 22;
int led_green_4 = 23;



int t_1 = 13;
int e_1 = 12;
int t_2 = 14;
int e_2 = 27;
int t_3 = 26;
int e_3 = 25;
int t_4 = 33;
int e_4 = 32; //34,35,36,39 pin for input only



int d_1, d_2, d_3, d_4;
int cm_1, cm_2, cm_3, cm_4;
int p1, p2, p3, p4;             //Return the parking status to app



void setup() {
  Serial.begin (9600);//Initialize the serial port for debugging output
information
```

```
  //BT.begin(9600); //Set Bluetooth initial value (from AT mode to see the bao
rate)


  pinMode(led_red_1, OUTPUT);
  pinMode(led_green_1, OUTPUT);
  pinMode(led_red_2, OUTPUT);
  pinMode(led_green_2, OUTPUT);
  pinMode(led_red_3, OUTPUT);
  pinMode(led_green_3, OUTPUT);
  pinMode(led_red_4, OUTPUT);
  pinMode(led_green_4, OUTPUT);
  pinMode(t_1, OUTPUT);
  pinMode(e_1, INPUT);       //Read the potential of the echo
  pinMode(t_2, OUTPUT);
  pinMode(e_2, INPUT);
  pinMode(t_3, OUTPUT);
  pinMode(e_3, INPUT);
  pinMode(t_4, OUTPUT);
  pinMode(e_4, INPUT);
}



void loop() {
  car_01();  //Read parking status of parking space 1 (cm, p, led)
  car_02();
  car_03();
  car_04();



Serial.println(String("A: ") + cm_1 + String("/ B: ") + cm_2 + String("/ C: ")
+ cm_3 + String("/ D: ") + cm_4);
```

```
  byte packet[5];
 packet[0] = 97;   //The key value that is verified by the app
  packet[1] = p1; //Parking condition, "0" is vacant parking space; 1" is not stoppable for cars
  packet[2] = p2;
  packet[3] = p3;
  packet[4] = p4;
Serial.println(String("p1: ") + p1 + String("/ p2: ") + p2 + String("/ p3: ") + p3 + String("/ p4: ") + p4);



  /*if(BT.available() > 0) {                    //Check whether the Bluetooth connection is successful
    if(BT.read() == 97) {                       Check whether you receive the key value
      Serial.println("Connect succeed!");
      for(int i = 0; i < 5; i++) {
        BT.write(packet[i]);                    //Send packets sequentially to the app
        Serial.println(packet[i]);
      }
    }
  }*/
  delay(2000);
}



void car_01(){
  //Parking space 001
  digitalWrite(t_1, LOW);  //Give Trig a low potential for 5 μs
  delayMicroseconds(5);
  digitalWrite(t_1, HIGH); //Give Trig a high potential for 10 μs
  delayMicroseconds(10);
```

```
  digitalWrite(t_1, LOW);
  d_1 = pulseIn(e_1, HIGH);  //Time at reception of high potential
  cm_1 = (d_1/2) / 29.1;       //Convert time to distance, in cm


  if (cm_1>=10) {                              /*When the distance is greater
than 10 cm, parking space No. 1 is "vacant"*/
    p1=0;                                     /*Send back "p1 value" to "0" to
APP*/
    digitalWrite(led_green_1, HIGH);          //Green light on, stopper
    digitalWrite(led_red_1, LOW);
  }
  else {                                       /*When the distance is less than
10 cm, parking space No. 1 is "in use"*/
    p1=10;                                     /*Send back the "p1 value" to
the APP*/
    digitalWrite(led_green_1, LOW);
    digitalWrite(led_red_1, HIGH);             //The red light is on, and it
cannot be stopped
  }
}


void car_02(){
  //Parking space 002
  digitalWrite(t_2, LOW);
  delayMicroseconds(5);
  digitalWrite(t_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(t_2, LOW);
  d_2 = pulseIn(e_2, HIGH);
  cm_2 = (d_2/2) / 29.1;
```

```
  if (cm_2>=10) {
    p2=0;
    digitalWrite(led_green_2, HIGH);
    digitalWrite(led_red_2, LOW);
  }
  else {
    p2=10;
    digitalWrite(led_green_2, LOW);
    digitalWrite(led_red_2, HIGH);
  }
}


void car_03(){
  //Parking space003
  digitalWrite(t_3, LOW);
  delayMicroseconds(5);
  digitalWrite(t_3, HIGH);
  delayMicroseconds(10);
  digitalWrite(t_3, LOW);
  d_3 = pulseIn(e_3, HIGH);
  cm_3 = (d_3/2) / 29.1;


  if (cm_3>=10) {
    p3=0;
    digitalWrite(led_green_3, HIGH);
    digitalWrite(led_red_3, LOW);
  }
  else {
    p3=10;
    digitalWrite(led_green_3, LOW);
```

```
    digitalWrite(led_red_3, HIGH);
  }
}


void car_04(){
  //Parking space 004
  digitalWrite(t_4, LOW);
  delayMicroseconds(5);
  digitalWrite(t_4, HIGH);
  delayMicroseconds(10);
  digitalWrite(t_4, LOW);
  d_4 = pulseIn(e_4, HIGH);
  cm_4 = (d_4/2) / 29.1;


  if (cm_4>=10) {
    p4=0;
    digitalWrite(led_green_4, HIGH);
    digitalWrite(led_red_4, LOW);
  }
  else {
    p4=5;
    digitalWrite(led_green_4, LOW);
    digitalWrite(led_red_4, HIGH);
  }
}
```