PROJECT

SMART PARKING

Project descripton:

Welcome to the revolution of parking efficiency! In this presentation, we will explore how the Internet ofThings (IoT) is transforming parking solutions. Discover the power of IoT in optimizing parking spaces, reducing congestion, and enhancing user experience. Get ready to delve into the world of smart parking!

Parking congestion, inefficient utilization of spaces, and frustrated drivers are common problems in urban areas. Traditional parking systems lack real-time information and smart management. IoToffers a solution by enabling real-time monitoring, intelligent parking guidance, and seamless payment mechanisms. Let's explore how IoT can revolutionize parking efficiency

By leveraging IoTtechnology, parking becomes smarter and more efficient. Connected sensors and devices provide real-time data on available parking spaces, allowing drivers to find vacant spots effortlessly. Smart parking solutions also enable automated payment systems, reducing the hassle of manual transactions. Let's dive deeper into the components of IoT-enabled smart parking.

Sensors play a crucial role in smart parking solutions. They detect the presence or absence of vehicles in parking spaces, transmitting data to a central system. Realtimedata on parking occupancy enables efficient management, accurate guidance, and predictive analytics. With IoT, parking operators can make data-driven decisions to optimize parking resources and enhance overall efficiency.

Finding an available parking spot can be frustrating. Intelligentparkingguidance systems use IoT data to direct drivers to vacant spaces quickly. Mobile apps, digital signage, and in-car navigation systems provide real-time guidance, reducing search time and traffic congestion. With IoT-enabled guidance, drivers can have a stress-free parking experience

Traditional parking payment methods often involve manual transactions and long queues. IoT-enabledsmart parking solutions offer seamless payment mechanisms. Integrated mobile payment apps and automatic billing systems make the payment process convenient and efficient. By eliminating the need for physical tickets or cash, IoT simplifies the parking payment experience for both drivers and parking operators.

The adoption of IoT for smart parking brings numerous benefits. It optimizes parking space utilization, reduces traffic congestion, and enhances user experience. Drivers save time and fuel by quickly finding parking spots, while parking operators gain insights for efficient resource allocation. With IoT, revolutionize parking efficiency and create a seamless parking ecosystem

Implementation:

PROGRAMMING THE PIC MICROCONTROLLER:

The following steps should be followed to embed the code into the PIC microcontroller,

STEP 1: Download and install the PIC C Compiler

STEP 2: Open the software and select File->New->Source File

STEP 3: Write the code, compile it and run.

STEP 4: Then dump the code into pic microcontroller using pic kit loader

PROGRAMMING THE RASPBERRY PI:

The following steps should be followed to program the raspberry pi

STEP 1: Establish the cloud connection, signup for an Real VNC account

STEP 2: Download VNC viewer and use same credentials to signup.

STEP 3: On the raspberry pi , run vnc server and make note of the IP address and then enter into the host.

STEP 4: Enable the features that we need

STEP 5: Then the raspberry pi is coded in the VNC viewer using python.

OPENING PUTTY:

STEP 1: Download and install the putty software

STEP 2: Open the putty software and specify the serial line as connection type

STEP 3: Then it enters into putty and it is just like a hyper terminal

STEP 4: The data from the zigbee receiver is displayed here when it is connected with the system.

IMAGE PROCESSING ANALYSIS:

As the number of vacant spaces will be displayed in software as the sensor sends the data and vacant spaces will get reduced when it captures the number plate. When the vehicle enters the parking area the number plate of the vehicle is captured and recognised with the help of camera and raspberry pi respectively. When the number plate is recognised it is automatically allocated to the vacant slot available. The recognition of the number plate produces the image thresh in which it is segmented. After this in the VNC Viewer it displays the command in the python shell as license plate is detected

RESULTS IN PUTTY :

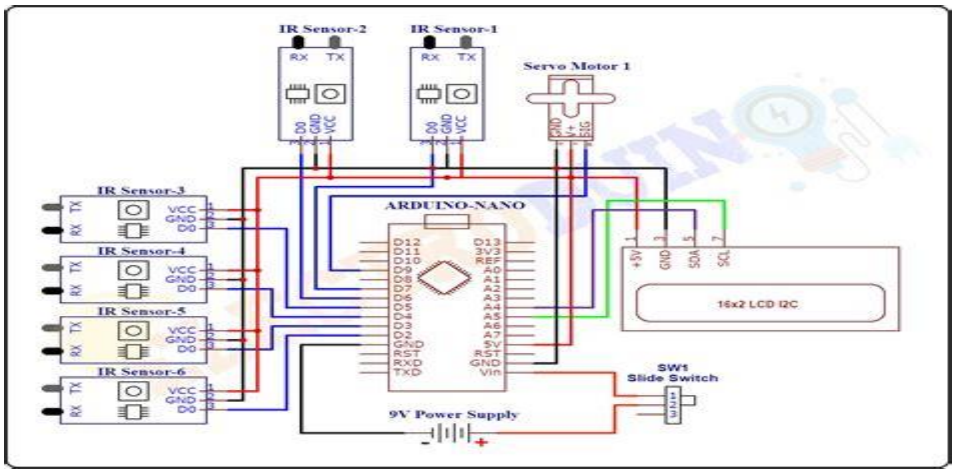
As when the license plate is detected it sends data to the putty software and displays the detected number plate and also shows the slot number in which the driver has to park their own vehicle from that time the timer gets started for payment purposes. When the vehicle is parked in the allocated slot and when it exited from the parking area the same image processing is made. After this the timer gets stopped and displays the amount to be paid. These timing and the payment details are updated to the server.

UPDATION IN THE SERVER:

When the information is sent from the sensors and the raspberry pi to the pic microcontroller and the same data is updated to the IoT server. We can login to it to find the slot information such as number of entries and its corresponding cost that to be paid

In this field chart the slot information is displayed based on the entry and the exit of the vehicle. The x-axis shows the time and the y-axis shows the cost. We can have the data which is stored in the server for future purposes. The field data feed can be extracted into various formats like csv and xml format. Each field can also be extracted individually.

CIRCUIT DIAGRAM:



SOURCE CODE:

4.1.FRONT END CODE:

```
#define ECHO_PIN1 15 //Pins for Sensor 1

#define TRIG_PIN1 2 //Pins for Sensor 1

#define ECHO_PIN2 5    //Pins for Sensor 2
#define TRIG_PIN2 18   //Pins for Sensor 2

#define ECHO_PIN3 26  //Pins for Sensor 3
#define TRIG_PIN3 27   //Pins for Sensor 3


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, LOW);
int duration = pulseIn(ECHO_PIN2, HIGH);
return duration * 0.034 / 2;
}

float readDistance3CM() {
digitalWrite(TRIG_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 = distance3 > 200;
```

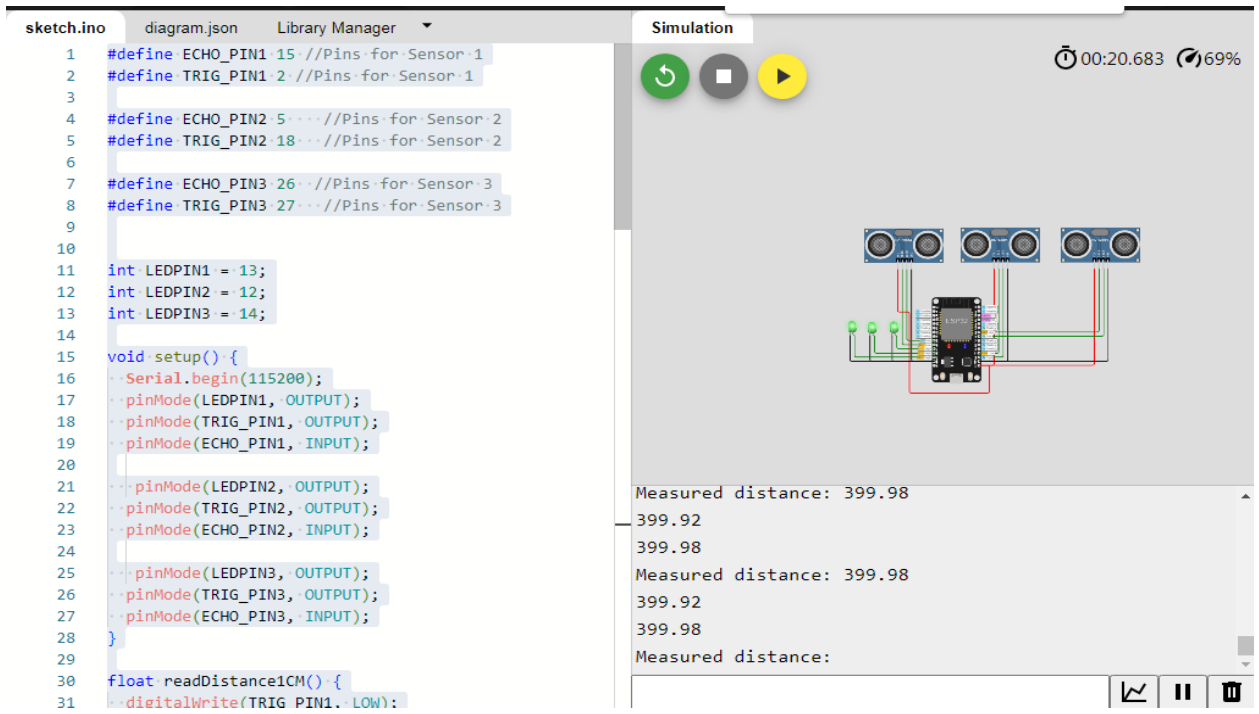
```
digitalWrite(LEDPIN3, isNearby3);

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

4.2.BACK END CODE:

```
{
"version": 1,
"author": "Surya K",
"editor": "wokwi",
"parts": [
{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 168.01, "left": -54.47, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 10.18, "left": 222.47, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic2", "top": 10.18, "left": 7.37, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic3", "top": 11.1, "left": -199.42, "attrs": {} },
{
"type": "wokwi-led",
"id": "led1",
"top": 215.93,
"left": -245.43,
"attrs": { "color": "green" }
},
{
"type": "wokwi-led",
"id": "led2",
"top": 217.94,
"left": -202.14,
"attrs": { "color": "green" }
},
{
"type": "wokwi-led",
"id": "led3",
"top": 216.99,
```

"left": -154.69,
"attrs": { "color": "green" }
}
],
"connections": [
["esp:TX0", "$serialMonitor:RX", "", []],
["esp:RX0", "$serialMonitor:TX", "", []],
["ultrasonic1:VCC", "esp:3V3", "red", ["v0"]],
["ultrasonic1:GND", "esp:GND.1", "black", ["v0"]],
["ultrasonic2:VCC", "esp:3V3", "red", ["v0"]],
[
"ultrasonic3:VCC",
"esp:3V3",
"red",
["v97.89", "h25.31", "v186.97", "h171.78", "v-63.59"]
],
["ultrasonic3:GND", "esp:GND.2", "black", ["v0"]],
["led1:C", "esp:GND.2", "black", ["v0"]],
["led2:C", "esp:GND.2", "black", ["v0"]],
["led3:C", "esp:GND.2", "black", ["v0"]],
["led1:A", "esp:D13", "green", ["v0"]],
["led2:A", "esp:D12", "green", ["v0"]],
["led3:A", "esp:D14", "green", ["v0"]],
["ultrasonic3:TRIG", "esp:D27", "green", ["v0"]],
["ultrasonic3:ECHO", "esp:D26", "green", ["v0"]],
["ultrasonic2:GND", "esp:GND.1", "black", ["v0"]],
["ultrasonic2:ECHO", "esp:D15", "green", ["v0"]],
["ultrasonic2:TRIG", "esp:D2", "green", ["v0"]],
["ultrasonic1:ECHO", "esp:D5", "green", ["v0"]],
["ultrasonic1:TRIG", "esp:D18", "green", ["v0"]]
],
"dependencies": {}
}

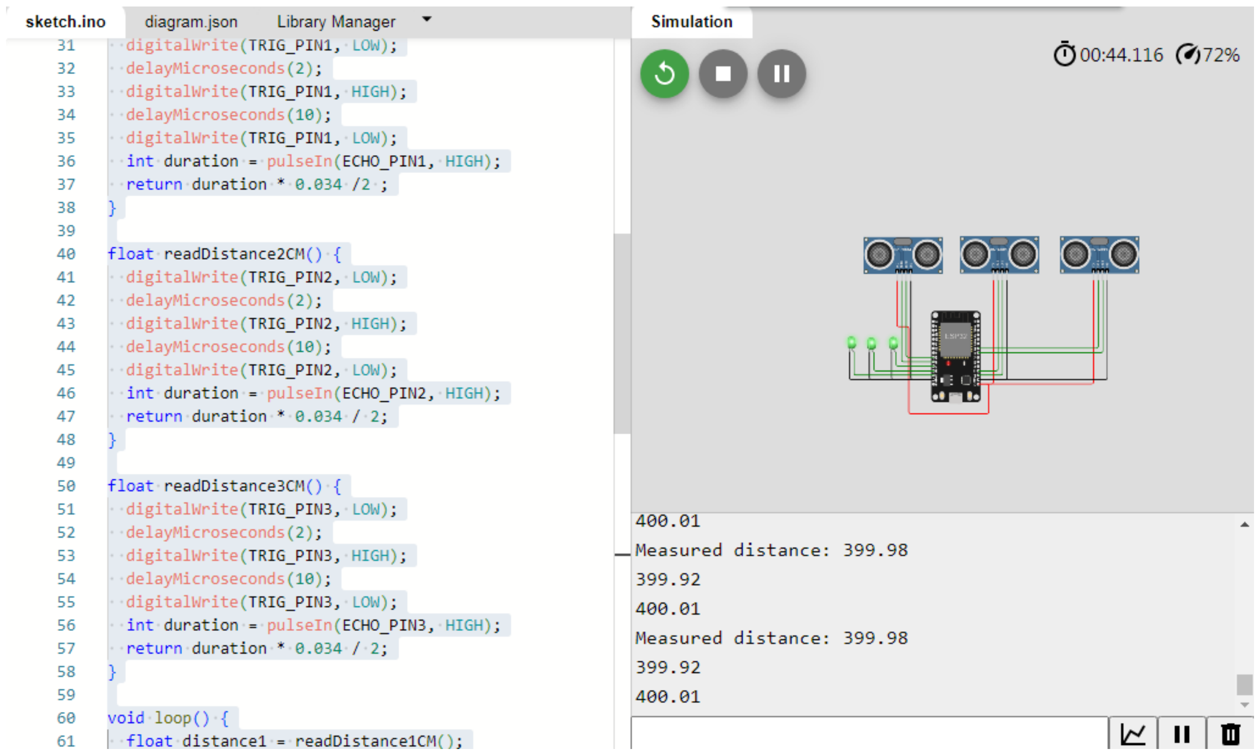
SAMPLE OUTPUT SCREENSHOT:



```
#define ECHO_PIN1 15 //Pins for Sensor 1
#define TRIG_PIN1 2 //Pins for Sensor 1

#define ECHO_PIN2 5    //Pins for Sensor 2
#define TRIG_PIN2 18   //Pins for Sensor 2

#define ECHO_PIN3 26  //Pins for Sensor 3
#define TRIG_PIN3 27   //Pins for Sensor 3


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);
```

```
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
399.92
399.98
Measured distance:
```

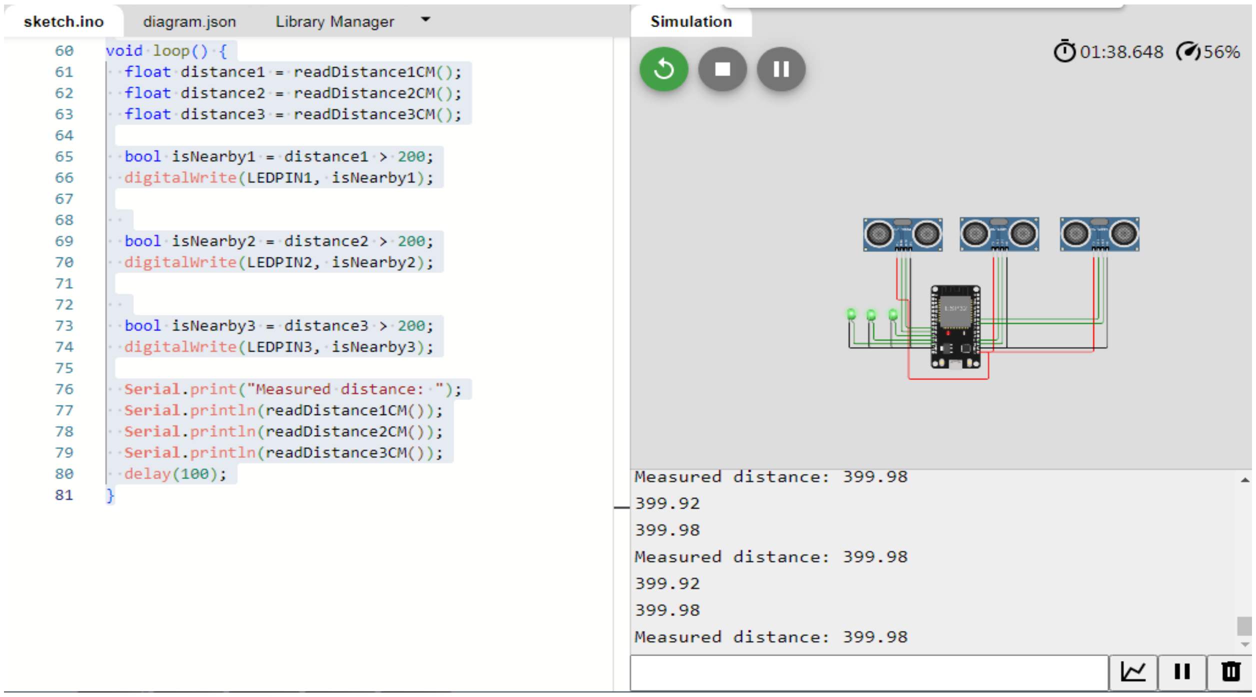
Fig:.4.1.1.initialize the code



```
  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, LOW);
  int duration = pulseIn(ECHO_PIN2, HIGH);
  return duration * 0.034 / 2;
}

float readDistance3CM() {
  digitalWrite(TRIG_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();
```

```
400.01
Measured distance: 399.98
399.92
400.01
Measured distance: 399.98
399.92
400.01
```

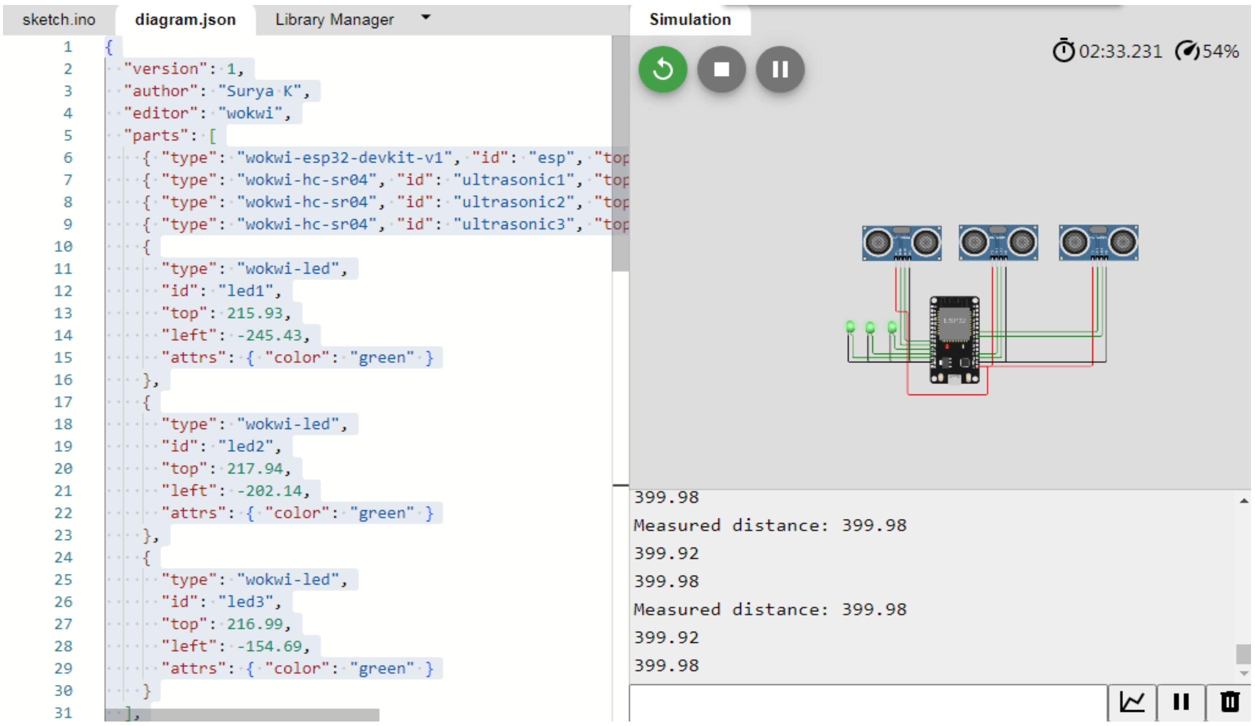
Fig:.4.1.2.processing



```
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 = distance3 > 200;
  digitalWrite(LEDPIN3, isNearby3);

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

```
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
```

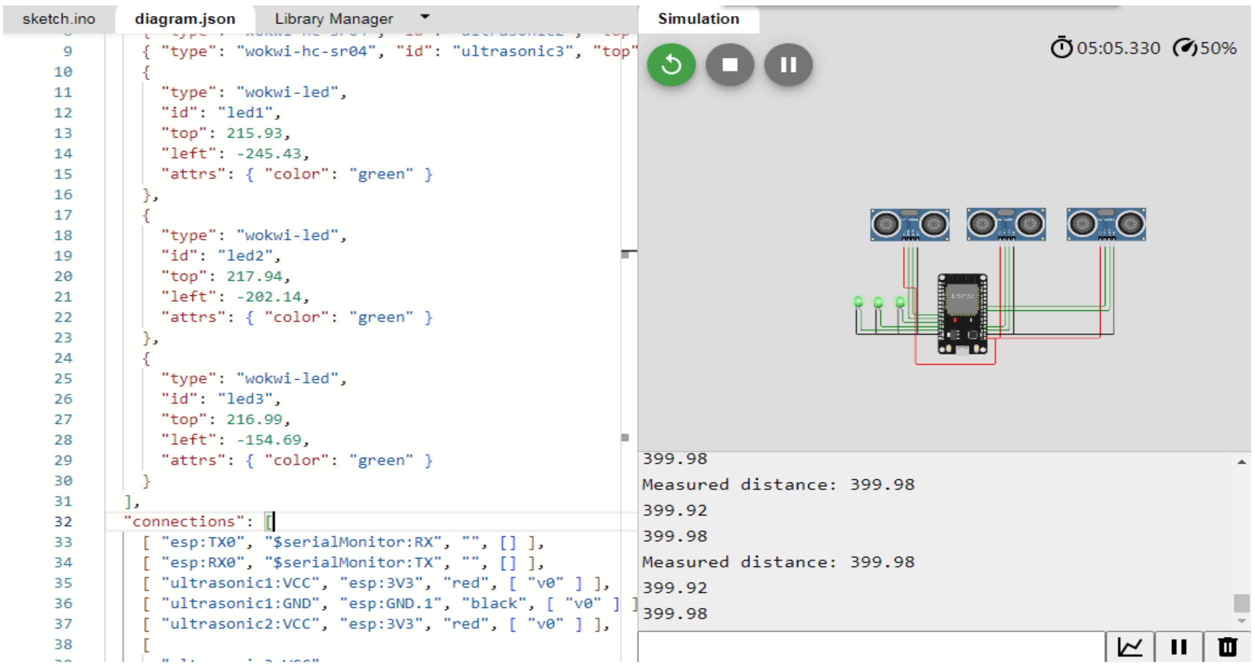
Fig:.4.1.3.ending the process



```
{
  "version": 1,
  "author": "Surya K",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top
    { "type": "wokwi-hc-sr04", "id": "ultrasonic2", "top
    { "type": "wokwi-hc-sr04", "id": "ultrasonic3", "top
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 215.93,
      "left": -245.43,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 217.94,
      "left": -202.14,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": 216.99,
      "left": -154.69,
      "attrs": { "color": "green" }
    }
  ],
```

```
399.98
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
399.92
399.98
```

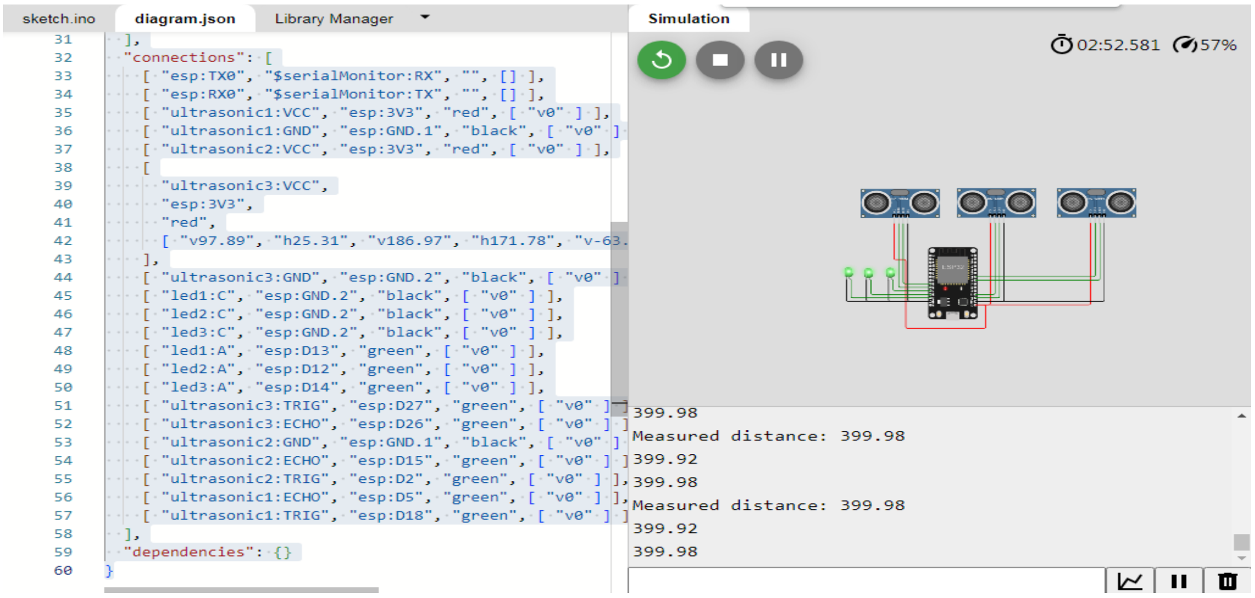
Fig:.4.2.1.inital state



```
    { "type": "wokwi-hc-sr04", "id": "ultrasonic3", "top"
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 215.93,
      "left": -245.43,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 217.94,
      "left": -202.14,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": 216.99,
      "left": -154.69,
      "attrs": { "color": "green" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]
    [ "ultrasonic2:VCC", "esp:3V3", "red", [ "v0" ] ],
    [
```

```
399.98
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
399.92
399.98
```

Fig:.4.2.2.processing



```
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ]
    [ "ultrasonic2:VCC", "esp:3V3", "red", [ "v0" ] ],
    [
      "ultrasonic3:VCC",
      "esp:3V3",
      "red",
      [ "v97.89", "h25.31", "v186.97", "h171.78", "v-63.
    ],
    [ "ultrasonic3:GND", "esp:GND.2", "black", [ "v0" ]
    [ "led1:C", "esp:GND.2", "black", [ "v0" ] ],
    [ "led2:C", "esp:GND.2", "black", [ "v0" ] ],
    [ "led3:C", "esp:GND.2", "black", [ "v0" ] ],
    [ "led1:A", "esp:D13", "green", [ "v0" ] ],
    [ "led2:A", "esp:D12", "green", [ "v0" ] ],
    [ "led3:A", "esp:D14", "green", [ "v0" ] ],
    [ "ultrasonic3:TRIG", "esp:D27", "green", [ "v0" ]
    [ "ultrasonic3:ECHO", "esp:D26", "green", [ "v0" ]
    [ "ultrasonic2:GND", "esp:GND.1", "black", [ "v0" ]
    [ "ultrasonic2:ECHO", "esp:D15", "green", [ "v0" ]
    [ "ultrasonic2:TRIG", "esp:D2", "green", [ "v0" ] ],
    [ "ultrasonic1:ECHO", "esp:D5", "green", [ "v0" ] ],
    [ "ultrasonic1:TRIG", "esp:D18", "green", [ "v0" ]
  ],
  "dependencies": {}
}
```

```
399.98
Measured distance: 399.98
399.92
399.98
Measured distance: 399.98
399.92
399.98
```

Fig:.4.2.3.Ending of the process