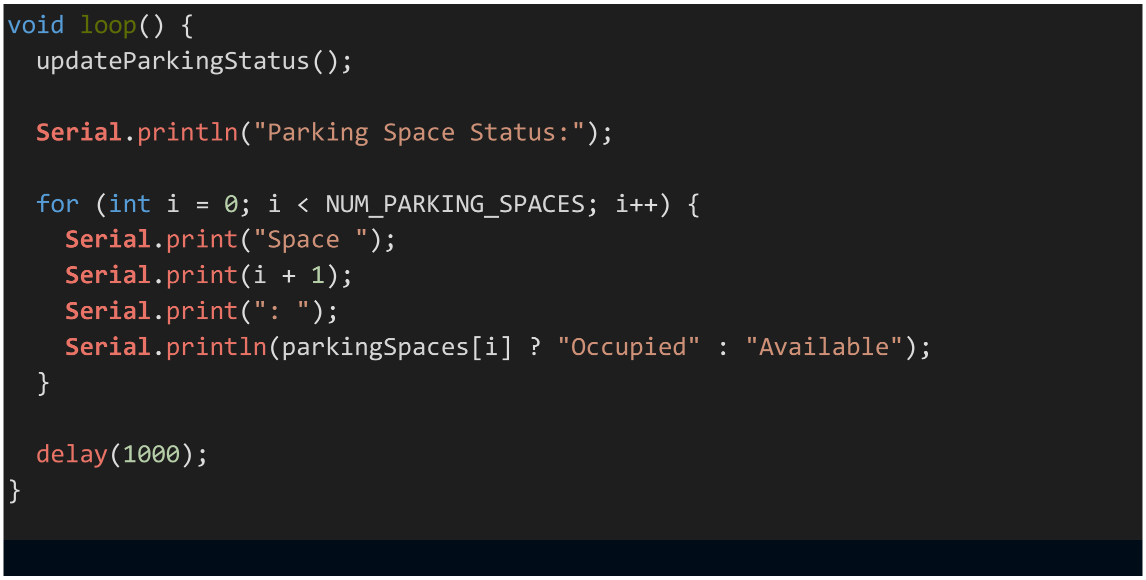
Phase 3: Development

IoT Sensor Integration and Raspberry Pi Configuration

Code Implementation using Wowki

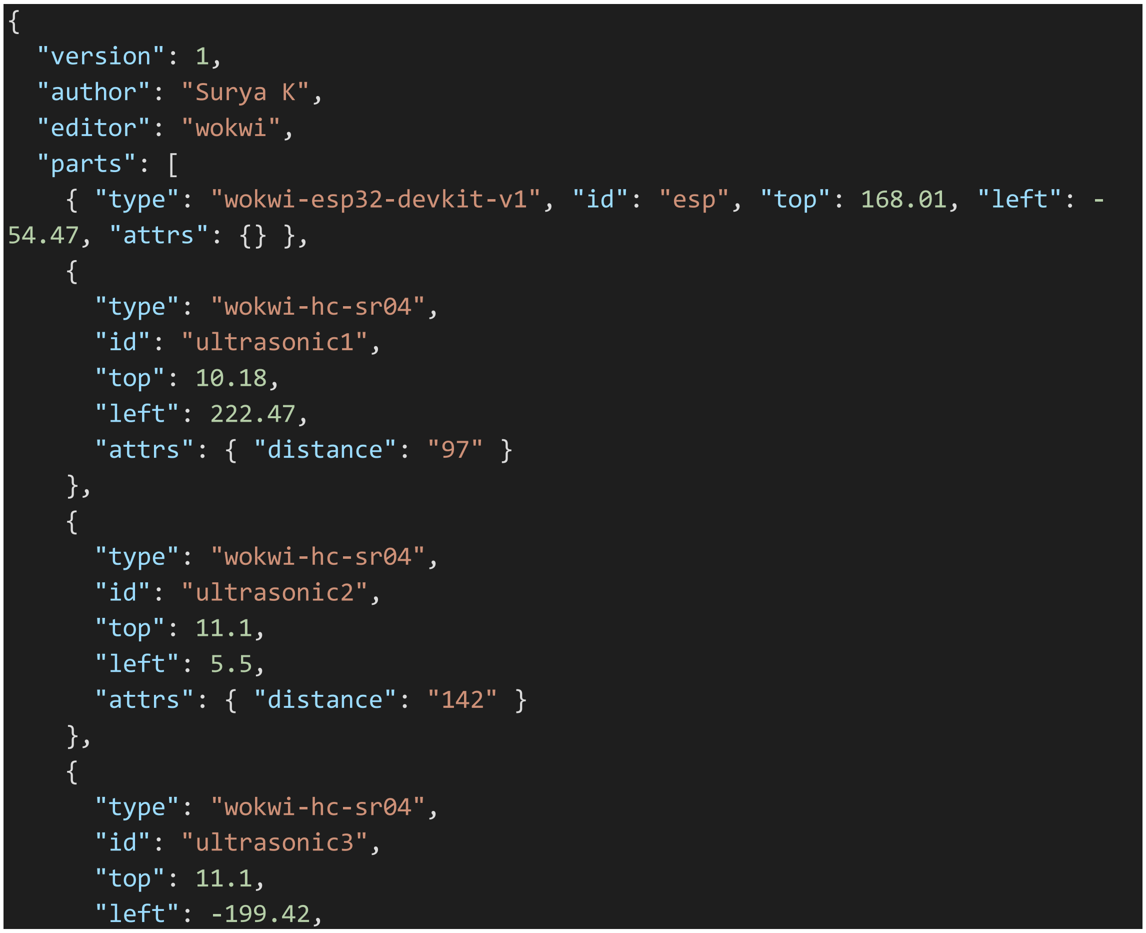
Sketch.ino

```
const int NUM_PARKING_SPACES = 3;

const int ECHO_PINS[NUM_PARKING_SPACES] = {15, 5, 26};
const int TRIG_PINS[NUM_PARKING_SPACES] = {2, 18, 27};
const int LED_PINS[NUM_PARKING_SPACES] = {13, 12, 14};

bool parkingSpaces[NUM_PARKING_SPACES] = {false, false, false};

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

  for (int i = 0; i < NUM_PARKING_SPACES; i++) {
    pinMode(ECHO_PINS[i], INPUT);
    pinMode(TRIG_PINS[i], OUTPUT);
    pinMode(LED_PINS[i], OUTPUT);
  }
}

float readDistanceCM(int TRIG_PIN, int ECHO_PIN) {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2 ;
}

// Data structure to keep track of the parking space status
void updateParkingStatus() {
  for (int i = 0; i < NUM_PARKING_SPACES; i++) {
    int distance = readDistanceCM(TRIG_PINS[i], ECHO_PINS[i]);
    parkingSpaces[i] = (distance < 200.0);
    digitalWrite(LED_PINS[i], parkingSpaces[i]);
  }
}
```

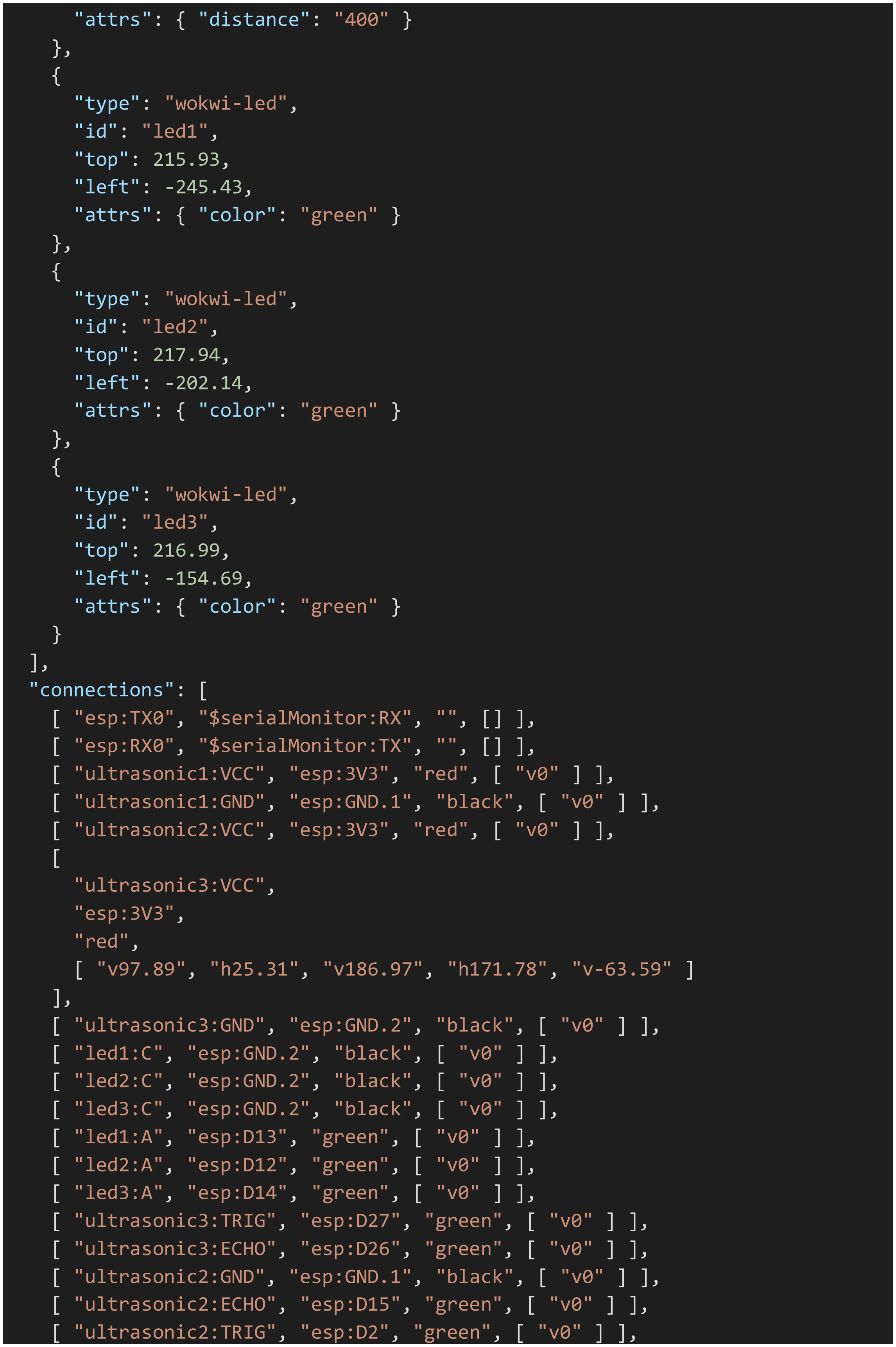


```
void loop() {
  updateParkingStatus();

  Serial.println("Parking Space Status:");

  for (int i = 0; i < NUM_PARKING_SPACES; i++) {
    Serial.print("Space ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.println(parkingSpaces[i] ? "Occupied" : "Available");
  }

  delay(1000);
}
```

.json:

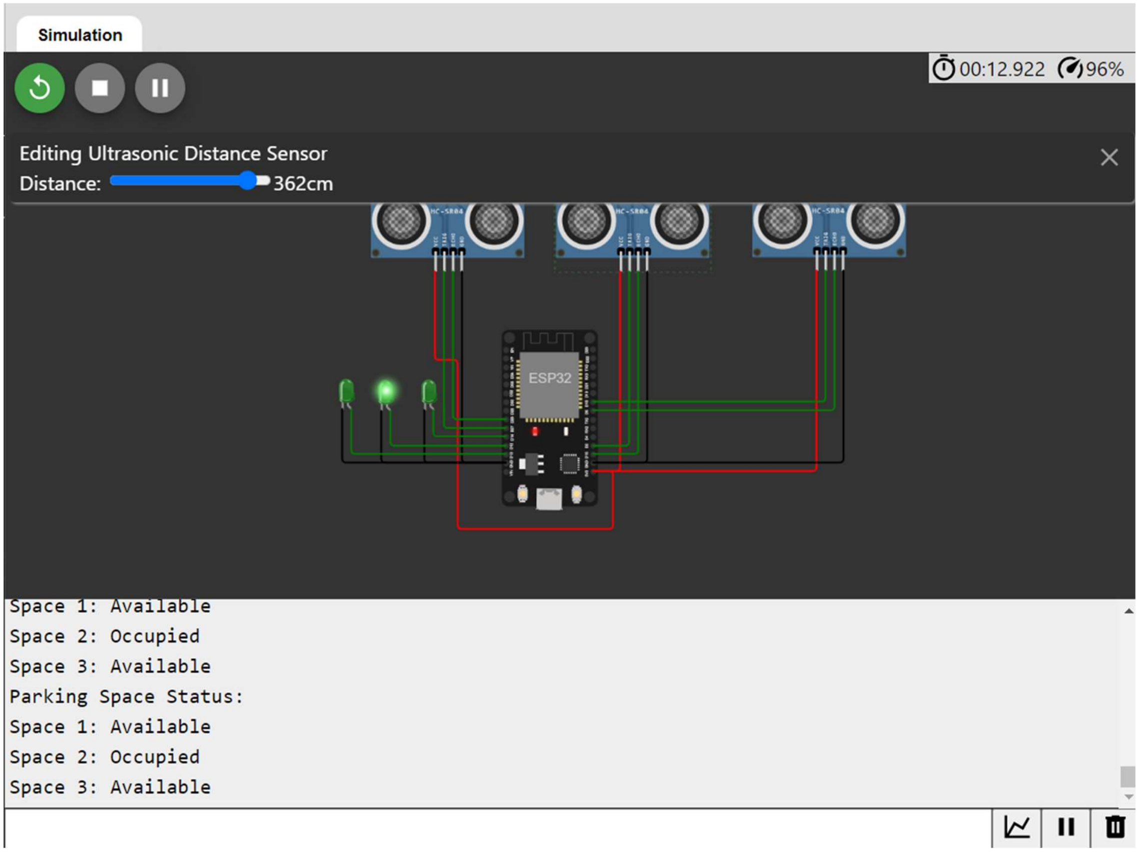


```
{
  "version": 1,
  "author": "Surya K",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 168.01, "left": -54.47, "attrs": {} },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": 10.18,
      "left": 222.47,
      "attrs": { "distance": "97" }
    },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic2",
      "top": 11.1,
      "left": 5.5,
      "attrs": { "distance": "142" }
    },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic3",
      "top": 11.1,
      "left": -199.42,
```



```
      "attrs": { "distance": "400" }
    },
    {
      "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": {}
}
```

Output:



Editing Ultrasonic Distance Sensor

Distance: 362cm

```
Space 1: Available
Space 2: Occupied
Space 3: Available
Parking Space Status:
Space 1: Available
Space 2: Occupied
Space 3: Available
```

In this phase, we successfully integrated IoT sensors with the Raspberry Pi to detect parking space occupancy. The Arduino code effectively manages sensor data, providing real-time status updates for each parking space.