Thursday, May 25, 2023 11:54 AM REPORT

Project Title: IoT-based Smart Parking System using ESP8266 and IR Sensors **1. Introduction:**

The IoT-based Smart Parking System is designed to monitor the occupancy status of parking slots and transmit the data to the cloud platform for further analysis and utilization. The system utilizes ESP8266, an affordable and widely-used Wi-Fi-enabled microcontroller, along with IR sensors to detect the presence of vehicles in parking slots. The status of each slot is then published to the Adafruit IO platform, allowing users to access real-time parking information remotely.

2. Components Used:

- ESP8266 board: It serves as the main microcontroller and provides Wi-Fi connectivity. - IR Sensors: Used to detect the presence of vehicles in parking slots.
- Jumper wires: To establish connections between the components.
- Breadboard or PCB: Provides a platform for mounting and connecting the components. **3. Schematics:**

The following schematic diagram illustrates the connection of the IR sensors to the ESP8266 board:

```
 +------------------+
| ESP8266 |
| |
| D1 (GPIO5)---|----IR Sensor 1
 | D2 (GPIO4)---|----IR Sensor 2
 | D5 (GPIO14)-|----IR Sensor 3
 | D6 (GPIO12)-|----IR Sensor 4
 +------------------+
```

4. Procedure of Connection:

Follow the steps below to establish the connections between the components:

1. Connect the VCC pin of each IR sensor to the 3.3V power supply pin on the ESP8266 board. 2. Connect the GND pin of each IR sensor to the GND pin on the ESP8266 board. 3. Connect the output pin of IR Sensor 1 to pin D1 (GPIO5) on the ESP8266 board. 4. Connect the output pin of IR Sensor 2 to pin D2 (GPIO4) on the ESP8266 board. 5. Connect the output pin of IR Sensor 3 to pin D5 (GPIO14) on the ESP8266 board. 6. Connect the output pin of IR Sensor 4 to pin D6 (GPIO12) on the ESP8266 board.

Ensure that the connections are secure and there are no loose connections or short circuits. **5. Software

Implementation:**

To implement the software part of the project, follow these steps:

1. Set up the Arduino IDE and install the required libraries: ESP8266WiFi, Adafruit_MQTT, and Adafruit_MQTT_Client.

2. Open the Arduino IDE and create a new sketch.
3. Copy the provided code into the sketch.
4. Modify the code to replace the placeholders with your Wi-Fi credentials (WLAN_SSID and WLAN_PASS) and Adafruit IO credentials (AIO_USERNAME and AIO_KEY). 5. Upload the code to the ESP8266 board.
6. Open the Serial Monitor to view the status and debug information.

6. System Operation:

Once the system is powered on and the code is uploaded successfully, the ESP8266 board starts executing the program. The IR sensors continuously monitor the occupancy status of the parking slots. Whenever a change in status is detected, the corresponding value is published to the Adafruit IO platform using MQTT.

The status of each parking slot can be monitored by accessing the Adafruit IO platform. The data can be visualized in real-time, allowing users to check the availability of parking slots remotely.

7. Conclusion:

The IoT-based Smart Parking System **8. Code:**

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#define WLAN_SSID "YourWiFiSSID"
#define WLAN_PASS "YourWiFiPassword"

#define AIO_SERVER "io.adafruit.com"
#define AIO_SERVERPORT 1883
#define AIO_USERNAME "YourAdafruitIOUsername"
#define AIO_KEY "YourAdafruitIOKey"

WiFiClient client;

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/Adafruit_MQTT_Publish slot1 =

Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/port1");

/****************************** IR Sensor Configuration
***********************************/

const int sensorPin1 = D1; // Digital input pin for sensor 1 const int sensorPin2 = D2;
// Digital input pin for sensor 2 const int sensorPin3 = D5; // Digital input pin for sensor
3
```



```
const int sensorPin3 = D5; // Digital input pin for sensor 3 const int sensorPin4 = D6;
// Digital input pin for sensor 4

/****************************** IR Sensor Configuration
```

```
**********************************/

void MQTT_connect();

void setup() {
 Serial.begin(115200);
  delay(10);

  Serial.println(F("Adafruit MQTT demo"));
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
 delay(500);
    Serial.print(".");
 }
  Serial.println();

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  pinMode(sensorPin1, INPUT);
  pinMode(sensorPin2, INPUT);
  pinMode(sensorPin3, INPUT);
  pinMode(sensorPin4, INPUT);
}

void loop() {
 MQTT_connect();

  bool carDetected1 = digitalRead(sensorPin1);
  bool carDetected2 = digitalRead(sensorPin2);
  bool carDetected3 = digitalRead(sensorPin3);
  bool carDetected4 = digitalRead(sensorPin4);

  if (carDetected1) {
 if (!slot1.publish(1)) {
 Serial.println(F("Failed to publish slot 1 status"));
    }
 } else if (carDetected2) {
    if (!slot1.publish(2)) {
 Serial.println(F("Failed to publish slot 2 status"));
    }
 } else if (carDetected3) {
    if (!slot1.publish(3)) {
 Serial.println(F("Failed to publish slot 3 status"));
    }
```

```
 }
 } else if (carDetected4) {
   if (!slot1.publish(4)) {
 Serial.println(F("Failed to publish slot 4 status"));
   }
 }

  delay(2000);

  // ping the server to keep the MQTT connection alive if (!mqtt.ping()) {
 mqtt.disconnect();
  }
}

void MQTT_connect() {
 int8_t ret;

  if (mqtt.connected()) {
 return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) {
 Serial.println(mqtt.connectErrorString(ret));
   Serial.println("Retrying MQTT connection in 5 seconds...");
   mqtt.disconnect();
   delay(5000);
   retries--;
   if (retries == 0) {
 while(1);
   }}}
```