SMART PARKING SYSTEM

COMPONENTS USED:

- ESP8266MOD
- 2 resistors of 220ohms
- 2 leds (red and blue)
- 1 servo motor sg90
- 3 IR sensors
- Breadboard and jumper wires
- 1 micro USB data cable

STEP 1: SETUP THE CIRUIT

- On Breadboard, create a power rail using jumper wires to connect the power bank(or any battery which have output of 5V-3A) to the breadboard.
- Connect the VCC and GND pins of ESP8266MOD to the power rail + and - respectively.
- Connect the VCC and GND pins of each IR sensor to the power rail + and - respectively. Connect the OUT pin of first IR sensor to D1 pin of ESP8266MOD, second IR sensor to D2 pin of ESP8266MOD and third IR sensor to D3 pin of ESP8266MOD.
- Connect the servo motor's VCC and GND pins to the power rail + and - respectively. Connect the signal pin of servo motor to D4 pin of ESP8266MOD.
- Connect the anode (longer leg) of red LED to D5 pin of ESP8266MOD through a 220 ohm resistor and cathode (shorter leg) to GND rail.
- Connect the anode (longer leg) of blue LED to D6 pin of ESP8266MOD through a 220 ohm resistor and cathode (shorter leg) to GND rail.

STEP 2: UPLOAD THE CODE

- Open Arduino IDE and install the ESP8266 board from the board manager if you haven't already by going to Files -> Preferences -> Additional Board Manager URLs and adding this URL: https://arduino.esp8266.com/stable/package_esp8266com_index.json
- Go to Tools -> Board -> Board Manager, search for "ESP8266" and install "ESP8266 By ESP8266 Community".
- Go to Tools -> Board and select "NodeMCU 1.0 (ESP-12E Module)".

- Install the Servo library by going to Sketch -> Include Library -> Manage Libraries, searching for "Servo" and installing it.
- Same for "ESP8266WiFi" library.
- Same for "PubSubClient" library.
- Copy and paste the code from smart_parking_cpp_code_version_mqtt.ino into the Arduino IDE.
- Replace the placeholders for WiFi SSID, WiFi Password, MQTT Broker address, MQTT Username, and MQTT Password with your actual credentials in the code.
- To get those parameters you need to set up a MQTT broker. You can use free brokers like Mosquitto MQTT and follow this few steps:
  i. Go to https://mosquitto.org/download/
  ii. Download and install the broker according to your OS.
  iii. Run the broker on your local machine or server.
  iv. Connect the laptop in which the broker is installed and the ESP8266 to the same WiFi network.
  v. Go to command prompt and type "ipconfig" to get the IPv4 address of your machine. This will be your MQTT Broker address.
  vi. Go to mosquitto installation directory and open the "mosquitto.conf" file in a text editor with admin access and make sure below lines are present and uncommented:

  ```
  listener 1883 0.0.0.0
  allow_anonymous true
  ```

  vii. Save the file and restart the mosquitto broker.
  viii. Now check whether the broker is working or not by using Admin Powershell commands like:

  ```
  Get-Service mosquitto
  ```

  If the service is running, then your broker is set up successfully and If not run it manually.

  ```
  Start-Service mosquitto
  ```

  Again check the status using Get-Service command and Even if it is not running, then their might be some error in the configuration file. Check the log file in the installation directory for errors.
  ix. Now you can use the IPv4 address of your machine as MQTT Broker address in the code. You can keep the username and password fields blank if you have set "allow_anonymous true" in the configuration file.
  x. Now you have got MQTT Broker address.

xi. For WiFi SSID and Password, use the credentials of the WiFi network to which your Laptop is connected.
For eg-If your WiFi SSID is "Home_Network" and password is "mypassword123", then replace the placeholders in the code as:

```
const char* ssid = "Home_Network";
const char* password = "mypassword123";
```

- Connect your ESP8266MOD to your computer using a micro USB data cable.
- In Arduino IDE, go to Tools -> Port and select the port to which your ESP8266MOD is connected.
- Click on the Upload button (right arrow icon) to compile and upload the code to your ESP8266MOD.

STEP 3: NODE-RED SETUP

- Install Node-RED on your system by following the instructions at https://nodered.org/docs/getting-started/local
- Open Node-RED by running the command `node-red` in your terminal or command prompt.
- Open your web browser and go to `http://localhost:1880` to access the Node-RED interface.
- In the Node-RED interface, install the MQTT nodes by going to the menu (top right corner) -> Manage palette -> Install tab, then search for "node-red-contrib-mqtt-broker" and click on "Install".
- Import the provided Node-RED flow(nodered_dashboard.json) for the smart parking system by clicking on the menu -> Import -> Clipboard, then paste the flow JSON and click "Import".
- Double-click on the MQTT input nodes in the flow to configure them. Set the server to your MQTT broker address (the same one used in the ESP8266 code[mqtt_server]) and set the topic to match the topics used in the ESP8266 code.
  i.e. The node (Vehicle Object Detection) should have topic as "parking/object" and for node (Status MQTT) should have topic as "parking/status".
- (Optional Step to Debug) Drag and drop a debug and mqtt input node from the left sidebar to the workspace.
  Set the topic of mqtt input to "#" to subscribe to all topics. Connect the output of mqtt input node to the input of debug node.
- Keep the debug tab(a lady bug logo) open (right sidebar) to see the incoming messages from the ESP8266MOD.
- Deploy the flow by clicking on the "Deploy" button (top right corner).

STEP 4: TEST THE SYSTEM

1. MQTT TESTING USING NODE-RED DEBUG TAB

- Open the debug tab (a lady bug logo) in the right sidebar to see the incoming messages.
- Open a another pc connected to the same network as the MQTT broker.
- Install Mosquitto MQTT client on the PC.
- Open Admin Powershell on that PC and use the following command to subscribe to the topics:

  ```
  .\mosquitto_pub -h <MQTT_BROKER_IP> -t "parking/#" -m"hello from 2nd PC"
  ```

  Replace `<MQTT_BROKER_IP>` with the actual IP address of your MQTT broker.
- Now, In node-RED debug tab, you should see the message "hello from 2nd PC" indicating that the MQTT communication is working.
- If not and you are sure that mqtt is running properly, then check the firewall settings of your system and allow the mosquitto broker port(1883) through it.

2. Node-RED Testing

- Open the Node-RED dashboard by going to `http://localhost:1880/ui` in your web browser.
- You should see the smart parking system interface with status indicators.
- Go to '/' and check whether the mqtt broker is connected or not. If connected, it will show "Connected" in green color.
- If not connected, check the MQTT broker settings in the MQTT input nodes and ensure that the broker is running.

3. SMART PARKING SYSTEM TESTING

- Power on the ESP8266MOD by connecting it to the power bank or your power source.
- Wait for a few seconds for the system to initialize and let esp8266mod gets connected to your Wifi(You can check the device connected option if you are using your mobile as hotspot).
- Place an object (like a book or a box) in front of the first IR sensor to simulate a vehicle approaching the parking area.
- Place the object in front of first IR sensor, then led 1 must glow, and if you place the object in front of both IR sensor(1 and 2), then led 2 must glow.
- If first led glows means it bike and if second led glows means its a car.
- Now place the object in front of the third IR sensor to simulate the vehicle reaching the parking spot. The servo motor should rotate to open the gate, allowing the vehicle to "park".
- Simultaneously, check the Node-RED dashboard (accessible at `http://localhost:1880/ui`) to see the status updates of the parking system.

- Also check the debug tab in Node-RED to see the messages being published by the ESP8266MOD.

CONCLUSION

- You have successfully set up a smart parking system using ESP8266MOD, IR sensors, a servo motor, and MQTT communication with Node-RED for monitoring.
- This system can be further enhanced by adding features like real-time notifications, multiple parking spots, and integration with mobile applications for a complete smart parking solution.

REFERENCES

- https://nodered.org/docs/getting-started/local
- https://mosquitto.org/documentation/
- https://arduino.esp8266.com/stable/package_esp8266com_index.json
- https://www.arduino.cc/en/Guide/ArduinoESP8266
- https://www.arduino.cc/en/Reference/Servo
- https://pubsubclient.knolleary.net/

CONTRIBUTORS

- Omkar Chandgaonkar (https://github.com/11byte)
- Aditya Chaudhari (https://github.com/Classadi)
- Sanika Ghorad (https://github.com/Sanikaghorad05)
- Paresh Gupta (https://github.com/Paresh-0007)

LICENSE

ACKNOWLEDGEMENTS

- We would like to thank the open-source community for providing valuable resources and libraries that made this project possible.