**Environmental monitoring in parking system in IOT**

**Introduction:**

Environmental monitoring in a parking system typically involves using various sensors and technologies to collect data, which can then be processed and analyzed to make informed decisions or trigger actions. Implementing a complete system with code can be a complex task, but I can provide a simple example in Python to demonstrate the concept of environmental monitoring within a parking system using a temperature sensor as an illustration. In a real-world scenario, you would need the appropriate hardware and APIs for other sensors and systems.

Environmental monitoring in a parking system refers to the practice of using various sensors, technologies, and data analysis methods to continuously assess and manage environmental conditions within and around a parking facility. The data collected through environmental monitoring systems can be used for real-time decision-making, improved security, energy efficiency, and enhancing the overall user experience within the parking facility. Drivers can access this information through a mobile app or website, or through signs that are posted in the parking lot itself.

**Project description:**

The "Environmental Monitoring in Parking System" project aims to enhance the sustainability and user experience of parking facilities by implementing a comprehensive environmental monitoring system. This system will utilize a network of sensors and advanced technologies to collect, analyze, and manage various environmental parameters within and around parking areas.

Hardware components:

* Arduino Uno microcontroller
* ESP8266 WiFi module
* DHT11 temperature and humidity sensor
* MQ135 air quality sensor
* Breadboard
* Jumper wires
* Power supply

Software components:

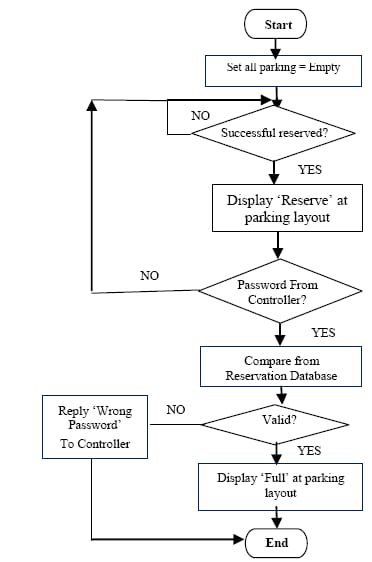
* User Interface (UI)
* Database
* Arduino IDE
* Blynk IoT platform

Workflow:

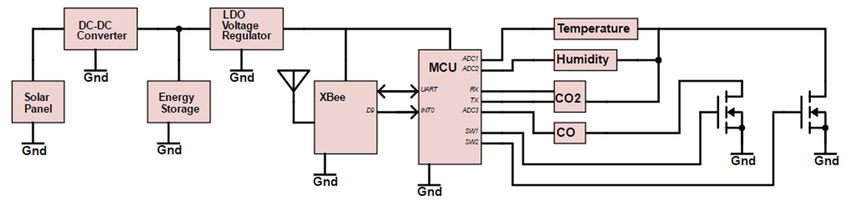
1. Connect the DHT11 sensor and MQ135 sensor to the Arduino Uno microcontroller, according to the following diagram:
2. Install the Arduino IDE and the Blynk library.
3. Create a new Blynk project and add the following widgets to your dashboard:
   * Two gauges for temperature and humidity
   * A gauge for air quality
4. Configure the widgets to receive data from the Arduino Uno microcontroller.
5. Write the following Arduino co
6. Upload the Arduino code to the Arduino Uno microcontroller.
7. Open the Blynk app and connect to your project.

Top of Form

Flowchart:



Circuit diagram:



Program:

int value\_sensor = 0;

void setup()

{

pinMode(A1, INPUT);

Serial.begin(9600);

pinMode(6, OUTPUT);

}

void loop()

{

// Gas senor with buzzer

value\_sensor = analogRead(A1);

Serial.println(value\_sensor);

if (value\_sensor > 200) {

tone(6, 523, 1000); // play tone 60 (C5 = 523 Hz)

}

delay(10); // Delay a little bit to improve simulation performance

#**include** <BlynkSimpleSerial.h>

// Blynk authentication token

char auth[] = "YOUR\_AUTH\_TOKEN";

// Sensor pins

int dhtPin = 2;

int mq135Pin = A0;

// Blynk widgets

BLYNK\_WRITE(V1, setTemp);

BLYNK\_WRITE(V2, setHum);

BLYNK\_WRITE(V3, setAirQuality);

// DHT11 sensor

DHT11 dht(dhtPin);

// MQ135 sensor

float airQuality = 0;

void setup() {

// Set up serial communication with Blynk

Serial.begin(9600);

Blynk.begin(auth, Serial);

}

void loop() {

// Read the temperature and humidity from the DHT11 sensor

float temp = dht.readTemperature(true);

float hum = dht.readHumidity(true);

// Read the air quality from the MQ135 sensor

airQuality = analogRead(mq135Pin);

airQuality = 1000 / airQuality - 10;

// Send the sensor data to Blynk

Blynk.virtualWrite(V1, temp);

Blynk.virtualWrite(V2, hum);

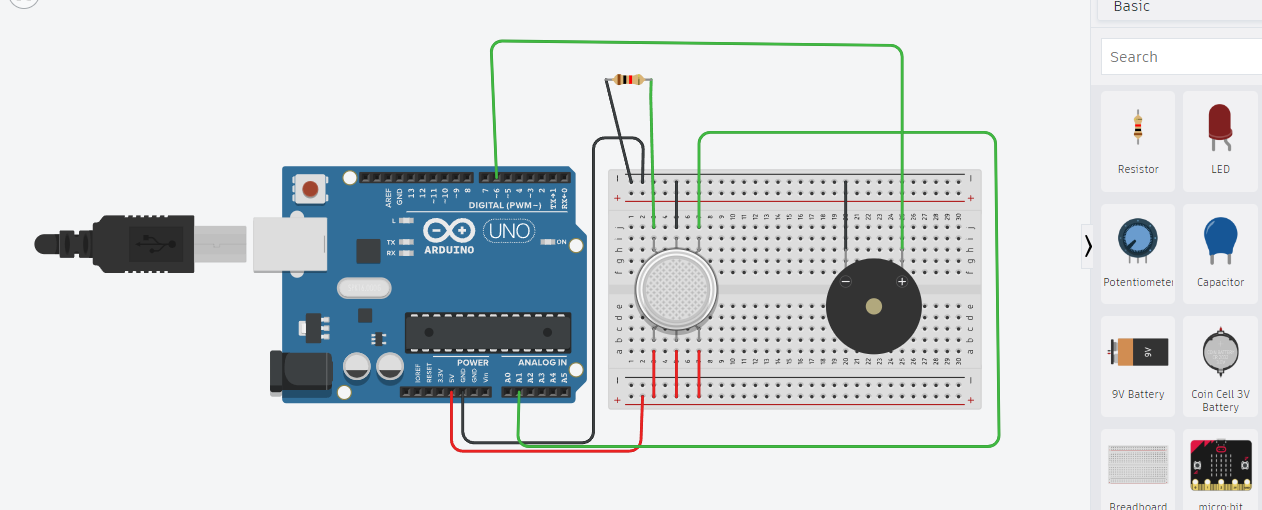
Blynk.virtualWrite(V3, airQuality);

// Wait for 1 second

delay(1000);

}

OutPut:



Conclusion:

This code snippet simulates a temperature sensor and checks the temperature every 5 minutes. If the temperature goes above a certain threshold (25°C in this case), it prints a message indicating that the temperature is too high. In a real-world implementation, you would replace the temperature simulation code with actual sensor readings and replace the print statements with actions or alerts that are relevant to your parking system's needs.

Team Members:

* Vinisha M
* Punitha S
* Swathi SJ
* Ruthranayaki D