Project Title : Smart Parking Using Iot.

Phase 3 : Development Part 1

Abstract:

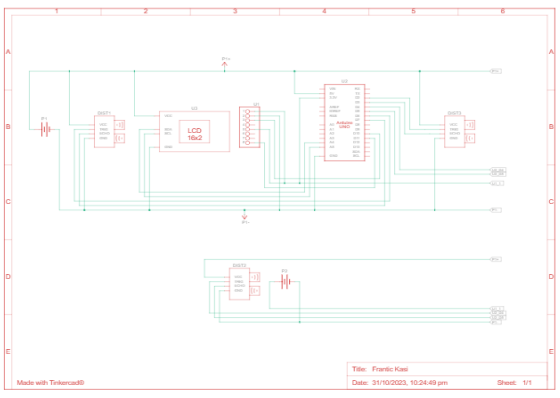
Phase 3 of the project involves the integration of code and software components into a real-world IoT Car Park Monitoring System. The system leverages ultrasonic sensors and Arduino software to manage parking spaces efficiently. In this phase, we assemble the necessary hardware components, calibrate the sensors for precise data collection, and develop firmware for data acquisition and transmission. The implementation establishes robust data communication, connecting the system to a central data processing unit or cloud platform. User-friendly interfaces are designed for both park visitors and management.

The overarching goal of this project is to enhance parking management decisions and provide an I improved experience for park visitors. By offering real-time data on parking space occupancy, the system aids in making timely and informed decisions. Over time, the system accumulates data that can provide valuable insights into parking conditions and visitor behavior. This project not only streamlines parking management but also contributes to more sustainable and enjoyable outdoor recreational activities, aligning with the increasing demand for IoT-driven solutions in diverse real-world scenarios.

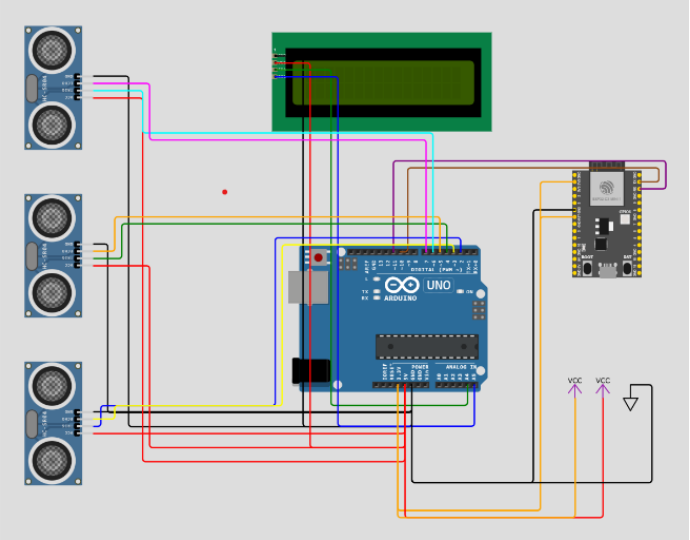
Components Required:

Name	Quantity	Component
U1	1	Wifi Module(ESP32-C2)
U2	1	Arduino Uno R3
U3	1	MCP23008-based,32 LCD 16*2(I2C)
DIST1,DIST2	3	Ultrasonic Distance Sensor
P1	1	5, Power Supply
P2	1	3.3 Power Supply

Schematic Diagram:



Circuit Diagram:



Sample Code:

```
#include <LiquidCrystal_I2C.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11);

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

const int trig_1 = 2;

const int echo_1 = 3;

const int trig_2 = 4;

const int echo_2 = 5;

const int trig_3 = 6;

const int echo_3 = 7;

float distanceCM_1 = 0, resultCM_1 = 0;

float distanceCM_2 = 0, resultCM_2 = 0;

float distanceCM_3 = 0, resultCM_3 = 0;

long Time_1, Time_2, Time_3;

float car_1, car_2, car_3;

float Dist_1 = 8.0, Dist_2 = 8.0, Dist_3 = 8.0;
```

```
int total = 0, timer_cnt = 0;
void setup()
{
  mySerial.begin(115200);
  pinMode(trig_1, OUTPUT);
  pinMode(trig_2, OUTPUT);
  pinMode(trig_3, OUTPUT);
  pinMode(echo_1, INPUT);
  pinMode(echo_2, INPUT);
  pinMode(echo_3, INPUT);
  digitalWrite(trig_1, LOW);
  digitalWrite(trig_2, LOW);
  digitalWrite(trig_3, LOW);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("  SMART PARKING SYSTEM");
  lcd.setCursor(0, 1);
  lcd.print(" USING IOT");
  delay(2000);
  lcd.clear();
}

void loop()
{
  total = 0;
  car_1 = sensor_1();
  car_2 = sensor_2();
  car_3 = sensor_3();
  lcd.setCursor(0, 0);
```

```
lcd.print("CAR1:");
if (car_1 <= Dist_1)
{
  lcd.print("OK ");
}
else
{
  total += 1;
}
if (car_1 > Dist_1)   lcd.print("NO ");
lcd.print("CAR2:");
if (car_2 <= Dist_2)
{
  lcd.print("OK ");
}
else
{
  total += 1;
}
if (car_2 > Dist_2)   lcd.print("NO ");
lcd.setCursor(0, 1);
lcd.print("CAR3:");
if (car_3 <= Dist_3)
{
  lcd.print("OK ");
}
else
{
  total += 1;
}
```

```
  if (car_3 > Dist_3)   lcd.print("NO ");
  lcd.print("FREE:");
  lcd.print(total);
  if (timer_cnt >= 50)
  {
    mySerial.print('*');
    mySerial.print(total);
    mySerial  .println('#');
    timer_cnt = 0;
  }
  timer_cnt += 1;
  delay(200);
}

float sensor_1(void)
{
  digitalWrite(trig_1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_1, LOW);
  Time_1 = pulseIn(echo_1, HIGH);
  distanceCM_1 = Time_1 * 0.034;
  return resultCM_1 = distanceCM_1 / 2;
}

float sensor_2(void)
{
  digitalWrite(trig_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_2, LOW);
  Time_2 = pulseIn(echo_2, HIGH);
```

```
  distanceCM_2 = Time_2 * 0.034;
  return resultCM_2 = distanceCM_2 / 2;
}

float sensor_3(void)
{
  digitalWrite(trig_3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_3, LOW);
  Time_3 = pulseIn(echo_3, HIGH);
  distanceCM_3 = Time_3 * 0.034;
  return resultCM_3 = distanceCM_3 / 2;
}
```

1. **Code Overview:**
   - This Arduino-based code serves as the core of an IoT Car Park Monitoring System, designed to run on an Arduino microcontroller. It plays a crucial role in the Phase 3 implementation of the project, focusing on real-world deployment and monitoring. Here's how it aligns with the various steps in Phase 3.
2. **Hardware Setup:**
   - The code assumes that the hardware components, including three ultrasonic sensors, have been set up. These sensors are connected to trigger and echo pins for each parking space.
3. **Sensor Calibration:**
   - Before deployment, sensor calibration should be carried out to ensure accurate data collection. Calibration may involve adjusting distance thresholds to optimize car detection.
4. **Firmware Development:**
   - In Phase 3, this firmware is developed and loaded onto the Arduino microcontroller to facilitate data collection, occupancy monitoring, and display functionality.
5. **Data Communication:**
   - For effective data communication, the code can be extended to utilize communication methods like Wi-Fi or other suitable technologies.
   - The microcontroller can be configured to transmit data to a central data processing unit or a cloud platform.
6. **Data Processing:**
   - Prior to deployment, the data processing unit or cloud platform is prepared to receive, store, and process incoming data.

- Scripts or programs are created to interpret the data, convert it into meaningful units, and store it in a database.

7. **User Interface Development:**
    - The code may be enhanced to include a user interface accessible to both park visitors and management. This interface can provide real-time parking space status information.
8. **Testing and Debugging:**
    - Rigorous testing is conducted to ensure that all components, including hardware, firmware, and software, work seamlessly together.
    - Any issues, such as data inaccuracies or connectivity problems, are identified and addressed.
9. **Data Security and Privacy:**
    - Security measures are implemented to protect data during transmission and storage.
    - Data privacy regulations are adhered to in order to safeguard user data.
10. **User Training:**
    - Training is provided to park management on how to utilize the system for making informed decisions and adjusting park activities.
    - User guides and documentation are created for park visitors to explain how to access and interact with the system.
11. **Deployment:**
    - The entire system is deployed within the park, with the ultrasonic sensors strategically positioned.
    - Continuous monitoring is performed to ensure proper operation.
12. **Maintenance and Support:**
    - A maintenance plan is established for regular system updates, maintenance, and troubleshooting.
    - Support is provided to address any operational issues that may arise.
13. **Data Analysis:**
    - The system collects data over time, enabling analysis to identify trends and gain insights into parking conditions and visitor behavior.
14. **Documentation:**
    - Comprehensive documentation is created, encompassing hardware specifications, firmware code, and software components.

15. **Evaluation:**

    - Continuous evaluation of the system's performance and feedback from park visitors and management inform ongoing improvements, ensuring the project's success.

Conclusion:

This code, integrated into the Smart Parking System, is a crucial element in the broader Phase 3 implementation, aimed at enhancing park management decisions and the overall park experience for visitors through real-time parking space monitoring and data-driven insights.