It seems like you want to convert the provided information on simulating an ESP32 and DHT22 sensor into a document for a smart parking project. Here's a modified introduction and the first few sections of your document:

Smart Parking System Project

Phase 3: Building and Developing with IoT Devices

In this phase of the Smart Parking System project, we will delve into the process of deploying IoT devices and developing a Python script to collect and process data for efficient parking management. Our objective is to create a smart parking system using IoT technology.

Introduction

The Smart Parking System project aims to optimize the parking experience by employing IoT devices to monitor parking spaces, provide real-time information to drivers, and enable efficient parking management. In this phase, we will demonstrate how to simulate an ESP32 microcontroller and a DHT22 sensor to collect environmental data relevant to parking space monitoring.

Simulating IoT Devices

In this section, we will guide you through the process of setting up the environment for simulating the ESP32 and DHT22 sensor, which will be the foundation of our smart parking system.

1. Accessing Wokwi Platform

To initiate this phase, we need to access the Wokwi platform, which serves as our virtual workspace for simulating the IoT devices. Follow these steps:

- Start by visiting the Wokwi website at wokwi.com.

2. Creating a New Project

Creating a new project on Wokwi is the initial step in setting up your virtual environment for the Smart Parking System. Here's how:

- Click the "New Project" button to create a new project. This project will serve as the workspace for your smart parking simulation.

3. Selecting the ESP32

Selecting the appropriate hardware component is crucial for your IoT project. In this case, we are simulating an ESP32 microcontroller. Follow these steps:

- In the "Select a board" section, type "ESP32" in the search bar and choose an ESP32 board model (e.g., ESP32 Dev Module). This selection emulates the hardware component for our smart parking system.

4. Adding the DHT22 Sensor

To measure environmental data relevant to parking space conditions, we'll simulate the use of a DHT22 sensor. Here's how to include it in your simulation:

- In the components panel on the left, search for "DHT22" or "DHT11."
- Drag and drop the DHT22 component onto the virtual breadboard area. This action replicates the physical process of connecting a DHT22 sensor to the ESP32 board.

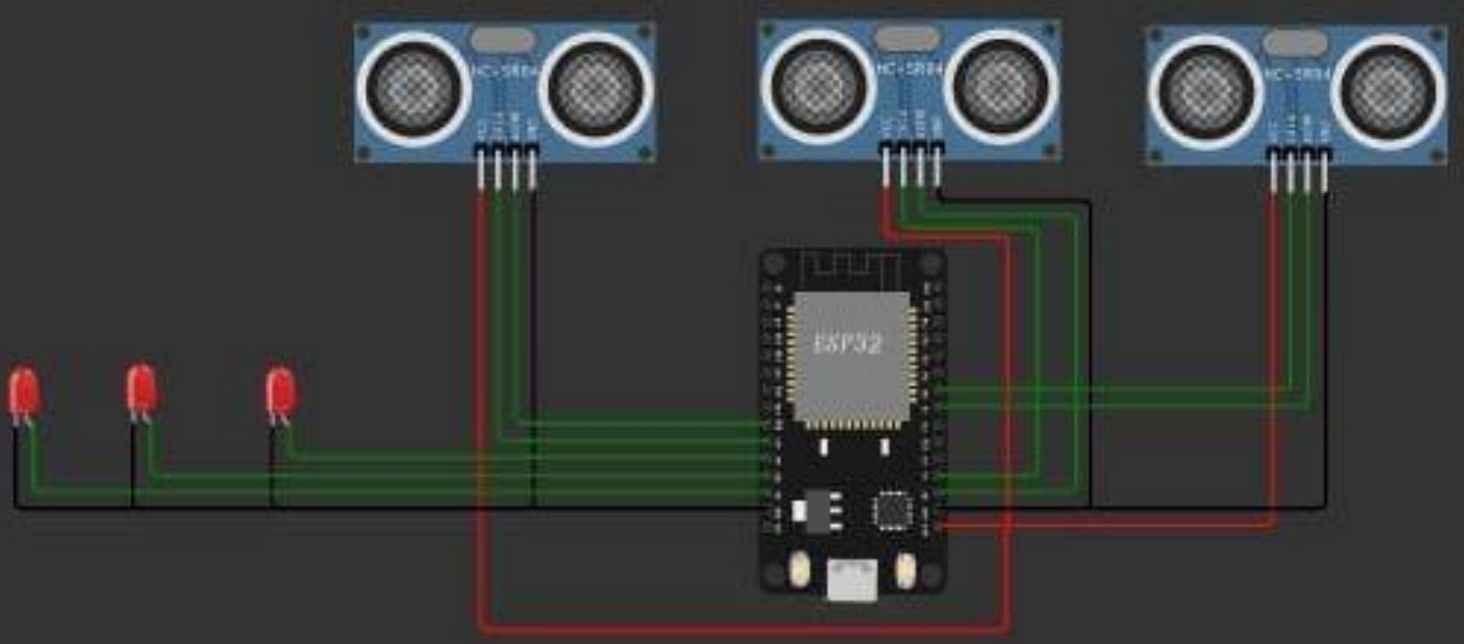
Stay tuned for the next sections where we'll detail the process of connecting the components, writing Python code, and running the simulation to collect essential data for our smart parking system.

```
.#include <Wire.h>
 #include <LiquidCrystal_I2C.h>
 LiquidCrystal_I2C lcd(0x27,16,2);
 #include <Servo.h>     Servo myservo1;    int IR1 = 4;
// IR Sensor 1    int IR2 = 7; // IR Sensor 2    int Slot =
4;      //Enter Total number of parking Slots    int
flag1 = 0;
 int flag2 = 0;
void setup()
```

```
{    lcd.init();       lcd.backlight();
pinMode(IR1,                INPUT);
pinMode(IR2,                INPUT);
myservo1.attach(9);
myservo1.write(100);
lcd.setCursor (0,0);    lcd.print("
ARDUINO    ");       lcd.setCursor
(0,1);          lcd.print("   PARKING
SYSTEM ");    delay (2000);
 lcd.clear();
 }
 void loop(){     if(digitalRead (IR1) == LOW &&
flag1==0){    if(Slot>0){flag1=1;
if(flag2==0){myservo1.write(0); Slot = Slot-1;}
 }else{           lcd.setCursor
(0,0);     lcd.print("   SORRY
:( ");    lcd.setCursor (0,1);
lcd.print(" Parking Full ");
delay (3000);    lcd.clear();
 }

--}
 if(digitalRead (IR2) == LOW && flag2==0){flag2=1;
if(flag1==0){myservo1.write(0); Slot = Slot+1;}
 }
 if(flag1==1 && flag2==1){
delay                    (1000);
myservo1.write(100);
flag1=0, flag2=0;
```

```
}
 lcd.setCursor            (0,0);
lcd.print("  WELCOME!  ");
lcd.setCursor            (0,1);
lcd.print("Slot    Left:    ");
lcd.print(Slot);
 }
```


ESP32