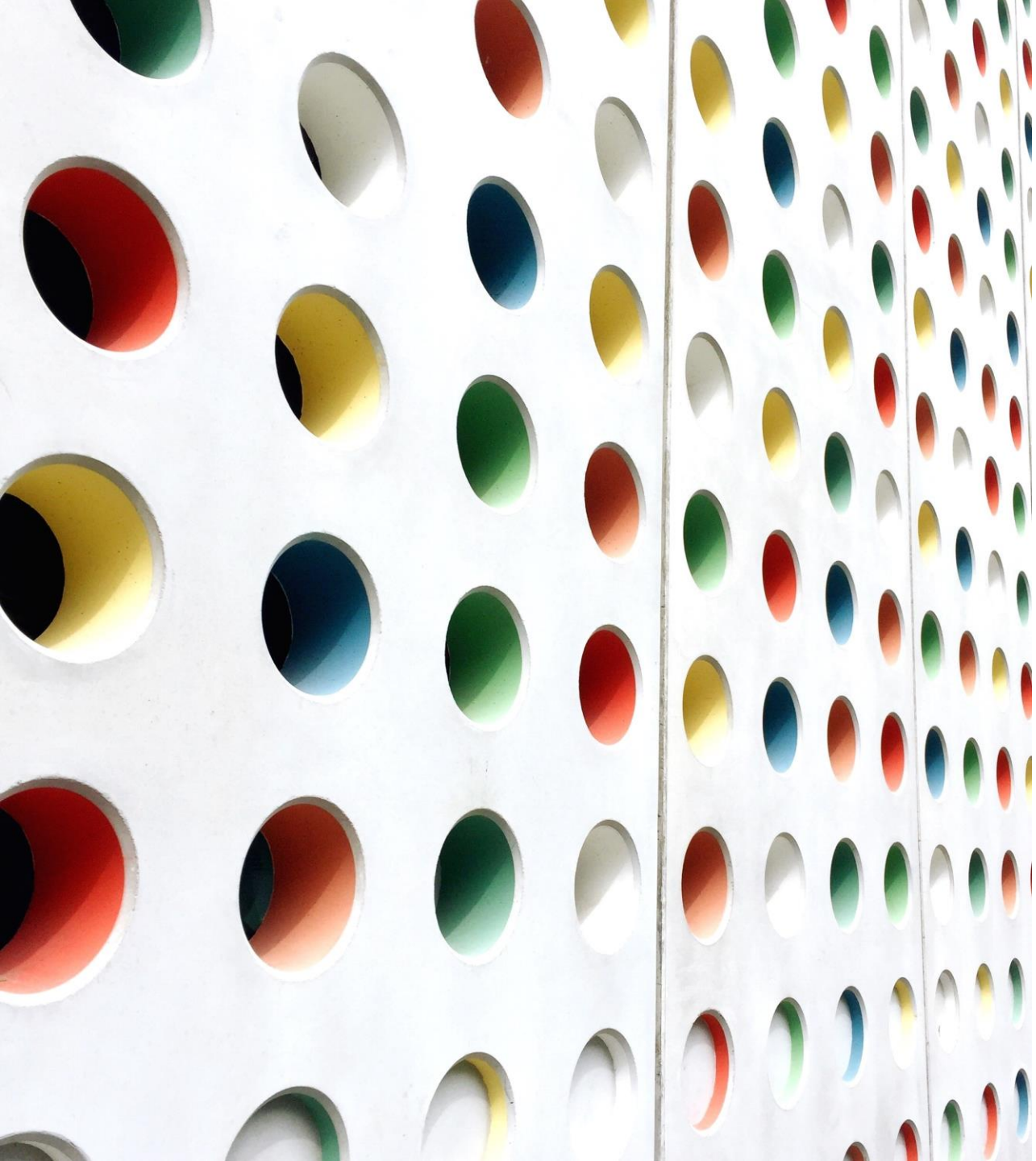


SMART PARKING

USING

INTERNET OF THINGS

BASIC DETAILS OF TEAM :

Team Name : proj_224089_Team1

Team Based : Smart Parking

Team Leader Name : Manivannan J

Institution Name : Chendu College of Engineering and Technology

Theme : Smart Parking using Integrated Sensing and Control

SMART PARKING SYSTEM :

- In this project we are discuss about smart parking.Nowadays, we use automobiles to save time, but often struggle to find parking spaces.Some parking places are manually operated,causing issues such as slow gate opening and high fees.

- Smart parking system can be implemented in places like hospitals ,shopping malls,theaters,tourist places and more.

WORKING DISCRIPTION :

- As a car approaches the entrance gate,a infrared motion sensor detects its presence and transmits this information to an Arduino board

- The Arduino board is pre-programmed with the knowledge of the available parking sites.If a parking slot is vacant,it acticates the gate to open.if no slots are available,it displays a message to the car owner indicating that all parking slots are occupied.

- once the car is parked in a slot,an occupancy sensor detects the presence of the car and uses a counter to keep track of the parking duration.

- In the event of a fire,a flame-detecting sensor on the premises detects the fire and sends a signal to the Arduino board.

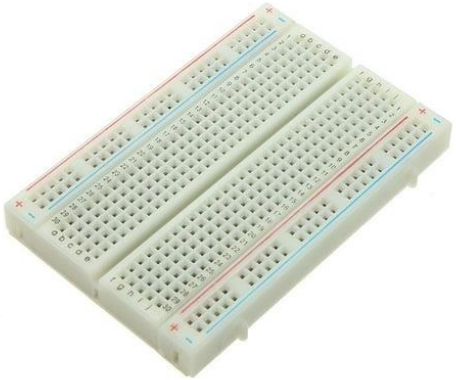
- The Arduino board,upon receiving the fire signal,activates an alarm system to alert relevant personal pr occupants about the fire emergency.

BENEFITS OF SMART PARKING :

- Improved Efficiency: Real-time monitoring helps drivers find available parking spaces quickly.
- Reduced Traffic Congestion: Efficient parking reduces circling, easing traffic congestion.
- Cost Savings: Drivers save on fuel and parking fees, while parking lot owners optimize revenue.
- Environmentally Friendly: Fewer cars circling mean reduced emissions and improved air quality.
- Reduced Vehicle Theft :Real time monitoring deters theft and vandalism.
- Remote Management :operators can manage facilities remotely,reducing maintenance costs.
- Scalability:easily expand the system to accommodate growing parking demands.

COMPONENTS REQUIRED :

- ✓ HC-SR04 Ultrasonic Distance Sensor
- ✓ Raspberry Pi Pico
- ✓ ESP32
- ✓ Mini breadboard
- ✓ Jumper cables



PIN Connection:

For the first sensor:

- VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico (usually labeled as VBUS or VSYS).
- GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico (labeled as GND).
- Trigger pin: Connect to Pin 0 (GP0) on the Raspberry Pi Pico.
- Echo pin: Connect to Pin 1 (GP1) on the Raspberry Pi Pico.

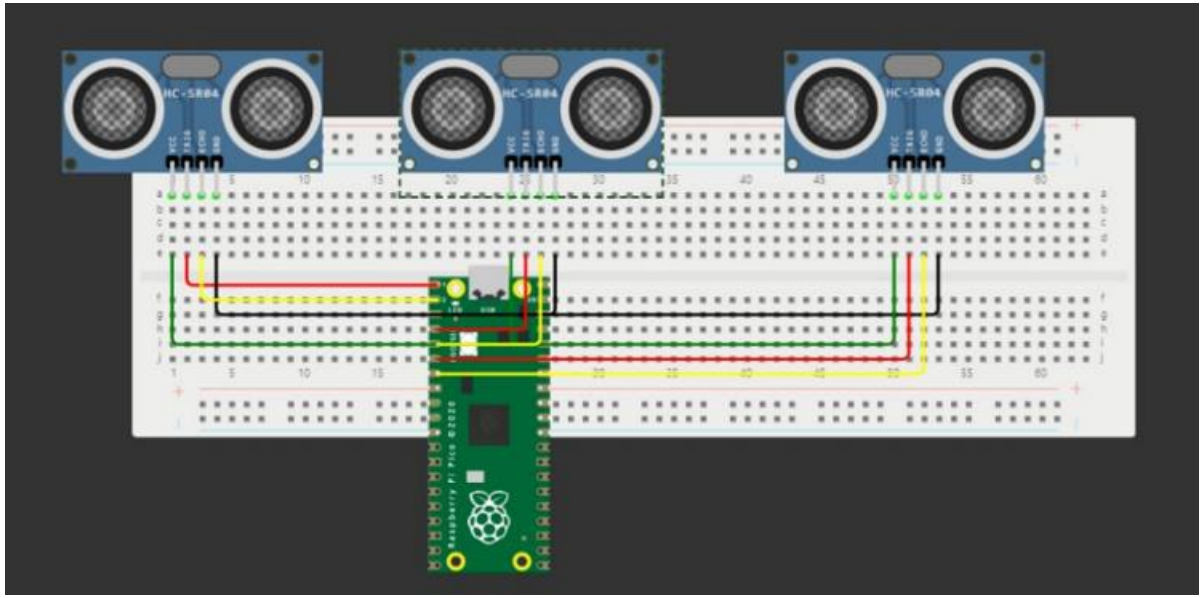
For the second sensor:

- VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico.
- GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico.
- Trigger pin: Connect to Pin 2 (GP2) on the Raspberry Pi Pico.
- Echo pin: Connect to Pin 3 (GP3) on the Raspberry Pi Pico.

For the third sensor:

- VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico.
- GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico.
- Trigger pin: Connect to Pin 4 (GP4) on the Raspberry Pi Pico.
- Echo pin: Connect to Pin 5 (GP5) on the Raspberry Pi Pico. Diagram

DIAGRAM :



MICRO PYTHON CODE :

```
import network
import urequests as requests
from machine import Pin
import time

# Wi-Fi credentials
WIFI_SSID = "Your_WiFi_SSID"
WIFI_PASSWORD = "Your_WiFi_Password"

# Your Firebase Realtime Database URL
FIREBASE_URL = "https://your-firebase-database-url.firebaseio.com"

# Initialize Wi-Fi connection
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)

# Pins connected to the ultrasonic sensors (replace with your specific sensor setup)
sensor_pins = [Pin(2, Pin.OUT), Pin(4, Pin.OUT), Pin(5, Pin.OUT)]  # Use the appropriate GPIO pins
```

```
# Function to measure distance from the ultrasonic sensor
def measure_distance(trigger_pin, echo_pin):
    trigger_pin.on()
    time.sleep_us(10)
    trigger_pin.off()

while echo_pin.value() == 0:
        pulse_start = time.ticks_us()

    while echo_pin.value() == 1:
        pulse_end = time.ticks_us()

    pulse_duration = time.ticks_diff(pulse_end, pulse_start)
    distance = (pulse_duration / 58)     # Convert to
centimeters

    return distance

# Function to send data to Firebase
def send_data_to_firebase(occupancy_data):
    data = {
        "occupancy_data": occupancy_data
    }
```

```
 print("Failed to send occupancy data to Firebase.")
response = requests.post(FIREBASE_URL +
"/restroom_data.json", json=data)
    if response.status_code == 200:
        print("Occupancy data sent to Firebase successfully.")
    else:
# Main loop to read occupancy data and send to Firebase
while True:
    try:
        occupancy_data = []
        for sensor_pin in sensor_pins:
            distance = measure_distance(sensor_pin, Pin(12,
Pin.IN))  # Echo pin connected to GPIO 12
            print(f"Distance: {distance} cm")

            # Adjust the threshold for occupancy based on your
sensor setup
            if distance < 10:
                occupancy_data.append(1)
            else:
                occupancy_data.append(0)
send_data_to_firebase(occupancy_data)
    except Exception as e:
```

```
print("Error reading occupancy data:", e)

# Adjust the sleep time as needed (e.g., every few seconds)
    time.sleep(5)
```

JAVA SCRIPT CODE :

```
import { initializeApp } from "firebase/app";
imporimport { initializeApp } from "firebase/app";
import { getDatabase, ref, onValue } from "firebase/database";

// Your Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyD-kAKiBxqSdcRA8h4CnSy0BooBFSKMgYg",
  authDomain: "chromatic-being-313507.firebaseapp.com",
  projectId: "chromatic-being-313507",
  storageBucket: "chromatic-being-313507.appspot.com",
  messagingSenderId: "236991601138",
  appId: "1:236991601138:web:34bfd1b00d06eef4e3336b",
  measurementId: "G-SWNGZT49YC"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

```
// Get a reference to your Firebase Realtime Database
const db = getDatabase(app);
const dataRef = ref(db, '/occupancy_data'); // Update with your actual path
t { getDatabase, ref, onValue } from "firebase/database";

// Your Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyD-kAKiBxqSdcRA8h4CnSy0BooBFSKMgYg",
  authDomain: "chromatic-being-313507.firebaseapp.com",
  projectId: "chromatic-being-313507",
  storageBucket: "chromatic-being-313507.appspot.com",
  messagingSenderId: "236991601138",
  appId: "1:236991601138:web:34bfd1b00d06eef4e3336b",
  measurementId: "G-SWNGZT49YC"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Get a reference to your Firebase Realtime Database
const db = getDatabase(app);
const dataRef = ref(db, '/occupancy_data'); // Update with your actual path
```

```
// Listen for changes in the data
onValue(dataRef, (snapshot) => {
  const data = snapshot.val(); // Get the data from the snapshot
  console.log("Received data from Firebase:", data);

  // Add your code to display or process the data as needed
  // For example, update a web page element with the received data
});
```

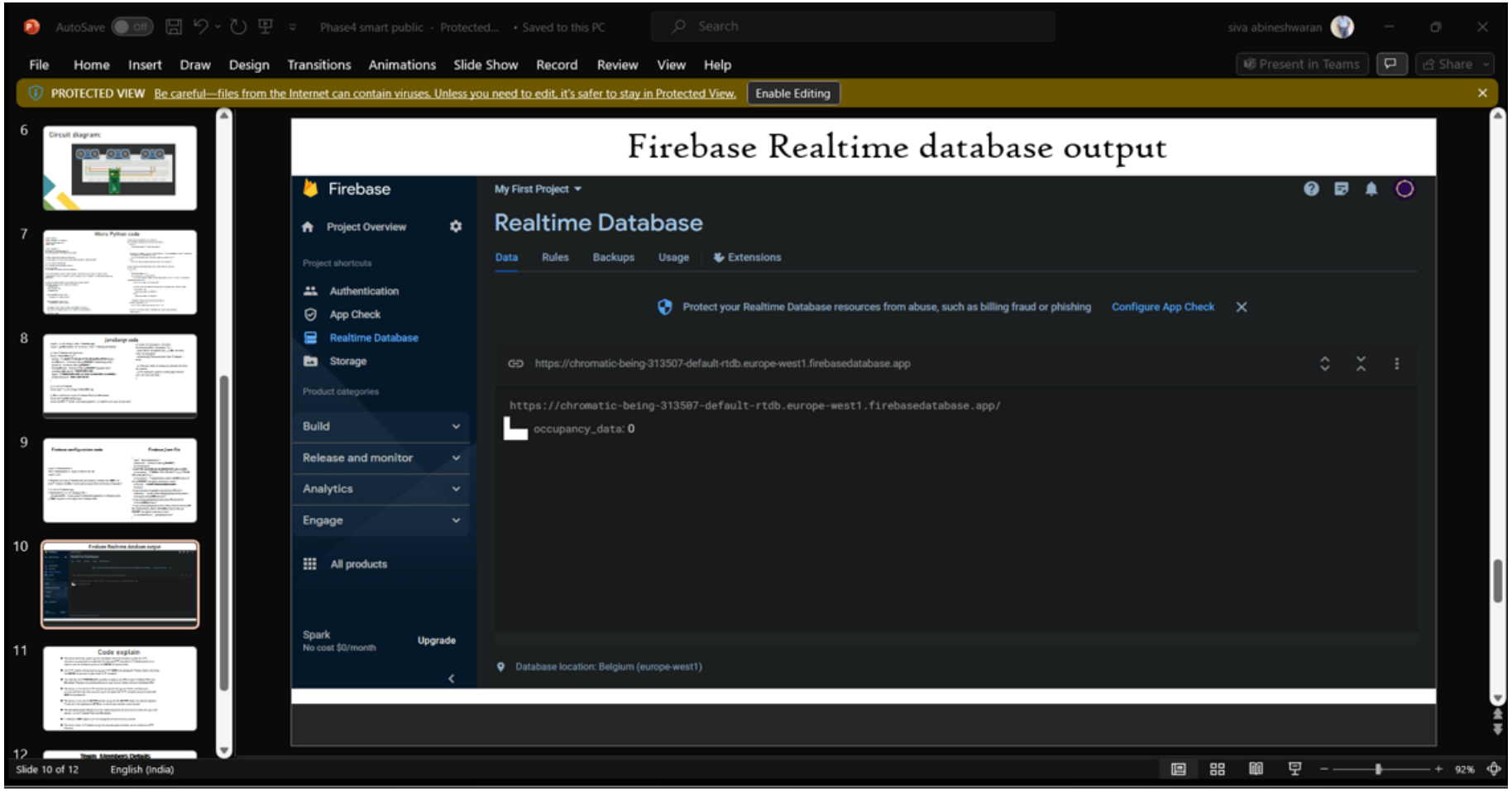
FIREBASE CONFIGURATION CODE

```
import firebase_admin
from firebase_admin import credentials, db
import time

# Replace with your Firebase service account credentials JSON
file
cred = credentials.Certificate("path/to/your-service-account-
key.json")

# Initialize Firebase app
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://your-firebase-database-
url.firebaseio.com'
}) POST request to the specified Firebase URL.
```

FIREBASE JSON FILE :


```
 "type": "service_account",
  "project_id": "chromatic-being-313507",
  "private_key_id":
"de45799fc4b2b89b3ef8e68562102ac0aac1531",
  "private_key": "-----BEGIN PRIVATE KEY-----\n\n-----END
PRIVATE KEY-----\n",
  "client_email": "firebase-adminsdk-nie2x@chromatic-being-
313507.iam.gserviceaccount.com",
  "client_id": "113997090536586644369",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase
-adminsdk-nie2x%40chromatic-being-
313507.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

FIREBASE REALTIME DATABASE OUTPUT



CODE EXPLAIN :

- The script starts by importing the necessary libraries network is used for Wi-Fi connectivity.urequests is imported for making HTTP requests to Firebase.machine is used to control hardware pins on the ESP32 microcontroller.

- the Wi-Fi credentials by specifying your Wi-Fi SSID and password. These credentials allow the ESP32 to connect to your local Wi-Fi network.

- You also set the FIREBASE_URL variable to specify the URL of your Firebase Realtime Database. Replace the placeholders with your actual credentials and database URL.

- The script initializes the Wi-Fi connection by activating the WLAN interface with wifi.active(True) and then connecting to the specified Wi-Fi network using the provided SSID and password.

- The script initializes the DHT22 sensor using the dht.DHT22 class. You should replace Pin(4) with the appropriate GPIO pin to which your sensor is connected.

- The send_data_to_firebase function takes temperature and humidity data as input and sends it to the Firebase Realtime Database.

- It creates a JSON object with the temperature and humidity values.

TEAM MEMBERS DETAILS

ROLE IN TEAM	NAME	BRANCH NAME	YEAR
TEAM LEADER	MANIVANNAN.J	CSE	III
TEAM MEMBER 1	TAMIL SELVAN.S.S	CSE	III
TEAM MEMBER 2	THARUN.J.S	CSE	III
TEAM MEMBER 3	SIVA ABINESHWARAN.K	CSE	III