



Smart Public Restroom

Using internet of thing to detect occupancy
status and send to a end point

Basic Details of the Team

Team Name : Proj_224080_Team_1

Team Leader Name: Akash.M

Team Based : Smart Public Restroom

Institute Name : Chendu college of engineering and technology

Theme : Smart Public Restroom with Integrated Sensing and Control

Smart Public Restroom Occupancy Monitoring with ESP32 and Firebase

Introduction

A smart public restroom system using Raspberry Pi Pico. Leveraging ultrasonic sensors, the code enables real-time occupancy monitoring and data transmission to a central cloud server. This prototype can be extended to facilitate efficient management of public restrooms, ensuring improved maintenance, and enhanced user experience.

Working Principle:

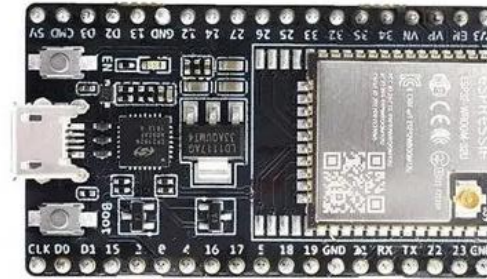
- ❑ By employing multiple ultrasonic sensors, strategically positioned within the public restroom area, the script continuously measures the proximity of individuals to each sensor.
- ❑ Upon detecting an individual within a certain range, the sensor registers the occupancy status.
- ❑ The Raspberry Pi Pico, equipped with Wi-Fi capabilities, facilitates the seamless transfer of this occupancy data to a cloud server.
- ❑ This data can be leveraged for real-time analysis, enabling authorities to monitor restroom traffic, optimize cleaning schedules, and ensure timely maintenance, thereby enhancing the overall user experience and satisfaction.

Components Required

01. Raspberry pi pico



02. Esp32



03. Ultra Sonic



04. Jumper cable



05. breadboard



Pin Connection

The First Sensor:

1. VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico.
2. GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico.

The Second Sensor:

1. VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico.
2. GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico.

The Third Sensor:

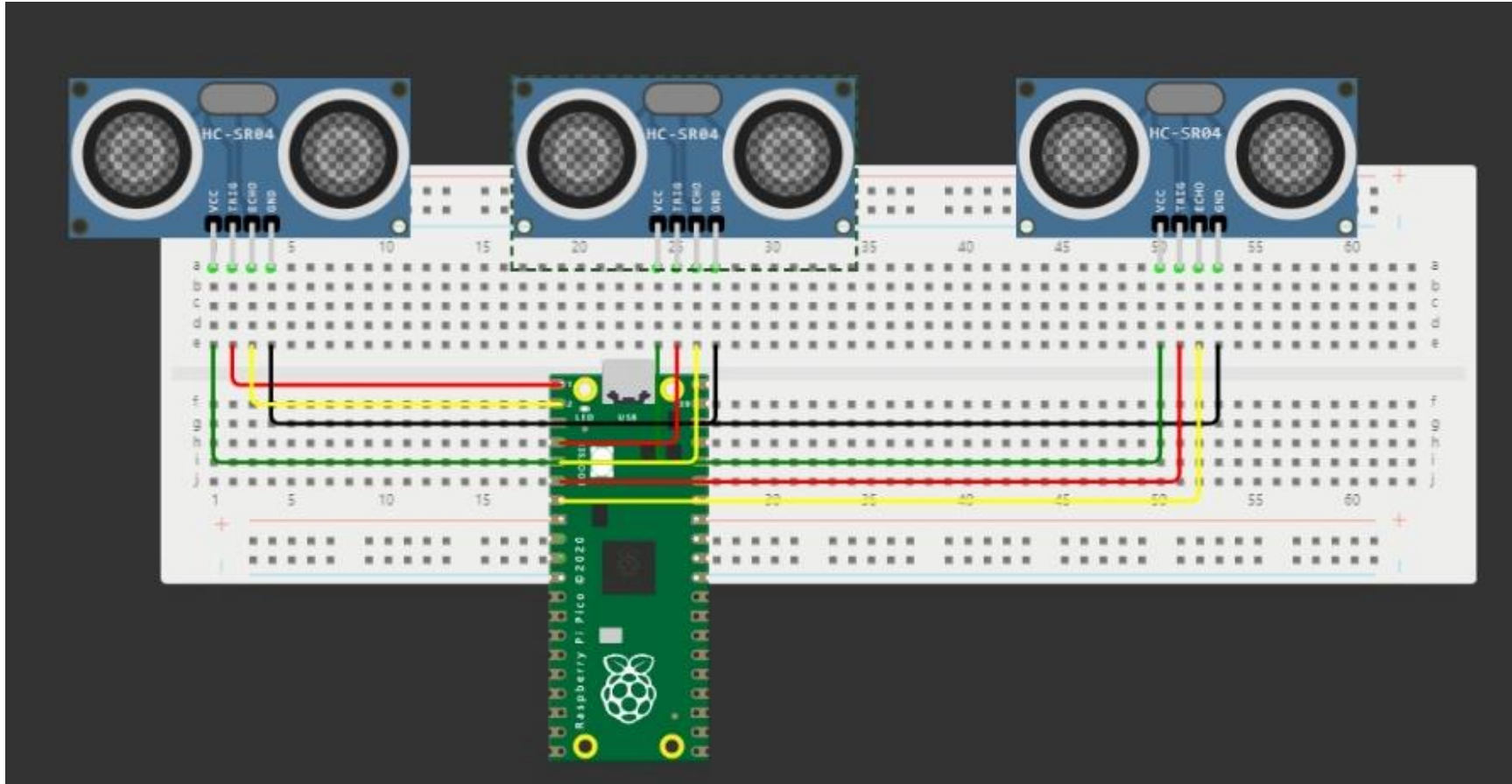
1. VCC Pin: Connect to the 5V pin on the Raspberry Pi Pico.
2. GND Pin: Connect to any of the ground pins on the Raspberry Pi Pico.

Trigger pin: Pin 0
Echo pin: Pin 1

Trigger pin: Pin 2
Echo pin: Pin 3

Trigger pin: Pin 4
Echo pin: Pin 5

Circuit 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
    }

    response = requests.post(FIREBASE_URL + "/restroom_data.json", json=data)
    if response.status_code == 200:
        print("Occupancy data sent to Firebase successfully.")
    else:
        print("Failed to send occupancy data to Firebase.")

# 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)
```

JavaScript code

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

// Your Firebase configuration
const firebaseConfig = {
  apiKey: "AlzaSyD-kAKiBxqSdcRA8h4CnSy0BooBFSKMgYg",
  authDomain: "chromatic-being-313507.firebaseio.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
});
```


Firestore configuration code

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

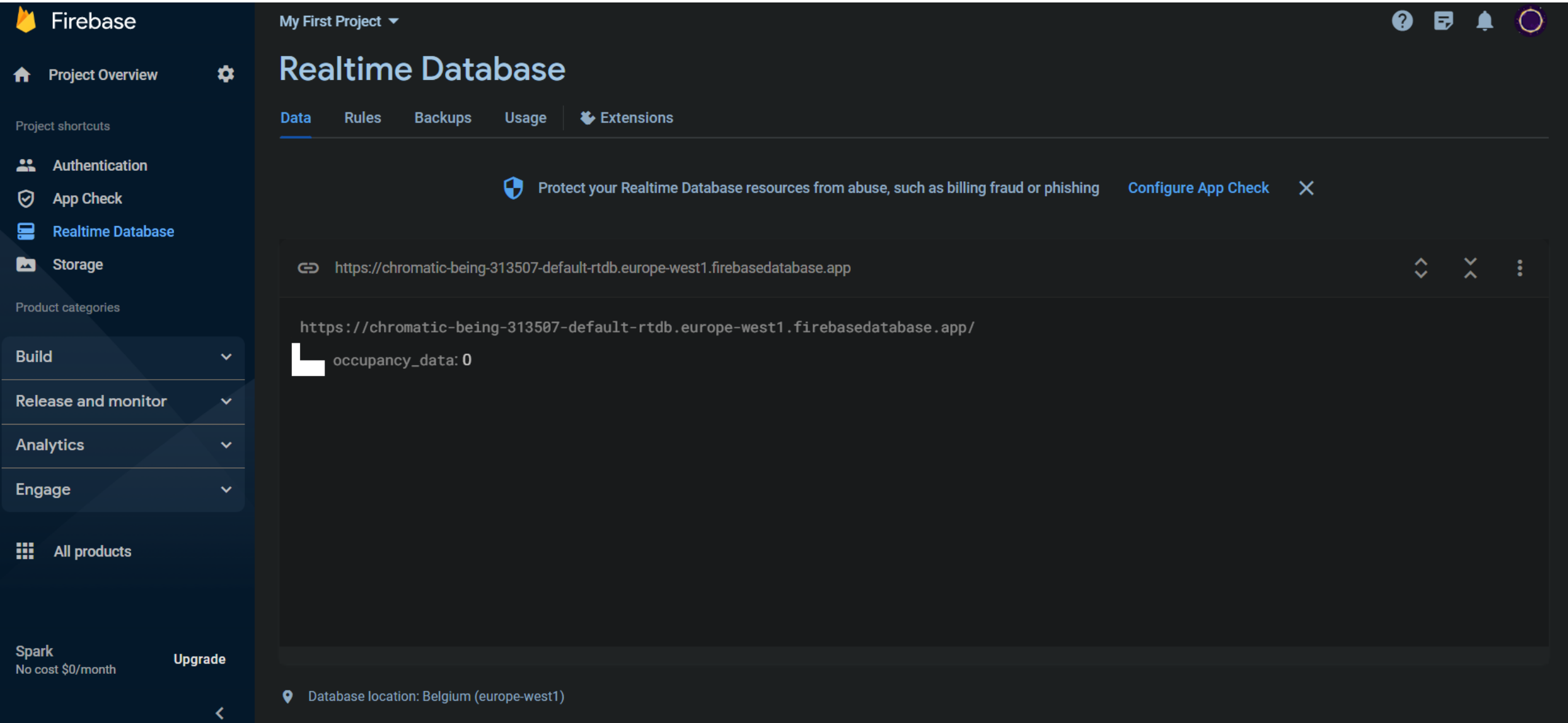
# Replace with your Firestore service account credentials JSON file
cred = credentials.Certificate("path/to/your-service-account-key.json")

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

Firestore 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": "11399709053586644369",
  "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/x50
9/firebase-adminsdk-nie2x%40chromatic-being-
313507.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

Firebase Realtime database output



The screenshot displays the Firebase Realtime Database interface. On the left is a dark sidebar with the 'Firebase' logo and navigation links: 'Project Overview', 'Authentication', 'App Check', 'Realtime Database' (highlighted), and 'Storage'. Below these are 'Product categories' (Build, Release and monitor, Analytics, Engage) and 'All products'. At the bottom left, it shows 'Spark No cost \$0/month' and an 'Upgrade' button. The main area is titled 'My First Project' and 'Realtime Database'. It has tabs for 'Data', 'Rules', 'Backups', 'Usage', and 'Extensions'. A security warning banner is present. The 'Data' tab shows a JSON output: `https://chromatic-being-313507-default-rtdb.europe-west1.firebaseio.com/` with a value of `occupancy_data: 0`. The database location is noted as 'Belgium (europe-west1)'.

My First Project ▾

Realtime Database

[Data](#) [Rules](#) [Backups](#) [Usage](#) [Extensions](#)

Protect your Realtime Database resources from abuse, such as billing fraud or phishing [Configure App Check](#) ✕

<https://chromatic-being-313507-default-rtdb.europe-west1.firebaseio.com/>

```
https://chromatic-being-313507-default-rtdb.firebaseio.com/  
└─ occupancy_data: 0
```

Database location: Belgium (europe-west1)

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.
- ❑ The data is sent to Firebase using the requests.post method, which makes an HTTP Protocol

Team Members Details

Role In Team	Name	Branch BE	Year
Team Leader	Akash	CSE	3 rd
Team Member 1	Ahmed abrarul hag	CSE	3 rd
Team Member 2	Gokula Kannan	CSE	3 rd
Team Member 3	Dhamodhar	CSE	3 rd