TEAM ID	530
PROJECT NAME	Smart parking using IOT

OBJECTIVE:

In our project we aim to provide real-time data transmission of available parking slot in parking area to the drivers through mobile app (Bylnk). The data from the sensors will be collected and transmitted in the cloud platform with the help of WIFI.

COMPONENTS DETAILS:

RASPBERRY PI PICO :

We used micro python raspberry pi as a board for our project. Micro Python on a Raspberry Pi Pico can be used in a smart parking system to create an efficient and intelligent parking management solution.

Micro Python simplifies the development process by providing a high-level programming environment that can run on the Raspberry Pi Pico, making it suitable for IOT and embedded projects like smart parking systems.

ULTRASONIC SENSOR (HC-SR04) :

By installing ultrasonic sensors in parking space. These sensors emit high-frequency sound waves and measure the time it takes for the sound waves to bounce back after hitting an obstacle (i.e., a vehicle).

When a vehicle occupies a parking space, the ultrasonic sensor will detect it based on the change in the return time of the sound waves.

By this we came to know whether the parking slot is occupied with vehicle or no not.

MOBILE APP - BYLNK :

We used Blynk to create a mobile app to receive data from the sensor, which is a popular Internet of Things (IoT) platform that allows us to build mobile applications to control and monitor your Raspberry Pi or other IoT devices. Blynk provides a user-friendly way to create smartphone apps to interface with various hardware and sensors connected to your Raspberry Pi.

CODE: Using micro python on Raspberry Pi Pico

```
from ultra import DistanceSensor
from time import sleep

dsa = DistanceSensor(echo=2, trigger=3)
dsb = DistanceSensor(echo=4, trigger=5)
dsc = DistanceSensor(echo=13, trigger=14)
dsd = DistanceSensor(echo=17, trigger=16)

while True:
    distance_a = dsa.distance * 100
    distance_b = dsb.distance * 100
    distance_c = dsc.distance * 100
    distance_d = dsd.distance * 100
    a = float(distance_a)
    b = float(distance_b)
    c = float(distance_c)
    d = float(distance_d)  # Convert to a floating-point number

    print(a)
    print(b)
    print(c)
    print(d)
    A="A"
    B="B"
    C="C"
    D="D"
    no=0

    def parking(distance, n,slot):
        if distance < 30:
        # Code to execute if the distance is less than 30
            print("Space is not free:"+slot)
            if(n==0):
                n=0
            else:
```

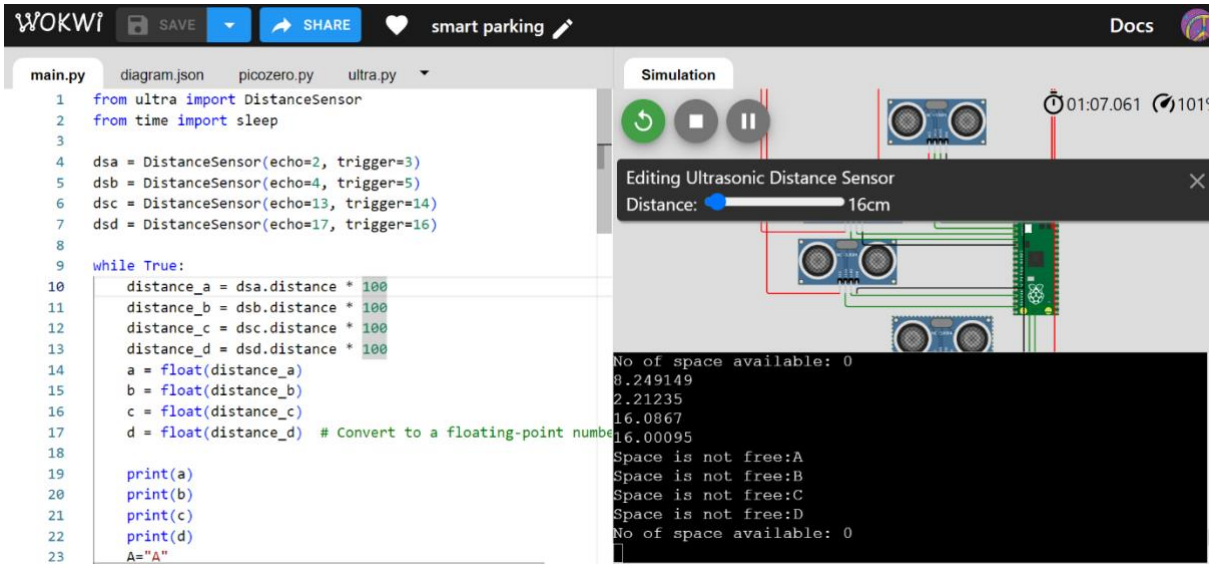
```
                n=n-1
        else:
        # Code to execute if the distance is not less than 30
            print("Space is free: "+slot)
            if(n==4):
                n=4
            else:
                n=n+1
        return n


    no=parking(a,no,A)
    no=parking(b,no,B)
    no=parking(c,no,C)
    no=parking(d,no,D)
    no=no
    print("No of space available:",no)



    sleep(0.1)
```

OUTPUT:



NOTE:

Due to unavailability of libraries for micro python on Raspberry Pi Pico we cannot able to transfer the data from sensor to cloud platform .

Such as urequest, WIFI , think speak libraries

So, we used ESP32 to work our project on real time data transfer.

ESP32:

The ESP32 is a versatile microcontroller and Wi-Fi module that can play several important roles in a smart parking system.

The ESP32's built-in Wi-Fi capabilities enable it to connect to the internet or a local network. It can communicate with a central server, cloud platform, or a mobile app to send and receive data related to parking space occupancy. This real-time communication is essential for keeping drivers informed about parking availability.

MOBILE APP INTEGRATION IN ESP32:

Mobile app is part of the smart parking system, the ESP32 can communicate with the app through APIs, allowing users to find available parking spaces.

CODE: Using ESP32

```
#define BLYNK_TEMPLATE_ID "TMPL3Fa6eQf9e"
#define BLYNK_TEMPLATE_NAME "Smart parking"
#define BLYNK_AUTH_TOKEN "C5yd-C-cAOwEMMiqohaSeoJRModhyTzl"
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


char ssid[] = "Wokwi-GUEST";
char pass[] = "";

BlynkTimer timer;

// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0)
```

```
{
  // Set incoming value from pin V0 to a variable
  int value = param.asInt();

  // Update state
  Blynk.virtualWrite(V1, value);
}

// This function is called every time the device is connected to the
Blynk.Cloud
BLYNK_CONNECTED()
{
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.pn
g");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-
started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  // by using this we can send any value at any time.
  // we should not sent more than 10 values per second.
  Blynk.virtualWrite(V2, millis() / 1000);
}


#define ECHO_A 2
#define TRIG_A 18
#define ECHO_B 4
#define TRIG_B 5
#define ECHO_C 13
#define TRIG_C 14
#define ECHO_D 17
#define TRIG_D 16
int a,b,c,d;
int Slot;
//char full[] = "Full";
//char fr[] = "Free";
void setup() {
  // entering setup code here, to run once:
  Serial.begin(115200);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
```

```
  pinMode(TRIG_A, OUTPUT);
  pinMode(ECHO_A, INPUT);
  pinMode(TRIG_B, OUTPUT);
  pinMode(ECHO_B, INPUT);
  pinMode(TRIG_C, OUTPUT);
  pinMode(ECHO_C, INPUT);
  pinMode(TRIG_D, OUTPUT);
  pinMode(ECHO_D, INPUT);
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
}

void loop() {
  //writing our main code here, to run repeatedly:
  a=check(TRIG_A,ECHO_A);
  if(a==0){
     Blynk.virtualWrite(V1, 0 );
  }
  else{
     Blynk.virtualWrite(V1, 1);
  }
  b=check(TRIG_B,ECHO_B);
  if(b==0){
     Blynk.virtualWrite(V2, 0 );
  }
  else{
     Blynk.virtualWrite(V2, 1);
  }
  c=check(TRIG_C,ECHO_C);
  if(c==0){
     Blynk.virtualWrite(V3, 0 );
  }
  else{
     Blynk.virtualWrite(V3, 1);
  }
  d=check(TRIG_D,ECHO_D);
  if(d==0){
     Blynk.virtualWrite(V4, 0 );
  }
  else{
     Blynk.virtualWrite(V4, 1);
  }
  Serial.printf("%d %d %d %d \n",a,b,c,d);
  Slot=a+b+c+d;
  Serial.printf("slots free=%d \n",Slot);
  Blynk.virtualWrite(V0, Slot);
  delay(2000); // this speeds up the simulation
```

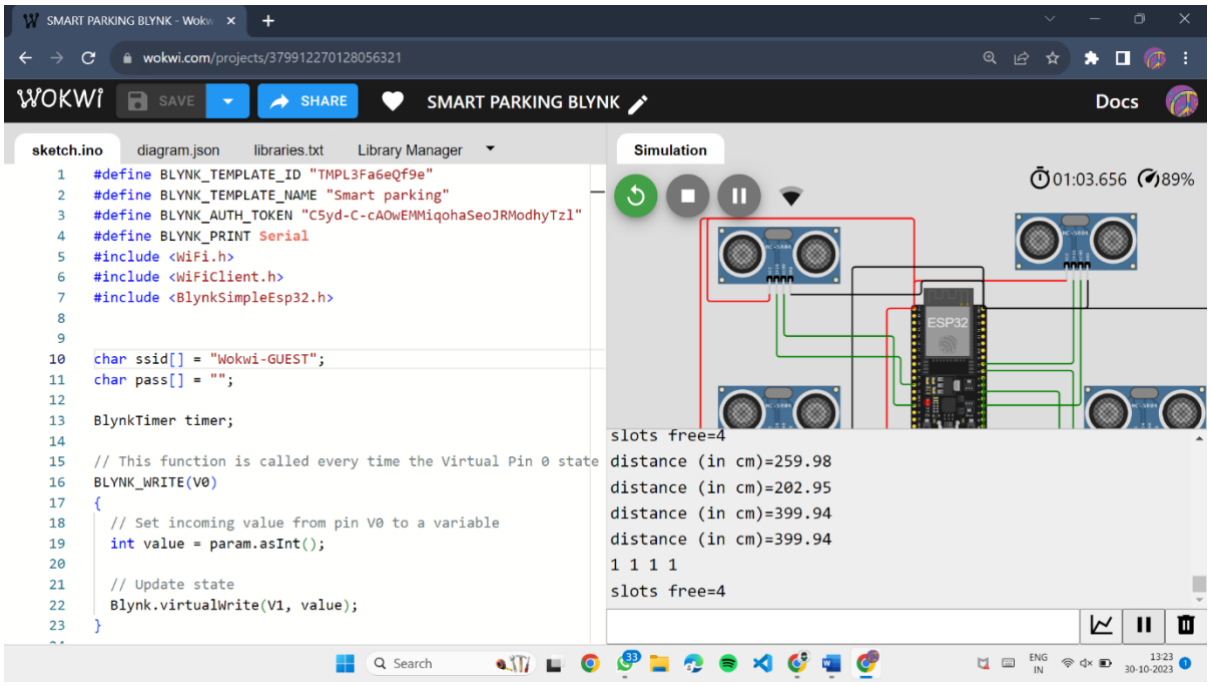
```
}
float check(int trigerpin, int echopin){
    float distance, duration;
    digitalWrite(trigerpin, LOW);
    delay(2);
    digitalWrite(trigerpin, HIGH);
    delay(10);
    digitalWrite(trigerpin, LOW);
    duration=pulseIn(echopin,HIGH);
    distance=duration/2*0.034;
    Serial.print("distance (in cm)=");
    Serial.println(distance);
    delay(10);
    if(distance>30){
      return 1;

    }
    else{
      return 0;
    }
 }
```

OUTPUT:

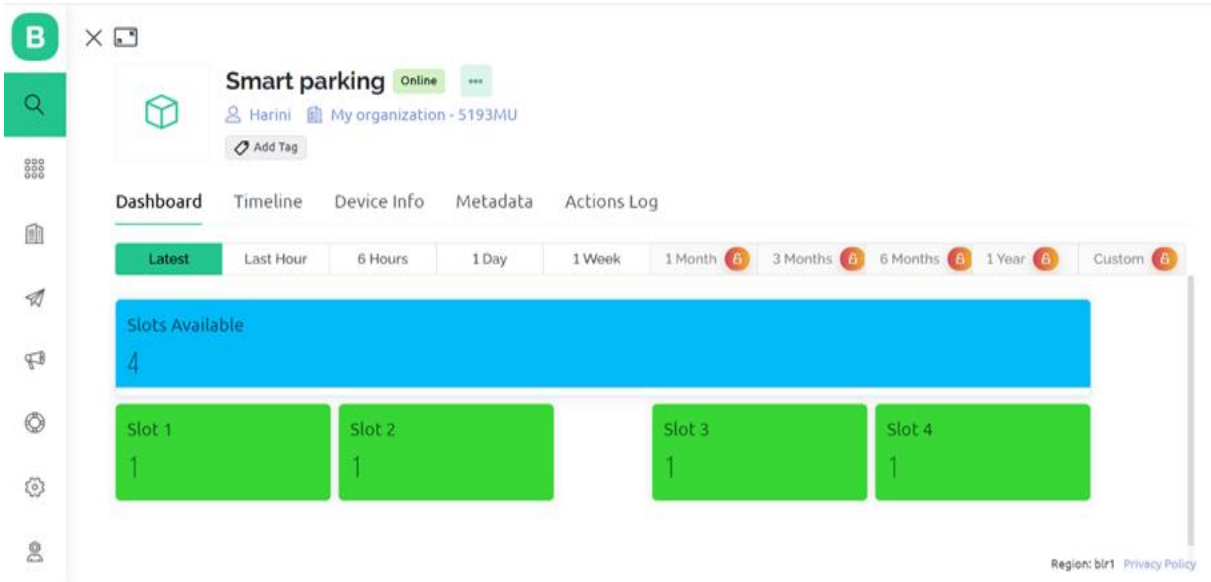
Stimulation:

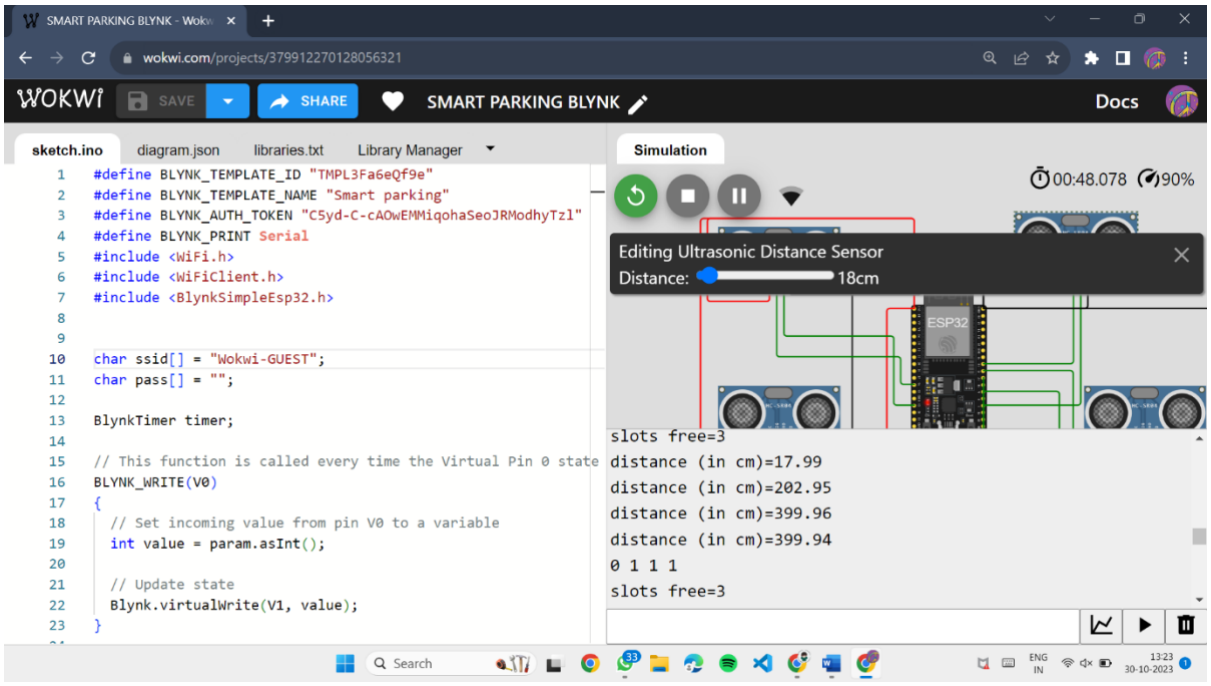
It shows the available parking slot in the parking area and also the number of slots free. By receiving data from the ultrasonic sensor. By running our code it connect to the WIFI and Blynk .



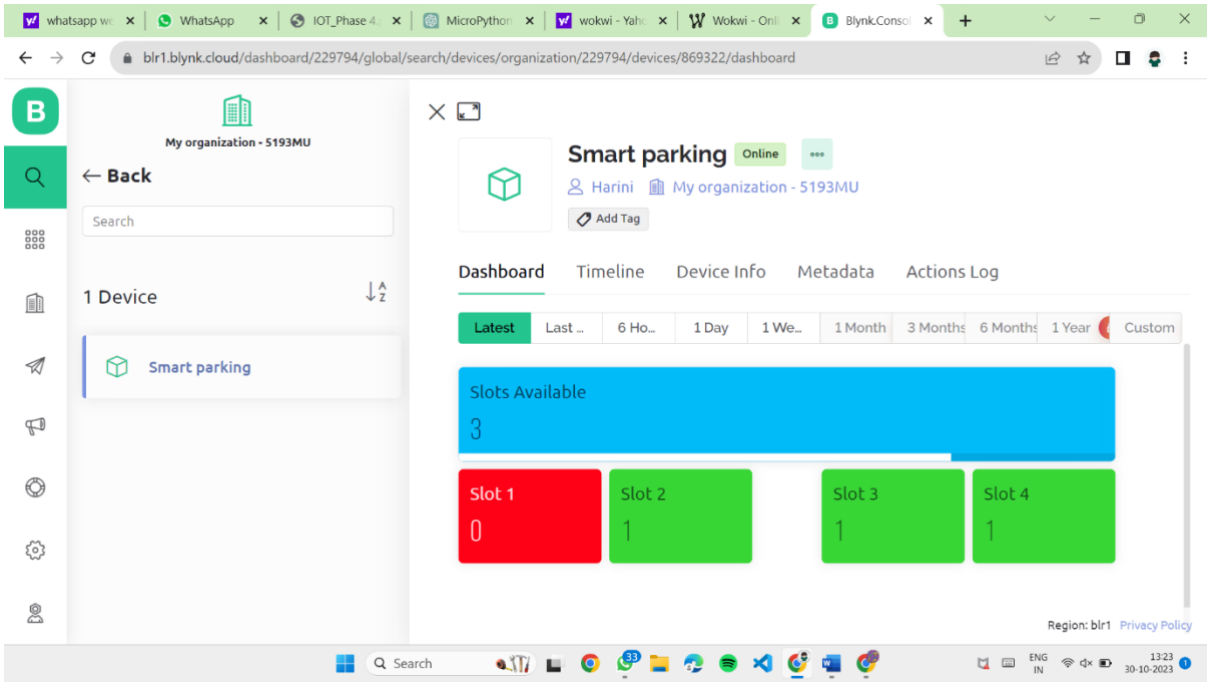
Blynk output:

Blynk receives the data given by the sensors and shows the available parking slot in the parking area which makes effective parking to the drivers.





Here, 3 slots where free, also it mentions the distance in the parking slot in centimetre. By connecting it into Blynk using WIFI shows real time parking space availability.



In Blynk number of slots available is shown. The free parking slots were indicated in green label and occupied slots were indicated in red label.

CONCLUSION:

This represents a Transformative solution to urban parking inefficiencies. By integrating Raspberry Pi, sensors, and cameras, parking space availability and enhances user experiences. Real-time parking availability information, harmony to reduce congestion and pollution, making urban living more sustainable. The project's comprehensive documentation ensures long-term effectiveness, and it paves the way for smarter, more efficient urban mobility.