I.Project Description:

Improving vehicle scheduling:

Optimizing vehicle schedules can help to reduce waiting times and improve service frequency.

Designing efficient routes:

Designing efficient routes can help to reduce travel times and fuel consumption.

Coordinating different modes of transportation:

Coordinating different modes of transportation, such as buses and trains, can make it easier for passengers to get around.

Improving infrastructure:

Investing in public transportation infrastructure, such as new bus lanes and train stations, can improve the overall quality of service.

Public transportation optimization is a complex problem, as it involves a variety of factors, such as passenger demand, vehicle availability, and budget constraints. However, a number of optimization techniques can be used to find solutions that improve the overall efficiency and effectiveness of public transportation systems.

One common approach to public transportation optimization is to use mathematical models. These models can be used to represent the complex relationships between the different factors involved in public transportation systems. Once a model has been developed, it can be used to evaluate different scenarios and identify the one that produces the best outcome.

II. Project Implementation:

Data Collection and Processing:

Data Sources:

Gather data on passenger demand, routes, stops, and schedules.

Utilize real-time data, historical usage patterns, and geographic information.

1.**Data Processing**:

Clean and preprocess the collected data.

Create a digital map of routes, stops, and passenger demand.

2.**Route Planning and Scheduling**:

Optimization Algorithms:

Implement algorithms to optimize routes and schedules.

Consider factors such as traffic conditions, road closures, and passenger preferences.

Frequency and Capacity:

Determine the frequency and capacity of each vehicle on different routes.

3. **Real-Time Monitoring:**

GPS and Tracking Systems:

Implement GPS tracking on vehicles.

Continuously monitor vehicle locations in real-time.

Adjustment Mechanism:

Develop a system to adjust routes and schedules based on real-time traffic and demand fluctuations.

4. Passenger Information System:

Real-Time Information:

Develop a platform to provide real-time information to passengers about routes, schedules, and delays.

Implement mobile apps and digital signage at stops for updates.

5. Ticketing and Payment:

Electronic Ticketing:

Implement electronic ticketing systems, including contactless payment options**.**

6. Safety and Security:

Surveillance and Emergency Systems:

Implement surveillance systems on vehicles and at stops.

Develop emergency response systems for unexpected situations.

7. Feedback and Improvement:

Passenger Feedback:

Create a mechanism for collecting feedback from passengers and drivers.

Utilize data analytics to identify areas for improvement.

8. Environmental Considerations:

Eco-Friendly Practices:

Implement practices and technologies to reduce the environmental impact of public transportation.

9. Cost Management:

Operational Cost Monitoring:

Develop a system to continuously monitor and manage operational costs.

10. Integration:

System Integration:

Ensure seamless integration with other transportation systems, city infrastructure, and data sources.

11. Development Platforms and Technologies:

Programming Languages:

Choose appropriate programming languages (e.g., Python, Java) for backend and frontend development.

Database:

Implement a robust database system to store and retrieve data efficiently.

Web and Mobile Development:

Develop web and mobile applications for passenger interfaces.

APIs and Microservices:

Implement APIs and microservices for modular and scalable development.

12. Testing and Deployment:

Testing:

Conduct thorough testing, including unit testing, integration testing, and user acceptance testing.

Deployment:

Deploy the system in stages, considering scalability and user adoption.

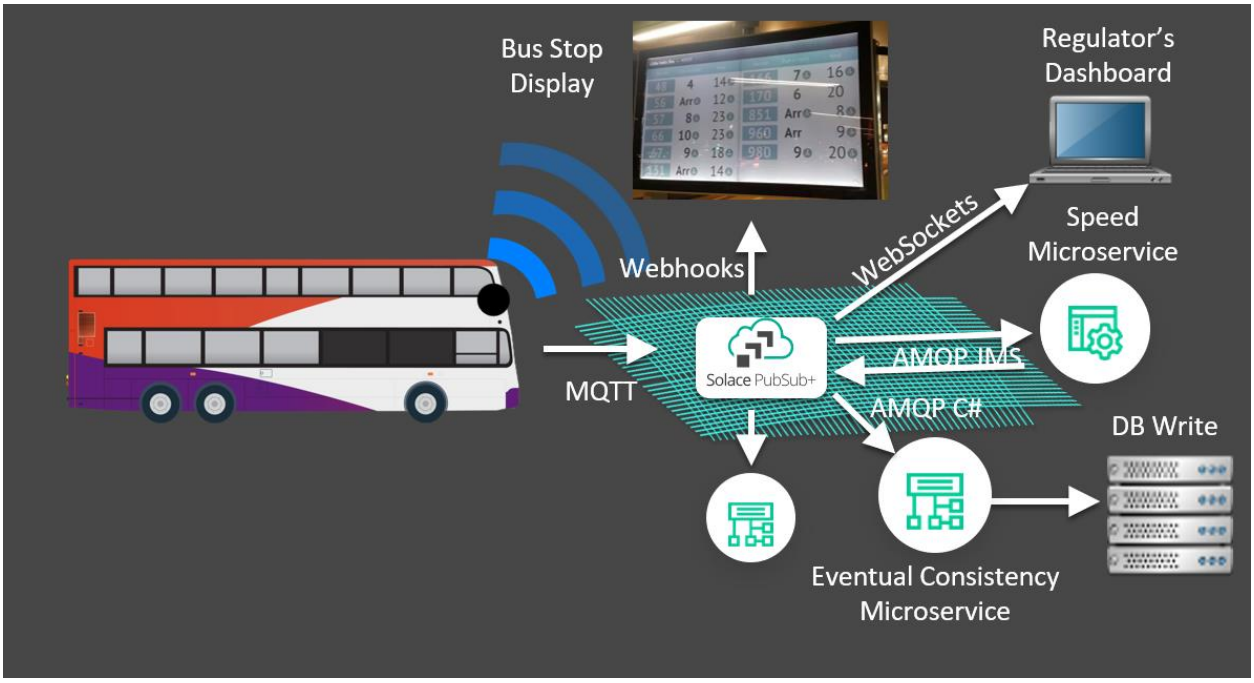
13. Maintenance and Updates:

Regular Maintenance:Establish a maintenance plan to address issues and ensure system reliability.

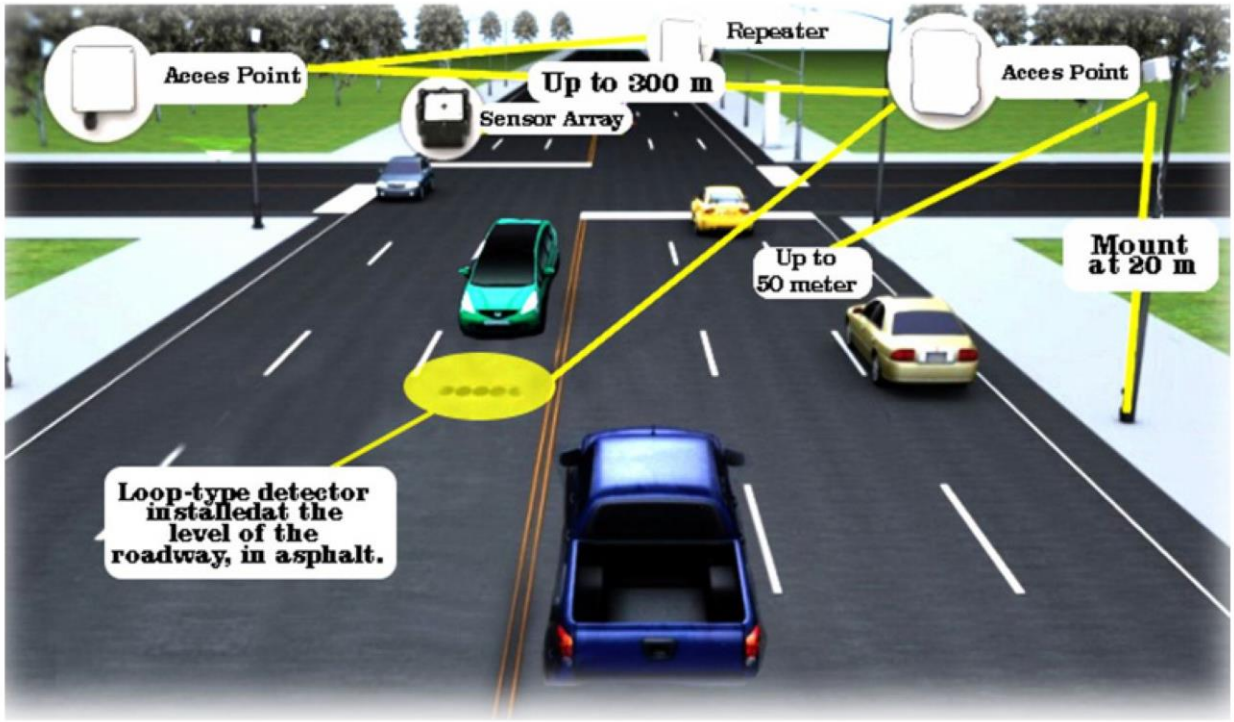
Updates and Enhancements:Plan for periodic updates and enhancements based on feedback and technological advancements.

III. Circuit Diagram:

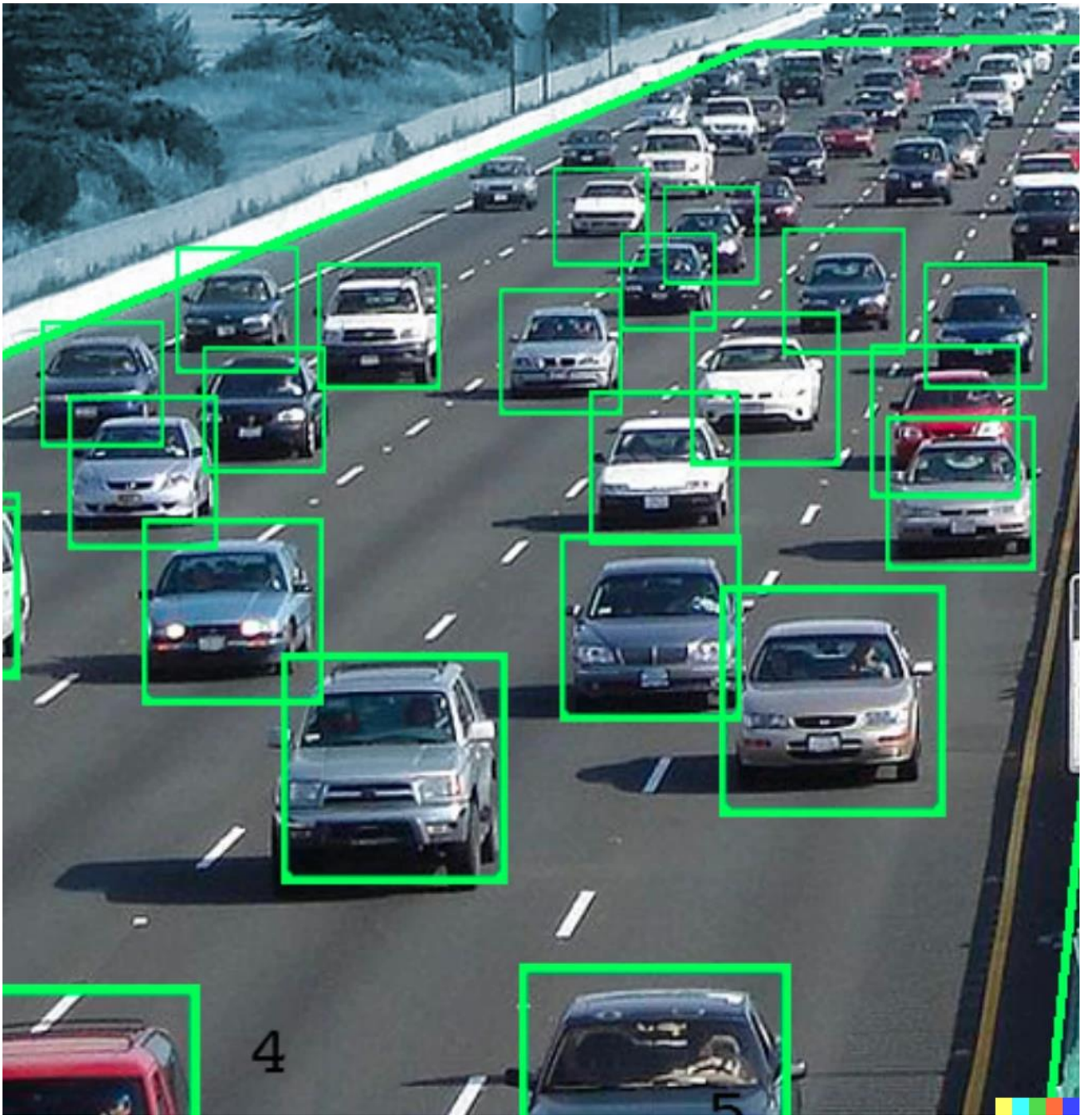
Working process of public transportation optimization:



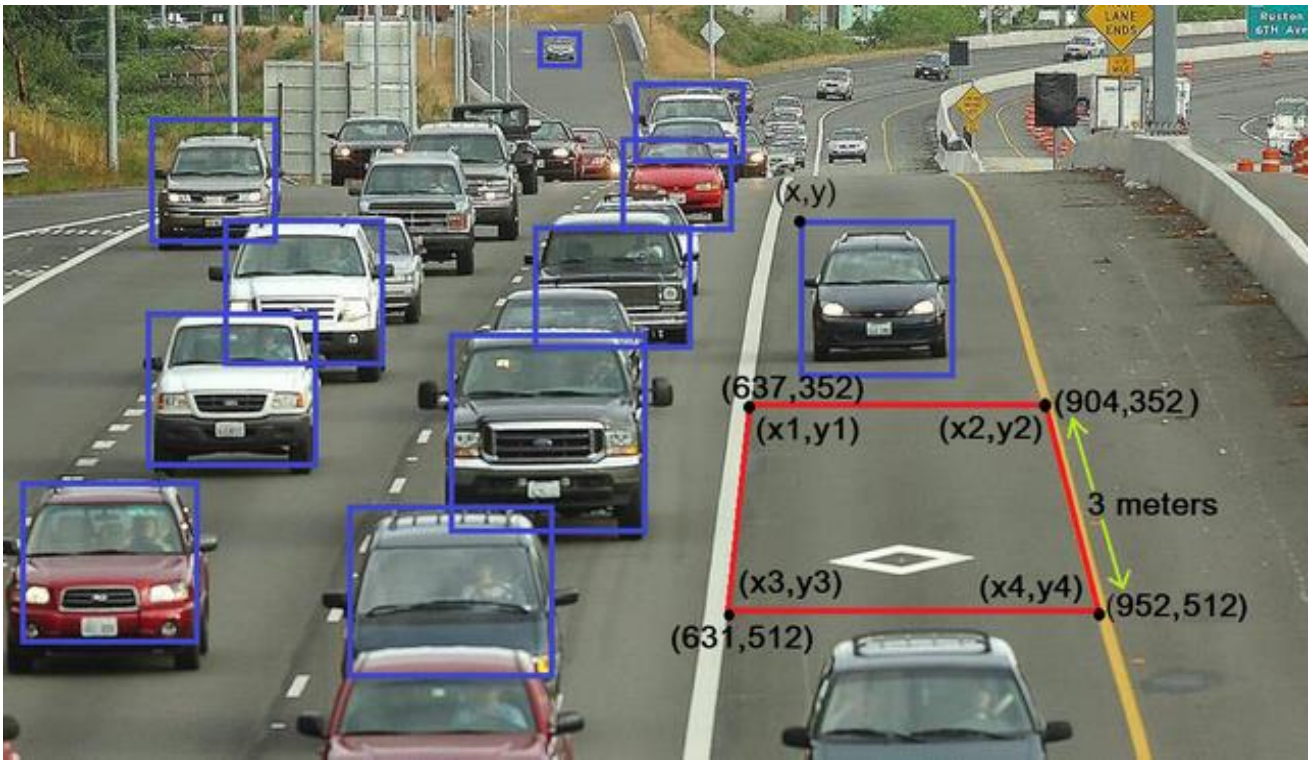
Real-time example of public transportation optimization:



Vehicle detection to control the public transport :



Public transportation optimization using IoT connected Camera's:



IV. Source Code:

4.1.. Front Code:(Python)

```
#define BLYNK_TEMPLATE_ID "TMPL26V4fGv5q"

#define BLYNK_TEMPLATE_NAME "Test"

#define BLYNK_AUTH_TOKEN "XEHxNF_Ur1Nt2p7wB5B20dNI1ZUwj34P"


#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>


int duration1 = 0;

int distance1 = 0;

int duration2 = 0;

int distance2 = 0;

int dis1 = 0;

int dis2 = 0;

int dis_new1 = 0;

int dis_new2 = 0;

int entered = 0;

int left = 0;

int inside = 0;
```

```
#define LED 2
#define PIN_TRIG1 15
#define PIN_ECHO1 14
#define PIN_TRIG2 13
#define PIN_ECHO2 12
BlynkTimer timer;

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";   // your network SSID (name)
char pass[] = "";
#define BLYNK_PRINT Serial

long get_distance1() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG1, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG1, LOW);

  // Read the result:
  duration1 = pulseIn(PIN_ECHO1, HIGH);
  distance1 = duration1 / 58;
  return distance1;
}
```

```
long get_distance2() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG2, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG2, LOW);

  // Read the result:
  duration2 = pulseIn(PIN_ECHO2, HIGH);
  distance2 = duration2 / 58;
  return distance2;
}

void myTimer() {
  Serial.println("100");
  dis_new1 = get_distance1();
  dis_new2 = get_distance2();
  if (dis1 != dis_new1 || dis2 != dis_new2){
    Serial.println("200");
    if (dis1 < dis2){
      Serial.println("Enter loop");
      entered = entered + 1;
      inside = inside + 1;
```

```
    digitalWrite(LED, HIGH);
    Blynk.virtualWrite(V0, entered);
    Blynk.virtualWrite(V2, inside);
    dis1 = dis_new1;
    delay(1000);
    digitalWrite(LED, LOW);
   }
  if (dis1 > dis2){
    Serial.println("Leave loop");
    left = left + 1;
    inside = inside - 1;
    Blynk.virtualWrite(V1, left);
    Blynk.virtualWrite(V2, inside);
    dis2 = dis_new2;
    delay(1000);
   }

 }


}

 void setup() {
```

```
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  pinMode(PIN_TRIG1, OUTPUT);
  pinMode(PIN_ECHO1, INPUT);
  pinMode(PIN_TRIG2, OUTPUT);
  pinMode(PIN_ECHO2, INPUT);
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 8080);
  timer.setInterval(1000L, myTimer);
}
void loop() {
  Blynk.run();
  timer.run();
}
```

4.2.. Backend Code:(Python)

```
{
  "version": 1,
  "author": "MANOJ S",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -12.8, "left": 180.8, "attrs": {} },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
```

```
    "top": -31.94,
    "left": -1.1,
    "attrs": { "distance": "240" }
  },
  {
    "type": "wokwi-hc-sr04",
    "id": "ultrasonic2",
    "top": -32.14,
    "left": -184.37,
    "attrs": { "distance": "104" }
  },
  {
    "type": "wokwi-led",
    "id": "led1",
    "top": 196.6,
    "left": 95.24,
    "attrs": { "color": "red" }
  }
],
"connections": [
  [ "esp:TX0", "$serialMonitor:RX", "", [] ],
  [ "esp:RX0", "$serialMonitor:TX", "", [] ],
  [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
```

```
    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],
    [ "ultrasonic1:TRIG", "esp:D13", "green", [ "v0" ] ],
    [ "ultrasonic1:ECHO", "esp:D12", "green", [ "v0" ] ],
    [ "ultrasonic2:GND", "esp:GND.2", "black", [ "v93.61", "h255.99", "v-19.13", "h1.42" ] ],
    [ "ultrasonic2:VCC", "esp:VIN", "red", [ "v111.32", "h288.11", "v-26.93" ] ],
    [ "ultrasonic2:ECHO", "esp:D14", "green", [ "v97.15", "h262.44", "v-53.14" ] ],
    [ "ultrasonic2:TRIG", "esp:D15", "green", [ "v102.11", "h397.16", "v-36.14" ] ],
    [ "led1:C", "esp:GND.1", "black", [ "v7.36", "h195.18", "v-106.29" ] ],
    [ "led1:A", "esp:D22", "green", [ "v0.98", "h178.81", "v-149.51" ] ]
  ],
  "dependencies": {}
}
```

4.3.. Sample Output:

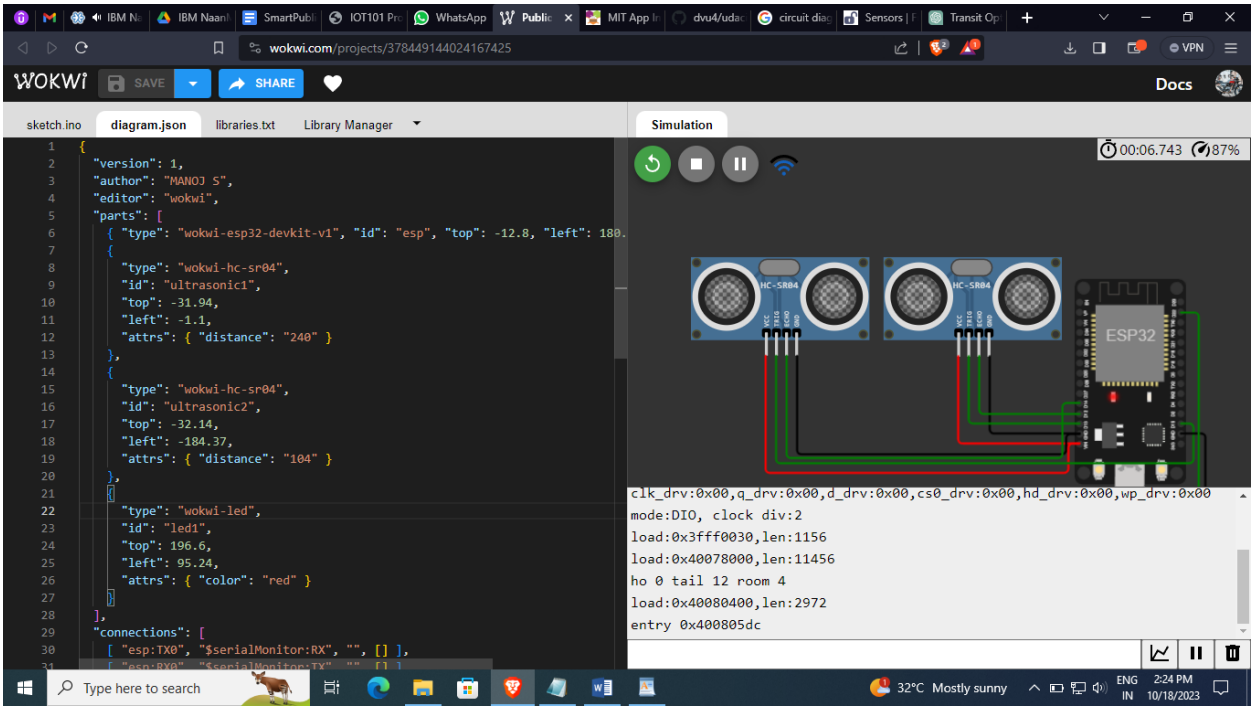


Fig: Start the IoT Module

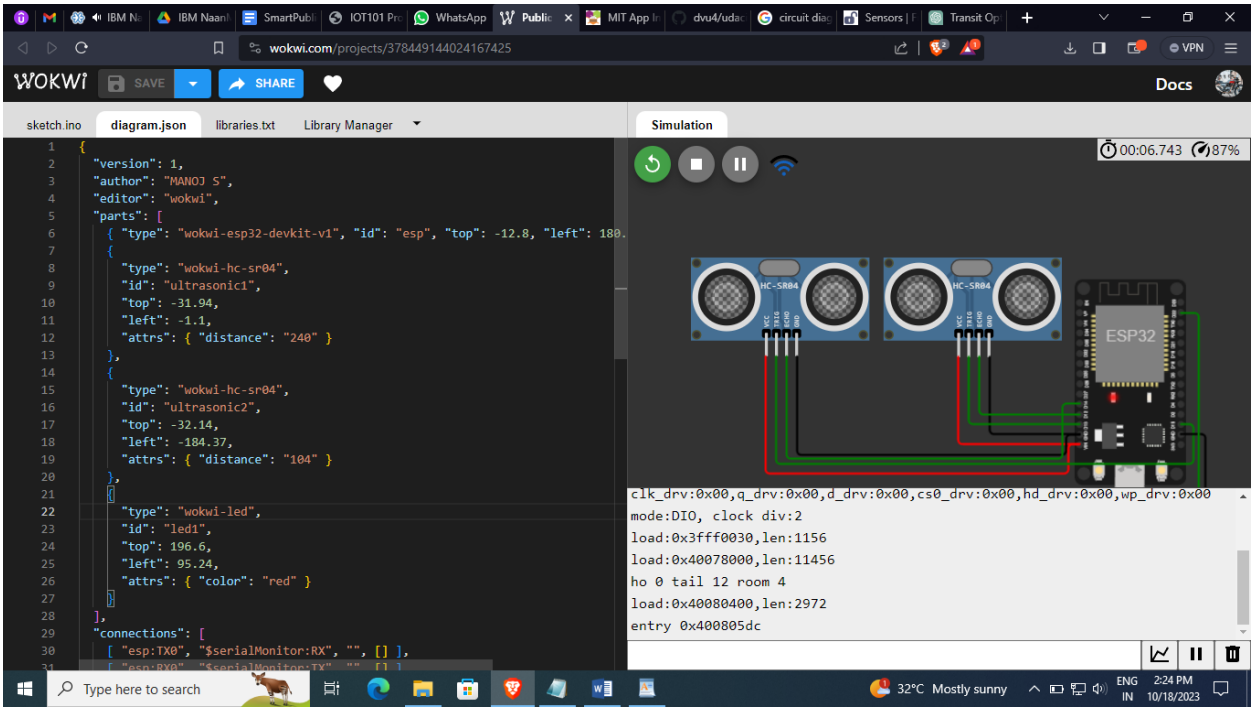


Fig: Initializing the IoT Module

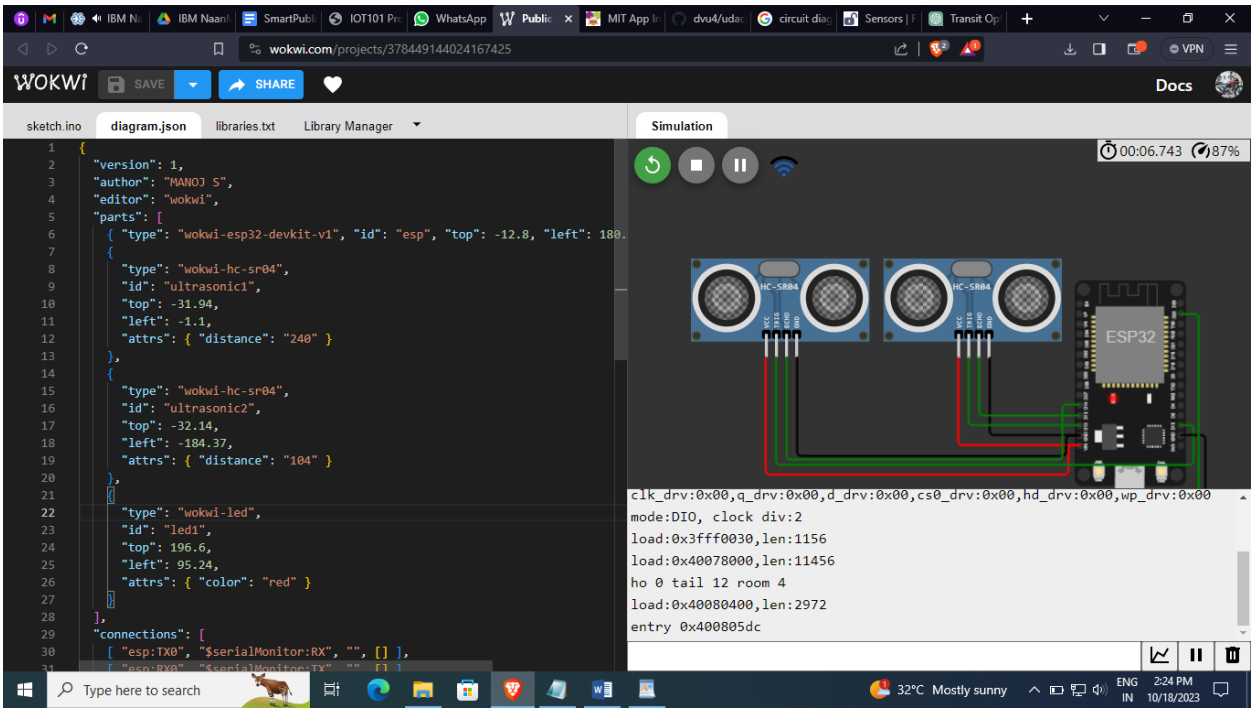
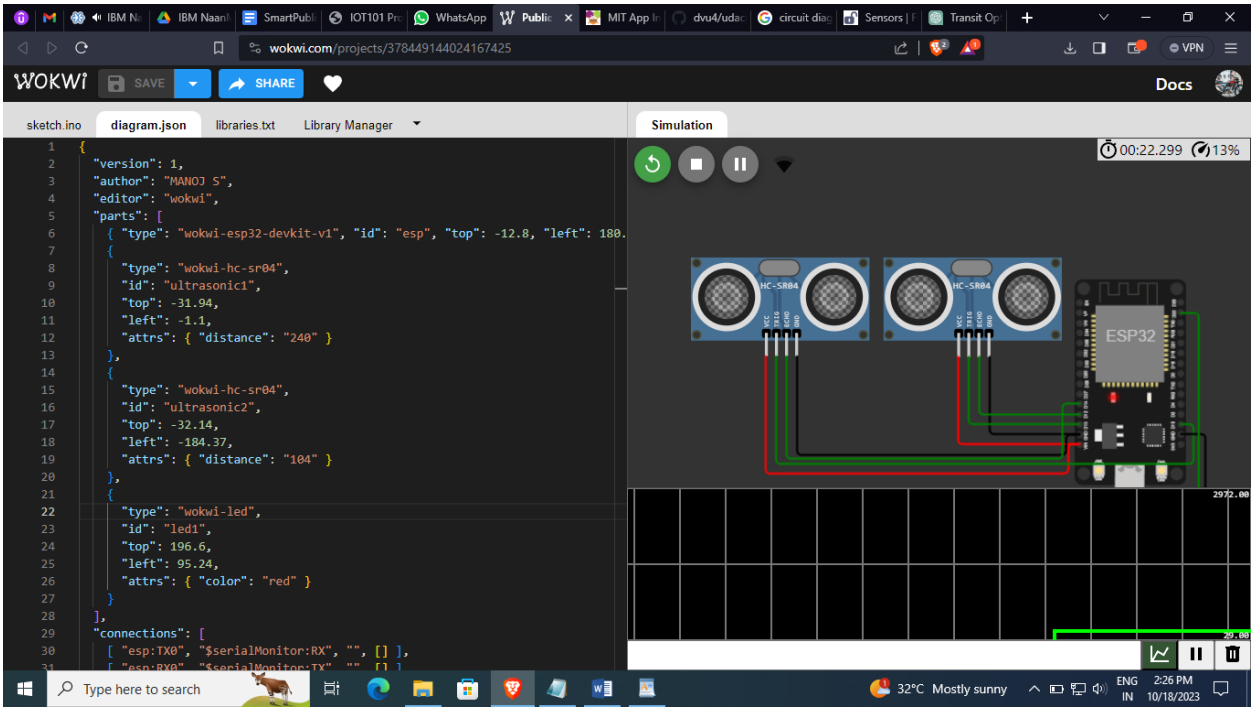


Fig: Started the intialize



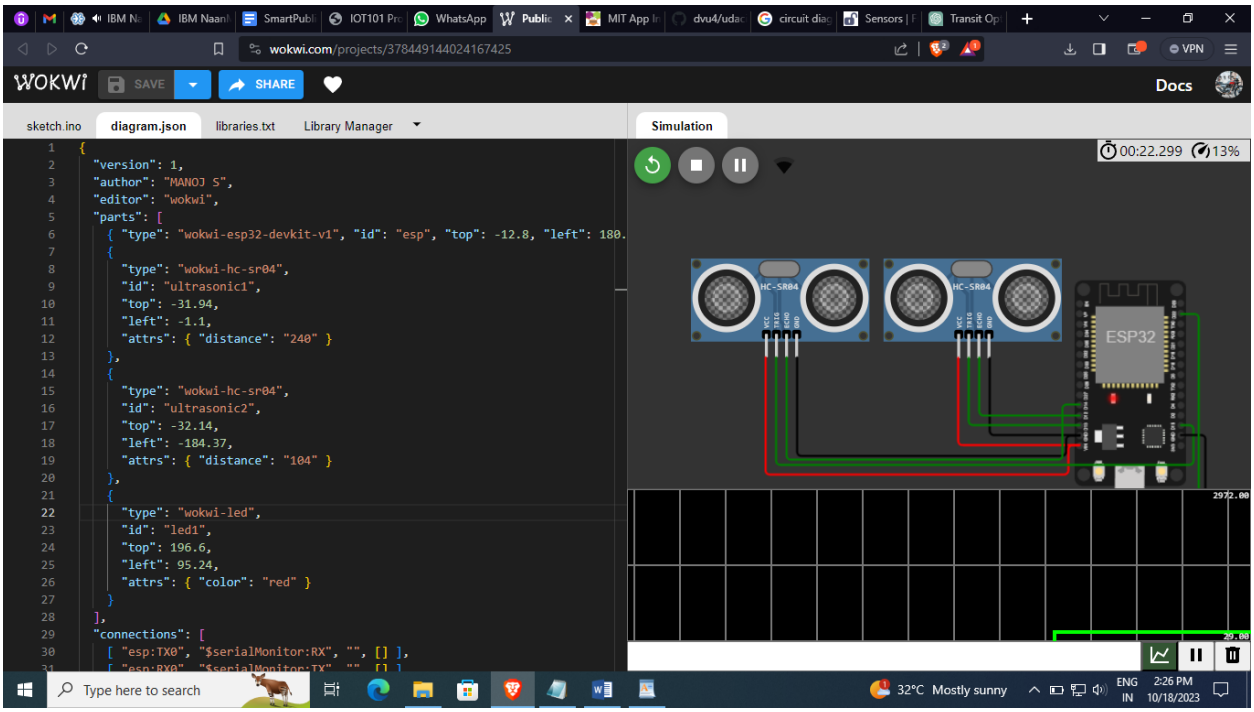


Fig:Optimal of program

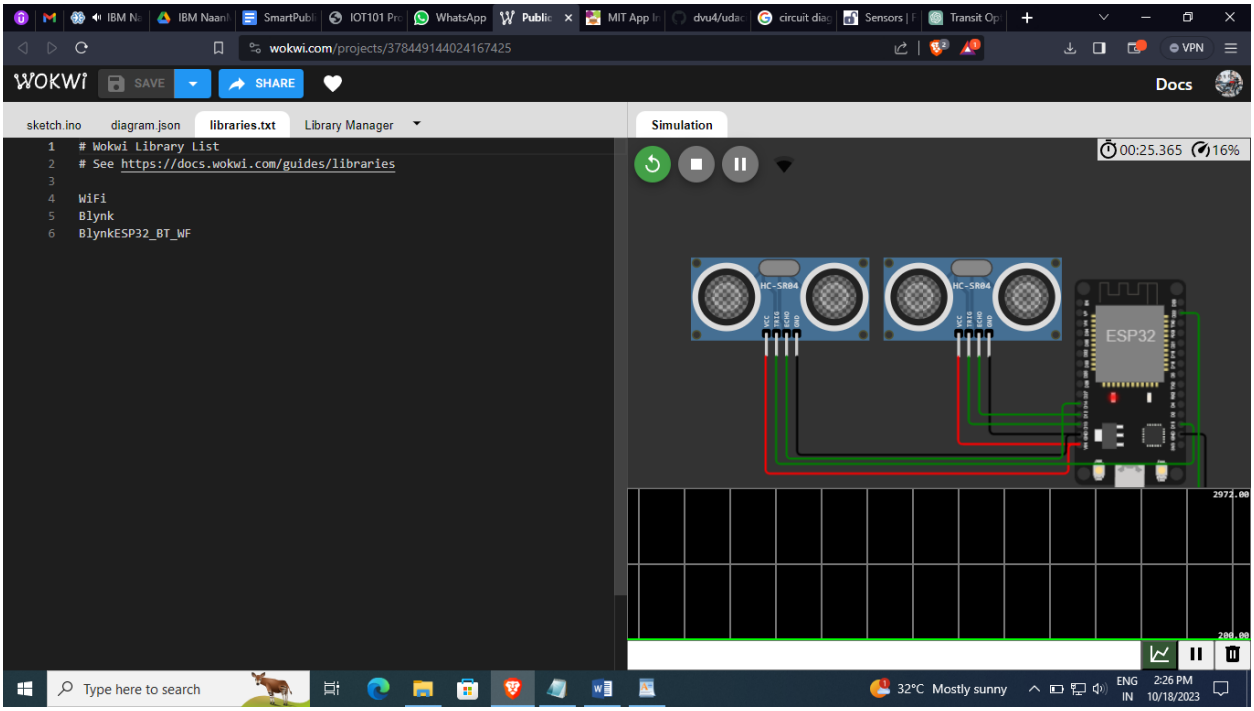


Fig: Initialize the code

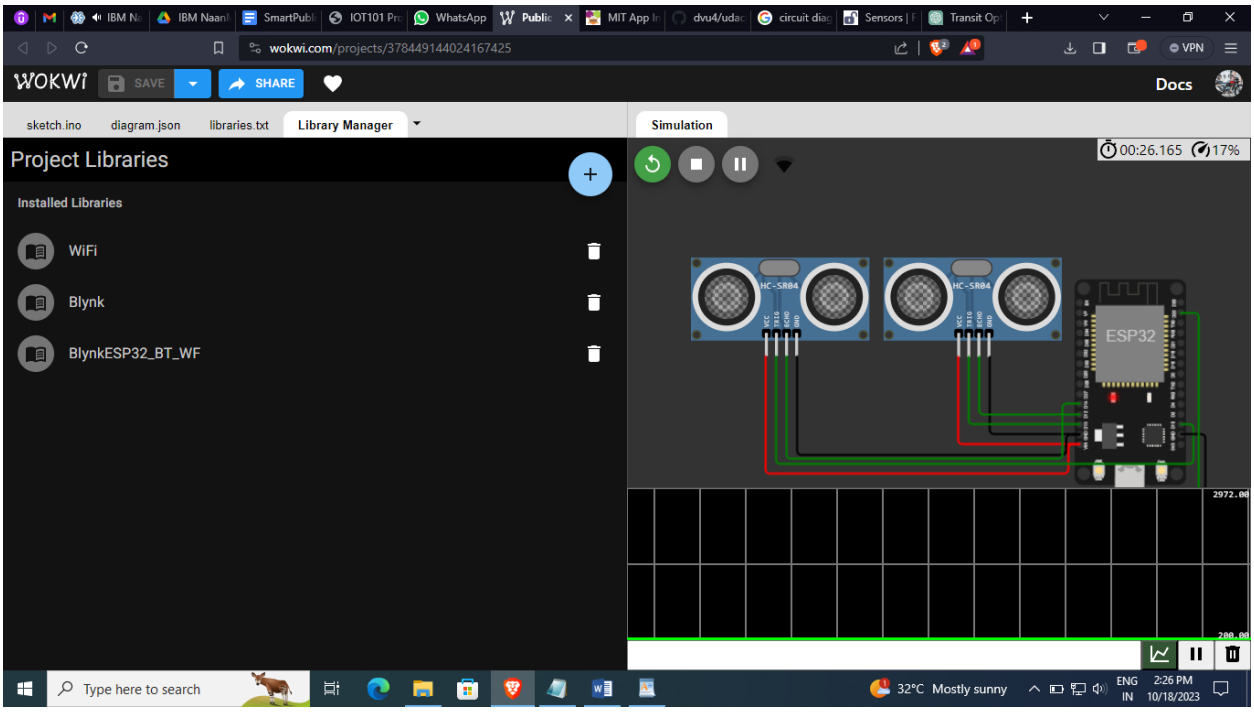
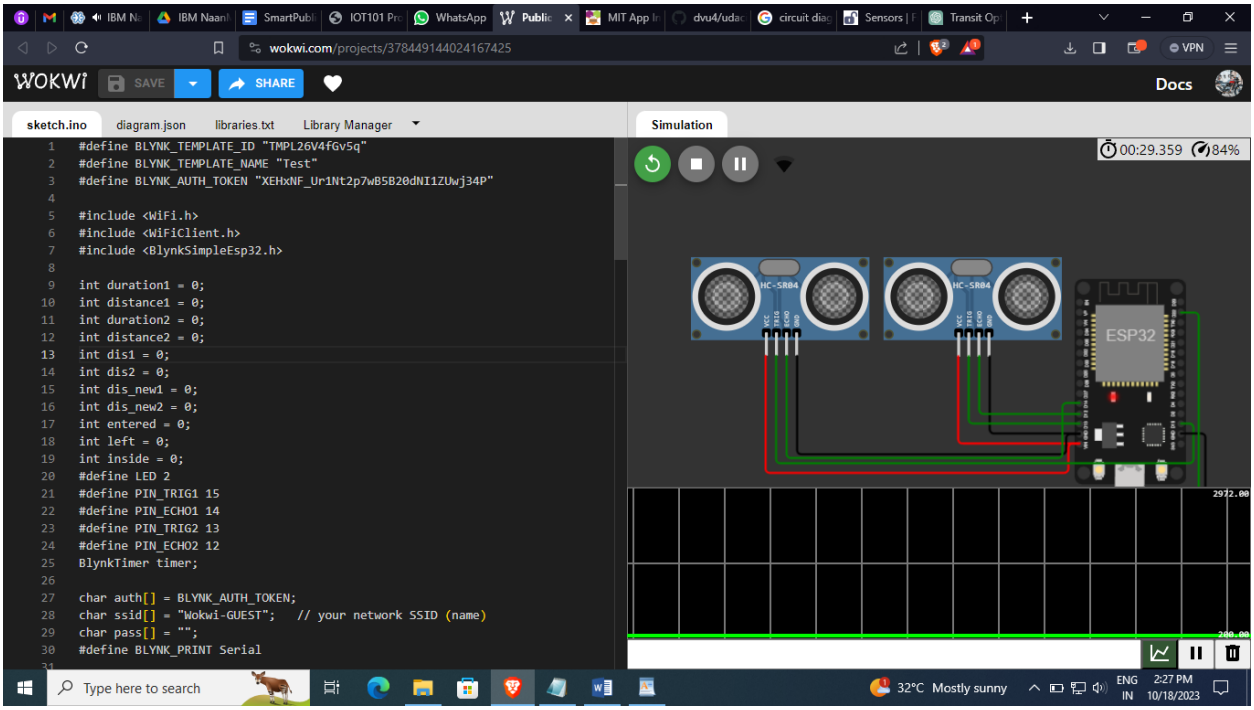
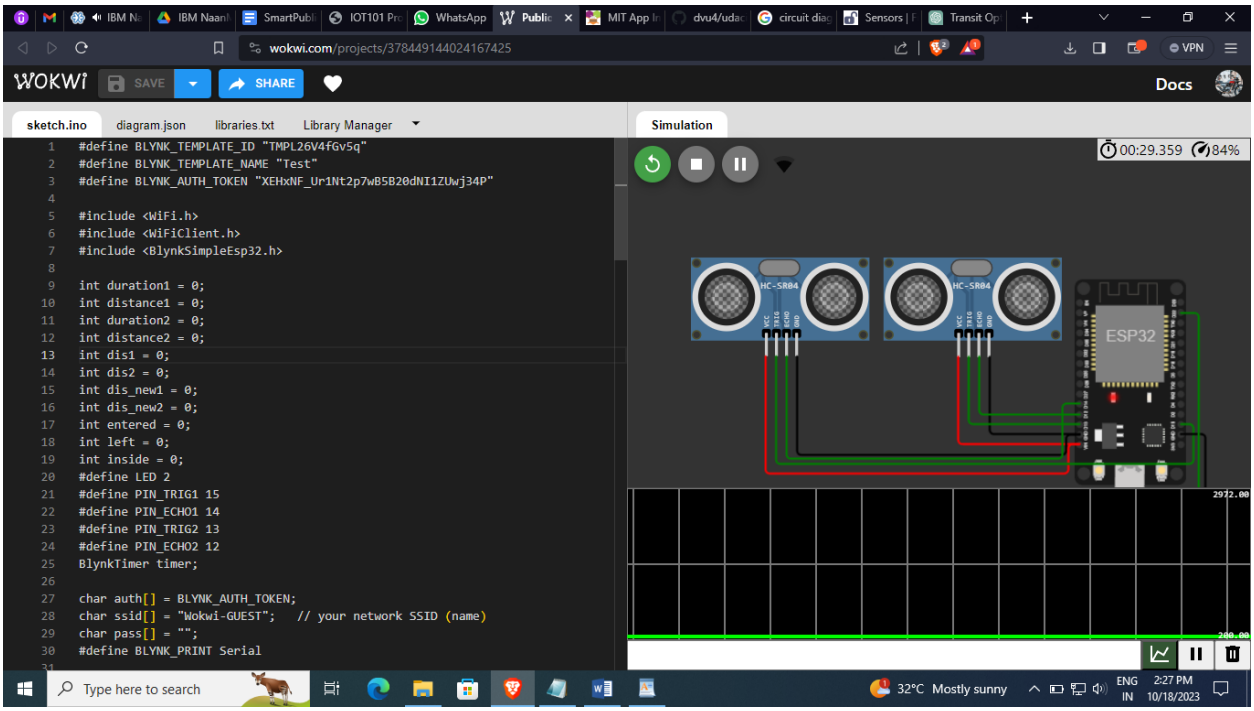


Fig describe the optimization of US Wave


wokwi.com/projects/378449144024167425
WOKWI
SAVE
SHARE
Docs
sketch.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:29.359
84%
HC-SR04
HC-SR04
ESP32
2972.00
280.00
#define BLYNK_TEMPLATE_ID "TMPL26V4fGv5q"
#define BLYNK_TEMPLATE_NAME "Test"
#define BLYNK_AUTH_TOKEN "XEHxNF_Ur1Nt2p7wB5B20dNI1ZUwj34P"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
int duration1 = 0;
int distance1 = 0;
int duration2 = 0;
int distance2 = 0;
int dis1 = 0;
int dis2 = 0;
int dis_new1 = 0;
int dis_new2 = 0;
int entered = 0;
int left = 0;
int inside = 0;
#define LED 2
#define PIN_TRIG1 15
#define PIN_ECHO1 14
#define PIN_TRIG2 13
#define PIN_ECHO2 12
BlynkTimer timer;
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST"; // your network SSID (name)
char pass[] = "";
#define BLYNK_PRINT Serial
Type here to search
32°C Mostly sunny
ENG
IN
2:27 PM
10/18/2023


wokwi.com/projects/378449144024167425
WOKWI
SAVE
SHARE
Docs
sketch.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:29.359
84%
HC-SR04
HC-SR04
ESP32
2972.00
280.00
#define BLYNK_TEMPLATE_ID "TMPL26V4fGv5q"
#define BLYNK_TEMPLATE_NAME "Test"
#define BLYNK_AUTH_TOKEN "XEHxNF_Ur1Nt2p7wB5B20dNI1ZUwj34P"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
int duration1 = 0;
int distance1 = 0;
int duration2 = 0;
int distance2 = 0;
int dis1 = 0;
int dis2 = 0;
int dis_new1 = 0;
int dis_new2 = 0;
int entered = 0;
int left = 0;
int inside = 0;
#define LED 2
#define PIN_TRIG1 15
#define PIN_ECHO1 14
#define PIN_TRIG2 13
#define PIN_ECHO2 12
BlynkTimer timer;
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST"; // your network SSID (name)
char pass[] = "";
#define BLYNK_PRINT Serial
Type here to search
32°C Mostly sunny
ENG
IN
2:27 PM
10/18/2023