PUBLIC TRANSPORT OPTIMIZATION

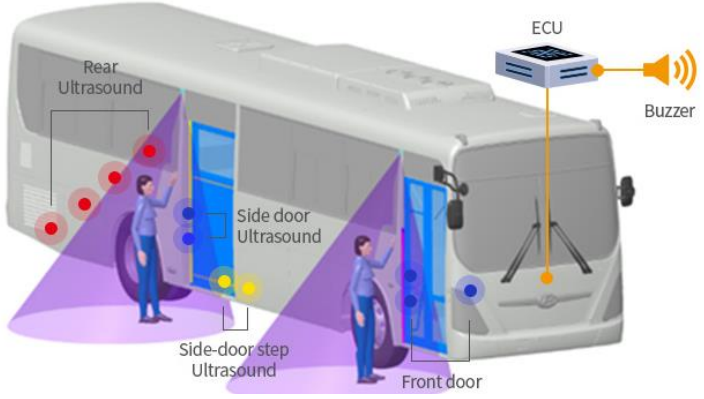
Phase-5: Documentation and Submission

PROJECT OBJECTIVE:

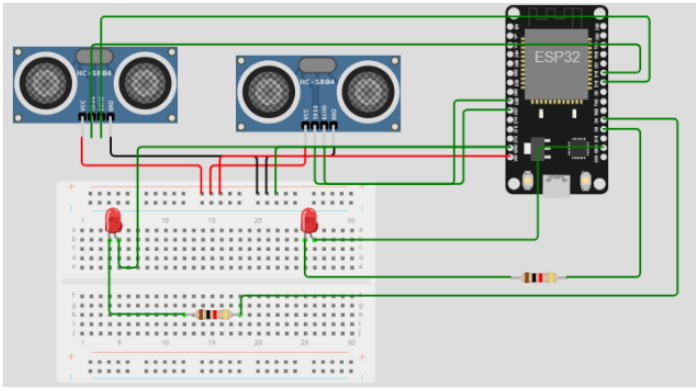
Objective of the project is to optimize the operation of the public transport vehicles (i.e., Buses, etc.) using IoT technology to predict arrival time and real-time locations of the vehicles and reduce waiting times, inform traffic status and rider-ship details to the passengers. These processes are improving the Quality of Services on Public Transports.

IOT DEVICE SETUP:

- Hardware components like **ESP32 Microcontroller, HC-SR04-Ultrasonic Sensors, Breadboard, Wi-Fi Modem, Neo-M6 GPS Module, connecting wires and SSD1306 Display** are used as an IoT devices in this project. **Python** programming language is dumped into the controller. The IoT protocols like **MQTT, AMQP** and **CoAP** are used to transfer the IoT data to the cloud server and also retrieve the data.
- Ultrasonic sensors placed at the entrance/exit doorway of the transport vehicles

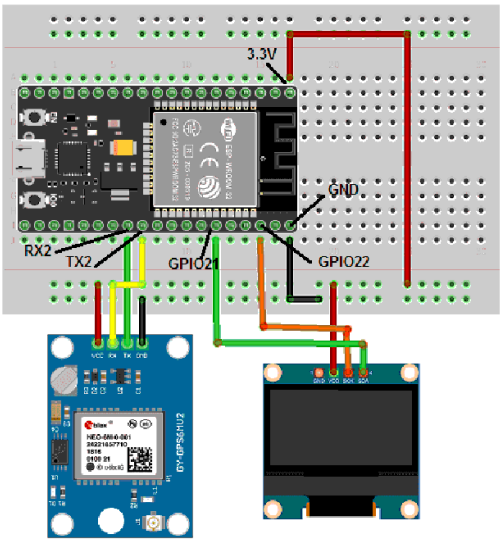


- **Passenger Exit and Enter Detector** using Ultrasonic Sensors and ESP32 Controller



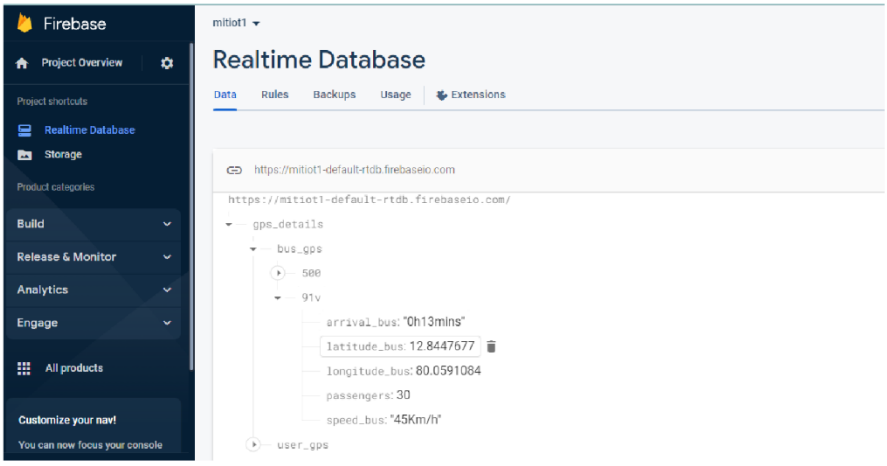
- **GPS Module** placed inside the vehicle (Controller section). It predicts real-time location and Speed of the vehicles. These parameters can used to calculate the arrival time.

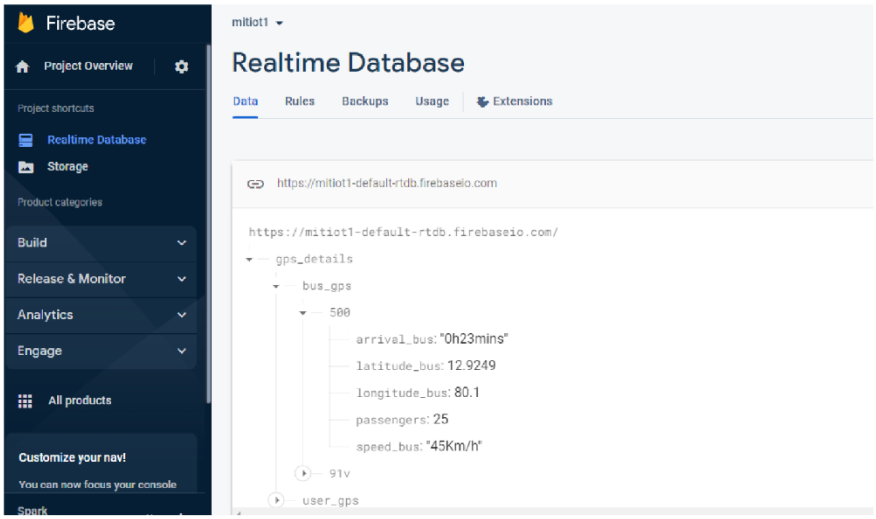
- **SSD1306 Display** project the GPS module collected data. Below fig. is a schematic



PLATFORM DEVELOPMENT:

The IoT platform built on cloud-based architecture. It employs microservices for data ingestion, real-time processing, predictive analytics and reporting. In this IoT project select the cloud-based platform as a **Google Firebase** to store the real-time data of the public transports and users. It is an open-source cloud platform. Using the python script each embedded node uploads the data to the **Firebase cloud** using internet. Some other cloud platforms are (**HiveMQ (based on MQTT broker), AWS, MS AZURE).**





CODE IMPLEMENTATION:

- **Before implement python program we ensure python libraries are available or not.**
- **Python Code for Passenger detector and Uploading data to the real-time Firebase database.**

```
import time
import math
from machine import Pin

#Importing firebase tools
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db

#Connecting wifi
import network
print("Connecting to WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
  print(".", end="")
  time.sleep(0.1)
print(" Connected!")

# Initialize Firebase Admin SDK
cred = credentials.Certificate('mitiot1-firebase-adminsdk-g6nw3-d317fcae89.json')
firebase_admin.initialize_app(cred,{'databaseURL':'https://mitiot1-default-
 rtdb.firebaseio.com/'})
bus_id=db.reference('gps_details/bus_gps/91V')

#Initialize Pins
trigger1 = Pin(19, Pin.OUT)
echo1 = Pin(18, Pin.IN)
trigger2 = Pin(27, Pin.OUT)
echo2 = Pin(26, Pin.IN)

distance_detection = 10
#Initial Enter/Exit Passengers
enter = 0
exit = 0
Total=0

# Function to measure distance
def measure_distance1():
    pulse_duration1=0
    pulse_start1=0
    pulse_end1=0
```

```
    distance1 = 50
    # Send a 10us pulse on the trigger pin
    trigger1.value(1)
    time.sleep_us(10)
    trigger1.value(0)

    # Wait for the echo pin to go high
    while echo1.value() == 0:
        pulse_start1 = time.ticks_us()

    # Wait for the echo pin to go low
    while echo1.value() == 1:
        pulse_end1 = time.ticks_us()

    # Calculate the pulse duration and convert to distance (in centimeters)
    pulse_duration1 = time.ticks_diff(pulse_end1, pulse_start1)
    distance1 = (pulse_duration1 / 2) / 29.1  # Speed of sound in air is
 approximately 343 meters per second

    return distance1

def measure_distance2():
    pulse_duration2=0
    distance2 = 50
    # Send a 10us pulse on the trigger pin
    trigger2.value(1)
    time.sleep_us(10)
    trigger2.value(0)

    # Wait for the echo pin to go high
    while echo2.value() == 0:
        pulse_start2 = time.ticks_us()

    # Wait for the echo pin to go low
    while echo2.value() == 1:
        pulse_end2 = time.ticks_us()

    # Calculate the pulse duration and convert to distance (in centimeters)
    pulse_duration2 = time.ticks_diff(pulse_end2, pulse_start2)
    distance2 = (pulse_duration2 / 2) / 29.1  # Speed of sound in air is
 approximately 343 meters per second

    return distance2


try:
    while True:
        distance1 = measure_distance1()
        print("Distance1_US1: {:.2f} cm".format(distance1))
        distance2 = measure_distance2()
        print("Distance2_US2: {:.2f} cm".format(distance2))
        if (distance1 < distance_detection):
            print("Passenger Enter!")
```

```
            Total+=1
            Pin(4,Pin.OUT).value(1)
            time.sleep(5)
            Pin(4,Pin.OUT).value(0)
            time.sleep(5)
        elif (distance2 < distance_detection):
            print("Passenger Exit!")
            Total-=1
            Pin(2,Pin.OUT).value(1)
            time.sleep(5)
            Pin(2,Pin.OUT).value(0)
            time.sleep(5)
        else:
            Pin(4,Pin.OUT).value(0)
            time.sleep(5)
            Pin(2,Pin.OUT).value(0)
            time.sleep(5)

        print("Total Passengers: {: }".format(Total))

        #Uploading Data to Firebase
        data={'passengers': Total}
        bus_id.update(data)

        time.sleep(5) # Wait for a second before taking another reading

except KeyboardInterrupt:
    pass
```

- **Python Code for GPS Tracker and Upload/Get the data from Firebase for some calculations**

```
import time
import machine
from micropyGPS import MicropyGPS
from machine import Pin, I2C
import ssd1306
import _thread

#Connecting wifi
import network
print("Connecting to WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
  print(".", end="")
  time.sleep(0.1)
print(" Connected!")

#Importing firebase tools
```

```
  import firebase_admin
  from firebase_admin import credentials
  from firebase_admin import db

  #Initialize Pins
  i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
  oled_width = 128
  oled_height = 64
  oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

  oled.text('Hello', 0, 0)
  oled.text('ESP32', 0, 10)
  oled.show()

  def main():

      uart = machine.UART(1, rx=16, tx=17, baudrate=9600, bits=8, parity=None,
stop=1, timeout=5000, rxbuf=1024)

      gps = MicropyGPS()
      latitudes=[]
      longitudes=[]
      timestamps=[]
      speeds=[]
      i=0
      j=0

      #get user gps data from firebase
      la=db.reference('gps_details/user_gps/latitude_user').get()
      lo=db.reference('gps_details/user_gps/longitude_user').get()
      lat1 = float(la)
      lon1 = float(lo)

      def arrival_time(lat1, lon1, lat2, lon2)
          def distance_finder(lat1, lon1, lat2, lon2):
              # Radius of the Earth in kilometers
              R = 6371
              lat1 = math.radians(lat1)
              lon1 = math.radians(lon1)
              lat2 = math.radians(lat2)
              lon2 = math.radians(lon2)
              dlat = lat2 - lat1
              dlon = lon2 - lon1
              a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) *
math.sin(dlon/2)**2
              c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
              dist= R * c
              return dist
          distance=distance_finder(lat1, lon1, lat2, lon2)
          local_time1= (distance / speed)
          time_interval='{0:02.0f}:{1:02.0f}'.format(*divmod(local_time1 * 60, 60))
          hours, minutes = map(int, '{}'.format(time_interval).split(':'))
          arrival='{}h{}mins'.format(hours,minutes)
```

```
            return arrival

        def calculate_speed(latitudes, longitudes, timestamps):
            speeds = []
            j=i-1
            lat1 = latitudes[i]
            lon1 = longitude[i]
            lat2 = latitudes[j]
            lon2 = longitudes[j]
            time_interval = timestamps[i] - timestamps[j]
            distance = distance_finder(lat1, lon1, lat2, lon2)
            speed = distance / time_interval
            return speed


        while True:
            buf = uart.readline()
            if uart.any():
               for char in buf:
                   gps.update(chr(char))  # Note the conversion to to chr, UART
    outputs ints normally

                print('UTC Timestamp:', gps.timestamp)
                timestamps.append(gps.timestamp)
                print('Date:', gps.date_string('long'))
                print('Latitude:', gps.latitude)
                latitudes.append(gps.latitude)
                print('Longitude:', gps.longitude_string())
                longitudes.append(gps.longitude)
                if i=>1:
                     print("Speed:",calculate_speed(latitudes, longitudes,
    timestamps))
                print('Horizontal Dilution of Precision:', gps.hdop)
                print('Altitude:', gps.altitude)
                print('Satellites:', gps.satellites_in_use)
                print()
                i+=1

                # Send data to Firebase
                arrival_bus_time = arrival_time(lat1, lon1, lat2, lon2)
                bus_speed = calculate_speed(latitudes, longitudes, timestamps)
                bus_id=db.reference('gps_details/bus_gps/91V')
                bus_id.update()

                data = {
                     'latitude_bus' gps.latitude,
                     'longitude_bus' :gps.longitude ,
                     'speed_bus' : bus_speed,
                     'arrival_bus': arrival_bus_time
                    }

                bus_id.update(data)
                print("Data sent to Firebase:", data)
```

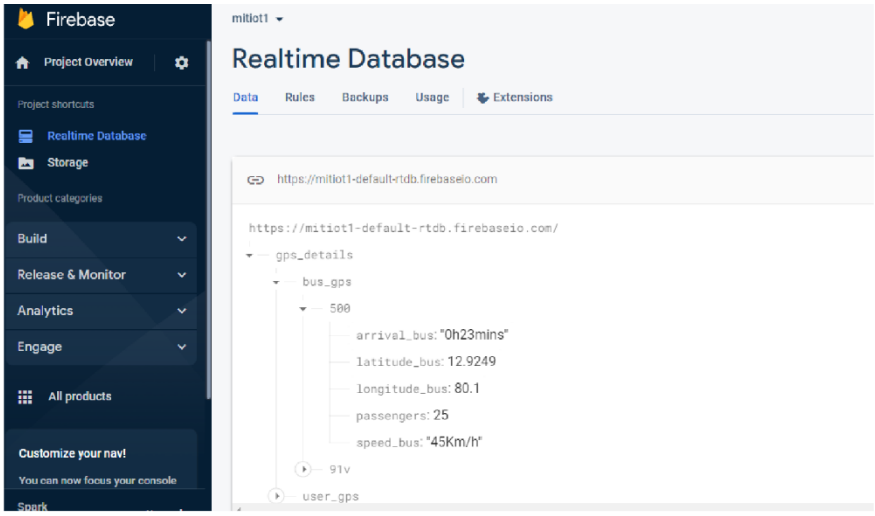
```
            oled.fill(0)
            y = 0
            dy = 10
            oled.text("{}".format(gps.date_string('s_mdy')), 0, y)
            oled.text("Sat:{}".format(gps.satellites_in_use), 80, y)
            y += dy
            oled.text("{:02d}:{:02d}:{:02.0f}".format(gps.timestamp[0],
gps.timestamp[1], gps.timestamp[2]),  0, y)
            y += dy
            oled.text("Lat:{}{:3d}'{:02.4f}".format(gps.latitude[2],
gps.latitude[0], gps.latitude[1]),  0, y)
            y += dy
            oled.text("Lon:{}{:3d}'{:02.4f}".format(gps.longitude[2],
gps.longitude[0], gps.longitude[1]),  0, y)
            y += dy
            oled.text("Alt:{:0.0f}ft".format(gps.altitude * 1000 / (12*25.4)),  0,
y)
            y += dy
            oled.text("HDP:{:0.2f}".format(gps.hdop),  0, y)
            oled.show()

    def startGPSthread():
        _thread.start_new_thread(main, ())

    if __name__ == "__main__":
      print('...running main, GPS testing')
      main()
```

- **Output after program Executed:**

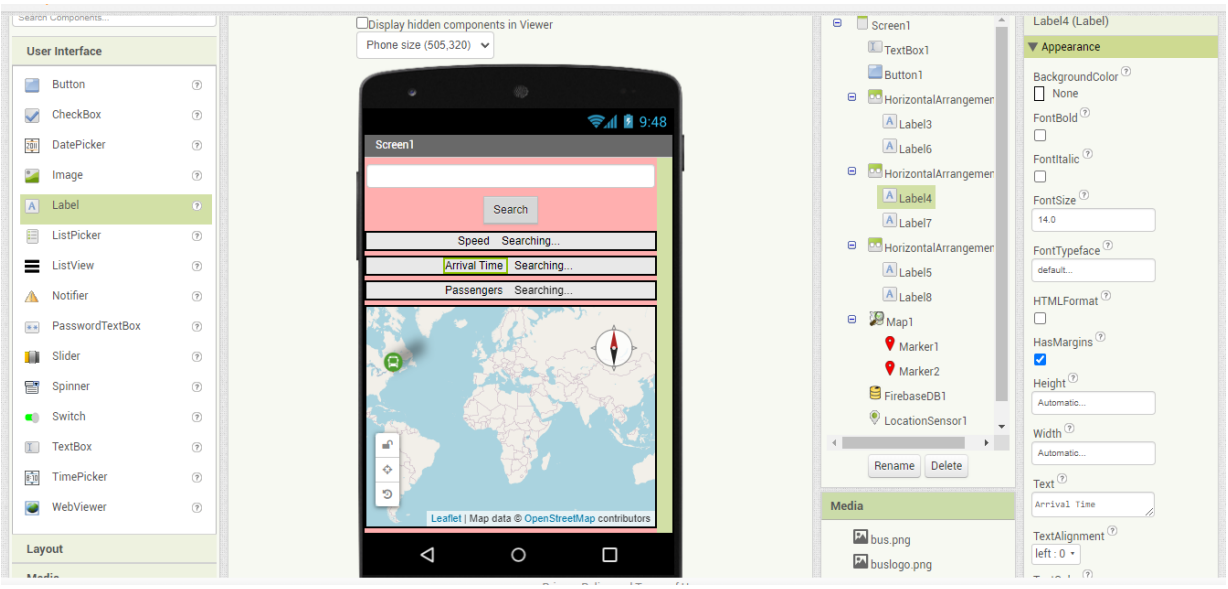
  While running a python script in controller the collected data from IoT devices upload securely after a particular time delay.



DATA-SHARING PLATFORM:

The data transfer from cloud to public platform using the potocols to ensure the security. The **MIT App Inventor** is a open-source platform for create the web based app and display the real-time locations and ridership details of the public transport. There many open-source are available like **Kodular, Flutter, Blynk Server, etc.**

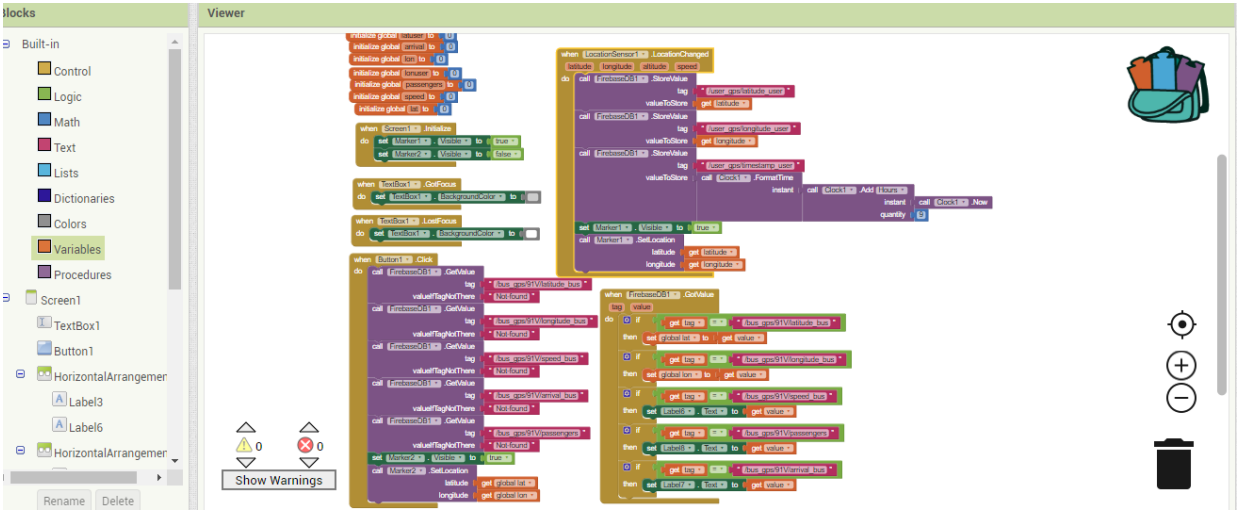
- **Initialize the User Interface:**



MIT App Inventor is a free mobile apps creator based-on pick and paste technology without no coding required. User can easily modify the user interface. FirebaseDB1 parameter in the MIT App inventor used to get the data form the desired **Firebase Real-Time Database.**

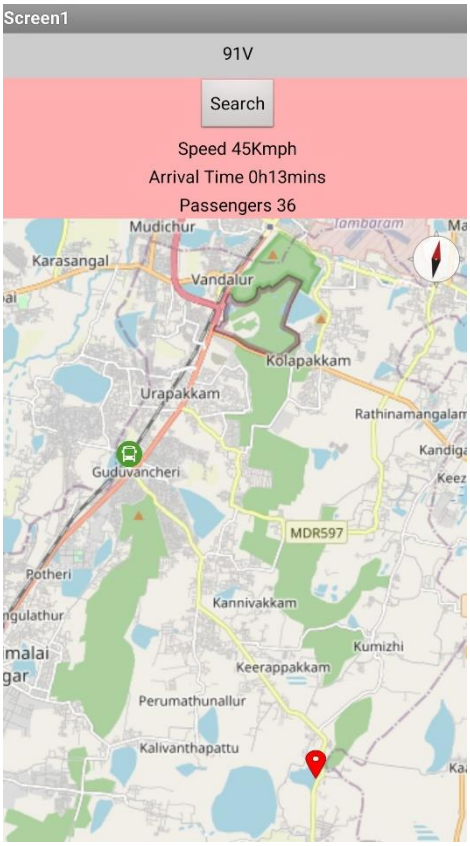
- **Assign functions to each block(parameters):**

Without the coding we can easily assign tasks to each block for required application oriented. Task can be assigned for each event occurs in the screen. The following figure show how task assigned to this IoT project.



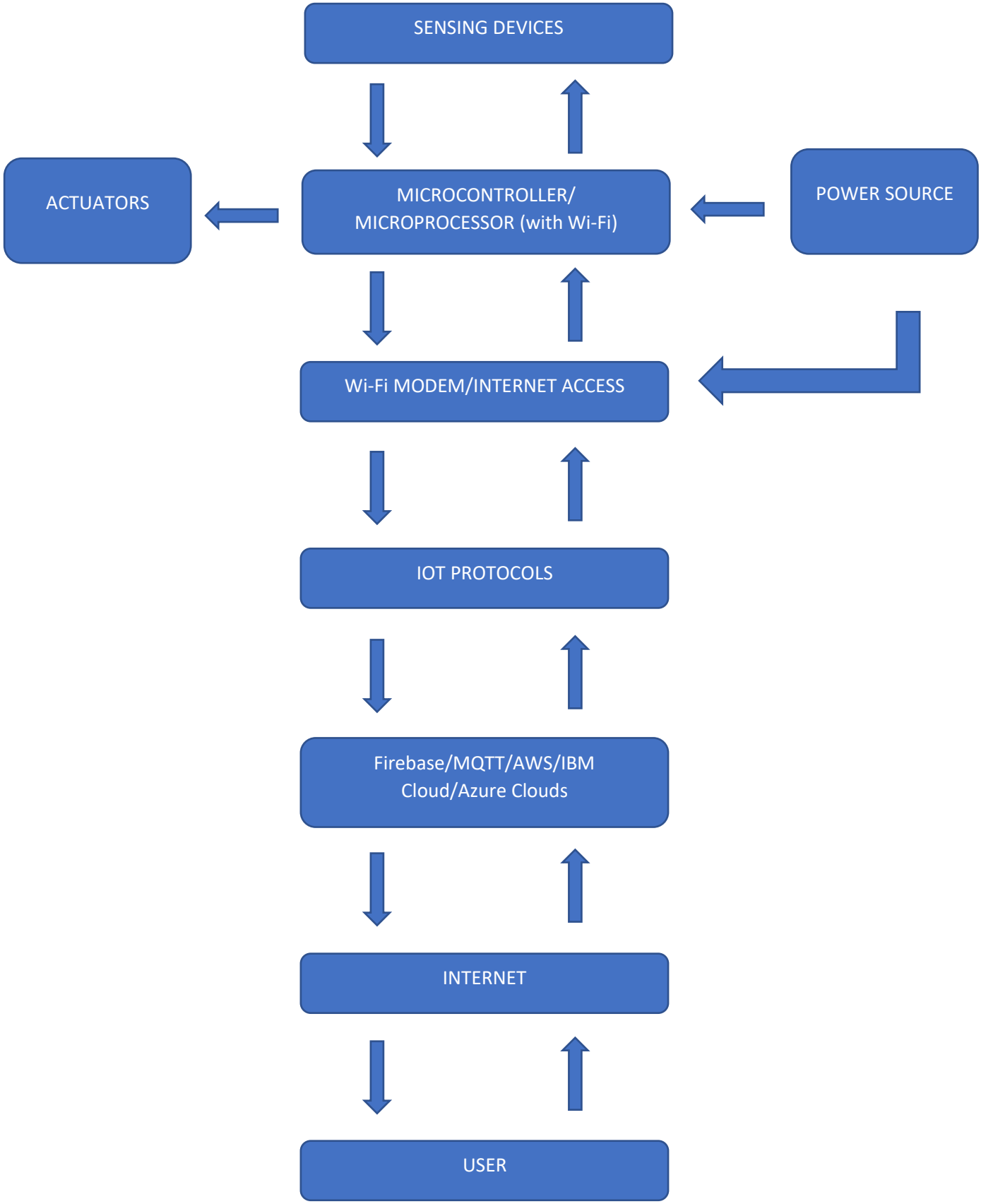
(this fig. shows example of bus id 91V)

- **Output at the passenger mobile:**



After enter the bus id passenger can easily know about the real-time location, arrival time, speed and passenger counts from the IoT devices implemented in Public Transport using Firebase(or other cloud server) Database.

FLOWCHART:


SENSING DEVICES
ACTUATORS
MICROCONTROLLER/
MICROPROCESSOR (with Wi-Fi)
POWER SOURCE
Wi-Fi MODEM/INTERNET ACCESS
IOT PROTOCOLS
Firebase/MQTT/AWS/IBM
Cloud/Azure Clouds
INTERNET
USER