TRACK : TRINIT_BYTEHOGS_IOT_02

TEAM : BYTE HOGS

TOPIC : Drift

Approach Note:

Problem Statement:

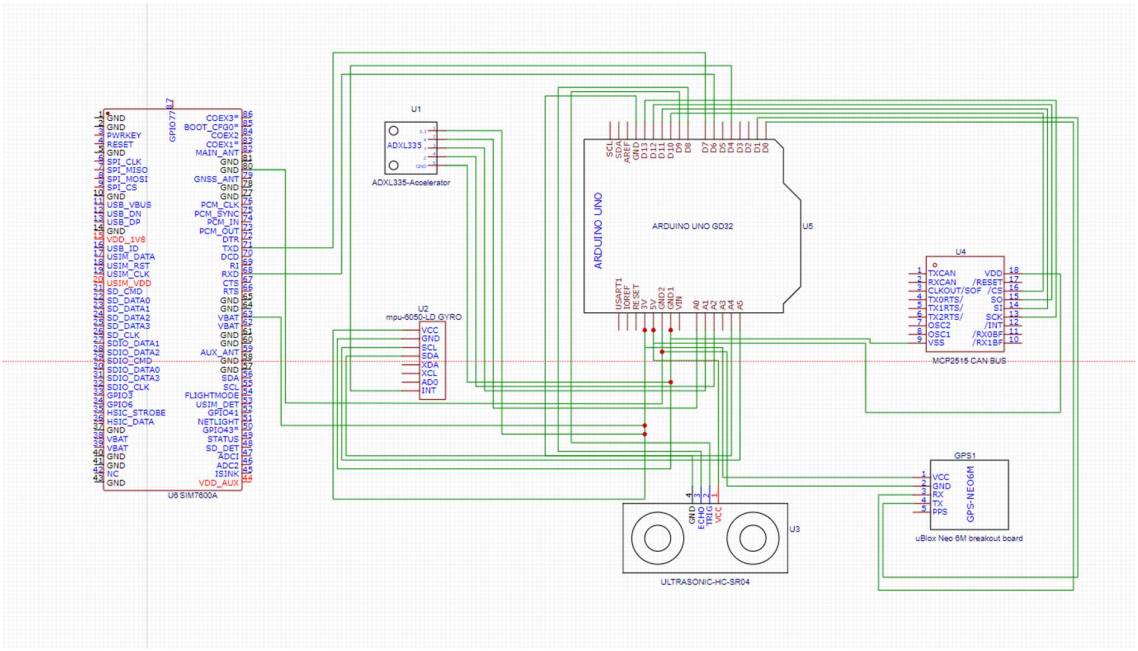
- To design an IoT edge module for each vehicle which replaces the need for traffic lights at intersections so that vehicles communicate with each other and smoothen the traffic flow.
- Implementing V2X communication that should help solve whenever there is a traffic hold-up.

Solution:

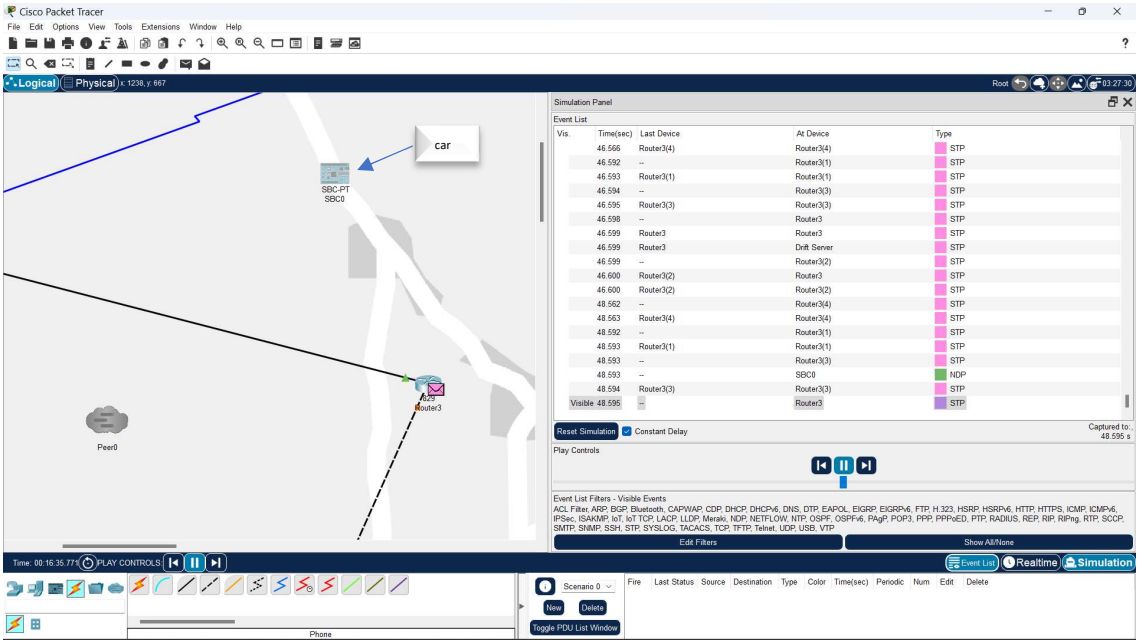
The proposed solution for this problem statement is to use Vehicle-to-Everything (V2X) communication technology, Long-Term Evolution (LTE) communication, and sensors along with an Arduino microcontroller to design an IoT edge module for each vehicle. The IoT edge module will function as a replacement for traffic lights at intersections.

Here are the steps from which we implemented the solution:

- Design the IoT edge module: The IoT edge module should have an Arduino microcontroller, LTE communication module, GPS sensor, and other necessary sensors such as an accelerometer, gyroscope, and GSM etc. These sensors will be used to measure the vehicle's speed, direction, and position.

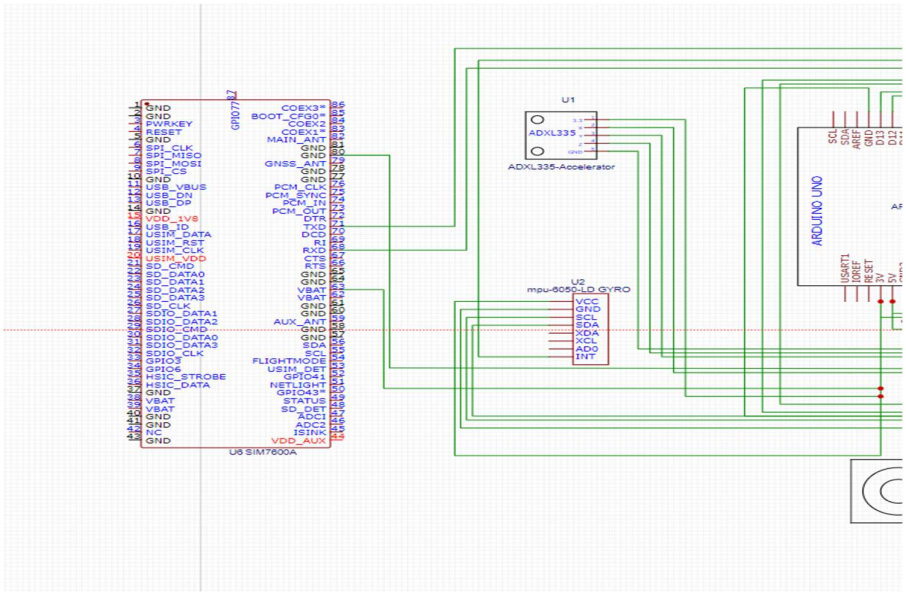


Integrate V2V communication: The IoT edge module is integrated with V2V communication technology. This will enable vehicles to communicate with each other and exchange data such as their speed, direction, and position.



Implement traffic flow algorithms: The IoT edge module should implement algorithms that can analyse the data received from other vehicles and predict the flow of traffic. This will allow vehicles to smoothly navigate through intersections without the need for traffic lights.

Integrate LTE communication: The IoT edge module should also be integrated with LTE communication. This will enable the vehicles to



communicate with the central server, which can monitor the traffic flow in real-time and make necessary adjustments to improve traffic flow.

Test the solution: Finally, the solution should be tested in real-world scenarios to ensure its effectiveness.

By implementing this solution, vehicles will be able to communicate with each other and the central server, allowing for smoother traffic flow and reducing the need for traffic lights at intersections.

The overall working of this circuit can be explained as follows:

A smart driving system is implemented using V2X communication, which integrates various sensors to provide various driving-related data. The data from these sensors is processed and sent to the other vehicles or a central server through the LTE module.

Let's explain the circuit logic for each of the sensors:

GPS Neo 6M:

The GPS Neo 6M module receives data from GPS satellites and provides the current position of the vehicle in terms of latitude and longitude coordinates. The module is connected to the microcontroller through the serial communication pins, RX (D0) and TX (D1). The microcontroller reads the data from the module and processes it.

HC SR04:

The HC SR04 is an ultrasonic sensor used for measuring the distance between the sensor and an object. The sensor has two pins, trigger (D9) and echo (D8). The microcontroller sends a trigger signal to the sensor through the trigger pin, and the sensor sends back an echo signal after reflecting off the object. The duration of the echo signal is proportional to the distance between the sensor and the object. The microcontroller measures the duration of the echo signal and calculates the distance.

MCP2515:

The MCP2515 is a CAN (Controller Area Network) controller used to communicate with other vehicles or with a central server. The MCP2515 is connected to the microcontroller through the SPI communication pins, SCK (D13), SI (D11), SO (D12), and chip select (D10). The microcontroller communicates with the MCP2515 to send and receive data on the CAN network.

MPU-6050:

The MPU-6050 is an accelerometer used to measure the acceleration and orientation of the vehicle. The accelerometer is connected to the microcontroller through the I2C communication pins, SCL (A5) and SDA (A4), and an interrupt pin (D4). The microcontroller communicates with the accelerometer to read the acceleration and orientation data and processes it.

ADXL335:

The ADXL335 is an accelerometer used to measure the acceleration of the vehicle in the x, y, and z-axis. The accelerometer is connected to the microcontroller through the analog input pins, X (A0), Y (A1), and Z (A2). The microcontroller reads the analog values from the accelerometer and converts them into acceleration values.

SIM7600:

The SIM7600 is an LTE module used to send and receive data from the other vehicles or a central server. The LTE module connected to the microcontroller through the serial communication pins, TXD (D7) and RXD (D6). The microcontroller communicates with the LTE module to send and receive data through the LTE network.

ROUTER:

-The router plays a vital role in providing internet connectivity for the LTE module. The LTE module communicates with the other vehicles or a central server through the LTE network, but it requires an internet connection to do so. The router provides the internet connection to the LTE module by connecting to the internet service provider.

-The router acts as a gateway between the LTE module and the internet. The data received by the LTE module is transmitted to the router, and the router forwards the data to the internet, where it reaches the other vehicles or a central server. Similarly, the data received from the other vehicles, or a central server is transmitted to the router, and the router forwards the data to the LTE module, which is then processed by the microcontroller.

-In short, the router acts as an intermediary that allows the LTE module to communicate with the other vehicles or a central server over the internet. Without the router, the LTE module would not be able to send and receive data through the LTE network.

CELLULAR TOWER:

-the cell tower plays a crucial role in providing the communication network for the LTE module. The LTE module communicates with other vehicles or a central server through the LTE network. The cell tower is the infrastructure that enables this communication by providing a network coverage area for the LTE module.

-The cell tower is responsible for transmitting and receiving data from the LTE module. When the LTE module sends data, it sends the data to the nearest cell tower. The cell tower then relays the data to the other vehicles or the central server through the LTE network. Similarly, when data is received from the other vehicles or the central server, it is transmitted to the nearest cell tower, which then relays the data to the LTE module.

-In short, the cell tower acts as the backbone of the LTE network and provides the communication infrastructure for the LTE module. Without the cell tower, the LTE module would not be able to communicate with other vehicles or the central server.

In summary,

all the sensors are connected to the microcontroller (Arduino uno), which processes the data from the sensors and communicates with other vehicles or a central server through the LTE module. The processed data can used for various driving-related applications, such as traffic management, accident prevention, etc.

Coding Part:

- **CIRCUIT**

The code is for an Arduino-based system that collects data from several sensors and communicates with a server using LTE.

- The code starts by including various libraries required for the different sensors and communication protocols.

- The software serial communication is set up for the GPS module on RX pin 0 and TX pin 1, and for the SIM7600 LTE module on TX pin 7 and RX pin 6.

- The HC-SR04 sonar sensor is set up with the trigger pin connected to pin 9 and the echo pin connected to pin 8. The maximum distance it can detect is set to 200 cm.

- The MPU6050 accelerometer and gyroscope sensor is set up with I2C communication.

- The ADXL335 accelerometer sensor is set up with its X-axis connected to analog pin A0, Y-axis to A1, and Z-axis to A2.

- In the setup () function, serial communication started at a baud rate of 115200. The GPS and MPU6050 communication is started and the SIM7600 is initialized with a delay of 3 seconds to allow it to start up properly. An "AT" command is sent to the SIM7600 to check if it's working, and the response is printed to the serial monitor.

- In the loop() function, the code reads data from each of the sensors and sends it to the server via the LTE module. The GPS data is read and printed to the serial monitor. The distance from the HC-SR04 sonar measured and the data from the ADXL335 and MPU6050 is read. The sensor data is then formatted as strings and sent to the server via the LTE module, with a delay of 5 seconds between each transmission.

➢ **SHORTEST PATH**

I. We have implemented the Dijkstra's algorithm, which is used to find the shortest path between a source node and a destination node in a graph. Here's a step-by-step explanation of the code:

II. Import the PriorityQueue class from the queue module.

III. Define the ShortestPathV2X function which takes as inputs a graph, the source node, and the destination node.

IV. Initialize an array distances where each element represents the distance of a node in the graph from the source node. All distances are initially set to infinity. The distance of the source node is set to 0.

V. Create a visited set to keep track of the nodes that have already have been processed.

VI. Create a priority queue pq to store the nodes. The priority queue implemented using the PriorityQueue class.

VII. Add the source node to the priority queue with a distance of 0.

VIII. The main loop continues until the priority queue is empty. In each iteration, the node with the lowest distance is removed from the priority queue and processed. If the node has already been processed, the loop continues to the next iteration.

IX. The node is added to the visited set to indicate that it has been processed.

X. For each of the neighbours of the current node, calculate the tentative distance to the neighbours by adding the distance of the current node to the value returned by the get_v2x_info function.

XI. If the tentative distance is less than the current distance of the neighbours, update the distance of the neighbours and add it to the priority queue.

XII. The function returns the distances array after all nodes have been processed.

XIII. Define the get_v2x_info function, which retrieves the Vehicle-to-everything (V2X) communication information between a node and its neighbours. In this implementation, the function simply returns 1 for simplicity.

XIV. Define the graph as a dictionary where each key is a node and its value is a list of its neighbour's.

XV. Set the source and destination nodes.

XVI. Call the ShortestPathV2X function and print the result, which is an array representing the distances of all nodes from the source node.