Transport Of Next Generation(TONG)

Track: Smart Transportation

-Meet TONG, an intelligent traffic management model which efficiently manages traffic in the city, ensures the fastest path to the users, guides them through fog and also detects accidents and alerts the authorities.

Team Halo:

Vanshika Jalan

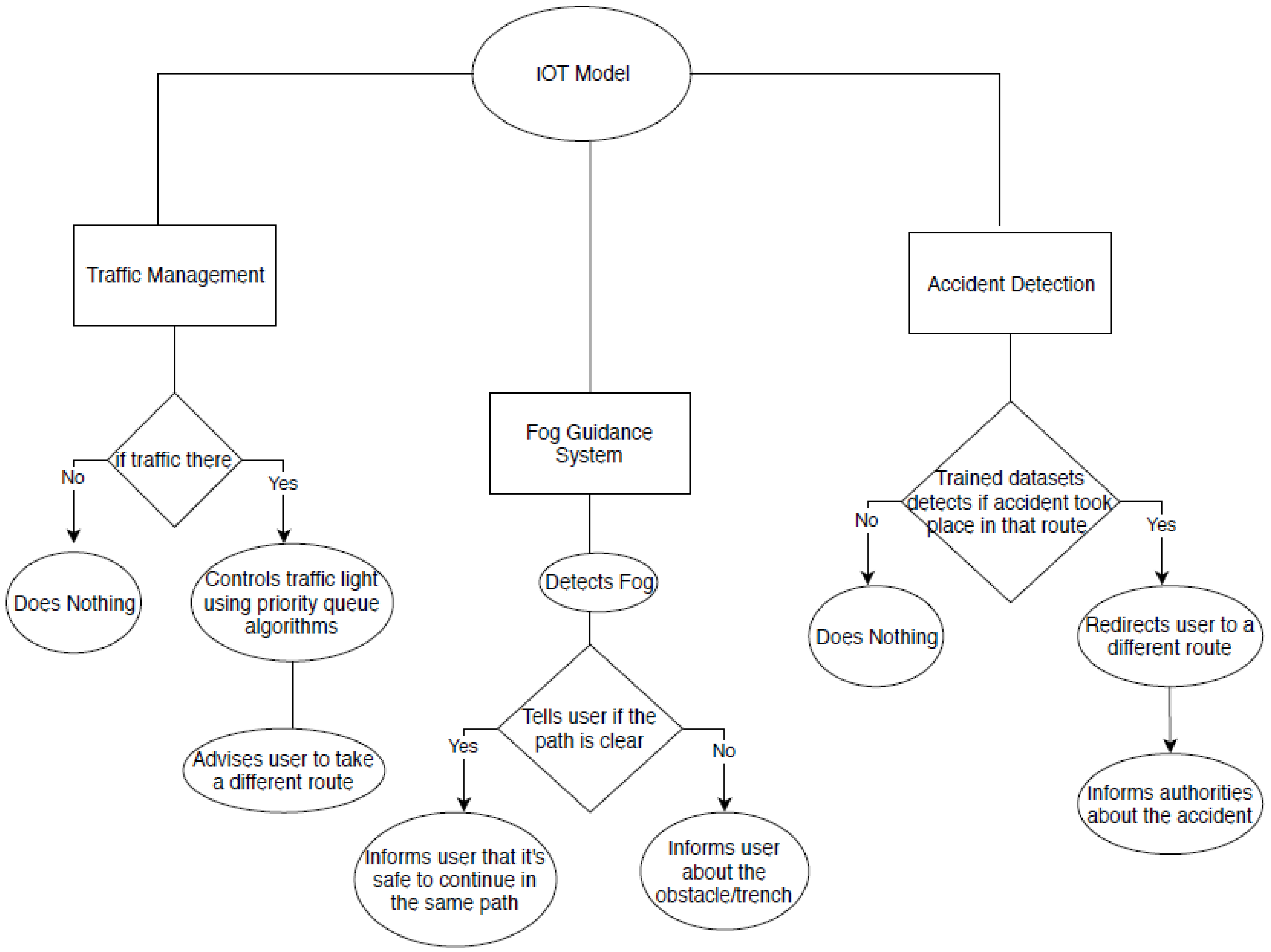
Chanchal Soni

U Sai Rutwik

Sathwik Amburi

Abstract

- Traffic management has been a major problem in the cities since a very long time and with increase in IOT technologies, we're at an era where this can be managed in a more efficient way using certain implementations. We're proposing an intelligent traffic management model alongside a fog guidance and accident detection systems which are managed using IOT. We'll be using an array of sensors such as pressure and proximity sensors which would be spread across the city and along with these, we'll be using OpenCV through traffic cameras and implementing the following:
- 1. Detects traffic and controls the signal lights using priority queue algorithms to decrease the traffic in the city as a whole.
- 2. Identifies accidents and reports it to the authorities.
- 3. Advices the users to take alternate routes to prevent cluttering.
- 4. Fog guidance system detects fog and tells the user about the obstacle/trench forth(if there).

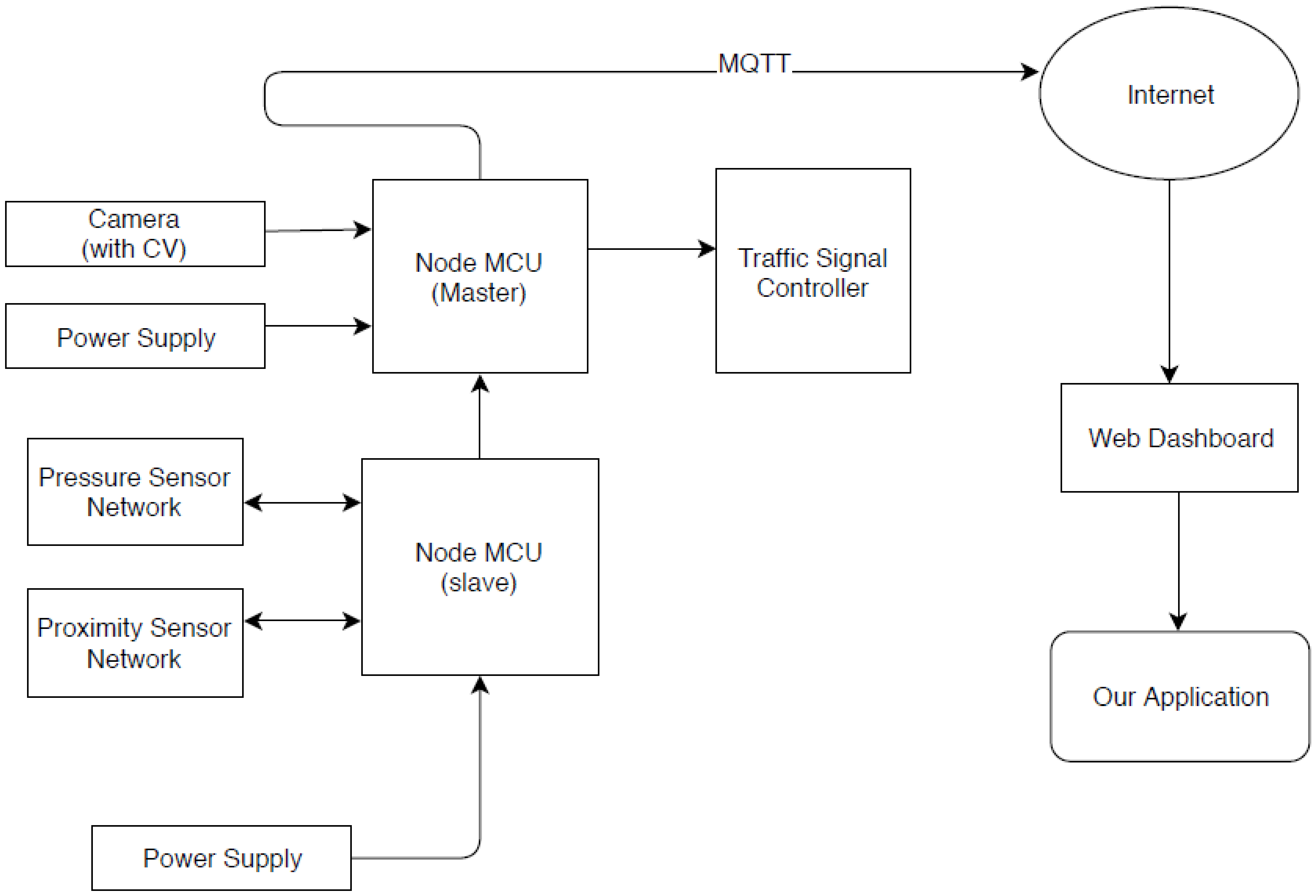

IOT Model
Traffic Management
Fog Guidance System
Accident Detection
if traffic there
No
Yes
Does Nothing
Controls traffic light using priority queue algorithms
Advises user to take a different route
Detects Fog
Tells user if the path is clear
Yes
No
Informs user that it's safe to continue in the same path
Informs user about the obstacle/trench
Trained datasets detects if accident took place in that route
No
Yes
Does Nothing
Redirects user to a different route
Informs authorities about the accident

Software Technologies:

1. Python 3 with Open CV - To train our system to identify the conditions we need (we’ll be using some pretrained models too)
2. Arduino IDE with ESP8266WiFi, ESP8266HTTPClient, Arduino Json and Wire Libraries- To program Arduino/NodeMcu's (First 3 libraries comes in handy for ESP8266 and the Wire Library is for I2C communications)
3. Android Studio for App interface/HTML, CSS, JavaScript for Web interface.

Hardware Equipment:

1. Cameras - To stream the traffic
2. NodeMcu (Master) - To transmit data to our client and to slave NodeMcus and to switch traffic lights based on the traffic
3. Proximity sensors - To measure the distance between our client and the obstacle
4. Pressure sensors - To act as a switch to turn on proximity sensor only if the required mechanical stress conditions are met
5. NodeMcu (Slaves) - To act as a bridge between proximity, pressure sensors and the Master NodeMcus
6. Jumpers - For connecting with Arduino and sensors
7. And of course LEDS - To act as traffic lights


MQTT
Internet
Camera (with CV)
Node MCU (Master)
Traffic Signal Controller
Power Supply
Web Dashboard
Pressure Sensor Network
Node MCU (slave)
Proximity Sensor Network
Our Application
Power Supply

## Modules
Our project is divided into 2 systems - Fog absent, Fog present

#Modules when there is no Fog:
Module - 1 (Analyzing the traffic)
In this module input is traffic analyzed by Open CV and output is sending the data to the master NodeMcu

Module - 2 (Switching traffic lights)
Here input is the data about the traffic send to the master NodeMcu and the master NodeMcu switches the traffic lights depending on the data about the traffic received by the master NodeMcu as output

Module - 3 (Sending data to cloud)
Here input is the data about the traffic and accidents received by the master NodeMcu and the master NodeMcu sends data to cloud using MQTT

Module- 4 (Notifying the user)
Here input is the data about the traffic sent to cloud and output is notifying the user about the traffic in his path and suggesting him a better path

Module - 5 (Notifying the authorities)
Here input is the data about the accidents sent to cloud and the output is notifying the authorities about the accident

#Modules when it's foggy:
1. Module - 1 (Analyzing the traffic)
In this module input is traffic analyzed by Open CV and output is detection of fog if present and sending it to slave it to slave NodeMcus near to the person using our app through master NodeMcu

2. Module -2 (Letting Proximity sensors take the lead)
Th input is the data of fog presence sent by the master NodeMcu to the slave NodeMcus and the output is activating all the proximity sensors to estimate the traffic as there won't be a clear view through camera during heavy fog conditions and sending all the data to their respective slave NodeMcus

3. Module - 3 (Sending traffic data to cloud)
The input is the data about traffic sent to master NodeMcu through slave NodeMcus and the output is master NodeMcu sending the traffic data to cloud

4. Module - 4 (Getting Pressure Sensor into work)
The input is the data about fog sent by the master NodeMcu to the slave NodeMcus and output is NodeMcu telling Pressure Sensor to measure pressure

5. Module - 5 (Identifying obstacle)
The input is NodeMcu telling Pressure Sensor to measure pressure and if the measured pressure is above the value we set it sends output to it's NodeMcu telling there is an obstacle

6. Module - 6 (Positioning the obstacle)
The input is NodeMcu telling the Proximity Sensor to locate the obstacle near it and the output is the proximity sensor sending the data about the obstacles location back to it's NodeMcu

7. Module – 7 (Sending obstacle data to cloud)
The input is the data about obstacle sent to Master NodeMcu through Slave NodeMcu and the output is the Master NodeMcu sending data to cloud using MQTT

8. Module - 8 (Notifying the user)
The input is the data sent to cloud and the output is notifying user about traffic and obstacles in front of them

How are we going to do it?
(how we plan to do it and what our model is going to do.)

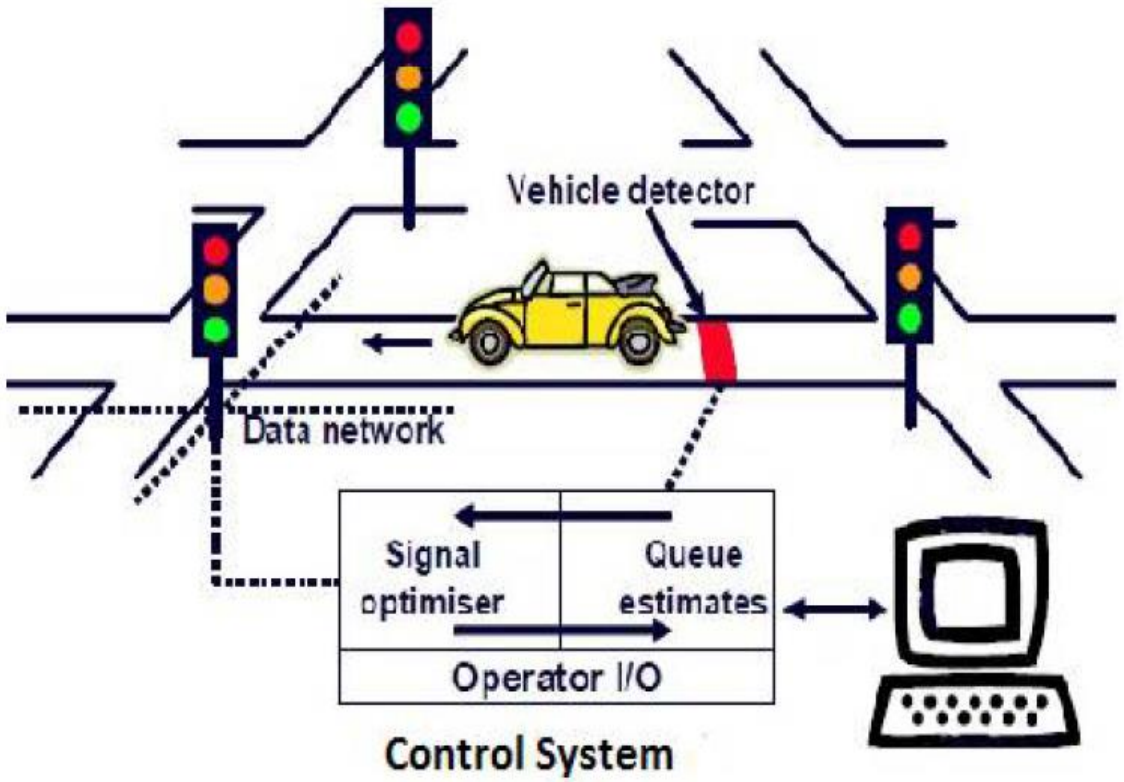
1) We’ll be using pre-trained models as well as train the model to detect vehicles to determine the traffic using OpenCV (Also to detect fog.)
2) As soon as the module detects traffic, the lights turn green and the traffic is minimised using priority queue algorithms and users planning to take the route will be alerted to take an alternate route

   **Our aim is minimise the traffic in the city as a whole.*
3) Using pressure and proximity sensors (which will be spread across the road) we’ll be able determine the presence of obstacles, we’ll also be able to create an accurate map of all the vehicles which will be backed up by the OpenCV…Therefore letting us build a module which intelligently determines the efficient way to clear the traffic

   Note: The pressure sensors in real life will set to detect only after a certain range, therefore avoiding unnecessary triggers of the sensors which can be caused n number of factors

   **Model is going to communicate with the sensors and the users using ESP8266 microcontroller in our project(since ours is just a prototype), a better wireless communication controller will be used in real time.*

4) In the presence of fog, the cameras will detect it and activate the proximity sensors to guide the users and ensure safe travel.
5) The trained model will be also be able to detect accidents and report it to authorities ensuring emergency services and traffic clearance int the route where the accident took place



ACCESS POINT
ANT
ITAP
WIRELESS SENSOR
D1
D2
D3
D4
Dx
3 m

*Pictures from the internet.