200280111084 Aryan Modi

Smart Traffic Monitoring and Management System

Electronics and Communication Engineering

Vodafone Idea Foundation

Index Terms

Cloud, MQTT, NodeMCU, Arduino, Traffic Control System, IoT, Thingspeak

I. Objective

The main objective of this project is to design and implement a system for real time traffic monitoring using Internet of things (IoT) and sensing technology. One part of the project was involved in implementing a real time traffic control system to control the traffic, while other part was involved in sending the data to the cloud whereit can be visualized as well as from cloud it can be made available to local user in real time through a mobile application.

II. Implemented Attributes

1) IR sensors to get vehicles count.
2) Sound sensor to detect and provide fast passage to emergency vehicles.
3) LCD display to display the current number of vehicles in each lane.
4) LED signals are used for road intersection. LEDs are controlled by Arduino UNO board.
5) Real time data is being collected in arduino, which is being sent to NodeMCU via serial communication.
6) From NodeMCU, collected data is being sent to Thingspeak IoT platform(cloud) via internet for storing and processing purposes. MQTT protocol is used for wireless network.
7) From cloud data can also be accessed using a Mobile App. Such that a local user can easily get information about current traffic status in that particular area.

III. Hardware Used

1) Arduino Mega 2560 board
2) NodeMCU - ESP8266
3) Infrared (IR) sensors
4) Sound sensors
5) LEDs

- RED
- GREEN (optional)

6) 16x2 LCD display

IV. Software Dependencies and Requirements

- Arduino IDE
- Thingspeak IoT platform.
- Any mobile app compatible to be used with Thingspeak server.

V. Description

In order to provide an efficient and centrally connected solution this traffic management system is designed. IoT is been used to make system efficient and connected. The primary element of this system is the wireless sensor nodes consisting of different IR and sound sensors. The sensors interact with the outside world and sends data to micro controller. On each lane there are two IR sensors, one for increasing the count when vehicles pass in front of it and another to decrements the count (which is placed near the junction), so that number of vehicles can be precisely known on each lane. A Sound Sensor on each lane is used to sense any emergency vehicle like ambulance, so that signal of that lane can be turn Green or (turn off the Red signal light) to provide fast passage to the ambulance. As soon as any emergency vehicleis detected on the road that particular lane's signal turns green or (turn off the Red signal light). All of the sensor's data is collected in Arduino,which in turn runs the required algorithm to control traffic signal, based on the data. The collected data is furthersent to NodeMCU via serial communication, from NodeMCU data is again sent to the Thingspeak IoT platformvia Internet for storing and processing purpose. Minimizing power requirements in IoT is one of the driving forces. Hence MQTT (Message Queuing Telemetry Transport) protocol is used to connect NodeMCU to the cloud. It is lightweight, power efficient and requires less bandwidth and also provide QoS to support message reliability.

Further a third-party mobile app is also used which is compatible with Thingspeak IoT platform and can be used to provide users with the current status of traffic label in any particular area.

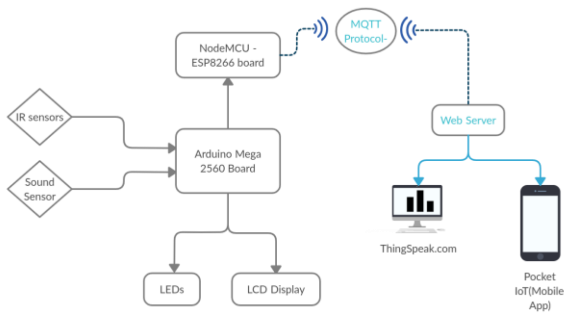


Fig. 1. Configuration Diagram

VI. Algorithm

This is the algorithm used to control traffic lights based on vehicle density on each lane: 8 IR sensors (I1, I2, I3, I4, I5, I6, I7, I8) are used in total, 2 on each lane, say:

- Number of vehicles are (V1, V2, V3, V4) respectively on each lane.
- On lane 1 (V1): I1(increments the count), I2(decrements the count)
- On lane 2 (V2): I3(increments the count), I4(decrements the count)
- On lane 3 (V3): I5(increments the count), I6(decrements the count)
- On lane 4 (V4): I7(increments the count), I8(decrements the count)

Algorithm:

1) Start
2) Get number of vehicles using IR sensors on each lane say (V1, V2, V3, V4)
3) Get input from sound sensor
4) if (emergencyVehicleOnRoad == 1), then turn off red signal for that lane say for lane L for time T.
5) else Compare (V1, V2, V3, V4), select the lane with highest number of vehicles, say L1, turn off red signal for that lane for time T(=15s) and red signal for other lanes.
6) Compare (V2, V3, V4), select the lane with highest number of vehicles, say L2 turn off red signal for that lane for time T and red signal for other lanes.
7) Compare (V3, V4), select the lane with highest number of vehicles, say L3 turn off red signal for that lane for time T and red signal for other lanes.
8) Turn off red signal on for last remaining lane.
9) Send data to server at the interval of say t (= 10s) seconds.
10) All time display data on LCD screen.
11) Repeat from step 2.

A. *Method to get the count using IR sensors*

If we try to use conventional method, that whenever an object comes in front of IR sensor, just start counting, this method gives wrong answer, as the time a vehicle spend in front of IR sensor is random. Hence to tackle this problem another method is used in which number of pulses are calculated. Whenever sensors state changes from 0 to 1 which generates a pulse, and the number of pulses generated gives the count of vehicles, and which will be independent of the time vehicle spends in front of the sensor.

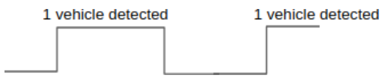
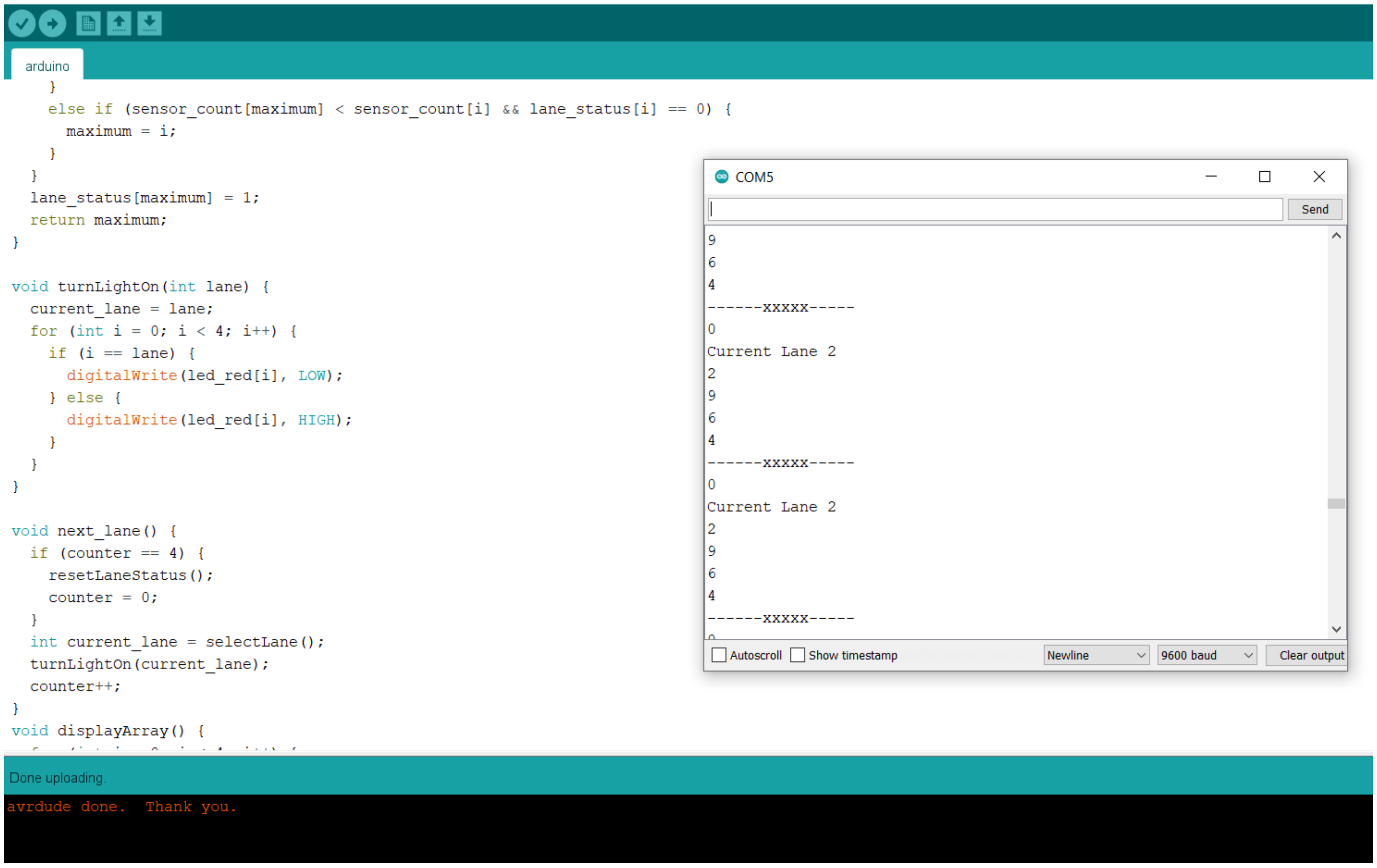


Fig. 2. IR Sensor Pulse Diagram

VII. Sample Output

Output in Serial Monitor, which shows data being received in arduino through sensors, being sent to NodeMCU and then to server.

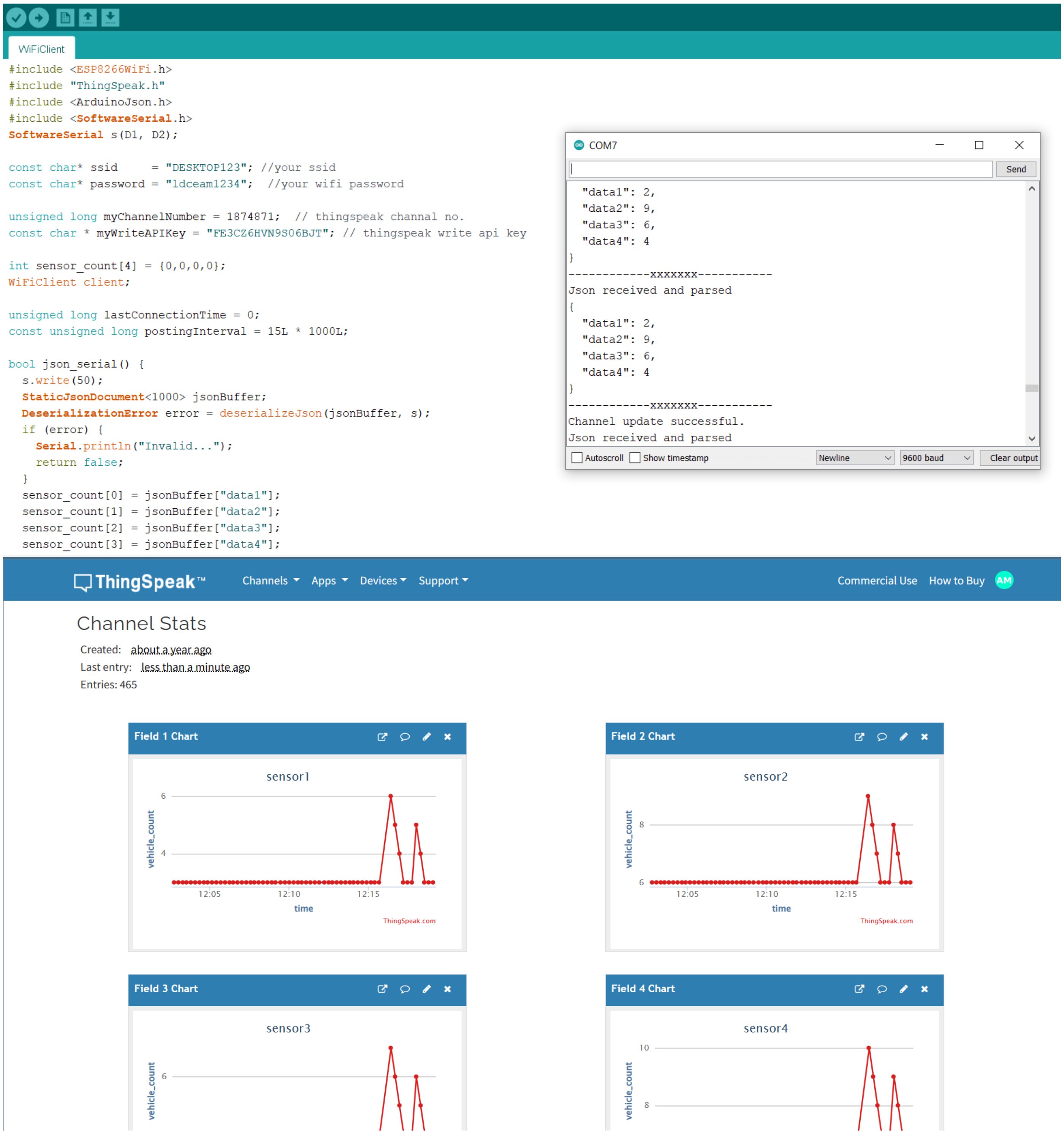


```
    }
    else if (sensor_count[maximum] < sensor_count[i] && lane_status[i] == 0) {
      maximum = i;
    }
  }
  lane_status[maximum] = 1;
  return maximum;
}

void turnLightOn(int lane) {
  current_lane = lane;
  for (int i = 0; i < 4; i++) {
    if (i == lane) {
      digitalWrite(led_red[i], LOW);
    } else {
      digitalWrite(led_red[i], HIGH);
    }
  }
}

void next_lane() {
  if (counter == 4) {
    resetLaneStatus();
    counter = 0;
  }
  int current_lane = selectLane();
  turnLightOn(current_lane);
  counter++;
}
void displayArray() {
```

```
9
6
4
------xxxxx-----
0
Current Lane 2
2
9
6
4
------xxxxx-----
0
Current Lane 2
2
9
6
4
------xxxxx-----
```

Newline 9600 baud

```
Done uploading.
avrdude done.  Thank you.
```



```
#include <ESP8266WiFi.h>
#include "ThingSpeak.h"
#include <ArduinoJson.h>
#include <SoftwareSerial.h>
SoftwareSerial s(D1, D2);

const char* ssid     = "DESKTOP123"; //your ssid
const char* password = "ldceam1234";  //your wifi password

unsigned long myChannelNumber = 1874871;  // thingspeak channal no.
const char * myWriteAPIKey = "FE3CZ6HVN9S06BJT"; // thingspeak write api key

int sensor_count[4] = {0,0,0,0};
WiFiClient client;

unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 15L * 1000L;

bool json_serial() {
  s.write(50);
  StaticJsonDocument<1000> jsonBuffer;
  DeserializationError error = deserializeJson(jsonBuffer, s);
  if (error) {
    Serial.println("Invalid...");
    return false;
  }
  sensor_count[0] = jsonBuffer["data1"];
  sensor_count[1] = jsonBuffer["data2"];
  sensor_count[2] = jsonBuffer["data3"];
  sensor_count[3] = jsonBuffer["data4"];
```

```
  "data1": 2,
  "data2": 9,
  "data3": 6,
  "data4": 4
}
------------xxxxxxx-----------
Json received and parsed
{
  "data1": 2,
  "data2": 9,
  "data3": 6,
  "data4": 4
}
------------xxxxxxx-----------
Channel update successful.
Json received and parsed
```

Channel Stats

Created: about a year ago
Last entry: less than a minute ago
Entries: 465

Fig. 3. Output in Serial Monitor: Left side is Arduino Serial Monitor, right side is NodeMCU serial Monitor.

VIII. User Manual

A. NodeMCU Setup and Code Description

- Pin D1 is Rx
- Pin D2 is Tx

Change following parameters in code before uploading it to your device:

- ssid: ssid of the Wi-Fi network
- password: password of your Wi-Fi network
- mqttUserName[]: your username
- mqttPass[]: thingspeak channel pass key
- writeAPIKey[]: thingspeak channel write key

B. Arduino UNO Setup and Code Description

Pin Configuration

- Pin 0 is Rx
- Pin 1 is Tx
- For serial communication connect Pin 10, 11 of arduino to Pin D2, D1 of NodeMCU respectively.
- Pin (10, 11, 12, 13) are Red LEDs pin for lane 1, 2, 3, 4 respectively.
- Optional if you want better user experience than you can use the Green LEDs pin for lane 1, 2, 3, 4 respectively.
- Pin 14 is sound sensor pin for lane 4.
- Pin (2, 3, 4, 5) are IR sensor incrementor pin for lane 1, 2, 3, 4 respectively.
- Pin (6, 7, 8, 9) are IR sensor decrementor pin for lane 1, 2, 3, 4 respectively.
- LCD pins are connected to SDA(A4) and SCL(A5)

Code Description

- TIME: this it the time interval for which green light will remain on for any particular lane.
- UPDATE_SERVER_TIME: this is the time interval at which Arduino sends data to NodeMCU via serial communication.
- json_serial() : this function is responsible for sending serial data.
- read_sensor() : this function is responsible for taking input from sensors.
- selectLane() : this is function responsible for controlling LEDs based on density based algorithm.
- displayArray() : this function is responsible for displaying data on LCD screen connected to Arduino Board.

C. Thingspeak IoT platform setup

1) Create new channel on Thingspeak.
2) Channel should contain four field to take data of each lane.
3) Update channel configurations in NodeMCU code to connect it.

D. Mobile App setup

1) Download Pocket IoT app from playstore.
2) Fill channel Key to see current data.