SMART TRAFFIC LIGHT CONTROLLER

May 9, 2019

Supervisor : Prof. Tamal Pal
Titash Kumar Das (510515043)
Antara Hembarm (510515031)
Sushovan Saha (510515040)
Akash Deep (510515038)
Bhavya Agarwal (510515035)
Department of Computer Science and Technology

CERTIFICATE

This is to certify that this project entitled .... by ....
....
...., submitted in fulfillment of the requirements for the degree of Bachelor of Technology in COMPUTER SCIENCE AND TECHNOLOGY of the IIEST, Shibpur during the academic year 2018-19, is a bonafide record of work carried out in a satisfactory way.

Project Guide:..................................
Date:....................
Place:.................

ACKNOWLEDGEMENT

Working on this project on "SMART TRAFFIC LIGHT CONTROLLER"; was a source of immense knowledge to us. We would like to express our sincere gratitude towards Prof. Tamal Pal for her guidance and valuable support throughout the course of this project work. We acknowledge with a deep sense of gratitude, the encouragement and inspiration received from our faculty members and colleagues. We would also like to thank our parents and friends for their love and support.

Contents

INTRODUCTION

A traffic light is a signaling device positioned at a road intersection, pedestrian crossing, or other location in order to indicate when it is safe to drive, ride, or walk using a universal color code. the traffic lights for vehicles commonly have three main lights, a red light that means stop,a green light that mean go and yellow that means ready to stop. However for the pedestrians, there have only two lights, a red light and a green light that mean go and stop respectively. The traffic lights have given many benefits to all road users. Besides reducing the number of accidents, it made the traffic flow smoothly and possibly could save people time.

OBJECTIVE

Current Traffic Light System

The traffic signal timers have a fixed time period to switch traffic between different directions. Due to this, the vehicles have to wait for a long time span even if the traffic density is very less. There are many issues with this fixed time period system.

Traffic light cause the heavy traffic jams

Increasing the number of vehicle in road, have cause the heavy traffic jams. This is usually happened at the main junctions commonly at the morning, before office hours and at the evening, after the office hours. The main effect of this matter is increasing time wasting of the people at the road.

No traffic, but the road user still need to wait

The traffic light has contributed more wasting time people at road. At the certain junction, sometime there is no traffic. But because the traffic light is still red, the road users should wait until the light turns to green.

Emergency car stuck in traffic jam

Usually, during traffic jam, the emergency vehicle, such as ambulance, fire brigade and police will be stuck especially at the traffic light junction. This is because the road users waiting for the traffic light turn to green. This is very critical problem because it can prevent the emergency case become complicated and involving life.

If the traffic signal timer can be programmed to be manipulated with the continuously varying traffic density, the problem of traffic congestion can be reduced to a significantly lower levels. Our objective is to develop a traffic signal controller, which can be sensitive to changing traffic management policy, and aim to achieve a wide range of transport and environmental objectives in addition to minimize the vehicle delays and stops.

EXISTING WORK

Round Robin Algorithm

Every lane is allotted a fixed green light cycle. Thus, it allows vehicles of each lane equal chance to cross the intersection junction.

Disadvantage : Since, the time slack is fixed, it's just the same algorithm which is used in normal traffic. Also it doesn't consider any case of emergency vehicle.

Emergency Vehicle Priority Algorithm

Whenever an emergency vehicle enters a lane, the lane is prioritised first to cross the intersection.

Disadvantage : The lanes with no emergency vehicle may suffer from starvation i.e. the vehicles in these lanes may not get chance to cross the intersection point for a long duration.

PLAN OF WORK

Density based approach has been taken for this project. The project is based on IOT, so we have divided the work in layers:

- **Perception Layer:** Image Capturing using arduino has been done in this stage.
- **Networking Layer:** Send the image to a Server Computer using wifi module
- **Processing Layer:**
  - Detection of number of cars in each lane from the video.
  - Store the data
  - Calculate the time to be assigned for each lane according to the number of cars present in the lanes.
- **Application Layer:** Give the red and green signal according to the calculated scheduling data.

IMPLEMENTATION

To implement the plan we have used both software and hardware components. The video capturing and the display of the output (showing the traffic light signal) are done through Arduino-uno. The video processing part and signal-time calculation is done on the server (computer). The whole process is done using the following steps.

Perception Layer

In this layer Image Capturing is done using OV7670 Camera Module. OV7670 can capture picture and send that using Arduino-Uno. **Hardware Components :**

- Arduino Uno
- OV7670 camera module
- Resistor 4.75K ohm
- Resistor 7K ohm
- Breadboard

Software Components :

- Arduino Ide
- JDK

Arduino Introdution

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

Arduino Uno Board

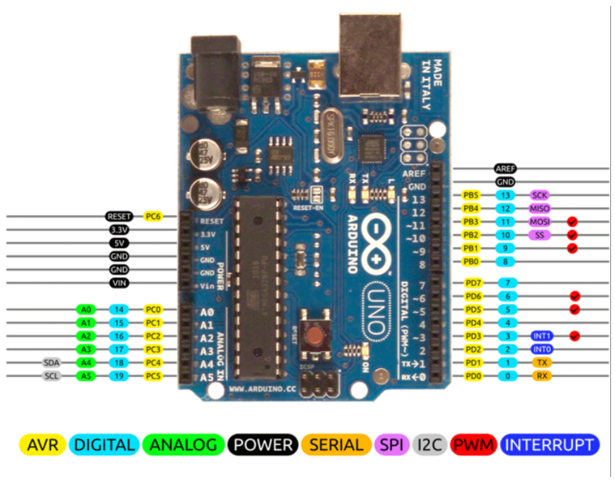
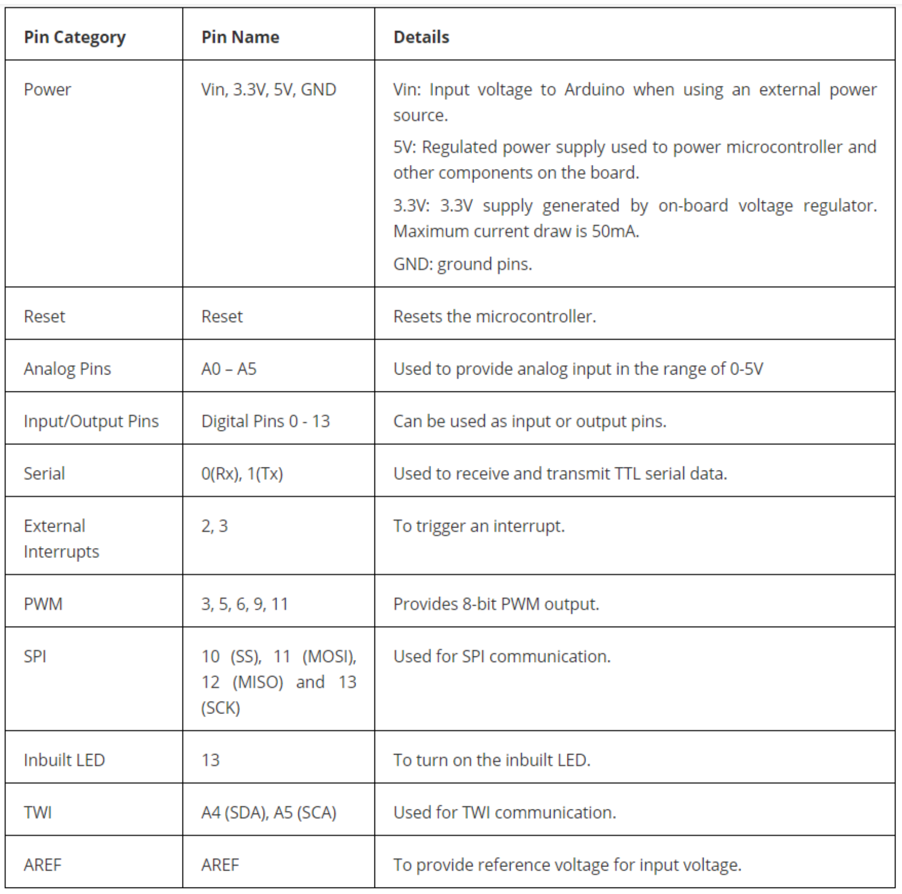


Figure 1: Arduino Uno Board

The Arduino Uno board is a microcontroller based on the ATmega328. It has 14 digital input/output pins in which 6 can be used as PWM outputs, a 16 MHz ceramic resonator, an ICSP header, a USB connection, 6 analog inputs, a power jack and a reset button. This contains all the required support needed for microcontroller. In order to get started, they are simply connected to a computer with a USB cable or with a AC-to-DC adapter or battery. It is featured by the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Pin Diagram of Arduino Uno Board



Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Figure 2: Pin Diagram of Arduino Uno Board

Arduino Software

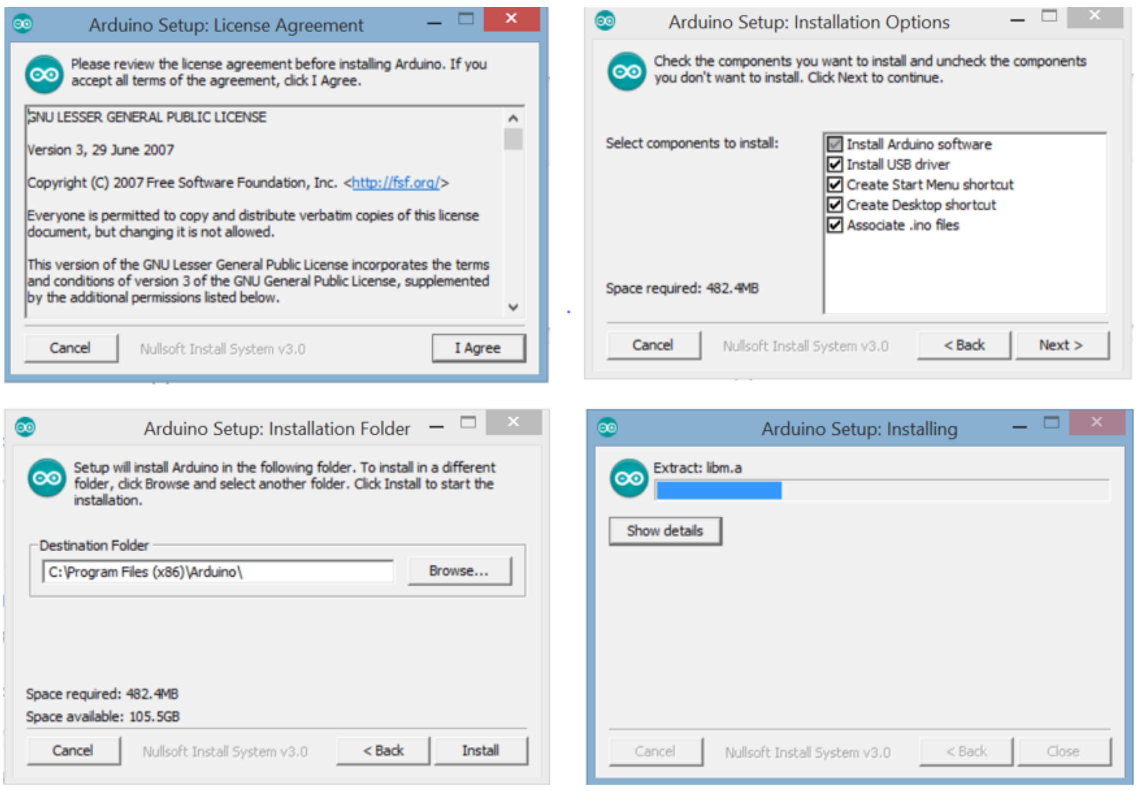
The Arduino Software (IDE) allows us to write programs and upload them to our board. There are two options:

1. If we have a reliable Internet connection, we should use the online editor (Arduino Web Editor). It will allow us to save our sketches in the cloud, having them available from any device and backed up. We will always have the most up- to-date version of the IDE without the need to install updates or community generated libraries.

2. If you would rather work offline, you should use the latest version of the desktop ide(Arduino Uno).

Installation of Arduino software (Desktop IDE) :

- Go to the official site - https://www.arduino.cc/en/Main/Software
- Click on `Windows Installer, for Windows XP and up`
- Click on `Just Download`
- Arduino Software will be downloaded.
- Click on the Arduino Software and Install it.


Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 482.4MB

Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 482.4MB
Space available: 105.5GB

Arduino Setup: Installing

Extract: libm.a

Figure 3: Installation Stages of Arduino

OV7670 Camera Module

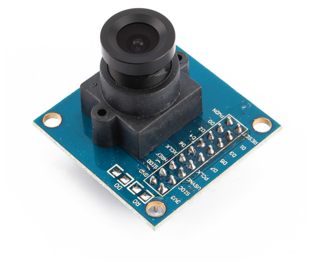


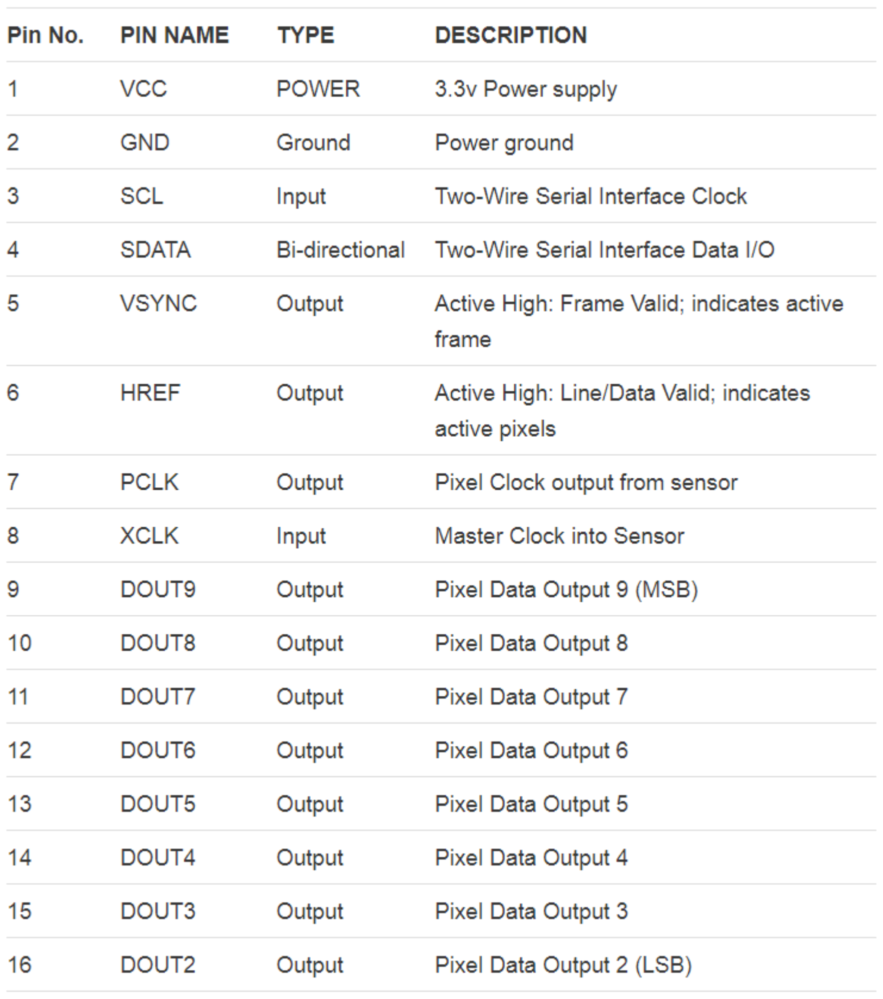
Figure 4: OV7670 Camera Module

The OV7670 image sensor is a small size, low voltage, single- chip VGA camera and CMOS image processor for all functions. It provides full-frame, sub-sampled or windowed 8-bit images in various formats, controlled through the Serial Camera Control Bus (SCCB) interface. The camera module is powered from a single +3.3V power supply, and external clock source for camera module XCLK pin. The OV7670 camera module built-in onboard LDO regulator only requires single 3.3V power and can be used inArduino STM32,Chipkit , FPGA and etc.

Specification of OV7670 Module

- Optical size 1/6 inch, Resolution 640×480 VGA
- Onboard regulator, only single 3.3V supply needed
- Mounted with high quality F1.8 / 6mm lens
- High sensitivity for low-light operation
- Automatic image control functions.
- ISP includes noise reduction and defect correction
- Supports LED and flash strobe mode ,Flicker (50/60 Hz) auto detection
- Saturation level (UV), Edge enhancement level auto adjust

Pin diagram of OV7670 Camera module



Pin No.	PIN NAME	TYPE	DESCRIPTION
1	VCC	POWER	3.3v Power supply
2	GND	Ground	Power ground
3	SCL	Input	Two-Wire Serial Interface Clock
4	SDATA	Bi-directional	Two-Wire Serial Interface Data I/O
5	VSYNC	Output	Active High: Frame Valid; indicates active frame
6	HREF	Output	Active High: Line/Data Valid; indicates active pixels
7	PCLK	Output	Pixel Clock output from sensor
8	XCLK	Input	Master Clock into Sensor
9	DOUT9	Output	Pixel Data Output 9 (MSB)
10	DOUT8	Output	Pixel Data Output 8
11	DOUT7	Output	Pixel Data Output 7
12	DOUT6	Output	Pixel Data Output 6
13	DOUT5	Output	Pixel Data Output 5
14	DOUT4	Output	Pixel Data Output 4
15	DOUT3	Output	Pixel Data Output 3
16	DOUT2	Output	Pixel Data Output 2 (LSB)

Figure 5: Pin diagram of OV7670 Camera module

Connection between OV7670 and Arduino-Uno Board

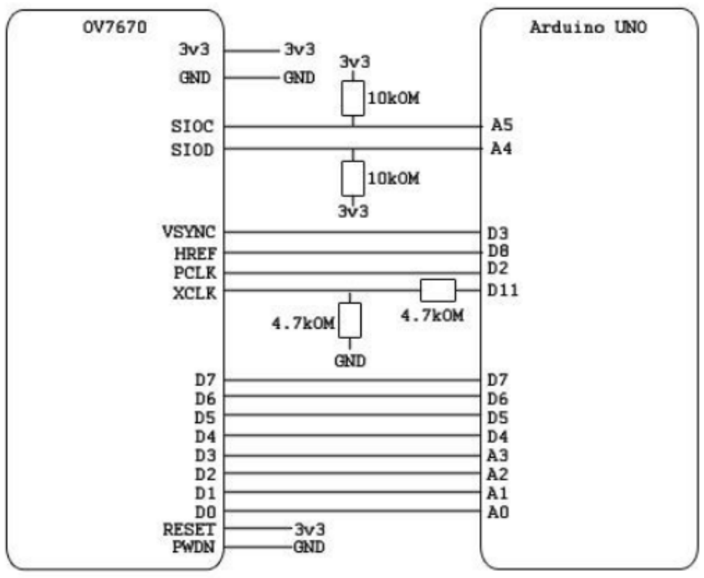


Figure 6: Connection between Arduino and OV7670

After making this connection we have run 2 programs:

- **Arduino Program:** This program helps to capture the image and send it to Arduino-Uno. Arduino-Uno board sends the image to the local machine using the serial connection and the image is stored in the machine.
- **Java Program:** This program reads the image sent by arduino and converts it to bit-map generating a black white image and stores it in a directory. This picture can be used for proccesing.

Network Layer

Hardware Components :

- Arduino Uno
- HC-12 wireless communication module
- Breadboard
- Jumping wires

Software Components :

- Arduino IDE

HC-12 wireless communication module :

Here are some specification of HC-12 wireless serial port communication module :

- Its wireless working frequency band is from 433.4 MHz to 473.0 MHz
- It has a total of 100 channels with a stepping of 400 KHz between each channel
- Transmitting power is from -1dBm (0.79mW) to 20dBm (100mW)
- Receiving sensitivity is from -117dBm (0.019pW) to -100dBm (10pW).

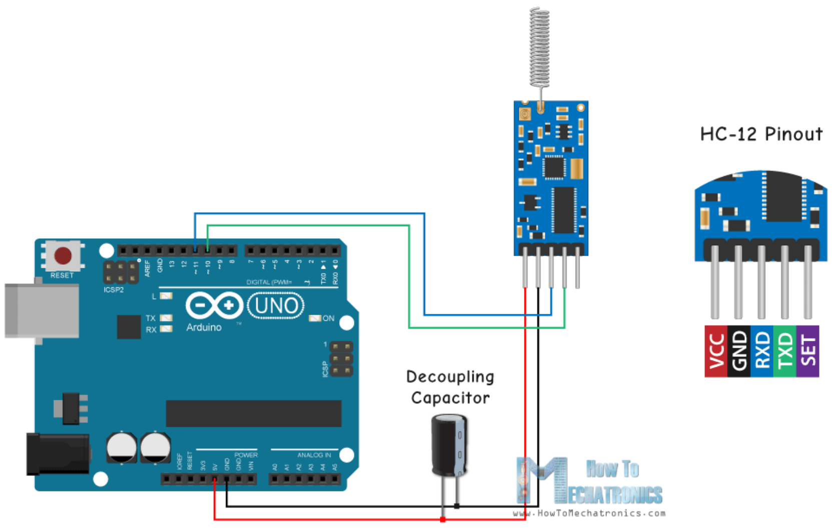
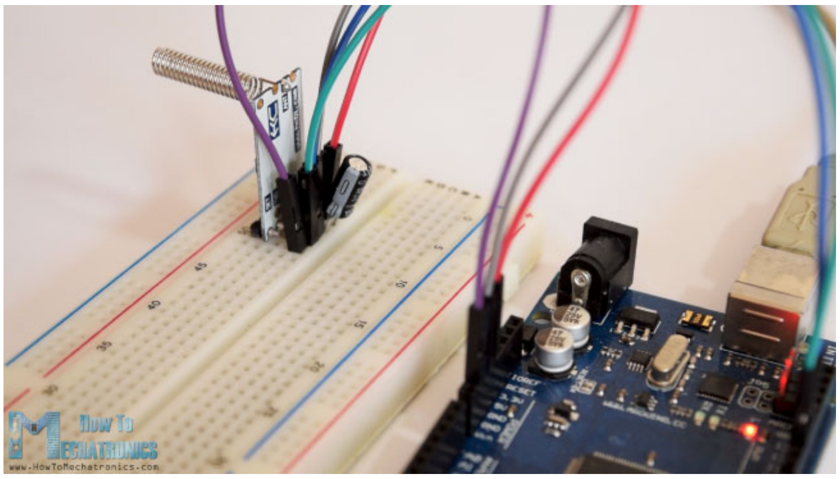


Figure 7: Circuit Diagram

How the code works:

So once we click the Send button, at the first Arduino, the while loop with the Serial.available() function will become true and using the HC12.write() function we will send the image data from the serial monitor to the HC-12 module. This module will transfer the data wirelessly to the second HC-12 module, so at the second Arduino the while loop with the HC12.available() function will become true and using the Serial.write() function the data will be sent to the serial monitor.



Processing Layer

Video Processing:

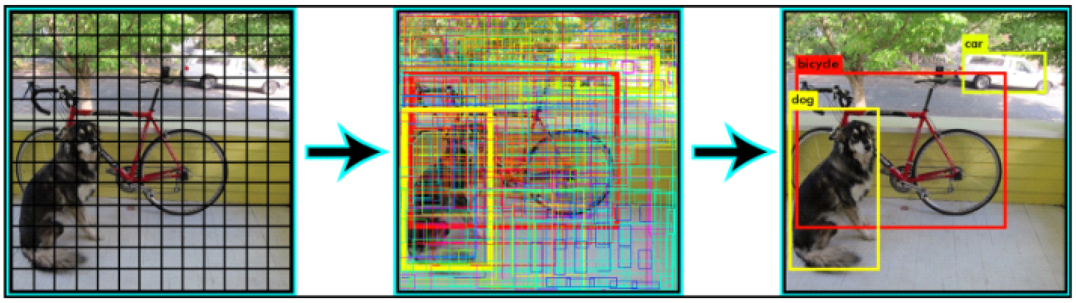
We are using YOLO detection system to detect vehicles from videos captured by Arduino camera .Let us take a brief look at YOLO and its advantages.

What is YOLO?

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a map (mean average precision) of 57.9 percentage on COCO test-dev.

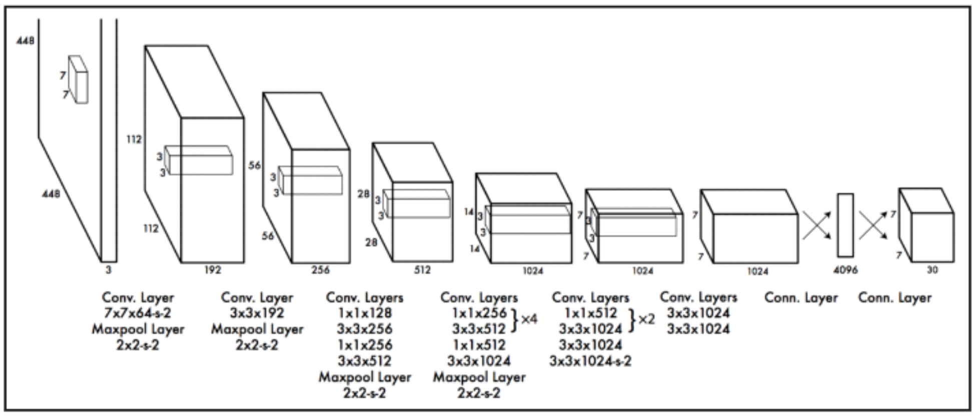
How YOLO works?

The major concept of YOLO is to build a CNN network to predict a (7, 7, 30) tensor. It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location. YOLO performs a linear regression using two fully connected layers to make 7×7×2 boundary box predictions (middle picture). To make a final prediction, we keep those with high box confidence scores (greater than 0.25) as our final predictions (right picture).



The class confidence score for each prediction box is computed as:
class confidence score = box confidence score x conditional class probability
It measures the confidence on both the classification and the localization (where an object is located). We may mix up those scoring and probability terms easily. Here are the mathematical definitions for your future reference.

Network Design :
YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolu-


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

tion layers use 1 × 1 reduction layers alternatively to reduce the depth of the features maps. For the last convolution layer, it outputs a tensor with shape (7, 7, 1024). The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs 7×7×30 parameters and then reshapes to (7, 7, 30), i.e. 2 boundary box predictions per location. A faster but less accurate version of YOLO, called Fast YOLO, uses only 9 convolutional layers with shallower feature maps.

Single Shot Detector (SSD) :

SSD attains a better balance between swiftness and precision. SSD runs a convolutional network on input image only one time and computes a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to foresee the bounding boxes and categorization probability.

SSD also uses anchor boxes at a variety of aspect ratio comparable to Faster-RCNN and learns the off-set to a certain extent than learning the box. In order to hold the scale, SSD predicts bounding boxes after multiple convolutional layers. Since every convolutional layer functions at a diverse scale, it is able to detect objects of a mixture of scales.

YOLO Vs SSD :

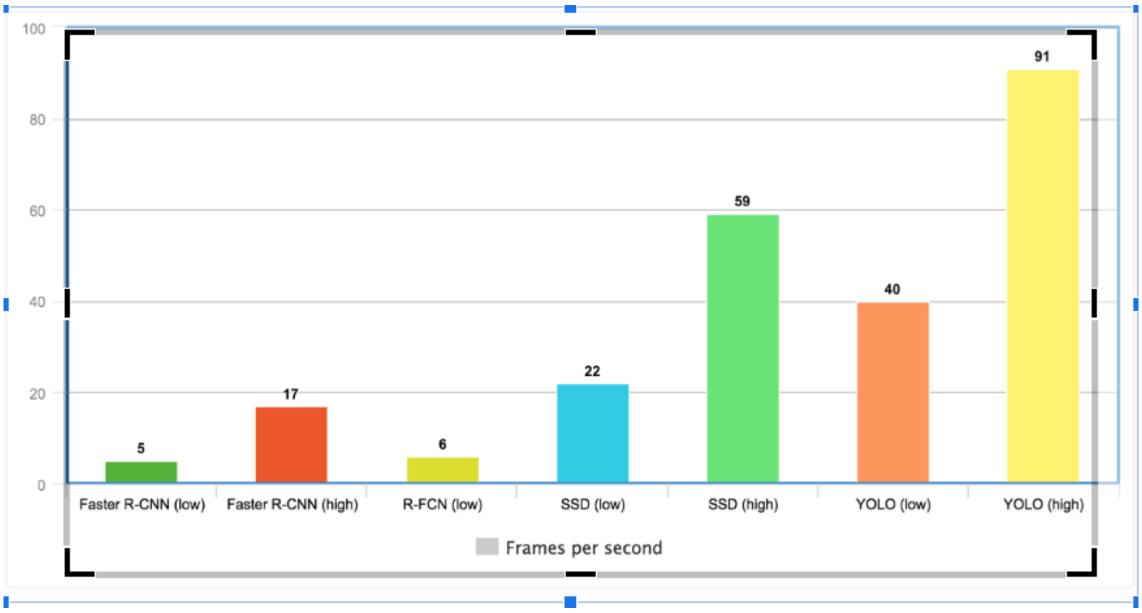
- SSD is a healthier recommendation. However, if exactness is not too much of disquiet but you want to go super quick, YOLO will be the best way to move forward. First of all, a visual thoughtfulness of swiftness vs precision trade-off would differentiate them well.

  SSD is a better option as we are able to run it on a video and the exactness trade-off is very modest. While dealing with large sizes, SSD seems to perform well, but when we look at the accurateness numbers when the object size is small, the performance dips a bit.

- The trade-off between speed and accuracy is accompanied with computational power available. The YOLO model is suitable for high-speed outputs, where accuracy is not that high... whereas SSDs provide higher accuracies with high-speed outputs with a higher computation time.

  Hence choose SSDs on good microprocessors, else YOLO is the goto for microprocessor-based computations.

- SSDs: mAP: 0.83 Time /epoch: 12 minutes
  YOLOs: mAP: 0.85 Time/epoch: 11 minutes



FUTURE WORK :

As the Traffic system is very much time sensitive and so in order to make our algorithm more efficient we can not afford high computational cost in terms of time to count the number of cars. So we have taken an image frame before a unit of time then counted the no. of vehicles in the image frame instead of constant monitoring on the lane. As our Algorithm takes only the number of vehicle present in the lane before a unit of time of making decision so, this approach of processing a single frame reduces the computational cost by a large margin than processing a video instead. Again, we do not need very powerful machines in terms of CPU and GPU to process a single frame of image.

But if we have powerful resources(powerful CPUs and GPUs) installed at server side, then we can afford constant monitoring on the lanes by processing larger number of frames per unit time in order to get better result and make the algorithm more efficient.

Installation:

- To implement YOLO - Go to https://pjreddie.com/darknet/yolo/
- Copy these commands and run in terminal sequentially-
  ```
  git clone https://github.com/pjreddie/darknet
  cd darknet
  Make
  ```

- We already have the config file for YOLO in the cfg/ subdirectory. we have to download the pre-trained weight file.
  wget https://pjreddie.com/media/files/yolov3.weights

For counting cars in a single Frame:

- Go to https://github.com/guillelopez/python-traffic-counter-with-yolo-and-sort and clone it.
- Go to Specified Folder named (TRAFFIC_COUNTER) and there in the inputs folder put the image for detection.
- Now put the pre-trained weight file (yolov3.weights) in the yolo-coco folder
- Run the command-» python mod.py –input input/car.bmp –output output/cars.bmp –yolo yolo-coco
- Now we get the classified objects in the image with no. of counts of the car

Photo Taken from Alipore Foot Bridge:

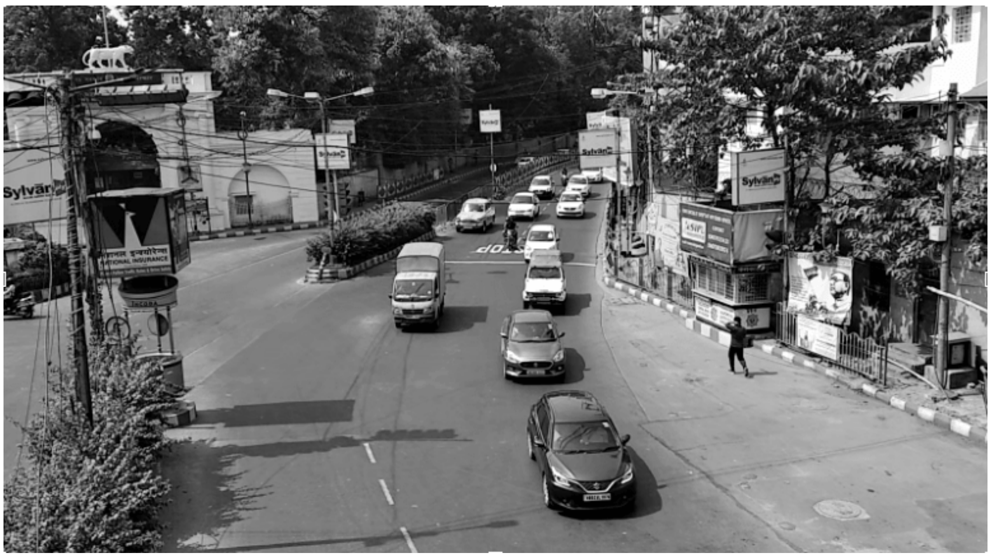


Figure 8: Image with cars

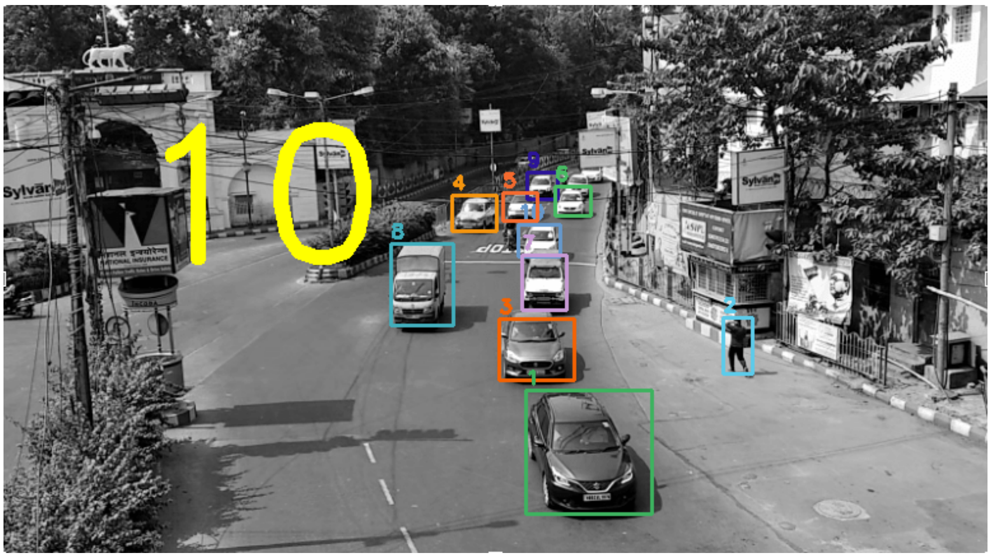
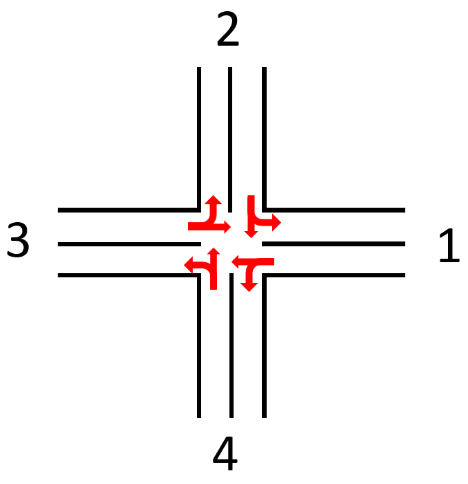
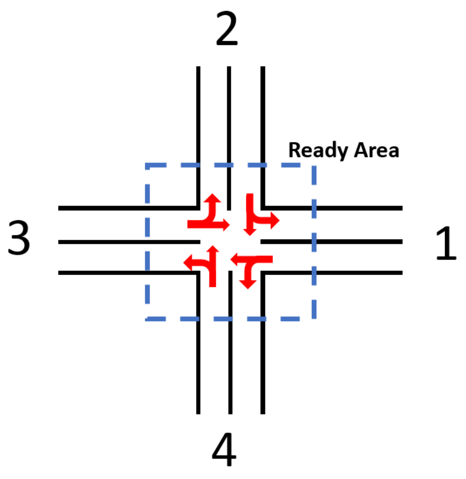


Figure 9: Detection of cars

Scheduling Algorithm



Here, we introduce a trafc light scheduling algorithm that considers the real-time trafc characteristics of the surrounding road segments at each signalized road intersection. In our work, we consider the typical four-leg road intersection. In the proposed algorithm, the trafc intersection is seen as a shared processor among four ows of trafc. Vehicles arrive at the road intersection at different estimated times, so each ow of trafc can be seen as a set of successive processes. Each process contains one or more vehicles that travel through the road intersection during the green phase of the trafc light. We determine the size of each process based on the number of vehicles located in the ready area during the data gathering phase (i.e., trafc density of each trafc ow).



Ready Area :

The ready area is a virtual dened area around each road intersection.The boundaries of each ready area are set based on the maximum allowable green time of the trafc light for each trafc ow. The ready area guarantee a fair share of the road intersection among the competing trafc ows. The ready area aims also to split the successive processes on each ow of trafc. The period of time (T) required by all vehicles in any process to pass the trafc light, is computed by using the below Equation

$T = a + n/stf$

In this equation a is a constant that accounts for the startup delay of the very rst vehicle in each set of vehicles, gathered in one process, the best value of a should be set empirically; and n is the no of vehichle in the respective lane. stf is the trafc speed of each trafc ow in terms of no of vehichles per sec, which crosses the lane.

Algorithm :

```
#initialization
 initialize tmax=90 sec
 initialize a =0.05 sec
 initialize priority_queue q <- pair(null,null)

 //stf array consists of each lane traffic flow speed provided
from statistical annalysis
```

```
#algo

for each lane i 1 to 4:
        c= count of no of vehichles in i'th lane within ready
           area length
        q.put(pair(c,i))

while !q.empty():
        pair p= q.pop()
        index i= p.second // max lane index
        if(!visit[i]):
                index j // complimentary lane of i
                visit[i]=1
                visit[j]=1
                n1= no of car in lane i
                n2= no of car in lane j
                t=timecalc(i,j,n1,n2)
                tassign=schedule(t,tmax) // assigned time
                                            for those lane



timecalc(i,j,n1,n2):
        t1= a + n1/stf[i] // time to be assign for lane i
        t2= a + n2/stf[j] // time to be assign for lane j
        if(t1>t2):
                return t1
        return t2  // return max of t1 and t2

schedule(t,tmax):
        if(t<tmax):
                return t
        return tmax
```

Application Layer

Signaling (LED)

We are taking LEDs as traffic light signal. A hardware circuits arrangement has been made for lighting up the LEDs. The circuit is given below.

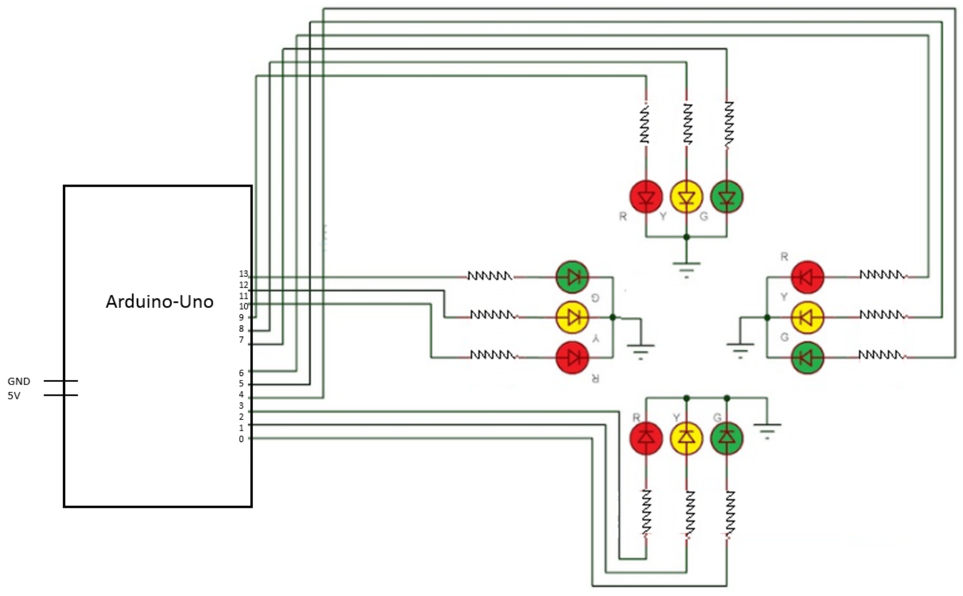


Figure 10

Output

Performance Evaluation Parameter:

- **Queueing delay**

  In our experiments, we compare the delay per vehicles, caused by the trafc light queuing delay. We then compare the total delay required for all detected vehicles inside the ready area at any certain time to cross the road intersection.

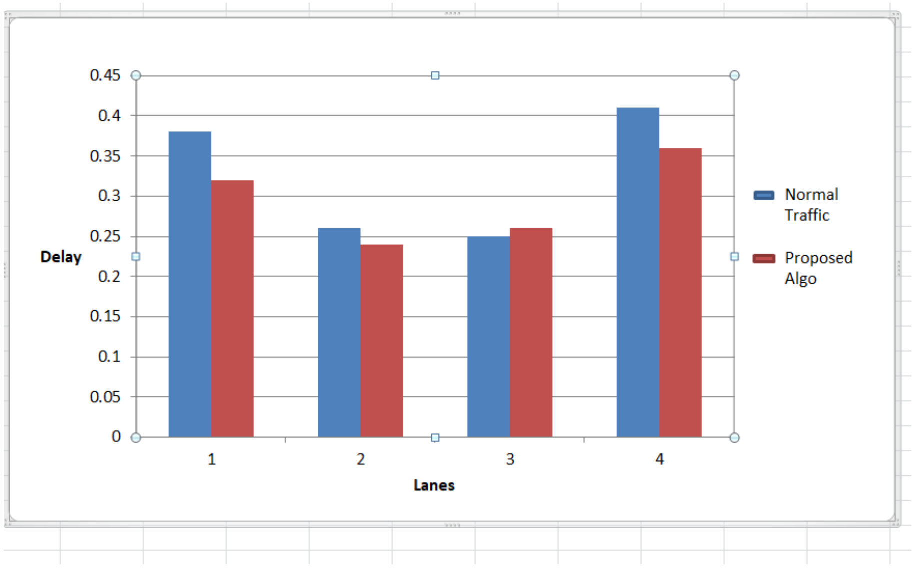


Figure 11: Peak Hours

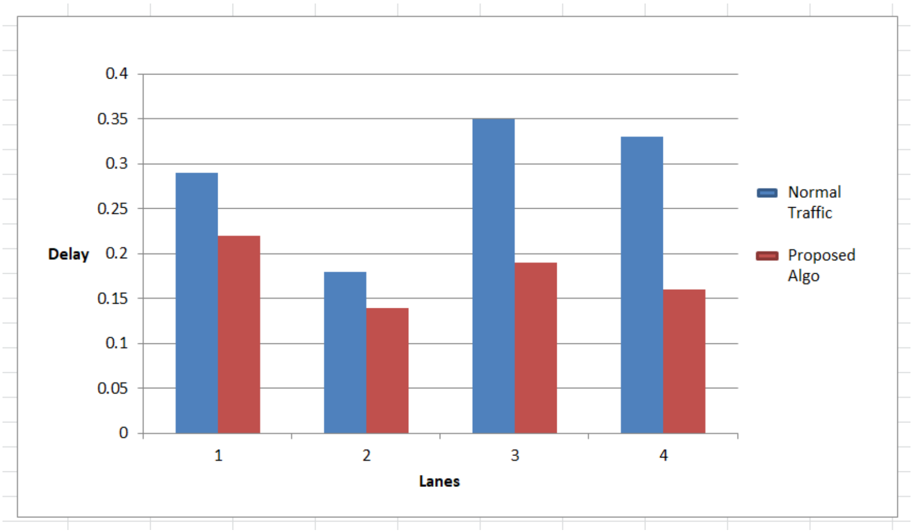


Figure 12: Non Peak Hours

We have measured the total delay that is required by all detected vehicles, at a certain point of time to cross an intersection. In general the above mentioned algorithm decreases the queuing delay of traveling vehicles by 25 percent on average compared to normal traffic scheduling algorithm for all trafc density scenarios.

- **Throughput**
  At the same time, our proposed algo increases the throughput of each road intersection by 30

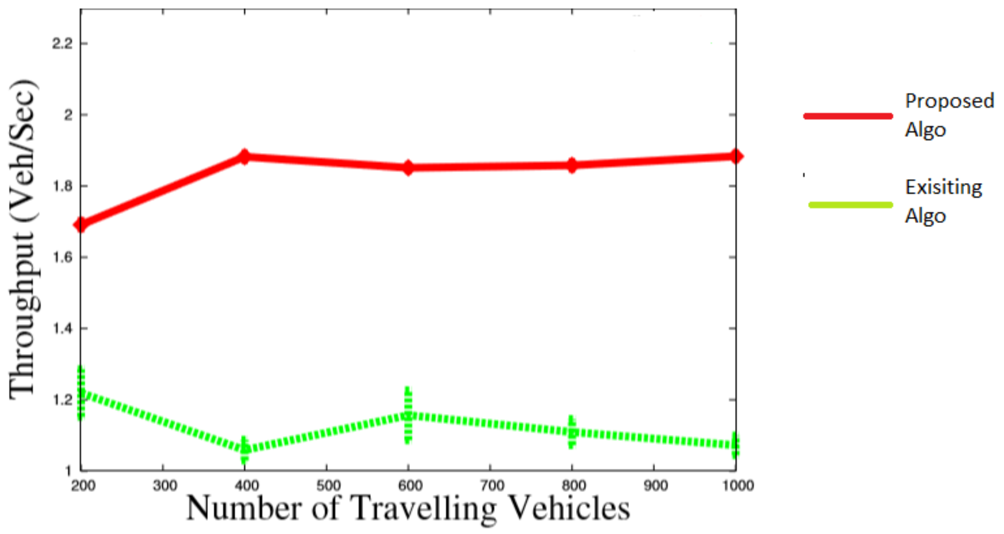


Figure 13: Throughput vs No of Vehichle

REFERENCES

- http://dspace.unimap.edu.my/dspace/bitstream/123456789/2857/6/Introduction.pdf
- https://github.com/ahmetozlu/tensorflow _object_counting_api
- https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- https://components101.com/microcontrollers/arduino-uno
- https://create.arduino.cc/projecthub/techmirtz/visual-capturing-with-ov7670-on-arduino-069ebb
- https://www.arduino.cc/en/Guide/Windows
- https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/