

**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**  
**University M'Hamed BOUGARA – Boumerdès**



**Institute of Electrical and Electronic Engineering**  
**Department of Electronics**

Project Report Presented in Partial Fulfilment of  
the Requirements of the Degree of

**‘MASTER’**

**In Electronics**

Option: **Computer Engineering**

Title:

**Design & Implementation of  
Smart traffic and streetlight  
systems**

Presented by:

- **TALBI Fatima Zahra Hamida**

Supervisor:

- **Mr. Mohamed Nadjib HAMADACHE**

Registration Number ...../2019

*To my family, and friends...*

## Acknowledgements

*First of all, I would like to praise almighty ALLAH for giving me the power to realize this project.*

*I would like to express my profoundest and sincerest respect and gratitude to my supervisor Mr. HAMADACHE for his support, guidance and patience during the realization of this project.*

*Many thanks are given to all the professors and workers at the IGEE for guiding me during my cursus.*

*I Would like to express my gratitude to the scientific club WAMEEDH. Special Thanks to Nazim, Abdenour, Sylia Imane, and Bouchra for their support, their encouragement, and their help to achieve this project, preparing and filming the final test.*

*At last, and not least, I would like to thank all my family's members and friends; they have been a source of inspiration and encouragement for years.*

*No acknowledgement would be completed without expressing my appreciation and thankfulness for the maintenance laboratory workers for their support and help.*

## **Abstract**

Smart city is seen to be one of the most dominant aspects in the last few years, and one special application of the internet of things. Smart traffic light and streetlight systems are major components of smart city infrastructure and are of an important use nowadays to ease the way of our life.

The first part is about building a smart traffic light system that has been implemented by using OpenCV software and PYTHON programming language that aims to have smart way for signal management which will ultimately be a cost effective solution. The system includes a camera placed facing a lane that will capture images of the road on which we want to control traffic. These images are efficiently processed to know the traffic density. According to the processed data, a microcontroller will send command to the LED to manage the traffic.

The second part is about implementing a smart streetlight system; it is the one which automatically switches ON and OFF the lights according to the situation. It automatically senses the movements of the object within a particular limit, The smart street light controller installed on the street light pole will control LED street lighting depending on movements of the object in the street, The control system will switch ON and OFF the street lights at needed timings and can also vary the intensity of the streetlight according to the necessity.

# Table of contents

Acknowledgements.....	III
Abstract.....	IV
Table of contents .....	V
List of figures .....	VII
List of tables .....	VIII
Nomenclatures.....	IX
Introduction.....	1
Chapter 1: Introduction to the Smart Traffic .....	2
and Streetlight Systems.....	2
1.1 TheProblem .....	2
1.2 The Solution .....	2
1.3 The Global SystemDiagram .....	2
1.4 Smart street light system .....	3
1.5 Smart traffic light system .....	4
1.6 Summary .....	4
Chapter 2: The Hardware Design and Implementation.....	5
Introduction .....	5
2.1 The ComponentsList .....	5
2.1.1 Raspberry pi 3 .....	5
2.1.2 The Arduino UNO .....	5
2.1.3 The camera.....	6
2.1.4 Infra Red (IR) sensors.....	6
2.1.5 Light dependent resistor (LDR).....	7
2.1.6 The light emitting diodes (LEDs).....	7
2.1.7 Resistors.....	7
2.2 The streetlight interfacing circuit .....	8
2.3 The traffic light interfacing circuit .....	9
2.4 The system power consumption .....	10
2.5 Summary.....	10
Chapter 3: The streetlight system .....	11
Introduction .....	11
3.1 Designing Methodology .....	11
3.2 The software codes .....	12
3.3 Results and discussions.....	16
3.4 Summary.....	17

Chapter 4: The traffic light system .....	18
Introduction .....	18
4.1 Image processing .....	18
4.1.1 Transform frame into NumPy array .....	18
4.1.2 Changing colorspace .....	18
4.1.3 Threshold the HSV colorspace .....	19
4.1.4 Blurring .....	19
4.1.9 Morphological operations .....	20
4.2 Designing Methodology .....	21
4.2.1 Raspberry pi 3 and USB camera interfacing .....	21
4.2.2 Building the environment .....	21
4.2.3 Traffic light assembly .....	21
4.2.4 Wiring .....	21
4.3 Software code .....	22
4.3.1 Python .....	22
4.3.2 OpenCV .....	22
4.3.3 Flowchart of the program .....	23
4.4 Results and discussion .....	24
4.4.1 Setup of the raspberry pi 3 .....	24
4.4.2 Installation of openCV libraries .....	24
4.5 Summary .....	27
Conclusion .....	28
References .....	29
Appendix A: Raspberry pi 3 .....	30
Appendix B: Arduino Uno Board and ATmega328 Microcontroller .....	32

# List of figures

## Chapter 1

Figure 1.1: Smart traffic light System Diagram.....	3
Figure 1.2: Street light System Diagram .....	3

## Chapter 2

Figure 2.1: Raspberry pi 3 .....	5
Figure 2.2: Arduino UNO.....	6
Figure 2.3: JEWAY webcam .....	6
Figure 2.4: IR sensor .....	6
Figure 2.5: working IR .....	6
Figure 2.6: Light Dependent Resistor .....	7
Figure 2.7: Light emitting diode .....	7
Figure 2.8: Working principle of resistors.....	7
Figure 2.9: Resistors.....	7
Figure 2.10: Street light circuit .....	8
Figure 2.11: Traffic light circuit.....	9

## Chapter 3

Figure 3.1: The architecture design of smart street light control system .....	11
Figure 3.2: Initialization of variables and setup of Arduino pins .....	12
Figure3.3: The LDR loop .....	13
Figure 3.4: The code of the two first IR sensors.....	14
Figure3.5: The code of the two last IR sensors.....	15
Figure3.6: The output of LEDs during the daytime .....	16

## Chapter 4

Figure 4.1: Traffic light assembly .....	21
Figure 4.2: Traffic light wiring .....	21
Figure 4.3: Global view of the code .....	23
Figure 4.4: Focus on counter and analysis.....	23
Figure 4.5: Masks.....	25
Figure 4.6: Threshold .....	25
Figure 4.7: the HSV colorspace and find contour window .....	26
Figure 4.8: The system environment .....	26
Figure 4.9: The intersection of the lanes .....	26

## Appendix A

Figure A.1: Raspberry pi 3 GPIO.....	30
--------------------------------------	----

## Appendix B

Figure B.1: ATmega 328 Pinmap to Arduino UNO .....	31
--	----

## List of tables

Table 2.1: The Components Power Consumption .....	10
Table 3.1: Derived results after implementation.....	17
Table A.1: Raspberry pi 3 technical specifications .....	29
Table A.2: Board connections.....	30
Table B.1: Arduino UNO Board 328 technical specifications .....	31



## Nomenclatures

<b>IOT</b>	Internet of things
<b>IR sensor</b>	Infra-Red sensor
<b>OPENCV</b>	Open computer vision
<b>NUMPY</b>	Numerical PYTHON
<b>RGB</b>	Red Green Blue
<b>HSV</b>	Hue Saturation Value
<b>GUI</b>	Graphical user interface
<b>NOOBS</b>	New out of box system

# Introduction

The smart city concept integrates information and communication technology (ICT), and various physical devices connected to the network (the Internet of things or IoT) to optimize the efficiency of city operations and services and connect to citizens. Smart city technology allows city officials to interact directly with both community and city infrastructure and to monitor what is happening in the city and how the city is evolving.

A smart city is one of the most important applications in internet of things. In Smart cities there are several domains to be managed. It manages the urban security, energy management, automated transportation, water distribution and so on. IoT solves manage various problem arise in cities. It controls the pollution, shortage of energy supplies, traffic congestion and so on. In the Smart cities the sensor is enabled with internet connection. It helps the user to have a better parking, reduce traffic, communicate with road side objects, better garbage collection and smart lightning [1]. In all cases, the aim of these developments in the field is the necessity to replace human intervention.

Nowadays, many countries suffer from the traffic congestion problems that affect the transportation system in cities and cause serious dilemma. In spite of replacing traffic officers and flagmen by automatic traffic systems, the optimization of the heavy traffic jam is still a major issue to be faced. At the same time, in the whole world, enormous electric energy is consumed by the street lamps, which are automatically turn on when it becomes dark and automatically turn off when it becomes bright. It is a huge waste of energy in the whole world and that should be changed.

The development of a smart traffic light and smart streetlight project that can address these problems will be time and cost effective. In this project, a smart traffic light control system that overcome many defects and improves the traffic management and a smart streetlight system in which lights ON when needed and light OFF when not needed are developed and implemented.

# **Chapter 1: Introduction to the Smart Traffic and Streetlight Systems**

In modern times, there is a wide variety of technologies and procedures used to implement a smart traffic light control system and smart streetlight. Nevertheless, the one that have caught a lot of attention from both the researchers and hobbyists due to its simplicity is the use of microcontrollers or systems on chip as the brain of the system which connect through a GSM module, Wi-Fi module or Ethernet module to the internet where other technologies are used to monitor the sensors data sent from the hardware and control the different parts of the systems.

## **1.1 The Problem**

As the problem of urban traffic congestion spreads, there is a pressing need for the introduction of advanced technology and equipment to improve the state-of-the-art of traffic control. Traffic problems nowadays are increasing because of the growing number of vehicles and the limited resources provided by current infrastructures.

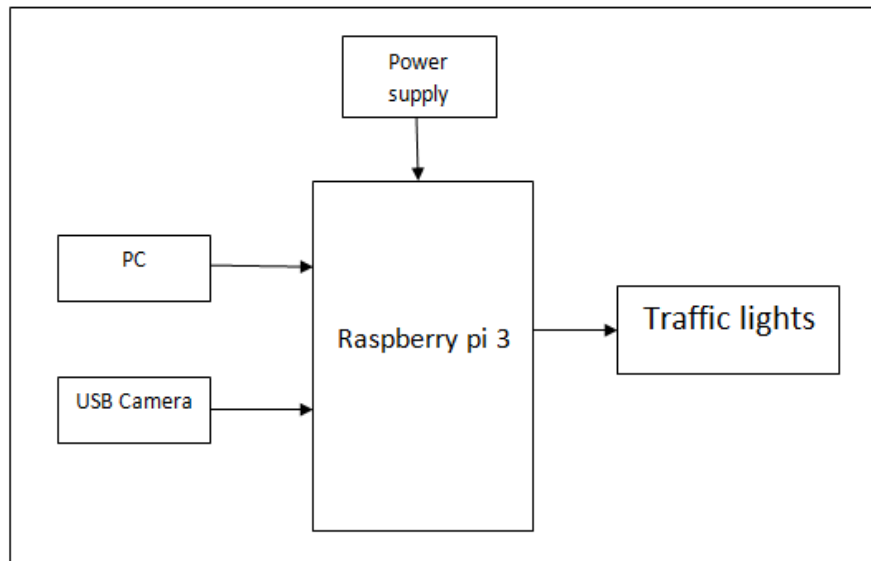
Each year, in the world, several trillion kWh of electrical energy are expended on street lighting. A high consumption means a high amount of generated energy, which in turn translates into a high level of noxious emissions.

## **1.2 The Solution**

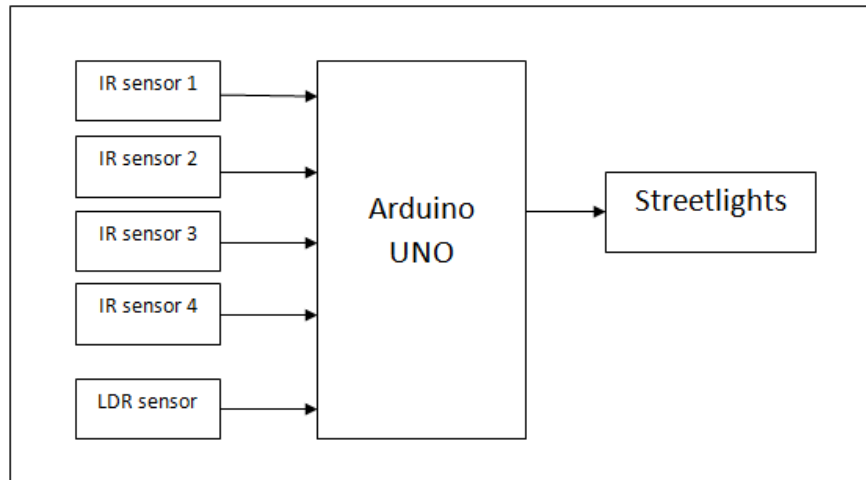
The solution to the problems mentioned earlier is to design a smart traffic light system and a smart streetlight system that can save time, costs and energy waste.

## **1.3 The Global System Diagram**

This section discusses the different parts that make this project, in addition to the technologies used to implement each part. Before jumping into the hardware part and software part, the entire system diagram must be established first, as the system goes large, one must take into consideration that it can be divided into smaller parts to address each part specifically. The following diagrams represent the global system structure.



**Figure 1.1:** *Smart traffic light Diagram*



**Figure 1.2:** *Smart streetlight Diagram*

Each part of the system is discussed into more details in the following sections.

## 1.4 Smart street light system

The different components are interconnected using the Arduino UNO board and the C programming language to control the entire system, this part is equipped with different sensors (IR sensors, Light dependent resistor LDR) and light emitting diodes to make the smart street light system.

## **1.5 Smart traffic light system**

This part consists of An Image Processing Based Intelligent Traffic Control System by using Raspberry pi 3 implemented using PYTHON Programming Language and openCV libraries. The lane with highest traffic density is given priority over all the others. “Image Processing Based Intelligent Traffic control using Raspberry Pi” technique overcomes all the limitations of the earlier (in use) techniques used for controlling the traffic. Earlier in automatic traffic control use of timer had a drawback that the time is being wasted by green light on the empty. This technique avoids this entire problem.

## **1.6 Summary**

In this chapter, the different parts of the system have been identified, in the next chapters the design and implementation of each part will be discussed in more details

# Chapter 2: The Hardware Design and Implementation

## Introduction

In this chapter, the description and interconnection of the different components is discussed in more details where the circuits' schematic is introduced. The development and implementation of smart traffic light system and smart street light system is divided into two steps, the first step is the circuit design where the different components are interfaced with the microcontroller and the second step is writing the program for the microcontroller.

## 2.1 The Components List

This section is to list all the used components, the components selection is based on three parameters:

- To reduce the hardware size.
- To reduce the power consumption.
- The decrease the product cost.

### 2.1.1 Raspberry pi 3

Raspberry pi 3 is a development board in PI series [Appendix A]. It can be considered as a single board computer that works on LINUX operating system. The board not only has tons of features it also has terrific processing speed making it suitable for advanced applications. It has Ethernet port, 4 USB ports, full HDMI port, micro USB power, an SD card slot and 40 GPIO pins. It has also combined 3.5mm audio jack and composite video, camera interface, display interface, and video Core IV 3D graphic core. [3]



**Figure 2.1:** *Raspberry pi*

### 2.1.2 The Arduino UNO

Arduino Uno is a microcontroller board based on the ATmega328P [Appendix B]. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.[5]



**Figure 2.2:** *Arduino UNO*

### 2.1.3 The camera

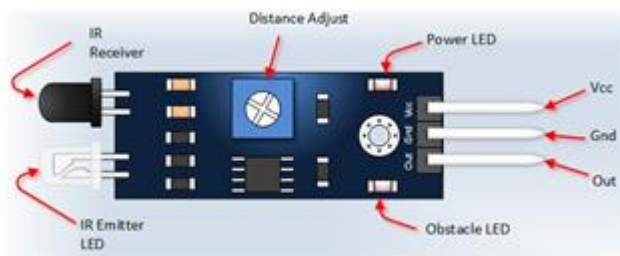
The used camera is a Jeway JW-5329 feet high 1080P HD 2.0 MP USB Computer Webcam (130cm-Cable).



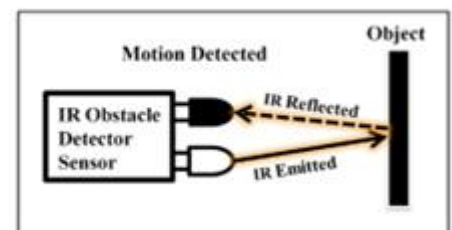
**Figure 2.3:** *JEWAY webcam*

### 2.1.4 Infra Red (IR) sensors

An IR sensor consists of an infrared-transmitter, an infrared-receiver and a potentiometer for adjusting the distance, shown in Fig 2.4. Whenever an object passes in front of a sensor, the emitted rays hit the surface of an object and reflect to the receiver of the sensor so it will consider this as a motion (as shown in Figure 2.5). It is a heat sensitive sensor and used for detection of motion. Based on the intensity of the reception by the IR receiver, the output of the sensor is defined.[6]



**Figure 2.4:** *IR sensor*



**Figure 2.5:** *Working principle of an IR*

### 2.1.5 Light dependent resistor (LDR)

LDR is a Light Dependent Resistor whose resistance is dependent on the light impinging on it. The resistance offered by the sensor decreases with the increase in light strength and increases with the decrease in light strength. This device is used for detection of day-time and night-time because when sunlight falls on it, it will consider as day-time, and when there is no sunlight falls on it, it will be regarded as a night. This resistor (LDR) must be inserted next to a normal resistor in series and based on the resulted voltage. These are very beneficial, especially in light/dark sensor circuits and help in automatically switching ON /OFF the street lights. [8]



**Figure 2.6:***Light dependent resistor (LDR)*

### 2.1.6 The light emitting diodes (LEDs)

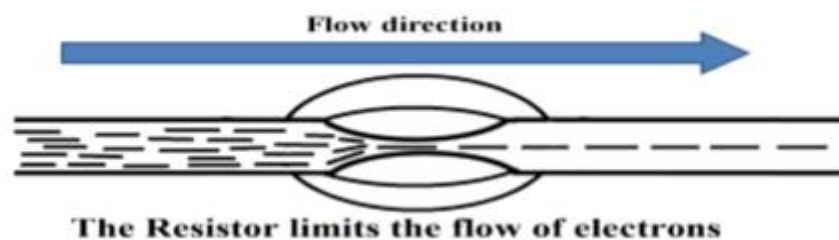
A LED (light-emitting diode) is a PN junction diode which is used for emitting visible light when it is activated. When the voltage is applied over its elements, electrons regroup with holes within the LED, releasing energy in the form of photons which gives the visible light. LEDs may have the Dim/full capability.



**Figure 2.7:***Light emitting diode (LED)*

### 2.1.7 Resistors

A resistor is a passive electronic component, used with other electronic components such as LEDs and sensors to prevent or limit the flow of electrons through them as illustrated in Fig. 2.8. It works on the principle of Ohm's law which prevent overflow of voltage.



**Figure 2.8:***Working principle of resistor*

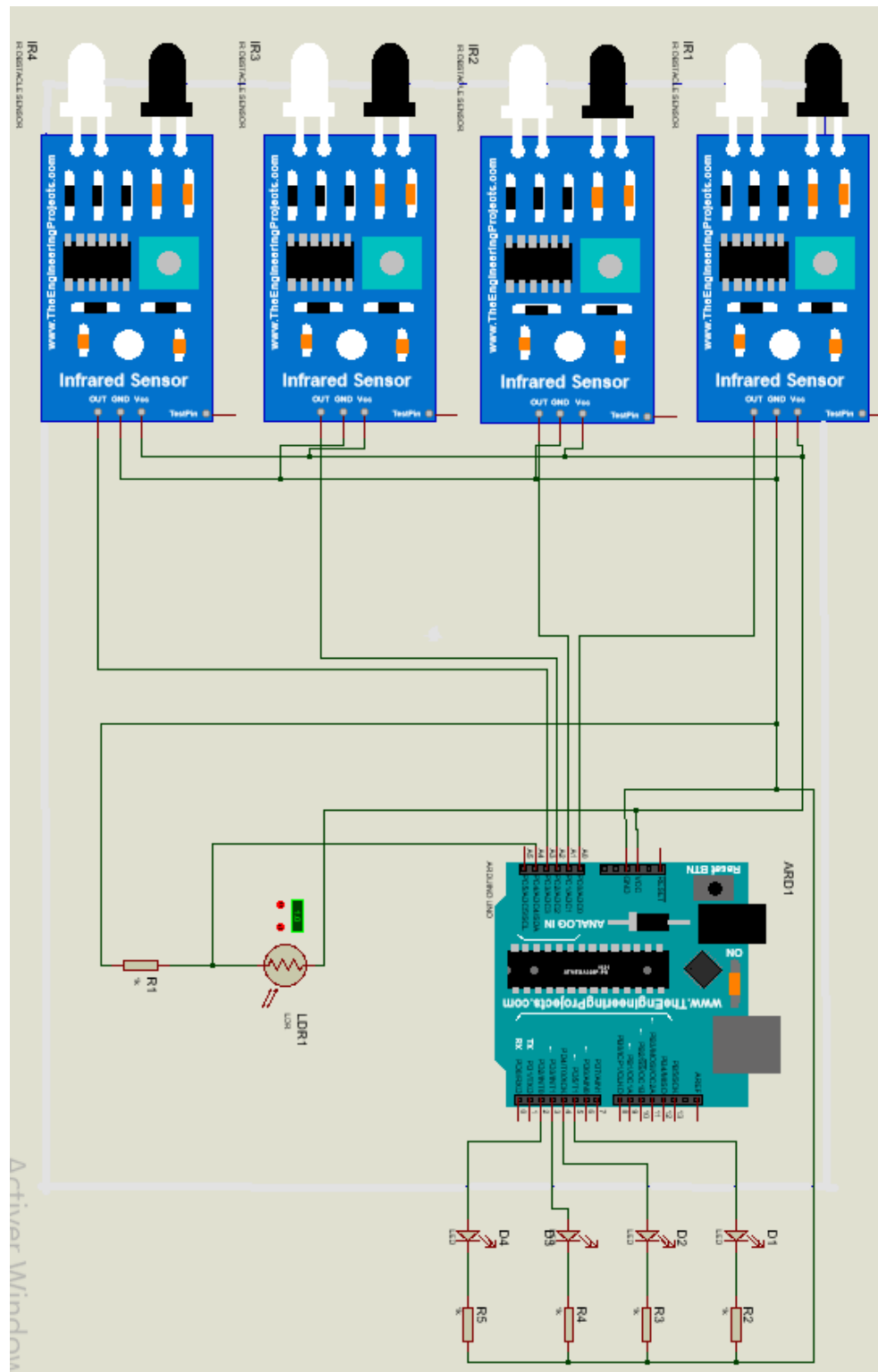


**Figure 2.9:***Resistors*



## 2.2 The streetlight interfacing circuit

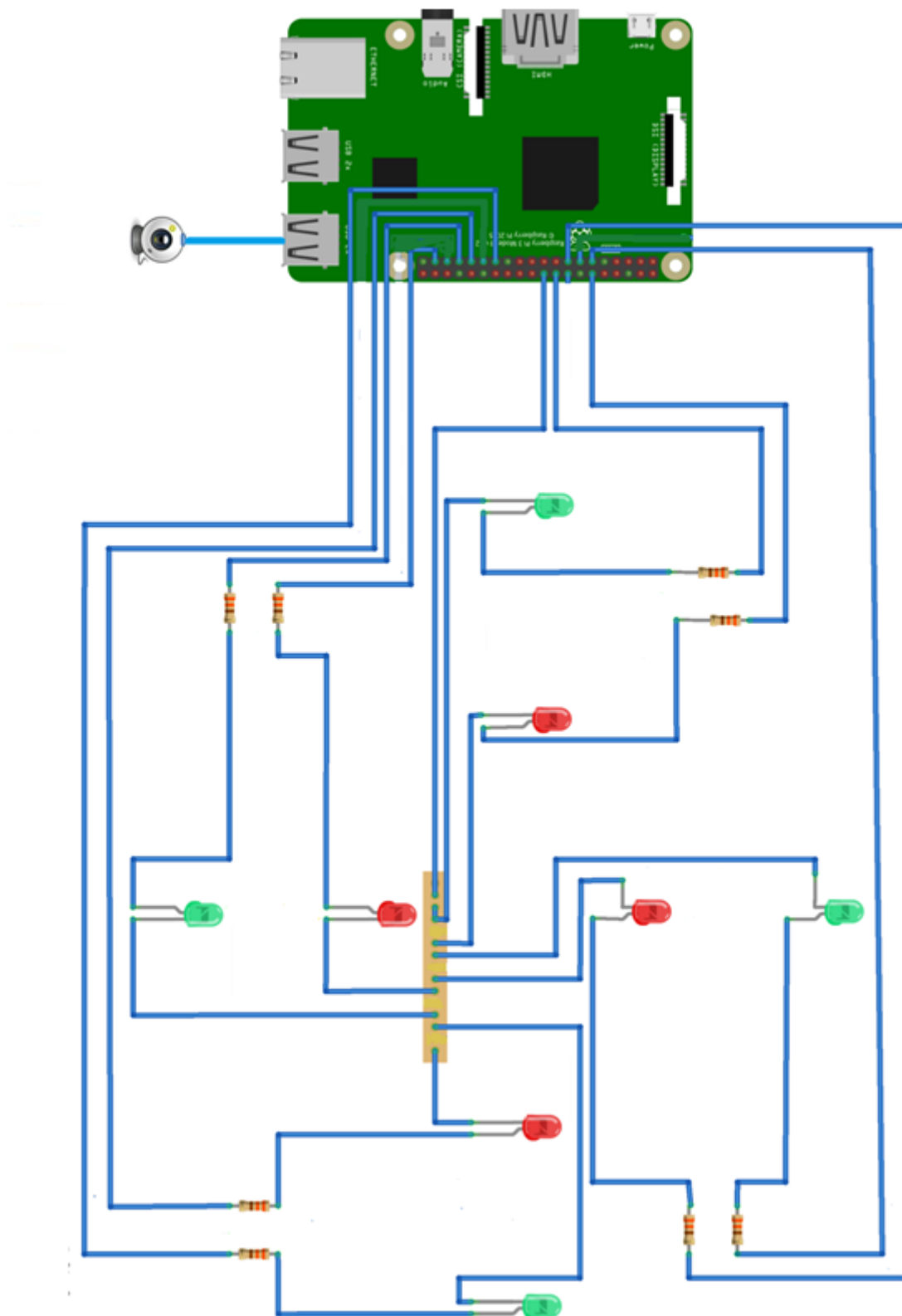
In this part the interfacing of the different components of streetlight system is introduced (Figure 2.10).



**Figure 2.10:** *Streetlight circuit*

## 2.3 The traffic light interfacing circuit

The circuit of the traffic light system is introduced in figure 2.11.



**Figure 2.11:** *Traffic light circuit*

## 2.4 The system power consumption

In this section, the system's power consumption is calculated by summing all the components power consumption. The table below (Table 2.1) contains each component's power consumption.

The component name	Number of used components	Power consumption (mW)
Raspberry pi 3	1	1200
Arduino UNO	1	232.5

**Table 2.1:** *The Components Power Consumption*

- The Arduino Uno board draws a maximum of 233mW and since this board supplies all the sensors, the light dependent resistor, the LEDs, we don't have to compute their power consumption.
- The raspberry pi 3 B consumes 1200mW; it supplies the USB camera, the servomotor and the LEDs.

To find the total power being consumed by the system we sum the power consumption of all the components:

$$P_{Total} = 1200 + 232.5 = 1432.5 \text{ mW}$$

The total power consumed by the system is 1432.5 mW.

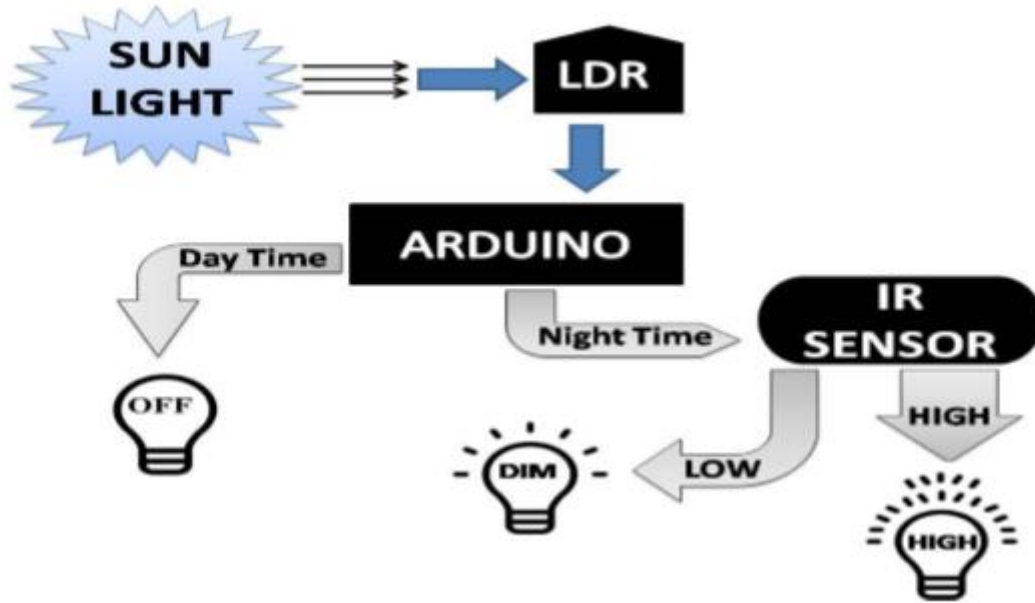
## 2.5 Summary

This chapter covered the used hardware components and the circuits used to implement this project. At the end, the system power consumption was introduced.

## Chapter 3: The streetlight system

### Introduction

Street lights play a vital role in our environment and also plays a critical role in providing light for safety during night-time travel. In this scenario, when the street lights are in working functionality over the whole night that consumes a lot of energy and reduces the lifetime of the electrical equipment such as electric bulb etc. Especially in cities' streetlights, it is a severe power consuming factor and also the most significant energy expenses for a city. In this regard, an intelligent lighting control system can decrease street lighting costs up to 70% [9] and increase the durability of the equipment.



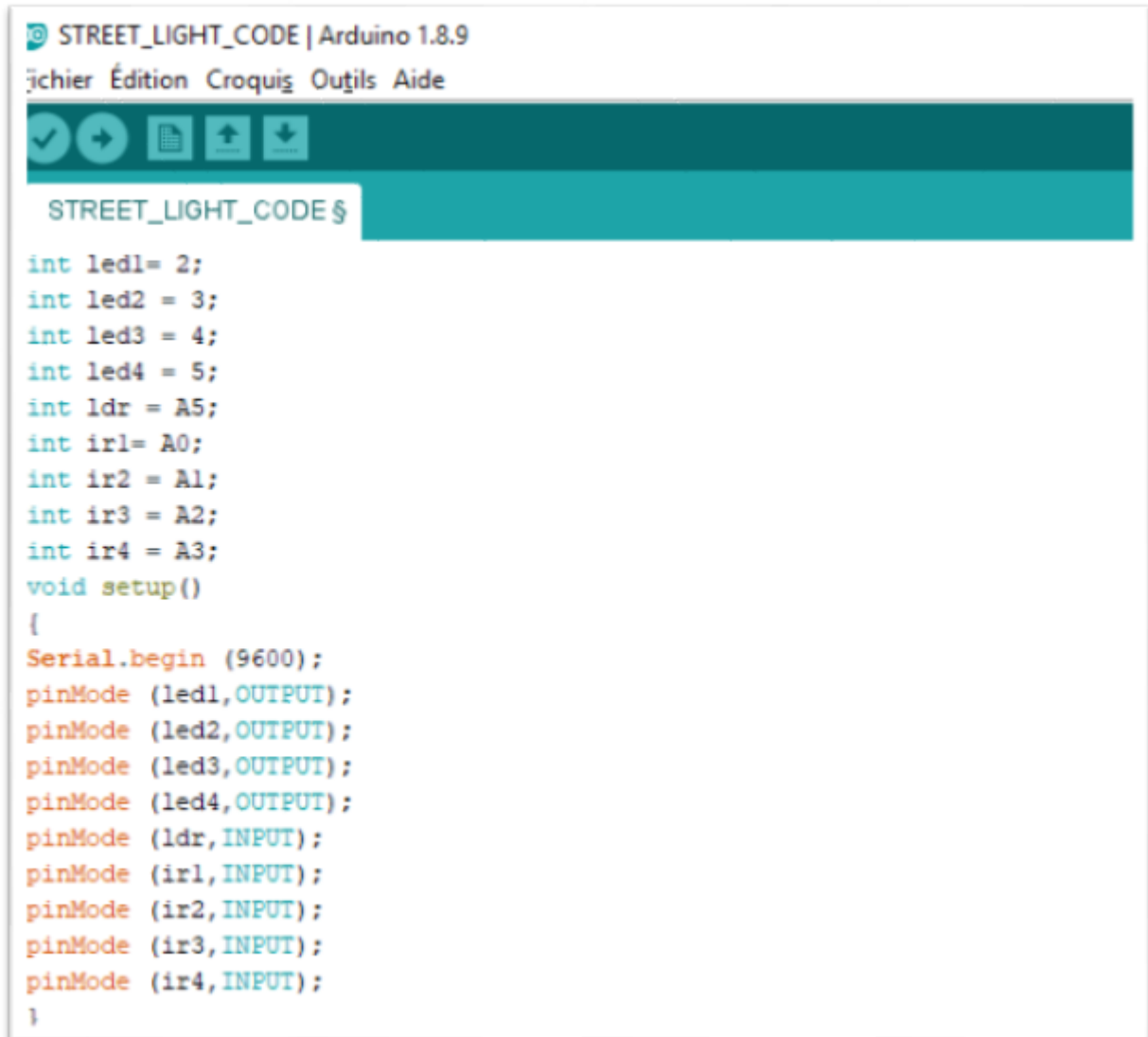
**Figure 3.1:** The architecture design of Smart Street light control system.

For the simplicity of discussion, **figure 3.1** illustrates the overall working mechanism and the features of the proposed lighting concept. Firstly, LDR will sense the intensity value of sunlight and send it to Arduino which will judge if the received value is above the threshold level (which is set independently by the user from the discrete value: 0-1023), then it will consider it as daytime and LEDs will remain OFF, or if the received value below the threshold level, Arduino will consider it as a night-time. In the night-time, if the value of IR obstacle detector sensor is LOW and detects no object, then DIM LEDs (half of its maximum voltage) will glow, or if IR obstacle detector value is HIGH and detects any object, then HIGH LEDs (full of its maximum voltage) will glow.

### 3.1 Designing Methodology

One leg of LDR sensor is connected to Arduino analog pin number A5 and another leg to VCC pin and same with a resistor to the ground port of Arduino. In addition, the threshold value is adjusted to 500 from the discrete values (0-1023) for understanding whether it is day or night. After that, all the positive terminals of the LEDs are connected to pin number 2, 3, 4 and 5, depicting the streetlights as the outputs of the Arduino signals. Furthermore, the grounds of all the LEDs are connected with resistors to Ground port. The IR sensors are connected to the Arduino port from pin A0, A1, A2 and A3, respectively, which is the input signal to the Arduino board. Similarly, the grounds of all the IR sensors are connected to GND port and all VCC of IR sensors are attached to Arduino 5V pin

After connecting all these devices to the corresponding pins in Arduino, the Arduino Software from the official website “[www.arduino.cc](http://www.arduino.cc)” is downloaded and installed. Then Arduino Uno is connected to the computer using the USB cable and installed the driver software on the computer to write, compile and run the software code on Arduino software.

The image shows a screenshot of the Arduino IDE interface. The title bar reads "STREET\_LIGHT\_CODE | Arduino 1.8.9". Below the title bar is a menu bar with "Fichier", "Édition", "Croquis", "Outils", and "Aide". Underneath the menu bar is a toolbar with icons for checking, running, saving, uploading, and downloading. The main text area contains the following C++ code:

```
STREET_LIGHT_CODE $  
  
int led1= 2;  
int led2 = 3;  
int led3 = 4;  
int led4 = 5;  
int ldr = A5;  
int irl= A0;  
int ir2 = A1;  
int ir3 = A2;  
int ir4 = A3;  
void setup()  
{  
  Serial.begin (9600);  
  pinMode (led1,OUTPUT);  
  pinMode (led2,OUTPUT);  
  pinMode (led3,OUTPUT);  
  pinMode (led4,OUTPUT);  
  pinMode (ldr,INPUT);  
  pinMode (irl,INPUT);  
  pinMode (ir2,INPUT);  
  pinMode (ir3,INPUT);  
  pinMode (ir4,INPUT);  
}
```

**Figure 3.2:** *Initialization of variables and setup of Arduino pins*

### 3.2 The software codes

Arduino code is written in C++ with an addition of special methods and functions. C++ is a human-readable programming language. When you create a ‘sketch’ (the name given to Arduino code files), it is processed and compiled to machine language. The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you’ll be typing up your code before uploading into the board you want to program.



STREET\_LIGHT\_CODE | Arduino 1.8.9

Fichier Édition Croquis Outils Aide

STREET\_LIGHT\_CODE \$

```
void loop()
{
  Serial.println(analogRead(A5));
  int ldrStatus = analogRead (ldr);

  if (ldrStatus <=500)
  {
    digitalWrite(led1, HIGH);
    analogWrite (led1,255/5);
    digitalWrite(led2, HIGH);
    analogWrite (led2,255/5);
    digitalWrite(led3, HIGH);
    analogWrite (led3,255/5);
    digitalWrite(led4, HIGH);
    analogWrite (led4,255/5);
  }
}
```

**Figure 3.3:** *The LDR loop*

The image shows a screenshot of the Arduino IDE interface. At the top, the title bar reads "STREET\_LIGHT\_CODE | Arduino 1.8.9". Below it is a menu bar with "Fichier", "Édition", "Croquis", "Outils", and "Aide". A toolbar with icons for saving, running, and other functions is visible. The main text area shows the following code:

```
STREET_LIGHT_CODE $

    if (analogRead(A0)<300)          // IR1 CODE
    {
        digitalWrite(led1,HIGH);
        analogWrite(led1,255);
        delay(1000);// micro second
    }
    else
    {
        digitalWrite(led1,HIGH);
        analogWrite(led1,255/5);
    }

    if (analogRead(A1)<300)          // IR2 CODE
    {
        digitalWrite(led2,HIGH);
        analogWrite(led2,255);
        delay(1000);// micro second
    }
    else
    {
        digitalWrite(led2,HIGH);
        analogWrite(led2,255/5);
    }
```

**Figure 3.4:** *The code of the two first IR sensors*



```
STREET_LIGHT_CODE | Arduino 1.8.9
Fichier Édition Croquis Outils Aide

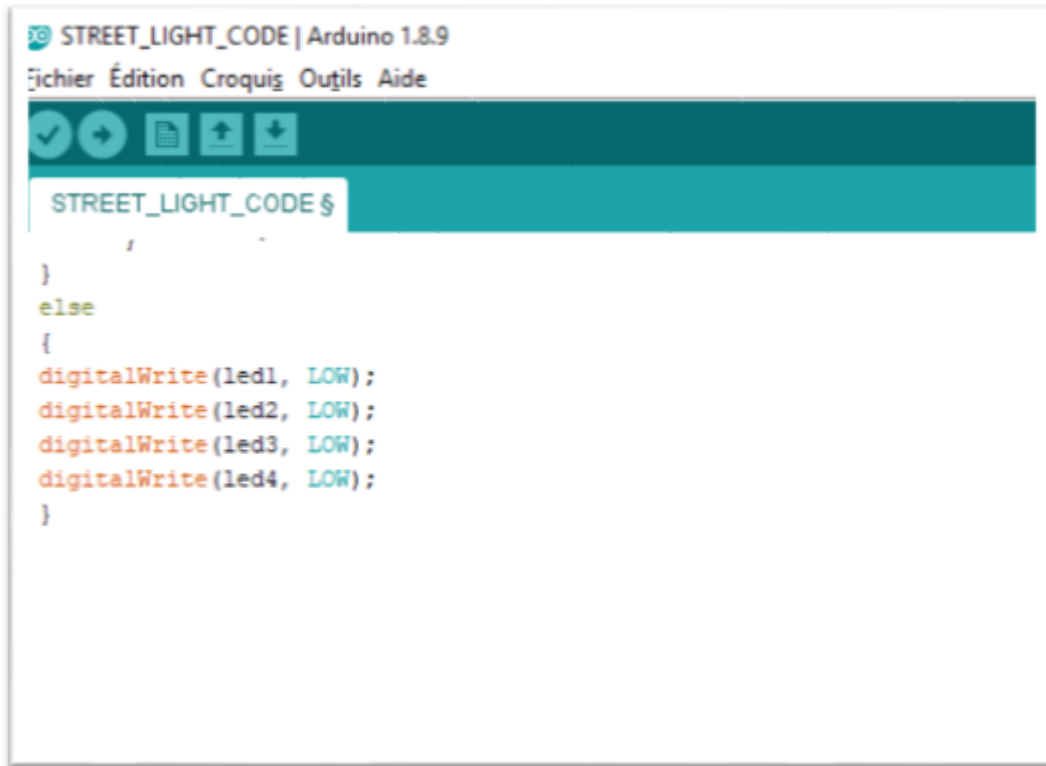
STREET_LIGHT_CODE $

    if (analogRead(A2) < 300)          // IR3 CODE
    {
        digitalWrite(led3, HIGH);
        analogWrite(led3, 255);
        delay(1000); // micro second
    }
    else
    {
        digitalWrite(led3, HIGH);
        analogWrite(led3, 255/5);
    }

    if (analogRead(A3) < 300)          // IR4 CODE
    {
        digitalWrite(led4, HIGH);
        analogWrite(led4, 255);
        delay(1000);                  // micro second
    }
    else
    {
        digitalWrite(led4, HIGH);
        analogWrite(led4, 255/5);
    }
```

**Figure 3.5:** *The code of the two last IR sensors*





**Figure 3.6:** *The output of LEDs during the daytime*

### 3.3 Results and discussions

In the beginning, the LDR sensor will sense the light intensity in the atmosphere at that time and consequently sends the data to Arduino. After receiving the data, Arduino will convert it into different discrete values from 0 to 1023 (In which 0 represents maximum darkness and 1023 represents maximum brightness) and then it will adjust the output voltage accordingly from 0 to 2.5v/5v (Dim/High) depending upon the received value (0-1023) by comparing with threshold value. So, the output will be 2.5v in the complete darkness (night time) if the received value is less than the threshold value. As a result, Dim LEDs will glow that is the half of maximum brightness, and when there is completely shine (daytime), the received value will be higher than the threshold value, and the output voltage would be 0v resulting the LEDs to be entirely switched OFF.

Initially, the IR obstacle detection sensor will be HIGH. So, when there is no vehicle or obstacle in front of the sensor, IR Transmitter does continuously transmit the IR light. Whenever, a car or any other object blocks any of the IR sensors, then the emitted rays will reflect the IR receiver after hitting the object, then microcontroller will sense it as a motion. In simple words, when any object passed in front of the first IR sensor, the corresponding LEDs will be turned from DIM to HIGH (5v) by the microcontroller. As the object moves forward and blocks the next IR sensor, the next two LEDs will be turned to HIGH from DIM, and the LEDs from the previous set is switched to DIM from HIGH. The process continues this way for the entire IR obstacle detector sensors and LEDs. These kinds of application can be implemented in the headlights of vehicles, street lights, parking lights of hotels, malls and homes, and it can be very beneficial.

**Table 3.1:** *Derived results after implementation*

Device name	Input Data	Verified results	Remarks
Arduino board testing	Digital signal	Switching of LEDs at different intervals.	Hardware is accurate
Light dependent resistor testing	Outside light intensity value	Dim/High LEDs glows according to light intensity.	Hardware is accurate
IR sensor testing	Sense motion	High LEDs glows whenever it detects motion.	Hardware is accurate

### 3.4 Summary

The proposed streetlight automation system is a cost effective and the safest way to reduce power consumption. It helps us to get rid of today's world problems of manual switching and most importantly, primary cost and maintenance can be decreased easily. The LED consumes less energy with cool-white light emission and has a better life than high energy consuming lamps. Moving to the new & renewable energy sources, this system can be upgraded by replacing conventional LED modules with the solar-based LED modules. With these efficient reasons, this presented work has more advantages which can overcome the present limitations. Keep in mind that these long-term benefits; the starting cost would never be a problem because the return time of investment is very less. This system can be easily implemented in street lights, smart cities, home automation, agriculture field monitoring, timely automated lights, parking lights of hospitals, malls, airport, universities and industries etc

# Chapter 4: The traffic light system

## Introduction

In today's world, traffic lights are essential for a safe road. However, many times, traffic lights can be annoying in situations where someone is approaching the light just as it is turning red. This wastes time, especially if the light is preventing a single vehicle from getting through the intersection when there is nobody else on the road. This project is a smart traffic light that uses live object detection from a camera to count the number of cars on each road [10].

Using OpenCV on the Raspberry Pi [11], the information gathered will be run through code that controls the LEDs via the GPIO. Depending on these numbers, the traffic light will change, letting cars through in the most optimal order. In this case, the lane with the most cars would be let through so that the lane with fewer cars would be idling, reducing air pollution. This would eliminate situations when many cars are stopped while there are no cars on the intersecting road. Not only does this save time for everybody, but it also saves the environment. The amount of time people are stopped at a stop sign with their engine idling increases the amount of air pollution, so by creating a smart traffic light, we are able to optimize the light patterns so that cars spend the least time possible with their vehicle stopped [12]. Ultimately, this traffic light system could be implemented in cities, suburbs, or even rural areas to be more efficient for people would reduce air pollution.

## 4.1 Image processing

In electrical engineering and computer science, image processing is any form of signal processing for which the input is an image, such as photographs or frames of video; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing are also possible.

To process the images of this project, we go through the following steps:

### 4.1.1 Transform frame into NumPy array

NumPy is a package popularly used for scientific computing with Python. NumPy contains a number of useful concepts such as array objects (for representing vectors, matrices, images and much more) and linear algebra functions. The NumPy array object lets you do important operations such as matrix multiplication, transposition, solving equation systems, vector multiplication, and normalization, which are needed to do things like aligning images, warping images, modelling variations, classifying images, grouping images, and so on. When we loaded images, we converted them to NumPy array objects with the `np.array()` but didn't mention what that means.

Arrays in NumPy are multi-dimensional and can represent vectors, matrices, and images. An array is much like a list (or list of lists) but is restricted to having all elements of the same type. Unless specified on creation, the type will automatically be set depending on the data. NumPy arrays are used to process frame by frame and to be able to access the pixel RGB values.

### 4.1.2 Changing colorspace

There are more than 150 color-space conversion methods available in OpenCV. But we are interested into only one which is most widely used, BGR  $\leftrightarrow$  HSV

HSV colour system is based on the **H**ue shift, **S**aturation and **V**alue. Unlike the RGB colour system,

which has to do with "implementation details" regarding the way RGB displays colour, HSV has to do with the "actual colour" components. Another way to say this would be RGB is the way computers treat colour, and HSV try to capture the components of the way we humans perceive colour. In the HSV representation of color, hue determines the color you want, saturation determines how intense the color is and value determines the lightness of the image.

The main reason to work on the HSV version of an image is because in RGB, we cannot separate colour information from luminance. HSV or Hue Saturation Value is used to separate image luminance from colour information. Using Hue component makes the algorithms less sensitive (if not invariant) to lighting variations.

HSV is more robust towards external lighting changes. This means that in cases of minor changes in external lighting (such as pale shadows, etc.) Hue values vary relatively lesser than RGB values. For example; two shades of red colour might have similar Hue values, but widely different RGB values. In real life scenarios such as object tracking based on colour, we need to make sure that our program runs well irrespective of environmental changes as much as possible. So, we prefer HSV colour thresholding over RGB.

#### 4.1.3 Threshold the HSV colorspace

To isolate the colours, we have to apply multiple masks. A low threshold and high threshold mask for hue, saturation and value. Anything pixel within these thresholds will be set to 1 and the remaining pixels will be zero. Thresholding the HSV image is used to get only blue colours and to exclude black and white (road and strips).

#### 4.1.4 Blurring

Smoothing and blurring techniques help us in eliminating noises from our image. There are various types of smoothing and blurring techniques available at direct disposal from openCV. Each and every technique has its own advantages as well as disadvantages.

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noises. It actually removes high frequency content (eg: noise, edges) from the image. So edges are blurred a little bit in this operation. Well, there are blurring techniques which doesn't blur the edges too. OpenCV provides mainly four types of blurring techniques.

#### 4.1.5 Averaging

This is done by convolving image with a normalized box filter. It simply takes the average of all the pixels under kernel area and replaces the central element. This is done by the function **cv2.blur()** or **cv2.boxFilter()**.

#### 4.1.6 Gaussian blurring

In this, instead of box filter, Gaussian kernel is used. It is done with the function, **cv2.GaussianBlur()**. We should specify the width and height of kernel which should be positive and odd. We also should specify the standard deviation in X and Y direction, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as same as sigmaX. If both are given as zeros, they are calculated from kernel size. Gaussian blurring is highly effective in removing Gaussian noise from the image.

#### 4.1.7 Median blurring

Here, the function **cv2.medianBlur ()** takes median of all the pixels under kernel area and central element is replaced with this median value. This is highly effective against salt-and-pepper noise in the images. Interesting thing is that, in the above filters, central element is a newly calculated value which may be a pixel value in the image or a new value. But in median blurring, central element is always replaced by some pixel value in the image. It reduces the noise effectively. Its kernel size should be a positive odd integer.

#### 4.1.8 Bilateral Filtering

**cv2.bilateralFilter ()** is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. We already know that Gaussian filter takes the neighbourhood around the pixel and find its Gaussian weighted average. This Gaussian filter is a function of space alone, that is, nearby pixels is considered while filtering. It doesn't consider whether pixels have almost same intensity. It doesn't consider whether pixel is an edge pixel or not. So it blurs the edges also, which we don't want to do. Bilateral filter also takes a Gaussian filter in space, but one more Gaussian filter which is a function of pixel difference. Gaussian function of space make sure only nearby pixels are considered for blurring while Gaussian function of intensity difference make sure only those pixels with similar intensity to central pixel is considered for blurring. So it preserves the edges since pixels at edges will have large intensity variation.

#### 4.1.9 Morphological operations

Morphological operations are a set of operations that process images based on shapes. They apply a structuring element to an input image and generate an output image.

The most basic morphological operations are two: **Erosion and Dilation**.

##### 4.1.9.1 Erosion

In the Erosion, it erodes away the boundaries of foreground objects. It is used to remove small white noises from the images. Erosion can also be used to detach two connected images.

- A kernel is formed from an image. The kernel is a matrix, where the order is odd, like 3, 5, and 7.
- A pixel of image is chosen. This pixel will be chosen as 1 only if all the pixels under the kernel are 1. Otherwise it will be eroded.
- So all pixels near the boundary will be discarded.
- So the thickness of foreground object decreases.

##### 4.1.9.2 Dilation

In the Dilation, it increases the object area. The Erosion can remove the white noises, but it also shrinks our image, so after Erosion, if Dilation is performed, we can get better noise removal results. The Dilation can also be used to joins some broken parts of an object.

- A kernel is formed from an image. The kernel is a matrix, where the order is odd, like 3, 5, and 7.
- A pixel of image is chosen. This pixel will be chosen as 1 only if all the pixels under the kernel are 1.
- Increases the white region or the size of foreground objects of the image

## 4.2 Designing Methodology

### 4.2.1 Raspberry pi 3 and USB camera interfacing

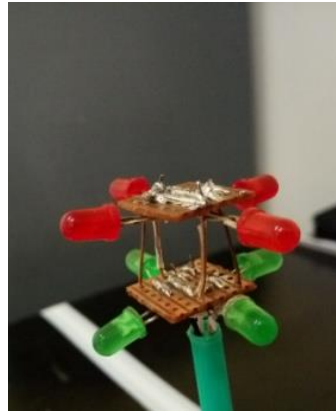
The hardware that will be used for this project is a Raspberry Pi 3, a camera module, and various electronic hardware for the light itself. The raspberry pi is connected with a camera through the USB port, it is powered up using 5v external power supply, and it is remotely accessed via laptop using VNC server and VNC viewer.

### 4.2.2 Building the environment

We make a 75cm square wood pallet; drill a hole that fits around the PVC pipe on one corner. The roads are marked with some white tape. Once everything is finalized, we paint the wood pallet with some black spray paint to clean up the look of the main frame.

### 4.2.3 Traffic light assembly

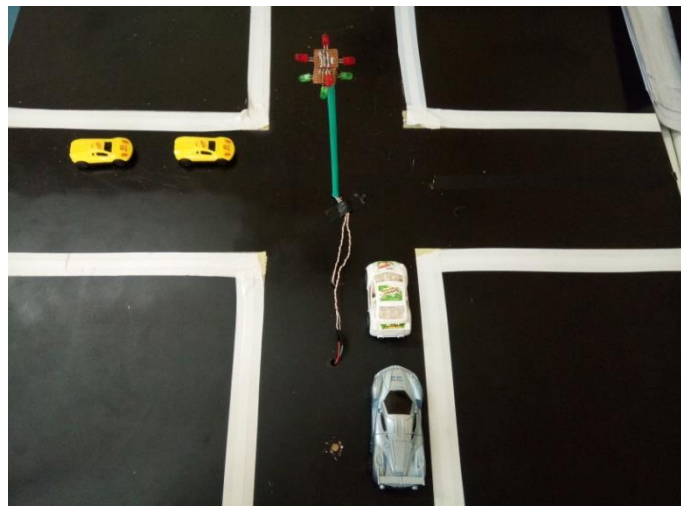
Each opposing set of LEDs share an anode and all of them share a common cathode (ground). There should be a total of 5 input wires: 1 for each pair of LEDS (4) + 1 ground wire. Solder and assemble the traffic light.



**Figure 4.1:** *Traffic light assembly*

### 4.2.4 Wiring

On wood pallet, we drill a hole that can fit a bundle of wires. A rough hole is fine as long as you can pass the wires through. We snake the wires through the holes. Finally we add a pipe on the center of the wood pallet for the traffic light.



**Figure 4.2:** *Traffic light wiring*

## 4.3 Software code

Traffic light code is written in PYTHON in addition to the use of OPENCV libraries.

### 4.3.1 Python

Python is a powerful modern computer programming language. It bears some similarities to FORTRAN, one of the earliest programming languages, but it is much more powerful than FORTRAN. Python allows you to use variables without declaring them, it determines types implicitly, and it relies on indentation as a control structure. You are not forced to define classes in Python unlike Java, but you are free to do so when convenient. Python was developed by Guido van Rossum, and it is free software. Python can be obtained without spending any money. But Python is also free in other important ways, for example you are free to copy it as many times as you like, and free to study the source code, and make changes to it. There is a worldwide movement behind the idea of free software, initiated in 1983 by Richard Stallman. Like shell scripts, Python can automate tasks like batch renaming and moving large amounts of files. It can be used just like a command line with IDLE, Python's REPL (read, eval, print, loop) function.

However, there are more useful things you can do with Python. For example, you can use Python to program multiple things like:

- Web applications
- Desktop applications and utilities
- Special GUIs
- Small databases
- 2D games

Python also has a large collection of libraries, which speeds up the development process. There are libraries for everything you can think of; game programming, rendering graphics, GUI interfaces, web frameworks, and scientific computing. Many things done in C can be done in Python but not all. Python is generally slower at computations than C, but its ease of use makes Python an ideal language for prototyping programs and designing applications that aren't computationally intensive.

### 4.3.2 OpenCV

Open CV Stands for Open computer vision it is source library of functions. It is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

### 4.3.3 Flowchart of the program

The algorithm of the microcontroller will be introduced through a flowchart.

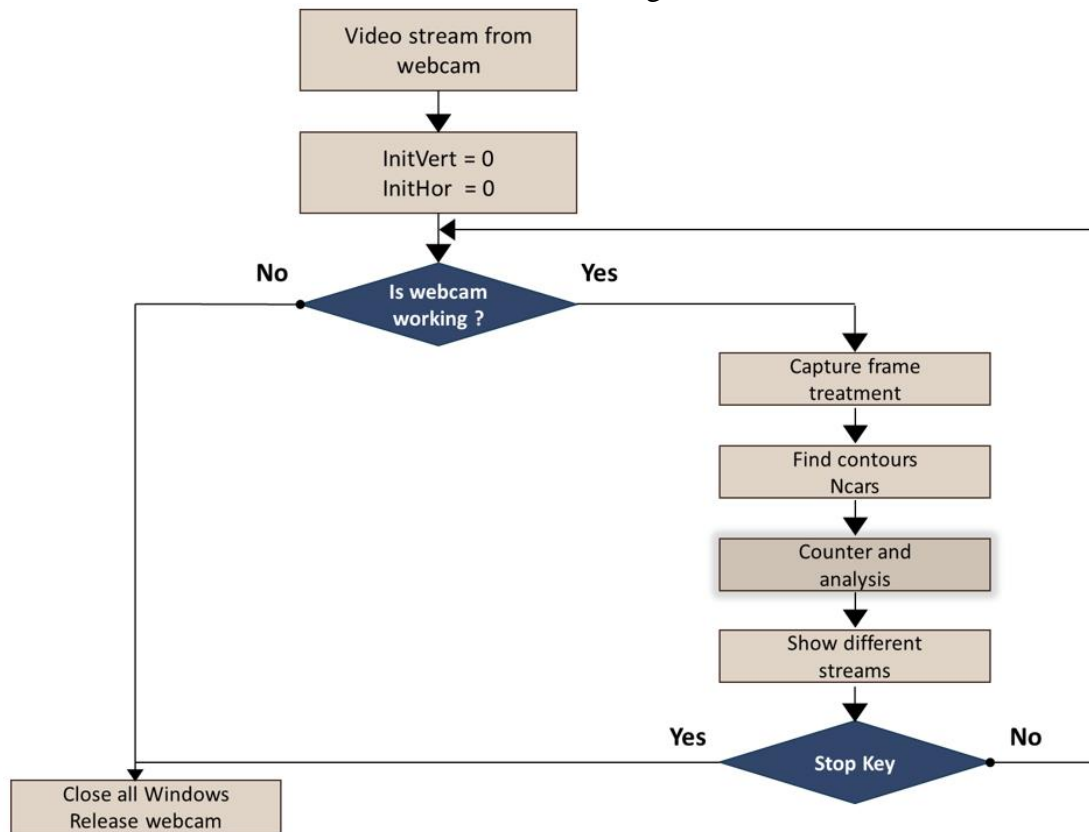


Figure 4.3: Global view of the code

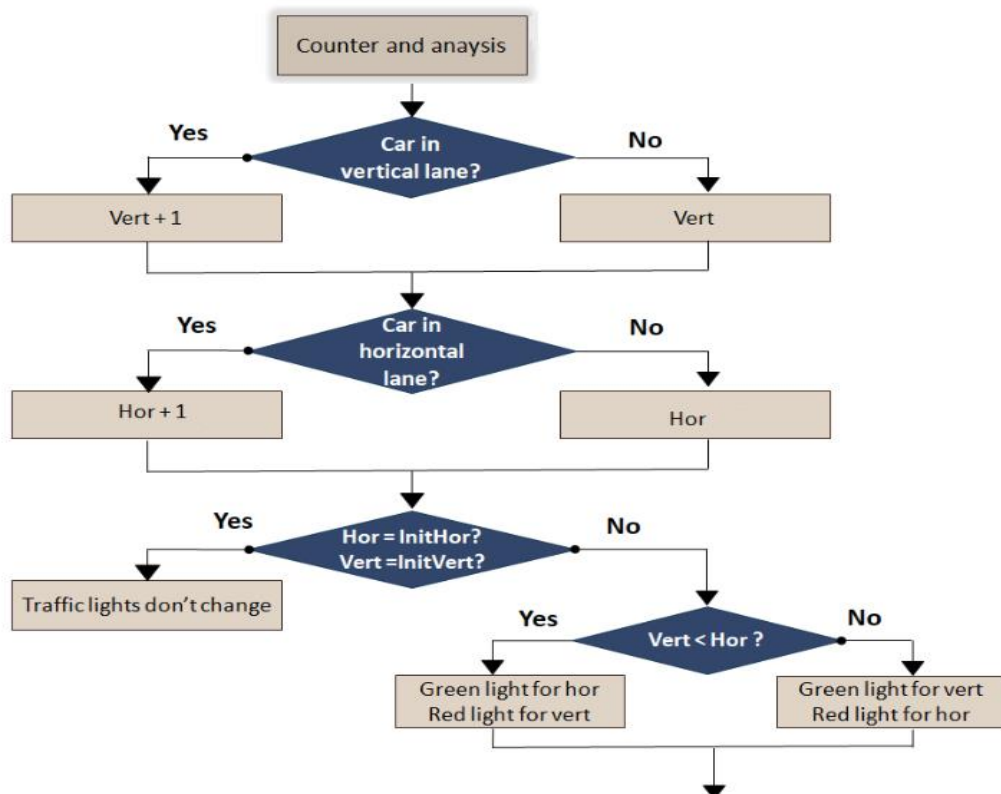


Figure 4.4: Focus on counter and analysis



## 4.4 Results and discussion

### 4.4.1 Setup of the raspberry pi 3

Before using the Raspberry Pi, we need an SD card with an operating system installed or with the New Out Of Box System (NOOBS) on it from *Raspberrypi.org*

The setup of the raspberry pi goes through multiple steps:

- Insert the SD card on the raspberry pi.
- Connect monitor, mouse and a keyboard.
- Power the board with a micro USB connector.
- Once the power is turned ON the raspberry pi will run on the OS installed in the memory card and will start from boot.
- Once all drivers are checked the PI will ask for authorization, this is set by default and can be changed.
- After authorization you will reach desktop where all application program development starts.

Once the setup of Raspberry Pi is finished, we want to connect to the raspberry pi remotely and use it through a network link, in which case we only need power and a network connection. We refer to this as connecting remotely to the Raspberry Pi over the network using VNC, and Connecting remotely to the Raspberry Pi over the network using SSH.

To access the pi remotely using VNC server we go through these steps

- On the Raspberry Pi, we enable the VNC server.
- Using a terminal window or via SSH, we execute the instruction *ifconfig* to discover the private IP address.
- On the device we want to take control, in our case it is a laptop, we download VNC Viewer. For best results, we use the compatible app from RealVNC.
- We enter the Raspberry Pi's private IP address into VNC Viewer.

### 4.4.2 Installation of openCV libraries

The installation of openCV libraries goes over multiple steps

- First of all, we update the package lists by running the command  
`sudo apt-get update && sudo apt-get upgrade`
- A reboot is necessary after updating  
`sudo reboot`
- Install all the important tools and libraries needed for OpenCV
  - 1<sup>st</sup> package: `sudo apt-get install build-essential cmakepkg-config`
  - 2<sup>nd</sup> package: `sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev`
  - 3<sup>rd</sup> package: `sudo apt-get install libavcodec-dev libaavformat-dev libswscale-dev libv4l-dev`
  - 4<sup>th</sup> package: `sudo apt-get install libxvidcore-dev libx264-dev`
  - 5<sup>th</sup> package: `sudo apt-get install libgtk2.0-dev libgtk-3-dev`
  - 6<sup>th</sup> package: `sudo apt-get install libatlas-base-dev gfortan`
- Install python 3  
`sudo apt-get install python3-dev`
- Install python 3 pip  
`sudo apt-get install python3-pip`
- Install openCV library  
`Pip3 install opencv-python`

After installing everything, we checked for python 3 if it is working, we get an error:

```
importError: libQtGui.so.4: cannot open shared object file: No such file or directory
```

That means we have some missing library, so we have to install it using

```
sudo apt-get install libqtgui4
```

After rechecking the installation of libraries the python software works fine.

After running the program, multiple treatments are applied to the frame; figure 4.5 represents the frame after applying the masks.



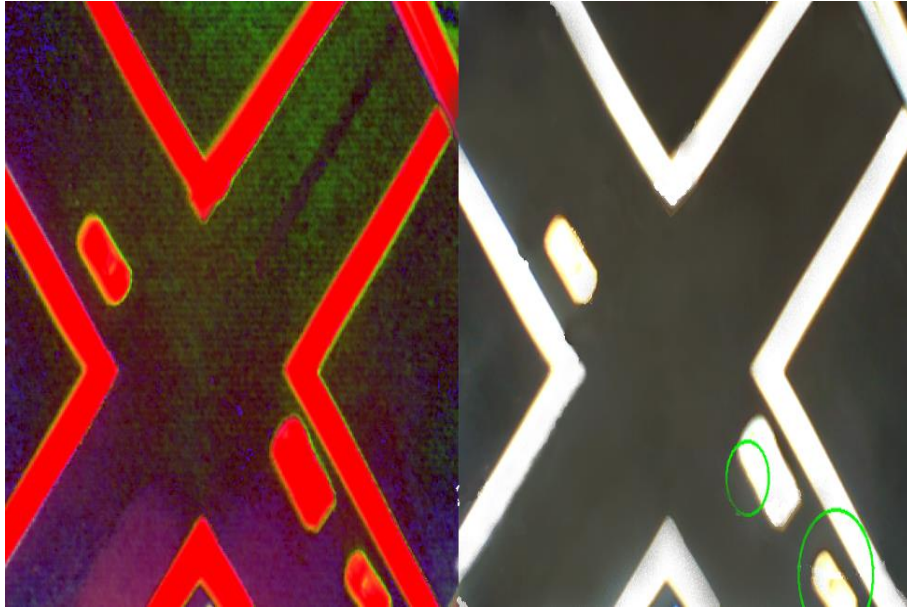
**Figure 4.5:** *masks*

The figure 4.6 represents the frame thresholding.



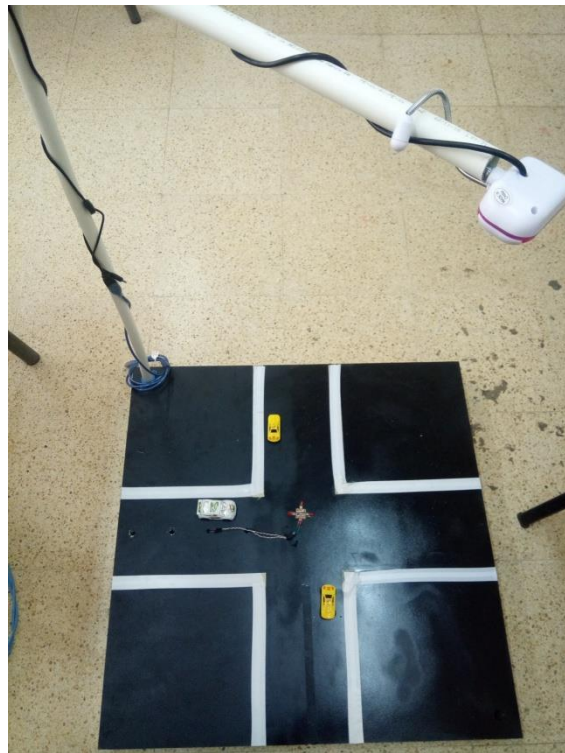
**Figure 4.6:** *Threshold*

At the end we get the HSV colorspace window and the find contour frame.

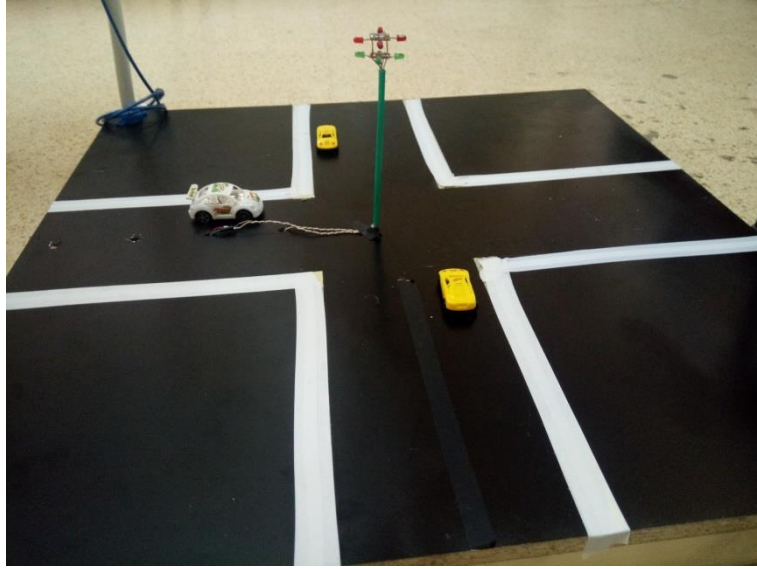


**Figure 4.7:** *HSV color spacing and find contour frame*

The whole system is represented in figures 4.8 and 4.9



**Figure 4.8:** *The system environment*



**Figure 4.9:** *The intersection of lanes*

## 4.5 Summary

In this chapter a method for estimating the traffic using OpenCV is presented. This is done by using the camera images captured from the road lanes. Each image is processed separately and the number of cars has been counted. This system guarantees that the average waiting time of the vehicle in front of traffic signal will be lesser than present traffic control systems, also the techniques and algorithms used in this project are more effective as compared to the previous systems. The advantages of this new method include such benefits as low cost, easy setup and relatively good accuracy and speed. Because this method has been implemented using OpenCV software, production costs are low while achieving high speed and accuracy.

## Conclusion

Automation systems are being preferred over the manual mode because it reduces the use of electrical energy. These automation systems play an essential role in making our daily life more comfortable and facilitate users from ceiling fans to washing machines and in other applications.

In this project, we have explored a portion of a smart city. This project has two major phases; the first stage is to design a smart street light system, which consists of switching ON and OFF the lights depending on night and object's detection. The second stage is about implementing a smart traffic light, which detect a busy and non-busy road and give green light depending on the traffic density on each road.

To implement the smart street light and smart traffic light, we needed to understand the concept of operation of IR sensors, the LDR, and the techniques of image processing. The study showed that image processing is a better technique to control the state change of the traffic light. It shows that it can reduce the traffic congestion and avoids the time being wasted by a green light on an empty road. It is also more consistent in detecting vehicle presence because it uses actual traffic images. After that, we passed by interfacing each component with the microcontroller, this led to implementing the algorithm to automate the two systems.

The contribution of this project was to create a full system that consists of the smart street light with smart traffic light; this full system can decrease the time spent on traffic jam and the high energy wasted over the world, besides it helps to increase the performance of the emergency vehicles, high road surveillance, and safety measurements

A future work to this project aims at harvesting the energy from renewable energy sources like sun and to effectively use the harvested energy for the benefit of mainly the remote villages (villagers) facing the serious power problems. As we are moving towards more advancement we require more power so use of renewable resources is useful and advantageous.

We can also have a network of cameras placed on roads intersections in the whole city. This would facilitates better control and better coordination. Also, we can add neural networks to estimate the speed of the vehicle, to recognize the cars number plate and the accidents that happen on roads.

## References

- [1] Simon Monk, “raspberry pi cookbook”, O’Reilly, December 2013
- [2] Santoshi Gupta, UtkarshaRaikar, Bhavna Mohite, Priyanka Patil, Renuka Molavade, “Image Processing Based Intelligent Traffic Control System by Using Raspberry Pi”, International Journal for Research in Applied Science and Engineering Technology, Vol.6, No.4, April.2018, accessed on march.2019, [online]. Available: <https://www.ijraset.com/files/serve.php?FID=14720>
- [3] Raspberry pi, “raspberry pi 3”, [Online], Accessed March 2019. Available: <https://www.raspberrypi.org/>
- [4] Michael Margolis, Nicolas Weldon, “Arduino cookbook”, O’Reilly, March 2011
- [5] Arduino, "Arduino UNO", [Online], Accessed march 2019 Available: <https://store.arduino.cc/arduino-uno-rev3>.
- [6] G. Benet, F. Blanes, J.E. Simó and P. Pérez, “Using infrared sensors for distance measurement in mobile robots,” Rob. Auton. Syst., vol. 40, no. 4, pp. 255-266, 2002
- [7] A. S. Jalan, “A survey on automatic street lightning system on Indian streets using Arduino,” Int. J. Innovative Res. Sci. Eng. Technol., vol. 6, no. 3, pp. 4139-4144, 2017.
- [8] H. Satyaseel, G. Sahu, M. Agarwal and J. Priya, “Light intensity monitoring & automation of street light control by Iot,” Int. J. Innovations Adv. Comput. Sci., vol. 6, no. 10, pp. 34-40, 2017.
- [9] S. A. E. Mohamed, “Smart street lighting control and monitoring system for electrical power saving by using VANET,” Int. J. Commun. Network Syst. Sci., vol. 6, pp. 351-360, 2013.
- [10] K.Vidhya, A.BazilaBanu.”Density Based Traffic Signal System”, International Journal of Innovative Research in Science, Engineering and Technology, Volume 3, Special Issue 3, March 2014.
- [11] S.Lokesh, T.Prahlad Reddy, “An Adaptive Traffic Control System Using Raspberry PI” IJESRT [Lokesh, 3(6): June, 2014] ISSN: 2277-9655, Scientific Journal Impact Factor: 3.449 (ISRA), Impact Factor: 1.852
- [12] Aman Dubey, Akshdeep, Sagar Rane.”Implementation of an Intelligent Traffic Control System and Real Time Traffic Statistics Broadcasting”.

## Appendix A: Raspberry pi 3

The Raspberry Pi is a single-board computer created by the Raspberry Pi Foundation, a charity formed with the primary purpose of reintroducing low-level computer skills to children in the UK. The aim was to rekindle the microcomputer revolution from the 1980s, which produced a whole generation of skilled programmers. Even before the computer was released at the end of February 2012, it was clear that the Raspberry Pi had gained a huge following worldwide and has now sold over 2 million units.

Several generations of Raspberry Pi have been released. The first generation Raspberry Pi 1 Model B was released in February 2012, followed by the simpler and cheaper Model A. In 2014, the Foundation released a board with an improved design, Raspberry Pi 1 Model B+. These boards are approximately credit-card sized. Improved A+ and B+ models were released a year later. The Raspberry Pi 2, which added more random-access memory, was released in February 2015. A Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015. On 28 February 2017, the Raspberry Pi Zero W was launched. Raspberry Pi 3 Model B was released in February 2016 with a 1.2 GHz 64-bit quad core processor, on-board Wi-Fi, Bluetooth and USB boot capabilities. On Pi Day 2018 the Raspberry Pi 3 Model B+ was launched with a faster 1.4 GHz processor and a three-times faster gigabit Ethernet or 2.4 / 5 GHz dual-band Wi-Fi (100 Mbit/s). Other features are Power over Ethernet (PoE), USB boot and network boot (an SD card is no longer required).

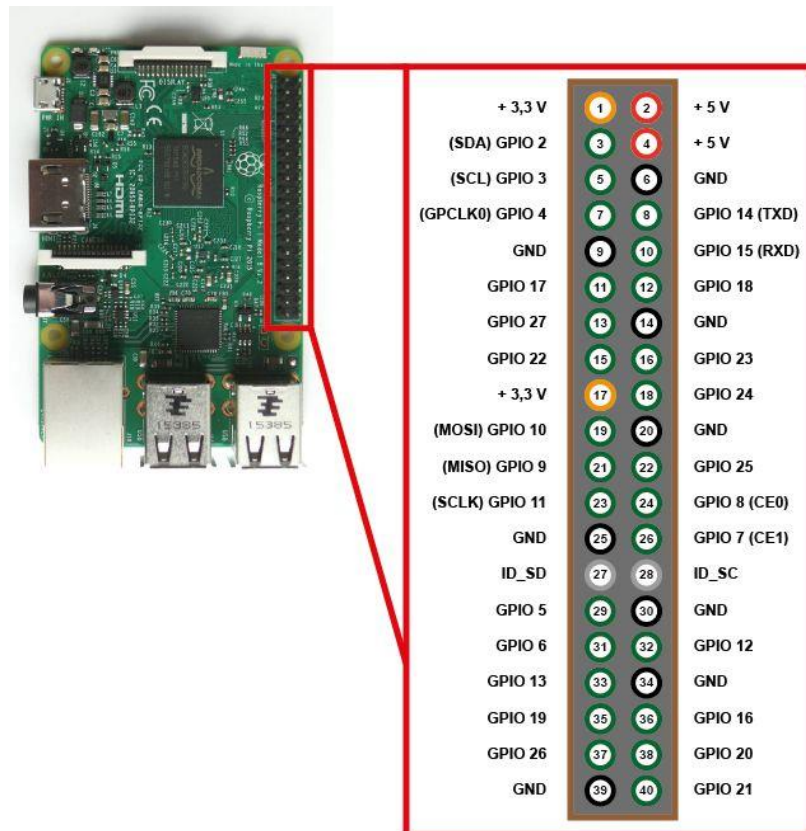
The full specifications of the raspberry pi 3 and the board connections are presented in Table A.1 and Table A.2 respectively. And the GPIO are showed in Figure A.1

**Table A.1:** *Raspberry pi 3 Technical Specifications*

Microprocessor	Broadcom BCM2837 64bit Quad Core Processor
Processor operating voltage	3.3v
Raw voltage input	5v . 2A power source
Maximum current through each I\O pin	16 mA
Maximum total current drawn from all I\O pin	54 mA
Flash memory (operating system)	16Gbyte SSD memory card
Internal RAM	1Gbyte DDR2
Clock Frequency	1.2 GHz
GPU	Dual Core IV multimedia Co-Processor. Provides Open GLES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high –profile decode. Capable of 1.5Gpixel/s, 1.5Gtexel/s or 24GFLOPS with texture filtering and DMA infrastructure
Ethernet	10/100 Ethernet
Wireless connectivity	BCM43143 (802.11 b/g/n Wireless LAN AND Bluetooth 4.1)
Operating Temperature	-40°C to +85°C

**Table A.2:** Board connections

Name	Description
Ethernet	Base T Ethernet socket
USB	2.0 four sockets
Audio output	3.5mm Jack and HDMI
Video output	HDMI
Camera connector	15-pin MIPI camera serial interface (CSI-2)
Display connector	Display Serial Interface (DSI) 15 ways flat flex cable connector with two data lanes and a clock lane.
Memory card slot	Push/Pull Micro SDIO



**Figure A.1:** Raspberry pi 3 GPIO



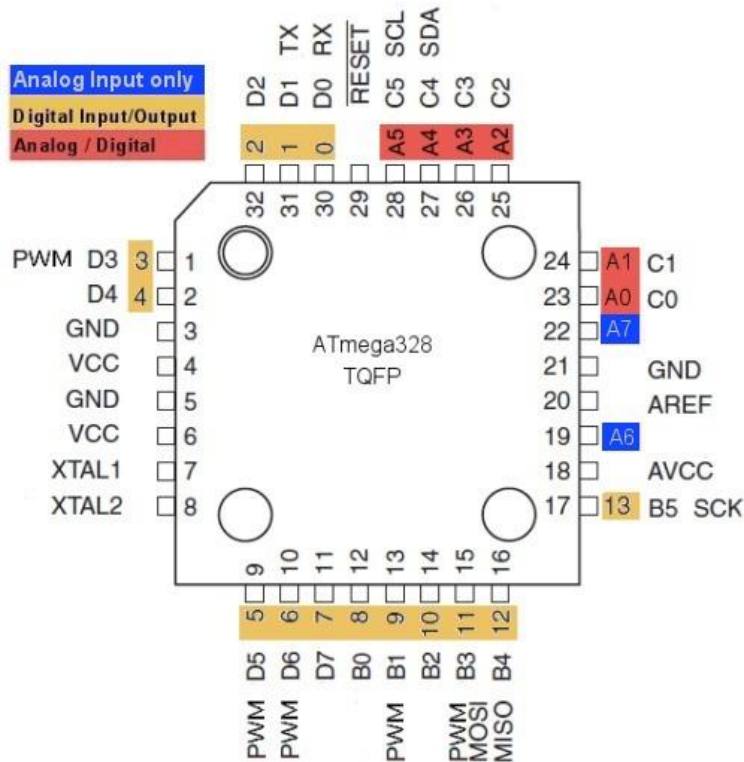
## Appendix B: Arduino Uno Board and ATmega328 Microcontroller

The Arduino Uno is a microcontroller board based on the ATmega328. Arduino is an open-source, prototyping platform and its simplicity makes it ideal for hobbyists to use as well as professionals. The Arduino Uno has 14 digital input/output pins of which 6 can be used as PWM outputs, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

The full specifications of the board are presented in table B.1 and the schematic of the microcontroller is in figure B.1

**Table B.1:** *Arduino UNO Board 328 Technical Specifications*

Microcontroller	ATmega328
Operating Voltage	5v
Input Voltage (recommended)	7-12v
Input Voltage (limits)	6-20v
Digital I\O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC current per I\O Pin	40 Ma
DC current for 3.3v Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock speed	16 MHz



**Figure B.1:** ATmega328Pinmap to Arduino Uno

