# IoT Traffic Light System for Smart City


A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering Under National University

## Submitted By

**Emam Mehedi**

**Reg. No:** 16502001012

**Session:** 2016-17

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NATIONAL UNIVERSITY, DHAKA, BANGLADESH**

**September 2022**

# APPROVAL

The project is titled **“IoT Traffic Light System for Smart City”** submitted by Reg No: 16502001012 to the development of Computer Science and Engineering. Habibullah Bahar College, Dhaka, Bangladesh has been accepted as satisfactory for partial fulfillment of the required studio style and content for a Bachelor's Degree in Computer Science and Engineering.

**Signature of Internal Examiner**

**Signature of External Examiner**

**Supervisor**
**Muhammad Muktadir**
Associate Professor
Department of CSE
Habibullah Bahar College

# DECLARATION

We hereby, declare that the work presented in this project "**IoT Traffic Light System for Smart City"** is the outcome of the investigation performed by use under the supervision of **Muhammad Muktadir**, Associate Professor, CSE Department, Habibullah Bahar College, Dhaka, Bangladesh. We also declare that no part of this project and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

**Supervised by:**

_____

**Muhammad Muktadir**

**Associate Professor**

**Department of Computer Science and Engineering**

**Habibullah Bahar College**

**Submitted by:**

**Emam Mehedi**

**Reg. No.: 16502001012**

# ABSTRACT

Four Way Traffic Lights Circuit using Arduino ESP8266 D1 Mini. In this traffic light project, we are going to design a circuit, to control traffic lights on a four-way signal. This circuit is designed by Arduino ESP8266 D1 Mini timer and a decade counter. The timer generates pulses and these pulses are fed to the ten-stage decade. Traffic signal lights are very Important to regulate vehicles and traffic on roads, simple four-way traffic light circuit is designed with timer and counter IC HC595. we know each traffic signal light setup will have three colors and representing Red for STOP, Yellow for WAIT, and Green for GO, those signals are works based on time intervals. Traffic light has proved to be an amazing way to stop the vehicular collisions and control the traffic jams in today's modern era where everyone owns the different types of vehicles.

# ACKNOWLEDGEMENTS

All praises are for the almighty Allah for giving us strength, without which I could not afford to attempt this research work.

I would like to express my sincere and heartiest gratitude to my honorable thesis supervisor **Muhammad Muktadir** Associate Professor, Dept. of Computer Science and Engineering, Dhaka City College, for his continuous motivation, guidance and keen encouragement, which helped me throughout the time of my research work. Noting is comparable to his keen advice.

I would like to think all the members of the board examiner for their precious time in understanding my work and their insightful comment. I would like to thank all of my friends for their co-operation. Finally, yet importantly, I am grateful to my parents for their continuous support and co-operation.

I also express our sincere thanks and gratitude to Prof. **Muhammad Muktadir** (Head of the Department) for their support and guidance and constant encouragement for completion of project report.

I also thankful to other respected teachers of the department, who helped me in a number of ways by providing various resources and moral support.

Any omission in this brief acknowledgment does not mean lack of gratitude.

# List of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

# Chapter 1
# Introduction

## 1.1 Introduction

Dhaka city is the fastest growing city in Bangladesh. As a result, number of cars are increasing considerably. Technology has some solutions to overcome the traffic congestion [1]. Smart cities are one idea to implement in such a city like Dhaka. Smart Traffic Light Controller [2] is not a new system, but it needs a lot of improvements. This system adds a database of the traffic congestion and gives a real time feedback about congestions in junctions. This system is part from smart city. It focuses on controlling the traffic lights by using IoT devices [2].
The system detects how long is the congestion in one direction and will give the traffic light a suitable time to let all vehicles passing the junction smoothly. The system gives the priority for passing for emergency vehicles [7].

## 1.2 Motivation for the work

In Dhaka city, traffic lights operate in a systematic way which cause a lot of delay for drivers in some cases. It gives every direction a specific time without reading the current traffic jam for each lane. Traffic flow [3] needs to be more smoothly. Connecting traffic light with a smart system to control the timing is the aim. Dhaka Municipality doesn't have records about how much is the jam in road junction. So, it will be difficult to recognize which junction needs to be replaced with a bridge. Quality of road can be improved by using smart systems.

## 1.3 Project Objectives

At first present traffic problem at any city of Bangladesh is practically observed. Then current traffic control system is critically observed to check its feasibility to solve the present problem in traffic control. Then some past research work regarding on traffic control system is studied. Finally, four system are proposed to regulate the traffic control properly at the city of its different cross junction point.
The work that I have chosen is a 4 Way Traffic controller. The basic idea behind the design is to avoid the collision of vehicles by providing appropriate signals to different directions for a limited time slot, after which the next waiting drivers will be given same treatment. In this way

a cycle will be established which will control the traffic. In the First switching-based traffic light control circuit, the traffic lights are controlled using switches i.e., when the first switches are on the green light is on in East and West and the Red Light is on in North and South. Similarly, when the second switch is on the green light is on in the North and south and the red light is on in the East and west and when the third switch is on all yellow lights are on. In the counter-based traffic light control circuit, we are going to design a circuit, to control traffic lights on a four-way signal. This circuit is designed by Arduino ESP8266 D1 Mini and a decade counter. The timer generates pulses and these pulses are fed to ten stage decade counter. The ten-stage decade counter have a memory of ten. It can count up to ten pulses. So, for every peak at clock, the counter admits it as an event and remembers it. The number of events that counter memorized outputted by corresponding pin. Traffic light control system using microprocessor and also using microcontroller is a unique traffic light controller makes simple use of C++ language programming [8] with Intel Atmel microprocessor [1]. It permits accident-free control as a separate set of signals has been assigned to a direction. For instance, if one desires to move towards north, east or west from south, he is provided a single light signal for the respective directions. Consequently, the probability of confusion leading to an accident is reduced.

This method will be applied to complete our project:

- Collection of data from books and internet.
- Collection of required components from local market and online shop.
- The concept of the design is based on using a microcontroller for processing.
- Designing a schematic for creating the circuit board.
- Programming for the microcontroller (ESP8266).
- Setting all the components in a PCB board and soldering.
- Turn on the device experimental.
- And finally, run the system using microcontroller.

## 1.4 Project Goals

Vehicles moving on a path, travelling from one point to another along that path. Traffic congestion is a condition on road networks that occurs as use increases, and is characterized by slower speeds, longer trip times, and increased vehicular queuing. The most common example is the physical use of roads by vehicles. Road traffic congestion control involves directing

vehicular and pedestrian traffic around a construction zone, accident or other road disruption, thus ensuring the safety of emergency response teams, construction workers and the general public. Road Traffic Congestion control involves controlling the congestion of vehicles or number of vehicles on a particular road and ensuring continuous vehicle flow without unpardonable delay. Traffic signals involves controlling traffic flow [3] depending on automated timers and lights.

## 1.5 Application technology

The application of technology tools and devices in the teaching and learning processes. It involves the usage, knowledge, skill and competence in the use technology in solving problem or performing specific function during and after academic activities.

## 1.6 Application server

The project requires the following software tools in order to function properly.

**Arduino IDE:**

The Arduino Integrated Development Environment is a cross-platform application that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

**Proteus Simulation:**

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

**C++Programming:**

C++ is a middle-level programming language developed by Bjarne Stroustrup starting in 1979 at Bell Labs. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This C++ tutorial adopts a simple and practical approach to describe the concepts of C++ for beginners to advanced software engineers.

**Draw Back:**

Enterprise Architect & Smart draw are used to create or draw Various types of Diagrams and Model.

# Chapter 2

# Components of Project

# Chapter 2
# Components of Project

## 2.1 Arduino D1 Mini (ESP8266)

The Arduino ESP8266 D1 Mini is an open-source microcontroller board based on the Microchip ESP8266 microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino ESP8266 D1 Mini and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

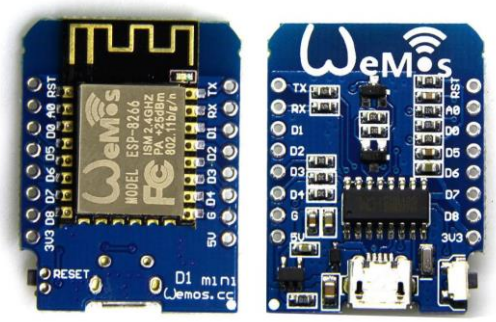


*Figure 2.1 Arduino ESP8266 D1 Mini*

The word "Nano" means "one" in Italian and was chosen to mark the initial release of the Arduino Software. The Nano board is the first in a series of USB-based Arduino boards, and it and version 1.0 of the Arduino IDE were the reference versions of Arduino, now evolved to newer releases. The ESP8266 on the board comes preprogrammed with a boot loader that allows uploading new code to it without the use of an external hardware programmer.

While the Nano communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

### 2.1.1 Background:

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003 Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language.[1]

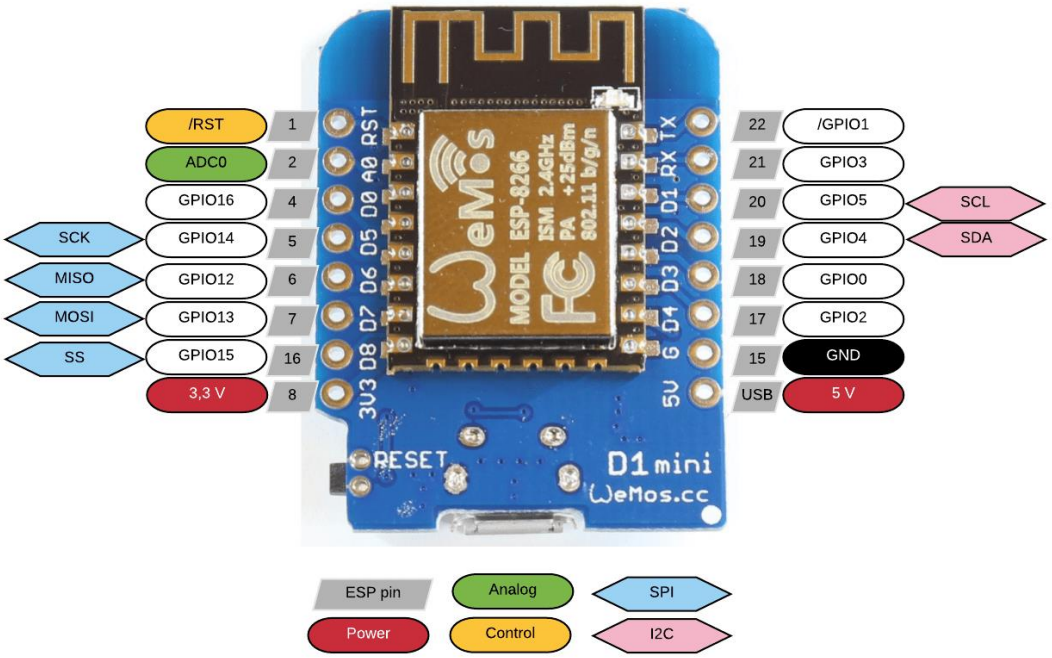


*Figure 2.2 Nano Pinout*

The project goal was to create simple, low-cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.[8] In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino. Early arduino boards used the FTDI USB-to-serial driver chip and an ATmega168.The Nano

differed from all preceding boards by featuring the ESP8266 microcontroller and an ATmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

### 2.1.2 History:

The Arduino [18] project was started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $50, a considerable expense for many students. In 2003 Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas. Casey Reas is known for co-creating, with Ben Fry, the Processing development platform. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller. In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino.

The initial Arduino core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis, but Barragán was not invited to participate.

Following the completion of the Wiring platform, lighter and less expensive versions were distributed in the open-source community.

It was estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

### 2.1.3 Hardware:

Arduino is open-source hardware [19]. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Although the hardware and software designs are freely available under copy left licenses, the developers have requested the name Arduino to be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official

product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in Arduino.

An early Arduino board with an RS-232 serial interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller [19] (ATmega8, ATmega168, ESP8266, ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features. The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012.The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed shields. Multiple and possibly stacked shields may be individually addressable via an I²C serial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the Lily Pad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default boot loader of the Arduino ESP8266 D1 Mini is the optiboot boot loader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Nano boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the Nanofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

An official Arduino ESP8266 D1 Mini R2 with descriptions of the I/O locations The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Nano provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers.

### 2.1.4 Required Software:

A program for Arduino hardware may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit AVR and 32-bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

### 2.1.5 Integrated Development Environment (IDE):

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

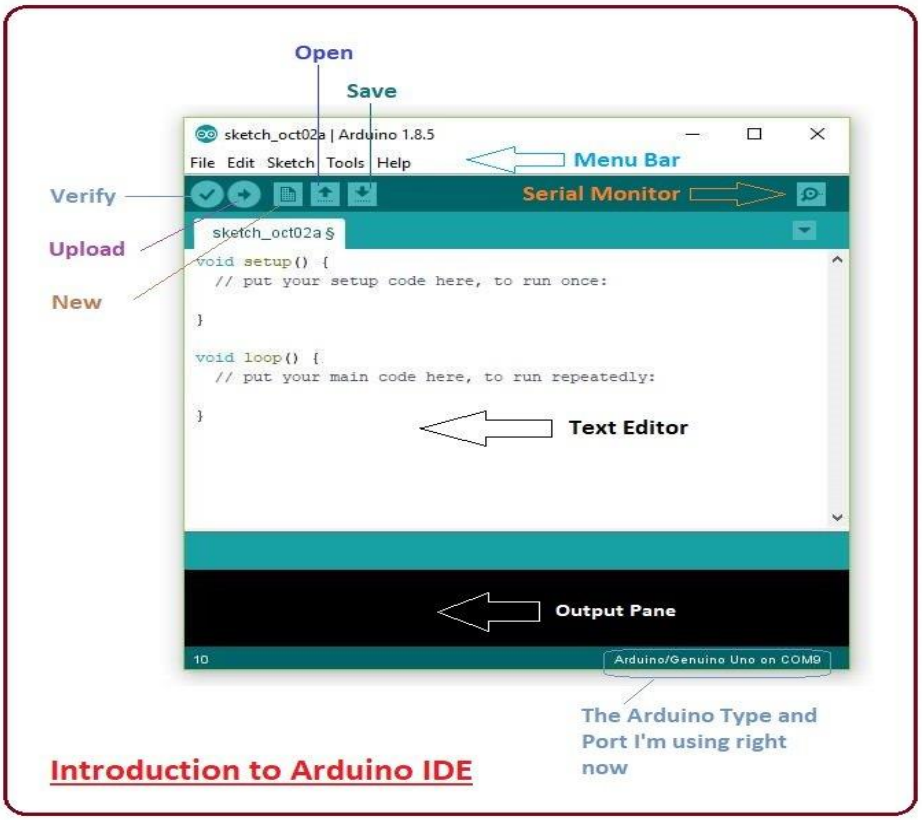


*Figure 2.3 Arduino ESP8266 D1 Mini*

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

### 2.1.6 Arduino Basic Sketch:

A sketch is a program written with the Arduino IDE. Sketches are saved on the development computer as text files with the file extension ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension ide.

**A minimal Arduino C/C++ program consists of only two functions:**

**setup():** This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch. It is analogous to the function main().

**loop():** After setup() function exits (ends), the loop() function is executed repeatedly in the main program. It controls the board until the board is powered off or is reset. It is analogous to the function while(1).

### 2.1.7 Blink example:

Most Arduino boards contain a light-emitting diode (LED) and a current limiting resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program used by beginners, akin to Hello, World!, is "blink", which repeatedly blinks the on-board LED integrated into the Arduino board. This program uses the functions pin Mode(), digital Write(), and delay(), which are provided by the internal libraries included in the IDE environment. This program is usually loaded into a new Arduino board by the manufacturer.

### 2.1.8 Libraries:

The open-source nature of the Arduino project has facilitated the publication of many free software libraries that other developers use to augment their projects.

- 3.4.9 Technical specifications

- Microcontroller: Microchip ESP8266
- Operating Voltage: 5 Volts
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by boot loader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

### 2.1.8 Pins:

**General pin functions:**

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it's off.

VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Reset: Typically used to add a reset button to shields which block the one on the board.

Special pin functions

Each of the 14 digital pins and 6 analog pins on the Nano can be used as an input or output, using pin Mode(), digital Write(), and digital Read() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Nano has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function.

In addition, some pins have specialized functions:

Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

PWM (pulse-width modulation): 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analog Write() function.

SPI (Serial Peripheral Interface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

TWI (two-wire interface) / I²C: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

AREF (analog reference): Reference voltage for the analog inputs.

### 2.1.9 Communication:

The Arduino/Genuino Nano has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ESP8266 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows serial communication on any of the Nano's digital pins.

### 2.1.10 Applications:

I. Arduboy, a handheld game console based on Arduino
II. Arduinome, a MIDI controller device that mimics the Monome
III. Ardupilot, drone software and hardware
IV. ArduSat, a cubesat based on Arduino.
V. C-STEM Studio, a platform for hands-on integrated learning of computing, science, technology, engineering, and mathematics (C-STEM) with robotics.
VI. Data loggers for scientific research.
VII. OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars
VIII. OpenEVSE an open-source electric vehicle charger
IX. XOD, a visual programming language for Arduino

## 2.2 Adaptor

12V 5A Micro Power Adpater. For using with Arduibo /Raspberry Pi 3 Model A+/B/B+/Zero and running any other high current devices.

Specifications:

Input: 100V - 240V AC, 50Hz/60Hz

Output: 5V 2A


Figure 2.4 Adapter

## 2.3 74HC595

The 74HC595 is an 8-bit Serial In – Parallel Out Shift Register [9], i.e., it can receive(input) data serially and control 8 output pins in parallel.

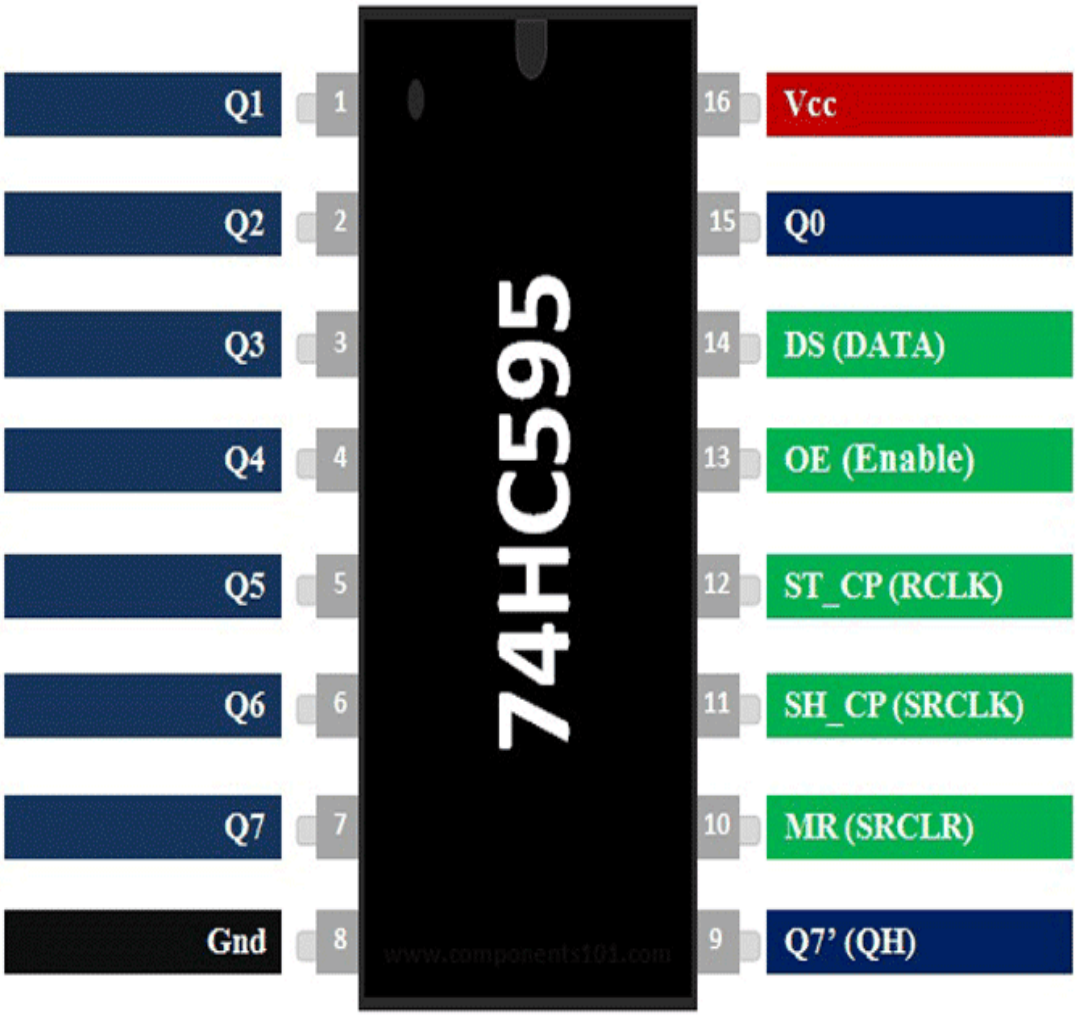


Figure 2.5 HC595

This comes in very handy where do not have enough GPIO [9] pins on our MCU/MPU to control the required number of outputs. It is often used in projects where relatively a large number of LED's has to be controlled through the Microcontroller. It can also be used to interface LCD screen since they can act as the data bit for the LCD displays. It can also be used to control 5V loads like relays through a 3.3V microcontroller since the high-level voltage is only 3.15. So, if you are looking for an IC to save on your GPIO pins on the Microcontroller, then this IC might be the right choice for you.

The **74HC595 shift register [9]** is commonly used with microcontrollers or microprocessors to expand the GIPO functionalities. It requires only 3 pins connected to the MCU, which are Clock, Data, and Latch. It has a wide operating voltage from 2V to 6V. An application circuit of the IC is shown below:

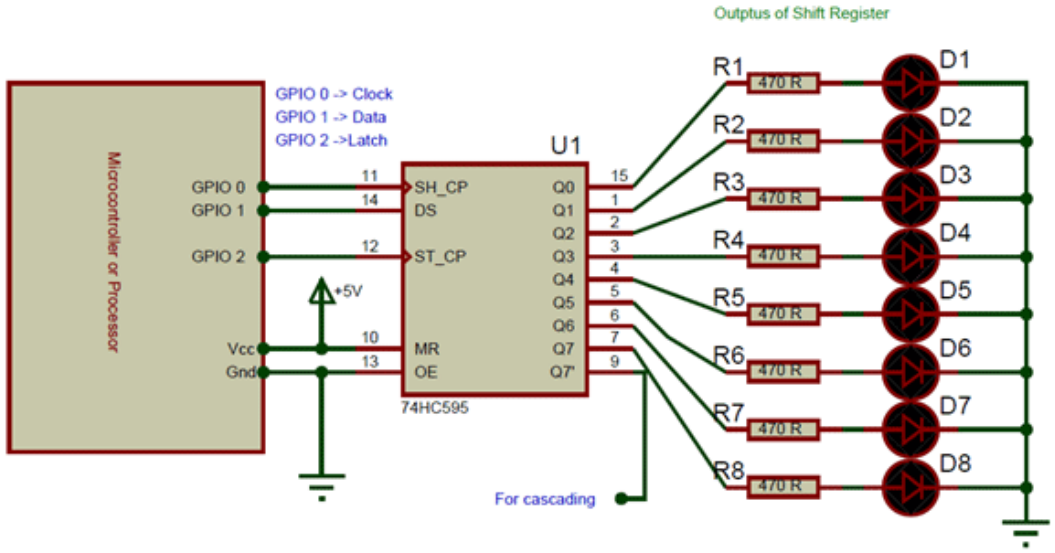


Figure 2.6 Working Circuit

The pins 11, 14 and 12 are connected to the GPIO pins of the Microcontroller. In which pin 11 is the clock which sends a constant pulse to keep timing. The pin 14 is Data which actually sends the Data about which output pins has to stay low and which should go high. The Pin 12 is the Latch which updates the received the data to the output pins when made high, this pin can also be permanently held high. The below image will help you understand better.

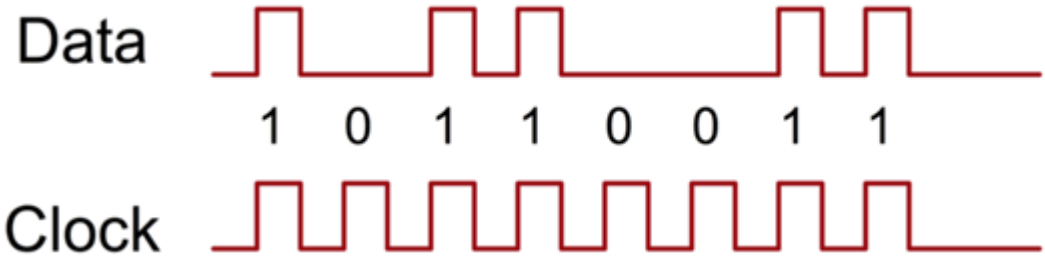


Figure 2.7 HC595 Output

As you can see the clock is continues train of pulses, and the data goes high only at the respective place where the output has to get high. Here for example the binary value 0b10110011 is passed to the microcontroller. The pin Master reset (MR) is used to reset the outputs, when not in use it is held high to VCC. similarly, the pin should be held low when not in use.

Another important advantage of the **74hc595 IC** is that it can be cascaded to control more than 8 outputs. To do this, we use the Q7' (pin 9), this pin should be connected to the data pin of the second 74HC595 IC. This way the first 8-bit sent from MCU will be used by the 1st IC and the second 8-bit will be used by the 2nd IC.

**Applications**

- Expand the GPIO pin on a MCU/MPU
- LED Matrix/Cube Projects
- Interface LCD
- Cascading applications
- High logic level controller

## 2.4 Push button:

Figure 2.8 Push Button

In electronics and technology, a reset button is a button that can reset a device. Reset buttons are found on circuit breakers to reset the circuit. This button can cause data corruption so this button often doesn't exist on many machines. Usually, in computers and other electronics devices, it is present as a small button, possibly recessed in to the case or only accessible by a pin or similar thin object, to prevent it being pressed accidentally.

### 2.5.1 Resistor

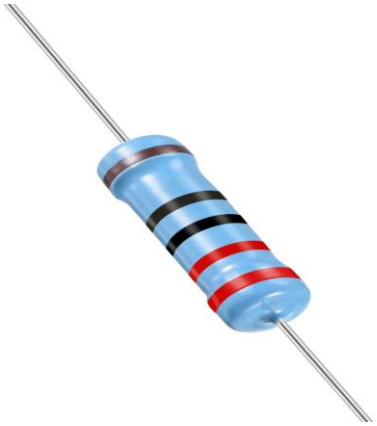

*Figure 2.9 Resistor*

A resistor (also known as an electrical resistor) is defined as a two-terminal passive electrical element that provides electrical resistance to current flow. Resistance is a measure of the opposition to the flow of current in a resistor. The larger a resistor's resistance, the greater the barrier against the flow of current. There are many different types of resistors, such as a thermistor.

In an electrical and electronic circuit, the primary function of a resistor is to "resist" the flow of electrons, i.e., electric current. That is why it is called a "resistor".

Resistors are passive electrical elements. This means they cannot deliver any energy to the circuit, and instead, they receive energy and dissipates it in the form of heat as long as a current flowing through it.

Different resistors are used in an electrical and electronic circuit to limit the current flow or produce voltage drops. Resistors are available in many different resistance values from fractions of Ohm (Ω) to millions of Ohms.

According to ohm's law, the voltage (V) across a resistor is directly proportional to the current (I) flowing through it. Where the resistance R is the constant of proportionality. In an electrical and electronic circuit, resistors are used to limit and regulate current flow, divide voltages, adjust signal levels, bias active elements, etc.

For example, many resistors are connected in series used to limit the current flowing through the light-emitting diode (LED).[9]

## 2.5.2 Potentiometer

Figure 2.10 Potentiometer

Potentiometers are manually adjustable variable resistors with three terminals. Two terminals are connected to the ends of the resistive element, while the third terminal is connected to a sliding contact called Wiper. So, by rotating the wiper, the resistance of the Potentiometer can be adjusted to its maximum value. The position of the wiper determines the output voltage of the Potentiometer. Potentiometer is commonly called as "Pot meter" or simply "Pot". "Log (Logarithmic) Pots" are used as volume control and "Lin (Linear) Pots" are used for other applications. Different materials are used as resistive material in Pots. This may be Carbon, Cermet, Wire wound, Conductive plastic or metal film. Usually, the value of the Pot is printed on its body. But the miniature "Trim pots" have codes to represent its value. For example, 10K = 103 etc. Checkout the different types of Potentiometers.

## 2.5.3 Buzzer

Buzzer is a kind of electronic sound receiver with integrated structure. It is widely used as a voice device in electronic products like computers, printers, copying machines, alarm apparatus, electronic toys, auto electronic devices, telephones, etc. In this experiment, we are going to use micro: bit to drive buzzer and make its sound circulate between high frequency and low frequency just like alarm song. And we will present its sound frequency on micro: bit with bar chart format.

*Figure 2.11 Buzzer*

## 2.6 PCF8574

The PCF8574 [17] device is an 8-bit I/O expander for the two-line bidirectional bus (I2C) is designed for 2.5-V to 5.5- V VCC operation. It provides general-purpose remote I/O expansion for most micro-controller families via the I2C interface (serial clock, SCL, and serial data, SDA, pins). The PCF8574 device provides an open-drain output (INT) that can be connected to the interrupt input of a microcontroller. An interrupt is generated by any rising or falling edge of the port inputs in the input mode. After time, tiv, INT is valid. Resetting and reactivating the interrupt circuit is achieved when data on the port is changed to the original setting or data is read from, or written to, the port that generated the interrupt. Resetting occurs in the read mode at the acknowledge bit after the rising edge of the SCL signal, or in the write mode at the acknowledge bit after the high-to-low transition of the SCL signal. Interrupts that occur during the acknowledge clock pulse can be lost (or be very short) due to the resetting of the interrupt during this pulse. Each change of the I/Os after resetting is detected and, after the next rising clock edge, is transmitted as INT. Reading from, or writing to, another device does not affect the interrupt circuit.

*Table 2.1 Address Pinout [17]*

| Address of PCF8574 | | | | | | | | 7-bit hexadecimal address without R/W |
|---|---|---|---|---|---|---|---|---|
| A6 | A5 | A4 | A3 | A2 | A1 | A0 | R/W | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | - | 20h |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | - | 21h |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | - | 22h |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | - | 23h |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | - | 24h |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | - | 25h |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | - | 26h |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | - | 27h |

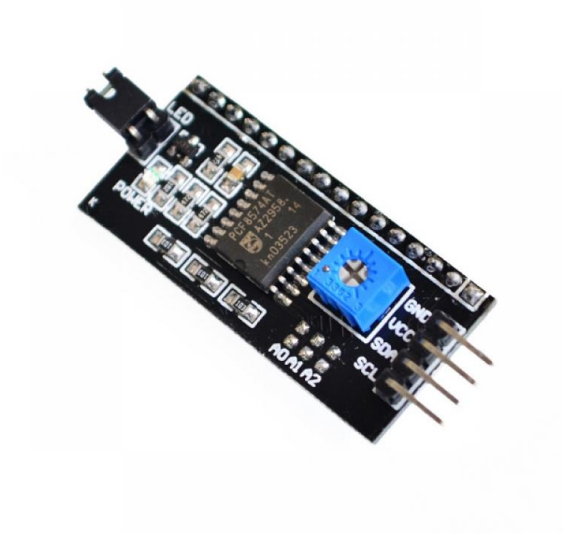


*Figure 2.12 LCD 1602*

## 2.7 LED Light

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device. Appearing as practical electronic components in 1962, the earliest LEDs emitted low-intensity infrared (IR) light. Infrared LEDs are used in remote-control circuits, such as those used with a wide variety of consumer electronics. The first visible-light LEDs were of low intensity and limited to red. Modern LEDs are available across the visible, ultraviolet (UV), and infrared wavelengths, with high light output.

Figure 2.13 LED

## 2.8 ESP8266-12E


*Figure 2.14 ESP8266-12E*

**ESP-12E [6]** is a miniature **Wi-Fi module** present in the market and is used for establishing a wireless network connection for microcontroller or processor. The core of ESP-12E is **ESP8266EX [16]**, which is a high integration wireless SoC (System on Chip). It features ability to embed Wi-Fi capabilities to systems or to function as a standalone application. It is a low-cost solution for developing IoT applications.

### 2.8.1 Pin Configuration

The ESP-12E module has twenty-two pins and we will describe function of each pin below.

*Table 2.2 ESP8266 Pinout [16]*

| Pin | Name | Description |
|---|---|---|
| 1 | RST | Reset Pin of the module |
| 2 | ADC | Analog Input Pin for 10-bit ADC (0V to1V) |
| 3 | EN | Module Enable Pin (Active HIGH) |
| 4 | GPIO16 | General Purpose Input Output Pin 16 |
| 5 | GPIO14 | General Purpose Input Output Pin 14 |
| 6 | GPIO12 | General Purpose Input Output Pin 12 |
| 7 | GPIO13 | General Purpose Input Output Pin 13 |
| 8 | VCC | +3.3V Power Input |
| 9 | CS0 | Chip selection Pin of SPI interface |
| 10 | MISO | MISO Pin of SPI interface |
| 11 | GPIO9 | General Purpose Input Output Pin 9 |
| 12 | GPIO10 | General Purpose Input Output Pin 10 |
| 13 | MOSI | MOSI Pin of SPI interface |
| 14 | SCLK | Clock Pin of SPI interface |
| 15 | GND | Ground Pin |
| 16 | GPIO15 | General Purpose Input Output Pin 15 |
| 17 | GPIO2 | General Purpose Input Output Pin 2 |
| 18 | GPIO0 | General Purpose Input Output Pin 0 |
| 19 | GPIO4 | General Purpose Input Output Pin 4 |
| 20 | GPIO5 | General Purpose Input Output Pin 5 |
| 21 | RXD0 | UART0 RXD Pin |
| 22 | TXD0 | UART0 TXD Pin |

### 2.8.2 Features and Electrical Characteristics

- Wireless Standard: IEEE 802.11 b/g/n protocol
- Frequency Range: 2.412 - 2.484 GHz
- Serial Transmission: 110 - 921600 bps, TCP Client 5
- SDIO 2.0, SPI and UART Interface available
- PWM available
- One ADC channel available
- Programmable GPIO available
- Wireless Network Type: STA / AP / STA + AP
- Security Type: WEP / WPA-PSK / WPA2-PSK
- Encryption Type: WEP64 / WEP128 / TKIP / AES
- Network Protocol: IPv4, TCP / UDP / FTP / HTTP
- Operating Voltage: 3.3V
- Maximum current allowed to draw per pin: 15mA
- Power down leakage current of < 10uA
- Integrated low power 32-bit MCU
- Onboard PCB Antenna
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW
- Operating Temperature: -40ºC to +125 ºC

### 2.8.3 Overview of ESP-12E

ESP-12E is a member of 'ESP-XX' series. Although all of them are based on ESP8266 SoC [2] they differ in on output pins, flash memory and antenna type. These modules numbered from **ESP-01 to ESP-15** and are best in performance and cost. Many engineers use these modules to setup a wireless communication between two applications. For data sharing and IoT you will find these modules Ideal.

### 2.8.4 How to use the ESP-12E

This module does not have complex circuitry or programming so using this module is very easy. We will construct a simple application circuit for understanding the module working.

**Steps for setting up a simple application circuit:**

- Connect positive +3.3V power to the module.
- Interface module to a microcontroller or ARDUINO [1] using UART (Connect RXD of ESP to RXD of µC & TXD of ESP to TXD of µC).
- Download the libraries for the module from the internet. For ARDUINO, the IDE will have pre-installed libraries. If you do not have them just update the libraries from ARDUINO website.
- Write the program for setting up the baud rate and data exchange.
- Send data to the module for transmitting through Wi-Fi or Receive data from the module that was transmitted via Wi-Fi.
- There is another way for setting up the module which is to bypass microcontroller and directly connect the module to PC using FTDI. After interface you can use serial monitor to communicate with the module.

## 2.9 AMS1117 3.3V


*Figure 2.15 AMS1117*

The AMS1117 [9] series of adjustable and fixed voltage regulators are designed to provide up to1A output current and to operate down to 1V input-to-output differential. The dropout voltage of the device is guaranteed maximum 1.3V, decreasing at lower load currents. On-chip trimming adjusts the reference voltage to 1.5%. Current limit is set to minimize the stress under overload conditions on both the regulator and power source circuitry. The AMS1117 devices are pin compatible with other three-terminal SCSI regulators and are offered in the low-profile surface mount SOT-223 package, in the 8L SOIC package and in the TO-252 (DPAK) plastic package.

# Chapter 3

# Requirements & Analysis

# Chapter 3
# Requirements & Analysis

## 3.1 Introduction

Our project aims to eliminate the delay on roads by reducing traffic on road automatically using embedded system. The timing of signal is adjusted according to traffic level on each road. The road which has level more than other road then this road assign green signal and for others have red is assign. It is also providing the additional functionality of release the emergency vehicle on its occurrence that means when emergency vehicle occurs. This circuit is designed by Arduino ESP8266 D1 Mini [1] timer and a decade counter. The timer generates pulses and these pulses are fed to the ten-stage decade. Traffic signal lights are very Important to regulate vehicles and traffic on roads, simple four-way traffic light circuit is designed with timer and counter IC HC595 [9]. we know each traffic signal light setup will have three colors and representing Red for STOP, Yellow for WAIT, and Green for GO, those signals are works based on time intervals. Traffic light has proved to be an amazing way to stop the vehicular collisions and control the traffic jams in today's modern era where everyone owns the different types of vehicles.[3]

## 3.2 Design and Implementation Smart Traffic Light

This is a summary of the above article done by Kareem; Jabbar. In the article, Authors begin by talking about traffic congestion increasing. The main objective is reducing the waiting time in emergency situations. Two parts are used, software part and hardware part. The software part gets signals from the other part to control traffic lights by using two methods. The first one is by calling the system to turn the traffic light to green in the required lane. The second one is by using IR sensors to detect the specific vehicle (like ambulance vehicle). Advantages: Open Lane for specific vehicles in emergency cases. Two ways of controlling the system (GSM and IR sensors). Disadvantages: Calling the system using GSM technology may fail regarding to no balance or weak coverage area. Using IR sensors to detect a specific car before reaches the junction is very difficult to implement and may cause some malfunctions in the system. In closing, the aim of the project is to reduce the crowding in the emergency situations. [5]

## 3.3 Ambulance with Automatic Traffic Light Control

The paper focuses on emergency conditions to save human's life. Reaching the hospital without delay caused by the ambulance at the road intersections. The aim is to let the ambulance arrives hospital as fast as possible. The system consists of two parts. The first one is related to patient's status. Basic information about the patient is send to the hospital for the treatment. The second part is related to traffic light. Ambulance driver clear the path by controlling traffic light. My project involved mainly in Arduino boards. This research paper is focusing in Arduino as a popular SBC around the world. Authors explained about its family, architecture and shields used on it. Also, an explanation of how to configure An Arduino board using IDE. It will be very useful for my project. The authors use IoT within the existing internet infrastructure to save patient's life while using ambulance vehicle. The project controls the traffic signals at the needed time. Arduino Uno board, Arduino Ethernet shield board, different sensors are used as hardware components. Arduino Software, Arduino Environment and Cayenne app are used as software components. [5]

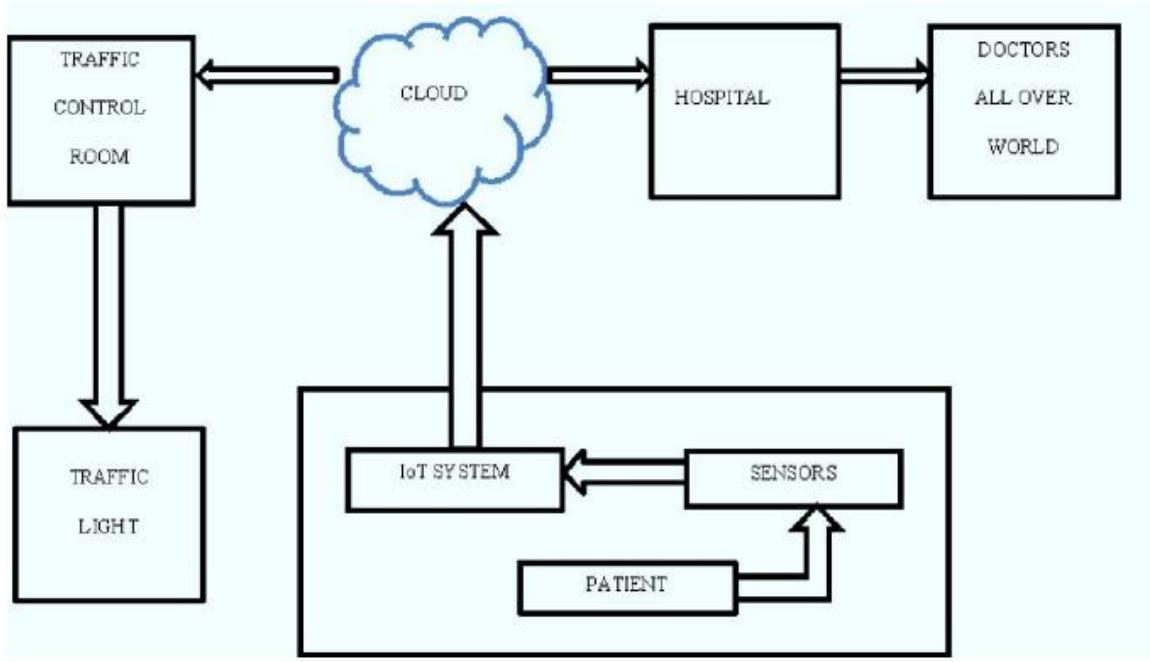


*Figure 3.1 IoT Block*

## 3.4 Innovative System for Road using IoT

The project is mainly about traffic congestion as a worldwide problem. The project is for Bangladesh as it is the most populous country. The traffic lights there are based on fixed time concept. Trying to apply smart devices on the road with a low-cost innovative technology to

solve the problem. The used devices are ultrasonic sensors, light sensors motion sensors camera and IoT devices [2]. The devices use the exciting network infrastructure.

The system uses cameras for detecting traffic congestion and controlling signals. There is one camera for each direction fixed alongside the traffic light. The camera takes sequence of images and analyze the image to calculate the number of vehicles and control the traffic light.

# Chapter 4

# Design

# Chapter 4
# Design

## 4.1 System Architectural Design

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.[4]

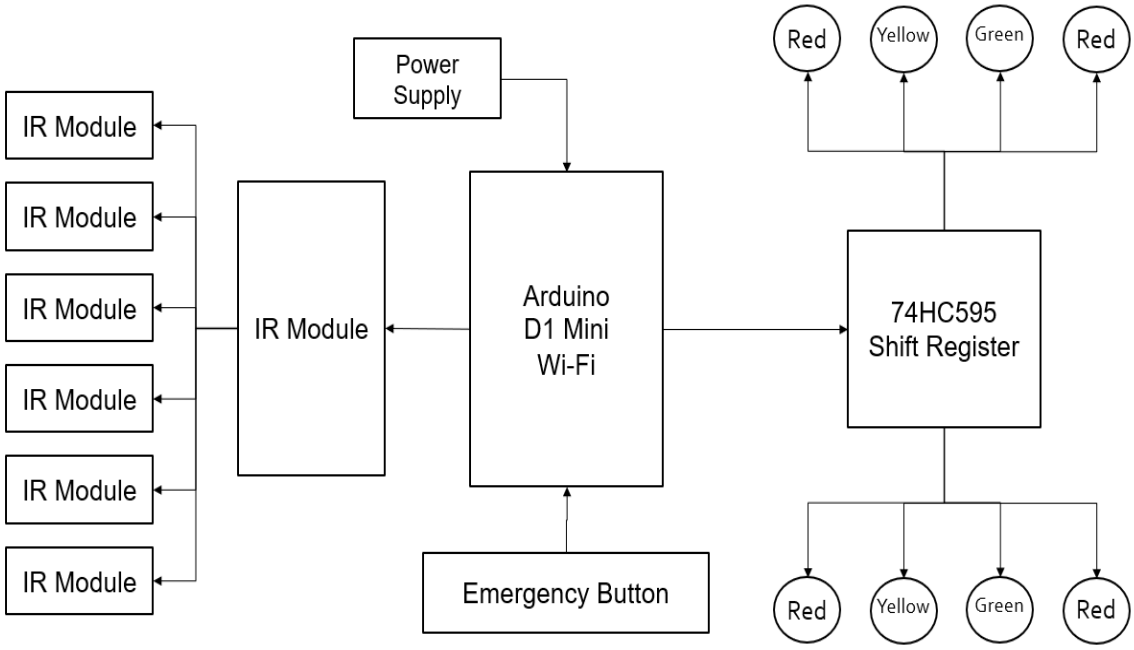


*Figure 4.1 Block Diagram*

## 4.2 Circuit Schematic Design

Schematic design is a rough construction drawing that offers a general overview of a project's basic features and construction cost estimates, allowing you to determine if your concept fits within the project budget. With schematic designs, your team turns ideas into physical drawings that you can look at and edit to help craft your construction project, and prepare you for the next phase of your architectural plan. Schematic design is the first phase of the architectural design process, which includes design development, construction documents, bidding, and construction administration.[9]

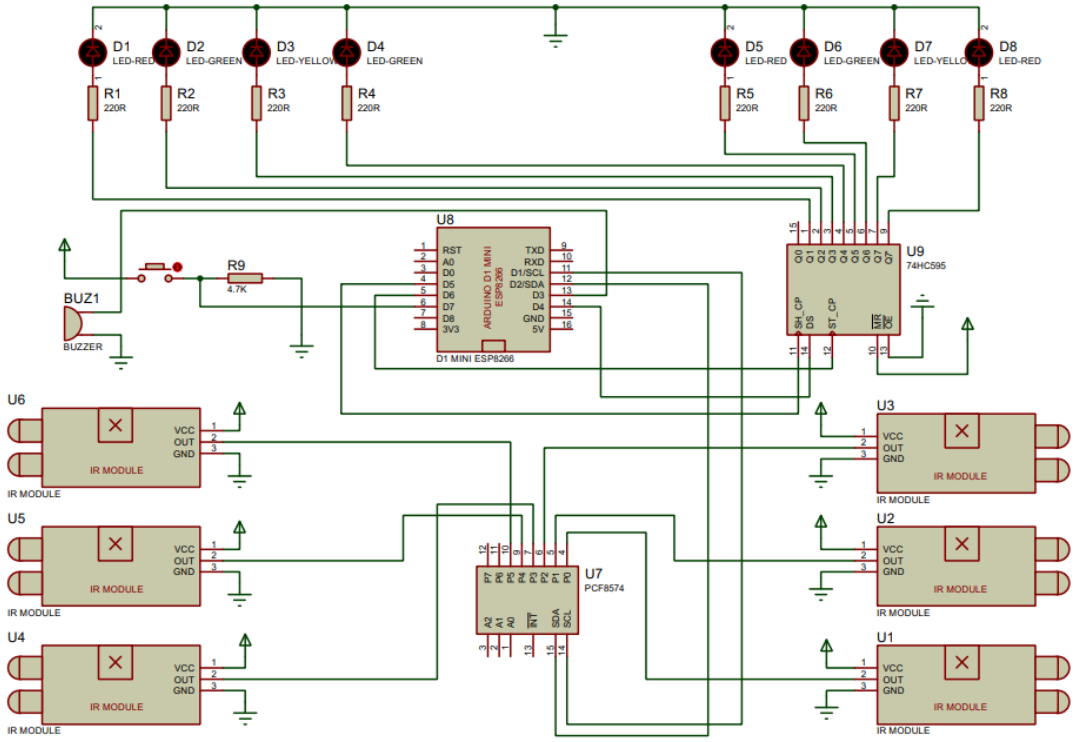


*Figure 4.2 Schematic Design*

## 4.3 Flowchart

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan. It's a common process analysis tool and one of the seven basic quality tools.

Elements that may be included in a flowchart [7] are a sequence of actions, materials or services entering or leaving the process (inputs and outputs), decisions that must be made, people who become involved, time involved at each step, and/or process measurements.

### 4.3.1 WHEN TO USE A FLOWCHART

- To develop understanding of how a process is done
- To study a process for improvement
- To communicate to others how a process is done
- When better communication is needed between people involved with the same process
- To document a process
- When planning a project

### 4.3.2 FLOWCHART BASIC PROCEDURE

Once you've determined that a flowchart is a right tool for the job, continue with these steps:

**1. Identify tasks:** The process may seem straightforward from a broad perspective. But, if you're not the one in the weeds, then rely on the team that is to help you outline the steps and tasks with you.

**2. Compile the necessary information:** You should know the exact steps, the variables and events that may cause the process to deviate, and also who is responsible for each step along the way.

**3. Double-check the process:** Gather critical stakeholders to review this outline of events to ensure the information is accurate.

**4. Create the flowchart:** Now, it's time to get to drawing! You can use the basic symbols mentioned above or go into more detail with BPMN, or Business Process Modeling and Notation.

As previously mentioned, you have options as to how to create your process flowchart. You can choose from:

**Drawing by hand** - you'll need a pen and paper.

**Online software -** use graphing software to make a flowchart digital

**Business process management software -** create digital flowcharts that can then go a step further and be analyzed or even executed by the system.

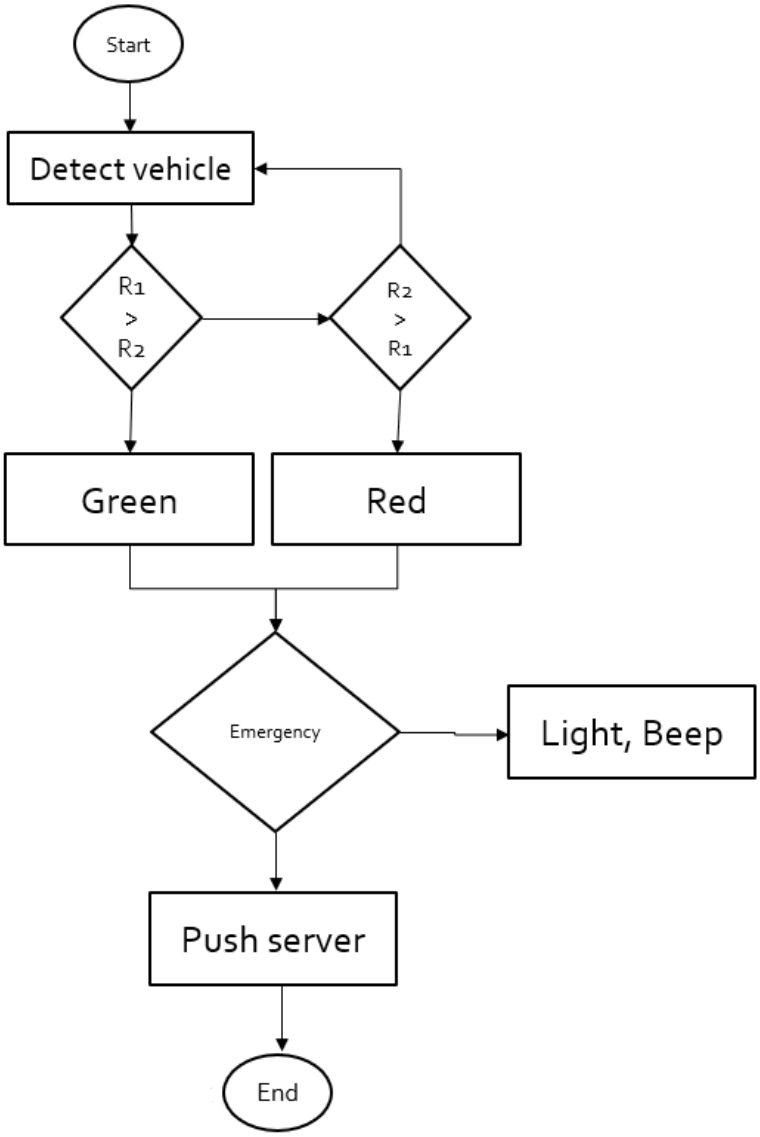


*Figure 4.3 Flowchart*

# Chapter 5

# Implementation

# Chapter 5
# Implementation

## 5.1 Introduction

A step-by-step evaluation process has been executed to validate three main functionalities: (i) map processing and selection of message board location, (ii) vehicle data collection and processing, and (iii) dashboards.

## 5.2 System comparison

The proposed system model is compared with existing systems discussed in the related works section based on functional aspects. The functionality is evaluated with respect to research objectives. The real-time monitoring of traffic scenarios and reach out to the public through traffic updates is the main objective of the proposed system. Hence, the three parameters 'real-time monitoring', 'real-time signal controlling', 'real-time public updates' are defined as parameters to evaluate the functionality. Similarly, another objective is to support message broadcasting by administrators on any unusual traffic incidents and monitoring dashboards for administrators, this leads to the fourth parameter 'administrator updates. The final parameter is 'System scalability' indicates the integration of additional hardware/software components. To evaluate the proposed system models, the defined parameters are assigned 3 values between 0 and 5: (i) 'not addressed' = 0, 'partially addressed' = 3, 'completely addressed' = 5. The 'partially addressed' and 'completely addressed' parameters are defined to differentiate the intensity of the functional requirements covered in the study. The 'partially addressed' parameter indicates that the study support only minimum functionality, whereas the 'completely addressed' parameter, indicates the support to different functional requirements. For eg, the functionality of real-time traffic update is only through traffic signals in studies 3, 4, and 5, whereas study 1 and 2 also provide the update on the intensity of traffic congestion. Similarly, study 5 provides a few kinds of administrative reports through the application layer and the system design uses fixed surveillance cameras; however, it does not provide a real-time dashboard. Considering this, the 'partially addressed' score has been assigned to the system for administrator updates and system scalability parameters.

## 5.3 PCB Design

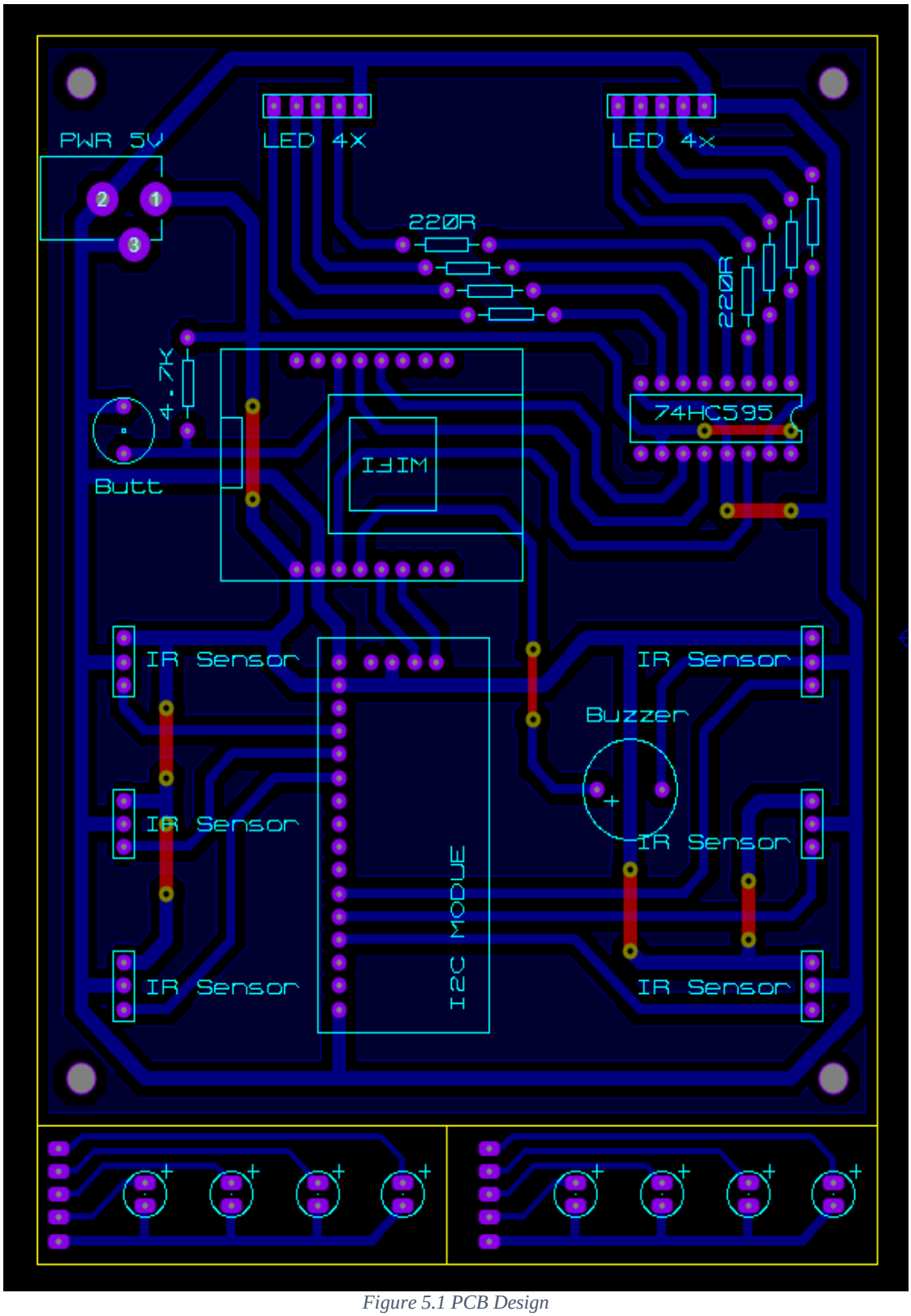


*Figure 5.1 PCB Design*

## 5.4 Real Hardware

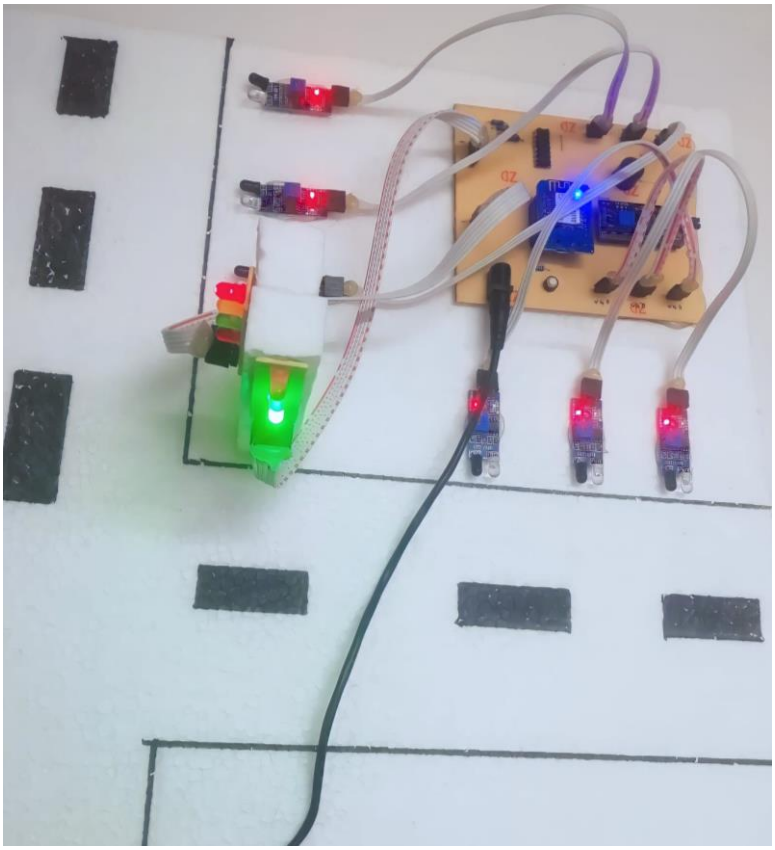

*Figure 5.2 Hardware full*

## 5.5 Discussion

The study proposed a system model for real-time traffic updates in an IoT context to assist drivers. The system has three main functionalities: (i) map processing, (ii) traffic data collection, (iii) visualization, and storage. The system uses an existing free wiki map to collect the road information and extract the message unit location. The data collection layer is built on magnetic sensors to detect vehicles and estimate the length of the vehicle and road occupancy. The feasibility of the system is demonstrated through a prototype. Each of the modules is evaluated individually, and the results of the evaluation are satisfactory in terms of accuracy. The system architecture establishes a WiFi-based communication model between sensors and the IoT platform. The roadside magnetic sensors and the micro-controller follows a client-server WiFi communication. The sensor nodes A and C act as client and sensor nodes B and D as the server to estimate vehicle length at entry and exit points of the road segment. The sensor node B acts as a server and sensor node A acts as a client while estimating the road occupancy measure. The sensor node B sends the message to the roadside unit through WiFi and to the real-time data to the IoT platform. As part of the prototype, the Nodemcu microcontroller is

used and a more powerful unit can be used in real-time. The central traffic management system can also connect to the IoT platform in the same way, which is beyond the scope of this work. This research selected a magnetic sensor for prototype implementation due to its low cost and availability. In an end-to-end real-time implementation, the readymade PCBs (e.g., iVCCS, PRS, etc.) can be used instead of magnetic sensors as all of them are already proven and have shown high accuracy in vehicle detection, speed estimation, magnetic length estimation, and physical length estimation/vehicle classification.

# Appendix

**Source Code:**

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <Adafruit_PCF8574.h>

#define SH D5
#define ST D6
#define DS D4
#define butt D7
#define buzz D3

#define IR1  0
#define IR2  1
#define IR3  2
#define IR4  4
#define IR5  5
#define IR6  6

#define eled1 0
#define gled1 1
#define yled1 2
#define rled1 3
#define eled2 4
#define gled2 5
#define yled2 6
#define rled2 7

#define ROAD1 0
#define ROAD2 1
#define RED    1
#define YELLOW 2
#define GREEN  3
#define EMERG  4
```

```
#define input(x) pcf.digitalRead(x)

const char *ssid = "Imran";
const char *pass = "12345678";
const char *link0 = "http://esinebd.com/projects/traffic/update_machine.php";
const char *link1 = "http://esinebd.com/projects/traffic/stat_machine.php";

WiFiClient client;
HTTPClient http;
Adafruit_PCF8574 pcf;

bool flip;
bool timeMode, update, serverMode, emrg1, emrg2;
byte density1, density2, pDensity1, pDensity2;
byte sec;
long pMs, prevMs;

void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_AP_STA);
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  pinMode(SH, OUTPUT);
  pinMode(ST, OUTPUT);
  pinMode(DS, OUTPUT);
```

```
  pinMode(butt, INPUT);
  pinMode(buzz, OUTPUT);

  Wire.begin(D2, D1);
  while (!pcf.begin(0x27, &Wire)) {
    delay(500);
    Serial.print("_");
  }
  Serial.println();
  Serial.println("PCF OK.");
  for (byte p = 0; p < 8; p++) {
    pcf.pinMode(p, INPUT);
  }

  traffic(ROAD1, GREEN);
  traffic(ROAD2, GREEN);
}

void loop() {
  if (!digitalRead(butt)) {
    digitalWrite(buzz, 1);
    timeMode = !timeMode;
    Serial.println((String)"TimerMode: " + timeMode);
    if (timeMode == 1) delay(1000);
    else delay(500);
    digitalWrite(buzz, 0);
  }

  if (!input(IR1) && !input(IR2) && !input(IR3)) density1 = 3;
  else if (!input(IR1) && !input(IR2) && input(IR3)) density1 = 2;
  else if (!input(IR1) && input(IR2) && input(IR3)) density1 = 1;
  else if (input(IR1) && input(IR2) && input(IR3)) density1 = 0;

  if (!input(IR4) && !input(IR5) && !input(IR6)) density2 = 3;
```

```
else if (!input(IR4) && !input(IR5) && input(IR6)) density2 = 2;
else if (!input(IR4) && input(IR5) && input(IR6)) density2 = 1;
else if (input(IR4) && input(IR5) && input(IR6)) density2 = 0;

if (pDensity1 != density1) {
  update = 1;
  pDensity1 = density1;
}
if (pDensity2 != density2) {
  update = 1;
  pDensity2 = density2;
}

if (serverMode == 0) {
  if (timeMode == 1) {
    if (millis() - pMs >= 1000) {
      if (sec == 3) {
        traffic(ROAD1, GREEN);
        traffic(ROAD2, RED);
      }
      else if (sec == 10) {
        traffic(ROAD1, YELLOW);
        traffic(ROAD2, YELLOW);
      }
      else if (sec == 13) {
        traffic(ROAD1, RED);
        traffic(ROAD2, GREEN);
      }
      else if (sec == 20) {
        traffic(ROAD1, YELLOW);
        traffic(ROAD2, YELLOW);
      }
      sec++;
      if (sec == 21) sec = 0;
```

```
      pMs = millis();
    }
  }
  else {
    if (update == 1) {
      Serial.print((String)"R1: " + density1);
      Serial.println((String)" | R2: " + density2);
      if (density1 > density2) {
        traffic(ROAD1, YELLOW);
        traffic(ROAD2, YELLOW);
        delay(2000);
        traffic(ROAD1, GREEN);
        traffic(ROAD2, RED);
      }
      else if (density2 > density1) {
        traffic(ROAD1, YELLOW);
        traffic(ROAD2, YELLOW);
        delay(2000);
        traffic(ROAD1, RED);
        traffic(ROAD2, GREEN);
      }
      else if (density1 == 0 && density2 == 0) {
        traffic(ROAD1, GREEN);
        traffic(ROAD2, GREEN);
      }
      update = 0;
    }
  }
}

if (millis() - prevMs >= 1000) {
  serverControl();
  if (emrg1 == 1) output(eled1, 1);
  else output(eled1, 0);
```

```
    if (emrg2 == 1) output(eled2, 1);
    else output(eled2, 0);
    if (emrg1 == 1 || emrg2 == 1) digitalWrite(buzz, !emrg1);
    else digitalWrite(buzz, 0);

    prevMs = millis();
  }
}

void serverControl() {
  int httpCode;
  if (flip == 0) {
    flip = 1;
    String link = (String)link0 + "?rd1=" + density1 + "&rd2=" + density2;
    if (http.begin(client, link)) {
      httpCode = http.GET();
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
        Serial.println(http.getString());
    }
  }
  else {
    flip = 0;
    if (http.begin(client, link1)) {
      httpCode = http.GET();
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
        String resp = http.getString();
        Serial.println(http.getString());

        if (resp.indexOf("server=1") != -1) serverMode = 1;
        else if (resp.indexOf("server=0") != -1) serverMode = 0;

        if (serverMode == 1) {
```

```
        if (resp.indexOf("road1=1") != -1) traffic(ROAD1, GREEN);
        else if (resp.indexOf("road1=0") != -1) traffic(ROAD1, RED);

        if (resp.indexOf("road2=1") != -1) traffic(ROAD2, GREEN);
        else if (resp.indexOf("road2=0") != -1) traffic(ROAD2, RED);

        if (resp.indexOf("butt1=1") != -1) emrg1 = 1;
        else if (resp.indexOf("butt1=0") != -1) emrg1 = 0;

        if (resp.indexOf("butt2=1") != -1) emrg2 = 1;
        else if (resp.indexOf("butt2=0") != -1) emrg2 = 0;
      }
    }
  }
 }
}

void traffic(byte road, byte m) {
 if (road == ROAD1) {
  if (m == RED) output(rled1, 1);
  else output(rled1, 0);
  if (m == YELLOW) output(yled1, 1);
  else output(yled1, 0);
  if (m == GREEN) output(gled1, 1);
  else output(gled1, 0);
 }
 else {
  if (m == RED) output(rled2, 1);
  else output(rled2, 0);
  if (m == YELLOW) output(yled2, 1);
  else output(yled2, 0);
  if (m == GREEN) output(gled2, 1);
  else output(gled2, 0);
 }
```

```
}

void output(byte n, bool state) {
  static byte leds = 0;
  bitWrite(leds, n, state);
  for (byte i = 7; i >= 0; i--) {
    digitalWrite(DS, bitRead(leds, i));
    digitalWrite(SH, 1); digitalWrite(SH, 0);
    if (i == 0) break;
  }
  digitalWrite(ST, 1); digitalWrite(ST, 0);
}
```

# Future Works

There are some limitations on the proposed model, which need to be enhanced further. The proposed system uses WiFi to communicate between devices; however, their energy consumptions and solutions to recharge them are not considered in this study. Alternate solutions such as solar charging or charging from street lights can be further looked at. Similarly, the proposed model is tested only in the context of the single-lane road as intended. However, it would be useful to test the system in a multi-lane scenario to identify the false detections.

For future directions, the proposed system could be further improved considering different aspects. The first dimension is suggesting an optimal route for the drivers based on real-time data.

The dynamic traffic signal control functionality is also considered as future work. In this case, the communication of roadside display units and traffic signals have to be established. Another aspect is the real-time implementation of the system including the IoT security features [1,52] in the communication layer; the prototype has to be extended to a complete end-to-end system with central server communication. Moreover, the integration of IoT security, communication between display units, and traffic signals will be investigated in future research.

- Appropriate to estimate traffic congestions on collector roads using road occupancy measure
- Update residents on real-time traffic messages through roadside display units
- Monitor the road density of smart campuses especially during peak hours and help to improve mobility
- Assist authorities to broadcast important traffic incident messages
- Provide a real-time dashboard to monitor the traffic updates

# Conclusion

IOT based road can improve travel time, road safety and reduce traffic congestions. It will enable police officers to view real-time traffic condition. As a step ahead ESP8266 can be linked to ThingSpeak Platform to perform analyses of traffic data. Augmented reality used in traffic will increase safety and comfort for the drivers as well as pave the way for autonomous driving functions. With the help of augmented reality critical information such as speed and navigation path can be seen while looking at the road ahead (on the windscreen).

# References


1. Hao Dong, Xingguo Xiong and Xuan Zhang, "Design and Implementation of a Real Time Traffic Light Control System Based on FPGA", *Proceeding of the 1st Conference on ASEE*, April 3–5, 2014.
2. Azura Che Soh, Lai Guan Rhung and Haslina Md. Sarkan, "MATLAB Simulation of Fuzzy Traffic Light Controller for Multilane Isolated Intersection", *International Journal on Computer Science and Engineering*, vol. 2, pp. 924-933, 2010.
3. N. Dinesh Kumar, G. Bharagava Sai and K. Shiva Kumar, *Traffic Control System using LABVIEW Global Journal of Advanced Engineering Technologies*, vol. 2, pp. 47-49, 2013.
4. Ashwini Y. Dakhole and Mrunalini P. Moon, "Design of Intelligent Traffic Control System Based on ARM", *International Journal of Advanced Research in Computer Science and Management Studies,* vol. 1, pp. 76-80, 2013.
5. Carlos Paiz, Christopher Pohl and Mario Porrmann, "Hardware in the loop Simulations for FPGA Based Digital Control Design", *Informatics in Control Automation and Robotics Lecture Notes Electrical Engineering*, vol. 15, pp. 355-372, 2008.
6. B. Dilip, Y. Alekhya and P. D. Bharathi, "FPGA Implementation of an Advanced Traffic Light Controller using Verilog HDL", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 7, 2012.
7. H. C. N. Premachandra, T. Yendo, M. P. Tehrani, T. Yamazato, H. Okada, T. Fujii, et al., "High-speed-camera image processing based LED traffic light detection for road-to-vehicle visible light communication", *Intelligent Vehicles Symposium (IV)*, pp. 793-798, June 2010.
8. B. Zhou, J. Cao, X. Zeng and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system", *Vehicular Technology Conference Fall (VTC 2010-Fall)*, vol. 72nd, pp. 1-5, September 2010.
9. R. Cucchiara, M. Piccardi and P. Mello, "Image analysis and rule-based reasoning for a traffic monitoring system", *Intelligent Transportation Systems IEEE Transactions*, vol. 1, no. 2, pp. 119-130, 2000.
10. S. Indu, M. Gupta and A. Bhattacharyya, "Vehicle tracking and speed estimation using optical flow method", *International Journal on Engineering Science and Technology*, vol. 3, no. 1, pp. 429-434, 2011.

11. R. Pazoki, A. S. Rad, D. Habibi, F. Paknejad, S. Kobraee, B. Yerli, et al., "An Effective Method of Head lamp and Tail lamp Recognition for Night Time Vehicle Detection", *G. C. World Academy of Science Engineering and Technology*, pp. 68.
12. R. K. Nath and S. K. Deb, "On road vehicle/object detection and tracking using template", *Indian Journal of Computer Science and Engineering*, vol. 1, no. 2, pp. 98-107, 2010.
13. Kingsley Monday Udofia, Joy Omoavowere Emagbetere and Frederick Obataimen Edeko, "Dynamic traffic signal phase sequencing for an isolated intersection using ANFIS", *Automation Control and Intelligent Systems*, vol. 2, no. 2, pp. 21-26, 2014.
14. W. Zhang, Q. J. Wu, G. Wang and X. You, "Tracking and pairing vehicle headlight in night scenes", *Intelligent Transportation Systems IEEE Transactions*, vol. 13, no. 1, pp. 140-153, 2012.
15. Anna Merine George and Palika Shetty S, "Fuzzy Controller for an Image based Traffic System", *International Journal of Management IT and Engineering (IJMIE)*, vol. 2, no. 6, pp. 291-305, 2012.
16. NXP Semiconductors datasheet - PCF8574; PCF8574A - Remote 8-bit I/O expander for I2C-bus with interrupt, *http://www.pibits.net/code/raspberry-pi-pcf8574-example.php*
17. Espressif Systems ESP8266 WiFi module, *https://microcontrollerslab.com/esp8266-pinout-reference-gpio-pins/*
18. *D. Mellis, M. Banzi, D. Cuartielles, and T. Igoe, "Arduino: An open electronic prototyping platform, " in Proc. CHI, vol. 2007, 2007*
19. *"Arduino FAQ – With David Cuartielles"*. Malmö University. April 5, 2013. Retrieved 2014-03-24