



---

**Graduation Project-2**

**“RFID-Based Intelligent Traffic  
Control System”**

**Final Report**

---

**Submitted by:**

Afaf Hamada Fawzy  
Farouk Sherif Bedier  
Michael Hany Philip  
Mohammed Emad Salem  
Nardine Elia Aziz

**Supervised by:**

Dr. Hassan Mostafa Hassan

# Acknowledgment

---

Listed below are the names of the people who provided us with significant help in developing our graduation project in addition to our sponsor NTRA. To all we extend our sincere thanks.

\*\*\*\*\*

Dr. Hassan Mostafa Hassan

Eng. Maged

Eng. Menna

# Abstract

---

Traffic congestion is one of the worldwide problems occurring in a lot of countries despite their region or whether or not the country is a developed country or a developing country. The usual traditional solutions would be building more roads, bridges or tunnels. However, this project discusses a solution from a different scope and a different angle. The RFID Based Intelligent Traffic Control System is a 'smart' solution to the above mentioned problem. Based on the data the traffic light reads from each street it acts accordingly deciding which street is to be served. The decision is made based on data received from the antennas indicating traffic congestion he decision is made based on data received from the antennas indicating traffic congestion or the presence of pedestrians or emergency cars and act accordingly instead of the traditional static solutions of the ordinary traffic lights which in some cases may lead to worsening the situation in terms of traffic jams. The system is composed of several hardware and software tools integrated together to compose the logic and the execution decisions of the project itself. These tools and design procedures are discussed in deeper details throughout the report. For this the demonstration is modeled on a T-intersection traffic light.

## Table of Contents

Chapter 1.....	1
1.1.    RFID Technology .....	1
<b>1.1.1.    What is RFID?</b> .....	1
<b>1.1.2.    Uses of RFID</b> .....	2
<b>1.1.3.    Types of RFID</b> .....	4
<b>1.1.4.    Components of RFID</b> .....	6
<b>1.1.5.    How it works</b> .....	7
1.2.    Labview and NI MyRIO .....	8
<b>1.2.1.    About NI</b> .....	8
<b>1.2.2.    What is LabVIEW?</b> .....	8
<b>1.2.3.    What is myRIO?</b> .....	8
1.3.    Problem Definition .....	9
<b>1.3.1.    Statistics</b> .....	11
Chapter 2.....	13
2.1.    Problem Solution .....	14
<b>2.1.1.    Proposed Solution</b> .....	14
2.1.2.    System Flowchart .....	21
2.1.3.    System Boolean Equations .....	23
2.2.    Simulation .....	25
<b>2.2.1.    SUMO</b> .....	25
<b>2.2.2.    TraCI4Matlab</b> .....	27
<b>2.2.3.    Simulation Description</b> .....	27
<b>2.2.4.    Simulation Steps</b> .....	28
<b>2.2.5.    Simulation Results</b> .....	29
Chapter 3.....	35
3.1.    Graphical User Interface (GUI).....	35
<b>3.1.1.    Database</b> .....	35
<b>3.1.2.    Graphical User Interface (GUI)</b> .....	38
3.2.    LabVIEW .....	41
<b>3.2.1.    LabVIEW blocks description</b> .....	41
Chapter 4.....	86
4.1.    Block Diagrams .....	87
4.2.    Radio Frequency Identification .....	89

<b>4.2.1. Description of the used RFID</b> .....	89
<b>4.2.2. Connection to Server</b> .....	92
4.3. NI MyRIO .....	94
<b>4.3.1. Connection to Database</b> .....	94
<b>4.3.2. Input/Output Map</b> .....	96
Chapter 5.....	99
5.1. Basic T-intersection Structure .....	100
<b>5.1.1. The Steel Structure</b> .....	100
<b>5.1.2. Modeling the Streets</b> .....	103
5.2. Testing the Hardware .....	104
Chapter 6.....	106
6.1. Possible Future Improvements .....	107
References .....	108
Appendix A.....	109
A.1. Intelligent Traffic Light system with pedestrians .....	109
<b>A.1.1. ODMatrix for Three Streets with Pedestrians</b> .....	114
<b>A.1.2. Incrementing Priority</b> .....	115
A.2. Intelligent Traffic Light System Without pedestrians .....	116
<b>A.2.1. ODMatrix for Three Streets without Pedestrians</b> .....	118
A.3. Ordinary Traffic Light System .....	118
A.4. Intelligent Traffic Light System with Dynamic Green Timing .....	121
Appendix B .....	126
Appendix C .....	129
A.1. Requirements .....	129
A.2. Installation Guide .....	130

## Table of Figures

FIGURE 1.1 RFID READING MULTIPLE TAGS ..... 2

FIGURE 1.2 96-BIT STRING OF THE RFID TAG ..... 7

FIGURE 1.3 VEHICLE TYPES DISTRIBUTION (ZIAD NAKAT ET AL., 2013) ..... 12

FIGURE 1.4 CONGESTION HOURS (ZIAD NAKAT ET AL., 2013)..... 12

FIGURE 2.5 T-SHAPED INTERSECTION ..... 14

FIGURE 2.6 DIRECTIONS OF VEHICLES FROM STREET #1..... 15

FIGURE 2.7 DIRECTIONS OF VEHICLES FROM STREET #2..... 15

FIGURE 2.8 DIRECTIONS OF VEHICLES FROM STREET #3..... 16

FIGURE 2.9 FLOW CHART OF THE PROPOSED SOLUTION ..... 22

FIGURE 2.10 RESULTS OF THE SIMULATION FOR 50 VEHICLES PER HOUR..... 29

FIGURE 2.11 RESULTS OF THE SIMULATION FOR 100 VEHICLES PER HOUR..... 30

FIGURE 2.12 RESULTS OF THE SIMULATION FOR 150 VEHICLES PER HOUR..... 30

FIGURE 2.13 RESULTS OF THE SIMULATION FOR 200 VEHICLES PER HOUR..... 31

FIGURE 2.14 RESULTS OF THE SIMULATION FOR 250 VEHICLES PER HOUR..... 32

FIGURE 2.15 RESULTS OF THE SIMULATION FOR 300 VEHICLES PER HOUR..... 32

FIGURE 3.16 SHOWS THE GUI WITH THE DATA NEEDED TO BE ENTERED FOR A SUCCESSFUL DATABASE UPDATE ..... 38

FIGURE 3.17 SHOWS HOW THE DATA OF A VEHICLE ENTERS THE VEHICLES DATABASE TABLE ..... 39

FIGURE 3.18 SHOWS THE GUI OF THE TRACKING AND REPORTING VEHICLES..... 40

FIGURE 3.19 SHOWS HOW THE DATA OF A VEHICLE ENTERS THE TRACKING DATABASE TABLE ..... 40

FIGURE 3.20 START BLOCK..... 41

FIGURE 3.21 GREEN TRAFFIC LIGHT FOR STREET #1..... 42

FIGURE 3.22 CONDITIONS OF Q1 ..... 43

FIGURE 3.23 CONDITIONS OF N1 ..... 45

FIGURE 3.24 CONDITIONS OF THE LEFT LANE GREEN TRAFFIC LIGHT FOR STREET #1 ..... 47

FIGURE 3.25 CONDITIONS OF RIGHT LANE GREEN TRAFFIC LIGHT FOR STREET #1 ..... 48

FIGURE 3.26 CONDITIONS OF YELLOW TRAFFIC LIGHT FOR STREET #1 ..... 49

FIGURE 3.27 CONDITIONS OF SIGNAL Y1 ..... 50

FIGURE 3.28 RED TRAFFIC LIGHT FOR STREET #1 ..... 51

FIGURE 3.29 SUB-BLOCK 3..... 51

FIGURE 3.30 GREEN FOR PEDESTRIANS IN STREET #1..... 52

FIGURE 3.31 GREEN TRAFFIC LIGHT FOR STREET #2..... 53

FIGURE 3.32 CONDITIONS OF Q1 ..... 54

FIGURE 3.33 CONDITIONS OF N2 ..... 56

FIGURE 3.34 CONDITIONS OF THE LEFT LANE GREEN TRAFFIC LIGHT FOR STREET#2 ..... 58

FIGURE 3.35 CONDITIONS OF THE RIGHT LANE GREEN TRAFFIC LIGHT FOR STREET #2 ..... 59

FIGURE 3.36 CONDITIONS OF YELLOW TRAFFIC LIGHT FOR STREET #2 ..... 60

FIGURE 3.37 CONDITIONS OF SIGNAL Y2 ..... 61

FIGURE 3.38 CONDITIONS OF RED TRAFFIC LIGHT FOR STREET #2..... 62

FIGURE 3.39 GREEN FOR PEDESTRIANS IN STREET #2..... 63

FIGURE 3.40 GREEN TRAFFIC LIGHT FOR STREET #3..... 64

FIGURE 3.41 CONDITIONS OF Q3 ..... 65

FIGURE 3.42 CONDITIONS OF N3 ..... 67

FIGURE 3.43 CONDITIONS OF THE LEFT LANE GREEN TRAFFIC LIGHT FOR STREET #3 ..... 69

FIGURE 3.44 CONDITIONS OF THE RIGHT LANE GREEN TRAFFIC LIGHT FOR STREET #3 ..... 70

FIGURE 3.45 CONDITIONS OF YELLOW TRAFFIC LIGHT FOR STREET #3 ..... 71

FIGURE 3.46 CONDITIONS OF SIGNAL Y3 ..... 72

FIGURE 3.47 CONDITIONS OF RED TRAFFIC LIGHT FOR STREET #3..... 73

FIGURE 3.48 CONDITIONS OF GREEN OF PEDESTRIANS IN STREET #3 .....	74
FIGURE 3.49 YELLOW LIGHT TIMER .....	76
FIGURE 3.50 GREEN LIGHT TIMER.....	76
FIGURE 3.51 DYNAMIC TIMING BLOCK.....	77
FIGURE 3.52 TIMER BLOCK.....	78
FIGURE 3.53 STRUCTURE-1.....	79
FIGURE 3.54 STRUCTURE-2.....	80
FIGURE 3.55 PEDESTRIANS TIMER.....	81
FIGURE 3.56 INTERRUPT TIMER.....	82
FIGURE 3.57 PRIORITY .....	83
FIGURE 3.58 PRIORITY BLOCK .....	85
FIGURE 4.59 BLOCK DIAGRAM SHOWING THE RFID-BASED INTELLIGENT TRAFFIC LIGHT SYSTEM .....	87
FIGURE 4.60 BLOCK DIAGRAM OF ADDING NEW VEHICLES TO DATABASE .....	88
FIGURE 4.61 BLOCK DIAGRAM OF THE TRACKING DATABASE .....	88
FIGURE 4.62 RFID READER USED IN THE SYSTEM.....	90
FIGURE 4.63 ANTENNA USED IN THE SYSTEM .....	91
FIGURE 4.64 RFID TAG USED IN THE SYSTEM.....	92
FIGURE 4.65 EXAMPLE OF THE NOTIFICATION LOG DATABASE .....	93
FIGURE 4.66 NI MYRIO.....	94
FIGURE 4.67 DATABASE CONNECTIVITY TOOLKIT IMPLEMENTATION ON LABVIEW .....	95
FIGURE 4.68 MAPPING OF INPUT PUSHBUTTONS ON MYRIO.....	96
FIGURE 4.69 MAPPING OF STREET #1 OUTPUTS ON MYRIO .....	97
FIGURE 4.70 MAPPING OF STREET #2 OUTPUTS ON MYRIO .....	97
FIGURE 4.71 MAPPING OF STREET #2 OUTPUTS ON MYRIO .....	98
FIGURE 4.72 MAPPING OF STREET #3 OUTPUTS ON MYRIO .....	98
FIGURE 5.73 ANTENNA PLANTED ON ONE OF THE WOODEN PLATES.....	100
FIGURE 5.74 SETUP PLACE ON THE MAQUETTE.....	101
FIGURE 5.75 TRAFFIC LIGHTS.....	101
FIGURE 5.76 EMERGENCY VEHICLES WITH RFID TAGS .....	102
FIGURE 5.77 CIVILIAN VEHICLES WITH RFID TAGS.....	102
FIGURE 5.78 FACING VEHICLE INFRONT OF ANTENNA.....	103
FIGURE 5.79 SCALED VERSION OF THE T-INTERSECTION .....	103





# Chapter 1

---

## Introduction

### 1.1. RFID Technology

#### 1.1.1. What is RFID?

**R**FID stands for Radio Frequency Identifier. It is a form of wireless communication that uses radio waves to identify and track objects. It is a flexible, automatic identification technology that transmits the identity (in the form of a unique serial number) of an object or person wirelessly using radio waves. It comes under the category of automatic identification technologies. This technology saves time and effort exerted in entering data manually. Each RFID tag has its own memory. Recent developments in RFID technology have increased its readings range.

RFID takes the barcoding concept and digitizes it for the modern world providing some advantages over barcoding.

**Advantages of RFID over barcode:**

- Uniquely identify an individual item beyond just its product type
- Identify items without direct line-of-sight
- Identify many items (up to 1,000s) simultaneously
- Identify items within a range between a few centimeters to several meters.



Figure 1.1 RFID reading multiple tags

**1.1.2. Uses of RFID**

RFID can be used to track and monitor the physical world automatically and with accuracy. It can tell you what an object is, where it is, and even its condition, which is why it is integral to the development of the Internet of Things (IoT). It allows the physical world itself to become an information system, automatically sensing what is happening, sharing related data, and responding.

## **RFID Applications**

### **1. Asset Tracking**

Companies can put RFID tags on assets that are lost or stolen, that are underutilized or that are just hard to locate at the time they are needed.

### **2. Manufacturing**

RFID has been used in manufacturing plants to track parts automatically, reducing errors and the need to have people look for parts needed on the manufacturing line, which increases throughput and manages the production of different versions of the same product.

### **3. Supply Chain Management**

RFID technology has been used in closed loop supply chains or to automate parts of the supply chain within a company's control for years.

### **4. Retailing**

Retailers are currently focused on improving supply chain efficiency and making sure product is on the shelf when customers want to buy it.

### **5. Payment Systems**

It is used to pay for road tolls without stopping. Quick service restaurants are using the active RFID tags to pay for meals at drive-through windows. Also, it is a convenient way to pay for bus, subway and train rides.

### **6. Security and Access Control**

RFID has long been used as an electronic key to control who has access to office buildings or areas within office buildings. The first access control systems used low-frequency RFID tags. The advantage of RFID is it is convenient (an employee can hold up a badge to unlock a door, rather than looking for a key or swiping a magnetic stripe card) and because there is no contact between the card and reader, there is less wear and tear, and therefore less maintenance.

### 1.1.3. Types of RFID

RFID systems can be divided in two broad categories of RFID systems- passive and active. Also, it can be broken down by the frequency band within which they operate: low frequency, high frequency, and ultra-high frequency.

#### **RFID Frequencies**

Frequency refers to the size of the radio waves used to communicate between RFID systems components. It operates in **low frequency (LF)**, **high frequency (HF)** and **ultra-high frequency (UHF) bands**. Radio waves behave differently at each of these frequencies.

#### **LF RFID**

The LF band covers frequencies from 30 KHz to 300 KHz. Typically LF RFID systems operate at 125 KHz, although there are some that operate at 134 KHz. This frequency band provides a short read range of 10 cm, and has slower reading speed than the higher frequencies, but one of the important advantages of LF RFID that it is not very sensitive to radio wave interference. LF RFID is used in access control and livestock tracking.

#### **HF RFID**

The HF band ranges from 3 to 30 MHz. Most HF RFID systems operate at 13.56 MHz with read ranges between 10 cm and 1 m. HF systems experience moderate sensitivity to interference. HF RFID is commonly used for ticketing, payment, and data transfer applications.

#### **UHF RFID**

The UHF frequency band covers the range from 300 MHz to 3 GHz. There is some variance in frequency from region to region, UHF Gen2 RFID systems in most countries operate between 900 and 915 MHz. The read range of passive UHF systems can be as long as 12 m, and UHF RFID has a faster data transfer rate than LF or HF.

UHF RFID is the most sensitive to interference, but many UHF product manufacturers have found ways of designing tags, antennas, and readers to keep performance high even in difficult environments. Passive UHF tags are easier and cheaper to manufacture than LF and HF tags.

UHF RFID is used in a wide variety of applications, ranging from retail inventory management, to pharmaceutical anti-counterfeiting, to wireless device configuration. The bulk of new RFID projects are using UHF opposed to LF or HF, making UHF the fastest growing segment of the RFID market.

## **Passive, Active, and BAP RFID Systems**

### **Active RFID Systems**

In active RFID systems, tags have their own transmitter and power source. Usually, the power source is a battery. Active tags broadcast their own signal to transmit the information stored on their microchips. It operates in the ultra-high frequency (UHF) band and offer a range of up to 100 m. In general, active tags are used on large objects, such as rail cars, big reusable containers, and other assets that need to be tracked over long distances.

#### **Types of active tags:**

##### **Transponders:**

Wake up when they receive a radio signal from a reader, and then power on and respond by transmitting a signal back. Because transponders do not actively radiate radio waves until they receive a reader signal, they conserve battery life.

##### **Beacons:**

Used in most real-time locating systems (RTLS), in order to track the precise location of an asset continuously. They are not powered on by the reader's signal. Instead, they emit signals at pre-set intervals. Depending on the level of locating accuracy required.

### **Passive RFID Systems**

In passive RFID systems, the reader and reader antenna send a radio signal to the tag. The RFID tag then uses the transmitted signal to power on, and reflect energy back to the reader.

Passive RFID systems can operate in the low frequency (LF), high frequency (HF) or ultra-high frequency (UHF) radio bands. Passive system ranges are limited by the power of the tag's backscatter (the radio signal reflected from the tag back to the reader), they are typically less than 10 m. Because passive tags do not require a power source or transmitter, and only require a tag chip and antenna, they are cheaper, smaller, and easier to manufacture than active tags.

### **Battery-Assisted Passive (BAP) Systems**

Most passive RFID tags use the energy from the RFID reader's signal to power on the tag's chip and backscatter to the reader, while BAP tags use an integrated power source (usually a battery) to power on the chip, so all of the captured energy from the reader can be used for backscatter. Unlike transponders, BAP tags do not have their own transmitters.

## **1.1.4. Components of RFID**

An RFID system has *readers* and *tags* that communicate with each other by radio.

### **RFID Tags**

An RFID tag is comprised of an integrated circuit (called an IC or chip) attached to an antenna that has been printed, etched, stamped or vapor-deposited onto a mount which is often a paper substrate or PolyEthylene Therephtalate (PET).

### **RFID Readers**

An RFID reader, also known as an interrogator, is a device that provides the connection between the tag data and the enterprise system software that needs the information. The reader communicates with tags that are within its field of operation.

### 1.1.5. How it works

#### Tag chip

It is previously programmed with:

- Tag identifier
- Unique serial number

Also the tag chip contains a memory bank that stores the items' unique tracking identifier which is known as electronic product code (EPC).

#### Electronic product code (EPC)

The EPC has 96-bit string of data which is printed to the tag by an RFID printer this 96-bit string of data are divide as following:

1. **The first 8-bits (Header):** identifies the version of the protocol
2. **The next 28-bits:** identifies the organization that manages the data for this tag
3. **The next 24-bits:** identifies the kind of the product
4. **The last 36-bits:** are unique serial number for a particular tag

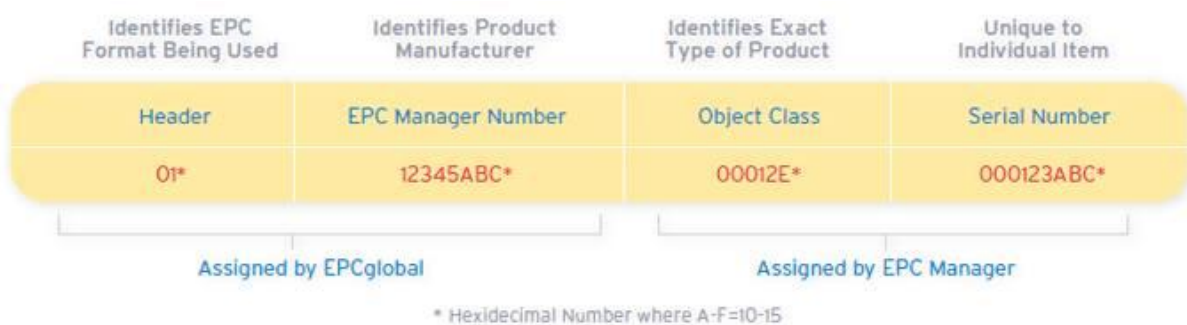


Figure 1.2 96-bit string of the RFID Tag

#### Tag antennas

It turns the chip on by receiving signal energy and passes it to the chip. There are many ways these tag antennas are integrated on the tag; they can be printed, etched, or stamped with conductive ink, or even vapor deposited onto labels. Tags may have single or multiple antennas.

The problem of a single antenna is that it can result in “dead zones”, while a tag with dual antenna can eliminate these dead zones.

## 1.2. Labview and NI MyRIO

### 1.2.1. About NI

National Instruments is an American company that produces automated test equipment and virtual instrumentation software. NI was founded in 1976 by Dr. James Truchard and Jeff Kodosky. NI provides technological solutions to accelerate productivity. This is done by integration of hardware and software which make the tools easier and more user-friendly to use. NI produced many hardware and software products. In this product, one of hardware and another of software products are used. In fact they are not only used, they are considered the core of the project.

### 1.2.2. What is LabVIEW?

LabVIEW is one of NI software products. LabVIEW is a development environment designed specifically to accelerate the productivity of engineers. It enables users to visualize and code using a graphical programming syntax. This helps users figure out their errors easily as they see the product during simulation. LabVIEW is a flexible programming environment that enables one to build unique applications; it can be used in taking simple measurements or prototyping with FPGA technology. It can also be used for the simulation of academic study by professors.

### 1.2.3. What is myRIO?

It is a Hardware compact device that works on or implements the simulated LabVIEW programs. NI myRIO uses a dual-core ARM® Cortex™-A9 real-time processing and Xilinx FPGA customizable Input and output. It is used to program onboard devices and use external sensors. NI myRIO mixes between FPGA programming and LabVIEW software, in other words, it provides the option to program the processor in either LabVIEW or C/C++.



## 1.3. Problem Definition

People, nowadays, depend on their vehicles as the main mean of transportation. There are many reasons why people depend on their vehicles not the public transportation system, for example: some people think that their vehicle is more comfortable; others think that their vehicles are faster; others just enjoy their privacy; others do not have the luxury of choice because their city/country does not offer public transportation. Everyone has his/her own reason to take their cars instead of the bus/train.

In some places in the world, where most people use public transportation instead of their own vehicles, the number of vehicles on the streets is very low causing the time duration to drive to the desired destination to be short. In all other places in the world where the number of vehicles on the streets is very high causes traffic congestion that causes the time duration to drive to the desired destination to increase.

Traffic congestion is one of the main problems that face the citizens all over the globe. Traffic congestion causes so many negative impacts; for example:

- Arriving late for work/school/college
- Waste of time (non-productive activity)
- Stress causing road rage
- Emergency cars will not be able to arrive on time
- Increase in carbon dioxide emissions causing air pollution
- Waste of fuel
- Wear and tear of vehicle due to accelerating and braking more frequently
- Higher chances of collision

These negative impacts and more are the reason that forces the officials to work as fast as possible to solve the problems that causes traffic congestion. There are many solutions that decrease the amount of time to reach one's destination. Some of these solutions are:

- Use bridges/tunnels
- Build more multi-lane streets
- Separate lane for emergencies
- Road pricing (to decrease number of drivers)
- Public transportation (serve more people than a normal vehicle using less space)

These are all practical and efficient solutions; however, they are static solutions, which means that if any of the design variables (number of vehicles, pedestrians, etc...) changes, the solution will not be enough until another solution design is implemented. Therefore, a dynamic system, using Intelligent Transportation System, would be more preferable to solve most of the issues.

This project's concern is with the traffic light systems. The ordinary traffic light system has a static time Red, Yellow and Green time that has already been chosen to decrease the average waiting time for all vehicles in the intersection. This system has limited capabilities which cause the average waiting time to not reach its minimum.

In order to decrease the average waiting time more than that of the ordinary traffic light system, the system needs to have answers to some questions:

1. How to act if the street is:
  - a) Empty?
  - b) Semi-crowded?
  - c) Extremely crowded?
2. How to act if there are any emergency cars in any of the streets?
3. How to act if there are any pedestrians who want to cross any of the streets?

No ordinary system can answer these questions with their limited capabilities; therefore, a dynamic system is needed for better performance.

### 1.3.1. Statistics

Cairo has more than 19 million inhabitants, which are more than one-fifth of Egypt's population. By year 2027, Cairo's population is expected to increase up to 24 million inhabitants, so its importance in economy will consequently increase. Cairo is also one of the most important cities contributing to the Egyptian economy in terms of GDP (Gross Domestic Product) and jobs. According to a study recently released by the World Bank, Cairo's traffic costs Egypt EGP 47 billion annually and is expected to reach EGP 105 billion by 2030. According to the study, about four percent of Egypt's GDP is lost because of traffic congestion; this takes into account the cost of time wasted (50 percent), delay expenses (31 percent), and health costs (19 percent). Waiting while traffic is still, wastes a lot of time, that could be used any more productive activities. Congestion also causes unnecessary fuel consumption, wear and tear of the used vehicles, emission of harmful gases that reduces air quality, makes transportation more expensive for businesses and makes Cairo an unpleasant location to initiate any business.

In definite places such as 6th of October Bridge and the Ring Road at Carrefour Al Maadi, Traffic volumes range from 3000 vehicles per hour per lane to 7000 vehicle per hour per lane in congestion hours. For local streets such as Al Dokki Street and Gesr Al Suez Street, traffic volumes are in range of 1000 vehicle per hour per lane to 4000 vehicle per hour per lane during peak hours.

Cars, mostly private ones, are the dominant way of transportation in Cairo, in both main streets and corridors. Given that the most dominantly used vehicles are the private cars and taxis, it is a normal consequence that Cairo is very congested. Most speeds on corridors (internal roads) are in the range of 50 to 60 percent of normal driving speeds, and in main streets, they sometimes reach 20 to 30 percent, this means that driving to work may take double its normal time. What is worse is that this congestion isn't observable only on peak congestion hours; it is in the same range all day. The figure below represents the percentage of each vehicle type in local streets.

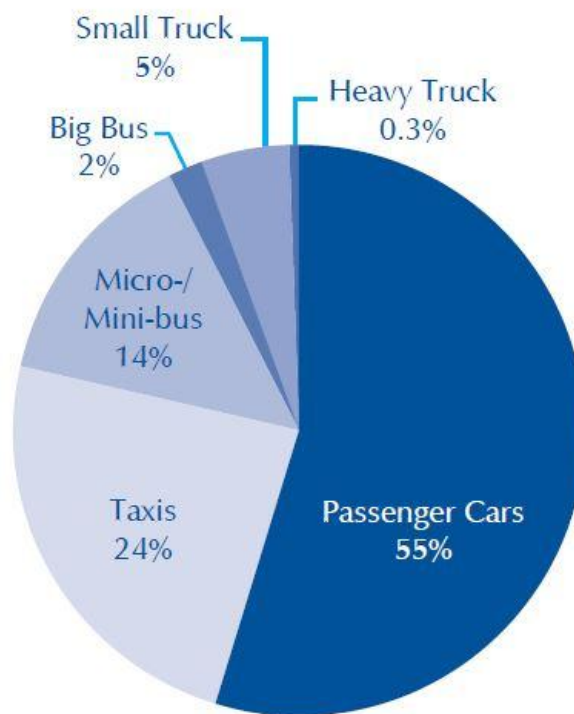


Figure 1.3 Vehicle types distribution (Ziad Nakat et al., 2013)

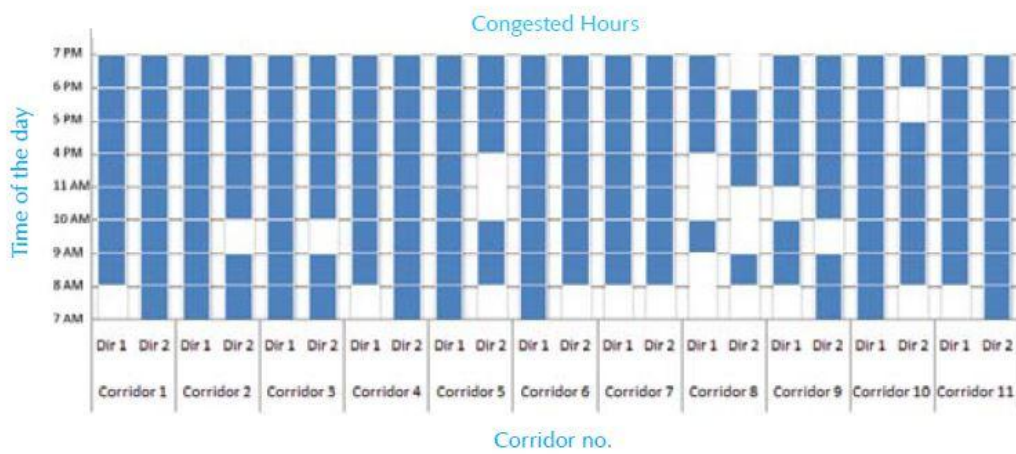


Figure 1.4 Congestion hours (Ziad Nakat et al., 2013)

# Chapter 2

---

## Simulation of Problem Solution

**T**his chapter focuses on the approach of improving the statistics discussed in the previous chapter. What matters most to drivers is the average time they need to arrive to their destination(s). Therefore, the most influential variable affecting the average arrival time is the average ‘waiting time’ in every intersection/ traffic light.

Therefore, this project is mainly concerned with reducing the average waiting time for every traffic light possible. For this to be achieved, a more Intelligent Transportation System (ITS) for the traffic lights should be used other than the basic static systems already implemented.

## 2.1. Problem Solution

### 2.1.1. Proposed Solution

Using the available technology of the RFID helps counting the cars in queue and keeping track of how the flow of vehicles is going at an identified intersection monitored by the traffic light. This is essential for detecting problems either at an early stage after it has occurred to solve it before the situation gets worse, or may even at some occasions avoid the problem from the start before it even occurs.

The project was simulated on a T-shaped intersection due to the availability of three RFID Antennas; however it was enough to demonstrate all the possible solutions this project has taken into consideration as shown in figure 2.1.

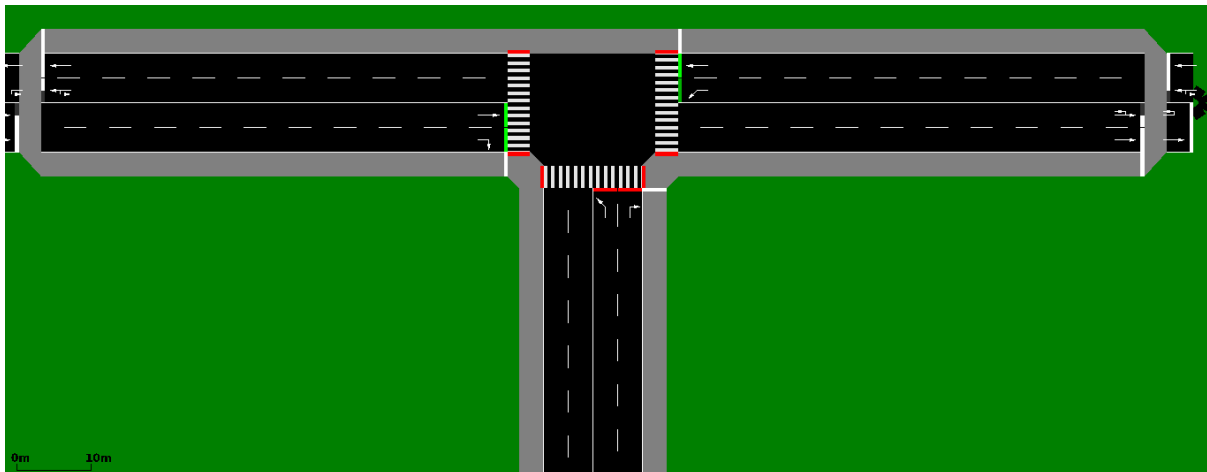


Figure 2.5 T-shaped intersection

This intersection forces the traffic light system to allow only one street to have green light to pass. To demonstrate how the vehicles pass this intersection, they follow one of two green light signals per street depending on what route the driver is intending to take.

Table 2.1 Street #1 directions

Red	Yellow	Green 1	Green 2
Stop	Get ready to stop	Turn left	Turn right

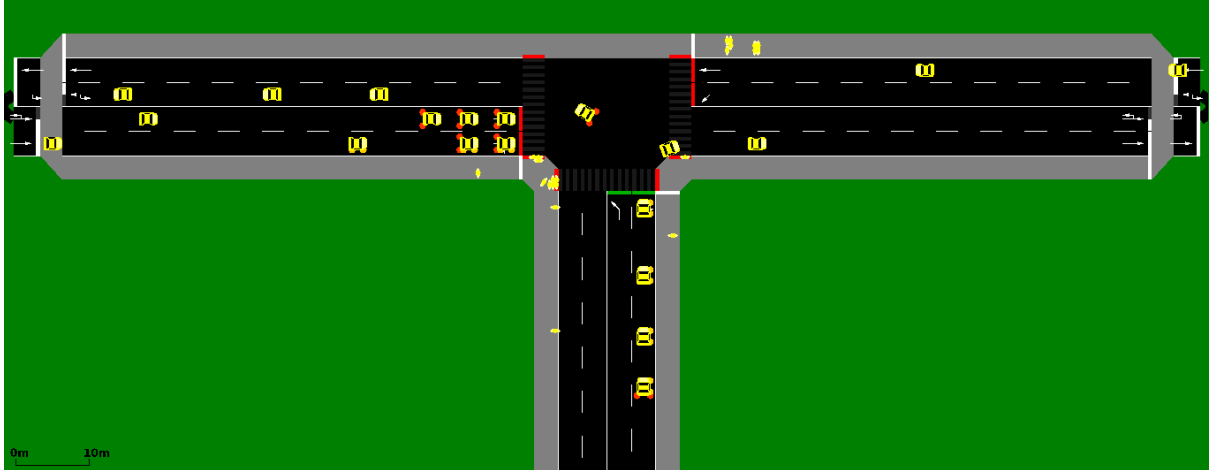


Figure 2.6 Directions of vehicles from street #1

Table 2.2 Street#2 directions

Red	Yellow	Green 1	Green 2
Stop	Get ready to stop	Drive straight	Turn right



Figure 2.7 Directions of vehicles from street #2

Table 2.3 Street#3 directions

Red	Yellow	Green 1	Green 2
Stop	Get ready to stop	Turn left	Drive straight

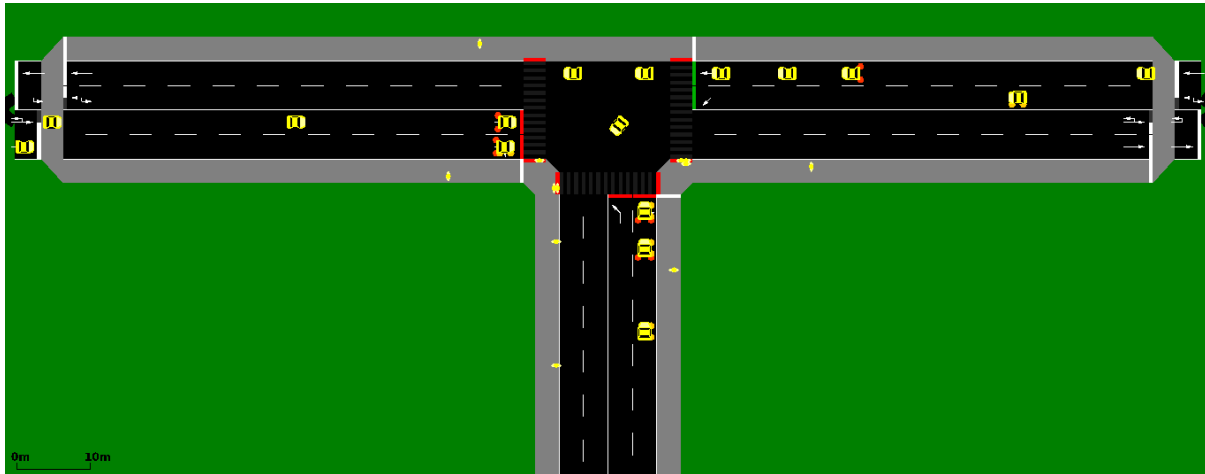


Figure 2.8 Directions of vehicles from street #3

After reaching the design of the intersection and after understanding how the vehicles move according to the previous tables and figures, it is time now to think of solutions to reduce the average waiting time of the vehicles at this specific intersection. However, the rules will apply at any other intersection design.

Before thinking of a solution, one must study the problem very well. As discussed in Chapter 1, the problem with the ordinary traffic light systems is that it does not have answers to some questions like how to act if: The street is empty, semi-crowded or extremely crowded? There are any emergency cars in any of the streets? There are any pedestrians who want to cross any of the streets?

With the technology at hand, the previously mentioned questions can be answered; therefore, it opens up the gates to new practical and intelligently efficient solutions to the transportation problems.



Answering the questions previously asked:

- How to act if the street is empty, semi-crowded or extremely crowded?
  - Turn on the green light to the streets that have more than one car in them
  - Change the green light timer according to the number of cars present in each street
  - A hybrid solution that combines turning on the green light to the streets that have more than one car in them and change the green light timer according to the number of cars present in each street
  
- How to act if there are any emergency cars in any of the streets?
  - Wait for its turn if the streets are not crowded
  - Interrupt the system to turn on the green light on the street where the emergency vehicle is
  - Keep the green light on until the emergency vehicle passes
  - A hybrid solution that combines interrupting the system to turn on the green light on the street where the emergency vehicle is and keeping the green light on until the emergency vehicle passes
  
- How to act if there are any pedestrians who want to pass any of the streets?
  - Turn on the red light to the lanes that routes the vehicle to the street that a pedestrian wants to pass
  - Interrupt the system if a pedestrian is stuck because he did not have a turn
  - Prioritize the pedestrians according to the maximum waiting time
  - Hybrid solution that combines all the previous solutions

To get the minimum average waiting time, some of the solutions mentioned above showed very good results in the simulations done. Therefore, these simulations were chosen to be implemented in this project.

The chosen solutions are:

1. Hybrid solution that combines turning on the green light to the streets that have more than one car in them and change the green light timer according to the number of cars present in each street
2. Hybrid solution that combines interrupting the system to turn on the green light on the street where the emergency vehicle is and keeping the green light on until the emergency vehicle passes
3. Hybrid solution that combines turning on the red light to the lanes that routes the vehicle to the street that a pedestrian wants to pass, interrupting the system if a pedestrian is stuck because he did not have a turn and prioritizing the pedestrians according to the maximum waiting time

## **Scenarios**

### **Design Scenarios**

In this intelligent traffic light system there are two types of scenarios:

- Normal scenarios
- Special scenarios

### **Normal scenarios**

1. All the three streets are empty with/without pedestrians.
2. Having cars in only one of the three streets and the other two are empty.
3. Having cars in two of the streets, and the third is empty with/without pedestrians.
4. Crowded streets; have cars in the three streets with/without pedestrians.
5. Having cars in one or more of the three streets with pedestrians.

### **Special scenarios**

1. There is an emergency car in one of the streets
2. Have emergency cars in more than one street

**Dealing and switching between scenarios:**

By default each traffic light will be green for time ' $t_i$ ' then yellow for time ' $t_{i1}$ '

**Scenario #1:** All streets are empty (Normal scenario #1), the traffic light of all streets will be red and the pedestrians will be allowed to pass even if there are no pedestrians present, until any car arrives at any of the three streets then a green light will be turned on to this street.

Switch from Normal scenario#1 to Normal scenario#2 or Normal scenario#3.

**Scenario #2:** Having cars only in one of the three streets (Normal scenario #2), its traffic light will be green all the time, until any car arrives at any of the other two streets or arrive at both of them. At this instance the yellow light of this street begins counting ( $t_{i1}$ ) (given that  $t_i$  has finished counting). Then start switching the green light between streets.

Switch from Normal scenario#2 to Normal scenario#3 or Normal scenario#4.

**Scenario #3:** Having cars in only two of the streets, so we will switch between these two traffic lights only.

**Scenario #4:** Having crowded streets, so we will switch between the three traffic lights.

**Scenario #5:** Having cars in one or more of the three streets with pedestrians; for each green light switching:

- Check the priority
  - Based on maximum waiting time of the pedestrians.
- Determine which street to be crossed by pedestrians.
- Turn on left/right green light based on in which street the pedestrians are passing.

**If a special scenario occurs:**

**Scenario #6:** If an emergency car arrives at the street with the green traffic light, it will not turn off the green light even if its green count  $t_i$  finishes.

**Scenario #7:** If the traffic light is green and an emergency car arrives at any of the other two streets, it switches directly to yellow regardless its green counter  $t_i$  and after  $t_{li}$  counts, the traffic light of the street with the emergency car turns green until the emergency passes.

**Scenario #8:** If an emergency car arrives at a street with the yellow traffic light, after  $t_{li}$  finishes, it switches to green again (in case that there is no emergency car in the other two streets).

**Scenario #9:** If an emergency car arrives after another one has arrived the latter will have to wait until the former emergency car passes the intersection.

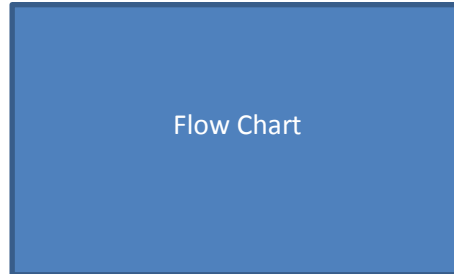
## 2.1.2. System Flowchart

There is no better way to describe a system other than a flow chart to demonstrate how the system changes states depending on the values of many variables within the system itself.

The flowchart shown in figure 2.5 describes the whole system.

Table 2.4 Flowchart Variable Definitions

<u>Variable</u>	<u>Physical Meaning</u>
Green 1	Street #1 Green Traffic Light
Green 1 Left	Street #1 Turn Left Green Traffic Light
Green 1 Right	Street #1 Turn Right Green Traffic Light
Yellow 1	Street #1 Yellow Traffic Light
Green 2	Street #2 Green Traffic Light
Green 2 Left	Street #2 Drive Straight Green Traffic Light
Green 2 Right	Street #2 Turn Right Green Traffic Light
Yellow 2	Street #2 Yellow Traffic Light
Green 3	Street #3 Green Traffic Light
Green 3 Left	Street #3 Turn Left Green Traffic Light
Green 3 Right	Street #3 Drive Straight Green Traffic Light
Yellow 3	Street #3 Yellow Traffic Light
All Red	Red Traffic Light for All Streets
$X_1, X_2, X_3$	Vehicle Presence Indicator for Streets 1,2,3
$X_4, X_5, X_6$	Emergency Vehicle Presence Indicator for Streets 1,2,3
$T_1, T_2, T_3$	Green Light Timer for Streets 1,2,3
$T_{11}, T_{12}, t_{13}$	Yellow Light Timer for Streets 1,2,3
$T_p$	Pedestrians Crossing Timer
$T_{P51}, T_{P52}, T_{P53}$	Pedestrians Waiting Timer for Streets 1,2,3
E	Emergency Vehicle Presence Indicator in any Street
$P_1, P_2, P_3$	Pedestrians Presence Indicator for Streets 1,2,3
$P_{r1}, P_{r2}, P_{r3}$	Pedestrians Priority for Streets 1,2,3
$N_1, N_2, N_3$	State Indicator for Streets 1,2,3



**Figure 2.9** Flow Chart of the proposed solution

## 2.1.3. System Boolean Equations

$$\begin{aligned}
 \text{Green 1} \equiv & (Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot X_4 + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot \overline{X_3} + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot X_4 \\
 & + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot X_1 + Y_3 \cdot T_{13} \cdot X_4 \\
 & + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot X_1 + R \cdot (T_p + E) \cdot X_4 \\
 & + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot N_1) \cdot (\overline{P_2} \cdot \overline{P_3})
 \end{aligned}$$

$$\begin{aligned}
 \text{Green 1 Left} \equiv & (Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot X_4 + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot \overline{X_3} \\
 & + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot X_4 + Y_2 \cdot T_{12} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot X_1 + Y_3 \cdot T_{13} \cdot X_4 \\
 & + Y_3 \cdot T_{13} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p53}} \cdot X_1 + R \cdot (T_p + E) \cdot X_4 \\
 & + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot N_1) \cdot P_{r3} \cdot P_3
 \end{aligned}$$

*Green 1 Right*

$$\begin{aligned}
 \equiv & (Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot X_4 + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot \overline{X_3} \\
 & + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot X_4 + Y_2 \cdot T_{12} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot X_1 \\
 & + Y_3 \cdot T_{13} \cdot X_4 + Y_3 \cdot T_{13} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p53}} \cdot X_1 + R \cdot (T_p + E) \cdot X_4 \\
 & + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot N_1) \cdot P_{r2} \cdot P_2
 \end{aligned}$$

$$\begin{aligned}
 \text{Yellow 1} \equiv & (\overline{T_{p51} + T_{p52} + T_{p53}} \cdot \overline{E} \cdot \overline{X_4} \cdot (X_2 + X_3) \cdot (T_1 + X_5 + X_6 + \overline{X_1}) \\
 & + (T_{p51} + T_{p52} + T_{p53}) \cdot \overline{E} \cdot T_{p51}
 \end{aligned}$$

$$\begin{aligned}
 \text{Green 2} \equiv & (Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot X_5 + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot \overline{X_1} + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot X_5 \\
 & + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot X_2 + Y_1 \cdot T_{11} \cdot X_5 \\
 & + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot X_2 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot X_5 \\
 & + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot (\overline{P_1} \cdot \overline{P_3})
 \end{aligned}$$

$$\begin{aligned}
 \text{Green 2 Left} \equiv & (Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot X_5 + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot \overline{X_1} \\
 & + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot X_5 + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot X_2 + Y_1 \cdot T_{11} \cdot X_5 \\
 & + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot X_2 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot X_5 \\
 & + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot P_{r1} \cdot P_1
 \end{aligned}$$

*Green 2 Right*

$$\begin{aligned}
&\equiv (Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot X_5 + Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p52}} \cdot \overline{X_3} \cdot \overline{X_1} \\
&+ Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot X_5 + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot X_2 + Y_1 \cdot T_{11} \cdot X_5 \\
&+ Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot X_2 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot X_5 \\
&+ R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot P_{r3} \cdot P_3
\end{aligned}$$

$$\begin{aligned}
\text{Yellow 2} &\equiv (\overline{T_{p51} + T_{p52} + T_{p53}}) \cdot \overline{E} \cdot \overline{X_5} \cdot (X_1 + X_3) \cdot (T_2 + X_4 + X_6 + \overline{X_2}) \\
&+ (T_{p51} + T_{p52} + T_{p53}) \cdot \overline{E} \cdot T_{p52}
\end{aligned}$$

$$\begin{aligned}
\text{Green 3} &\equiv (Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot \overline{X_2} + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot X_6 \\
&+ Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot X_3 + Y_2 \cdot T_{12} \cdot X_6 \\
&+ Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p53}} \cdot X_3 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 \\
&+ R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot (\overline{P_1} \cdot \overline{P_2})
\end{aligned}$$

$$\begin{aligned}
\text{Green 3 Left} &\equiv (Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot \overline{X_2} \\
&+ Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot X_6 + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot X_3 + Y_2 \cdot T_{12} \cdot X_6 \\
&+ Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p53}} \cdot X_3 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 \\
&+ R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot P_{r2} \cdot P_2
\end{aligned}$$

*Green 3 Right*

$$\begin{aligned}
&\equiv (Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 + Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{T_{p53}} \cdot \overline{X_1} \cdot \overline{X_2} \\
&+ Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot X_6 + Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{T_{p51}} \cdot \overline{X_2} \cdot X_3 + Y_2 \cdot T_{12} \cdot X_6 \\
&+ Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{T_{p53}} \cdot X_3 + R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot X_6 \\
&+ R \cdot (T_p + E) \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{N_1} \cdot \overline{N_2}) \cdot P_{r1} \cdot P_1
\end{aligned}$$

$$\begin{aligned}
\text{Yellow 3} &\equiv (\overline{T_{p51} + T_{p52} + T_{p53}}) \cdot \overline{E} \cdot \overline{X_6} \cdot (X_1 + X_2) \cdot (T_3 + X_4 + X_5 + \overline{X_3}) \\
&+ (T_{p51} + T_{p52} + T_{p53}) \cdot \overline{E} \cdot T_{p53}
\end{aligned}$$

$$\begin{aligned}
\text{All Red} &\equiv (Y_1 \cdot T_{11} \cdot \overline{X_5} \cdot \overline{X_6} \cdot \overline{X_4} \cdot T_{p51}) + (Y_2 \cdot T_{12} \cdot \overline{X_6} \cdot \overline{X_4} \cdot \overline{X_5} \cdot T_{p52}) \\
&+ (Y_3 \cdot T_{13} \cdot \overline{X_4} \cdot \overline{X_5} \cdot \overline{X_6} \cdot T_{p53})
\end{aligned}$$



## 2.2. Simulation

### 2.2.1. SUMO

"Simulation of Urban MObility", or "SUMO" for short, is an open source, microscopic, multi-modal traffic simulation. It allows to simulate how a given traffic demand which consists of single vehicles moves through a given road network. The simulation allows addressing a large set of traffic management topics. It is purely microscopic: each vehicle is modeled explicitly, has an own route, and moves individually through the network. Simulations are deterministic by default but there are various options for introducing randomness.

The development of SUMO started in the year 2000. The major reason for the development of an open source, microscopic road traffic simulation was to support the traffic research community with a tool with the ability to implement and evaluate own algorithms. This tool has no need for regarding all the needed things for obtaining a complete traffic simulation such as implementing and/or setting up methods for dealing with road networks, demand, and traffic controls. By supplying such a tool, the DLR wanted to make the implemented algorithms more comparable by using a common architecture and model base, and gain additional help from other contributors.

Two major design goals are approached: the software shall be fast and shall be portable. Due to these goals, the very first versions were developed to be run from the command line only - no graphical interface was supplied at first and all parameter had to be inserted manually. This increased the execution speed by leaving off slow visualization.

In addition, due to these goals, the software was split into several parts. Each of them has a certain purpose and must be run individually. This is something that makes SUMO different to other simulation packages where, for instance, the dynamical user assignment is made within the simulation itself, not via an external application like SUMO. This split allows an easier extension of each of the applications within the package because each is smaller than a monolithic application that does everything.

Moreover, it allows the usage of faster data structures, each adjusted to the current purpose, instead of using complicated and ballast-loaded ones. Still, this makes the usage of SUMO a little bit uncomfortable in comparison to other simulation packages. As there are still other things to do.

The features of SUMO:

- Includes all applications needed to prepare and perform a traffic simulation (network and routes import and simulation)
- Simulation
  - Space-continuous and time-discrete vehicle movement
  - Different vehicle types
  - Multi-lane streets with lane changing
  - Different right-of-way rules, traffic lights
  - A fast OpenGL graphical user interface
  - Manages networks with several 10,000 edges (streets)
  - Fast execution speed (up to 100,000 vehicle updates/s on a 1GHz machine)
  - Interoperability with other application at run-time
  - Network-wide, edge-based, vehicle-based, and detector-based outputs
  - Supports person-based inter-modal trips
- Network Import
  - Imports VISUM, Vissim, Shapefiles, OSM, RoboCup, MATsim, OpenDRIVE, and XML-Descriptions
  - Missing values are determined via heuristics
- Routing
  - Microscopic routes - each vehicle has an own one
  - Different Dynamic User Assignment algorithms
- High portability
  - Only standard c++ and portable libraries are used
  - Packages for Windows main Linux distributions exist
- High interoperability through usage of XML-data only
- Open source (GPL)

### **2.2.2. TraCI4Matlab**

TraCI4Matlab is an API (Application Programming Interface) developed in Matlab that allows the communication between any application developed using Matlab and the urban traffic simulator SUMO. The functions that comprise TraCI4Matlab implement the TraCI (Traffic Control Interface) application level protocol, which is built on top of the TCP/IP stack, so the application developed in Matlab, which is the client, can access and modify the simulation environment provided by the server (SUMO). TraCI4Matlab allows controlling SUMO objects such as vehicles, traffic lights, junctions, etc, enabling applications like traffic lights predictive control and dynamic route assignment, among others.

### **2.2.3. Simulation Description**

After installing SUMO and TraCI4Matlab, a simulation of the suggested solutions can be done to provide many results; such as:

- Making sure the suggested solutions could be implemented in real life
- Calculating the average waiting time of all vehicles
- Getting the minimum average waiting time for all vehicles
- Getting the mean and variance of vehicles each second
- Run more than one simulation with different scenarios and compare between them

#### 2.2.4. Simulation Steps

1. Create node file that defines the borders of the streets
2. Create edge files that defines the streets connection with nodes
3. Create connection file that adds pedestrian crossings and limit lanes to certain turns
4. Convert the three files mentioned in the first three steps into one network file
5. Create a route file that assigns the flow of vehicles and pedestrians
6. Write the traffic light control equations using Matlab
7. Write the function that allows Matlab to gather data concerning the number of vehicles on every street
8. Write the function that allows Matlab to gather data concerning the speed of each vehicle on every street
9. By using the above mentioned functions, Matlab can control the traffic lights using the count of the vehicles gathered, also Matlab can calculate the average waiting time by observing the speed of each vehicle and, if the speed is below a certain threshold, it starts counting the number of seconds till the vehicle passes the intersection
10. Get the average waiting time for all vehicles on all streets for a certain observing time
11. Additional steps:
  - a. Create two loops
    - i. One for the number of vehicles
    - ii. One for the green light timer
  - b. Run the above mentioned program
  - c. Save the average waiting times calculated in a matrix
  - d. Get the minimum average waiting time for every row (which shows the average number of cars generated in a certain observing time interval)

### 2.2.5. Simulation Results

To simulate this project, one has to create all scenarios that could happen to allow for a variety of choices among green light timings. The scenarios implemented are:

1. The ordinary (static) traffic light system
2. Intelligent traffic light system without pedestrians
3. Intelligent traffic light system with pedestrians
4. Intelligent traffic light system with pedestrians with generation of vehicles in two streets only

The result of the above mentioned scenarios will be shown in the following figures. For all scenarios the observed time interval is one hour and the green lime timings range from 15 seconds to 75 seconds. The four scenarios' results are shown in one plot to make it easier to compare among them.

Figure 2.6 shows the results of four scenarios when the probability of the generation of vehicles is 50 vehicles/hour

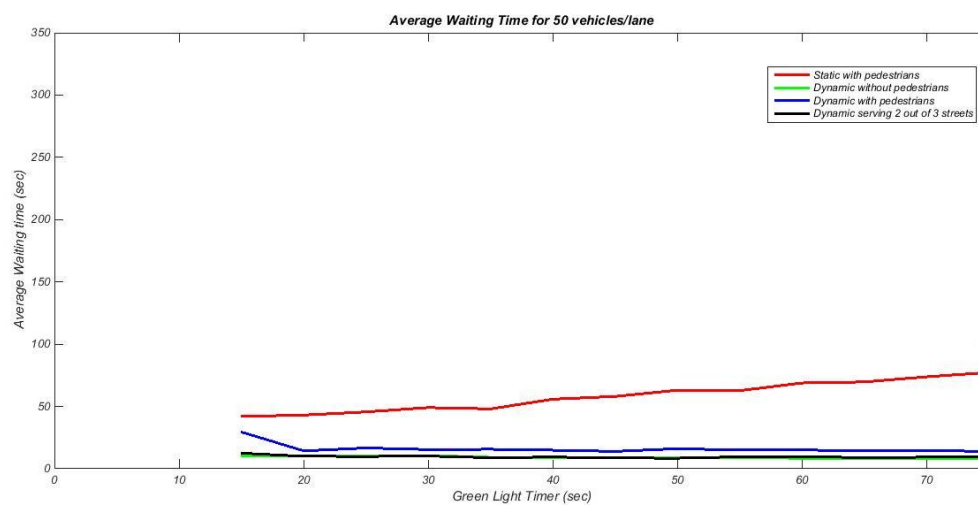


Figure 2.10 Results of the simulation for 50 vehicles per hour

Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	41.98 Seconds	15 Seconds
Dynamic without pedestrians	9.181 Seconds	35 Seconds
Dynamic with pedestrians	13.94 Seconds	45 Seconds
Dynamic serving 2 streets	8.864 Seconds	35 Seconds

Figure 2.7 shows the results of four scenarios when the probability of the generation of vehicles is 100 vehicles/hour

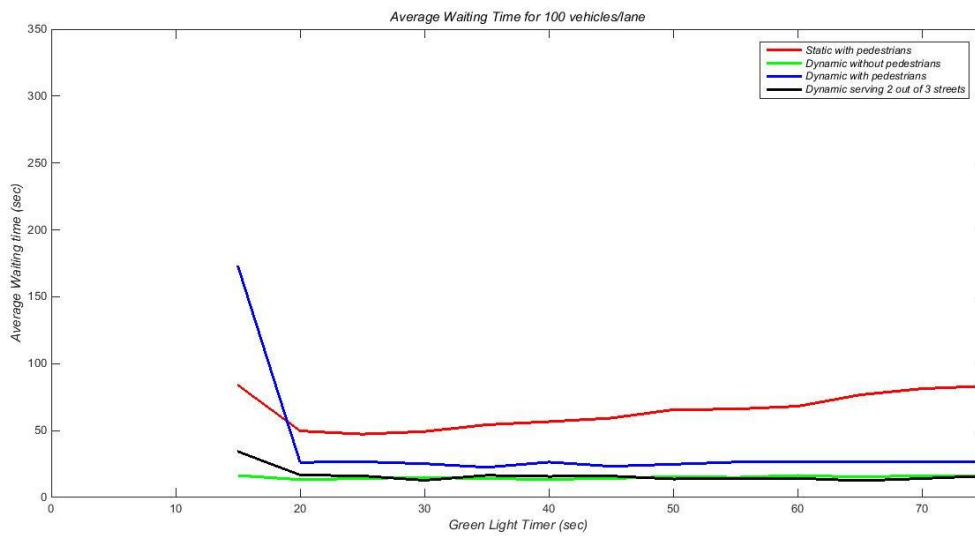


Figure 2.11 Results of the simulation for 100 vehicles per hour

Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	47.37 Seconds	25 Seconds
Dynamic without pedestrians	13.65 Seconds	40 Seconds
Dynamic with pedestrians	22.56 Seconds	35 Seconds
Dynamic serving 2 streets	12.86 Seconds	30 Seconds

Figure 2.8 shows the results of four scenarios when the probability of the generation of vehicles is 150 vehicles/hour

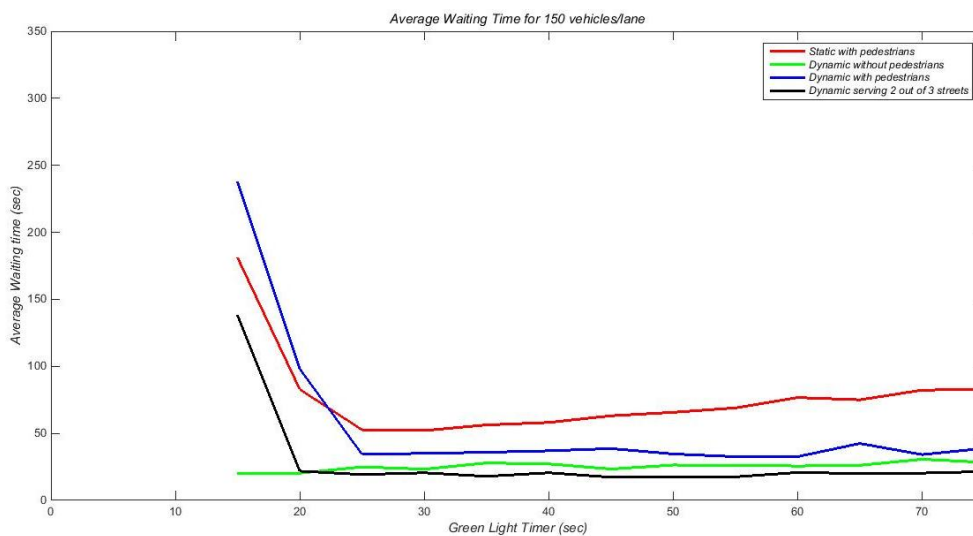


Figure 2.12 Results of the simulation for 150 vehicles per hour

Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	52.07 Seconds	30 Seconds
Dynamic without pedestrians	23.28 Seconds	30 Seconds
Dynamic with pedestrians	32.61 Seconds	55 Seconds
Dynamic serving 2 streets	17.94 Seconds	35 Seconds

Figure 2.9 shows the results of four scenarios when the probability of the generation of vehicles is 200 vehicles/hour

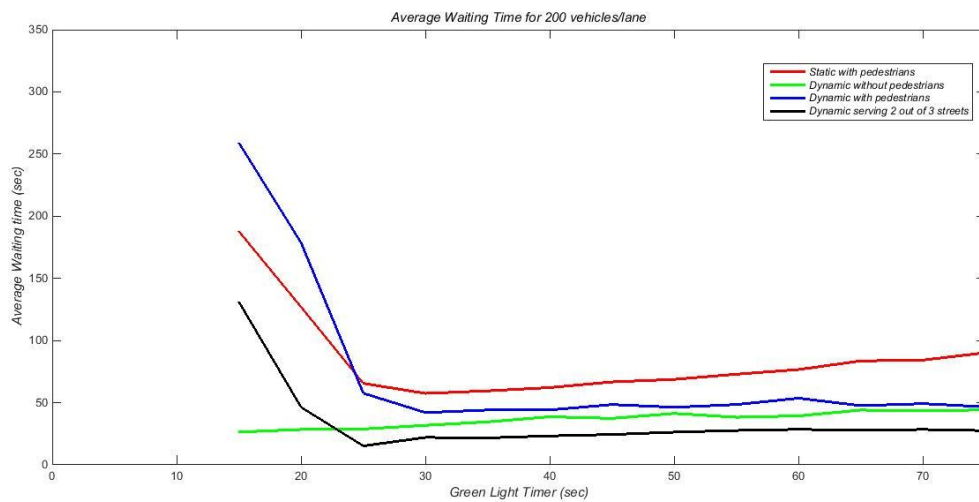


Figure 2.13 Results of the simulation for 200 vehicles per hour

Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	57.47 Seconds	30 Seconds
Dynamic without pedestrians	28.83 Seconds	25 Seconds
Dynamic with pedestrians	42.04 Seconds	30 Seconds
Dynamic serving 2 streets	21.7 Seconds	35 Seconds

Figure 2.10 shows the results of four scenarios when the probability of the generation of vehicles is 250 vehicles/hour

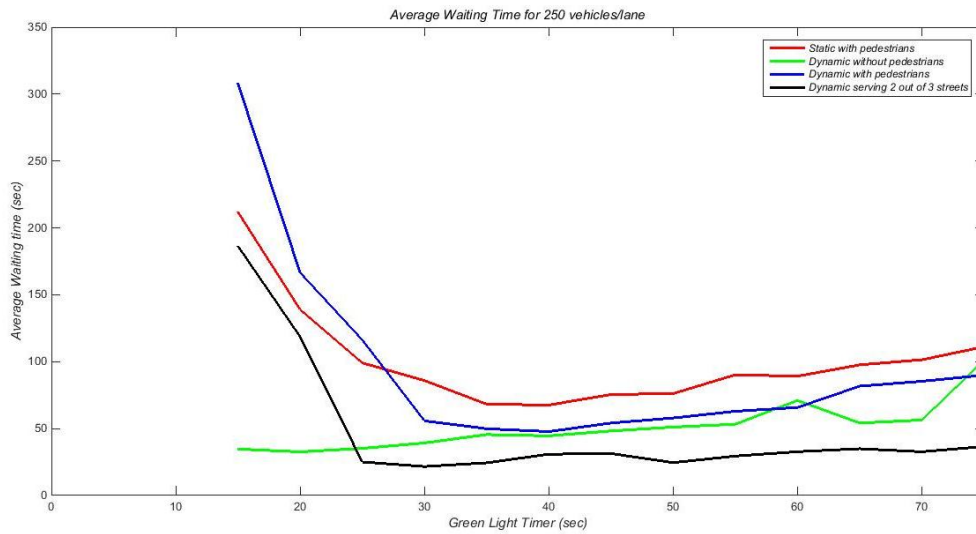


Figure 2.14 Results of the simulation for 250 vehicles per hour

Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	67.61 Seconds	40 Seconds
Dynamic without pedestrians	32.63 Seconds	20 Seconds
Dynamic with pedestrians	47.77 Seconds	40 Seconds
Dynamic serving 2 streets	21.65 Seconds	30 Seconds

Figure 2.11 shows the results of four scenarios when the probability of the generation of vehicles is 300 vehicles/hour

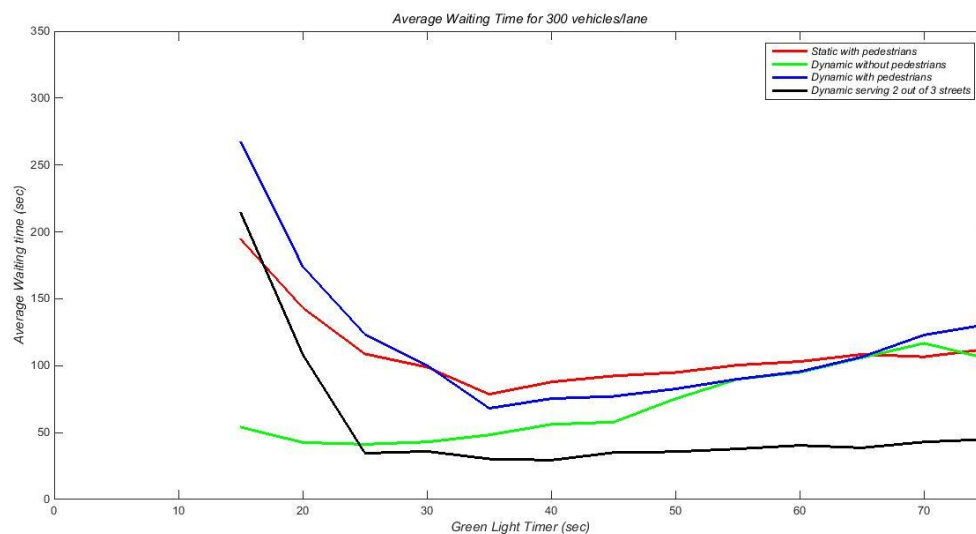


Figure 2.15 Results of the simulation for 300 vehicles per hour



Simulation name	Minimum average waiting time	Green Time
Static with pedestrians	78.66 Seconds	35 Seconds
Dynamic without pedestrians	41.28 Seconds	25 Seconds
Dynamic with pedestrians	68.18 Seconds	35 Seconds
Dynamic serving 2 streets	29.41 Seconds	40 Seconds

One can observe from the results shown in the previous figures that the intelligent traffic light system implemented works better when:

1. The number of vehicles generated per hour is low
2. The number of streets served is lower e.g. 1 or 2

The results shown in the previous figures are in the shape of a parabola-like with its minimum value somewhere in the middle of the graph. The reason for the parabola-like shape is that when the green light timer is too low in comparison with the number of vehicles, the average waiting time for each street will be high because a few number of vehicles will be allowed to pass in one iteration. However, when the green light timer increases that it becomes so high, it will make the other 2 streets wait more. Therefore, the minimum average waiting time will be somewhere in the middle.

For simulation number 2 (Intelligent Traffic light system without pedestrians), the graph is not as high as the other three graphs. The reason is that the pedestrians time is 10 seconds, and at the beginning the green light timer is 15 seconds. Therefore, there will always be one lane not allowed to pass except for 5 seconds, if there are pedestrians. If there are no pedestrians like simulation number 2, both lanes will be allowed to pass for the whole 15 seconds, so the average waiting time will be lower than the other three simulations.

In order to solve the problem that the system does not work efficiently when large number of cars per hour is generated, the dynamic green light timing is used. This hybrid system works using some conditions:

- Uses the same equations mentioned before
- Counts the number of vehicles and change the green light timer accordingly

The timings of the green light timer were easy to get from the previously shown results. After simulating one last time using the dynamic green light timing with the following conditions:

- The scenario used is intelligent traffic light system with pedestrians
- The flow generation of vehicles changes its probability randomly
- The timer changes at the instance of turning on a green light and stays constant until a switch of green light occurs

- If

$$\# \text{ of vehicles} / \text{street} \leq 4 \rightarrow \text{Timer} = 25 \text{ seconds}$$

- If

$$4 < \# \text{ of vehicles} / \text{street} \leq 10 \rightarrow \text{Timer} = 30 \text{ seconds}$$

- If

$$\# \text{ of vehicles} / \text{street} > 10 \rightarrow \text{Timer} = 40 \text{ seconds}$$

The average waiting time calculated after running this simulation with the improved condition is **42.0353 seconds** which improved the average waiting time of the 250 and 300 vehicles per hour.

# Chapter 3

---

## Software Setup

### 3.1. Graphical User Interface (GUI)

**T**his Section is concerned with how the end-user interacts with the database by either altering it or adding new fields to it such that the system is up to date and synchronized with all the vehicles interacting with it and maintaining a user friendly interface.

#### 3.1.1. Database

The database consists of 3 tables:

- Vehicles Database
- Notification Log
- Tracking Table

Each table contributes in its own use to provide the user and the system with the data and information required by either.

The vehicles database contains the following data about the vehicles involved in the system.

**Table 3.5 Vehicles' Database Fields and definitions**

<u>Title</u>	<u>Description</u>
SerialNumber	Serial Number of the RFID Tag on Vehicle
PlateNumber	Vehicle Plate Number
Category	What Type of Vehicle is it
Stolen	Whether or not the Vehicle is Reported Stolen
LicenseExpiryDate	Expiry Date of the vehicle's License
PersonalID	ID of the Vehicle Owner
LastSeenTime	What Time was the Vehicle Last Seen
LastSeenPlace	Where was the Vehicle Last Seen
VIN	Vehicle Identification Number (Chassis Number)

This table contains all the data needed concerning identifying each vehicle as a unique one and all the relative data that may come in good use. Whenever someone buys a new car/ sells it/ changes a detail about it, this table has to be updated immediately to match the real relative data.

The second table is the Notifications Log Table which has the following data:

**Table 3.6 Notification Log Database Fields and Definitions**

<u>Title</u>	<u>Description</u>
NotificationLog	Unique identifier to every time an antenna reads something
AntennaNumber	Identifies the antenna number which delivered the readings
SerialNumber	The serial number of the RFID tag read by the antennas
NotificationLogDate	The exact Date and Time this field was read

This table is basically controlled by the RFID reader and is not edited or altered manually by any user. However its data is very critical since it contains the antenna readings which are used to determine the state and the scenarios to be followed which were discussed in the previous chapter.

Moreover, this table is very important in making the vehicles tracking table which is very important to retrieve data from when a car is reported stolen in order to track and retrieve it as fast as possible.

The final table in the database is the Tracker table with the following data.

**Table 3.7 Tracker Database Fields and Definitions**

Title	Description
Serial Number	The serial number of the RFID tag read by the antennas
TrafficLightNumber	The Traffic Light ID where the antenna made the reading
TimeSeen	The Exact time and date where the vehicle was seen

This table basically records the vehicle movements and updates itself from the data in the notifications log. The table updates itself if the field it is checking falls under one of the following conditions:

- The Serial number is not found in the tracker table
- The Serial number is found already but it has been more than two minutes since it was last recorded in the table
- The Serial number is found already but the vehicle has been spotted in a new traffic light.


With the three discussed tables the database is complete and ready to serve the system with all the information it needs to function properly.

### 3.1.2. Graphical User Interface (GUI)

#### 3.1.2.1. Adding New Cars to Database

Now a simple user friendly interface is ready to be used once the database has been set up. The users in this case are the employees of the General Directorate of Traffic. The GUI allows them to enter new cars to the system for example when someone buys a new car. The Scenario goes as follows:

- RFID tag properly planted into the vehicle
- Vehicle is placed in front of the reading antenna
- Only one car should be detected by the reading antenna, otherwise an error message shows up and the database is not updated
- If no car is in front of the antenna, another error message appears and the database is not updated
- If one of the vehicle's field information is missing an error message appears and the database is no updated.



The image shows a screenshot of a graphical user interface window titled "New Car Registration". The window contains several input fields for data entry:

- PersonalID**: A text input field.
- Plate Number**: A text input field.
- Category**: A dropdown menu with the text "Please select a category..." and a downward arrow.
- VIN**: A text input field.
- Expiry Date**: Three dropdown menus for the day (dd), month (mm), and year (yyyy).

A "Register" button is located at the bottom right of the form area.

Figure 3.16 shows the GUI with the data needed to be entered for a successful database update

Once the above steps are properly executed the vehicles table in the database is updated and the new vehicle is now ready to be served by the Intelligent Traffic System.

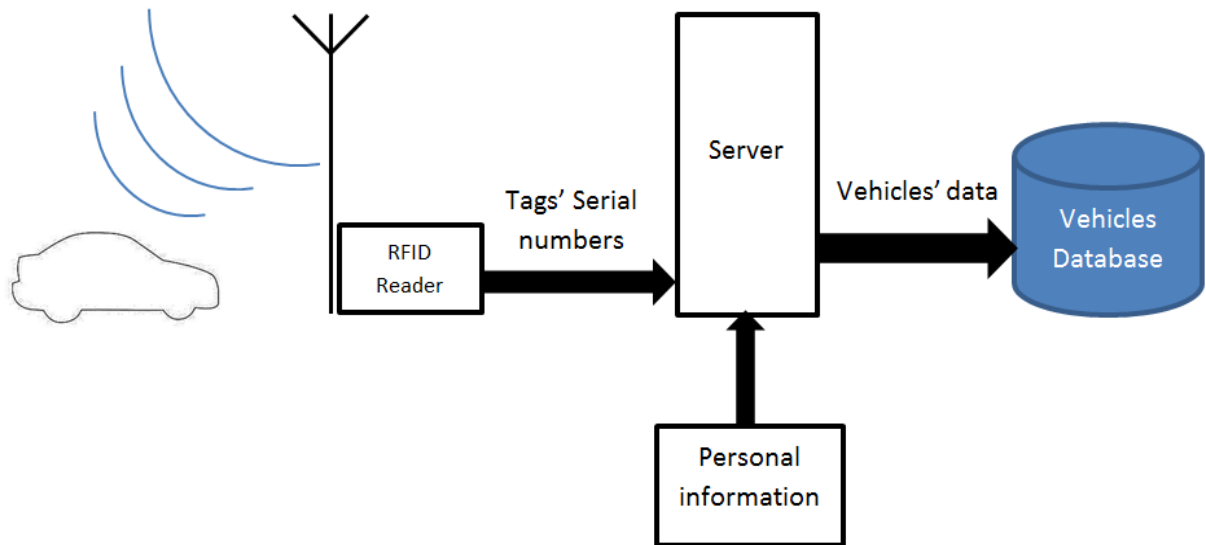


Figure 3.17 shows how the data of a vehicle enters the Vehicles Database Table

### 3.1.2.2. Tracking Vehicles

When disaster strikes and somebody's vehicle has been stolen, the obvious step that the owner takes is that they report that to the police department. In turn, this is reported to the General Directorate of Traffic where the employee there uses the GUI to update the 'stolen' status in the vehicles table as shown in figure 3.3.

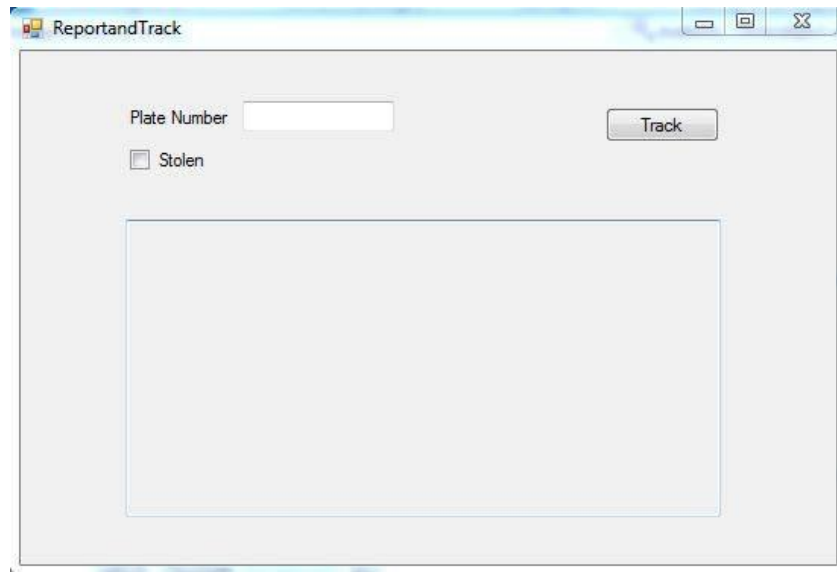


Figure 3.18 shows the GUI of the tracking and reporting vehicles

Basically all what happens is:

- The employee enters the vehicle's Plate Number and checks the Stolen checkbox
- The GUI updates the status of the car in the vehicles table to 'stolen'
- When the employee presses the track button, the GUI retrieves the data from the Tracker table to show where was the car recently seen and when.
- If the checkbox is un-ticked then the GUI only shows tracking information without reporting it

The data is then used by the police to track down the stolen car and retrieve it whilst consuming much less time and effort by limiting the areas they need to search for or even follow the driving pattern to expect where to find the car. Figure 3.4 explains how the tracking of the vehicles work.

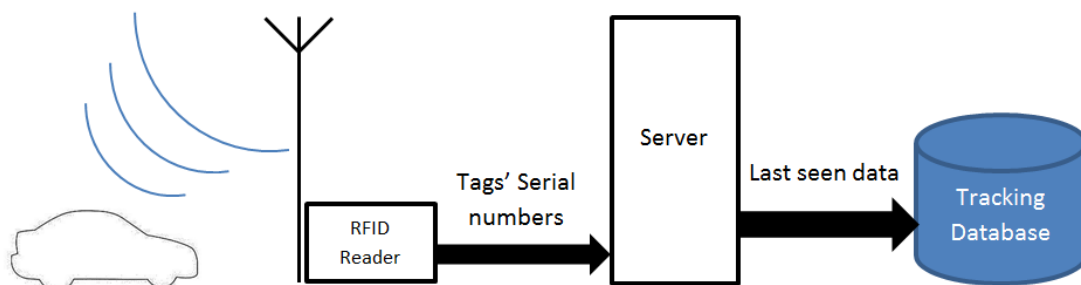


Figure 3.19 shows how the data of a vehicle enters the Tracking database table



## 3.2. LabVIEW

### 3.2.1. LabVIEW blocks description

#### Definitions for some Signals:

- E: It represents the existence of any emergency ( $X4 + X5 + X6$ )
- Z: It represents any interrupt from pedestrians ( $T_{p51} + T_{p52} + T_{p53}$ )
- All\_Red: It represents the case when interrupt occurs. All traffic red lights are ON

#### Start

It is considered as the initial condition of our system. The initial condition includes:

- The traffic light is Green for street #1, and is Red for the other two streets by setting
  - M1 and N1 = "True", M11 and M12 = "False", M2 = "False" and R1="False"
  - M3 and N2 = "False", M31 and M32 = "False", M4 = "False" and R2="True"
  - M5 and N3 = "False", M51 and M52 = "False", M6 = "False" and R3="True"
- All the timers are reset (T1, T11, T2, T12, T3, T13, Tp, Tp51, Tp52, Tp53)
- All priorities are set to "False".

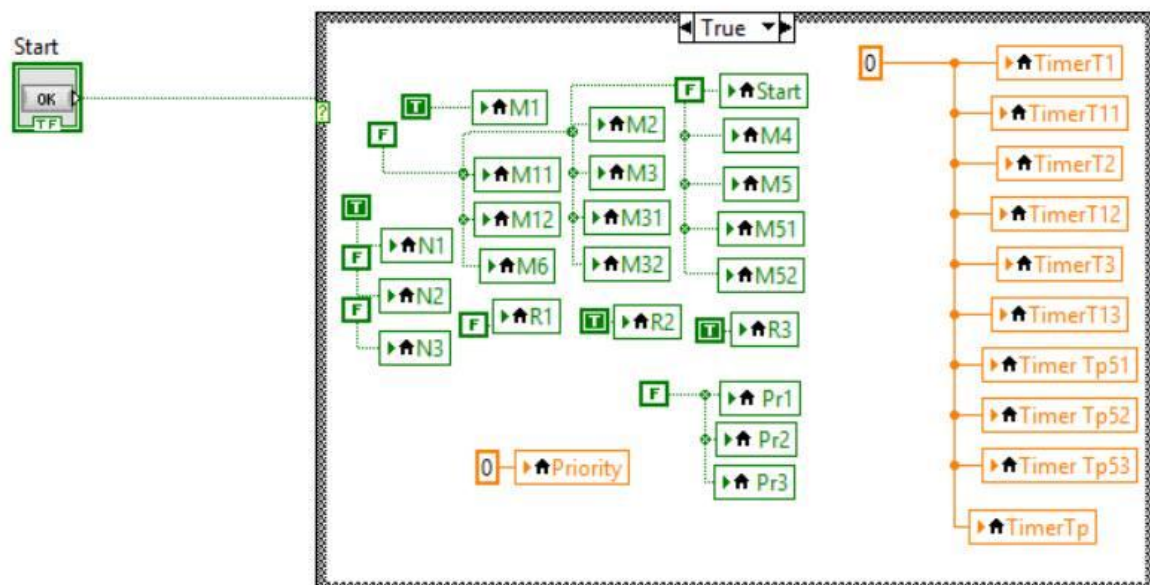


Figure 3.20 Start block

### Green light for street #1

There are three signals.

- M1: It is a signal of the normal case (i.e. when both right traffic light and left traffic light of street #1 are ON)
- M11: It is a signal when the left traffic light only is ON (i.e. there are pedestrians in street #3)
- M12: It is a signal when the right traffic light only is ON (i.e. there are pedestrians in street #2)

### Green light for both lanes

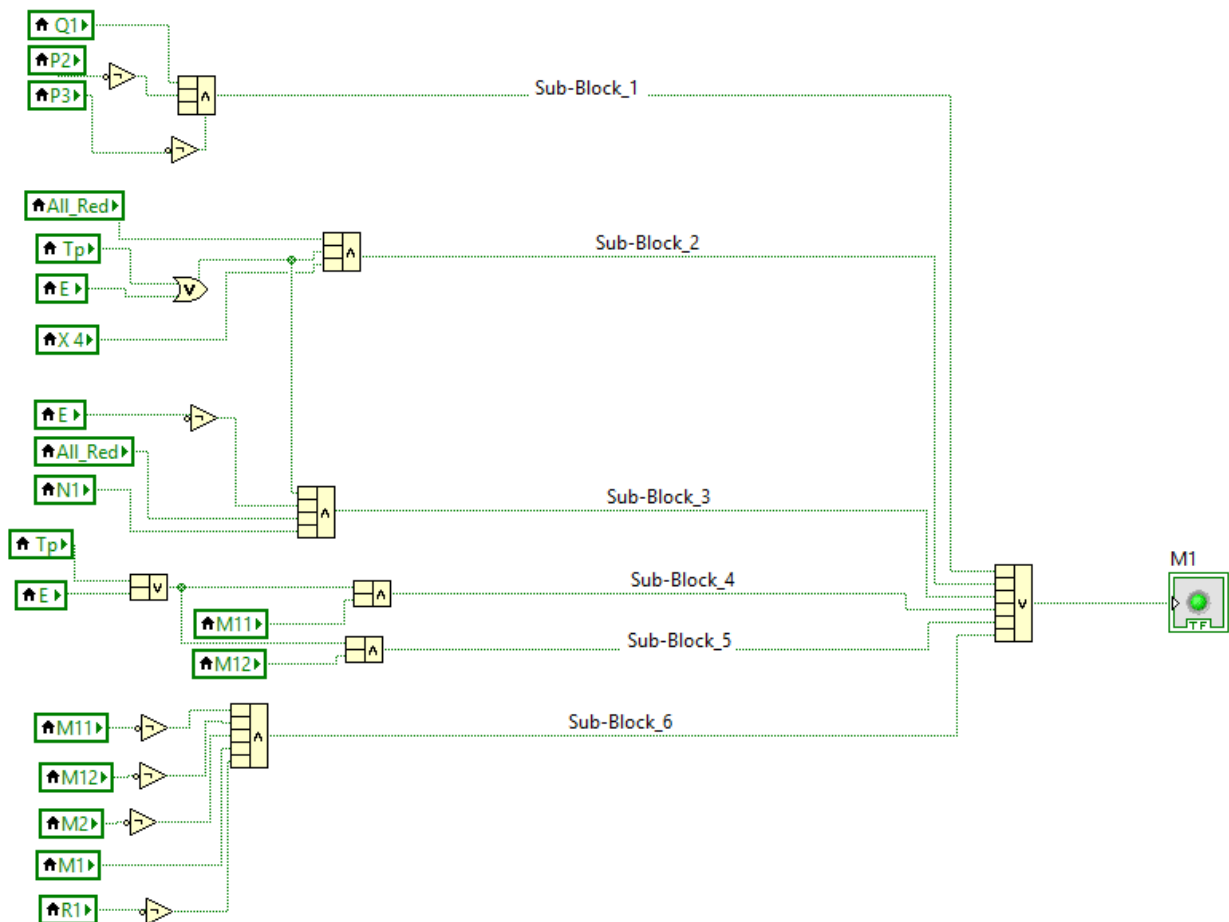


Figure 3.21 Green traffic light for street #1

M1 consists of 6 sub-blocks; each sub-block represents a case that will turn on the green light for Street1 as shown in figure 3.22. These cases do not have to be satisfied together. If only one case is satisfied, Green-1 will be turned ON. The following statements will discuss each sub-block in details.

**Sub-block 1**

Sub-block 1 is responsible for turning on the green light for street #1 with the assumption that there are no pedestrians in street #2 and street #3. There is a signal Q1 which has 6 conditions, if any of these conditions is satisfied while the condition that there are no pedestrians in street #2 and street #3 is also satisfied, the green light for street #1 will be turned on.

**Conditions of Q1:**

Illustrate the normal case without pedestrians. Figure 3.7 shows these conditions.

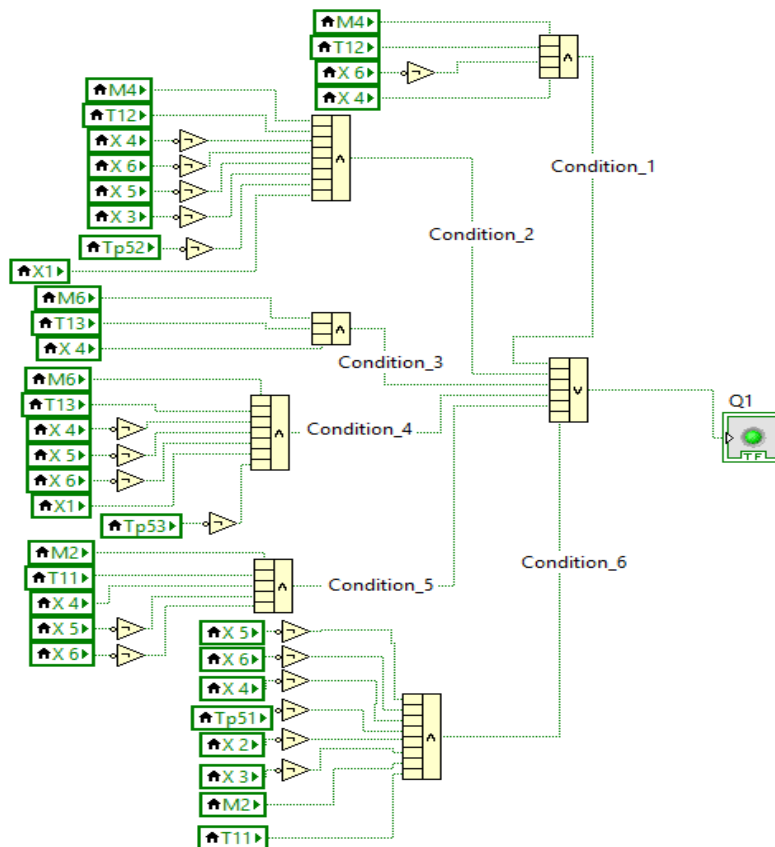


Figure 3.22 Conditions of Q1

**Condition #1:**

It states that if the current state is Yellow in street #2, its timer has been finished, and there is no emergency in the in-order street (Street #3), while there is an emergency in street #1. Subsequently, the green light of street #1 will be turned on.

**Condition #2:**

It states that if the current state is the Yellow of street #2, its timer has been finished, there is no emergency in any of the three streets, there are no cars in the in-order street (street #3), while there are cars in street #1, and that there is no interrupt from pedestrians in street #2 (assuming that there are no pedestrians). Subsequently, the green light for street #1 will be turned on.

**Condition #3:**

It states that if the current state is the yellow light in street #3, and its timer has been finished, and there is an emergency in street #1, hence, the green light for street #1 will be turned on.

**Condition #4:**

It states that if the current state is the yellow light in street #3, yellow light's timer has been finished, there is no emergency in any of the three streets, and we have cars in street #1, given that there are no interrupts from pedestrians in street #3 (assuming that there are no pedestrians). Subsequently, the green light for street #1 will be turned on.

**Condition #5:**

It states that if the current state is yellow light in street #1, and its timer has been finished, while an emergency in this street (street #1) has come during the Yellow light time, and there are no other emergencies in the other two streets, then the green light will be turned on again for street #1.

**Condition #6:**

It states that if the current state is yellow light in street #1, its timer has been finished, there are no emergencies in any of the three streets, there are no cars in street #2 and street #3, and there is no interrupt from pedestrians in street #1, Therefore, the green for street #1 will be turned on.

I.e. This case happens when a car that has been read (by RFID reader) in street #2 or street #3, illegally crosses the road (Violates the traffic light), or there is an instantaneous false reading by the RFID reader (rarely happens).

**Sub-block 2**

Sub-block 2 is responsible for turning on the green light for street #1, when the current state is the state that all traffic lights are Red, and the timer for the pedestrians to cross the street is finished or there is an emergency in street #1.

**Sub-block 3**

Sub-block 3 is responsible for turning on the green light for street #1, when the current state is the state that all traffic lights are Red, the timer for pedestrians to cross the street is finished, and there are no emergencies in any of the three streets. In addition, the previous state was the green of street #1. The Previous state is indicated by the signal N1.

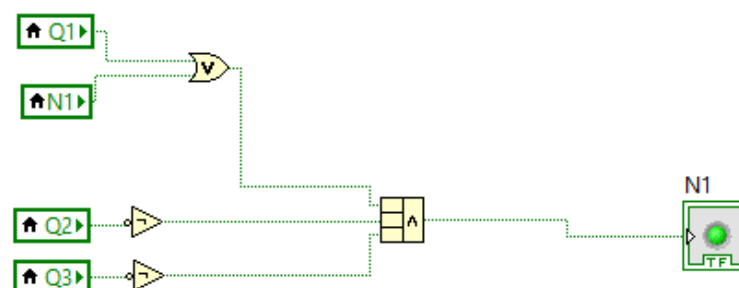
**Conditions of signal N1:**

Figure 3.23 Conditions of N1

The signal N1 is the same as the signal of Q1 but with a self-latch and an exit condition. So, it is always “true” if the state did not change either to Green light of street #2 (Q2= “true”) or Green light of street #3 (Q3 = “true”) as shown in figure 3.23.

#### **Sub-block 4**

Sub-block 4 is responsible for turning on the green light of street #1, when the current state is the Left Green light state, and there is an emergency or the time for pedestrians to cross the street is finished. Thus, we return to green light of street #1.

#### **Sub-block 5**

Sub-block 5 is responsible for turning on the green light of street #1, when the current state is the Right Green light state, and there is an emergency or the time for the pedestrians to cross the street is finished. Thus, we return to green light of street #1.

#### **Sub-block 6**

Sub-block 6 is responsible for the self-latch and exit of M1. M1 will keep its value “True”, whenever Left Green light, Right Green light, Yellow light of street #1 and Red light of street #1 are not changed to become “True”.

### Green light for the left lane

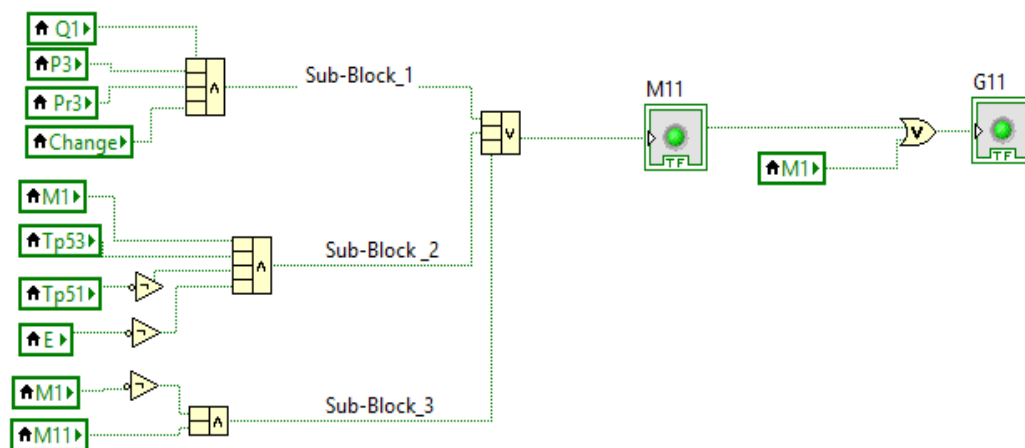


Figure 3.24 Conditions of the left lane green traffic light for street #1

M11 consists of 3 sub-blocks; each sub-block represents a case that will turn on the Left Green traffic light only in street #1 as shown in figure 3.24.

#### Sub-block 1

Sub-block 1 is responsible for turning M11 ON, when there are pedestrians in street #3, the priority is for pedestrians in street #3 and any of Q1 conditions is satisfied.

#### Sub-block 2

Sub-block 2 is responsible for turning M11 ON, when the current state is the state that both left and right green traffic light of street-1 are ON, there is an interrupt from pedestrians in street #3, there is no interrupt from pedestrians in street #1 and there is no emergency.

#### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M11. M11 will keep its value unless the state returns to M1.

G11 is the output signal of traffic light that turned ON if one of the following two cases is satisfied as shown in figure(3.24).

- The normal green traffic light is ON (M1)
- The left green traffic light is ON (M11)

### Green Light for the right lane

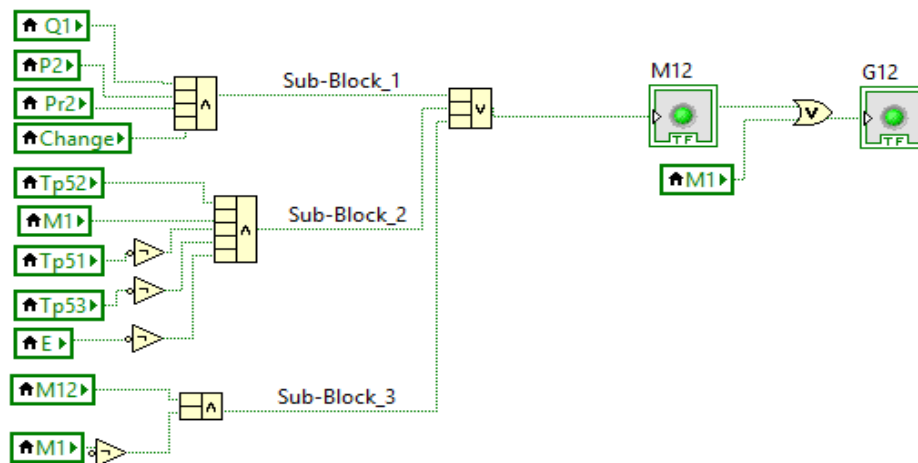


Figure 3.25 Conditions of right lane green traffic light for street #1

M12 consists of 3 sub-blocks; each sub-block represents a case that will turn on the right Green traffic light only in street #1 as shown in figure 3.25.

#### Sub-block 1

Sub-block 1 is responsible for turning M12 ON, when there are pedestrians in street #2, the priority is for pedestrians in street #2 and any of Q1 conditions is satisfied.

#### Sub-block 2

Sub-block 2 is responsible for turning M12 ON, when the current state is the state that both left and right green traffic light of street-1 are ON, there is an interrupt from pedestrians in street #2, there is no interrupt from both pedestrians in street #1 and street #3, and there is no emergency.

#### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M12. M12 will keep its value unless the state returns to M1.

G12 is the output signal of traffic light that turned ON if one of the following two cases is satisfied as shown in figure 3.25.

- The normal green traffic light is ON (M1)
- The left green traffic light is ON (M12)



### Yellow light for street #1

M2 represent the Yellow traffic light for street #1. It consists of 3 sub-blocks as shown in figure (3.26). If any condition of these sub-blocks is satisfied, the current state will change from Green traffic light to Yellow traffic light of street #1. In the following part each sub-block is discussed in details:

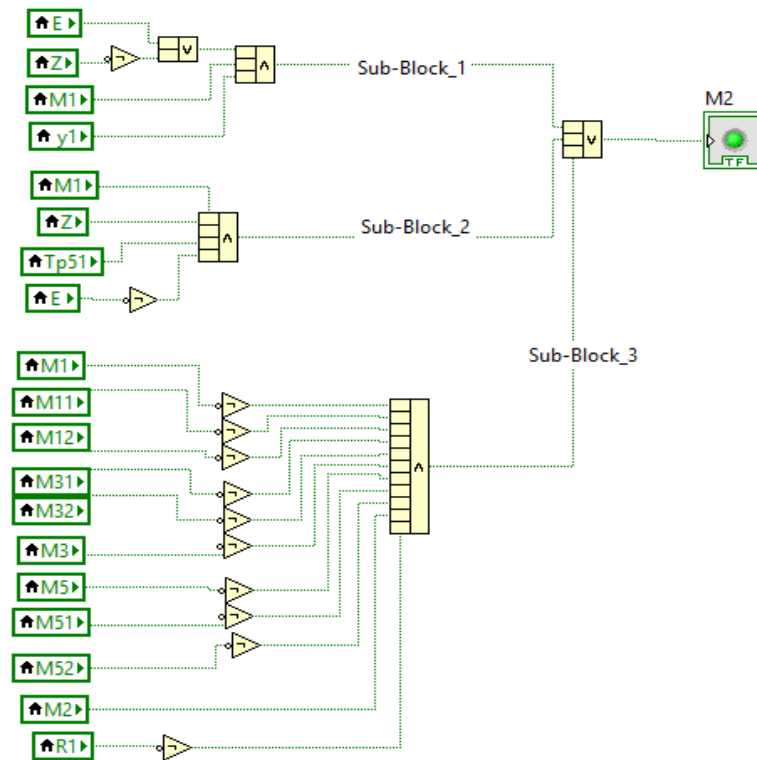


Figure 3.26 Conditions of yellow traffic light for street #1

### Sub-block 1

Sub-block 1 is responsible for turning M2 ON, when the current state is the green of street #1, without any interrupt from pedestrians or there is an emergency, and if any of the conditions of signal y1 shown in figure 3.27 is satisfied.

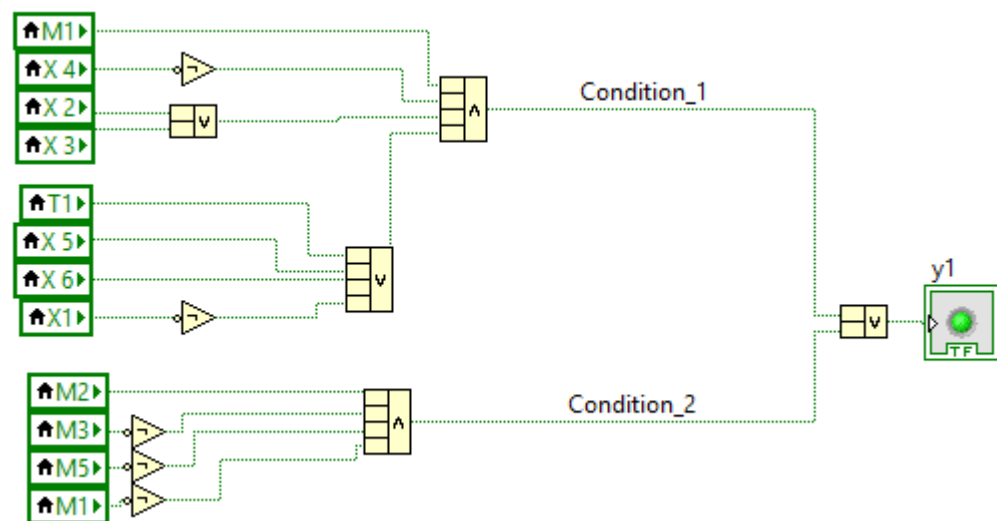
**Conditions of signal y1:**

Figure 3.27 Conditions of signal Y1

**Condition #1:**

It states that if the current state is the green of street #1 with no emergencies in street #1, and there are cars either in street #2 or in street #3, in addition, the timer of the green traffic light is finished or there is an emergency in either street #2 or street #3 or there are no cars in street #1, then y2 will be turned ON.

**Condition #2:**

It represents the self-latch and exit of y1. Y1 will hold its value unless, the green traffic light of any of the three streets turned ON.

**Sub-block 2**

Sub-block 2 is responsible for turning M2 ON, when the current state is the green of street #1, there is an interrupt from pedestrians in street #1, and there is no emergency in any of the three streets.

**Sub-block 3**

Sub-block 3 is responsible for the self-latch and exit of M2. M2 will keep its value unless, the green traffic light either Left green, right green or both of them of any of the three streets turned on or the Red traffic light of street #1 is ON.

**Red light for street #1**

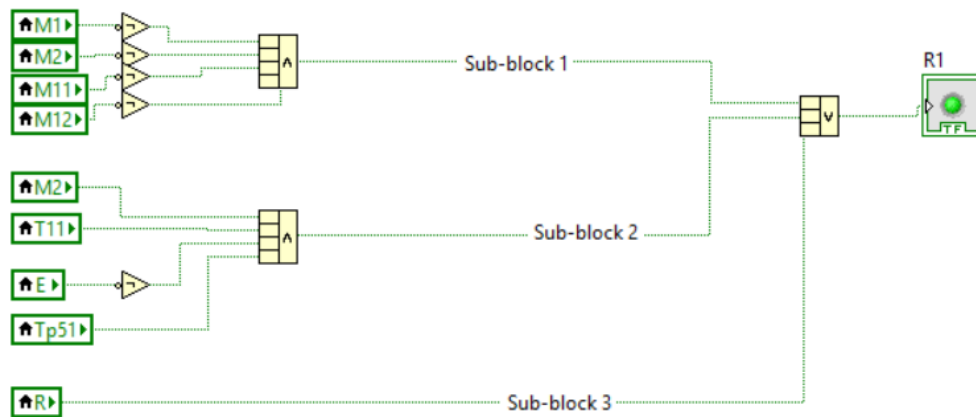


Figure 3.28 Red traffic light for street #1

R1 consists of 3 sub-blocks; each sub-block represents a case that will turn on the red light for street #1 as shown in figure 3.28. The following will discuss each sub-block in details.

**Sub-block 1**

Sub-block 1 is responsible for turning on the red light for street #1, when the green traffic light and yellow traffic light of street #1 is OFF.

**Sub-block 2**

Sub-block 2 is responsible for turning on the red light of street #1, when there are no cars in any of the three streets, and there are pedestrians in any of the three streets.

**Sub-block 3**

Sub-block 3 is responsible for turning on the red light of street #1, when the current state is the yellow light of street #1 and its timer is finished, no emergency in any of the three streets and there is an interrupt from pedestrians in street #1. As shown in figure 3.29.

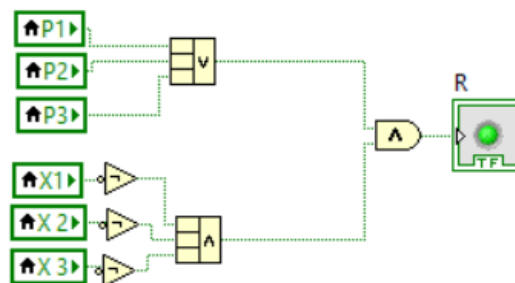


Figure 3.29 Sub-block 3

**Green light for pedestrians in street #1**

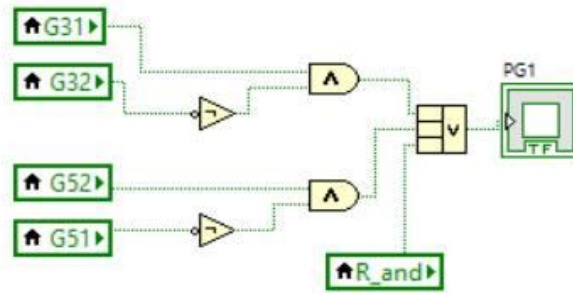


Figure 3.30 Green for pedestrians in street #1

The pedestrians in street #1 are allowed to cross the street if only the left green traffic light of street #2 is ON and its right green traffic light is OFF, or if the right green traffic light of street #3 is ON and its left green traffic light is OFF, or in case of all traffic light are red due to an interrupt from any pedestrians. The figure above shows the conditions of Pedestrians' green traffic light.

### Green light for street #2

There are three signals.

- M3: It is a signal of the normal case (i.e. when both right traffic light and left traffic light of street #1 are ON)
- M31: It is a signal when the left traffic light only is ON (i.e. there are pedestrians in street #3)
- M32: It is a signal when the right traffic light only is ON (i.e. there are pedestrians in street #2)

### Green light for both lanes

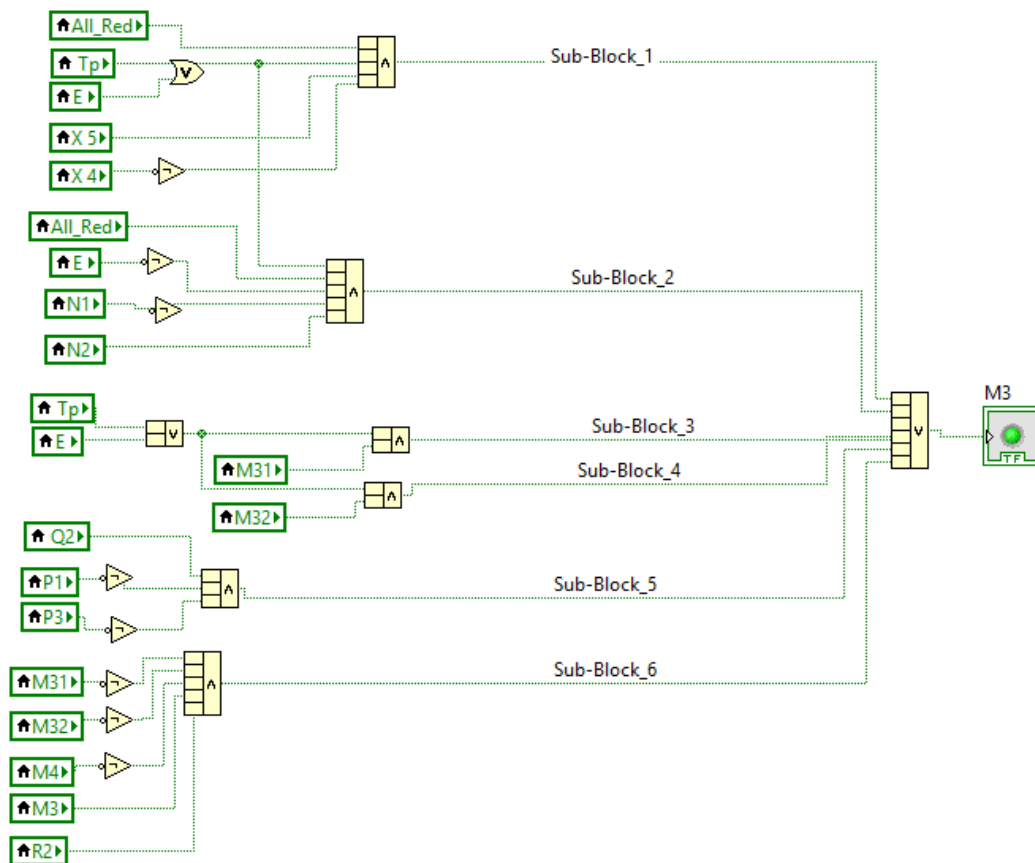


Figure 3.31 Green traffic light for street #2

M3 consists of 6 sub-blocks; each sub-block represents a case that will turn on the green light for street #2 as shown in figure 3.31. These cases do not have to be satisfied together. If only one case is satisfied, Green-2 will be turned ON. The following statements will discuss each sub-block in details.

**Sub-block 1**

Sub-block 1 is responsible for turning on the green light for street #2 with the assumption that there are no pedestrians in street #1 and street #3. There is a signal Q2 which has 6 conditions, if any of these conditions is satisfied while the condition that there are no pedestrians in street #3 and street #1 is also satisfied, the green light for street #2 will be turned on.

**Conditions of Q2:**

Illustrate the normal case without pedestrians, the following figure shows these conditions.

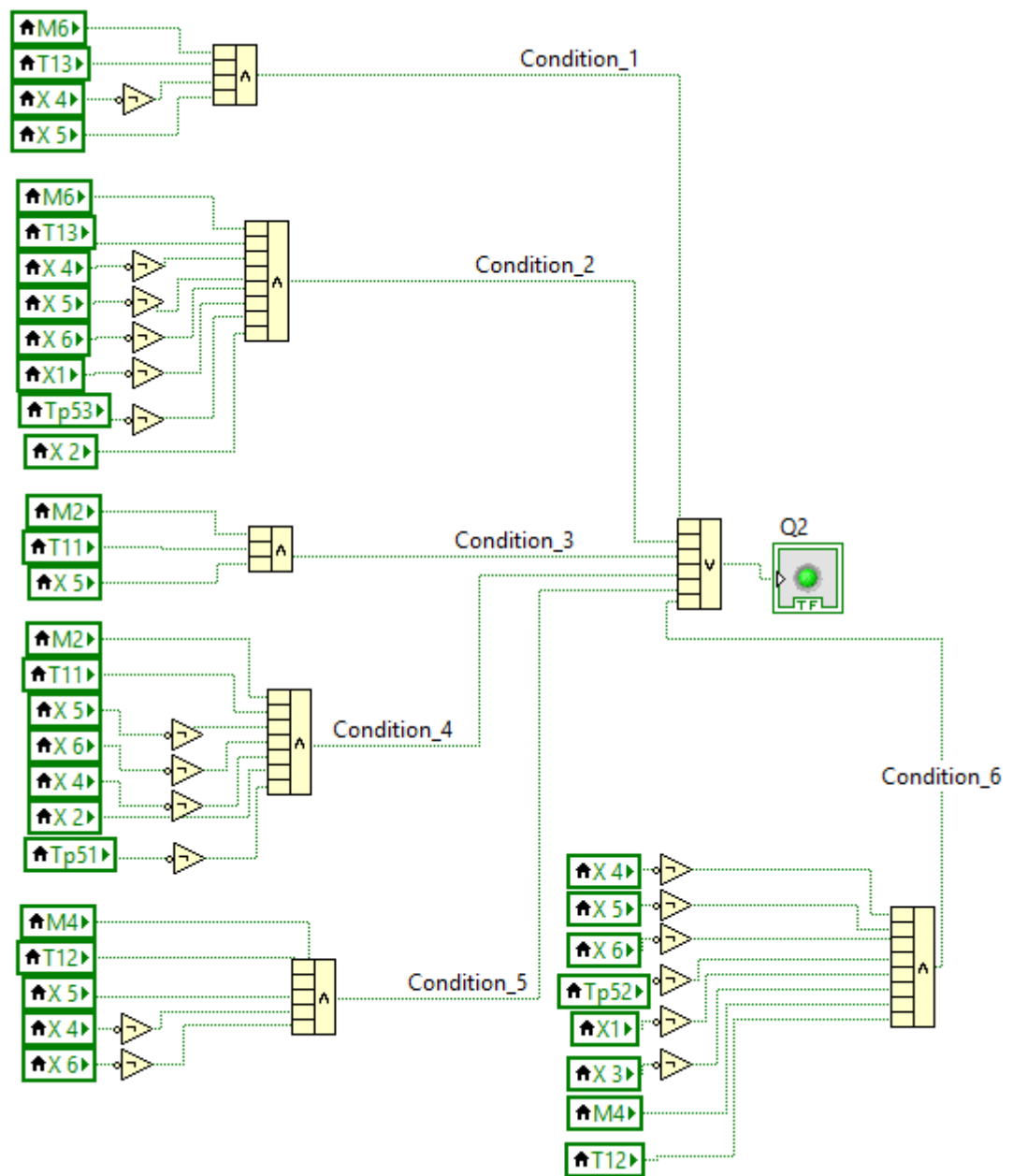


Figure 3.32 Conditions of Q1

**Condition #1:**

It states that if the current state is Yellow in street #3, its timer has been finished, and there is no emergency in the in-order street (Street #1), while there is an emergency in street #2. Subsequently, the green light of street #2 will be turned on.

**Condition #2:**

It states that if the current state is the Yellow of street #3, its timer has been finished, there is no emergency in any of the three streets, there are no cars in the in-order street (street #1), while there are cars in street #2, and that there is no interrupt from pedestrians in street #3 (assuming that there are no pedestrians). Subsequently, the green light for street #2 will be turned on.

**Condition #3:**

It states that if the current state is the yellow light in street #1, and its timer has been finished, and there is an emergency in street #2, hence, the green light for street #2 will be turned on.

**Condition # 4:**

It states that if the current state is the yellow light in street #1, yellow light's timer has been finished, there is no emergency in any of the three streets, and we have cars in street #2, given that there are no interrupts from pedestrians in street #1 (assuming that there are no pedestrians). Subsequently, the green light for street #2 will be turned on.

**Condition # 5:**

It states that if the current state is yellow light in street #2, and its timer has been finished, while an emergency in this street (street #2) has come during the Yellow light time, and there are no other emergencies in the other two streets, then the green light will be turned on again for street #2.

**Condition # 6:**

It states that if the current state is yellow light in street #2, its timer has been finished, there are no emergencies in any of the three streets, there are no cars in street #1 and street #3, and there is no interrupt from pedestrians in street #2, Therefore, the green for street #1 will be turned on.

I.e. This case happens when a car that has been read (by RFID reader) in street #1 or street #3, illegally crosses the road (Violates the traffic light), or there is an instantaneous false reading by the RFID reader (rarely happens).

### Sub-block 2

Sub-block 2 is responsible for turning on the green light for street #2, when the current state is the state that all traffic lights are Red, and the timer for the pedestrians to cross the street is finished or there is an emergency in street #2 and no emergency in street #1.

### Sub-block 3

Sub-block 3 is responsible for turning on the green light for street #2, when the current state is the state that all traffic lights are Red, the timer for pedestrians to cross the street is finished, and there are no emergencies in any of the three streets. In addition, the previous state was the green of street #2. The Previous state is indicated by the signal N2.

### Condition for signal N2:

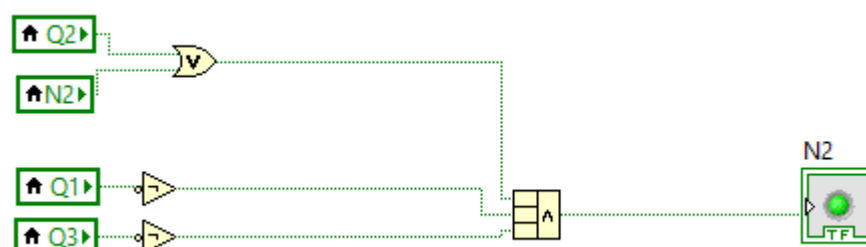


Figure 3.33 Conditions of N2

The signal N2 is the same as the signal of Q2 but with a self-latch and an exit condition. So, it is always “true” if the state did not change either to Green



light of street #1 (Q1= “true”) or Green light of street #3 (Q3 = “true”) as shown in figure 3.33.

#### **Sub-block 4**

Sub-block 4 is responsible for turning on the green light of street #2, when the current state is the Left Green light state, and there is an emergency or the time for pedestrians to cross the street is finished. Thus, we return to green light of street #2.

#### **Sub-block 5**

Sub-block 5 is responsible for turning on the green light of street #2, when the current state is the Right Green light state, and there is an emergency or the time for the pedestrians to cross the street is finished. Thus, we return to green light of street #2.

#### **Sub-block 6**

Sub-block 6 is responsible for the self-latch and exit of M3. M3 will keep its value “True”, whenever Left Green light, Right Green light, Yellow light of street #2 and Red light of street #2 are not changed to become “True”.

#### **Green light for the left lane**

M31 consists of 3 sub-blocks; each sub-block represents a case that will turn on the Left Green traffic light only in street #1 as shown in figure 3.34.

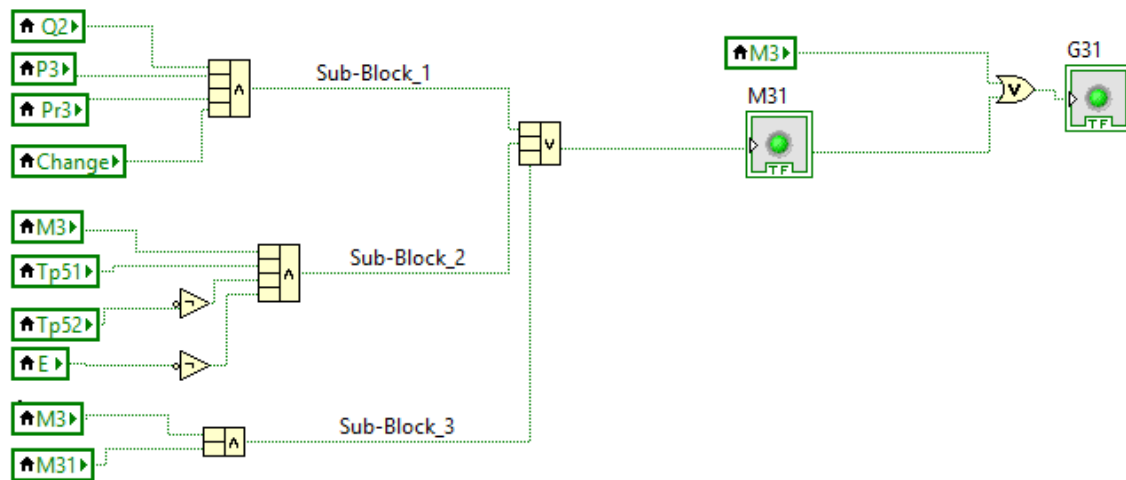


Figure 3.34 Conditions of the left lane green traffic light for street#2

### Sub-block 1

Sub-block 1 is responsible for turning M31 ON, when there are pedestrians in street #3, the priority is for pedestrians in street #3 and any of Q2 conditions is satisfied.

### Sub-block 2

Sub-block 2 is responsible for turning M31 ON, when the current state is the state that both left and right green traffic light of street-2 are ON, there is an interrupt from pedestrians in street #1, there is no interrupt from pedestrians in street #2 and there is no emergency.

### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M31. M31 will keep its value unless the state returns to M3.

G31 is the output signal of the green traffic light, turned ON if one of the following two cases is satisfied as shown in figure 3.34.

- The normal green traffic light is ON (M3)
- The left green traffic light is ON (M31)

### Green light for the right lane

M32 consists of 3 sub-blocks; each sub-block represents a case that will turn on the right Green traffic light only in street #3 as shown in figure 3.35.

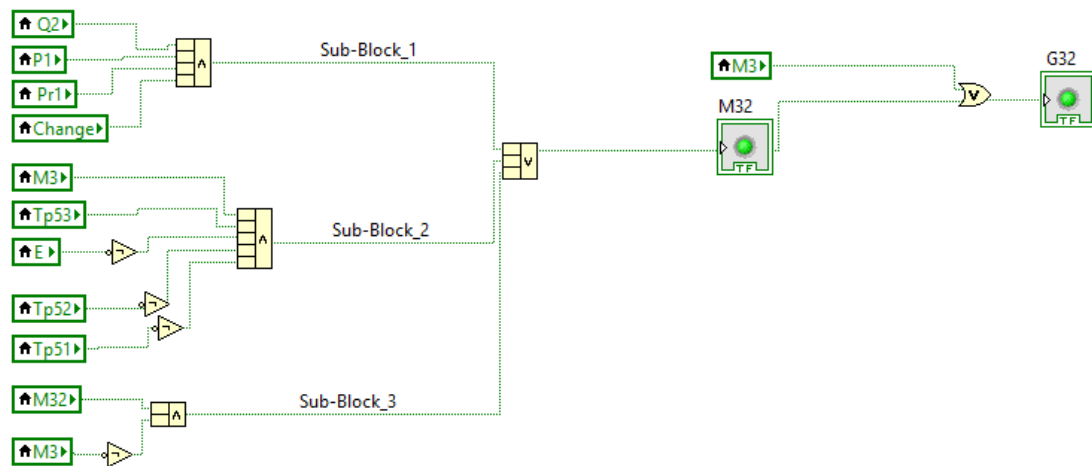


Figure 3.35 Conditions of the right lane green traffic light for street #2

### Sub-block 1

Sub-block 1 is responsible for turning M32 ON, when there are pedestrians in street #1, the priority is for pedestrians in street #1 and any of Q2 conditions is satisfied.

### Sub-block 2

Sub-block 2 is responsible for turning M32 ON, when the current state is the state that both left and right green traffic light of street-2 are ON, there is an interrupt from pedestrians in street #3, there is no interrupt from both pedestrians in street #1 and street #2, and there is no emergency.

### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M32. M32 will keep its value unless the state returns to M3.

G32 is the output signal of the green traffic light, turned ON if one of the following two cases is satisfied as shown in figure 3.35.

- The normal green traffic light is ON (M3)
- The left green traffic light is ON (M32)

### Yellow light for street #2

M4 represent the Yellow traffic light for street #2. It consists of 3 sub-blocks as shown in figure 3.36. If any condition of these sub-blocks is satisfied, the current state will change from Green traffic light to Yellow traffic light of street #2. In the following part each sub-block is discussed in details:

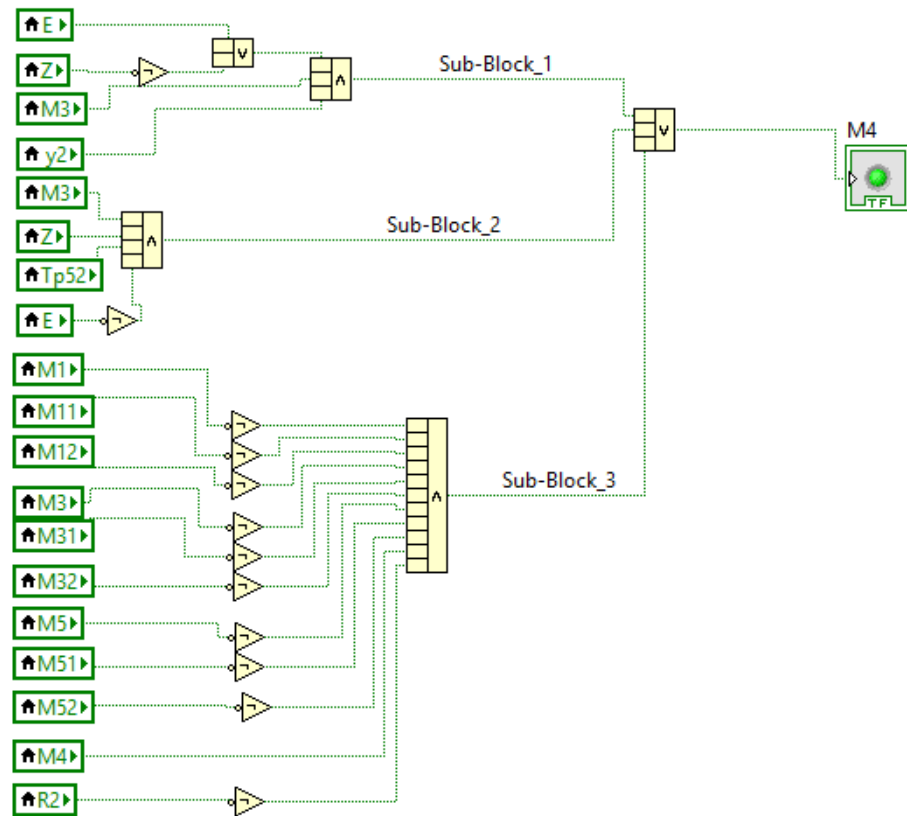


Figure 3.36 Conditions of yellow traffic light for street #2

**Sub-block 1**

Sub-block 1 is responsible for turning M4 ON, when the current state is the green of street #2, without any interrupt from pedestrians or there is an emergency, and if any of the conditions of signal y2 shown in figure (Y2) is satisfied.

**Conditions of signal y2:**

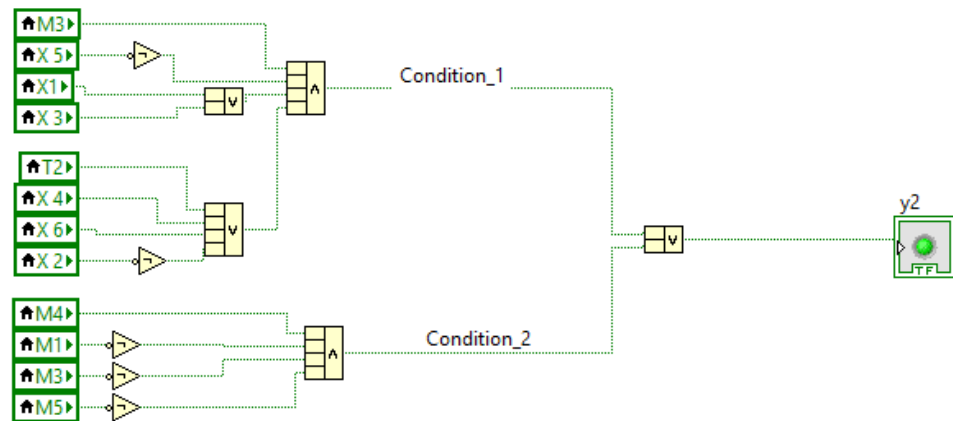


Figure 3.37 Conditions of signal Y2

### Condition #1:

It states that if the current state is the green of street #2 with no emergencies in street #2, and there are cars either in street #1 or in street #3, In addition, the timer of the green traffic light is finished or there is an emergency in either street #1 or street #3 or there are no cars in street #2, then y2 will be turned ON.

### Condition #2:

It represents the self-latch and exit of y2. Y2 will hold its value unless, the green traffic light of any of the three streets turned ON.

### Sub-block 2

Sub-block 2 is responsible for turning M4 ON, when the current state is the green of street #2, there is an interrupt from pedestrians in street #2, and there is no emergency in any of the three streets.

### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M4. M4 will keep its value unless, the green traffic light either Left green, right green or both of them of any of the three streets turned on or the Red traffic light of street #2 is ON.

### Red light for street #2

R2 consists of 3 sub-blocks; each sub-block represents a case that will turn on the red light for street #2 as shown in figure 3.38. The following will discuss each sub-block in details.

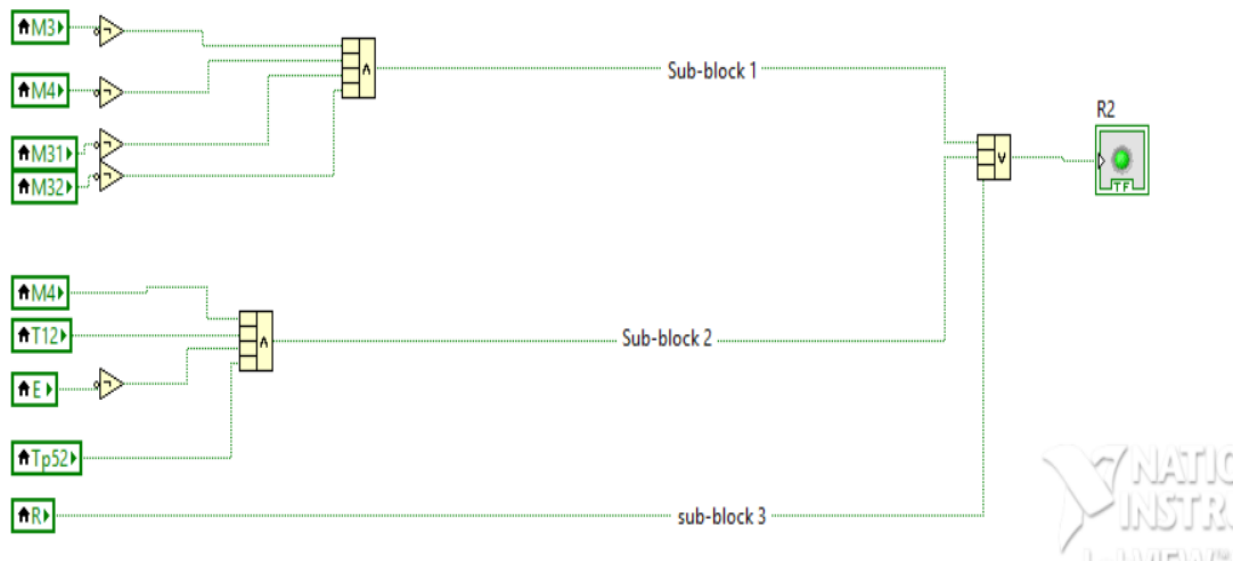


Figure 3.38 Conditions of red traffic light for street #2

### Sub-block 1

Sub-block 1 is responsible for turning on the red light for street #2, when the green traffic light and yellow traffic light of street #2 is OFF.

### Sub-block 2

Sub-block 2 is responsible for turning on the red light of street #2, when there are no cars in any of the three streets, and there are pedestrians in any of the three streets.

### Sub-block 3

Sub-block 3 is responsible for turning on the red light of street #2, when the current state is the yellow light of street #2 and its timer is finished, no emergency in any of the three streets and there is an interrupt from pedestrians in street #2 as shown in figure 3.38.

### Green light for pedestrians in street #2

The pedestrians in street #2 are allowed to cross the street if only the left green traffic light of street #3 is ON and its right green traffic light is OFF, or if the right green traffic light of street #1 is ON and its left green traffic light is OFF, or in case of all traffic light are red due to an interrupt from any pedestrians. The following figure shows the conditions of pedestrians' green traffic light.

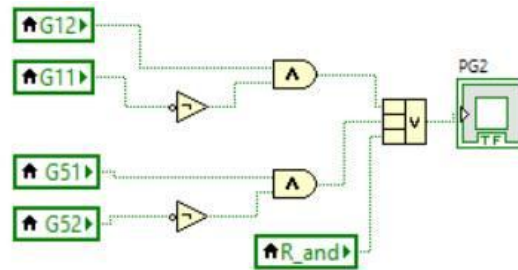


Figure 3.39 Green for pedestrians in street #2

### Green light for street #3

There are three signals.

- M5: It is a signal of the normal case (i.e. when both right traffic light and left traffic light of street #3 are ON)
- M51: It is a signal when the left traffic light only is ON (i.e. there are pedestrians in street #2)
- M32: It is a signal when the right traffic light only is ON (i.e. there are pedestrians in street #1)

### Green light for both lanes

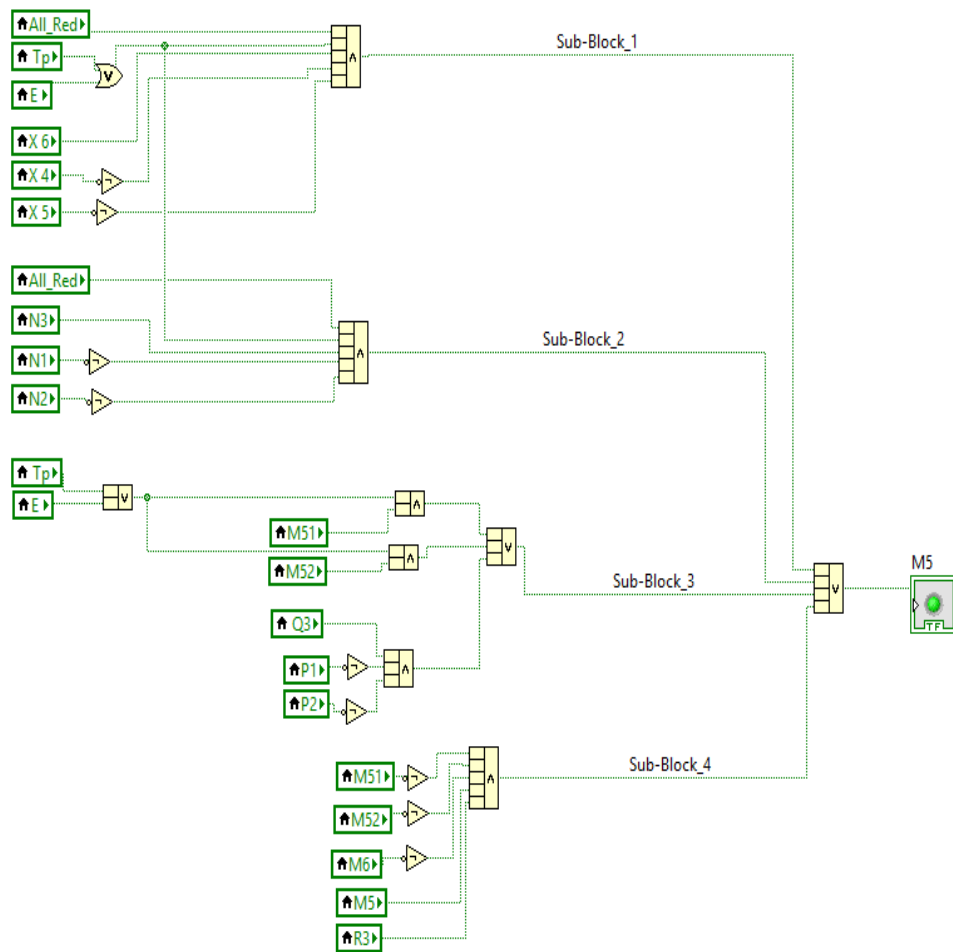


Figure 3.40 Green traffic light for street #3

M5 consists of 6 sub-blocks; each sub-block represents a case that will turn on the green light for street #3 as shown in figure 3.40. These cases do not have to be satisfied together. If only one case is satisfied, Green-3 will be turned ON. The following statements will discuss each sub-block in details.

**Sub-block 1**

Sub-block 1 is responsible for turning on the green light for street #3 with the assumption that there are no pedestrians in street #1 and street #2. There is a signal Q3 which has 6 conditions, if any of these conditions is satisfied while the condition that there are no pedestrians in street #1 and street #2 is also satisfied, the green light for street #3 will be turned on.

**Conditions of Q3:**



Illustrate the normal case without pedestrians, the following figure shows these conditions.

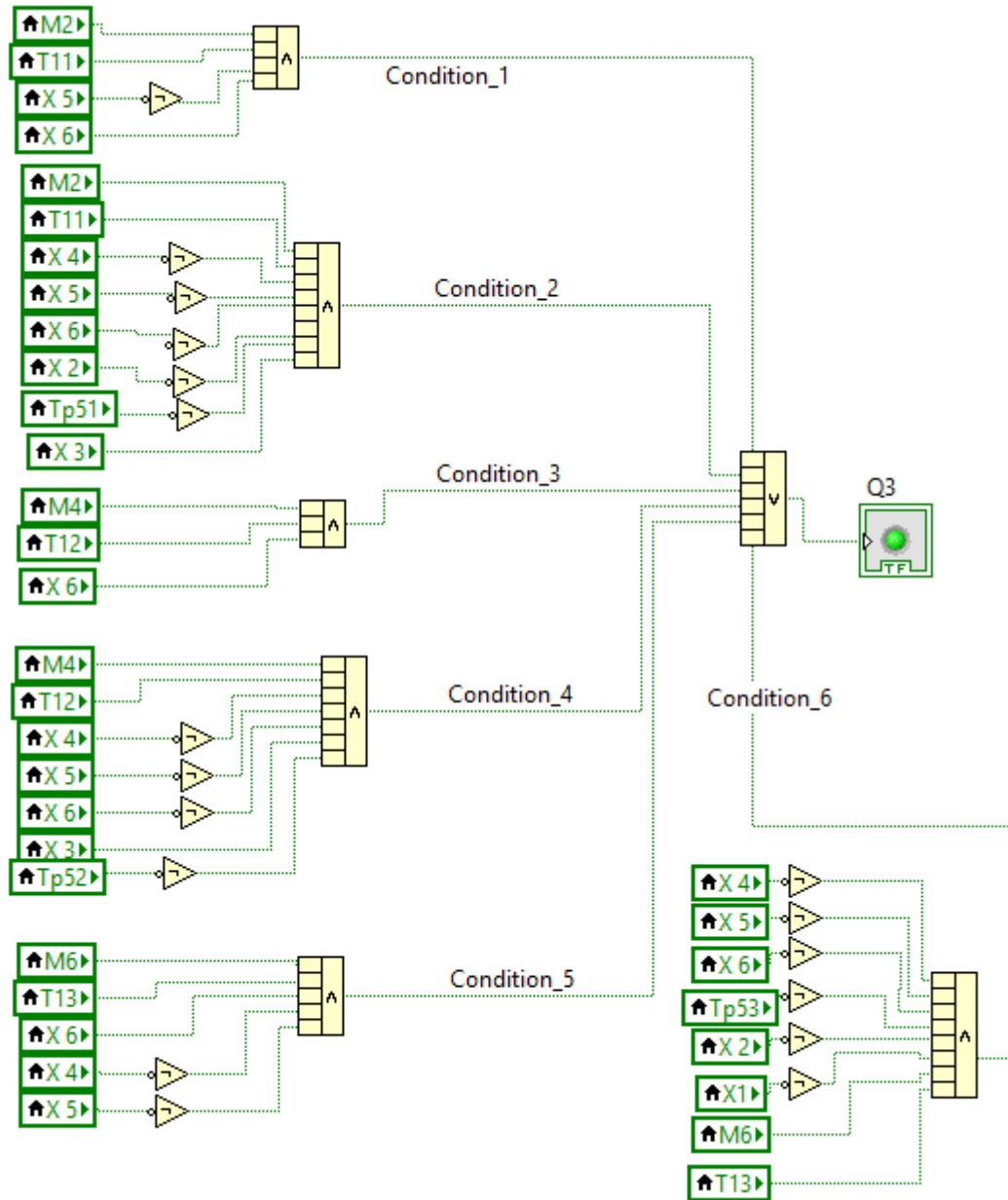


Figure 3.41 Conditions of Q3

**Condition #1:**

It states that if the current state is Yellow in street #2, its timer has been finished, and there is no emergency in the in-order street (Street #1), while there is an emergency in street #3. Subsequently, the green light of street #3 will be turned on.

**Condition #2:**

It states that if the current state is the Yellow of street #1, its timer has been finished, there is no emergency in any of the three streets, there are no cars in the in-order street (street #2), while there are cars in street #3, and that there is no interrupt from pedestrians in street #1 (assuming that there are no pedestrians). Subsequently, the green light for street #3 will be turned on.

**Condition #3:**

It states that if the current state is the yellow light in street #2, and its timer has been finished, and there is an emergency in street #3, hence, the green light for street #2 will be turned on.

**Condition #4:**

It states that if the current state is the yellow light in street #2, yellow light's timer has been finished, there is no emergency in any of the three streets, and we have cars in street #3, given that there are no interrupts from pedestrians in street #2 (assuming that there are no pedestrians). Subsequently, the green light for street #3 will be turned on.

**Condition #5:**

It states that if the current state is yellow light in street #3, and its timer has been finished, while an emergency in this street (street #3) has come during the Yellow light time, and there are no other emergencies in the other two streets, then the green light will be turned on again for street #3.

**Condition #6:**

It states that if the current state is yellow light in street #3, its timer has been finished, there are no emergencies in any of the three streets, there are no cars in street #1 and street #2, and there is no interrupt from pedestrians in street #3, Therefore, the green for street #3 will be turned on.

I.e. This case happens when a car that has been read (by RFID reader) in street #1 or street #2, illegally crosses the road (Violates the traffic light),

or there is an instantaneous false reading by the RFID reader (rarely happens).

### Sub-block 2

Sub-block 2 is responsible for turning on the green light for street #2, when the current state is the state that all traffic lights are Red, and the timer for the pedestrians to cross the street is finished or there is an emergency in street #3 and there are no emergencies in street #1 and street #2.

### Sub-block 3

Sub-block 3 is responsible for turning on the green light for street #3, when the current state is the state that all traffic lights are Red, the timer for pedestrians to cross the street is finished, and there are no emergencies in any of the three streets. In addition, the previous state was the green of street #3. The Previous state is indicated by the signal N3.

#### Condition for signal N3:

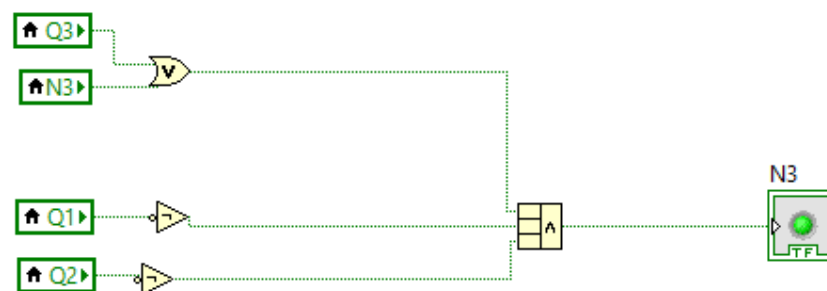


Figure 3.42 Conditions of N3

The signal N3 is the same as the signal of Q3 but with a self-latch and an exit condition. So, it is always “true” if the state did not change either to Green light of street #1 (Q1= “true”) or Green light of street #2 (Q2 = “true”) as shown in figure 3.42.

### Sub-block 4

Sub-block 4 is responsible for turning on the green light of street #3, when the current state is the Left Green light state, and there is an emergency or the time for pedestrians to cross the street is finished. Thus, we return to green light of street #3.

#### **Sub-block 5**

Sub-block 5 is responsible for turning on the green light of street #3, when the current state is the Right Green light state, and there is an emergency or the time for the pedestrians to cross the street is finished. Thus, we return to green light of street #3.

#### **Sub-block 6**

Sub-block 6 is responsible for the self-latch and exit of M5. M5 will keep its value (“True”), whenever Left Green light, Right Green light, Yellow light of street #3 and Red light of street #3 are not changed to become “True”.

#### **Green light for the left lane**

M51 consists of 3 sub-blocks; each sub-block represents a case that will turn on the Left Green traffic light only in street #3 as shown in figure 3.43.

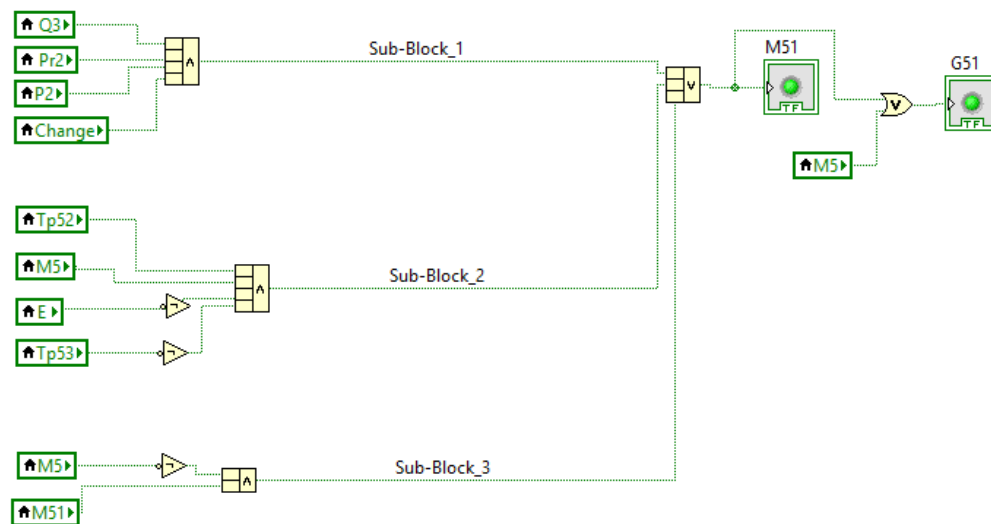


Figure 3.43 Conditions of the left lane green traffic light for street #3

### Sub-block 1

Sub-block 1 is responsible for turning M51 ON, when there are pedestrians in street #2, the priority is for pedestrians in street #2 and any of Q3 conditions is satisfied.

### Sub-block 2

Sub-block 2 is responsible for turning M51 ON, when the current state is the state that both left and right green traffic light of street-3 are ON, there is an interrupt from pedestrians in street #2, there is no interrupt from pedestrians in street #3 and there is no emergency.

### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M51. M51 will keep its value unless the state returns to M5.

G51 is the output signal of the green traffic light, turned ON if one of the following two cases is satisfied as shown in figure 3.43:

- The normal green traffic light is ON (M5)
- The left green traffic light is ON (M51)

### Green light for the right lane

M52 consists of 3 sub-blocks; each sub-block represents a case that will turn on the right Green traffic light only in street #3 as shown in figure 3.44.

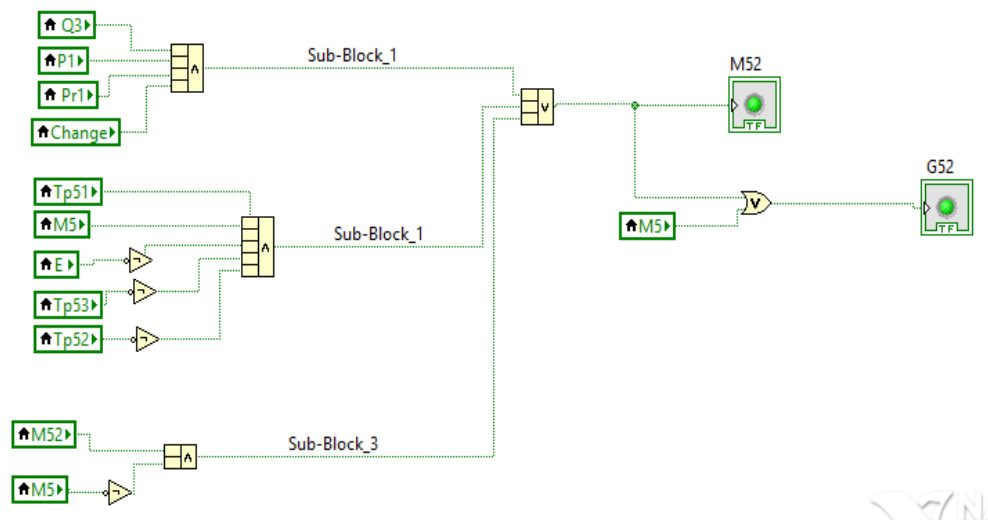


Figure 3.44 Conditions of the right lane green traffic light for street #3

### Sub-block 1

Sub-block 1 is responsible for turning M52 ON, when there are pedestrians in street #1, the priority is for pedestrians in street #1 and any of Q3 conditions is satisfied.

### Sub-block 2

Sub-block 2 is responsible for turning M52 ON, when the current state is the state that both left and right green traffic light of street-3 are ON, there is an interrupt from pedestrians in street #1, there is no interrupt from both pedestrians in street #2 and street #3, and there is no emergency.

### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M52. M52 will keep its value unless the state returns to M5.

G52 is the output signal of the green traffic light, turned ON if one of the following two cases is satisfied as shown in figure 3.44:

- The normal green traffic light is ON (M5)
- The left green traffic light is ON (M52)

### Yellow light for street #3

M6 represent the Yellow traffic light for street #3. It consists of 3 sub-blocks as shown in figure 3.30. If any condition of these sub-blocks is satisfied, the current state will change from Green traffic light to Yellow traffic light of street #3. In the following part each sub-block is discussed in details:

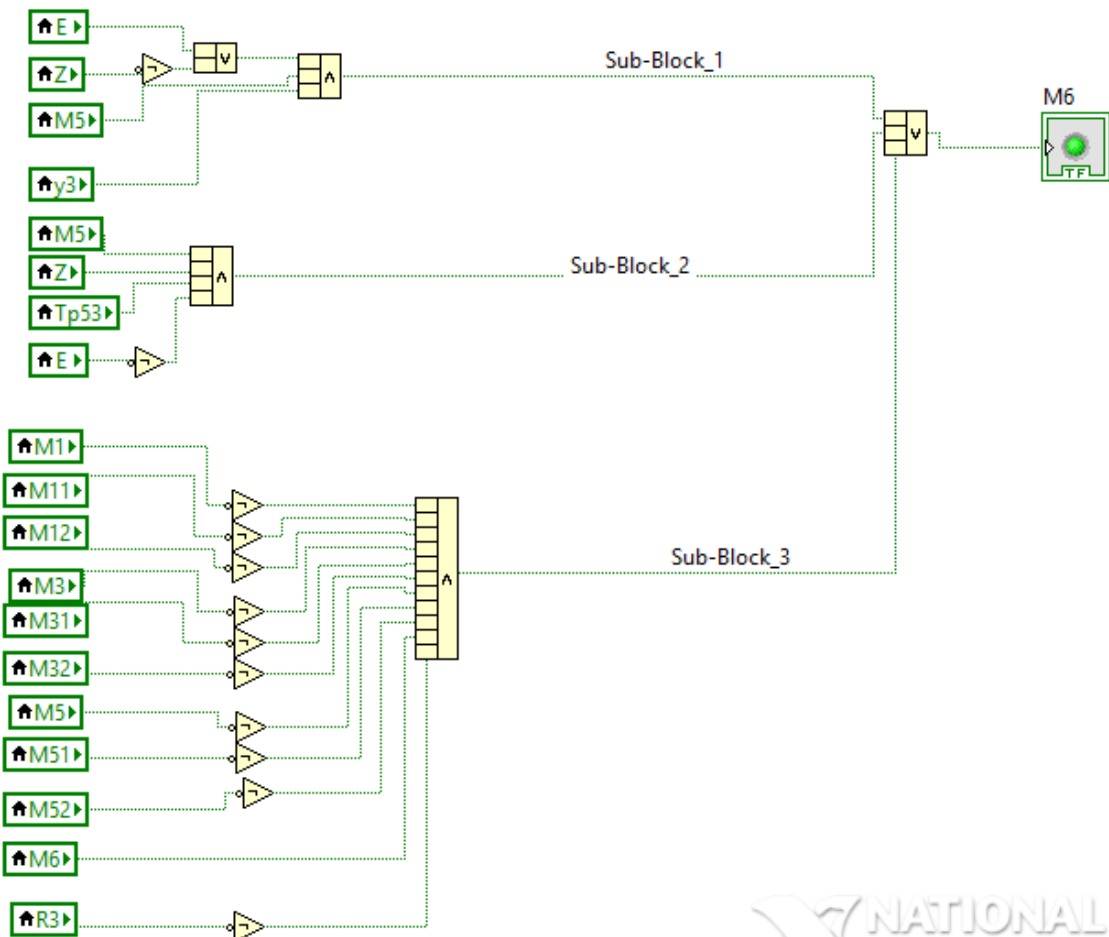


Figure 3.45 Conditions of yellow traffic light for street #3

### Sub-block 1

Sub-block 1 is responsible for turning M6 ON, when the current state is the green of street #3, without any interrupt from pedestrians or there is an emergency, and if any of the conditions of signal y3 shown in figure 3.45 is satisfied.

**Conditions of signal y3:**

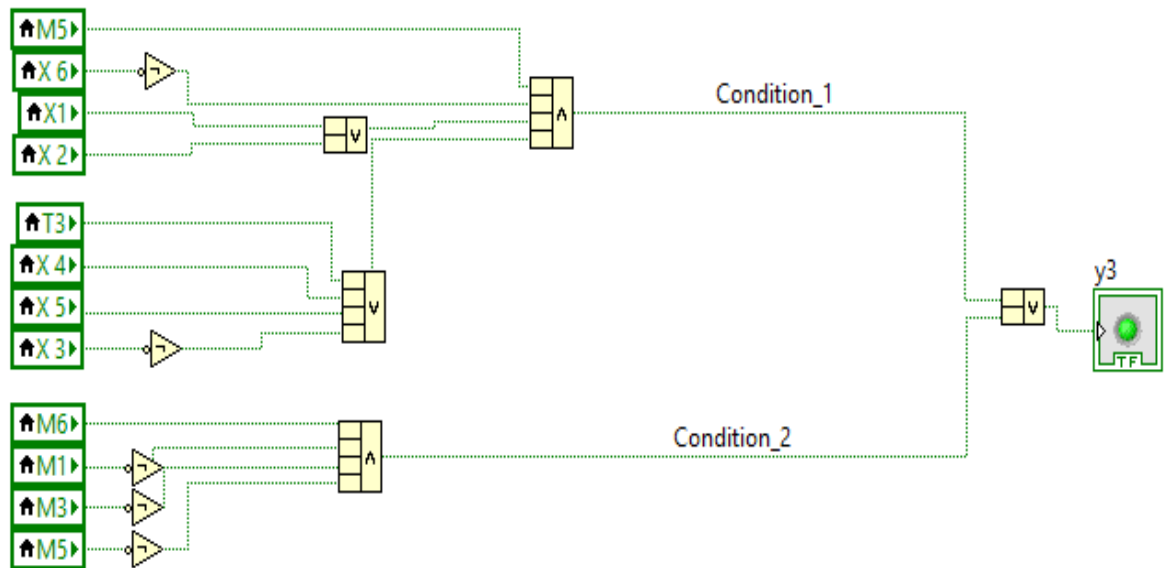


Figure 3.46 conditions of signal Y3

**Condition #1:**

It states that if the current state is the green of street #3 with no emergencies in street #3, and there are cars either in street #1 or in street #2, in addition, the timer of the green traffic light is finished or there is an emergency in either street #1 or street #2 or there are no cars in street #1, then y3 will be turned ON.

**Condition #2:**

It represents the self-latch and exit of y3. Y3 will hold its value unless, the green traffic light of any of the three streets turned ON.

**Sub-block 2**

Sub-block 2 is responsible for turning M6 ON, when the current state is the green of street #3, there is an interrupt from pedestrians in street #3, and there is no emergency in any of the three streets.



### Sub-block 3

Sub-block 3 is responsible for the self-latch and exit of M6. M6 will keep its value unless, the green traffic light either Left green, right green or both of them of any of the three streets turned on or the Red traffic light of street #3 is ON.

### Red light for street #3

R3 consists of 3 sub-blocks; each sub-block represents a case that will turn on the red light for street #3 as shown in figure 3.47. The following will discuss each sub-block in details.

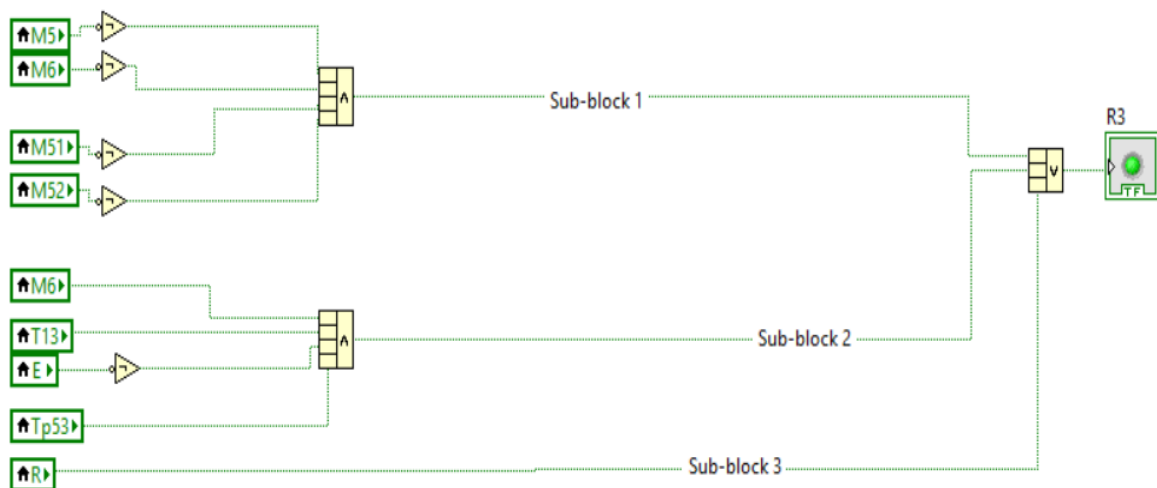


Figure 3.47 Conditions of red traffic light for street #3

### Sub-block 1

Sub-block 1 is responsible for turning on the red light for street #3, when the green traffic light and yellow traffic light of street #3 is OFF.

### Sub-block 2

Sub-block 2 is responsible for turning on the red light of street #3, when there are no cars in any of the three streets, and there are pedestrians in any of the three streets.

### Sub-block 3

Sub-block 3 is responsible for turning on the red light of street #3, when the current state is the yellow light of street #3 and its timer is finished, no emergency in any of the three streets and there is an interrupt from pedestrians in street #3 as shown in figure 3.47.

### Green light for pedestrians in street #3

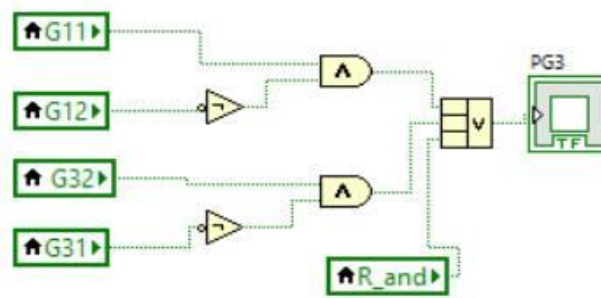


Figure 3.48 conditions of green of pedestrians in street #3

The pedestrians in street #3 are allowed to cross the street if only the left green traffic light of street #1 is ON and its right green traffic light is OFF, or if the right green traffic light of street #2 is ON and its left green traffic light is OFF, or in case of all traffic light are red due to an interrupt from any pedestrians. The following figure shows the conditions of Pedestrians' green traffic light.

### Cars' traffic light Timer

It decides the time for green and yellow traffic light. For the green state the time is set according to the number of cars in the street. While the yellow state time is constant for all cases (=3). The timer block consists of two main blocks; Dynamic timing block and Timer block. The first block is considered to be a multiplexer that chooses a suitable time for the green traffic light according to the number of cars. The second block is the timer itself, which counts the time decided by the first block.

Cars' traffic light timer has only one input that shows the current state as shown in figure 3.49 for green timer and figure 3.50 for yellow light. All states are added as inputs to the block "Replace array subset" that changes these inputs in to binary numbers. These binary numbers are converted in to an integer number. Each integer number represents a case as shown in table 3.9.

Table 3.8 Binary to integer case mapping

Current state	Case number
Traffic light for street #1: Green	1
Traffic light for street #2: Green	2
Traffic light for street #3: Green	4
Traffic light for street #1: Yellow	8
Traffic light for street #2: Yellow	16
Traffic light for street #3: Yellow	32

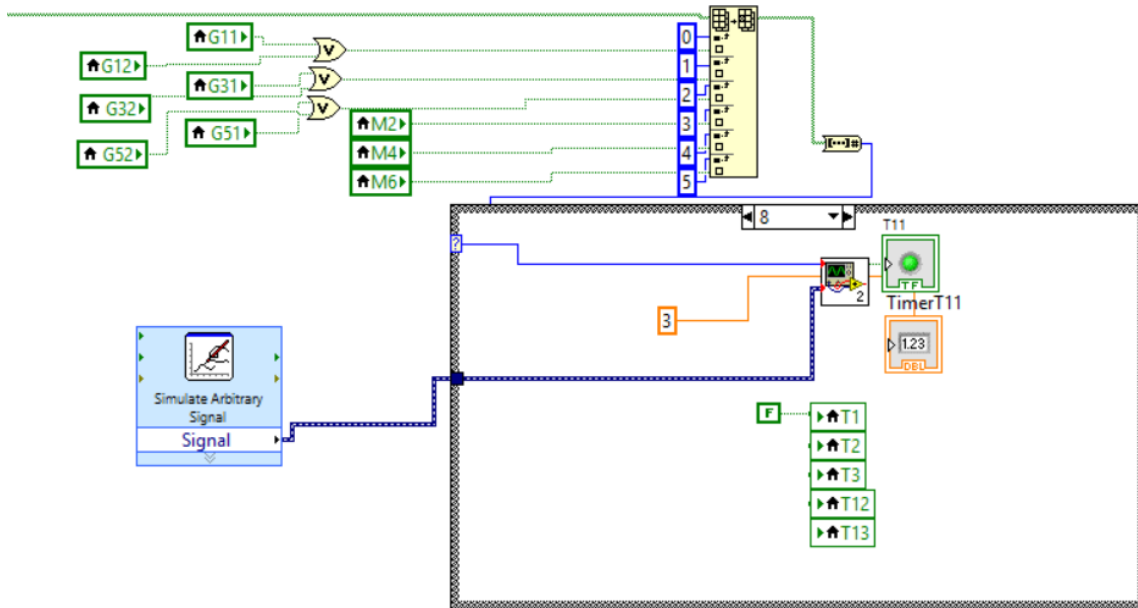


Figure 3.49 Yellow light timer

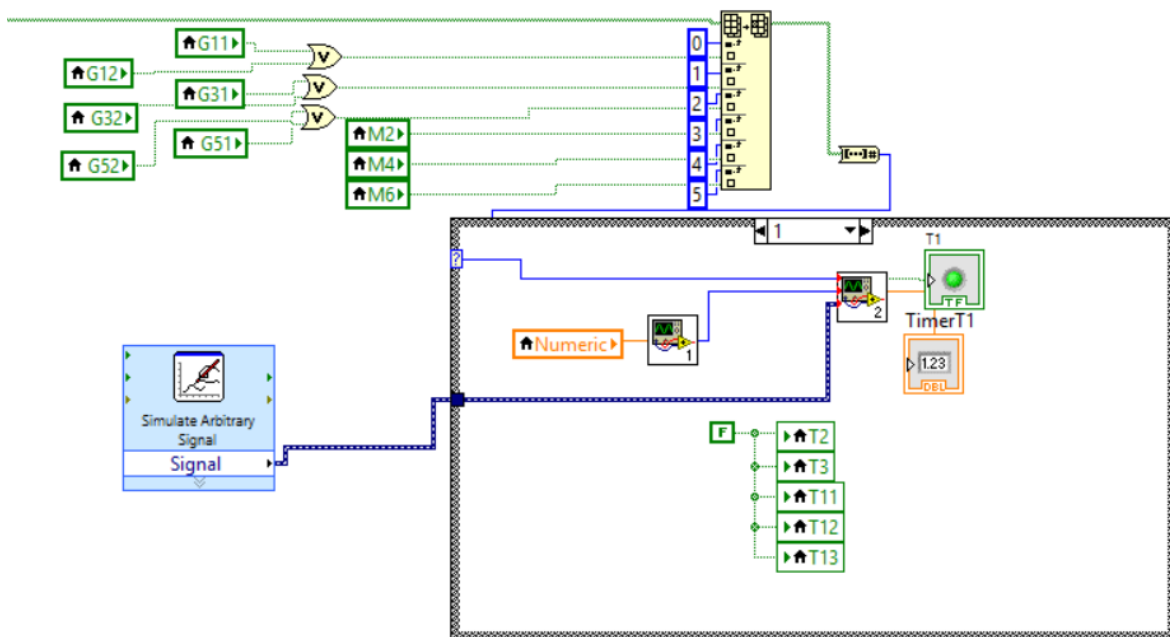


Figure 3.50 Green light timer

### Dynamic timing block

This block It has 1 input and 1 output pins as shown in figure 3.51.

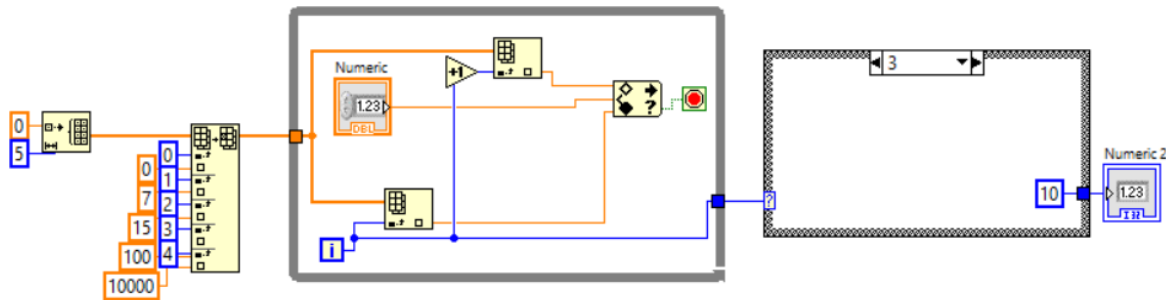


Figure 3.51 Dynamic timing block

**Input Pin:** It is the number of the cars in the current green traffic light.

**Output Pin:** It is the time that will be taken by the green traffic light.

Dynamic timing block contains an array that is initialized by values [0 7 15 100 10000], this array represents the number of cars. For example, in the first iteration; if the input number of cars is within the range from 0 to 7, then break the loop, and the output time is set to be 2 seconds. Else, it will continue comparing the number of the input cars with the next ranges, until the number of cars falls within one of these ranges shown in table 3.10.

Table 3.9 Dynamic time table

Range of cars	Time
0-4 cars	25 seconds
5-10 cars	30 seconds
11-100 cars	40 seconds

### Timer block

It has 3 input and 2 output pins as shown in figure 3.52.

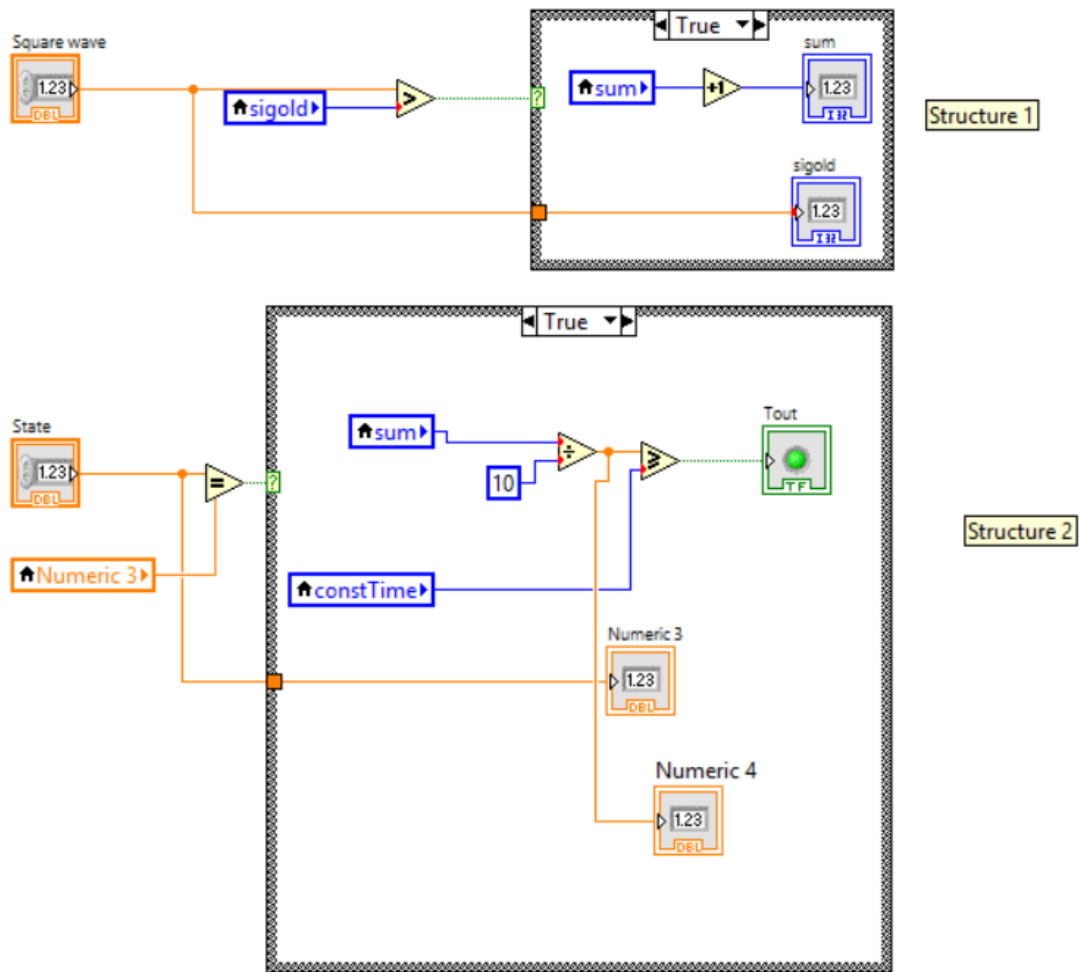


Figure 3.52 Timer block

#### Input pins:

1. **State:** It indicates the current state of the traffic light
2. **Time:** It is the time for each state.
3. **Signal:** It is a square wave

#### Output pins:

1. **T<sub>i</sub>** : Indicator to show if the time specified to the current state has finished or not.
2. **Timer T<sub>i</sub>** : It is a display for the time count.

There are two main structures in Timer block, the first structure shown in figure 3.37 is for comparing the input signal (square wave) value with its previous value to know if there is a positive edge trigger. Table 3.11 contains some important parameters for the square signal.

Table 3.10 Timer signal parameters

Parameter	value
Minimum amplitude	0
Maximum amplitude	1
Period	0.1 second

If the square wave value is greater than its pervious value (sigold), then increment local variable called “Sum” which represents the time. So, if the value of the square wave is zero, and comparing it the pervious value (assume = 0). Then this will gives “False”. In the false case, the sigold takes the current value of the square wave. After half the period of the signal, the value changes from “0” to “1”. Now, the current value of the signal is greater than the previous value, and then this will give “True”. In the “true” case as shown in figure 3.53, the “Sum” is incremented and the sigold takes the current value of the square wave. In the next period the current value will be “0” and previous value is “1”, which give false when comparing. This means that sum is incremented only at the Positive Edge of the square signal.

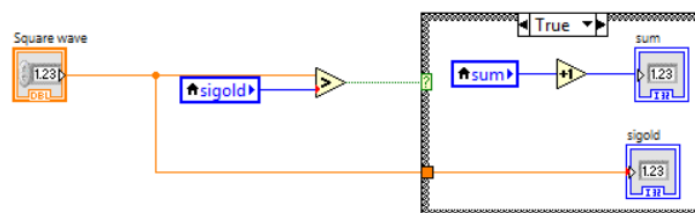


Figure 3.53 Structure-1

The second structure shown in figure 3.54 is for calculating the value of the timer. If the current state does not change, the value of “sum” (representing the time) being calculated in the first structure is compared with the target time. If it is greater than or equal our target time, therefore the timer indicator gives “True”. The value of the timer is displayed to the output Ti.

Note: the value of “Sum” is divided by 10 as it incremented by 1 every 0.1 second.

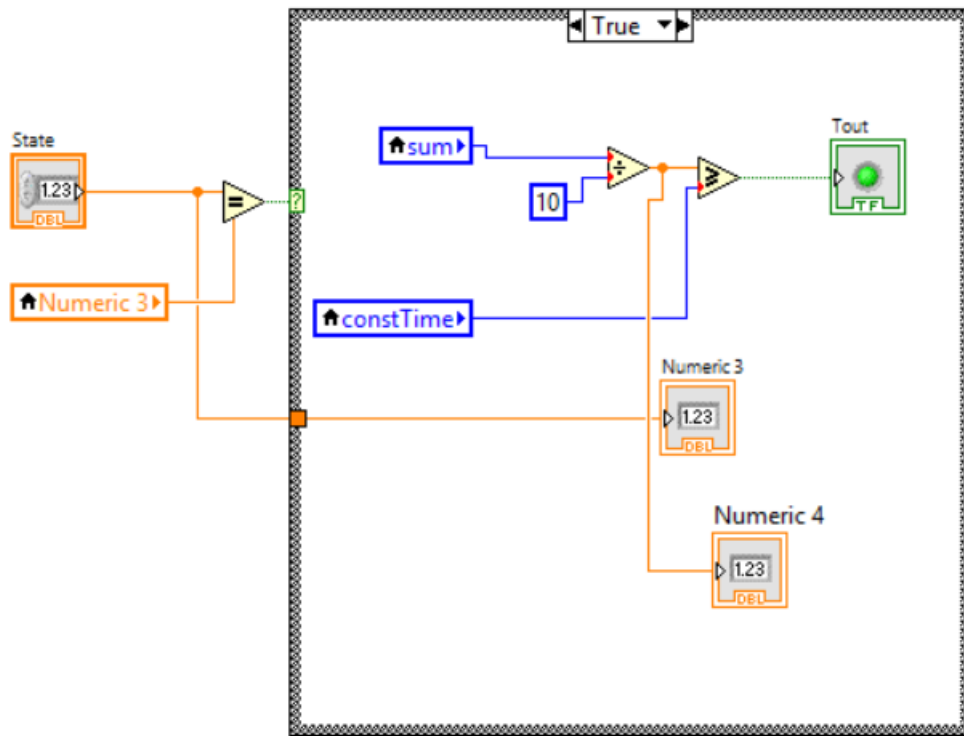


Figure 3.54 Structure-2



### Pedestrians' traffic light Timer

It is the time taken by the pedestrian to cross any of the three streets. Pedestrians' timer starts when they are allowed to cross the street. As shown in figure 3.55 Pedestrians' timer starts once there is only one side (Left or Right) green traffic light is ON, or in case of all traffic lights are red due to an interrupt from Pedestrians' interrupt Timer. The time allowed for this timer is set to be smaller than the time allowed for green traffic light, so after the pedestrians cross the road in their specific time, the green traffic light returns to its normal case (both right and left green traffic light).

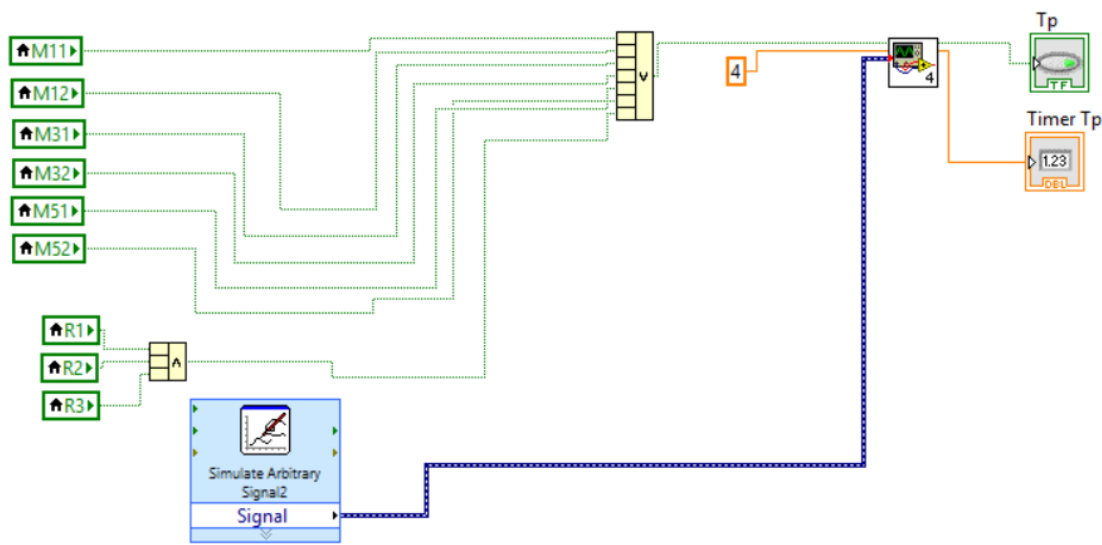


Figure 3.55 Pedestrians timer

### Pedestrians' interrupt Timer

It is the maximum time the pedestrians will wait before interrupting the current green traffic light of cars. Each street has an interrupt timer for its pedestrians. The interrupt timer starts once the pedestrians arrives and press on the push button. It is constant for all pedestrian in all streets, and it is said to be 5 minutes. As shown in figure 3.56 the same timer block discussed before is used.

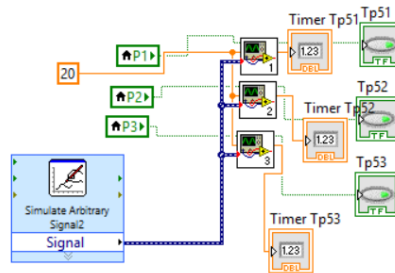


Figure 3.56 Interrupt timer

### Priority Block

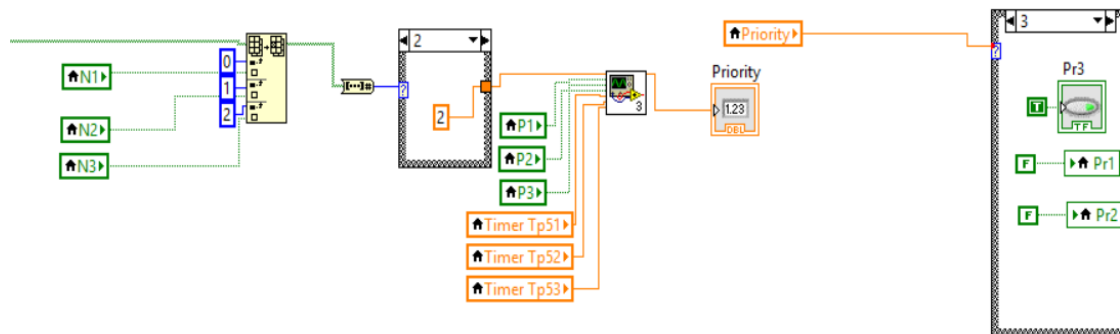


Figure 3.57 Priority

Pedestrians are selected to cross the street according to their priority. Pedestrians with larger waiting time are given higher priority. Priority is checked and changed every time the state of the traffic light is changed depending on N1, N2 and N3. The following figure illustrates the priority for pedestrians.

The priority block has 7 input and 3 output pins as shown in figure 3.57.

#### Input pins:

1. **State:** it represents the current state depending on N1, N2 and N3. It takes definite values 1, 2 and 3.
2. **Ped1:** It represent if there are pedestrians in street #1
3. **Ped2:** It represent if there are pedestrians in street #2
4. **Ped3:** It represent if there are pedestrians in street #3
5. **Timer Tp51:** it represents the time that the pedestrians in street #1 wait before an interrupt occurs.
6. **Timer Tp52:** it represents the time that the pedestrians in street #2 wait before an interrupt occurs.
7. **Timer Tp53:** it represents the time that the pedestrians in street #3 wait before an interrupt occurs.

#### Output pins:

**Priority number:** it represents the pedestrians' street number that is selected to cross the road.

### Operation of Priority Block

- 1- If the state hasn't changed, the priority remains as it is.
- 2- If the state has changed:

The case structure inside the priority block has only one input that shows the existence of pedestrians in the three streets as shown in figure 3.58. All pedestrians are added as inputs to the block "Replace array subset" that changes these inputs in to binary numbers. These binary numbers are converted in to an integer number. Each integer number represents a case as shown in table 3.12.

Table 3.11 Binary to integer case mapping

Current state	Case number
Ped1	1
Ped2	2
Ped3	4
Ped1 and Ped2	3
Ped1 and Ped3	5
Ped2 and Ped3	6
Ped1 and Ped2 and Ped3	7

According to the case number the priority is determined by:

- a- If case number equals 1, then Priority will be given to pedestrians in street #1.
- b- If case number equals 2, then Priority will be given to pedestrians in street #2.
- c- If case number equals 4, then Priority will be given to pedestrians in street #3.
- d- If case number equals 3, then we will compare between waiting time of pedestrians in street #1 and 2 (Tp51 and Tp52), priority will be given to the pedestrians of longer waiting time.
- e- If case number equals 5, then we will compare between waiting time of pedestrians in street #1 and 3 (Tp51 and Tp53), priority will be given to the pedestrians of longer waiting time.
- f- If case number equals 6, then we will compare between waiting time of pedestrians in street #2 and 3 (Tp52 and Tp53), priority will be given to the pedestrians of longer waiting time.
- g- If case number equals 7, then we will compare between waiting time of pedestrians in street #1, 2 and 3 (Tp51, Tp52 and Tp53), priority will be given to the pedestrians of longer waiting time.

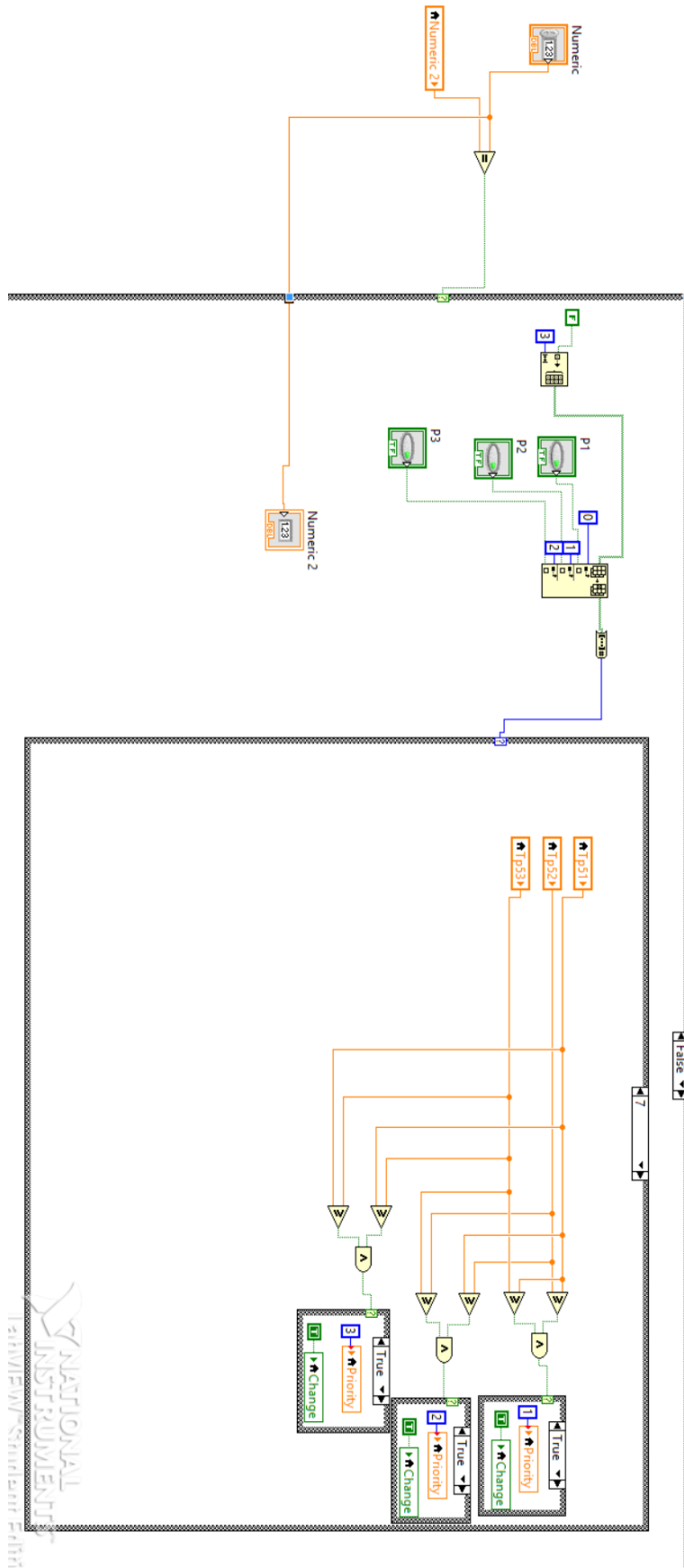


Figure 3.58 Priority block

# Chapter 4

---

## Hardware Setup

The previous chapters have introduced the problem, suggested a solution, simulated and implemented the design using software applications. This chapter will introduce all the hardware equipment that were used to help implement this system.

The function of the system is that it counts the number of vehicles, emergency cars and pedestrians and accordingly it decides who has the priority to cross the intersection. Therefore, for the system to count, it needs an equipment to read the number of cars (RFID) and to decide who has the priority to cross the intersection it need a controller (NI MyRIO).

For better illustration of the whole system, this chapter includes block diagrams that show how every equipment in the system interfaces with one another, description of the used RFID and a description of NI MyRIO.

## 4.1. Block Diagrams

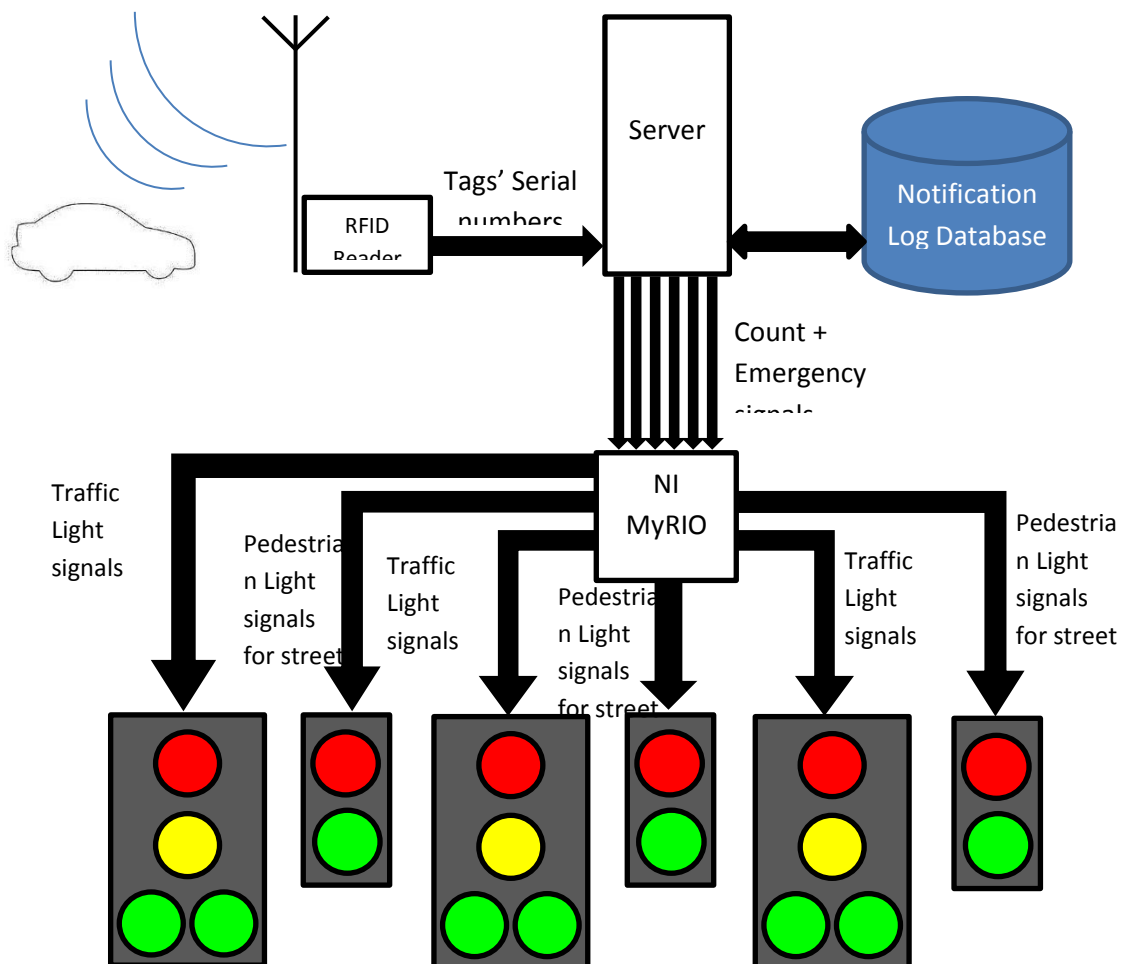


Figure 4.59 Block diagram showing the RFID-based Intelligent Traffic Light System

The block diagram shown in figure 4.1 describes the system exactly how it is implemented. First the antenna receives the serial number of the tag and the RFID reader, which is connected to the antenna via coaxial cable, reads and sends the data to the server. Using queries, the server is able to count the cars in every street and send it to NI MyRIO, and if there is an emergency, it sends a signal to indicate where the emergency is. Then NI MyRIO processes these signals to decide which street has the priority to have green light.

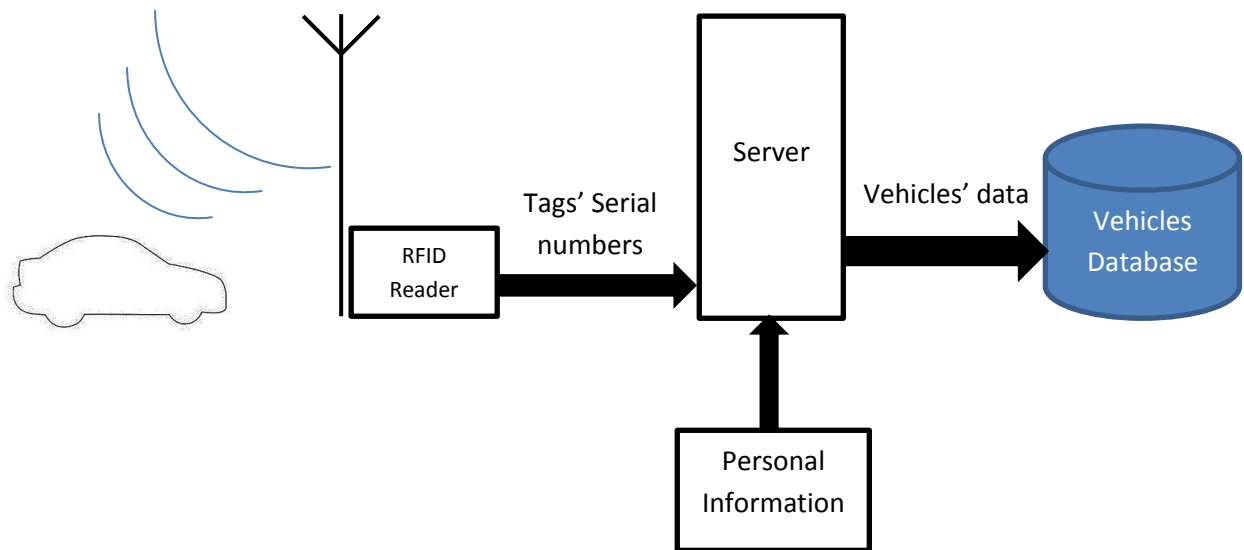


Figure 4.60 Block diagram of adding new vehicles to Database

The block diagram shown in figure 4.2 describes how the vehicle’s data is added to the “Vehicle’s Database” as described in chapter 2. The owner enters his personal information and his vehicle’s information, and then when registering the vehicle, the server waits for a car to appear in front of the antenna and add its serial number to the database as well.

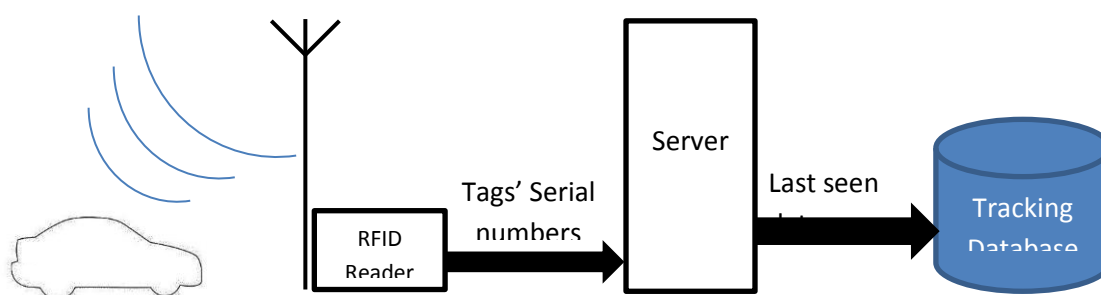


Figure 4.61 Block diagram of the tracking database

The block diagram shown in figure 4.3 describes how the vehicle’s tracking data is added to the “Tracking Database” as discussed in Chapter 2. The serial numbers of all the vehicles in front of the antenna are sent to the server and then the server adds them with some conditions to the database.



## 4.2. Radio Frequency Identification

The Radio Frequency Identification (RFID) has been introduced in this book in Chapter 1. In this section, the RFID technology used in this project will be discussed. One of the most important questions asked before designing the system was what specifications of the RFID should be chosen and how will it affect the performance of the project. The answers to these questions will be discussed in this section as well.

### 4.2.1. Description of the used RFID

The RFID technology used in this system consists of 3 main components:

1. Reader
  - a. Supply power to the antenna
  - b. Receives signals from the antenna
  - c. Convert the received signal to a serial number
2. Antenna
  - a. Broadcast signal to tags
  - b. Receives signals from tags
3. Tag
  - a. Reflects its own serial number through the signal that came from the antenna

This project models a traffic light system. This means that in order to model the system to scale, a long range RFID technology (long range reader, long range antennas and long range tags) must be used to be able to hang the antenna as high as the traffic lights and in the meantime it must be able to capture all the tags stuck on all vehicles in the range designed.

The system models a T-intersection; therefore, three antennas must be used, one for each street, to differentiate whether the vehicles are on street #1, street #2 or street #3. However, the antennas can read tags that are on another street; for example: a vehicle on street #2 could be read by the antennas hung on street #2 and street #1. This problem can be solved by getting a directive antenna or an antenna with a narrow beam width.

As any intersection, there will be more than one vehicle arriving at the same time which means more than one reading of tags at the same time. Therefore, the reader must be able to interpret and convert the signals received from the antennas as fast as it could without dropping any signal caused by interference of two readings.

To summarize the specifications needed:

- Long range RFID technology
- Three directive antennas
- Ability to read multiple tags at an instance

**The specifications could be met by the following RFID components:**

1. **Reader:** ID ISC.MRMU102-A UHF Reader Module developed by FEIG ELECTRONIC GmbH

Table 4.12 Specifications of the RFID reader

Power Supply	24V DC
Maximum Output Power	500 mW
Read Range	Up-to 12 m
Applications	Standard UHF Applications with readings > 3m
Features	Anti-Collision Real-time Clock

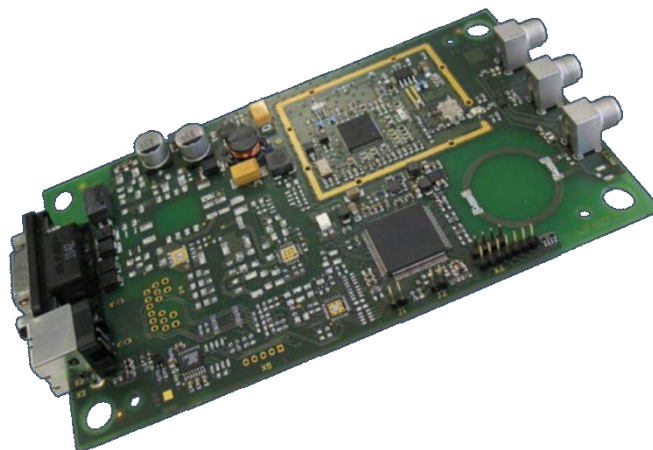


Figure 4.62 RFID Reader used in the system

**2. Antenna:** ID ISC.ANT.U270/270 developed by FEIG ELECTRONIC GmbH

Table 4.13 Specifications of RFID antennas

Dimension	270 x 270 x 57 mm <sup>3</sup>
3 dB beam width	65° x 65°
Gain	9 dBic
Polarization	Circular
Weight	1.21g



Figure 4.63 Antenna used in the system

### 3. Tag: Passive RFID tags

Table 4.14 Specifications of RFID tags

Power source	Energy transfer from reader via RF
Availability of tag power	Within field of antenna
Strength from antenna to tag	Very high
Communication Range	Short (up to 10m)

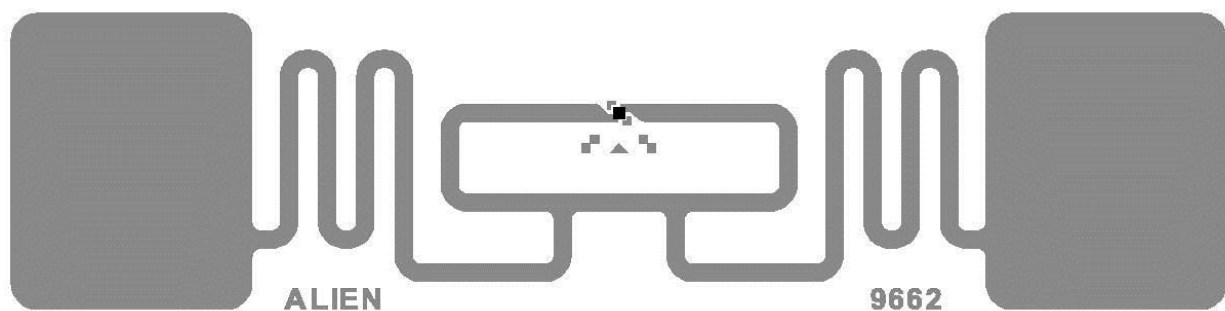


Figure 4.64 RFID Tag used in the system

#### 4.2.2. Connection to Server

After getting the components with the specifications mentioned above, the system needs one more step to be able to count the vehicles in the streets of the intersection. The reader must be connected to the database to be able to process the data sent from the reader. With these data, the system will be able to know the number of vehicles in all streets, emergency (if any) and the time any vehicle arrived at the intersection.

##### **The data sent to the database from the reader are:**

1. Unique identifier
2. Antenna Number
3. Serial Number
4. Time and date of arrival

The reader is used in Buffered Read Mode which means that the read tag will be buffered in the server for further use. See Appendix C for instructions on how to install the RFID Buffer Read Mode service on the PC. This allows the database to save the data sent by the reader in the Notification Log Database as shown in figure 4.1, which is updated whenever a tag passes in front of the antenna.

After that it is as simple as writing a query to get the count of the vehicle and everything the system needs. Figure 4.7 shows an example of the Notification Log Database.

NotificationLogId	ReaderIPAddr...	AntennaNum...	SerialNumber	UserMemory	NotificationLogDate
a38dd83a-bc5a-49c4-ae92-2f818932b54f		1	000240000000000000000000	NULL	2016-06-05 16:58:11.217
d1a2ab45-44a9-4287-8ca3-b90953efa82c		1	001500000000000000000000	NULL	2016-06-05 16:58:11.213
a180cfe6-3580-4da1-a90d-c63b1bb5a721		1	001700000000000000000000	NULL	2016-06-05 16:58:11.217

Figure 4.65 Example of the Notification Log Database

## 4.3. NI MyRIO



Figure 4.66 NI MyRIO

The controller used for this system is NI MyRIO, shown in figure 4.8. The controller's job is to read from the server, process the readings and output the traffic and pedestrian light signals.

### 4.3.1. Connection to Database

First, Labview, the language used to program NI MyRIO, has the option to write queries to read from and write into database, but due to license issues, this project could not use this option. However, the reading from the database was done using an intermediate C# program that reads from the database and writes into a .txt file and save it in the MyRIO memory. See Appendix B for more information.

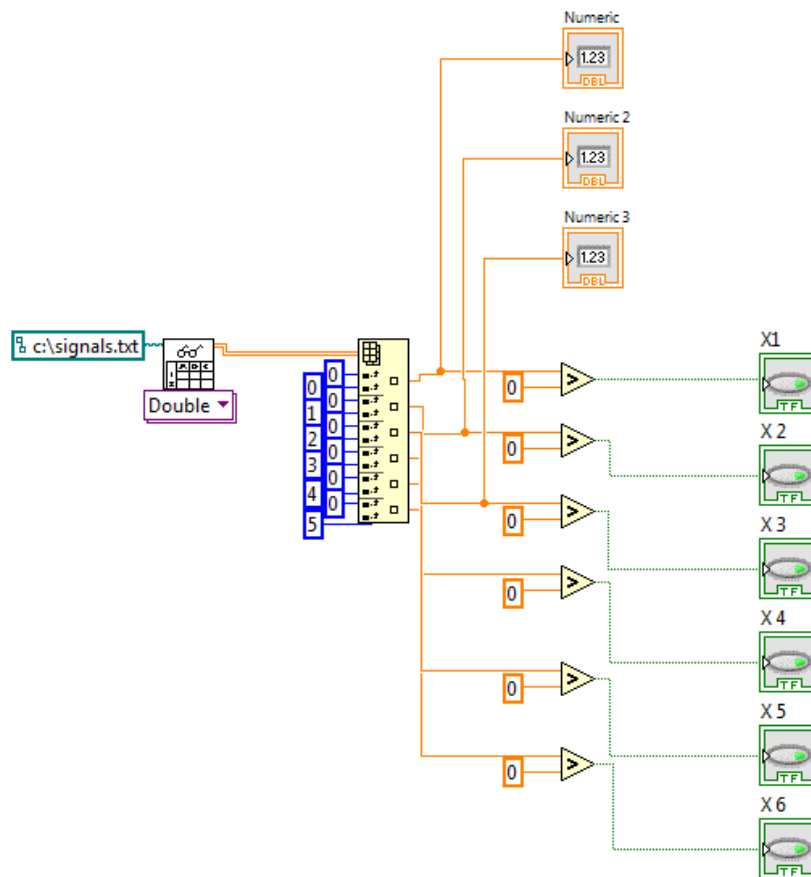


Figure 4.67 Database Connectivity Toolkit implementation on Labview

After the intermediate program saves the readings from the database into a .txt file in MyRIO memory, using Labview:

- Open .txt file
- Convert into a spreadsheet
- Insert every cell in the spread sheet into array
- Compare the output of the array with zero to get Boolean values

### 4.3.2. Input/Output Map

For this system, there are 2 inputs:

1. From database
2. From pedestrians pushbuttons

There are three input pushbuttons, so the mapping of the inputs is as shown in table 4.4 and figure 4.10.

Table 4.15 Mapping of input pushbuttons on Myrio

Input	Pin Number
Pushbutton #1	C0
Pushbutton #2	C1
Pushbutton #3	C2

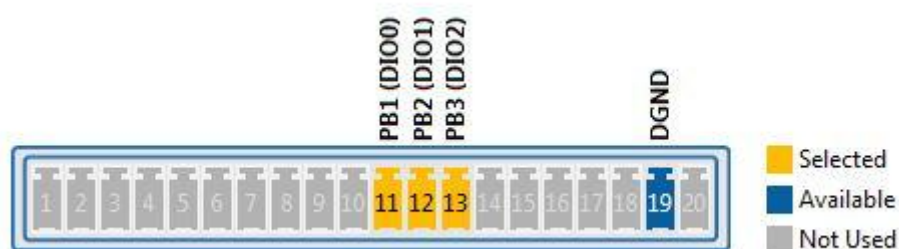


Figure 4.68 Mapping of input pushbuttons on MyRIO

After the inputs are processed the controlling part begins. NI MyRIO controls three streets every street has:

1. Red traffic light
2. Yellow traffic light
3. Left green traffic light
4. Right green traffic light
5. Red pedestrians light
6. Green pedestrians light

Every street needs 6 output signals, which is 18 output signals for the three streets. These signals will be mapped according to the tables 4.5, 4.6 and 4.7 respectively and figures 4.11, 4.12, 4.13 and 4.14 respectively.



Table 4.16 Mapping of Street #1 outputs on MyRIO

Output for Street #1	Pin Number
Red traffic light	A6
Yellow traffic light	A7
Left green traffic light	A8
Right green traffic light	A9
Red pedestrians light	A10
Green pedestrians light	A11

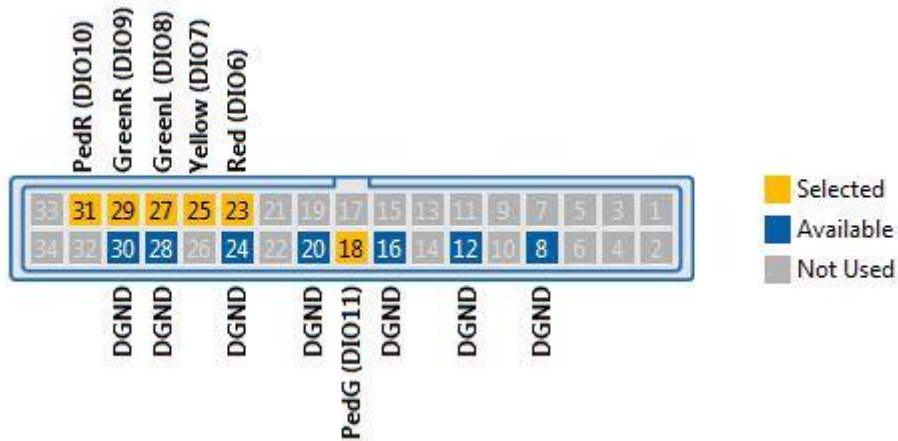


Figure 4.69 Mapping of Street #1 outputs on MyRIO

Table 4.17 Mapping of Street #2 outputs on MyRIO

Output for Street #2	Pin Number
Red traffic light	A12
Yellow traffic light	A13
Left green traffic light	A14
Right green traffic light	A14
Red pedestrians light	C6
Green pedestrians light	C7

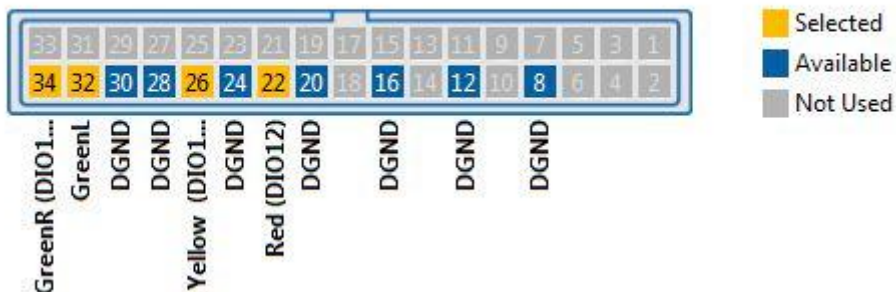


Figure 4.70 Mapping of Street #2 outputs on MyRIO

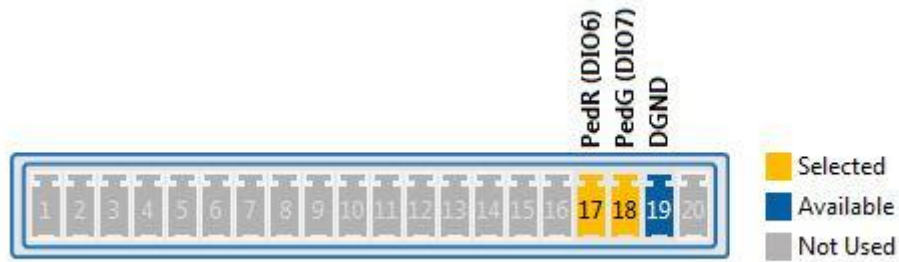


Figure 4.71 Mapping of Street #2 outputs on MyRIO

Table 4.18 Mapping of Street #3 outputs on MyRIO

Output for Street #3	Pin Number
Red traffic light	A0
Yellow traffic light	A1
Left green traffic light	A2
Right green traffic light	A3
Red pedestrians light	A4
Green pedestrians light	A5

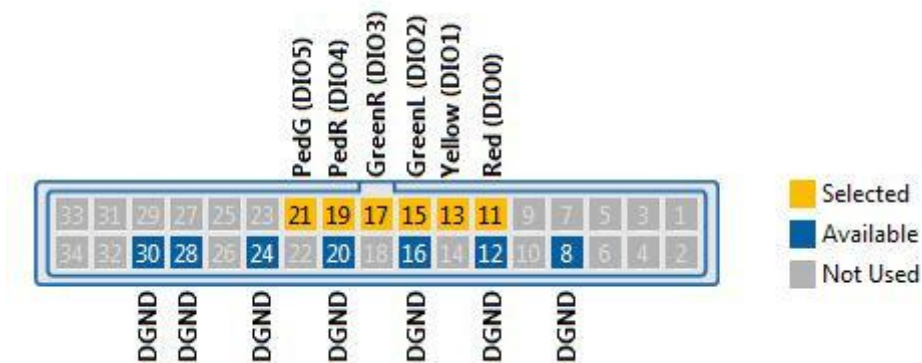


Figure 4.72 Mapping of Street #3 outputs on MyRIO

# Chapter 5

---

## Maquette Designing and Testing

**T**his chapter is mainly concerned with the design of the maquette, including the streets, the traffic lights and the antenna holders. In addition to that there will be an explanation of the tests conducted on the maquette to make sure it is fully functional.

This chapter describes the moment of truth to what this whole project was about, demonstrated in a smaller scale model that accounts for a real life T-intersection with the traffic lights and all the real life variables it could face.

## 5.1. Basic T-intersection Structure

The basic structure for the maquette's T-intersection consists of two basic components:

- The steel structure for the intersection
- Printed streets for each of the three streets in the intersection

### 5.1.1. The Steel Structure

The steel structure had 3 wooden plates where the antennas are planted as shown in figure 5.1 to read the RFID tags planted on the cars which model the vehicles in the street. This steel structure models the intersection itself of the three streets. Three out of the four sides of the cubic structure represent those three streets.



Figure 5.73 Antenna planted on one of the wooden plates

The fourth side of the steel structure of the maquette is the setup place for the Laptop modeling the system server and the MyRIO (previously discussed in chapter 1).



Figure 5.74 Setup place on the maquette

The traffic light was modeled using red, yellow and green LEDs just as it would look like on a real traffic light plus an extra green and red ones for the pedestrians to indicate whether they are able to cross the street or not.

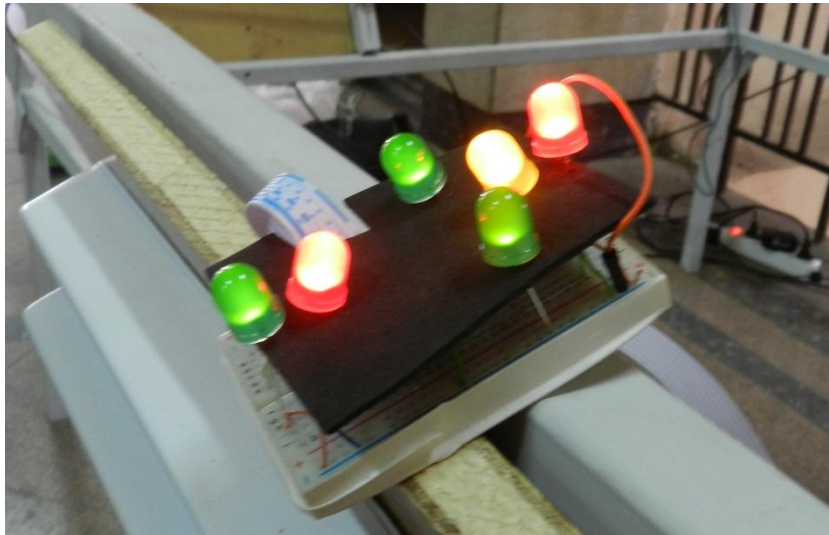


Figure 5.75 Traffic lights

Notice that there are two green LEDs per traffic light to distinguish between those wanting to turn right and left

As previously mentioned vehicles fall under one of two main categories. Vehicles are either classified as emergency vehicles or civilian vehicle as shown in figure 5.4 and 5.5 respectively. The first step to declaring a vehicle under one of the two categories is to set up the tag on the vehicle.



Figure 5.76 Emergency Vehicles with RFID Tags



Figure 5.77 Civilian Vehicles with RFID Tags

The next step is setting up the vehicle to be facing antenna number one and verifying the details of the vehicle via the GUI as explained in Chapter three.



Figure 5.78 Facing vehicle in front of antenna

### 5.1.2. Modeling the Streets

The streets were modeled using printed posters to divide each street into lanes and give a scaled version of the real life streets fitted to the project's maquette. Figure 5.6 shows the scaled printed streets of the maquette.

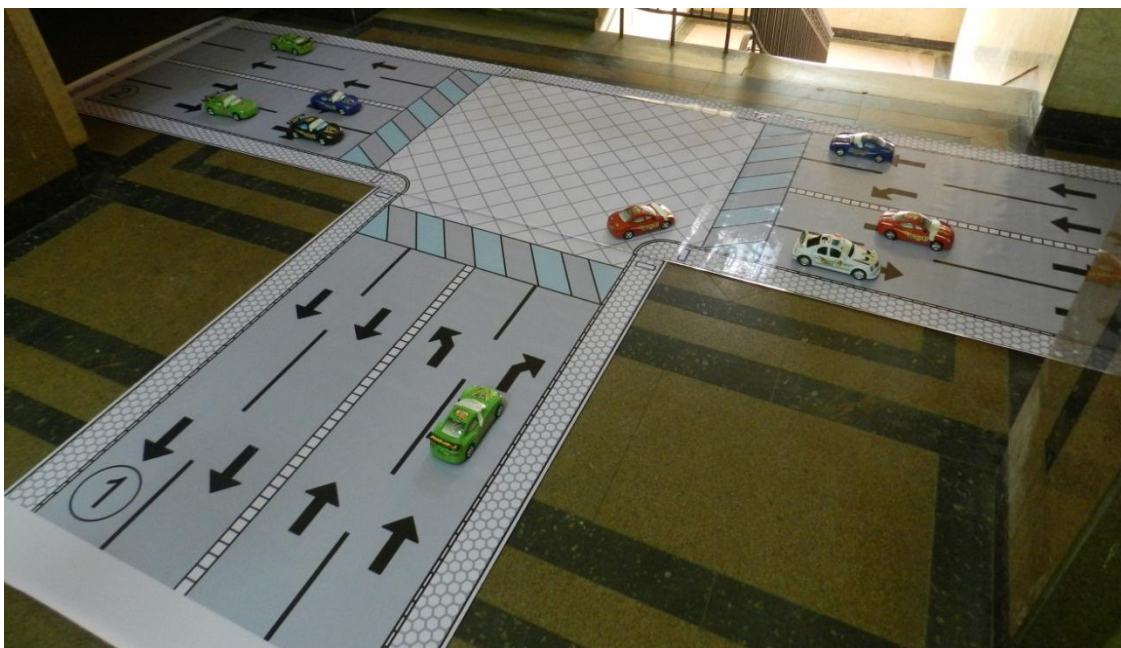


Figure 5.79 Scaled version of the T-intersection

## 5.2. Testing the Hardware

After the hardware has been fully set up now it is mandatory to test the functionalities of the hardware to make sure it fully responds to what it was designed to do. To do this, several scenarios were made to put the project to the test

**Test#1:** Putting cars at each traffic light but with different traffic intensity at each. In other words, the total number of cars at Street#1 > Street#2 > Street#3.

**To Pass the Test:** Green traffic light time for street#1 should be greater than that of street#2 which is greater than that of street#3

**Test#2:** Cars are available only at two streets only while the third is empty

**To Pass the Test:** Traffic light keeps on switching between the two streets only and no green light is ever given to the empty street

**Test#3:** Cars are available at only one street

**To Pass the Test:** Green Light is only given to the street with cars until there are no more cars

**Test#4:** Emergency Vehicle arrives at a certain street

**To Pass the Test:** Traffic light turns green for the street having the emergency vehicle and stays on the green light till the emergency vehicle passes.

**Test#5:** Emergency vehicles arrive simultaneously at two different streets

**To Pass the Test:** Green light is given to the emergency car arriving first till it passes then green light is given to the second car till it passes.

**Test#6:** Pedestrians want to pass one of the streets.

**To Pass the Test:** When it is time for pedestrians to pass, the green lights of the other two streets that might lead to a cross-over must not light up as well as the green light of the street they are passing.



**Test#7:** If pedestrians request passing a street whilst others request passing another street moments after

**To Pass the Test:** Make sure the ones that have had the longest waiting time are the ones that pass first

**Test #8:** If pedestrians have waited a very long time without getting a turn to pass.

**To pass the Test:** System interrupt occurs where all the Vehicles traffic lights go red and let the pedestrian pass, then go back to completing the original round.

**Test#9:** Emergency vehicle arrives while pedestrians are crossing the road.

**To Pass the Test:** Priority is always given to the emergency vehicle and therefore the pedestrians light goes red till the emergency vehicle passes through.

After all the tests have been verified, it is for sure now that the hardware is working as designed on the software discussed in previous chapters.

# Chapter 6

---

## Conclusion

**I**n conclusion, the project has shown a great deal of efficiency supported by the simulation results explained in chapter 2. Given the fact that the project targets on fixing a worldwide problem that is present even in the most advanced countries that invest a great deal of money hoping to decrease the average wait time of vehicles in traffic intersections.

The project does not only target decreasing the average wait time but it targets finding the point of providing the least possible Wait Time within the project's logical capabilities. This is done through a series of simulations to identify the optimum point that would lead to optimal performance.

The accumulation of this performance improvement in every traffic light intersection increases the overall improvement even more and that is where the real strength comes.

## 6.1. Possible Future Improvements

Though the project improves the traffic by a relatively reasonable range it is possible and encouraged to widen the scope of the project and find better ways that could be implemented to make the project even more efficient.

Some of the possible future improvements would be:

- Creating a direct link with the police department to alarm them (in real time) when a stolen car is seen at any traffic light
- The possibility of safely killing the engine of a car which has been detected and reported as stolen without causing hazards
- Creating priority codes for each emergency passing such that the traffic light would know which emergency vehicle is more in need of the time. (Sometimes an emergency vehicle serving an extremely dangerous case would come moments after a one serving a less dangerous case; therefore the second vehicle should be given the priority )
- Adding more antennas to detect if a car has illegally crossed a red light.
- Possibilities of allowing vehicles to communicate with one another or communicate with traffic light by means of exchanging data.
- Creating a mobile application integrated with the project for different parties (civilians, police, ministry of transportation)

# References

---

*<http://www.ni.com/company/about-ni/>*

*<http://www.ni.com/myrio/what-is/>*

*<https://en.wikipedia.org/wiki/LabVIEW>*

*<http://www.ni.com/myrio/>*

*<http://www.worldbank.org/content/dam/Worldbank/TWB-Executive-Note-Eng.pdf>*

*<http://www.impinj.com/resources/about-rfid/>*

*<http://www.rfidjournal.com/articles/view?1334/>*

*[http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931\\_read-41000/](http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/)*

# Appendix A

---

## Matlab Codes

### A.1. Intelligent Traffic Light system with pedestrians

```
clc
clear
Street3 = 'ggrrrrrrr';
Street3L = 'ngrrrrrrg';
Street3R = 'grrrrrrgr';
Yellow3 = 'yyrrrrrrr';

Street2 = 'rrrrggrrr';
Street2L = 'rrrrrggr';
Street2R = 'rrrrggr';
Yellow2 = 'rrrryyrrr';

Street1 = 'rrggrrrrr';
Street1L = 'rrrgrrgr';
Street1R = 'rrgrrrrg';
Yellow1 = 'rryyrrrrr';

ALLRED = 'rrrrrrggg';

priority=1;
totaltime=tic();
```

```

tr=[20 20 20 50 80 80];
for vh=6:6
    A=vh*50;
    ODMatrixP(A);
    vehtime=tic();
    for tr=6:6
        cycletime=tic();
        x=system('sumo-gui -c
C:\Users\Lenovo\Desktop\GPshortcuts\Project\gp.sumocfg&');
        traci.init();
        step=0;
        traci.trafficlights.setRedYellowGreenState('0', Street1);
        U1=1>0;
        U2=1<0;
        U3=1<0;
        NUM=1;
        p1=1<0;
        p2=1<0;
        p3=1<0;
        M=1;
        R=0;
        T1=tr*5;
        Tp51=300;
        Tp52=300;
        Tp53=300;
        Tp=10;
        T2=5;
        t1=0;
        t2=0;
        tp=0;
        tp51=0;
        tp52=0;
        tp53=0;
        pr=1;
        waits=cell(0);
        waitn=[0 0 0];
        [pr]=incPr(pr,1,p1,p2,p3);
        while step<3600
            x1n(step+1)= traci.edge.getLastStepVehicleNumber('5to0');
            x2n(step+1)= traci.edge.getLastStepVehicleNumber('1to0');
            x3n(step+1)= traci.edge.getLastStepVehicleNumber('2to0');

            p1= length(traci.edge.getLastStepPersonIDs('1to0'))>0;
            p2= length(traci.edge.getLastStepPersonIDs('0to5'))>0;
            p3= length(traci.edge.getLastStepPersonIDs('5to0'))>0;

            if (p1)
                tp51=tp51+1;
            end
            if (p2)
                tp52=tp52+1;
            end
            if (p3)
                tp53=tp53+1;
            end
            Z=(Tp51<=tp51) || (Tp52<=tp52) || (Tp53<=tp53);

            x1=x1n(step+1)>0;
            x2=x2n(step+1)>0;
            x3=x3n(step+1)>0;

```

```

    Q1=((M==2 && t2>=T2 && Tp51>tp51 && ~x2 && ~x3) || (M==4 &&
t2>=T2 && Tp52>tp52 && x1 && ~x3) || (M==6 && t2>=T2 && Tp53>tp53 && x1));
    Q2=((M==4 && t2>=T2 && Tp52>tp52 && ~x1 && ~x3) || (M==6 &&
t2>=T2 && Tp53>tp53 && ~x1 && x2) || (M==2 && t2>=T2 && Tp51>tp51 && x2));
    Q3=((M==6 && t2>=T2 && Tp53>tp53 && ~x1 && ~x2) || (M==2 &&
t2>=T2 && Tp51>tp51 && ~x2 && x3) || (M==4 && t2>=T2 && Tp52>tp52 && x3));
    U1= (Q1 || (U1 && ~Q2 && ~Q3));
    U2= (Q2 || (U2 && ~Q1 && ~Q3));
    U3= (Q3 || (U3 && ~Q1 && ~Q2));
    if (U1 && NUM~=1) || (U2 && NUM~=2) || (U3 && NUM~=3)
        if (U1)
            [pr]=incPr(pr,1,p1,p2,p3);
            NUM=1;
        elseif (U2)
            [pr]=incPr(pr,2,p1,p2,p3);
            NUM=2;
        else
            [pr]=incPr(pr,3,p1,p2,p3);
            NUM=3;
        end
    end

    if ((Q1 && ~p2 && ~p3) || (M==11 && tp>=Tp) || (M==12 &&
tp>=Tp) || (R==1 && Tp<=tp && U1))
        M=1;
        t1=t1+1;
        tp=0;
        t2=0;
        R=0;
        traci.trafficlights.setRedYellowGreenState('0', Street1);

    elseif ((Q1 && p3 && pr==3) || (M==1 && Tp51>tp51 && Tp53<=tp53
&& Z) )
        tp53=0;
        M=11;
        t1=t1+1;
        tp=tp+1;
        t2=0;
        R=0;
        traci.trafficlights.setRedYellowGreenState('0', Street1L);

    elseif ((Q1 && p2 && pr==2) || (M==1 && Tp51>tp51 && Tp53>tp53
&& Tp52<=tp52 && Z))
        tp52=0;
        M=12;
        t1=t1+1;
        tp=tp+1;
        t2=0;
        R=0;
        traci.trafficlights.setRedYellowGreenState('0', Street1R);

    elseif (M==1 && ~Z && ((x2 || x3) && (t1>=T1 || ~x1)) || (M==1
&& Z && Tp51<=tp51))
        M=2;
        t1=0;
        tp=0;
        t2=t2+1;
        R=0;
        traci.trafficlights.setRedYellowGreenState('0', Yellow1);

```

```

        elseif ((Q2 && ~p1 && ~p3) || (M==31 && tp>=Tp) || (M==32 &&
tp>=Tp) || (R==1 && Tp<=tp && U2))
            M=3;
            t1=t1+1;
            tp=0;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Street2);

        elseif ((Q2 && p1 && pr==1) || (M==3 && Tp52>tp52 &&
Tp51<=tp51 && Z) )
            tp51=0;
            M=31;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Street2L);

        elseif ((Q2 && p3 && pr==3) || (M==3 && Tp52>tp52 && Tp51>tp51
&& Tp53<=tp53 && Z) )
            tp53=0;
            M=32;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Street2R);

        elseif (M==3 && ~Z && ((x1 || x3) && (t1>=T1 || ~x2)) || (M==3
&& Z && Tp52<=tp52) )
            M=4;
            t1=0;
            tp=0;
            t2=t2+1;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Yellow2);

        elseif ((Q3 && ~p1 && ~p2) || (M==51 && tp>=Tp) || (M==52 &&
tp>=Tp) || (R==1 && Tp<=tp && U3))
            M=5;
            t1=t1+1;
            tp=0;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Street3);

        elseif ((Q3 && p2 && pr==2) || (M==5 && Tp53>tp53 &&
Tp52<=tp52 && Z) )
            tp52=0;
            M=51;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0', Street3L);

        elseif ((Q3 && p1 && pr==1) || (M==5 && Tp53>tp53 && Tp52>tp52
&& Tp51<=tp51 && Z) )
            tp51=0;

```



```

M=52;
t1=t1+1;
tp=tp+1;
t2=0;
R=0;
traci.trafficlights.setRedYellowGreenState('0', Street3R);

elseif (M==5 && ~Z && ((x1 || x2) && (t1>=T1 || ~x3)) || (M==5
&& Z && Tp53<=tp53) )
M=6;
t1=0;
tp=0;
t2=t2+1;
R=0;
traci.trafficlights.setRedYellowGreenState('0', Yellow3);

elseif ((M==2 && T2<=t2 && Tp51<=tp51) || (M==4 && T2<=t2 &&
Tp52<=tp52) || (M==6 && T2<=t2 && Tp53<=tp53))
tp51=0;
tp52=0;
tp53=0;
R=1;
t1=0;
tp=0;
t2=0;
M=0;
traci.trafficlights.setRedYellowGreenState('0', ALLRED);
else
if (M==1 || M==3 || M==5 || M==11 || M==12 || M==31 || M==32
|| M==51 || M==52)
t1=t1+1;
elseif (M==2 || M==4 || M==6)
t2=t2+1;
else
tp=tp+1;
t1=0;
t2=0;
end
if (M==11 || M==12 || M==31 || M==32 || M==51 || M==52)
tp=tp+1;
end
end

z1=traci.edge.getLastStepVehicleIDs('5to0');
z2=traci.edge.getLastStepVehicleIDs('1to0');
z3=traci.edge.getLastStepVehicleIDs('2to0');

[waits, waitn]=inWait2(z1,waits,waitn,step,1);
[waits, waitn]=inWait2(z2,waits,waitn,step,2);
[waits, waitn]=inWait2(z3,waits,waitn,step,3);

traci.simulationStep();
step=step+1;
end
traci.close();
av1(vh,tr,1)=mean(x1n);
av1(vh,tr,2)=var(x1n);
av2(vh,tr,1)=mean(x2n);
av2(vh,tr,2)=var(x2n);
av3(vh,tr,1)=mean(x3n);

```

```

    av3(vh, tr, 2) = var(x3n);

    k = find(waitn(:, 2) == 3599, 1);
    waitn(k:end, :) = [];
    [R, C] = size(waitn);
    av(vh, tr) = sum(waitn(:, 2) - waitn(:, 1)) / R - 1;
    cyc(vh, tr) = toc(cycletime);
end
vehtimecyc(vh) = toc(vehtime);
end
ttime = toc(totaltime);

```

### A.1.1. ODMatrix for Three Streets with Pedestrians

```

function [] = ODMatrixP(A)
a = A ./ 3600;
str1 = '<routes>\n';
str2 = '<vType id="Car" accel="0.8" decel="4.5" sigma="0.5" length="2.5"
minGap="2.5" maxSpeed="16.67" guiShape="passenger"/>\n\n';
str3 = '<flow id="5" begin="0" end="3600" probability="%d" from="in5"
to="out1" via="5to0 0to1" type="Car"/>\n';
str4 = '<flow id="6" begin="0" end="3600" probability="%d" from="in5"
to="out2" via="5to0 0to2" type="Car"/>\n';
str5 = '<flow id="1" begin="0" end="3600" probability="%d" from="in1"
to="out2" via="1to0 0to2" type="Car"/>\n';
str6 = '<flow id="2" begin="0" end="3600" probability="%d" from="in1"
to="out5" via="1to0 0to5" type="Car"/>\n';
str7 = '<flow id="3" begin="0" end="3600" probability="%d" from="in2"
to="out5" via="2to0 0to5" type="Car"/>\n';
str8 = '<flow id="4" begin="0" end="3600" probability="%d" from="in2"
to="out1" via="2to0 0to1" type="Car"/>\n';

str = [str1 str2 str3 str4 str5 str6 str7 str8];
fileID = fopen('gp.rou.xml', 'w');
if (size(A) > 1)
    fprintf(fileID, str, a(1,2), a(1,3), a(2,1), a(2,3), a(3,1), a(3,2));
else
    fprintf(fileID, str, a, a, a, a, a, a);
end
fclose(fileID);

AP = floor(linspace(0, 3600, 300));

ped1 = [AP' (0:299)' ones(300,1)];
ped2 = [AP' (300:599)' 2*ones(300,1)];
ped3 = [AP' (600:899)' 3*ones(300,1)];

peds = [ped1; ped2; ped3];
P = sortrows(peds);

fileID = fopen('gp.rou.xml', 'a+');
for i = 1:900
    if (P(i,3) == 1)
        pro = '1to0' to = '0to2';
        pro2 = '45' arrivalPos = '-30';
    elseif (P(i,3) == 2)
        pro = '5to0' to = '2to0';
        pro2 = '45' arrivalPos = '-30';
    end
end

```

```

else
    pro='0to5" to="0to1';
    pro2='30" arrivalPos="-45';
end
ppp='<person id="person%d" depart="%d">\n<walk from="%s"
departPos="%s"/>\n</person>\n';
fprintf(fileID, ppp, P(i,2),P(i,1),pro,pro2);
end
fprintf(fileID, '\n</routes>');
fclose(fileID);

```

### A.1.2. Incrementing Priority

```

function [X]=incPr(x,n,p1,p2,p3)
if (n==1)
    X=x+1;
    if (X>3)
        X=1;
    end
    if (p2 && X==2)
        X=X;
    elseif (p3 && X==3)
        X=X;
    else
        X=X+1;
        if (X>3)
            X=1;
        end
    end
    if (X==n)
        X=X+1;
        if (X>3)
            X=1;
        end
    end
end
if (n==2)
    X=x+1;
    if (X>3)
        X=1;
    end
    if (p1 && X==1)
        X=X;
    elseif (p3 && X==3)
        X=X;
    else
        X=X+1;
        if (X>3)
            X=1;
        end
    end
    if (X==n)
        X=X+1;
        if (X>3)
            X=1;
        end
    end
end
end
end

```

```

if (n==3)
    X=x+1;
    if (X>3)
        X=1;
    end
    if (p1 && X==1)
        X=X;
    elseif (p2 && X==2)
        X=X;
    else
        X=X+1;
        if (X>3)
            X=1;
        end
    end
end
if (X==n)
    X=X+1;
    if (X>3)
        X=1;
    end
end
end
end

```

## A.2. Intelligent Traffic Light System Without pedestrians

```

clc
clear
Street3 = 'ggrrrrrrr';
Street3L = 'rgrrrrrrr';
Street3R = 'grrrrrrrr';
Yellow3 = 'yyrrrrrrr';

Street2 = 'rrrrggrrr';
Street2L = 'rrrrrgrrr';
Street2R = 'rrrrgrrrr';
Yellow2 = 'rrrryyrrr';

Street1 = 'rrggrrrrr';
Street1L = 'rrrgrrrrr';
Street1R = 'rrgrrrrrr';
Yellow1 = 'rryyrrrrr';

priority=1;
totaltime=tic();
for vh=1:6
    A=50*vh;
    ODMatrix(A);
    for tr=3:15
        x=system('sumo -c
C:\Users\Lenovo\Desktop\GPshortcuts\Project\gp2.sumocfg');
        traci.init();
        step=0;
        traci.trafficlights.setRedYellowGreenState('0', Street1);
        M=1;
        T1=tr*5;
        T2=5;
        t1=0;
        t2=0;
        waits=cell(0);
    end
end

```

```

waitn=[0 0 0];
while step<3600
    x1n(step+1)= traci.edge.getLastStepVehicleNumber('5to0');
    x2n(step+1)= traci.edge.getLastStepVehicleNumber('1to0');
    x3n(step+1)= traci.edge.getLastStepVehicleNumber('2to0');

    x1=x1n(step+1)>0;
    x2=x2n(step+1)>0;
    x3=x3n(step+1)>0;
    if (t2>=T2 &&((M==6 && x1) || (M==4 && ~x3)))
        traci.trafficlights.setRedYellowGreenState('0', Street1);
        M=1;
        t2=0;
    elseif (t2>=T2 && ((M==6 && ~x1) || (M==2 && x2)))
        traci.trafficlights.setRedYellowGreenState('0', Street2);
        M=3;
        t2=0;
    elseif (t2>=T2 && ((M==4 && x3) || (M==2 && ~x2)))
        traci.trafficlights.setRedYellowGreenState('0', Street3);
        M=5;
        t2=0;
    elseif (M==1 && ((x2 || x3) && (t1>=T1 || ~x1)))
        traci.trafficlights.setRedYellowGreenState('0', Yellow1);
        M=2;
        t1=0;
    elseif (M==3 && ((x1 || x3) && (t1>=T1 || ~x2)))
        traci.trafficlights.setRedYellowGreenState('0', Yellow2);
        M=4;
        t1=0;
    elseif (M==5 && ((x1 || x2) && (t1>=T1 || ~x3)))
        traci.trafficlights.setRedYellowGreenState('0', Yellow3);
        M=6;
        t1=0;
    else
        if (M==1 || M==3 || M==5)
            t1=t1+1;
        else
            t2=t2+1;
        end
    end

    z1=traci.edge.getLastStepVehicleIDs('5to0');
    z2=traci.edge.getLastStepVehicleIDs('1to0');
    z3=traci.edge.getLastStepVehicleIDs('2to0');

    [waits, waitn]=inWait2(z1,waits,waitn,step,1);
    [waits, waitn]=inWait2(z2,waits,waitn,step,2);
    [waits, waitn]=inWait2(z3,waits,waitn,step,3);

    traci.simulationStep();
    step=step+1;
end
traci.close();
av1(vh,1)=mean(x1n);
av1(vh,2)=var(x1n);
av2(vh)=mean(x2n);
av2(vh,2)=var(x2n);
av3(vh)=mean(x3n);
av3(vh,2)=var(x3n);

```

```

k=find(waitn(:,2)==3599,1);
waitn(k:end,:)=[];
[R, C]=size(waitn);
av(vh,tr)=sum(waitn(:,2)-waitn(:,1))/R-1
end

end
toc(totaltime)

```

### A.2.1. ODMatrix for Three Streets without Pedestrians

```

function []=ODMatrix(A)
a=A./3600;
str1='<routes>\n';
str2='<vType id="Car" accel="0.8" decel="4.5" sigma="0.5" length="2.5"
minGap="2.5" maxSpeed="16.67" guiShape="passenger"/>\n\n';
str3='<flow id="5" begin="0" end="3600" probability="%d" from="in5"
to="out1" via="5to0 0to1" type="Car"/>\n';
str4='<flow id="6" begin="0" end="3600" probability="%d" from="in5"
to="out2" via="5to0 0to2" type="Car"/>\n';
str5='<flow id="1" begin="0" end="3600" probability="%d" from="in1"
to="out2" via="1to0 0to2" type="Car"/>\n';
str6='<flow id="2" begin="0" end="3600" probability="%d" from="in1"
to="out5" via="1to0 0to5" type="Car"/>\n';
str7='<flow id="3" begin="0" end="3600" probability="%d" from="in2"
to="out5" via="2to0 0to5" type="Car"/>\n';
str8='<flow id="4" begin="0" end="3600" probability="%d" from="in2"
to="out1" via="2to0 0to1" type="Car"/>\n';
str=[str1 str2 str3 str4 str5 str6 str7 str8];
fileID = fopen('gp.rou.xml','w');
if (size(A)>1)
    fprintf(fileID,str, a(1,2), a(1,3), a(2,1), a(2,3), a(3,1), a(3,2));
else
    fprintf(fileID,str, a, a, a, a, a, a);
end
fprintf(fileID,'</routes>');
fclose(fileID);

```

## A.3. Ordinary Traffic Light System

```

clc
clear
Street3 = 'ggrrrrrrr';
Street3L = 'rgrrrrrrg';
Street3R = 'grrrrrrrr';
Yellow3 = 'yyrrrrrrr';

Street2 = 'rrrrggrrr';
Street2L = 'rrrrrgrrg';
Street2R = 'rrrrgrrrr';
Yellow2 = 'rrrryyrrr';

Street1 = 'rrggrrrrr';
Street1L = 'rrrgrrrrg';
Street1R = 'rrgrrrrrr';

```

```

Yellow1 = 'r ryyrrrrrr';

priority=1;
totaltime=tic();
for vh=5:6
    A=50*vh
    ODMatrix(A);
    for tr=3:15
        if ~(vh==5 && tr<11)
            x=system('sumo -c
C:\Users\Lenovo\Desktop\GPshortcuts\Project\gp2.sumocfg');
            traci.init();
            step=0;
            traci.trafficlights.setRedYellowGreenState('0', Street1);
            M=1;
            T1=tr*5
            T2=5;
            Tp=10;
            t1=0;
            t2=0;
            tp=0;
            waits=cell(0);
            waitn=[0 0 0];
            while step<3600
                if (M==6 && t2>=T2)
                    t2=0;
                    M=11;
                    traci.trafficlights.setRedYellowGreenState('0',
Street1L);
                elseif (M==11 && tp>=Tp)
                    tp=0;
                    M=1;
                    traci.trafficlights.setRedYellowGreenState('0',
Street1);
                elseif (M==1 && t1>=T1-Tp)
                    t1=0;
                    M=2;
                    traci.trafficlights.setRedYellowGreenState('0',
Yellow1);
                elseif (M==2 && t2>=T2)
                    t2=0;
                    M=31;
                    traci.trafficlights.setRedYellowGreenState('0',
Street2L);
                elseif (M==31 && tp>=Tp)
                    tp=0;
                    M=3;
                    traci.trafficlights.setRedYellowGreenState('0',
Street2);
                elseif (M==3 && t1>=T1-Tp)
                    t1=0;
                    M=4;
                    traci.trafficlights.setRedYellowGreenState('0',
Yellow2);
                elseif (M==4 && t2>=T2)
                    t2=0;
                    M=51;
                    traci.trafficlights.setRedYellowGreenState('0',
Street3L);
                elseif (M==51 && tp>=Tp)
                    tp=0;

```

```

M=5;
traci.trafficlights.setRedYellowGreenState('0',
Street3);
elseif (M==5 && t1>=T1-Tp)
t1=0;
M=6;
traci.trafficlights.setRedYellowGreenState('0',
Yellow3);
else
if (M==1 || M==3 || M==5)
t1=t1+1;
elseif (M==2 || M==4 || M==6)
t2=t2+1;
else
tp=tp+1;
end
end
z1=traci.edge.getLastStepVehicleIDs('5to0');
z2=traci.edge.getLastStepVehicleIDs('1to0');
z3=traci.edge.getLastStepVehicleIDs('2to0');

[waits, waitn]=inWait2(z1,waits,waitn,step,1);
[waits, waitn]=inWait2(z2,waits,waitn,step,3);
[waits, waitn]=inWait2(z3,waits,waitn,step,2);

traci.simulationStep();
step=step+1;
end
traci.close();

k=find(waitn(:,2)==3599,1);
waitn(k:end,:)=[];
[R, C]=size(waitn);
av(vh,tr)=sum(waitn(:,2)-waitn(:,1))/R-1
end
end
end
% w1=5*(1:12);
% w2=60*(1:5);
% surf(w1,w2,av)
toc(totaltime)

```



## A.4. Intelligent Traffic Light System with Dynamic Green Timing

```

clc
clear
Street3 = 'ggrrrrrrr';
Street3L = 'rgrrrrrrg';
Street3R = 'grrrrrrgr';
Yellow3 = 'yyrrrrrrr';

Street2 = 'rrrrggrrr';
Street2L = 'rrrrrgrrg';
Street2R = 'rrrrgrrr';
Yellow2 = 'rrrryyrrr';

Street1 = 'rrggrrrrr';
Street1L = 'rrrgrrrrr';
Street1R = 'rrgrrrrrg';
Yellow1 = 'rryyrrrrr';

ALLRED = 'rrrrrrggg';

priority=1;
totaltime=tic();
    x=system('sumo-gui -c
C:\Users\Lenovo\Desktop\GPshortcuts\Project\gp.sumocfg&');
    traci.init();
    step=0;
    traci.trafficlights.setRedYellowGreenState('0', Street1);
    U1=1>0;
    U2=1<0;
    U3=1<0;
    NUM=1;
    p1=1<0;
    p2=1<0;
    p3=1<0;
    M=1;
    R=0;
    T1=10;
    Tp51=300;
    Tp52=300;
    Tp53=300;
    Tp=10;
    T2=5;
    t1=0;
    t2=0;
    tp=0;
    tp51=0;
    tp52=0;
    tp53=0;
    pr=1;
    waits=cell(0);
    waitn=[0 0 0];
    [pr]=incPr(pr,1,p1,p2,p3);
    while step<3600
        x1n(step+1)= traci.edge.getLastStepVehicleNumber('5to0');
        x2n(step+1)= traci.edge.getLastStepVehicleNumber('1to0');
        x3n(step+1)= traci.edge.getLastStepVehicleNumber('2to0');

```

```

p1= length(traci.edge.getLastStepPersonIDs('lto0'))>0;
p2= length(traci.edge.getLastStepPersonIDs('0to5'))>0;
p3= length(traci.edge.getLastStepPersonIDs('5to0'))>0;

if (p1)
    tp51=tp51+1;
end
if (p2)
    tp52=tp52+1;
end
if (p3)
    tp53=tp53+1;
end
Z=(Tp51<=tp51) || (Tp52<=tp52) || (Tp53<=tp53);

x1=x1n(step+1)>0;
x2=x2n(step+1)>0;
x3=x3n(step+1)>0;
Q1=(M==2 && t2>=T2 && Tp51>tp51 && ~x2 && ~x3) || (M==4 &&
t2>=T2 && Tp52>tp52 && x1 && ~x3) || (M==6 && t2>=T2 && Tp53>tp53 && x1));
Q2=(M==4 && t2>=T2 && Tp52>tp52 && ~x1 && ~x3) || (M==6 &&
t2>=T2 && Tp53>tp53 && ~x1 && x2) || (M==2 && t2>=T2 && Tp51>tp51 && x2));
Q3=(M==6 && t2>=T2 && Tp53>tp53 && ~x1 && ~x2) || (M==2 &&
t2>=T2 && Tp51>tp51 && ~x2 && x3) || (M==4 && t2>=T2 && Tp52>tp52 && x3));
U1= (Q1 || (U1 && ~Q2 && ~Q3));
U2= (Q2 || (U2 && ~Q1 && ~Q3));
U3= (Q3 || (U3 && ~Q1 && ~Q2));
if ((U1 && NUM~=1) || (U2 && NUM~=2) || (U3 && NUM~=3))
    if (U1)
        [pr]=incPr(pr,1,p1,p2,p3);
        NUM=1;
    elseif (U2)
        [pr]=incPr(pr,2,p1,p2,p3);
        NUM=2;
    else
        [pr]=incPr(pr,3,p1,p2,p3);
        NUM=3;
    end
end

if(Q1)
    if (x1n(step+1)<=4)
        T1=25;
    elseif (x1n(step+1)<=10)
        T1=30;
    else
        T1=40;
    end
elseif(Q2)
    if (x2n(step+1)<=4)
        T1=25;
    elseif (x2n(step+1)<=10)
        T1=30;
    else
        T1=40;
    end
elseif(Q3)
    if (x3n(step+1)<=4)
        T1=25;

```

```

        elseif (x3n(step+1)<=10)
            T1=30;
        else
            T1=40;
        end
    end
end

    if ((Q1 && ~p2 && ~p3) || (M==11 && tp>=Tp) || (M==12 &&
tp>=Tp) || (R==1 && Tp<=tp && U1))
        M=1;
        t1=t1+1;
        tp=0;
        t2=0;
        R=0;
        traci.trafficlights.setRedYellowGreenState('0',
Street1);

        elseif ((Q1 && p3 && pr==3) || (M==1 && Tp51>tp51 &&
Tp53<=tp53 && Z) )
            tp53=0;
            M=11;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Street1L);

            elseif ((Q1 && p2 && pr==2) || (M==1 && Tp51>tp51 &&
Tp53>tp53 && Tp52<=tp52 && Z))
                tp52=0;
                M=12;
                t1=t1+1;
                tp=tp+1;
                t2=0;
                R=0;
                traci.trafficlights.setRedYellowGreenState('0',
Street1R);

                elseif (M==1 && ~Z && ((x2 || x3) && (t1>=T1 || ~x1))
|| (M==1 && Z && Tp51<=tp51))
                    M=2;
                    t1=0;
                    tp=0;
                    t2=t2+1;
                    R=0;
                    traci.trafficlights.setRedYellowGreenState('0',
Yellow1);

                    elseif ((Q2 && ~p1 && ~p3) || (M==31 && tp>=Tp) || (M==32
&& tp>=Tp) || (R==1 && Tp<=tp && U2))
                        M=3;
                        t1=t1+1;
                        tp=0;
                        t2=0;
                        R=0;
                        traci.trafficlights.setRedYellowGreenState('0',
Street2);

```

```

        elseif ((Q2 && p1 && pr==1) || (M==3 && Tp52>tp52 &&
Tp51<=tp51 && Z) )
            tp51=0;
            M=31;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Street2L);

        elseif ((Q2 && p3 && pr==3) || (M==3 && Tp52>tp52 &&
Tp51>tp51 && Tp53<=tp53 && Z) )
            tp53=0;
            M=32;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Street2R);

        elseif (M==3 && ~Z && ((x1 || x3) && (t1>=T1 || ~x2))
|| (M==3 && Z && Tp52<=tp52) )
            M=4;
            t1=0;
            tp=0;
            t2=t2+1;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Yellow2);

        elseif ((Q3 && ~p1 && ~p2) || (M==51 && tp>=Tp) || (M==52
&& tp>=Tp) || (R==1 && Tp<=tp && U3))
            M=5;
            t1=t1+1;
            tp=0;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Street3);

        elseif ((Q3 && p2 && pr==2) || (M==5 && Tp53>tp53 &&
Tp52<=tp52 && Z) )
            tp52=0;
            M=51;
            t1=t1+1;
            tp=tp+1;
            t2=0;
            R=0;
            traci.trafficlights.setRedYellowGreenState('0',
Street3L);

        elseif ((Q3 && p1 && pr==1) || (M==5 && Tp53>tp53 &&
Tp52>tp52 && Tp51<=tp51 && Z) )
            tp51=0;
            M=52;
            t1=t1+1;
            tp=tp+1;
            t2=0;

```

```

R=0;
traci.trafficlights.setRedYellowGreenState('0',
Street3R);

elseif (M==5 && ~Z && ((x1 || x2) && (t1>=T1 || ~x3))
|| (M==5 && Z && Tp53<=tp53) )
M=6;
t1=0;
tp=0;
t2=t2+1;
R=0;
traci.trafficlights.setRedYellowGreenState('0',
Yellow3);

elseif ((M==2 && T2<=t2 && Tp51<=tp51) || (M==4 && T2<=t2
&& Tp52<=tp52) || (M==6 && T2<=t2 && Tp53<=tp53))
tp51=0;
tp52=0;
tp53=0;
R=1;
t1=0;
tp=0;
t2=0;
M=0;
traci.trafficlights.setRedYellowGreenState('0',
ALLRED);

else
if (M==1 || M==3 || M==5 || M==11 || M==12 ||M==31 ||
M==32 ||M==51 ||M==52)
t1=t1+1;
elseif (M==2 || M==4 || M==6)
t2=t2+1;
else
tp=tp+1;
t1=0;
t2=0;
end
if (M==11 || M==12 ||M==31 || M==32 ||M==51 ||M==52)
tp=tp+1;
end
end

z1=traci.edge.getLastStepVehicleIDs('5to0');
z2=traci.edge.getLastStepVehicleIDs('1to0');
z3=traci.edge.getLastStepVehicleIDs('2to0');

[waits, waitn]=inWait2(z1,waits,waitn,step,1);
[waits, waitn]=inWait2(z2,waits,waitn,step,2);
[waits, waitn]=inWait2(z3,waits,waitn,step,3);

traci.simulationStep();
step=step+1;
end
traci.close();
k=find(waitn(:,2)==3599,1);
waitn(k:end,:)=[];
[R, C]=size(waitn);
av=sum(waitn(:,2)-waitn(:,1))/R-1
toc(totaltime)

```

# Appendix B

---

## Database Connection Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Permissions;
using System.Threading.Tasks;
using System.IO;
using System.Data.SqlClient;
using System.Net;

namespace Database_Connectivity_Toolkit
{
    class Program
    {
        private void OnChanged(object source, FileSystemEventArgs e)
        {
        }
        static void Main(string[] args)
        {
            int count1 = 0;
            int count2 = 0;
            int count3 = 0;
            int count4 = 0;
            int count5 = 0;
            int count6 = 0;
        }
    }
}
```

```

while (true)
{
    SqlConnection myConnection = new SqlConnection("server=LENOVO-PC;" +
        "Trusted_Connection=yes;" +
        "database=RFIDNotificationService;" +
        "connection timeout=30");

    string str1 = "use RFIDNotificationService select count(DISTINCT
    SerialNumber) " +
        "from NotificationLog where AntennaNumber=1";
    string str2 = "use RFIDNotificationService select count(DISTINCT
    SerialNumber) " +
        "from NotificationLog where AntennaNumber=2";
    string str3 = "use RFIDNotificationService select count(DISTINCT
    SerialNumber) " +
        "from NotificationLog where AntennaNumber=4";
    string str4 = "use RFIDNotificationService select count(DISTINCT
    NotificationLog.SerialNumber) from NotificationLog, VehiclesDB " +
        "where NotificationLog.AntennaNumber=1 AND
    NotificationLog.SerialNumber=VehiclesDB.SerialNumber " +
        "AND VehiclesDB.Category>=4";
    string str5 = "use RFIDNotificationService select count(DISTINCT
    NotificationLog.SerialNumber) from NotificationLog, VehiclesDB " +
        "where NotificationLog.AntennaNumber=2 AND
    NotificationLog.SerialNumber=VehiclesDB.SerialNumber " +
        "AND VehiclesDB.Category>=4";
    string str6 = "use RFIDNotificationService select count(DISTINCT
    NotificationLog.SerialNumber) from NotificationLog, VehiclesDB " +
        "where NotificationLog.AntennaNumber=4 AND
    NotificationLog.SerialNumber=VehiclesDB.SerialNumber " +
        "AND VehiclesDB.Category>=4";
    string str7 = "use RFIDNotificationService delete NotificationLog
    where NotificationLogDate<getdate()-0.0000185185185185185";
    string str8 = "select SerialNumber, AntennaNumber into Temp from
    NotificationLog" +
        " EXCEPT select SerialNumber, TrafficLightNumber from
    Tracker_Table" +
        " Insert into Tracker_Table select * , getdate() from temp" +
        " select SerialNumber, AntennaNumber into Temp1 from
    NotificationLog" +
        " EXCEPT select SerialNumber, AntennaNumber from Temp" +
        " select temp1.SerialNumber, temp1.AntennaNumber, getdate() AS
    timeseen" +
        " into temp2 from temp1 left join Tracker_Table ON" +
        " temp1.SerialNumber=Tracker_Table.SerialNumber" +
        " where getdate()>Tracker_Table.TimeSeen+0.00138888888" +
        " Insert into Tracker_Table select * from temp2 Drop table Temp" +
        " Drop table Temp1 drop table temp2";

    SqlCommand myCommand1 = new SqlCommand(str1, myConnection);
    SqlCommand myCommand2 = new SqlCommand(str2, myConnection);
    SqlCommand myCommand3 = new SqlCommand(str3, myConnection);
    SqlCommand myCommand4 = new SqlCommand(str4, myConnection);
    SqlCommand myCommand5 = new SqlCommand(str5, myConnection);
    SqlCommand myCommand6 = new SqlCommand(str6, myConnection);
    SqlCommand myCommand7 = new SqlCommand(str7, myConnection);
    SqlCommand myCommand8 = new SqlCommand(str8, myConnection);
    try
    {
        myConnection.Open(); // ERROR OCCURS HERE
        using (SqlDataReader reader = myCommand1.ExecuteReader())

```

```

    {
        if (reader.Read())
        {
            count1 = reader.GetInt32(0);
        }
    }
    using (SqlDataReader reader = myCommand2.ExecuteReader())
    {
        if (reader.Read())
        {
            count2 = reader.GetInt32(0);
        }
    }
    using (SqlDataReader reader = myCommand3.ExecuteReader())
    {
        if (reader.Read())
        {
            count3 = reader.GetInt32(0);
        }
    }
    using (SqlDataReader reader = myCommand4.ExecuteReader())
    {
        if (reader.Read())
        {
            count4 = reader.GetInt32(0);
        }
    }
    using (SqlDataReader reader = myCommand5.ExecuteReader())
    {
        if (reader.Read())
        {
            count5 = reader.GetInt32(0);
        }
    }
    using (SqlDataReader reader = myCommand6.ExecuteReader())
    {
        if (reader.Read())
        {
            count6 = reader.GetInt32(0);
        }
    }
    myCommand8.ExecuteNonQuery();
    myCommand7.ExecuteNonQuery();
}
catch (System.Exception ex)
{
    Console.WriteLine("Can not open connection ! ");
}
finally
{
    myConnection.Close();
}
string line = count1 + "\t" + count2 + "\t" + count3 + "\t" + count4 +
"\t" + count5 + "\t" + count6;
using (StreamWriter file = new StreamWriter(@"Z:\c\signals.txt"))
{
    file.WriteLine(line);
}
Console.WriteLine(line);
System.Threading.Thread.Sleep(500);
}
}

```



# Appendix C

---

## RFID Installation Guide

### A.1. Requirements

1. Microsoft .Net Framework Version 4.5.1
2. SQL Server 2008 R2
3. ISOStart 2014
4. C++ Redistributable File

## A.2. Installation Guide

1. Install .Net Framework version 4.5.1
2. Install SQL Server 2008 R2
3. Restore Database with name (RFIDNotificationService)
4. Install C++ Redistributable File
5. Install ISOStart And Configure Reader(s) on Buffered Mode
6. Install RFIDBufferReadSetup and Change Configuration in Config File:
  - i. Use USB mode
  - ii. Change reading mode to buffer mode
  - iii. Choose which antenna you are connected on
7. Press “Start” → right click on “Computer” → press “Manage” → expand “Services and Applications” → press “Services” → restart RFIDBufferRead. (make sure you are not connected on ISOStart 2014)
8. Now anything that the reader reads will go directly into the RFIDNotificationService database into the NotificationLog table