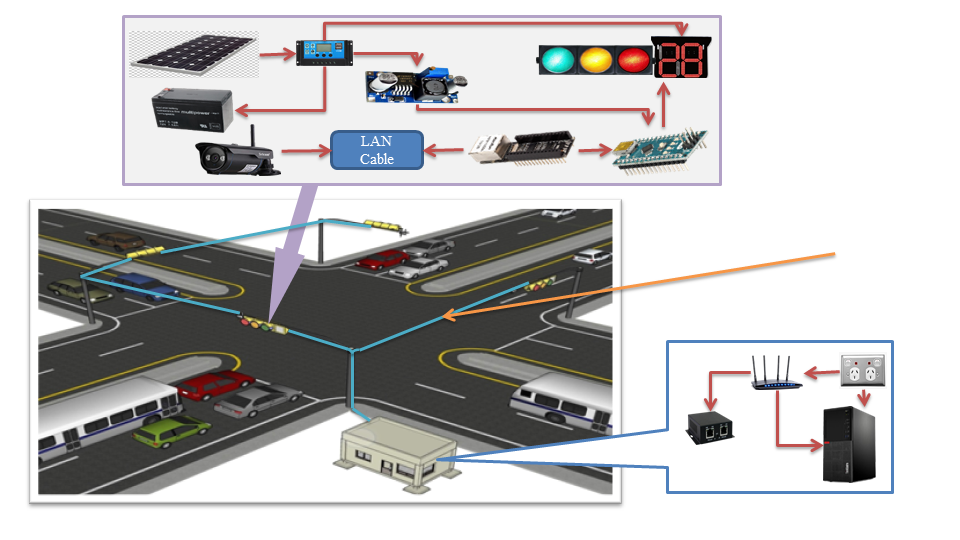
**CHAPTER 4**

**IMPLEMENTATION, SIMULATION AND TEST RUN**

This chapter contains the simulation of object detection, program implementations for UDP server, installing YOLO framework and OpenCV, camera angle calculation the integrating hardware with Arduino and ENC2J580 and seven segments display and test run for this system.

4.1. **Methodology**

The design of the adaptive traffic signal control consists of two main parts; traffic control unit and main control unit. The traffic control unit consists of Arduino Nano, ENC28J60, shift register seven segment display, CCTV IP camera and solar power supply. The main control unit consist computer and router. Traffic Control Unit sends video live stream and receives timing cycle from main control unit. At the main control unit, it process live stream video with YOLO framework and OpenCV to subtract traffic flow and critical lane volume. Then calculate timing cycle for the next time and send to traffic control unit with UDP protocol. Traffic control unit received data from main control unit and running timing cycle and sending video live stream.



Main Control Unit

Ethernet Communication

Traffic Control Unit

Figure 4.1. System Overview of Adaptive Traffic Signal Control System

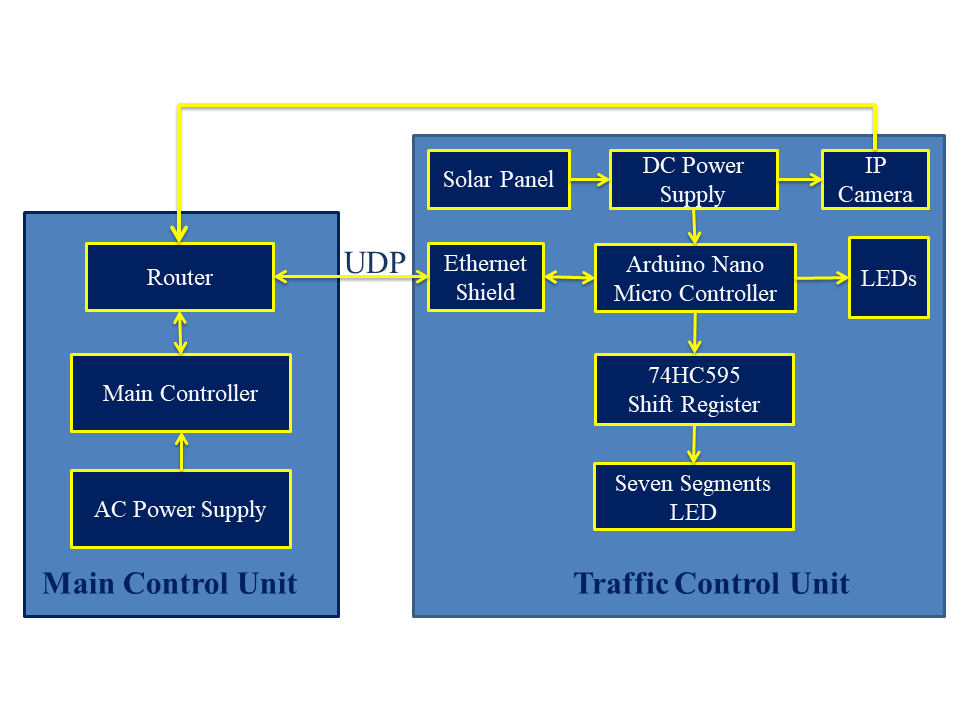


Figure 4.2. Block Diagram of Proposed System

4.2. **Required Library Installing**

In this system operate hardware and software program. Some of the programs are implement with open source python libraries such OpenCV, numpy etc.

4.2.1. Install Darknet

Darknet is an open-source framework that supports Object Detection and Image Classification tasks in the form of Convolutional Neural Networks. Darknet is mainly known for its implementation of the YOLO algorithm (You Only Look Once), which has demonstrated state of the art performance when it comes to real-time object detection. First install Python3, OpenCV and download from https://github.com/pjreddie/darknet. Open command windows and run make. CUDA 10.0 is also required with at least Nvidia GTX 1050. CUDA toolkit is available from https://developer.nvidia.com, download and install according operation system such as Windows, Linux or Mac. Requirements are:

1. Windows or Linux
2. CMake >= 3.8 for modern CUDA support
3. CUDA 10.0
4. OpenCV >= 2.4
5. cuDNN >= 7.0 for CUDA 10.0
6. GPU with CC >= 3.0

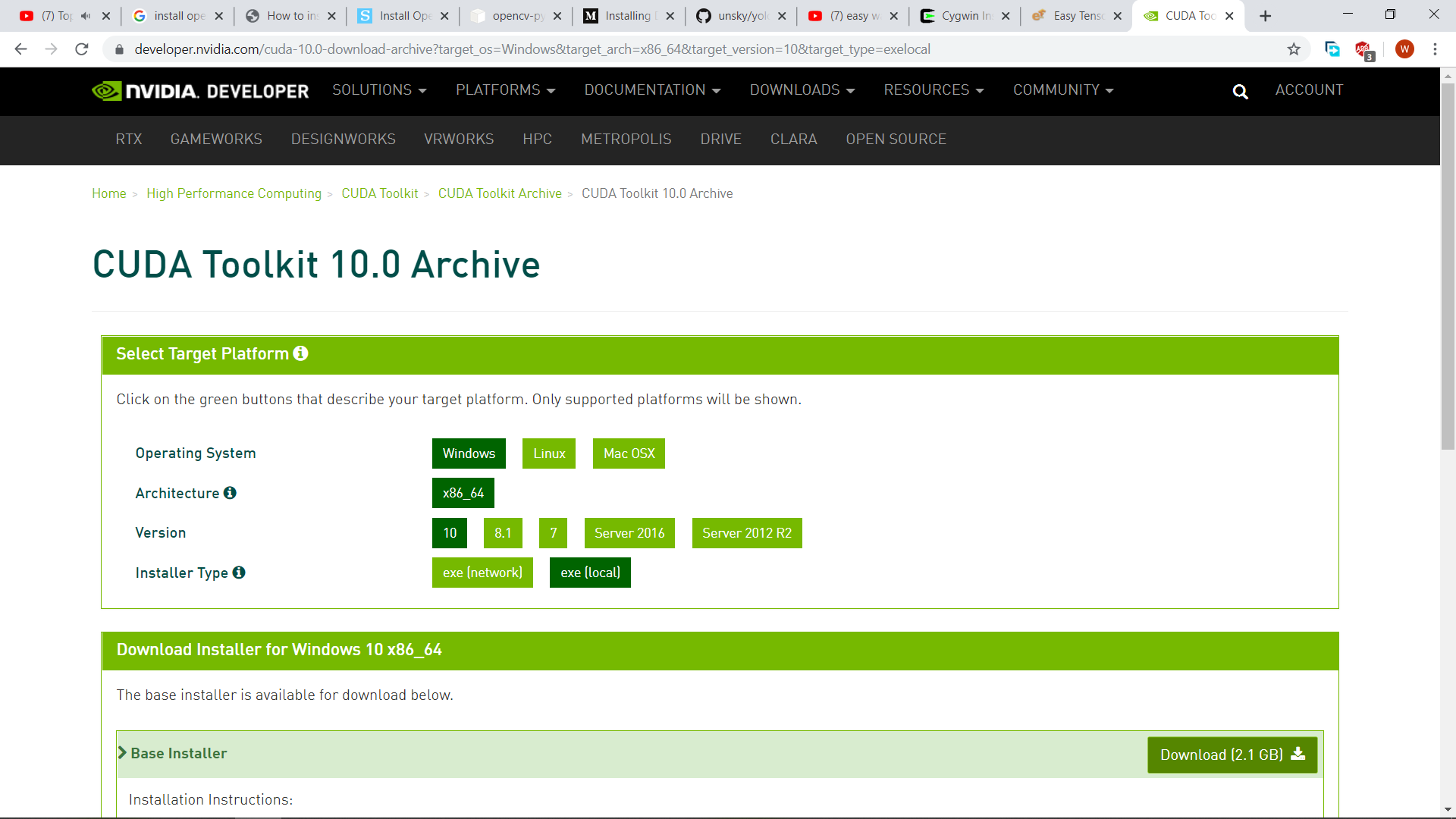


Figure 4.3. Download CUDA form the Nvidia

4.2.2. Training COCO Dataset

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

1. Object segmentation
2. Recognition in context
3. Super pixel stuff segmentation
4. 330K images (>200K labeled)
5. 1.5 million object instances
6. 80 object categories
7. 91 stuff categories
8. 5 captions per image
9. 250,000 people with key points

The first thing for any Computer Vision task is data collection. For using the YOLO object detector, each .jpg file requires a .txt file with the same name. Each line of the text file is of the following format <object-class> <x\_center> <y\_center> <width> <height> , where: <object-class> is an integer corresponding to the class ranging from 0 to num\_classes-1. <x\_center> and <y\_center> are the co-ordinates of the center of the bounding box rectangle.<width> and <height> are float values relative to the the dimensions of the image. (<width> = <absolute\_width> / <image\_width>) .To convert bounding box annotations to the required YOLO format, either [convert2Yolo](https://github.com/ssaru/convert2Yolo) or [Yolo\_mark](https://github.com/AlexeyAB/Yolo_mark" \t "_blank) can be used. Next, navigate to the data directory in the YOLOv3 repository, and store the images of the dataset in an Images folder, and the text file annotations in a labels folder. Now that dataset labels are in the required format, need to create a train-test split. I chose to create a test set containing 10% of the images in the dataset. Run the Python script from the data directory. This will create 2 text files within the data directory: train.txt and valid.txt containing paths to each image in the train and test set respectively.

Dataset is ready to use, we can begin training. Before we start, compile the darknet repository with the make command. To compile with specific options, such as GPU, CUDNN and OPENCV, edit the Makefile following [these instructions](https://github.com/AlexeyAB/darknet#how-to-compile-on-linux). This will create a darknet executable. Now let’s move on to training. First, create file yolo-obj.cfg (in the cfg directory) and copy the contents of the selected models’s config file. Make the following changes in the file.

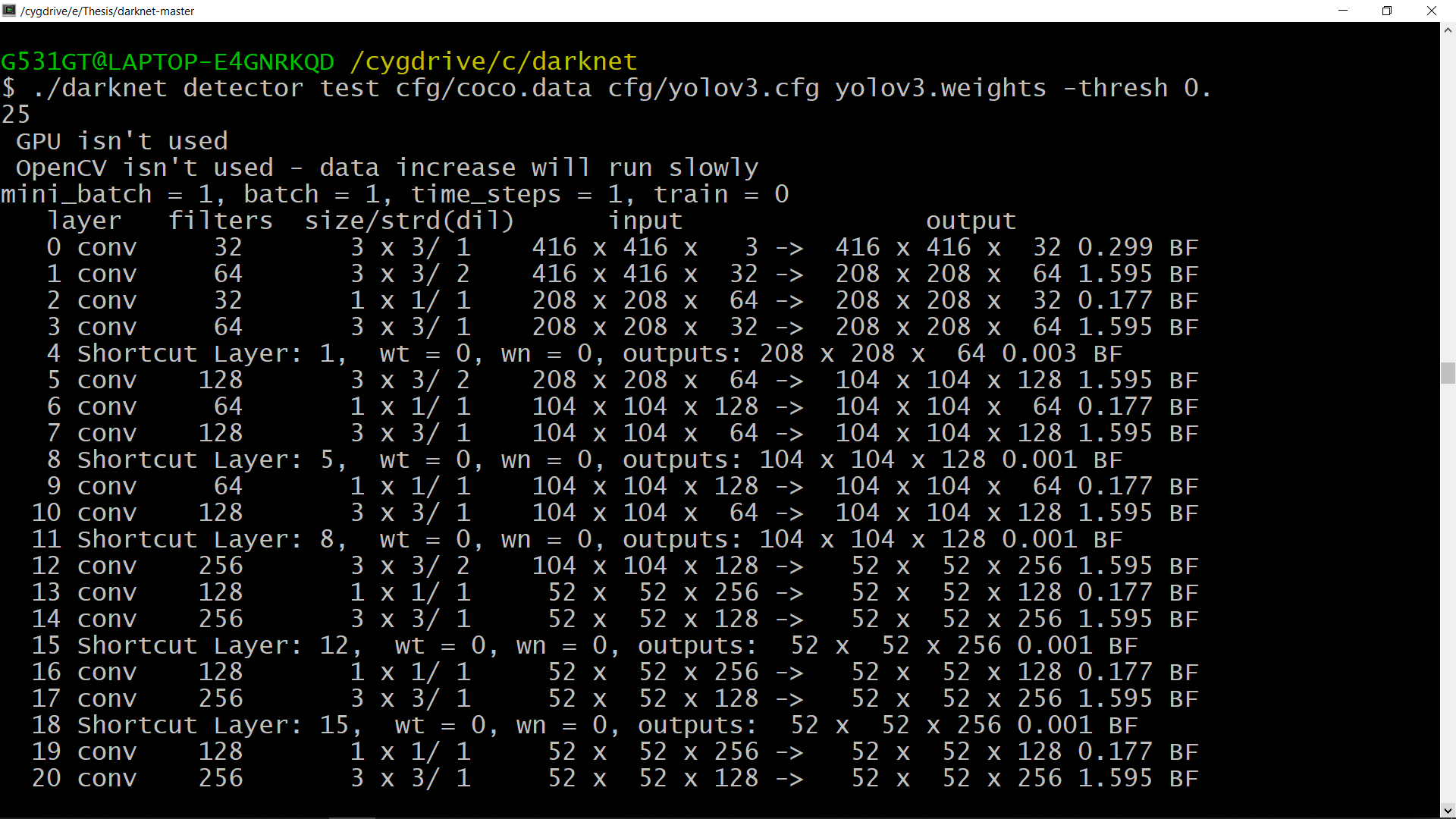


Figure 4.4. Training COCO Dataset with Darknet

4.2.3. Setup MySQL Database

To use Media Server with a MySQL database, you must download and install a MySQL server and ODBC driver, and configure Media Server to connect to the database through the driver. To set up a MySQL Media Server database on Windows. Download and install a MySQL server and MySQL Connector/ODBC (which contains the Unicode driver). During installation, set up a user account with superuser privileges. For instructions, refer to the MySQL documentation on www.mysql.com. Configure the database server for use with Media Server:

1. Open the configuration or options file for the MySQL server (usually named my.ini).
2. So that Media Server can send large amounts of binary data (images) to the database, set the configuration parameter max\_allowed\_packet =1073741824.
3. Save and close the configuration file.
4. Add the MySQL bin directory path to the PATH environmental variable.
5. Open the mysql command line tool:In the Windows Command Prompt, run the command: mysql -u userName –p
6. Enter your password when prompted.
7. Run a CREATE DATABASE command to create a new database.

Run the my.sql script provided in the Media Server installation directory. This script sets up the database schema that Media Server requires.

1. Close the mysql command-line tool: quit
2. In the Windows Command Prompt, run the following command:

mysql -u userName -p -v -D databaseName -e "source path/my.sql"

If security is not a consideration, grant all privileges. Start the mysql command-line tool: mysql, run the GRANT commands, GRANT CREATE TEMPORARY TABLES ON databaseName.\* TO username, GRANT SELECT, INSERT, UPDATE, DELETE ON databaseName.\* TO username, GRANT EXECUTE ON databaseName.\* TO username. And then close the mysql command-line tool: quit.

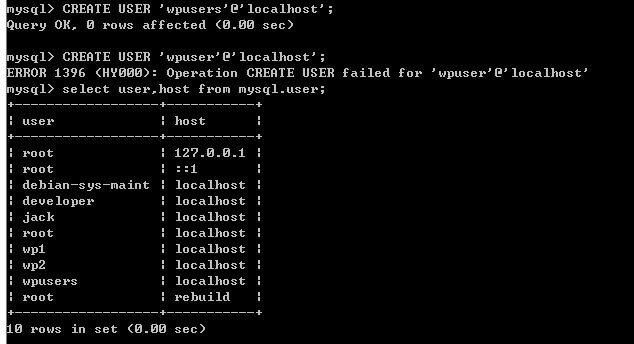


Figure 4.5. MySQL Database Configuration with Command Prompt

4.3. **Program Implementation**

The main control unit and traffic control unit are synchronous with UDP server. UDP server program is implementation with Python programming and run on main control unit. At traffic control unit, UDP client program is also run.

4.3.1. Server Program Implementation

At main control unit, setup for network configuration and IP address is 192.168.0.100. So UDP server program use this IP address for LAN communication. Python network socket library, argparse library and mysql connector library are required. MySQL connector library is connected to MySQL Database and Python programming. In this program, receive data from four clients and check each client send or fail. One of these clients is fail, server is waiting these clients and when all is accept, calculate and sending timing cycle to each client.

4.3.2. Flow chart for Server Program

This is flow chart for server program.

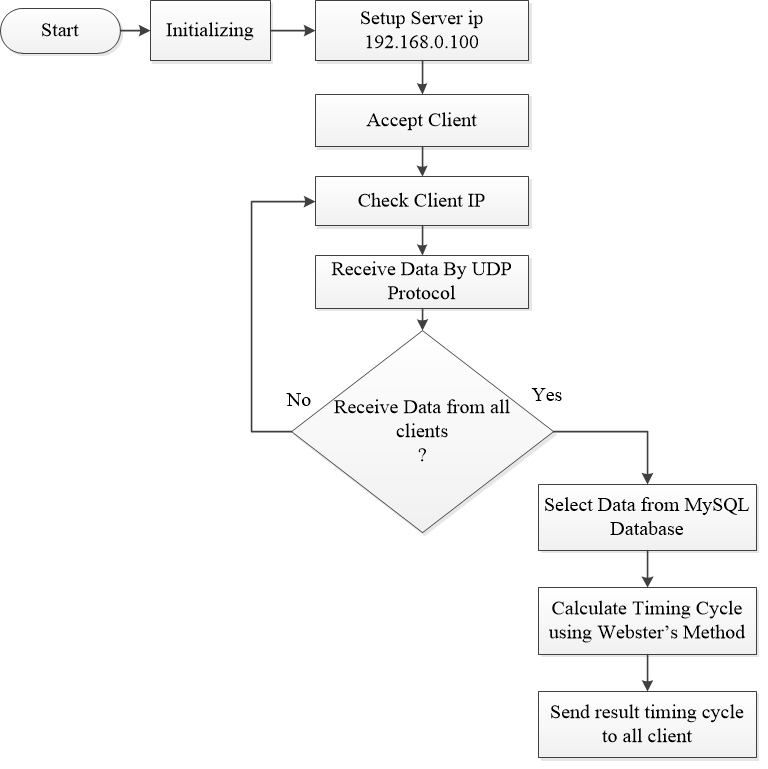


Figure 4.6. Flowchart of Server Program

4.3.3. Objects Detection Program Implementation

To collect traffic data, process live stream video from CCTV IP camera using openCV and YOLO Framework. It use deep neural network to classified and detected object from an image or video. This program is also used Python programming. In this program, the number of traffic flow gets from each lane and this results are store in MySQL Database. Saturation flow is collected by Background subtraction techinque and critical lane volume is collected by YOLO Framework.

4.3.4. Flowchart for Objects Detection Program

The following is flow chart for Objects Detection Program.

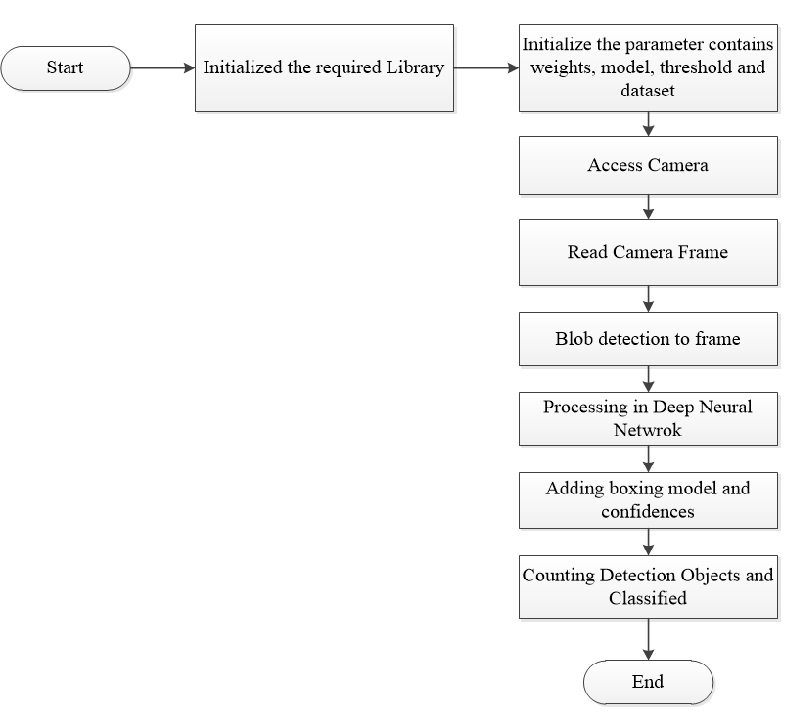


Figure 4.7. Flowchart of Object Detection Program

4.3.5. Traffic Control Unit Program Implementation

Traffic Control Unit Program is used Arduino C programming. In this program use some of library for ENC2J860.. When ENC2J860 received data, read it from SPI communication and split green time value and red time value. And then compare these two value and large value is primary for corresponding phase. The countdown timing and display number at seven segment display. At the end of timing cycle, traffic control unit send data to server for the next timing cycle.

4.3.6. Flowchart for Traffic Control Unit Program

This is the flowchart of Traffic Control Unit Program.

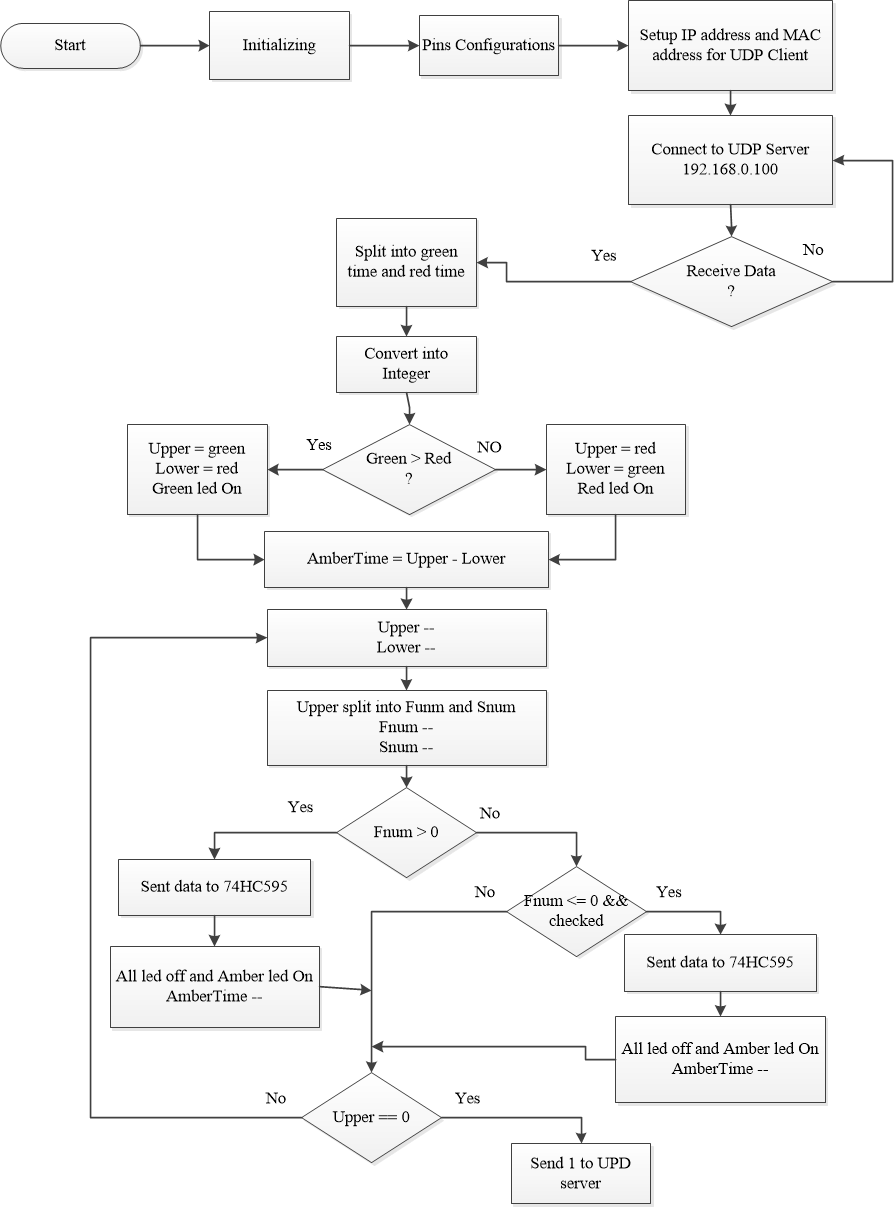


Figure 4.8. Flowchart of Traffic Control Unit Program

**4.4. Simulation Result**

Object detection and Background subtraction are simulated in MATLAB and Python Programming. YOLO is simulated with Python Programming.

4.4.1. Simulation Result with Python

This is background subtraction simulation result with python and openCV.



Figure 4.9. Background image at Zaylay Junction



Figure 4.10. Foreground Image at Zaylay Junction

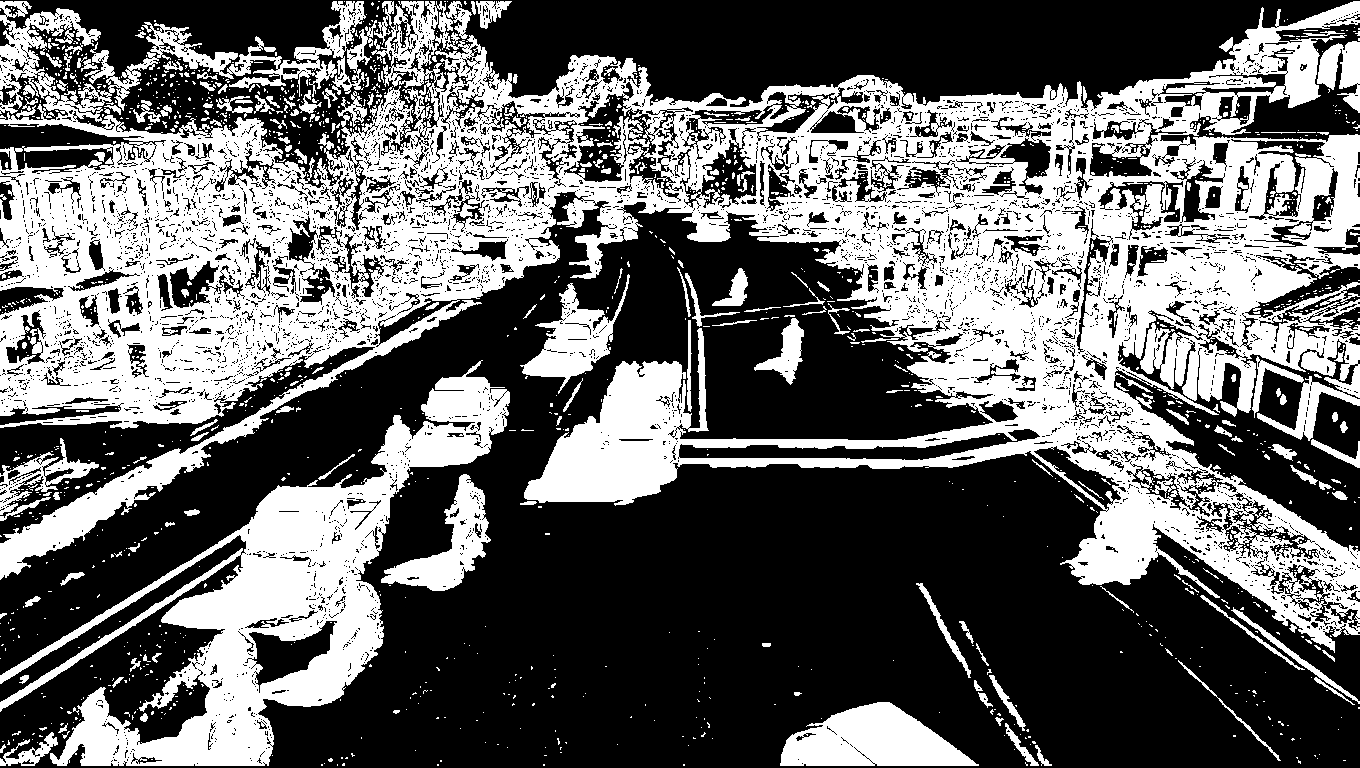


Figure 4.11. Simulation Result of Background Subtraction with Python

In this result, background image content are disable and foreground image content are available. Some of white regions on middle is moving objects and other white region at each side is environmental noise such motion of leaves by winds and camera motion. So background image and foreground image different is much required more. But this can easily detect vehicles on lane.

The following result is car detection with YOLO and openCV.

Figure 4.12. Input Image for YOLO Detection

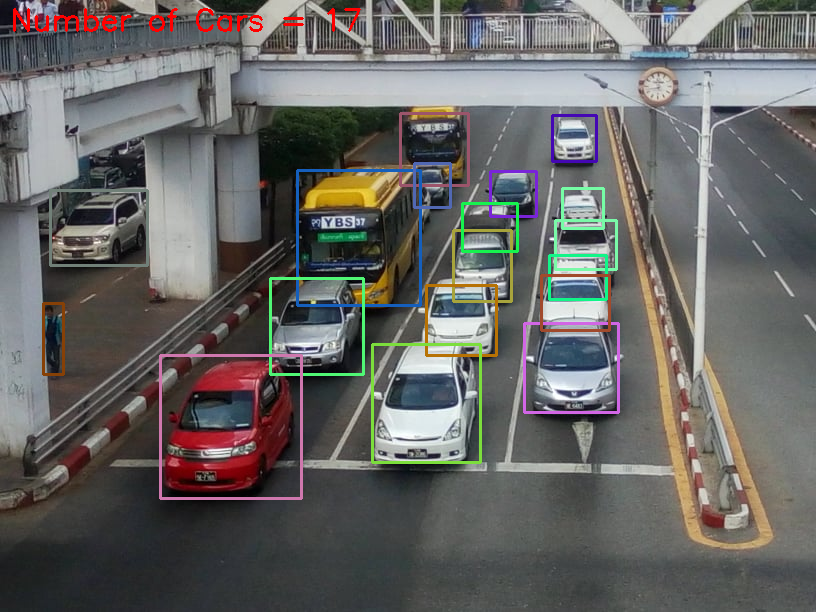
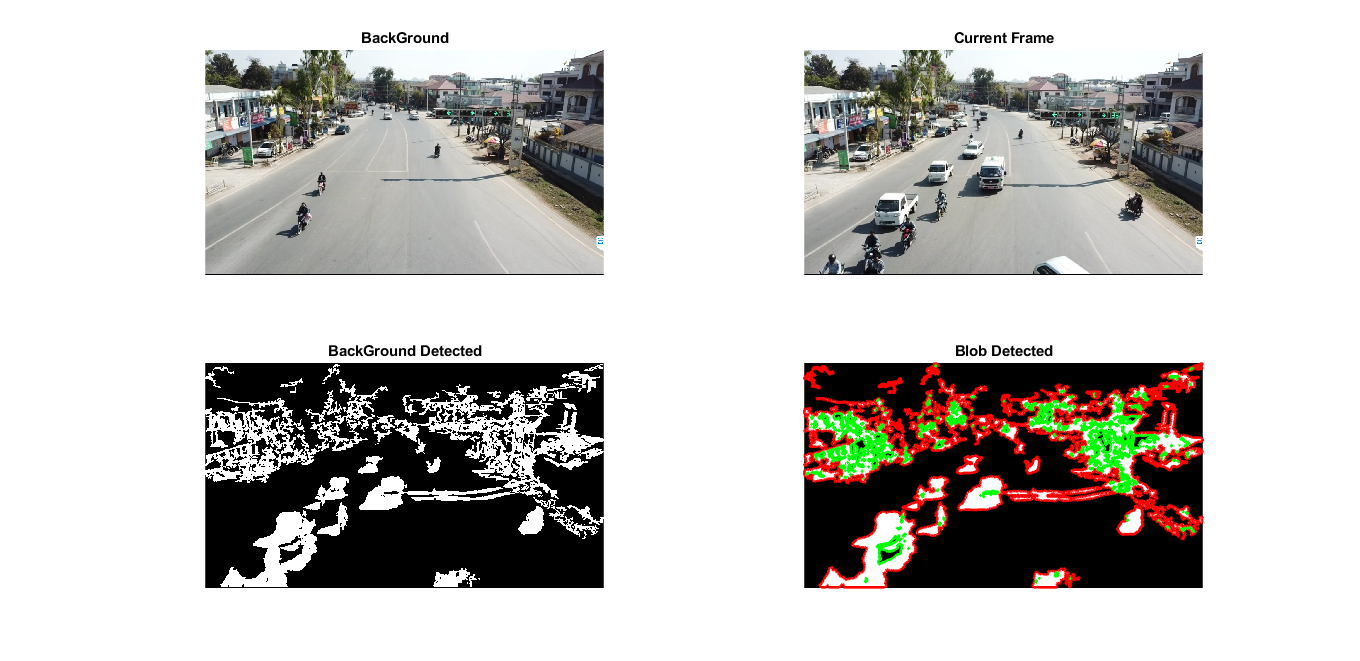


Figure 4.13. Output Result Image of YOLO Detection

In this result, YOLO can detect various of vehicles and counting the number of vehicles.

4.4.2. Simulation Result with MATLAB

This is MATLAB simulation for background subtraction technique.

Figure 4.12. Background Image at Zaylay Junction

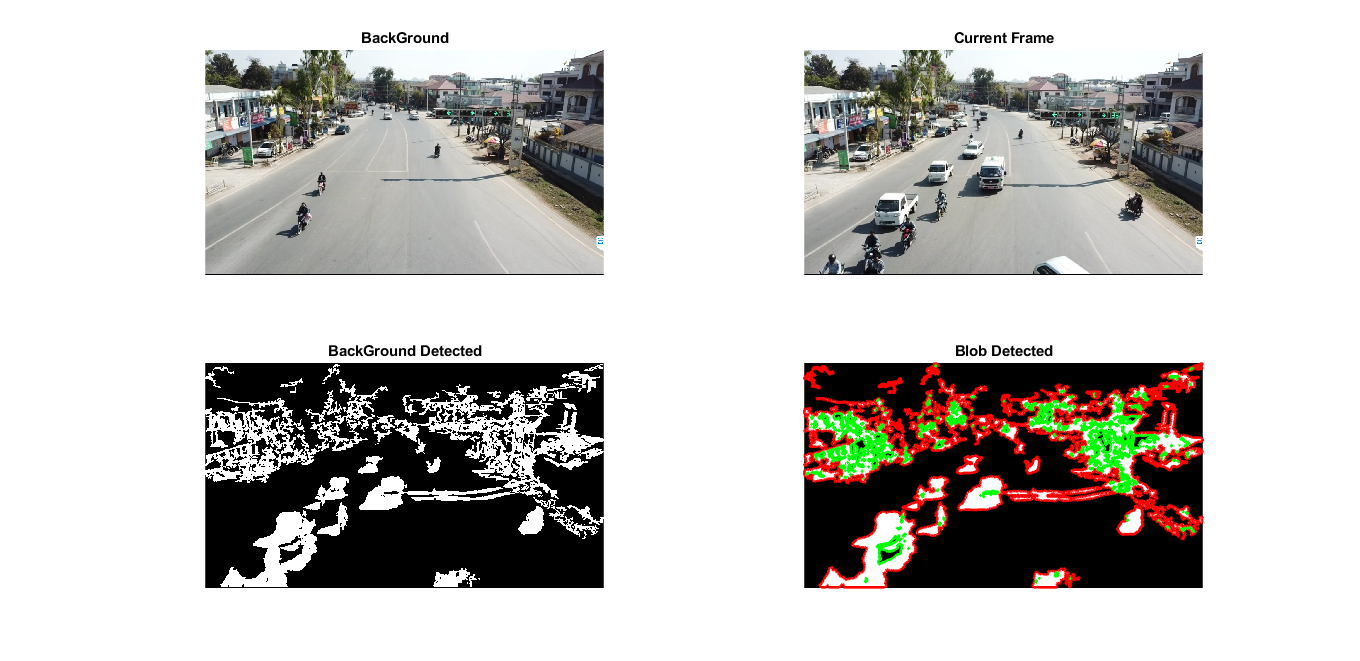


Figure 4.13. Foreground Image at Zaylay Junction

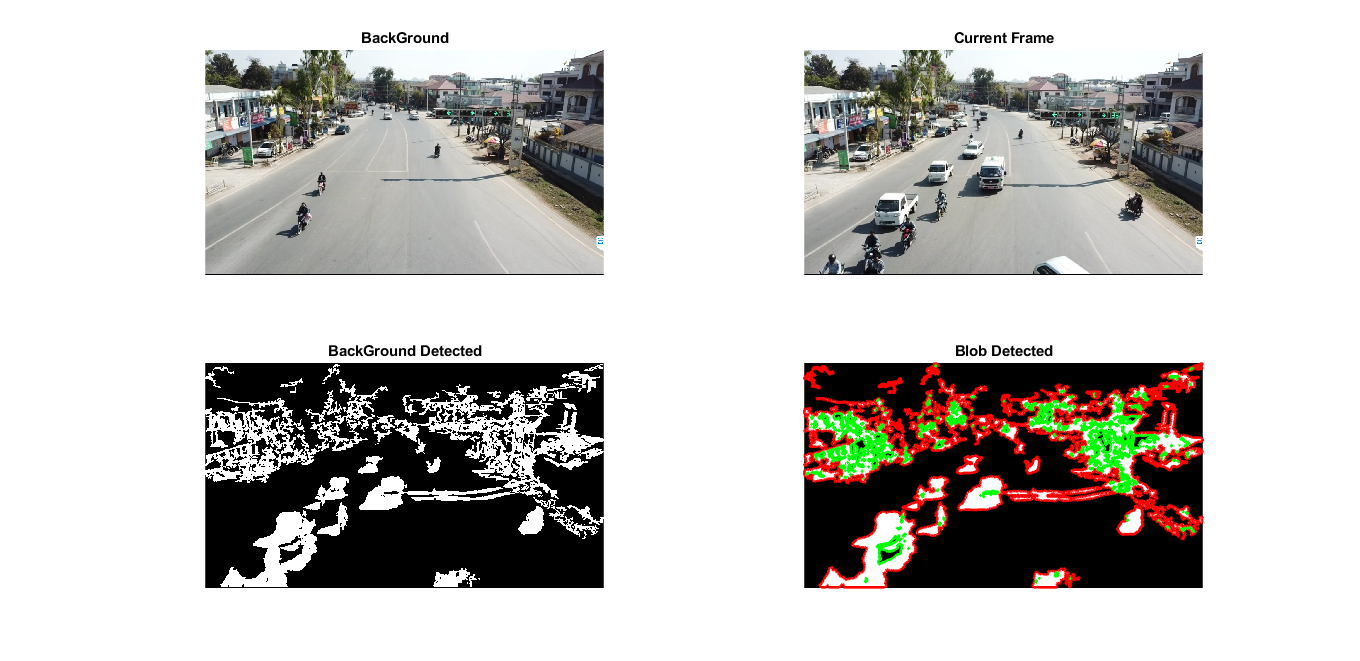


Figure 4.13. Simulation Result with MATLAB

**4.5. Camera Angle Calculation**

Video image processor cameras can be deployed to view upstream or downstream traffic. The primary advantage of upstream viewing is that incidents are not blocked by the resultant traffic queues as described in Table 4.1. However, tall vehicles such as trucks may block the line of sight, and headlights may cause blooming of the imagery at night. With upstream viewing, headlight beams can be detected as vehicles in adjacent lanes on curved road sections. Downstream viewing conceals cameras mounted on overpasses so that driver behavior is not altered. Downstream viewing also makes vehicle identification easier at night through the information available in the taillights and enhances track initiation because vehicles are first detected when close to the camera.

Table 4.1. Different between Upstream and Downstream Camera View.

|  |  |
| --- | --- |
| Upstream Viewing | Downstream Viewing |
| Headlight blooming, glare on wet pavement, headlight beams detected in adjacent lanes on curved road sections. | Camera on overpass concealed from drivers. |
| More block age from tall trucks. | More information from taillights available for braking indication, vehicle classification, turning movement identification, and tracking. |
| Traffic incidents not blocked by resulting traffic queues. | Easier to acquire vehicles that are closer to the camera for tracking algorithm implementation. |

Although some manufacturers quote a maximum surveillance range for a VIP of ten times the camera mounting height, conservative design procedures limit the range to smaller distances because of factors such as road configuration, congestion level, vehicle mix, and inclement weather. The impact of reduced headway on the effective surveillance range is calculated from the distance *d* (along the roadway from the base of the camera mounting structure to the vehicles in question) at which the VIP can distinguish between two closely spaced vehicles. The distance *d* depends explicitly on camera mounting height, vehicle separation or gap, and vehicle height.



Figure 4.14 Camera Field of Views at Traffic Junction

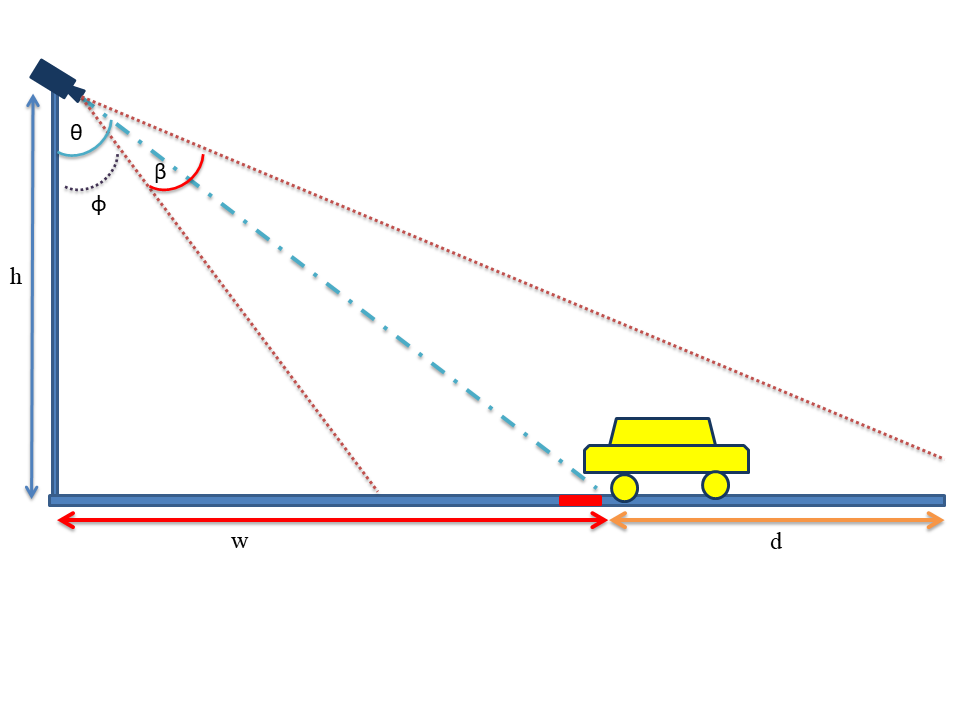


Figure 4.15 Camera Angle Calculation for Field of Views

The angle of camera is calculation with following.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**4.6. Implementation of System and Test Run**

The purposed system consists of two units. Each unit is implemented with respective components and test run.

4.6.1. Implementation of Traffic Control Unit

Traffic Control Unit contains Arduino Nano, ENC2J860 Ethernet Module, 74HC595 shift register, leds, seven segments display and solar power supply. ENC2J860 connects to Arduino Nano at digital Pin 10,11, 12, 13. 74HC595 shift register pin 12 is connected to digital pin 2, data pin ( 14) is pin 3 and clock pin (11) is attached to digital pin 4 of Arduino Nano. To control color of seven segments display, digital pin 8 and 9 are used. Traffic Signal Light are connect to digital pin 5,6 and 7.

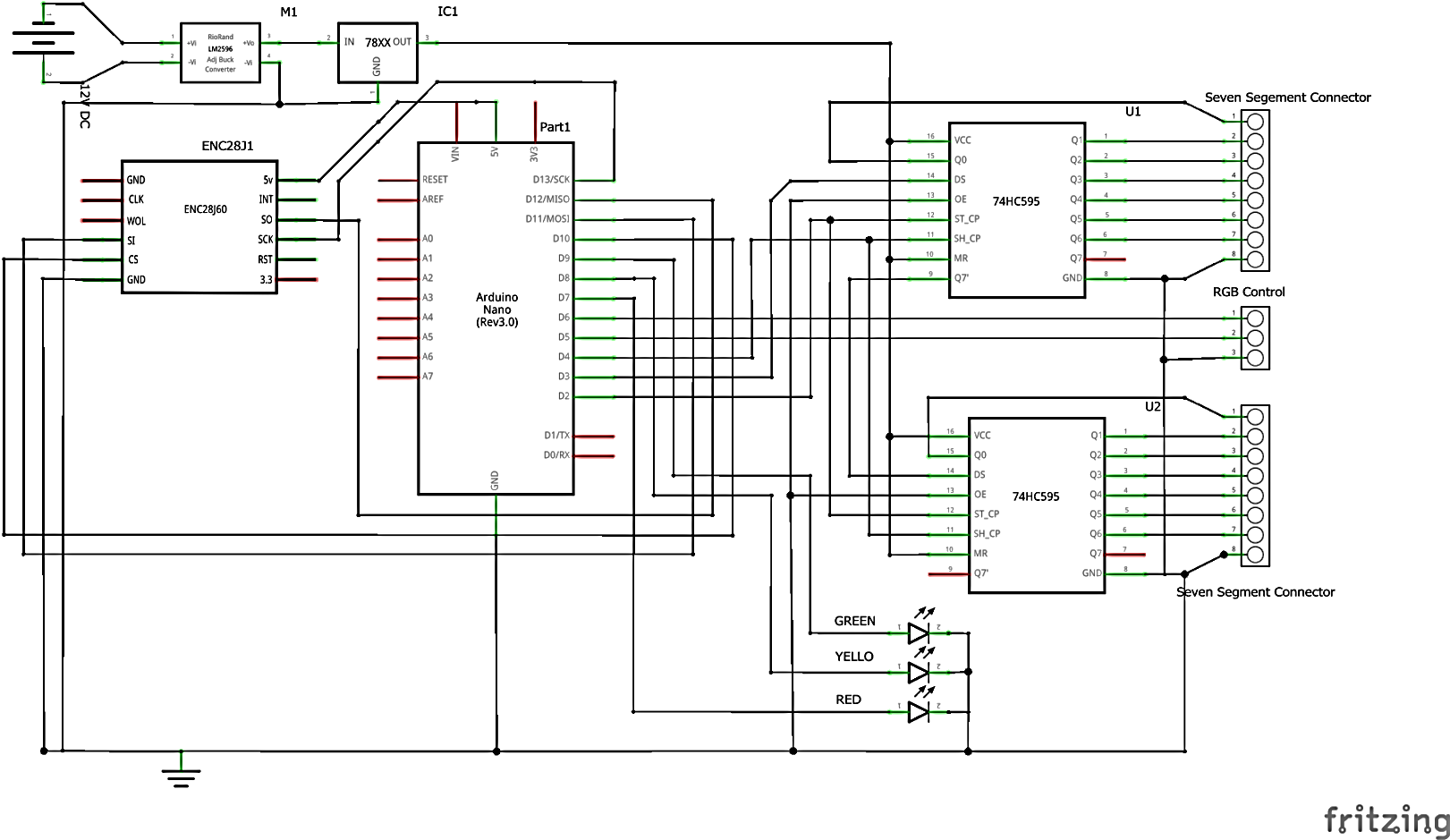


Figure 4.16. Circuit Diagram of Traffic Control Unit

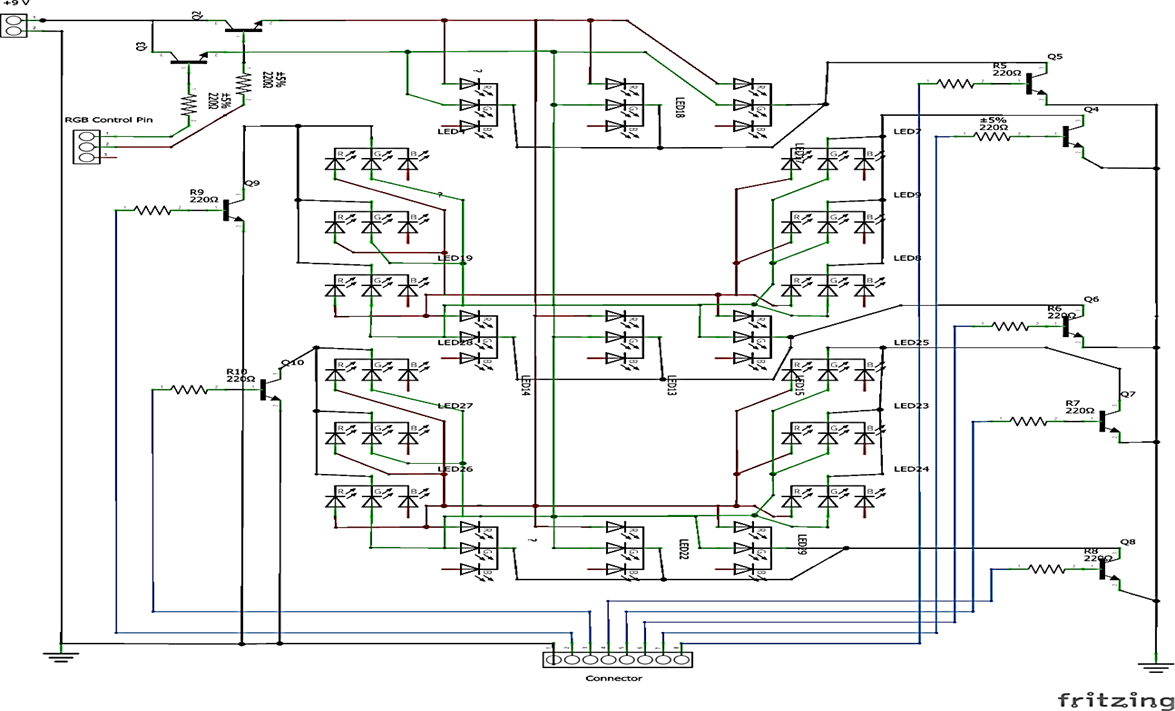


Figure 4.17. Circuit Diagram of Tri Color Seven Segments Display

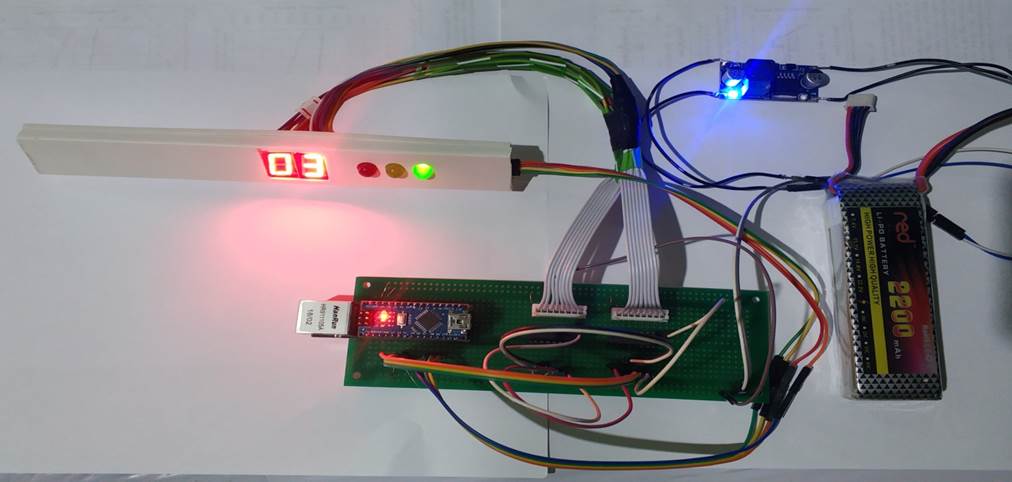


Figure 4.18. Implementation of Traffic Control Unit for One Lane

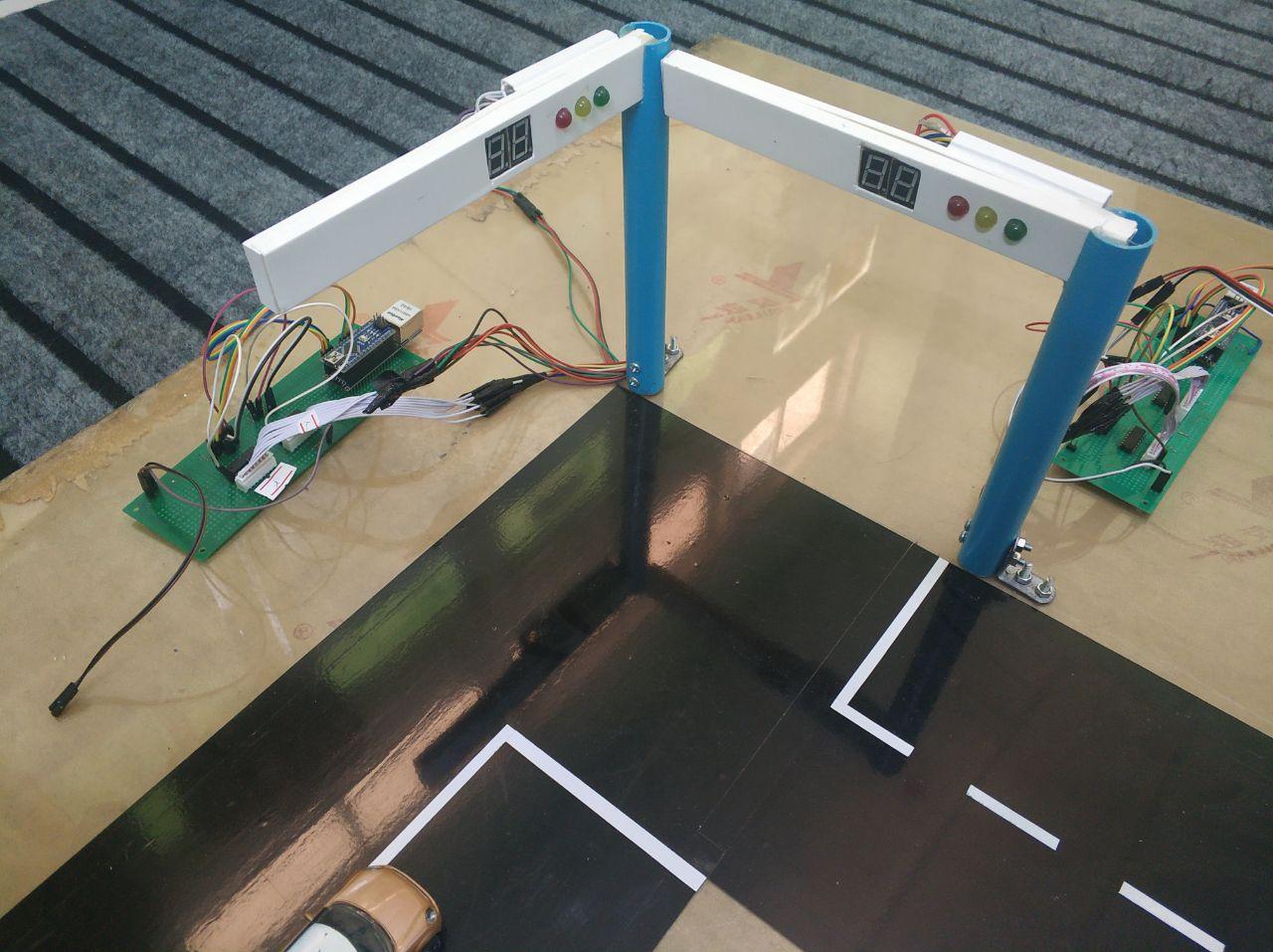


Figure 4.19. Implementation of Traffic Control Unit for Two Lane

4.6.2. Implementation of Main Control Unit

In the Main Control unit, change configuration and connection with router. At router, enable DHCP server and limit subnet mask for required clients. The router IP address is 192.168.0.1 and default gate is 192.168.0.1. Computer is main server and IP address is 192.168.0.100.

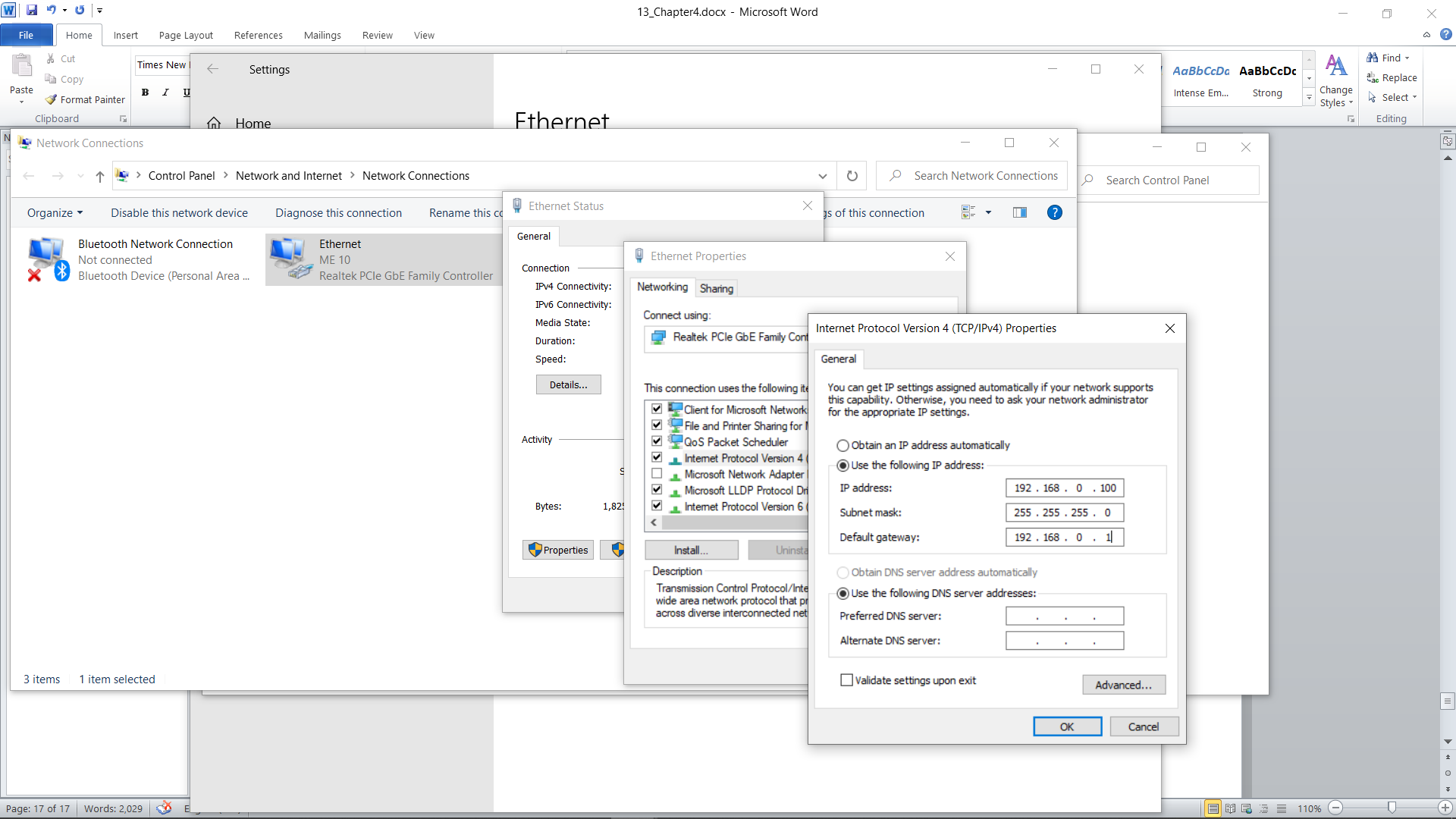


Figure 4.20. Network Configuration for Computer

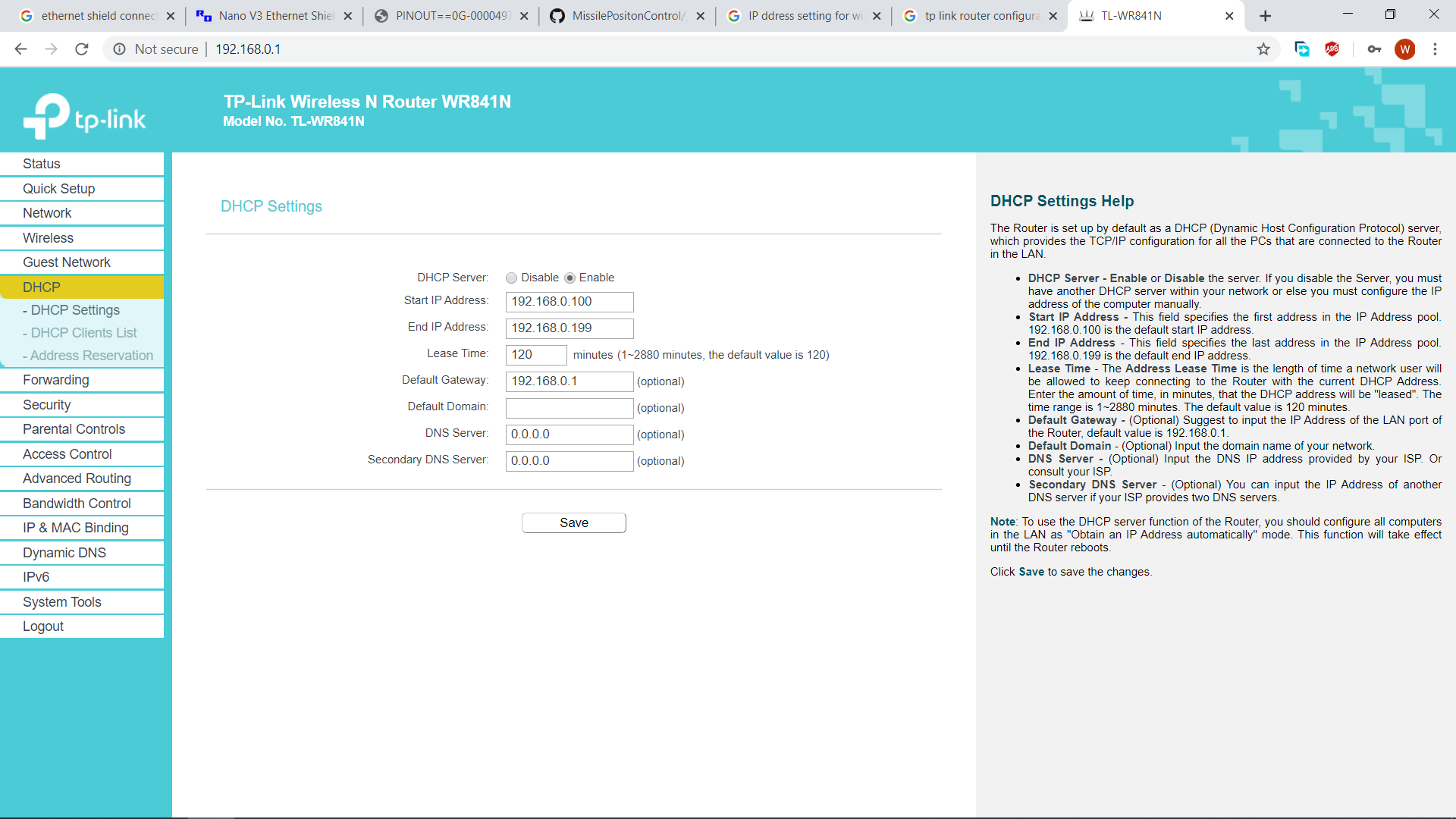


Figure 4.21. DHCP Configuration of Router

4.6.3. Testing System

All of units are connecting to respective components. Then the purposed system is testing.