7
CROSS-JUNCTION TRAFFIC CONTROL SYSTEM VIA SENSOR FEEDBACK

Ahmad Shafeeq August Fouzy, Sophan Wahyudi Nawawi, Muhammad Ruzairi Ramli, Leong Yong Liang, and Noel Ng Fong Nian, Khairul Hamimah Abas, Mohd. Fadzli Abdul Shaib

7.1 INTRODUCTION

There are two types of junctions at the highways. One of the designs is two main roads are grade-separated and is called an interchange. All designs are to ensure continuous traffic flow. Examples of the two most common types of highway interchanges are shown in Figure 7.1.

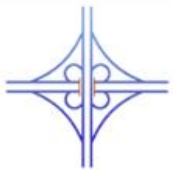
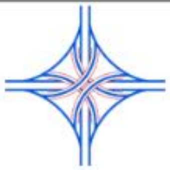
Type	Diagram	Example	Description
Cloverleaf			2 levels. Can be found in most Malaysian expressway crossings. Easier to construct but occupies larger area.
Stack			4 levels. Expensive to construct but occupies less area. One of the most efficient interchange.

Figure 7.1: Two type of cross-junction

An at-grade junction is called an intersection. Cross intersections with traffic light control are found everywhere within an urban area where vehicles are abundant. It's can also be found at the exit of a highway, where it connects to a local road. The cross intersection commonly being constructed below or above a highway depending on the type of land or terrain. The most common type of such cross intersection in Malaysia are illustrated in Figure 7.2.

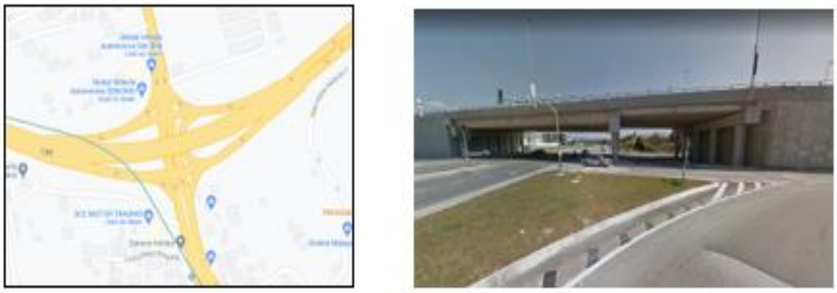


Figure 7.2: The graphical and actual picture of cross intersection in Malaysia

The growth of modern technology in Malaysia causes an increase in the number of vehicles, and a lack of effective traffic management would lead to huge economic losses such as energy consumption, greenhouse gas emissions, and time waste. Traffic congestion has become one of the most significant impediments to city economic development. It can be resulting in high fuel consumption, increased commute costs, and pollution of the environment especially in the city [1]. As a result, we require an efficient traffic light system to meet rising demand, reduce vehicle user time spent at traffic lights, and reduce carbon emissions from vehicles that must stop and then resume their journey. There are 3 main traffic light control system methods:

(a) Timer – where the light cycles are pre-programmed and always loop with the same pre-set time.
(b) Inductor loop sensor – where metal (vehicles) sensors are being installed under the tar road to detect the presence of cars/motorcycles on top. More loops can be installed some distance apart with each other on the same road to determine the queue length, and microcontrollers can decide an optimum traffic light wait time based on the sensor data.
(c) Machine vision – cameras are installed on the cross intersection to detect the number of vehicles in queue using object detection

algorithms. With these precise constant data feedback, the traffic light system can operate more efficiently under some pre-set parameters.

7.2 RELATED WORKS

A short review of the challenges and the state of the art for Cross-Junction Traffic Control System via Sensor Feedback can be found in section 7.2.1.

7.2.1 Related Works

Traffic control and monitoring have become an attention today due to various purposes that can be used and one of them is by using video sensors and also lead to significant advances in computer vision fields. Video processing has been used in commercial and research systems. An example of the system is Reading People Tracker (RPT) that can use multiple cameras that were later used in the development of the system called AVITRACK that can monitor airplane servicing operations. Another project is FP5 INTERVUSE which is an artificial vision network-based system that can monitor the ground traffic at airports. The system uses the Autoscope® Solo Wide Area Video Vehicle Detection System which has been successfully deployed worldwide for monitoring and controlling road traffic [2].

Wireless Sensor Networks has met this proposal interest where it has been proved to solve issues of traffic due to its flexibility and work well for application domains that require low power, cost, and maintenance. By using sensors, traffic jams can be solved in the junction which also can reduce accidents among residents focused in city areas [3].

IoT has been used with the system as needed as today's technologies [4][5]. Deployment of wireless sensor networks on the road, on traffic lights, and in specific places, for example, hospitals to monitor the traffic to avoid traffic jams by providing the shortest route to the users. There is also a system that can measure the density of the road congestion created at road crossings. Which gives a priority on a road that consists of more cars than other roads that have fewer. Next, there is the system that detects vehicles using the RSSI signal that is installed along the road that will receive the radio frequency emitted by Bluetooth beacons on the other side of the road that can detect the type of vehicles on the road, such as trucks or cars [4].

In a congested intersection, traffic signals are the most practical form of traffic control. If one lane has more traffic than the others, current traffic signals are ineffective at controlling traffic. Intelligent or "Smart Traffic Control" is a system that can measure the vehicle density in a lane at a 4-way intersection and then determine the priority using the software [6]. As a result, traffic issues have become more prevalent in recent years, and current traffic light supervisors have drawbacks because they depend on predefined systems that do not allow for real-time adjustment. The delay is longer due to the constant periods of green, orange, and red signals. A proposed technology known as the "Smart Traffic Control System" has been developed to bring traffic light control more effective [7]. The red and green light timings will be intelligently determined based on traffic on nearby roads.

7.3 METHODOLOGY

7.3.1 System Architecture

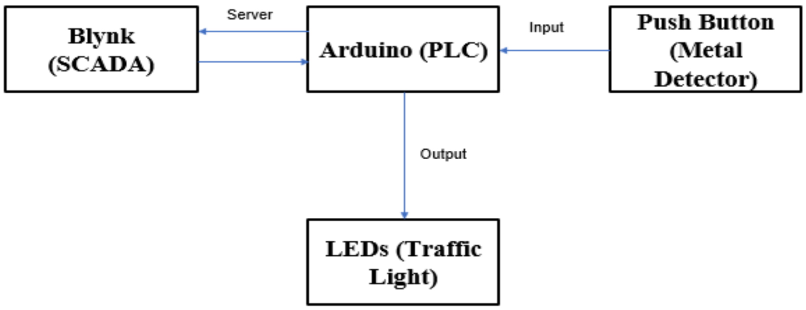


Figure 7.3: The flow of the system architecture

A system architecture is a conceptual model that defines the structure, behaviors, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system[8][9]. For this project, our system architecture is shown in Figure 7.3.

For the input of the system, the push button was used as a metal detector. The metal detector is a sensor that detects the presence of a car on that particular road. Then, the signal was sent to the Arduino to identify the number of car/s is that are detected by sensor 1 only or detected by sensor 1 and sensor 2. After that, the LED (traffic light) will react to the signal as an output of the system. In the meantime, we can monitor and control the traffic light using Blynk (SCADA)[5][10].

7.3.2 System Architecture

Illustration is a tool of communication that helps to simplify complex ideas into clear messages; links concepts, captures feelings, and shapes the narrative in order to deliver effective messages.

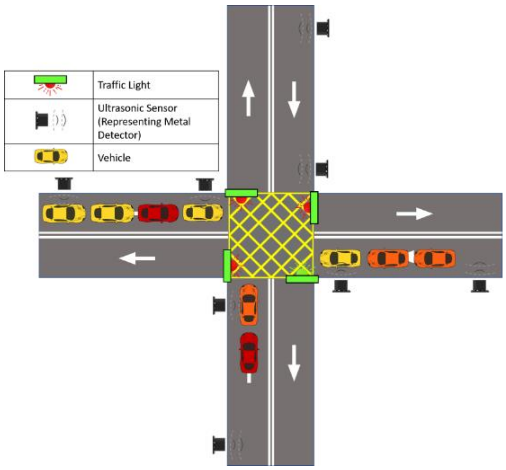


Figure 7.4: The system illustration for the cross-junction traffic light

The system architecture for the cross-junction traffic light that has two sensors of each junction as shown in Figure 7.4. In a real scenario, traffic police officers need to be at traffic lights to ensure smooth traffic during peak hours. In this case, traffic police officers have to precede routes that have a lot of cars. This is the same function as using sensors at each intersection to give a long time for the green traffic light on a congested road.

7.3.3 PLC using Arduino Uno

Hardware used for this project consists of 8 inputs which are switches that act like a sensor on each junction of a traffic light junction are illustrated in Figure 7.5. There are 8 outputs of LEDs consist of 4 green and 4 red lights following traffic light color. 4 resistors of 220Ω for LEDs and 330Ω for switches to control enough current through the devices to avoid overcurrent.

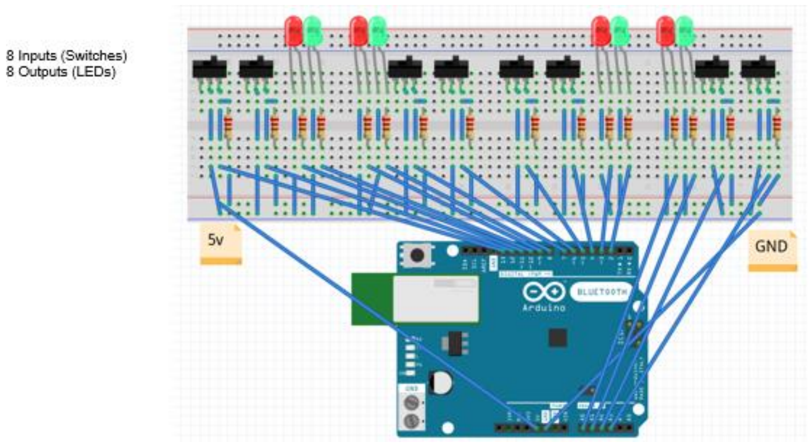


Figure 7.5: The hardware arrangement/schematic

The representation of PLC part of this project is an Arduino Uno used to compile a sequence of coding write up using Arduino IDE software. Lastly, a jumper wire and a breadboard are used to connect all the elements. Based on the traffic light system using an Arduino code in Appendix D until Appendix G, the traffic light's first initial condition is set to all junctions so that the red light will be started when there is no car at the junction. It gives initial high output at the pin assigned to red LEDs. After detecting the first car in the junction, the first push-button is activated and the coding reacts by changing the LED from red light to green light for 0.5s to let the car move. If both sensors detect a car means there will be more than one car in the junction. Here the coding will react and change the red light to green light but this time will be longer than 0.5s.

7.3.4 SCADA using Blynk

The Scada part of this project is using a Blynk app to control an Arduino used to control the traffic light. Based on the Figure 7.6, the digital pin is used the same as in the Arduino output pin to control the LED light only using the apps. Real-time graphs are also implemented to monitor the traffic light. It will be used to collect data on the time taken for a traffic jam to be settled.

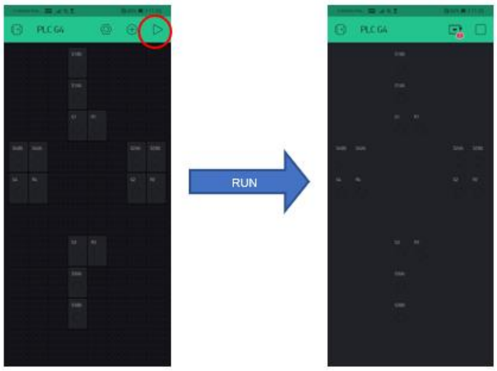
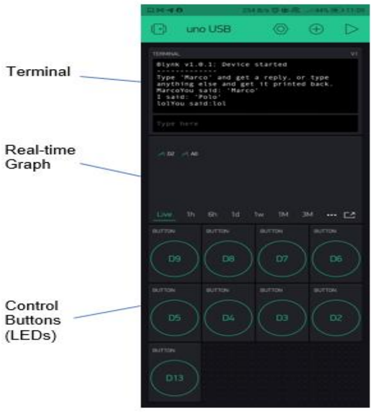


Figure 7.5: The view of the apps SCADA using Blynk



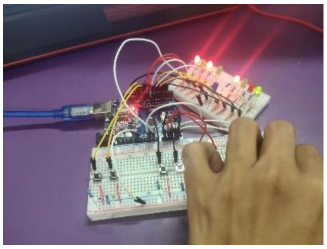
```
Blynk v1.0.1: Device started
-------------
Type 'Marco' and get a reply, or type
anything else and get it printed back.
MarcoYou said: 'Marco'
I said: 'Polo'
lolYou said:lol
```

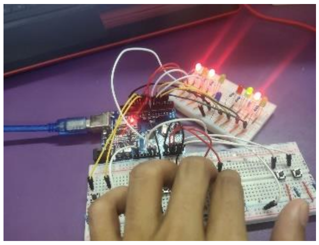
Figure 7.6: The view of the Blynk apps used in the project

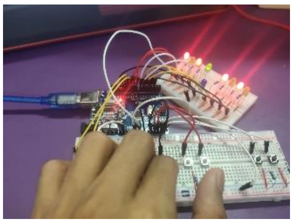
Referring to Appendix A, Appendix B, and Appendix C, the coding used to connect the Blynk application to Arduino is described. It starts by defining the Blynk to the code and putting the correct Authentication Token from the apps that act as an address to differentiate this project from another[5]. Continuing with pin coding from the application, output feedback is implemented to make sure the connection is correct and accurate. The "Marco" input with a response "Polo" with the same words output is proof that the connection is working.

7.4 RESULTS AND DISCUSSION

The result in Figure 7.7 shows that the output LED works exactly with the execution code in Arduino, all green lights will light after the 1st button is pushed and will light longer after both buttons are pushed.


(a)


(b)


(c)

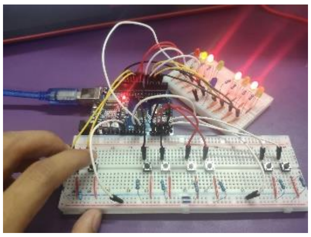
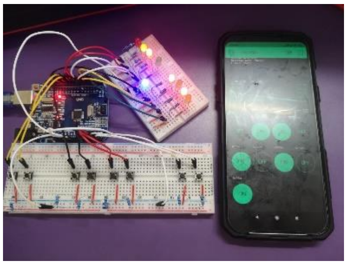
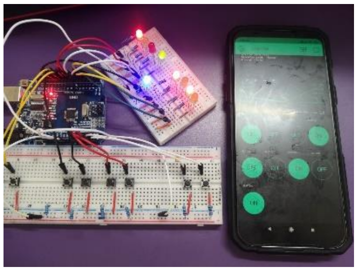
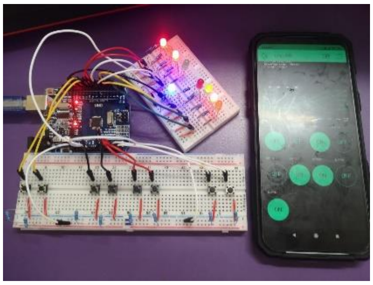

(d)

Figure 7.7: Green light after pushing a push button or sensors detect a car (a) 1st Junction (b) 2nd Junction (c) 3rd Junction (d) 4th Junction.


(a)


(b)


(c)

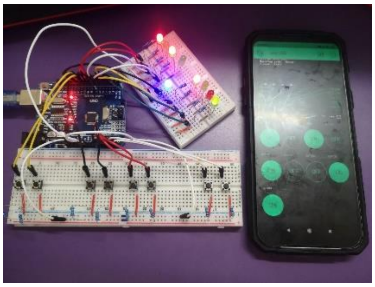

(d)

Figure 7.8: Green light after pushing a assign pin in the Blynk apps or controlling the traffic light using Blynk apps (a) 1st Junction (b) 2nd Junction (c) 3rd Junction (d) 4th Junction

While the result in Figure 7.8 shows that the assigned pin in the Blynk application is correct and can control the LEDs light only using the apps and can also be controlled at such a distance for about multiple meters without any obstacle that blocks the wave (Figure 7.9-Figure 7.10).

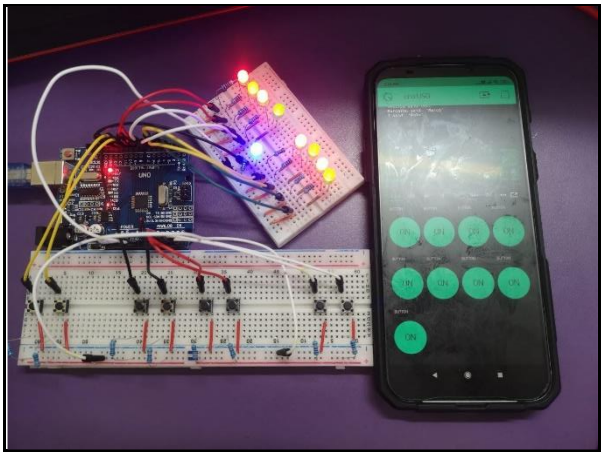


Figure 7.9: All light is light up to show the apps can control every single LEDs

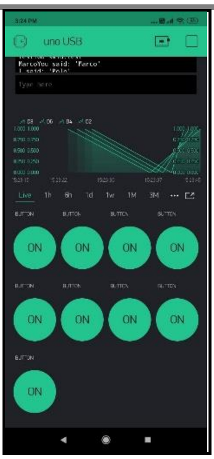

```
MarcoYou said: 'Marco'
I said: 'Polo'
```

Figure 7.10: Live data streams can be seen to be used to collect data for documentation purposes

Last but not least, all coding and hardware successfully perform together to get the result. In reality, a traffic light consists of 3 colors which are red, yellow, and green, but for this project due to less amount of pins that can be assigned to be output from the Arduino, the yellow light needs to be left out, and priority given to only red and green light. The limited budget also affects the project, for example, the Blynk apps can only use for data streams only up to 4 pins which is in this case only considering green light output, many features can be added for the future works. The project works successfully and can be implemented in real life with a more powerful PLC and Scada to be used that will benefit the consumer, can reduce traffic jams, reduce emissions of carbon and help to maintain the sustainability of the earth.

7.5 CONCLUSION

This chapter presents the cross-junction traffic control system with sensor feedback for the traffic control with surveillance device and monitoring. Overall, the results of the study show that this technology gives users a more comfortable experience and satisfaction while at traffic lights. The system can minimize the amount of time spent at traffic lights while also reducing carbon emissions.

REFERENCES

[1] W.H. Lee and C.Y. Chiu, “Design and Implementation of a smart traffic signal control system for smart city applications,” *Sensors (Switzerland)*, 20(2) ,2020, doi: 10.3390/s20020508

[2] A. Koutsia, T. Semertzidis, K. Dimitropoulos, N. Grammalidis and K. Georgouleas, "Intelligent traffic monitoring and surveillance with multiple cameras," 2008 International Workshop on Content-Based Multimedia Indexing, London, UK, 2008, pp. 125-132, doi: 10.1109/CBMI.2008.4564937.

[3] Srivastava, Juhi & Tsb, Sudarshan. (2013). Intelligent traffic management with wireless sensor networks. Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA. 1-4. 10.1109/AICCSA.2013.6616429.

[4] Adil Hilmani, Abderrahim Maizate, Larbi Hassouni, "Automated Real-Time Intelligent Traffic Control System for Smart Cities Using Wireless Sensor Networks", Wireless Communications and Mobile Computing, vol. 2020, , 28 pages,https://doi.org/10.1155/2020/8841893

[5] Blynk, “Tutorial: Control Arduino over USB with Blynk app. No shield required.,” *YouTube*. Mar. 21, 2017,[Online]. Available: https://www.youtube.com/watch?v=fgzvoan_3_w.

[6] Theak, “How to install socat on windows | socat,” *YouTube*. Sep. 30, 2020, Accessed: Jul. 15, 2021. [Online]. Available: https://www.youtube.com/watch?v=tYo6KCmelM8.

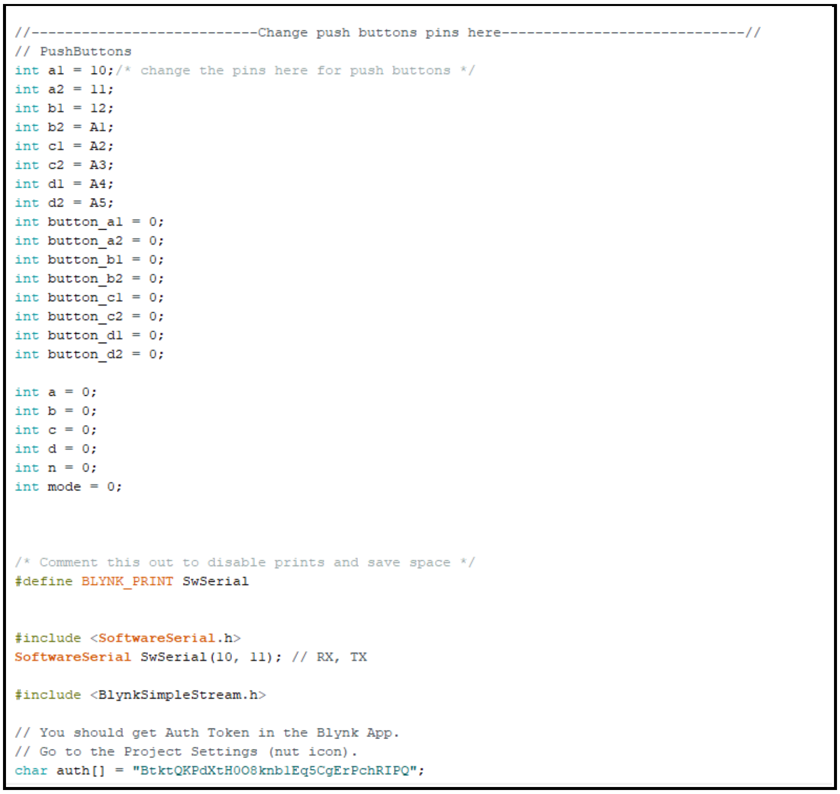
[7] Zhong F, Wang J and Cheng X 2013 *Application of Intelligent Traffic Control Based on PLC*, 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Hangzhou, China, March 22-23, pp 1044-1048

[8] Yousef K M, Al-Karaki J N and Shatnawi A M 2010 Intelligent Traffic Light Flow Control System Using Wireless Sensors Networks, *Journal of Information Science and Engineering* 26 753-768

[9] Khattak M A 2011 PLC Based Intelligent Traffic Control System, *International Journal of Electrical & Computer Sciences* **11**(6) 69-73

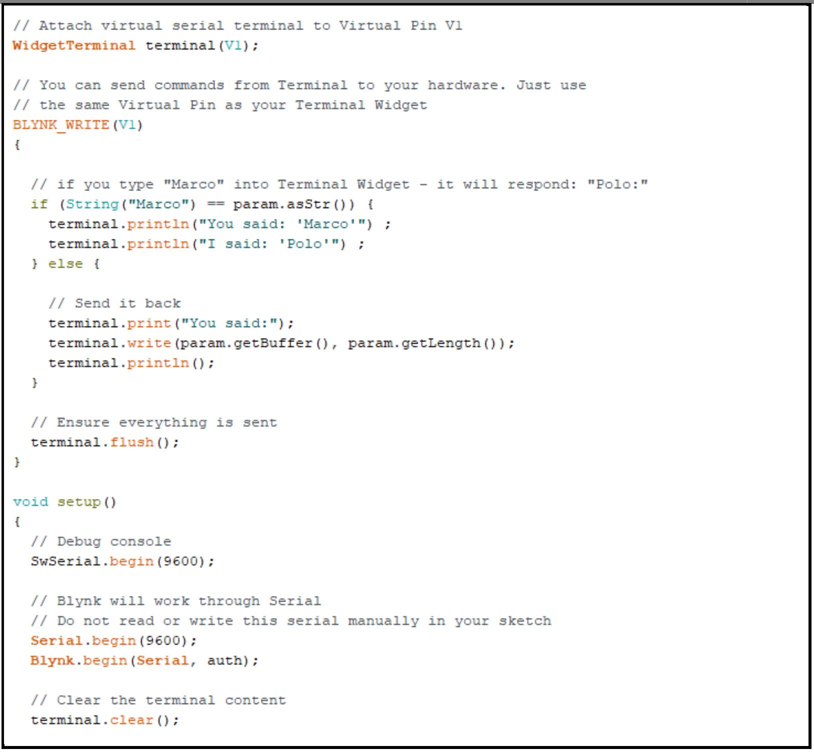
[10] Srivastava M D, Prerna, Sachin S, Sharma S and Tyagi U 2012 Smart Traffic Control System Using PLC and SCADA*, International Journal of Innovative Research in Science, Engineering and Technology* **1**(2) 169-172

APPENDICES



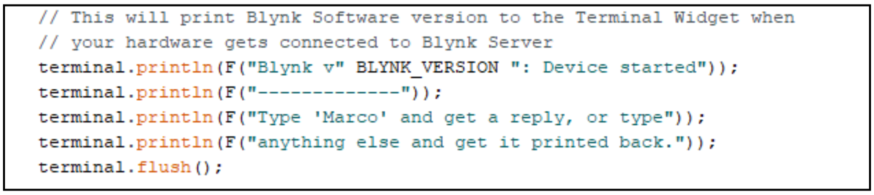
```
//---------------------------Change push buttons pins here-----------------------------//
// PushButtons
int a1 = 10;/* change the pins here for push buttons */
int a2 = 11;
int b1 = 12;
int b2 = A1;
int c1 = A2;
int c2 = A3;
int d1 = A4;
int d2 = A5;
int button_a1 = 0;
int button_a2 = 0;
int button_b1 = 0;
int button_b2 = 0;
int button_c1 = 0;
int button_c2 = 0;
int button_d1 = 0;
int button_d2 = 0;

int a = 0;
int b = 0;
int c = 0;
int d = 0;
int n = 0;
int mode = 0;



/* Comment this out to disable prints and save space */
#define BLYNK_PRINT SwSerial


#include <SoftwareSerial.h>
SoftwareSerial SwSerial(10, 11); // RX, TX

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "BtktQKPdXtH0O8knblEq5CgErPchRIPQ";
```

Appendix A: The first part of coding consists of assigning input of push button, initializing input button and integer a,b,c,d equal to zero.



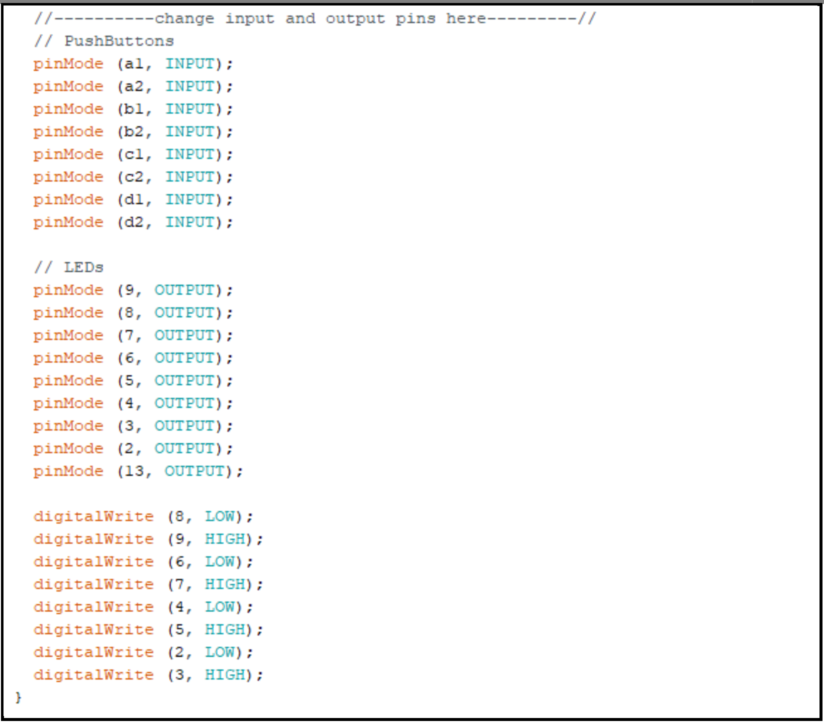
```
// Attach virtual serial terminal to Virtual Pin V1
WidgetTerminal terminal(V1);

// You can send commands from Terminal to your hardware. Just use
// the same Virtual Pin as your Terminal Widget
BLYNK_WRITE(V1)
{

  // if you type "Marco" into Terminal Widget - it will respond: "Polo:"
  if (String("Marco") == param.asStr()) {
    terminal.println("You said: 'Marco'") ;
    terminal.println("I said: 'Polo'") ;
  } else {

    // Send it back
    terminal.print("You said:");
    terminal.write(param.getBuffer(), param.getLength());
    terminal.println();
  }

  // Ensure everything is sent
  terminal.flush();
}

void setup()
{
  // Debug console
  SwSerial.begin(9600);

  // Blynk will work through Serial
  // Do not read or write this serial manually in your sketch
  Serial.begin(9600);
  Blynk.begin(Serial, auth);

  // Clear the terminal content
  terminal.clear();
```

Appendix B: This part consists of controlling Arduino over USB using Blynk apps.



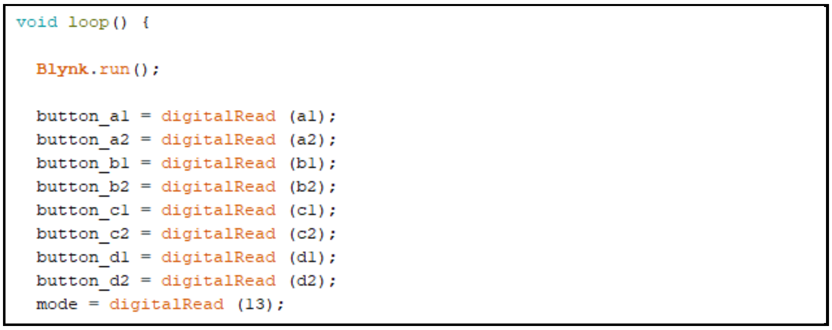
```
// This will print Blynk Software version to the Terminal Widget when
// your hardware gets connected to Blynk Server
terminal.println(F("Blynk v" BLYNK_VERSION ": Device started"));
terminal.println(F("-------------"));
terminal.println(F("Type 'Marco' and get a reply, or type"));
terminal.println(F("anything else and get it printed back."));
terminal.flush();
```

Appendix C: This code to print a connected Arduino with Blynk statement and to test if the Blynk really connects with the Arduino.



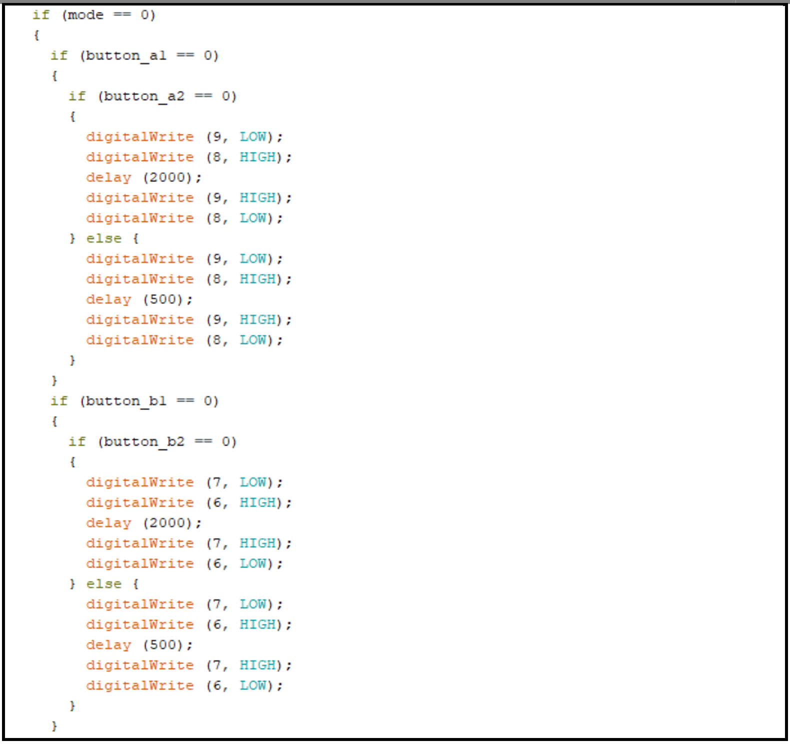
```
  //----------change input and output pins here---------//
  // PushButtons
  pinMode (a1, INPUT);
  pinMode (a2, INPUT);
  pinMode (b1, INPUT);
  pinMode (b2, INPUT);
  pinMode (c1, INPUT);
  pinMode (c2, INPUT);
  pinMode (d1, INPUT);
  pinMode (d2, INPUT);

  // LEDs
  pinMode (9, OUTPUT);
  pinMode (8, OUTPUT);
  pinMode (7, OUTPUT);
  pinMode (6, OUTPUT);
  pinMode (5, OUTPUT);
  pinMode (4, OUTPUT);
  pinMode (3, OUTPUT);
  pinMode (2, OUTPUT);
  pinMode (13, OUTPUT);

  digitalWrite (8, LOW);
  digitalWrite (9, HIGH);
  digitalWrite (6, LOW);
  digitalWrite (7, HIGH);
  digitalWrite (4, LOW);
  digitalWrite (5, HIGH);
  digitalWrite (2, LOW);
  digitalWrite (3, HIGH);
}
```

Appendix D: This code consists of the traffic light system part by assigning the pin from the Arduino to be input or output



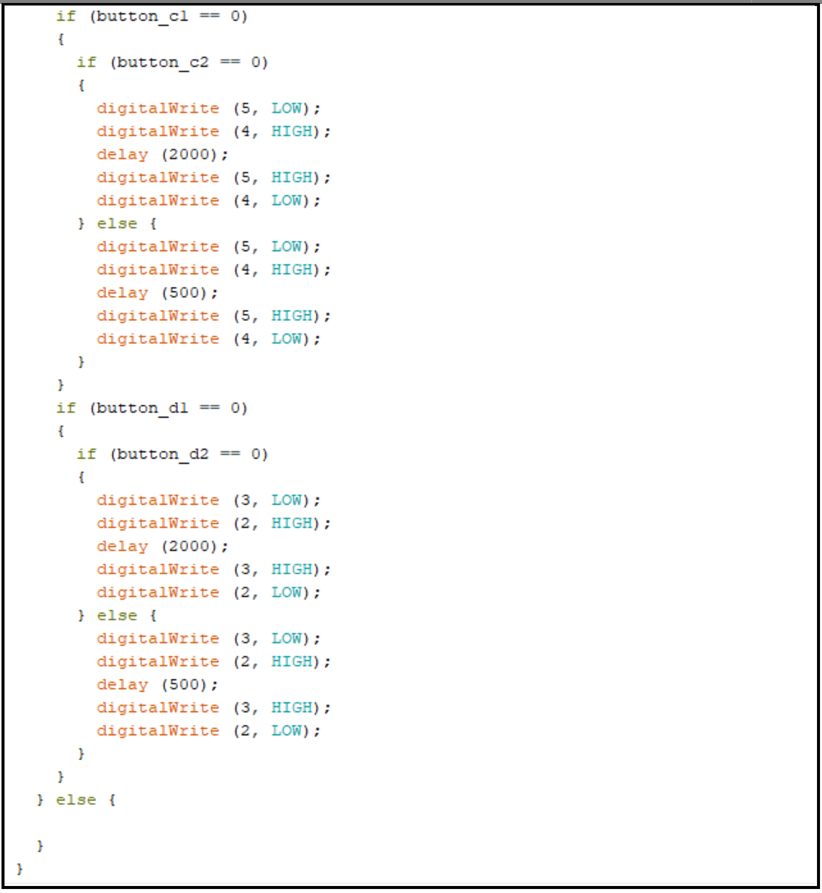
```
void loop() {

  Blynk.run();

  button_a1 = digitalRead (a1);
  button_a2 = digitalRead (a2);
  button_b1 = digitalRead (b1);
  button_b2 = digitalRead (b2);
  button_c1 = digitalRead (c1);
  button_c2 = digitalRead (c2);
  button_d1 = digitalRead (d1);
  button_d2 = digitalRead (d2);
  mode = digitalRead (13);
```

Appendix E: This code consists of reading input of the assigned push button



```
if (mode == 0)
{
  if (button_a1 == 0)
  {
    if (button_a2 == 0)
    {
      digitalWrite (9, LOW);
      digitalWrite (8, HIGH);
      delay (2000);
      digitalWrite (9, HIGH);
      digitalWrite (8, LOW);
    } else {
      digitalWrite (9, LOW);
      digitalWrite (8, HIGH);
      delay (500);
      digitalWrite (9, HIGH);
      digitalWrite (8, LOW);
    }
  }
  if (button_b1 == 0)
  {
    if (button_b2 == 0)
    {
      digitalWrite (7, LOW);
      digitalWrite (6, HIGH);
      delay (2000);
      digitalWrite (7, HIGH);
      digitalWrite (6, LOW);
    } else {
      digitalWrite (7, LOW);
      digitalWrite (6, HIGH);
      delay (500);
      digitalWrite (7, HIGH);
      digitalWrite (6, LOW);
    }
  }
```

Appendix F: This code consists of the first two junctions that will give output if the sensors detect the car in the junction.



```
    if (button_c1 == 0)
    {
      if (button_c2 == 0)
      {
        digitalWrite (5, LOW);
        digitalWrite (4, HIGH);
        delay (2000);
        digitalWrite (5, HIGH);
        digitalWrite (4, LOW);
      } else {
        digitalWrite (5, LOW);
        digitalWrite (4, HIGH);
        delay (500);
        digitalWrite (5, HIGH);
        digitalWrite (4, LOW);
      }
    }
    if (button_d1 == 0)
    {
      if (button_d2 == 0)
      {
        digitalWrite (3, LOW);
        digitalWrite (2, HIGH);
        delay (2000);
        digitalWrite (3, HIGH);
        digitalWrite (2, LOW);
      } else {
        digitalWrite (3, LOW);
        digitalWrite (2, HIGH);
        delay (500);
        digitalWrite (3, HIGH);
        digitalWrite (2, LOW);
      }
    }
  } else {

  }
}
```

Appendix G: This code consists of the last two junctions that react the same with the previous junction.