School of Science and Technologies

RMIT UNIVERSITY

OENG1207 – Digital Fundaments

GROUP PROJECT REPORT

Traffic Lights with Pedestrian Button

GROUP INFORMATION

Group 2, Team 10

Tuesday, 13:00 – 15:00

STUDENTS

Nguyen Quoc Hoang	s3697305
Phan Ngoc Quang Anh	s3810148
Nguyen Tan Huy	s3864185

SUBMISSION DUE DATE

September 27, 2020

I. Introduction and Background

Traffic lights are common everywhere and can be found at almost every road intersection that play an important role in controlling the flows of traffic. In the final project of Digital Fundamental course, we were required to design a traffic light system in MATLAB/Simulink and implement our model on an Arduino microcontroller board.

Our project was divided into two phases, the first one was Group Milestone 1 which the requirement was to design an automated traffic light system using 'Counter Limited' block. The second phases required a Stateflow block and a switch that acts as a pedestrian crossing button [1]. The whole project consists of three traffic lights (red, yellow, green) and two pedestrian lights (red, green) and multple Simulink logic blocks depending on the requirements of each phase.

II. Design and Methodology

Group milestone 1

In order to construct our automated traffic light system, we came up with an appropriate algorithm for the project. According to the requirements, a full sequence of this particular traffic light system would last for a total of 16 counts and then reset itself. Therefore, we need a timer counting from 0 to 15 (16 counts) and it is a fundamental component of the system. Here is our algorithm:

- Step 1: Initiate a timer that counts from 0 to 15
- Step 2: When counts are fewer than 7 (7 counts), the **green traffic light** and the **red pedestrian light** are turned on while others stay off. Vehicles are allowed to go while pedestrian must not cross the street
- Step 3: When counts are more than 6 and fewer than 10 (3 counts), only the **yellow traffic light** and **red pedestrian light** are turned on. Traffic slows down, pedestrians are still not allowed to cross.
- Step 4: **Red traffic light** as well as **green pedestrian light** is on if counts are more than 9 (6 counts). Cars must stop while pedestrian can cross the street.

- Step 5: Go back to Step 1.

Simulink blocks and their usages:

- 'Counter Limited' block: set up a timer that counts from 0 to 15. The counts are also input values.
- 'AND Logical Operator' blocks: control input values to reach desired outputs.
- 'Scope' block: display data demonstrating the operation of the system with respect to time.
- 'Lamp' blocks: visualize the traffic lights for better demonstration.
- 'Output' block: get the outputs.

Our Simulink model for the Milestone 1's requirements is as followed:

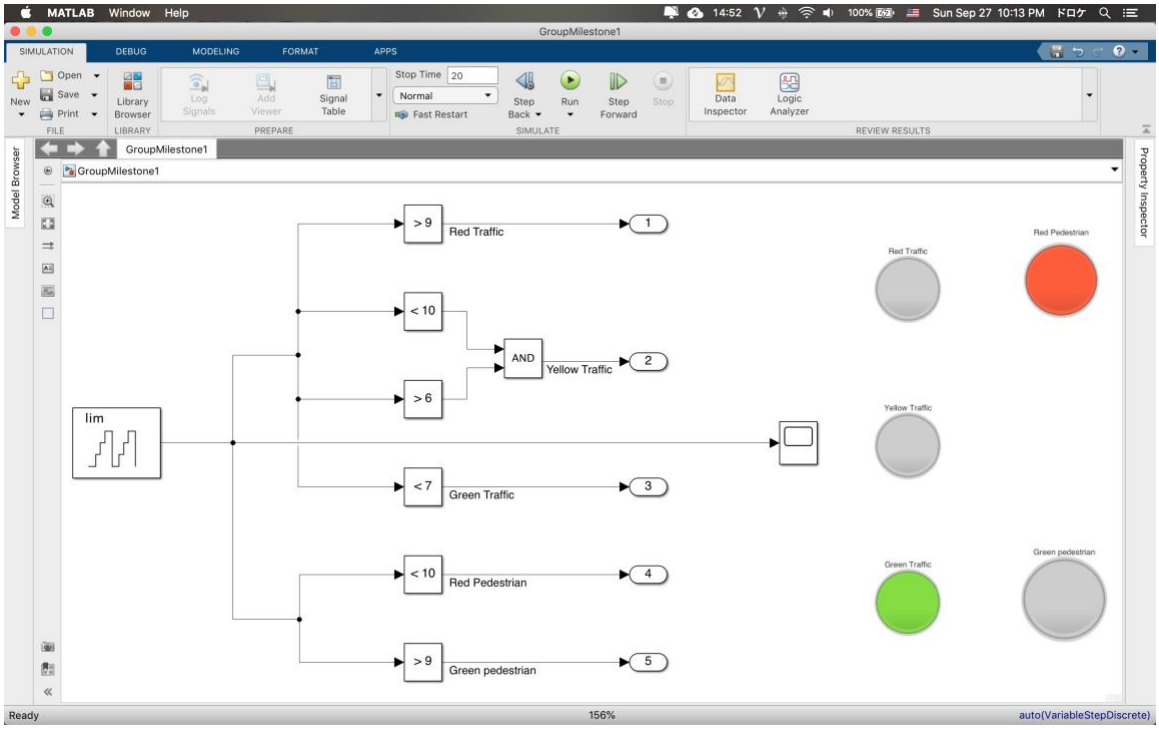

Figure 1: Milestone 1's model

The model above is showing an example of the system when counts is less than 7, which means cars are being allowed to go whereas pedestrians are required to stop.

Final project

The traffic light system with a **pedestrian crossing button** operates as a form of finite state machine with 3 states.

- State 1 (default state): The **pedestrian crossing button** is not pushed.
  + **Green traffic light** is on, cars can go.
  + **Red pedestrian light** is on, pedestrians must stop.
  + Other lights are off.

- Sate 2: The pedestrian button is pushed.
  + **Yellow traffic light** is on, cars are alerted
  + **Red pedestrian light** is on; pedestrians are still not allowed to cross.
  + Other lights are off.

- State 3: After 3 seconds in stage 2.
  + **Red traffic light** turns on, vehicles must stop.
  + **Green pedestrian light** is on, pedestrian can cross the street.
  + Other lights are off.
  + Go back to state 1 after 6 seconds.

The following Flow chart illustrates briefly three difference states of our system:

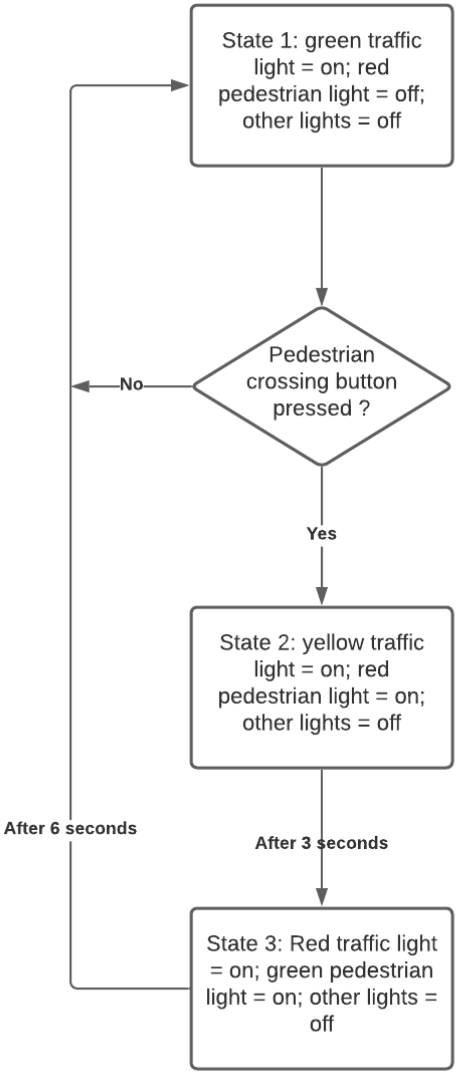

Figure 2: Final Project's Flow chart

Simulink blocks and their usages:

- 'Chart' block from Stateflow library: create the states of a state machine along with conditions for the transition between different states.
- 'Slider Switch' block: simulate the pedestrian crossing button/switch.
- 'Scope' block: display data demonstrating the operation of the system with respect to time.
- 'Lamp' blocks: visualize the operation of the traffic light system.

The following screenshot provides a testing evidence for our Final project's model:

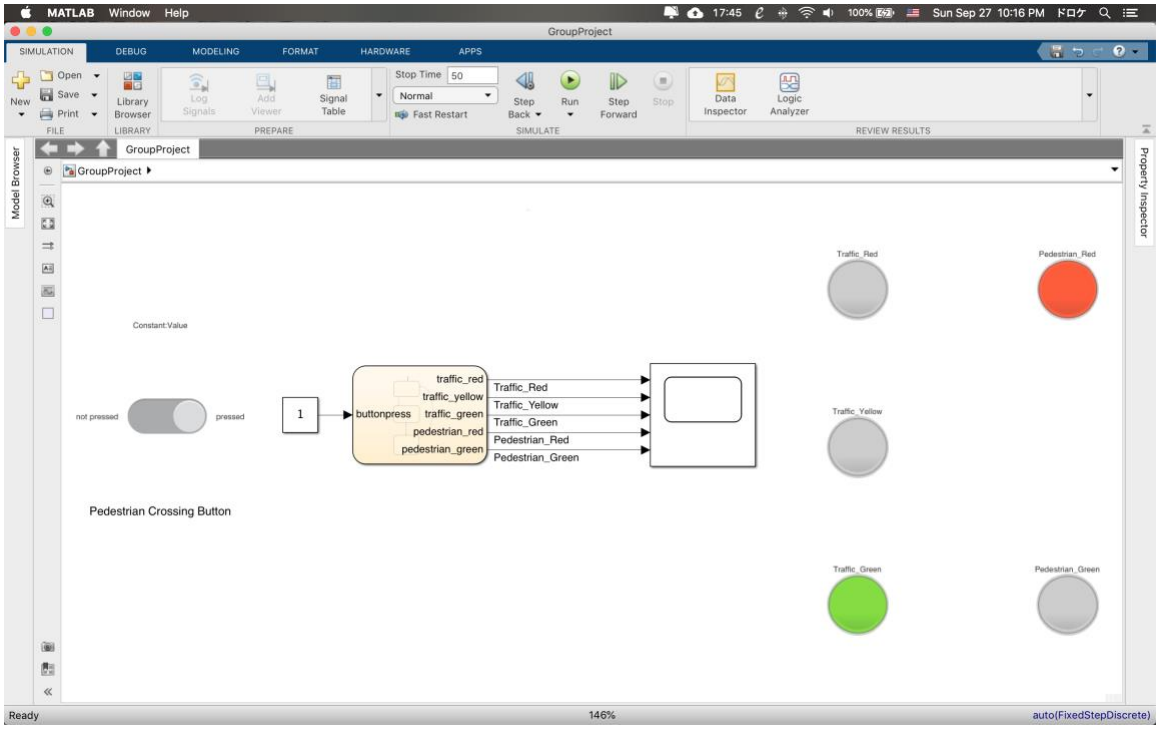

Figure 3: Final Project's model

We use state 1 as an example here, where traffic light is green and pedestrian light is red. It is the default state and only switch to state 2 after the button is pressed.

III. Output and Testing

For our practical session, we replaced the switch with an input from a momentary switch in the Arduino package for Simulink. We also added output signals from the Stateflow block to 5 output pins on the Arduino board. We configure our model correspondingly to the input and output pin numbers, and later on deployed on the Arduino Uno board.

For the components, we used a total of 5 LEDs as in our system for the traffic and pedestrian lights, 6 resistors, a momentary switch which acts as a pedestrian crossing button. We put our components on a breadboard and connect with the Arduino board using multiple wires. Lastly, we connected our system to the laptop

and deployed our Simulink model onto the hardware. The following photo shows our components and wiring:

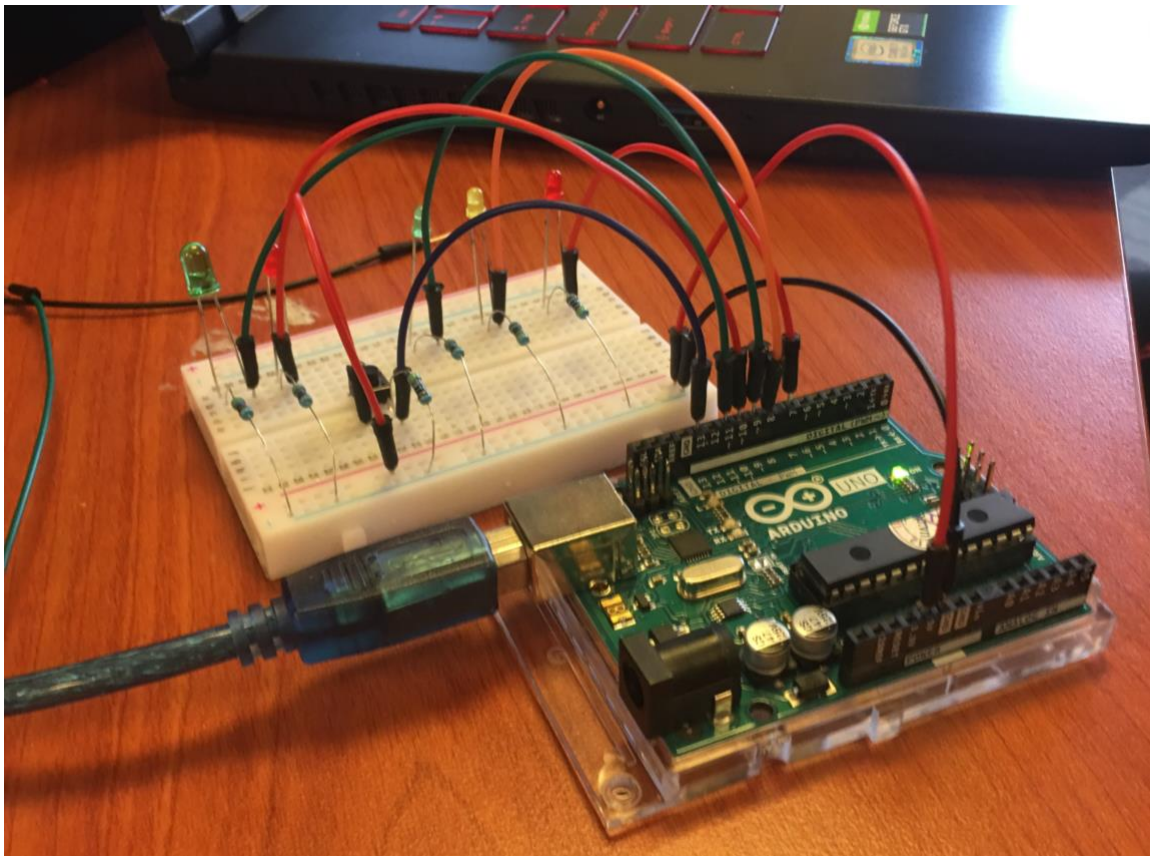

Figure 4: Wiring and configurations

We successfully deployed our model onto the board. As we start up the board, it would always remain in state 1 unless we pressed the button. Both the lights on our simulation and the LEDs on breadboard changed at the same time.

State 1:

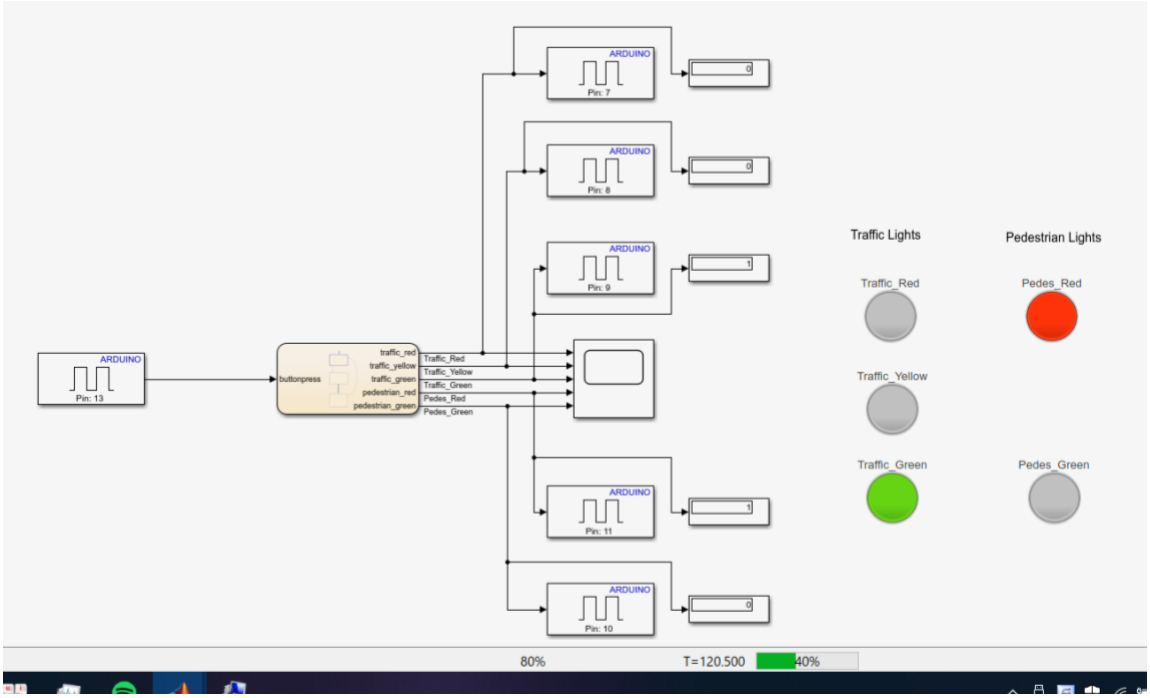

Figure 5: State 1 -Simulation

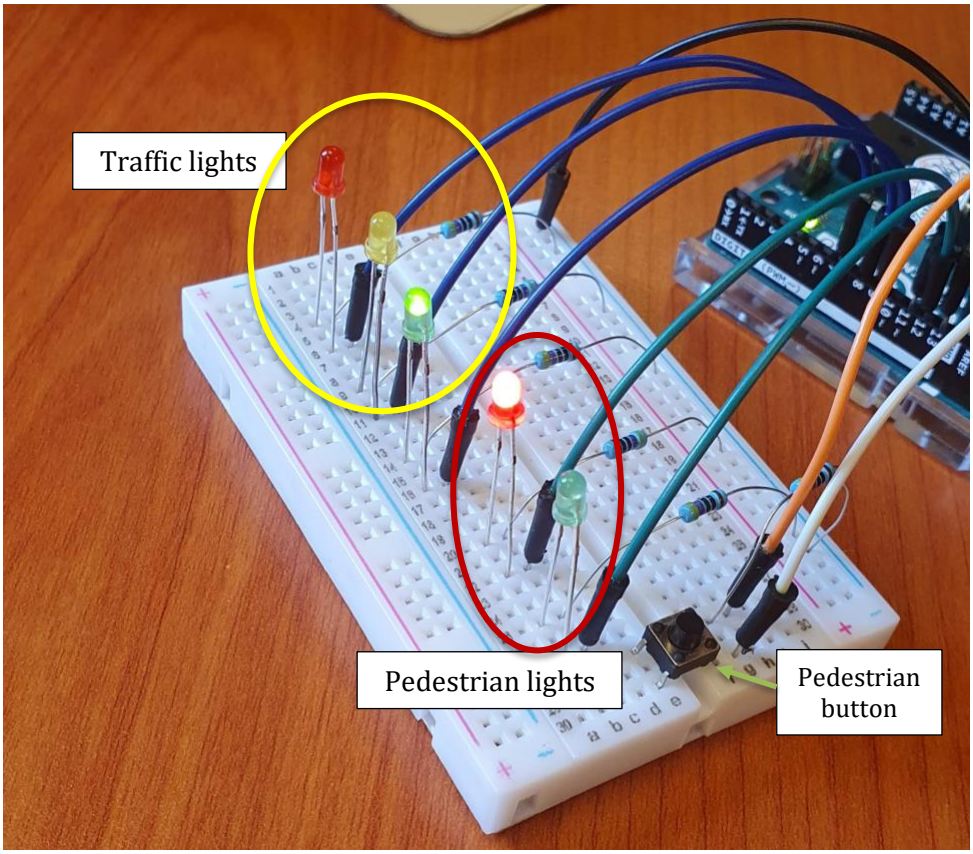

Figure 6: State 1 – Hardware

As seen from the 2 pictures above, when the system is in state 1, the traffic light is green and the pedestrian light is red. It will remain in this state until we press the button and the system will go to state 2.

State 2:

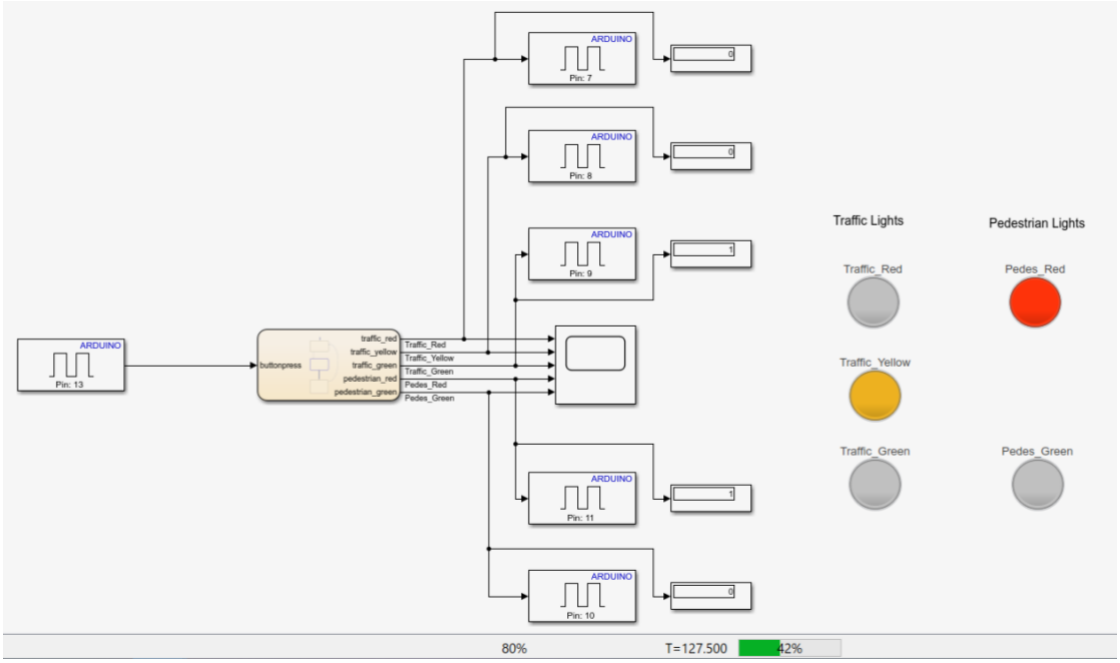

Figure 7: State 2 - Simulation

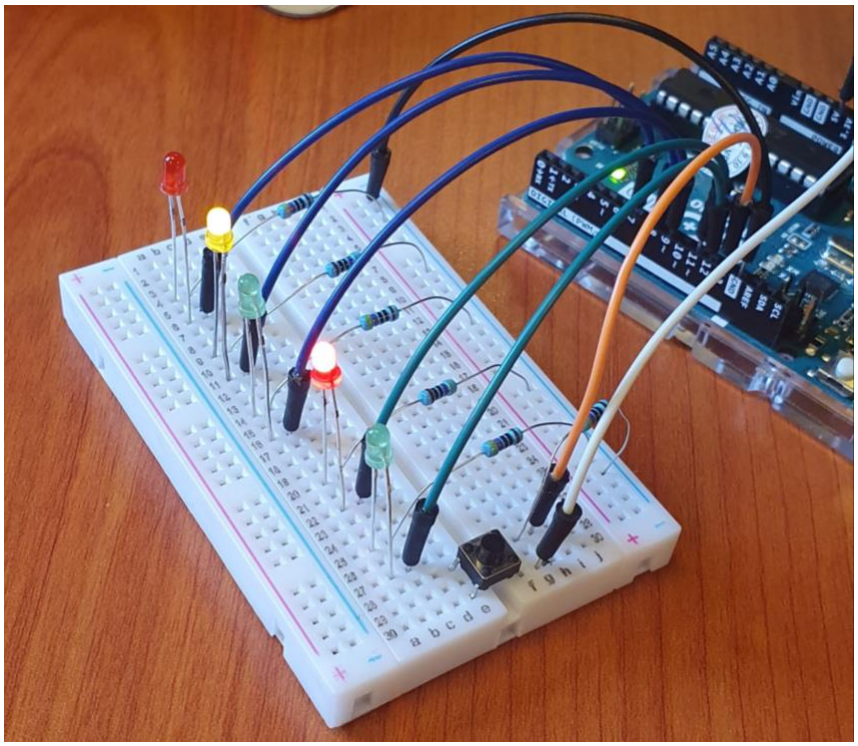

Figure 8: State 2 - Hardware

The system will go to state 2 three seconds after the pedestrian button is pressed. In this state, traffic light will turn yellow to signal vehicles to slow down, preparing for pedestrian to cross.

State 3:

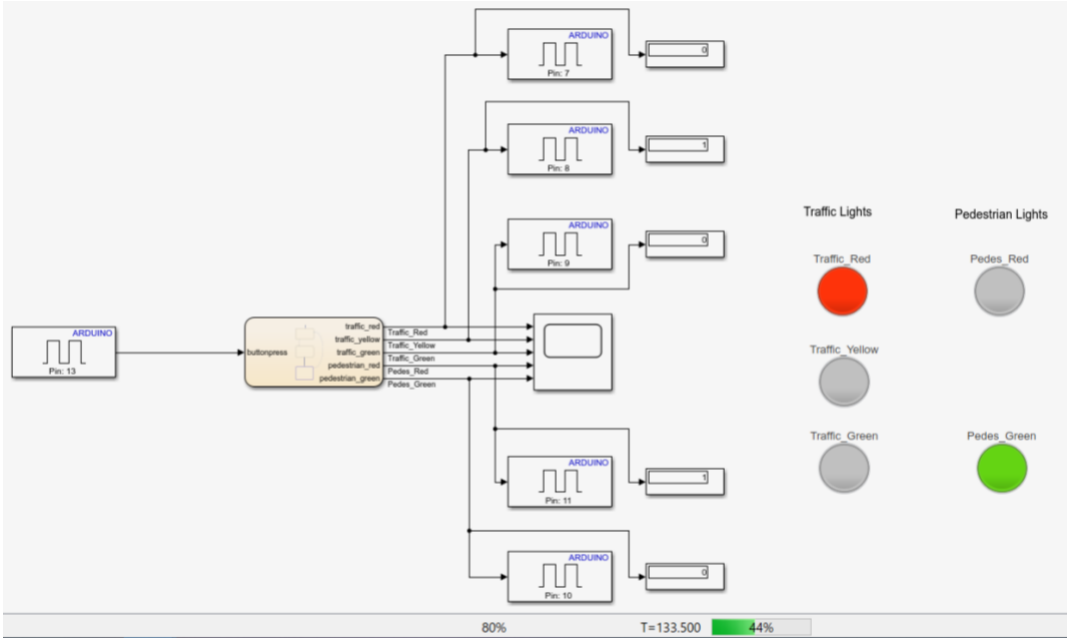

Figure 9: State 3 - Simulation

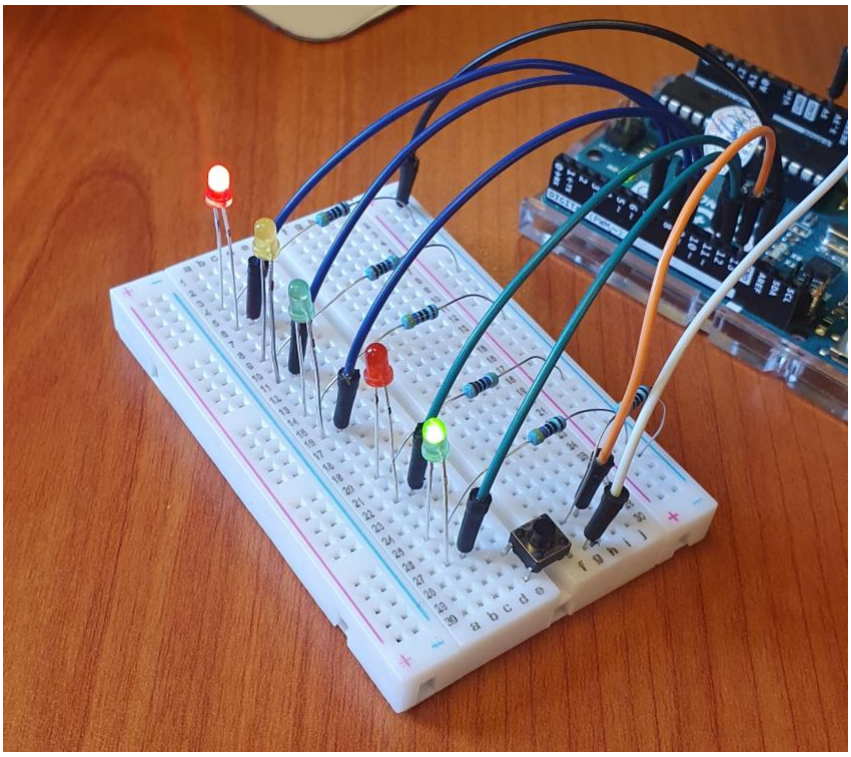

Figure 10: State 3 – Hardware

This final state will stop all vehicles and allow 6 seconds for the pedestrians to cross the street. After 6 seconds, the system will switch back to state 1, allows the traffic flow to continue.

We can display our results in another way using scope. We can tell when a light is on or off when its value is 1 or 0, respectively:

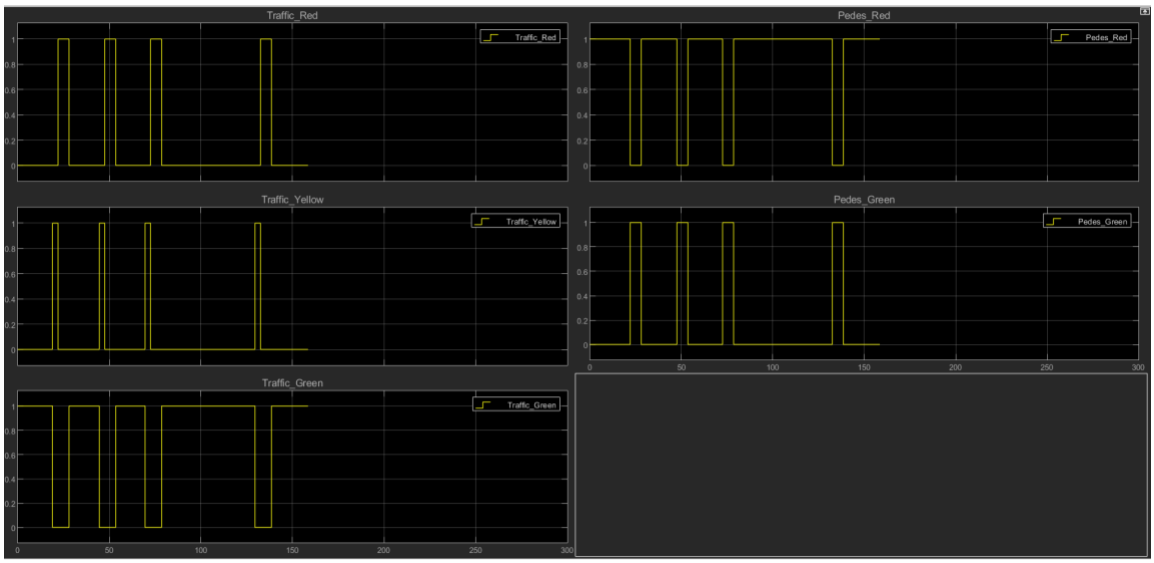

Figure 11: Result from Scope

The 3 graphs on the left side show the signal for the traffic lights, which are red, yellow, green from top to bottom. The 2 graphs on the right show the pedestrian lights: red and green. As we can see, if a signal of one light is red, the signal of its green light is green, and vice versa. This has proven that our model has worked successfully, both in the simulation and on hardware.

IV. Conclusion

The system revolves around the very interesting topic of traffic lights and their autonomy. Most importantly, this project allows us to have a better understanding of Simulink in MATLAB, as well as the techniques of deploying a software model on a hardware to work. Also, the system we created has fulfilled all the requirements of the task and our automatic traffic light system worked very smoothly and did not crash at all in the demonstration. What we might be able improve on our system in

the future is adding symbols that pedestrians can cross the street when the light is red or stop when the light is green so they can be more easily recognized. We could also make our model into real traffic poles, with countdown timer to allow drivers and pedestrians know how much time they have left.

V. Appendix

Our final system with Arduino input/output ports is attached bellowed:

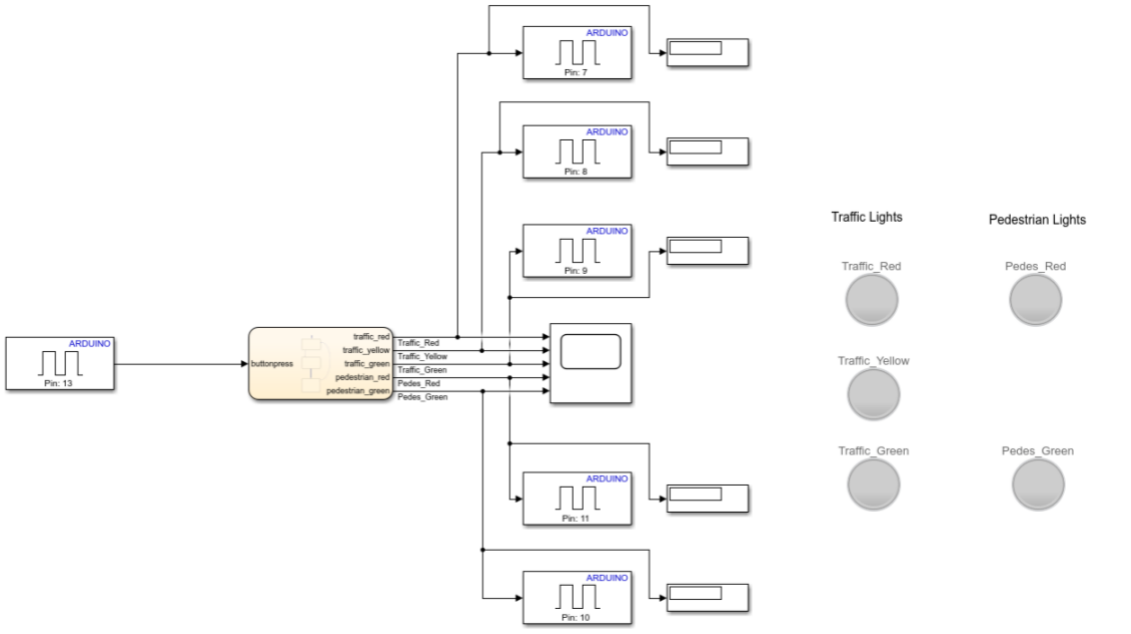

Figure 12: Final model with hardware implementations

VI. Reference

[1] Alexandru Fechete, “Group Project Specifications”, 2020.