Stoplight

Engineering 1281H

Autumn, 2023

Sreekar Kutagulla

Samara Buchanan

Shay Verma

Eddie Kuper

J.Davidson 8:00

Date of Experiment: 11/15/23

Date of Submission: 11/28/23

With the myriad of things that humans cannot observe or react to, it has become increasingly important to use modern computers, that are capable of running billions processes per second, to simulate and slow down such things that humans have trouble seeing. For example, a simple stoplight at the intersection of two streets can be determined by a multitude of factors such as timing, pedestrians, amount of cars etc... As a result, using simulations, engineers and scientists can model such factors and run tests to find certain deliverables, improving the safety and efficiency of objects like stoplights. Specifically, in this lab, groups of Ohio State students did just that, employing the use of a Proteus controller and simulator to build a small-scale physical and simulated stoplight. The purpose of the experiment is to learn how to create simulations through Proteus and provide information about the utility of such simulations.

To complete this experiment, the groups needed an FEH Proteus Controller, a computer with an application such as VSCode that can run C/C++, the Proteus Project zip folder, a MicroSD, a USB adapter, and a charger. Once the group attained all these materials, they opened the Proteus Project zip folder in VSCode and inserted the MicroSD card into the USB which was then inserted into the computer. The code was then compiled onto the MicroSD card which was then removed and inserted into the Proteus Controller. Once the controller was turned on, the code was edited such that it included appropriate libraries, pin declaration statements, and touch and release logic. Then, each stoplight was connected to a certain pinouts, and the code was further modified to operate a mock stoplight. Once the physical Proteus was completed and returned, the groups filled in code sections to create a simulated stoplight on the Proteus simulator on the computers.

Data from the physical Proteus was not gained during this experiment as the code did not compile on the computers and the MicroSD card had code from previous trials. By the end of the period, the code compiled on the computers and the MicroSD card was wiped clean. However, the simulation part of the experiment yielded a cyclical stoplight, running as intended. It was clear that the simulation was more advantageous as it was able to run locally on a computer and has less needs than the physical experiment. The physical experiment relied heavily on the MicroSD card while the simulation depended on a working zip folder and terminal.

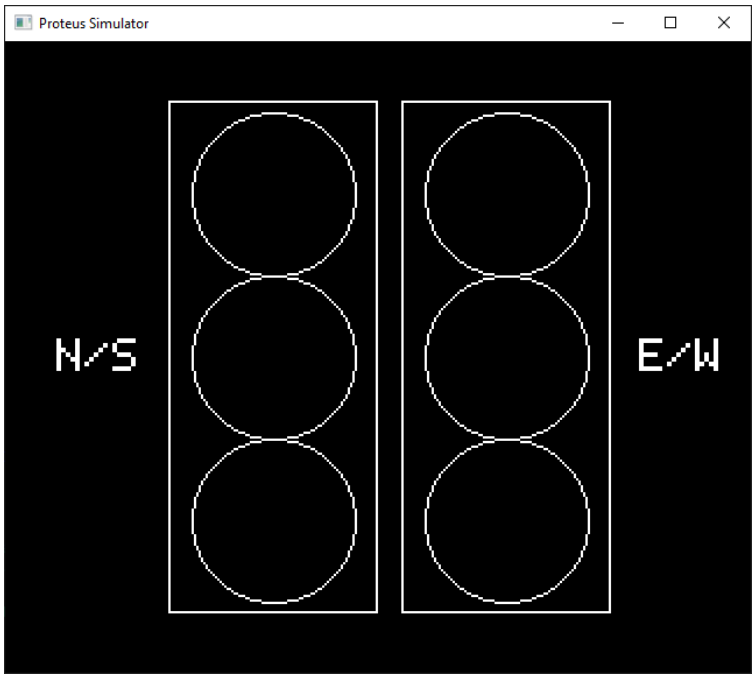
The use of such simulations, like the experiment, allows engineers to provide information about the factors unseen to the human eye which affect the safety of individuals. Data can show the dangers of such factors and their effects on the efficiencies of systems like stoplights.

Stoplight Worksheet

Instructions
For this Post Lab Worksheet, you will be converting the stoplight code that you wrote in lab to work on the Proteus Simulator. The Proteus Simulator is a desktop-based simulation that works on Windows or MacOS.

Using the instructions found on the FEH Proteus website, install Visual Studio Code and any other required tools, then follow the instructions to compile and run the code provided in Simulator_Project.zip. Example output is shown below in Figure 1. The stoplight on the left represents a stoplight facing the North and South direction, and the stoplight on the right represents a stoplight facing in the East and West direction. Note that the Proteus simulator comes preinstalled on computers in HI206, HI208, HI214, and HI216.

Figure 1: Proteus Simulator Stoplight base code.



Review the DrawStoplight function. This function is responsible for drawing either the N/S or E/W stoplight. It accepts two integers, orientation and activeLight. The variable orientation is 0 for the N/S light, and 1 for the E/W light. The variable activeLight is 0 if the light is red, 1 if the light is yellow, and 2 if the light is green.

Fill in the missing code in DrawStoplight. There are 6 places to add code, which are labelled with // FILL IN. The code you are filling in is responsible for drawing the light on the stoplight that is currently turned on.

In the main function, use the DrawStoplight function to implement your stoplight logic. Use the same stoplight cycle that was used in lab, but without the walk and don't walk lights. For bonus points, you may implement the double red bonus from the lab.

Proteus Simulator Screenshots and Reflection

1. Insert a screenshot of your program when the N/S light is green, and the E/W light is red.

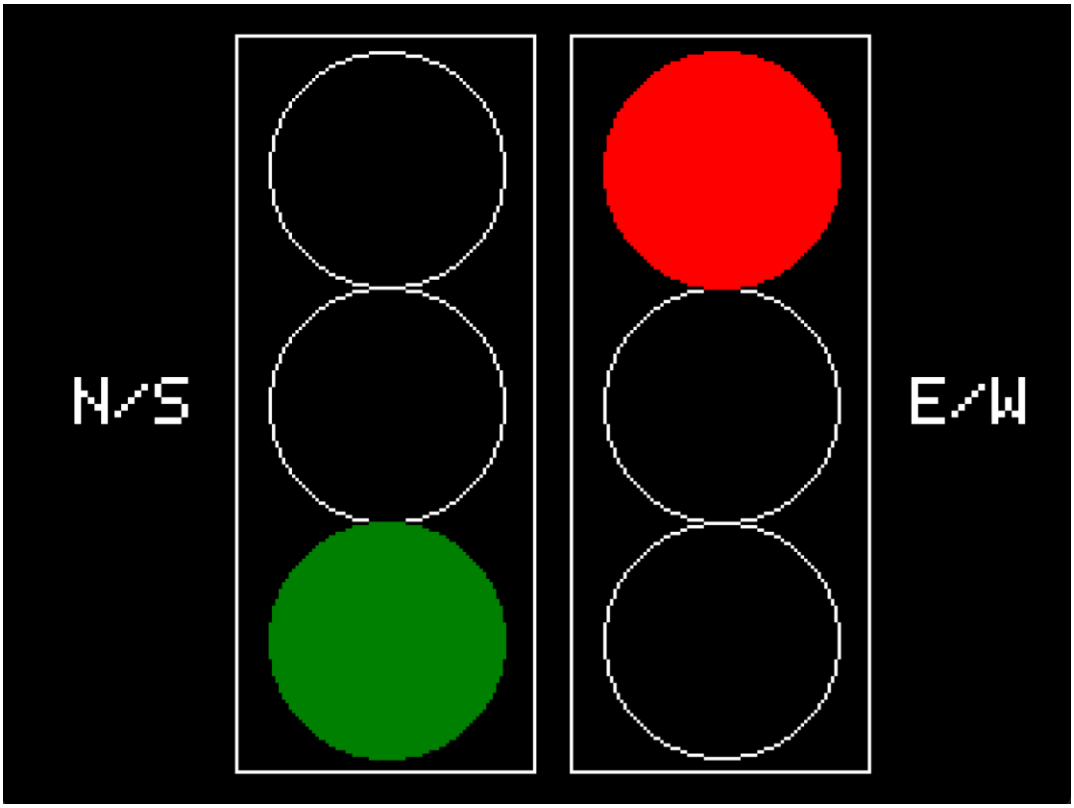


Figure 2: Green N/S Light.

2. Insert a screenshot of your program when the N/S light is yellow, and the E/W light is red.

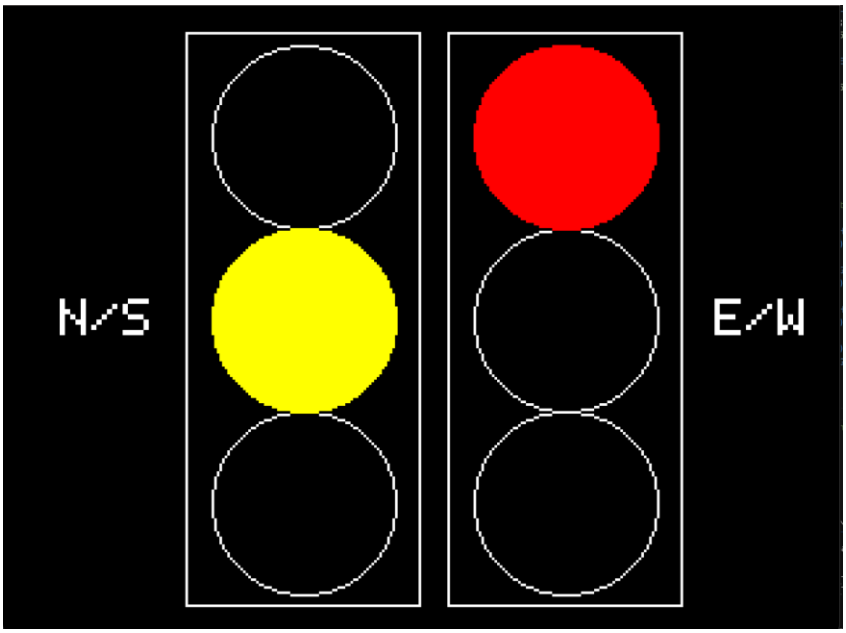


Figure 3: Yellow N/S Light.

3. Insert a screenshot of your program when the N/S light is red. If you completed the double red bonus, your E/W light will also be red. If not, it will be green.

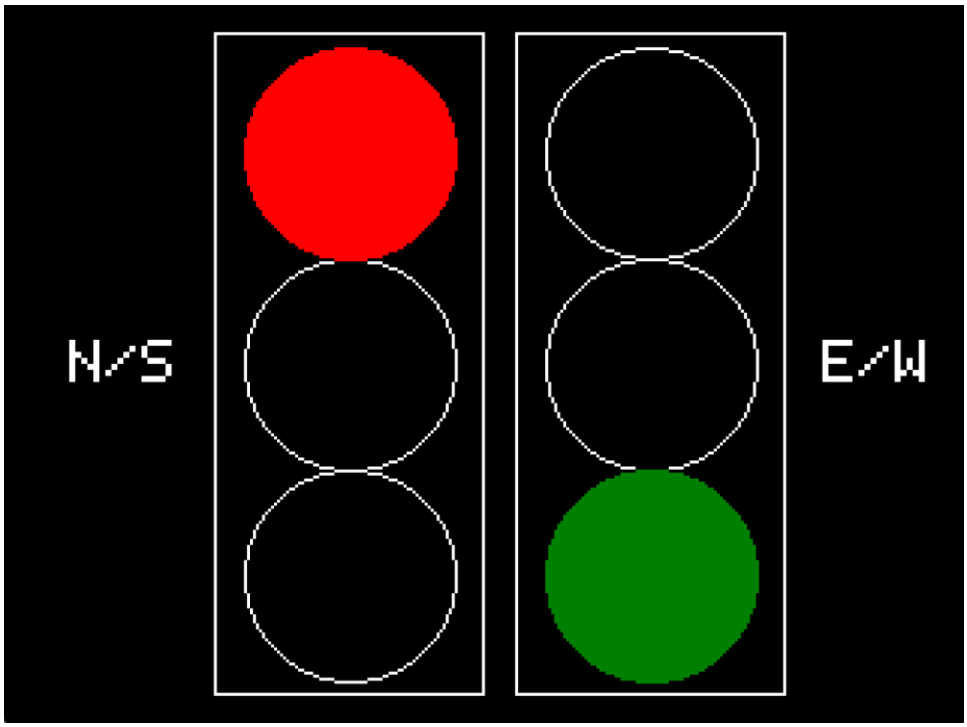


Figure 4: Red N/S Light.

4. Copy your code from *main.cpp* and paste it here. If you completed the double red bonus, highlight the portion of your code that is responsible for the bonus.

```
#include "FEHLCD.h"
#include "FEHUtility.h"


#define N_S 0
#define E_W 1
#define RED_LIGHT 0
#define YELLOW_LIGHT 1
#define GREEN_LIGHT 2


/*
* Draws a stoplight to the screen
* orientation = 0 if N/S, 1 if E/W
* activeLight = 0 for red, 1 for yellow, 2 for green
*/
void DrawStoplight(int orientation, int activeLight) {
LCD.SetFontColor(LCD.White);
LCD.WriteAt("N/S", 20, 112);
LCD.WriteAt("E/W", 270, 112);
if (orientation == N_S) {
// N/S
// Clear what was there before by drawing black over it
LCD.SetFontColor(LCD.Black);
LCD.FillRectangle(70, 10, 90, 220);
// Draw the outline of the stoplight
```

```
LCD.SetFontColor(LCD.White);
LCD.DrawRectangle(70, 10, 90, 220);
LCD.DrawCircle(115, 50, 35);
LCD.DrawCircle(115, 120, 35);
LCD.DrawCircle(115, 190, 35);


// Draw the light that is currently active
if (activeLight == RED_LIGHT) {
LCD.SetFontColor(RED);
LCD.FillCircle(115,50,35);
} else if (activeLight == YELLOW_LIGHT) {
LCD.SetFontColor(YELLOW);
LCD.FillCircle(115,120,35);
} else if (activeLight == GREEN_LIGHT) {
LCD.SetFontColor(GREEN);
LCD.FillCircle(115,190,35);
}
} else if (orientation == E_W) {
// E/W
// Clear what was there before by drawing black over it
LCD.SetFontColor(LCD.Black);
LCD.FillRectangle(170, 10, 90, 220);
// Draw the outline of the stoplight
LCD.SetFontColor(LCD.White);
LCD.DrawRectangle(170, 10, 90, 220);
LCD.DrawCircle(215, 50, 35);
LCD.DrawCircle(215, 120, 35);
LCD.DrawCircle(215, 190, 35);
```

```
// Draw the light that is currently active
if (activeLight == RED_LIGHT) {
LCD.SetFontColor(RED);
LCD.FillCircle(215,50,35);
} else if (activeLight == YELLOW_LIGHT) {
LCD.SetFontColor(YELLOW);
LCD.FillCircle(215,120,35);
} else if (activeLight == GREEN_LIGHT) {
LCD.SetFontColor(GREEN);
LCD.FillCircle(215,190,35);
}
}
LCD.Update();
}


/* Entry point to the application */
int main() {
// Infinite loop so the stoplights run until the program is closed
while (1) {
DrawStoplight(N_S, GREEN_LIGHT);
DrawStoplight(E_W, RED_LIGHT);
Sleep(1.0);
DrawStoplight(N_S, YELLOW_LIGHT);
DrawStoplight(E_W, RED_LIGHT);
Sleep(1.0);
DrawStoplight(E_W, GREEN_LIGHT);
DrawStoplight(N_S, RED_LIGHT);
Sleep(1.0);
DrawStoplight(N_S, RED_LIGHT);
```

```
DrawStoplight(E_W, YELLOW_LIGHT);
Sleep(1.0);


// Add logic to make stoplights function
// FILL IN
}
return 0;
}
```