Train Project Notebook

Submitted to:
Instructor Dr. Bixler
GTA Isa Fernandez

Created by:
Team A

Kai Chuen Tan
Frances Severding
Annette Petty
Stephen Doucet

Engineering 1181
The Ohio State University
Columbus, OH
24th April 2017

Table of Contents

List of Figures and Tables

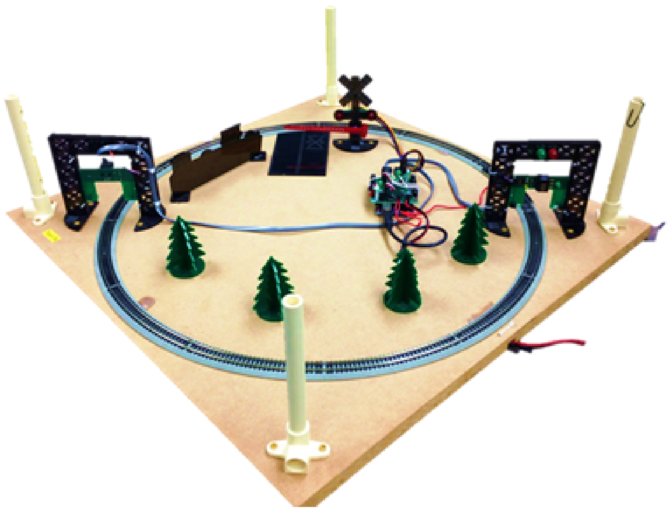


Figure 1 : N-Scale Model Train Set

A MATLAB program that met specific requirements for a train operating system was designed and tested in lab by using control of the N-Scale Model Train Set (*Figure 1*) via interaction with an Arduino microcontroller.

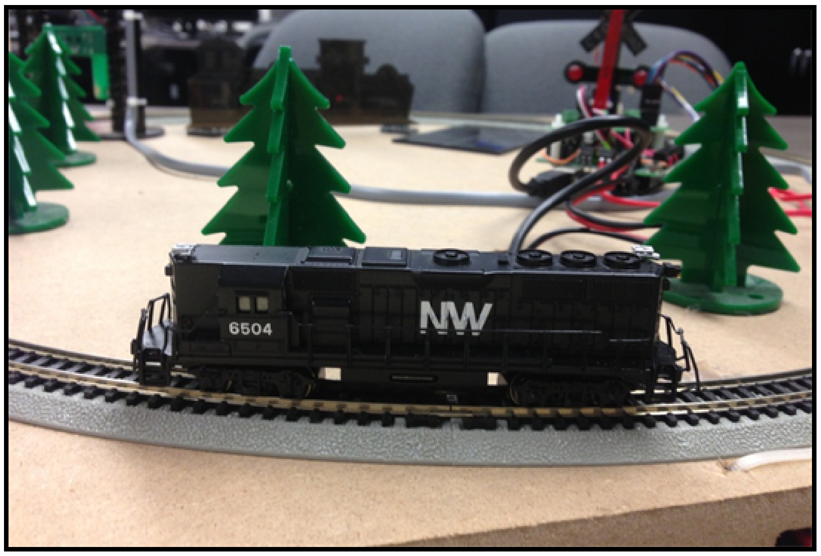


Figure 2 : N-Scale Train Locomotive

Figure 2 shows the N-Scale Train Locomotive on the N-Scale Model Train Set; N-scale is 1/160 of the real train's size. The width of the track was 9 mm between rails, which were made of metal for conductivity. The speed and direction of the train could be controlled by an Arduino Microcontroller.

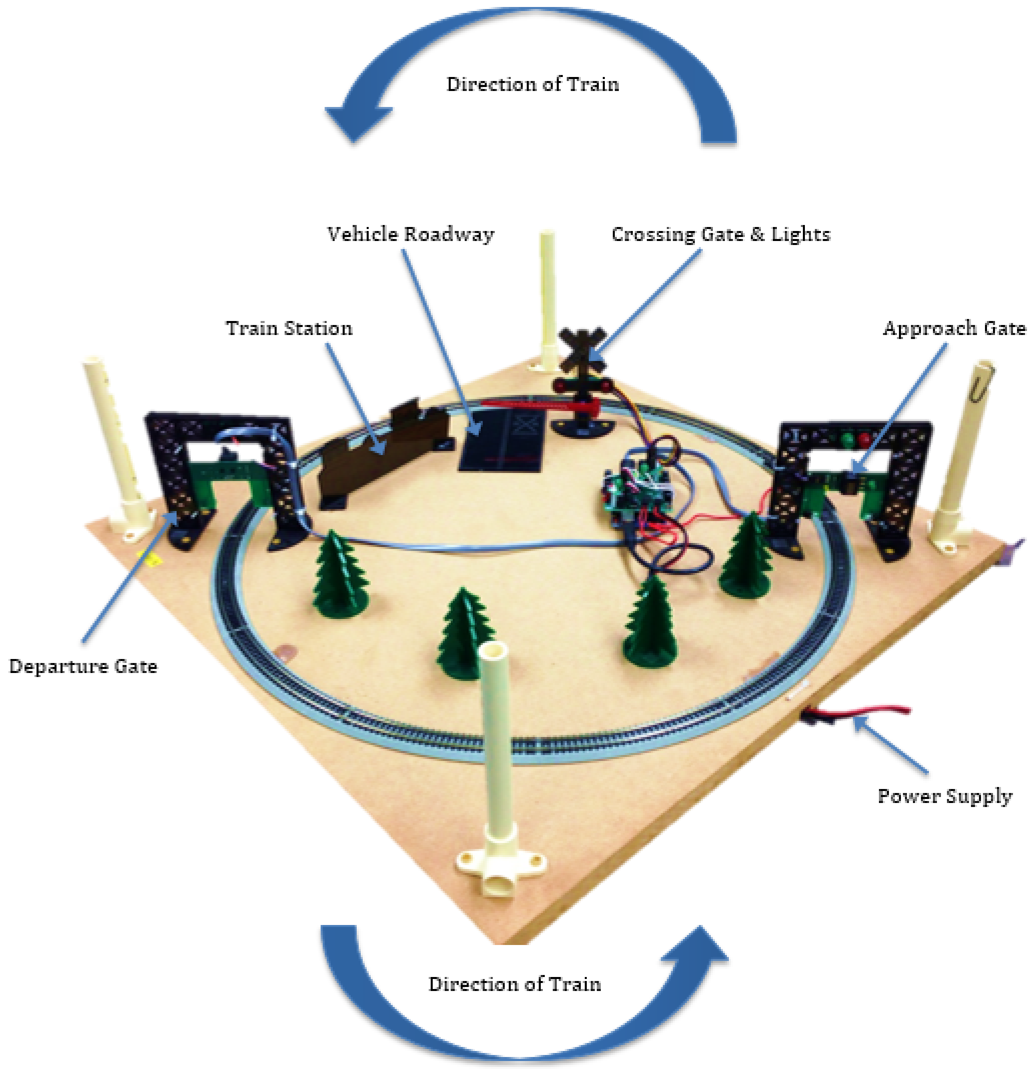


Figure 3 : Train Track Layout

Figure 3 illustrates the complete layout of the train track. The N-Scale Train Set consisted of a departure gate (with break beam sensors, red LED, and green LED), approach gate (with break beam sensors, red LED, and green LED), train station, vehicle roadway, and a crossing gate with 2 red LEDs. The N-Scale Train Locomotive moved in anti-clockwise direction. When the departure gate's break beam sensors detected the train, the train would move at maximum speed. After the train crossed the approach gate, the two red LEDs would flash alternatively, and the crossing gate would close after a delay.

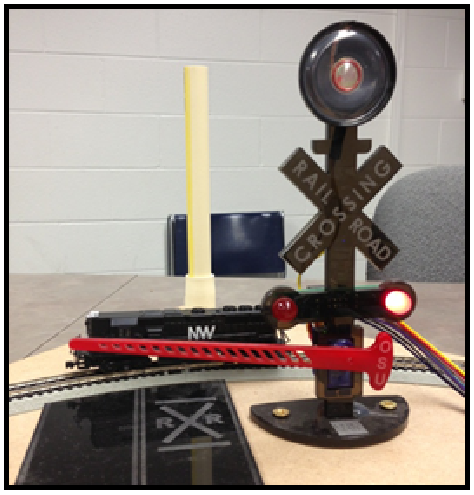


Figure 4 : Crossing Gate

Figure 4 shows the railroad gate, which was controlled by a servo motor that was attached to the Arduino Microcontroller. The flashing red LEDs are to warn drivers that the train was arriving soon and that they should stop their cars. The crossing gate was to prevent the cars from crossing the railway.

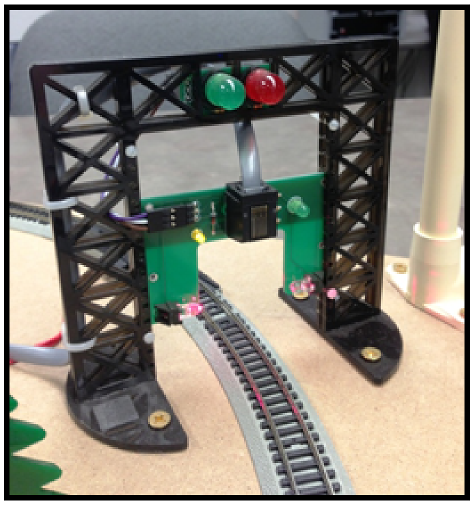


Figure 5 : Break Beam Sensor

Figure 5 shows one of the two gates with break beam sensors, red LED, and green LED. Each break beam sensor has an LED that was facing a photoreceptor. The LED constantly shot a red beam towards the photoreceptor. While the beam was unobstructed, a low numerical value would be sent to MATLAB; when the red beam was obstructed, a significantly higher value would be returned to MATLAB. After the train passed through the gate, the red LED on top of the gate would be switched on and the green LED on top the gate would be switched off.

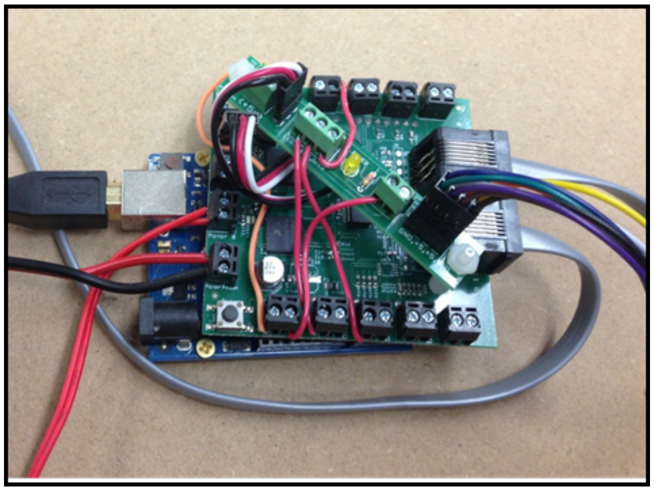


Figure 6 : Arduino Microcontroller

Figure 6 displays an open-sourced microcontroller, which served as the “brain” of the project. The Arduino Microcontroller had 12 digital and 6 analog pins available. It was programmed using a subset of C++, but students would be using a MATLAB program that used keywords like motorSpeed to execute commonly-used Arduino code segments.

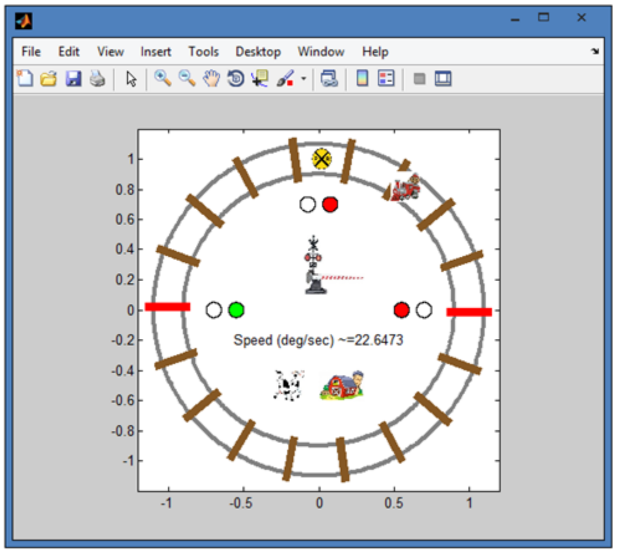

File Edit View Insert Tools Desktop Window Help
Speed (deg/sec) ~=22.6473
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

Figure 7 : Train Simulator

Figure 7 shows a working train simulator in action. Students could work on the train project in a computer lab or from home by setting up the train set simulator in conjunction with MATLAB. The simulator also demonstrated the train movement and location, the train speed, the break beam sensors, the flashing crossing lights, the crossing gate, and it was able to differentiate between the approach and departure sides of the track.

Serial Port which the Arduino is attached	4
Value at which the gate is vertical (10-170)	66
The first LED to turn on (left or right)	left (LED 14)

Table 1 : Settings of the Crossing Gate

Table 1 above shows the crossing gate settings. The serial port which the Arduino was attached to was number 4. The value which opened the gate vertically was 66 and the first LED to turn on was the left LED, which was port number 14.

	1st Value	**2nd Value**	**3rd Value**
Unobstructed Sensor values	64	64	64
Obstructed Sensor values	399	399	400

Table 2 : Unobstructed and Obstructed Break Beam Sensor Values

Table 2 above displays three unobstructed sensor values and three obstructed sensor values. The unobstructed sensor values were always consistent at 64. The obstructed sensor values were 399 for most of the time, but also read 400 sometimes. Although the obstructed sensor values were consistent the majority of the time, it was important to code in proper parameters that included all possible values.

motorSpeed (0-255)	Time for 1/2 circuit (sec)	Train Speed (mph)
255	**2.2545**	**0.8907**
230	**2.4081**	**0.8339**
200	**2.6374**	**0.7614**
190	**2.8255**	**0.7107**
170	**3.2165**	**0.6243**

Table 3 : Speed of the train (mph) and Time for Half Circuit at a Certain Motor Speed

Table 3 above shows the speed of the train in miles per hour (mph) and time taken for a half circuit at a certain motor speed in seconds (s). As the motor speed decreased, the time taken for the half circuit increased, with the train speed decreasing as well.

Executive Summary

The *Train Project Lab* allowed students to work in a team to produce a quality software package coded in MATLAB that automates control of a railway system that could be sold to different cities. The software produced in this lab is able to handle the train in both urban and rural environments and allows the client of the software to select which environment to run. This gave students experience in software development and experience in developing an application for a client able to handle the specified criteria. It also gave students experience working in a software development team with multiple developers coming together to produce a final product and allowed students to observe their code in action using the Arduino Microcontroller to visually observe the code instructions being executed.

Train Lab A explored the different MATLAB commands to control the railway system. Students collected data that would be critical to completing *Train Lab B*. Students connected the Arduino to MATLAB and observed to which port it was attached. Once the Arduino was attached, students began testing different settings of the crossing gate to determine which number allowed the gate to enter its vertical position. *Table 1* (page 7) displays this data. The Arduino was attached to Serial Port 4. It was determined that the integer value of "66" was the position where the gate was vertical, and LED 14 was the LED on the left. Students then began testing the gate values to determine what the obstructed values and unobstructed values were. These values are displayed in *Table 2* (page 7). It was determined that each time the sensor was unobstructed, the value was the integer 64. The obstructed sensor values differed, but were always either the integers 399 or 400. The students then tested the train motor speed and the time taken to move halfway around the tracks at a certain speed. These values are displayed in *Table 3* (page 8). It was determined that as the motor speed decreased, the time to complete the half circuit increased.

Train Lab B consisted of students working together to complete the project as outlined. In this lab students worked as a team to express ideas and discuss algorithms of how to automate the railway system. Before the lab, students created pseudocode of their ideas of how to potentially solve the automation problem and were given the chance to start implementing it in the lab. Students used the information collected in *Train Lab A* to help solve this problem as students had already figured out how to move the train, which LEDs corresponded to which LED number, which values properly moved the gate to the open or closed position, and which values were the obstructed or unobstructed values on the gate sensors. *Train Lab C* gave students a chance to demonstrate their final code. Students had 40 minutes to make adjustments to their code or work on extra credit assignments. After the 40 minute problem solving session, the instructional staff viewed and tested each team's project individually.

There are many sources of potential error in this lab. One source of potential error is if the gate sensors were either delayed or did not work all together. In *Train Lab C* the code that Team A wrote ran perfectly on the Train Simulator but many times the train failed to be detected in the departure gate when running on the Arduino. This problem was believed to be solved by saving the value of the departure gate each check and not check the value again in the "if" statement to get rid of the chance the train might leave the departure gate and the value gets set back to unobstructed before the departure code had the chance to run. This seemed to have fixed the problem as the train ran fine when tested again on the Arduino, but once again failed to work upon demonstration. However, the code worked fine on the simulator during the demonstration, which indicates that the departure gate sensor either did not return the correct values or was failing to detect the train.

Introduction

A series of experiments were conducted in the *Train Project Lab* to solve a real-world problem by using MATLAB programming language and applying several effective problem-solving strategies in a team setting. In order to achieve the main purpose, engineering students had worked together to familiarize themselves with the Arduino microcontroller and had learned several fundamental Arduino commands to obtain feedback data from sensors with MATLAB. This *Train Project Notebook* was created to address the importance of programming skills to solve real-world engineering problems. The specifics of this case were to create a software package that could control a railroad crossing gate system and could meet the client's particular requirements for a train operating system by using MATLAB, a miniature train set, and the Arduino microcontroller. This software package was able to adjust the delay time of closing the railroad crossing gate by prompting users to input a number to choose between urban and rural communities. If users chose rural for the community setting, there would be a longer warning time between flashing lights and the crossing gate moving down because cars travelled faster in rural areas than in dense urban areas. This *Train Project Notebook* contains list of figures and results in table form, program descriptions, discussion, conclusion, full MATLAB code, initial pseudocode, final pseudocode, and final flowchart.

Train Project Lab was divided into three sections, *Train Project Lab 7A, Train Project Lab 7B,* and *Train Project Lab 7C*. In *Train Project Lab 7A*, a group of engineering students were required to complete 8 tasks to familiarize themselves with the Arduino microcontroller. In Task 1 and Task 2, students learned to connect the Arduino controller to the MATLAB program. Students used one of the Arduino commands which was "team_Advance = arduino('COM4')" and then MATLAB would establish a connection to the Arduino microcontroller. Next, the objectives of Task 2 were to establish a connection between the Arduino microcontroller and the servo by executing the command "team_Advance.servoAttach(1)" to check the servo connection status by coding the command "team_Advance.servoStatus" and to control the servo by using the command "team.Advance_servoWrite(1,170)".

In Task 3, students were required to set up the LEDs by executing the command "team_Advance.pinMode" to attach LEDs to the Arduino microcontroller and the command "team_Advance.digitalWrite" to switch on or switch off LEDs. Furthermore, Task 4 required engineering students to enable the miniature train to move, controlling the motion of trains by using the command "team_Advance.motorRun". Students could make the train move forward, backward, and stop by typing 'forward,' 'backward,' and 'release' for the second argument in the parenthesis of the "team_Advance.motorRun" command. Besides that, the speed of the train could be controlled by executing the command "team_advance.motorSpeed".

In Task 5, students learned to set up the break beam sensors of the departure gate and the approach gate by using the command "team_Advance.analogRead" multiple times to obtain accurate obstructed and unobstructed values. Students needed to add an argument value which was the analog port number in the parenthesis of the "team_Advance.analogRead" command to enable MATLAB to interpret the values from either approach gate or departure gate. Tasks 6 and 7 required the writing of a control program in a script file that included group name, individual name, class information, a "clear all" command, a “close all” command, a “clc” command, and a “delete(instrfindall)” command and started the program with an infinite while-loop to run the program non-stop. In Task 7, students used the “tic” and “toc” commands to record the time taken to complete half of the track circuit and thereby calculate the speed of the train. In Task 8, students were required to make sure the lab was cleaned up and to stop the moving train by executing the command “team_Advance.motorSpeed(1,0)”.

In *Train Project Lab 7B* and *7C*, the students coded a quality and effective solution to meet the client’s particular requirements for a train operating system by demonstrating it with a miniature train set, the Arduino microcontroller, and MATLAB. The objectives of *Train Project Lab 7B* and *7C* were to decrease the speed of the train, to make the crossing gate’s LEDs flash alternately and close the crossing gate after a delay when the break beam sensors detected the train at the approach gate. When the break beam sensors detected the train at the departure gate, the speed of the train would increase; LEDs of the crossing gates would switch off, and the crossing gate would open vertically. Lastly, the train and all the aforementioned commands were required to run repeatedly without any unexpected errors.

Program Description for Developers

List of MATLAB General Commands

● clear all	- Clears all variables.
● close all	- Closes all figures.
● clc	- Clears the command window.
● delete (instrfindall)	- Clears any previous Arduino connections.
● tic	- Starts the stopwatch.
● toc	- Records the time from the stopwatch.

List of Arduino Commands

● team_Advance = arduino('COM4')	- Sets up a connection from the Arduino to MATLAB.
● team_Advance.servoAttach	- Sets up a connection to the crossing gate's servo motor.
● team_Advance.servoWrite	- Sets up a usable range of angle of the crossing gate which is between 10 and 170.
● team_Advance.motorRun	- Sets up the train's direction which can be forward, reverse or stop.
● team_Advance.motorSpeed	- Sets up the train's speed between 170 and 255, and 0 to stop the train.
● team_Advance.pinMode	- Establishes a connection to the LED crossing gate lights, LED departure gate lights, and the LED approach gate lights.
● team_Advance.digitalWrite	- Switches on and off the LED lights.
● team_Advance.analogRead	- Reads the values of break beam sensors from departure gate and approach gate.

List of All Variables

● right_LED = 15	- The variable is the crossing gate right LED, and its digital pin is 15.
● left_LED = 14	- The variable is the crossing gate left LED, and its digital pin is 14.
● red_Approach_LED = 6	- The variable is the approach gate red LED, and its digital pin is 6.
● green_Approach_LED = 7	- The variable is the approach gate green LED, and its digital pin is 7.
● red_Departure_LED = 8	- The variable is the departure gate red LED, and its digital pin is 8.
● green_Departure_LED = 7	- The variable is the departure gate green LED, and its digital pin is 9

● approach_Gate = 2	- The variable is the approach gate, and its port number is 2.
● departure_Gate = 3	- The variable is the departure gate, and its port number is 3.
● train_Minimum_Speed = 170	- The variable is the train minimum speed and its value is 170.
● train_Maximum_Speed = 255	- The variable is the train maximum speed and its value is 255.
● community = input ('Please type 1 for Urban or 2 for Rural:\n')	- Prompt user to input an integer 1 or 2 to choose between Urban and Rural.
● right_Flash = 0.75	- Sets the time value, 0.75 to delay the flashes of crossing gate right LED.
● left_Flash = 1.25	- Sets the time value, 1.25 to delay the flashes of crossing gate left LED.
● timeline = toc	- The variable, timeline will be equal to the value of the toc, which is the time taken of the train travelled for upper half of the track.
● right_Flash = right_Flash + 1	- Override the previous interval of the righ_LED flashes by adding 1.
● left_Flash = left_Flash + 1	- Override the previous interval of the left_LED flashes by adding 1.

At the beginning of MATLAB Code, "clear all", "close all", "clc", and "delete (instrfindall)" were coded to clear all variables in the Workspace, to close all figures, to clear all commands in the Command Window, and to disconnect any previous Arduino connections. Next, the crossing gate's right LED, the crossing gate's left LED, the approach gate's red LED, the approach gate's green LED, the departure gate's red LED, the departure gate's green LED, the approach gate, and the departure gate variables were assigned to their digital pin and port number value which were 15, 14, 6, 7, 8, 9, 2, and 3, respectively. Moreover, variables of the minimum and maximum speed of the train were assigned values of 170 and 255, respectively.

After all the 10 variables were initialized to a value as mentioned above, there were a few commands coded to connect all devices, LED lights, and sensors. The command "team_Advance = arduino('COM4')" was coded to set up a connection from the Arduino Microcontroller to MATLAB and the command "team_Advance.servoAttach(1)" was also coded to attach the servo motor and let MATLAB control the crossing gate. In order to connect the red LED of approach gate, the green LED of approach gate, the red LED of departure gate, the green LED of departure gate, the right LED of crossing gate, and the left LED of crossing gate to their pin number or port number, the command "team_Advance.pinMode(#, 'output')" was coded. Besides that, the command “team_Advance.digitalWrite” was coded to turn on the red LED of the departure gate and the green LED of the approach gate, and to switch off the red LED of the approach gate and the green LED of the departure gate. Lastly, the crossing gate was opened vertically by using the command “team_Advance.servoWrite(1,66)”.

Before making the train run, the variable “community” was used to prompt the user to input a value of 1 or 2 to choose between the urban area’s test run and the rural area’s test run, respectively. After the variable “community” was assigned to a value of 1 or 2, the train would start to move in a forward direction with maximum speed by using the “team_Advance.motorRun(1, ‘forward’)” and “team_Advance.motorSpeed(1, train_Maximum_Speed)” commands. Then, the statement “while 1” was executed to make the program run into an infinite while-loop. In the while-loop, a for-loop statement was used to read the values of the approach gate for 5 times to ensure an accurate reading was recorded by using the “team_Advance.analogRead(approach_Gate)” command. If the value of the approach gate’s break beam sensor or “team_Advance.analogRead(approach_Gate) was greater than 250, the timer would be started by using the command “tic”. Furthermore, “right_Flash” and “left_Flash” interval variables would be initialized to values which were 0.75 and 1.25, respectively and the speed of train would also be changed to the minimum speed by executing the“team_Advance.motorSpeed(1, train_Minimum_Speed)” command.

Next, a for-loop statement was used again to read the values of the departure gate for 5 times to ensure an accurate reading was recorded by using the “team_Advance.analogRead(departure_Gate)” command. Then, a while-loop was used once again after the for-loop in the if-statement. This statement was that while the value of the departure gate’s sensor or “team_Advance.analogRead(departure_Gate) was less than 250, the commands in the while-loop would keep on executing until the conditional statement was not met. Under this while-loop, the variable “timeline” was assigned to the value of “toc,” and the for-loop was used to read the values of the departure gate for 5 times to ensure an accurate reading was recorded by using the “team_Advance.analogRead(departure_Gate)” command. Moreover, the red LED of the approach gate and the green LED of the departure gate were turned on, and the red LED of the departure gate and the green LED of the approach gate were turned off by using the “team_Advance.digitalWrite(red_Approach_LED,1)”, “team_Advance.digitalWrite(green_Departure_LED,1)”, “team_Advance.digitalWrite(red_Departure_LED,0)”, and “team_Advance.digitalWrite(green_Approach_LED,0)” commands.

Furthermore, an if-statement was used to decide the delay time for closing the crossing gate. If the value of the variable “community” was 1 (the urban test run), the delay time for closing the gate would be 1 second. Therefore, if the statement was used in the if-statement to close the crossing gate by using “if timeline >= 1” and “team_Advance.servoWrite(1,170)”; else if the value of the variable “community” was 2 (the rural test run), the delay time for closing the gate would be 0.5 seconds. Hence, the if statement was used again in the elseif statement to close the crossing gate by using “if timeline >= 0.5” and “team_Advance.servoWrite(1,170)”. Next, another if-statement was coded to make the crossing gate’s LEDs flash alternatively. If the “timeline” variable was greater than the assigned value for the “right_Flash” variable (which was the right LED interval), the right LED of the crossing gate would turn on by using the “team_Advance.digitalWrite(right_LED,1)” command; at the same time, the left LED of the crossing gate would remain off by using the “team_Advance.digitalWrite(left_LED,0)” command. Then, the right LED interval was incremented by one to set it equal to the new right LED interval. Using an ‘elseif’ statement, if the “timeline” variable was greater than the assigned value for the “left_Flash” variable (which was the left LED interval), the left LED of the crossing gate would turn on by using the “team_Advance.digitalWrite(left_LED,1)” command; at the same time, the right LED of the crossing gate would remain off by using the “team_Advance.digitalWrite(right_LED,0)” command. Then, the left LED interval was incremented by one to set it equal to the new left LED interval. After that, the if-else statement and while-loop were terminated.

Last but not least, another if-statement was coded for the value of the departure gate sensor reading which was “team_Advance.analogRead(departure_Gate)”. If the value of the departure gate was greater than 250, the time of the stopwatch would be reset by using the “tic” command; both crossing gate’s LEDs would be switched off by using the “team_Advance.digitalWrite” command; the speed of the train would be set to the maximum by using the “team_Advance.motorSpeed(1, train_Maximum_Speed)” command; and lastly the gate would be opened vertically by using the “team_Advance.servoWrite(1, 66)” command. Finally, the if-statement and the infinite while-loop were ended by using the “end” statement.

Program Description for General Users

The first couple lines of code clear anything left over from previously run programs and also clear the input window of all previous commands. Then a connection between MATLAB and the Microcontroller was established. The Microcontroller is where the MATLAB commands are sent; in other words, it is what understands the MATLAB code and gives the train instructions. Then, all initial setup was completed including moving the gate to its starting position. The user was then asked if they would like to run the train in an urban or rural environment. Once the user responded to this question, the train's speed was set to its initial value, and the train moved forward. The program then entered a loop that ran until the user closed the program. The program would initially check the approach gate sensor until the train passed through it. When the train was detected to pass through the approach sensor, the train's speed was set to the minimum speed. After a small delay determined by whether the train was running in a rural or urban environment, the lights began flashing and after another small delay (also determined by the environment in which the train was running), the gate closed. During this time, the program is also checking the departure gate sensor to determine when it should run the code for when the train has passed that gate. When the train was detected to pass through the departure gate, the lights would stop flashing, the gate would move back up, and the train's speed would be set to the maximum train speed. The train would then start checking the approach gate sensor to determine when the train has passed the approach gate. This completes the loop for the program. The program will continue to run in this sequence, checking the two gates and running the specified code until the user decides to quit the program.

Discussion

When running the code for the train, everything did not work perfectly. After the fifth time of the train going around the track, the gate would no longer go down when passing through the approach gate. The source of this error is still unidentified since the code worked on the simulator. It was suspected that the sensor missed when the train passed through and therefore did not go through the loop that put down the gate and flashed the lights. This was modified to so that the gate saved what was checked last rather just in case it missed it. This code worked for a while but eventually the gate would stop going down again. Since the code worked on the simulator it was thought that there was possibly something wrong with the sensor and that it was not working correctly. The group was also asked to change the timing so that the lights starting flashing before the gate went down. To do this, the delay time for the lights was reduced.

To start off, the group was given a list of commands to run to connect the Arduino board, the train, the crossing gate, and the lights. This is the first part of the MATLAB code. The original code had two sections, one for the rural setting and one for the urban setting. This made the code very long. Later, the code was changed and a loop was added so that the code was far less repetitive. The main obstacle that was faced was when the crossing gate stopped going down as discussed above.

Conclusion

The *Train Project Lab* required the members of the group to develop a software package using MATLAB that met all the needs of the client. The task was to create a code to control a train as well as the crossing gates. The code was required to work for rural and urban settings. In *Train Project Lab 7A*, the group became familiar with the commands used to write the code. This included moving the train at different speeds as well as stopping the train, moving the crossing gate, and turning the crossing gates on and off. In *Train Project Lab 7B* and *7C,* the group worked on the code that would meet the client's requirements. This included the slowing the train down in the urban setting, speeding it up in the rural setting, the crossing gate coming down at the appropriate time, and the lights on the crossing gate flashing at the appropriate time.

It initially proved difficult to write a code that worked perfectly. A working code was written initially but was later revised to make it more efficient as well as shorter. Previous knowledge of common looping was used to write the code. An infinite while-loop was used to ensure the code could run continually. For-loops were used to read the break beam sensors. If-elseif loops were used in the for-loops to tell the train, lights, and crossing gate what to do. A trial-error method was used to fix any bugs found. Problems were identified and were then deduced to figure out why they were happening. Then, a line of code that could possibly fix the problem was written and then tested. When running the code on the simulation, it worked correctly. However, when it was run on the mini train set, sometimes the break beam sensors did not pick up that the train had passed through. This indicated that there could have possibly been a mechanical problem with the sensors.

This lab taught the group how to use the teamwork skills that were built up over the course of the semester to solve a real-world problem. Problem-solving skills and knowledge of MATLAB and loops were used to produce the final software package. Effective communication was one of the most important skills required when working on this project. The effective communication allowed the group to write code that worked correctly, in the time given. Along with meeting the deadline, effective communication allowed the group to produce a software package that fulfilled the requirements.

Appendix A : Full MATLAB Code (with comments)

```
%Clear all variables, command window, and previous arduino connections.
clear all;
close all;
clc;
delete(instrfindall);

% Initialization---------------------------------------------------------

% Initialize the variable right_LED to 15.
right_LED = 15;

% Initialize the variable left_LED to 14.
left_LED = 14;

% Initialize the variable red_Approach_LED to 6.
red_Approach_LED = 6;

% Initialize the variable green_Approach_LED to 7.
green_Approach_LED = 7;

% Initialize the variable red_Departure_LED to 8.
red_Departure_LED = 8;

% Initialize the variable green_Departure_LED to 9.
green_Departure_LED = 9;

% Initialize the variable approach_Gate to 2.
approach_Gate = 2;

% Initialize the variable departure_Gate to 3.
departure_Gate = 3;

% Initialize the variable train_Minimum_Speed to 170;
train_Minimum_Speed = 170;

% Initialize the variable train_Maximum_Speed to 255;
train_Maximum_Speed = 255;

% Attach and connect all the devices and sensors.--------------------------

% Connect the Arduino to MATLAB.
team_Advance = arduino('COM4');

% Attach the servo and the Arduino.
team_Advance.servoAttach(1);
```

```
% Attach the red LED of approach gate to digital pin 6.
team_Advance.pinMode(red_Approach_LED,'output');

% Set the red LED of approach gate as default which is off.
team_Advance.digitalWrite(red_Approach_LED,0);

% Attach the green LED of approach gate to digital pin 7.
team_Advance.pinMode(green_Approach_LED,'output');

% Set the green LED of approach gate as default which is on.
team_Advance.digitalWrite(green_Approach_LED,1);

% Attach the red LED of departure gate to digital pin 8.
team_Advance.pinMode(red_Departure_LED,'output');

% Set the red LED of departure gate as default which is off.
team_Advance.digitalWrite(red_Departure_LED,1);

% Attach the green LED of departure gate to digital pin 9.
team_Advance.pinMode(green_Departure_LED,'output');

% Set the green LED of departure gate as default which is on.
team_Advance.digitalWrite(green_Departure_LED,0);

% Attach the left LED to digital pin 14.
team_Advance.pinMode(left_LED,'output')

% Attach the right LED to digital pin 15.
team_Advance.pinMode(right_LED,'output')

% -------------------------------------------------------------------------

% Make sure the crossing gate was open when the train is located at the
train station.

% Control the servo and open the crossing gate vertically.
team_Advance.servoWrite(1,66);

% -------------------------------------------------------------------------

% Prompt user to choose between urban and rural.
community = input('Please type 1 for Urban or 2 for Rural:\n');

% -------------------------------------------------------------------------
```

```
    % Use if-else selection statement to run the train in different
      community.
    % Enable the train to move forward.
    team_Advance.motorRun(1,'forward');

    % Set the motor speed to train initial speed which is 255.
    team_Advance.motorSpeed(1,train_Maximum_Speed);


% Use infinite while-loop to execute the program infinite times.
while 1

      % Receive the current status of the approach break beam
      % sensors 5 times.

      for i = 1 : 5
         team_Advance.analogRead(approach_Gate);
      end

      % If the approach break beam sensor was interrupted.
      if team_Advance.analogRead(approach_Gate) > 250

      % Start stopwatch.
      tic;


            % Initialize the variable right_Flash to 0.25.
            right_Flash = 0.25;
            % Initialize the variable left_Flash to 0.75.
            left_Flash = 0.75;


      % Set the motor speed to train minimum speed which is 170.
        team_Advance.motorSpeed(1,train_Minimum_Speed);

      % Receive the current status of the departure break beam
      % sensors for 5 times.
       for i = 1 : 5
            team_Advance.analogRead(departure_Gate);
      end


      % While the train still not yet arrive at the departure gate.
      while team_Advance.analogRead(departure_Gate) < 250

            % Assign timeline to toc to update the timeline.
            timeline = toc;
```

```
% Receive the current status of the departure break beam sensors
5 times.
for i = 1 : 5
    team_Advance.analogRead(departure_Gate);
end

% Switch on both red_Approach_LED and green_Departure_LED, and
% switch off both green_Approach_LED and red_Departure_LED.
team_Advance.digitalWrite(red_Approach_LED,1);
team_Advance.digitalWrite(green_Approach_LED,0);
team_Advance.digitalWrite(red_Departure_LED,0);
team_Advance.digitalWrite(green_Departure_LED,1);

if community == 1
      % Delay on closing the crossing gate for 1 second by
        using if statement.
      if timeline >= 1
            % Close the crossing gate horizontally.
            team_Advance.servoWrite(1,170);
      end

else if community == 2
    % Delay on closing the crossing gate for 0.5 second by using
      if statement.
      if timeline >= 0.5
            % Close the crossing gate horizontally.
            team_Advance.servoWrite(1,170);
      end

      end
end

% Use if-else statement in while-loop to make both LEDs flash
% alternatively.
if timeline >= right_Flash

    % Switch on right LED.
    team_Advance.digitalWrite(right_LED,1);

    % Switch off left LED.
    team_Advance.digitalWrite(left_LED,0);

    % Increment the right_Flash by 1.
    right_Flash = right_Flash + 1;
```

```
            elseif timeline >= left_Flash

                % Switch on left LED.
                team_Advance.digitalWrite(left_LED,1);

                % Switch off right LED.
                team_Advance.digitalWrite(right_LED,0);

                % Increment the left_Flash by 1.
                left_Flash = left_Flash + 1;
            end
      end
      end

      % If the train passes the departure gate.
      if team_Advance.analogRead(departure_Gate) > 250

      % Switch off the right LED.
      team_Advance.digitalWrite(right_LED,0);

      % Switch off the left LED.
      team_Advance.digitalWrite(left_LED,0);

      % Open the crossing gate.
      team_Advance.servoWrite(1,66);

      % Set the train speed to maximum.
      team_Advance.motorSpeed(1,train_Maximum_Speed);

      % Reset the stopwatch.
      tic;

      % Switch off both red_Approach_LED and green_Departure_LED, and
      % switch on both green_Approach_LED and red_Departure_LED.
        team_Advance.digitalWrite(red_Approach_LED,0);
        team_Advance.digitalWrite(green_Approach_LED,1);
        team_Advance.digitalWrite(red_Departure_LED,1);
        team_Advance.digitalWrite(green_Departure_LED,0);
      end
end
```

Appendix B : Final Testing Checklist

Group : Team A Instructor : Dr. Bixler and Ms. Fernandez (GTA)

Class Time : 2:20 p.m. Date: 10th April 2017

This sheet must be filled out and signed by a member of the Instructional Staff by the end of Lab C. The Instructor/TA must watch the program be initiated through MATLAB and will record the results below.

At approach sensor:		**Yes**	**No**	**Points Earned (0-4)**	**Comments**
Train speed decreases					
LEDs begin to flash (alternately)					
Gate Routine	Closes after a delay				
	"Rural" delay				
	"Urban" delay				
			Points Subtotal	**/ 20**	
At departure sensor:		**Yes**	**No**	**Points Earned (0-4)**	**Comments**
Train speed increases					
LEDs turn off					
Gate opens					
Runs repeatedly/indefinitely, without any new errors					
			Points Subtotal	**/ 16**	
			Total Points Earned	**/ 36**	
General Comments:					

Instructor/TA's Signature: ______________________ Date: __________

Appendix C : Initial Pseudocode

```
Initialize all variables

Set the train initial speed
Move the train forward

Use a while loop to execute the program an infinite number of times

    Detect whether the train is in one of the gates and which gate it is in if so

    Use If statement to run code if the train is in or has passed the approach gate
        Start the stopwatch
        Set the train speed to the minimum train speed
        Flash the gate lights after a small delay (determined by stopwatch)
        Close the gate after another delay/after the lights have flashed and before the train
                reaches the gate

    Use else if to run code if the train is in or has passed the departure gate
        Set the train speed to the maximum train speed
        Stop the flashing lights
        Open the gate
        Stop and reset the stopwatch
        end else statement for the departure gate

    end if block for the gates

end and restart infinite while loop to run the checks again
```

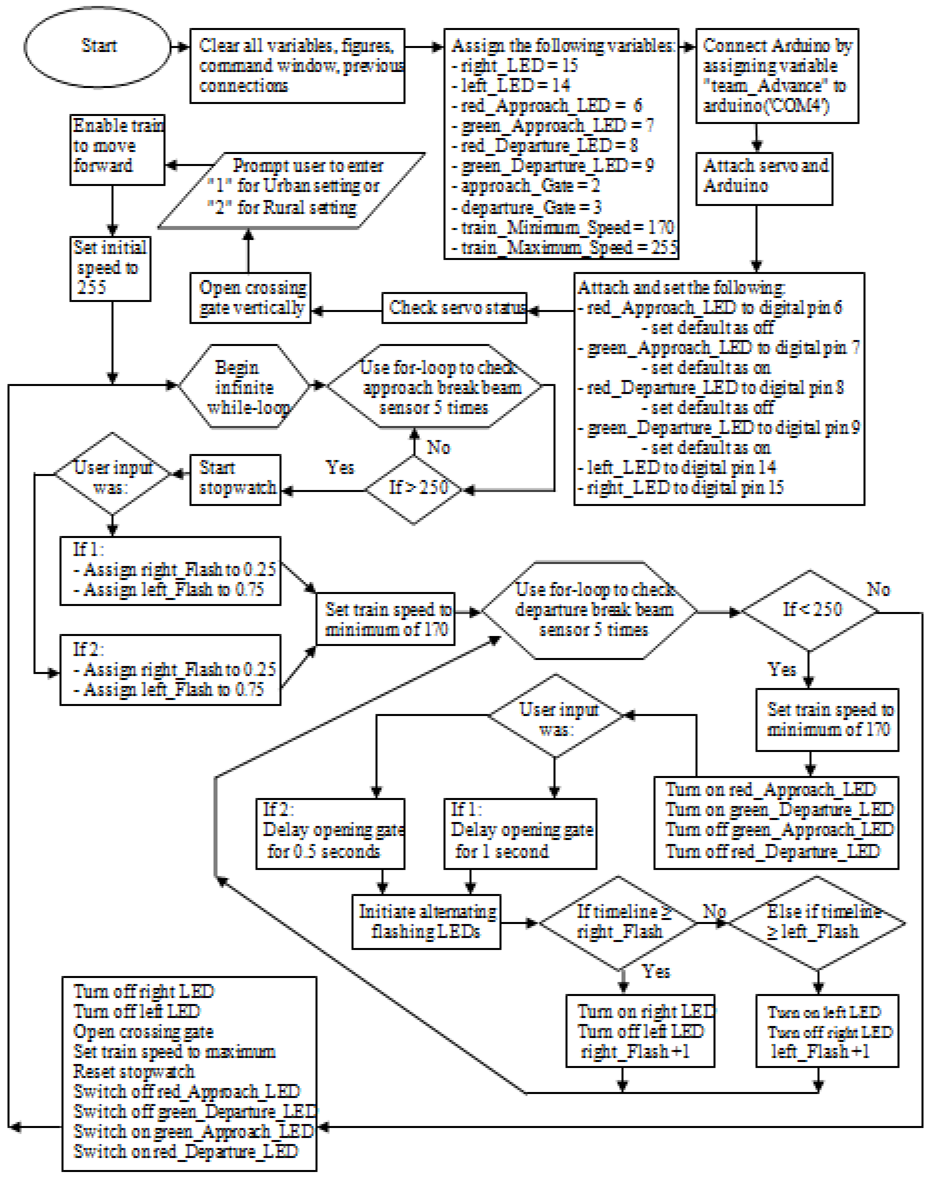
Appendix D : Final Pseudocode

```
Clear all variables.
Close all figures.
Clear the command window.
Clear any previous arduino connections.

Assign the variable, right_LED to the port number of right LED which is 15.
Assign the variable, left_LED to the port number of left LED which is 14.
Assign the variable, red_Approach_LED to the port number of red LED in approach gate which is 6.
Assign the variable, green_Approach_LED to the port number of green LED in approach gate which is 7.
Assign the variable, red_Departure_LED to the port number of red LED in departure  gate which is 8.
Assign the variable, green_Departure_LED to the port number of green LED in departure gate which is 9.
Assign the variable, approach_Gate to the port number of approach gate which is 2.
Assign the variable, departure_Gate to the port number of departure gate which is 3.
Assign the variable, train_Minimum_Speed to 170.
Assign the variable, train_Maximum_Speed to 255.

Attempt a connection with Arduino by assigning a variable, team_Advance to arduino(‘COM4’).
Attach the servo and the Arduino.
Attach the red LED of approach gate to digital pin 6 which is also the variable, red_Approach_LED.
Set the red LED of approach gate as default which is off.
Attach the green LED of approach gate to digital pin 7 which is also the variable, green_Approach_LED.
Set the green LED of approach gate as default which is on.
Attach the red LED of departure gate to digital pin 8 which is also the variable, red_Departure_LED.
Set the red LED of departure gate as default which is off.
Attach the green LED of departure gate to digital pin 9 which is also the variable, green_Departure_LED.
Set the green LED of departure gate as default which is on.
Attach the left LED to digital pin 14 which is also a variable, left_LED.
Attach the right LED to digital pin 15 which is also a variable, right_LED.

Check the servo status.
Control the servo and open the crossing gate vertically.

Prompt user to type either number 1 for Urban or number 2 for Rural.

Enable the train to move forward.
Set the motor speed to train initial speed which is 255.
```

```
Use infinite while-loop to execute a program infinite times.

    Use for-loop to receive the current status of the approach break beam sensors 5 times.

    If the approach break beam sensor value is more than 250

        Start the stopwatch.

        If the user input is 1 which is urban

            Assign a variable, right_Flash to 0.25 which means the flashing right LED will be
            delay by 0.25 seconds.
            Assign a variable, left_Flash to 0.75 which means the flashing left LED will be
            delay by 0.75 seconds.

        Else if the user input is 2 which is rural.

            Assign a variable, right_Flash to 0.25 which means the flashing right LED will be
            delay by 0.25 seconds.
            Assign a variable, left_Flash to 0.75 which means the flashing left LED will be
            delay by 0.75 seconds.

          end if statement

        end if statement

        Set the motor speed to train minimum speed which is 170.

        Use for-loop to receive the current status of the departure break beam sensors for 5
        times.

        While the departure gate break beam sensor is still less than 250

            Set the motor speed to train minimum speed which is  170 again.

            Switch on both red_Approach_LED and green_Departure_LED.
            Switch off both green_Approach_LED and red_Departure_LED.
```

```
                        If the user input is 1 which is urban

                                Use if to delay on opening the crossing gate for 1 second.
                                end if statement.

                        Else if the user input is 2 which is rural

                                Use if to delay on opening the crossing gate for 0.5 seconds.
                                end if statement
                        end if statement
                        end if statement

                        Use if-else statement in while-loop to make both LED flash alternatively.
                        If the timeline is more than or equal to right_Flash

                                Switch on the right LED.
                                Switch off the left LED.
                                Increment the right_Flash by 1

                        Else if timeline is more than or equal to left_Flash.

                                Switch on the left LED.
                                Switch off the right LED.
                                Increment the left_Flash by 1

                        end if-else statement
                end while statement.
        end if statement.

        If the departure gate break beam sensor value is more than 250

                Switch off the right LED.
                Switch off the left LED.
                Open the crossing gate.
                Set the train speed to maximum.
                Reset the stopwatch.
                Switch off both red_Approach_LED and green_Departure_LED.
                Switch on both green_Approach_LED and red_Departure_LED.

        end if statement.
end infinite while statement
```

Appendix E : Final Flowchart



Start
Clear all variables, figures, command window, previous connections
Assign the following variables: - right_LED = 15 - left_LED = 14 - red_Approach_LED = 6 - green_Approach_LED = 7 - red_Departure_LED = 8 - green_Departure_LED = 9 - approach_Gate = 2 - departure_Gate = 3 - train_Minimum_Speed = 170 - train_Maximum_Speed = 255
Connect Arduino by assigning variable "team_Advance" to arduino('COM4')
Attach servo and Arduino
Attach and set the following: - red_Approach_LED to digital pin 6 - set default as off - green_Approach_LED to digital pin 7 - set default as on - red_Departure_LED to digital pin 8 - set default as off - green_Departure_LED to digital pin 9 - set default as on - left_LED to digital pin 14 - right_LED to digital pin 15
Check servo status
Open crossing gate vertically
Prompt user to enter "1" for Urban setting or "2" for Rural setting
Enable train to move forward
Set initial speed to 255
Begin infinite while-loop
Use for-loop to check approach break beam sensor 5 times
No
If > 250
Yes
Start stopwatch
User input was:
If 1: - Assign right_Flash to 0.25 - Assign left_Flash to 0.75
If 2: - Assign right_Flash to 0.25 - Assign left_Flash to 0.75
Set train speed to minimum of 170
Use for-loop to check departure break beam sensor 5 times
If < 250
No
Yes
Set train speed to minimum of 170
Turn on red_Approach_LED Turn on green_Departure_LED Turn off green_Approach_LED Turn off red_Departure_LED
User input was:
If 2: Delay opening gate for 0.5 seconds
If 1: Delay opening gate for 1 second
Initiate alternating flashing LEDs
If timeline ≥ right_Flash
No
Else if timeline ≥ left_Flash
Yes
Turn on right LED Turn off left LED right_Flash +1
Turn on left LED Turn off right LED left_Flash +1
Turn off right LED Turn off left LED Open crossing gate Set train speed to maximum Reset stopwatch Switch off red_Approach_LED Switch off green_Departure_LED Switch on green_Approach_LED Switch on red_Departure_LED

Lab Participation Agreement

Team Name	A
Date	24th April 2017
Lab Title	Train Project
Instructor	Dr. Bixler and Ms. Fernandez (GTA)

We as a team agree to have actively contributed towards the lab deliverables. We have used only approved materials and processes as documented in our course material. Furthermore, each team member has equally contributed to the analysis and documentation involved.

Team Member	Printed name	Duties performed during lab (brief)
	Signature	Duties performed outside class (brief)
1	Kai Chuen Tan	Write codes with MATLAB most of the time and familiarize with the train setup.
	Kai Chuen Tan	Write codes and completed part of the Notebook.
2	Stephen Doucet	Recorded data, assisted with coding and troubleshooting
	Stephen Doucet	Completed parts of the Notebook, proofreading and editing, flowchart
3	Annette Petty	Write and debug MATLAB code. Assisted with troubleshooting.
	Annette Petty	Optimized code. Completed and proofread part of notebook.
4	Frances Severding	Assisted with recording data. Read the lab procedure and kept group on task.
	Frances Severding	Completed part of the notebook.

Train Project Grading Guidelines (Project Notebook)

Section	Point Breakdown	Points Earned
Cover Page	**5**	
Table of Contents	**5**	
List of Figures and Tables	**5**	
Project Notebook Body	**110**	
Executive Summary: Follow the guidelines for executive summaries provided in the Technical Communications Guide to give a summary of the train project lab.	**25**	
Introduction: Clearly define the purpose of the train project lab, describing the objectives of each of the tasks.	**15**	
Program Description: Describe the program/code in words for developers and general users, separately.	**25**	
Discussion: Provide brief explanations of what occurred during the train's run including worksheet statistics. Describe the obstacles faced in writing the final code and how they were overcome.	**25**	
Conclusion: Develop a conclusion from the results obtained in the train project lab	**20**	
Writing Style	**25**	
Grammar: See Technical Communication Guide for guidelines	**10**	
Organization and Progression: Make sure the notebook has natural breaks, clearly labeled tabs, ideas are properly separated by paragraphs, and ideas are fluently connected.	**15**	
Appendix	**50**	
Provide the full MATLAB code, checklists, initial and final pseudocode, final flowchart and any other aspects of the results not covered in the Discussion section. Format the additional results in accordance with the Technical Communication Guide.	**45**	
Lab Participation Agreement	**5**	