CE-442 Mobile Robotics Laboratory Report #2

MATLAB and Simulink

Teleoperation with Keyboard Control and
Stoplight State of a Simulated Mobile Robot

Heidi Gwinner

Dylan Lozon

Quinn Austin

Kettering University
College of Engineering
Department of Electrical and Computer Engineering

CE-442: Mobile Robotics
Dr. Girma Tewolde

Apr 17, 2024

I. OBJECTIVES

This lab was primarily designed to reinforce the skills learned in the MATLAB, Simulink, and Stateflow onramp laboratory assignments. By utilizing all three of these tools, the two tasks verified whether the necessary skills were obtained to complete future labs.

Utilize MATLAB, Simulink, and Stateflow tools to solve the following problems:

1. Teleoperation to control a simulated mobile robot on a given map

   In this first task, we simulated the direct user control of a robot using MATLAB's graphing and user input detection capabilities. This task focused on the ability to effectively utilize MATLAB's interpreted programming language.

   - Allow a user to interactively control the movement of a mobile robot using the 'WASD' keys on a keyboard by creating a 10x10 map with a small red square object that represents a mobile robot.

2. Simulate a traffic light controller using Simulink and Stateflow

   In the second task, we simulated the states and logic required to make a stoplight function in two modes: Normal and Override. This task focused on the ability to utilize Simulink and Stateflow to achieve completion.

I. TASK ONE: Simulated Teleoperation Control

We used MATLAB Live Script to simulate a robot being controlled directly by a human operator. To achieve this simulation, we first defined the limits of the plot, the robot's initial position, and the robot's step size.

On the initial plot, we represent the robot using a red square and set the axes limits to one through ten.

The program then enters the main loop, which checks if the user has pressed one of four of the defined keys while focused on the figure window. When one of these four keys is pressed, the robot moves accordingly. If the robot has left its boundaries, it is forced back within them. We limit the maximum and minimum bounds of the robot's movements to one-tenth less than the limits of the graph. The graph is then updated and continues through the loop.

The initial state of the product can be seen below in Figure 1 and the final product has been demonstrated in this video.

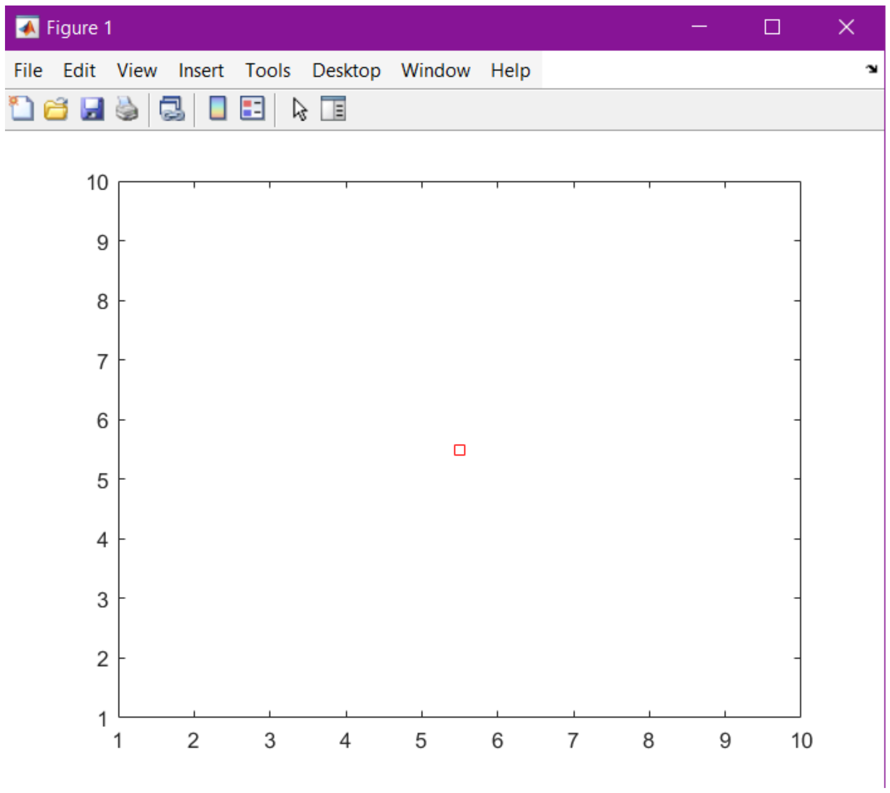

Figure 1: Initial state of the output graph for Task 1.

II. TASK TWO: Simulated Stoplight

This task required using Simulink and Stateflow to simulate a stoplight. We used a repeating sequence stair to generate the traffic lamp's mode, which serves as the system's input. Then, the input and each light's state is sent to a scope, which allows each signal to be viewed and verified.

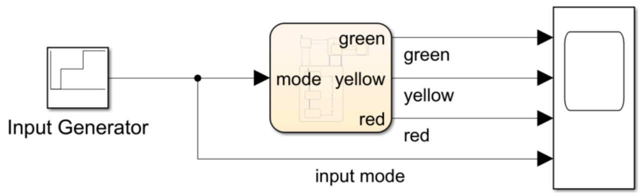

Figure 2: The Simulink block diagram.

The Stateflow chart, shown in Figure 3, takes a value from the input, 0, 1, or 2, and decides which state the stoplight is in: Off, Normal, or Override, respectively. The program's output can be seen in this video.

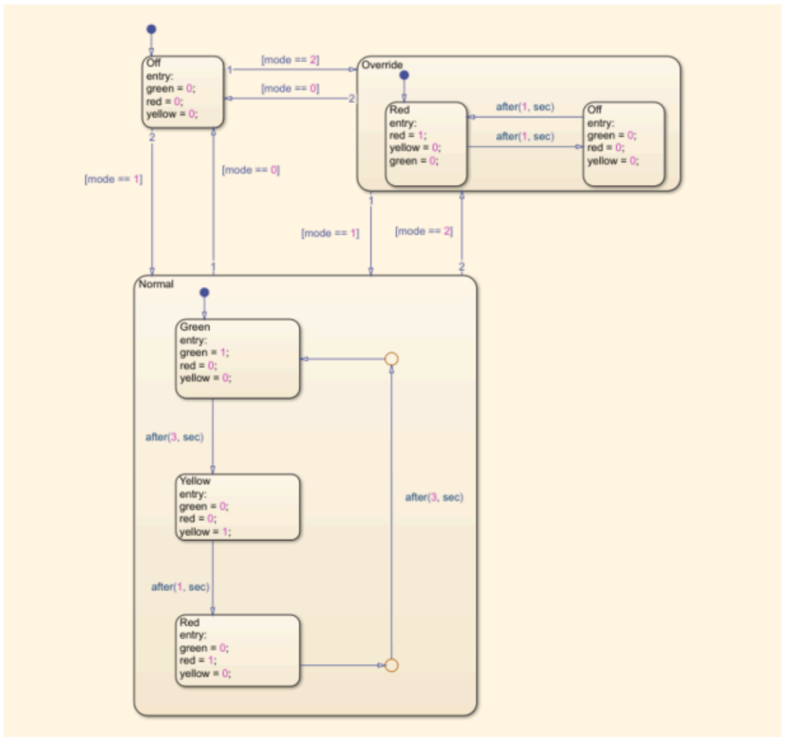

Figure 3: The Stateflow diagram.

III. CONCLUSIONS

This lab expanded on previously learned material from Lab 1. In the first task, MATLAB is used to create a simulated teleoperated robot. With task two, Simulink and Stateflow are utilized to simulate two possible modes of a traffic light. This combined tests that the basics of MATLAB, Simulink, and Stateflow are understood, ensuring that we can take on more difficult challenges in the future.

IV. SOURCE CODE

% Declare and Initialize Variables

```
minRobotX = 1;

maxRobotX = 10;


minRobotY = 1;

maxRobotY = 10;


robotX = mean([minRobotX, maxRobotX]);

robotY = mean([minRobotY, maxRobotY]);


step = .1; % How far the robot moves each frame
```

% Draw the initial plot

```
plot(robotX, robotY, "rs");

xlim([minRobotX, maxRobotX]);

ylim([minRobotY, maxRobotY]);
```

% Read user input and move the robot

```
while 1

   waitforbuttonpress;
```

```
input = get(gcf, 'CurrentCharacter');


% Update the position of the robot based on the input
switch input
    case 'w'
        robotY = robotY + step;
    case 'a'
        robotX = robotX - step;
    case 's'
        robotY = robotY - step;
    case 'd'
        robotX = robotX + step;
end % input switch statement
```

% Enforce the robot's boundaries

```
if (robotX <= minRobotX)
    robotX = minRobotX + step;
elseif (robotX >= maxRobotX)
    robotX = maxRobotX - step;
end


if (robotY <= minRobotY)
    robotY = minRobotY + step;
elseif (robotY >= maxRobotY)
    robotY = maxRobotY - step;
end
```

% Update the plot with the robot's position

```
plot(robotX, robotY, "rs");
```

```
   xlim([minRobotX, maxRobotX]);

   ylim([minRobotY, maxRobotY]);


end % program while loop
```