Project Spyn: Final Report

By Abhikya Ananth

Table of Contents

Summary:

A new ridesharing company, Spyn, is looking to gain a competitive edge through the exploration of new markets and the development of new products and services. Local to Arizona, the company has asked our team to design an autonomous vehicle that can help safely transport handicapped passengers. Over the past ten weeks, our engineering team has come up with a

design plan for a new car that fulfills their requirements. In the early stages of our project, we were able to make a model of the vehicle design and split production to make an assembly line consisting of four stations. We also identified the cost constraints and created a Gantt chart. After brainstorming and gathering requirements, we were able to adapt our design and implement it into a working wheel-chair accessible, autonomous vehicle that safely picks up and drops off a passenger. We were able to write a script that utilizes touch and color sensors through Matlab and run our program with Robosim. Using the Rocket model and the EasyFare maze in Robosim, we completed a total of six implementations and two milestones in order to finalize a program that enables the robot to navigate through the maze. In our final demonstration, we were able to execute our design with a successful run that looked as follows: the vehicle turned around so that the sensors are facing forward, then moved forward until it touched a wall, activating a touch sensor and prompting the robot to move slightly backwards and turn through the angleRel function. After turning the robot continued to move until it detected a color – either red, yellow, or green. The robot is programmed to adhere to traffic laws in which red prompts the robot to stop for two seconds, and yellow and green exit automatic control and switch to keyboard control so the user can safely pick up and drop off the passenger respectively.

Team Members and Contributions

Tiffany Ticlo: For the both milestones, I completed and submitted the documentation required for the implementation. I shared my screen during the lab so that we could work together in finishing our code. I think overall our team had shared efforts in completing the scripts, but specifically, I contributed with the left and right arrow implementations to the keyboard controls. In the maze navigation milestone, I worked on the beginning part of the code in which the program enters the first section of while loops. The robot turns around using the relative angle

motor function and moves forward – hitting the wall in front of it and activating the touch sensor. Once the touch sensor is activated, the robot moves backwards slightly and then turns using the relative angle function again and continues to move forward. When the robot touches the next wall, it moves backwards and turns right again until it reaches the red line where it stops for two seconds.

Sohan Rayabharapu: Regarding the keyboard control milestone, I contributed with the down and up arrow implementations to the keyboard controls. These both go in the switch statement, which takes key as a parameter and relies on user input to move. For example, when the user presses either the up or down key on their keyboard, the robot would move accordingly. More specifically, the down arrow would use motors A and B to manually move at a speed of -20, or moving it backwards, and same with the up arrow, except the speed of moving motors A and B would be at 20, and would move it forward. For the maze navigation implementation, I worked on the part of the code right after the robot stops at the red line and goes forward. This part involves the robot looking for the color yellow, which signals that the robot should go into manual mode, allowing for the user to manually pick up the passenger. After that, I made it so that after the user picks up the passenger, they just need to press q, and then it goes into automatic mode. It then goes on to find the red line, and pauses for 2 seconds after that, then moves towards the next wall.

Jared Guthrie: For the keyboard control milestone, I helped to define the numerical values used in the keyboard control code as well as controlled the robot itself in the final implementation. These two processes went very well together as when I tested it I could immediately alter the code to benefit the smoothness of the machine. For the maze navigation milestone, As the robot moves forward, it will eventually hit a wall, triggering one of the touch sensors, where it will then turn around, using the Angle Rel function, at a reduced speed so the passenger doesn't fall off. After that, it will turn angle A to the left, so that it will face the coming wall. Then I had it go to said wall, using similar while loops to what we had previously, waiting for one of the touch sensors to then activate.

Max Hested: In the keyboard control implementation, I tested out each numerical value and was able to decide on which numerical values were suitable for the program itself to run smoothly and safely. I was able to give this information to Tiffany on how to run the Robot by following all of the safety guidelines. I was also able to figure out the keyboard controls for picking up the passenger and send that information over to Tiffany as well. With the maze navigation milestone, I worked on the second half of the code. When the touch sensors activate, the robot will move backwards and to the left, similar to the two loops we had previously to turn to where the robot is now, only slightly adjusting the values to account for the inconsistencies in the robot hitting the wall. I then had the robot move forward, to go to the next wall.

Abhikya Anath: For the keyboard control milestone I contributed in the designing of the robot and initially helped in the functioning of the code. I also helped in the up and down arrow implementations and helped in checking the code completely and actively participated in making changes if there were any looking at the criteria provided. I also recorded the video for our demo. Overall as a team we have all out in equal efforts to make project spyn work and it was a team effort. I coded the final stretch of the program with the maze navigation, where as the robot turns left and heads toward the wall, it will then hit the wall near the green line, which will then trigger the touch sensor again, moving the robot back a little, then left, towards the green line. And, as the robot senses the green line with its color sensor, it will switch to manual mode, which then allows the user to drop off the passenger safely.

Requirements:

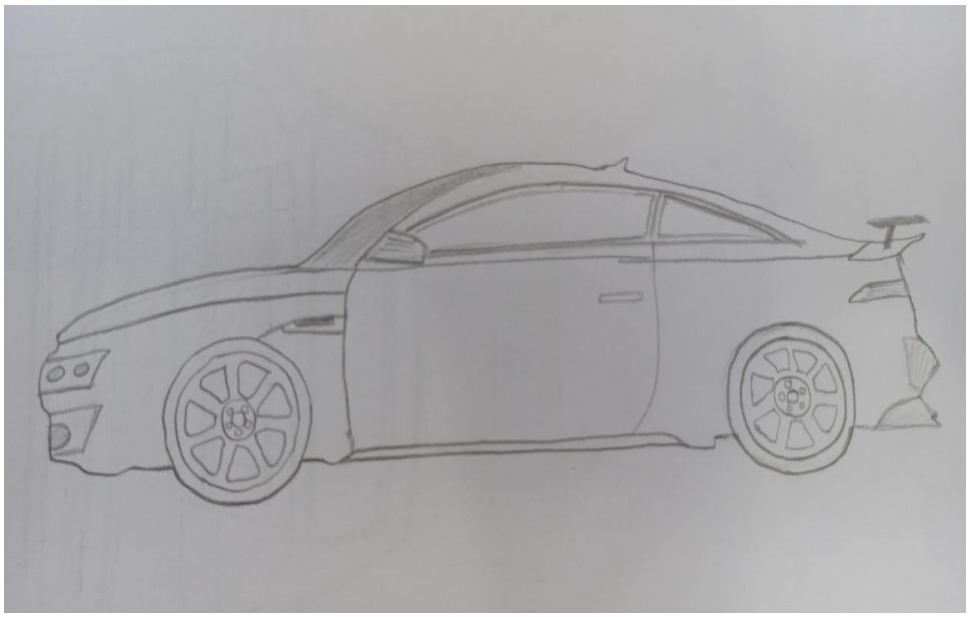
The project requirements were as follows (functional, performance, and cost constraints):

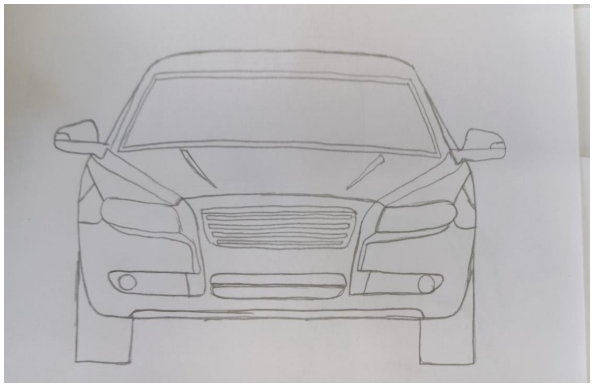
- The project must be completed across the ten weeks
- Car must be entirely automatic
- Cars will be electric, with energy levels monitored
- Through Spyn, a passenger can be picked up at any time of day and at any point on the road
- The pick-up location is yellow

- Drop-off location is green
- Robot must adhere to traffic laws
    - Stop at red for two seconds
    - Yellow must initiate keyboard control for pick-up
    - Green must initiate keyboard control for drop-off
- Required dimensions: typical wheelchair has a length of 42” (106.7 cm), a height of 36” (91.4 cm), seat heights around 19.5” (49.5 cm), and a width of 25” (63.5 cm)
- The minimum size of a standard parking space shall be nine feet wide and eighteen feet long
- There is no predetermined budget for the project; cost will be determined based on the weight of the vehicle
- Safety of the passenger is among top priority

Design Proposal:

Prototype Design 1:





Prototype design 1 description:

This prototype has a similar layout to a sedan. The vehicle features two doors through which a ramp will extend via sensors to allow easy entry and exit into the car for disabled passengers. Through a button on the app, the passenger can then turn on the moving ramp feature (similar to a treadmill mechanism). Inside the vehicle, there will be a large touch-screen that displays the passenger's route, features an emergency SOS button, shows battery levels, and allows passengers to confirm payment.

Prototype Design 2:

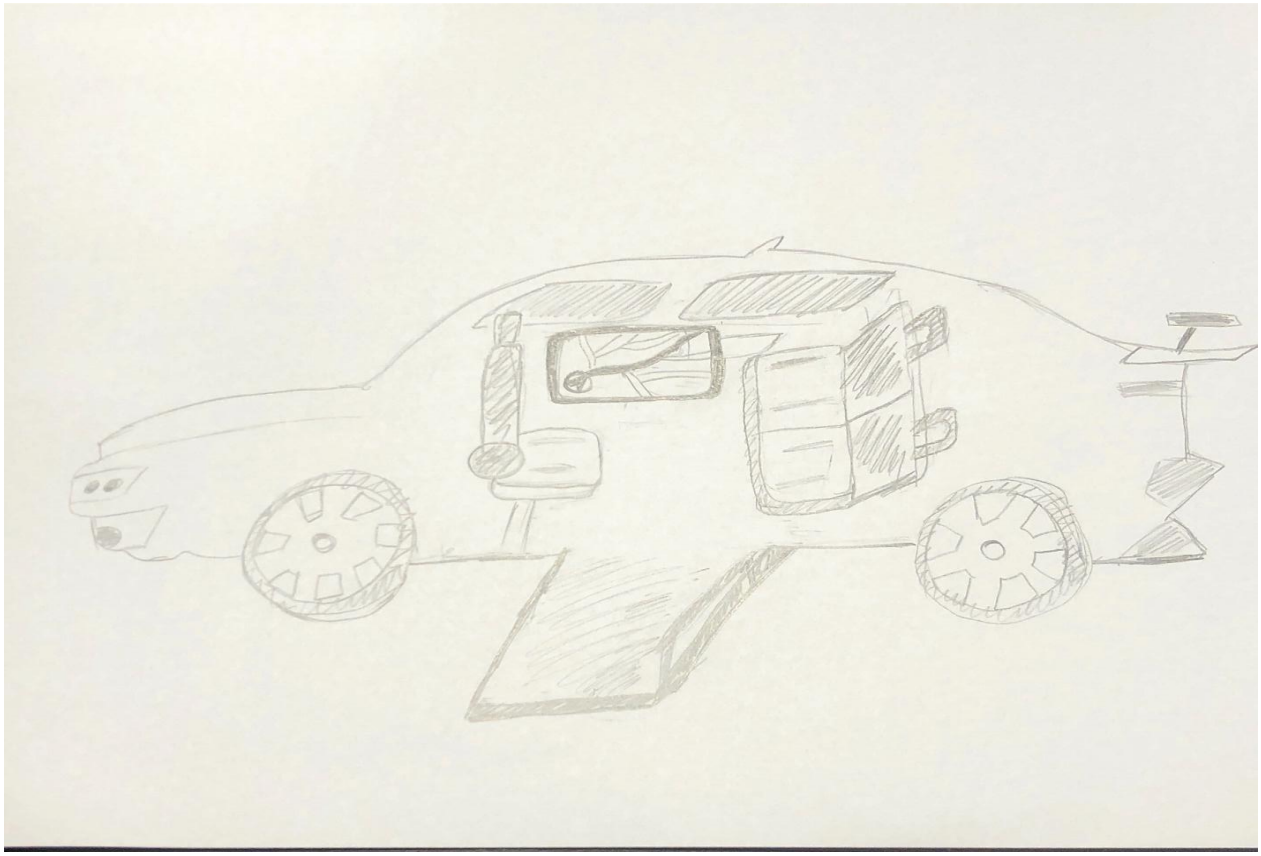
Prototype design 2 description:

The prototype we have designed has a van layout and is specially modified for people with wheelchairs and similar disabilities who may not be comfortable traveling in normal cars. For the purpose of easy on and off loading, we have installed a ramp that automatically extends via a sensor that detects if there is enough space for the ramp to be deployed. The car will also feature motion and color sensors. These sensors will be able to detect objects surrounding the car; and, the car will stop when detecting the color red and go when detecting the color green. This design is more spacious, but will cost more to produce.

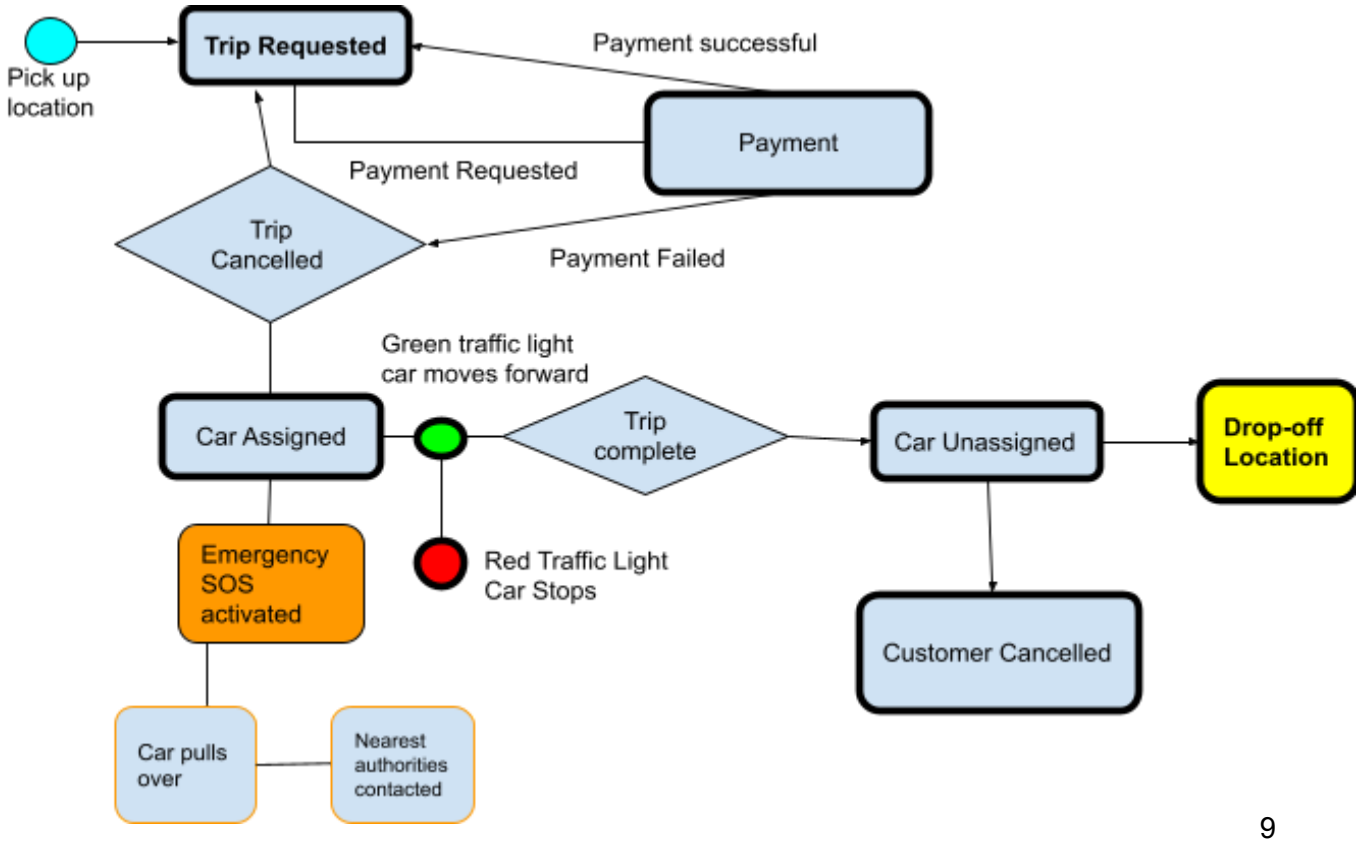
Design functionality:

The prototype we have selected (prototype 1) is a sedan-style vehicle that allows handicap and eldery passengers to ride safely and comfortably. With a large two door design, a ramp can easily be installed and deployed. The ramp also moves through the app – similar to a treadmill. This makes the vehicle wheelchair accessible and provides easy on and off loading. Inside the vehicle, there will be a large touch-screen that displays the passenger's route, features an emergency SOS button, shows battery levels, and allows passengers to confirm payment. The maximum amount of passengers our prototype can accommodate is three people. However, for handicap passengers, the number of passengers is limited to one. Cost is based on the weight of the vehicle – since our vehicle is smaller in size, production will be slightly less expensive. The car is entirely automatic: the passenger will input their pick up and drop off location through the app, an emergency SOS feature can be accessed at any point during the trip where the passenger feels unsafe or needs assistance, and sensors will be used to detect colors (red is stop, green is go) and surrounding objects. These design components ensure that the project requirements are met.

Interior design of the car:



Behavioral Design



Summary of Feedback:

Our feedback for the Gantt chart was to slightly adjust the timeline. Initially, we had displayed a timeline starting from the very beginning of Project Spyn; so, we altered the chart to only display the tasks for upcoming labs. In regards to our design proposal, it was suggested that we incorporate a unique feature. Based on this suggestion, we decided to add a moving ramp. This ramp will extend via sensor – it will detect the passenger in front of the ramp. Once the passenger is ready to enter the car, they can press a button on the app that allows the ramp to move (similar to the motion of a treadmill).

Overall Design Proposal:

Within this implementation, we also created a gantt chart that aided in the progression of the overall project. We made slight adjustments each week to ensure maximum productivity, but for the most part, our gantt chart remained similar to what we initially created. Based on our design proposal ideas above, when executing our code in Robosim, we selected the Rocket robot. We felt that the model best reflected our ideal vehicle design. One of the main features that we wanted to have was the wheelchair lift to pick up and drop off the passenger.

Final Gantt Chart

Project Spyn: Gantt Chart

How does your implementation meet the requirements?

Within the timeline, our team was able to write a MatLab script for an autonomous, wheel-chair accessible vehicle that runs in Robosim. Using the Rocket robot and EasyFare maze, our robot efficiently navigates through the maze using the AngleRel function and touch and color sensors. Once the touch sensors are activated, the motors stop, the robot moves slightly backwards, and then turns. The color sensor works as follows: if red is detected, the robot stops for two seconds, if yellow (pick up location) is detected, the robot stops and picks up the passenger with keyboard control, if green (drop off location) is detected, the robot stops, and drops off the passenger with keyboard control. The color sensors essentially help to ensure that the robot abides to the traffic laws. A successful run looks as follows: the vehicle turns around so that the sensors are facing forward, then moves forward until it touches a wall, activating a touch sensor and prompting the robot to move slightly backwards and turn through the AngleRel function. After turning, the robot will continue to move until it detects a color – either red, yellow, or green. Red prompts the robot to stop for two seconds, and yellow and green exit automatic control and switch to keyboard control so the user can safely pick up and drop off the passenger respectively. In the earlier stages of the project, we identified the cost constraints, created an assembly line for production to maximize profit, and created a Gantt chart which we referenced throughout the duration of the project. In our final live demonstration today, we displayed a successful run. Ultimately, our robot is able to navigate through the maze effectively and safely transport a handicap passenger; therefore, our design and code meet the necessary requirements.

Our MatLab Code

Youtube Link with Demo:

https://youtu.be/nJGhPTooJBA

MatLab Code:

```
brick.SetColorMode(3,2);

color = brick.ColorCode(3);

reading = brick.TouchPressed(2);
distance = brick.UltrasonicDist(4);
global key;
InitKeyboard();

while 1
% robot turns around to leave the box and moves forward
   brick.MoveMotorAngleRel('A', 100, 358, 'Brake');
   brick.WaitForMotor('A');
   pause(2);
   brick.MoveMotor('AB', -40);
   break;
end
while 1
% robot searches for wall, and when it does, the touch sensor activates and goes back a little bit
   reading = brick.TouchPressed(2);
   if reading
      brick.MoveMotorAngleRel('AB', 100, 180, 'Brake');
      brick.WaitForMotor('AB');
      pause(2);
      break;
   end
end
% robot turns using angleRel function to the right and moves forward until another wall is detected
while 1
   brick.MoveMotorAngleRel('B', 100, 300, 'Brake');
   brick.WaitForMotor('B');
   pause(1);
   brick.MoveMotor('AB', -40);
   break;
end
```

```
% when the robot touches the wall, touch sensor activates, moving the robot backwards slightly and then
turning to the right
while 1
   reading = brick.TouchPressed(2);
   if reading
      brick.MoveMotorAngleRel('AB', 100, 550, 'Brake');
      brick.WaitForMotor('AB');
      pause(2);
      break;
   end
End
 brick.MoveMotorAngleRel('B', 100, 300, 'Brake');
 brick.WaitForMotor('B');
 pause(2);
 brick.MoveMotor('AB', -40);

% once robot reaches red line it stops for 2 seconds then continues to move forward
while 1
  color = brick.ColorCode(3);

     if color == 5

        brick.StopMotor('AB');

       break;
     end
end
pause(2);
brick.MoveMotor('AB', -40);

% once robot reaches yellow line, motors stop moving and keyboard controls begin to pick up passenger
% pressing q will quit manual mode and the robot goes back into automatic mode
while 1
  color = brick.ColorCode(3);

     if color == 4

        brick.StopMotor('AB');

       break;
     end

end
pause(2);
```

```
while 1
      pause(.1)
       switch key
          case 'space'
             brick.MoveMotor('C',1000);
             pause(5)
             brick.StopAllMotors();
          case 'a'
             brick.MoveMotor('C', -1000 );
             pause(5)
             brick.StopAllMotors();
          case 'uparrow'
             brick.MoveMotor('AB', -20);
             pause(1)
             brick.StopAllMotors();
          case 'downarrow'
             brick.MoveMotor('AB' , 10);
             pause(1)
             brick.StopAllMotors();
          case 'rightarrow'
             brick.Move Motor('A', -4)
             brick.MoveMotor('B', 4);
             pause(1)
             brick.StopAllMotors();
           case 'leftarrow'
             brick.Move Motor('A', 4)
             brick.MoveMotor('B', -4);
             pause(1)
             brick.StopAllMotors();
           case 'q'
             brick.MoveMotor('AB', -10);
             break;
       end
end
while 1
  color = brick.ColorCode(3);

     if color == 5

        brick.StopMotor('AB');

       break;
     end
end
```

```
pause(2);
brick.MoveMotor('AB', -10);
while 1
   reading = brick.TouchPressed(2);
   if reading
      brick.MoveMotorAngleRel('AB', 20, 300, 'Brake');
      brick.WaitForMotor('AB');
      pause(2);
      break;
   end
end
while 1
   brick.MoveMotorAngleRel('A', 20, 400, 'Brake');
   brick.WaitForMotor('A');
   pause(2);
   break;
end
brick.MoveMotor('AB', -20);
while 1
   reading = brick.TouchPressed(2);
   if reading
      brick.MoveMotorAngleRel('AB', 20, 310, 'Brake');
      brick.WaitForMotor('AB');
      pause(2);
      break;
   end
end
brick.MoveMotorAngleRel('A', 20, 420, 'Brake');
brick.WaitForMotor('A');
pause(2);
brick.MoveMotor('AB', -20);
while 1
   reading = brick.TouchPressed(2);
   if reading
      brick.MoveMotorAngleRel('AB', 20, 330, 'Brake');
      brick.WaitForMotor('AB');
      pause(2);
      break;
   end
end
brick.MoveMotorAngleRel('A', 20, 420, 'Brake');
brick.WaitForMotor('A');
pause(2);
brick.MoveMotor('AB', -10);
```

```
while 1
  color = brick.ColorCode(3);

     if color == 3
        brick.StopMotor('AB');
       break;
     end

end
pause(2);
while 1
      pause(.1)
       switch key
          case 'space'
             brick.MoveMotor('C',1000);
             pause(5)
             brick.StopAllMotors();
          case 'a'
             brick.MoveMotor('C', -1000 );
             pause(5)
             brick.StopAllMotors();
          case 'uparrow'
             brick.MoveMotor('AB', -20);
             pause(1)
             brick.StopAllMotors();
          case 'downarrow'
             brick.MoveMotor('AB' , 10);
             pause(1)
             brick.StopAllMotors();
          case 'rightarrow'
             brick.MoveMotor('A', -4)
             brick.MoveMotor('B', 4);
             pause(1)
             brick.StopAllMotors();
           case 'leftarrow'
             brick.MoveMotor('A', 4)
             brick.MoveMotor('B', -4);
             pause(1)
             brick.StopAllMotors();
           case 'q'
             break;
       End
end
```