CmpE 443
Fall 2019
Wall Follower Car

NFS
Gürkan Demir
Burak İkan Yıldız
Enes Turan Özcan
Ufuk Yılmaz

January 3, 2020

Contents

1 Introduction

In this final project, we are expected to build a toy car that will follow along a wall. For the car, we use 1 Motor Controller, 2 LDRs, 4 LEDs, 1 potentiometer, 1 speedometer and 1 Ultrasonic sensor. The car will have two separate operating modes. One of them is Test mode which will be used to test the car using serial communication. The other one is Autonomous mode which car follow along a wall itself.

In this report, we will introduce the progress of our group: **NFS** on the final project of CmpE 443 (Embedded Systems) Course. The actions and the conditions are explained in detail below. We prepared this report in order to explain the hardware connections and software-hardware relations.

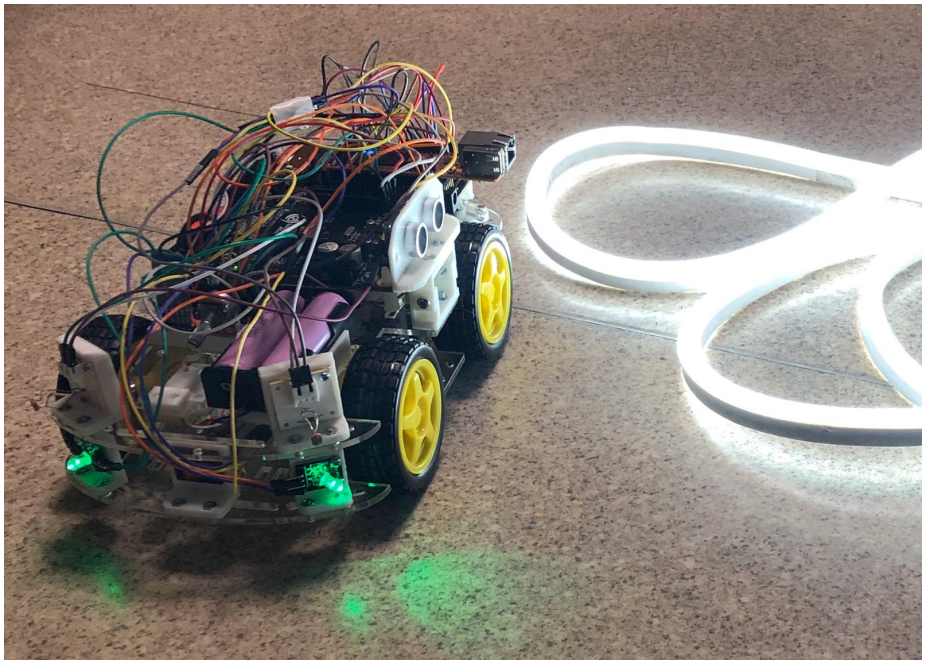

Figure 1: NFS wall follower car.

2 Requirements

2.1 Global Requirements

- When the device stops, all LEDs shall be turned off.
- While the device is moving forward, only the LEDs at the front shall be turned on.

- While the device is moving backward, only the LEDs at the back shall be turned on.
- While the device is turning left (counter-clockwise direction), the LEDs at the left shall blink twice a second. Other LEDs must stay turned off.
- While the device is turning right (clockwise direction), the LEDs at the right shall blink twice a second. Other LEDs must stay turned off.
- Whenever the ASCII text "STATUS" followed by a carriage return and a line feed is sent to the device via serial communication, the device should return a status information.
- An external potentiometer must be used to control the velocity of the car. The potentiometer must allow us to set the speed from 0% to 100%, where 0% being literally no movement and 100% being the maximum speed.

Status information is a JSON object. It must be sent by the device whenever the STATUS command is sent to the device. Structure of object:

- distance: (Integer in cm) The distance from the wall, as measured by the Ultrasonic Sensor.
- light_level_left: (Integer between 0-1023) The light level as measured by the LDR on the left.
- light_right_left: (Integer between 0-1023) The light level as measured by the LDR on the right.
- op_mode: (String AUTO—TEST) The current operating mode of the device.

2.2 Test Requirements

- When entered, the device should send the ASCII text "TESTING" followed by a carriage return and a line feed via serial communication.
- Whenever the ASCII text "LEFT" followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn counter-clockwise 90 degrees in its place and then stop.
- Whenever the ASCII text "RIGHT" followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn clockwise 90 degrees in its place and then stop.
- Whenever the ASCII text "FORWARD" followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving forward.

- Whenever the ASCII text “BACK” followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving backward.
- Whenever the ASCII text “STOP” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should stop moving.
- Whenever a command word followed by a carriage return and a line feed is sent, the device should echo the sent command back via the serial communication followed by a carriage return and a line feed.
- Whenever LDRs detect a light source (a light source (more than 300 lumens) which is much brighter than the room light), the car must stop.When the light source is gone, the device must continue doing whatever it was doing before.
- Whenever the ASCII text “AUTO” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should switch to AUTONOMOUS MODE.

2.3 Autonomous Requirements

- When switched to AUTONOMOUS MODE, the device should send the ASCII text “AUTONOMOUS” followed by a carriage return and a line feed via serial communication.
- When switched to AUTONOMOUS MODE, the device must wait until the ASCII text “START” followed by a carriage return and a line feed is sent through serial communication.
- While moving and next to a wall, the device should move along the wall, within 15 to 35 centimeters from the wall.
- Once the device faces the LED strip at the end of the road while moving, it must stop and send the ASCII text “FINISH” followed by a carriage return and a line feed through serial communication.
- Whenever the ASCII text “TEST” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should switch to TEST MODE.

3 Block Diagrams

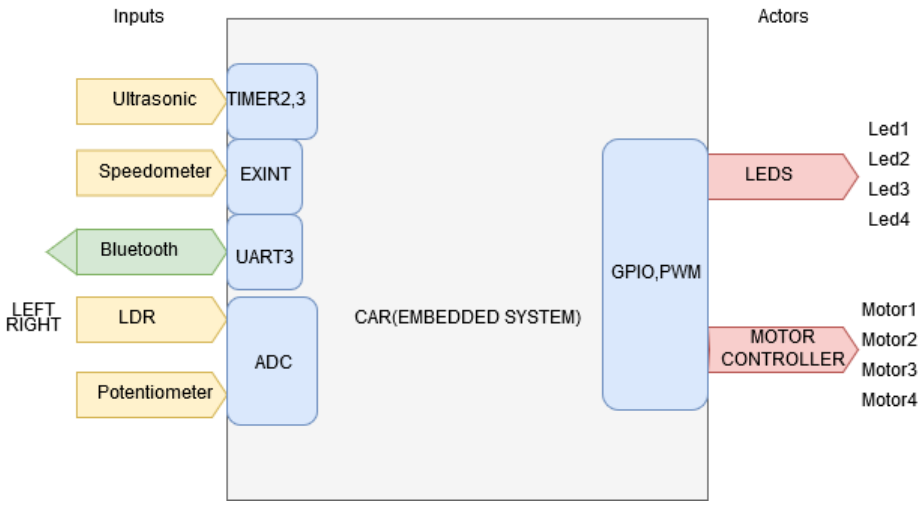

Figure 2: Block diagram of NFS wall follower car.

For this project we use 1 Motor Controller, 2 LDRs, 4 LEDs, 1 potentiometer, 1 speedometer and 1 Ultrasonic sensor.

3.1 Inputs

1. Ultrasonic Sensor
2. Speedometer
3. Bluetooth
4. LDRs
5. Potentiometer

Ultrasonic sensor, connected to Timer2 and Timer 3, is used to measure the distance from the wall in autonomous mode.

Speedometer, connected to EXINT, is used to calculate the number of turning counts of motor.

Bluetooth, connected to UART3, is used to get commands via the mobile application. Car generates a response according to type of command.

LDRs, connected to ADC, are used to detect a light source which is much brighter than room light.

Potentiometer, connected to ADC, is used to change speed of the car.

3.2 Outputs

1. LEDs
2. Motor Controller
3. Bluetooth

LEDs, connected to GPIO and PWM1, is used to represent direction of the car.

Motor controller, connected to GPIO and PWM0, is used configure direction of the car. Motor controller is connected to 4 motor, where 2 of them on the left and other 2 are on the right.

Bluetooth is used to response to commands via mobile application.

Details of pin configurations are listed in the following sections.

4 Flowcharts

4.1 Initialization

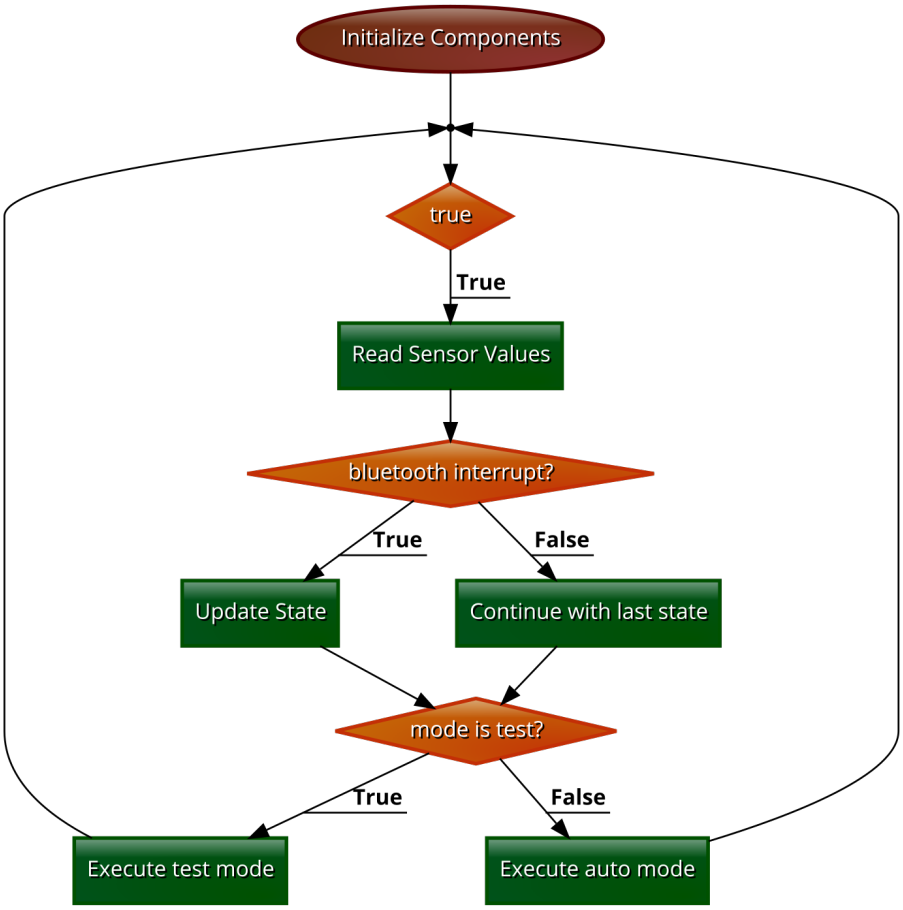

Figure 3: Flowchart of initialization.

4.2 Test Mode

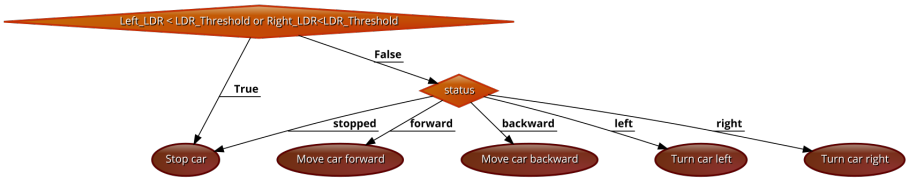

Figure 4: Flowchart of test mode.

4.3 Auto Mode

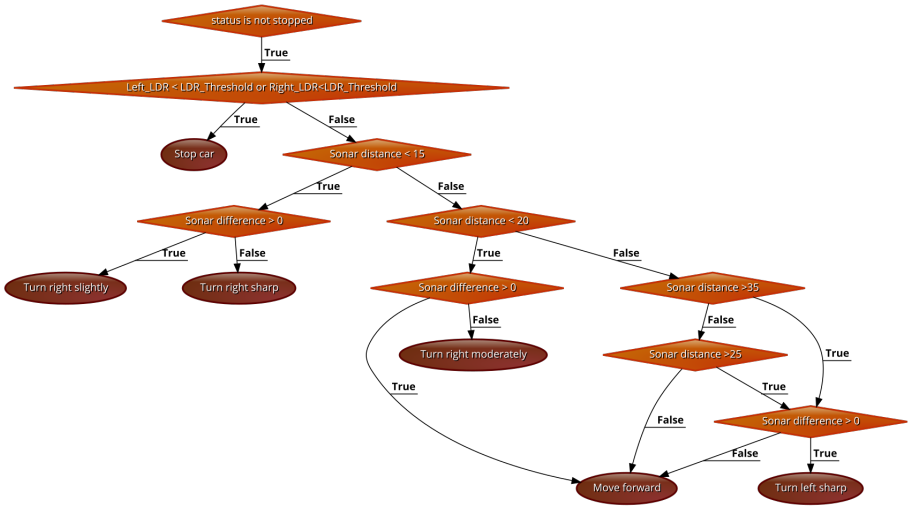

Figure 5: Flowchart of auto mode.

5 Pin Configurations

5.1 LEDs

LED	+ Pin	+ Functionality	- Pin	- Functionality
Front Left	P1.6(P27)	GPIO	P1.20(P7)	PWM1[2]
Front Right	P0.21(P8)	GPIO	P1.20(P7)	PWM1[2]
Back Left	P1.23(P6)	GPIO	P1.20(P7)	PWM1[2]
Back Right	P1.24(P5)	GPIO	P1.20(P7)	PWM1[2]

Table 1: LEDs configuration in board.

We have 4 LEDs in our car. Leds represent the direction of the car. When car is moving forward, LEDs at the front are turned on. When car is moving backward, LEDs at the back are turned on. When car is turning right, LEDs at the right are blinked 2 times in a second. When car is turning left, LEDs at the left are blinked 2 times in a second.

LEDs positive pin is connected to GPIO, and negative pin is connected to PWM1. In order to blink leds when turning right or left we use PWMs. When the GPIO value is LOW, it means led is turned off. When it is HIGH, leds on or off condition depends on the PWM value, when PWM side is LOW it turns on led. Depending on duty cycles, leds are turned on, off or blinked.

Expected functionalities:

1. When the device stops, all LEDs shall be turned off.
2. While the device is moving forward, only the LEDs at the front shall be turned on.
3. While the device is moving backward, only the LEDs at the back shall be turned on.
4. While the device is turning left (counter-clockwise direction), the LEDs at the left shall blink twice a second. Other LEDs must stay turned off.
5. While the device is turning right (clockwise direction), the LEDs at the right shall blink twice a second. Other LEDs must stay turned off.

5.2 Motors

Motors	**Enable Pin**	**Enable Functionality**	**IN1 Pin**	**IN2 Pin**
Motor Left	P1.2(P30)	PWM0[1]	P1.30(P19)(GPIO)	P1.31(P20)(GPIO)
Motor Right	P1.3(P29)	PWM0[2]	P0.4(P34)(GPIO)	P0.5(P33)(GPIO)

Table 2: Motors configuration in board.

We have 1 motor controller in our car. Motor controller is connected to 4 motors, where 2 of them is on the left and others are on the right. Motors on the left, and motors on the right are connected to each other. In others words, motors on the left have common IN1, IN2, and PWM, same applies for motors on the right.

In order to move car forward, we have to give HIGH value for IN1, IN3 and LOW value for IN2, IN4. In order to move car backward, we have to LOW value for IN1, IN3 and HIGH value for IN2, IN4. In order to turn car right, IN1, IN4 values must be HIGH, and others must be LOW, vice versa is applied for turning left.

Expected functionalities:

1. Whenever the ASCII text “LEFT” followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn counter-clockwise 90 degrees in its place and then stop.

2. Whenever the ASCII text “RIGHT” followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn clockwise 90 degrees in its place and then stop.

3. Whenever the ASCII text “FORWARD” followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving forward.

4. Whenever the ASCII text “BACK” followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving backward.

5. Whenever the ASCII text “STOP” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should stop moving.

6. Whenever a command word followed by a carriage return and a line feed is sent, the device should echo the sent command back via the serial communication followed by a carriage return and a line feed.

5.3 HM10 - Bluetooth

HM10 Bluetooth	**RX Pin**	**RX Functionality**	**TX Pin**	**TX Functionality**
HM10 Bluetooth	P0.0(P9)	U3_TXD	P0.1(P10)	U3_RXD

Table 3: HM10 configuration in board.

Bluetooth is used to communication between car and mobile phone. When command arrives from mobile phone, car responses accordingly. HM10 is connected to UART3.

Expected functionalities:

1. Whenever the ASCII text “STATUS” followed by a carriage return and a line feed is sent to the device via serial communication, the device should return a status information.

2. When entered, the device should send the ASCII text “TESTING” followed by a carriage return and a line feed via serial communication.

3. Whenever the ASCII text “LEFT” followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn counter-clockwise 90 degrees in its place and then stop.

4. Whenever the ASCII text “RIGHT” followed by a carriage return and a linefeed is sent to the device via serial communication, the device must turn clockwise 90 degrees in its place and then stop.

5. Whenever the ASCII text “FORWARD” followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving forward.

6. Whenever the ASCII text “BACK” followed by a carriage return and a line feed is sent to the device via serial communication, the device should start moving backward.

7. Whenever the ASCII text “STOP” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should stop moving.

8. Whenever a command word followed by a carriage return and a line feed is sent, the device should echo the sent command back via the serial communication followed by a carriage return and a line feed.

9. When switched to AUTONOMOUS MODE, the device should send the ASCII text “AUTONOMOUS” followed by a carriage return and a line feed via serial communication.

10. When switched to AUTONOMOUS MODE, the device must wait until the ASCII text “START” followed by a carriage return and a line feed is sent through serial communication.

11. Once the device faces the LED strip at the end of the road while moving, it must stop and send the ASCII text “FINISH” followed by a carriage return and a line feed through serial communication.

12. Whenever the ASCII text “TEST” followed by a carriage return and a linefeed is sent to the device via serial communication, the device should switch to TEST MODE.

5.4 LDRs

LDR	Pin	Functionality
Left	P0.26(P18)	ADC0[3]
Right	P0.25(P17)	ADC0[2]

Table 4: LDRs configuration in board.

In order to detect light which is much brighter than room condition, we use 2 LDRs where one of them is located on the left and another is located on the right. LDRs are connected ADC.

Analog to Digital converter pins are used to convert the resistor values which are analog and turn to digital for the computer to execute. Normally, maximum value of ADC can be 0xFFF which is 4095, but in this project we are expected to scale it to 0-1023 for LDRs.

Expected functionalities:

1. Whenever LDRs detect a light source (a light source (more than 300 lumens) which is much brighter than the room light), the car must stop. When the light source is gone, the device must continue doing whatever it was doing before.

2. Once the device faces the LED strip at the end of the road while moving, it must stop and send the ASCII text "FINISH" followed by a carriage return and a line feed through serial communication.

5.5 Potentiometer

Potentiometer	Pin	Functionality
Potentiometer	P0.24(P16)	ADC0[1]

Table 5: Potentiometer configuration in board.

In our car, in order to specify the speed we use potentiometer. Potentiometer is connected to ADC. Analog to Digital converter pins are used to convert the resistor values which are analog and turn to digital for the computer to execute. Normally, maximum value of ADC can be 0xFFF which is 4095, but in this project we are expected to scale it to 0-100 for potentiometer.

Expected functionalities:

1. An external potentiometer must be used to control the velocity of the car. The potentiometer must allow us to set the speed from 0% to 100%, where 0% being literally no movement and 100% being the maximum speed.

5.6 Speedometer

Speedometer	Pin	Functionality
Speedometer	P2.10(P23)	EINT0

Table 6: Speedometer configuration in board.

Speedometer, is used to calculate the number of turning counts of motor. It is connected to EXINT. When interrupt occurs, it increments the counter by 1.

Expected functionalities:

1. Calculate motor turn count, and stop the motor according to given value.

5.7 Ultrasonic Sensor

Ultrasonic Sensor	Trigger Pin	Trigger Functionality	Echo Pin	Echo Functionality
Ultrasonic Sensor	P0.9(P11)	T2_MAT3	P0.23(P15)	T3_CAP0

Table 7: Ultrasonic Sensor configuration in board.

In order to follow the wall, we use ultrasonic sensor which measures the distance between car and wall. Sensor's trigger pin is connected to pin with functionality T2_MAT3 and echo pin is connected to pin with functionality T3_CAP0.

Expected functionalities:

1. While moving and next to a wall, the device should move along the wall, within 15 to 35 centimeters from the wall.

6 Circuit Schematic

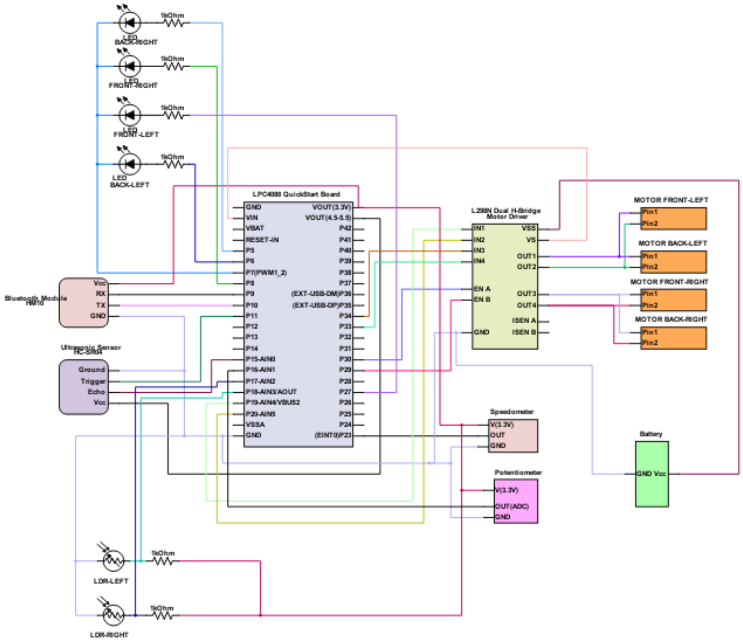

Figure 6: Circuit schematic of NFS wall follower car.

7 Sequence Diagrams

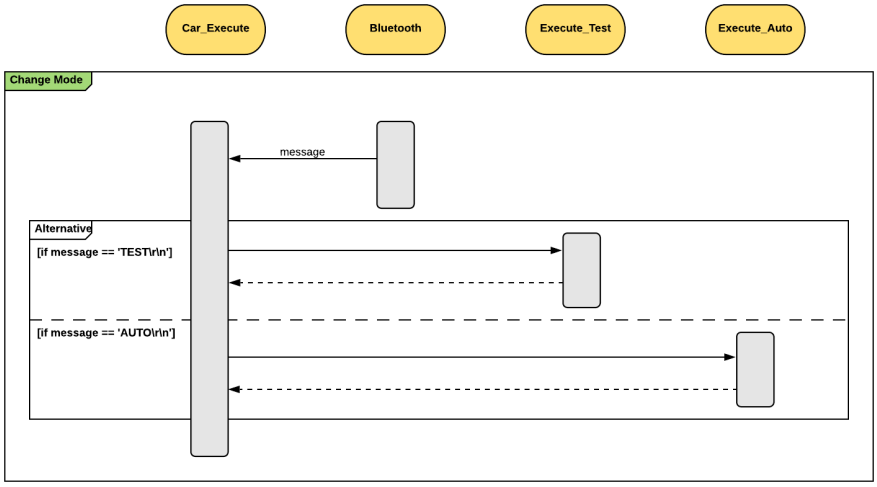

Figure 7: Choosing mode according to message from bluetooth.

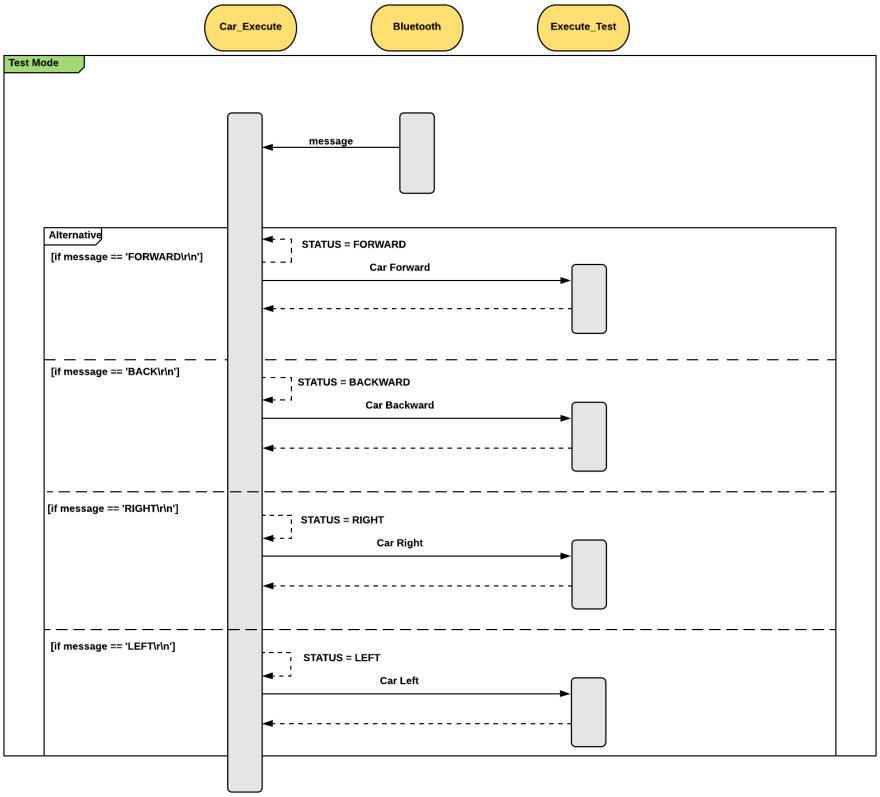

Figure 8: Sequence diagram of test mode.

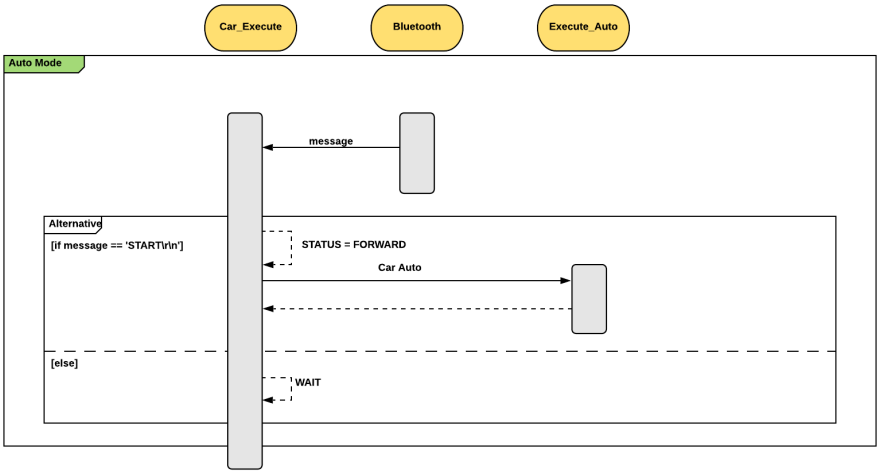

Figure 9: Sequence diagram of auto mode.

8 Pseudo Code

Algorithm 1 Initialization

```
Initialize components;
while True do
    Read sensor values;
    if Bluetooth interrupt? then
        Update state;
    else
        Continue with last state;
    end
    if mode is test? then
        Execute test mode;
    else
        Execute auto mode;
    end
end
```

8.1 Test Mode

Algorithm 2 Test mode algorithm

```
if Left_LDR < LDR_Threshold or Right_LDR < LDR_Threshold then
    Stop car;
else
    if status is stopped then
        Stop car;
    else if status is forward then
        Move car forward;
    else if status is backward then
        Move car backward;
    else if status is left then
        Turn car left;
    else if status is right then
        Turn car right;
```

8.2 Auto Mode

Algorithm 3 Auto mode algorithm

```
if status is not stopped then
    if Left_LDR < LDR_Threshold or Right_LDR < LDR_Threshold then
        Stop car;
    else if Sonar distance < 15 then
        if Sonar difference > 0 then
            Turn right slightly;
        else
            Turn right sharp;
    else if Sonar distance < 20 then
        if Sonar difference > 0 then
            Move forward;
        else
            Turn right moderately;
    else if Sonar distance > 35 then
        if Sonar difference > 0 then
            Turn left sharp;
        else
            Move forward;
    else if Sonar distance > 25 then
        if Sonar difference > 0 then
            Turn left sharp;
        else
            Move forward;
    else
        Move forward;
```

9 How To Use

1. Download the source code files from Github Link
2. In order setup environment, from Keil Website download and install Keil uVision5application (MDK-ARM)
3. Connect the “micro-B” part to the HDK USB port of the LPC4088 Quick-Start Board.
4. Install mbed Windows Serial Software from here.
5. Enter Keil uVision5 and open Pack Installer, Select NXP Devices and select LPC4000_DFP and install that pack.
6. Close the Keil.

7. Connect the LEDs, Motors, HM10, LDRs, Potentiometer, Speedometer and Ultrasonic Sensor as specified in section 5.
8. Open the project with Keil.
9. Build the project in Keil.
10. Load the project to the LPC4088 Board.

10 Expense List

1. MM Jumper Cable, 5 TL
2. MF Jumper Cable, 5 TL
3. FF Jumper Cable, 5 TL
4. 5 Leds, 0.5 TL
5. 10 Resistors, 0.5 TL

11 Conclusion

During the CMPE443 course this semester, we have learned the basics, use cases and the impacts of the embedding system in our daily lives. We got used to low-level programming and embedded system design along with its concepts a few of which are interrupts, hardware components, port, and pin structure. Most importantly, we are now aware of how to extract the required information from an embedded system manual. Lab hours were also an opportunity to perform out theoretical knowledge on real-world tasks. In short, this course was an amusing experience for four of us.

As the final project of the course, we have designed a car and programmed its all required functionalities. This project was an opportunity for us to consolidate our knowledge gained throughout the semester and turn them into practice. It also enabled us to assemble the experience learned during lab hours under a single project. One of the most important objectives of the project - at least in our opinion, was the work-sharing among team members. For some tasks, we assigned a particular member to implement it and for the rest, we either worked as four or assigned it to two or three of the members. Apart from the embedded system issues, completing this project as teamwork was an essential aspect of it.

The most challenging part - actually the only one, of the project that we have faced during the implementation and experiment phase was to perform the automated motion. We first decided on the logic for the automated movement and started implementation. We were all set on the code. However, when we

tested the behavior of the car on the path, we got unexpected results inconsistent with the conditions written. After the debugging session, we concluded that the ultrasonic sensor we use was not consistently bringing data. This was probably because of the reflection of the waves when a certain angle with the transmitter has occurred. We overcome this issue by filtering the sensor data. This was an extremely informative part of the project for us.

As the last words before we end the conclusion, we want to mention a few issues that might have been better during the process. The limitation of working hours with the car in the lab was a huge burden both on us and the assistants. We were not able to work on the project other than the open lab hours and this made the implementation period way longer. Also, some of the components used in the car were not working stable and caused hours of unnecessary and inconclusive efforts of us. For instance, the speed of a motor was varying without changing the value on the code but instead, its speed was changing arbitrarily according to the external conditions. We have overcome this by changing the motor. We suggest that it would be better to let us take our car away from the lab and making sure the components function properly.