Project Robotics

2223 1.1 Bsc TI01 Basis Robotica
Semih Can Karakoç (695258)
ALTIVT1a
10-11-2022

Table of contents

Problem 1 – Vision Board

1.1 STARR-1

Situation or Task

Situation is at home.
Task: making a vision board of my education. I am going to make it on the 6th of September It will probably take 3 hours to make it.
I have to make this vision board for my study. Making it, is already a big problem to begin with. It will make clear what my final goals are.

Action

1. Open PowerPoint to create the vision board.
2. Setup my mind (my goals, my expectations of this education/of my future career) Searching for images for the vision board and gathering the pictures for the next step.
3. Making a collage of pictures that I will use for in my vision board.
4. Saving my work when I am finished.
5. Upload the file to Moodle.

Result

I made my vision board at home and it look 3 hours to make it. I wanted to have photos of my future self in there, so I had to search them.
The result is very well. I used PowerPoint to make the collage of photo's. I then searched for photo's that I need, like I have written in the action section.
When I was finished, I put the collage in this word document and saved it.

Reflection

My experience with making a collage was pretty good. I just had to look for creative photos and that took a lot of time.
I have learnt how to make a beautiful vision board in PowerPoint.
I was lucky enough, that I didn't encounter any big problems. So making the vision board took some less time than I anticipated. (took 2 hours instead of 3)
I have 10 photos in my collage, but I wanted more, but I couldn't find more photos. So that was my only little problem I actually encountered.

1.2 STARR-2

Situation or Task

Situation is at school
Task: I will do it at 6 Sep.
My goals are to install all the development environments. I plan to do it in 6 hours.
I need to download all of it, because I need it for my study. It is for my study Technical Informatics. Robotics is about working with Debian and programming in C, so that's why I need it.

Action

1. Download virtual box
2. Download Debian
3. Download extension file for virtual box
4. Install Virtual Box and the extension
5. Install Debian into virtual box
6. Download GCC on Linux (in Debian)
7. Install GCC on Linux (in Debian)

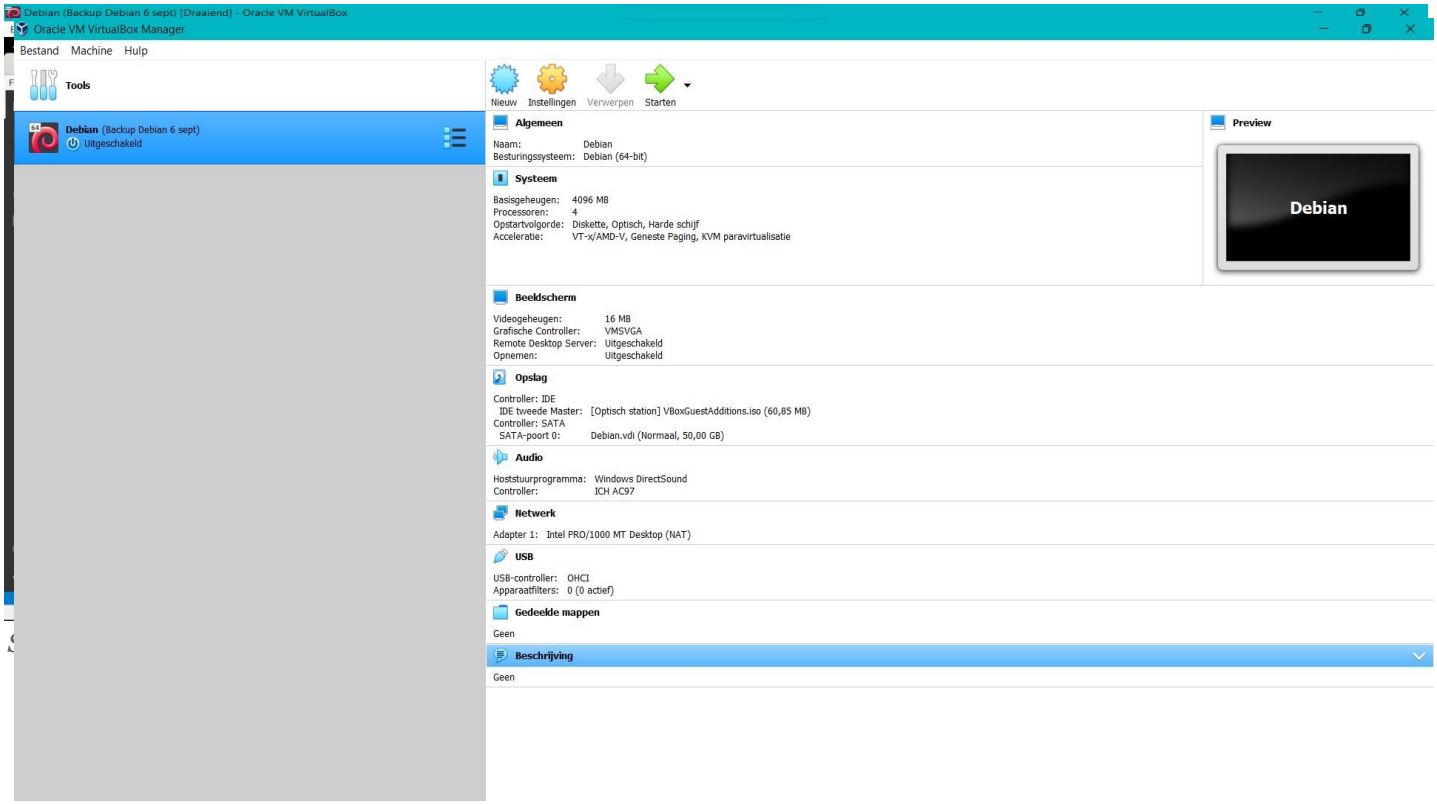
Result

I did this at school and at home, it took probably 1 day, which is estimated 8 hours I think.
I downloaded and installed everything, but the first time installing Debian it didn't work. The problem was that I had chosen the wrong CPU-core, I solved the problem by giving Debian more CPU cores. I have learnt how to download and install a virtual box and Debian. Perhaps in the future I need to download and install it again and now I know how to do it. So in general: installing Debian 11, GCC and Virtual Box went successfully.

Reflection

My experience with downloading and installing these development programs was actually pretty straight forward. The problem was at first sight frustrating, but later when I solved the problem everything went really good.

1.2.1 Proof of installation



Screenshot 2: a screenshot of me running Oracle VM VirtualBox on my laptop.

1.3 Visionboard

1.3.1 Explanation of my visionboard and study motivation

I will learn about databases, embedded security, programming and much more. I think I will see myself later as a technical programmer and engineer. I think I will work as a software engineer or a computer engineer, jobs in that kind of direction. I also find it cool to work and program robots. I am also interested in digital technic, working with bits and ALU's. Ultimately, I will probably have a job where I can develop new things with a team.

I am prepared to study at home and finishing all my problems no matter what happens. My schedule is okay. I work at the McDonald's in Heerhugowaard as my side business (max. 10h/w). I have done that because I want to have enough time to study. I climb every Saturday as my sport to stay fit and healthy. I want to use the off days for my study and if I have enough time in the weekends, I will also use the weekends for studying. My studying locations are school and at home. I do not get distracted that easily, because I am most likely to finish things before stopping. My friends always ask me to join their game, but now with my study, I think I have to reject them for a while… And if I ever need help, I'll just ask my fellow students for help and otherwise my father or the teacher at school.

Problem 2 – Research Project

2.1 Summary

In this problem, I figured out how the robot's ultra-sonic sensor and LEDs work and how to apply them efficiently at certain times. I also figured out from the theoretical side how the components work and how to code them.

2.2 Motivation

Free roaming robots typically need to find objects and move to these objects to perform (inter)actions with these objects. In the project robotics a free roaming robot is used which needs to detect objects and react to them.

2.3 Problem

There is no code available to find the nearest object to the robot and to move in a straight line to the object, without hitting it.

2.4 Goal

The LEDs on the front and the rear show the distance to all surrounding objects using color coding and the robot moves to the nearest object.

2.5 Main research question

How can the robot show the distance to all surrounding objects and move to the nearest object?

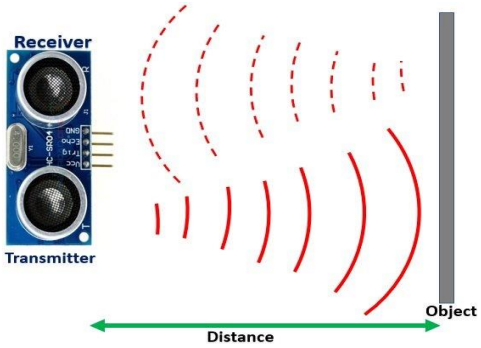
Sub questions:

1. How accurate is the measurement of the ultrasonic sensor?
2. Which code ensures that the LEDs represent the most accurate color coding corresponding to the distance to objects around the robot?
3. How accurate is the time-to-distance conversion of the wheels?

2.6 Theoretical framework

2.6.1 Explain how the ultrasonic sensor works (and the servo motor to which it is attached and which code is used to control the sensor.

The ultrasonic sensor (HC-SR04) emits a high frequency (18kHz – 200 kHz) sound forward and then bounces off an object, then the sound comes back into the sensor's microphone and converts the sound into an electric signal. You can control and measure this electric signal in an oscilloscope With spongy objects, such as rubber clothing and stiff materials, this high frequency sound wave will not reflect, because these objects absorb sound very well. While the sensor is emitting this, the computer is also measuring the time about how long it takes the sound to come back. This allows the computer to calculate the exact distance between the robot and the object with a physics formula.

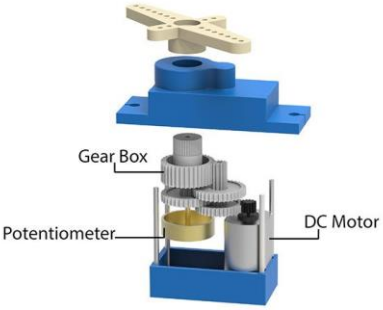


The formula to calculate exact distance between robot and object:

$$s = \frac{1}{2} * t_{sound\ waves} * v_c$$

$s = distance\ (meters)$
$t_{sound\ waves} = total\ time\ of\ sound\ waves\ transmitting\ and\ emitting\ (seconds)$
$v_c = speed\ of\ sound = 343\ (meters/seconds)$

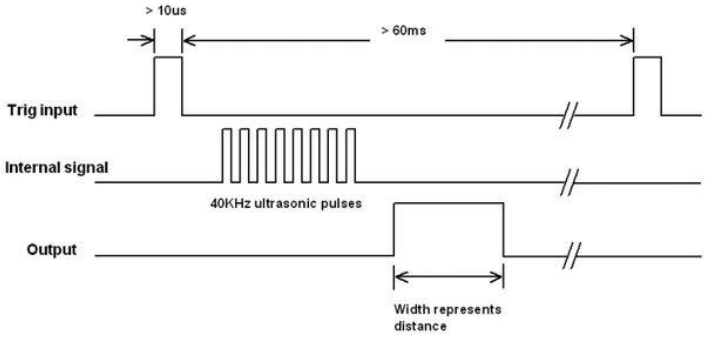
After the distance is determined, the code will use the measured distance to adjust the appropriate speed and/or adjust the appropriate turning angle of the wheels.



A servomotor is a general used name for driving linear or rotary objects. It is a constant movement controllable motor: everything is constant, like the moment, position, acceleration and speed. Servomotors are mostly used for steering transport, in this case for steering the ultrasonic sensor. It uses a gearbox to get the constant speed and it also uses a potentiometer to know if it has turned enough, otherwise it will correct itself.

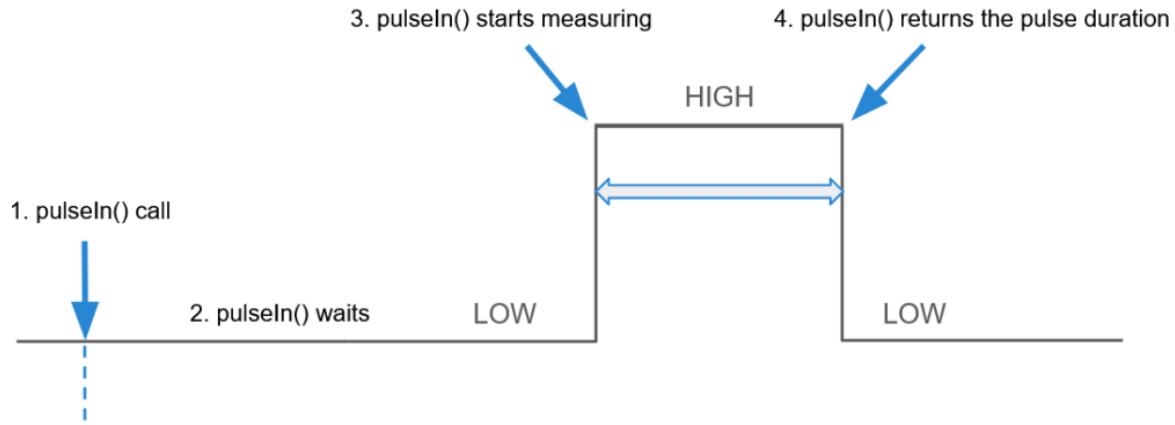
The servomotor in the robot is able to turn 180 degrees around to measure more distances. You can code that the servomotor turns 10 times 18 degrees.

Schematic of an oscillogram with low and high frequencies (PWM, high-low echo's)



The code I use to perform measurements looks like this (I added comments to explain the code):

```
digitalWrite(PIN_SONIC_TRIG, LOW); // Initializes the sonar sensor: set trigger pin to 0
delayMicroseconds(2);  // A delay of 2 micro seconds
digitalWrite(PIN_SONIC_TRIG, HIGH); // Set tPin to 1: this emits the high frequency sound.
delayMicroseconds(10); // Delays it for 10 micro seconds
digitalWrite(PIN_SONIC_TRIG, LOW); // Set tPin to 0: stops emitting high frequency sound.
duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Calculates duration between 0 to 1 to 0
distance = duration * 0.034 / 2; // Time * Speed of Sound / 2. (retour: see theory)
Serial.print("Afstand = "); // Prints “Afstand = “
Serial.print(distance); // Prints the distance variable value. (could be anything)
Serial.println(" centimeter\n"); // Prints “ centimeter“ with a new line.
```

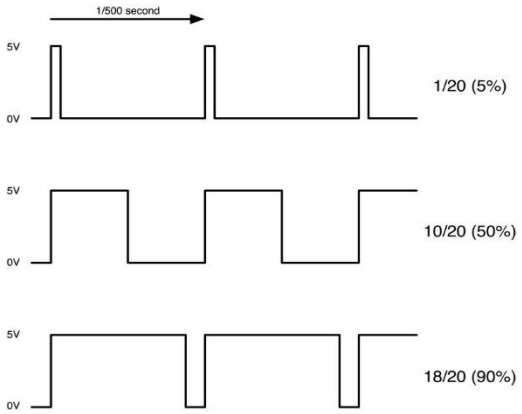


2.6.2 Explain how the colors (LEDs) work, how the LEDs are coded on the robot and which code is used to control the LEDs.

You have many different LEDs, few of them are: RGB LEDs and single-color LEDs. The robot we use uses RGB LEDs. An RGB LED has four pins, namely:

1. this is the long pin and is used as the cathode (+)
2, 3 and 4. Are the three anodes (-), these provide the colors red, green and blue, because you can make all colors with them.

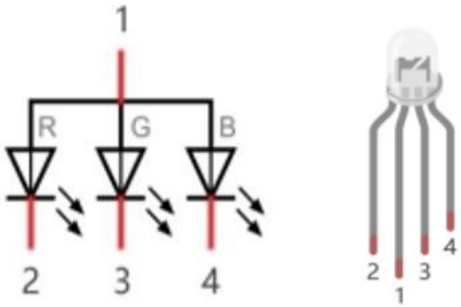
The RGB LEDs are able to emit 256 maximum brightness levels with PWM, which are in total (0-255). PWM is a pulse of an electric signal, the signal is either high or low, that is why the oscillograms all have square figures. How higher the brightness is, how bigger thicker the squares get. Amount of colors is determined by this calculation: 256^3. Each pin supports 2^8 colors and by powering 256 by 3, because of 3 pins, you get all 16777216 colors. The robot has in total 10 RGB LEDS: five each side.



You can use the LEDs for decoration, but that is not very efficient. However, you can also use them usefully as indicators for when objects get close. You can code this with if the distance between the object and the robot gets smaller and smaller (so with an if-statement) first an orange light will light up and then it will become more red and brighter.

All colors, wavelengths, voltages and material put into a table:

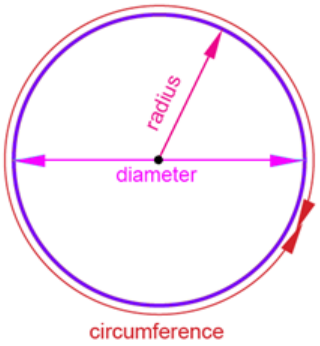
Color		Wavelength (nm)	Forward Voltage (V)
	Ultraviolet (UV)	<400	3.1-4.4
	Violet	400-450	2.8-4.0
	Blue	450-500	2.5-3.7
	Green	500-570	1.9-4.0
	Yellow	570-590	2.1-2.2
	Orange	590-610	2.0-2.1
	Red	610-760	1.6-2.0
	Infrared (IR)	>760	>1.9




```
meten(); // Calling the function I explained in section 2.6.1
if (distance <= 100 && distance >= 50) { // if the dist. = less equal to 100 and more equal
to 50 do the following thing:
  strip.setAllLedsColor(0x00FF00); // set LEDs to green color
  } else if (distance <= 50 && distance >= 20){ // if the first condition wasn't right check
if the distance is less equal to 50 and if the distance is more equal to 20.
    strip.setAllLedsColor(0xFFA500); // set LEDs to orange color
    } else if (distance <= 20){ // check if distance is less equal than 20
      strip.setAllLedsColor(0xFF0000); // set LEDs to red color
      } else { // if all conditions are false do this:
        strip.setAllLedsColor(0, 0, 0); // turn all LEDs off.
        }
```

2.6.3 Explain how the circumference of the wheels relates to distance travelled.

When the robot starts driving with a tire that has a small circumference it covers less distance than a tire with a larger circumference. The distance travelled in one revolution of a tire is equal to the circumference of that tire. Therefore, a tire with a larger circumference travels farther than a tire with a smaller circumference. The advantage of a small tire is that you need less force but more speed. A larger wheel requires more force and less speed. This is because the larger wheel creates more friction with the road surface. Thus, that circumference is equal to the distance covered in one revolution.



The circumference is calculatable by multiplying pi by the diameter:

$$c = \pi * d$$

$c = circumference\ of\ the\ wheel$
$d = diameter$

The area is calculatable by multiplying pi by the radius squared:

$$A = \pi * r^2$$

$A = area\ (surface)\ of\ the\ wheel$
$r = radius$

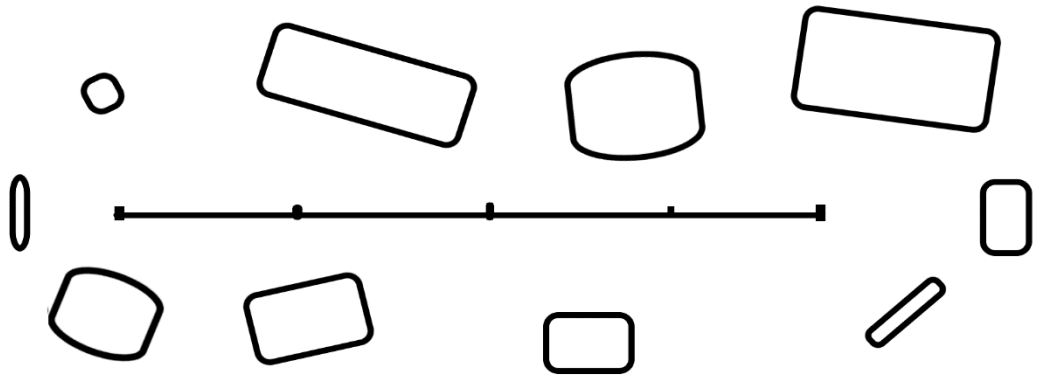
2.7 Methodology

2.7.1 Sub question 1

To answer sub question 1 a series of experiments is done where objects with different shapes are placed around a line and the ultrasonic sensor on the robot is positioned in steps from left to right to measure the distance in the respective position. The robot starts at the beginning of the line and moves along the line to stop at the indicated positions to measure the distance of objects at that position. In a series of indicated positions form left to right the distance is measured. Each distance is measured 10 times. During the measurement at a stop the robot turns around by using the wheels to detect the objects that are behind the robot.

The following form is used to record the results of each experiment:

Schematic of the line and the position of the stops, objects, and their shape:



Position of robot in mm from start	*Position of ultrasonic sensor in degrees*	*Distance measured in mm*									
0	**0**	**270**	**270**	**270**	**200**	**270**	**540**	**530**	**830**	**210**	**270**
0	35	270	200	860	530	490	830	820	470	800	540
0	70	540	540	470	490	820	410	400	520	530	540
0	105	540	510	420	390	380	290	290	380	400	400
0	140	380	400	380	290	270	290	290	290	290	290
300	**0**	**210**	**210**	**210**	**200**	**210**	**480**	**500**	**190**	**200**	**210**
300	35	210	210	190	530	470	500	480	470	470	480
300	70	470	470	470	480	500	500	500	510	530	500
300	105	520	520	500	500	500	240	240	260	340	500
300	140	500	270	250	240	230	240	240	230	240	240
600	**0**	**320**	**330**	**330**	**330**	**330**	**760**	**360**	**320**	**320**	**320**
600	35	320	320	300	340	760	760	760	750	760	760
600	70	760	760	760	750	760	300	290	350	760	760
600	105	760	760	310	290	300	300	300	290	290	290
600	140	290	290	290	290	300	300	300	300	300	300
900	**0**	**490**	**490**	**490**	**480**	**480**	**460**	**470**	**470**	**480**	**490**
900	35	480	480	470	470	460	500	480	470	470	470

900	70	460	470	470	490	500	530	530	590	510	500
900	105	490	540	650	530	530	480	490	480	500	520
900	140	520	490	480	480	480	480	480	480	480	480
1200	**0**	**270**	**270**	**270**	**270**	**270**	**280**	**270**	**260**	**270**	**260**
1200	35	270	270	260	260	280	410	410	330	280	280
1200	70	280	280	340	410	420	410	390	400	390	390
1200	105	420	390	400	410	410	380	380	410	400	410
1200	140	410	400	410	400	380	380	380	380	380	380
After 180 degree turn											
1200	**0**	**310**	**310**	**310**	**310**	**310**	**270**	**270**	**260**	**270**	**280**
1200	35	310	310	310	310	320	690	690	280	280	280
1200	70	270	270	340	690	690	510	510	720	690	690
1200	105	700	700	730	520	510	510	520	510	500	510
1200	140	500	500	510	500	490	630	480	490	490	490
900	**0**	**140**	**140**	**140**	**140**	**140**	**470**	**470**	**480**	**150**	**140**
900	35	140	140	470	470	470	470	470	470	460	470
900	70	460	470	470	470	470	480	480	480	480	480
900	105	480	480	480	480	480	340	550	490	480	480
900	140	480	480	490	590	570	800	790	790	790	790
600	**0**	**280**	**280**	**280**	**270**	**280**	**280**	**270**	**270**	**280**	**280**
600	35	270	270	270	280	280	560	630	470	290	280
600	70	280	280	470	640	580	560	560	560	590	580
600	105	70	90	570	560	560	70	1120	660	470	560
600	140	560	570	650	1130	1120	1130	1120	1120	1120	1120
300	**0**	**460**	**1490**	**470**	**450**	**470**	**460**	**460**	**450**	**480**	**480**
300	35	540	470	450	460	460	30	30	530	450	450
300	70	460	70	40	50	50	30	30	50	30	40
300	105	70	530	50	70	70	360	350	40	30	30
300	140	70	80	340	350	350	360	360	360	360	360
0	**0**	**300**	**300**	**290**	**300**	**300**	**380**	**390**	**390**	**300**	**300**
0	35	290	280	300	730	730	310	320	330	690	690
0	70	730	330	290	280	280	1170	130	270	320	320
0	105	320	320	240	150	140	130	130	130	150	140
0	140	140	140	140	130	130	130	130	130	130	130

2.7.2 Sub question 2

To answer sub question 2 a distance to color chart is created and code to detect objects around the robot, based on the experiments of sub question 1. This color chart is then tested by repeating the experiments of sub question 1 and recording the color visible for that position. The robot turns around to measure the distance to the objects behind the robot and shows the distance using the color chart on both the front and the rear of the robot.

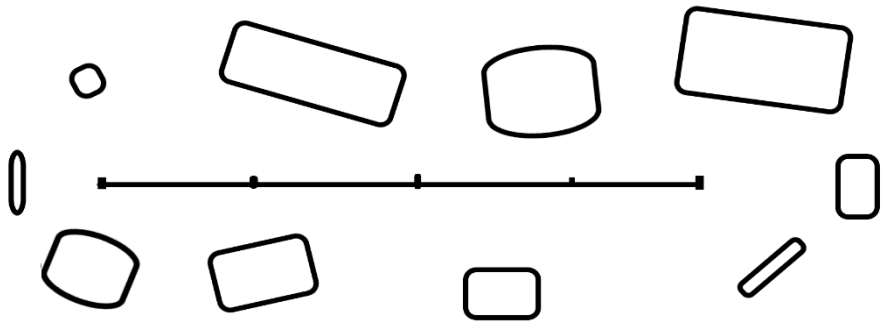
The experiments can be repeated to test several possible color charts, if the results of the previous experiments are not satisfactory.

The following table is used to define the color chart:

Range from in mm	*Range up to and including in mm*	*Color code*
400	600	0x00FF00 (green)
200	400	0xFFA500 (orange)
0	200	0xFF0000 (red)

The following form is used to record the results of each experiment:

Schematic of the line and the position of the objects



Position of ultrasonic sensor in degrees	*Distance measured in mm*										*Color visible in front after 360 degree turn in color code* *G (green) = 0x00FF00* *O (orange) = 0xFFA500* *R (red) = 0xFF0000*									
0	**270**	**270**	**270**	**200**	**270**	**540**	**530**	**830**	**210**	**270**	O	O	O	R	O	G	G	G	O	O
35	270	200	860	530	490	830	820	470	800	540	O	R	G	G	G	G	G	G	G	G
70	540	540	470	490	820	410	400	520	530	540	G	G	G	G	G	G	G	G	G	G
105	540	510	420	390	380	290	290	380	400	400	G	G	G	O	O	O	O	O	G	G
140	380	400	380	290	270	290	290	290	290	290	O	G	O	O	O	O	O	O	O	O
0	**210**	**210**	**210**	**200**	**210**	**480**	**500**	**190**	**200**	**210**	O	O	O	O	O	G	G	R	O	O
35	210	210	190	530	470	500	480	470	470	480	O	O	R	G	G	G	G	G	G	G
70	470	470	470	480	500	500	500	510	530	500	G	G	G	G	G	G	G	G	G	G
105	520	520	500	500	500	240	240	260	340	500	G	G	G	G	G	O	O	O	O	G
140	500	270	250	240	230	240	240	230	240	240	G	O	O	O	O	O	O	O	O	O

0	**320**	**330**	**330**	**330**	**330**	**760**	**360**	**320**	**320**	**320**	O	O	O	O	O	G	O	O	O	O
35	320	320	300	340	760	760	760	750	760	760	O	O	O	O	G	G	G	G	G	G
70	760	760	760	750	760	300	290	350	760	760	G	G	G	G	G	O	O	O	G	G
105	760	760	310	290	300	300	300	290	290	290	G	G	O	O	O	O	O	O	O	O
140	290	290	290	290	300	300	300	300	300	300	O	O	O	O	O	O	O	O	O	O
0	**490**	**490**	**490**	**480**	**480**	**460**	**470**	**470**	**480**	**490**	G	G	G	G	G	G	G	G	G	G
35	480	480	470	470	460	500	480	470	470	470	G	G	G	G	G	G	G	G	G	G
70	460	470	470	490	500	530	530	590	510	500	G	G	G	G	G	G	G	G	G	G
105	490	540	650	530	530	480	490	480	500	520	G	G	G	G	G	G	G	G	G	G
140	520	490	480	480	480	480	480	480	480	480	G	G	G	G	G	G	G	G	G	G
0	**270**	**270**	**270**	**270**	**270**	**280**	**270**	**260**	**270**	**260**	O	O	O	O	O	O	O	O	O	O
35	270	270	260	260	280	410	410	330	280	280	O	O	O	O	O	G	G	O	O	O
70	280	280	340	410	420	410	390	400	390	390	O	O	O	G	G	G	G	G	G	G
105	420	390	400	410	410	380	380	410	400	410	G	O	G	G	G	O	O	G	G	G
140	410	400	410	400	380	380	380	380	380	380	G	G	G	G	O	O	O	O	O	O
After 180 degree turn																				
0	**310**	**310**	**310**	**310**	**310**	**270**	**270**	**260**	**270**	**280**	O	O	O	O	O	O	O	O	O	O
35	310	310	310	310	320	690	690	280	280	280	O	O	O	O	O	G	G	O	O	O
70	270	270	340	690	690	510	510	720	690	690	O	O	O	G	G	G	G	G	G	G
105	700	700	730	520	510	510	520	510	500	510	G	G	G	G	G	G	G	G	G	G
140	500	500	510	500	490	630	480	490	490	490	G	G	G	G	G	G	G	G	G	G
0	**140**	**140**	**140**	**140**	**140**	**470**	**470**	**480**	**150**	**140**	R	R	R	R	R	G	G	G	R	R
35	140	140	470	470	470	470	470	470	460	470	R	R	G	G	G	G	G	G	G	G
70	460	470	470	470	470	480	480	480	480	480	G	G	G	G	G	G	G	G	G	G
105	480	480	480	480	480	340	550	490	480	480	G	G	G	G	G	O	G	G	G	G
140	480	480	490	590	570	800	790	790	790	790	G	G	G	G	G	G	G	G	G	G
0	**280**	**280**	**280**	**270**	**280**	**280**	**270**	**270**	**280**	**280**	O	O	O	O	O	O	O	O	O	O
35	270	270	270	280	280	560	630	470	290	280	O	O	O	O	O	G	G	G	O	O
70	280	280	470	640	580	560	560	560	590	580	O	O	G	G	G	G	G	G	G	G
105	70	90	570	560	560	70	1120	660	470	560	R	R	G	G	G	R	G	G	G	G
140	560	570	650	1130	1120	1130	1120	1120	1120	1120	G	G	G	G	G	G	G	G	G	G
0	**460**	**1490**	**470**	**450**	**470**	**460**	**460**	**450**	**480**	**480**	G	G	G	G	G	G	G	G	G	G
35	540	470	450	460	460	30	30	530	450	450	G	G	G	G	G	R	R	G	G	G
70	460	70	40	50	50	30	30	50	30	40	G	R	R	R	R	R	R	R	R	R
105	70	530	50	70	70	360	350	40	30	30	R	G	R	R	R	O	O	R	R	R
140	70	80	340	350	350	360	360	360	360	360	R	R	O	O	O	O	O	O	O	O

0	**300**	**300**	**290**	**300**	**300**	**380**	**390**	**390**	**300**	**300**	O	O	O	O	O	O	O	O	O	O
35	290	280	300	730	730	310	320	330	690	690	O	O	O	G	G	O	O	O	G	G
70	730	330	290	280	280	1170	130	270	320	320	G	O	O	O	O	G	O	O	O	O
105	320	320	240	150	140	130	130	130	150	140	O	O	O	R	R	R	R	R	R	R
140	140	140	140	130	130	130	130	130	130	130	R	R	R	R	R	R	R	R	R	R

2.7.3 Sub question 3

To answer sub question 3 experiments are done where the wheels are turned for a specific duration and the actual distance is recorded and compared to the theoretical distance. Each time is measured 10 times.

The following form is used to record the results of each experiments:

Time in seconds	Theoretical distance in mm	Measured distance in mm									
0.5	530	587	582	589	575	591	583	597	594	593	595
0.6	636	681	675	676	689	692	684	683	688	693	692
0.7	743	767	786	771	770	787	775	782	785	787	776
0.8	849	871	880	872	875	876	869	885	873	865	866
0.9	955	962	963	967	964	959	964	956	965	967	969
1	1061	1052	1036	1053	1042	1064	1049	1059	1069	1063	1053
2	2122	1695	1885	1912	1895	1891	1935	1909	1945	1912	1914
3	3182	2693	2695	2689	2686	2694	2697	2698	2697	2692	2695
4	4243	3476	3473	3475	3480	3471	3482	3473	3475	3472	3477
5	5304	4395	4394	4396	4391	4389	4392	4399	4387	4393	4395
10	10608	8765	8762	8755	8769	8770	8766	8761	8764	8765	8762
15	15912	13279	13274	13274	13278	13281	13274	13273	13271	13277	13278
20	21216	17593	17591	17587	17598	17596	17589	17585	17594	17592	17596
40	42432	35169	35194	35153	35164	35126	35149	35142	35167	35163	35161
60	63648	50396	50391	50389	50394	50397	50401	50399	50400	50393	50392

$$\boldsymbol{Theoretical\ distance\ (S_T) = RPS * circumference * time}$$

$$Circumference\ of\ the\ wheel = \pi * diameter = 221\ cm$$

$$Diameter\ of\ the\ wheel = 70.34\ cm$$

$$Revolutions\ Per\ Second = \frac{285\ RPM}{60} = 4.75\ RPS\ (I\ also\ measured\ it\ in\ slowmotion)$$

2.8 Results

2.8.1 Sub question 1 & 2

The results of sub question 1 are quite interesting because some of the measurements are not very accurate. Looking at the results in section 2.7 “Methodology”, we see that the first five measurements go well. If we then look at the last five measurements, we see that the first three of them are higher compared to the other two measurements. This is probably, because the servo motor is running at a high speed and then the ultrasonic sensor starts measuring too early.

This problem is best observed through the results of section 2.7.2: “sub question 2”. There you will see the letters “R”, “O” and “G”. These stand for *red*, *orange* and *green*. If we then look closely, it can be seen that in some measurements there is indeed a small deviation. For example, on page 19, at the bottom of measurement “140”. There you can also see that there are two R's first and only then the rest of the O's. So this is because the servo motor is running too fast and there is too little delay before the measurement starts.
Here is a link to a YouTube video I made for this sub question: https://youtu.be/cx0oobsjSIo

2.8.2 Sub question 3

For sub question 3, I took the code from the IR controller and changed the IR commands to my own code. I coded a delay of say 500ms in the switch statement. Then I measured everything with a tape measure and entered it in the results table. What I noticed in the results is that the measured distance does not quite match the theoretical distance. This is mainly because in reality resistors are present. Here have not taken into account in the calculation theoretical distance. In measurements like the one of 1 second, you see that the measured distance is less than the theoretical distance. So this is due to the rolling resistance and the resistance of the floor. The longer the distance becomes, the more the resistance becomes, so the distance traveled becomes less.

You can also see that the first measurement has higher values than the theoretical distance. This is because the robot struggles with the delay 500ms. As the robot accelerates its motors, it has to stop again immediately, making it slightly longer than it should be.

I calculated the theoretical distance by multiplying the RPS (revolutions per second) by the circumference of the wheel and time. I found this out while answering the theoretical questions. You have to think of the circumference of the wheel as a straight line on the ground and the RPS is how many times it travels the circumference in one second. I measured the RPS in slow motion just to be sure, here I came up with about 4.75 revolutions in a second. Here a link to my YouTube video of my RPS provision: https://youtu.be/zOJPOWZni5o

2.9 Conclusion

From my results, you can see that I managed to get the robot to look around and measure its environment. I did this by having the servo motor turn and measure every 35 degrees. These measurements are then printed into the serial monitor. Also, the robot uses its LEDs as a source to show that an object could potentially be present in front of it. From my results you can see back that at least it works fine. The only problem I could have seen coming was that there was too little delay, so the robot started scanning too fast. So the robot uses its lighting as a warning source to show whether there is anything in front of it or not by showing different colors. The robot can possibly be programmed to then drive to the nearby object.

2.10 Follow-up research

What I would like to discover in a follow-up research is, how to make the robot follow the line perfectly and do the 180 degree turn better. I would also like to explore how to make the measurements more accurate. Furthermore, I would also like to investigate what kind of relationship there is between distance traveled, speed and rolling resistance of the wheels. If I am honest, I would also like to learn how to run code more efficiently than I have currently. In follow-up research, I would also like to learn how to have the robot memorize objects and then indicate where the objects were.

2.11 References

Here you put references to the sources you have found (websites, code you have found, etcetera).

2.11.1 Websites

MAGNETIC INNOVATIONS. Servomotor. 2022. Visited on 14-9-2022,
https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor
Danny Jost. Ultrasonic Sensor. 2019. Visited on 14-9-2022,
https://www.magneticinnovations.com/nl/veelgestelde-vragen/hoe-werkt-een-servo-motor/
Microcontrollerslab. HC-SR04. 2022. Visited on 14-9-2022,
https://microcontrollerslab.com/hc-sr04-ultrasonic-sensor-raspberry-pi-pico-micropython-tutorial/
A Minority and Woman-owned Business Enterprise. PWM. 2012. Visited on 14-9-2022,
https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/theory-pwm
LastMinuteEngineers. LED. 2022. Visited on 14-9-2022,
https://lastminuteengineers.com/light-emitting-diode-led/
Wikimedia Foundation, Inc.. Revolutions per minute. 11 August 2022, at 07:38 (UTC). Visited on 21-9-2022,
https://en.wikipedia.org/wiki/Revolutions_per_minute
The Robotics Back-End. Arduino pulseIn() function. 2022. Visited on 5-10-2022,
https://roboticsbackend.com/arduino-pulsein-function/

2.11.2 Figures

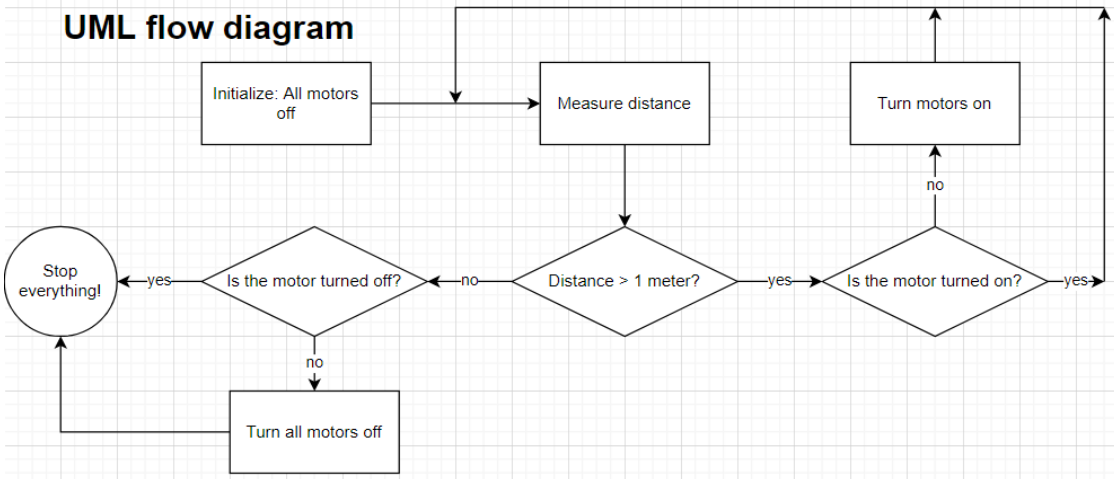
Microcontrollerslab. HC-SR04 (photo). 2022. Visited on 14-9-2022,
https://microcontrollerslab.com/wp-content/uploads/2014/12/Ultrasonic-sensor-HC-SR04-working.jpg
Circumference Of Circle. Circumference Of Circle. 2022. Visited on 14-9-2022,
https://circumferenceofcircle.com/media/django-summernote/2022-03-09/c7c26967-5110-4487-8a66-dbb1a9003c33.png
Microcontrollerslab. HC-SR04. 2022. Visited on 14-9-2022,
https://microcontrollerslab.com/wp-content/uploads/2014/12/Timing-diagram-HC-SR04.jpg
Agilo Research Pvt. Ltd.. What is a Servo Motor. 2022. Visited on 14-9-2022,
https://learn.thestempedia.com/wp-content/uploads/2018/05/Servo-Motor-600x461.jpg
The Robotics Back-End. Arduino pulseIn() function. 2022. Visited on 5-10-2022,
https://roboticsbackend.com/wp-content/uploads/2021/06/arduino_pulsein_explained-1024x375.png

2.12 Code

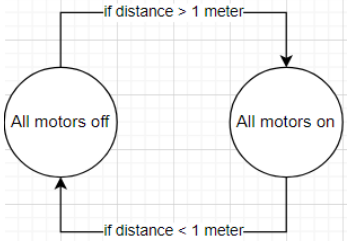
2.12.1 UML flow diagrams and state diagrams

Theoretical question 1:

UML flow diagram

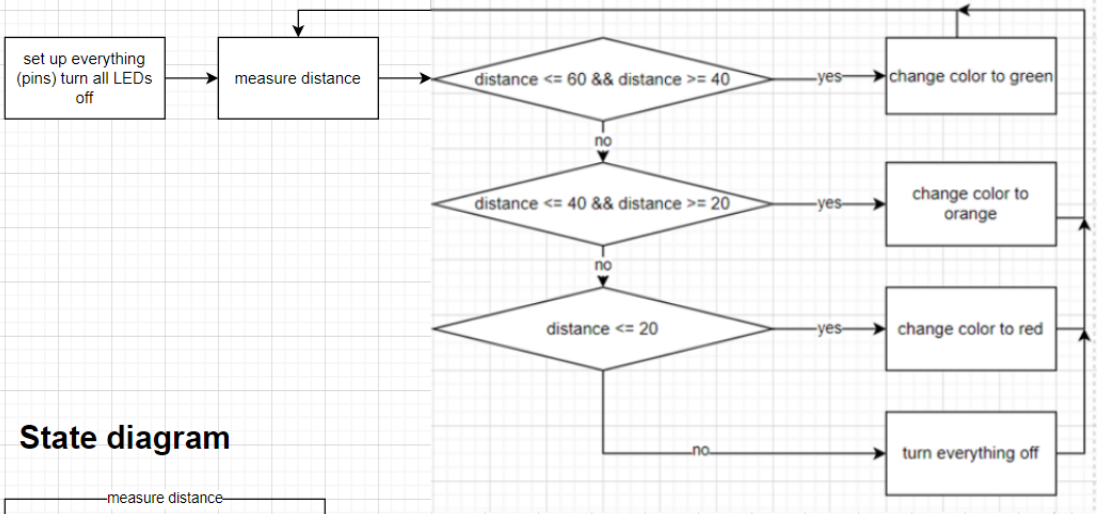


State diagram

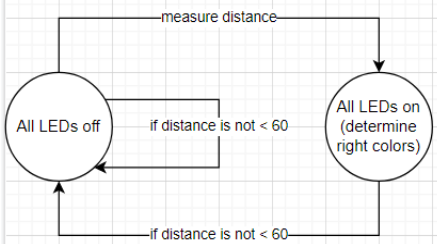


Theoretical question 2:

UML flow diagram

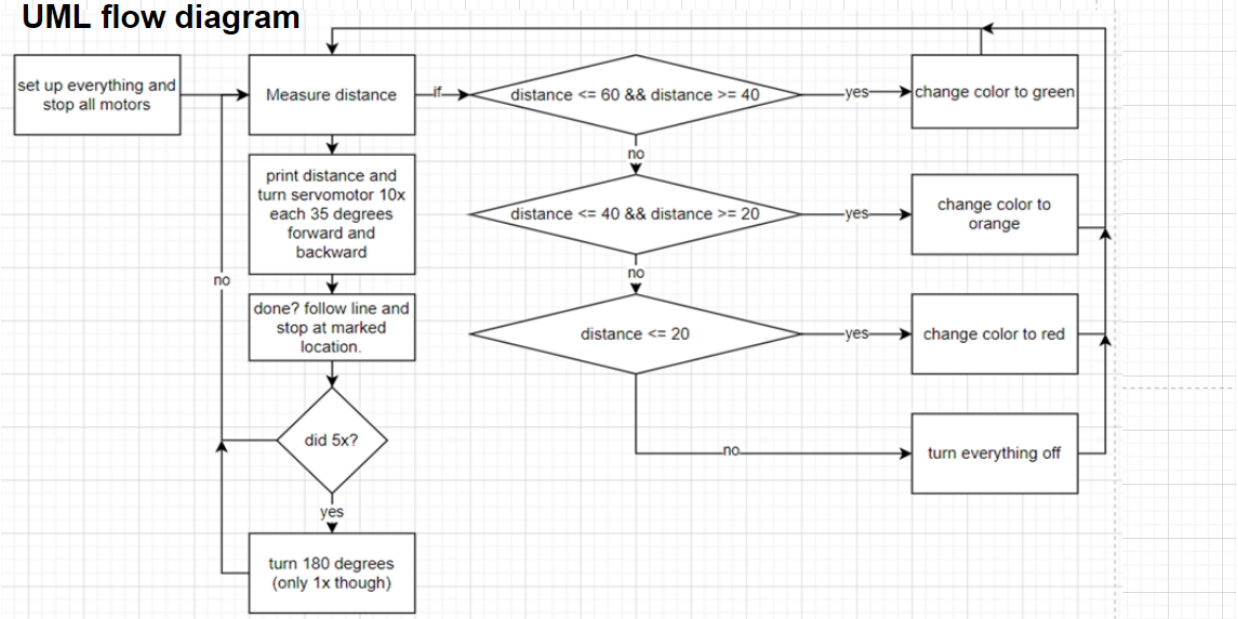


State diagram

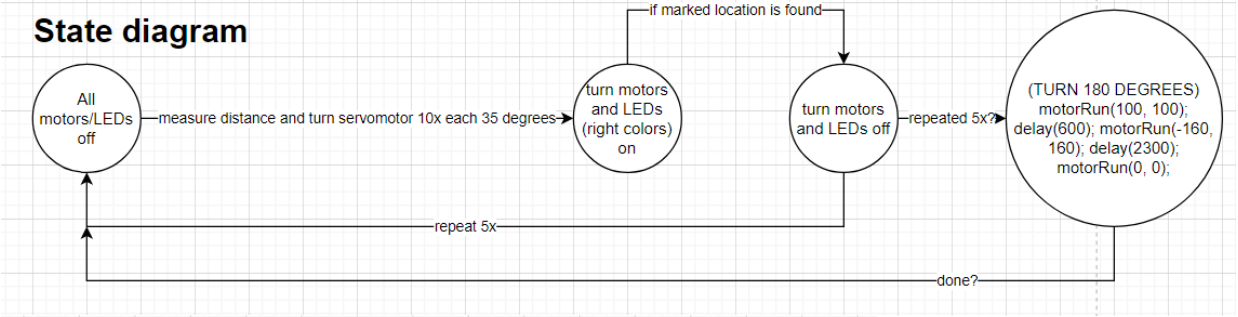


Sub question 1 & 2:

UML flow diagram

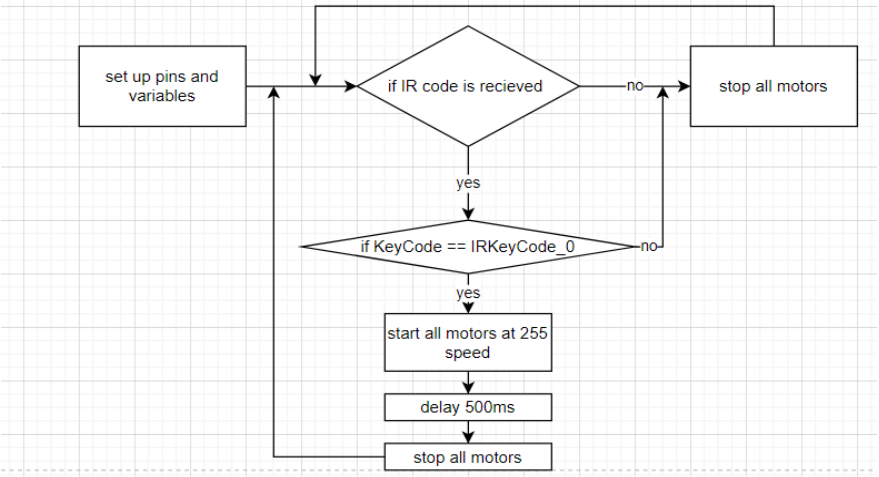


State diagram

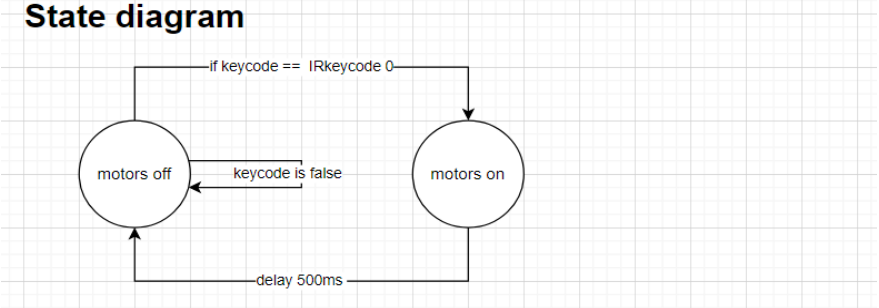


Sub question 3:

UML flow diagram



State diagram



2.12.2 Code for theoretical question 1

```
/* INCLUDE NEEDED LIBRARY */
#include <Servo.h>

/* DEFINE SONAR PINS */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* DEFINE SONAR CONSTANTS */
#define MAX_DISTANCE    300   //cm
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // calculate timeout
#define SOUND_VELOCITY  340  //soundVelocity: 340m/s
/* DEFINE MOTOR PINS */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6

/* DECLARE IMPORTANT VARIABLES */
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement
int speedr = 0, speedl = 0; // variable for motor right and left
int dirL = 0, dirR = 0; // variable for motor directions

/* SETUP FUNCTION */
void setup() {
  Serial.begin(9600);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
}

/* MAIN LOOP FUNCTION */
void loop() {
  /* MOTOR DRIVER */
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
```

```
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R

  /* MEASURE DISTANCE */
  digitalWrite(PIN_SONIC_TRIG, LOW); // Initializes the sonar sensor: set trigger pin to 0
  delayMicroseconds(2); // A delay of 2 micro seconds
  digitalWrite(PIN_SONIC_TRIG, HIGH); // Set tPin to 1: this emits the high frequency sound.
  delayMicroseconds(10); // Delays it for 10 micro seconds
  digitalWrite(PIN_SONIC_TRIG, LOW); // Set tPin to 0: stops emitting high frequency sound.
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Calculates duration between 0 to 1 to 0
  distance = duration * 0.034 / 2; // Time * Speed of Sound / 2. (retour: see theory)
  Serial.print("Afstand = "); // Prints “Afstand = “
  Serial.print(distance); // Prints the distance variable value. (could be anything)
  Serial.println(" centimeter\n"); // Prints “ centimeter“ with a new line.

  /* MAIN PART OF LOOP */
  if (distance > 100) {  // check if distance is greater than 100 cm
    if (speedr > 0 && speedl > 0) {  // check if speedright and speedleft are > 0
      Serial.print("\nMotoren staan aan!\n"); // print that motors are on
    } else {       // otherwise
      speedr = 255, speedl = 255; // go max speed forward
    }
  } else { // otherwise
    if (speedr == 0 && speedl == 0) { // check if speedright and left equals 0
      Serial.print("\nMotoren staan uit!\n"); // print motors are off
    } else {       // otherwise
      speedr = 0, speedl = 0;  // set motors off
    }
  }
}
```

2.12.3 Code for theoretical question 2

```
/* INCLUDE NEEDED LIBRARIES */
#include <Servo.h>
#include <Freenove_WS2812B_RGBLED_Controller.h>

/* DEFINE SONAR PINS */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* DEFINE SONAR CONSTANTS */
#define MAX_DISTANCE    300   //cm
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // calculate timeout
#define SOUND_VELOCITY  340  //soundVelocity: 340m/s
/* DEFINE MOTOR PINS */
#define PIN_DIRECTION_RIGHT 3 // Hier worden de pinnen gedifined
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* DEFINE LED RELATED ADRESS & PINS */
#define I2C_ADDRESS  0x20
#define LEDS_COUNT   10  //it defines number of lEDs.
/* DEFINE BATTERY PIN */
#define PIN_BATTERY     A0 //define battery pin

/* DECLARE IMPORTANT VARIABLES */
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement

/* LED CONTROLLER: */
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB);

/* SETUP FUNCTION */
void setup() {
  !strip.begin();
  Serial.begin(9600);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
}

/* MEASURE DISTANCE FUNCTION */
void meten() {
  digitalWrite(PIN_SONIC_TRIG, LOW); // Initializes the sonar sensor: set trigger pin to 0
  delayMicroseconds(2); // A delay of 2 micro seconds
  digitalWrite(PIN_SONIC_TRIG, HIGH); // Set tPin to 1: this emits the high frequency sound.
  delayMicroseconds(10); // Delays it for 10 micro seconds
  digitalWrite(PIN_SONIC_TRIG, LOW); // Set tPin to 0: stops emitting high frequency sound.
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Calculates duration between 0 to 1 to 0
```

```
  distance = duration * 0.034 / 2; // Time * Speed of Sound / 2. (retour: see theory)
  Serial.print("Afstand = "); // Prints “Afstand = “
  Serial.print(distance); // Prints the distance variable value. (could be anything)
  Serial.println(" centimeter\n"); // Prints “ centimeter“ with a new line.
}

/* MAIN LOOP FUNCTION */
void loop() {
  meten(); // Calling the function I explained in section 2.6.1
  if (distance <= 100 && distance >= 50) { // if the dist. = less equal to 100 and more
equal to 50 do the following thing:
    strip.setAllLedsColor(0x00FF00); // set LEDs to green color
  } else if (distance <= 50 && distance >= 20){ // if the first condition wasn’t right check
if the distance is less equal to 50 and if the distance is more equal to 20.
    strip.setAllLedsColor(0xFFA500); // set LEDs to orange color
  } else if (distance <= 20){ // check if distance is less equal than 20
    strip.setAllLedsColor(0xFF0000); // set LEDs to red color
  } else { // if all conditions are false do this:
    strip.setAllLedsColor(0, 0, 0); // turn all LEDs off.
  }
}
```

2.12.4 Code for sub question 1 & 2

```
/* INCLUDE NEEDED LIBRARY */
#include <Servo.h>             //include servo library
#include <Freenove_WS2812B_RGBLED_Controller.h> //include RGB_controller library

/* DEFINE SERVO PIN */
#define PIN_SERVO      2
/* DEFINE SONAR PINS */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* DEFINE SONAR CONSTANTS */
#define MAX_DISTANCE    300  // Maximum distance of Sonar in CMs
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // Calculate timeout
#define SOUND_VELOCITY  340  // SPEED OF SOUND: 340 m/s
/* DEFINE MOTOR PINS */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* DEFINE BATTERY AND BUZZER PIN */
#define PIN_BATTERY     A0
#define PIN_BUZZER      A0
/* DEFINE LED PIN AND I2C ADDRESS */
#define I2C_ADDRESS  0x20   // I2C BUS = LEDs address pin
#define LEDS_COUNT   10
/* DEFINE LINE TRACKING PINS */
#define PIN_TRACKING_LEFT  A1
#define PIN_TRACKING_CENTER A2
#define PIN_TRACKING_RIGHT  A3

/* DECLARE IMPORTANT VARIABLES */
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement
int draai = 0; // variable for the turning counter
int herhaling; // variable for the scanner repeater
int draaiteller;
int aanuit;

/* INITIALIZE SERVO */
Servo servo;              //create servo object
byte servoOffset = 0;     //change the value to calibrate servo, if needed.

/* INITIALIZE LED CONTROLLER */
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB); //LED strip controller
(must use, otherwise the LEDs do not function)

/* SETUP FUNCTION */
void setup() {
  !strip.begin(); // function that set ups and starts the LED strip
  Serial.begin(9600); // sets the data rate in bits per second for serial data transmission.
```

```
  pinMode(PIN_TRACKING_LEFT, INPUT); //
  pinMode(PIN_TRACKING_RIGHT, INPUT); //
  pinMode(PIN_TRACKING_CENTER, INPUT); //
  pinMode(PIN_DIRECTION_LEFT, OUTPUT); // sets the PIN_DIRECTION_LEFT to output mode
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT); // sets the PIN_MOTOR_PWM_LEFT to output mode
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT); // sets the PIN_DIRECTION_RIGHT to output mode
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT); // sets the PIN_MOTOR_PWM_RIGHT to output mode
  pinMode(PIN_BUZZER, OUTPUT); // sets the PIN_BUZZER to output mode
  pinMode(PIN_SONIC_TRIG, OUTPUT);// set trigPin to output mode
  pinMode(PIN_SONIC_ECHO, INPUT); // set echoPin to input mode
  servo.attach(PIN_SERVO);        //initialize servo motor
  servo.write(90 + servoOffset);  // change servoOffset to calibrate servomotor, but this
isn't needed.
}

/* (standard sketch) MOTOR DRIVER FUNCTION */
void motorRun(int speedl, int speedr) {
  int dirL = 0, dirR = 0; // declare variables
  if (speedl > 0) { //checks if speed LEFT is greater than 0
    dirL = 0; // if yes, set direction L to 0
  }
  else { //else set dirL to 1
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) { // same for this but inversed
    dirR = 1;
  }
  else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* SERVO ANGLE TURNING FUNCTION */
void angleTurner() {
  draai++; // adds +1 each time until it gets to an int of 10.
  Serial.print("Times robot has turned servomotor: ");  // prints how many times the
servomotor has turned
  Serial.print(draai); // variable that reference to how many times it has turned
  Serial.println("\n"); // prints new line
  Serial.print("Servomotor angle = "); // prints the exact angle of the servomotor
  Serial.print(angle); // variable that reference to the exact angle
  Serial.println("\n"); // prints new line
```

```
  servo.write(angle); // this function sets the servo to the variable, which is the right
angle
}

/* MEASURE DISTANCE FUNCTION */
void measureDistance() {
  digitalWrite(PIN_SONIC_TRIG, LOW); // Initializes the sonar sensor: set trigger pin to 0
  delayMicroseconds(2); // A delay of 2 micro seconds
  digitalWrite(PIN_SONIC_TRIG, HIGH); // Set tPin to 1: this emits the high frequency sound.
  delayMicroseconds(10); // Delays it for 10 micro seconds
  digitalWrite(PIN_SONIC_TRIG, LOW); // Set tPin to 0: stops emitting high frequency sound.
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Calculates duration between 0 to 1 to 0
  distance = duration * 0.034 / 2; // Time * Speed of Sound / 2. (retour: see theory)
  Serial.print("Afstand = "); // Prints “Afstand = “
  Serial.print(distance); // Prints the distance variable value. (could be anything)
  Serial.println(" centimeter\n"); // Prints “ centimeter“ with a new line.
}

/* ALARM LIGHTS (LEDs) FUNCTION */
void alarmLights() {
  if (distance <= 60 && distance >= 40) { // if measured distance is less and equal to 20
AND distance is greater and equal to 10 do:
    strip.setAllLedsColor(0x00FF00); // set color to orange (hexa)
  } else if (distance <= 40 && distance >= 20) {
    strip.setAllLedsColor(0xFFA500); // set color to orange (hexa)
  } else if (distance <= 20) { // else if the distance is less do:
    strip.setAllLedsColor(0xFF0000); // color code red ON
  } else {
    digitalWrite(PIN_BUZZER, LOW);  //turn off buzzer
    strip.setAllLedsColor(0, 0, 0); // turn of all lights
  }
}

/* MAIN LOOP FUNCTION */
void loop() {
  int z = 0;     // z is a simple variable that counts how many times the robot has to scan
(see for loop), this is set to a max. of 5 times.
  int angle = 0; // angle is also a variable that counts at what angle it is turning around,
this one is very important.

  if (herhaling < 2) { // Checks if the robot has scanned twice.
    for (angle = 0; angle < 140; delay(500)){ // For-loop, for the servo angle
      if (draai <= 9) { // code is going to check if the servomotor has turned at a max. of
10 times.
        angleTurner();  // Call angle turner function
        for(z = 0; z <= 4; z++) { // this loop, loops 5 times and stops after that 0-4.
          measureDistance(); // Call distance measuring function
          if (draai == 10) { // if the turns are equal to 10 times do: (this code patches
the glitch, of endless buzzing and light)
            digitalWrite(PIN_BUZZER, LOW);  //turn off buzzer
```

```
            strip.setAllLedsColor(0, 0, 0); // turn of all LEDS
          }
          alarmLights();  // call alarm lights (show LED colors)
        }
      }
      angle = angle + 35;   // sets angle to angle plus 35 degrees
      Serial.println("\n"); // prints new line
    }

    for (angle >= 140; angle >= 0; delay(500)){
      if (draai <= 9) {  // checks if draai is equal or less than 9, because Sonar has to
turn only 10 times, and not more!
        angleTurner();   // Call angle turner function
        for(z = 0; z <= 4; z++) { // this loop, loops 5 times and stops after that 0-4.
          measureDistance(); // Call distance measuring function
          if (draai == 10) { // if the turns are equal to 10 times do: (this code patches
the glitch, of endless buzzing and light)
            digitalWrite(PIN_BUZZER, LOW);  //turn off buzzer
            strip.setAllLedsColor(0, 0, 0); // turn of all LEDS
          }
          alarmLights();  // call alarm lights (show LED colors)
        }

        if (angle == 140) { // IF angle of servo is equal to 140 degrees, because otherwise
it will miss one angle (instead of x2).
          angleTurner();  // Call function angleTurner
          for(z = 0; z <= 4; z++) { // this loop, loops 5 times and stops after that 0-4.
            measureDistance();  // call function measureDistance
            if (draai == 10) { // if the turns are equal to 10 times do: (this code patches
the glitch, of endless buzzing and light)
            digitalWrite(PIN_BUZZER, LOW);  //turn off buzzer
            strip.setAllLedsColor(0, 0, 0); // turn of all LEDS
            }
            alarmLights();  // call function alarmLights
          }
        }
      }
      angle = angle - 35; // set angle to angle minus 35 degrees
      Serial.println("\n"); // print newline
    }
  }
  if (herhaling < 1) { // checks if herhaling is less than 1
    motorRun(100, 100); // go forward (motors on)
    delay(800); // delay 800 ms
    motorRun(0, 0); // stop all motors!
    for (int teller = 0; teller < 1;) { // loop 1x
      if (digitalRead(PIN_TRACKING_LEFT) == 1 && digitalRead(PIN_TRACKING_CENTER) == 1) { //
check if line tracking left and center are on
      motorRun(0, 0); // turn off motors
      teller++;  // add +1 to teller
```

```
      } else if (digitalRead(PIN_TRACKING_RIGHT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1) { // check if line tracking right and center are on
      motorRun(0, 0); // Stop all motors
      teller++; // add +1 to teller
      } else if (digitalRead(PIN_TRACKING_CENTER) == 1) { // check if line tracking center
is on
      motorRun(70, 70); // Turn motors on
      } else {
      motorRun(0, 0); // Stop all motors
      }
    }
    delay(1500);  // delay 1500 ms
    draai = 0;    // set draai to 0
    herhaling++;  // do herhaling + 1
  }
  if (draaiteller == 4) { // check if draaiTeller == 4 (times servo has turned)
    aanuit++; // add +1 to aanuit
  }
  if (aanuit == 1) {  // check if aanuit == 1
    motorRun(100, 100);  // motors on
    delay(600); // wait 0.6 seconds
    motorRun(-160, 160); // turn around
    delay(2300);  // for 2.3 seconds
    motorRun(0, 0); // stop all motors!
    aanuit++; // aanuit +1 (which now becomes 2)
    draai = 0;  // reset draai
    draaiteller = -1; // set draaiteller to -1
    herhaling = 1;  // set herhaling to 1
    /* Print variable draaiteller (x servo has turned and print herhaling (0 = before turn
and 1 = after 180 turn)) */
    Serial.print("\n");
    Serial.print(draaiteller);
    Serial.print("\n");
    Serial.print(herhaling);
    Serial.print("\n");
  }
  if (draaiteller < 3) {  // check if draaiteller is less than 3
    draaiteller++;  // if so: add +1 to draaiteller
    herhaling--;  // substract 1 from herhaling
    Serial.print("\n");
    Serial.print(herhaling);  // print herhaling
    Serial.print("\n");
  } else {
    draaiteller++;  // add +1 to draaiteller
  }
}
```

2.12.5 Code for sub question 3

```
/* INCLUDE NEEDED LIBRARY */
#include <IRremote.h>

/* DEFINE IR-TIMEOUT TIME AND CARSPEED */
#define IR_UPDATE_TIMEOUT     110
#define IR_CAR_SPEED          255
/* DEFINE MOTOR PINS */
#define PIN_DIRECTION_LEFT  4
#define PIN_DIRECTION_RIGHT 3
#define PIN_MOTOR_PWM_LEFT  6
#define PIN_MOTOR_PWM_RIGHT 5
/* DEFINE REMOTE PINS */
#define PIN_IRREMOTE_RECV 9
#define PIN_SPI_CE      9
#define PIN_SPI_CSN     10
#define PIN_SPI_MOSI    11
#define PIN_SPI_MISO    12
#define PIN_SPI_SCK     13
/* DEFINE IR REMOTE KEYCODES */
#define IR_REMOTE_KEYCODE_POWER    0xFFA25D
#define IR_REMOTE_KEYCODE_MENU    0xFF629D
#define IR_REMOTE_KEYCODE_MUTE    0xFFE21D
#define IR_REMOTE_KEYCODE_MODE    0xFF22DD
#define IR_REMOTE_KEYCODE_UP      0xFF02FD
#define IR_REMOTE_KEYCODE_BACK    0xFFC23D
#define IR_REMOTE_KEYCODE_LEFT    0xFFE01F
#define IR_REMOTE_KEYCODE_CENTER  0xFFA857
#define IR_REMOTE_KEYCODE_RIGHT   0xFF906F
#define IR_REMOTE_KEYCODE_0       0xFF6897
#define IR_REMOTE_KEYCODE_DOWN    0xFF9867
#define IR_REMOTE_KEYCODE_OK      0xFFB04F
#define IR_REMOTE_KEYCODE_1     0xFF30CF
#define IR_REMOTE_KEYCODE_2     0xFF18E7
#define IR_REMOTE_KEYCODE_3     0xFF7A85
#define IR_REMOTE_KEYCODE_4     0xFF10EF
#define IR_REMOTE_KEYCODE_5     0xFF38C7
#define IR_REMOTE_KEYCODE_6     0xFF5AA5
#define IR_REMOTE_KEYCODE_7     0xFF42BD
#define IR_REMOTE_KEYCODE_8     0xFF4AB5
#define IR_REMOTE_KEYCODE_9     0xFF52AD

/* INITIALIZE IR RECIEVER CONTROLLER INSTANCES */
IRrecv irrecv(PIN_IRREMOTE_RECV);
decode_results results;

/* U (unsigned) 32 (bits) VARIABLES */
u32 currentKeyCode, lastKeyCode;
u32 lastIRUpdateTime = 0;
```

```
/* GLOBAL VARIABLE(S) */
bool isStopFromIR = false;  // true or false

/* SETUP FUNCTION */
void setup() {
  irrecv.enableIRIn(); // START
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
}

/* MOTOR DRIVER FUNCTION */
void motorRun(int speedl, int speedr) {
  int dirL = 0, dirR = 0;
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* MAIN LOOP FUNCTION */
void loop() {
  if (irrecv.decode(&results)) { // Check if called function called address (&) of results
    isStopFromIR = false; // Set isStopFromIR to FALSE
    currentKeyCode = results.value; // Set currentKeyCode to results.value (value of the
result)
    if (currentKeyCode != 0xFFFFFFFF) { // Check if keyCode is not equal to nothing (error)
      Serial.println(currentKeyCode); // print keycode
      lastKeyCode = currentKeyCode; // set lastkeycode to the current one
    }
    switch (lastKeyCode) {  // switch statement (for more cases - in this case 1 case)
        case IR_REMOTE_KEYCODE_0:   // If keycode == keycode 0
          motorRun(IR_CAR_SPEED, IR_CAR_SPEED); // set speed to 255 (defined on start of
code)
          delay(500); // delay 500ms
```

```
          motorRun(0, 0); // stop all motors
          break;
        default: // (else) default state
          motorRun(0, 0); // stop all motors
          break;
    }
    irrecv.resume(); // Receive the next value
    lastIRUpdateTime = millis();  // Set lastIRUpdateTime to elapsed run time
  }
  else {
    if (millis() - lastIRUpdateTime > IR_UPDATE_TIMEOUT) // Check if elapsed time - last ir
updated time is greater than the update timeout (110)
    {
      if (!isStopFromIR) {  // check if IR is not stopped
        isStopFromIR = true;  // Set to true
        motorRun(0, 0); // stop all motors
      }
      lastIRUpdateTime = millis();  // Set lastIRUpdateTime to elapsed run time
    }
  }
}
```

2.13 STARR: Problem 2

Situation or Task

Situation is at home and at school.
Task: writing a professional document about project robotics and especially problem 2. Writing about this problem and working out the code and etc. will probably take up to 40 hours. I will finish this problem in about two weeks or so.

Action

1. Start my Word document
2. Begin to copy paste the syllabus questions into chapter problem 2.
3. Make up the neatness of the document: add chapters, paragraphs, page breaks and more.
4. Answer the all questions of the syllabus and work out the theoretical questions and add figures if needed.
5. Add references to the references list.
6. Open draw.io and make all the UML diagrams
7. Save them and paste them into Word document
8. Write codes according to UML diagrams (make changes, if needed)
9. Save codes and test them
10. Save serial output values for sub question
11. Paste the values into tables in Word document
12. Do a control check of problem 2, just to be sure that I have filled in everything.
13. Save the file
14. Upload it to Moodle.

Result

I wrote this code at school and at home, it took probably 12 days, which is estimated 52 hours. While the results were fine, there were things that could have been better. Some measurements of sub question 2 were not quite accurate. I probably could have improved this, but that will come later in follow-up research. The servo motor probably moved too fast while the robot was still scanning. Other than that, the results of sub question 3 did make reasonable sense and I had seen something like that coming. As distance traveled and time increases, the delta between theoretical and actual distance increases. This is due to the resistances taking place at that time. Indeed, it took me 35 hours. Including setting up/building robot, examining robot, debugging robot, creating UMLs, writing code, learning to write code and more.

Reflection

Otherwise, the other questions went well. I did find it unfortunate that the batteries came in so late, though, because that left me way behind and I'm sure I'm not the only one. Anyway, I also found it difficult at first to start programming because I have no experience with it. But I went through the C manual myself and watched some videos on YouTube, so that worked out fine in the end.

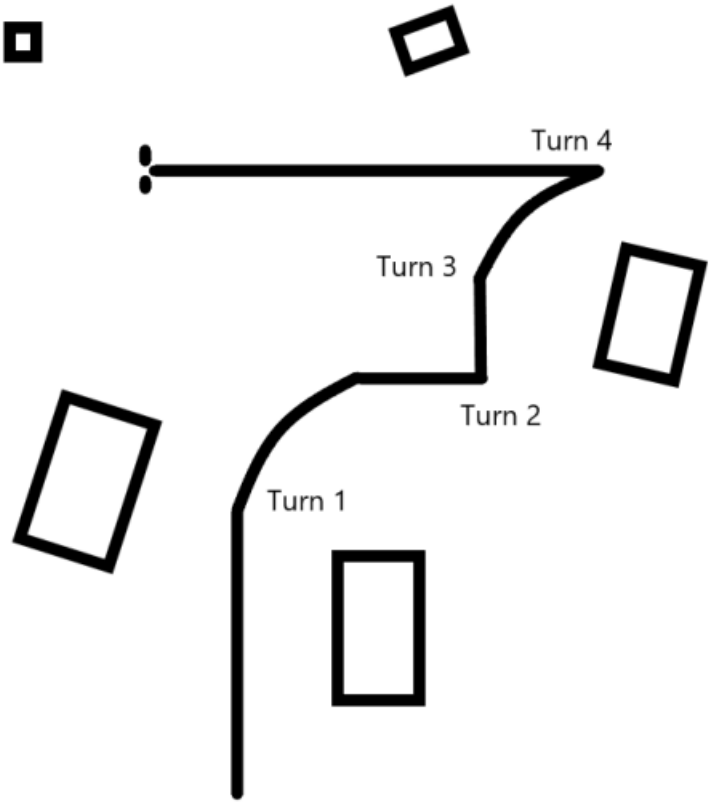
Problem 3 – Line Tracer and Object Finder

3.1 Summary

In this problem I made the robot follow a line and save measured distances until the end of the parkour. After the robot is done, it has to do an 180 degree turn and show with the LEDs the measured distances at the same place.

3.2 Motivation

Make the robot follow a line and find objects. Create a line pattern with a minimum of 4 turns, with at least one 45 degree and one 90 degree turn. Place at least 3 objects at least 30 cm from the line. The objects should be on both sided of the line. It also has to use LEDs for detecting objects.



3.3 Problem

There is no code available to detect objects and remembering positions. There is also no code for LEDs and following the line.

3.4 Goal

The robot should detect the objects on the way from the start to the finish and remember the positions and direction of the objects. After completion the robot should return from the finish to the start and light up the LEDs at the correct locations without detection, but by using its memory.

3.5 Results

The results were good, but also not good. The measuring process and time recording is going well. When the robot is finished it has to drive back to its starting place while showing the right colors in the right places. This goes pretty well until the robot gets to the turn, where it builds up deceleration, because the tires have too much friction with the ground and until the time the robot has accelerated, it has to stop again for the lights. This I do find unfortunate, but otherwise everything works!

I recorded a video, where the code works and where you also can see the delay building up, because of friction:

https://youtube.com/shorts/x5AOXFjo63s

rework version:

https://youtu.be/yHpYx57XF5Q

3.6 Conclusion

Looking at the results it can be seen that although the problem is solved, but the code is not quite optimal. In a subsequent study I might fix this. It's not a very big problem because the principle is there, it's just a matter of resistance and other things I should have taken into account.

3.7 Follow-up research

Looking at the result, I could improve some additional things. For example, I can add extra delay which then takes resistance into account.

3.8 References

3.8.1 Websites

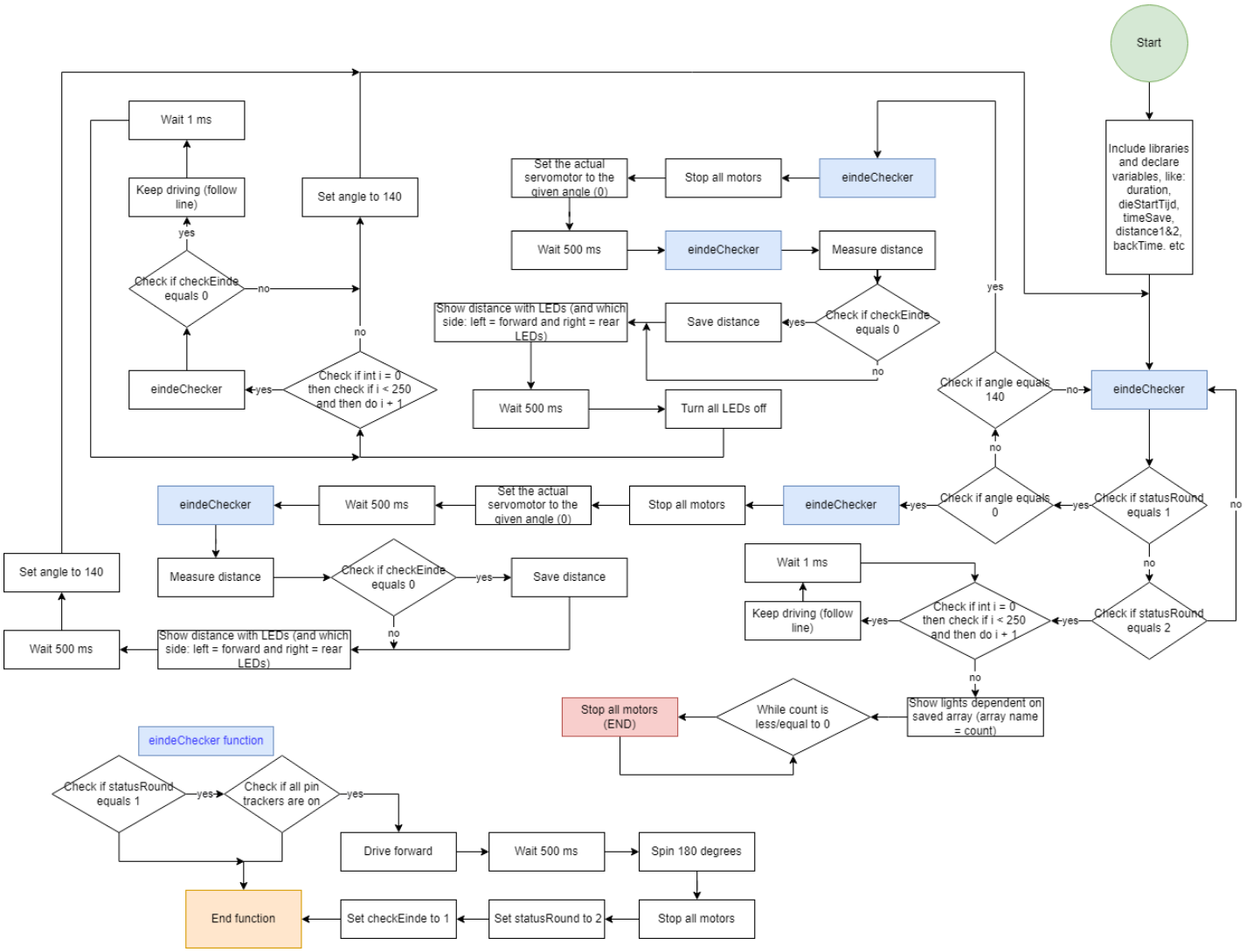
I have not used any websites

3.8.2 Figures

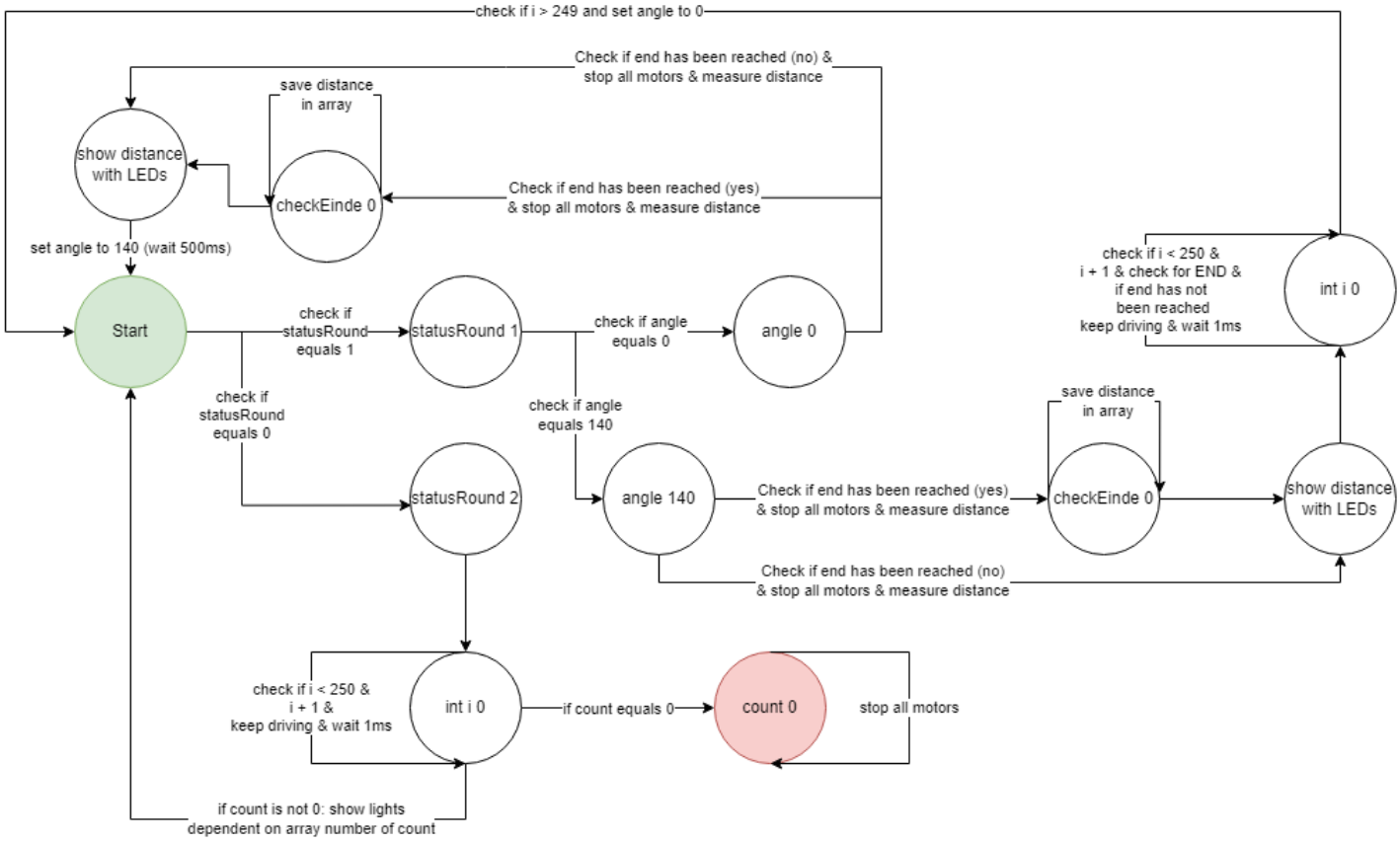
Draw.io - app.diagrams. 2007 - 2022 Jan Bodnar. Visited on 26-10-2022,
https://app.diagrams.net/

3.9 Code

3.9.1 UML flow diagram



3.9.2 UML state diagram



3.9.3 Arduino code problem 3

```
/* INCLUDE NEEDED LIBRARIES */
#include <Servo.h>
#include <Freenove_WS2812B_RGBLED_Controller.h>
/* SERVOMOTOR PIN: */
#define PIN_SERVO      2
/* SONAR PINS: */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* SONAR RELATED DEFINED VALUES: */
#define MAX_DISTANCE    300  // MAXIMUM DISTANCE MEASURABLE
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // CALCULATE TIMEOUT (increase max. distance)
#define SOUND_VELOCITY  340  // SPEED OF SOUND: 340m/s
/* MOTOR PINS: */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* LEDS PINS: */
#define I2C_ADDRESS  0x20
#define LEDS_COUNT   10
/* LINE TRACKING PINS: */
#define PIN_TRACKING_LEFT   A1
#define PIN_TRACKING_CENTER A2
#define PIN_TRACKING_RIGHT  A3

/* PUBLIC VARIABLE LIST */
long duration;              // variable for the duration of sound wave travel
unsigned long dieStartTijd; // variable for function millis(), which measures elapsed time
int timeSave[100] = {};     // Array for saving elapsed time
int distance, distance2;     // variable for the distance measurement (dis1 the initial var
and dis2 is for going back (memorize))
int backTime;               // This variable is needed to count backwards (elaps. time - 1,
per sec)
int angle = 0, count = 0, eindeTime = 0, count2 = 0;  // vars
int statusRound = 1;        // variable that sets the status to 1 or 2. 1 = measuring fase
and 2 = going back and showing colors fase
int dataSave[100] = {};     // Array for saving measured distance
int distanceRight, distanceLeft;
int y = 0;
int checkEinde = 0;
int notLeft = 0;

Servo servo;                // Create servo object
byte servoOffset = 0;       // Change the value to Calibrate servo
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB); // LED controller,
which sets up the parameters

/* SET UP FUNCTION */
void setup() {
```

```
  !strip.begin();           // Start LED strips
  Serial.begin(9600);       // Starts data communcation, which is needed for Serial Monitor
  /* Set up all digitalReads to the defined pins */
  pinMode(PIN_TRACKING_LEFT, INPUT);
  pinMode(PIN_TRACKING_RIGHT, INPUT);
  pinMode(PIN_TRACKING_CENTER, INPUT);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
  servo.attach(PIN_SERVO);
}

/* LED ALARM LIGHTS FUNCTION */
void alarmLights() {
  if (angle == 0) {                             // RIGHT SIDE (REAR
LEDS)
    if (distance <= 60 && distance >= 45) {        // Check if distance is less&equal to
1000 AND check if distance is more&equal to 50
      strip.setLedColor(5, 0, 255, 0);  // green
      strip.setLedColor(6, 0, 255, 0);  // green
      strip.setLedColor(7, 0, 255, 0);  // green
      strip.setLedColor(8, 0, 255, 0);  // green
      strip.setLedColor(9, 0, 255, 0);  // green
    } else if (distance <= 45 && distance >= 30){  //Check if distance is less&equal to 50
AND check if distance is more&equal to 20
      strip.setLedColor(5, 255, 127, 0);  // orange
      strip.setLedColor(6, 255, 127, 0);  // orange
      strip.setLedColor(7, 255, 127, 0);  // orange
      strip.setLedColor(8, 255, 127, 0);  // orange
      strip.setLedColor(9, 255, 127, 0);  // orange
    } else if (distance <= 30){                    // Check if distance is less&equal to 20
      strip.setLedColor(5, 255, 0, 0);  // red
      strip.setLedColor(6, 255, 0, 0);  // red
      strip.setLedColor(7, 255, 0, 0);  // red
      strip.setLedColor(8, 255, 0, 0);  // red
      strip.setLedColor(9, 255, 0, 0);  // red
    } else {                                       // IF ALL OF THESE ABOVE ARE ALL FALSE
DO:
      strip.setLedColor(5, 0, 0, 0);  // red
      strip.setLedColor(6, 0, 0, 0);  // red
      strip.setLedColor(7, 0, 0, 0);  // red
      strip.setLedColor(8, 0, 0, 0);  // red
      strip.setLedColor(9, 0, 0, 0);  // red
    }
  } else if (angle == 140) {
    if (distance <= 60 && distance >= 45) {        // Check if distance is less&equal to
1000 AND check if distance is more&equal to 50
```

```
      strip.setLedColor(0, 0, 255, 0);  // green
      strip.setLedColor(1, 0, 255, 0);  // green
      strip.setLedColor(2, 0, 255, 0);  // green
      strip.setLedColor(3, 0, 255, 0);  // green
      strip.setLedColor(4, 0, 255, 0);  // green
    } else if (distance <= 45 && distance >= 30){  // Check if distance is less&equal to 50
AND check if distance is more&equal to 20
      strip.setLedColor(0, 255, 127, 0);  // orange
      strip.setLedColor(1, 255, 127, 0);  // orange
      strip.setLedColor(2, 255, 127, 0);  // orange
      strip.setLedColor(3, 255, 127, 0);  // orange
      strip.setLedColor(4, 255, 127, 0);  // orange
    } else if (distance <= 30){                    // Check if distance is less&equal to 20
      strip.setLedColor(0, 255, 0, 0);  // red
      strip.setLedColor(1, 255, 0, 0);  // red
      strip.setLedColor(2, 255, 0, 0);  // red
      strip.setLedColor(3, 255, 0, 0);  // red
      strip.setLedColor(4, 255, 0, 0);  // red
    } else {                                       // IF ALL OF THESE ABOVE ARE ALL FALSE
DO:
      strip.setLedColor(0, 0, 0, 0);  // red
      strip.setLedColor(1, 0, 0, 0);  // red
      strip.setLedColor(2, 0, 0, 0);  // red
      strip.setLedColor(3, 0, 0, 0);  // red
      strip.setLedColor(4, 0, 0, 0);  // red
    }
  }
}

/* LINE FOLLOWING DRIVING FUNCTION */
void driving() {
  // Check if left&right are equal to 1 and center sensor is NULL
    if (digitalRead(PIN_TRACKING_CENTER) == 1){   // Check if sensor CENTER is equal to 1
    motorRun(100, 100);                             // Set motor PWM speed to L=100 AND
R=100
    notLeft = 0;
  } else if (digitalRead(PIN_TRACKING_LEFT) == 1){   // Check if sensor LEFT is equal to 1
    motorRun(-255, 255);
    notLeft = 0;                          // Correction
  } else if (digitalRead(PIN_TRACKING_RIGHT) == 1){  // Check if sensor RIGHT is equal to 1
    motorRun(255, -255);
    notLeft = 0;                            // Correction
  } else if (digitalRead(PIN_TRACKING_RIGHT) == 1 && digitalRead(PIN_TRACKING_CENTER) == 1){
// Check if sensors RIGHT AND CENTER are equal to 1
    motorRun(255, -255);
    notLeft = 0;                                                          // Extra
correction
  } else if (digitalRead(PIN_TRACKING_LEFT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1){  // Check if sensor LEFT AND CENTER are equal to 1
    motorRun(-255, 255);
```

```
    notLeft = 0;                                                    // Extra
correction
  // Check if ALL sensors are equal to NULL:
  } else if (digitalRead(PIN_TRACKING_LEFT) == 0 && digitalRead(PIN_TRACKING_CENTER) == 0 &&
digitalRead(PIN_TRACKING_RIGHT) == 0) {
    // motorRun(-255, 255);
    if (notLeft == 0) {
    motorRun(-245, 245);
    delay(300);
    }
    if (digitalRead(PIN_TRACKING_LEFT) == 0 && digitalRead(PIN_TRACKING_CENTER) == 0 &&
digitalRead(PIN_TRACKING_RIGHT) == 0) {
      motorRun(230, -230);
      notLeft = 1;
    }
  } else {            // Else, if other statements are false:
    motorRun(0, 0);   // Set motor PWM to 0
  }
}

/* MOTORRUN FUNCTION (from standard sketch) */
void motorRun(int speedl, int speedr) {
  int dirL = 0, dirR = 0;
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* DATA SAVER (saves measured distances in array) FUNCTION */
void dataSaver() {
  for (int j = 0; j < 1; j++) {         // Standard for-loop
    dataSave[count] = distance;         // Set array (count - which gets added by one each
time the function is called) equal to measured distance
    Serial.print("\n");                 // Print newLine
    Serial.print("dataSave waarde = "); // Print datasave value = ...
```

```
    Serial.print(dataSave[count]);      // Print the actual value from that array
    Serial.print("\ndataSave count = "); // Print datasave value = ...
    Serial.print(count);
    Serial.print("\n");                 // Print newLine
    count++;                            // Add +1 to variable "count"
  }                                     // END for-loop
}

/* TIME SAVER (saves elapsed time in array) FUNCTION */
void timeSaver() {
    timeSave[count2] = dieStartTijd;    // Set array (count2 - which gets added by one each
time the function is called) equal to Elapsed time var (dieStartTijd)
    Serial.print("\n");                 // Print newLine
    Serial.print("tijdSave waarde = "); // Print datasave value = ...
    Serial.print(timeSave[count2]);     // Print the actual value from that array
    Serial.print("\n");                 // Print newLine
    count2++;                           // Add +1 to variable "count2"
}

/* MEASURE DISTANCE FUNCTION */
void meten() {
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Set sonar pin to low
  delayMicroseconds(2);                 // 2 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, HIGH);   // Set sonar pin to high
  delayMicroseconds(10);                // 10 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Reads the echoPin, returns the sound wave travel
time in microseconds
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Recieve duration of sonar ECHO
  distance = duration * 0.034 / 2;      // distance becomes: duration of echo multiplied by
sound velocity and that all devided by 2 (retour of wave)
  // Serial.print("Measured distance = "); // Print measured distance = ...
  // Serial.print(distance);              // Print actual distance value
  // Serial.println(" cm\n");             // End with measured unit and a newLine
}

/* LINE TRACKING BACK DRIVING (but for going back, after 180 degree turn) FUNCTION */
void terugDriving() {
   if (digitalRead(PIN_TRACKING_CENTER) == 1){
    motorRun(100, 100);
    } else if (digitalRead(PIN_TRACKING_LEFT) == 1){
      motorRun(-255, 255);
    } else if (digitalRead(PIN_TRACKING_RIGHT) == 1){
      motorRun(255, -255);
    } else if (digitalRead(PIN_TRACKING_RIGHT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1){
      motorRun(255, -255);
    } else if (digitalRead(PIN_TRACKING_LEFT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1){
      motorRun(-255, 255);
```

```
    } else if (digitalRead(PIN_TRACKING_LEFT) == 0 && digitalRead(PIN_TRACKING_CENTER) == 0
&& digitalRead(PIN_TRACKING_RIGHT) == 0) {
      motorRun(255, -255);
      } else {
        motorRun(0, 0);
      }
}

/* LEDS DISTANCE SHOW (for showing distance on way back) FUNCTION */
void showLightsRight() {                              // RIGHT SIDE (REAR
LEDS)
  if (distanceRight <= 60 && distanceRight >= 45) {        // Check if distance is
less&equal to 1000 AND check if distance is more&equal to 50
      strip.setLedColor(5, 0, 255, 0);  // green
      strip.setLedColor(6, 0, 255, 0);  // green
      strip.setLedColor(7, 0, 255, 0);  // green
      strip.setLedColor(8, 0, 255, 0);  // green
      strip.setLedColor(9, 0, 255, 0);  // green
  } else if (distanceRight <= 45 && distanceRight >= 30) {  // Check if distance is
less&equal to 50 AND check if distance is more&equal to 20
      strip.setLedColor(5, 255, 127, 0);  // orange
      strip.setLedColor(6, 255, 127, 0);  // orange
      strip.setLedColor(7, 255, 127, 0);  // orange
      strip.setLedColor(8, 255, 127, 0);  // orange
      strip.setLedColor(9, 255, 127, 0);  // orange
  } else if (distanceRight <= 30) {                   // Check if distance is less&equal to
20
      strip.setLedColor(5, 255, 0, 0);  // red
      strip.setLedColor(6, 255, 0, 0);  // red
      strip.setLedColor(7, 255, 0, 0);  // red
      strip.setLedColor(8, 255, 0, 0);  // red
      strip.setLedColor(9, 255, 0, 0);  // red
    } else {                                       // IF ALL OF THESE ABOVE ARE ALL FALSE
DO:
      strip.setLedColor(5, 0, 0, 0);  // red
      strip.setLedColor(6, 0, 0, 0);  // red
      strip.setLedColor(7, 0, 0, 0);  // red
      strip.setLedColor(8, 0, 0, 0);  // red
      strip.setLedColor(9, 0, 0, 0);  // red
    }
}

void showLightsLeft() {
  if (distanceLeft <= 60 && distanceLeft >= 45) {        // Check if distance is less&equal
to 1000 AND check if distance is more&equal to 50
      strip.setLedColor(0, 0, 255, 0);  // green
      strip.setLedColor(1, 0, 255, 0);  // green
      strip.setLedColor(2, 0, 255, 0);  // green
      strip.setLedColor(3, 0, 255, 0);  // green
      strip.setLedColor(4, 0, 255, 0);  // green
```

```
    } else if (distanceLeft <= 45 && distanceLeft >= 30){  // Check if distance is
less&equal to 50 AND check if distance is more&equal to 20
      strip.setLedColor(0, 255, 127, 0);  // orange
      strip.setLedColor(1, 255, 127, 0);  // orange
      strip.setLedColor(2, 255, 127, 0);  // orange
      strip.setLedColor(3, 255, 127, 0);  // orange
      strip.setLedColor(4, 255, 127, 0);  // orange
    } else if (distanceLeft <= 30){                    // Check if distance is less&equal to
20
      strip.setLedColor(0, 255, 0, 0);  // red
      strip.setLedColor(1, 255, 0, 0);  // red
      strip.setLedColor(2, 255, 0, 0);  // red
      strip.setLedColor(3, 255, 0, 0);  // red
      strip.setLedColor(4, 255, 0, 0);  // red
     } else {                                        // IF ALL OF THESE ABOVE ARE ALL FALSE
DO:
      strip.setLedColor(0, 0, 0, 0);  // red
      strip.setLedColor(1, 0, 0, 0);  // red
      strip.setLedColor(2, 0, 0, 0);  // red
      strip.setLedColor(3, 0, 0, 0);  // red
      strip.setLedColor(4, 0, 0, 0);  // red
    }
}

/* LIGHTS CHECKER (checks which time stamp belongs to which time stamp was saved)FUNCTION */
void lightsChecker() {
  motorRun(0, 0);                          // Set motors to 0 PWM (off)
  count--;
  if (count > 0) {
      distanceRight = dataSave[count];
      Serial.println("\n DatasaveRight count = ");
      Serial.print(count);
      showLightsRight();
      count--;
      distanceLeft = dataSave[count];
      Serial.println("\n DatasaveLeft count = ");
      Serial.print(count);
      showLightsLeft();
      delay(1000);
      strip.setAllLedsColor(0, 0, 0);
  }
}

void eindeChecker() {
  if (statusRound == 1) {
    if (digitalRead(PIN_TRACKING_LEFT) == 1 && digitalRead(PIN_TRACKING_CENTER) == 1 &&
digitalRead(PIN_TRACKING_RIGHT) == 1) {
      motorRun(100, 100);  // drive
      delay(500);
      for (int i = 0; i < 1200; i++) { // Standard for-loop
```

```
        driving();               // Call terugDriving() function
        delay(1);                     // Delay 1 ms
      }                               // SUM delay = 250 ms
      eindeTime = 1;         // Set eindeTime to 0, which freezes the timer (millis())
      statusRound = 2;       // Set statusRound to 2, which means set status to 2 in the loop
function
      checkEinde = 1;
    }
  }
}

/* MAIN LOOP FUNCTION */
void loop() {
  if (eindeTime < 1) {              // Check if eindeTime (variable for freezing the timer)
is less than 1
    dieStartTijd = millis() / 1000; // Starts the timer and keeps adding one second, until
eindeTime becomes = 1
  }
  eindeChecker();
  if (statusRound == 1) {           // Check if status of robot is equal to 1 (1 = start,
measuring dist. etc. AND 2 = memorizing state while driving back to start)
  switch (angle) {                  // Switch statement for angle of servomotor (var: angle)
    case 0:                         // Case 0 (when angle = 0 degree)
      eindeChecker();
      motorRun(0, 0);                 // Set all motors off
      servo.write(angle);             // Set servo motor to given angle (which is 0 degree)
      Serial.print("Case 1:\n");      // Print "Case 1: " with a newLine
      delay(500);                     // Delay 500 ms
      eindeChecker();
      meten();                        // Call meten() function
      if (checkEinde == 0) {
        dataSaver();                    // Call dataSaver() function
      }
      alarmLights();                  // Call alarmLights() function
      delay(500);                     // Delay 500 ms, (SUM DELAY = 1000 ms)
      angle = 140;                    // Set angle variable to 140 (for the next case)
      break;                          // Break this case

    case 140:                          // Case 140 (when angle = 140 degrees)
      eindeChecker();
      motorRun(0, 0);                   // Set all motors off
      servo.write(angle);               // Set servo motor to given angle (which is 140
degrees)
      Serial.print("Case 140:\n");      // Print "Case 140: " with a newLine
      delay(500);                       // Delay 500 ms
      eindeChecker();
      // timeSaver();                      // Call timeSaver() function
      meten();                          // Call meten() function
      if (checkEinde == 0) {
        dataSaver();                   // Call dataSaver() function
```

```
      }                   // Call dataSaver() function
      alarmLights();                      // Call alarmLights() function
      delay(500);                         // Delay 500 ms
      strip.setAllLedsColor(0, 0, 0);     // Set all LEDs off
      for (int i = 0; i < 250; i++) { // Standard for-loop
        eindeChecker();
        if (checkEinde == 0) {
        driving();                        // Call driving() function
        delay(1);                         // Delay 1 ms
        }
      }                                  // SUM delay = 1000 ms, because for loop is ran 1000
times
      angle = 0;                          // Set angle to 0 degree
      break;                              // Break case: 140
    }
  } else if (statusRound == 2) {      // Check if status of robot is equal to 2 (going back
to start and showing distances with LEDs)
      for (int i = 0; i < 250; i++) { // Standard for-loop
        driving();                // Call terugDriving() function
        delay(1);                      // Delay 1 ms
      }                                // SUM delay = 250 ms
      lightsChecker();                 // Call lightsChecker() function
      while (count <= 0) {
        motorRun(0, 0);
      }
  }
}
```

3.10 STARR: Problem 3

Situation or Task

Situation is at home and at school.
Task: writing code for problem 3 and making sure it all works and writing an article about it. This problem will probably take up to 25 hours. I will finish this problem in about a weeks or so.

Action

1. Start draw.io and make UMLs
2. Start IDE and have the UMLs open on my other monitor
3. Start coding according to the UMLs
4. Test the code and improve when needed
5. Save the code
6. Make a video for proving the code actually works
7. Upload that video to YouTube
8. Answer the questions in Word document
9. Save everything
10. Upload it to Moodle.

Result

I wrote this code at school and at home, it took probably 9 days, which is estimated 46 hours I think. The results were good in themselves, the only problem was that the lights came on earlier than they should have, this is mainly due to the friction with the ground, so the acceleration is not enough. Adding an extra delay should fix this.

Reflection

Anyway, the code works and the problem is sufficiently solved. I do regret that at the last minute this is happening. Next time better!

Problem 4 – Database

4.1 Summary

In this problem, I first figured out how to build an interface for a database using the programming language c. In the process, we did an exercise with the EDS database in class and I learned a lot from that. I made a UML diagram and based on that I wrote the interface code for the f1 database. The code could have been made more universal, which would have had more than 400 lines, but unfortunately I had too little time and experience for that.

4.2 Motivation

Install the MariaDB server and MariaDB client in Debian. The MariaDB server is a program without a user interface. This is called a daemon. This program can only be approached via a network connection. MariaDB also create the client program. This program is a user interface on the command line and it converts commands from the user to network packets and sends it to the MariaDB server.

4.3 Problem

There is no MariaDB downloaded and installed in Debian and there is no database available yet. There is also no user interface for MariaDB, so I have to create one via c code.

4.4 Goal

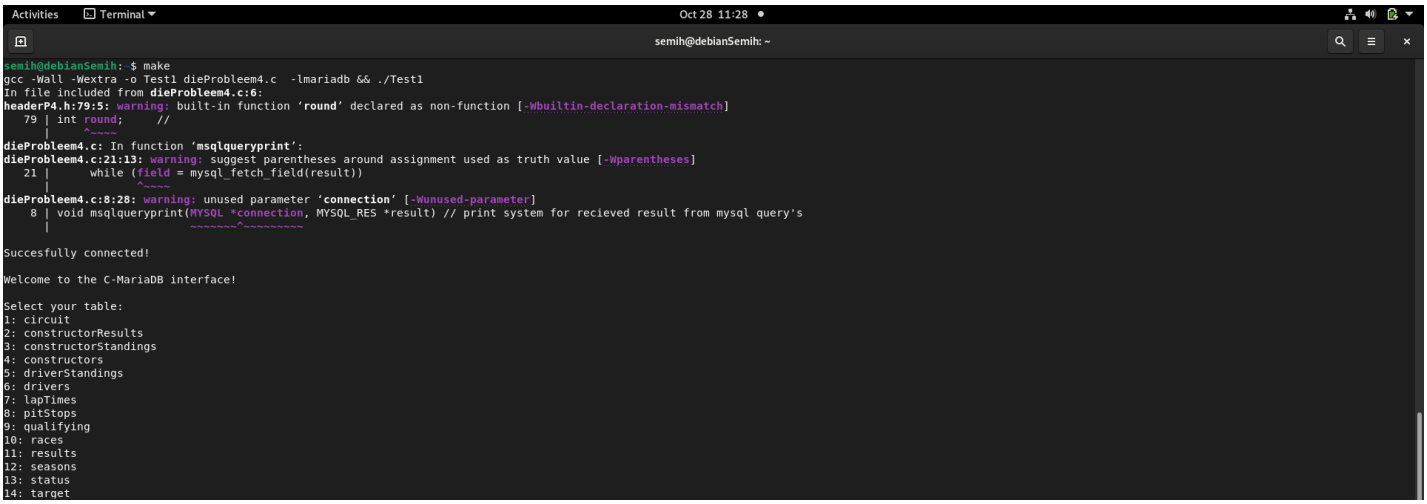
My goal is to write a code in c programming language that allows the user to interact with the f1 database. The user should be able to add, modify, delete and show all the rows in different tables

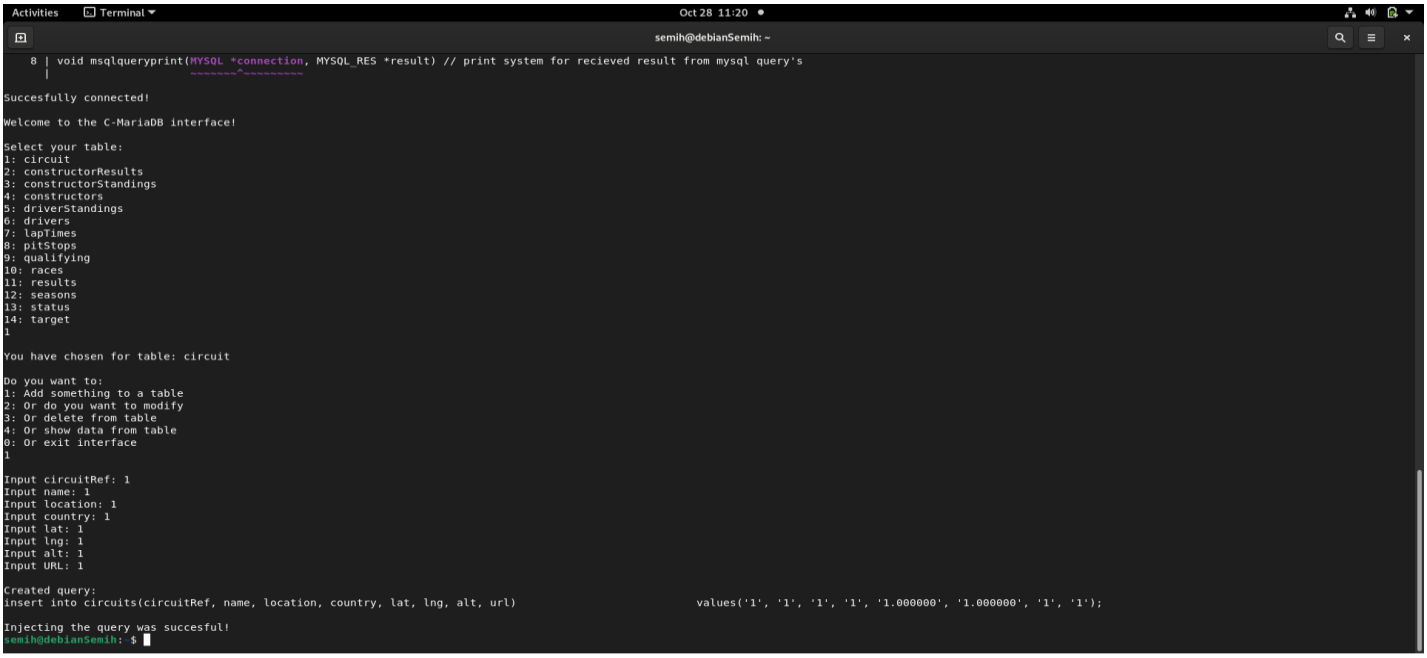
4.5 Results and screen output

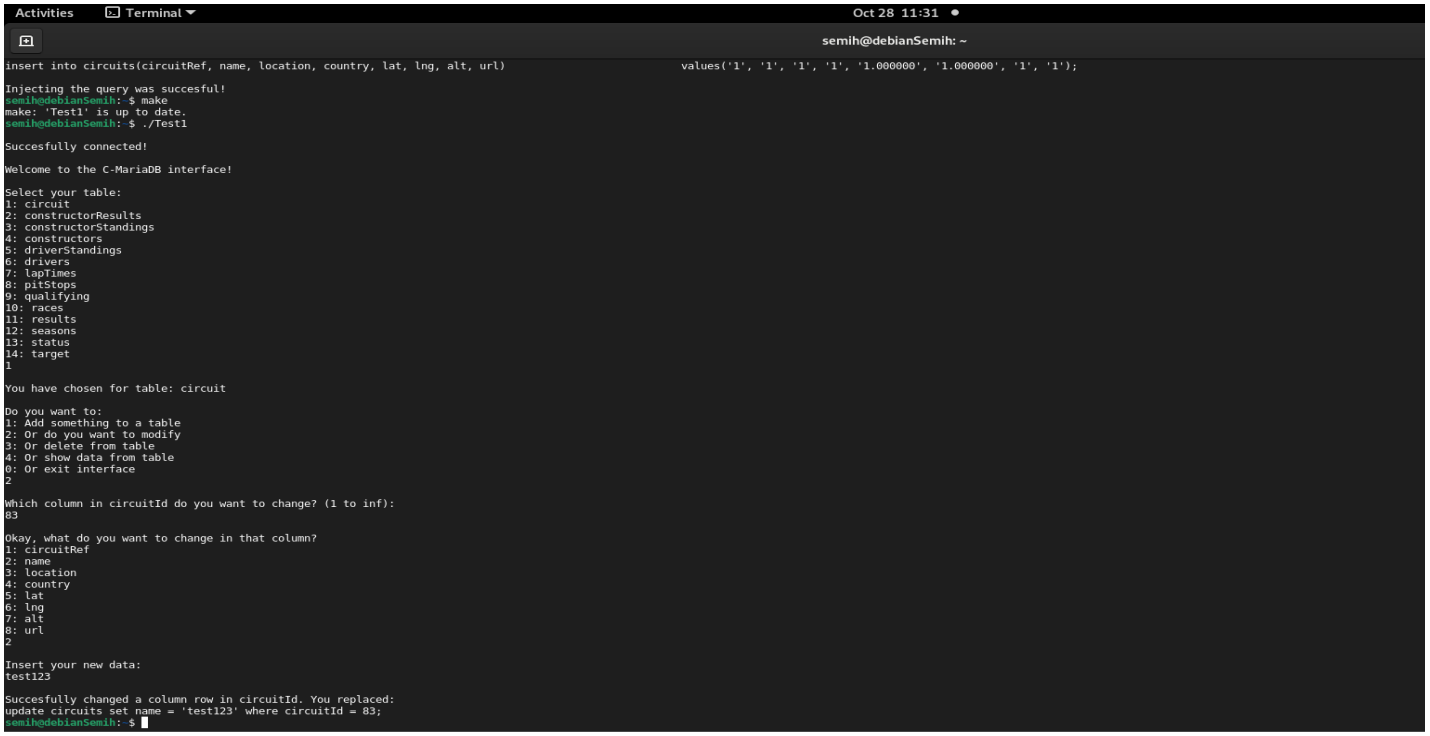
4.5.1 Results

The results, to my mind, were okay. Personally, I found the code to be very long. Unfortunately, I ran out of time to improve the code, otherwise I would have done so. Then I could have made the code more universal, reducing the number of lines. Other than that, the code does work and you can add, modify, delete and show rows from all 14 tables, so I am uber happy that I managed to get everything working. To import the f1 database I searched on the internet how to do that. I made myself root user and then I created a database with mariadb, then I used this command: "mysql -u username -p school < f1.sql"

4.5.2 screen output






Activities
Terminal
Oct 28 11:31
semih@debianSemih: ~
insert into circuits(circuitRef, name, location, country, lat, lng, alt, url) values('1', '1', '1', '1', '1.000000', '1.000000', '1', '1');
Injecting the query was succesful!
semih@debianSemih: $ make
make: 'Test1' is up to date.
semih@debianSemih: $ ./Test1
Succesfully connected!
Welcome to the C-MariaDB interface!
Select your table:
1: circuit
2: constructorResults
3: constructorStandings
4: constructors
5: driverStandings
6: drivers
7: lapTimes
8: pitStops
9: qualifying
10: races
11: results
12: seasons
13: status
14: target
1
You have chosen for table: circuit
Do you want to:
1: Add something to a table
2: Or do you want to modify
3: Or delete from table
4: Or show data from table
0: Or exit interface
2
Which column in circuitId do you want to change? (1 to inf):
83
Okay, what do you want to change in that column?
1: circuitRef
2: name
3: location
4: country
5: lat
6: lng
7: alt
8: url
2
Insert your new data:
test123
Succesfully changed a column row in circuitId. You replaced:
update circuits set name = 'test123' where circuitId = 83;
semih@debianSemih: $


make: 'Test1' is up to date.
semih@debianSemih: $ ./Test1
Succesfully connected!
Welcome to the C-MariaDB interface!
Select your table:
1: circuit
2: constructorResults
3: constructorStandings
4: constructors
5: driverStandings
6: drivers
7: lapTimes
8: pitStops
9: qualifying
10: races
11: results
12: seasons
13: status
14: target
1
You have chosen for table: circuit
Do you want to:
1: Add something to a table
2: Or do you want to modify
3: Or delete from table
4: Or show data from table
0: Or exit interface
3
Which column in circuitId do you want to delete? (1 to inf):
83
Succesfully deleted a column row in circuitId. You deleted:
delete from circuits where circuitId = 83;
semih@debianSemih: $


semih@debianSemih: $ make
make: 'Test1' is up to date.
semih@debianSemih: $ ./Test1
Succesfully connected!
Welcome to the C-MariaDB interface!
Select your table:
1: circuit
2: constructorResults
3: constructorStandings
4: constructors
5: driverStandings
6: drivers
7: lapTimes
8: pitStops
9: qualifying
10: races
11: results
12: seasons
13: status
14: target
1
You have chosen for table: circuit
Do you want to:
1: Add something to a table
2: Or do you want to modify
3: Or delete from table
4: Or show data from table
0: Or exit interface
0
semih@debianSemih: $

4.6 Conclusion

Looking at my results, I can conclude that the problem was successfully solved. However, I do regret that the code is so long.

4.7 Follow-up research

Looking back at the results, I find that everything works well, but what could definitely be improved is the length of the code. I could have made functions of parts that come back more often in the code. Also, I would have liked to try a next study to make a universal code. Such a code will then probably be faster and shorter!

4.8 References

4.8.1 Websites

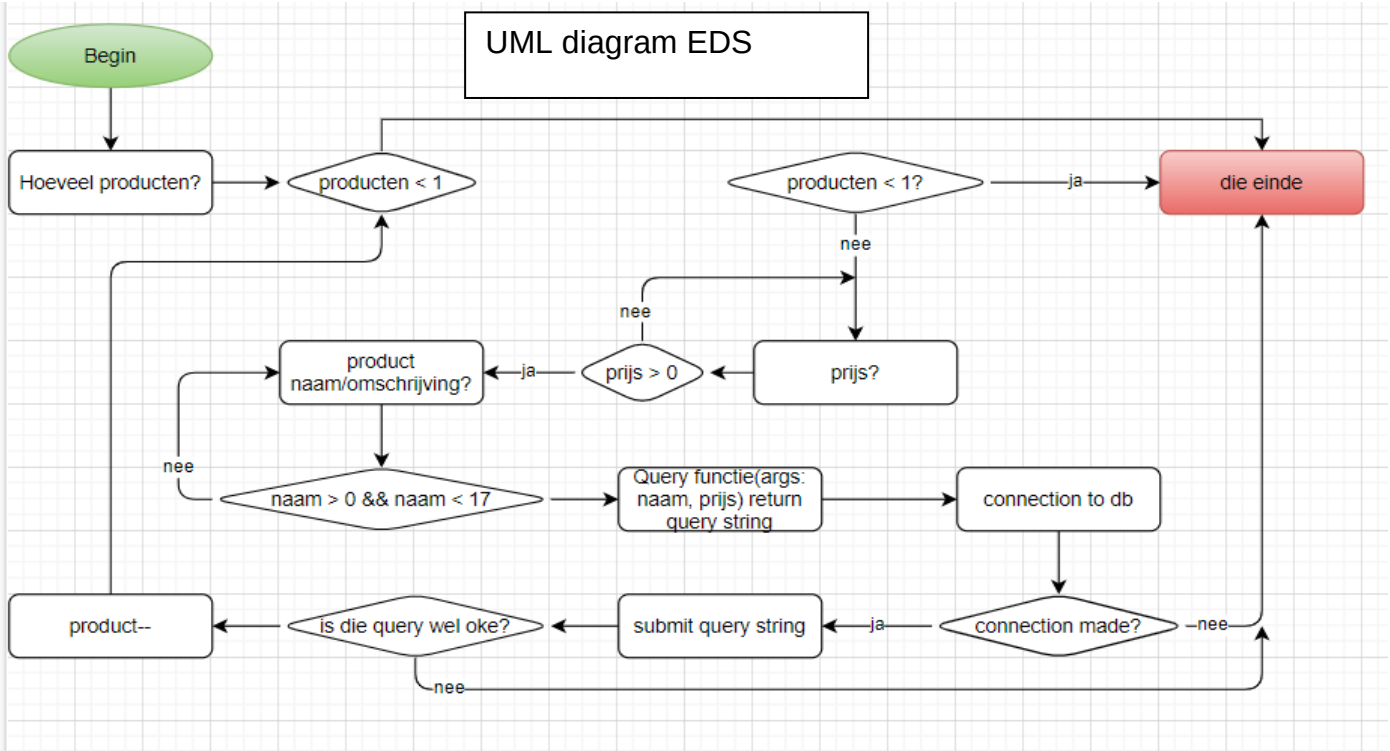
ZETCODE admin(at)zetcode.com. JGraph Ltd. 2005-2022. Visited on 26-10-2022,
https://zetcode.com/db/sqlitec/
DigitalOcean. DigitalOcean, LLC. All rights reserved. 2022. Visited on 23-10-2022,
https://www.digitalocean.com/community/tutorials/how-to-import-and-export-databases-in-mysql-or-mariadb

4.8.2 Figures

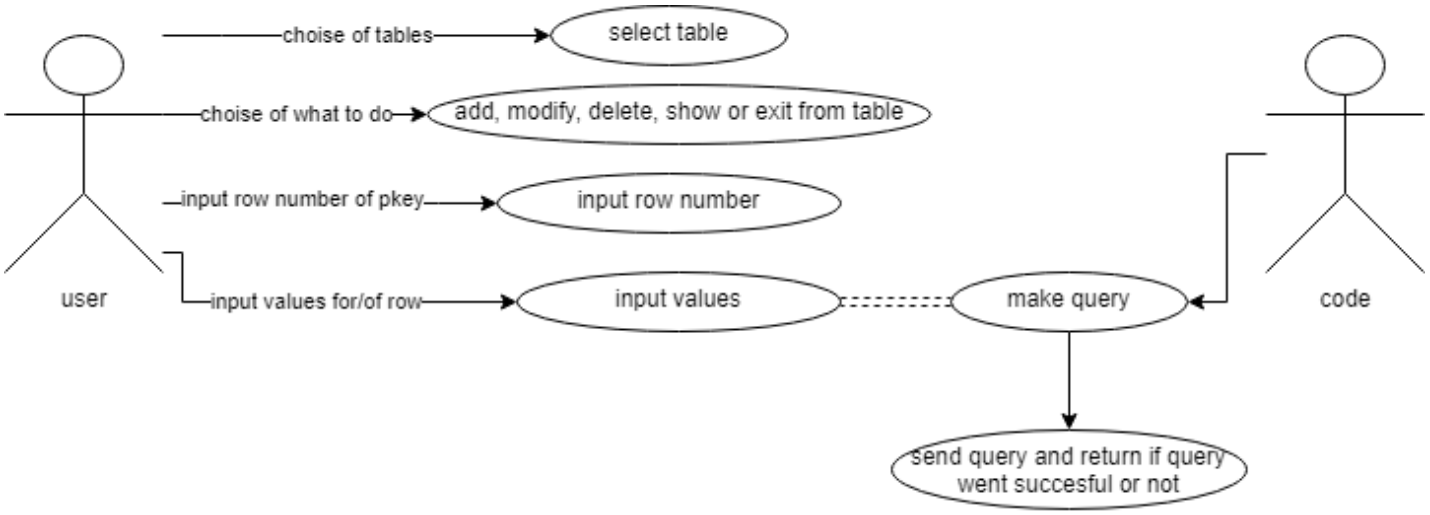
Draw.io - app.diagrams. 2007 - 2022 Jan Bodnar. Visited on 26-10-2022,
https://app.diagrams.net/

4.9 Code

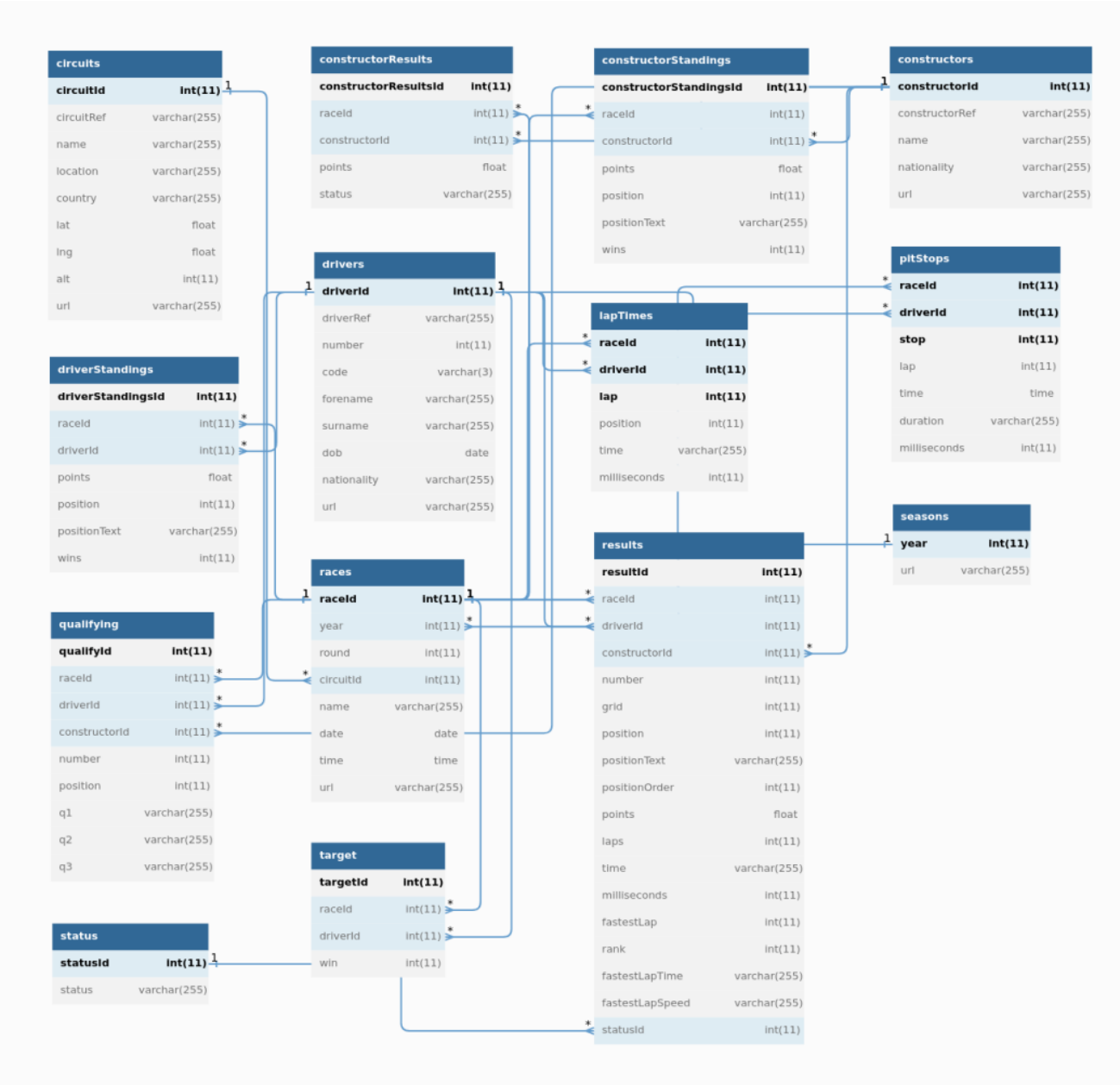
4.9.1 UML flow diagram for EDS db



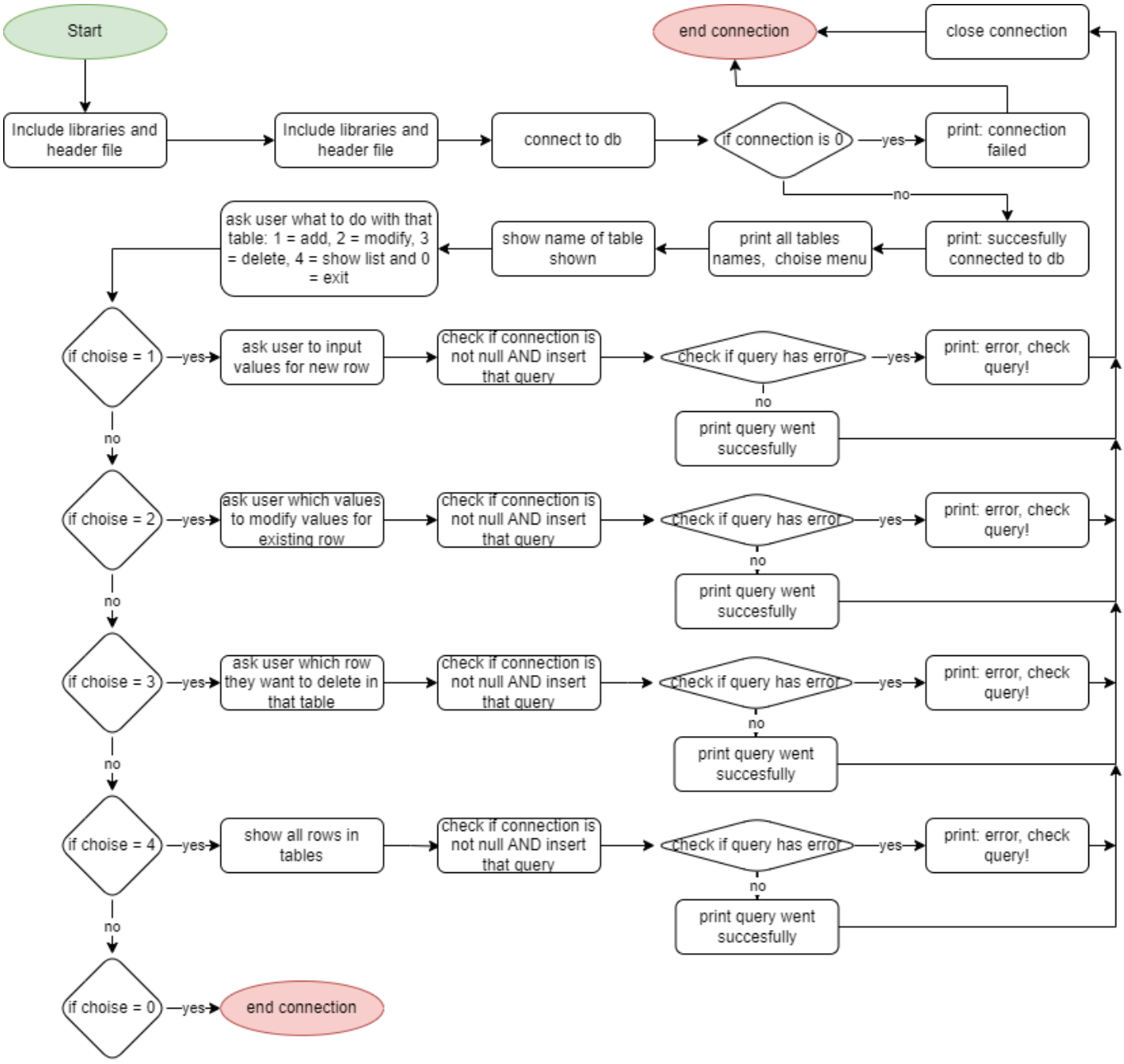
4.9.2 UML user case diagram



4.9.3 Relational model of f1 database



4.9.4 UML flow diagram f1 database interface



4.9.5 Main code for EDS DB

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

#define DEBUG

#define MINIMUM_AANTAL_PRODUCTEN    1
#define MAXIMUM_NAAM_LENGTE         16
#define MAXIMUM_QUERY_LENGTE        1024

int main(void)
{
    unsigned int uiAantalProducten;
    float fProductPrijs;
    char acNaamProduct[MAXIMUM_NAAM_LENGTE];
    char query[MAXIMUM_QUERY_LENGTE];

    /* create pointer to db connection */
    MYSQL *connection = mysql_init(NULL);

    char *user = "pipo";
    char *pass = "theclown";
    char *db = "EDS";

    /* hoeveel producten invoeren ? */
    printf("Hoeveel producten wil je invoeren?\n");
    scanf("%d", &uiAantalProducten);

#ifdef DEBUG
    printf("\nIngevoerde aantal producten: %d\n", uiAantalProducten);
#endif

    /* aantal producten > dan minimaal vereist? */
    if(uiAantalProducten < MINIMUM_AANTAL_PRODUCTEN)
    {
        printf("Onvoldoende producten in te voeren\n");
        return 0;
    }

    /* zolang aantal producten > 0... */
    while(uiAantalProducten > 0)
    {
#ifdef DEBUG
        printf("Aantal producten over: %d\n", uiAantalProducten);
#endif

        printf("\nGeef prijs van het product:\n");
        scanf("%f", &fProductPrijs);
```

```
#ifdef DEBUG
    printf("\nIngevoerde prijs: %.2f\n", fProductPrijs);
#endif

        printf("\nGeef naam van het product:\n");
        scanf("%s", acNaamProduct);

#ifdef DEBUG
        printf("\nIngevoerde product naam: %s\n", acNaamProduct);
#endif

        /* connect to db as username + password */
        mysql_real_connect(connection,
                           "localhost",
                           user,    // username
                           pass,    // password
                           db,      // database naam
                           0,
                           NULL,
                           0);

        /* check if connection is there */
        if(connection == NULL)
        {
            printf("\nConnection failed, bye!\n");

            mysql_close(connection);    // try close db connection

            return 0;
        }

#ifdef DEBUG
            printf("\nStill alive....\n");
#endif

        /* clear char array for query */
        memset(query, '\0', sizeof(query));

        sprintf(query,
            "insert into product(description, price) \
            values(\'%s\', %.2f);", acNaamProduct, fProductPrijs);

#ifdef DEBUG
        printf("\nCreated query:\n%s\n", query);
#endif

        // send query to db via connection
        if(mysql_query(connection, query) != 0)
        {
```

```
            printf("Mayby wrong username or password?\n");
            printf("%s\n", mysql_error(connection));

            mysql_close(connection);    // try close db connection

            return 0;
        }

#ifdef DEBUG
        printf("\nInserting product succesful!\n");
#endif
        uiAantalProducten = uiAantalProducten - 1;
    }

    mysql_close(connection);    // try close db connection

    return 0;
}
```

4.9.6 Makefile code for f1 DB interface

```
Problem4:   Probleem4.c
    gcc -Wall -Wextra -o Problem4 Probleem4.c $(mariadb_config --libs) -lmariadb
```

4.9.7 Header code for f1 DB interface

```
#ifndef HEADERP4_H
#define HEADERP4_H

/* debugger and maximal length for queries */
//#define DEBUG // If the debug is enabled, you will get extra prints in terminal (handy for testing). Turn off by making this line a comment (or not if you want to debug)
#define MAXIMUM_QUERY_LENGTE    4096

/* variable for query string */
char query[MAXIMUM_QUERY_LENGTE];

/* variables for DB connection */
char *user = "pipo";        // username
char *pass = "theclown";    // password
char *db = "school";        // database name

/* variables for choice menu */
int modifychoice;           //
char newData[256];          //
char columnreplacer[16];    //

// VARIABLES FOR ALL TABLES (*some variables may be double!*):
/* variables for circuits - extra comments: */
int circuitId;              //
char circuitRef[255];       //
char name[255];         //
char location[255];     //
char country[255];          //
float lat;                  //
float lng;                  //
int alt;                    //
char url[255];              //

/* variables for constructorResults - extra comments: */
int constructorResultsId;   //
int raceId;                 //
int constructorId;          //
float points;               //
char status[255];           //

/* variables for constructorStandings - extra comments: */
int constructorStandingsId; //
int position;               //
char positionText[255];     //
int wins;                   //

/* variables for constructors - extra comments: */
char constructorRef[255];   //
char nationality[255];      //
```

```
/* variables for driverStandings - extra comments: */
int driverStandingsId;      //
int driverId;               //

/* variables for drivers - extra comments: */
char driverRef[255];        //
int number;                 //
char code[3];               //
char forename[255];         //
char surname[255];          //
char dob[255];              // format: dd-mm-yyyy

/* variables for lapTimes - extra comments: */
int lap;                    //
char time[255];             // format: hh:mm:ss (00:00:00)
int milliseconds;           //

/* variables for pitStops - extra comments: */
int stop;                   //
char duration[255];         //

/* variables for qualifying - extra comments: */
int qualifyId;              //
char q1[255];               //
char q2[255];               //
char q3[255];               //

/* variables for races - extra comments: */
int year;                   //
int round;                  //
char date[255];             // format: 0000-00-00

/* variables for results - extra comments: */
int resultId;               //
int grid;                   //
int positionOrder;          //
int laps;                   //
int fastestLap;             //
int rank;                   //
char fastestLapTime[255];   //
char fastestLapSpeed[255];  //
int statusId;               //

/* variables for seasons - extra comments: */
// All variables are already declared!

/* variables for status - extra comments: */
// All variables are already declared!
```

```
/* variables for target - extra comments: */
int targetId;                //
int win;                     //

#endif
```

4.9.8 Main code for f1 DB interface

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

#include "headerP4.h"

void msqlqueryprint(MYSQL *connection, MYSQL_RES *result) // print system for recieved
result from mysql query's
            {
                int num_fields = mysql_num_fields(result); // set num_fields as number of
fields in mysql query result

                MYSQL_ROW row;
                MYSQL_FIELD *field;

                while ((row = mysql_fetch_row(result)))
                {
                for (int i = 0; i < num_fields; i++)
                {
                    if (i == 0)
                    {
                    while (field = mysql_fetch_field(result))
                    {
                        printf("%s ", field->name);
                    }
                    printf("\n");
                    }
                    printf("%s  ", row[i] ? row[i] : "NULL");
                }
                }
                mysql_free_result(result); // Clear result argument
                printf("\n");
            }

int main(void)
{
    /* create pointer to DB connection */
    MYSQL *connection = mysql_init(NULL);

    /* Connect to database with username + password and DB name */
    mysql_real_connect(connection,
                    "localhost",    // @localhost (device host)
                    user,           // username (pipo)
                    pass,           // password (theclown)
                    db,             // database naam (school)
                    0,              // NULL
                    NULL,           // NULL
                    0);             // NULL
```

```
    /* Checks if connection is there */
    if(connection == NULL)
    {
        printf("\nConnection failed, bye!\n");  // No connection? Print: "Connection failed,
bye!"
        mysql_close(connection);               // Now try to close database connection
        return 0;
    } else {                                   // If the connection wasn't failed:
    #ifdef DEBUG
        printf("\nSuccesfully connected!\n");
    #endif
    }

    int tablechoice;                // This is a variable for the choosing the table from
db: (1 - 14)

    /* Start the program by printing a welcome message and asking which table the user wants
to use: */
    printf("\nWelcome to the C-MariaDB interface!\n");
    printf("\nSelect your table: \n1: circuit\n2: constructorResults\n3:
constructorStandings\n4: constructors\n5: driverStandings\n6: drivers\n7: lapTimes\n8:
pitStops\n9: qualifying\n10: races\n11: results\n12: seasons\n13: status\n14: target\n");
    scanf("%d", &tablechoice);      // Scans for user-input (1 - 14 integer)

    switch (tablechoice)            // On the basis of the number selected, the following
case is performed (case 1 - 14):
    {
    case 1:                         // Case 1: table circuits
        printf("\nYou have chosen for table: circuit\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze;                  // This variable is for choosing what the user wants to
do with that table
        scanf("%d", &keuze);        // Scans for user-input (0 - 3)

        if (keuze == 1)             // Checks if keuze is equal to 1
        {
            /* The code will ask for the user to input values for the new list for in the
table */
            printf("\nInput circuitRef: ");
            scanf("%s", circuitRef);
            printf("Input name: ");
            scanf("%s", name);
            printf("Input location: ");
            scanf("%s", location);
            printf("Input country: ");
            scanf("%s", country);
            printf("Input lat: ");
            scanf("%f", &lat);
```

```
            printf("Input lng: ");
            scanf("%f", &lng);
            printf("Input alt: ");
            scanf("%d", &alt);
            printf("Input URL: ");
            scanf("%s", url);

            /* Clear character array for query */
            memset(query, '\0', sizeof(query)); // (basicly clears all values in variable
"query"", which is needed")

            /* This function adds a query to the variable query and this query will be
inserted into the db */
            sprintf(query,
                "insert into circuits(circuitRef, name, location, country, lat, lng, alt,
url) \
                values(\'%s\', \'%s\', \'%s\', \'%s\', \'%f\', \'%f\', \'%d\', \'%s\');",
                circuitRef, name, location, country, lat, lng, alt, url);

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);           // Little DEBUG line for be
secure
            #endif

            if(mysql_query(connection, query) != 0)            // Checks if the connection
failed or if the query is false.
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));    // Prints the error message
in output

                    mysql_close(connection);                   // Try close datebase
connection

                    return 0;
                }

            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");   // DEBUG line for making
sure everything went right
            #endif

            mysql_close(connection);                           // Close database
connection!
            break;
        }
        else if (keuze == 2)         // Checks if choice is equal to 2, which is MODIFYING a
line of data in a table.
        {
```

```
            /* Ask user to input circuitId (primary key (unique)) and what to change in that
specific line of data and to put in new data */
            printf("\nWhich column in circuitId do you want to change? (1 to inf): \n");
            scanf("%d", &circuitId);
            printf("\nOkay, what do you want to change in that column?\n1: circuitRef\n2:
name\n3: location\n4: country\n5: lat\n6: lng\n7: alt\n8: url\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);   // Scans for user-input of new data

            if (modifychoice == 1)  // Checks which number has been chosen (1 - 8). This one
checks it for modify choise == 1.
            {
                char columnreplacer[16] = "circuitRef";         // Variable for the name of
that variable from database, which is need for that specific variable
                memset(query, '\0', sizeof(query));             // Clear query values

                /* This function modifies a line of data in the table circuitId */
                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);   // UPDATE "table_name" SET "columnName)" = "newData" WHERE "tableId"
= "Id_of_that_table";

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query); // DEBUG line
                #endif

                if(mysql_query(connection, query) != 0)         // Checks if the connection
failed or if the query is false.
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));    // Prints the error message
in output

                    mysql_close(connection);                    // Try close datebase
connection

                    return 0;
                }
            }
            else if (modifychoice == 2)                         // ETC. for the rest: same
concept but with other variables
            {
                char columnreplacer[16] = "name";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);
```

```
                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "location";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "country";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
```

```
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "lat";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "lng";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
```

```
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
                char columnreplacer[16] = "alt";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 8)
            {
                char columnreplacer[16] = "url";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update circuits set %s = \'%s\' where circuitId = %d;", columnreplacer,
newData, circuitId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in circuitId. You replaced:
\n%s\n", query);
                #endif
```

```
                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze == 3) // Checks if keuze is equal to 3, which is: DELETE ROW
function.
        {
            /* Ask user to input ID of that specific tableId, that the user wants to delete
*/
            printf("\nWhich column in circuitId do you want to delete? (1 to inf): \n");
            scanf("%d", &circuitId);            // Scan for user-input which contains that
ID

                memset(query, '\0', sizeof(query));     // Clear query values

                sprintf(query,
                    "delete from circuits where circuitId = %d;", circuitId); // Query:
DELETE FROM "tableName" WHERE "tableID" = "userInputID";

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in circuitId. You deleted:
\n%s\n", query); // DEBUGGER
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // Try close db connection

                    return 0;
                }
        }
        else if (keuze == 4)
        {

            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from circuits;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
```

```
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query

        }
        else if (keuze == 0)    // Checks if keuze is equal to 0, which is exiting the
interface.
        {
            return 0;           // Exit program
        }
        else                    // If none of these numbers were put in: warn user and close
interface:
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }

    break;                      // END OF CASE 1 (table: CIRCUITID)

    case 2:
        printf("\nYou have chosen for table: constructorResults\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze2;
        scanf("%d", &keuze2);

        if (keuze2 == 1)
        {
            printf("\nInput raceId: ");
            scanf("%d", &raceId);
            printf("Input constructorId: ");
            scanf("%d", &constructorId);
            printf("Input points: ");
            scanf("%f", &points);
            printf("Input status: ");
            scanf("%s", status);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into constructorResults(raceId, constructorId, points, status) \
                values(\'%d\', \'%d\', \'%f\', \'%s\');",
                raceId, constructorId, points, status); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
```

```
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze2 == 2) // MODIFY
        {
            printf("\nWhich column in constructorResults do you want to change? (1 to inf):
\n");
            scanf("%d", &constructorResultsId);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
constructorId\n3: points\n4: status\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorResults set %s = \'%s\' where constructorResultsId =
%d;", columnreplacer, newData, constructorResultsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorResults. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
```

```
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "constructorId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorResults set %s = \'%s\' where constructorResultsId =
%d;", columnreplacer, newData, constructorResultsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorResults. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "points";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorResults set %s = \'%s\' where constructorResultsId =
%d;", columnreplacer, newData, constructorResultsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorResults. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
```

```
            {
                char columnreplacer[16] = "status";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorResults set %s = \'%s\' where constructorResultsId =
%d;", columnreplacer, newData, constructorResultsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorResults. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze2 == 3) // DELETE ROW
        {
            printf("\nWhich column in constructorResultsId do you want to delete? (1 to
inf): \n");
            scanf("%d", &constructorResultsId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from constructorResults where constructorResultsId = %d;",
constructorResultsId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in constructorResults. You
deleted: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
```

```
        }
        else if (keuze2 == 4)
        {

            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from constructorResults;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze2 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 3:
        printf("\nYou have chosen for table: constructorStandings\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze3;
        scanf("%d", &keuze3);

        if (keuze3 == 1)
        {
            printf("\nInput raceId: ");
            scanf("%d", &raceId);
            printf("Input constructorId: ");
            scanf("%d", &constructorId);
            printf("Input points: ");
            scanf("%f", &points);
            printf("Input position: ");
            scanf("%d", &position);
            printf("Input positionText: ");
            scanf("%s", positionText);
            printf("Input wins: ");
            scanf("%d", &wins);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
```

```
                "insert into constructorStandings(raceId, constructorId, points, position,
positionText, wins) \
                values(\'%d\', \'%d\', \'%f\', \'%d\', \'%s\', \'%d\');",
                raceId, constructorId, points, position, positionText, wins); // this is an
insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze3 == 2) // MODIFY
        {
            printf("\nWhich column in constructorStandingsId do you want to change? (1 to
inf): \n");
            scanf("%d", &constructorStandingsId);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
constructorId\n3: points\n4: position\n5: positionText\n6: wins\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
```

```
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "constructorId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "points";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
```

```
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "position";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "positionText";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
                #endif
```

```
                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "wins";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructorStandings set %s = \'%s\' where
constructorStandingsId = %d;", columnreplacer, newData, constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructorStandings. You
replaced: \n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze3 == 3) // DELETE ROW
        {
            printf("\nWhich column in constructorStandingsId do you want to delete? (1 to
inf): \n");
            scanf("%d", &constructorStandingsId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from constructorStandings where constructorStandingsId = %d;",
constructorStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in constructorStandings. You
deleted: \n%s\n", query);
```

```
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze3 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from constructorStandings;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze3 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 4:
        printf("\nYou have chosen for table: constructors\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze4;
        scanf("%d", &keuze4);
        if (keuze4 == 1)
        {
            printf("\nInput constructorRef: ");
            scanf("%s", constructorRef);
            printf("Input name: ");
            scanf("%s", name);
            printf("Input nationality: ");
            scanf("%s", nationality);
            printf("Input url: ");
            scanf("%s", url);
```

```
            /* clear char array for query */
        memset(query, '\0', sizeof(query));

        sprintf(query,
            "insert into constructors(constructorRef, name, nationality, url) \
            values(\'%s\', \'%s\', \'%s\', \'%s\');",
            constructorRef, name, nationality, url); // this is an insert query

        #ifdef DEBUG
        printf("\nCreated query:\n%s\n", query);
        #endif

        if(mysql_query(connection, query) != 0)
        {
            printf("Mayby wrong username or password?\n");
            printf("%s\n", mysql_error(connection));

            mysql_close(connection);    // try close db connection

            return 0;
        }

        // send query to db via connection
        #ifdef DEBUG
        printf("\nInjecting the query was succesful!\n");
        #endif

        mysql_close(connection);    // try close db connection
        break;
    }
    else if (keuze4 == 2) // MODIFY
    {
        printf("\nWhich column in constructorsId do you want to change? (1 to inf):
\n");
        scanf("%d", &constructorId);
        printf("\nOkay, what do you want to change in that column?\n1:
constructorRef\n2: name\n3: nationality\n4: url\n");
        scanf("%d", &modifychoice);
        printf("\nInsert your new data:\n");
        scanf("%s", newData);

        if (modifychoice == 1)
        {
            char columnreplacer[16] = "constructorRef";
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "update constructors set %s = \'%s\' where constructorId = %d;",
columnreplacer, newData, constructorId);
```

```
                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructors. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "name";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructors set %s = \'%s\' where constructorId = %d;",
columnreplacer, newData, constructorId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructors. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "nationality";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructors set %s = \'%s\' where constructorId = %d;",
columnreplacer, newData, constructorId);

                #ifdef DEBUG
```

```
                printf("\nSuccesfully changed a column row in constructors. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "url";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update constructors set %s = \'%s\' where constructorId = %d;",
columnreplacer, newData, constructorId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in constructors. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze4 == 3) // DELETE ROW
        {
            printf("\nWhich column in constructorId do you want to delete? (1 to inf): \n");
            scanf("%d", &constructorId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from constructors where constructorId = %d;", constructorId);

                #ifdef DEBUG
```

```
                printf("\nSuccesfully deleted a column row in constructors. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze4 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from constructors;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze4 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 5:
        printf("\nYou have chosen for table: driverStandings\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze5;
        scanf("%d", &keuze5);

        if (keuze5 == 1)
        {
            printf("\nInput raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input points: ");
```

```
            scanf("%f", &points);
            printf("Input position: ");
            scanf("%d", &position);
            printf("Input positionText: ");
            scanf("%s", positionText);
            printf("Input wins: ");
            scanf("%d", &wins);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into driverStandings(raceId, driverId, points, position,
positionText, wins) \
                values(\'%d\', \'%d\', \'%f\', \'%d\', \'%s\', \'%d\');",
                raceId, driverId, points, position, positionText, wins); // this is an
insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze5 == 2) // MODIFY
        {
            printf("\nWhich column in driverStandingsId do you want to change? (1 to inf):
\n");
            scanf("%d", &driverStandingsId);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
driverId\n3: points\n4: position\n5: positionText\n6: wins\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);
```

```
            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
```

```
            {
                char columnreplacer[16] = "points";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "position";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "positionText";
```

```
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "wins";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update driverStandings set %s = \'%s\' where driverStandingsId = %d;",
columnreplacer, newData, driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in driverStandings. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze5 == 3) // DELETE ROW
        {
            printf("\nWhich column in driverStandingsId do you want to delete? (1 to inf):
\n");
```

```
            scanf("%d", &driverStandingsId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from driverStandings where driverStandingsId = %d;",
driverStandingsId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in driverStandings. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze5 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from driverStandings;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze5 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;


    case 6:
        printf("\nYou have chosen for table: drivers\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
```

```
        int keuze6;
        scanf("%d", &keuze6);
        if (keuze6 == 1)
        {
            printf("\nInput driverRef: ");
            scanf("%s", driverRef);
            printf("Input number: ");
            scanf("%d", &number);
            printf("Input code (MAX: 3 chars): ");
            scanf("%s", code);
            printf("Input forename: ");
            scanf("%s", forename);
            printf("Input surname: ");
            scanf("%s", surname);
            printf("Input dob (ex: 1993-12-14): ");
            scanf("%s", dob);
            printf("Input nationality: ");
            scanf("%s", nationality);
            printf("Input url: ");
            scanf("%s", url);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into drivers(driverRef, number, code, forename, surname, dob,
nationality, url) \
                values(\'%s\', \'%d\', \'%s\', \'%s\', \'%s\', \'%s\', \'%s\', \'%s\');",
                driverRef, number, code, forename, surname, dob, nationality, url); // this
is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif
```

```
            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze6 == 2) // MODIFY
        {
            printf("\nWhich column in driverId do you want to change? (1 to inf): \n");
            scanf("%d", &driverId);
            printf("\nOkay, what do you want to change in that column?\n1: driverRef\n2:
number\n3: code (3 chars!)\n4: forename\n5: surname\n6: dob (date)\n7: nationality\n8:
url\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "driverRef";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "number";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
```

```
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "code";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "forename";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif
```

```
                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "surname";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "dob";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
```

```
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
                char columnreplacer[16] = "nationality";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 8)
            {
                char columnreplacer[16] = "url";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update drivers set %s = \'%s\' where driverId = %d;", columnreplacer,
newData, driverId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in drivers. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));
```

```
                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze6 == 3) // DELETE ROW
        {
            printf("\nWhich column in driverId do you want to delete? (1 to inf): \n");
            scanf("%d", &driverId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from drivers where driverId = %d;", driverId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in drivers. You deleted: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze6 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from drivers;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze6 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
```

```
            return 0;
        }
        break;

    case 7:
        printf("\nYou have chosen for table: lapTimes\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze7;
        scanf("%d", &keuze7);

        if (keuze7 == 1)
        {
            printf("\nInput raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input lap: ");
            scanf("%d", &lap);
            printf("Input position: ");
            scanf("%d", &position);
            printf("Input time (ex: 1:48.699): ");
            scanf("%s", time);
            printf("Input milliseconds: ");
            scanf("%d", &milliseconds);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into lapTimes(raceId, driverId, lap, position, time, milliseconds) \
                values(\'%d\', \'%d\', \'%d\', \'%d\', \'%s\', \'%d\');",
                raceId, driverId, lap, position, time, milliseconds); // this is an insert
query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
```

```
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze7 == 2) // MODIFY
        {
            printf("\nWhich column in lapTimes do you want to change? Please enter raceId (1
to inf): \n");
            scanf("%d", &raceId);
            printf("\nWhich column in lapTimes do you want to change? Please enter driverId
(1 to inf): \n");
            scanf("%d", &driverId);
            printf("\nWhich column in lapTimes do you want to change? Please enter \"lap\"
number (1 to inf): \n");
            scanf("%d", &lap);

            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
driverId\n3: lap\n4: position\n5: time (ex: 1:48.699)\n6: milliseconds\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
```

```
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "lap";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
```

```
                char columnreplacer[16] = "position";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "time";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "milliseconds";
```

```
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE lapTimes set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
lap = %d;", columnreplacer, newData, raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in lapTimes. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze7 == 3) // DELETE ROW
        {
            printf("\nWhich column in lapTimes do you want to delete? Please enter raceId (1
to inf): \n");
            scanf("%d", &raceId);
            printf("\nWhich column in lapTimes do you want to delete? Please enter driverId
(1 to inf): \n");
            scanf("%d", &driverId);
            printf("\nWhich column in lapTimes do you want to delete? Please enter \"lap\"
number (1 to inf): \n");
            scanf("%d", &lap);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from lapTimes WHERE raceId = %d AND driverId = %d AND lap =
%d;", raceId, driverId, lap);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in lapTimes. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));
```

```
                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze7 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from lapTimes;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze7 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 8:
        printf("\nYou have chosen for table: pitStops\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze8;
        scanf("%d", &keuze8);

        if (keuze8 == 1)
        {
            printf("\nInput raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input stop: ");
            scanf("%d", &stop);
            printf("Input lap: ");
            scanf("%d", &lap);
            printf("Input time (ex: 1:48.699): ");
            scanf("%s", time);
            printf("Input duration: ");
            scanf("%s", duration);
            printf("Input milliseconds: ");
```

```
            scanf("%d", &milliseconds);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into pitStops(raceId, driverId, stop, lap, time, duration,
milliseconds) \
                values(\'%d\', \'%d\', \'%d\', \'%d\', \'%s\', \'%s\', \'%d\');",
                raceId, driverId, stop, lap, time, duration, milliseconds); // this is an
insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze8 == 2) // MODIFY
        {
            printf("\nWhich column in pitStops do you want to change? Please enter raceId (1
to inf): \n");
            scanf("%d", &raceId);
            printf("\nWhich column in pitStops do you want to change? Please enter driverId
(1 to inf): \n");
            scanf("%d", &driverId);
            printf("\nWhich column in pitStops do you want to change? Please enter stop (1
to inf): \n");
            scanf("%d", &stop);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
driverId\n3: stop\n4: lap\n5: time (ex: 1:48.699)\n6: duration\n7: milliseconds\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);
```

```
            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
```

```
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "stop";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "lap";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
```

```
            {
                char columnreplacer[16] = "time";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "duration";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
```

```
                char columnreplacer[16] = "milliseconds";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE pitStops set %s = \'%s\' WHERE raceId = %d AND driverId = %d AND
stop = %d;", columnreplacer, newData, raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in pitStops. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze8 == 3) // DELETE ROW
        {
            printf("\nWhich column in pitStops do you want to delete? Please enter raceId (1
to inf): \n");
            scanf("%d", &raceId);
            printf("\nWhich column in pitStops do you want to delete? Please enter driverId
(1 to inf): \n");
            scanf("%d", &driverId);
            printf("\nWhich column in pitStops do you want to delete? Please enter \"stop\"
number (1 to inf): \n");
            scanf("%d", &stop);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from pitStops WHERE raceId = %d AND driverId = %d AND stop =
%d;", raceId, driverId, stop);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in pitStops. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
```

```
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze8 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from pitStops;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze8 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 9:
        printf("\nYou have chosen for table: qualifying\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze9;
        scanf("%d", &keuze9);

        if (keuze9 == 1)
        {
            printf("Input raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input constructorId: ");
            scanf("%d", &constructorId);
            printf("Input number: ");
            scanf("%d", &number);
            printf("Input position: ");
            scanf("%d", &position);
            printf("Input q1: ");
            scanf("%s", q1);
```

```
            printf("Input q2: ");
            scanf("%s", q2);
            printf("Input q3: ");
            scanf("%s", q3);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into qualifying(raceId, driverId, constructorId, number, position,
q1, q2, q3) \
                values(\'%d\', \'%d\', \'%d\', \'%d\', \'%d\', \'%s\', \'%s\', \'%s\');",
                raceId, driverId, constructorId, number, position, q1, q2, q3); // this is
an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze9 == 2) // MODIFY
        {
            printf("\nWhich column in qualifyId do you want to change? (1 to inf): \n");
            scanf("%d", &qualifyId);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
driverId\n3: constructorId\n4: number\n5: position\n6: q1\n7: q2\n8: q3\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
```

```
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "constructorId";
                memset(query, '\0', sizeof(query));
```

```
                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "number";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "position";
                memset(query, '\0', sizeof(query));

                sprintf(query,
```

```
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "q1";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
                char columnreplacer[16] = "q2";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);
```

```
                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 8)
            {
                char columnreplacer[16] = "q3";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update qualifying set %s = \'%s\' where qualifyId = %d;",
columnreplacer, newData, qualifyId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in qualifying. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze9 == 3) // DELETE ROW
        {
            printf("\nWhich column in qualifyId do you want to delete? (1 to inf): \n");
            scanf("%d", &qualifyId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from qualifying where qualifyId = %d;", qualifyId);
```

```
                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in constructors. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze9 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from qualifying;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze4 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 10:
        printf("\nYou have chosen for table: races\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze10;
        scanf("%d", &keuze10);

        if (keuze10 == 1)
        {
            printf("Input year: ");
            scanf("%d", &year);
            printf("Input round: ");
```

```
            scanf("%d", &round);
            printf("Input circuitId: ");
            scanf("%d", &circuitId);
            printf("Input name: ");
            scanf("%s", name);
            printf("Input date (ex: 0000-00-00): ");
            scanf("%s", date);
            printf("Input time (ex: 00:00:00): ");
            scanf("%s", time);
            printf("Input url: ");
            scanf("%s", url);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into races(year, round, circuitId, name, date, time, url) \
                values(\'%d\', \'%d\', \'%d\', \'%s\', \'%s\', \'%s\', \'%s\');",
                year, round, circuitId, name, date, time, url); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze10 == 2) // MODIFY
        {
            printf("\nWhich column in raceId do you want to change? (1 to inf): \n");
            scanf("%d", &raceId);
            printf("\nOkay, what do you want to change in that column?\n1: year\n2:
round\n3: circuitId\n4: name\n5: date\n6: time\n7: url\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
```

```
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "year";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "round";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
```

```
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "circuitId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "name";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
```

```
                char columnreplacer[16] = "date";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "time";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
                char columnreplacer[16] = "url";
                memset(query, '\0', sizeof(query));
```

```
                sprintf(query,
                    "update races set %s = \'%s\' where raceId = %d;", columnreplacer,
newData, raceId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in races. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze10 == 3) // DELETE ROW
        {
            printf("\nWhich column in raceId do you want to delete? (1 to inf): \n");
            scanf("%d", &raceId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from races where raceId = %d;", raceId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in races. You deleted: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze10 == 4)
        {
            memset(query, '\0', sizeof(query));
```

```
            sprintf(query,
                "select * from races;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze10 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 11:
        printf("\nYou have chosen for table: results\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze11;
        scanf("%d", &keuze11);

        if (keuze11 == 1)
        {
            printf("Input raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input constructorId: ");
            scanf("%d", &constructorId);
            printf("Input number: ");
            scanf("%d", &number);
            printf("Input grid: ");
            scanf("%d", &grid);
            printf("Input position: ");
            scanf("%d", &position);
            printf("Input positionText: ");
            scanf("%s", positionText);
            printf("Input positionOrder: ");
            scanf("%d", &positionOrder);
            printf("Input points: ");
            scanf("%f", &points);
            printf("Input laps: ");
            scanf("%d", &laps);
            printf("Input time (ex: +1:08.94): ");
            scanf("%s", time);
            printf("Input milliseconds: ");
```

```
            scanf("%d", &milliseconds);
            printf("Input fastestLap: ");
            scanf("%d", &fastestLap);
            printf("Input rank: ");
            scanf("%d", &rank);
            printf("Input fastestLapTime (ex: 1:29.558): ");
            scanf("%s", fastestLapTime);
            printf("Input fastestLapSpeed (ex: 213.166): ");
            scanf("%s", fastestLapSpeed);
            printf("Input statusId: ");
            scanf("%d", &statusId);

            /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into results(raceId, driverId, constructorId, number, grid,
position, positionText, positionOrder, points, laps, time, milliseconds, fastestLap, rank,
fastestLapTime, fastestLapSpeed, statusId) \
                values(\'%d\', \'%d\', \'%d\', \'%d\', \'%d\', \'%d\', \'%s\', \'%d\',
\'%f\', \'%d\', \'%s\', \'%d\', \'%d\', \'%d\', \'%s\', \'%s\', \'%d\');",
                raceId, driverId, constructorId, number, grid, position, positionText,
positionOrder, points, laps, time, milliseconds, fastestLap, rank, fastestLapTime,
fastestLapSpeed, statusId); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze11 == 2) // MODIFY
        {
            printf("\nWhich column in resultId do you want to change? (1 to inf): \n");
```

```
            scanf("%d", &resultId);
            printf("\nOkay, what do you want to change in that column?\n1: raceId\n2:
driverId\n3: constructorId\n4: number\n5: grid\n6: position\n7: positionText\n8:
positionOrder\n9: points\n10: laps\n11: time\n12: milliseconds\n13: fastestLap\n14:
rank\n15: fastestLapTime\n16: fastestLapSpeed\n17: statusId\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
```

```
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "constructorId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "number";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));
```

```
                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 5)
            {
                char columnreplacer[16] = "grid";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 6)
            {
                char columnreplacer[16] = "position";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection
```

```
                    return 0;
                }
            }
            else if (modifychoice == 7)
            {
                char columnreplacer[16] = "positionText";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 8)
            {
                char columnreplacer[16] = "positionOrder";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
```

```
            }
            else if (modifychoice == 9)
            {
                char columnreplacer[16] = "points";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 10)
            {
                char columnreplacer[16] = "laps";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 11)
```

```
            {
                char columnreplacer[16] = "time";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 12)
            {
                char columnreplacer[16] = "milliseconds";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 13)
            {
                char columnreplacer[16] = "fastestLap";
```

```
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 14)
            {
                char columnreplacer[16] = "rank";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 15)
            {
                char columnreplacer[16] = "fastestLapTime";
                memset(query, '\0', sizeof(query));
```

```
                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 16)
            {
                char columnreplacer[16] = "fastestLapSpeed";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 17)
            {
                char columnreplacer[16] = "statusId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
```

```
                    "update results set %s = \'%s\' where resultId = %d;", columnreplacer,
newData, resultId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in results. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze11 == 3) // DELETE ROW
        {
            printf("\nWhich column in resultId do you want to delete? (1 to inf): \n");
            scanf("%d", &resultId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from results where resultId = %d;", resultId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in results. You deleted: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze11 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from results;");
```

```
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze11 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 12:
        printf("\nYou have chosen for table: seasons\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze12;
        scanf("%d", &keuze12);

        if (keuze12 == 1)
        {
            printf("\nInput year value -> *NOTE: value < 1950 & value > 2017 PLEASE!*: \n");
            scanf("%d", &year);
            printf("Input url: ");
            scanf("%s", url);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into seasons(year, url) \
                values(\'%d\', \'%s\');",
                year, url); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
```

```
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze12 == 2) // MODIFY
        {
            printf("\nWhich column in seasons do you want to change? Please enter year value
(1 to inf): \n");
            scanf("%d", &year);
            printf("\nOkay, what do you want to change in that column?\n1: year\n2: url\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "year";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE seasons set %s = \'%s\' WHERE year = %d;", columnreplacer,
newData, year);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in seasons. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "url";
                memset(query, '\0', sizeof(query));

                sprintf(query,
```

```
                    "UPDATE seasons set %s = \'%s\' WHERE year = %d;", columnreplacer,
newData, year);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in seasons. You replaced:
\n%s\n", query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze12 == 3) // DELETE ROW
        {
            printf("\nWhich column in seasons do you want to delete? Please enter year value
(1 to inf): \n");
            scanf("%d", &year);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from seasons WHERE year = %d;", year);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in pitStops. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze12 == 4)
        {
            memset(query, '\0', sizeof(query));
```

```
            sprintf(query,
                "select * from seasons;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze12 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;

    case 13:
        printf("\nYou have chosen for table: status\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze13;
        scanf("%d", &keuze13);

        if (keuze13 == 1)
        {
            printf("Input status: ");
            scanf("%s", status);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into status(status) \
                values(\'%s\');",
                status); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
```

```
                }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze13 == 2) // MODIFY
        {
            printf("\nWhich column in statusId do you want to change? (1 to inf): \n");
            scanf("%d", &statusId);
            printf("\nOkay, what do you want to change in that column?\n1: status\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "status";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "update status set %s = \'%s\' where statusId = %d;", columnreplacer,
newData, statusId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in statusId. You replaced:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
        }
        else if (keuze13 == 3) // DELETE ROW
        {
            printf("\nWhich column in statusId do you want to delete? (1 to inf): \n");
            scanf("%d", &statusId);

                memset(query, '\0', sizeof(query));
```

```
                sprintf(query,
                    "delete from status where statusId = %d;", statusId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row from statusId. You deleted:
\n%s\n", query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze13 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from status;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze13 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }

    break;

    case 14:
        printf("\nYou have chosen for table: target\n");
        printf("\nDo you want to: \n1: Add something to a table\n2: Or do you want to
modify\n3: Or delete from table\n4: Or show data from table\n0: Or exit interface\n");
        int keuze14;
        scanf("%d", &keuze14);

        if (keuze14 == 1)
```

```
        {
            printf("\nInput targetId: ");
            scanf("%d", &targetId);
            printf("Input raceId: ");
            scanf("%d", &raceId);
            printf("Input driverId: ");
            scanf("%d", &driverId);
            printf("Input win: ");
            scanf("%d", &win);

                /* clear char array for query */
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "insert into target(targetId, raceId, driverId, win) \
                values(\'%d\', \'%d\', \'%d\', \'%d\');",
                targetId, raceId, driverId, win); // this is an insert query

            #ifdef DEBUG
            printf("\nCreated query:\n%s\n", query);
            #endif

            if(mysql_query(connection, query) != 0)
            {
                printf("Mayby wrong username or password?\n");
                printf("%s\n", mysql_error(connection));

                mysql_close(connection);    // try close db connection

                return 0;
            }

            // send query to db via connection
            #ifdef DEBUG
            printf("\nInjecting the query was succesful!\n");
            #endif

            mysql_close(connection);    // try close db connection
            break;
        }
        else if (keuze14 == 2) // MODIFY
        {
            printf("\nWhich column in target do you want to change? Please enter targetId (1
to inf): \n");
            scanf("%d", &targetId);

            printf("\nOkay, what do you want to change in that column?\n1: targetId\n2:
raceId\n3: driverId\n4: win\n");
            scanf("%d", &modifychoice);
            printf("\nInsert your new data:\n");
```

```
            scanf("%s", newData);

            if (modifychoice == 1)
            {
                char columnreplacer[16] = "targetId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE target set %s = \'%s\' WHERE targetId = %d;", columnreplacer,
newData, targetId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in target. You replaced: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 2)
            {
                char columnreplacer[16] = "raceId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE target set %s = \'%s\' WHERE targetId = %d;", columnreplacer,
newData, targetId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in target. You replaced: \n%s\n",
query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
```

```
            }
            else if (modifychoice == 3)
            {
                char columnreplacer[16] = "driverId";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE target set %s = \'%s\' WHERE targetId = %d;", columnreplacer,
newData, targetId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in target. You replaced: \n%s\n",
query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
            else if (modifychoice == 4)
            {
                char columnreplacer[16] = "win";
                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "UPDATE target set %s = \'%s\' WHERE targetId = %d;", columnreplacer,
newData, targetId);

                #ifdef DEBUG
                printf("\nSuccesfully changed a column row in target. You replaced: \n%s\n",
query);
                #endif


                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
            }
```

```
        }
        else if (keuze14 == 3) // DELETE ROW
        {
            printf("\nWhich column in target do you want to delete? Please enter targetId (1
to inf): \n");
            scanf("%d", &targetId);

                memset(query, '\0', sizeof(query));

                sprintf(query,
                    "delete from target WHERE targetId = %d;", targetId);

                #ifdef DEBUG
                printf("\nSuccesfully deleted a column row in target. You deleted: \n%s\n",
query);
                #endif

                if(mysql_query(connection, query) != 0)
                {
                    printf("Mayby wrong username or password?\n");
                    printf("%s\n", mysql_error(connection));

                    mysql_close(connection);    // try close db connection

                    return 0;
                }
        }
        else if (keuze14 == 4)
        {
            memset(query, '\0', sizeof(query));

            sprintf(query,
                "select * from target;");
            mysql_query(connection, query);
            MYSQL_RES *result = mysql_store_result(connection); // store results of query
                    printf("\nPrinting selected table...\n\n");
                    msqlqueryprint(connection, result); // print results of query
                }
        else if (keuze14 == 0)
        {
            return 0;
        }
        else
        {
            printf("\nThat is not an option! Closing...\n");
            return 0;
        }
        break;
    }
}
```

4.10 STARR: Problem 4

Situation or Task

Situation is at school and home.
Task: installing MariaDB server and client in Debian. After that I have to write a C code that can interact with the MariaDB server. The user interface has to do following things: view, add, modify and delete several tables and columns of data inside of tables. I also have make a relational model of the database that I use (F1). Before writing the code I also have to make an UML diagram and state diagram.

Action

1. Open my code editor (IDE)
2. Make UML and state diagram
3. Put these diagrams in Word document
4. Write the code using these diagrams
5. Test it and improve it if needed
6. Done? Save everything
7. Copy and paste code into Word document
8. Save document
9. Upload file to Moodle

Result

I wrote this code at school and at home, it took probably 6 days, which is estimated 33 hours I think. To import the f1 database I searched on the internet how to do that. I made myself root user and then I created a database with mariadb, then I used this command: "mysql -u username -p school < f1.sql". The result is okay, because I think if I had some more knowledge in programming, that the code would have been a bit shorter. But I got it working, so I am fine with that. The Makefile and header files were also not that difficult to make. The code itself is also not that difficult to understand. Another thing was that I used the old syllabus, which said that there should be one source file, but in the new updated syllabus they changed it to “more” source files. This problem itself just took a lot of time, that’s it.

Reflection

My experience with writing this code was okay. I just started learning C and I had to use the snowboard code to make the other code work.
I have learnt how to write a C code that is interactable with the user, which can input values into a DB.

Problem 5 – Exit Finder

5.1 Summary

In this problem, I had to figure out how to get the robot to make sure it can find the exit in a box and then drive straight back in and stop.

5.2 Motivation

Make the robot find the exit. Place the robot in a large box with an hole matching the size of the robot. The robot should search and find the exit and pass through it, then move back in and go in a straight line to the starting point. Make sure the robot cannot detect the exit from the starting point without moving first.

5.3 Problem

There is no code available to find the exit and go back into the box in a straight line.

5.4 Goal

My goal is to make a code that will let the robot find an exit. The robot has to go through the hole in a straight line and do an 180 degree turn and go back into the box.

5.5 Results

I honestly think the results were okay, but the code is not quite optimal yet. In a next research, I would have had more time to make the robot drive back in a straight line better by measuring. I thought about implementing this, but I ended up not doing it because I also had to study for math and solve the other problems. Finding the exit could have been better. Now the robot measures the two differences in terms of distance, but I actually wanted it calculated with a mathematical formula.

Below you can find a link of the video:

https://youtu.be/8BwtveRf2D0

rework version:

https://youtube.com/shorts/xrM_6-0-OjA?feature=share

5.6 Conclusion

The problem is solved, but the code is still not quite the way I wanted it. So my conclusion is that the problem is solved, but the code could have been better, perhaps something for future follow-up research.

5.7 Follow-up research

If there is ever a follow-up research, I want to make sure this robot knows how to find the exit in a mathematical way and then drive forward without hitting the walls. In doing so, I also want to make sure that when the robot returns to its box, it also properly measures the distances around it.

5.8 References

5.8.1 Websites

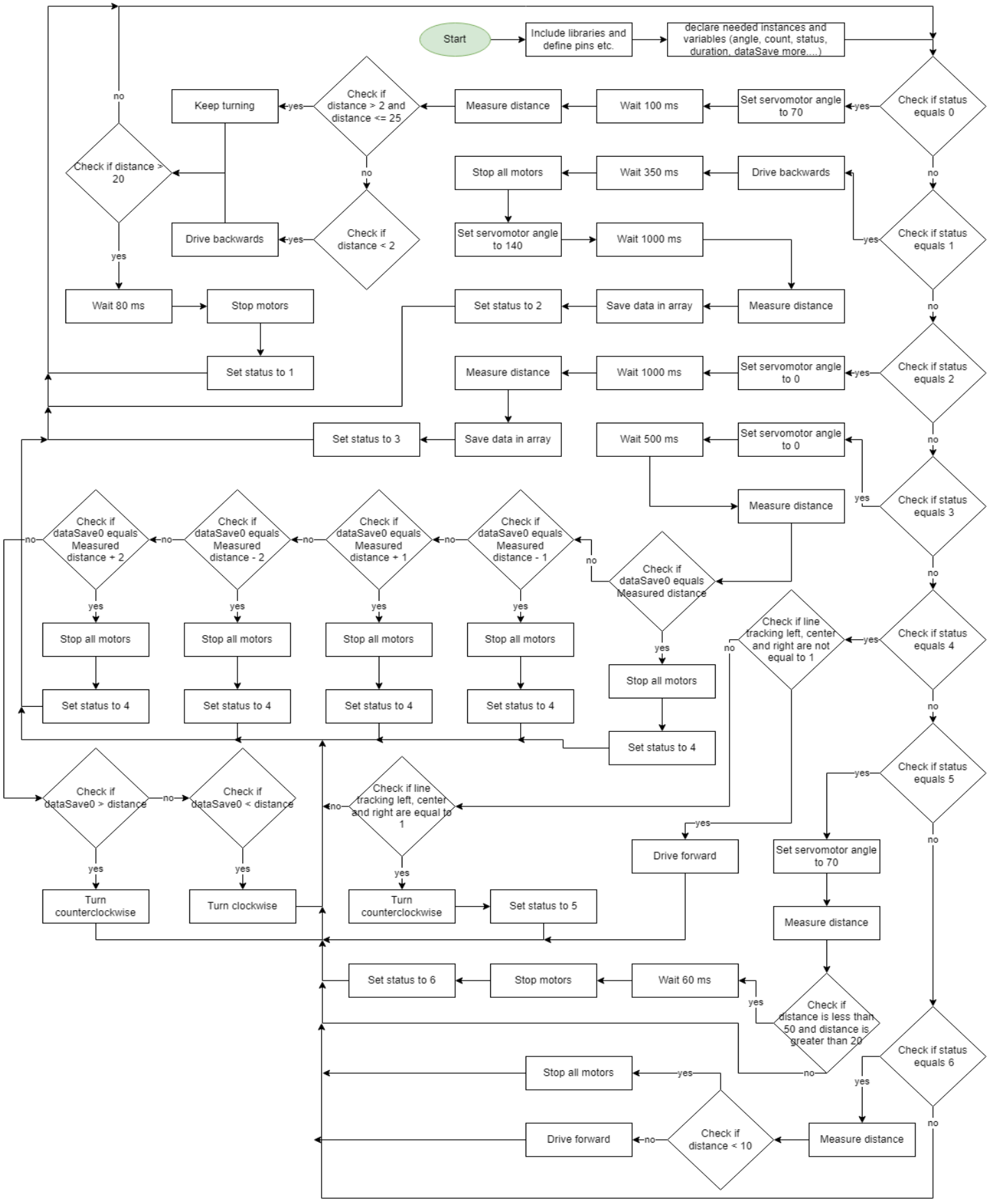
I have not used any sources online, I only used my brain and draw.io for making UML diagram

5.8.2 Figures

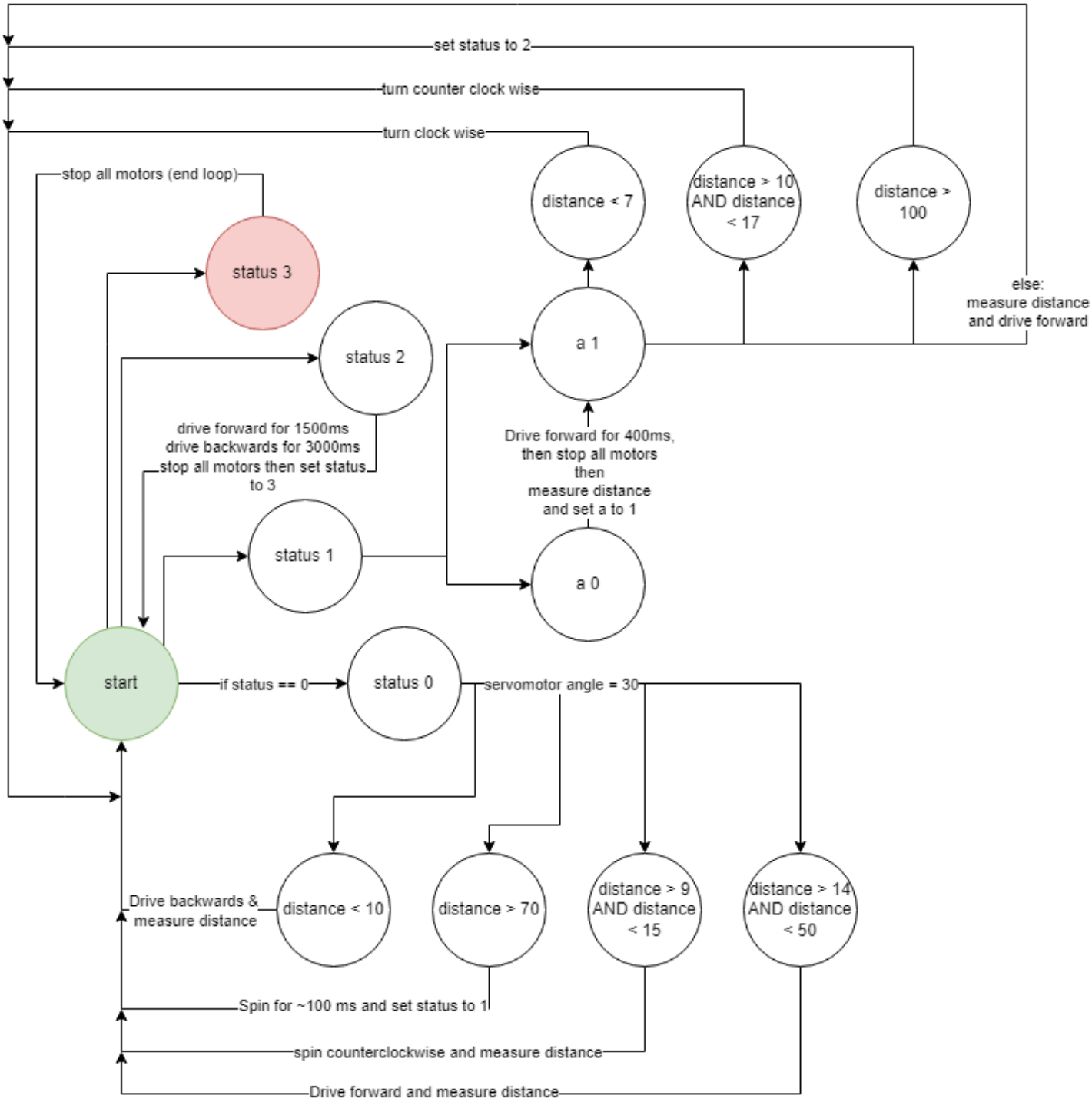
Draw.io - app.diagrams. JGraph Ltd. 2005-2022. Visited on 26-10-2022,
https://app.diagrams.net/

5.9 Code

5.9.1 UML flow diagram



5.9.2 UML state diagram



5.9.3 Code for problem 5

```
/* INCLUDE NEEDED LIBRARIES */
#include <Servo.h>
#include <Freenove_WS2812B_RGBLED_Controller.h>
/* SERVOMOTOR PIN: */
#define PIN_SERVO      2
/* SONAR PINS: */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* SONAR RELATED DEFINED VALUES: */
#define MAX_DISTANCE    300  // MAXIMUM DISTANCE MEASURABLE
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // CALCULATE TIMEOUT (increase max. distance)
#define SOUND_VELOCITY  340  // SPEED OF SOUND: 340m/s
/* MOTOR PINS: */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* LEDS PINS: */
#define I2C_ADDRESS  0x20
#define LEDS_COUNT   10
/* LINE TRACKING PINS: */
#define PIN_TRACKING_LEFT   A1
#define PIN_TRACKING_CENTER A2
#define PIN_TRACKING_RIGHT  A3

/* GLOBAL IMPORTANT VARIABLES */
long duration; // variable for the duration of sound wave travel
int status = 0; // Status of robot (finding exit or going back into box)
int distance; // variable for the distance measurement
int angle = 0; // angle = angle of servomotor and count for counting +1 each time for
datasave
int turnCounter = 0;
int turnChecker = 0;
int a = 0;

/* INSTANCES AND LED CONTROLLER */
Servo servo;              //create servo object
byte servoOffset = 0;     //change the value to Calibrate servo
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB);

/* SETUP FUNCTION */
void setup() {
  !strip.begin();
  Serial.begin(9600);
  pinMode(PIN_TRACKING_LEFT, INPUT);
  pinMode(PIN_TRACKING_RIGHT, INPUT);
  pinMode(PIN_TRACKING_CENTER, INPUT);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
```

```
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
  servo.attach(PIN_SERVO);
  servo.write(90 + servoOffset);
}

/* MOTOR DRIVING FUNCTION */
void motorRun(int speedl, int speedr) {  // Functie ddie ervoor zorgt dat de motoren goed
functioneren. Het is een void, dus zonder return.
  int dirL = 0, dirR = 0;
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* MEASURE DISTANCE FUNCTION */
void meten() {
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Set sonar pin to low
  delayMicroseconds(2);                 // 2 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, HIGH);   // Set sonar pin to high
  delayMicroseconds(10);                // 10 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Reads the echoPin, returns the sound wave travel
time in microseconds
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Recieve duration of sonar ECHO
  distance = duration * 0.034 / 2;      // distance becomes: duration of echo multiplied by
sound velocity and that all devided by 2 (retour of wave)
  Serial.print("Measured distance = "); // Print measured distance = ...
  Serial.print(distance);               // Print actual distance value
  Serial.println(" cm\n");              // End with measured unit and a newLine
}

/* EXIT FINDER FUNCTION */
void exitFinder() {
```

```
  servo.write(30);
  meten();  // Measure distance
  if (distance > 70) { // Check if there is and exit
    motorRun(180, -180);  // correction
    delay(100);
    status = 1;
  } else if (distance < 10) { // distance is too little
    motorRun(-100, -100); // drive backwards
    meten();  // measure distance
  } else if (distance > 9 && distance < 15) { // spin around until another distance has been
seen
    motorRun(-180, 180);
    meten();
  } else if (distance > 14 && distance < 50) { // check for oppertunity for driving forward
    motorRun(80, 80);
    meten();
  }
}

/* MAIN LOOP FUNCTION */
void loop() {
  switch (status) { // Switch case of status
    case 0: // Case 0:
      exitFinder(); // Run exitFinder code
      break;
    case 1:
      servo.write(140); // set angle of servo to 140
      delay(100); // wait 100ms
      meten();  // measure distance
      if (a == 0) { // check if a equals 0
        motorRun(80, 80); // drive forward
        delay(400); // wait 400 ms
        motorRun(0, 0); // stop all motors
        meten();  // measure distance
        a = 1;  // set a to 1
      }
        if (distance < 7) { // check if distance is less than 7
          motorRun(140, -180); // turn around cwise
        } else if (distance > 10 && distance < 17) { // else check if distance > 10 and
check if distance < 17
          motorRun(-140, 140);  // turn ccwise
        } else if (distance > 100) { // check if distance is > 100 (exit has been found)
          status = 2; // status becomes 2
        } else {  // else
           meten(); // measure distance
          motorRun(70, 70);   // drive forward
        }
      break;
    case 2:
      motorRun(100, 100); // drive forwards
```

```
      delay(1500);  // wait 1.5s
      motorRun(-100, -100);   // drive backwards
      delay(3000);  // wait 3s
      motorRun(0, 0); // stop all motors
      servo.write(70);  // set servo to middle (70)
      delay(500); // wait 500ms
      status = 3; // set status to 3 (end)
      break;
    case 3: // END
      motorRun(0, 0); // stop all motors
      break;
  }
}
```

5.10 STARR: Problem 5

Situation or Task

Situation is at school and home.
Task: my job I to make sure the robot can find an exit from a box with an exit. Once the robot finds the exit, it must follow it and then drive straight ahead. Then it has to make a turn and go straight back into the box.

Action

1. Open my browser and go to draw.io
2. Make UML diagram and state diagram
3. Save those
4. Paste them into Word document
5. Use them while writing Arduino code
6. Test code often (debugging code)
7. Add extra or change stuff if UML is not right
8. Change code
9. Change UML
10. Save both if done
11. Copy and paste code (and UML) into Word document
12. Answer questions in Word document
13. Save document
14. Upload to Moodle

Result

I wrote this code at school and at home, it took probably 6 days, which is estimated 32 hours I think. I think the result is not too good, although the code works. I just couldn't make more out of it than this. Maybe someday in follow-up research I can improve the code.

Reflection

My experience writing the code was okay, but I found the logic a bit tricky, especially since I didn't have too much time left because the batteries came in so late.

Problem 6 – Parking

6.1 Summary

In this problem, I figured out how to make a robot parallel park between two objects with one ultrasonic sensor. The results were successful and the code is not too long!

6.2 Motivation

The robot must be able to parallel park between two objects. Therefore, my motivation is to have the robot be able to park between a large parking space and be able to park between a short parking space.

6.3 Problem

Design and implement code to make the robot perform a parallel parking in between two objects.

6.4 Goal

My goal is to have the robot parallel park itself between two objects without hitting the objects. In addition, the robot must also be parked straight in.

6.5 Results

Vincent and I made a video of the robot running on the Arduino code I wrote. Vincent helped me a little with calibrating the robot. Then we made this video and I uploaded it to YouTube. I personally liked the results, some things could have been better, but I thought the basic principle was good enough. Parking didn't go very well at times, the wheels have a deviation to the right, but otherwise it actually went well.

Here is the link of the video we've made:
https://youtu.be/786aZiHS3WU

6.6 Conclusion

From my results, you can see that the code itself works well. Consider parking, occasionally the robot does get a little too close to the wall/object, but you can solve that by calibrating the values properly. So, I think I was able to solve this problem well, even if it was tricky!

6.7 Follow-up research

What I found unfortunate was that my robot only has 1 ultrasonic sensor. If the robot had multiple sensors, the process would have been easier and the code could have been less long. So maybe in the future I would be quite happy to write a new code using a robot with more sensors!

6.8 References

6.8.1 Websites

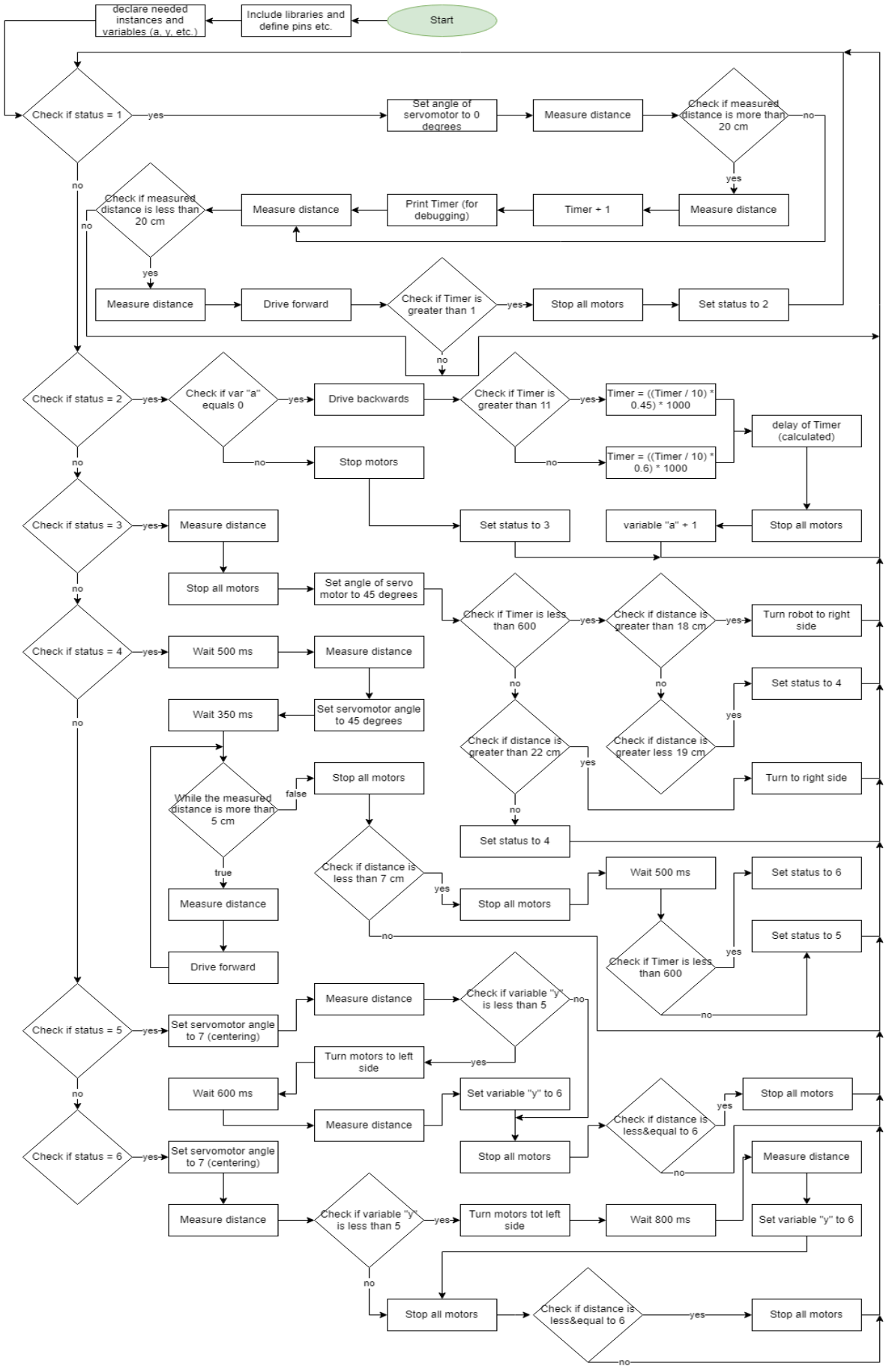
I have not used any sources online, I only used my brain and draw.io for making UML diagram

6.8.2 Figures

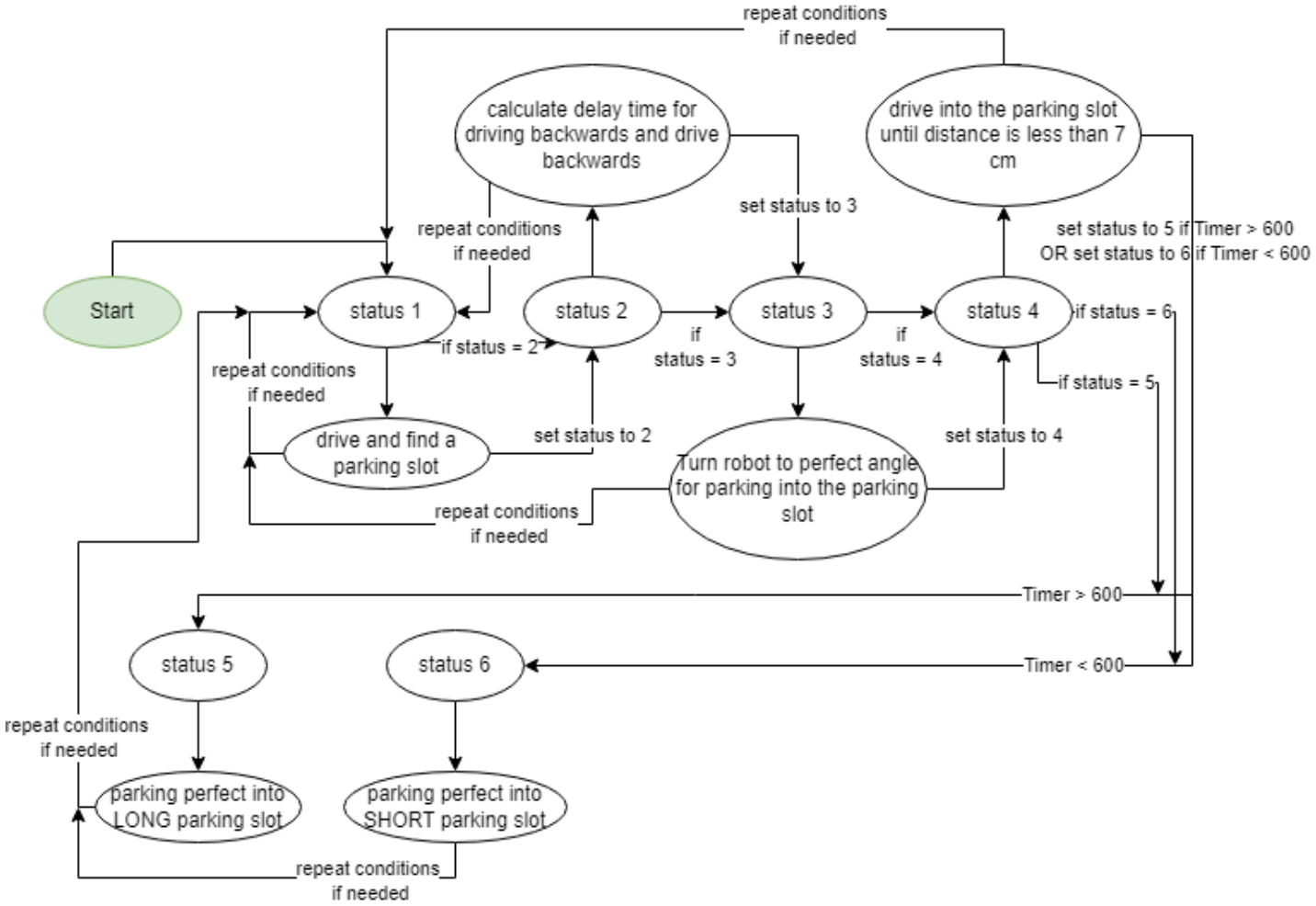
Draw.io - app.diagrams. JGraph Ltd. 2005-2022. Visited on 26-10-2022,
https://app.diagrams.net/

6.9 Code

6.9.1 UML flow diagram



6.9.2 UML state diagram



6.9.3 CODE FOR PROBLEM 6

```
/* INCLUDE NEEDED LIBRARIES */
#include <Servo.h>
#include <Freenove_WS2812B_RGBLED_Controller.h>
/* SERVOMOTOR PIN: */
#define PIN_SERVO      2
/* SONAR PINS: */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* SONAR RELATED DEFINED VALUES: */
#define MAX_DISTANCE    300  // MAXIMUM DISTANCE MEASURABLE
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // CALCULATE TIMEOUT (increase max. distance)
#define SOUND_VELOCITY  340  // SPEED OF SOUND: 340m/s
/* MOTOR PINS: */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* LEDS PINS: */
#define I2C_ADDRESS  0x20
#define LEDS_COUNT   10
/* LINE TRACKING PINS: */
#define PIN_TRACKING_LEFT   A1
#define PIN_TRACKING_CENTER A2
#define PIN_TRACKING_RIGHT  A3

/* GLOBAL IMPORTANT VARIABLES */
long duration;     // Variable for the duration of sound wave travel
float Timer = 0;   // Variable for measuring time
int distance;      // Variable for the distance measurement
int angle = 0;     // Variable for angle of servo
int status = 1;    // Status is the status of the robot (parking status, measuring status,
etc.)
int i = 0, a = 0, y = 0, controle = 0; // i, y and a are needed for a checking state and
controle is also needed for checking status

/* INSTANCES AND LED CONTROLLER */
Servo servo;
byte servoOffset = 0;
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB);

/* SETUP FUNCTION */
void setup() {
  !strip.begin();
  Serial.begin(9600);
  pinMode(PIN_TRACKING_LEFT, INPUT);
  pinMode(PIN_TRACKING_RIGHT, INPUT);
  pinMode(PIN_TRACKING_CENTER, INPUT);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
```

```
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
  servo.attach(PIN_SERVO);
  servo.write(90 + servoOffset);
}

/* MOTOR DRIVING FUNCTION */
void motorRun(int speedl, int speedr) {
  int dirL = 0, dirR = 0;
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* MEASURE DISTANCE FUNCTION */
void meten() {
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Set sonar pin to low
  delayMicroseconds(2);                 // 2 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, HIGH);   // Set sonar pin to high
  delayMicroseconds(10);                // 10 microseconds delay
  digitalWrite(PIN_SONIC_TRIG, LOW);    // Reads the echoPin, returns the sound wave travel
time in microseconds
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Recieve duration of sonar ECHO
  distance = duration * 0.034 / 2;      // distance becomes: duration of echo multiplied by
sound velocity and that all devided by 2 (retour of wave)
  Serial.print("Measured distance = "); // Print measured distance = ...
  Serial.print(distance);               // Print actual distance value
  Serial.println(" cm\n");              // End with measured unit and a newLine
}

/* MAIN LOOP FUNCTION */
void loop() {
  switch (status) { // Switch status of robot
```

```
    case 1: // Case 1: driving and scan for parking slot
      servo.write(angle); // Set servo to given angle value
      delay(50);  // Delay 50 ms
      meten();    // Measure distance (call meten)
      if (distance > 20) {  // Check if measured distance is more than 20 cm
        meten();  // Measure distance (call meten)
        Timer++;  // Do Timer + 1, until distance becomes more less than 20 cm
        Serial.println(Timer);  // Print value of Timer
      }
      meten(); // Measure distance (call meten)
      if (distance < 20) {  // Check if measured distance is less than 20 cm
        meten();  // Measure distance (call meten)
        motorRun(100, 100); // Set motors to 100 speed (forward - max 255 - not recomended)
        if (Timer > 1) {  // Check if Timer value is more than 1 (a brief check, before
going to status 2)
          motorRun(0, 0); // Stop all motors
          status = 2;     // Set status to 2
        }
      }
      break;
    case 2: // Case 2: drive back to perfect place, using math
      if (a == 0) { // Check if a equals 0, which it is only 1x
          motorRun(-100, -100); // Drive back
          if (Timer > 11) { // Check if Timer is more than 11 (long parking slot)
            Timer = ((Timer / 10) * 0.45) * 1000; // Use this calculation (perfect for big
parking slot)
            Serial.println(Timer);  // Print (for debugging code)
            Serial.println("\nTimer > 11\n");
          } else if (Timer < 10) { // Else check if Timer is less than 11 (short parking
slot)
            Timer = ((Timer / 10) * 0.6) * 1000; // Use this calculation (perfect for short
parking slot)
            Serial.println(Timer);  // Print (for debugging code)
            Serial.println("\nTimer < 26\n");
          }
          delay(Timer); // Use calculated Timer value for delay for driving back
          motorRun(0, 0); // Done? now STOP all motors
          a++;  // a + 1
      } else if (a == 1) {  // Check if a equals 1, which it becomes after driving back
        motorRun(0, 0); // Stop all motors
        status = 3; // Set robot status to 3
      }
      break;
    case 3: // Case 3: do pefect angle turn, dependent on Timer
      meten();  // Measure distance (call meten)
      motorRun(0, 0); // Stop motors
      angle = 45; // Set angle to 45 degrees
      servo.write(angle); // Set servomotor to that angle
      if (Timer < 600) {  // Check if Timer is less than 600 (short parking slot)
        if (distance < 69 && distance > 18) { // Check if distance is greater than 18 cm
```

```
          motorRun(200, -200);  // Turn robot to right side
        } else if (distance < 19) { // Else check if distance is less than 18 (Checking)
          status = 4; // Set status to 4
        }
      } else {  // ELSE if not short parking slot
        if (distance < 69 && distance > 22) { // Check if distance is greater than 22 cm
(more room)
          motorRun(200, -200);  // Turn around
        } else if (distance < 23) {   // Check if distnace is less than 23
          status = 4; // Now you can go to status 4
        }
      }
      break;
    case 4: // Case 4: driving into parking slot
      delay(500); // Wait 500 ms
      meten();  // Measure distance (call meten)
      angle = 45; // Set angle to 45 degrees
      servo.write(angle); // Print angle for debugging
      delay(350); // Delay of 350 ms
      while (distance > 5) {  // Check if distance is more than 5
        meten();  // Measure distance (call meten)
        motorRun(80, 80); // Drive forward until distance is not more than 5
      }
      motorRun(0, 0); // Stop all motors
      if (distance < 7) { // Check if distance is less than 7
        motorRun(0, 0); // Stop all motors
        delay(500); // Delay of 500 ms
        if (Timer < 600) {  // Check if timer was less than 600 (short parking slot)
          status = 6; // Set status to short parking slot (6)
        } else {
          status = 5; // Set status to long parking slot (5)
        }
      }
      break;
    case 5: // Case 5: parking perfect into LONG parking slot
      angle = 70;  // Set angle to center (70 degrees)
      servo.write(angle); // Set servomotor to that angle
      meten();  // Measure distance (call meten)
      if (y < 5) {  // Check if y is less than 5 (which it is initially)
        motorRun(-170, 185);  // Turn motors to left
        delay(600); // Delay 600 ms
        meten();  // Measure distance (call meten)
        y = 6;  // Set y to 6
      }
      motorRun(0, 0); // Stop all motors
      if (distance <= 6) { // Last check: if distance is less&equal to 6
        motorRun(0, 0); // Stop all motors
      }
      break;
    case 6: // Case 6: parking perfect into SHORT parking slot
```

```
      angle = 70; // Set angle to center (70 degrees)
      servo.write(angle); // Set servomotor to that angle
      meten();  // Measure distance (call meten)
      if (y < 5) {  // Check if y is less than 5 (which it is initially)
        motorRun(-170, 185);  // Turn motors to left
        delay(800); // Delay 600 ms
        meten();  // Measure distance (call meten)
        y = 6;  // Set y to 6
      }
      motorRun(0, 0); // Stop all motors
      if (distance <= 6) { // Last check: if distance is less&equal to 6
        motorRun(0, 0); // Stop all motors
      }
      break;
  }
}
```

6.10 STARR: Problem 6

Situation or Task

Situation is at school and home.
Task: to make an UML diagram and writing code for parking problem using that UML. When the code is done, it has to parallel park between two objects.

Action

1. Open my browser and go to draw.io
2. Make UML diagram and state diagram
3. Save those
4. Paste them into Word document
5. Use them while writing Arduino code
6. Test code often (debugging code)
7. Add extra or change stuff if UML is not right
8. Change code
9. Change UML
10. Save both if done
11. Copy and paste code (and UML) into Word document
12. Answer questions in Word document
13. Save document
14. Upload to Moodle

Result

I wrote this code at school and at home, it took probably 8 days, which is estimated 43 hours I think. The result is decent, because the code isn't that big, it even works efficiently! Although it did take a lot of time, it was worth it.

Reflection

My experience with coding is still not very good, but this time I improved. I learned to use functions. What I could have improved, is using words instead of values. Such as motorRun(stop, stop) instead of motorRun(0, 0). I will get there somehow in the future!

Problem 7 – Message Reading

7.1 Summary

In this problem, I figured out how to get a robot to read a message by using objects of different widths. For example, I used widths of 10 cm and 5 cm in my code. The robot must be able to read a binary code. 10 cm represents a 0 and 5 cm represents a 1.

7.2 Motivation

Make the robot read an 4 bit binary number. Use objects of 2 different sizes. For example one group of objects with a width of 5 cm and one group of objects with a width of 10 cm. Use 5 cm to represent a 1 and 10 cm to represent a 0. Then the number 5 can be represented by placing the objects in the order of 10, 5, 10, and 5, or 0101 in binary. The robot should move in a straight line along the objects, by using the line following sensor, and read the binary number using the ultrasonic sensor. The result of the number should be visible by using the colored LEDs.

7.3 Problem

No code is available that allows the robot to read a message by measuring two different objects in terms of length.

7.4 Goal

My goal is to make sure the robot can read numbers by measuring two different objects with different lengths. 5 cm should represent an 1 and 10 cm should represent a 0. So my goal is to make sure that the robot can read four bits of binary and convert that into numbers

7.5 Results

The results were quite good. When I saw this problem at the beginning I got a little scared, but the modeling and coding actually turned out not to be very difficult. The code works completely as it should. The robot keeps time when the distance becomes less than 20 cm. If it is a large distance, the time will also be large, so probably 10 cm long. The same goes for 5 cm. I store the values in an array and use it later to convert the 4 bits of code into decimal numbers, showing a fixed color.

Link to my video (proof):

https://www.youtube.com/shorts/n7xWnz6GVCM

rework version:

https://youtu.be/-881eiOVGHk

7.6 Conclusion

Looking back at the results I find that I solved the problem well. I did have some problems at first with the robot scanning too fast causing many errors, but I finally solved that with a delay. Then everything worked and I filmed the process and put it on YouTube. So you can see in the video that the number 5 is shown. Initially it shows a 10 cm object and then 5 cm and then 10 cm again and finally 5 cm. That's in binary 0101, which equals 0 + 4 + 0 + 1 = 5. In my program, that is the white color, which is what you see in the video.

7.7 Follow-up research

I do think there can be improvement. For example, being able to measure exactly whether it's 5 or 10 cm. Then you can also add other lengths, maybe for hexadecimal numbers. Or maybe you could even show words or play sounds with certain lengths. There are many more options available!

7.8 References

7.8.1 Websites

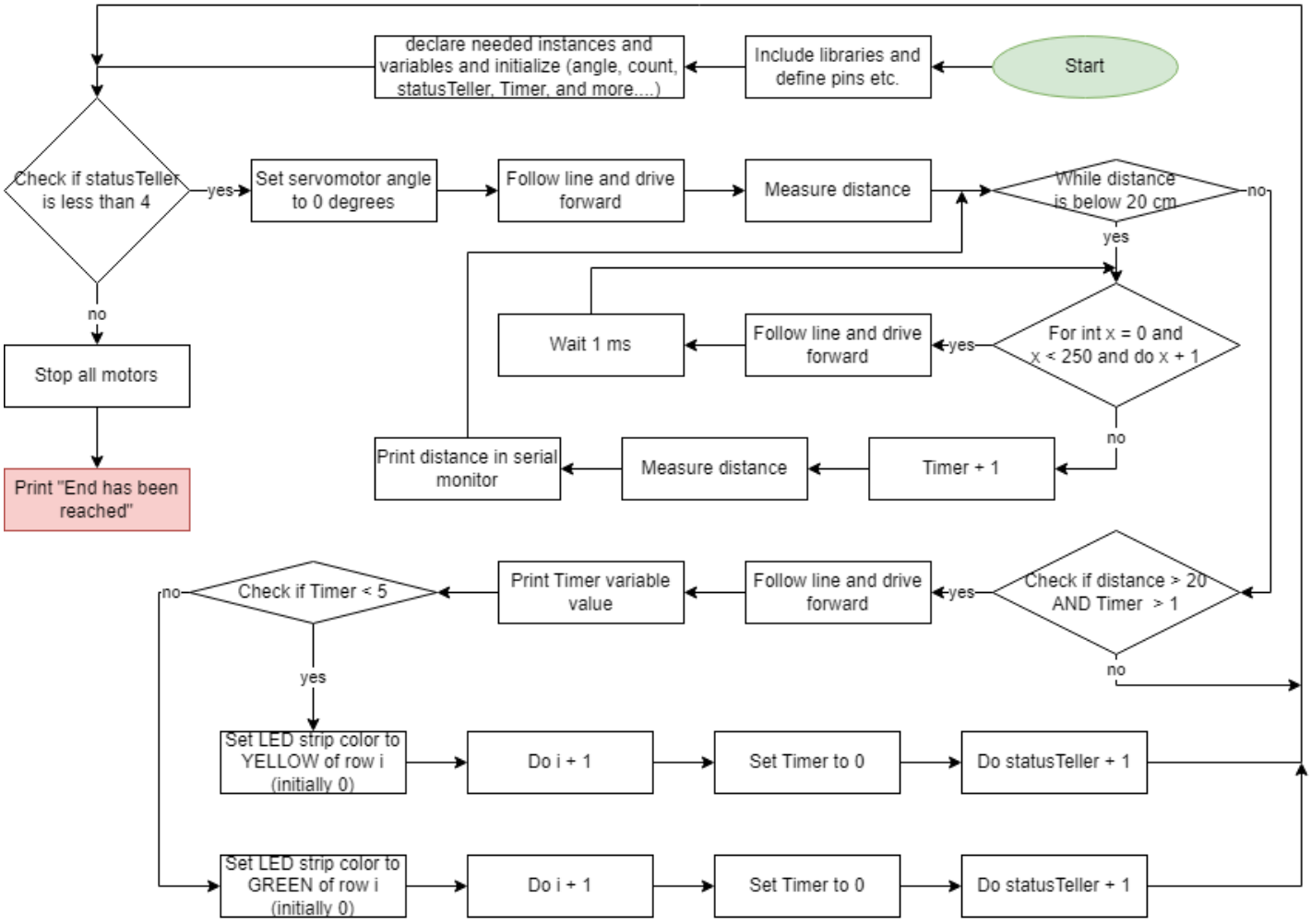
I have not used any sources online, I only used my brain and draw.io for making UML diagram

7.8.2 Figures

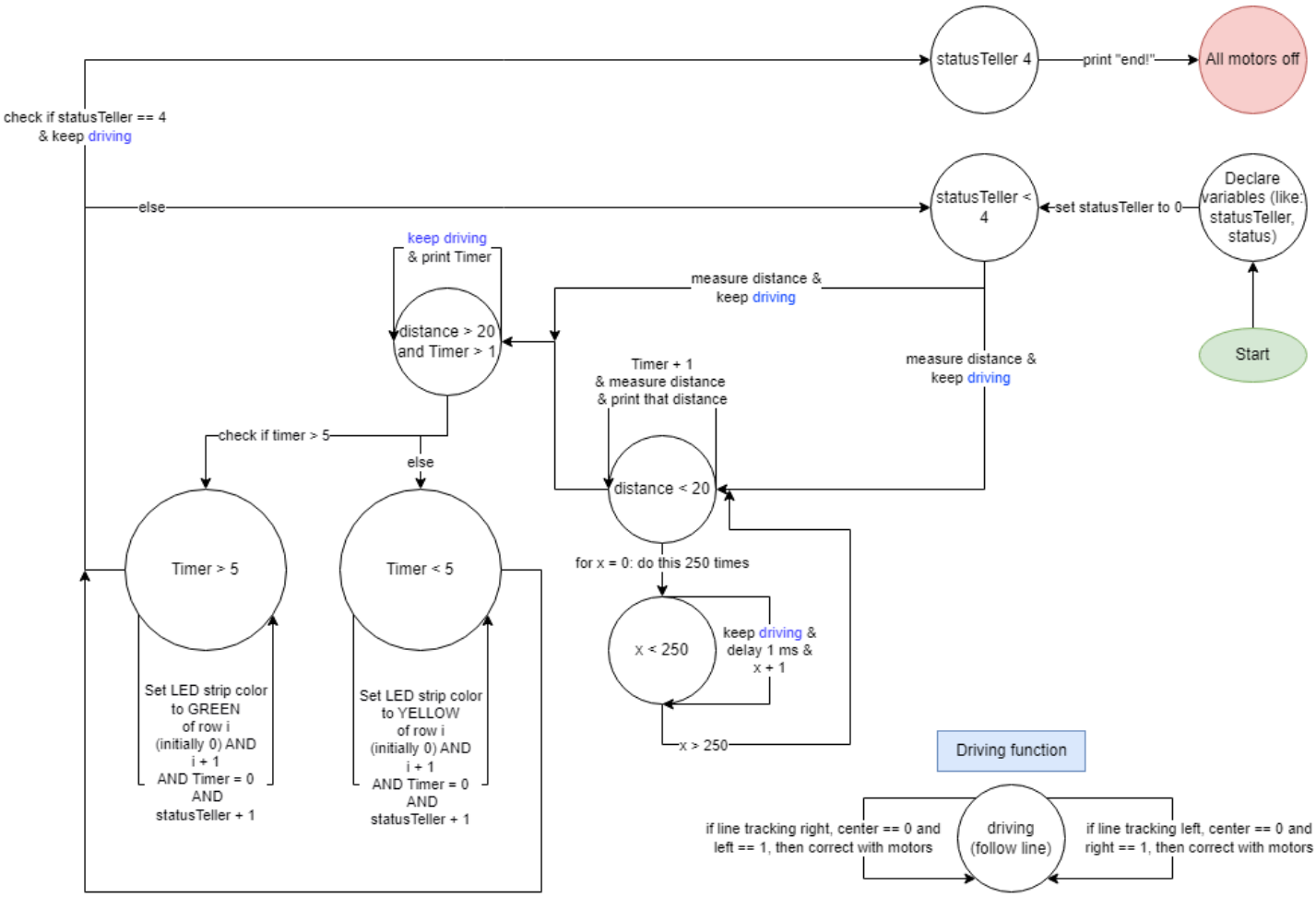
Draw.io - app.diagrams. JGraph Ltd. 2005-2022. Visited on 26-10-2022,
https://app.diagrams.net/

7.9 Code

7.9.1 UML flow diagram



7.9.2 UML state diagram



7.9.3 CODE FOR PROBLEM 7

```
// CODE PROBLEM 7 REWORK (RETAKE)
/* NEEDED LIBRARIES: */
#include <Servo.h>
#include <Freenove_WS2812B_RGBLED_Controller.h>
/* SERVOMOTOR PIN: */
#define PIN_SERVO      2
/* SONAR PINS: */
#define PIN_SONIC_TRIG    7
#define PIN_SONIC_ECHO    8
/* SONAR RELATED DEFINED VALUES: */
#define MAX_DISTANCE    300  // MAXIMUM DISTANCE MEASURABLE
#define SONIC_TIMEOUT   (MAX_DISTANCE*60) // CALCULATE TIMEOUT (increase max. distance)
#define SOUND_VELOCITY  340  // SPEED OF SOUND: 340m/s
/* MOTOR PINS: */
#define PIN_DIRECTION_RIGHT 3
#define PIN_DIRECTION_LEFT  4
#define PIN_MOTOR_PWM_RIGHT 5
#define PIN_MOTOR_PWM_LEFT  6
/* LEDS PINS: */
#define I2C_ADDRESS  0x20
#define LEDS_COUNT   10
/* LINE TRACKING PINS: */
#define PIN_TRACKING_LEFT   A1
#define PIN_TRACKING_CENTER A2
#define PIN_TRACKING_RIGHT  A3

/* INITIALIZE IMPORTANT VARIABLES */
long duration; // Variable for the duration of sound wave travel
int distance;  // Variable for the distance measurement
// 1: angle is the angle of servomotor.
// 2: count is a variable for counting +1 for the datasaver.
// 3: statusTeller is for changing the status of the robot (after 4 measurements)
// 4: Timer is for counting x number of code ran in ms, when distance is less than 20
// 5: i is Needed for doing +1 for individual LED on LED strip;
int angle = 0, count = 0, statusTeller = 0, Timer = 0, i = 0;
int status = 1;         // Status is the state of the robot in the loop (measuring and
showing colors)
int dataSave[100] = {}; // Array for saving binary number of measured time
Servo servo;            // Create servo object
byte servoOffset = 0;   // Change the value to Calibrate servo (IF NEEDED)
Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB); // LED strip
controller

/* SET UP FUNCTION */
void setup() {
  !strip.begin();
  Serial.begin(9600);
  pinMode(PIN_TRACKING_LEFT, INPUT);
  pinMode(PIN_TRACKING_RIGHT, INPUT);
```

```
  pinMode(PIN_TRACKING_CENTER, INPUT);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_SONIC_TRIG, OUTPUT);
  pinMode(PIN_SONIC_ECHO, INPUT);
  servo.attach(PIN_SERVO);
  servo.write(90 + servoOffset);
}

/* (STANDARD SKETCH) ELECTROMOTOR DRIVER FUNCTION */
void motorRun(int speedl, int speedr) {
  int dirL = 0, dirR = 0;
  if (speedl > 0) {
    dirL = 0;
  } else {
    dirL = 1;
    speedl = -speedl;
  }
  if (speedr > 0) {
    dirR = 1;
  } else {
    dirR = 0;
    speedr = -speedr;
  }
  digitalWrite(PIN_DIRECTION_LEFT, dirL); // set PIN_DIRECTION_LEFT to direction DL
  digitalWrite(PIN_DIRECTION_RIGHT, dirR); // set PIN_DIRECTION_RIGHT to direction DR
  analogWrite(PIN_MOTOR_PWM_LEFT, speedl); // set PIN_MOTOR_PWM_LEFT to direction only
forward L
  analogWrite(PIN_MOTOR_PWM_RIGHT, speedr); // set PIN_MOTOR_PWM_RIGHT to direction Only
forward R
}

/* LINE TRACKING DRIVING FUNCTION */
void driving() {
  if (digitalRead(PIN_TRACKING_LEFT) == 0 && digitalRead(PIN_TRACKING_CENTER) == 0 &&
digitalRead(PIN_TRACKING_RIGHT) == 0) { //Check if all line trackings sensors are off
      motorRun(0, 0); // Set all motors off
    } else if (digitalRead(PIN_TRACKING_CENTER) == 1){  // Check if line tracking center is
on
      motorRun(100, 100); // Set motors on (forward)
    } else if (digitalRead(PIN_TRACKING_LEFT) == 1){  // Check if line tracking left is on
      motorRun(-255, 255);  // Turn to right side
    } else if (digitalRead(PIN_TRACKING_RIGHT) == 1){   // Check if line tracking right is
on
      motorRun(255, -255);  // Turn to left side
    } else if (digitalRead(PIN_TRACKING_RIGHT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1){  // Check if line tracking right and center are on
      motorRun(255, -255);  // Turn to left side
```

```
    } else if (digitalRead(PIN_TRACKING_LEFT) == 1 && digitalRead(PIN_TRACKING_CENTER) ==
1){   // Check if line tracking left and center are on
      motorRun(-255, 255);  // Turn to right side
    } else {          // Else:
      motorRun(0, 0); // Set all motors off
  }
}

/* MEASURE DISTANCE FUNCTION */
void meten() {
  digitalWrite(PIN_SONIC_TRIG, LOW); // Initializes the sonar sensor: set trigger pin to 0
  delayMicroseconds(2); // A delay of 2 micro seconds
  digitalWrite(PIN_SONIC_TRIG, HIGH); // Set tPin to 1: this emits the high frequency sound.
  delayMicroseconds(10); // Delays it for 10 micro seconds
  digitalWrite(PIN_SONIC_TRIG, LOW); // Set tPin to 0: stops emitting high frequency sound.
  duration = pulseIn(PIN_SONIC_ECHO, HIGH); // Calculates duration between 0 to 1 to 0
  distance = duration * 0.034 / 2; // Time * Speed of Sound / 2. (retour: see theory prob2)
  // Serial.print("Afstand = "); // Prints "Afstand = ..."
  // Serial.print(distance); // Prints the distance variable value. (could be anything)
  // Serial.println(" centimeter\n"); // Prints " centimeter" with a new line.
}

/* (MAIN) LOOP FUNCTION */
void loop() {
  if (statusTeller < 4) { // Check if statusTeller (a counter) is less than four
    angle = 0;                // SET ANGLE TO 0 DEGREES
    servo.write(angle);       // TURN SERVOMOTOR TO THAT ANGLE
    driving();                // CALL FUNCTION "driving" (follow line)
    meten();                  // CALL FUNCTION "meten" (measure distance)
    while (distance < 20) {   // CHECK IF MEASURED DISTANCE IS LESS THAN 20
      for (int x = 0; x < 250; x++) { // Loop 250 times with delay of 1 ms
        driving();  // FOLLOW LINE
        delay(1);   // DELAY 1 ms (SUM = 250 ms)
      }
      Timer++;    // ADD +1 TO VARIABLE "Timer"
      meten();    // CALL FUNCTION "meten" (measure distance)
      Serial.println(distance); // Print distance value
    }
    if (distance > 20 && Timer > 1) { // CHECK IF DISTANCE IS GREATER THAN 20 AND CHECK IF
TIMER IS GREATER THAN 1
      driving();  // FOLLLOW THE LINE
      Serial.println("TIMER WAARDE: ");
      Serial.println(Timer);  // PRINT "Timer" value
      Serial.println("\n");
      if (Timer < 5) {                    // Check if "Timer" is less than 5 (5 cm object)
        strip.setLedColor(i, 255, 255, 0); // Set LED strip collor to yellow of row i
(initially 0)
        i++;                              // ADD +1 to i
        Timer = 0;                        // Reset "Timer" (to 0)
        statusTeller++;                // ADD +1 TO "statusTeller"
```

```
      } else {                              // Else (it must be 10 cm)
        strip.setLedColor(i, 0, 255, 0); // Set LED strip collor to green of row i
(initially 0)
        i++;                            // ADD +1 to i
        Timer = 0;                      // Reset "Timer"
        statusTeller++;                 // ADD +1 TO "statusTeller"
      }
    }
  } else {
    motorRun(0, 0);
    Serial.println("END has been reached!");
  }
}
```

7.10 STARR: Problem 7

Situation or Task

Situation is at school and home.
Task: my task is to write a code for the robot to read a message using objects of different widths. Also, while doing this, the robot must also follow a line. At the end, the robot has to show a color depending on the order of the objects.

Action

1. Open my browser and go to draw.io
2. Make UML diagram and state diagram
3. Save these files on my laptop
4. Paste them into the Word document
5. Use them while writing the main Arduino code
6. Test the code often times (debugging code)
7. Add extra or change if UML diagram is not right
8. Change code if needed
9. Change UML if needed
10. Save both files if done
11. Copy and paste code (and UML) into Word document
12. Answer questions in Word document
13. Save document
14. Upload to Moodle

Result

I wrote this code at school and at home, it took probably 7 days, which is estimated 36 hours I think. I think the result is quite good because the robot can park in both a small and a large parking lot.

Reflection

My experience with writing this went good. At the beginning I got a little scared when I heard that we had to do something in binary, but when I got to the assignment itself, it actually didn't seem very difficult. I mostly had to think logically and figure out how to measure time, and I succeeded!

Problem 8 – Robot Simulation

8.1 Summary

In this chapter, I figured out how to make a robot drive in an open plane with all kinds of objects without touching them. Therefore, the goal is to get as far as possible without touching objects. While researching and writing code and improving my UML diagrams, my code got better and better.

8.2 Motivation

At all positions where the robot has been a dot is placed. Write a program that makes the robot drive through as many open spaces as possible (and change the visited spaces in a dot).

8.3 Problem

No code is available to ensure that the robot safely avoids all obstacles. Make sure the robot uses its speed efficiently and does not crash!

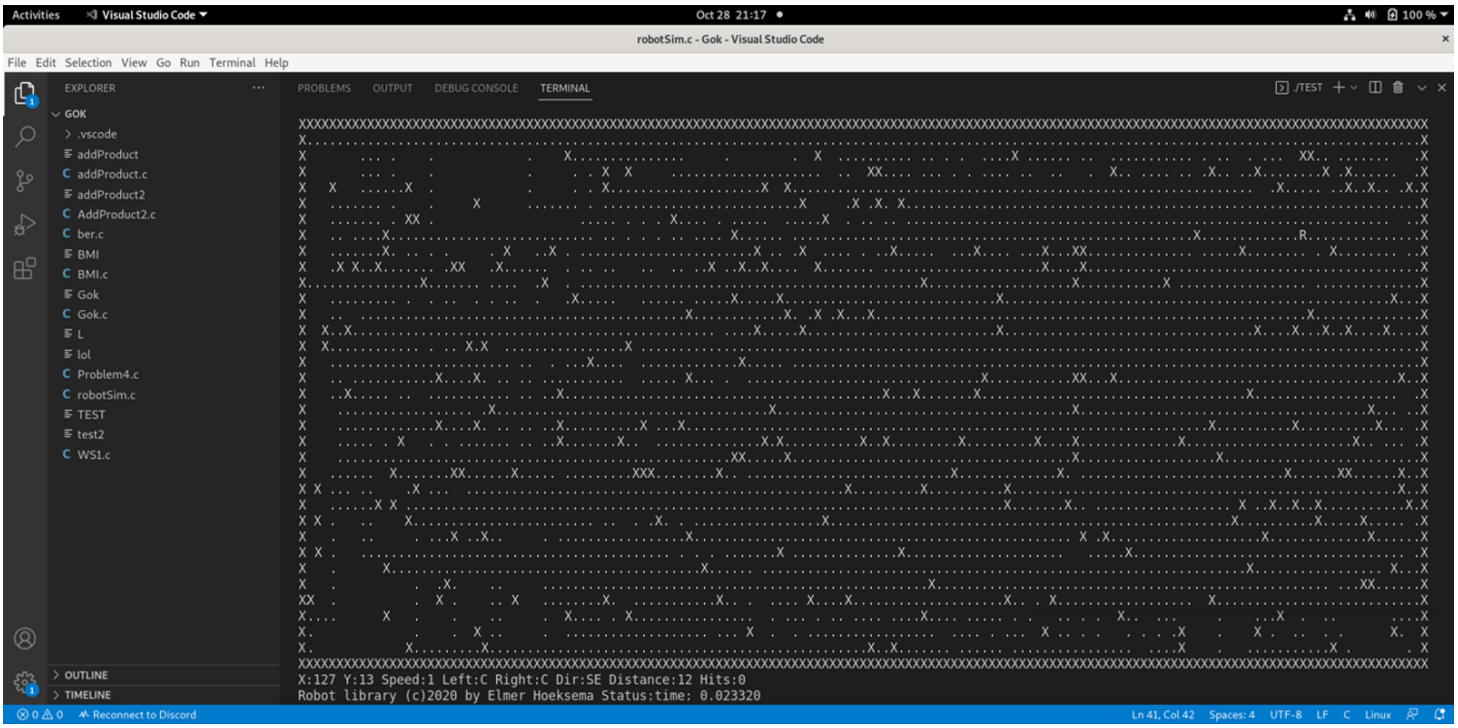
8.4 Goal

My goal is to write the code for this problem so that the robot in the simulation avoids all obstacles and can get to all places without colliding with the objects surrounded the robot.

8.5 Results

In itself, I was happy with the result. I ran the code you see below for 1 hour and the result was pretty okay. I actually wanted to make a code that works like a robot vacuum cleaner, that the robot goes from top to bottom, making it go everywhere. I ended up not being able to do this due to lack of time. I had started working on it, but due to lack of time I was unable to make it further. So then I just stuck to this code that I had initially made. Anyway. If you look at the image you can also see that the robot occasionally tilts for a reason I don't know, it does manage not to hit any objects though, so I was happy with that.

The result after hours of waiting:



8.6 Conclusion

My conclusion is that while the robot will eventually get everywhere, the code has not yet reached its maximum efficiency. In follow-up research, however, I would like to try to make the code even better by making an algorithm like what a Roomba also uses.

8.7 Follow-up research

In a follow-up research, I would like to try to write a code that works like an algorithm. The robot goes from top to bottom avoiding the obstacles and then just moves on. Eventually the robot will then get everywhere and the process will not take too long

8.8 References

8.8.1 Websites

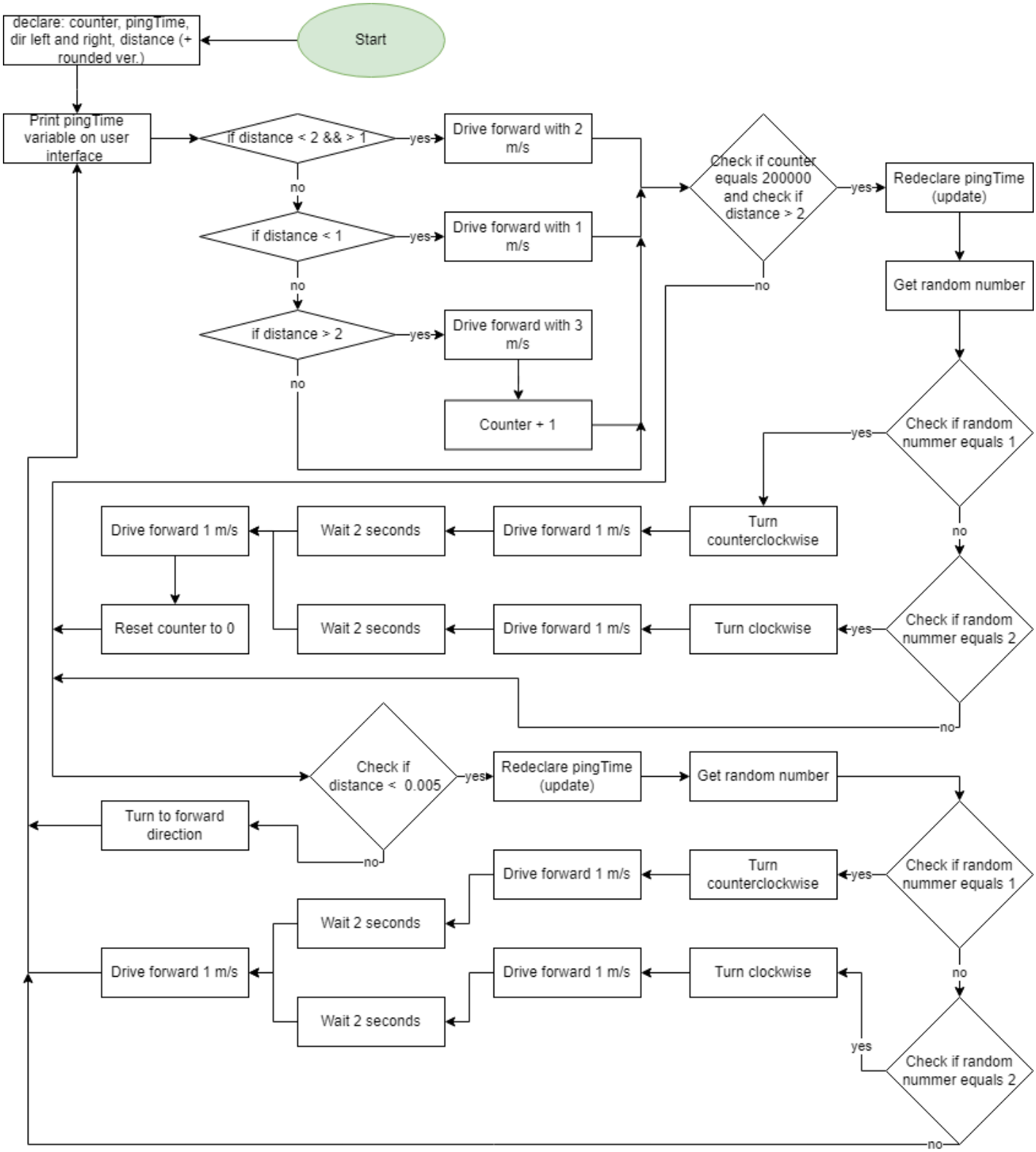
I have used Elmer's tutorial on YouTube

8.8.2 Figures

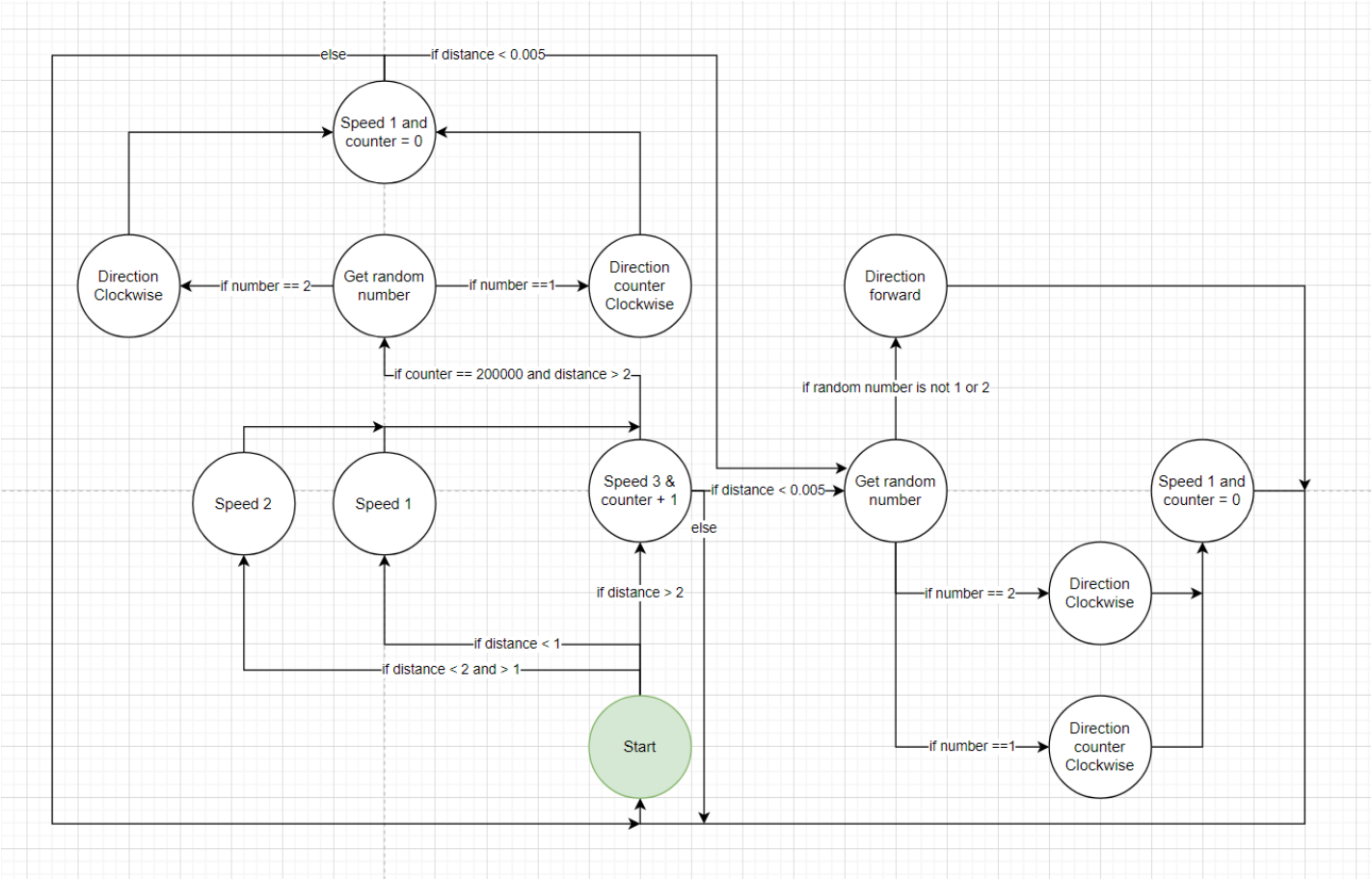
Draw.io - app.diagrams. JGraph Ltd. 2005-2022. Visited on 26-10-2022, https://app.diagrams.net/

8.9 Code

8.9.1 UML flow diagram



8.9.2 UML state diagram



8.9.3 Makefile for problem 8

```
Problem8:   robotSim.c
    gcc -o Problem8 -lrobot -lncurses -lpthread -lm robotSim.c
```

8.9.4 Code for problem 8

```
#include <stdio.h>
#include <robot.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int counter = 0;

void init()
{
    double pingTime = ping();
    directionLeft(CounterClockwise);
    directionRight(Clockwise);
}

void loop()
{
    /* Variables */
    double pingTime = ping();   // Ping = delay for measuring distance
    double distvPT = ((pingTime / 1000 / 2) * 3 * 100000000) / 1000; // Calculating distance
with scientific formula.
    double distPTR = round(((pingTime / 1000 / 2) * 3 * 100000000) / 1000); // Rounding
measured distance with function round();

    /* Print variables on user interface */
    dprintf(robotOut, " Pingtime: %f", pingTime); // print Pingtime

    if (distvPT < 2 && distvPT > 1)   // Check if measured distance is between 1 and 2
    {
        speed(2);   // Set speed to 2 m/s
    }
    else if (distvPT < 1)    // Check if measured distance is less than 1
    {
        speed(1);   // Set speed to 1 m/s
    }
    else if (distvPT > 2)   // Default (else)
    {
        speed(3);   // Set speed to 3 m/s
        counter++;  // Counter + 1
    }

    /* Road randomizer after ~2.5 meters */
    if (counter == 200000 && distPTR > 2) {  // Check if counter has reached 200000 AND
distance > 2 meter
        double pingTime = ping();   // Redeclare pingTime
        srand(time(NULL));          // Use random number function
        int getal = rand() % 2 + 1; // Declare getal as random number variable

        if (getal == 1) // Check if the randomized number is equal to 1
```

```
        {
            directionRight(CounterClockwise);   // Set right wheels to counter clock wise
direction (-speed)
            directionLeft(CounterClockwise);    // Set left wheels to counter clock wise
direction (-speed)
            speed(1);   // Set speed to 1 m/s
            wait(2);    // Wait 2 seconds (90 degree correction)
        }
        else if (getal == 2)    // Else check if randomized number equals 2
        {
            directionRight(Clockwise);  // Set right wheels to clock wise direction (-speed)
            directionLeft(Clockwise);   // Set left wheels to clock wise direction (-speed)
            speed(1);   // Set speed to 1 m/s
            wait(2);    // Wait 2 seconds (90 degree correction)
        }
        speed(1);       // Set speed to 1 m/s
        counter = 0;    // Reset counter variable (set to 0)
    }

    /* X checker and avoider with randomized side choice */
    if (distPTR < 0.005)    // Check if rounded measured distance is less than 0.005 (~0)
    {
        double pingTime = ping();   // Redeclare pingTime
        srand(time(NULL));          // Use random number function
        int getal = rand() % 2 + 1; // Declare getal as random number variable

        if (getal == 1) // Check if the randomized number is equal to 1
        {
            directionRight(CounterClockwise);   // Set right wheels to counter clock wise
direction (-speed)
            directionLeft(CounterClockwise);    // Set left wheels to counter clock wise
direction (-speed)
            speed(1);    // Set speed to 1 m/s
            wait(2);     // Wait 2 seconds (90 degree correction)
        }
        else if (getal == 2)    // Else check if randomized number equals 2
        {
            directionRight(Clockwise);  // Set right wheels to clock wise direction (-speed)
            directionLeft(Clockwise);   // Set left wheels to clock wise direction (-speed)
            speed(1);   // Set speed to 1 m/s
            wait(2);    // Wait 2 seconds (90 degree correction)
        }
        speed(1);   // Set speed to 1 m/s
    }
    else    // Else set wheels on forward direction
    {
        directionRight(Clockwise);              // Set right wheels to clock wise direction
        directionLeft(CounterClockwise);        // Set left wheels to clock wise direction
    }
}
```

8.10 STARR: Problem 8

Situation or Task

Situation is at school and home.
Task: downloading the “robot.h” library from Moodle. Unpacking the file and installing the library with terminal in Debian (following the tutorial). After I have done that I had to write a C code so that the robot can get everywhere without touching the X’s (marked places). It also has to be an efficient code, faster when it is possible and slower when needed. But firstly I have to make an UML diagram before actually coding. A state diagram is also needed.

Action

1. Download “robot.h” (by Elmer) library from Moodle
2. Install the library with terminal in Debian
3. Make UML and state diagram
4. Save these diagrams and paste them in Word
5. Write code using UML and state diagram
6. Test code and improve if needed.
7. Save code
8. Paste code in Word
9. Answer all questions in Word document
10. Save Word document
11. Upload file to Moodle

Result

I liked the result for a beginner in itself. The robot avoids all obstacles and drives at maximum speed when it can also goes in a random direction. This so he will get everywhere. I was also glad that the code didn't get too long. Functions I tried to use, but it wasn't really necessary

Reflection

My experience writing this code was good in itself. I think writing code does help me begin to understand better and better how everything works. I think I worked on it for about 7 days, which amounts to 40 hours. The UML diagrams help me with that as well. I did find it difficult to create a state diagram in this one, but I managed it to do. There was also a problem that the library lncurses did not work. We solved that together in class by using apt-get