

# SDU

## Assignment 2

Line Following Robot

by: Team (2/Thursday: 08-12) 7

Michalina Janik & Michał Krukowski,

[mikru21@student.sdu.dk](mailto:mikru21@student.sdu.dk),

[mijan21@student.sdu.dk](mailto:mijan21@student.sdu.dk)

### Our implementation:

Our robot purpose is to get from Start Node to End Node using the shortest possible path. Nodes are connected with black lines and these lines represent paths possible to choose from. Robot is following a given path in order to get to the final point. Robot also has 3 Green and 1 Red LEDs as information for the observer. One Green LED is lighting when the robot is powered, and 2 left represents which motor is currently working. Red one is lighting when a robot finds an obstacle on his path. This is other functionality of our robot

How to find the shortest Path:

We are using A\* algorithm implemented in Python: file: "A\_star/AStarAlg\_Assignment\_2.py". As a result, we are getting a list of ordered nodes with which you have to visit to get from one node to others. To use our program, you need to set starting node and ending node and then run the program:

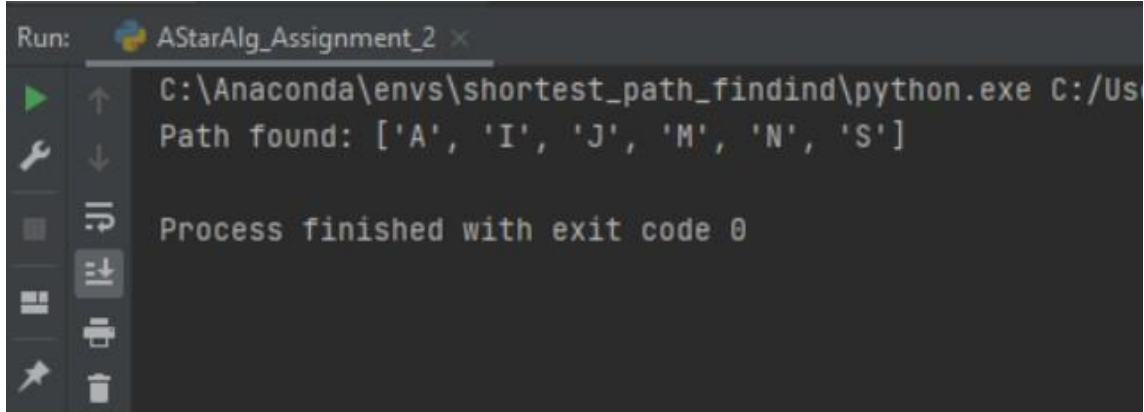
```
54  
55 ▶ if __name__ == '__main__':  
56     # AStar_algorithm('A', 'S')  
57     AStar_algorithm('A', 'S')  
58
```

How we use this information about path:

In order to make the robot drive, we are adding the path in the right order. When the Robot gets to the next intersection on a graph, the algorithm decides if the robot should: turn left/right, go

straight or if he gets to his destination. We provided two demo movies for two paths. From A to S & D to O:

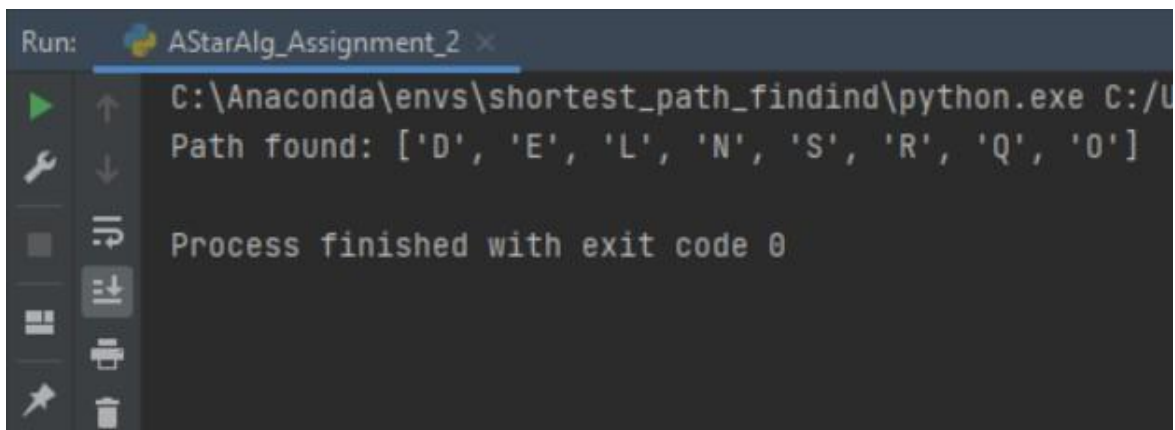
[Line following robot - Driving Demo 1:](#)



```
Run: AStarAlg_Assignment_2 x
C:\Anaconda\envs\shortest_path_findind\python.exe C:/Us
Path found: ['A', 'I', 'J', 'M', 'N', 'S']

Process finished with exit code 0
```

[Line following robot - Driving Demo 2](#)

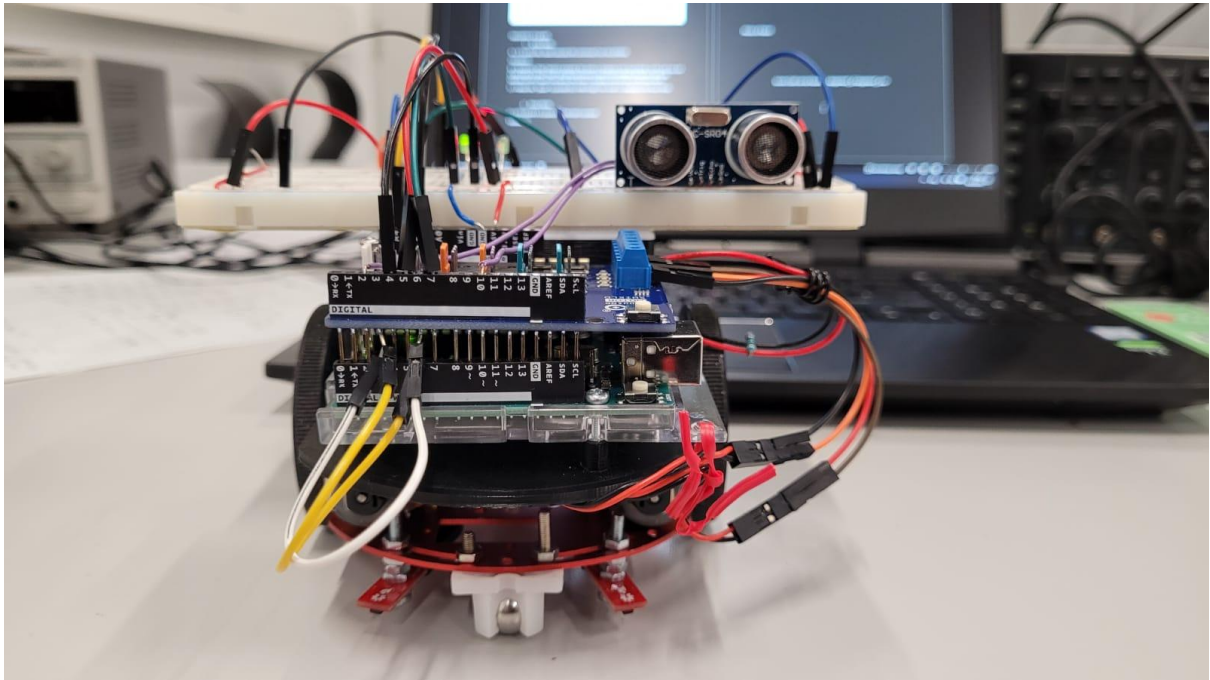


```
Run: AStarAlg_Assignment_2 x
C:\Anaconda\envs\shortest_path_findind\python.exe C:/U
Path found: ['D', 'E', 'L', 'N', 'S', 'R', 'Q', 'O']

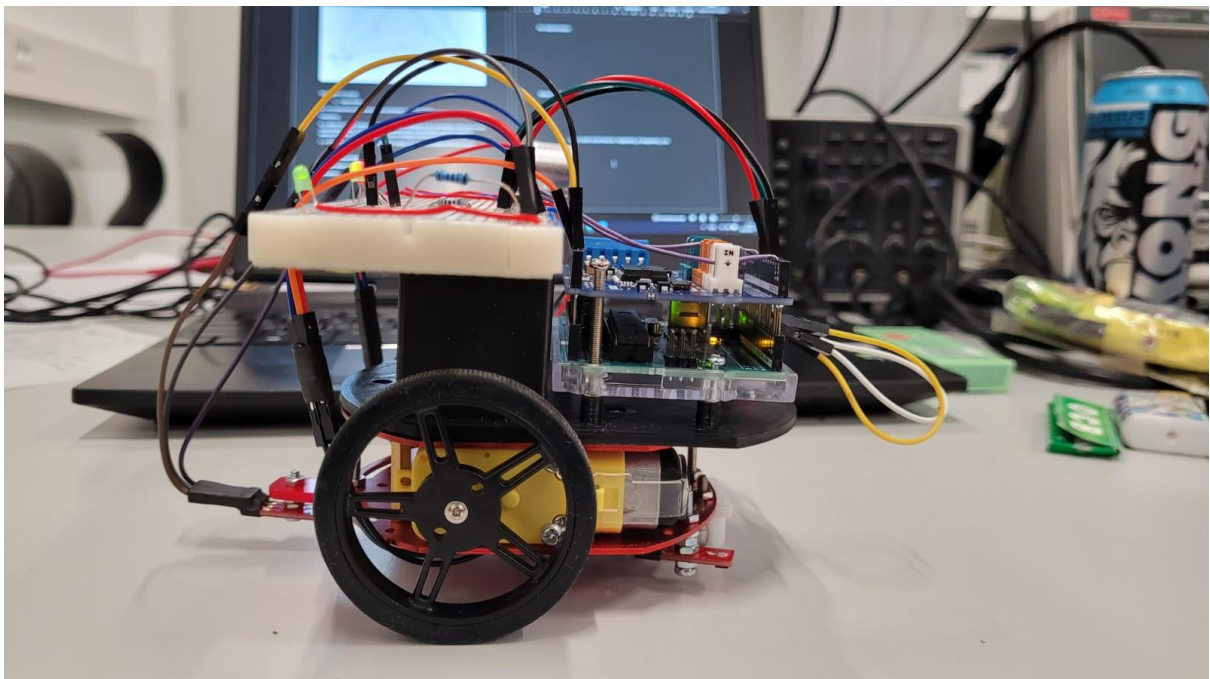
Process finished with exit code 0
```

## Photos of Robot:

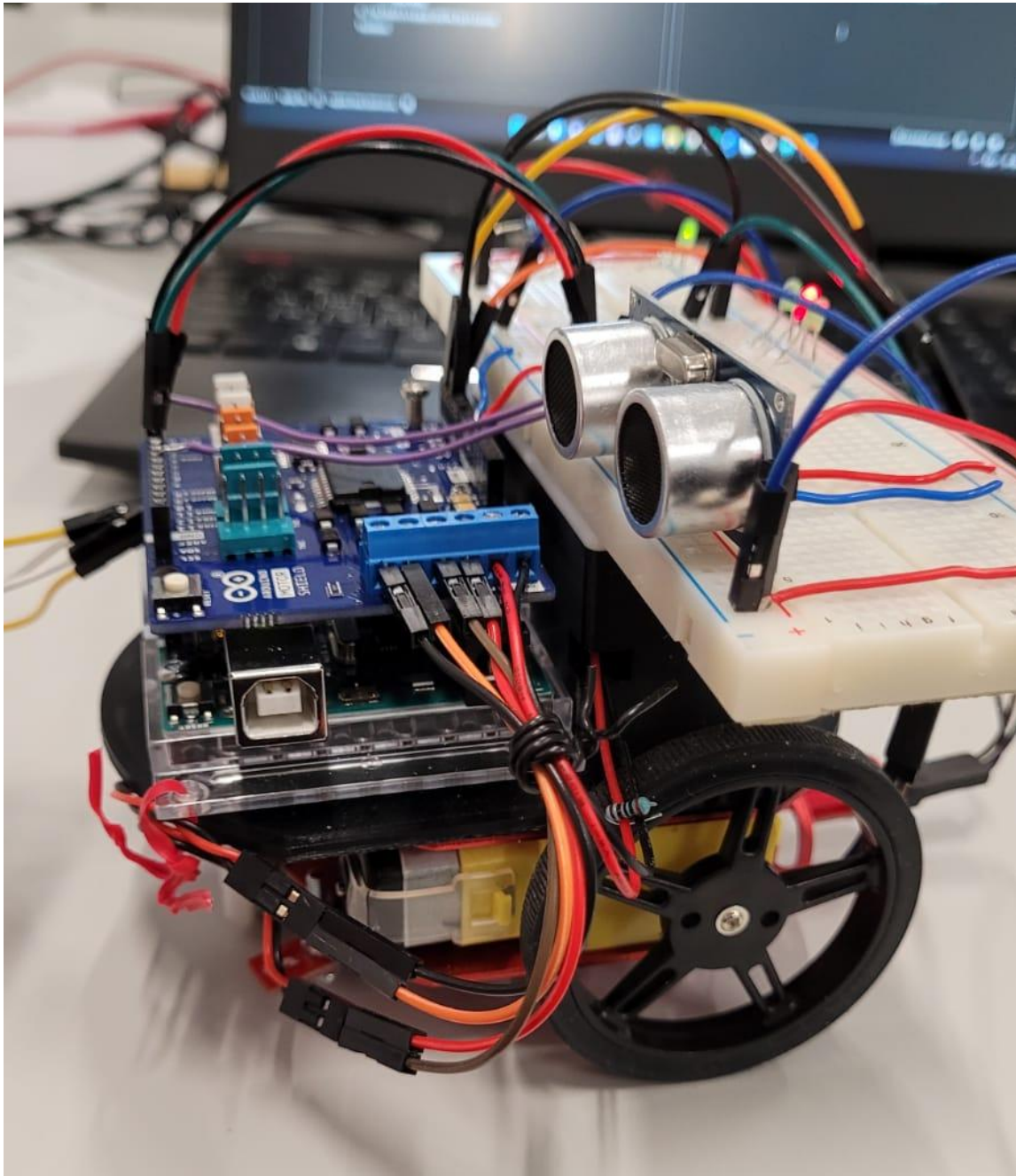
Front



Right side:

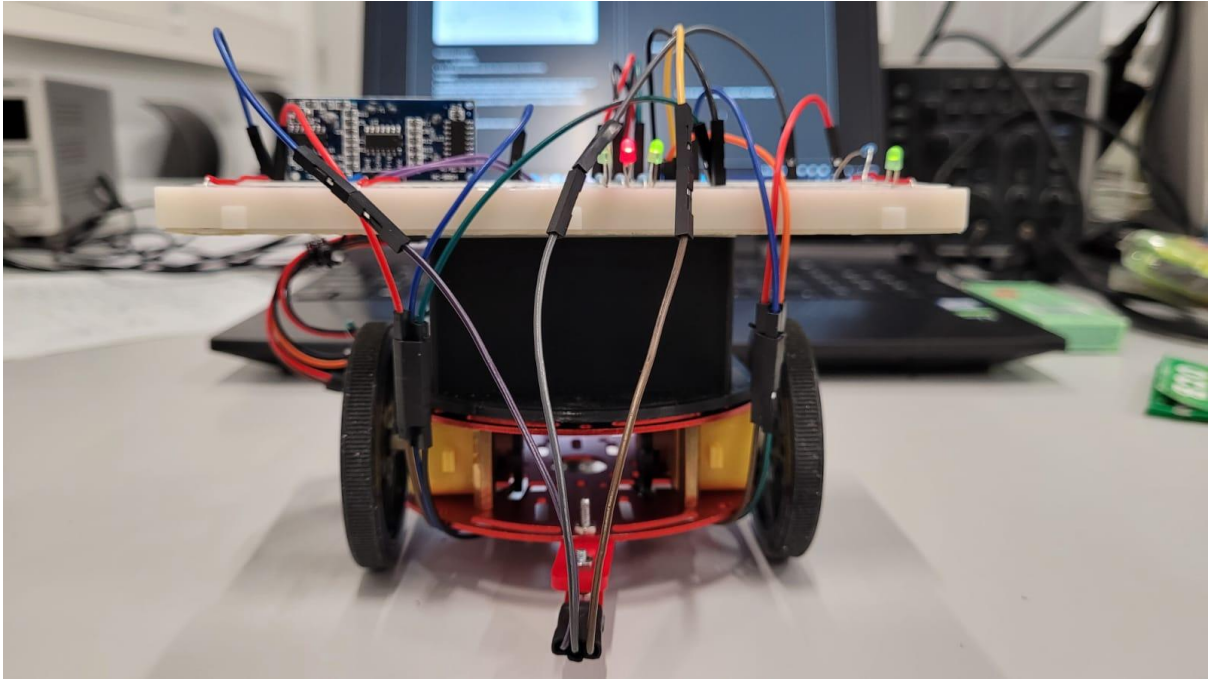


Left side:

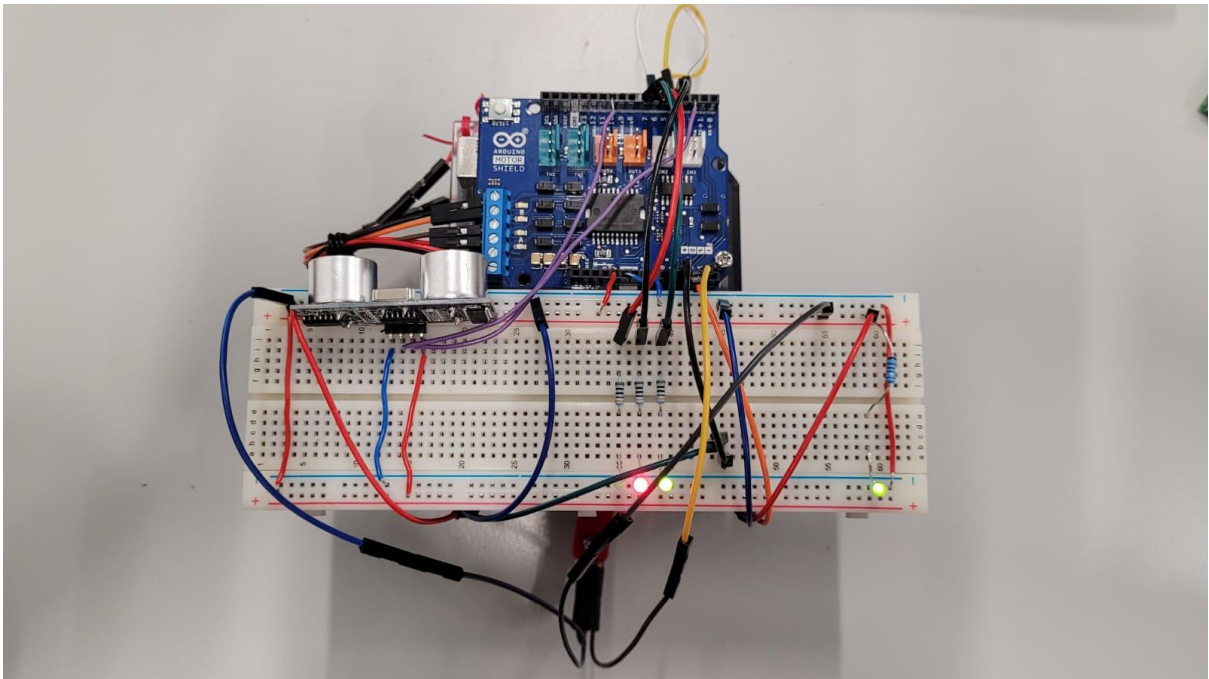




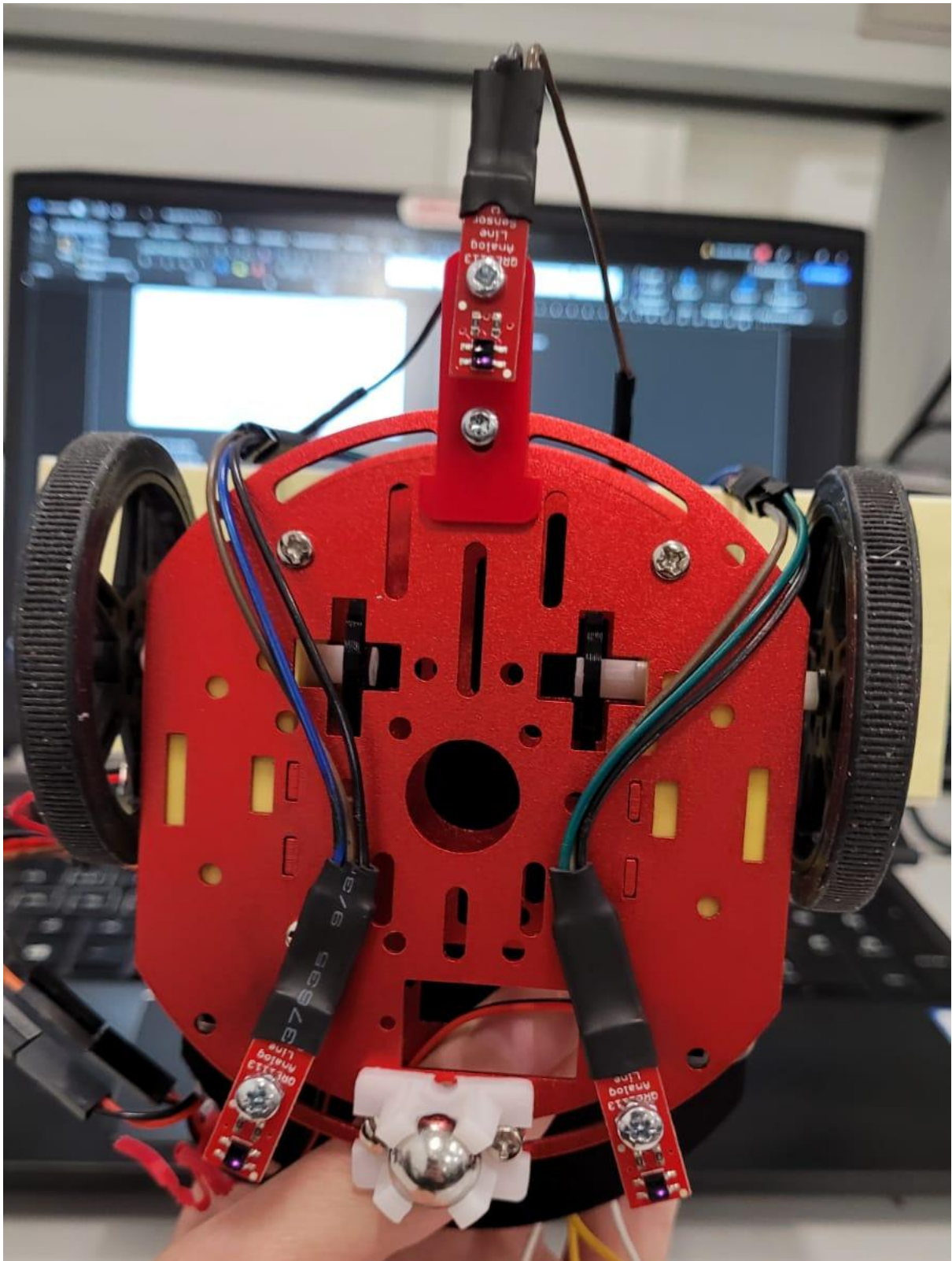
**Back:**



**View from above:**



View from below:



## Used Pins and its functions:

### Motor pins:

Function	Direction	Speed	Brake
Channel A	12	6*	9
Channel B	13	11	8

\*We have switched the pin 3 to pin 6, due to the fact that we needed pin 3 for the interrupted **function** in last **project**.

### Led pins:

Digital Pin	Function
4	Red LED, User Interface
5	Green LED, User Interface
7	Green LED, User Interface
POWER	Green LED, power

### IR sensor pins:

Analog Pin	Function
A2	Front-Left IR sensor
A3	Front-Right IR sensor
A4	Rear IR sensor

### Ultrasonic sensor HC-SR04 pins:

Digital Pin	Function
2	Echo pin
10	Trig pin

## Hardware - Required components:

1. Arduino Motor Shield microcontroller
2. IR sensor x3
3. Ultrasonic sensor
4. LED x4 & resistor x4
5. female and male wires
6. 6V power supply
7. DC motors x2

### 1. Arduino Motor Shield microcontroller:

This is an Arduino microcontroller, which is based on the L298 dual full-bridge. It is powered by an external power supply (6v battery) and has two separated channels A and B, which are used to connect DC Motors.

Each of the motors uses three pins of a microcontroller, so in total 6 digital pins are used to drive the robot.

It supplies power(5V) to a breadboard, where our components are embedded.

Three LEDs which create an interface for the user are connected to the microcontroller by its three digital pins: 4,5,7.

Three IR sensors are also connected to the Arduino via analog pins: A2, A3, A4.

The Ultrasonic sensor is connected via digital pins: 2, 10.

### 2. IR sensor

It is called "Line follower", because it provides a possibility for a robot to detect a dark line.

It measures the amount of reflected light, so it is able to detect transitions from light to dark surface.



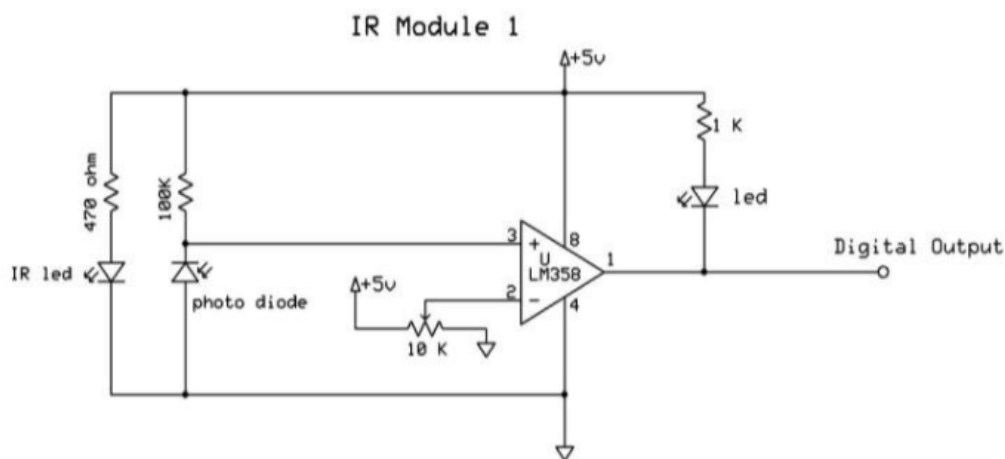
The IR sensor consists of a one IR emitter(IR LED) and a photo-transistor(IR detector).

IR emitter emits infrared light, which is outside of a visible spectrum. A phototransistor is a transistor sensitive to light.

When the area underneath the sensor is dark, the photo-transistor is off and the output is HIGH, otherwise the current is generated, and the signal is received by the detector, so the output is a LOW signal. IR sensor gives an analog output in a range of (0-1023) its value describes the amount of reflected IR LED light.

	DARK SURFACE	WHITE SURFACE
STATE	HIGH	LOW

### IR Sensor -schematic



### 3. Ultrasonic sensor HC-SR04

It is a sensor which can measure distance in a range of (2cm-4m) by using a sonar.

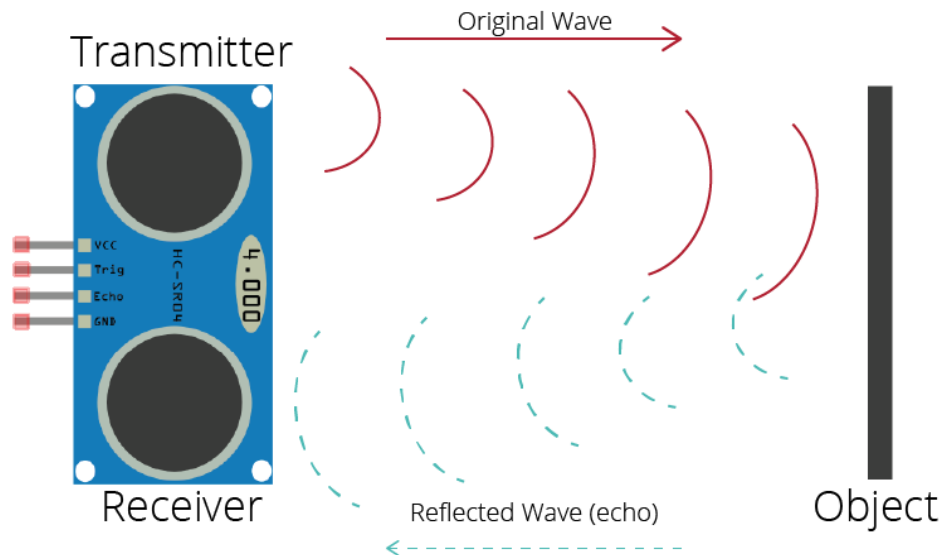
- The ultrasound transmitter(trig pin) emits a sound of a high frequency.
- The sound travels in the air, so if it detects an object it bounces back to the sender.
- In the result,a receiver(echo pin), takes the reflected sound (echo).

Distance from the obstacle can be calculated based on the time travel of the sound from a sender(trig pin) to a receiver(echo pin) and the sound speed value(0,034cm/ms). The calculated time is called "Pulse", and can be measured using the pulseIn() function, which detects on the echo pin the change of the state

Distance formula:

$$S = V \cdot t$$

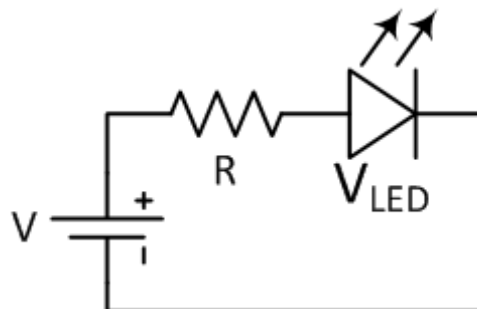
$$S = \frac{0.034cm/ms}{2} \cdot t$$



#### 4. LEDs & Resistors

Three LEDs are components used as an interface for the user to signalize the current state of the robot. They are connected to an Arduino with resistors between them. The fourth LED is connected directly to a power, so it signals whether the robot is supplied by the power or not.

An LED emits light when an electric current passes through it. The simplest circuit to power an LED is a voltage source with a resistor and an LED. The resistor is used to limit the current through the LED and to prevent the LED from burning out.



The resistance needed to power a LED is easy to calculate using Ohm's law. It can be done using the following equation:

$$R = \frac{V - V_{LED}}{I}$$

Where:

V - the voltage source,

V\_LED - LED voltage,

I - LED current.

## 5. **Female and male wires**

Wires are used to connect everything together.

## 6. **6V Power Supply**

As a power supply, we are using x4 1,5V battery in order to create a 6V power supply.

## 7. **DC motors x2**

In the project, we used two DC motors. DC motors convert direct current energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some mechanism to change the direction of current in part of the motor. In our motors, the H-Bridge mechanism is used. Arduino MOTOR SHIELD is responsible for controlling these motors.

## Software:

### The shortest path search algorithm: A\*

- Description of a function: *AStar\_algorithm(start\_node, end\_node)*:

1. First we need two sets: open\_set and closed\_set and two maps. The "g" map is used to store the distance from the starting node. The "parents" map/dictionary is used to store the parent node. The "g" dictionary is initially empty, but the "parents" store itself (the start node) from the beginning.
2. Then the cost to all its adjacent nodes needs to be calculated. So we need to find the neighbouring node with the lowest cost value called  $f(n)$  and create a condition of breaking a finding loop, which is reaching the destination node. If the current node is not the final one, we are putting it to the open\_set in the condition that it has not been already there and its parent node set.
  - " $f(n)$ " represents the final cost, which consists of a " $g(n)$ " const and " $h(n)$ " cost.
  - The " $g(n)$ " is a cost of traversing from one node to another.
  - The  $h(n)$  is an approximation cost, usually treated as a value of a distance between a given node and the final one.
3. If the g value of the neighbour is lower then the current node's value and the neighbour is in the closed\_set, so the current node is replaced by the neighbour's parent node.
4. If the current "g" value is lower than the previous one and its adjacent node is in open\_set, the current one is replaced by the neighbour parent, simultaneously the g value is set to the lower one.
5. If the adjacent node is not present in any of the sets (open\_set, closed\_set), the g value is set and then it is added to open\_set

- Description of a function: *get\_neighbors(v)*:

The function returns the adjacent node of a "v" node, otherwise it returns None.

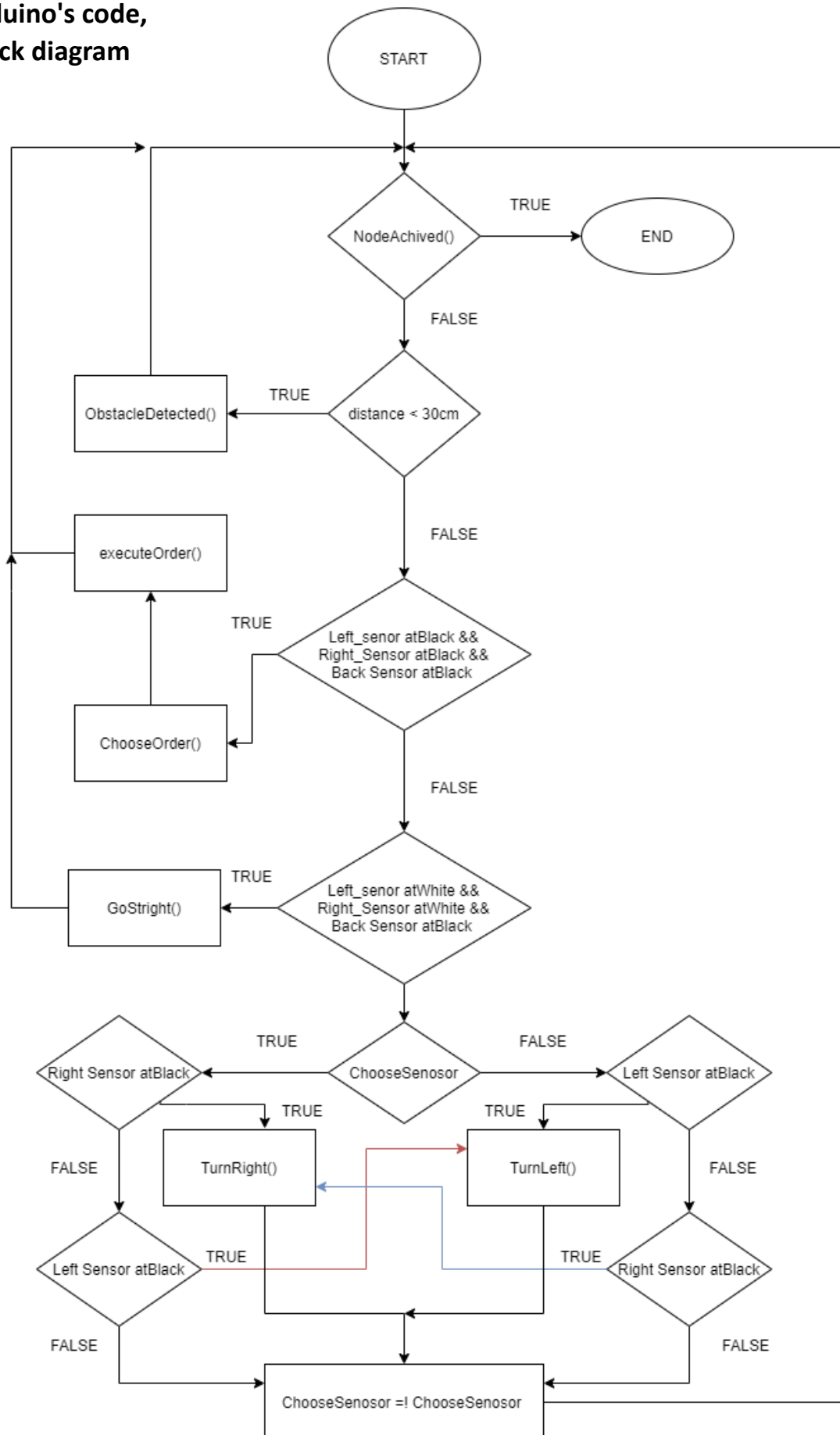
- Description of a function *heuristic(n)*:

The function returns the value of an approximate cost of a travel (in distance) from the given "n" node to the destination\_node. The values of the cost are statically stored in the dictionary.

- Description of a graph representation:

Graph is represented by an adjacent node dictionary, so each vertex(node) of a graph is mapped with its adjacent nodes and its weights(tuple).

## Arduino's code, Block diagram





### All Sensors States:

This table shows all possible states of sensors. It also shows which combination cause which action:

LEFT SENSOR	RIGHT SENSOR	BACK SENSOR	ACTION
Black	Black	Black	Intersection – chooseOrder().
Black	Black	White	NOP
Black	White	Black	Turn Left
Black	White	White	Turn Left
White	Black	Black	Turn Right
White	Black	White	Turn Right
White	White	Black	Go Straight
White	White	White	NOP

**\*\* If Black – High Signal, If White – Low Signal**

Source code can be found at :

*"Assignment\_2/ Assignment\_2.ino",*

*"A\_star/AStarAlg\_Assignment\_2.py"*