EXAMENSARBETE INOM MASKINTEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, SVERIGE 2021

Precision Court Sweeper Tennisborste

JENNY ALKE

MARIA SANDAHL

KTH
SKOLAN FÖR INDUSTRIELL TEKNIK OCH MANAGEMENT

Precision Court Sweeper

Thesis

JENNY ALKE AND MARIA SANDAHL

Bachelor's Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:32

Abstract

Tennis is a popular sport and in summer it's often played outside. When an outdoors tennis court has been used it needs to be brushed. First the whole court is brushed with a large brush and then the white lines with a smaller brush. The aim of this thesis was to design and build a working prototype of a robot who can do all of this by itself i.e sweep the court and then the white lines. The budget for components to the prototype was limited to 1 000 SEK. Tools and other resources such as 3D-printers, soldering equipment and laser cutters was provided by KTH for free. First information and inspiration about self-driving cars and driving patterns was collected and some important sources were old bachelor's thesis. Then, needed components and dimensions could be determined. In this project the main components were an Arduino Uno, two DC motors, an L298 H-bridge, an ultra sonic distance sensor, an on/off switch, AAA batteries and a 9 V battery.

The conclusions that could been drawn was that the robot can work good enough to sweep a court with only a preprogrammed path. However, to sweep the white lines, sensors would be necessary. It could also be concluded that a robot could sweep the court at the same speed as two people could do it.

Keywords: Mechatronics, Arduino Uno, DC-motor, Tennis court.

Referat

Tennis-borste

Tennis är en populär sport och på sommaren spelas den ofta utomhus. När en utomhustennisbana har använts måste den borstas. Först borstas hela banan med en stor borste och sedan de vita linjerna med en mindre borste. Syftet med denna uppsats var att designa och bygga en fungerande prototyp av en robot som kan göra allt detta av sig själv dvs. sopa tennisbanan och sedan de vita linjerna. Budgeten för komponenter till prototypen var begränsad till 1000 SEK. Verktyg och andra resurser så som 3D-skrivare, lödutrustning och laserskärare tillhandahölls av KTH gratis. Det första som gjordes var att samla information och inspiration om självkörande bilar och olika körmönster och några viktiga källor var gamla kandidatexamensuppsatser. Sedan kunde nödvändiga komponenter och dimensioner bestämmas. I detta projekt var huvudkomponenterna en Arduino Uno, två DC-motorer, en L298 H-brygga, en ultraljudssensor, en på/av-omkopplare, AAA-batterier och ett 9 V batteri.

Slutsatserna som kunde dras var att roboten kan fungera tillräckligt bra för att borsta en tennisbana med endast en förprogrammerad bana. För att sopa de vita linjerna skulle sensorer dock vara nödvändiga. En annan slutsats var att en robot kan sopa banan på samma tid som det krävs för två personer att sopa varsin halva.

Nyckelord: Mekatronik, Arduino, DC-motor, Tennisbana.

Acknowledgements

We would like to thank all of the following persons for the help provided during this project, making it possible for us to learn and complete this thesis:

Nihad Subasic, Examiner and course responsible for Degree Project in Mechatronics at KTH.

Staffan Qvarnström, Teacher in the course Degree Project in Mechatronics at KTH.

Amir Avdic, Teaching assistant in the course Degree Project in Mechatronics at KTH.

And of course we also would like to thank all of our friends and colleagues in this course who we have gotten inspired and helped by.

Contents

List of Figures

List of Tables

List of Abbreviatoins

3D	Three-dimensional
AC	Alternating current
CAD	Computer-aided design
DC	Direct current
ICSP	In Circuit Serial Programming
IR	Infra red
KTH	Kungliga Tekniska Högskolan
LIDAR	Light detection and ranging
PWM	Pulse Width Modulation
rpm	Rotations per minute
USB	Universal Serial Bus

Chapter 1

Introduction

1.1 Background

Tennis is a very popular sport and when the weather allows it, many people choose to play tennis outside. Those who have played tennis outside know that you have to sweep the court after your booked time slot. First you sweep the whole court with a large brush and then you sweep the white lines with a smaller brush. The aim of this thesis was to design and build a working prototype of a robot which can do all of this by itself i.e sweep the court and then the white lines as well. The time it takes for two people to sweep one half each is from personal experience estimated to six minutes.

1.2 Purpose

The purpose with the project was to find an effective way to sweep a tennis court. In order to do that, the following research questions had to be answered.

- How efficient can the robot be compared to the time it takes for two people to sweep each half?
- How precise is it possible to make the robot with only a predetermined and preprogrammed path?

1.3 Scope

The main goal of this thesis was to design and build a working prototype of a robot which can sweep a tennis court and the court's white lines all by itself. Since the project's duration was limited the goal was divided into two separate sub-goals with internal prioritization. First and foremost the robot should be able to sweep the big court, and after, if the time and resources allowed, the robot should be able to sweep the white lines on the court as well. The budget for components to the prototype was limited to 1 000 SEK. Tools and other resources were provided by KTH.

1.4 Method

The first step was to collect information and inspiration about self-driving cars and different driving patterns. Some important sources for this were old bachelor's thesis [8], [5], [6]. Then, needed components and dimensions could be determined. Here too were old bachelor's thesis' a great source [5], [16]. To easier answer the research questions a prototype was built and as a first step a simplified 3D simulation of the prototype was created in the program Acumen. The Acumen code can be found in appendix D. In this project an Arduino Uno, a microcontroller, has been used to control the robot. Other important components were two DC motors, an L298 H-bridge, an ultra sonic distance sensor and an on/off switch. The DC motors were given power from a shared voltage source consisting of ten 1,5 V batteries and the Arduino as well as the ultra sonic distance sensor used 5 V from a 9 V battery.

Chapter 2

Theory

2.1 Arduino Uno

Arduino Uno is a microcontroller board, see Figure 2.1, which has 14 digital pins, 6 analog inputs, a 16 MHz ceramic resonator, an USB port, a power jack and an ICSP header [4]. It is cheap and very easy to use, you only have to connect your arduino to a computer with a USB cable, an AC-to-DC adapter or a battery to get it going [4]. It is often used for constructing and programming of electronics and the programming language to control the board is C++ [19]. The analog inputs can be used to control the speed of motors by adjusting the PWM. Pulse Width Modulation (PWM) is a method used to create a square wave and there by control the voltage given to the motors which in turn enables controlling the speed. [18]

Figure 2.1. Arduino Uno [4]

2.2 DC motor

A DC (Direct current) motor is an electric motor, which turns electrical energy into mechanical energy, a principle sketch of a DC motor can be seen in Figure 2.2. This is done by electromagnetic induction. When the winding is connected to a DC motor an electric current is flowing in the winding and a magnetic field is induced.[5]

DC motors are easy to adjust without losing efficiency. In addition, they are silent, light weighted and cheap. In order to control a DC motor an H-bridge is needed.

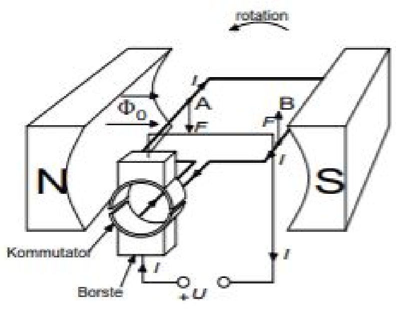


Figure 2.2. DC motor [9]

2.3 Ultra sonic distance sensor

An ultra sonic distance sensor can be used to perform distance measurements between moving or stationary objects. It uses a frequency of 40 000 Hz [12], which is way above the range of human hearing. It has one transmitter and one receiver. The transmitter emits an ultrasonic wave and the receiver receives the wave reflected back from the object. The distance d is calculated from the time between the emission and reception, see equation (2.1), where v is velocity of the sound and t is the time. It is divided by two because t is the time for the sound wave to go to the obstacle and back. [11]

$$d = \frac{vt}{2} \tag{2.1}$$

The sensor can perform precise distance measurements from 2cm to 4m with an accuracy of 3mm, a typical design of an ultra sonic distance sensor is presented in Figure 2.3. The sensor can be used to detect obstacles regardless of colour and material unless the surface is soft.

Figure 2.3. Ultra sonic distance sensor [1]

2.4 Driver card and H-bridge

An H-bridge makes it possible for an electrical circuit to let current flow in both directions. These circuits are often used in robotic and other uses where it's necessary to let the electrical motor go back and forth. The H-bridge has four switches (S1, S2, S3, S4), see Figure 2.4, that can be closed and opened. Depending on which switches are closed and opened the current will flow in different directions. The direction of the motor can be controlled by changing these switches [**DCPWM**].

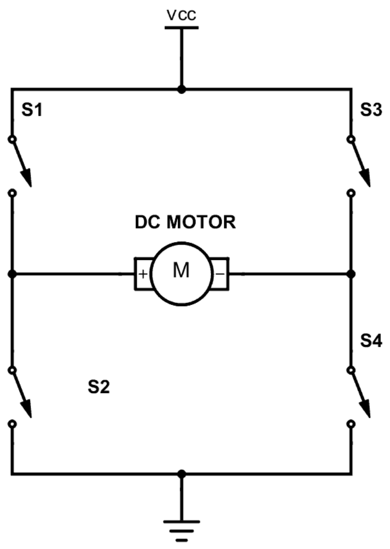


Figure 2.4. H-bridge [7]

In addition, the driver card can control the motors velocity. This is adjusted by the time the switches are on and off. If the switches always are on, the duty cycle is at 100% and if the switches always are off the duty cycle is at 0%.

2.5 Minimizing square

To cover the area of the whole court a minimizing square-route, shown in Figure 2.5, can be used. Different areal coverage efficiency was studied in the bachelor thesis Husbie 2.0. The study showed that the minimizing square movement was very efficient [16]. In addition, this is the ordinary way to sweep to get an even amount of gravel on the tennis court.

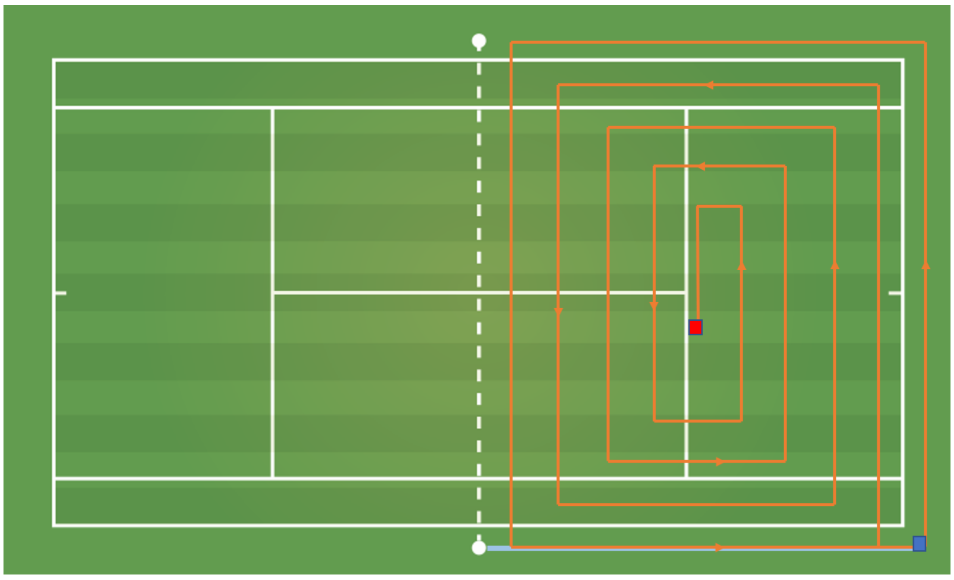


Figure 2.5. The square is minimized each lap. Made in Power Point.

The motion consist of straight and 90 degrees movements, see Figure 2.5. This goes on repeatedly and the square is decreased with a set value each lap.

Chapter 3

Demonstrator

3.1 Construction

The idea was to make the prototype's large brush about 20 cm, approximately a scale 1:10 in relation to a real standard brush, while the rest of the robot had to be larger than a scale 1:10 in order to fit all the components needed. However, the brush that actually was used on the prototype measured 10 cm in width but when programming the path it was assumed to be 20 cm. The large brush was placed at the back of the robot and the small one was supposed to be placed in the middle, in line with the front wheel, on the underside of the robot body. The actual prototype did not have a small brush and the calculations for the white lines were made with the center position of the robots underside. The robot has three wheels, one supporting swivel caster wheel without a motor and two larger rubber wheels, each connected to separate DC-motors. The DC-motors connects to an L298 dual full-bridge driver which in turn connects to a 15 V power source. The Arduino have a separate portable power source, a 9 V battery connecting to the micro-controller via a DC-cable. An ultrasonic distance sensor was placed in the front of the robot body in order to increase safety, both for users as well as the robot itself. The sensor signals if an object disturbs the robot's path and comes too close which makes the robot stop. A 3D model of the robot was made, see Figure 3.1.

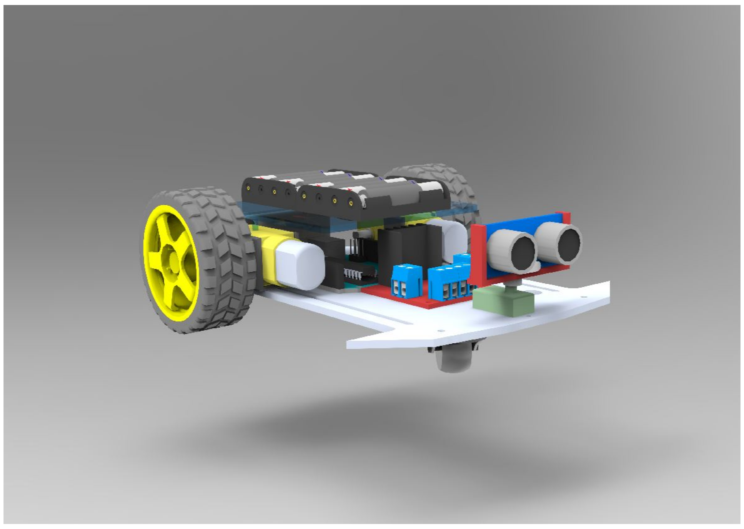


Figure 3.1. CAD model of the robot without the brushes. Made in Solid Edge.

3.2 Hardware

Sufficient current is required to power the robot. The DC motors used in this project is a gear motor MG-6-48 [13]. They require 6 V each and has a maximum speed of 230 rpm. The motors total requirement is 12 V. A power source of 15 V is therefore connected to the H-bridge, which is needed to be able to control the DC motors. The H-bridge has a power protection making it possible to use a power source of 15 V when only 12 V is needed. More information about the H-bridge can be found in appendix A.

The Arduino Uno requires 5 V, see appendix B. However more voltage was needed. This is probably because there is a power drop when components are connected. Therefore, a battery with 9 V is connected to the Arduino. Likewise the H-bridge, the Arduino also has a build in power protection which allows 9 V battery.

An on-off-switch was connected between the Arduino and the battery for allowing the user to start the sweeper on demand.

3.3 Electronics

The components were connected as described in the hardware section. To get a better view over the circuit and to understand how everything is connected an electrical circuit diagram was made. The result is shown in Figure 3.2 below. The H-bridge is connected to the Arduinos PWM pins to enable the Arduino to control the speed of the motors. The ultra sonic distance sensor is also connected to the Arduino's digital pins, the 5 V pin and ground.

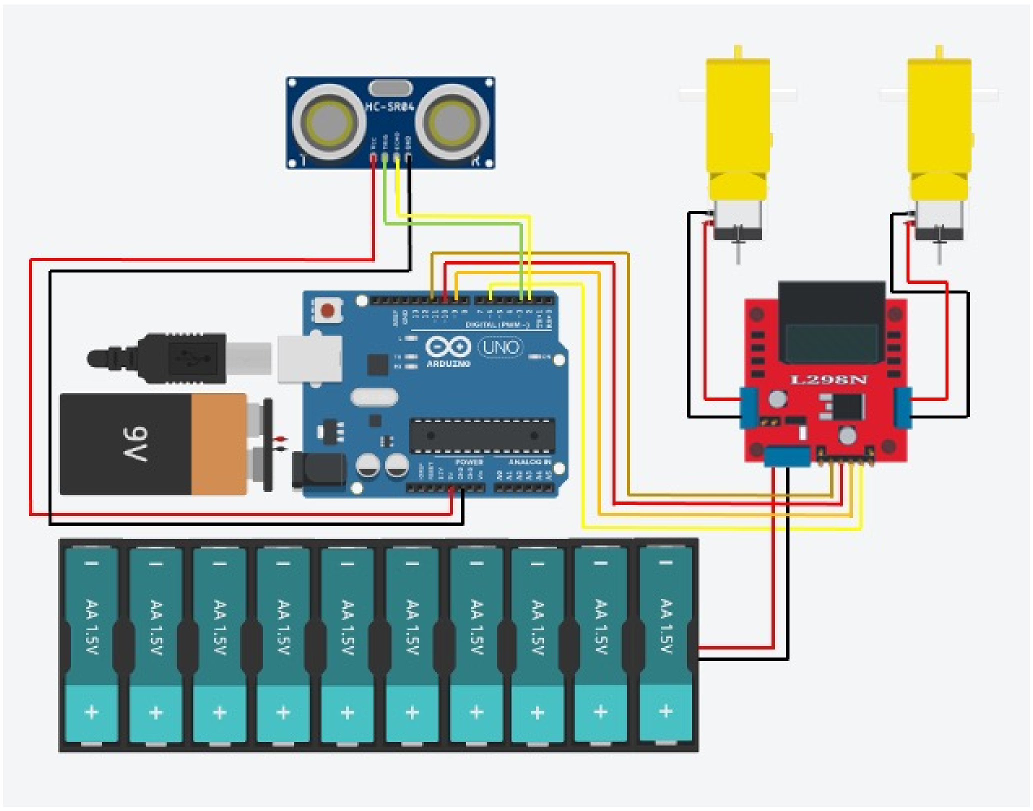


Figure 3.2. Electrical circuit diagram. Made in Tinkercad and Power Point.

3.4 Software

The microcontroller board that has been used to control the court sweeper and its components is an Arduino Uno. The Arduino code has been written in the programming language C++, the code can be found in appendix E, [17]. The Arduino can, with help from the H-bridge, control the two DC-motors and make the robot go forward in the wanted velocity and time. In addition, the Arduino also has control over the ultra sonic distance sensor, used to detect obstacles. A flow chart diagram was made to show the sequence of events for the court sweeper from start until finish, see Figure 3.3.

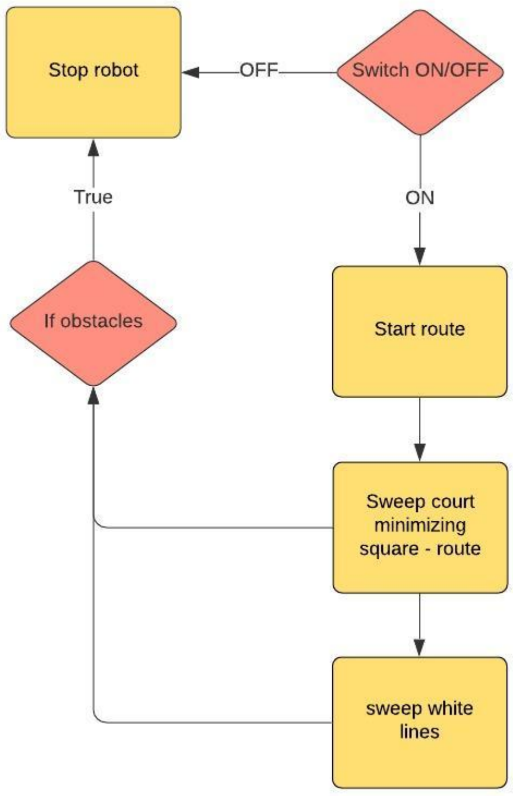


Figure 3.3. Flowchart for the arduino code. Made in Lucidchart.

The different distances the path consists of is converted to time intervals when the motors are on or off with a set PWM. This time is calculated according to equation (3.1). t is the time in seconds, $rotations$ is how many rotations the wheel has to spin to move a set distance, the maximum rpm and the maximum PWM are taken from the motor's and Arduino's specifications and PWM is chosen depending on the wanted speed on the wheels.

$$t = \frac{rotations}{rpm} = \frac{\frac{l}{d\pi}}{\frac{rpm_{max}PWM}{PWM_{max}}} \tag{3.1}$$

To turn 90 degrees one wheel has to move the distance calculated in equation (3.2) while the other wheel is still [10], see Figure 3.4.

$$d_{90^o} = \frac{130\pi}{4} \tag{3.2}$$

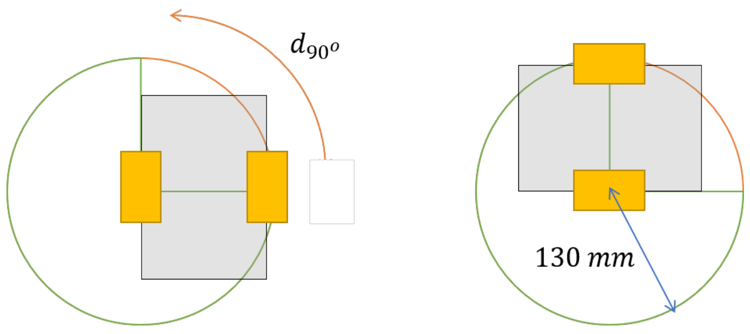


Figure 3.4. 90 degrees turn. Made in Power Point.

A tennis court can have different surfaces but they always have the same dimensions [15]. The robot will be programmed with a set start position and when placed and turned on it will start the preplanned route to sweep the court and then the white lines.

Chapter 4

Result

The average time that it took for the robot prototype to sweep a court and its white lines that was scaled down with a factor 10 was 1 minute and 29 seconds, this was calculated from the time of ten different test runs presented in Table 4.1.

Table 4.1. Table presenting the times from different test runs

Test run	Time in seconds
A	91
B	89
B	87
C	90
D	88
E	86
F	89
G	91
H	93
I	88

An illustration of the average path from the same test runs presented in the table above compared to the desired preprogrammed path is presented in Figures 4.1, 4.2, 4.3 and 4.4 down below.

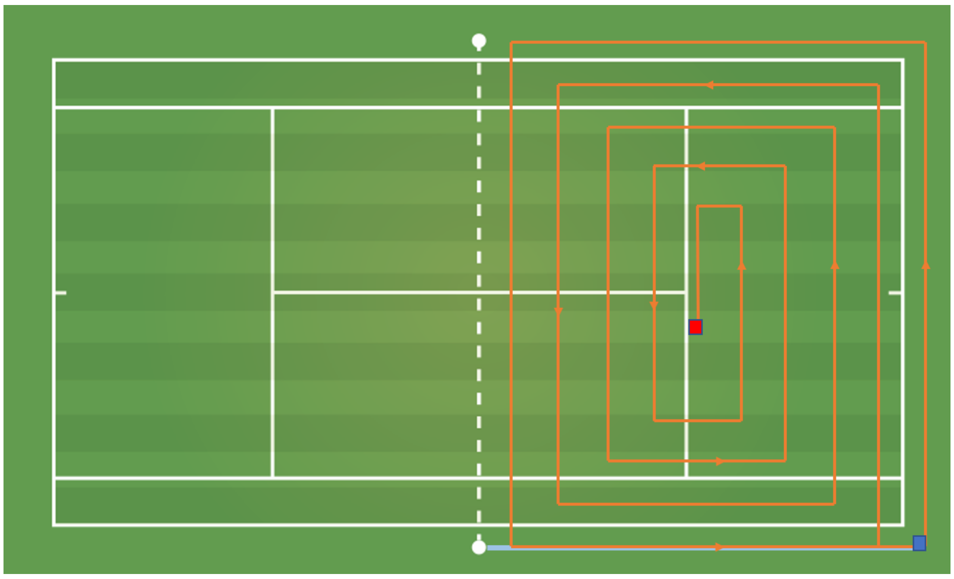


Figure 4.1. Preprogrammed path. Made in Power Point.

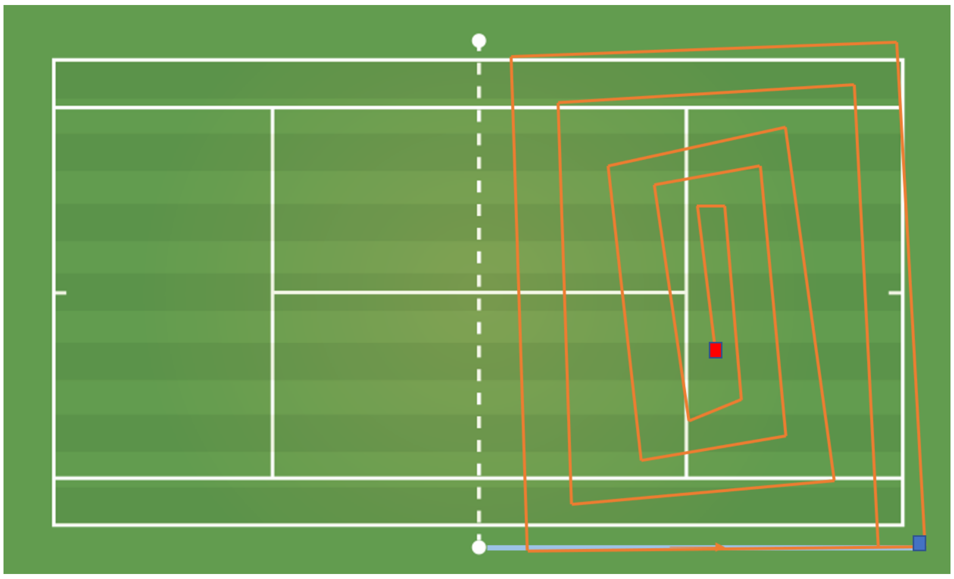


Figure 4.2. Average actual path of robot. Made in Power Point.

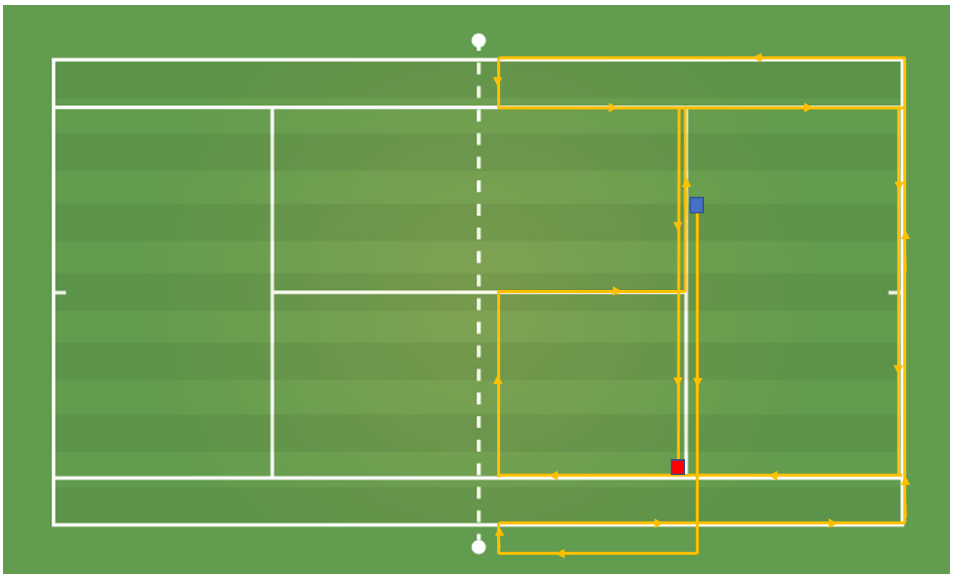


Figure 4.3. Preprogrammed path of white lines. Made in Power Point.

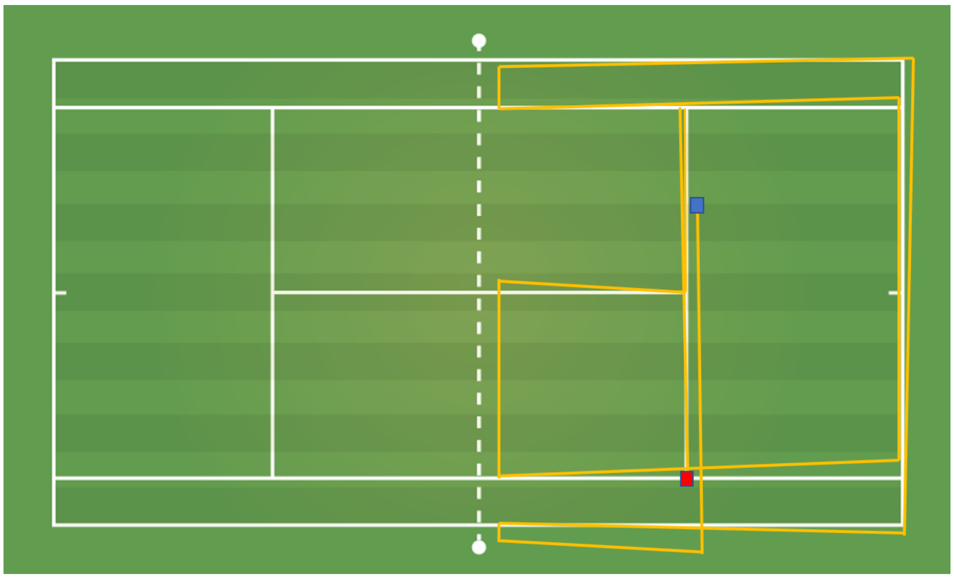


Figure 4.4. Average actual path of the white lines. Made in Power Point.

Chapter 5

Discussion

As seen in the previous chapter the actual route is not identical to the programmed path. The path the robot moves has been very different depending on what ground it drives on. The path it takes outside on asphalt or concrete is completely different from the path it takes inside on a wooden floor even if it is the same exact code. If the external conditions would've been optimal, this problem would probably not have occurred. The needed voltage for the DC motors depends on a lot of factors. The same voltage will not always give the same velocity. For example, the friction on the surface affect how much current is needed. In addition, the friction will always vary a little on the left and right wheel as well, since the surface is not constant. Lack of enough friction also makes the wheels slip, especially when turning and one wheel is supposed to be completely still. All of this will result in the robot not going exactly straight forward when wanted and the 90 degrees turn will not be precise enough. If the turn happens to be 92 degrees instead of 90, the angle will be 20 degrees wrong after only ten turns. A domino effect occurs. The same effect applies for other external factors. The center of gravity and the mass of the robot also affect which voltage is needed for each DC motor.

The robot is, in other words, very sensitive for these external factors. The prototype in this project is made with limited resources and therefore it is not perfect. The wheels are not perfectly aligned and the connections are a bit unstable which also adds to the domino affect. However, the prototype is sufficient for proof of concept and we do have enough to answer the research questions. So, how efficient can the robot be compared to the time it takes for two people to sweep each half? So the time it took for the prototype to sweep one half is approximately 1,5 minutes. Both the prototype and the test court was scaled down to the same proportion compared to a real tennis court. With that said, it is assumed that the time would be the same with the right dimensions. The whole court will then take 3 minutes to sweep which is less than six minutes, the estimated time it takes for two people to sweep each half.

The second research question also has been answered, how precise enough is it possible to make the robot with only a predetermined path? As said earlier in the discussion, the robot is very sensitive for external factors which result in the actual path not being the same as the preprogrammed path. The robot is therefore not precise enough and would need sensors to be. Possible sensors for this project will be discussed below in recommendations for further work.

Although the robot didn't function exactly as planned it would work good enough to sweep a court with sufficient result. Even if it couldn't move in perfect 90 degrees turns or completely straight it's path would still completely cover the whole court, see Figure 4.1 and 4.2. The risks however is that it could get caught up in the net when trying to move across the court to the opposite side and that it would finish in different places every time. However it is not precise enough to sweep the white lines since that part of the route requires more precision, see Figure 4.3 and 4.4.

Chapter 6

Recommendation for further work

Since the robot was much more sensitive for external factors than expected, some sensors would be needed to improve it's precision. With sensors it would be possible for the robot to always know where it is and in what direction it should go next. The problem with the domino effect would have been solved.

One option could be a LIDAR (Light Detection and Ranging) sensor, which is an optical remote sensing system that can identify the distance to an object by illuminating it with laser [2]. However, the LIDAR would be very complicated since it would have had a hard time identifying the net and the round poles would not be enough for the robot to identify a given position. The same problem would appear with the ultra sonic distance sensor since the net is relatively soft. An ultra sonic distance sensor works better when used as now, as a safety measure to avoid obstacles.

Another options would be to use an IR (infra red) sensor. It has one IR-led as a transmitter and one as a receiver. The measured photocurrent will be smaller when the sensor is pointed at a black surface than when pointed at a lighter surface. This makes detection of black and white surfaces possible [3]. This sensor could only been used to identify and follow white lines but not identify where on the court the robot is at a given time.

The best option for this project would probably be visual recognition. Computer vision could be used to recognize how the court looks like. A pixy camera would be an easy way to add vision to the robot and is easy to implement on the micro-controller used in this project, but was unfortunately too expensive for the limited budget. The pixy can learn to detect objects that you teach it [14]. It can find the net and from there get to the starting position on the court. More information about the sensors are presented in appendix C.

Another way to improve this project would be to use components with higher quality. Because of the limited budget, the quality on the DC motor, H-bridge and the construction over all is not as good as wanted. Better components would solve the problem with the unstable connections as well.

Whole bibliography

[1] RS components AB. *Ultrasonic Distance Sensor Module.* https://se.rs-online.com/web/p/sensor-development-tools/7813020/?cm_mmc=SE-PLA-DS3A-_-google-_-CSS_SE_EN_Raspberry_Pi_. visited on 2021-02-10.

[2] Abhi Abhinav. *A few things about LiDAR.* https://create.arduino.cc/projecthub/Abhinav_Abhi/arduino-lidar-917404. visited on 2021-02-10.

[3] Tess Antonsson and Sofia Jönsson. "Pac-King". DEGREE PROJECT IN MECHANICAL ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2019. URL: https://kth.diva-portal.org/smash/get/diva2:1373885/FULLTEXT01.pdf.

[4] Arduino.cc. *ARDUINO UNO REV3.* https://store.arduino.cc/arduino-uno-rev3. visited on 2021-02-10.

[5] Joel Bergman and Jonas Lind. "Robot Vacuum cleaner". DEGREE PROJECT IN ELECTRONICS AND COMPUTER ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2019. URL: http://www.diva-portal.org/smash/get/diva2:1373823/FULLTEXT01.pdf.

[6] Carl Bondesson and Erik Stigenius. "Autonom bil med enkapseldator och ultraljudssensorer". Examensarbete 15 hp, UPPSALA UNIVERSITET, Teknisk-naturvetenskaplig fakultet UTH-enheten. 2015. URL: https://www.diva-portal.org/smash/get/diva2:823450/FULLTEXT01.pdf.

[7] buildelectroniccircuits. *H-bridge concept.* https://www.build-electronic-circuits.com/h-bridge/. visited on 2021-02-11.

[8] Jacob Ekesund. "Self-driving car". EXAMENSARBETE INOM TEKNIK, GRUNDNIVÅ, KTH STOCKHOLM, SVERIGE 2016. URL: http://www.diva-portal.org/smash/get/diva2:955268/FULLTEXT01.pdf.

[9] Hans Johansson. *Elektroteknik.* Institutionen för Maskinkonstruktion: Kungliga Tekniska Högskolan, 2013.

[10] Khondker Jahid Reza Kazi Mahmud Hasan Abdullah-Al-Nahid. "Path planning algorithm development for autonomous vacuum cleaner robots". In: (2014). URL: https://www.diva-portal.org/smash/get/diva2:1213402/FULLTEXT01.pdf.

[11] P Bilik L Koval J Vanus. “Distance Measuring by Ultrasonic Sensor”. In: (2016). URL: https://reader.elsevier.com/reader/sd/pii/S2405896316326623.

[12] K. Bharat Kumar N. Anju Latha B. Rama Murthy. “Distance Sensing with Ultrasonic Sensor and Arduino”. In: (2016). URL: https://d1wqtxts1xzle7.cloudfront.net/55881151/V2I5-1189.pdf?1519398297=&response-content-disposition=inline%3B+filename%3DDistance_Sensing_with_Ultrasonic_Sensor.pdf&Expires=1620384687&Signature=GjeO-U~hDKXBvvYhyL9Gcqp9XJQd6zJ
d~Ryw1ecOYws3yz6gqCRVerkY9BGcJwdApakai1aXUdPo03~St0Jb8c2pxJGo1hZYy~-
WqJ7qe4j-7mINPWGesFuUgNFEQX7eWJYEi4cWlzXf3h8JqWedTsxb7hxpYKrIqJFTxQM95HGBetuyuY43e
~pDQKaCvrkuVeqiPx~eeG4KGLbyPzqre2VFc280eahH11o~MpekIZOz6CcFfKMESfj2Ok69CfYVQk3gTdE
_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

[13] Olimex. *MG-6-48*. https://www.olimex.com/Products/Robot-CNC-Parts/GearMotors/MG-6-48/. visited on 2021-04-07.

[14] Pixycam. *Introducing Pixy2*. https://pixycam.com/. visited on 2021-02-19.

[15] TSC Tennis court supply. *EVERYTHING TO KNOW ABOUT TENNIS COURT DIMENSIONS, SIZES, AND MEASUREMENTS*. https://www.tenniscourtsupply.com/everything-about-tennis-court-dimensions.html. visited on 2021-02-19.

[16] Erik Svensson and Daria Granström. “Husbie 2.0”. DEGREE PROJECT IN MECHANICAL ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2017. URL: http://www.diva-portal.org/smash/get/diva2:1199995/FULLTEXT01.pdf.

[17] CIRCUITO TEAM. *EVERYTHING YOU NEED TO KNOW ABOUT ARDUINO CODE*. https://www.circuito.io/blog/arduino-code/. visited on 2021-04-07.

[18] CIRCUITO TEAM. *PWM arduino*. https://www.arduino.cc/en/Tutorial/Foundations/PWM. visited on 2021-05-03.

[19] Lianzhi Jiang Zhijun Liu. “The Working Principle Of An Arduino”. In: (2011). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6997578.

Books only

[9] Hans Johansson. *Elektroteknik*. Institutionen för Maskinkonstruktion: Kungliga Tekniska Högskolan, 2013.

Articles only

[10] Khondker Jahid Reza Kazi Mahmud Hasan Abdullah-Al-Nahid. “Path planning algorithm development for autonomous vacuum cleaner robots”. In: (2014). URL: https://www.diva-portal.org/smash/get/diva2:1213402/FULLTEXT01.pdf.

[11] P Bilik L Koval J Vanus. “Distance Measuring by Ultrasonic Sensor”. In: (2016). URL: https://reader.elsevier.com/reader/sd/pii/S2405896316326623.

[12] K. Bharat Kumar N. Anju Latha B. Rama Murthy. “Distance Sensing with Ultrasonic Sensor and Arduino”. In: (2016). URL: https://d1wqtxts1xzle7.cloudfront.net/55881151/V2I5-1189.pdf?1519398297=&response-content-disposition=inline%3B+filename%3DDistance_Sensing_with_Ultrasonic_Sensor.pdf&Expires=1620384687&Signature=GjeO-U~hDKXBvvYhyL9Gcqp9XJQd6zJ
d~Ryw1ecOYws3yz6gqCRVerkY9BGcJwdApakai1aXUdPo03~St0Jb8c2pxJGo1hZYy~-
WqJ7qe4j-7mINPWGesFuUgNFEQX7eWJYEi4cWlzXf3h8JqWedTsxb7hxpYKrIqJFTxQM95HGBetuyuY43e
~pDQKaCvrkuVeqiPx~eeG4KGLbyPzqre2VFc280eahH11o~MpekIZOz6CcFfKMESfj20k69CfYVQk3gTdE
_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

[19] Lianzhi Jiang Zhijun Liu. “The Working Principle Of An Arduino”. In: (2011). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6997578.

Thesis’s only

[3] Tess Antonsson and Sofia Jönsson. “Pac-King”. DEGREE PROJECT IN MECHANICAL ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2019. URL: https://kth.diva-portal.org/smash/get/diva2:1373885/FULLTEXT01.pdf.

[5] Joel Bergman and Jonas Lind. “Robot Vacuum cleaner”. DEGREE PROJECT IN ELECTRONICS AND COMPUTER ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2019. URL: http://www.diva-portal.org/smash/get/diva2:1373823/FULLTEXT01.pdf.

[6] Carl Bondesson and Erik Stigenius. “Autonom bil med enkapseldator och ultraljudssensorer”. Examensarbete 15 hp, UPPSALA UNIVERSITET, Teknisk-naturvetenskaplig fakultet UTH-enheten. 2015. URL: https://www.diva-portal.org/smash/get/diva2:823450/FULLTEXT01.pdf.

[8] Jacob Ekesund. “Self-driving car”. EXAMENSARBETE INOM TEKNIK, GRUNDNIVÅ, KTH STOCKHOLM, SVERIGE 2016. URL: http://www.diva-portal.org/smash/get/diva2:955268/FULLTEXT01.pdf.

[16] Erik Svensson and Daria Granström. “Husbie 2.0”. DEGREE PROJECT IN MECHANICAL ENGINEERING, FIRST CYCLE, KTH STOCKHOLM, SWEDEN 2017. URL: http://www.diva-portal.org/smash/get/diva2:1199995/FULLTEXT01.pdf.

Websites only

[1] RS components AB. *Ultrasonic Distance Sensor Module*. `https://se.rs-online.com/web/p/sensor-development-tools/7813020/?cm_mmc=SE-PLA-DS3A-_-google-_-CSS_SE_EN_Raspberry_Pi_`. visited on 2021-02-10.

[2] Abhi Abhinav. *A few things about LiDAR*. `https://create.arduino.cc/projecthub/Abhinav_Abhi/arduino-lidar-917404`. visited on 2021-02-10.

[4] Arduino.cc. *ARDUINO UNO REV3*. `https://store.arduino.cc/arduino-uno-rev3`. visited on 2021-02-10.

[7] buildelectroniccircuits. *H-bridge concept*. `https://www.build-electronic-circuits.com/h-bridge/`. visited on 2021-02-11.

[13] Olimex. *MG-6-48*. `https://www.olimex.com/Products/Robot-CNC-Parts/GearMotors/MG-6-48/`. visited on 2021-04-07.

[14] Pixycam. *Introducing Pixy2*. `https://pixycam.com/`. visited on 2021-02-19.

[15] TSC Tennis court supply. *EVERYTHING TO KNOW ABOUT TENNIS COURT DIMENSIONS, SIZES, AND MEASUREMENTS*. `https://www.tenniscourtsupply.com/everything-about-tennis-court-dimensions.html`. visited on 2021-02-19.

[17] CIRCUITO TEAM. *EVERYTHING YOU NEED TO KNOW ABOUT ARDUINO CODE*. `https://www.circuito.io/blog/arduino-code/`. visited on 2021-04-07.

[18] CIRCUITO TEAM. *PWM arduino*. `https://www.arduino.cc/en/Tutorial/Foundations/PWM`. visited on 2021-05-03.

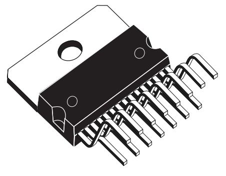
Appendix A

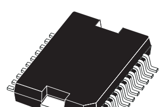
Full bridge driver

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)


Multiwatt15

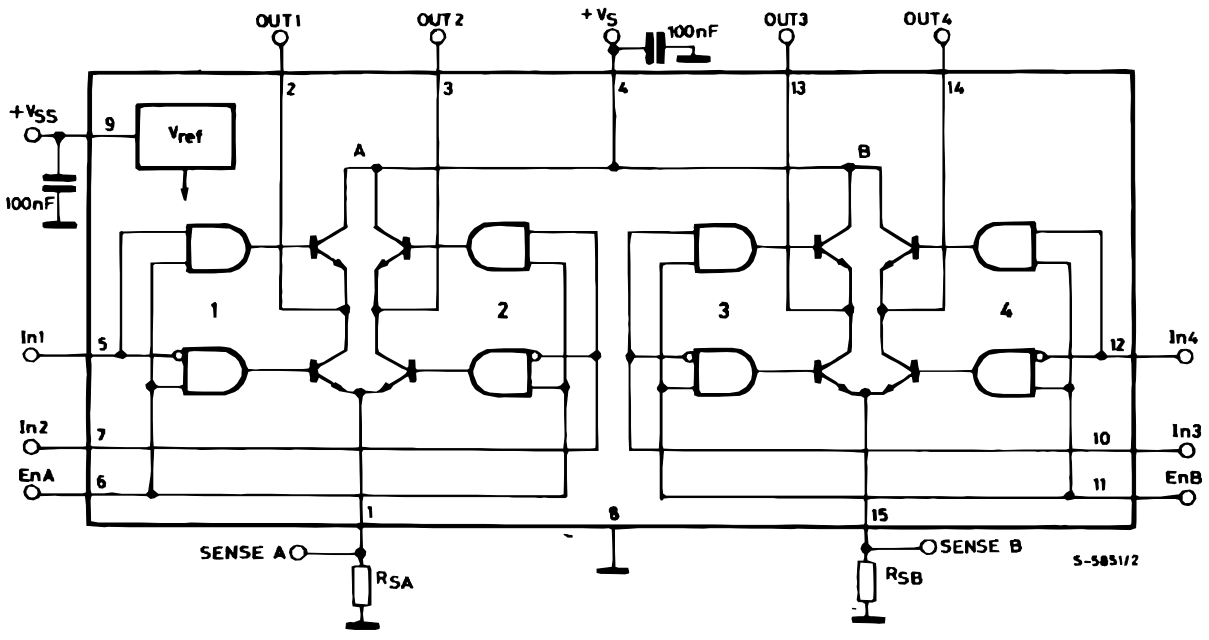

PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

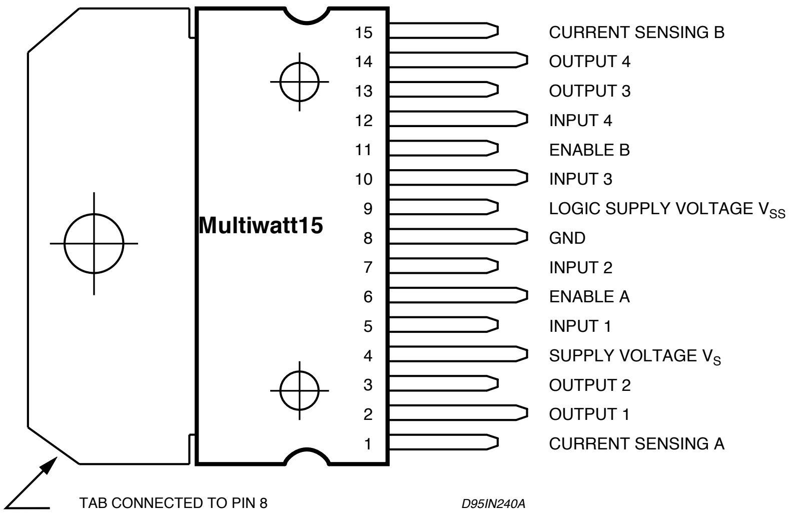
BLOCK DIAGRAM

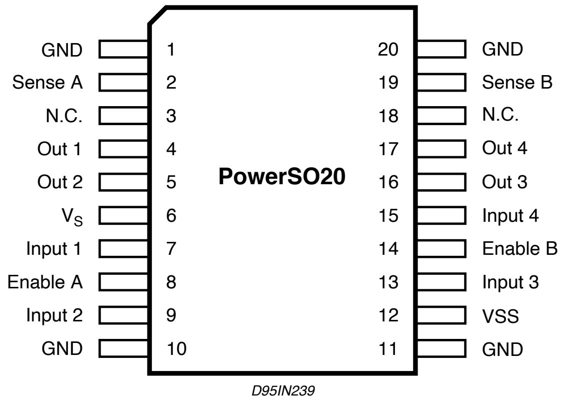


ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100μs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)





THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	μA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	μA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	μA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	μA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

Appendix B

Arduino UNO

Atmel®

ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32×8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

- I/O and packages
  - 23 programmable I/O lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
  - 2.7V to 5.5V for ATmega328P
- Temperature range:
  - Automotive temperature range: –40°C to +125°C
- Speed grade:
  - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
  - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
  - Active mode: 1.5mA at 3V - 4MHz
  - Power-down mode: 1μA at 3V

Appendix C

Sensor research

LIDAR

LIDAR, which stands for Light Detection and Ranging, is an optical remote sensing system that can identify the distance to an object by illuminating it with laser. LIDAR is often used to get an overview of the environment by providing 2D maps or object classification. The device consists of a transmitter which transmits a laser beam and a receiver which can identify the distance to the object hit by the laser by calculating the time it takes for the light to reach the object and then return. A mechanism built with a servo motor, an Arduino and a driver card with a builtin H-bridge sweeps the laser beam so it covers the required surroundings. [1]

The problem with using a LIDAR sensor in this project is that it wouldn't be able to identify the net. Therefore, it would be hard for the court sweeper to locate itself on the court.

IR sensor

IR sensors have both one transmitter and one receiver. An IR-LED as a transmitter that sends out infrared light. When the IR LED points at a dark surface almost all light is absorbed. If the surface is very light almost all light is reflected back. When reflected back, it enters the photo diode which works as a receiver, shown in figure 1. The measured photocurrent will be smaller when the sensor is pointed at a black surface than when pointed at a lighter surface. This makes detection of black and white surfaces possible.[2]

Using IR-sensors in this project could be a problem. Since the white lines on the tennis court at times will be covered with sand, the court sweeper will lose track of the path.

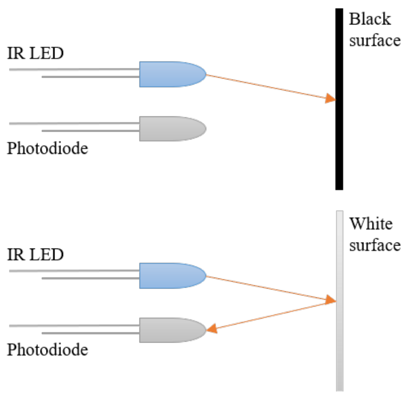


Figure 1. IR sensor pointing at black and white surface. Made in Power Point.

Visual recognition

Computer vision could been used to recognize how the court looks like. A pixy camera, seen in figure 2, would be an easy way to add vision to the robot and is easy to implement on the microcontroller used in this project. The pixy can learn to detect objects that you teach it. It can find the net and from there get to the starting position on the court.[4]

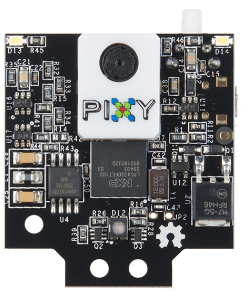


Figure 2. Pixy camera [3]

Using visual recognition would probably be the best way to solve this problem. It could also been used to detect the white lines.

Unfortunately, this project has some restrictions. The pixy camera costs almost 1000 swedish crones[3] and since we have other components to purchase as well, our budget won't allow it.

Due to the limited amount of time and financial resources none of the sensors was an optimal fit for this project. The LIDAR would be very complicated since it would have had a hard time identifying the net and the round poles would not be enough for the robot to identify a given position. The same problem would appear with the ultra sonic distance sensors since the net is relatively soft. The Pixy cam could have worked but was unfortunately a too expensive option.

Appendix D

Acumen code

```
// Code by Jenny Alke & Maria Sandahl
// Group 14

// Code for bachelors' thesis VT2021
// TRITA-ITM-EX 2021:32
// Last edited:25/5
// University: Royal Institute of Technology

// This is a code for an 3D simulation of
// a prototype of a robot constructed for
// sweeping a tennis court.


model Main(simulator) =
  initially
     _3D = (), _3DView = ()
  always
     _3D = ((Box
               center = (0,0,0)
                 size = (4,2,1)
                color = yellow
         transparency = 1), //Robot body


             (Box
               center = (2.5,0,-0.5)
                 size = (1,4,0.5)
                color = magenta
         transparency = 1) //Brush


         (Cylinder
               center = (-1,0,-0.5)
                 size = (0.1,0.27)
                color = green
         transparency = 1) //Small front wheel


         (Cylinder
               center = (0.7,-1,-0.28)
                 size = (0.25,0.5)
                color = green
         transparency = 1) //Left rear wheel


          (Cylinder
               center = (0.7,1,-0.28)
                 size = (0.25,0.5)
                color = green
         transparency = 1)), //Right rear wheel


       _3DView = ((0, -10, 5), (0, 1, -1))
```

Appendix E

Arduino Code

```
/*
 * Code by Jenny Alke & Maria Sandahl
 * Group 14
 *
 * Code for bachelors' thesis 2021
 * TRITA-ITM-EX 2021:32
 * Last edited:7/5 2021
 *
 * This is a code for an arduino uno robot,
 * a prototype of a robot constructed for
 * sweeping a tennis court. Using a minimizing
 * square function.
 */

#include <Wire.h>

//Pin number definition for ultrasonic distance sensor
const int trigPin = 3;
const int echoPin = 2;

//Variables for ultrasonic distanceSensors
long duration;
int distance;
const int minDistance = 0.001; //Minimum distance for obstacles [m]

unsigned long start_t;

const float Pi = 3.14159;
const int d = 0.07; //Diameter wheels [m]
const int b = 0.155; //Width car [m]

int O = d*Pi; //Circumference wheels [m]
int state = 0;

//Factors for converting distance to time
const float factor = (255*60000)/(230*O*60);
const float turnDistance = (((255*60000)/(230*O*60))*((130*Pi)/4)); //Distance for 90 degrees turn

//Pin number definition for DC-motors
const int motorLeftForward = 10;
const int motorLeftBack = 11;
const int motorRightForward = 9;
const int motorRightBack = 6;

//Defining which pins are inputs/outputs
void setup() {
  Serial.begin(9600);
  pinMode(motorLeftForward, OUTPUT);
  pinMode(motorLeftBack, OUTPUT);
  pinMode(motorRightForward, OUTPUT);
  pinMode(motorRightBack, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

//Function for driving forward
void forward(){
  digitalWrite(motorLeftForward, HIGH);
  analogWrite(motorLeftForward, 60);
  digitalWrite(motorLeftBack, LOW);
  analogWrite(motorLeftBack, 0);
  digitalWrite(motorRightForward, HIGH);
  analogWrite(motorRightForward, 60);
  digitalWrite(motorRightBack, LOW);
  analogWrite(motorRightBack, 0);
  }

//Function for driving backwards
void backwards(){
  digitalWrite(motorLeftForward, LOW);
  analogWrite(motorLeftForward, 0);
  digitalWrite(motorLeftBack, HIGH);
  analogWrite(motorLeftBack, 60);
  digitalWrite(motorRightForward, LOW);
  analogWrite(motorRightForward, 0);
  digitalWrite(motorRightBack, HIGH);
  analogWrite(motorRightBack, 60);
  }

//Function for turning left
void turnLeft(){
  digitalWrite(motorLeftForward, LOW);
  analogWrite(motorLeftForward, 0);
  digitalWrite(motorLeftBack, LOW);
  analogWrite(motorLeftBack, 0);
  digitalWrite(motorRightForward, HIGH);
  analogWrite(motorRightForward, 60);
  digitalWrite(motorRightBack, LOW);
  analogWrite(motorRightBack, 0);
  }

//Function for turning right
void turnRight(){
  digitalWrite(motorLeftForward, HIGH);
  analogWrite(motorLeftForward, 60);
  digitalWrite(motorLeftBack, LOW);
  analogWrite(motorLeftBack, 0);
  digitalWrite(motorRightForward, LOW);
  analogWrite(motorRightForward, 0);
  digitalWrite(motorRightBack, LOW);
  analogWrite(motorRightBack, 0);
  }

//Function for stoping
void stopRobot(){
  digitalWrite(motorLeftForward, LOW);
  digitalWrite(motorLeftBack, LOW);
  digitalWrite(motorRightForward, LOW);
  digitalWrite(motorRightBack, LOW);
  }


//Coding the path and lines, with switch state,
//one case for each "distance movement".
void loop() {

start_t = millis(); //Time

//The route for sweeping the court
switch (state){
case 0:
  if(start_t <= (factor*1.3)){
    forward();
    delay(1000);
    state = 1;
  }
  break;

case 1:
start_t = millis();
  if(start_t <= turnDistance){
    turnLeft();
    delay(1000);
    state = 2;
  }
  break;

case 2:
  start_t = millis();
    if(start_t <= (factor*1.3)){
      forward();
      delay(1000);
      state = 3;
    }
    break;

  case 3:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 4;
    }
    break;

case 4:
  start_t = millis();
    if(start_t <= (factor*1.3)){
      forward();
      delay(1000);
      state = 5;
    }
    break;

case 5:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 6;
    }
    break;

case 6:
  start_t = millis();
    if(start_t <= (factor*1.1)){
      forward();
      delay(1000);
      state = 7;
    }
    break;

case 7:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 8;
    }
    break;

case 8:
  start_t = millis();
    if(start_t <= (factor*1.1)){
      forward();
      delay(1000);
      state = 9;
    }
    break;

case 9:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 10;
    }
    break;

case 10:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 11;
    }
    break;

case 11:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 12;
    }
    break;

case 12:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 13;
    }
    break;

case 13:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 14;
    }
    break;

case 14:
  start_t = millis();
    if(start_t <= (factor*0.7)){
      forward();
      delay(1000);
      state = 15;
    }
    break;

case 15:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 16;
    }
    break;

case 16:
  start_t = millis();
    if(start_t <= (factor*0.7)){
      forward();
      delay(1000);
      state = 17;
    }
    break;

case 17:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 18;
    }
    break;

case 18:
  start_t = millis();
    if(start_t <= (factor*0.5)){
      forward();
      delay(1000);
      state = 19;
    }
    break;

case 19:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 20;
    }
    break;

case 20:
  start_t = millis();
    if(start_t <= (factor*0.5)){
      forward();
      delay(1000);
      state = 21;
    }
    break;

case 21:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 22;
    }
    break;

case 22:
  start_t = millis();
    if(start_t <= (factor*0.3)){
      forward();
      delay(1000);
      state = 23;
    }
    break;

case 23:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 24;
    }
    break;

case 24:
  start_t = millis();
    if(start_t <= (factor*0.3)){
      forward();
      delay(1000);
      state = 25;
    }
    break;

case 25:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 26;
    }
    break;

case 26:
  start_t = millis();
    if(start_t <= (factor*0.1)){
      forward();
      delay(1000);
      state = 27;
    }
    break;


case 27:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 28;
    }
    break;

//Route of the white lines begins here
case 28:
  start_t = millis();
    if(start_t <= (factor*0.7)){
      forward();
      delay(1000);
      state = 29;
    }
    break;

case 29:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 30;
    }
    break;

case 30:
  start_t = millis();
    if(start_t <= (factor*0.6)){
      forward();
      delay(1000);
      state = 31;
    }
    break;

case 31:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 32;
    }
    break;

case 32:
  start_t = millis();
    if(start_t <= (factor*0.2)){
      forward();
      delay(1000);
      state = 33;
    }
    break;

case 33:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 34;
    }
    break;

case 34:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 35;
    }
    break;

case 35:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 36;
    }
    break;


case 36:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 37;
    }
    break;

case 37:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 38;
    }
    break;

case 38:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 39;
    }
    break;

case 39:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 40;
    }
    break;

case 40:
  start_t = millis();
    if(start_t <= (factor*0.2)){
      forward();
      delay(1000);
      state = 41;
    }
    break;

case 41:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 42;
    }
    break;

case 42:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1800);
      state = 43;
    }
    break;

case 43:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 44;
    }
    break;

case 44:
  start_t = millis();
    if(start_t <= (factor*0.7)){
      forward();
      delay(1000);
      state = 45;
    }
    break;

case 45:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 46;
    }
    break;

case 46:
  start_t = millis();
    if(start_t <= (factor*0.9)){
      forward();
      delay(1000);
      state = 47;
    }
    break;

case 47:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 48;
    }
    break;

case 48:
  start_t = millis();
    if(start_t <= (factor*0.35)){
      forward();
      delay(1000);
      state = 49;
    }
    break;

case 49:
  start_t = millis();
    if(start_t <= turnDistance){
      turnRight();
      delay(1000);
      state = 50;
    }
    break;

case 50:
  start_t = millis();
    if(start_t <= (factor*0.6)){
      forward();
      delay(1000);
      state = 51;
    }
    break;

case 51:
  start_t = millis();
    if(start_t <= turnDistance){
      turnLeft();
      delay(1000);
      state = 52;
    }
    break;

case 52:
  start_t = millis();
    if(start_t <= (factor*0.35)){
      forward();
      delay(1000);
      state = 53;
    }
    break;

case 53:
  start_t = millis();
     if(start_t <= (factor*0.7)){
      backwards();
      delay(1000);
      state = 54;
    }
    break;

case 54:
    stopRobot();
break;
}

  //Ultrasonic distance sensor
  while(1){

    //Reading trig and echopin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH); //the time it takes for the sound wave to go back and fourth from the obstacle
    distance = duration * 0.34 / 2;    //distance from obstacle. Velocity sound = 0.034 [m/ms]

    //Stop if obstacle
    if((distance <= minDistance)){
        stopRobot();
    }
  }
}
```

TRITA TRITA-ITM-EX 2021:32