DEGREE PROJECT IN MECHANICAL ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Gobi

Automatic sand-spreading robot

TIM NASER

STAVROS NTOUVAS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Gobi

Automated sand-spreading robot

TIM NASER & STAVROS NTOUVAS

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

TRITA-ITM-EX 2021:10

Abstract

The purpose of this report was to research through the construction of a prototype the technical challenges associated with creating a robot that distributes sand on patios after snowfall. A robot that could complete this task should be able to know its position in an unknown terrain and traverse it in a predictable manner that allows for the even spread of the sand. In order to achieve stability and predictability of movement, stepper motor driven wheels were chosen as the steering method. The sand-spreading mechanism consists of a DC Motor connected to a 3D-printed disc with rectangular extrusions at its base. The wheels and chassis of the robot were likewise 3D-printed. Lastly, an Arduino MEGA board was the controller of choice.

Keywords: Mechatronics, stepper motors, servo, DC, snow, sand, spread

Referat

Automatiserad sandspridare för snö

Syftet med denna rapport var att genom konstruktionen av en prototyp undersöka de tekniska utmaningarna för att skapa en robot som distribuerar sand på uteplatser efter snöfall. En robot som kan slutföra denna uppgift bör kunna känna till sin position i en okänd terräng och färdas på den på ett förutsägbart sätt som möjliggör en jämn spridning av sanden.För att uppnå stabilitet och förutsägbarhet för rörelse valdes stegmotordrivna hjul som styrmetod. Sandspridningsmekanismen består av en likströmsmotor ansluten till en 3D-utskriven skiva med rektangulära extruderingar vid basen. Robotens hjul och chassi var också 3D-utskrivna. Slutligen var ett Arduino MEGA-kort den valfria styrenheten.

Nyckelord: Mekatronik, stegmotorer, servomotorer, DC-motorer, snö, sand, sprida

Acknowledgements

We would like to thank Nihad Subasic for his interesting and informative lectures. We would also like to thank Staffan Qvarnström for his counsel and commitment to assist all project groups in procuring their desired components on time. Last but not least, we would like to thank the other groups for inspiring us and motivating us to do better through their excellence.

Contents

List of Figures

List of Tables

List of Abbreviations

AC - Alternating Current
CPU - Central Processing Unit
DC - Direct Current
IDE - Integrated Development Environment
RAM - Random Access Memory
RCA - Route Calculation Algorithm
ROM - Read-Only Memory
rpm - Rounds Per Minute
TTA - Terrain Traveral Algorithm

Chapter 1

Introduction

This thesis studies the engineering challenges involved in designing and constructing autonomous solutions for spreading sand on surfaces covered with snow. In this document, the construction of the robot and the design of the software are described in detail. This chapter will explain the background and purpose of the thesis as well as specify the scope and method used for investigating this problem.

1.1 Background

In several Nordic countries, snowfall is a common occurrence during the winter. When the temperature reaches positive degrees and then back to negative again, snow turns into ice, which can cause residents to slip and possibly injure themselves. In the public sector, industrial grade equipment is used to spread salt or sand on the driveways in order to increase traction, which prevents automobile accidents. The corresponding solutions in the consumer market involve a great deal of personal effort since home owners need to spread the sand themselves. This project explores the possibility of creating an automated solution for private use.

1.2 Purpose

The main purpose of this thesis is to construct a robot that can spread sand evenly over a specific area with simple user commands. The following set of research questions need to be answered in order for the purpose to be realised:

- The robot does not use sensors of any kind in order to localize itself. Does that lead to deviations from its planned course?
- What rate of sand flow is achieved through the use of different nozzle sizes?
- How does the rotational velocity of the spinner affect the size of the area that is sanded?

1.3 Scope

The project focuses on constructing a prototype that can move across flat terrain. Focus is placed upon regulating the flow of the sand being spread and the robot speed so as to achieve an even distribution across the designated area. As a result of budget constraints, no analysis has been performed regarding the suitability of different tracks, resistance to cold weather or uneven terrain performance. Due to the limited torque that could be generated by the current motors, the sand is simulated by spreading sugar instead, since using actual sand would not have made it impossible for the robot to be mobile.

1.4 Method

The execution of this project can be divided into two main areas, hardware and software. Work on both areas took place concurrently with an initial emphasis on designing the main software and circuitry. Afterward, the non-moving parts of the robot where 3D-printed and assembled. This was an iterative process, whose main goal was to achieve robustness while removing unnecessary weight.

Chapter 2

Theory

This chapter describes the necessary theory for this project.

2.1 Micro-controller

The Micro-controller is a type of programmable computer that contains at least one Central Processing Unit (CPU), memory (ROM/RAM), and methods for users to generate desired outputs from inputs [1]. The inputs are received by the micro-controller in the form of electrical signals and the outputs can take many forms, depending on the peripherals that are connected to the micro-controller. The Arduino is an open-source micro-controller that was created for educational purposes by the Interaction Design Institute, Ivrea (IDII), in Italy [2].

2.2 H-bridge

An H-bridge consists of 4 transistors that allow change of direction of currents in a circuit. H-bridges can be used to manipulate the rotation of DC motors and bipolar stepper motors [3]. As shown in Figure 2.1, the sequence of transistors that are on respectively off will decide the path of the electrical current through the motor.

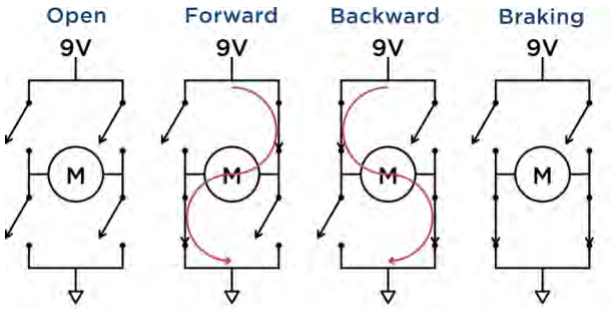


Figure 2.1: The figure shows a schematic representation of an H-bridge [4]

2.3 DC Motor

The Direct Current Motor (DC Motor) consists of an electric conductor shaped in the form of a loop that is enclosed by magnets. The magnets create a magnetic field, which in turn makes the loop rotate around its axis when a voltage source (e.g., a battery) is connected to it [5]. Even though the alternative choice of an alternating current (AC) motor is cheaper, the reason for using a DC motor is the superior ability to adjust the rotation speed without losing efficiency [6].

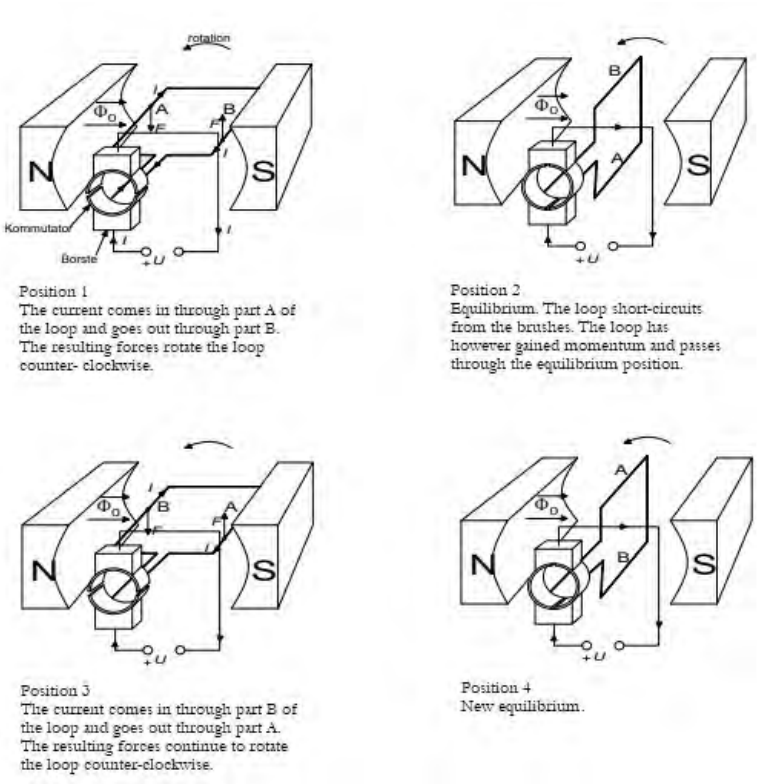


Figure 2.2: The mechanism of a DC Motor is described in four steps [5]

2.4 Stepper motor

Like the DC motor, the stepper motor also uses DC currents. The stepper motor has coils that act as magnets when infused by the DC current (see Figure 2.3). This way the rotor can be driven by activating the next pair of magnets [7]. While less efficient than the DC motor, the stepper motor excels at control and offers a more exact positioning and greater holding torque. Stepper motors allow tracking of positioning by counting the steps. Stepper motors come in two types, unipolar and bipolar [8]. The former offer higher rotational speeds and do not require the use of an H-bridge whereas the latter compensate with higher torque [9].

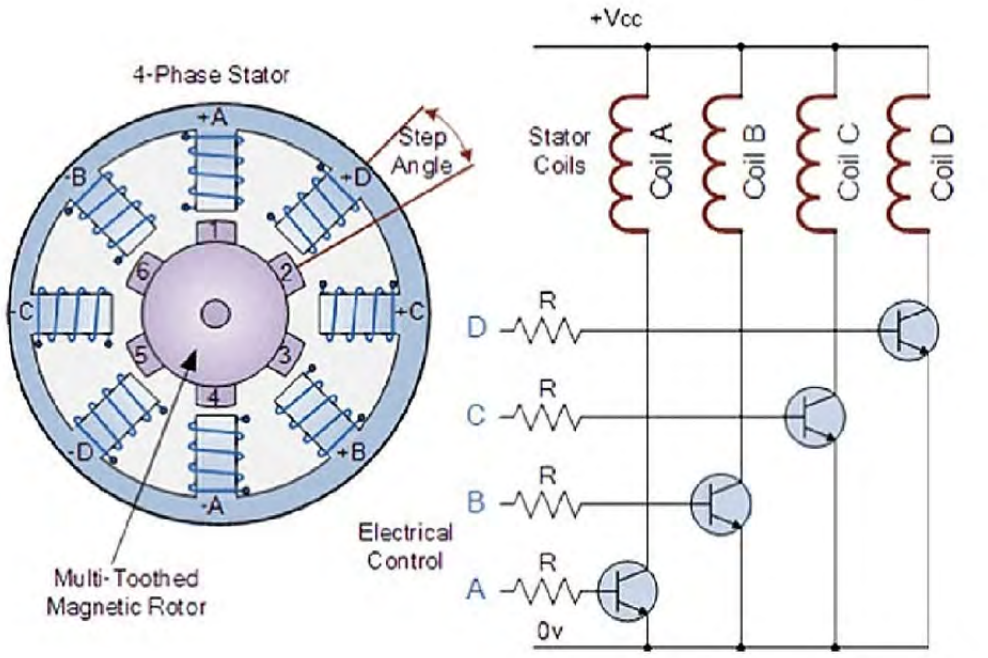


Figure 2.3: An example of a Stepper motor [10]

2.5 Servo motor

The servo motor is a type of motor most often utilised in applications where angle control, speed and peak torque are deemed important [11]. Servo motors used to be driven by direct current (DC). Advancements made during the last 30 years have made servo motors driven by alternating current (AC) the better option for industrial applications with larger loads. However, a type of servo motor potentially used in this type of project is a more economical DC servo motor. Servos of the aforementioned type are driven by DC motors, which in turn are connected to a potentiometer that keeps track of the amount of rotation. There are two basic types of servo motors for rotation, positional servo motors that offer great precision but can only turn 180 degrees and continuous rotation servo motors that can turn indefinitely in both directions, but are not as precise. Torque is generated through gears that are connected to the output shaft.

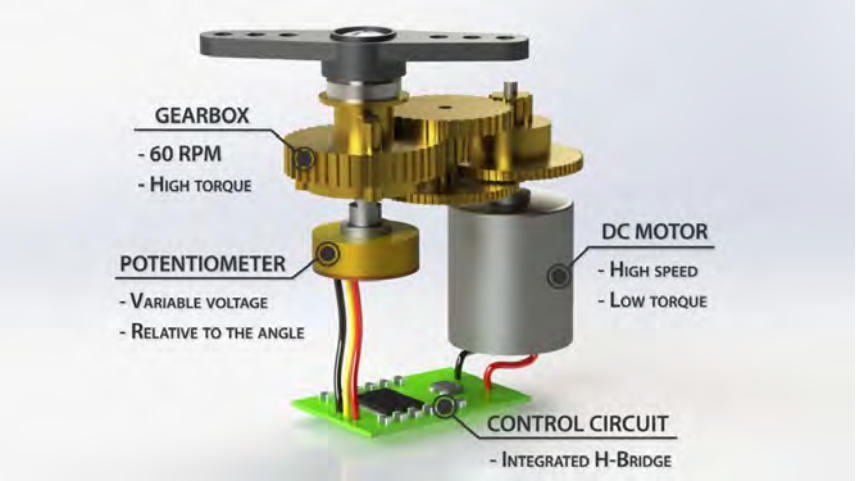


Figure 2.4: Graphical representation of a Servo motor and its components [12]

2.6 Sand Spreading Techniques

A common strategy is to use a funnel with a lid at the bottom drops sand down on a spinner (see figure 2.5) that is connected to a DC motor. The centripetal forces that are caused by the rapid rotational velocity of the spinner result in the sand particles being ejected out onto the ground. The diameter of the circular area that is covered in sand is called in this report as the ”span” of the mechanism, as it is demonstrated in Figure 2.6 .

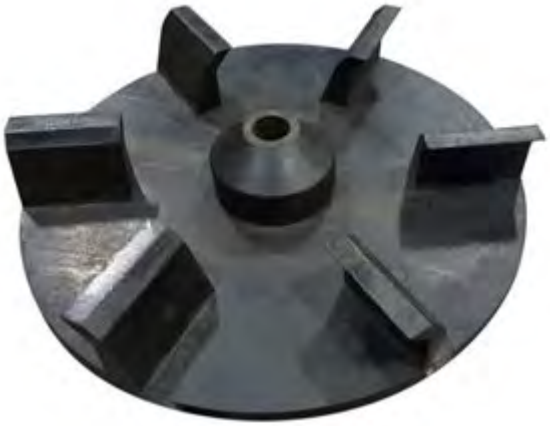


Figure 2.5: Example of a spinner [13]

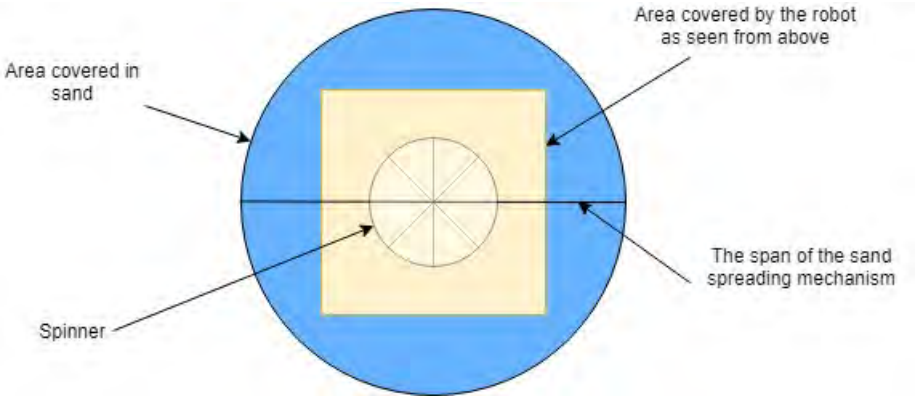


Figure 2.6: A top-down view of the expected function of the spinner.Made in draw.io [14]

Chapter 3

System Design

This chapter describes how the robot was designed in terms of hardware and software.

3.1 Hardware

This section describes the exact components that were used in the project and the way they were assembled.

3.1.1 Micro Controller

The micro controller model is Arduino MEGA 2560 Rev3 and it is mounted on the 3D printed main body of the construction. The Arduino MEGA is responsible for allocating current to the mechanical components according to the code instructions.

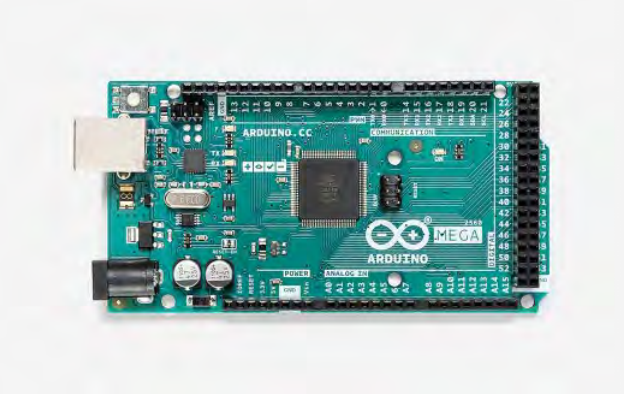


Figure 3.1: The Arduino Mega 2560 Rev3 [15]

3.1.2 Terrain Traversal

The robot traverses the terrain through the use of unipolar stepper motors with 3D printed wheels. The wheels are mounted on the shafts of the stepper motors, which are in turn mounted on the main body of the robot, two on each side. The stepper motors were chosen because they allow for creating specific instructions regarding the terrain traversal by counting the steps. The models in use are 28BYJ-48 , a very popular and accessible model that can divide each rotation in 2048 steps. Specifications of the motor in use are provided in Appendix C.

Figure 3.2: The 28BYJ-48 unipolar stepper motor [16]

3.1.3 Sand Spreading Mechanism

The sand is deposited inside a 3D printed funnel mounted on top of Gobi. At this point the lid at the lower end of the funnel is closed. Once the program starts, and Gobi starts its route, the DC motor that is friction fit inside the chassis turns the 3D printed spinner and the lid that blocks the sugar is opened. Sand falls inside the rotating spinner and is subsequently spread. When the robot takes a turn, the lid is temporarily closed again so that excess sugar at the turning points is minimized. The funnel is attached at its end to an attachable 3D printed nozzle in order to offer greater modularity to the design but also for testing purposes. The funnel's height is 160 mm and the diameter at its widest part is 180 mm. The components can be seen in Figures 3.3 and 3.4.

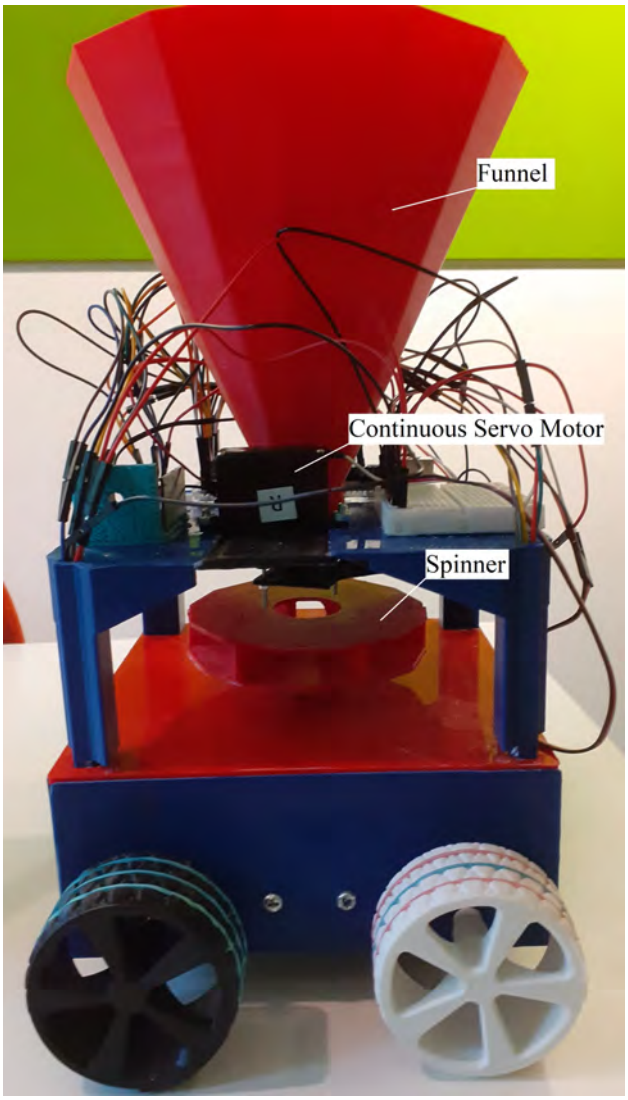


Figure 3.3: Side view of Gobi. Image taken by the authors

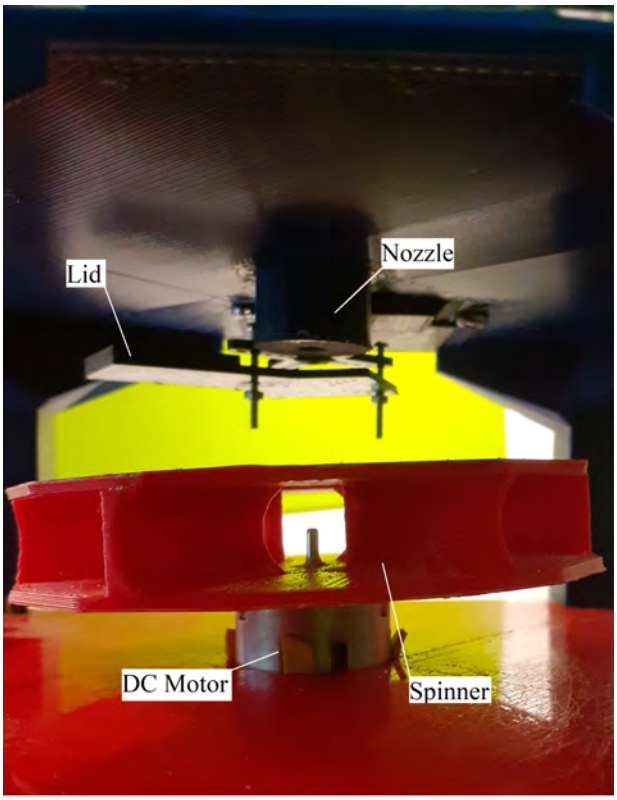


Figure 3.4: The sand spreading mechanism. Image taken by the authors

3.2 Software

The algorithms were developed in the Arduino integrated development environment (Arduino IDE), an open source software provided by the Interaction Design Institute, the developers of Arduino.

The user enters the lengths of the sides of the rectangular area that he/she wants to cover . Gobi is expected to cover that area in a rectangular spiral. A necessary condition is that the algorithm should be applicable for rectangular areas of arbitrary sizes.

3.2.1 Route Calculation

As shown in Figure 3.5, Gobi starts at the bottom left corner of the rectangular area and finishes at its centre. Gobi's motion can be described as a collection of straight forward motion that is interspersed with 90 degree turns. The first three straight distances are equal to the size of the respective dimensions that were entered by the user. From the fourth distance and for every consecutive distance that corresponds to an even number, the distance is reduced by a set amount. The amount depends

on the preferences of the user and the span of the robot (see chapter 2, section 2.6). A smaller number results in a more dense sand spreading pattern, which increases the amount of sand per area unit but limits the total area that can be covered.

Mathematically, this process can be described in the following manner: Let $\mathbf{A}$ denote a vector of real numbers whose every element is equal each straight distance needs to be covered

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \tag{3.1}$$

Let also x and y denote the horizontal and vertical dimensions of the area to be covered by sand (see Figure 3.5). Lastly, let s denote the span of the sand spreading mechanism. Then the first three elements of the array should be:

$$a_1 = y \tag{3.2}$$
$$a_2 = x \tag{3.3}$$
$$a_3 = y \tag{3.4}$$

The consecutive elements of the vector will be inserted in the following sequence:

$$\begin{aligned} a_4 &= x - 1 * s \\ a_5 &= y - 1 * s \\ a_6 &= x - 2 * s \\ a_7 &= y - 2 * s \\ &\vdots \end{aligned}$$
$$a_{i,even} = x - (\frac{i}{2} - 1) * s \tag{3.5}$$
$$a_{i+1} = x - (\frac{i}{2} - 1) * s \tag{3.6}$$

The series that is displayed in Equations 3.5 and 3.6 starts to come into effect after the third iteration of the while loop referenced in Figure 3.6.

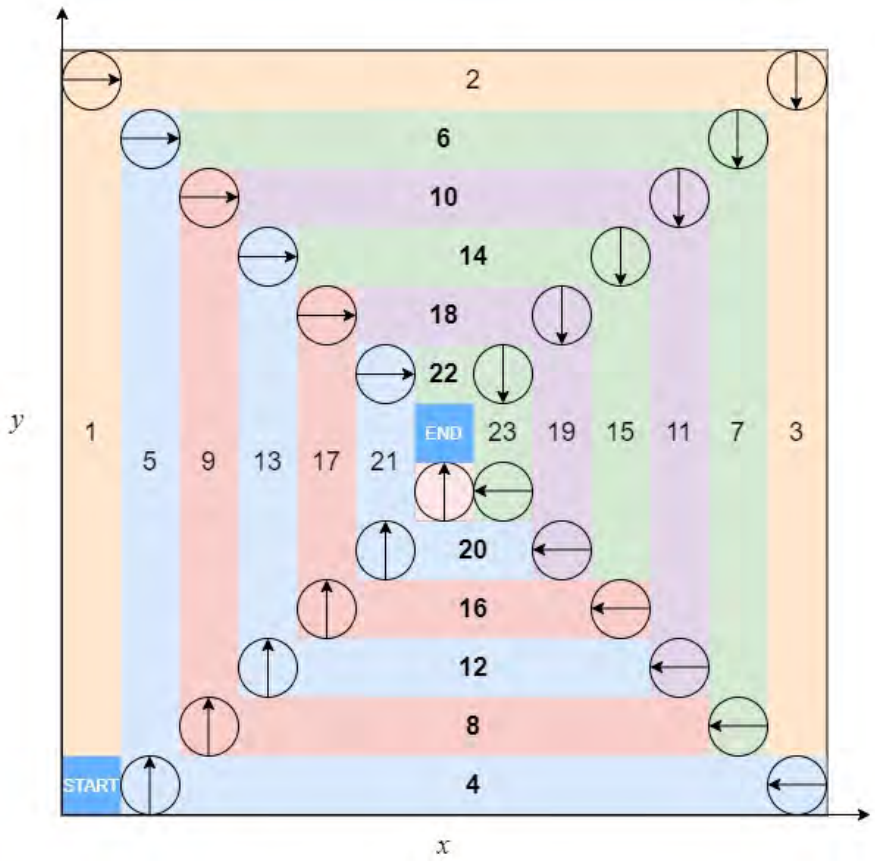


Figure 3.5: Gobi's path inside an arbitrary rectangular area. Each circle with an arrow denotes a turning point. All straight distances are marked with a number. Those with numbers in bold text denote the places where a distance reduction occurs, according to equations 3.5 and 3.6. Made in draw.io [14]

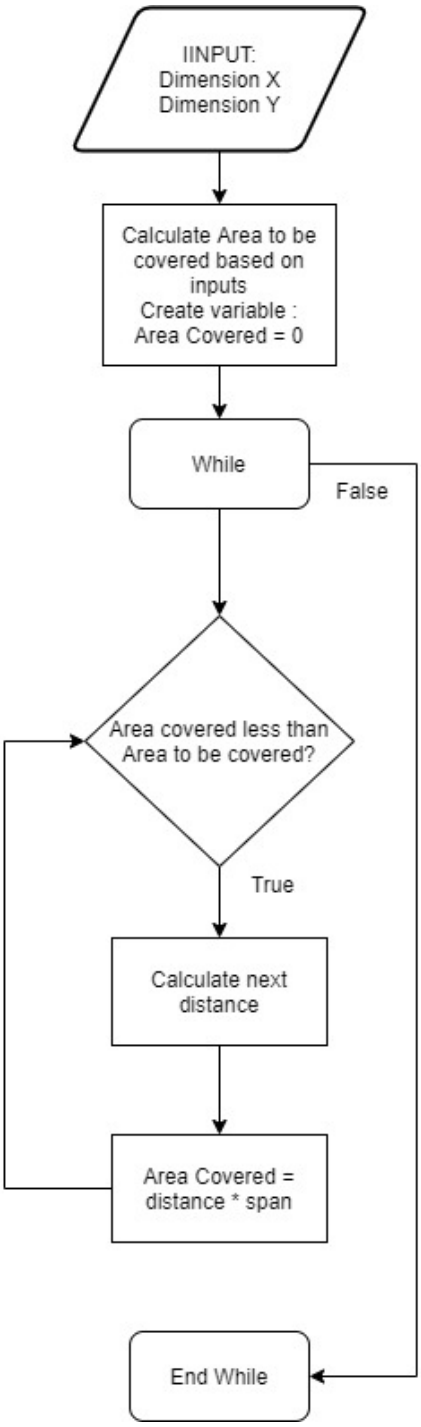


Figure 3.6: Flowchart of the code structure for calculating Gobi's route. The condition for exiting the loop is the Area covered equaling or exceeding in size the Area to cover. Made in draw.io[14]

3.2.2 Stepper Motor Operation

The stepper motor instructions were coded using the *AccelStepper* library [17]. The methods within this library are constructed in a way that receives number of steps as input. This means that the stepper motors can be instructed to rotate a predetermined amount of steps. The stepper motors used in this project have 2048 steps for one whole rotation (see Appendix C). Due to this fact, a function for converting distances to steps was created. Let d denote a distance in meters, r denote the radius of the wheels used, and $\mathrm{N_{rot}}$ the number of full rotations:

$$N_{rot} = \frac{d}{2\pi r} \tag{3.7}$$

$$Steps = N_{rot} * 2048 \tag{3.8}$$

$$Steps(d) = \frac{2048d}{2\pi r} \tag{3.9}$$

Using this conversion, the outputs from the Route Calculation Algorithm could be converted into inputs that are compatible with the *AccelStepper* library.

3.2.3 Main Code

The main code of the software receives the distances that were calculated by the Route Calculation Algorithm and creates instructions to the various components of Gobi. This process is illustrated in Figure 3.7.

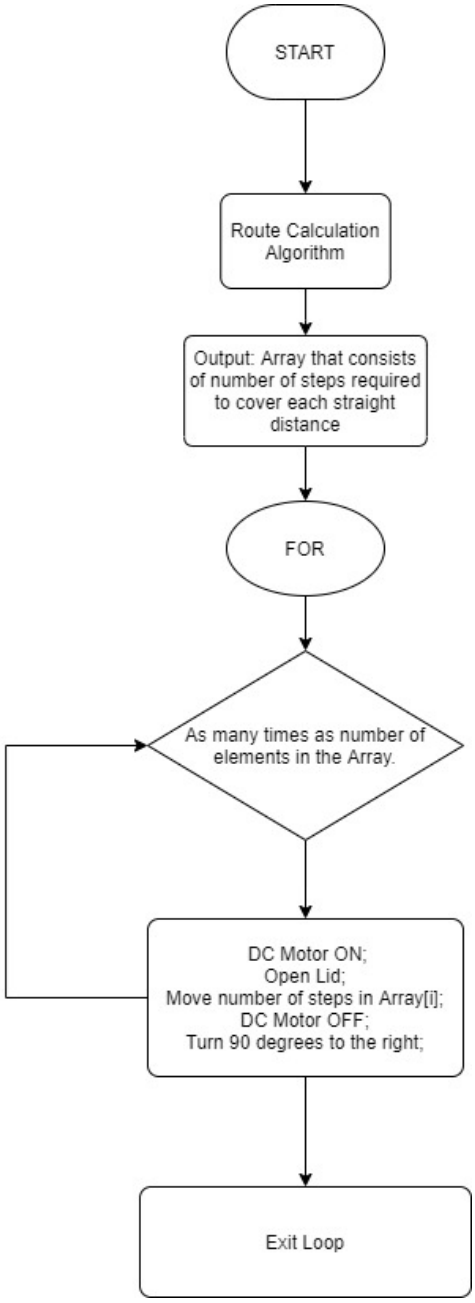


Figure 3.7: Flowchart of the main code structure, Made in draw.io[14]

Chapter 4

Experiments and Results

This chapter describes the experiments that were performed in order to test the research questions. All tests were performed with sugar as a substitute to sand due to the high density of sand making it impossible to use the robot with the current stepper equipment.

4.1 Experiments For Deviation From Course

The purpose of these experiments was to examine the first research question by determining how accurately the resulting movement of Gobi follows the software instructions. There were two types experiments that were used to test this question. First, Gobi was programmed to move in straight lines with the goal of determining whether or not there was any deviation during the forward motion part of the operation. The test was performed in the way shown in Figure 4.1. Afterwards, Gobi was operated according to its intended use, which is to cover rectangular areas as described in chapter 3. After each time that Gobi was operated, the course deviation was estimated by measuring the distance between the final stopping position of Gobi and the intended position, which is always the centre of the rectangular area. The second experiment is represented visually in Figure 4.2 .

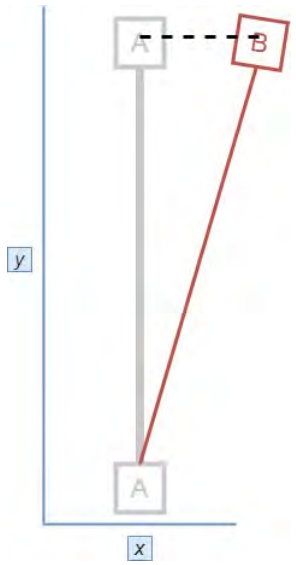


Figure 4.1: The deviation from course in a straight line experiment. Gobi starts at point A and ends its course at point B. The grey line depicts the programmed course of Gobi whereas the red line depicts course Gobi took during operation. The black dashed line represents the distance that was measured in the experiment. Made in draw.io [14]

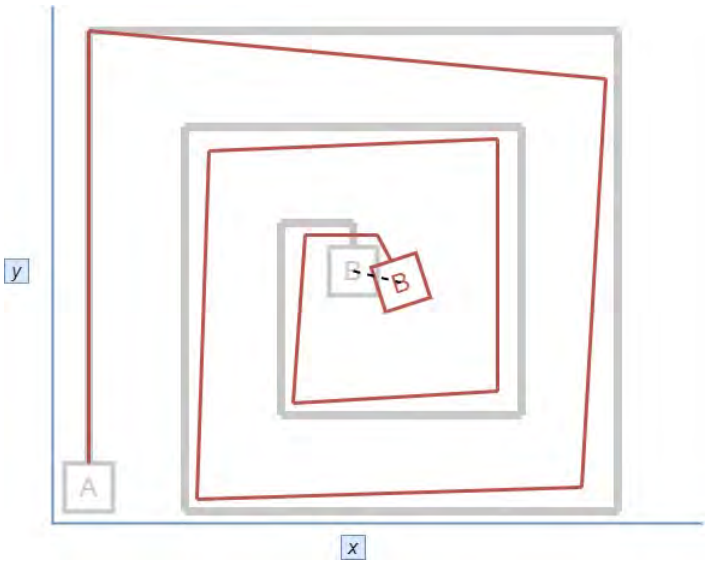


Figure 4.2: The deviation from course experiment. Gobi starts at point A and ends its course at point B. The grey line depicts the programmed course of Gobi whereas the red line depicts course Gobi took during operation. The black dashed line represents the distance that was measured in the experiment. Made in draw.io [14]

4.1.1 Experiments For Deviation From Course Results

The first version of the experiment, which intended to measure deviations from courses in straight lines resulted in no measurable deviations. The second version resulted in measurable deviations from the course, which are shown in Table 4.3.

Number of Turns	Average Deviation [cm]
5	10
8	18
11	35
14	45

Table 4.1: The values shown are averages after five tries

4.2 Testing of Flow for Various Nozzle Sizes

This experiment was conducted with aim of answering the second research question that was related to the effect that nozzle sizes have on the sand flow. The size of area that can be covered greatly depends on the amount of sand distributed per unit of time. Each nozzle size was tested five times. Upon activation, the robot was set in motion with the same speed of approximately 0,05 m/s. The mass of the sugar held inside the funnel was 100 g.

4.2.1 Testing of Sand Flow for Various Nozzle Sizes Results

Nozzle size [mm]	Time under flow [s]	Distance covered by robot [m]
4	no flow	0
5	69	3,79
6	44	2,42
7	26	1,43
8	13	0,71

Table 4.2: Distances covered by the robot while still being able to spread sugar particles

Figure 4.3: The nozzles that were used (4mm to 8 mm diameter from left to right. Picture taken by the authors

4.3 Effect of Spinner's Rotational Velocity

This experiment was performed with the purpose of determining how the rotational speed of the DC Motor affects the size of the area that is being covered in particles. In order to test the range of the sand based on the rotational velocity of the spinner, the robot was placed on a black surface and the DC motor was tested five times for each chosen rpm speed (see Figure 4.4). Afterwards, the average measured value was calculated.

4.3.1 Effect of Spinner's Rotational Velocity Results

Rotational Speed [rpm]	Average Distance [cm]
1900	10
2500	18
2800	35
3300	45

Table 4.3: Average distance that the majority of the particles traveled for a given rotational speed

Figure 4.4: Testing of spread of the particles as a function of the rotational velocity of the DC motors. Picture taken by the authors

Chapter 5

Discussion

The robot met all expectations and demands of a prototype for testing the research questions. The stepper motors moved according to our forward moving algorithm introduced in Section 3.2. The stepper motors also worked in sequence of first moving forward then rotating as mentioned in Section 3.2, as desired. The movement sequence resulted in the robot moving as explained in Figure 3.5 with minor deviations occurring every turning sequence.

The most difficult part to measure was the particle spread relative to the rotors rotational velocity, discussed in section 4.3. The main issue with measuring was the particles small size and the colour. As seen in Figure 4.4 black plastic bags were a viable solution for spotting and measuring the particle spread. The different nozzle sizes tested, as seen in Figure 4.3, were really great to find both the minimum and maximum limit of plausible nozzle size to flow speed ratio. The 6 mm and 7mm would also be enough to handle the small testing distance however, the 5mm nozzle had a good enough flow to provide enough friction and simultaneously spread in bigger areas if desired. Using the 8mm nozzle would result in the robot either running out of particles too soon or be filled with more particles than the stepper motors can carry in weight.

The main issue with the robot were the stepper motors. The weight of the robot was too high for them to maintain an adequate speed. As a result, the robot could not cover areas larger than approximately 1,3 m^2. Another issue was that the 3D printed chassis were optimised for having minimal material in order to minimize the weight of the robot. As a result, the chassis were susceptible to bending motions that cumulatively led to deviations in the course. The lack of sensors to provide feedback about the environment and subsequently correct the path taken by the robot exacerbated this problem.

Chapter 6

Conclusion

The prototype worked well enough to attain answers for all of the research questions that needed to be answered.

- The robot does not use sensors of any kind in order to localize itself. Does that lead to deviations from its planned course?

When Gobi was instructed to move in straight lines, there were no measurable deviations. However, there were some deviations in the path Gobi followed when it turned several times. The light construction of the robot in combination with possible imperfections in the terrain as well as the fact that the stepper motors were not operated simultaneously but instead were used in very brief intervals separately may have contributed to that fact.

- What rate of sand flow is achieved through the use of different nozzle sizes?

The sand spreading mechanism performed as expected, however due to the medium not being actual sand, but being sugar instead, the results may not be accurate. What was unexpected was that even though the sugar particles were very small, the 4mm nozzle gave no flow at all. To cover a long enough driving distance of at least 5 meters with minimum amount of weight the 5mm nozzle was used.

- How does the rotational velocity of the spinner affect the size of the area that is sanded?

Spreading the particles in a distance beyond two times the robot's length would be superfluous, therefore driving the rotor at 2800 rpm is considered the optimal choice for a spreading mechanism of this type.

Despite the issues that occurred during testing, Gobi managed to complete its meant task. The task being completing the movement according to the calculated movement algorithm while also spreading enough particles to cover the designated area with enough sand to provide friction.

Chapter 7

Future Work

The areas for improvement are many, both in terms of hardware and in terms of software. It became clear during the experiments that the torque output of the stepper motors in use was the main limiting factor in terms of the amount of sugar that could be transported. More powerful motors would enable the use of actual sand, which would make the experiments more accurate. Furthermore, a construction made of more robust materials would have made outdoor testing possible. Moreover, the lack of an obstacle avoidance algorithm greatly reduces the autonomy of the robot. An implementation of LiDAR as a sensor for self localization would have been a great improvement in that regard [18]. Additionally, a controller for regulating the sand flow depending on the terrain would also serve to make the robot more versatile [19]. Finally, the robot in its current form lacks the ability for simple user input, the intended area is declared inside the script itself. A user input device such as a small numerical pad in conjunction with a serial monitor would be a significant improvement.

Bibliography

[1] J. H. Davies, "Chapter 1 - embedded electronic systems and microcontrollers," in *MSP430 Microcontroller Basics*, J. H. Davies, Ed. Burlington: Newnes, 2008, pp. 1–20. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978075068276350002X

[2] (accessed: 15.02.2021) What is arduino? [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[3] B. Bellini, A. Arnaud, S. Rezk, and M. Chiossi, "An integrated h-bridge circuit in a hv technology," in *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, 2016, pp. 331–334. [Online]. Available: https://doi.org/10.1109/LASCAS.2016.7451077

[4] F. Bär. (15.02.2021) Building an arduino bluetooth robot car–part 2: Controlling dc motors. [Online]. Available: https://filderbaer.wordpress.com/2014/10/15/building-an-arduino-bluetooth-robot-car-part-2-controlling-dc-motors/

[5] H. Johansson, *Elektroteknik*. Stockholm, Sweden: KTH, Institutionen för Maskinkonstruktion Mekatronik, 2013.

[6] J. Bergman and J. Lind, "Robot vacuum cleaner," KTH, Stockholm, Sweden, DEGREE PROJECT IN ELECTRONICS AND COMPUTER, ENGINEERING, FIRST CYCLE, 2019. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1373823&dswid=-7420

[7] A. Antonova and H. Lundin, "Photobot: An exploring robot," KTH, Stockholm, Sweden, DEGREE PROJECT IN TECHNOLOGY, FIRST CYCLE, 2019. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1373575&dswid=-7420

[8] Z. Benhua, L. Chenghua, S. Shiming, and G. Lu, "Design on a unipolar and unidirectional stepper motor circuit," in *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, vol. 4, 2011, pp. 1795–1797. [Online]. Available: https://doi.org/10.1109/EMEIT.2011.6023452

[9] C. Cavallo. (accessed: 15.02.2021) Stepper motors vs. dc motors - what's the difference? [Online]. Available: https://www.thomasnet.com/articles/machinery-tools-supplies/stepper-motors-vs-dc-motors/

[10] B. Porter. (accessed:15.02.2021) What's a stepper motor driver why do i need it? [Online]. Available: https://all3dp.com/2/what-s-a-stepper-motor-driver-why-do-i-need-it/

[11] R. Krishnan, "Selection criteria for servo motor drives," *IEEE Transactions on Industry Applications*, no. 2, pp. 270–275, 1987. [Online]. Available: https://doi.org/10.1109/TIA.1987.4504902

[12] D. Nedelkovski. (accessed: 15.02.2021) How servo motor works and how to control servos using arduino. [Online]. Available: https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

[13] (accessed:15.02.2021) Elite pickup truck insert spreader. Meyer Products. [Online]. Available: https://www.meyerproducts.com/salt-spreaders/insert-hopper/elite-pickup-truck-insert-spreader

[14] S. GmbH. (accessed:20.05.21). [Online]. Available: https://drawio-app.com/product/

[15] Arduino. (2021) Arduino mega 2560 rev3. [Online]. (accessed:21.05.21) Available: https: //store.arduino.cc/arduino-mega-2560-rev3

[16] (accessed:21.05.21) 28byj-48 datasheet. MikroElektronika. [Online]. Available: https://www.digikey.se/sv/datasheets/mikroelektronika/mikroelektronika-step-motor-5v-28byj48-datasheet

[17] Mike McCauley, "Accelstepper (version 1.61, date accessed 2020-05-20)." [Online]. Available: https://www.airspayce.com/mikem/arduino/AccelStepper/

[18] NOAA. (accessed: 30.04.21) What is lidar? [Online]. Available: https://oceanservice.noaa.gov/facts/lidar.html

[19] T. Glad and L. Ljung, *Reglerteknik - Grundläggande teori*. ISBN: 978-91-44-02275-8.: Studentlitteratur, 2020.

Appendix A

Arduino Code

```
 /*
* University: Royal Institute of Technology, KTH
* Course : MF133XVT21, Degree Project in Mechatronics.
* TRITA number : ITMEX 2021:10
* Authors : Stavros Ntouvas & Tim Naser
* Program: CMAST
* Project: Gobi
*Finalised: 2021-05-07
*
*This script operates the robot in the wa described in the report
*/

//************************** Preamble ***************************

// Libraries
#include <AccelStepper.h>
#include <NewPing.h>
#include <Servo.h>

//*********************** Components Setup ******************

//---------------------------------- Stepper Motor Pins

// Define Motor Pins (Motor 1)

#define motor1Pin1  22
#define motor1Pin2  23
#define motor1Pin3  24
#define motor1Pin4  25
```

```
// Define Motor Pins (Motor 2)

#define motor2Pin1  26
#define motor2Pin2  27
#define motor2Pin3  28
#define motor2Pin4  29

// Define Motor Pins (Motor 3)

#define motor3Pin1  46
#define motor3Pin2  47
#define motor3Pin3  48
#define motor3Pin4  49

// Define Motor Pins (Motor 4)

#define motor4Pin1  50
#define motor4Pin2  51
#define motor4Pin3  52
#define motor4Pin4  53

// Define step for stepper motors
#define Step 4
//---------------------------------- Ultrasonic Sensor Pins

#define triggerPin 10
#define echoPin 10

//---------------------------------- DC MotorPins
int motorPin1 = 6;
int motorPin2 = 7;


//---------------------------------- Ultrasonic Sensor Constants
#define maxDistance 450   // 450 cm according to specifications
#define minDistance 2     // cm



//********************************** Initializations ******************************
// The sequence 1-3-2-4 is required for proper sequencing of 28BYJ48
AccelStepper stepper1(Step, motor1Pin1, motor1Pin3, motor1Pin2, motor1Pin4);
// Creates Stepper object
AccelStepper stepper2(Step, motor2Pin1, motor2Pin3, motor2Pin2, motor2Pin4);
```

```
AccelStepper stepper3(Step, motor3Pin1, motor3Pin3, motor3Pin2, motor3Pin4);
AccelStepper stepper4(Step, motor4Pin1, motor4Pin3, motor4Pin2, motor4Pin4);
Servo servo; // Creates servo object
NewPing sonar(triggerPin,echoPin,maxDistance);

//************************ Constants and Variables *********************
#define pi acos(-1.0) // accurate approximation of pi



//----------------------------------- Robot Dimensions

double r=32.5e-3; // Radius of the wheel
double per=pi*r*2; //circumference of wheel
double b = 0.340; // two times the side of the robot in m

//----------------------------------- Area to be covered
double x = 1; // distance in dim x in meters
double y = 1; // distance in dim y in meters
double AreaToCover = x*y; // Area that will have to be covered in sand
int Array[50]; // Array with prealocated number of elements


double AreaCovered = 0;
double n =1; // number of iterations


//double dy = y -n*b;

double dist=1;
double nRot=dist/per;
int stepsRequired=round(nRot*2048);

//----------------------------------- Iteration Integers
int j = 0; // Used to create the path Array
int k = 0; // Used to create servo motor arary

/*
 * Calculates span of angle that the Ultrasonic Sensor is
 * supposed to scan for based on what
 * distance between the robot and the obstacle is considered near enough
 */
int servoDelay = 25;
float distAcceptable = 0.200; // Distance which we deem to be too close
```

```
float alpha =180/pi*asin(0.5*b/distAcceptable);  // Output in degrees
int gamma = round( 90 - alpha); // Convert to output that is compatible with
                                // the servo library
int angleServoLeft = 180 - gamma; // Left limit of the servo motor
int angleServoRight = gamma; // Right limit of the sevo motor
int lengthof = angleServoLeft - angleServoRight;
int servoArray[360];
//int i = angleServoRight;



void Servo_open(){
  servo.write(120); // Continuous servo motor moves at a slow speed
  delay(550);
  servo.write(93); // contin. servo motor stops, according to theory it should
                   // be servo.write(90) but it
                   // is different for different motors
}

void Servo_close(){
  servo.write(66); // Continuous servo motor moves at a slow speed
  delay(550);
  servo.write(93); // contin. servo motor stops, according to theory it should
                   // be servo.write(90) but it
                   // is different for different motors
}

/*
 *  Turn( int angle) function.
 *  Input: Degrees that the robot should rotate, positive numbers are right,
 *  negative numbers are left
 *  Output: Rotation that corresponds to input
 */
void turn(int angle){
  if (angle>=0){
      double rad = angle * pi/180;
      double sector = rad * b/2;
      double Rotations = sector/per;
      int stepsTurn= round(2048*Rotations);
      movement_A(stepsTurn,stepsTurn,-stepsTurn,-stepsTurn);

  }
```

```
  else {
       double rad = angle * pi/180;
      double sector = rad * b/2;
      double Rotations = sector/per;
      int stepsTurn= round(2048*Rotations);
      movement_A(stepsTurn,stepsTurn,-stepsTurn,-stepsTurn);

  }
}


/*
 *  Movement_A( int Steps for motor 1, .... inte Steps for moto 4)
 *  Input: Amount of steps that each stepper motor is supposed to take,
 *  this function is used in conjuction with array
 *  Output: The stepper motors take the steps that they are instructed to take
 */
void movement_A(int Steps1, int Steps2, int Steps3, int Steps4) {
  int obstDistance = 200;

  stepper1.moveTo(Steps1);
  stepper2.moveTo(Steps2);
  stepper3.moveTo(-Steps3);
  stepper4.moveTo(-Steps4);


  while(stepper1.distanceToGo()!=0){

  stepper1.run();
  stepper2.run();
  stepper3.run();
  stepper4.run();

  }



}

void setup() {

  Serial.begin(9600); // Initialises Serial monitor for debugging purposes
```

```
  //***************** Define how fast the stepper motors **************
  //**************** are expected to rotate and accelerate ***********
  stepper1.setMaxSpeed(550.0);
  stepper1.setAcceleration(150.0);
  stepper1.setSpeed(200);
  stepper2.setMaxSpeed(550.0);
  stepper2.setAcceleration(150.0);
  stepper2.setSpeed(200);
  stepper3.setMaxSpeed(550.0);
  stepper3.setAcceleration(150.0);
  stepper3.setSpeed(200);
  stepper4.setMaxSpeed(550.0);
  stepper4.setAcceleration(150.0);
  stepper4.setSpeed(200);

/*
 * Creates an Array with all intended distances that are intended to be driven
 * in a straight line
 * Each element of the Array is then used as input for movement_A()
 * Not a function itself because it is only calculated once
 */

    byte ArrayIndex = 4;
  Array[0] = round(y*2048/per);
  Array[1] = round(x*2048/per);
  Array[2] = Array[0];
  while(AreaCovered < AreaToCover ){
  double dx = x - (ArrayIndex/2-1)*b;
  int Steps = round(2048*dx/per);
  Array[ArrayIndex]= Steps;
  AreaCovered = AreaCovered + dx*b;
  double dy = y - (ArrayIndex/2-1)*b;
  Steps = round(2048*dy/per);
  Array[ArrayIndex + 1]= Steps;
  AreaCovered = AreaCovered + dy*b;
  ArrayIndex = ArrayIndex +2;

        }
   /*
    * Creates the Array with the span of angles that
    * the servo motor will cover
    */
```

```
   for(int i = 0; i < lengthof; i++){
    servoArray[i] = angleServoRight + i;
    Serial.println(servoArray[i]);

}
  for(int i =lengthof; i<=(2*lengthof); i++){
    servoArray[i] = angleServoLeft - k;
    k++;

    Serial.println(servoArray[i]);

}


   servo.attach(8);// Creates a servo object. Attaches it to pin 8
   servo.write(90); delay(1000);




}

void loop() {
    Servo_open();
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, HIGH);
    movement_A(Array[j],Array[j],Array[j],Array[j]);
    stepper1.setCurrentPosition(0);
    stepper2.setCurrentPosition(0);
    stepper3.setCurrentPosition(0);
    stepper4.setCurrentPosition(0);
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
    Servo_close();
    turn(180);
    stepper1.setCurrentPosition(0);
    stepper2.setCurrentPosition(0);
    stepper3.setCurrentPosition(0);
    stepper4.setCurrentPosition(0);
    Serial.println("turn");
    j++;
```

```
}
```

Appendix B

Acumen Code

```
/* Acumen Assignment as part of Bachelor's Thesis in Mechatronics
 * Authors: Stavros Ntouvas and Tim Naser 2021.
 *  This is a model of the construction, which is a robot that
 * spreads sand evenly over a specific area with simple user commands.
 */

//****** Script initialization and definition of*****
//*******variables that will be used***
model Main(simulator) =
initially
 r = create Robot((0,0,0),blue), // Instance of Robot is created
 x=0, x'=0, x''=0,   // Variables that will allow the motion equations later on
 x2=0, x2'=0, x2''=0,
 z = 0
always
//********* Motion equations **************
 if x<20
  then x'' = -(x'-10)
  else if x'>0
         then x '' = -10
         else x '' = 0,
 r.pos = (x,0,0),
 x2'' = -100*(x2-x)-10*(x2'-x'),
 z = x-x2


//******* Object Definitions***************
model Robot(pos,col) =  // Robot object is defined
initially
 _3D = (),_Plot=()
```

```
always
 _3D = (Box center = pos+(0,0,0)
            color = 0.4*col + 0.6*white
            length = 8
            width = 6
            height = 3
        Cone center = pos + (0,0,4)
             color = 0.4*col + 0.6*red
             radius = 2
             length= 5
             rotation=(-pi/2,0,0)
        Cylinder center = pos + (3,3,-1)        // Front left wheel
                 color = 0.4*col + 0.6*blue
                 radius = 1.5
                 length = 1
        Cylinder center = pos + (3,-3,-1)       // Front right wheel
                 color = 0.4*col + 0.6*blue
                 radius = 1.5
                 length = 1
       Cylinder center = pos + (-3,-3,-1)      // Rear right wheel
                 color = 0.4*col + 0.6*blue
                 radius = 1.5
                 length = 1
       Cylinder center = pos + (-3,3,-1)      // Rear left wheel
                 color = 0.4*col + 0.6*blue
                 radius = 1.5
                 length = 1
        )
```

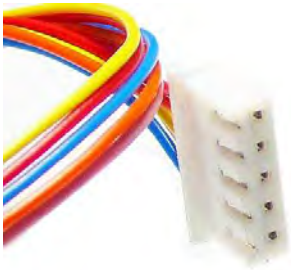
Appendix C

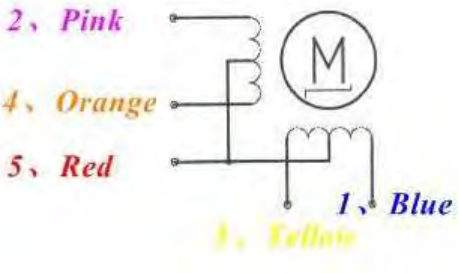
Data Sheet Stepper Motors

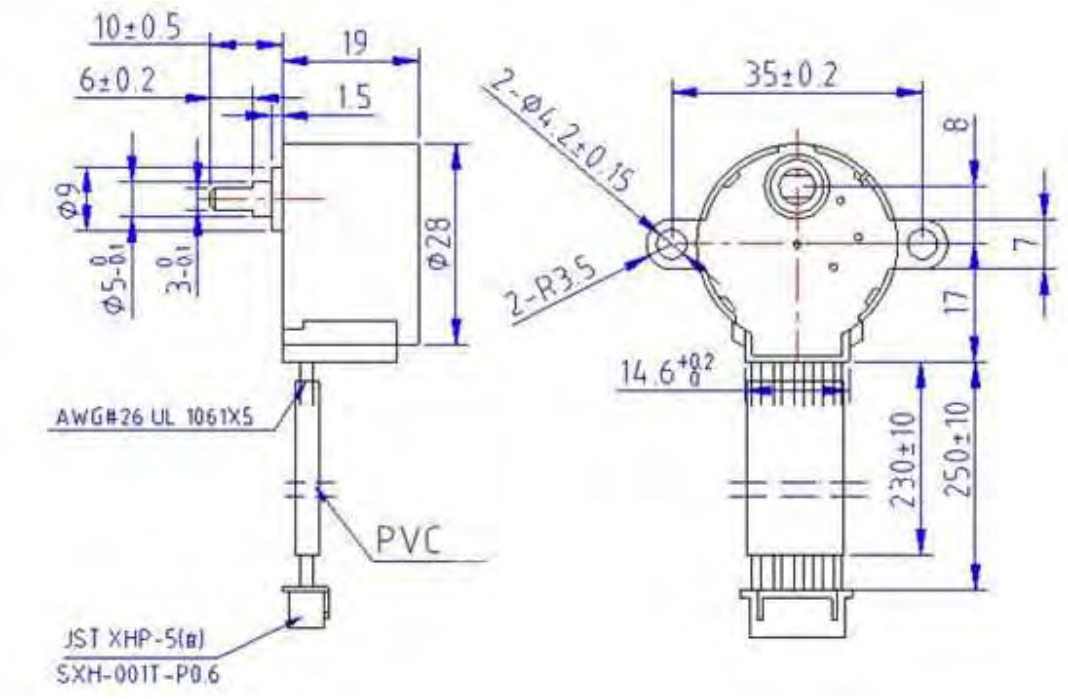
28BYJ-48 – 5V Stepper Motor

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V



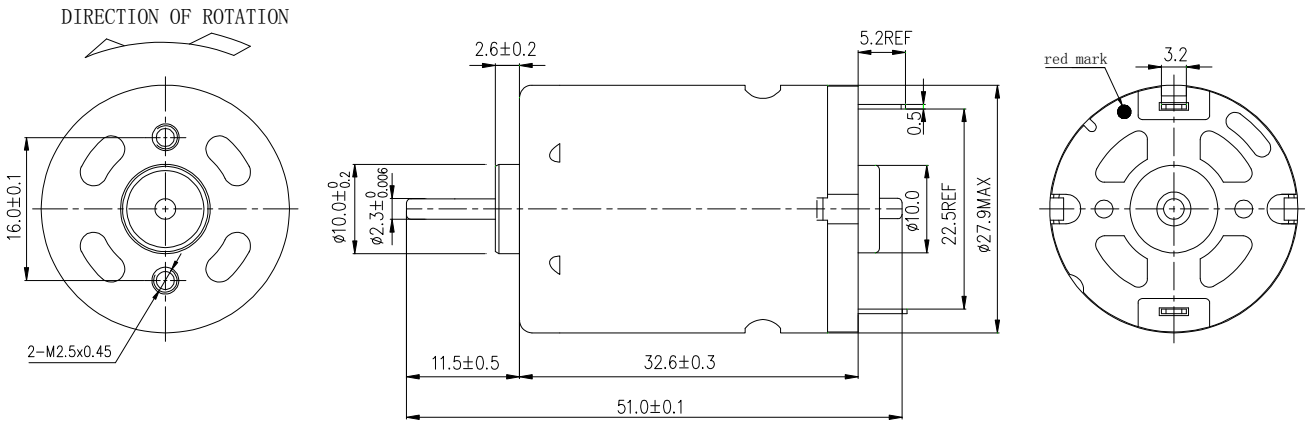




Appendix D

Data Sheet DC Motor

MOT3N

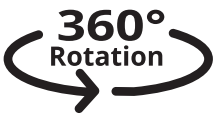

DIRECTION OF ROTATION
16.0±0.1
2-M2.5x0.45
2.6±0.2
5.2REF
0.5
ø10.0±0.2
ø2.3±0.006
ø10.0
22.5REF
ø27.9MAX
11.5±0.5
32.6±0.3
51.0±0.1
red mark
3.2

Appendix E

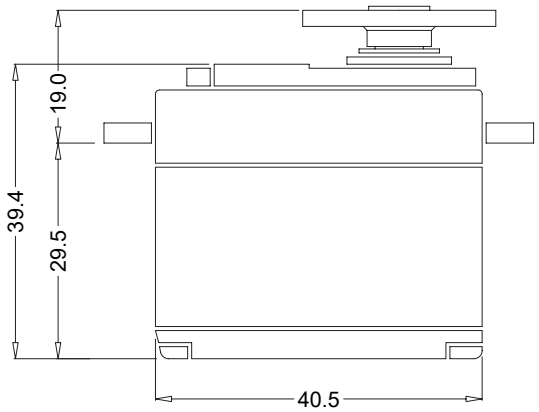
Data Sheet Servo Motor

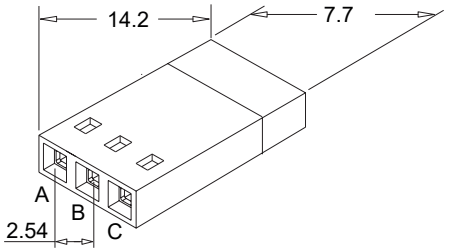
360° Continuous Rotation Servo

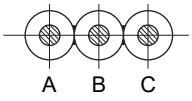
Gear motor for Arduino and Raspberry-Pi robotics projects.
Typical use: Model aircraft, cars and robots.
A continous servo rotates forward or backwards instead of moving to a given position.
Bidirectional rotation - pulse duration determines the speed and direction of rotation.



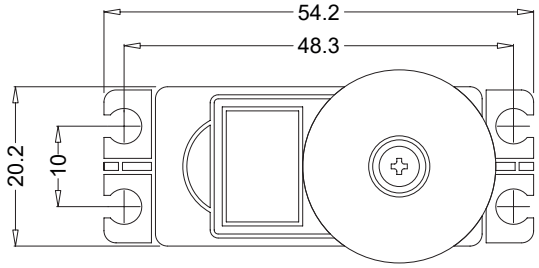
Item no: 90770
Model no: DS04-NFC
Weight: 38g.
Torque: 5,5 kg/cm (54 Ncm) (at 4,8 V).
Speed: 0,22 sec/60° ≈ 45 rpm (at 4,8 V).
Linear response to PWM (0-45 RPM) for easy ramping
Operating voltage: 4,8-6 V.
Operating temperature: -10 to 50 °C.
Current: < 1000 mA.
Cable length: 290 mm.
Connector type: JR / Futaba
Breadboard friendly connector 2,54 mm pitch.
Connector wire gauge: 28 AWG.
Control system: PWM (Pulse Width Modulation)
Counterclockwise rotation: 1-1,5 ms
No rotation (stop): 1,5 ms
Clockwise rotation: 1,5-2 ms
Pulse Frequency / Duty cycle: 50 Hz / 20 ms square wave



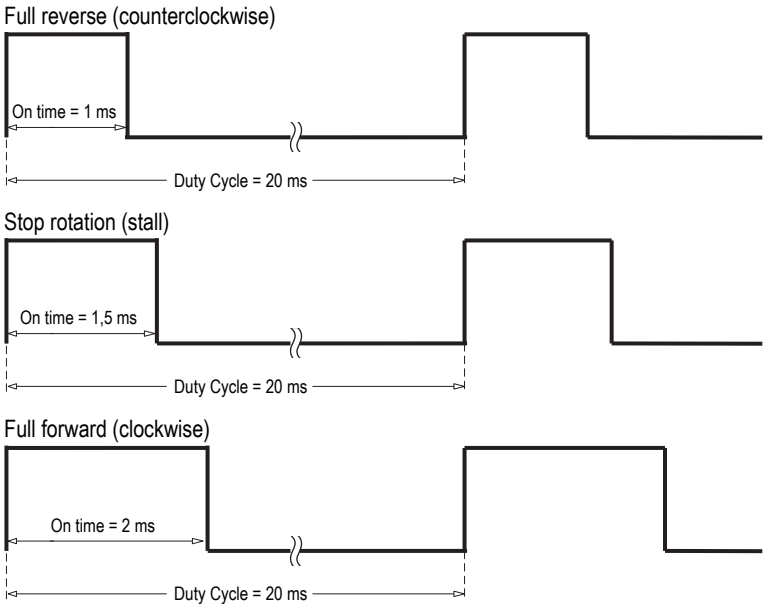




A - Black (-)
B - Red (+)
C - White (signal)



360° servo PWM signal timings



TRITA-ITM-EX 2021:10