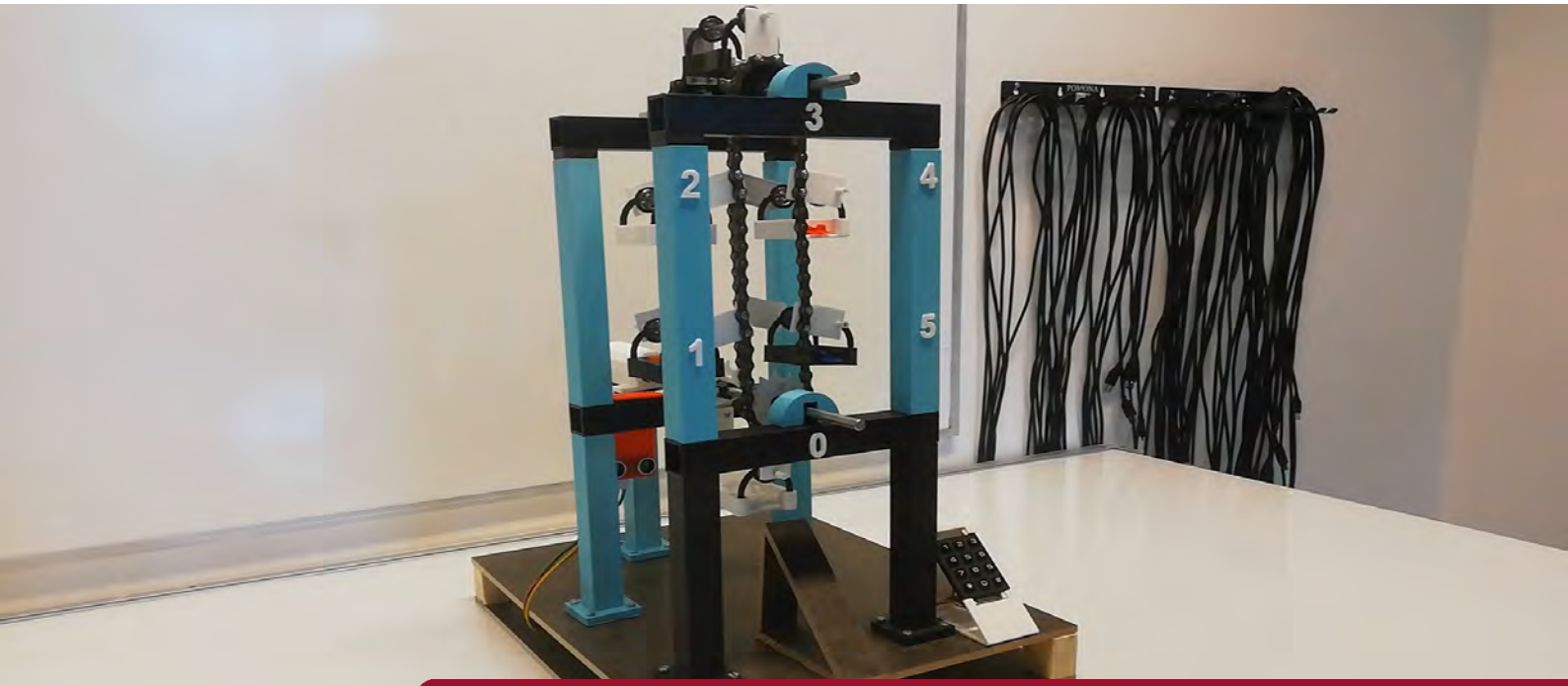

DEGREE PROJECT IN MECHANICAL ENGINEERING,
FIRST CYCLE, 15 CREDITS
*STOCKHOLM*, *SWEDEN 2021*

# Rotary parking system
# Roterande parkeringssystem

**RAYAN ALNAKAR**

**DANILO CATOVIC**




**KTH ROYAL INSTITUTE OF TECHNOLOGY**
**SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT**


# Rotary Parking System

RAYAN ALNAKAR
DANILO CATOVIC

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

TRITA-ITM-EX 2021:47

# Abstract

Automatic parking systems are intended to save a lot of space and create a more favourable parking experience. In this thesis the main goal was to create a functional and user-friendly rotary parking system. The system consists of a framework, DC-motor, driveline, ultrasonic distance sensor and platforms. Most of the construction consists of plastic, the remaining parts are made of metal. The system was evaluated using two experiments, one that measured the speed of the system and the other one measuring the level of accuracy. After the first experiment, it was concluded that the system works well for the corresponding speed. The second experiment showed that improvements of the accuracy can be made.

Keywords: mechatronics, Arduino, ultrasonic sensor, DC-motor, keypad.

# Referat

## Roterande parkeringssystem

Automatiska parkeringssystem är avsedda att spara utrymme och skapa en bättre parkeringsupplevelse. I denna avhandling var huvudmålet att skapa ett funktionellt och användarvänligt roterande parkeringssystem. Systemet består av ett ramverk, likströmsmotor, drivlina, ultraljudssensor och plattformar. Majoriteten av konstruktionen består av plast, de återstående delarna är gjorda av metall. Systemet utvärderades med hjälp av två experiment, ett som mätte systemets hastighet och ett annat som mätte noggrannheten. Efter det första experimentet drogs slutsatsen att systemet fungerar bra för motsvarande hastighet. Det andra experimentet visade att förbättringar av noggrannheten kan göras.

Nyckelord: mekatronik, Arduino, ultraljudsensor, DC-motor, knappsats.

# Acknowledgements

We would like to thank our supervisor Nihad Subasic for his guidance and availability throughout the project. A special thanks to assistant Amir Avdic who, among other things, helped us with 3D-printing, further understanding of the components and for many fruitful discussions.

We would also like to thank Staffan Qvarnström for providing us with materials and sharing his knowledge about soldering and other fields. Additionally, a special thanks to Tomas Östberg, for helping us create a functional driveline. Lastly, we would like to thank all the students who contributed with their support and feedback throughout the project.

*Rayan Alnakar, Danilo Catovic*
*Stockholm May 2021*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

The following list describes several abbreviations that will be later used within the body of the document:

| | |
|---|---|
| $3D$ | Three Dimensional |
| $CAD$ | Computer Aided Design |
| $DC$ | Direct Current |
| $Hz$ | Hertz |
| $LED$ | Light Emitting Diode |
| $PLA$ | Polylactic Acid |
| $PMW$ | Pulse Width Modulation |
| $RPS$ | Rotary Parking System |
| $SEK$ | Swedish Crown |
| $UDS$ | Ultrasonic Distance Sensor |
| $USB$ | Universal Serial Bus |
| $V$ | Volts |

# Chapter 1

# Introduction

This chapter will present an overview of the background, purpose, scope and method of the project.

## 1.1 Background

Automatic parking systems have existed for a long time and were created to improve conventional parking. They are intended to save a lot of space and to create a more favourable parking experience for the user. In 1905, the first automatic parking system was introduced in Paris [1]. The Garage Rue de Ponthieu consisted of an elevator mechanism that could hoist cars to different levels. The system is shown in figure 1.1.

Figure 1.1: First automatic parking system, Paris, 1905 [1].

Since then, autonomous systems in many fields have developed tremendously and are vital in today's society. A rotary parking system (RPS) is one of many examples of such systems [2]. This thesis aims to gain a better insight into the structure, function and constituent components of a RPS.

Figure 1.2: Modern day parking system [2]

## 1.2 Purpose

The main purpose of this project is to construct a prototype of an RPS. This project contains the three fundamental components of mechatronics, which are mechanics, electronics and programming. The following research questions will subsequently be studied:

1. How should the system be structured to recognize the shortest path down?
2. What is the maximum time required to retrieve an arbitrarily placed vehicle?
3. What level of accuracy is achieved upon retrieval of an arbitrarily placed vehicle?

## 1.3 Scope

It is of great importance to have a notion of what the project's scope in advance will be limited by. In this case, the constraints are in the form of time, budget, resources, attention to detail and level of accuracy. The monetary limit is 1000 SEK and the time span is from January until the beginning of May, when the completed thesis must be submitted.
This project will mainly focus on creating a fully functional prototype of an RPS and answer the research questions. Certain advanced calculations, for instance the analysis and optimization for strength of the framework of the parking system, will not be taken into consideration as they are not relevant to the main topics of the course. Therefore, this thesis aims to create a better understanding of the theory and how the different constituent components need to interact with each other. The plan is for the prototype to be designed to accommodate six model vehicles simultaneously.

### 1.3.1 Method

The RPS is intended to be constructed using a predetermined fixed budget and the resources at hand. The main concept of the RPS is for small model cars to enter

the parking platform one at a time, whereupon a sensor in the system senses the occupied space and lifts the car up in a rotational step. Later, the user should be able to retrieve the car using a display where a number associated with the occupied parking platform is entered, resulting in a rotational motion where the requested car finds the shortest path down.
Among the components required to build the RPS prototype are an Arduino UNO microcontroller, DC-motor, UDS, sprockets, roller chains and revolving platforms. Several of these components can be found in stock, while others need to be procured separately, either through purchase or 3D-printing.

# Chapter 2

# Theory

In this chapter, the necessary theory and conducted research to answer the research questions are presented.

## 2.1 Microcontroller: Arduino Uno

There are several microcontroller boards available to choose from. In this project, the Arduino UNO was selected for the RPS. With an open-source electronics prototyping platform, the circuit board can be embedded into a wide range of projects of varying levels of difficulty [3]. A microchip is used in an Arduino, which is a tiny device that can be programmed to sense the conditions in the physical world by using attached sensors. The Arduino UNO can interact with one or multiple outputs, such as LEDs, motors, and displays, by reacting to certain conditions and inputs [4]. The inputs and outputs can either be digital, which is binary information using only the states true or false, or analog where the information is continuous and it can therefore map input voltages into integer values between 0 and 1023. By connecting it to a computer with a USB cable, a sufficient amount of power will be supplied to the Arduino and enable it to be programmed. The code uploaded to the Arduino board is written in the C programming language. The name of the 8-bit microcontroller used in Arduino boards is ATMEGA328P.

## 2.2 Ultrasound

Sound waves with a frequency above 20 000 Hz are called ultrasound, which is undetectable to humans. It is used in many applications. It is utilized by some mammals and birds in echolocation and communication, helping them navigate and detect prey or obstacles [5]. By measuring the time from transmission to reception of the reflected ultrasound, the distance to the reflective structure can be calculated if the speed of the sound in the medium is known [6].

### 2.2.1 Ultrasonic Ranging Module HC - SR04

The HC - SR04 ultrasonic distance sensor (UDS) offers a non-contact 0,02 - 4,00 m measurement capability with an accuracy of up to three millimeters [8]. The constituent components of the modules are: ultrasonic transmitters, receiver and control circuit. On the HC - SR04 four pins are found; Power supply, Trigger Pulse Input, Echo Pulse Output and Ground.
The transmitter sends a high-frequency trigger signal which is reflected and received as an echo signal if an obstacle is found, By using the time interval between transmitting and receiving the signal, it enables the calculation of the distance to the obstacle since the velocity of sound in air is known. The figure 2.1 below illustrates the functionality of a UDS.

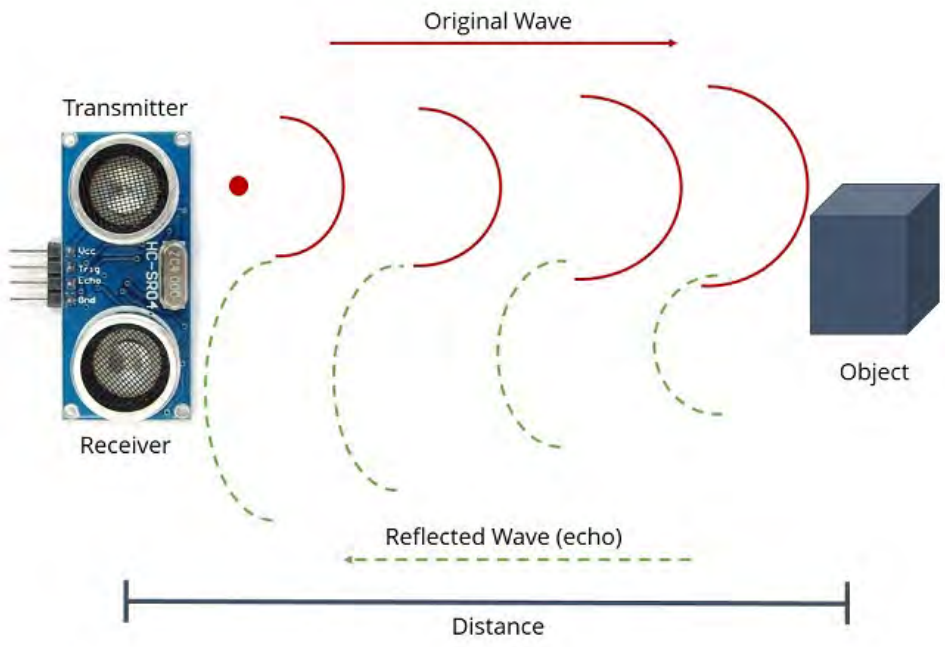



Figure 2.1: The functionality of UDS [7].

## 2.3 Potentiometer

A potentiometer is a commonly used electronic device. With the help of a potentiometer one can control resistance. Fundamentally, a potentiometer consists of three terminals and a dial [13]. By turning the dial, the resistance changes, which is why a potentiometer is often referred to as a variable resistor. When dealing with high power loads, such as motors, it is not efficient to use a potentiometer, simply because much of the energy is lost in the form of heat when passing through the resistance of the potentiometer.

## 2.4 DC-motor

The general use of motors is to generate torque. A DC-motor generates torque by converting electrical power into mechanical. In the figure below, 2.2, a DC-motor and its components are shown. When electrical power, such as a battery is connected to the commutator, a current will pass through the coil. When connected to a power source, the rotor coils which are surrounded by magnets will induce a magnetic field. The induced magnetic field generates a force that spins the motor. The direction of the force is controlled by the direction of the current that passes through the coils, a change of current direction will result in a clockwise or anticlockwise rotation [9].

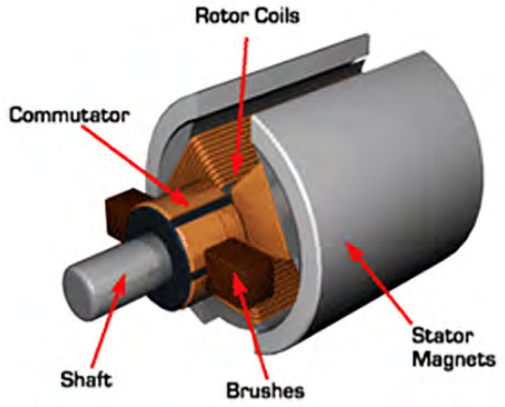



Figure 2.2: DC-motor [10].

### 2.4.1 Pulse Width Modulation

Pulse Width Modulation (PWM) is a technique that can be used to regulate DC-motors. The idea behind PWM and a potentiometer is to control the energy that is sent to a load. The potentiometer and PWM perform the same task but with different approaches. PWM works by turning the continuous signal on and off, by controlling the input signal in this way, different averages of the output signal can be generated. A very important concept in PWM is the duty cycle, which represents the duration of time during which the signal is on [11].

Figure 2.3 illustrates an example of a 50 % duty cycle. The output voltage can be calculated using the following equation:

$$V_{out} = V_{max} \cdot dutycycle \tag{2.1}$$

Given the equation 2.1 and numbers from figure 2.3 the average output voltage can be calculated to 6 $V$. By changing the duty cycle the speed of DC-motors can be controlled.

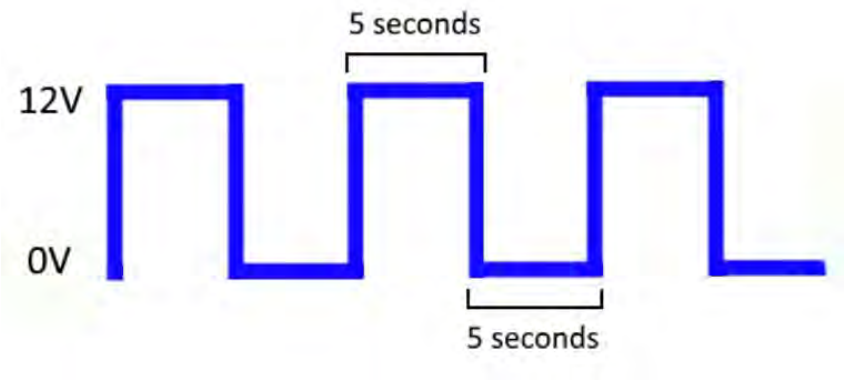



Figure 2.3: 50 % duty cycle [11].

## 2.5 H-bridge

An H-bridge is an electronic component that consists of four switches. By opening and closing the switches, the direction of the current changes, resulting in a change of polarity of the load. In robotics, an H-bridge is often used to control the direction of a DC-motor [12].

## 2.6 Keypad

An Arduino Keypad is a set of buttons arranged in rows and columns, resembling a matrix. The buttons on a keypad are referred to as keys. There are various types of keypads: 3x4 and 4x4 being two popular types. To detect any key pressing, the keypad is scanned to assess the row and column pin that is connected to the button. When entering a number on a keypad, the membrane switch between the row and column trace closes, enabling current to flow between a row pin and column pin [14].

# Chapter 3

# Demonstrator

In this chapter, the main problems are formulated and insight is given into the assembly and manufacturing processes.

## 3.1 Problem formulation

The goal of this project is to create a prototype of an RPS. In order to reach the goal, the following problems must be solved:

- Constructing the framework
- Regulate the DC-motor
- Design and build a functional driveline
- Installation of UDS that tracks the movement of the parking platforms

## 3.2 Electronics

In order to operate the RPS, one motor is required, which is powered by four AA 1.5 V batteries. To control the speed and direction of rotation of the motor, a custom-made L9997 motor driver is used. A data sheet for this specific motor driver could not be found. By using a UDS, it allows for regulating the stoppage position of the parking platforms, ensuring that they would not stop far away from their predetermined positions. With the help of the keypad, the user can choose between parking or retrieving a vehicle. Figure 3.1 shows the wiring of the constituent parts with the Arduino UNO board.

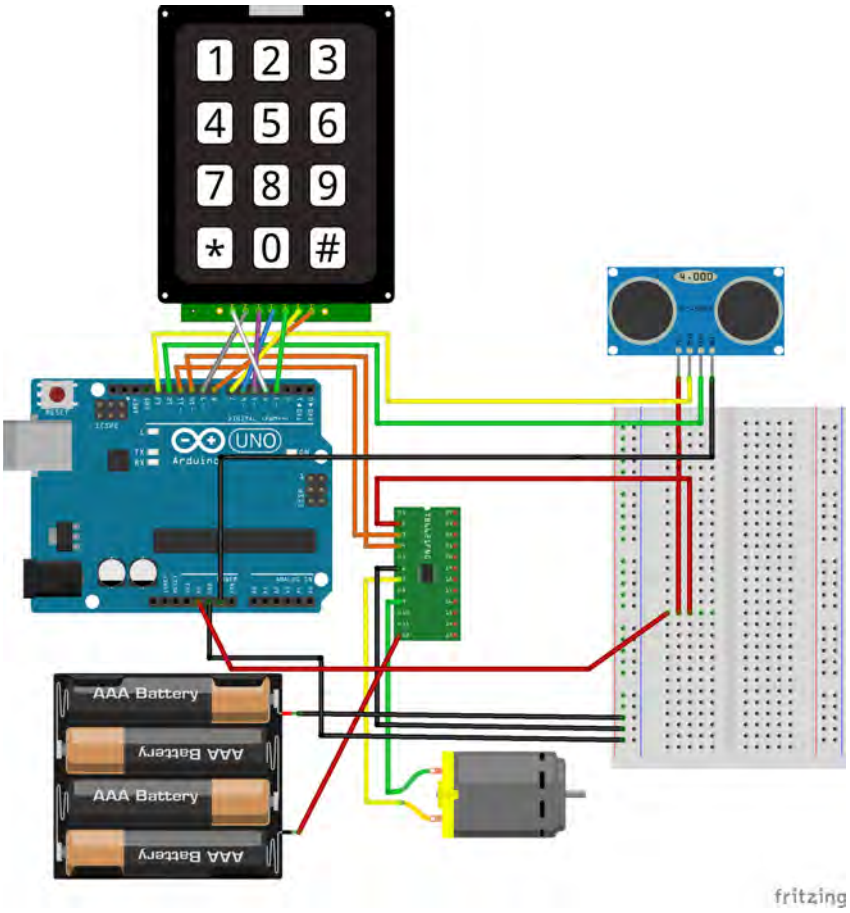



Figure 3.1: Arduino wiring diagram, created using Fritzing [15].

### 3.2.1 Microcontroller

An Arduino UNO board was the available type of microcontroller that would handle the components that operate the RPS. The Arduino board was programmed in the Arduino IDE using the C programming language. The digital input pins used on the Arduino board ranges from three to thirteen.

### 3.2.2 DC-motor

To set the system in motion, one DC-motor was used. The speed and direction of rotation of the motor was managed by the motor driver, whose input pins were connected to the digital input pins ten and eleven on the Arduino board. These are two out of six PWM pins on the Arduino UNO board that can simulate voltages between 0 V and 5 V.

### 3.2.3 Ultrasonic distance sensor

The UDS is located at the entrance of the RPS. It is attached to the plywood board under the platforms. Its task is to carry down the requested parking platform after input from the user.

In figure 3.2 below, an example of a user retrieving its vehicle can be seen. The user identifies the vehicle in place number four, whereby the number four on the keypad is entered to carry down the vehicle. When number four is entered, it tells the system to rotate clockwise (the shortest path down for that position) with the help of the motor driver. The system also tells the UDS to register each time the distance between the UDS and the lowest platform is measured, whereupon it adds one to the count.

The pink platform passes, and the system stops when the green platform is placed under the UDS. The sensor has now counted twice and tells the system to stop. This is possible because each position is fixed and preprogrammed. In comparison, if the user would like to bring vehicle number one down, the system is programmed to rotate anticlockwise and count once. After each vehicle retrieval, the system resets.

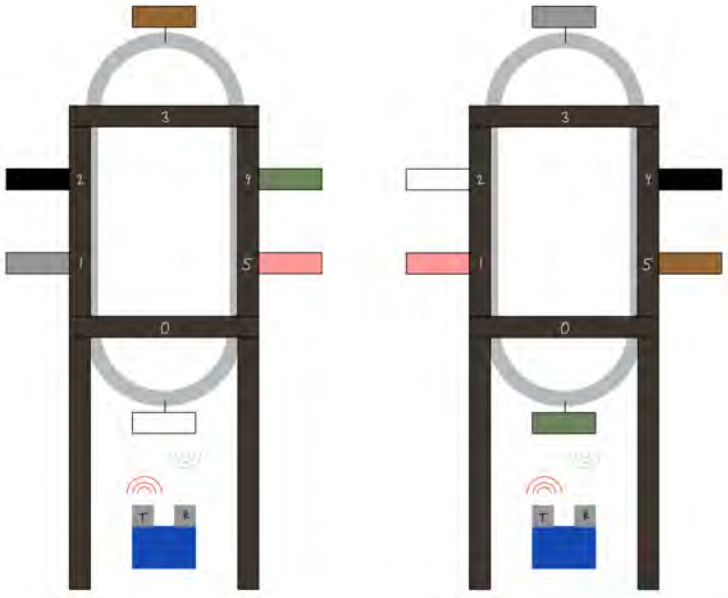



Figure 3.2: Illustration of a vehicle retrieval, created using Notability [18].

## 3.3 Hardware

This section will present an overview of how the mechanical components were constructed and assembled. Figure 3.3 below displays the completed RPS.

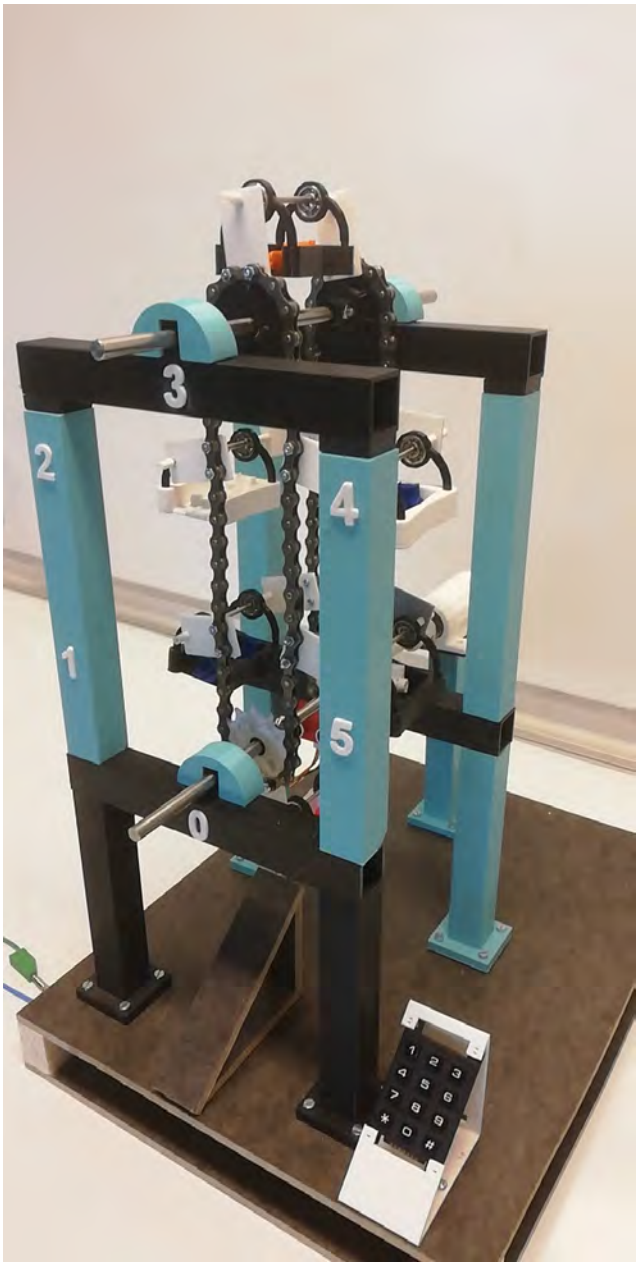



Figure 3.3: A picture of the completed RPS, taken by the authors.

### 3.3.1 Framework

The framework was designed using Solid Edge 2020 [16], where four criteria were taken into consideration. The framework needed to be stable, lightweight, adaptable for height adjustments and constructed in a way that aligns the axis both vertically and horizontally. Below is figure 3.4 showing the framework model.

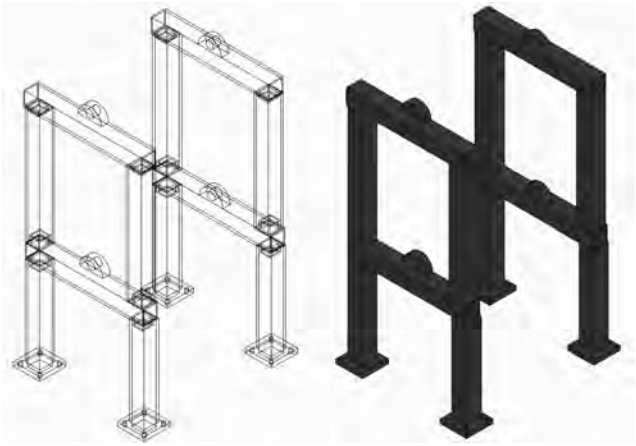


Figure 3.4: Framework model, created using Solid Edge 2020.

To stabilize the framework, the four legs were drilled onto a plywood board. The horizontal parts and bearing housings were 3D-printed as individual parts to align the axis properly. By having the parts of the framework being hollow, printing time and weight could be reduced. The different parts of the framework were assembled using glue.

In order to tighten or soften the chain, height adjustments were needed. It was accomplished by 3D-printing small puzzle like blocks that were placed between the upper vertical and horizontal parts of the framework.

### 3.3.2 Driveline

#### Ball bearings and bearing housing

Four bearings were purchased to allow rotation about the two main axis. The bearings will mainly endure radial forces. Since the radial forces would not be particularly impactful, bearing analysis was deemed not necessary. The bearing housing was designed in Solid Edge 2020 and 3D-printed using PLA filament with an Ultimaker printer.

#### Sprocket and chain

To transfer torque to the upper axis, two bicycle chains were attached to sprockets. The system contains two bicycle chains and four identical sprockets. The sprock-

ets were downloaded from Thingiverse [22]. Thingiverse is an open-source website designed for sharing user-created digital design files.
The opening of the sprocket is a hexagon. The mounting of the sprocket to the axis was made by 3D-printing a hexagon with an inner diameter matching the axis. The hexagon was glued to the sprocket and drilled into the axis to allow rotation. The bicycle chain was purchased and shortened with a mini chain brute to match the desired length. Figure 3.5 below shows a picture of the driveline of the RPS.

Figure 3.5: A picture of the driveline of the RPS, taken by the authors.

#### Shaft and shaft coupling

Two stainless steel shafts were purchased. The shaft of the DC-motor is connected via a shaft coupling. To connect the motor shaft to the axis, it was important that they were at the same horizontal height. This was made possible by 3D-printing a motor tower. The motor tower is connected to a motor housing. Afterwards the shaft coupling was 3D-printed. The shaft coupling is a cylindrical component that consists of two different inner diameters. One diameter that matches the motor shaft and the other matches the axis. The shaft coupling was fixed by drilling two holes in the shaft coupling itself and in each shaft, then the shaft coupling was screwed on. The motor tower, housing and shaft coupling were designed in Solid Edge 2020 and 3D-printed with an Ultimaker printer.

### 3.3.3 Parking platform

The RPS consists of six parking platforms. The platform was designed in Solid Edge 2020. Each platform was attached to the chain via a 3D-printed rectangle connected to an axis carrying the platform. Little ball bearings were installed in between the axis and parking platform to allow the platform to always maintain its place while rotating. Figure 3.6 below shows a picture of a parking platform.

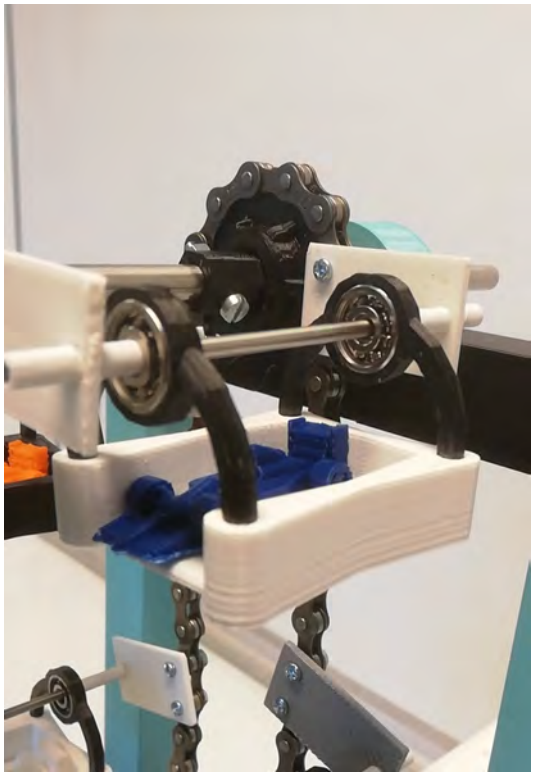


Figure 3.6: A picture of a parking platform, taken by the authors.

## 3.4 Software

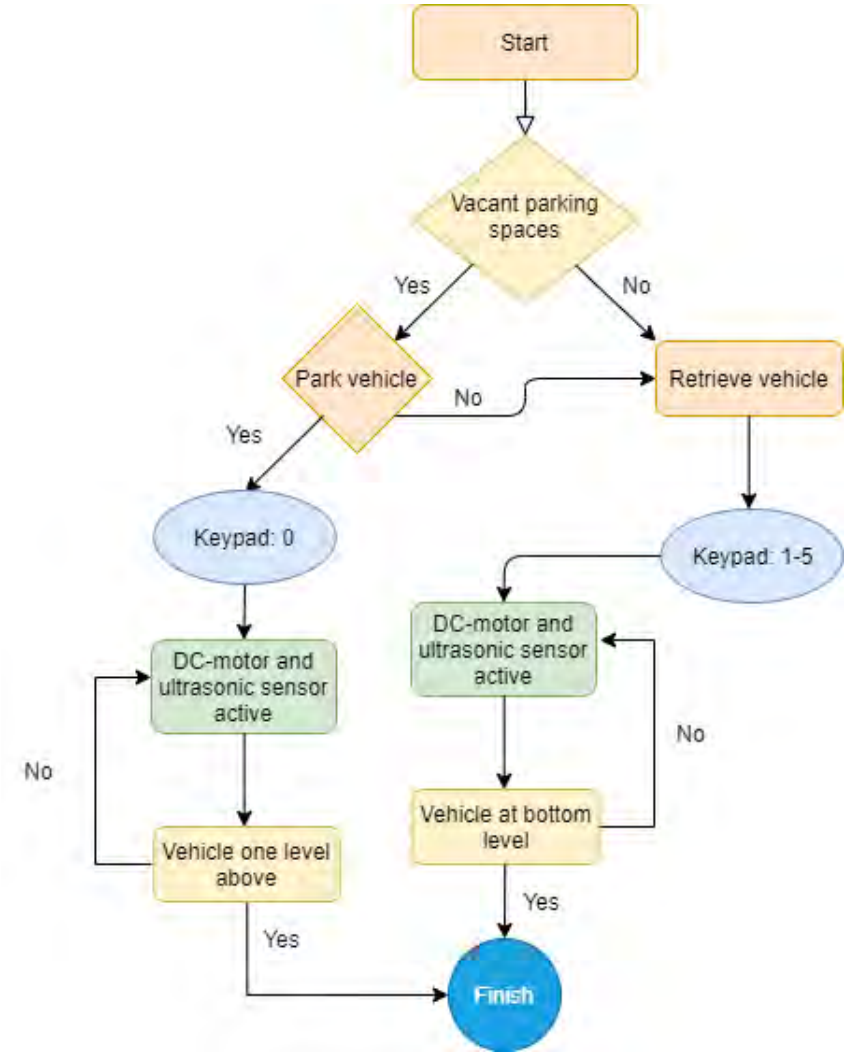



Figure 3.7: A flowchart illustrating the different stages of the RPS, created using app.diagrams.net [19].

A total of four softwares were utilized in this project. To model the prototype using a computer, Solid Edge 2020 was chosen. It is a 3D-CAD, solid modeling software. The open source Arduino software, an integrated development environment, was used to write code in the C programming language and upload it to the board. The simulation of the prototype was carried out in Acumen (found in Appendix ***B***), a tool to resemble models of cyber-physical systems. Ultimaker Cura, an open source slicing application, was used to upload the Solid Edge models to the 3D-printers.

The complete code for the Arduino software can be found in Appendix ***A***. The written code controls the Arduino board and the behaviour of its connected electronic parts: a DC-motor, a keypad and a UDS. The Keypad library was imported from the Hardware Abstraction libraries in order to use matrix style keypads with the Arduino.

The main part of the code is divided into six different cases using if statements, one for each parking platform in the RPS. Upon retrieval, each of the five upper platforms are programmed to choose the shortest path down, while the lowest platform is set up to move one step upwards to park a vehicle. The active keys on the keypad are zero through five, where the key zero sets the lowest platform in motion, and one through five for the upper ones.

# Chapter 4

# Results

To determine the results of the system, two experiments were devised in order to evaluate the quality and performance of the system. Two parameters were the main focus of the experiments; time and level of accuracy. The former was analyzed by conducting 20 tests under the same conditions for each platform to help increase the accuracy of the estimate and consequently make the experiment more reliable. The accuracy parameter was inspected by running the RPS 25 times for each platform and assessing in which region of the "semicircle" below a platform would end up in.

## 4.1 Speed

The purpose of the first experiment was to measure the time required for each platform to move downwards and stop at the bottom level. In the case of platform number zero where it always starts at the bottom level and moves upwards in a rotational motion, its collected data can be equated with the data for platform number one. This is due to the fact that both platforms move in the same direction and are stopped equally far from the starting point. In other words, the retrieval time of platform number one is the same as the parking time of platform number zero. This is also reflected in the Arduino software code, where both platforms are set in motion using the exact same code, respectively. In figure 4.1, five scatter charts for the retrieval time of the platforms are shown.

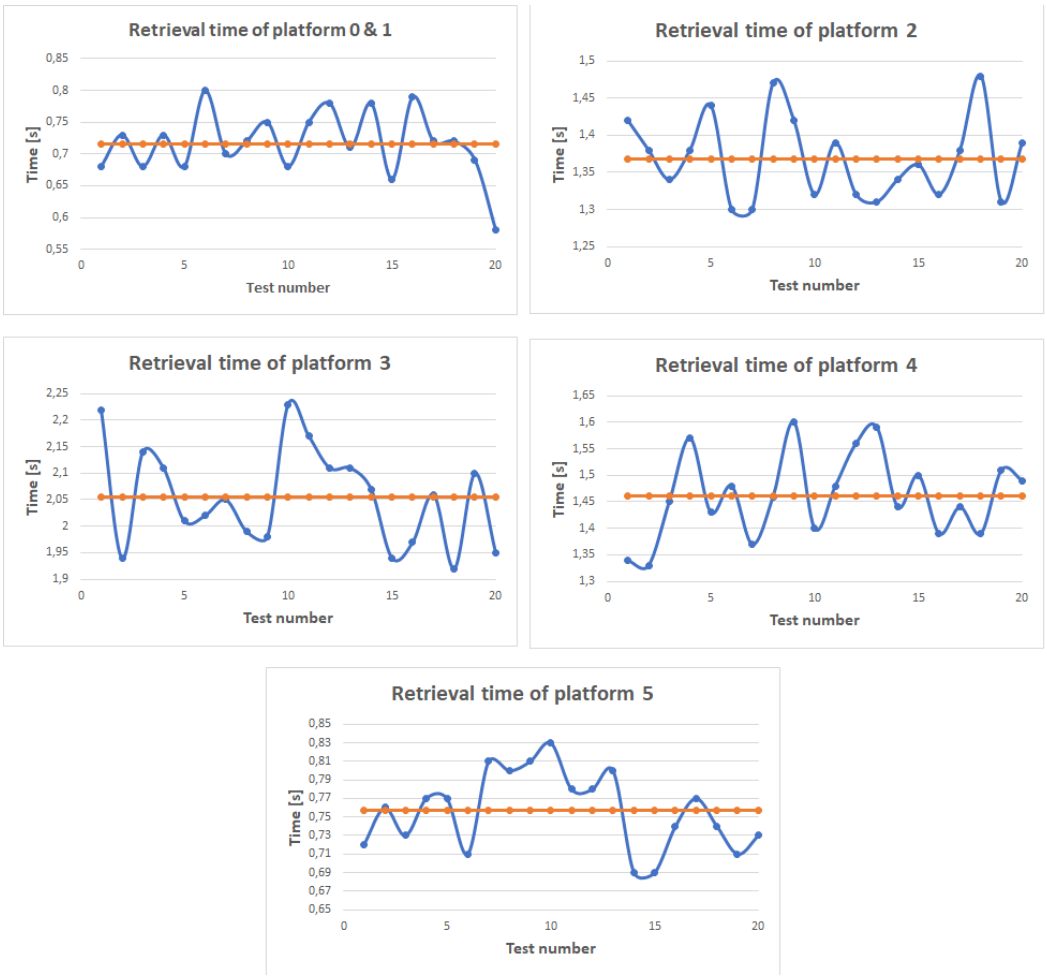



Figure 4.1: Five scatter charts displaying the retrieval time for the five platforms along with a line illustrating the average value of the 20 attempts performed for each platform. Created using Microsoft Excel [20].

To get an insight into the average amount of variability in the data set obtained, the standard deviation of the retrieval times for each platform was calculated. It measures the spread of a data distribution, meaning how far each score lies from the mean. A high standard deviation value suggests a more spread out data distribution, and a low standard deviation value indicates that data points are close to the mean. The standard deviation for each platform is displayed in table 4.1.

| | Platform 0 & 1 | Platform 2 | Platform 3 | Platform 4 | Platform 5 |
|---|---|---|---|---|---|
| Average time [s] | 0.7165 | 1.3685 | 2.0545 | 1.461 | 0.757 |
| Standard deviation | 0.0506 | 0.0545 | 0.0910 | 0.0771 | 0.0406 |

Table 4.1: Average retrieval time and standard deviation of the retrieval times for the six platforms.

## 4.2 Level of accuracy

The second experiment dealt with the evaluation of the level of accuracy achieved by the RPS. The precision was determined by estimating where a platform would end up after each run. Platforms zero through three move in anticlockwise direction. Conversely, platforms four and five move in the opposite direction, to maintain the condition of travelling the shortest path down upon retrieval. The stoppage area at the bottom level was simulated as a "semicircle", divided in five sections. In figures 4.2 and 4.3, the five different cases of stoppage accuracy are illustrated. The red, rectangular marking in each figure shows which current platform was being tested.

Every case was simulated 25 times, reaching a total of 125 test results, since platforms zero and one are represented as one individual case as outlined above. The green circle sectors (60 degrees) are the optimal stoppage areas for the platforms, the yellow ones (45 degrees each) less so, and the red circle sectors (15 degrees each) are the least desired areas for the platforms to stop in. The numbers in the colored areas indicate the number of times the current platform stopped there.

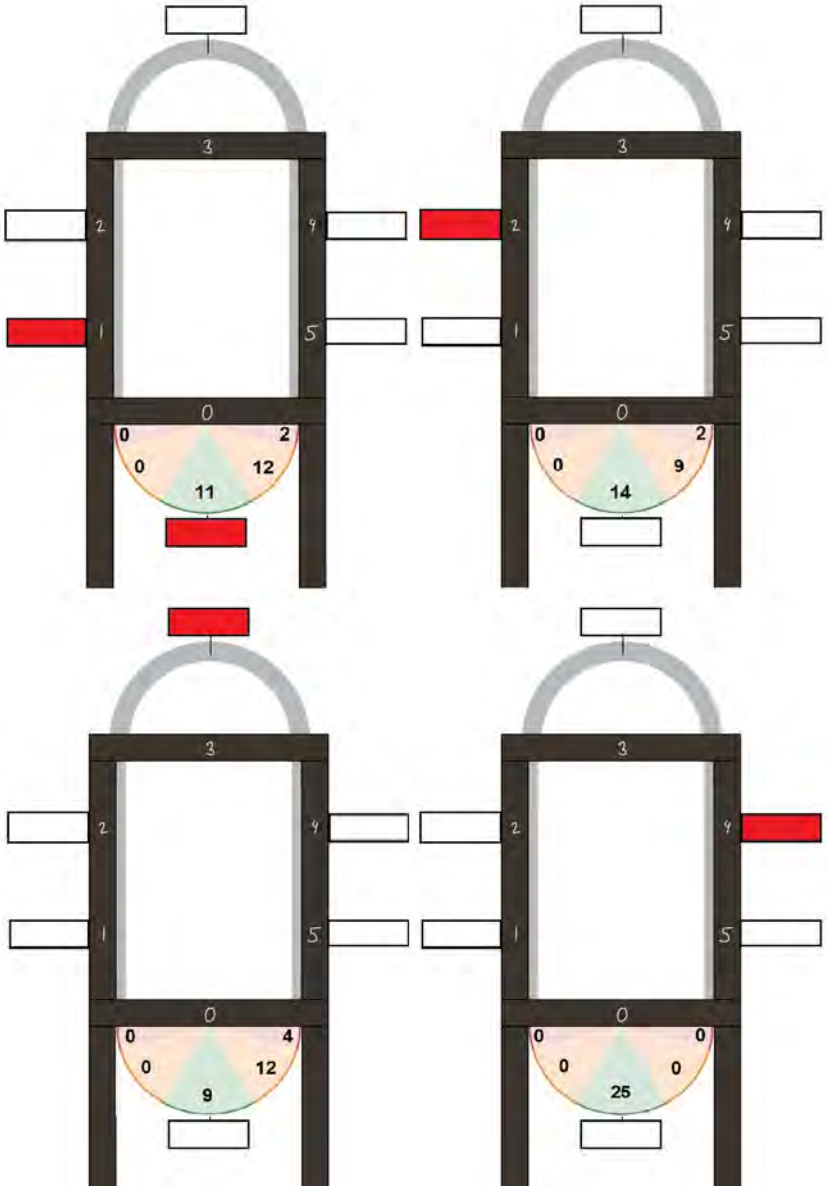



Figure 4.2: Level of accuracy for platforms zero through four. Created using Desmos and Microsoft Paint [17] [21].

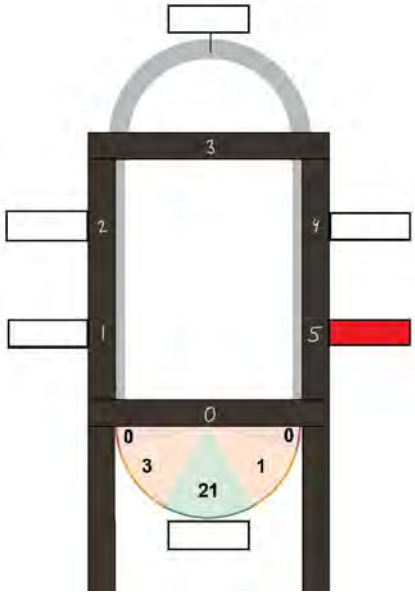



Figure 4.3: Level of accuracy for platform number five. Created using Desmos and Microsoft Paint [17] [21].

# Chapter 5

# Discussion and conclusions

This work has been very educational and fun. At the beginning of the work, the decision was made to build a large prototype in wood. A wooden frame was built and assembled with the remaining components. Tests were performed without any successful results. The problem with the wooden framework was that the two symmetrical parts that make up the framework were not identical. These undesirable properties resulted in the decision to dissolve the work on the wooden structure.

Regarding the ability to recognize platforms in motion, the UDS was not the only option in mind. Initially, a light sensor was considered for the recognition function. However, after some time trying to figure out how and if it should be implemented, it was decided that a UDS would be a more suitable choice. A light sensor would have required special conditions to function properly. One such condition would be for the moving platforms to rotate in front of and cover the light sensor perfectly. Consequently, moving platforms would be registered every time the brightness level is below a certain predetermined value.

The location of the UDS was of utmost importance. Every time a number on the keypad is entered, the RPS is activated, which in turn starts the UDS and DC-motor. The initial plan of having the UDS laying down facing up had to be changed. This was because the UDS would immediately register the platform at the bottom level when it was not meant to. Therefore, it led to the incorrect assumption that a platform had already passed, when in reality the RPS had just started and thus not had enough time to move one rotational step. This was solved by repositioning the UDS so that nothing covered it upon initiation of the RPS.

Other factors affecting the performance of the RPS were the platforms' distance to the UDS, DC-motor's rotational speed and the delay time of the UDS, written in the Arduino code. The latter refers to how often the UDS is told to measure the distance to the obstacle in front. All these factors needed to work in conjunction with each other to run the RPS properly. The optimal selected distance for the

UDS was five centimeters. This implies that every time a moving platform was in front of the UDS, the system would register it as if a platform had passed by, adding it to the counter. When platforms in motion are temporarily straight in front of the UDS, the distance of a few of the surfaces on the platforms is longer than five centimeters, where only one is below five. By having the predetermined distance at five centimeters, it reduces the risk of the system mistakenly counting a platform more than once.

The delay time for each platform was selected after multiple attempts to try to determine which would be the ideal number for a platform to stop in the middle of the semicircle below, since a single delay time for all platforms was not a viable solution. This had to do with the fact that the six platforms, although being placed symmetrically around the chain, moved to some extent differently. Consequently, this required manual adjustments to make them act accordingly. Platform one and five, for instance, should move similarly, although in opposite directions.

In the case of the speed of the DC-motor for the platforms, it was not possible to make them all move slowly at the same speed and stop at fairly acceptable areas in the semicircle below. This was due to resistance in the chain motion, often leading to the failure of the vehicle retrieval and platforms lurching helplessly. This was solved by selecting slightly different rotational speeds for the platforms to overcome the resistance in the chain motion, without significantly interfering with the stoppage accuracy.

The test results obtained in the two experiments explained earlier were to some extent as expected. When comparing the average retrieval time of opposite platforms, in other words platforms zero and five, as well as two and four, a similarity between retrieval times can be observed. The times 0.7165 and 1.3685 are almost equal in value to 1.461 and 0.757. Platform three has no equivalent as it is the only one that moves from top to bottom, with an average time of 2.0545 seconds, which of course is the longest retrieval time as it is positioned at the top. As it requires the longest time to move downwards, it is not unforeseen that the standard deviation for platform three is the highest compared to others. This stems from the fact that the retrieval times for platform three are more dispersed in relation to the average time.

It is worth mentioning that there is a small margin of error regarding the timekeeping of each retrieval. Every time the RPS was set in motion, by entering a number on the keypad with one hand, the other hand was used simultaneously to start the timer. This was carried out 25 times for each case as carefully as possible, however, it can still be regarded as a source of error due to human reaction time.

As for the level of accuracy results, some of them were a little surprising. Platforms zero through three that moved in anticlockwise direction all stopped in the middle and the two right circle sectors. This had to do with the fact that the UDS

was located behind the simulated semicircle, illustrated in figure 4.2. Whenever a rotating platform was about to stop, the predetermined delay time after the last platform recognition would cause it to move a little extra before it stopped.

The results of platforms four and five were unanticipated. The stoppage accuracy of the fifth platform was impressive while the fourth's was perfect. Presumably it was because they were the only two platforms that rotated in the opposite direction, resulting in a slightly different recognition timing.

In an attempt to get as accurate results as possible, they different cases were tested consecutively instead of 25 times for each case respectively. At first glance, the accuracy achieved by platforms zero through three might seem mediocre. It is worth mentioning, however, that in many of the test evaluations, it was quite difficult to determine which circle sector a platform had stopped in, often leading to the selection of the less accurate area instead of the circle sector in or close to the middle.

## 5.1 Conclusions

In general the overall implementation and function of the completed RPS are fairly similar to what was originally envisioned from the beginning. The results were pleasing and the mechanical, electrical and programmable elements adequately satisfied the initial requirements.

One of the most important aspects of the RPS was how the system would be structured to recognize the shortest path down. This was the first research question. In hindsight, this was achieved by manually programming each of the six platforms to move in a certain direction and stop with the help of a UDS at an appropriate place.

The two other research questions concerned the maximum retrieval time and level of accuracy achieved by the system. This resulted in the average retrieval time of 2.0545 seconds for the third platform, the longest time out of all of the platforms. When it comes to accuracy, 80 out of 125 attempts resulted in stoppage inside the green, optimal circle sectors. Many of the remaining attempts also stopped in close proximity to the middle, but not near enough to be included in the green area.

# Chapter 6

# Future improvements

For further work and improvements on this project, you could scale up the system, add safety features and improve the parking experience for the user.

By scaling up the system new factors must be taken into consideration. The first consideration is that you would have to deal with a larger load, resulting in a need for a stronger motor. You could also have to consider how to stack the cars to minimize the different loads in the system, and mechanically locking the system while not operating. A user-friendly display could be added to the system, displaying available parking lots, and interacting with the users while parking or retrieving their vehicle. Safety features could be added to the display, for instance the user might need to confirm questions like, is the car empty, is the engine switched off, and so on. A parking ramp could be added to the system. This would require a servo motor and infrared sensors counting entering and exiting vehicles.

There are many different approaches to make the system recognize where the platforms are located. The optimum would be for the system to be as user-friendly as possible. In this prototype, the user is forced to locate where their vehicle is parked, find the corresponding number of the platform, and enter it on the keypad to retrieve their vehicle. This is a solution that works but is not optimal because it requires too much user involvement. Since this is only a prototype such a solution works well. However, you would have to resort to other solutions to operate a real, scaled up RPS. For a scaled up version, every platform could have its own number and a camera that reads these numbers. When a user parks their vehicle they get a message on the display telling them which platform they have parked their vehicle at. Upon retrieval of the vehicle, they simply enter the same number.

# Bibliography


[1] Qurius, Vertical parking (1920-1976). https://www.q-park.com/blogs/newsitem/11761/vertical-parking-1920-1976 *Acquisition date: 2021-01-30*

[2] No specified author. *VW conquers the world*. 2012. https://www.economist.com/business/2012/07/07/vw-conquers-the-world *Acquisition date: 2021-04-11*

[3] Culkin, J. *Introduction to Arduino*. 2011. https://playground.arduino.cc/Main/ArduinoComic/ *Acquisition date: 2021-02-15*

[4] No specified author. *Arduino For Beginners*. https://www.makerspaces.com/wp-content/uploads/2017/02/Arduino-For-Beginners-REV2.pdf *Acquisition date: 2021-02-15*

[5] J. Philipsson, J-A. Dahlström, J. Malmquist, L. Grahm, R. Hall. National encyclopedia, *"ultraljud"*. http://www.ne.se/uppslagsverk/encyklopedi/lång/ultraljud *Acquisition date: 2021-02-15*

[6] Susannah J.Patey, James P.Corcoran: *Physics of ultrasound*, Anaesthesia Intensive Care Medicine, Volume 22, Issue 1, 2021, Pages 58-63, https://www.sciencedirect.com/science/article/pii/S1472029920302435 *Acquisition date: 2021-02-15*

[7] *Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino.* https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04 *Acquisition date: 2021-02-15*

[8] ElecFreaks, *Ultrasonic Ranging Module HC - SR04* https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf *Acquisition date: 2021-02-15*

[9] Magnet acadamy, *DC-motor* https://nationalmaglab.org/education/magnet-academy/watch-play/interactive/dc-motor *Acquisition date: 2021-02-15*

[10] DC-motor, *Brushed DC-motor construction*
`https://www.linearmotiontips.com/are-brushed-motors-suitable-for-industrial-applications/` *Acquisition date: 2021-02-15*

[11] What is duty cycle?, *Duty cycle*
`http://blog.actuonix.com/2017/04/what-is-duty-cycle.html`
*Acquisition date: 2021-02-15*

[12] Dahl, Ø. *What is an H-bridge?*
`https://www.build-electronic-circuits.com/h-bridge/`
*Acquisition date: 2021-02-15*

[13] *Potentiometer*
`http://www.ne.se.focus.lib.kth.se/uppslagsverk/encyklopedi/lång/potentiometer` *Acquisition date: 2021-02-15*

[14] Krishna Pattabiraman, *HOW TO SET UP A KEYPAD ON AN ARDUINO*
`https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/`
*Acquisition date: 2021-04-11*

[15] Fritzing, https://www.fritzing.org/, software
*Acquisition date: 2021-04-11*

[16] Solid Edge, 2020, Siemens, https://www.solidedge.siemens.com, CAD software.
*Acquisition date: 2021-04-11*

[17] Desmos, https://www.desmos.com/calculator, Online
*Acquisition date: 2021-04-11*

[18] Notability, https://apps.apple.com/se/app/notability/id360593530, Apple application
*Acquisition date: 2021-04-11*

[19] Diagrams, https://www.app.diagrams.net/, Online
*Acquisition date: 2021-04-11*

[20] Microsoft Excel, https://www.microsoft.com/microsoft-365/excel, software
*Acquisition date: 2021-04-11*

[21] Microsoft Paint, https://support.microsoft.com/sv-se/windows/skaffa-microsoft-paint-a6b9578c-ed1c-5b09-0699-4ed8115f9aa9, software
*Acquisition date: 2021-04-11*

[22] Thingiverse, *12 tooth sprocket for standard bicycle chain*
`https://www.thingiverse.com/thing:2474954`
*Acquisition date: 2021-04-11*

# Appendix A

# First Appendix - Arduino code

```
//Rotary parking system
//Date : 2021-04-15
//Written by: Rayan Alnakar & Danilo Catovic
//Examinor : Nihad Subasic
//TRITA-nr: TRITA-KTH-ITM 2021:47
//Course: MF133X

//The Arduino code:



#include <Keypad.h>

int trigPin = 13;    // TRIG pin
int echoPin = 12;    // ECHO pin

float duration_us, distance_cm;
int raknare = 0;

const int ROW_NUM = 4; //four rows
const int COLUMN_NUM = 3; //three columns

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
//connect to the row pinouts of the keypad
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};
```

```
//connect to the column pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3};

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM, COLUMN_NUM );

void setup(){
  Serial.begin(9600);

  // configure the trigger pin to output mode
  pinMode(trigPin, OUTPUT);
  // configure the echo pin to input mode
  pinMode(echoPin, INPUT);
}

double get_distance() {

    // generate 10-microsecond pulse to TRIG pin
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // measure duration of pulse from ECHO pin
    duration_us = pulseIn(echoPin, HIGH);

    // calculate the distance
    distance_cm = 0.017 * duration_us;

    // print the value to Serial Monitor
    Serial.print("distance: ");
    Serial.print(distance_cm);
    Serial.println(" cm");

    return distance_cm;
}

void rotateLift(int spots, int speed, int direction, int delay_time) {

    int raknare = 0;
    // If statement to determine the direction of the rotation of the DC-motor
    // and to set it in motion.
    if (direction == 1) {
        analogWrite(11, speed);
        analogWrite(10,0);
```

```
    } else {
        analogWrite(11, 0);
        analogWrite(10,speed);
    }
    while (true) {
      // If the distance to an obstacle in front is less than five,
      // add to the count.
        if (get_distance() < 5){

            Serial.println(raknare);
            raknare++;
            delay(delay_time);
            Serial.println(raknare);
            // if the counter is equal or more than the number of spots,
            // stop the DC-motor.
            if (raknare >= spots) {
                analogWrite(11,0);
                analogWrite(10,0);
                break;
            }
        }
    }
}
          // rotateLift(int spots, int speed, int direction, int delay_time)
void callLift(char key) {
    // A switch case depending on which number is entered on the keypad.
    switch(key) {
        // Different scenarious for the 6 platforms. As seen below,
        // platforms zero and one are operated with the same code.
        // The numbers inside "rotateLift" are adjusted to reach an
        // acceptable level of performance of the system.
        case '0':
        // First variable regulates how many platforms pass the
        // ultrasonic distance sensor before the system stops.
        case '1':
            rotateLift(1,75,1,100);
        break;
        // Second variable adjusts the speed of the DC-motor.
        // It ranges from zero to 255.
        case '2':
            rotateLift(2,70,1,125);
        break;
        // Third variable is either one or zero.
        // Decides the direction of rotation of the DC-motor.
```

```
        case '3':
            rotateLift(3,70,1,150);
        break;
        // Fourth variable is the delay time for the distance sensor.
        // It controls how ofter the sensor measure the distance in front.
        case '4':
            rotateLift(2,70,0,550);

        break;
        case '5':
            rotateLift(1,70,0,550);
        break;
    }
}

void loop(){
  char key = keypad.getKey();
  // Calling the callLift function in a loop to always
  // be active whenever a number on the keypad is entered.
  callLift(key);

}
```

# Appendix B

# Second Appendix - Acumen code

```
//Rotary parking system
//Date : 2021-03-28
//Written by: Rayan Alnakar & Danilo Catovic
//Examiner : Nihad Subasic
//TRITA-nr: TRITA-KTH-ITM 2021:47
//Course: MF133X

//The Acumen simulation code:


model Main (simulator) =
initially
  _3D = ((Box // 1st side: 1st lower, vertical red part
            center=(-0.3,0,0)
            size=(0.4,1,5)
            color=red
            rotation=(0,0,0))
          (Box // 1st side: 2nd lower, vertical red part
            center=(4.3,0,0)
            size=(0.4,1,5)
            color=red
            rotation=(0,0,0))

           (Box // 1st side: lower, horizontal black part
            center=(2,0,2.7)
            size=(5,1,0.4)
            color=black
            rotation=(0,0,0))

           (Box // 1st side: 1st upper, vertical blue part
```

```
 center=(4,0,6)
 size=(0.2,1,6.2)
 color=blue
 rotation=(0,0,0))

(Box // 1st side: 2nd upper, vertical blue part
 center=(0,0,6)
 size=(0.2,1,6.2)
 color=blue
 rotation=(0,0,0))

(Box // 1st side: upper, horizontal black part
 center=(2,0,9)
 size=(3.8,1,0.2)
 color=black
 rotation=(0,0,0))


(Box // 1st side: 1st chain represented with the yellow part
 center=(0.9,0.6,6)
 size=(0.2,0.2,6.2)
 color=yellow
 rotation=(0,0,0))

(Box // 1st side: 1st chain represented with the yellow part
 center=(3.1,0.6,6)
 size=(0.2,0.2,6.2)
 color=yellow
 rotation=(0,0,0))


// 4 cylinders representing the bicycle sprockets
(Cylinder
 center=(2,3.35,9)
 size=(0.25,0.8)
 color=green
 rotation=(0,0,0))


(Cylinder
 center=(2,3.35,4)
 size=(0.25,1)
 color=green
 rotation=(0,0,0))
```

```
 (Cylinder
  center=(2,0.6,9)
  size=(0.25,1)
  color=green
  rotation=(0,0,0))


 (Cylinder
  center=(2,0.6,4)
  size=(0.25,1)
  color=green
  rotation=(0,0,0))


  // 2 shafts between the bicycle sprockets
  (Cylinder
  center=(2,1.9,4)
  size=(2.8,0.25)
  color=blue
  rotation=(0,0,0))

  (Cylinder
  center=(2,1.9,9)
  size=(2.8,0.25)
  color=blue
  rotation=(0,0,0))


(Box // 2nd side: 1st lower, vertical red part
  center=(-0.3,4,0)
  size=(0.4,1,5)
  color=red
  rotation=(0,0,0))

(Box // 2nd side: 2nd lower, vertical red part
  center=(4.3,4,0)
  size=(0.4,1,5)
  color=red
  rotation=(0,0,0))

 (Box // 2nd side: lower, horizontal black part
  center=(2,4,2.7)
  size=(5,1,0.4)
```

```
 color=black
 rotation=(0,0,0))

(Box // 2nd side: 1st upper, vertical blue part
 center=(4,4,6)
 size=(0.2,1,6.2)
 color=blue
 rotation=(0,0,0))

(Box // 2nd side: 2nd upper, vertical blue part
 center=(0,4,6)
 size=(0.2,1,6.2)
 color=blue
 rotation=(0,0,0))

(Box // 2nd side: upper, horizontal black part
 center=(2,4,9)
 size=(3.8,1,0.2)
 color=black
 rotation=(0,0,0))

(Box // 1st side: 1st chain represented with the yellow part
 center=(0.9,3.3,6)
 size=(0.2,0.2,6.2)
 color=yellow
 rotation=(0,0,0))

(Box // 1st side: 1st chain represented with the yellow part
 center=(3.1,3.3,6)
 size=(0.2,0.2,6.2)
 color=yellow
 rotation=(0,0,0)))
```

TRITA-ITM-EX 2021:47