AUTOMATIC PERPENDICULAR PARKING OF A CAR- TRAILER SYSTEM

by

151220132072 SERDAR TEMELTAŞI

151220132078 MUHAMMED FATİH TÜZ

151220132048 BATUHAN TÜRKAN

151220132012 BERK BAYRAKDAR

A Graduation Project Report

Electrical Electronics Engineering Department

JUNE 2018

AUTOMATIC PERPENDICULAR PARKING OF A CAR- TRAILER SYSTEM

by

151220132072 SERDAR TEMELTAŞI

151220132078 MUHAMMED FATİH TÜZ

151220132048 BATUHAN TÜRKAN

151220132012 BERK BAYRAKDAR

A Report Presented in Partial Fulfilment of the Requirements for the Degree Bachelor of Science in Electrical Electronics Engineering

ESKISEHIR OSMANGAZI UNIVERSITY

JUNE 2018

AUTOMATIC PERPENDICULAR PARKING OF A CAR- TRAILER SYSTEM

by

151220132072 SERDAR TEMELTAŞI

151220132078 MUHAMMED FATİH TÜZ

151220132048 BATUHAN TÜRKAN

151220132012 BERK BAYRAKDAR

has been approved by

Supervisory Committee

_____

Prof. Dr. Osman PARLAKTUNA

_____

Asst. Prof. Dr. Burak KALECİ

_____

Asst. Prof. Dr. Gökhan DINDIŞ

ABSTRACT

In this study, the problem of automatic parking of cars and similar vehicles is addressed and it is aimed to give an example for the system that allows the vehicle to park on its own, and it is aimed to make it easier for the drivers to park their vehicles. Automatic Perpendicular Parking of a Car- Trailer System (APPCTS) is the process of parking trailer vehicles in the most appropriate way, considering all imaginable possibilities. While working on these possibilities, the system has been able to find a suitable parking space by scanning the environment with ultrasonic and infrared distance sensors, and by performing motor motion simultaneously with scanning operation. To perform all these operations, the system, uses Arduino MEGA microcontroller, which uses ATMEL's microchip Atmega328P. In addition, ultrasonic sensors and infrared distance sensor are used for automatic vehicle parking system.

Keywords: Car-Like-Mobile-Robot (CLMR), Perpendicular Parking, Autonomous Vehicle, Ultrasonic Distance Sensor, Infrared Distance Sensor, Arduino, Atmega328P

ÖZET

Bu çalışmada, römorka sahip otomobil ve benzeri araçların otomatik dikey park edilmesi sorunu ele alınmış ve aracın kendi başına dikey olarak park etmesini sağlayan sisteme örnek vermesi amaçlanmıştır. Böylece sürücülerin araçlarını park etmesini kolaylaştırmak hedeflenmiştir. Römorklu bir aracın otomatik dikey park edilmesi sistemi, tüm akla gelebilecek olasılıklar göz önünde bulundurularak, römork araçlarının en uygun şekilde park edilmesi işlemini gerçekleştirmeyi amaçlamaktadır. Sistem, ultrasonik ve kızıl ötesi mesafe sensörleri ile çevreyi tarayarak ve tarama işlemiyle aynı zamanlı olan motor hareketini gerçekleştirerek uygun bir park alanının bulunması ve gerekli manevralar ile park işleminin gerçekleştirilmesi maksadı ile geliştirilmiştir. Tüm bu işlemleri gerçekleştirilmesi için, prototip, ATMEL'in Atmega328P mikrokontrolörünü kullanan Arduino MEGA mikro denetleyicisini kullanır. Ayrıca park işlemi sırasında çevreyi taramak ve olası engelleri bulmak amacıyla ultrasonik sensörler ve kızılötesi mesafe sensörlerinden yardım alınmıştır.

Anahtar Kelimeler: Araba Benzeri Mobil Robot, Dikey Park, İnsansız Araç, Ultrasonik Mesafe Sensörü, Kızılötesi Mesafe Sensörü, Arduino, Atmega328P

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our advisor Prof. Dr. Osman PARLAKTUNA for the continuous support during our study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research and writing of this report. His extraordinary support during this paper motivated us to finish this project and paper.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Explanation
μ:	Micro
°C:	Celsius degree
d:	Distance in centimeters
C:	Speed in meters/seconds
T:	Time in seconds
Δt:	Time Difference
C_{air}:	Speed of Sound in Air
T_{air}:	Temperature of Air
Γ:	Heat Capacity Ratio
V:	Voltage
R:	Range
k:	Constant
R_{12}, R_1, R_2:	Gear Ratios
N_1, N_2, N_3:	Number of Teeth on Gears
S:	Steps per Revolution
Θ_{step}:	Step Angle
s:	Steps Taken at Each Centimeters
c:	Circumference
V_{max}:	Maximum Speed
t_{min}:	Minimum Time per Step
I_{max}:	Maximum Current
Ω:	Ohm
P_{max}:	Maximum Power Spent

Symbol	**Explanation**
W:	Watt
P_{DSRON}:	Power During the “on” Stage of D-S
R_{DSON}:	Resistance During the “on” Stage of D-S
I_{LOAD}:	Current of Load
P_{SWavg}:	Average Power Dissipation During Switching
V_{batt}:	Voltage Supplied via Battery
t_{sw}:	Time of Switching
R_{Sx}:	Resistance of the Sense Resistor
V_{REF}:	Reference Voltage
$I_{tripMAX}$:	Value of Current Limiting

Abbreviation	**Explanation**
Kg:	Kilogram
cm:	Centimeters
KHz:	Kilohertz
µS:	Microseconds
mA:	Milliampere
V:	Voltage
mS:	Milliseconds
kB:	Kilobyte
I/O:	Input/Output
g:	Gram
A:	Amepere

Abbreviation	Explanation
ISO:	International Organization for Standardization
IEC:	The International Electro-technical Commission
IEEE:	The Institute of Electrical and Electronics Engineers
USART:	Universal Synchronous/Asynchronous Receiver/Transmitter
UART:	Universal Asynchronous Receiver/Transmitter
TTL:	Transistor-Transistor Logic
IDE:	Integrated Development Environment
IR:	Infrared
m:	Meters
Atm:	Atmosphere
DMOS:	Diffusion Metal Oxide Semiconductor
DC:	Direct Current
LED:	Light Emitting Diode
GND:	Ground
USB:	Universal Serial Bus
ADC:	Analog-Digital Converter
PERT:	Program Evaluation Review Technique
CLMR:	Car-Like Mobile Robot
APPCTS:	Autonomous Perpendicular Parking of Car-Trailer System

1. INTRODUCTION

Since the invention of the cars, they have been preferred due to they make it possible to travel long distances in a short period of time and can carry heavy loads without needing human power. Therefore, the automobile market has always been a profitable area. The number of car sales, which was 39.2 million in 1990, approaches 81.57 million in 2018 [1] and revenue of the leading automotive manufacturers worldwide in 2016 changes between 104.13 billion dollars and 254.69 billion dollars [2]. Because of that, in recent years, many technological developments have come to the fore in the automotive sector. These new developments in car technology have attracted the attention of consumers. In USA, potentiality of purchasing a new car with an advanced technology in the next twenty-four months is 79% in millennials, 66% in generation-X, 42% in baby boomers [3]. In addition to improving the driving performance of the vehicles, the activities are being carried out in order to increase driving pleasure and make the journey more comfortable. One of these technologies is intelligent parking systems that we frequently encounter in new vehicles. This feature is that the vehicle has the ability to park on its own without needing anybody. Such manufacturers, like Ford [4], Volkswagen [5] and BMW [6] use and develop this technology.

The human factor plays a major role in traffic accidents. Turkish Statistical Institute reported, “Among the defects causing accidents, driver defects are the first with 89.6% [7]” Also, according to U.S National Safety Council, “ More than 50,000 crashes occur in parking lots and garage structures annually, resulting in 500 or more deaths and more than 60,000 injuries [8].“ Furthermore, U.S Federal Motor Carrier Safety Administration states, “In 2016, 4,440 large trucks and buses were involved in fatal crashes, a 2-percent increase from 2015 [9].” Technologies like intelligent parking system aim to prevent the accidents caused by humans by removing the human factor from the equation.

Accidents caused by vehicles made for personal use lead to less loss of property and life than vehicles with trailer systems. This project is aimed to reduce the accidents that may

occur during parking to 0% by eliminating human error factor with using autonomous perpendicular parking system. Also, monetary losses and loss of lives will decrease due to reduction of accidents. Moreover, by making the parking process autonomous, there will be a decrease in fuel consumption due to elimination of unnecessary maneuvers during parking.

1.1 Objectives of the Project

The goal of the project is to create a prototype for the system that allows the vehicle with a trailer to find empty parking area and to park on its own, and to make it easier for drivers to park their vehicles without any human factor and reduce the chance of an accident during parking process caused by human factor. With the help of the developed hardware and software, it is possible to calculate whether there is enough parking space to enable parking and to park the vehicle with the help of sensors on the vehicle without any intervention of the driver when sufficient space is available. Parking process will be ended when trailer of the vehicle has fully entered the established parking area.

These above goals raise the following core project objectives:

- To use variety of sensors in order to collect distance data from changing environmental conditions,
- To find empty space that is suitable for size of the vehicle to park with the aid of data collected from sensors,
- To use microcontroller to analyze the data ,
- To calculate angle between vehicle and trailer,
- To develop an algorithm to convert the obtained angle value into necessary motion information,
- To use microcontroller to control motors and thereby ensuring the necessary motion,
- To form an parking algorithm,
- To stop the system in case of any obstacle,

1.2 Literature Survey

From the beginning of the development to today, automatic parking systems are always attractive because they increase the safety of driving and increase significantly the quality of life. Today's automatic parking systems are very close to a fully self-parking system. Intelligent parking systems can detect the size of the parking area by combining various sensors, can notify the driver whether it is an appropriate size, and can park the vehicle.

Planning the direction between the initial point and end-point of the vehicle is the key problem of the accident-free parking. For this purpose, more than one methods can be employed. For example, Demirli and Khoshnejad investigated the parallel parking problem at interruption-free environment, so that with the help of sensors, parking in one backward movement is provided. In addition, they have used behavior-based Neuro-Fuzzy control strategy to estimate the angle of the vehicle during the parking motion [10]. This type of control strategy has applied for similar approach in our project in the first prototypes. Although this motion control can be suitable for interruption-free environment, motion in environments containing any obstacles must be handled with careful consideration. Additionally, Martinet, Kermorgant, Dominguez-Quijada and Pérez-Morales used a path-planning algorithm that is built around sensors, they stated that use of sensors to scan environment to form a motion path has a great efficiency [11]. Despite successful results of their work, it should not be forgotten that a single sensor should not be trusted when determining the path of motion. Because of this reason, multiple sensor types suitable for various environmental conditions must be employed in math-planning process. For these reasons, we used a variety of sensors to detect any obstacle that could be in different light, temperature, color and transparency conditions.

On the other hand, Gomez-Bravo,Cuesta and Ollero have done their work in both parallel and cross-park, and have tried to park in a single maneuver instead of multiple maneuvers in order to avoid collisions during parking with using decision of an arrangement of vector which cause a closed-path, but they stated that the need of a big parking space for successful

parking operation as weakness of this work [12]. This maneuver could be used for a vehicle that has no trailer; on the other hand, it could not be used for vehicles with a trailer that is longer than other vehicles. That is why in our project we used multiple maneuvers to complete parking process. Furthermore, Lee,Wei and Guo stated an argument that sensors have a problem with detection of empty parking space on both sides, so they used a laser camera to prevent insufficiency of sensors. As a result, the system they created manage to park the vehicle securely and quickly [13].

Manipulation of the car movement during the parking process is another complication. Speed of the vehicle and angle change depending of this is a very delicate calculation to prevent collusion during parking. During calculation of the speed and angle change, path that will be followed by the vehicle must be calculated. Paromtchik and Laugier used sinusoidal reference functions to develop an algorithm for parallel parking. They concluded that with these functions steering motion and speed of the vehicle could be determined correctly [14].

2. REQUIREMENTS SPECIFICATION

To be able to create a prototype for the system that allows the vehicle to find empty parking area and to park on its own the following requirements are ensured. In this section, formulas and their intended use are described.

2.1 Physical Requirements

Total weight of the system cannot exceed 10 kg. The dimensions of the platform on which the system will be installed cannot go beyond 26 cm × 16 cm. The platform should be made of metal alloy for good heat dissipation. Furthermore, platform should have openings on it for the purpose of robot components can be mounted onto it and the purpose of to increase heat dissipation. The vehicle must have two fixed back wheels and a wheel that can rotate 360 degrees on the platform where it is mounted. Two fixed back wheels must be mounted so that

they do not interfere with each other's movement in the rear half of the platform. Front caster wheel must be mounted in the front half of the platform, so that it could turn 360 degrees around itself. The gears to be used on the rear wheels must be the number of gears that will increase the torque. The diameter of the wheels used in main platform shall not exceed 7 cm. Dimension of the trailer system should be 20cm×10cm and there must be holes in which the wheels can be fitted. The distance and height between trailer and vehicle shall not prevent the installation of encoder that is used for encoding the angle information between the vehicle and the trailer.

2.2 Performance and Functionality Requirements

The system should be able to detect obstacles between 2 cm to 400 cm and must have an average forward speed of 1 cm/second. On the occasion that if system encounters with any obstacle during its movement, it should stop 10 cm away from the obstacle. The system will convert sound waves into distance measurements with a maximum accuracy of 3%. In addition, system will be able to detect obstacles on all four sides of platform. The system should not skid during movement. The angle of rotation of the system must be between 0 and 180 degrees. The trailer attached to the main platform could move without restriction.

Ultrasonic sensor which is used to detect distance data from environment should be able to measure distance between 3 cm to 400 cm and should have the ranging accuracy up to 1 cm. Angle of the measurement must be up to 15 degrees. Sensor should have working frequency of 40 kHz and should be able to provide trigger input signal of 10 uS. Additionally, it must work under 5 V and 15 mA. In addition, it to be used must produce adequate number of sound waves that is used to determine distance in 10 uS.

Infrared sensor, which is used to detect distance data from environment, should be able to assess space between 10 cm to 80 cm. It must have an operating voltage of 5V and must

have an average current consumption of maximum 30 mA. In addition, update period of sensor needs to be maximum of 45 ms.

The microcontroller should have operating voltage of 5V and should be able to support of current of minimum 1000 mA. Furthermore, microcontroller must have at least 24 digital I/O pins and 4 analog I/O pins and be able to support a memory of minimum 12 kB. Furthermore, microcontroller must have built-in over current and over voltage protection.

Stepper motor should be able to operate at 5V and must maximum current of 400 mA. Stepper motor should have minimum detent torque of 100 g-cm to prevent skidding and holding torque of 1080 g-cm.

Stepper motor driver must be able to supply continuous current of 1A to 2A per phase. Logic voltage of driver must not exceed the logic voltage of 5V. In addition, it should be able to provide variety of microstep resolutions. To protect the stepper motors, it must have over-temperature, under-voltage and over-current protections.

2.3 Environmental Requirements

The system should be able to operate at an ambient temperature range of -10 ^{o}C to 85 ^{o}C. The system shall not generate vibration at a level that could be damaging to its performance or that of other equipment.

In addition, sensors used in the system need to be not affected by transparency and color of objects. Likewise, dark environments must not disturb the functionality of the sensors. They must not give incorrect readings in environments with heavy build-up dirt, dust and high-moisture. In addition, they should operate in the given temperature range.

2.4 Health and Safety Requirements

The system will not expose humans to unhealthy levels of electromagnetic radiation and will meet conditions from safe operation identified in ANSI Std. C95.1 [15]. The system should stop moving if an obstacle is detected in the path.

2.5 Manufacturability and Maintainability Requirements

The system will have a modular design such that failed components can be replaced by a technician in under 10 minutes. Each component will be found easily in the market.

3. STANDARDS

The standards which most likely to affect the design and the requirements. The list below gives some of the types of standards that will be employed in a project and included in the requirements.

3.1 Safety

The project was created under safety conditions. Robotic elements are used in this project. Therefore, ISO 10218-2: 2011 [16] standards are taken into consideration. In this project, we used this standard to remove or reduce the fundamental hazards and hazardous situations that arise in the design.

ISO 10218-2:2011 (*Robot and robotic devices – Safety requirements*) describes the basic hazards and hazardous situations identified with these systems, and provides requirements to eliminate or adequately reduce the risks associated with these hazards.

3.2 Testing

This project has passed through several stages of testing. These stages are;

1. Stepper motor configuration
2. Distance sensors accuracy
3. Drivers' configuration
4. Parking sensitivity without trailer, and with trailer

While these test steps are in progress, ISO / IEC / IEEE 29119 [17] standards are taken into consideration. , we used this standard to be internationally accepted software testing that can be used in any software development life cycle or organization.

ISO/IEC/IEEE 29119 (*The International software Testing Standard*) Software Testing is an internationally agreed set of standards for software testing that can be used within any software development life cycle or organization. By implementing these standards, you will be adopting the only internationally recognized and agreed standards for software testing, which will provide your organization with a high-quality approach to testing that can be communicated throughout the world [18].

3.3 Communications

This project is created by using Universal Synchronous/Asynchronous Receiver/Transmitter (USART) communication protocol. USART provides the microcontroller with the interface necessary for communication with the computer. After the algorithm and coding process are established, UART is

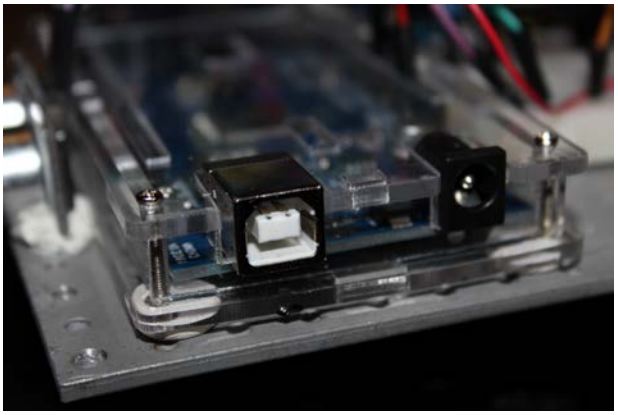

Figure 1. Arduino MEGA Communication Ports

employed to transfer this information between computer and microcontroller. The system's microcontroller have three UART for communicate to the any microcontroller. UART's TTL standards are chosen according to efficiency and necessity of the system.

3.4 Documentation

We have made this thesis and the documentation of the project according to the standards of the IEEE [19]. IEEE's standards of documentation can be found at their websites.

3.5 Design Methods

This project is formed with the aid of some electronical and mechanical components. Driving range, sensitivity, weight, size and price are all considered when selecting the vehicle and the sensors. Moreover, the transmission and communication characteristics of the selected add-ons were examined and a microcontroller with sufficient processing power was selected. Again, while choosing this microcontroller; Attention has been paid to whether or not you have enough Input / Output pins, the interface of the sensors, the status of the motor driver cards, or the integrated operation, size, weight and price. The system adopts IEEE 754-2008 [20] as a design method standard.

IEEE 754-2008 (*IEEE Standard for Floating-Point Arithmetic*): This standard specifies inter-change and arithmetic formats and methods for binary and decimal floating-point arithmetic in computer programming environments. This standard specifies exception conditions and their default handling. An implementation of a floating-point system conforming to this standard may be realized entirely in software, entirely in hardware, or in any combination of software and hardware. For operations specified in the normative part of this standard, numerical results and exceptions are uniquely determined by the values of the input data, sequence of opera-tions, and destination formats, all under user control.

3.6 Programming Languages

The system uses Arduino IDE compiler which have own language. The Arduino programming language is high level of C/C++. In the project, Arduino language is used to code the ultrasonic distance sensors and the sensitivity of trailer. To code the stepper motor and IR distance sensor, the system needs to perform mathematical operations. For this operations C libraries was necessity. These codes are programmed according to *ISO / IEC 9899: 2011* [21] standards.

ISO/IEC 9899:2011 specifies the form and establishes the interpretation of programs written in the C programming language. It specifies;

- The representation of C programs;
- The syntax and constraints of the C language;
- The semantic rules for interpreting C programs;
- The representation of input data to be processed by C programs;
- The representation of output data produced by C programs;
- The restrictions and limits imposed by a conforming implementation of C [22].

4. PATENTS

- Automatic parallel parking system, Patent No: US4735274A [23]
- Automatic parking device for automobile, Patent No: US4931930A [24]
- Automatic driving apparatus, Patent No: GB2304934B [25]
- Process and apparatus for assisting in the parking of a motor vehicle, Patent No: US6097314A [26]
- Procedure and device for an aiding in parking an automobile, Patent No: FR2771500A1 [27]
- Method and device to assist in displacement of a vehicle especially for the park Patent No: FR2785383B1 [28]

- Assistance device for controlling the moving of an automotive vehicle, especially in view of its parking, Patent No: EP1043213B1 [29]
- Parking assistance device and method, Patent No: EP1050866B1 [30]
- Automatic steering system for an unmanned vehicle, Patent No: US5923270A [31]
- An arrangement for guiding steering to assist parallel parking, Patent No: GB2357743A [32]
- Vehicle autonomous parking system, Patent No: US20170329346A1 [33]
- Parking autonomous vehicles, Patent No: US20150346727A1 [34]
- Method for performing autonomous parking process of motor vehicle e.g. passenger car, involves storing target position and/or last driven trajectory of vehicle in suitable device prior to start of autonomous vehicle parking operation, Patent No: DE102012008858A1 [35]
- Identifying cost-effective parking for an autonomous vehicle, Patent No: US20150241241A1 [36]
- System and method for enabling an autonomous vehicle to track a desired path, Patent No: US5684696A [37]
- Semi-autonomous parking control system for a vehicle providing tactile feedback to a vehicle operator Patent No: US5742141A [38]

5. THEORETICAL BACKGROUND

A number of formulas must be used to check the correctness of the requirements given in the system requirements and to correct them if necessary.

5.1 Ultrasonic Distance Sensor HC-SR04

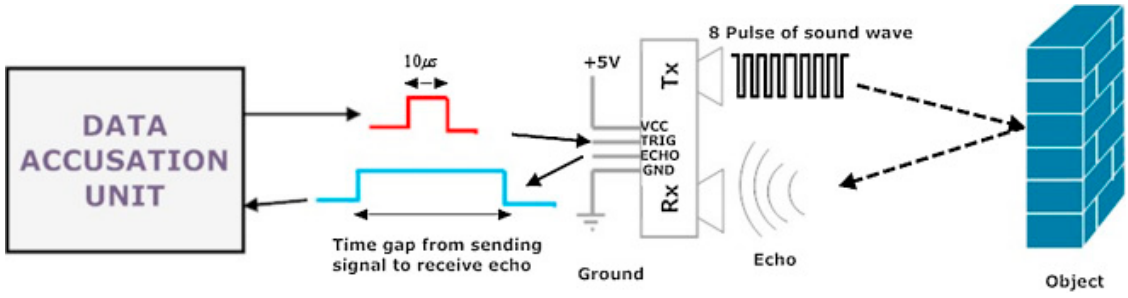

Figure 2. Working Principle of HC-SR04 [52].

For the purpose of calculating the distance between vehicle and obstacle, the HC-SR04 ultrasonic distance sensor is selected. This sensor can send a ping at a given moment and receive the ping rebound on an obstacle at another given moment. The ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when any obstacle objects it, it is reflected back towards the sensor, this reflected wave is observed by the ultrasonic receiver module.

According to HC-SR04 datasheet, the transmitter on sensor emits eight bursts of a directional 40 KHz ultrasonic wave when triggered. When the HC-SR04 sends the ultrasonic sound waves from transmitter to environment, the Trigger Pin goes from 0 V to 5 V and waits to sound wave reflects through an object. If sound waves are reflected, these waves hit the receiver. Transmitting and receiving time is used to calculate distance between obstacle and sensor. The following formulas are used to calculate this distance.

The general formula of the distance is given as;

$$d = C \times t \tag{1}$$

Where d is distance in meters, C is speed in meters/second and t is time in seconds.

The sensor transmits a ping at a specific time and receives the rebounded ping at another specific time. The difference between these time values gives an idea of the distance of an object. The following formula is used to calculate this time difference

$$\Delta t = t_2 - t_1 \quad (2)$$

Where t_1 is the time when the sensor transmits a ping and where t_2 is the time when the sensor receives the transmitted ping.

It should be noted that this time difference includes the sound wave generated by transmitter hits the obstacle and reaches the receiver. This means total value is combination of outgoing and incoming sound waves, but distance information should only cover the outgoing sound wave. For this reason, to measure the exact distance value between the sensor and the obstacle, the calculated time difference should be divided into two. Thus, equation (2) becomes the following form.

$$\Delta t = (t_2 - t_1) \div 2 \quad (3)$$

To be able to use equation (1), it is also necessary to know the speed. According to Zitzewitz, "The speed of sound in air depends on the temperature, with the speed increasing by about 0.6 m/s for each 1°C increase in air temperature. At room temperature (20°C), sound moves through air at sea level at a speed of 343 m/s. [39]". The speed of sound in air is generally calculated by the formula:

$$C_{air} = (331.296 + 0.606 \times T_{air}) \quad (4)$$

Where C_{air} is the speed of sound in air and T_{air} is the temperature of air in °C. 331.296, the speed at 0°C, is form on heat capacity ratio, γ, on 1 atm [40].

By using equations (1), (3) and (4), the generalized formula for distance calculation can be found as:

$$d = \frac{\Delta t}{2} \times C_{air} = (t_2 - t_1) \times (165.648 + 0.303 \times T_{air}) \quad (5)$$

5.2 Infrared Distance Sensor GP2Y0A41SK0F

The GP2Y0A41SK0F Infrared distance sensor is preferred for determination of the gap between vehicle and obstacle. This sensor uses infrared light to determine the distance data. Transmitter on sensor transmits an infrared light and if obstacle is in range of sensor, light is reflected to receiver of the sensor. According to intensity of the light reflected, distance value is determined. However, these distance value innately non-linear. A huge change in output voltage does not always mean a huge change in distance to an object. In order to get distance value, sensor output voltage must be converted to a distance value using series of function.

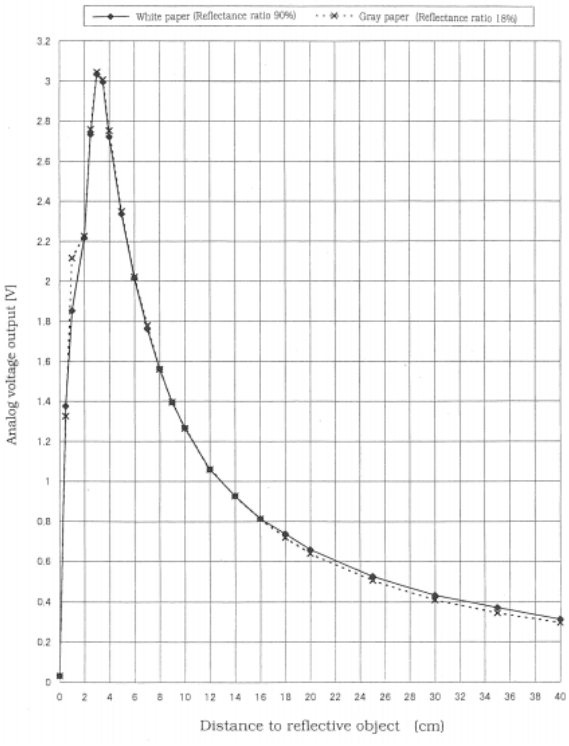

Figure 3. GP2Y0A41SK0F - Analog Output Voltage vs. Distance to Reflective Object [41].

The method to follow is to convert the analog output voltage to linearized voltage values by using approximation methods. This is done with taking small pieces from each segment of respond and converting them to small straight lines. However, this method must be applied differently for each sensor as each sensor has a unique response of output voltage vs. distance.

The sensor output voltage as a function of a distance to an obstacle is estimated by the following formula:

$$V = \frac{1}{(R + k)} \quad (6)$$

Where V is voltage, R is range and k is a constant, which is different for each sensor, luckily, the equation (6) generates a very straight line for output voltage. The inversion in equation (6) works as a linearizer function. However, as previously said, constant k is different for each sensor. That is why this constant must be found for our sensor. According to GP2Y0A41SK0F datasheet, this constant must be 0.42 [41]. Thus, the equation (6) becomes:

$$V = \frac{1}{(R + 0.42)} \tag{7}$$

After approximation, following stage is to find a straight-line approximation that correlates the voltage to the linearizing function, shown in equation (7). Finding suitable constants in basic line equation, which is given in equation (8), is done by using the voltage variable of equation (7) instead of y variable of line equation because y is equal to the linearized distance. These steps are shown below.

$$y = m \times x + b \tag{8}$$

$$\frac{1}{(R + 0.42)} = m \times V + b \tag{9}$$

Reorganizing the equation (9) terms gives range as a function of voltage, as shown:

$$R = \frac{1}{(m \times V + b)} - 0.42 \tag{10}$$

Finding the m and b variables of equation (10) is done by using the data from the calibration stage of the sensor. Figuring out the reversion of the linear data generates the m and b variables. Finally, equation (11) is obtained from the all work done above.

$$R = \frac{5461}{(V - 17)} - 2 \tag{11}$$

5.3 Stepper Motor LB82773-M1

In order to use the stepper motor in the most efficient way, a number of calculations should be made. First of all, the number of gears affects the operation of whole system. What should be noted here is that externally used gears affects the step angle of the system.

Each stepper in the system has a three external gears which are consists of 72-35-20 teeth. These gear systems are interact with the stepper motor's internal gear system. The relationship between gears and the effect of these gears on the step angle is given as follows:

$$R_{12} = \frac{N_1}{N_2} \tag{12}$$

Where R is the gear ratio, and where N_1, N_2 and N_3 are number of teeth on gears, respectively. If we use the equation (12), we can find the gear ratios between 72-teeth, 35-teeth gear and 20-teeth gears as:

$$R_{12} = \frac{72}{35} = 2.057142286 \tag{13}$$

$$R_{23} = \frac{35}{20} = 1.75 \tag{14}$$

With the help of the equation (15), total gear ratio of the external gear system is found at equation (16).

$$R = R_{12} \times R_{23} \tag{15}$$

$$R \ = 2.057142286 \ \times 1.75 \ = 3.5999 \tag{16}$$

Also, the stepper motor of the system has a specification that each step takes 7.5 degrees. Equation (17) is used to find how many steps are needed to turn the rotor of the stepper motor one revolution:

$$S = \frac{360}{\theta_{step}} \times R = \frac{360}{7.5} \times 3.5999 \cong 173 \ steps \tag{17}$$

Where *S* is the steps per revolution θ_{step} is the step angle and *R* is the gear ratio.

In addition, it is necessary to calculate the circumference of the wheel in order to determine how much angle is taken at each step. To do that basic circumference formula given in equation (18) and equation (20) are used, also results are given in equation (19) and equation (20).

$$\mathrm{c} = 2 \times \pi \times r \tag{18}$$

$$\mathrm{c} = 2 \times \pi \times 3.5 \cong 22cm \tag{19}$$

$$deg = \frac{360}{S} \tag{20}$$

$$deg = \frac{360}{173} = 2.080925° \tag{21}$$

To find how many steps stepper motor travels at one centimeter, one revolution is divided into circumference of the wheel. Then, result is divided into result found in equation (21). Formulization and corresponding results are given in equations (22) and (23).

$$\text{s}= ({}^{360}/_{c}) \div deg \tag{22}$$

$$s = \left(\frac{360}{22}\right) \div 2.080925 = 7.867 steps/cm \tag{23}$$

Where *s* is steps taken at each centimeter. Also the system travels 0.1271676389 centimeters at each step.

Stepper motors' area of use is generally for positioning. Stepper motors are made to work precise, and be able to stop at exact positions, so they are build for torque, not speed. However, when you speed up the stepper motor, they lose torque proportional to increase in speed. That is why the maximum speed that stepper motor can reach is also important. This value is inversely proportional to the motor's steps per revolution value. Maximum speed stepper motor can gain is given in equation (24).

$$\text{V}_{max} = \frac{V}{2 \times L \times I_{max} \times S} \tag{24}$$

Where V_{max} is maximum speed, V is applied voltage, L is stepper motor inductance, I_{max} is maximum current and S is steps per revolution. According to equation (24), maximum speed our stepper motor can reach is shown in equation (25).

$$\text{V}_{max} = \frac{5V}{2 \times 6.25\Omega \times 0.8A \times 173} = 2.89\, rev/sec \tag{25}$$

At the same time, minimum time for energizing the motor's windings during the each step is very important. According to this value, stepper stepping and delay time in microcontroller is decided during the project. Equation (26) shows minimum time per step formula, also result is given in equation (27).

$$t_{min} = \frac{2 \times L \times I_{max}}{V} \tag{26}$$

$$t_{min} = \frac{2 \times 6.25\Omega \times 0.8A}{5V} = 2ms \tag{27}$$

Where t_{min} is the minimum time per step, L is stepper motor inductance, I_{max} is maximum current and V is applied voltage to run the stepper motor. In addition, maximum power spent on stepper motor is given in equations (28) and (29).

$$P_{max} = I_{max} \times V \tag{28}$$

$$P_{max} = 0.8A \times 5V = 4W \tag{29}$$

Where P_{max} is maximum power spent, I_{max} is maximum current, V is applied voltage.

5.4 Stepper Motor Driver A4988

For using the stepper motors properly, appropriate stepper motor driver must be used. According to datasheet, "The A4988 is a complete micro-stepping motor driver with built-in translator for easy operation [42]". Although driver can support a voltage level up to 35 V, we use 12V supply to power up the driver. However, this does not mean we do not encounter with losses due to heat dissipation.

To calculate the thermal efficiency of the stepper motor driver, conduction losses and switching losses must be found. A4988 uses two DMOS Full Bridge to controller stepping of the stepper motor. This means each transistor has a resistance due to "on" state of drain to source legs. Thus, resistance on drain-to-source causes thermal dissipation. Equation (30) shows how to calculate this dissipation.

$$P_{DSRON} = R_{DSON} \times {I_{LOAD}}^2 \tag{30}$$

Where P_{DSRON} is power during the on stage of drain-to-source, R_{DSON} is the resistance and I_{LOAD} is the current.

To make a time difference between each stepping, capacitors along the whole driver is used [42]. These capacitors causes switching losses during the timing operation. These losses can be calculated using equations (31) and (32) [43].

$$P_{SWavg} = I_{LOAD} \times \frac{V_{BAT}}{{t_{SW}}^2} \times \int_{t_0}^{t_0+t_{SW}} t \times dt \tag{31}$$

$$P_{SWavg} = \frac{I_{LOAD} \times V_{BAT}}{2} \tag{32}$$

Where P_{SWavg} is average power dissipation during switching, V_{BAT} is voltage supplied via battery and t_{SW} is time of switching.

According to datasheet of the A4988 to find the maximum value of the current limiting, "The maximum value of current limiting is set by the selection of R_{Sx} and the voltage at the V_{REF} pin [42]". Equation (33) shows how to find this current value.

$$I_{TripMAX} = \frac{V_{REF}}{(8 \times R_S)} \quad (33)$$

Where maximum value of current limiting is shown by $I_{TripMAX}$, reference voltage level is shown by V_{REF} and the resistance of the sense resistor is shown by R_S.

6. METHODOLOGY

The analysis of the principles of inquiry in assembling the building blocks of the prototype is investigating in this section. Prototypes contain both hardware and software in terms of performing the finding the empty parking spot and parking process. For this reasons, this section is divided into three parts and each section is clarified in itself. System hardware, software and tools which used to construct the system is described in Section 6.

6.1 System Hardware

Automatic Perpendicular Parking of Car-trailer System uses multiple components to purpose of realization. The system uses both hardware and software to consolidate the autonomous parking of a car-trailer system. Hardware is used to purpose of gathering distance data from environment, analyzing the data collected from the environment and providing movement according to the results of analysis.

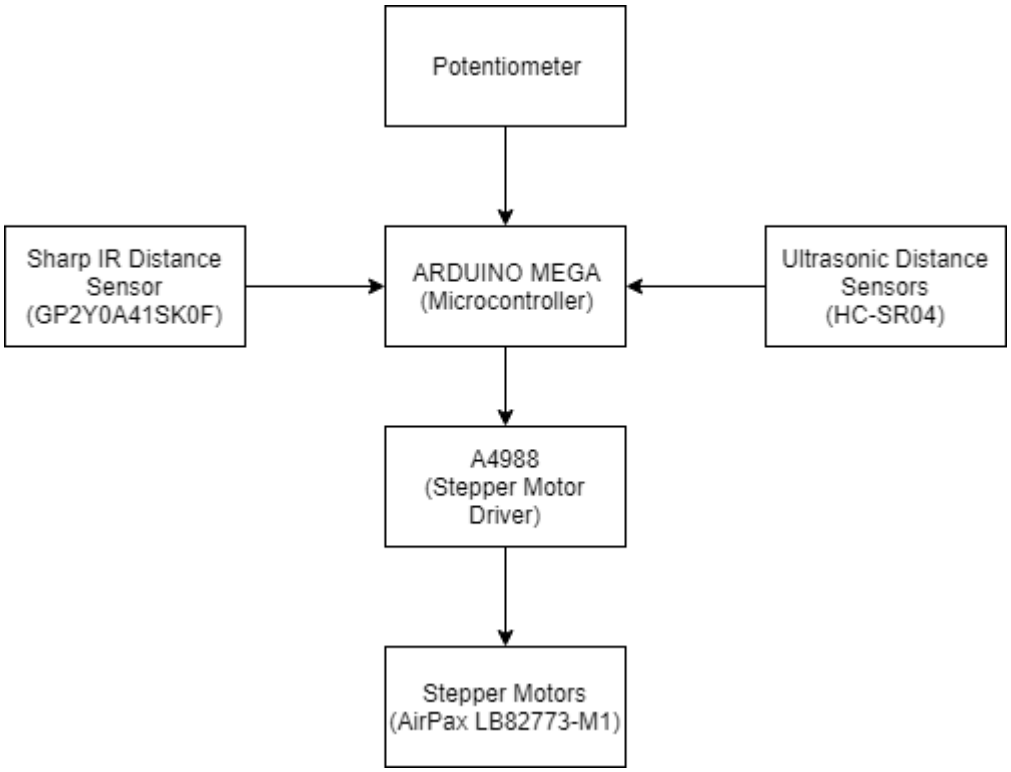

Figure 4. System Hardware Diagram

Hardware of the system is built with the cooperation of sensors, potentiometer, stepper motor drivers and bi-polar stepper motors. Sensors are used to collect the distance data which is used to determine whether there is an obstacle in front of the system. Potentiometer is used to provide angle information between car-like robot and trailer. Likewise, stepper motor driver and bi-polar stepper motor is used to provide movement of the system according to algorithm mentioned in the system software section. The methodology used in the creation of hardware and software given below, respectively.

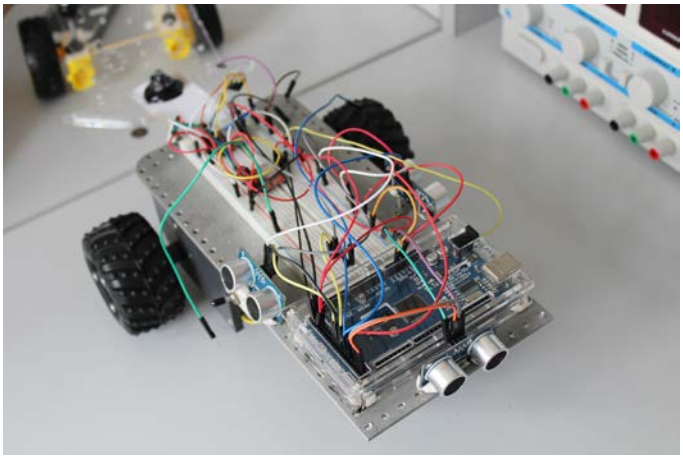

Figure 5. Automatic Perpendicular Parking of a Car-trailer System Hardware

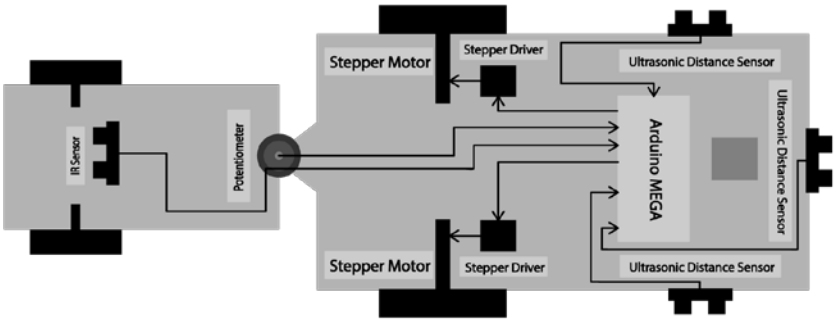

Figure 6. System's Hardware Placement

6.1.1 Step Motor

Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time.

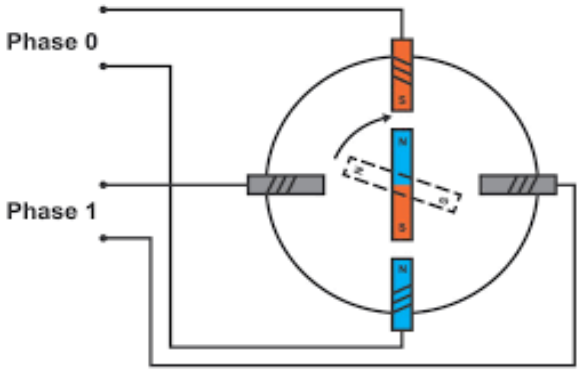

Figure 7. Bi-polar 2 Phase Stepper Motor Diagram [44]

In general, the robotic systems and needing precision systems require a sensitive motion, so these systems need step motor for controlling each step. The robot whose mission is to

autonomous parking robot needs each steps and each motions are controlled, so the stepper motors are used in this project.

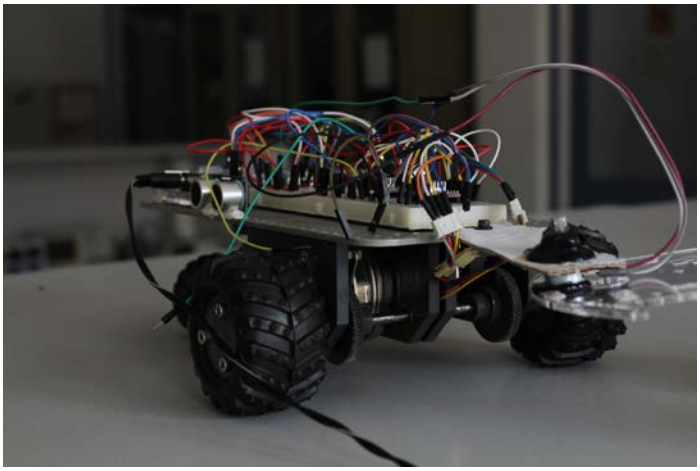

Figure 8. Bipolar 2 Phase Stepper Motors

The system uses two bi-polar two-phase step motors, AirPax LB82773-M1, for motion. When choosing a stepper motor, it was noted that the stepper motors must have torque that can carry the weight of the system. In addition, the stepper motors is selected based on the current-draw capability and the maximum amount of current the microcontroller could provide. Since per step made by stepper motors determines the precision of the process, the stepper motor is chosen which can provide the precision that is necessary to the system. These motors are used in motion for the driving mode and parking mode. When the sensors do not detected an obstacle or empty parking area, the stepper motors moves at equal speed and in equal steps. If the sensors detects any obstacle, the stepper motors stops and process of parking is stopped. If the sensors detect any empty parking area (30cm × 60cm), the robot moves according to robot parking algorithm.

6.1.2 Sensors

Sensor selection is made in this project to ensure the necessary sensitivity and adapt to changing environmental conditions. For instance, the IR sensor supports shorter range, and it is more sensitive than ultrasonic sensor. However, “An ultrasonic sensor's reaction is not subordinate upon the surface shading or optical reflectivity of the article. Detecting of a glass plate, a stoneware plate, a white plastic plate, and a sparkly aluminum plate is the same [45]”, in addition, the ultrasonic sensor cheaper than infrared sensor. Therefore, the infrared sensor is

used to measure the distance at the back of the robot, the ultrasonic sensor is placed the front of robot and sides of the robot.

6.1.2.1 The Ultrasonic Sensor

The HC-SR04 Ultrasonic Distance Sensor is a sensor used for detecting the distance to an object using sonar. It is ideal for any robotics projects you have which require you to avoid objects, by detecting how close they are you can steer away from them.

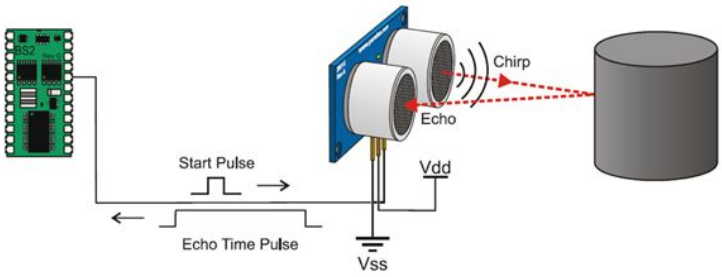

Figure 9. HC-SR04 Working Principle [46]

In this project, the ultrasonic sensors are placed the front of the robot and sides of the robot. The sensor, which is placed the front of the robot, detect the obstacles. If the sensor detect an obstacle at a distance of less than 15 cm, the system stop the system. Others ultrasonic sensors detect a park area (width of 30 cm and length of 70 cm), the system will enter parking mode.

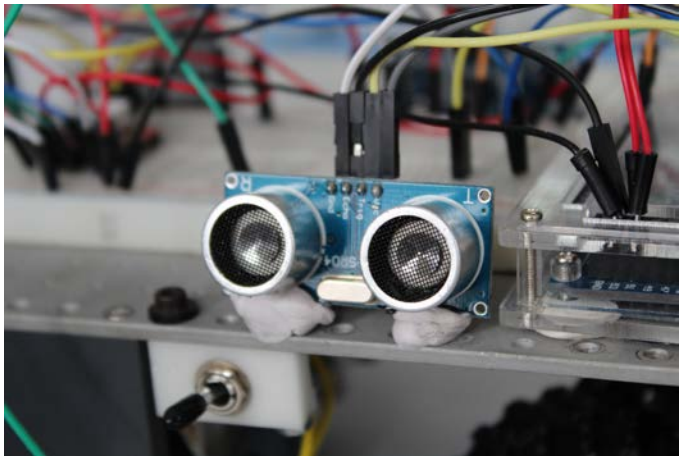

Figure 10. HC-SR04 Ultrasonic Distance Sensor

6.1.2.2 The Infrared Sensor

IR Sensors work by using a specific light sensor to detect a select light wavelength in the Infra-Red (IR) spectrum. By using an LED which produces light at the same wavelength as what the sensor is looking for, you can look at the intensity of the received light. When an object is close to the sensor, the light from the LED bounces off the object and into the light sensor. This results in a large jump in the intensity.

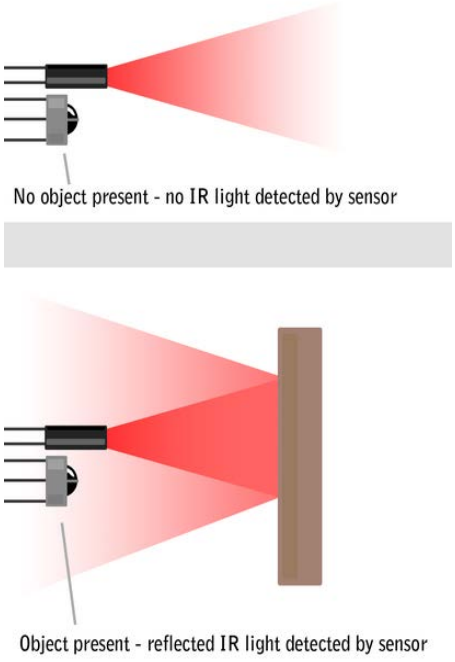

Figure 11. Sharp IR Distance Sensor Working Principle [53]

In this system, sharp GP2Y0A41SK infrared sensor is placed back of the robot for the measure the distance between the obstacle and robot. Cause of this model using in project is, it has the measurement values at the appropriate intervals required. If the infrared sensor measure 5 cm between the obstacle and robot, the system will finish parking.

6.1.3 Potentiometer

According to IEEE, potentiometer is “An instrument for measuring an unknown electromotive force or potential difference by balancing it, wholly or in part, by a known potential difference produced by the flow of known currents in a network of circuits of known electrical constants [47]” and “A resistor with an adjustable sliding contact that functions as an adjustable voltage divider [48].”

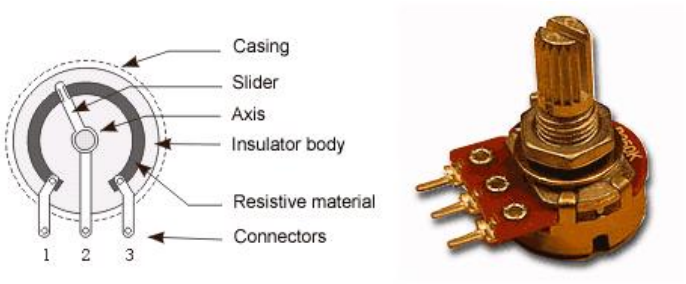

Figure 12. Potentiometer Pin-out [49]

In this project, potentiometer is placed between car-like robot and trailer. Reason why the system uses potentiometer is setting trailer in orientation. The total rotational capacity of the potentiometer is 270 degrees. If a 10 KΩ potentiometer is considered, when degrees is zero, the value of potentiometer is zero, when degrees is 270; the value of potentiometer is 10k ohm, approximately. The value of the potentiometer according to the value of the angle to which this value range will vary varies. Therefore, it will turn right or left in the parking process according to these angle values.

6.1.4 Arduino MEGA

According to SparkFun's Official Website, "Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a "microcontroller") and a piece of software, or IDE that runs on your computer, used to write and upload computer code to the physical board [50]."

The system uses Arduino MEGA as an analyzer. It analyzes data collected from the environment, and according to these data, it decides whether parking algorithm could be used or could not use. Also, it decides the motion of the system. If no obstacle is found, it introduces forward motion to the system with help of the stepper motors. Additionally, it analyzes the data from the potentiometer and uses this data in the process of parking.

6.1.5 Bipolar Stepper Motor Driver

A bipolar stepper motor driver is actually a little current amplifier consists of transistor, because stepper motor requires much more current than microcontroller can provide. Stepper motor drivers modify pulse signal coming from microcontroller into motor action to accomplish correct motion [51].

When deciding the stepper motor driver, the following features are taken into account.

- It should not require more current than the current that microcontroller can provide for,
- It should operate between normal operating voltages,
- It should provide enough current that is required for each stepper motor,
- It can support various micro-stepping resolutions.
- It can provide over current and over voltage protection.

6.2 Software

System realization is assembled with both hardware and software. In the previous section, hardware elements of the system are explained. How the system software is assembled is explained in this section.

Arduino MEGA microcontroller has been used to read and interpret the sensor information, to generate necessary signaling to control the stepper motors and to calculate significant maneuvers for parking on the prototype vehicle. The algorithm developed within the general working principle is based on the use of each individual modules properly to perform successful autonomous parking.

The autonomous parking system used on microcontroller is divided into three separate sections such as obstacle control, searching for empty parking area and parking process. In addition, each sub-section is given as flowchart figures to show the process. In case of accomplishing the parking process successfully, these three steps must be

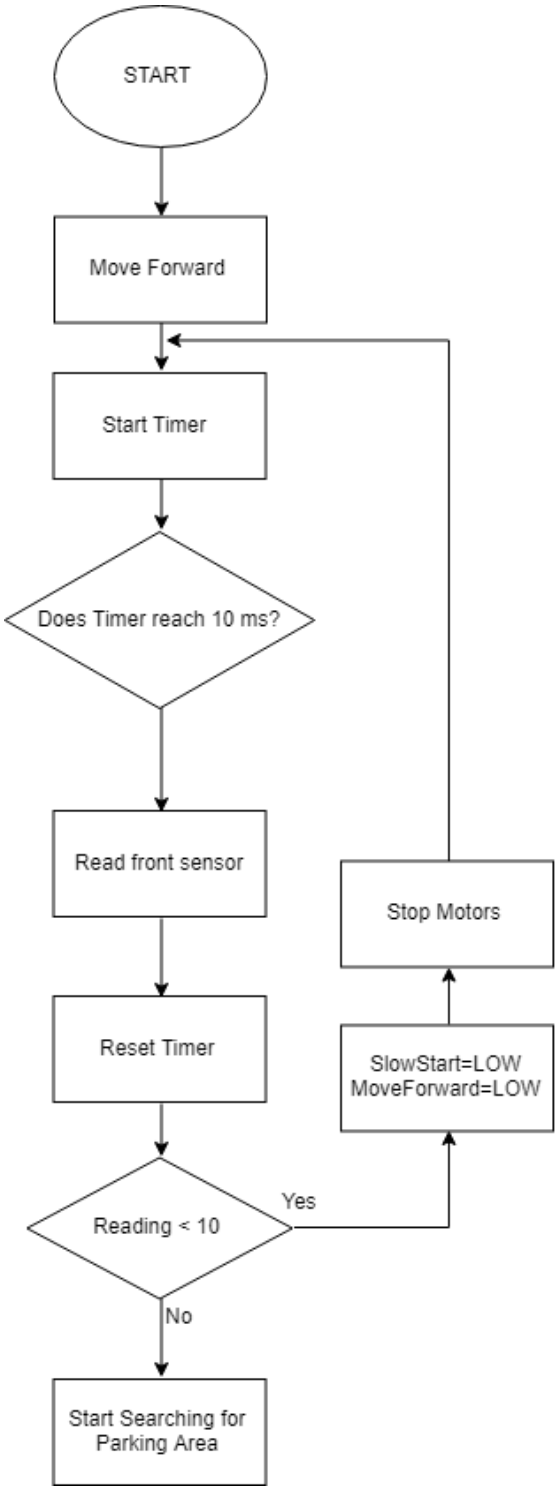

Figure 13. Flowchart of Obstacle Control Algorithm

6.2.1 Obstacle Control

Obstacle control algorithm is based on the operating principle of the ultrasonic distance sensor so that the vehicle stops if it encounters any obstacles during its movement. When the vehicle is autonomously parked, an autonomous brake system is activated when an obstacle is detected by means of an ultrasonic sensor in front of the vehicle.

Since more than one ultrasonic distance sensor is used in the system, timer in microcontroller is used to prevent any communication malfunction between microcontroller and ultrasonic sensors.

In addition, as mentioned in section 2.2, on the occasion that if system encounters with any obstacle during its movement, it should stop 10 cm away from the obstacle. Therefore, if the data read

from the microcontroller is less than 10 centimeters, the motion of the vehicle is paused. The autonomous brake system is a prevention procedure that works to prevent a possible accident or to have the least damage in case of a possible accident.

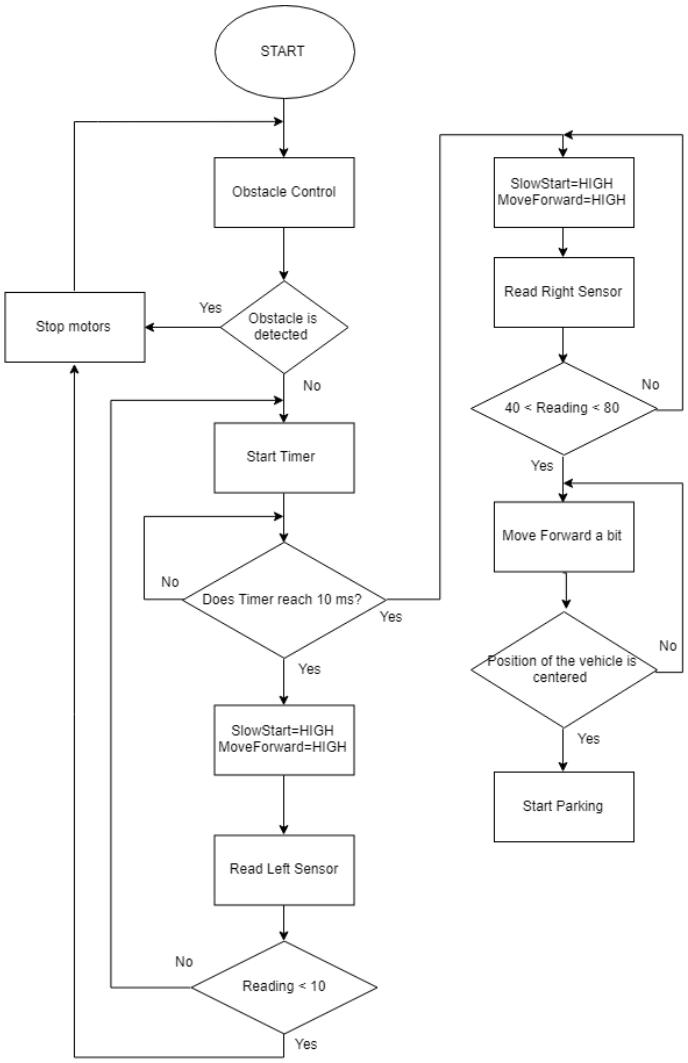

Figure 14. Flowchart of the Searching for Empty Parking Spot

When the vehicle is stopped, the timer of the microcontroller is reset to ensure that the data is read correctly at each time for each sensor. In addition, the value of 10ms is selected according to the value given in datasheet to prevent any failure on distance sensor.

Working principle of the sensor is explained in Section 6.1.2.1 and flowchart of the obstacle detection algorithm can be seen in Figure 13.

6.2.2 Empty Parking Area Search

The availability of an empty parking spot is very important in order to initiate the parking process. Size of the parking spot cannot exceed the dimensions of the vehicle mentioned in Section 2.1. Therefore, algorithm to be used is created in consideration of the width and length of the vehicle.

Obstacle detection is performed as described in the previous section. In case of any obstacle detected, microcontroller restrains vehicle's movement. If the right-hand distance to be parked by the vehicle is greater than 40 cm, the second distance measurement is started. If the second measured distance value is greater than the first value and less than 60 cm, it means that the distance from the obstacle to the gap is reached. After this operation, parking process starts.

Timer of the microcontroller is again used in the algorithm to ensure each sensor reads accurate data. Data is read from both the left and the right sensor every 10 milliseconds to increase accuracy. Acceleration of the start of the movement is provided with the help of the necessary flags. When "SlowStart" flag of the algorithm is set to HIGH, the movement of the vehicle is accelerated. In addition, "MoveForward" flag is used to provide an information on the direction of the movement. Positioning the vehicle at its center after the parking spot is found is done according to information given in Sections 2.1 and 5.3. When the vehicle is centered at the parking spot, parking process begins. Illustration of the finding empty parking area can be seen in Figure 16-(a) and Figure 16-(b). Flowchart of the searching empty parking spot can be found at Figure 14.

6.2.3 Parking Process

After finding suitable parking spot, explained in Section 6.2.2, parking process is started. Parking process consists of multiple maneuvers which are decided according to data read from the potentiometer. However, data from the potentiometer must be converted to digital values because value of the potentiometer is read as analog voltage values by microcontroller. This process is done by using necessary mapping function. Maneuvers to be made by the vehicle are decided according to the angle values from the potentiometer. Assuming that the center value of the potentiometer is 90 degrees, maneuvers are determined according to the angle values between 0 and 180 degrees.

If we acknowledge the fact that the parking space is on the right side of the vehicle and the vehicle parks vertically, in case of angle value read from the potentiometer is equal to 90 degrees or smaller than 90 degrees, the vehicle must move forward to the left. When the angle value drops below 10 degrees, the vehicle stops to reduce the vibration that can affect the angle value read from the potentiometer. The visual representation of this step can be seen from the Figure 17-(c). After first maneuver, angle value is read again. Whenever angle value is greater than 90 degrees, the vehicle starts to move forward to the right. If the angle read after this maneuver is equal to or greater than 135 degrees, it stops again for the same reason. Illustration of this step can been seen in Figure.17-(d) and Figure.17-(e). Third maneuver is used to align the trailer and vehicle by going backwards a bit. Whenever angle value is greater than 175 degrees, the vehicle starts to move to left until angle data equals to or less than 90 degrees. In this way, the vehicle and trailer become perpendicular to the parking spot. The visual representation of these steps can be seen in Figure.17-(g) and Figure.17-(h).

Microcontroller analyzes the distance information received from the IR sensor while the vehicle move backwards. When the distance data becomes equal or less than 5 cm, it stops and finished the parking process. This step is shown in Figure.17-(i). Some photographs from the moment of application of autonomous parking of the developed prototype vehicle are given in Figure 15.

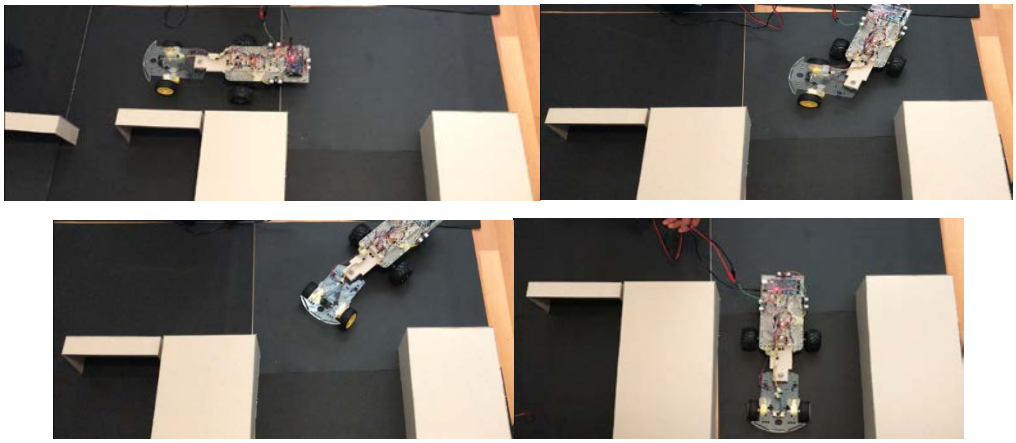

Figure 15. Moment of Application

Flowchart of the parking algorithm, steps of maneuvers and whole system can be found at Figure 16, Figure 17 and Figure 18.

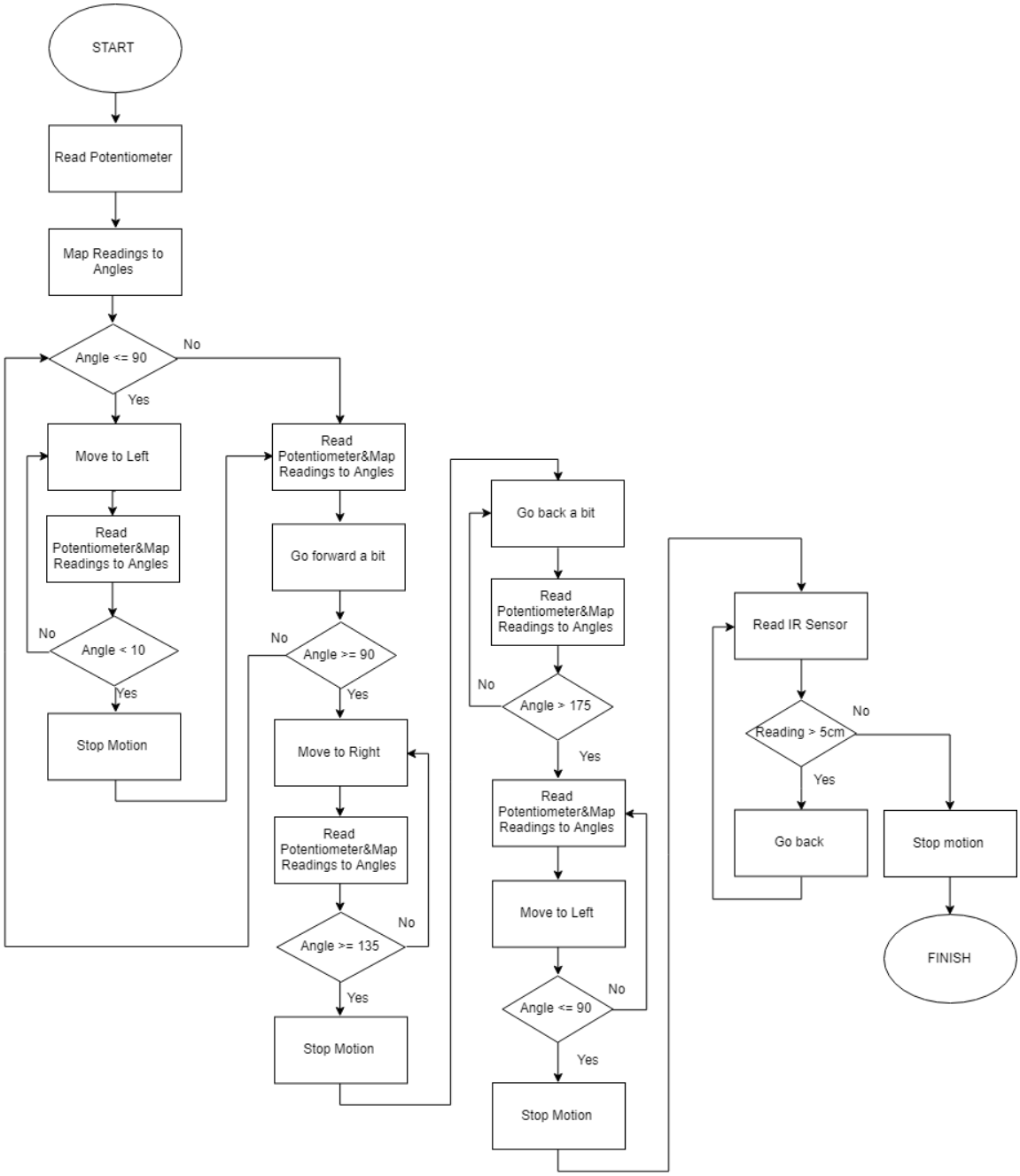

Figure 16. Flowchart of the Parking Process

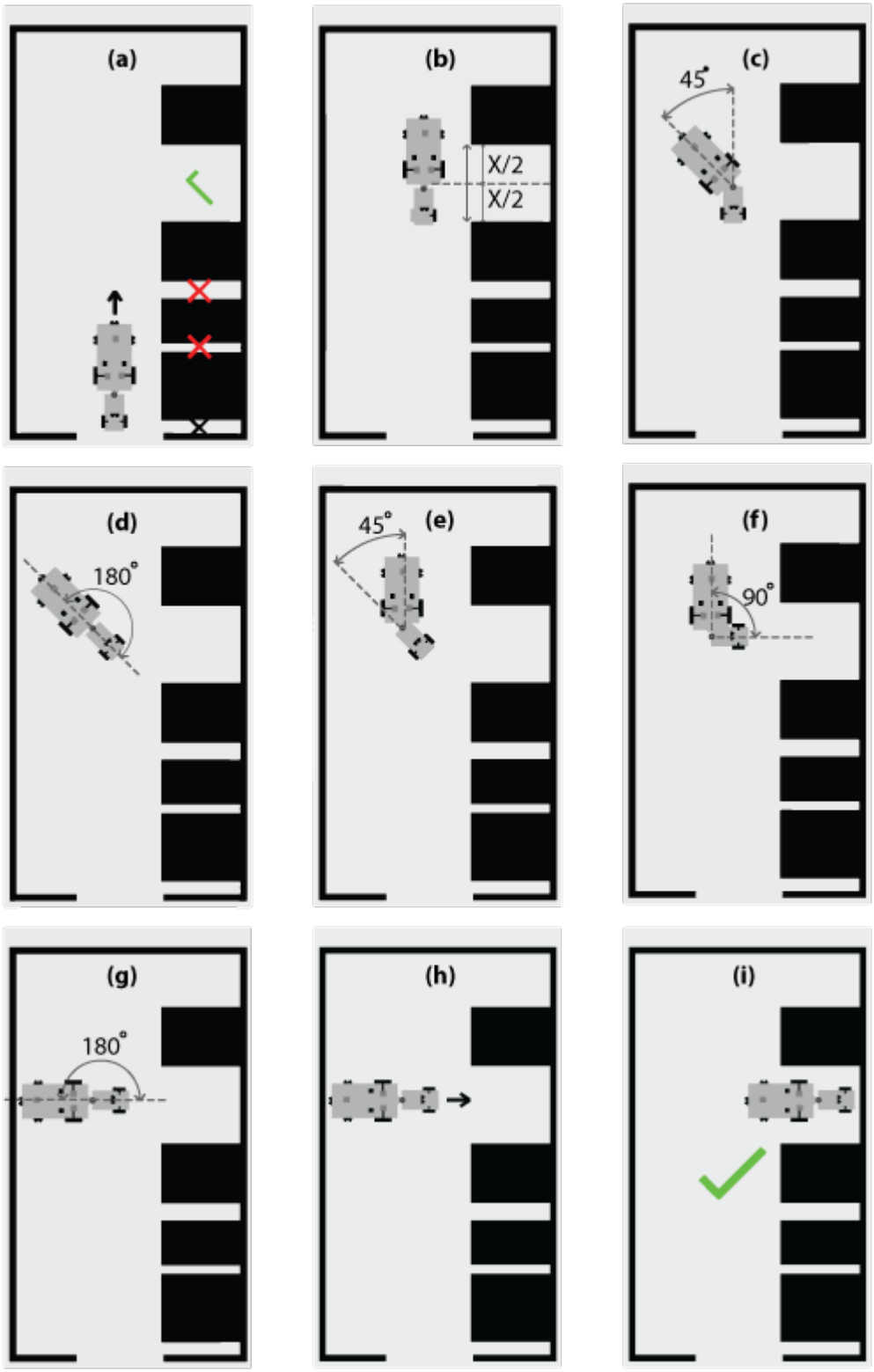

Figure 17. Maneuvers of Parking Process

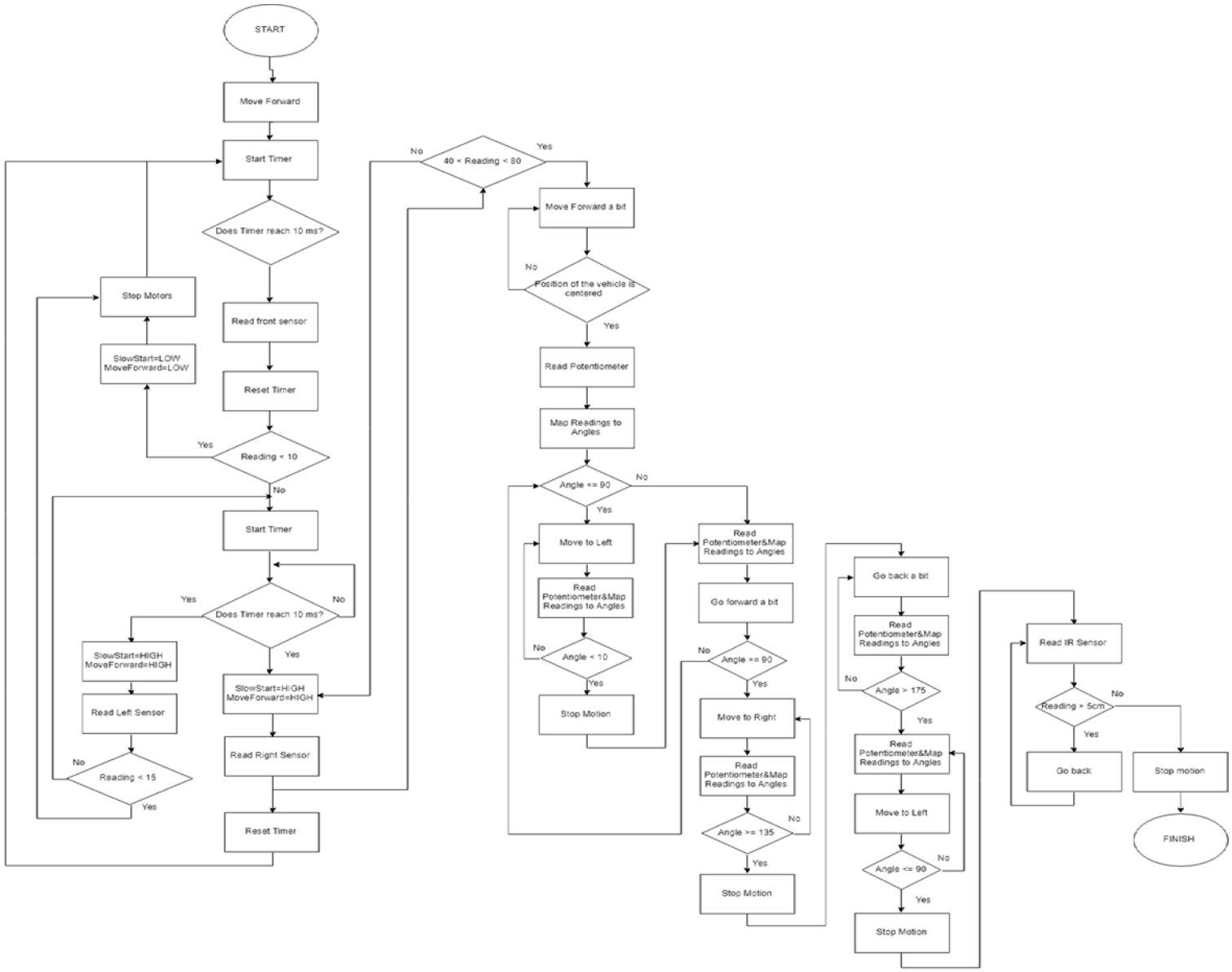

Figure 18. Flowchart of the Whole System

6.3 Tools

- Microsoft Visual Studio
- Arduino IDE
- Gantt Project
- Microsoft Office
- Microsoft Project
- Adobe Illustrator

7. EXPERIMENTS

Requirements for designing a system that grants the vehicle to find empty parking area and to park on its own are given in section two. Necessary experiments have been carried out in this section to see if the system meets these design requirements. Experiments in this section are created with the help of variety of software, mechanical components and electronic components. For a better understanding of the reader, experiment systems are given in schematics and experiment results are clarified in tabular forms and plots.

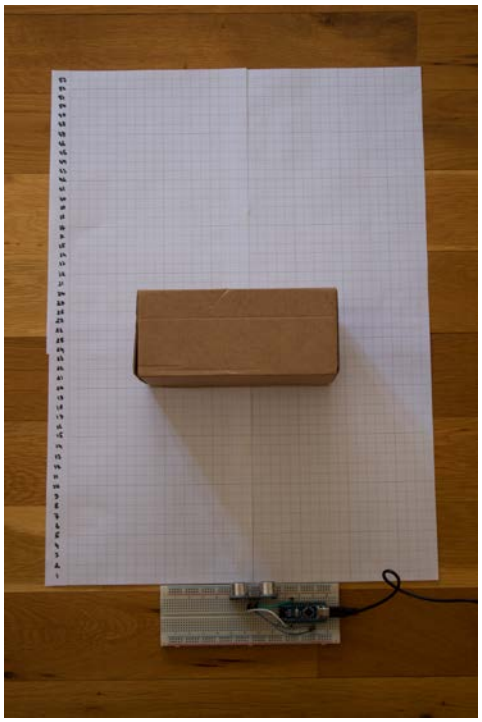

Figure 19. HC-SR04 Ultrasonic Distance Sensor Accuracy Test Against a Fixed Obstacle

7.1 HC-SR04 Ultrasonic Distance Sensor Accuracy Test Against a Fixed Obstacle

The ultrasonic distance sensor used in the system plays a very important role in terms of the measurement of the distance of the vehicle to the obstacle and the choice of the algorithm to be applied according to this distance value.

As mentioned in section 2.2, ultrasonic distance sensor must have a maximum accuracy of 3%. The experiment design presented in this section to test the accuracy of the ultrasonic distance sensor with fixed obstacle at 20cm made use of the breadboard, cables, HC-SR04 ultrasonic distance sensor, Arduino Nano, fixed obstacle, Arduino IDE and Visual C# .NET application (The codes written in the mentioned applications can be found in the Appendix A and Appendix B).

Main purpose of the experiment is to measure the accuracy of the sensor in case of fixed obstacle at 20cm. The reason for the obstacle being fixed at 20 cm is that the length of the area to be used in the motion tests will not exceed 400cm. That is why we decided to make controlled variable, the obstacle, is kept constant at 20cm, so the algorithm is written based on these accepted variables.

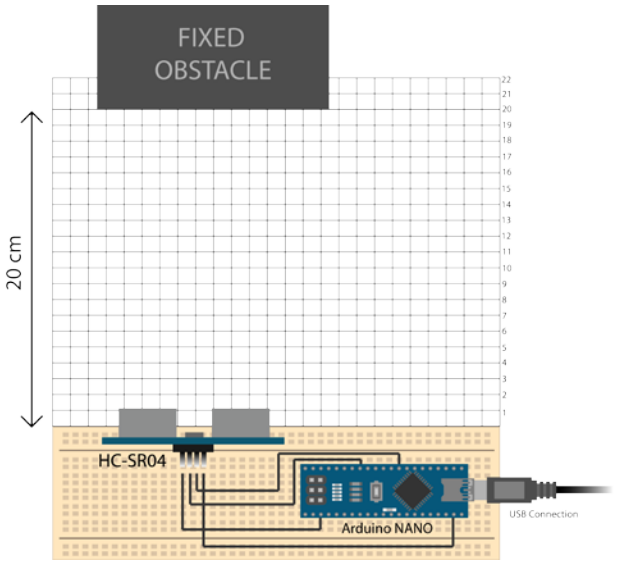

Figure 20. Sensor Accuracy Test Against a Fixed Obstacle - Test Schematic

Dependent variable of experiment is the distance difference between controlled variable and value measured by HC-SR04. This difference was used to find the accuracy of the sensor.

HC-SR04 is used to collect distance data from the environment in experiment. To use HC-SR04, echo and trigger pins of sensor is connected to pin 9 and pin 8 of the Arduino Nano. In addition, V_{CC} and GND pins of sensor is connected to the appropriate pins of the Arduino. The power of the Arduino is provided by USB connection to computer.

The process of getting distance data from the sensor is as follows. Trigger pin is activated via using Arduino Nano. After 10 µS, trigger pin is set to its initial state. Thus, HC-

SR04 is created necessary sound waves to measure distance between sensor and obstacle. Then, echo pin is used to read a pulse when its state becomes HIGH. When there is a sound wave returning to the sensor, timing starts and calculates the return time of the sound wave to the sensor in microseconds. This time is used to calculate distance between sensor and obstacle. The “flag” from the application, which is created with using Visual C #.NET, is waited for the calculated value to be sent to the application. If the flag from the program is the desired flag value, the calculated distance value is transferred to the program via serial port of computer.

Visual C# .NET application is used to convert data from the sensor to graph, tabular form and calculate the accuracy of the sensor. In application, serial port element is used to establish a connection between computer and application. On the other hand, timer element is

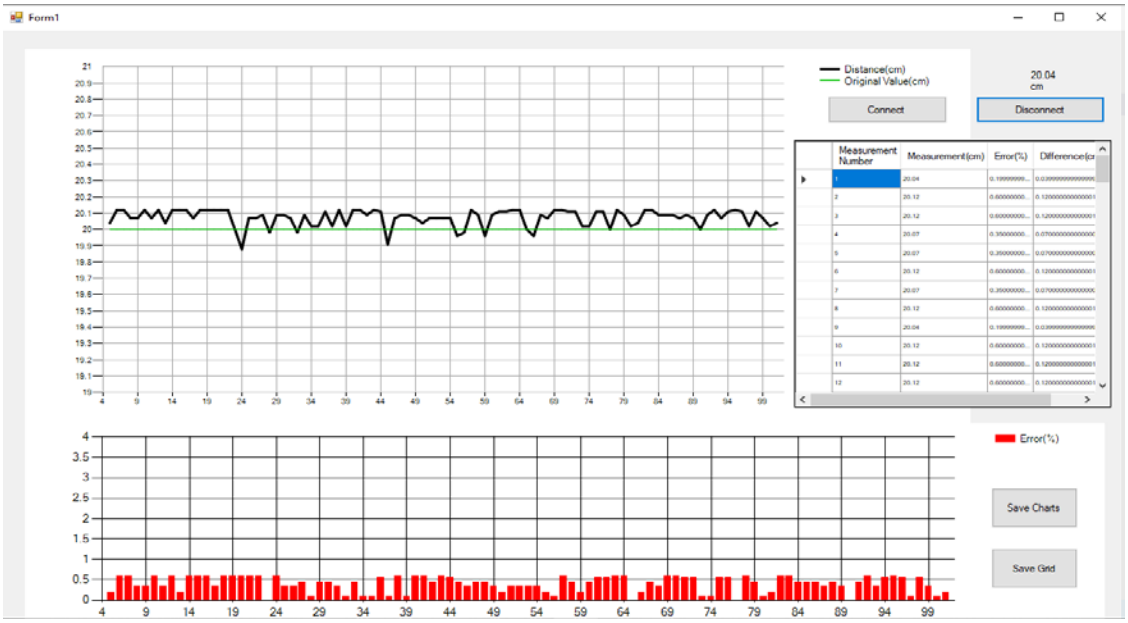

Figure 21. Visual C# .NET Application for Sensor Accuracy Test Against a Fixed Obstacle

used to read data from the serial port with the interval of 100 mS. Distance of fixed obstacle, distance data from the sensor and percentage of the error is shown on the graph using Chart elements. Also, these values are shown in tabular form using dataGridView element. In addition, two buttons are used to start and end connection between serial port and application. The experimental setup and the results obtained are given below.

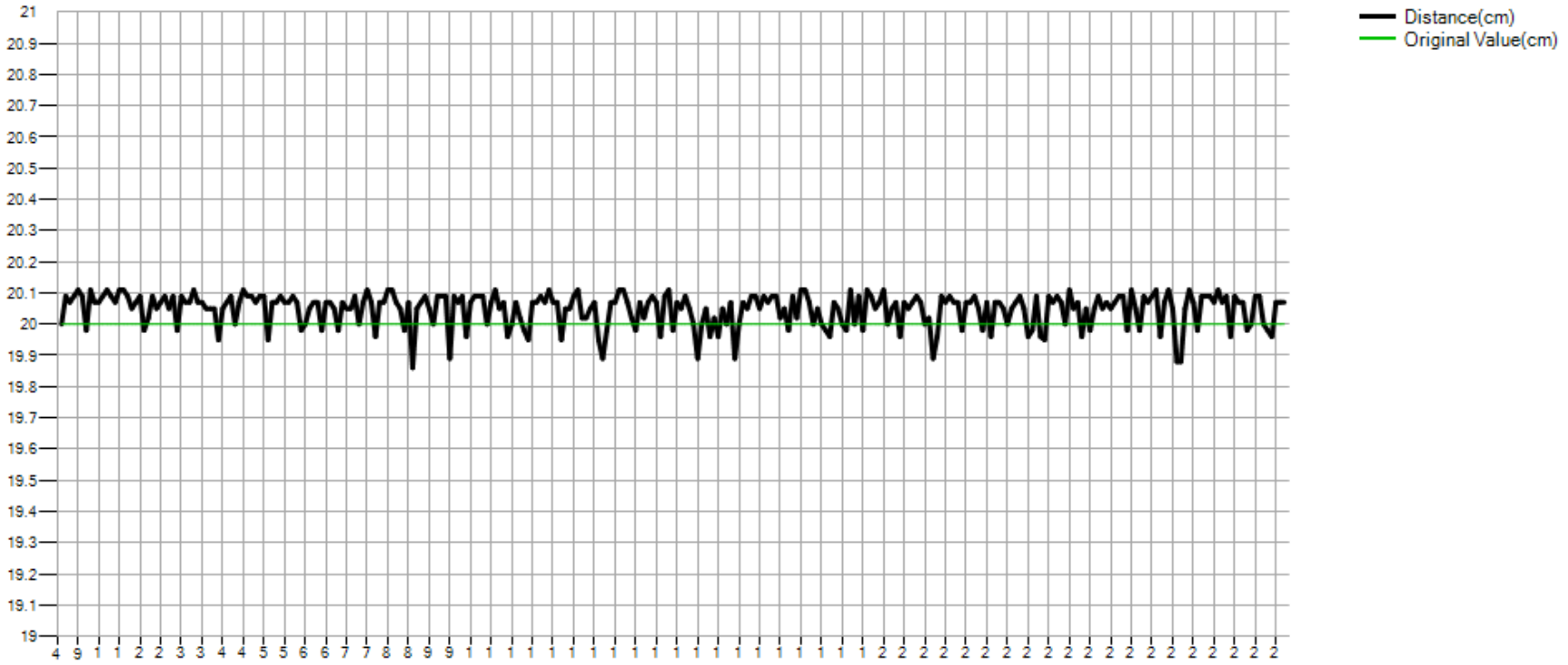

Figure 22. Accuracy Test Results - Measurement at each 100 ms and Error Value at each Measurement Plot

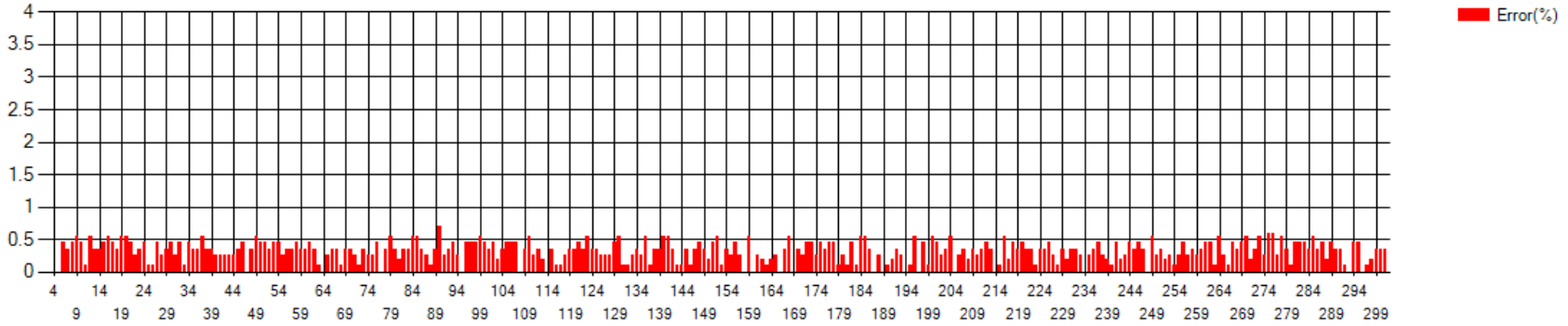

Figure 23. Accuracy Test Results - Number of Measurements Made at each 100 ms and Measured Value Plot

Table 1. Accuracy Test Results of Section 7.1 in Tabular Form

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
1	20	0	0
2	20.09	0.45	0.09
3	20.07	0.35	0.07
4	20.09	0.45	0.09
5	20.11	0.55	0.11
6	20.09	0.45	0.09
7	19.98	0.1	-0.02
8	20.11	0.55	0.11
9	20.07	0.35	0.07
10	20.07	0.35	0.07
11	20.09	0.45	0.09
12	20.11	0.55	0.11
13	20.09	0.45	0.09
14	20.07	0.35	0.07
15	20.11	0.55	0.11
16	20.11	0.55	0.11
17	20.09	0.45	0.09
18	20.05	0.25	0.05
19	20.07	0.35	0.07
20	20.09	0.45	0.09
20	20.09	0.45	0.09
21	19.98	0.1	-0.02
22	20.02	0.1	0.02
23	20.09	0.45	0.09
24	20.05	0.25	0.05
25	20.07	0.35	0.07
26	20.09	0.45	0.09
27	20.05	0.25	0.05
28	20.09	0.45	0.09
29	19.98	0.1	-0.02
30	20.09	0.45	0.09
31	20.07	0.35	0.07
32	20.07	0.35	0.07
33	20.11	0.55	0.11
34	20.07	0.35	0.07
35	20.07	0.35	0.07

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
36	20.05	0.25	0.05
37	20.05	0.25	0.05
38	20.05	0.25	0.05
39	19.95	0.25	-0.05
40	20.05	0.25	0.05
41	20.07	0.35	0.07
42	20.09	0.45	0.09
43	20	0	0
44	20.07	0.35	0.07
45	20.11	0.55	0.11
46	20.09	0.45	0.09
47	20.09	0.45	0.09
48	20.07	0.35	0.07
49	20.09	0.45	0.09
50	20.09	0.45	0.09
51	19.95	0.25	-0.05
52	20.07	0.35	0.07
53	20.07	0.35	0.07
54	20.09	0.45	0.09
55	20.07	0.35	0.07
56	20.07	0.35	0.07
57	20.09	0.45	0.09
58	20.07	0.35	0.07
59	19.98	0.1	-0.02
60	20	0	0
61	20.05	0.25	0.05
62	20.07	0.35	0.07
63	20.07	0.35	0.07
64	19.98	0.1	-0.02
65	20.07	0.35	0.07
66	20.07	0.35	0.07
67	20.05	0.25	0.05
68	19.98	0.1	-0.02
69	20.07	0.35	0.07
70	20.05	0.25	0.05
71	20.05	0.25	0.05

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
72	20.09	0.45	0.09
73	20	0	0
74	20.07	0.35	0.07
75	20.11	0.55	0.11
76	20.07	0.35	0.07
77	19.96	0.2	-0.04
78	20.07	0.35	0.07
79	20.07	0.35	0.07
80	20.11	0.55	0.11
81	20.11	0.55	0.11
82	20.07	0.35	0.07
83	20.05	0.25	0.05
84	19.98	0.1	-0.02
85	20.07	0.35	0.07
86	19.86	0.7	-0.14
87	20.05	0.25	0.05
88	20.07	0.35	0.07
89	20.09	0.45	0.09
90	20.05	0.25	0.05
91	20	0	0
92	20.09	0.45	0.09
93	20.09	0.45	0.09
94	20.09	0.45	0.09
95	19.89	0.55	-0.11
96	20.09	0.45	0.09
97	20.07	0.35	0.07
98	20.09	0.45	0.09
99	19.96	0.2	-0.04
100	20.07	0.35	0.07
101	20.09	0.45	0.09
102	20.09	0.45	0.09
103	20.09	0.45	0.09
104	20	0	0
105	20.07	0.35	0.07
106	20.11	0.55	0.11
107	20.05	0.25	0.05
108	20.07	0.35	0.07
109	19.96	0.2	-0.04
110	20	0	0
111	20.07	0.35	0.07
112	20.02	0.1	0.02
113	19.98	0.1	-0.02
114	19.95	0.25	-0.05
115	20.07	0.35	0.07
116	20.07	0.35	0.07
117	20.09	0.45	0.09
118	20.07	0.35	0.07
119	20.11	0.55	0.11
120	20.07	0.35	0.07
121	20.07	0.35	0.07
122	19.95	0.25	-0.05
123	20.05	0.25	0.05
124	20.05	0.25	0.05
125	20.09	0.45	0.09
126	20.11	0.55	0.11
127	20.02	0.1	0.02
128	20.02	0.1	0.02
129	20.05	0.25	0.05
130	20.07	0.35	0.07
131	19.95	0.25	-0.05
132	19.89	0.55	-0.11
133	19.98	0.1	-0.02
134	20.07	0.35	0.07
135	20.07	0.35	0.07
136	20.11	0.55	0.11
137	20.11	0.55	0.11
138	20.07	0.35	0.07
139	20.02	0.1	0.02
140	19.98	0.1	-0.02
141	20.07	0.35	0.07
142	20.02	0.1	0.02
143	20.07	0.35	0.07

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
144	20.09	0.45	0.09
145	20.07	0.35	0.07
146	19.96	0.2	-0.04
147	20.09	0.45	0.09
148	20.11	0.55	0.11
149	19.98	0.1	-0.02
150	20.07	0.35	0.07
151	20.05	0.25	0.05
152	20.09	0.45	0.09
153	20.05	0.25	0.05
154	20	0	0
155	19.89	0.55	-0.11
156	20	0	0
157	20.05	0.25	0.05
158	19.96	0.2	-0.04
159	20.02	0.1	0.02
160	19.96	0.2	-0.04
161	20.05	0.25	0.05
162	20	0	0
163	20.07	0.35	0.07
164	19.89	0.55	-0.11
165	20	0	0
166	20.07	0.35	0.07
167	20.05	0.25	0.05
168	20.09	0.45	0.09
169	20.09	0.45	0.09
170	20.05	0.25	0.05
171	20.09	0.45	0.09
172	20.07	0.35	0.07
173	20.09	0.45	0.09
174	20.09	0.45	0.09
175	20.02	0.1	0.02
176	20.05	0.25	0.05
177	19.98	0.1	-0.02
178	20.09	0.45	0.09
179	20.02	0.1	0.02

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
180	20.11	0.55	0.11
181	20.11	0.55	0.11
182	20.07	0.35	0.07
183	20	0	0
184	20.05	0.25	0.05
185	20	0	0
186	19.98	0.1	-0.02
187	19.96	0.2	-0.04
188	20.07	0.35	0.07
189	20.05	0.25	0.05
190	20	0	0
191	19.98	0.1	-0.02
192	20.11	0.55	0.11
193	20	0	0
194	20.09	0.45	0.09
195	19.98	0.1	-0.02
196	20.11	0.55	0.11
197	20.09	0.45	0.09
198	20.05	0.25	0.05
199	20.07	0.35	0.07
200	20.11	0.55	0.11

7.2 HC-SR04 Ultrasonic Distance Sensor Accuracy Test Against a Moving Obstacle

As mentioned in sections 2.2 and 7.1, measurement of the distance of the vehicle to the obstacle is very important, that is why sensor used in the system is critical aspect of the system. It should be noted that the vehicle is in motion even if the distance measurements to a fixed obstacle are crucial. Therefore, measurements made by sensors on the move are more important than measurements in steady state.

As mentioned in section 2.2 and 7.1, ultrasonic distance sensor must have a maximum accuracy of 3%. The experiment design presented in this section to test the accuracy of the HC-SR04 with moving obstacle at various distance values made use of the breadboard, cables, HC-SR04 ultrasonic distance sensor, Arduino Nano, moving obstacle, Arduino IDE and Visual C# .NET application (The codes written in the mentioned applications can be found in the Appendix A and Appendix C).

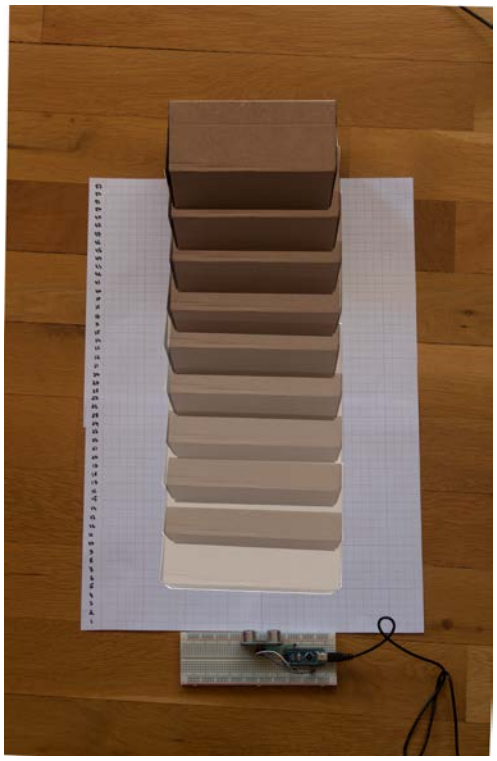

Figure 24. HC-SR04 Ultrasonic Distance Sensor Accuracy Test against a Moving Obstacle

Goal of the experiment is to calculate the accuracy of the sensor on condition of moving obstacle at various distance values. Sensor readings is the dependent variable of the experiment. Independent variable of the experiment is the distance of the obstacle between Arduino Nano and HC-SR04 in the experimental setup. The position of the obstacle is continuously increased by 1 cm during the experiment and the distance between sensor and microcontroller ranges from 4 cm to 100 cm. Controlled variables are the position of the breadboard, which contains sensor and microcontroller on it, graph paper with distance values on it, temperature of the room experiment is conducted. Temperature of the room is kept constant at 22°C.

Again, distance data from the experimental setup is provided with HC-SR04. Connection of the sensor and microcontroller and working principle of the sensor are as described in the previous section. Code piece used in the microcontroller generates a flag, which is used to trigger the process in Visual C# .NET application, after measuring the distance between obstacle and sensor.

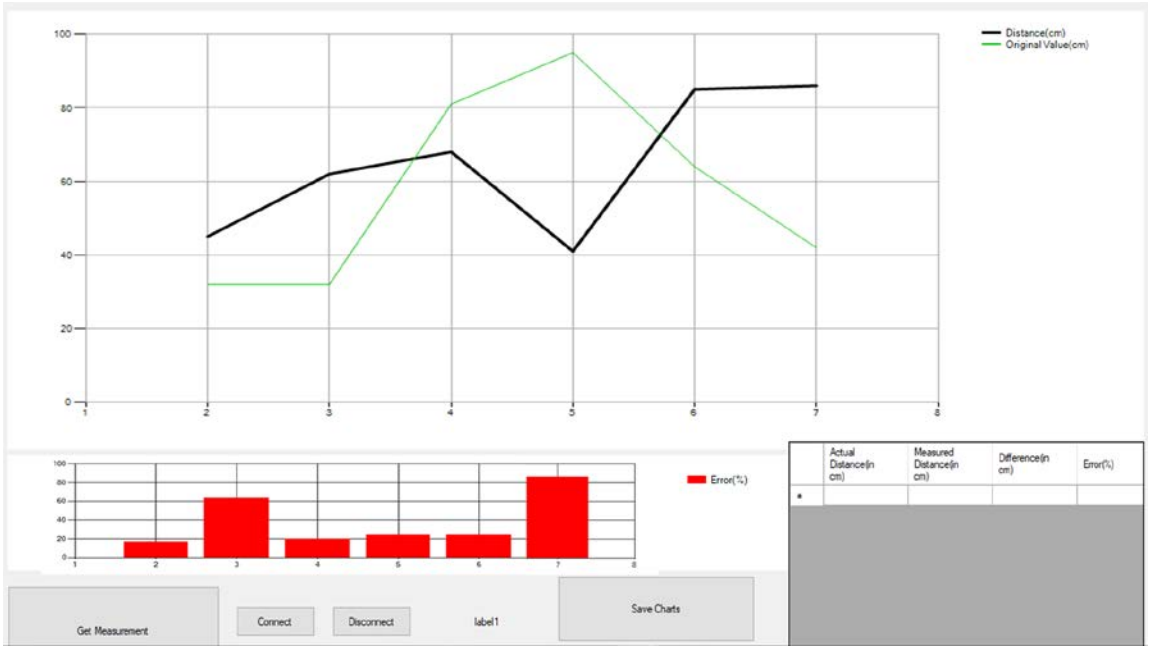

Figure 25. Visual C# .NET Application for Sensor Accuracy Test against a Moving Obstacle

Measured distance data from HC-SR04 is converted to graph and tabular form with help of the Visual C# .NET application.

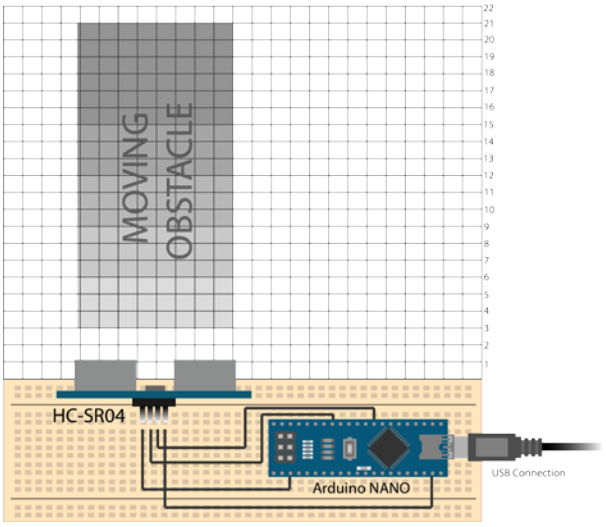

Figure 26. Sensor Accuracy Test Against a Moving Obstacle - Test Schematic

In C application, communication between Arduino board and computer is established with serialPort element. Also, timer element is used to reading data between Arduino and computer. Actual distance value, measured distance value and error percentage of

the measured data are exhibited on the chart using necessary elements. Additionally, actual distance, measured distance, difference and error values are listed in tabular form using dataGridView element. Two buttons, connect and disconnect, are used to start and stop the communication between microcontroller and computer. Label1 element displays the distance data currently read when communication is established. GetMeasurement button is used to read data from the sensor and SaveCharts button is used to save the chart in .png format.The experimental results obtained are given below as both tabular and graphical forms.

Table 2. Accuracy Test Results of Section 7.2 in Tabular Form

Actual Value(in cm)	Measured Value(in cm)	Difference(in cm)	Error (%)	Actual Value(in cm)	Measured Value(in cm)	Difference(in cm)	Error (%)
4	4.11	0.11	2.75	28	28.29	0.29	1.035714
5	5.12	0.12	2.4	29	29.11	0.11	0.37931
6	6	0	0	30	30.16	0.16	0.533333
7	7.12	0.12	1.714286	31	31.16	0.16	0.516129
8	8	0	0	32	32.12	0.12	0.375
9	9.11	0.11	1.222222	33	33.28	0.28	0.848485
10	10.18	0.18	1.8	34	34.39	0.39	1.147059
11	11.16	0.16	1.454545	35	35.4	0.4	1.142857
12	12.12	0.12	1	36	36.37	0.37	1.027778
13	13.12	0.12	0.923077	37	37.09	0.09	0.243243
14	14.25	0.25	1.785714	38	38.07	0.07	0.184211
15	15.26	0.26	1.733333	39	39.23	0.23	0.589744
16	16.16	0.16	1	40	40.3	0.3	0.75
17	17.16	0.16	0.941176	41	41.05	0.05	0.121951
18	18.23	0.23	1.277778	42	42.32	0.32	0.761905
19	19.21	0.21	1.105263	43	42.93	0.07	0.162791
20	20.18	0.18	0.9	44	43.81	0.19	0.431818
21	21.26	0.26	1.238095	45	45.21	0.21	0.466667
22	22.11	0.11	0.5	46	46.25	0.25	0.543478
23	23.29	0.29	1.26087	47	47	0	0
24	24.16	0.16	0.666667	48	48.12	0.12	0.25
25	25.19	0.19	0.76	49	49.37	0.37	0.755102
26	26.35	0.35	1.346154	50	50.12	0.12	0.24
27	27.1	0.1	0.37037	51	50.98	0.02	0.039216

Actual Value(in cm)	Measured Value(in cm)	Differ-ence(in cm)	Error (%)
52	52.14	0.14	0.269231
53	53.16	0.16	0.301887
54	54.07	0.07	0.12963
55	55.05	0.05	0.090909
56	56.21	0.21	0.375
57	56.98	0.02	0.035088
58	58.04	0.04	0.068966
59	59.25	0.25	0.423729
60	60.04	0.04	0.066667
61	60.93	0.07	0.114754
62	62	0	0
63	63.12	0.12	0.190476
64	64.07	0.07	0.109375
65	65.09	0.09	0.138462
66	66.14	0.14	0.212121
67	67.21	0.21	0.313433
68	67.88	0.12	0.176471
69	69.18	0.18	0.26087
70	70.42	0.42	0.6
71	70.82	0.18	0.253521
72	72.26	0.26	0.361111
73	73	0	0
74	74.02	0.02	0.027027
75	75.28	0.28	0.373333
76	76.11	0.11	0.144737
77	77.02	0.02	0.025974
78	78.3	0.3	0.384615
79	78.96	0.04	0.050633
80	80.33	0.33	0.4125
81	81.26	0.26	0.320988
82	82.51	0.51	0.621951
83	83.05	0.05	0.060241
84	84.37	0.37	0.440476
85	84.95	0.05	0.058824
86	86.02	0.02	0.023256
87	87.04	0.04	0.045977

Actual Value(in cm)	Measured Value(in cm)	Differ-ence(in cm)	Error (%)
88	87.98	0.02	0.022727
89	89.38	0.38	0.426966
90	90.67	0.67	0.744444
91	91.02	0.02	0.021978
92	92.14	0.14	0.152174
93	93	0	0
94	93.99	0.01	0.010638
95	95.07	0.07	0.073684
96	95.16	0.84	0.875
97	97.32	0.32	0.329897
98	98.31	0.31	0.316327
99	99	0	0
100	100.07	0.07	0.07

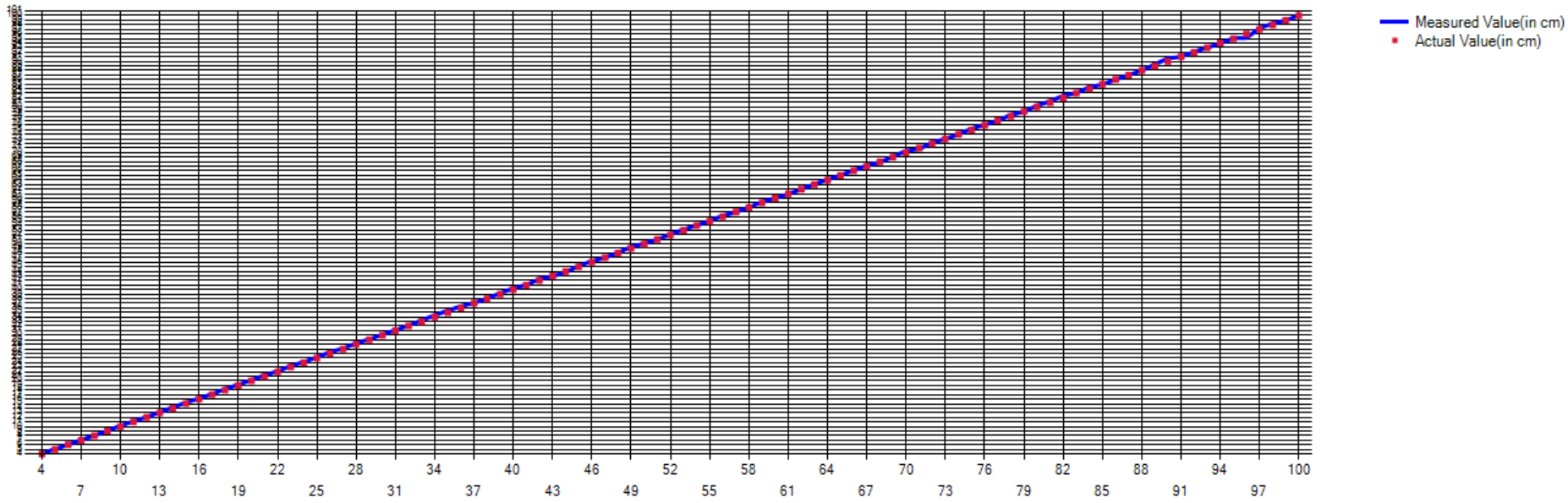

Figure 28. Accuracy Test Results - Actual Distance Values and Measured Distance Values Plot

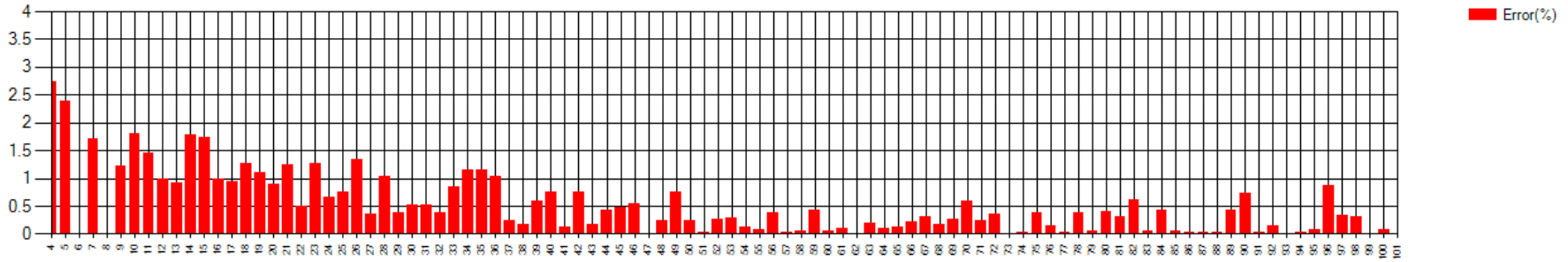

Figure 27. Accuracy Test Results - Actual Distance Values and Error Values at each Measurement Plot

7.3 GP2Y0A41SK Sharp Infrared Sensor Accuracy Test Against a Fixed Obstacle

The infrared distance sensor is one of the distance measurement components in system. The sensor used in system takes a very important part with regards to measurement of the distance of the vehicle to the object, and select the appropriate algorithm.

The infrared distance sensor must have a maximum certainty of 3%. Experiment is designed to test the accuracy of the infrared sensor with fixed obstacle at 20cm. Experiment consists of the breadboard, jumper cables, GP2Y0A41SK Sharp IR Sensor, Arduino Nano, fixed obstacle at 20cm, Arduino IDE and Visual C#. Net application (The codes written in the mentioned applications can be found in the Appendix B and Appendix D).

Primary intention of the experiment is to measure the certainty of the sensor in case of fixed object at 20cm. Motion test is performed at the area which has an length of maximum 400cm, so controlled variable of the experiment is selected as object's distance to sensor. Dependent variable of the experiment is the distance difference between controlled variable and value measured by infrared sensor. This difference is used to find the certainty of the sensor.

To be able to use the GP2Y0A41SK Sharp IR sensor, its voltage supply cable is connected to Arduino's 5V output pin as well as its ground cable is connected to Arduino's ground pin. Also, Sensor's data cable is connected to one of the analog pins of the Arduino Nano to ensure the data communication between sensor and Arduino is established. The power of the Arduino is provided by USB connection to computer.

Obtaining the distance information from the sensor is as follows. When the analog voltage read command from the microcontroller is provided, the sensor will send a reading to microcontroller. Conversion between voltage level to digital reading is provided with Analog-to-Digital converter of the microcontroller. The "flag" from the application is waited for the

measured value to be sent to the application. If the flag from the program is the wanted flag value, the measured distance value is sent to the application via serial port communication.

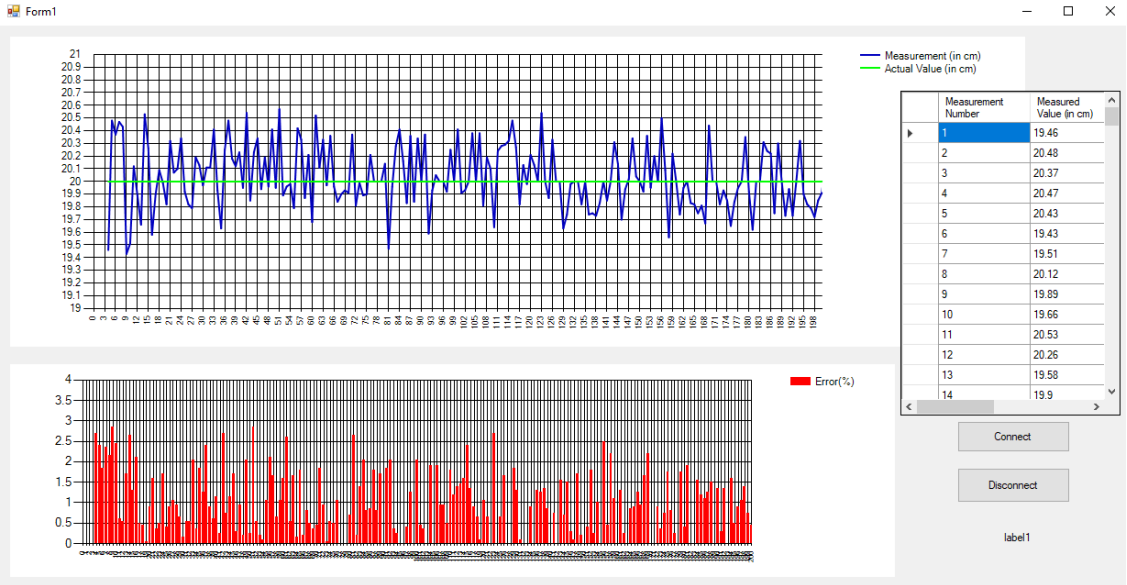

Figure 29. Visual C# .NET Application for IR Distance Sensor Accuracy Test Against a Fixed Obstacle

Visual C#. NET Application is used to convert measured distance values from the sensor to graph and tabular form. In application, SerialPort and Timer elements are used to provide the necessary communication between microcontroller and computer. Also, timer element is used to construct the timing between application and experiment setup. Distance of fixed obstacle, distance data from the sensor and percentage of the error are shown on the graph using Chart elements. In addition, these values are shown in tabular form using dataGridView element. In addition, two buttons are used to start and end connection between serial port and application. Saving charts and grid is achieved by clicking the specific chart or grid. The experimental setup and the results obtained are given below.

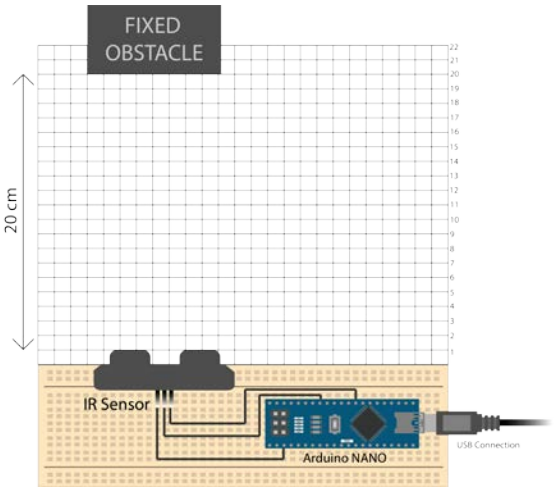

Figure 30. Sharp Infrared Sensor Precision Test Against a Fixed Obstacle - Test Schematic

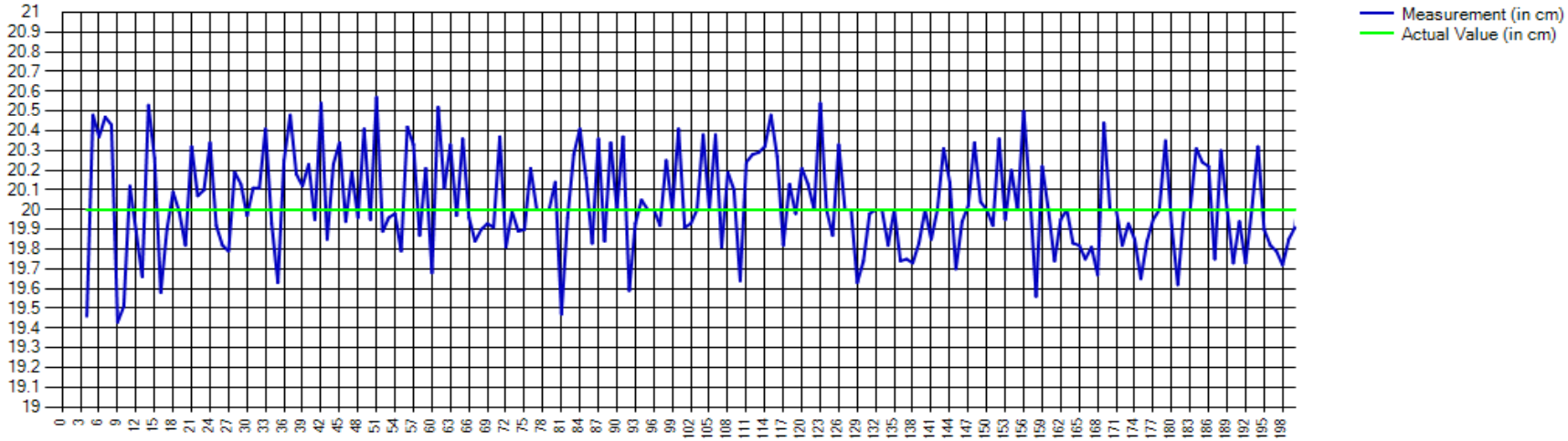

Figure 31. Accuracy Test Results - Actual Distance Values and Measured Distance Values Plot of the IR Sensor

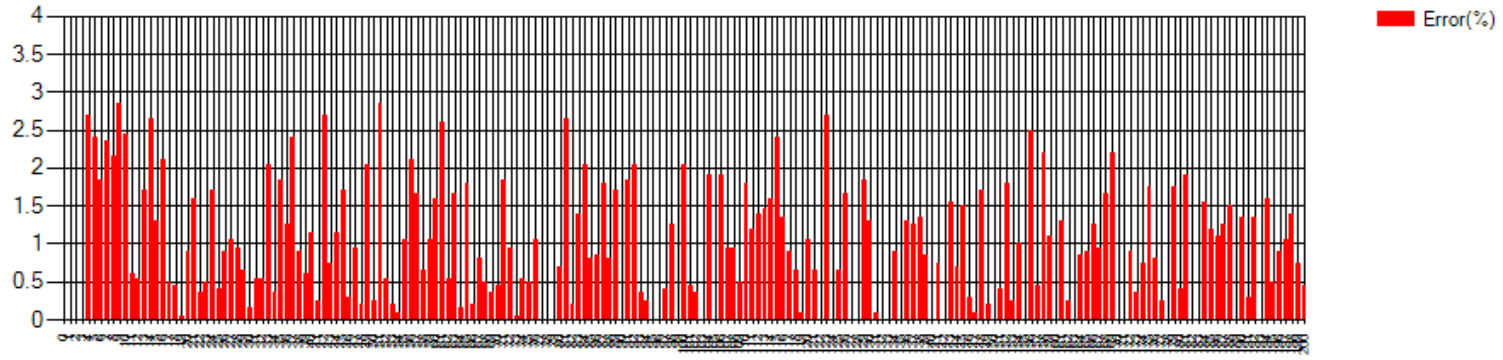

Figure 32. Accuracy Test Results - Actual Distance Values and Error Values at each Measurement Plot of IR Sensor

Table 3. Accuracy Test Results of Section 7.3 in Tabular Form

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)	Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
1	19.46	0.54	2.7	37	20.23	0.23	1.15
2	20.48	0.48	2.4	38	19.95	0.05	0.25
3	20.37	0.37	1.85	39	20.54	0.54	2.7
4	20.47	0.47	2.35	40	19.85	0.15	0.75
5	20.43	0.43	2.15	41	20.23	0.23	1.15
6	19.43	0.57	2.85	42	20.34	0.34	1.7
7	19.51	0.49	2.45	43	19.94	0.06	0.3
8	20.12	0.12	0.6	44	20.19	0.19	0.95
9	19.89	0.11	0.55	45	19.96	0.04	0.2
10	19.66	0.34	1.7	46	20.41	0.41	2.05
11	20.53	0.53	2.65	47	19.95	0.05	0.25
12	20.26	0.26	1.3	48	20.57	0.57	2.85
13	19.58	0.42	2.1	49	19.89	0.11	0.55
14	19.9	0.1	0.5	50	19.96	0.04	0.2
15	20.09	0.09	0.45	51	19.98	0.02	0.1
16	19.99	0.01	0.05	52	19.79	0.21	1.05
17	19.82	0.18	0.9	53	20.42	0.42	2.1
18	20.32	0.32	1.6	54	20.33	0.33	1.65
19	20.07	0.07	0.35	55	19.87	0.13	0.65
20	20.1	0.1	0.5	56	20.21	0.21	1.05
21	20.34	0.34	1.7	57	19.68	0.32	1.6
22	19.92	0.08	0.4	58	20.52	0.52	2.6
23	19.82	0.18	0.9	59	20.11	0.11	0.55
24	19.79	0.21	1.05	60	20.33	0.33	1.65
25	20.19	0.19	0.95	61	19.97	0.03	0.15
26	20.13	0.13	0.65	62	20.36	0.36	1.8
27	19.97	0.03	0.15	63	19.96	0.04	0.2
28	20.11	0.11	0.55	64	19.84	0.16	0.8
29	20.11	0.11	0.55	65	19.9	0.1	0.5
30	20.41	0.41	2.05	66	19.93	0.07	0.35
31	19.93	0.07	0.35	67	19.91	0.09	0.45
32	19.63	0.37	1.85	68	20.37	0.37	1.85
33	20.25	0.25	1.25	69	19.81	0.19	0.95
34	20.48	0.48	2.4	70	19.99	0.01	0.05
35	20.18	0.18	0.9	71	19.89	0.11	0.55
36	20.12	0.12	0.6	72	19.9	0.1	0.5

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
73	20.21	0.21	1.05
74	20	0	0
75	20	0	0
76	20	0	0
77	20.14	0.14	0.7
78	19.47	0.53	2.65
79	19.96	0.04	0.2
80	20.28	0.28	1.4
81	20.41	0.41	2.05
82	20.16	0.16	0.8
83	19.83	0.17	0.85
84	20.36	0.36	1.8
85	19.84	0.16	0.8
86	20.34	0.34	1.7
87	20	0	0
88	20.37	0.37	1.85
89	19.59	0.41	2.05
90	19.93	0.07	0.35
91	20.05	0.05	0.25
92	20	0	0
93	20	0	0
94	19.92	0.08	0.4
95	20.25	0.25	1.25
96	20	0	0
97	20.41	0.41	2.05
98	19.91	0.09	0.45
99	19.93	0.07	0.35
100	20	0	0
101	20.38	0.38	1.9
102	20	0	0
103	20.38	0.38	1.9
104	19.81	0.19	0.95
105	20.19	0.19	0.95
106	20.1	0.1	0.5
107	19.64	0.36	1.8
108	20.24	0.24	1.2

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
109	20.28	0.28	1.4
110	20.29	0.29	1.45
111	20.32	0.32	1.6
112	20.48	0.48	2.4
113	20.27	0.27	1.35
114	19.82	0.18	0.9
115	20.13	0.13	0.65
116	19.98	0.02	0.1
117	20.21	0.21	1.05
118	20.13	0.13	0.65
119	20	0	0
120	20.54	0.54	2.7
121	20	0	0
122	19.87	0.13	0.65
123	20.33	0.33	1.65
124	20	0	0
125	20	0	0
126	19.63	0.37	1.85
127	19.74	0.26	1.3
128	19.98	0.02	0.1
129	20	0	0
130	20	0	0
131	19.82	0.18	0.9
132	20	0	0
133	19.74	0.26	1.3
134	19.75	0.25	1.25
135	19.73	0.27	1.35
136	19.83	0.17	0.85
137	20	0	0
138	19.85	0.15	0.75
139	20	0	0
140	20.31	0.31	1.55
141	20.14	0.14	0.7
142	19.7	0.3	1.5
143	19.94	0.06	0.3
144	20.02	0.02	0.1

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
145	20.34	0.34	1.7
146	20.04	0.04	0.2
147	20	0	0
148	19.92	0.08	0.4
149	20.36	0.36	1.8
150	19.95	0.05	0.25
151	20.2	0.2	1
152	20	0	0
153	20.5	0.5	2.5
154	20.09	0.09	0.45
155	19.56	0.44	2.2
156	20.22	0.22	1.1
157	20	0	0
158	19.74	0.26	1.3
159	19.95	0.05	0.25
160	20	0	0
161	19.83	0.17	0.85
162	19.82	0.18	0.9
163	19.75	0.25	1.25
164	19.81	0.19	0.95
165	19.67	0.33	1.65
166	20.44	0.44	2.2
167	20	0	0
168	20	0	0
169	19.82	0.18	0.9
170	19.93	0.07	0.35
171	19.85	0.15	0.75
172	19.65	0.35	1.75
173	19.84	0.16	0.8
174	19.95	0.05	0.25
175	20	0	0
176	20.35	0.35	1.75
177	19.92	0.08	0.4
178	19.62	0.38	1.9
179	20	0	0
180	20	0	0

Measurement Number	Measurement (in cm)	Error (%)	Difference (in cm)
181	20.31	0.31	1.55
182	20.24	0.24	1.2
183	20.22	0.22	1.1
184	19.75	0.25	1.25
185	20.3	0.3	1.5
186	20	0	0
187	19.73	0.27	1.35
188	19.94	0.06	0.3
189	19.73	0.27	1.35
190	20	0	0
191	20.32	0.32	1.6
192	19.9	0.1	0.5
193	19.82	0.18	0.9
194	19.79	0.21	1.05
195	19.72	0.28	1.4
196	19.85	0.15	0.75
197	19.91	0.09	0.45
198	20.16	0.16	0.8
199	19.79	0.21	1.05
200	20	0	0

7.4 GP2Y0A41SK Sharp Infrared Sensor Accuracy Test Against a Moving Obstacle

In sections 2.2 and 7.3, analysis of the distance between vehicle and obstacle is deeply crucial. That is why sensors used in the system are critical aspect of the system. Although distance measurements are important between fixed obstacle and vehicle, it should be noted that vehicle is in motion during the process of parking. For this reason, calculation of the distance during the motion is more important than measurements between fixed object and vehicle.

The maximum error margin of the sensor used in our system should be 3%. The experiment design presented in this section is to evaluate whether or not the accuracy of the infrared sensor is sufficient to secure this necessity. The experimental setup is consists of breadboard, jumper cables, infrared sensor, Arduino Nano and a moving object. Also, Arduino IDE and Visual C#. NET application is used to measure the distance data and to visualize these data values (The codes can be found in the Appendix D and Appendix E).

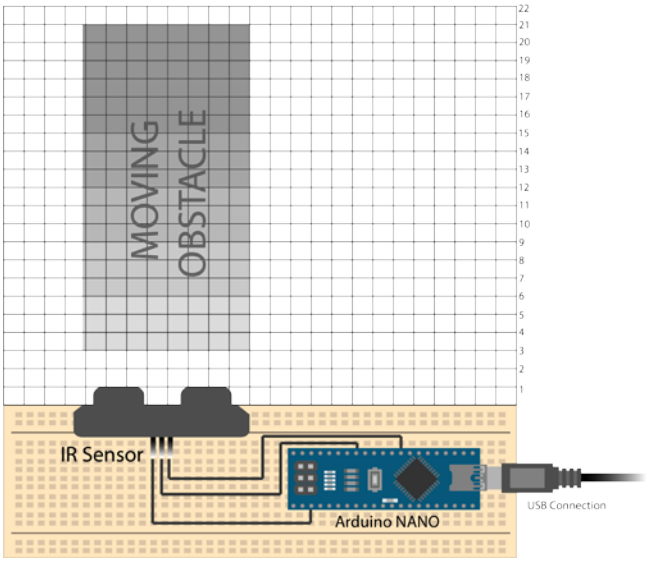

Figure 33. Sharp Infrared Sensor Accuracy Test Against a Moving Obstacle - Test Schematic

Purpose of the experiment is to determine whether or not the accuracy of the sensor on condition of moving obstacle at various distance values is enough. Independent variable of the experiment is selected as position of the obstacle. During the experiment, the position of the obstacle is continuously increased by one centimeter at each reading. Location of the bread-board, which contains sensor and microcontroller, and room temperature (22°C) is kept constant as controlled variables. In addition, sensor readings is the dependent variable of the experiment. Furthermore, the distance between sensor and obstacle ranges from 4cm to 30cm.

Connection between sensor and microcontroller is as described in the previous section. Code piece used in the microcontroller, again, generates a flag, which is used to trigger the process in Visual C# .NET application.

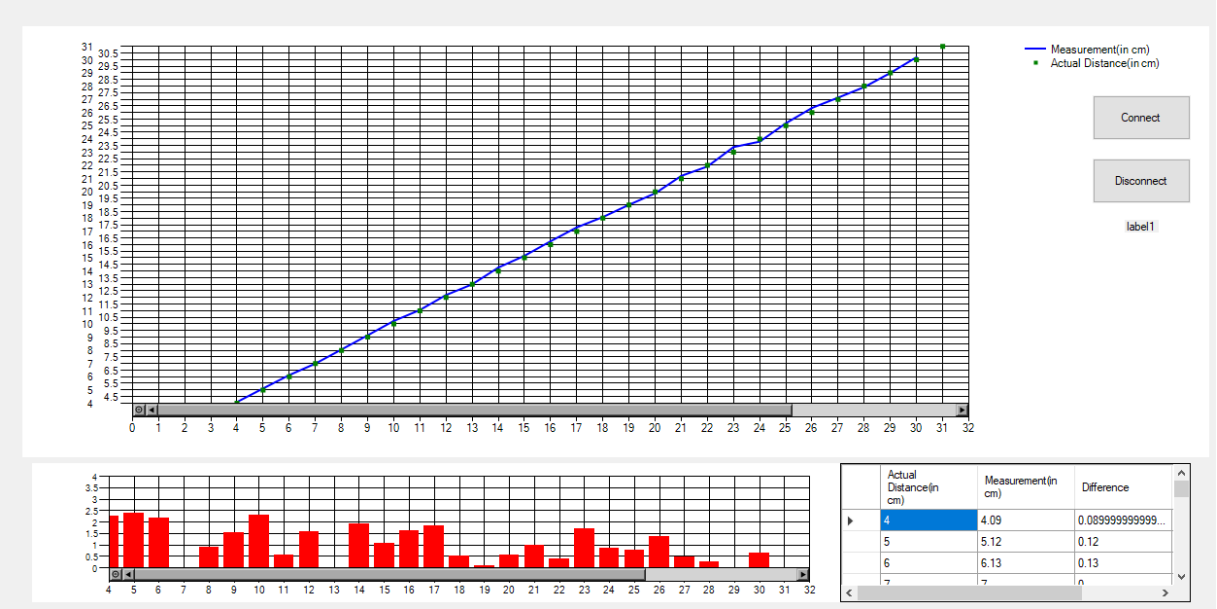

Figure 34. Visual C# .NET Application for IR Distance Sensor Accuracy Test Against a Moving Obstacle

In application, communication between microcontroller and computer is provided with SerialPort element. In addition, timer element is used to read data between microcontroller and

computer. Actual distance value, measured distance value and error percentage values are displayed on the chart using necessary elements in application. Also, these values are listed in tabular form. Connect and Disconnect buttons in application are used to start and end the communication. The experimental results obtained are given below as both tabular and graphical forms.

Table 4. Accuracy Test Results of Section 7.4 in Tabular Form

Actual Distance(in cm)	**Measurement(in cm)**	**Difference(in cm)**	**Error (%)**
4	4.09	0.09	2.25
5	5.12	0.12	2.4
6	6.13	0.13	2.166667
7	7	0	0
8	8.07	0.07	0.875
9	9.14	0.14	1.555556
10	10.23	0.23	2.3
11	11.06	0.06	0.545455
12	12.19	0.19	1.583333
13	13	0	0
14	14.27	0.27	1.928571
15	15.16	0.16	1.066667
16	16.26	0.26	1.625
17	17.31	0.31	1.823529
18	18.09	0.09	0.5
19	19.02	0.02	0.105263
20	19.89	0.11	0.55
21	21.21	0.21	1
22	21.92	0.08	0.363636
23	23.39	0.39	1.695652
24	23.8	0.2	0.833333
25	25.19	0.19	0.76
26	26.35	0.35	1.346154
27	27.13	0.13	0.481481
28	27.93	0.07	0.25
29	29	0	0
30	30.19	0.19	0.633333

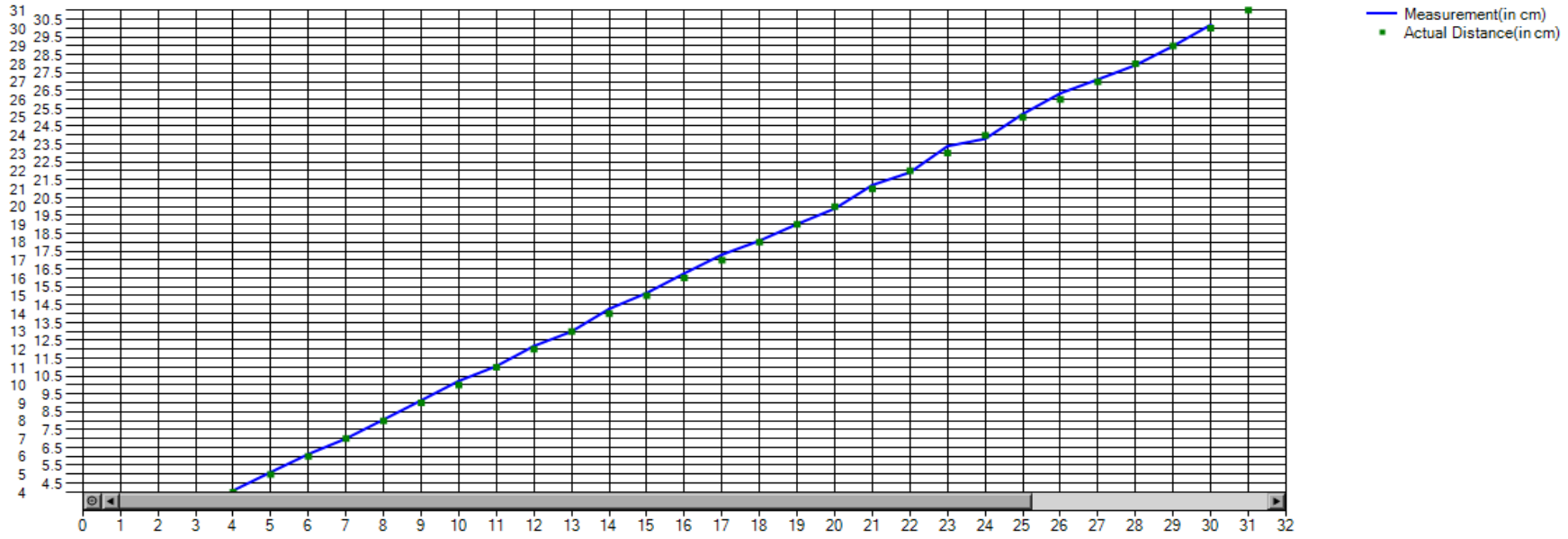

Figure 36. Accuracy Test Results - Actual Distance Values and Measured Distance Values Plot of the IR Sensor for Section 7.4

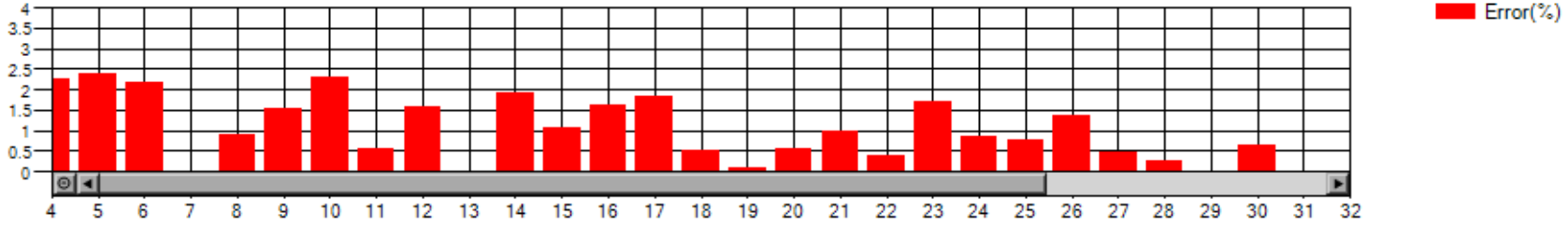

Figure 35. Accuracy Test Results - Actual Distance Values and Error Values at each Measurement plot of IR sensor for section 7.4

8. PROJECT PLAN

- **Work Package 1 - Literature Search:** Basic engineering and scientific principles, related technologies and existing solutions are searched.

- **Work Package 2 – Deciding on the Plan to be Followed While Realizing the Project and Task Sharing:** Considering the studies conducted on literature search, the methods and task distributions to be followed in the project are realized.

- **Work Package 3 – Basic Electronic Parts' Selection for the Project:** Selecting the components to meet the requirements of the system. Then, market research on selected parts.

- **Work Package 4 – Coding:** Control system is coded for ATmega2560 by using Arduino IDE. Developing algorithms for finding an empty parking area, obstacle detection and parking process. The working principles of the sensors, stepper motors and microcontroller are learned in detail, and realization of the system is achieved by coding.

- **Work Package 5 – Sensor tests:** The coded sensors are subjected to a series of tests to find out whether their accuracies are suitable for the system. If they are not suitable for the system, improvement and optimization studies are carried out.

- **Work Package 6 – Integration and testing of prepared software with microcontroller:** The codes for the sensor test on the computer are adapted to the microcontroller, and the microcontroller is integrated with the sensors. The generated algorithms are subjected to a series of tests with regard to position, reaction time, speed etc.

- **Work Package 7 – Mounting sensors, microcontroller and motors to the robotic vehicle:** Selected components are provided. The components are mounted in such a way that they do not violate the specified system requirements. Once the desired results are obtained from the tests of the required electronic units, the sensors and microcontroller are placed in the most appropriate way in terms of aerodynamic and sensor efficiency.

- **Work Package 8 – Performing the authentication tests of the final state of the project:** After the environment where the vehicle is parked is selected, the project is carried out for the verification of the project. The parking process will be tested and the necessary software and hardware changes will be made to finalize the project.

Table 5. Work Package, Resource and Duration

Work Package	**Resource**	**Duration (in days)**	**Duration (in weeks)**
1	Fatih, Serdar, Batuhan, Berk	4	0.57
2	Fatih, Serdar, Batuhan, Berk	6	0.86
3	Fatih, Serdar, Batuhan, Berk	8	1.14
4	Batuhan, Serdar	41	5.86
5	Berk, Batuhan	38	5.43
6	Fatih, Berk,Serdar	40	5.71
7	Fatih, Serdar, Batuhan, Berk	94	13.43
8	Fatih, Serdar, Batuhan, Berk	28	4
PROJECT COMPLETITION TIME		**113**	**16.14**

Work Package 1 - Literature Search	2/5/18	2/8/18	4	
Existing Solutions	2/5/18	2/8/18	4	Serdar,Fatih,Berk,Batuhan
Scientific Principles	2/5/18	2/7/18	3	Serdar,Fatih,Berk,Batuhan
Related Technologies	2/5/18	2/8/18	4	Serdar,Fatih,Berk,Batuhan
Work Package 2 - Deciding on the Plan to be Followed...	2/9/18	2/14/18	6	
Deciding the plan	2/9/18	2/11/18	3	Serdar,Fatih,Berk,Batuhan
Task Distribution	2/12/18	2/14/18	3	Serdar,Fatih,Berk,Batuhan
Work Package 3 - Basic Electronic Parts' Selection for ...	2/15/18	2/22/18	8	
Selecting Microcontroller	2/19/18	2/22/18	4	Serdar,Fatih,Berk,Batuhan
Selecting Ultrasonic Sensor	2/15/18	2/16/18	2	Serdar,Fatih,Batuhan
Selecting IR Sensor	2/15/18	2/16/18	2	Serdar,Berk
Selecting Stepper Motor	2/15/18	2/17/18	3	Serdar
Selecting Stepper Motor Driver	2/18/18	2/18/18	1	Serdar
Work Package 4 - Coding	2/23/18	4/4/18	41	
Stepper Motors Codes	2/23/18	4/4/18	41	Serdar
Creating the algortihm that performing parking	2/25/18	3/21/18	25	Serdar,Fatih
Creating the algortihm that find empty parking area	2/25/18	3/18/18	22	Berk,Batuhan
Ultrasonic Sensor Codes	2/23/18	3/22/18	28	Fatih,Batuhan
IR Sensor Codes	2/23/18	3/22/18	28	Serdar,Berk
Creating the algorthim that finds the obstacles	2/25/18	3/4/18	8	Serdar,Fatih
Work Package 5 - Sensor tests	3/23/18	4/29/18	38	
Ultrasonic Sensor Accuracy Tests	3/23/18	4/29/18	38	Serdar,Fatih,Batuhan
IR Sensor Accuracy Tests	3/23/18	4/29/18	38	Serdar,Fatih,Berk
Work Package 6 - Integration and testing of prepared s...	3/22/18	4/30/18	40	
Angle/Position Tests	3/22/18	4/18/18	28	Serdar,Fatih
IR Sensor Obstacle Tests	3/23/18	4/16/18	25	Serdar,Berk,Batuhan
Ultrasonic Sensor Obstacle Tests	3/23/18	4/16/18	25	Berk,Batuhan
Speed Tests	4/5/18	4/30/18	26	Serdar,Fatih
Work Package 7 - Mounting sensors, microcontroller a...	2/5/18	5/9/18	94	
Platform of the vehicle	2/26/18	3/26/18	29	Fatih
Mounting Microcontroller	2/5/18	2/25/18	21	Serdar,Fatih,Berk,Batuhan
Mounting Stepper Motors	5/1/18	5/9/18	9	Serdar
Mounting Stepper Motor Drivers	3/27/18	4/4/18	9	Batuhan
Mounting Ultrasonic Sensors	4/30/18	5/2/18	3	Berk
Mounting IR Sensors	4/30/18	5/4/18	5	Fatih
Work Package 8 - Performing the authentication tests ...	5/1/18	5/28/18	28	Serdar,Fatih,Berk,Batuhan

Figure 37. Task Names, Durations and Resources of Project

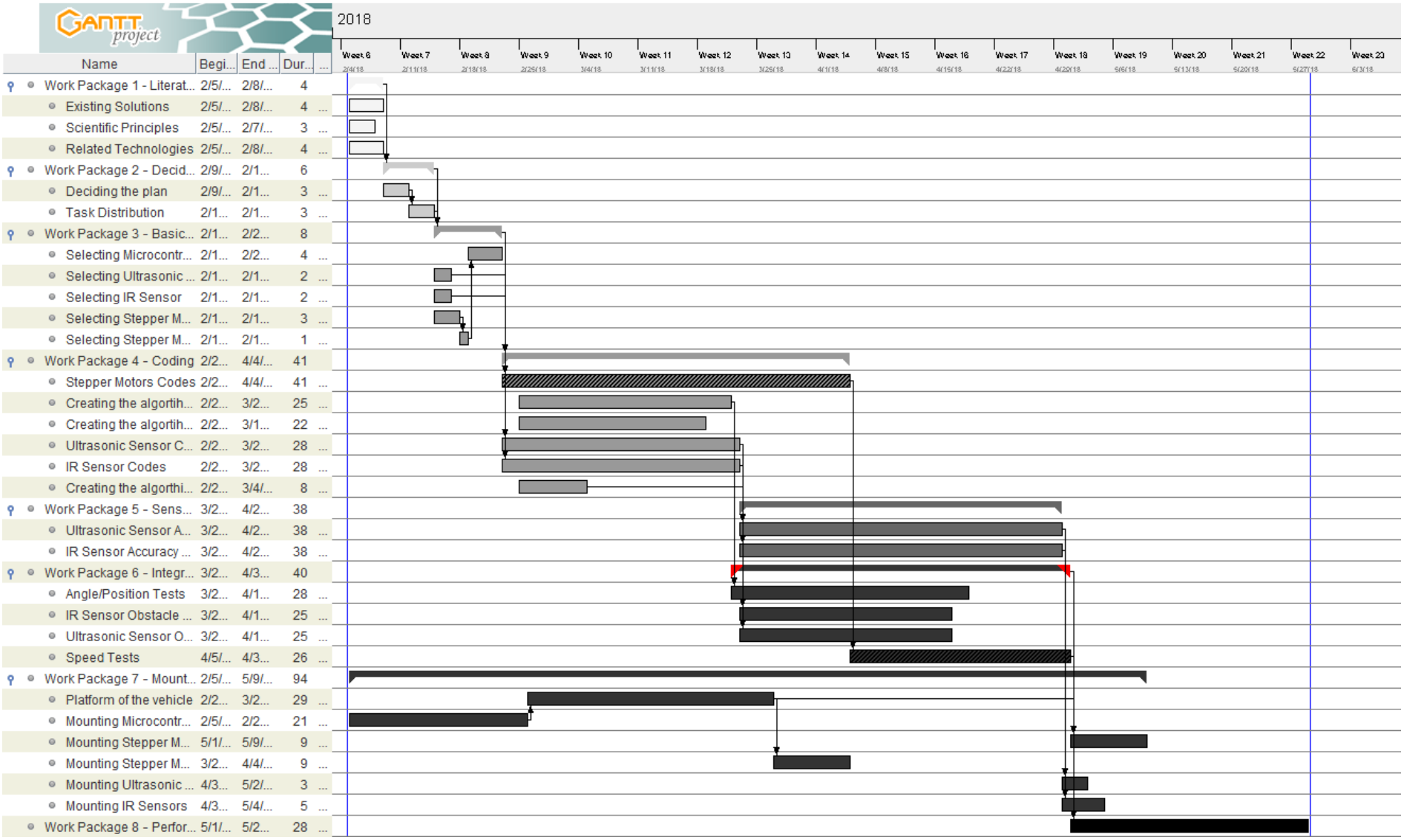

Figure 38. Gantt Diagram of the Project

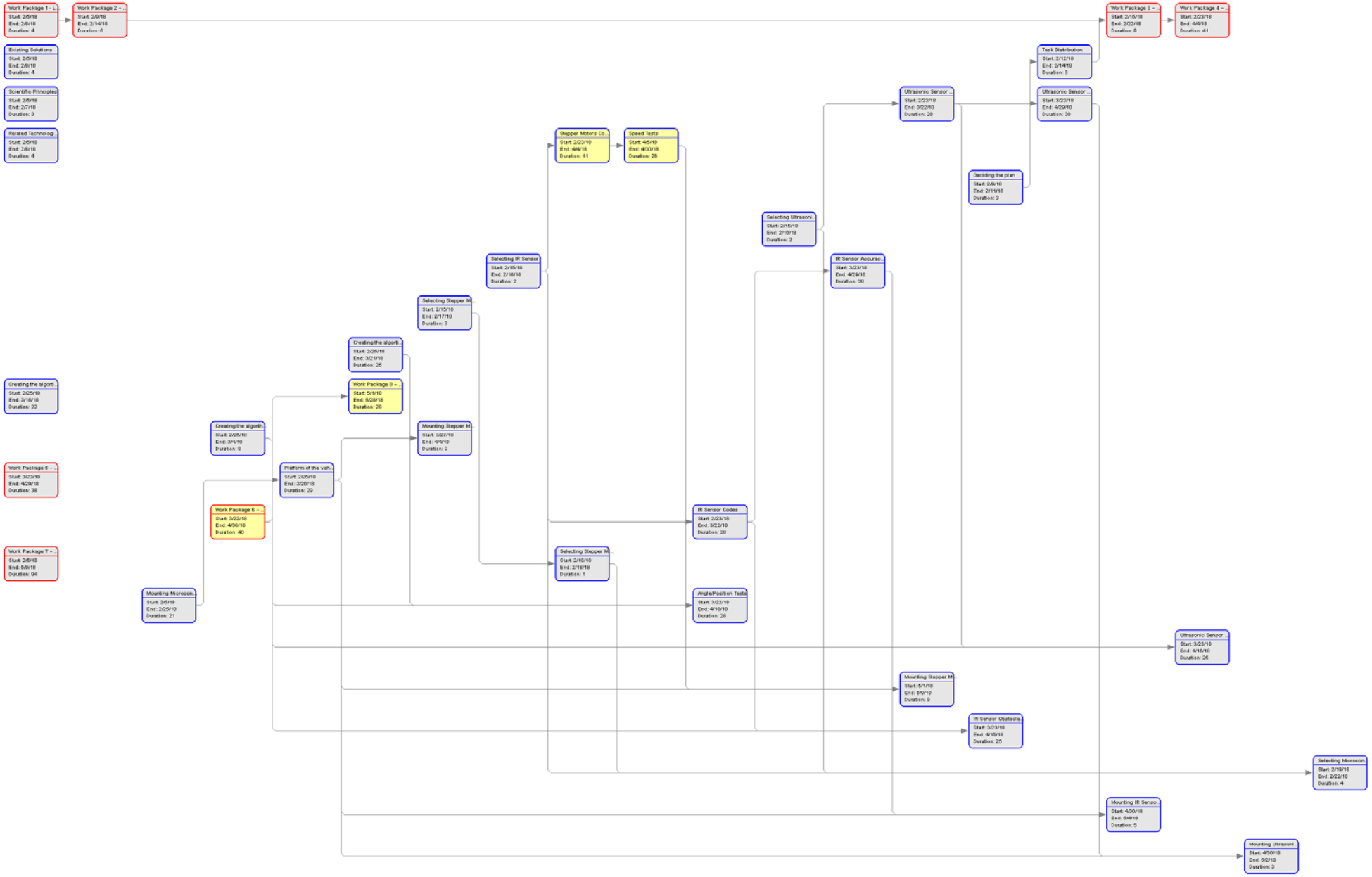

Figure 39. PERT Diagram of the Project

9. CONCLUSION

Within the scope of this study, a prototype has been developed in order to find suitable parking space and autonomous parking of cars and similar vehicles. With the hardware and software developed, the prototype vehicle can determine the space to be parked when there is enough space. Thus, the driver is able to park automatically with the help of sensors on the vehicle without any help.

The algorithm design is based on the use of required sensors and components for autonomous vehicle control and parking. The autonomous parking part is divided into two places: finding the appropriate parking space and then parking.

The most significant difference between this study and other similar studies is that evaluating the appropriate parking is done according to the dimensions of the vehicle in the software. A microcontroller board, a motor driver board, an infrared distance sensor and three ultrasonic distance sensor have been assisted in order to locate the parking lot and perform the necessary maneuvers. In the determination of the parking lot, the distance measurement was made according to the information taken continuously from each sensor placed on the right and left side of the vehicle and the appropriate parking space was determined.

Different scenarios have been tried and the prototype developed for these scenarios has been successful. Different boxes are used to represent the obstacles in application tests. The developed prototype has been tested for three conditions. These are respectively if the parking space is smaller than the size of the vehicle, there is a suitable parking space between the two obstacles and the appropriate parking space is at the end of the experiment setup. In order to eliminate the problem in the last parking maneuver, In order to increase the stability of the read data from the ultrasonic distance sensor, it was decided not to read the sensor on

the left side of the vehicle during parking process. The prototype vehicle developed within the scope of the study successfully carried out parking / non-parking in three situations.

In future studies, it is thought that automatic parking and autonomous parking works will be done based on image processing.

REFERENCES

[1] Statista, "International Car Sales 1990-2018," July 2018. [Online]. Available: https://www.statista.com/statistics/200002/international-car-sales-since-1990/. [Accessed 8 April 2018].

[2] Statista, "Car Manufacturers by Revenue," 2018. [Online]. Available: https://www.statista.com/statistics/232958/revenue-of-the-leading-car-manufacturers-worldwide/. [Accessed 8 April 2018].

[3] Statista, "Likelihood of consumers buying a car with new technology in the next 24 months in the United States as of February 2017, by generation," May 2017. [Online]. Available: https://www.statista.com/statistics/719360/likelihood-of-consumers-buying-a-car-with-new-technology-usa-by-age/. [Accessed 1 April 2018].

[4] Ford Motor Company, "Active Park Assist," 2018. [Online]. Available: https://owner.ford.com/how-tos/vehicle-features/convenience-and-comfort/active-park-assist.html. [Accessed 8 April 2018].

[5] Volkswagen, "Park Assist," 2018. [Online]. Available: http://www.volkswagen.co.uk/technology/parking-and-manoeuvring/park-assist. [Accessed 8 April 2018].

[6] BMW UK, "Intelligent Parking," 2018. [Online]. Available: https://www.bmw.co.uk/bmw-ownership/connecteddrive/driver-assistance/intelligent-parking. [Accessed 8 April 2018].

[7] Tuik.gov.tr, "Türkiye Istatistik Kurumu, Karayolu Trafik Kaza Istatistikleri, 2016," 21 June 2017. [Online]. Available: http://www.tuik.gov.tr/PreHaberBultenleri.do?id=24606. [Accessed 8 April 2018].

[8] National Safety Council, "Parking Lot Safety," 2016. [Online]. Available: https://www.nsc.org/road-safety/safety-topics/distracted-driving/parking-lot-safety. [Accessed 8 April 2018].

[9] U.S. DEPARTMENT OF TRANSPORTATION, "Large Truck and Bus Crash Facts 2016," 2018. [Online]. Available: https://www.fmcsa.dot.gov/safety/data-and-statistics/large-truck-and-bus-crash-facts-2016. [Accessed 8 April 2018].

[10] K. Demirli and M. Khoshnejad, "Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller," *Fuzzy Sets and Systems*, vol. 160, no. 19, pp. 2876-2891, 2009.

[11] D. Pérez-Morales, S. Domínguez-Quijada, O. Kermorgant and P. Martinet, "Autonomous parking using a sensor based approach," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016.

[12] F. Gómez-Bravo, F. Cuesta and A. Ollero, "Parallel and diagonal parking in nonholonomic autonomous vehicles," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 4, pp. 419-434, 2001.

[13] B. Lee, Y. Wei and Y. Guo, "Automatic Parking of Self-driving Car Based on LIDAR," in *2017 ISPRS Geospatial Week*, Wuhan, China, 2017.

[14] I. E. Paromtchik and C. Laugier, "Motion Generation and Control for Parking an Autonomous Vehicle," in *Proceedings of IEEE International Conference on Robotics and Automation*, Minneapolis, USA, 1996.

[15] C95.1-2005, *IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz*, IEEE, 2006.

[16] ISO 10218-2:2011, *Robots and Robotic Devices and Safety Requirements for Industrial Robots*, International Organization for Standardization, 2011.

[17] ISO/IEC/IEEE 29119-5:2016, *Software and Systems Engineering - Software Testing*, IEEE, 2016.

[18] Software Testing Standard, "The International Software Testing Standard," 2013. [Online]. Available: http://softwaretestingstandard.org/. [Accessed 2018].

[19] IEEE, "Style Manual," 2006. [Online]. Available: https://www.ieee.org/documents/style_manual.pdf. [Accessed 12 February 2018].

[20] IEEE 754-2008, *IEEE Standard for Floating-Point Arithmetic*, IEEE, 2008.

[21] ISO/IEC 9899:2011, *Information technology - Programming languages - C*, International Organization for Standardization, 2011.

[22] International Organization for Standardization, "Information technology -- Programming languages -- C," 2011. [Online]. Available: https://www.iso.org/standard/57853.html. [Accessed 2018].

[23] T. W. Good, J. Vilhauer and E. Jacob, "Automatic parallel parking system". USA Patent US4735274A, 5 April 1988.

[24] C. Ching-Wang and J.-M. Shyu, "Automatic parking device for automobile". USA Patent US4931930A, 5 June 1990.

[25] Y. Shimizu and T. Fukatsu, "Automatic driving apparatus". United Kingdom Patent GB2304934B, 19 November 1997.

[26] R. Siegfried and D. Jens, "Process and apparatus for assisting in the parking of a motor vehicle". USA Patent US6097314A, 1 August 2000.

[27] A. M. Palmier and B. Perrin, "Procedure and device for aiding in parking an automobile". France Patent FR2771500A1, 11 February 2000.

[28] B. Perrin, A. M. Palmier and M. Hamidi, "Method and device to assist in displacement of a vehicle especially for the park". France Patent FR2785383B1, 15 December 2000.

[29] B. Perrin and M. Hamidi, "Hilfseinrichtung zum Steuern der Bewegung eines Kraftfahrzeuges, insbesondere im Hinblick auf sein Einparken". Patent EP1043213B1, 28 March 2004.

[30] A. Lisaka, Y. Ueyama and N. Yasui, "Einparkhilfe-Assistenz-Vorrichtung". Patent EP1050866B1, 9 July 2003.

[31] M. Sampo, J. Puputti, K. Rintanen, H. Makela and M. Ojala, "Automatic steering system for an unmanned vehicle". USA Patent US5923270A, 13 July 1999.

[32] G. Asahi, H. Kuriya and K. Shimazaki, "An arrangement for guiding steering to assist parallel parking". United Kingdom Patent GB2357743A, 4 July 2001.

[33] R. Latotzki, "Vehicle autonomous parking system". USA Patent US20170329346A1, 16 November 2017.

[34] M. Ramanujam , "Parking Autonomous Vehicles". USA Patent US20150346727A1, 3 December 2015.

[35] C. Bonnet, A. Hiller, G. Kuenzel, M. Moser and H. Schiemenz, "Method for performing autonomous parking process of motor vehicle e.g. passenger car, involves storing target position and/or last driven trajectory of vehicle in suitable device prior to start of autonomous vehicle parking operation". Germany Patent DE102012008858A1, 8 November 2012.

[36] G. D. Cudak, C. J. Hardee and A. X. Rodriguez, "Identifying cost-effective parking for an autonomous vehicle". USA Patent US20150241241A1, 27 August 2015.

[37] P. N. Rao, D. H. Shin, W. L. Whittaker, K. W. Kleimenhagen, S. J. Singh, C. A. Kemner, W. J. Bradbury, C. L. Koehrsen, J. L. Peterson, L. E. Schmidt and L. J. Devier,

"System and method for enabling an autonomous vehicle to track a desired path". USA Patent US5684696A, 4 November 1997.

[38] J. L. Czekaj, "Semi-autonomous parking control system for a vehicle providing tactile feedback to a vehicle operator". USA Patent US5742141A, 21 March 1998.

[39] P. W. Zitzewitz, "Sound," in *Physics: Principles and Problems*, The McGraw-Hill Companies, Inc., 2005, p. 404.

[40] Wikipedia contributors, "Speed of Sound," Wikipedia, The Free Encyclopedia, 7 October 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Speed_of_sound&oldid=862956622. [Accessed 14 March 2018].

[41] Pololu Robotics&Electronics, "Analog Output vs. Distance to Reflective Object," in *GP2Y0A41SK0F datasheet*, 2002.

[42] Allegro MicroSystems, "DMOS Microstepping Driver with Translator," in *A4988 Datasheet*, 2009.

[43] M. Ekler, "Integrated Stepper Motor Driver Solutions Require Thermal Design Calculations," 7 January 2013. [Online]. Available: https://www.electronicdesign.com/power/integrated-stepper-motor-driver-solutions-require-thermal-design-calculations. [Accessed 13 March 2018].

[44] Reductor-motor, "Stepper motors," 2016. [Online]. Available: http://www.reductor-motor.com/eng-theory-stepper%20motor.html. [Accessed 1 June 2018].

[45] http://roboticsensors.blogspot.com, "Advantages and Disadvantages of ultrasonic distance sensor," 9 November 2015. [Online]. Available: http://roboticsensors.blogspot.com/2015/11/advantages-and-disadvantages-of.html. [Accessed 1 June 2018].

[46] A. Yılmaz, "Raspberry Pi İle Ultrasonik Sensör Örneği," 16 April 2016. [Online]. Available: https://aoyilmaz.wordpress.com/2016/04/16/raspberry-pi-ile-ultrasonik-sensor-ornegi/. [Accessed 1 June 2018].

[47] IEEE Std 100-2000, "The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition," p. 850, 2000.

[48] IEEE Std 100-2000, "The Authoritative Dictionary of IEEE Standards Terms, 7th ed," p. 851, 2000.

[49] diyot.net, "Potansiyometre Nedir?," 22 November 2014. [Online]. Available: http://diyot.net/potansiyometre-nedir/. [Accessed 1 June 2018].

[50] SparkFun Electronics, "What is an Arduino?," 2015. [Online]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino. [Accessed 1 June 2018].

[51] Oriental Motor, "Stepper Motors, Drivers and Controllers," [Online]. Available: http://www.orientalmotor.com/stepper-motors/index.html. [Accessed 2 June 2018].

[52] K. G. Panda, D. Agrawal, A. Nshimiyimana and A. Hossain, "Effects of environment on accuracy of ultrasonic sensor operates in millimetre range," *Perspectives in Science,* vol. Volume 8, pp. Pages 574-576, 2016.

[53] K. Hartman and N. Puckett, "00001," 21 October 2014. [Online]. Available: http://blog.ocad.ca/wordpress/digf6l01-fw201402-01/category/general-posts/page/7/. [Accessed 1 June 2018].

APPENDIX A

```
int echoPin=9;
int trigPin=8;

void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
}

void loop() {
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  float duration = pulseIn(echoPin, HIGH);
  float distance = (duration /2) / 28.5 ;

  if(Serial.read()=='1'){
    Serial.println(distance);
    }
  delay(5);
}
```

APPENDIX B

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace WindowsFormsApp3
{


    public partial class Form1 : Form
    {
        string result;
        string error;
        string difference;
        long maximum = 30;
        long minimum = 0;
        int row = 1;
        int rowNum = 1;
        int i = 0;
        double[] average;

        public Form1()
        {
            InitializeComponent();
            serialPort1.PortName = "COM4";
            serialPort1.BaudRate = 9600;

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            serialPort1.Open();
            timer1.Start();
            button1.Enabled = false;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            timer1.Stop();
            button1.Enabled = true;


        }

        private void timer1_Tick(object sender, EventArgs e)
```

```
        {

            chart1.ChartAreas[0].AxisY.Minimum = 19;  //Measured Value chart - Mini-
mum Y axis value is 19
            chart1.ChartAreas[0].AxisY.Maximum = 21;  //Measured Value chart - Maxi-
mum Y axis value is 21
            chart1.ChartAreas[0].AxisY.Interval = 0.1;      //Measured Value chart -
Interval of Y axis value is 0.1
            chart1.ChartAreas[0].AxisX.Interval = 5;  //Measured Value chart - Inter-
val of X axis value is 5

            chart2.ChartAreas[0].AxisY.Minimum = 0;         //Error chart - Minimum
Y axis value is 0
            chart2.ChartAreas[0].AxisY.Maximum = 4;         //Error chart - Maximum
Y axis value is 4
            chart2.ChartAreas[0].AxisY.Interval = 0.5;      //Error chart - Interval
of Y axis value is 0.5
            chart2.ChartAreas[0].AxisX.Interval = 5;  //Error chart - Interval of X
axis value is 5

            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;         // Measured
Value chart - zoomable chart
            chart1.ChartAreas[0].AxisX.ScaleView.Zoom(minimum,maximum); //MEasured
Value chart - zoomable between max and min values

            chart2.ChartAreas[0].AxisX.ScaleView.Zoomable = true;         //Error
chart - zoomable chart
            chart2.ChartAreas[0].AxisX.ScaleView.Zoom(minimum,maximum);   //Error
chart - zoomable between max and min values


            serialPort1.Write("1");

             //Write serial port the value "1", So Arduino can measure the distance
            result = serialPort1.ReadLine();
            if (result != null)
            {
                label1.Text = result + "cm";
                this.chart1.Series[0].Points.AddXY(minimum+5,result);
                                                                                //Measured
Distance Value Plot the graph
                row = dataGridView2.Rows.Add();

             //Add to row

                this.chart1.Series[1].Points.AddXY(minimum + 5, 20);
                                                                                //Fixed ob-
stacle distance

                dataGridView2.Rows[row].Cells[0].Value = rowNum;
                                                        //Measurement Number - on data-
GridView2
                dataGridView2.Rows[row].Cells[1].Value = result;
                                                        //Measurement - on data-
GridView2
                dataGridView2.Rows[row].Cells[2].Value = (((Convert.ToDouble(re-
sult)-20)/20)*100);                         //Error Percentage - on dataGridView2
                if ((((Convert.ToDouble(result) - 20) / 20) * 100) < 0)
                {
                    this.dataGridView2.Rows[row].Cells[2].Value =((((Convert.ToDou-
ble(result) - 20) / 20) * 100) * -1);   //Error Percentage - negative
                }
```

```
                dataGridView2.Rows[row].Cells[3].Value = Convert.ToDouble(result)-
20;                                                     //Difference - on data-
GridView2

                this.chart2.Series[0].Points.AddXY(minimum + 5, (((Convert.ToDou-
ble(result) - 20) / 20) * 100))                //Error chart - Error value

                if ((((Convert.ToDouble(result) - 20) / 20) * 100) < 0)
                {
                    this.chart2.Series[0].Points.AddXY(minimum + 5, (((Con-
vert.ToDouble(result) - 20) / 20) * 100) * -1);
                }



                rowNum++;
                row++;

                maximum++;
                minimum++;


            }


            serialPort1.DiscardInBuffer();
        // Stop the communication between serial port and application
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string pathImage1 = Environment.CurrentDirectory + "\\Chart1.png";
            chart1.SaveImage(pathImage1, System.Windows.Forms.DataVisualiza-
tion.Charting.ChartImageFormat.Png);

            string pathImage2 = Environment.CurrentDirectory + "\\Chart2.png";
            chart2.SaveImage(pathImage2, System.Windows.Forms.DataVisualiza-
tion.Charting.ChartImageFormat.Png);

            MessageBox.Show("Chart is saved !!!");
        }

        private void button4_Click(object sender, EventArgs e)
        {

        }
    }
}
```

APPENDIX C

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace WindowsFormsApp4
{
    public partial class Form1 : Form
    {

        long maximum = 30;
        long minimum = 0;
        int i = 5;
        string result;
        int row = 1;
        int rowNum = 1;


        public Form1()
        {
            InitializeComponent();
            serialPort1.PortName = "COM4";
            serialPort1.BaudRate = 9600;
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            serialPort1.Open();
            timer1.Start();
            button1.Enabled = false;
            button4.Enabled = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            timer1.Stop();
            button1.Enabled = true;

        }

        private void button4_Click(object sender, EventArgs e)
        {
            serialPort1.Write("1");                         //Write serial port the
value "1", So Arduino can measure the distance
            result = serialPort1.ReadLine();
```

```
            timer1.Stop();

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            chart1.ChartAreas[0].AxisY.Minimum = 0;  //Measured Value chart - Mini-
mum Y axis value is 0
            chart1.ChartAreas[0].AxisY.Maximum = 50; //Measured Value chart - Maxi-
mum Y axis value is 50
            chart1.ChartAreas[0].AxisY.Interval = 1; //Measured Value chart - Inter-
val of Y axis value is 1
            chart1.ChartAreas[0].AxisX.Interval = 1; //Measured Value chart - Inter-
val of X axis value is 1

            chart2.ChartAreas[0].AxisY.Minimum = 0;         //Error chart - Minimum
Y axis value is 0
            chart2.ChartAreas[0].AxisY.Maximum = 4;         //Error chart - Maximum
Y axis value is 4
            chart2.ChartAreas[0].AxisY.Interval = 0.5;      //Error chart - Interval
of Y axis value is 0.5
            chart2.ChartAreas[0].AxisX.Interval = 1; //Error chart - Interval of X
axis value is 5

            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;        // Measured
Value chart - zoomable chart
            chart1.ChartAreas[0].AxisX.ScaleView.Zoom(minimum, maximum); //MEasured
Value chart - zoomable between max and min values
            chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            chart1.ChartAreas[0].AxisY.ScaleView.Zoom(minimum,maximum);

            chart2.ChartAreas[0].AxisX.ScaleView.Zoomable = true;        //Error
chart - zoomable chart
            chart2.ChartAreas[0].AxisX.ScaleView.Zoom(minimum, maximum); //Error
chart - zoomable between max and min values
            chart2.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            chart2.ChartAreas[0].AxisY.ScaleView.Zoom(minimum, maximum);


            if ((result != null))
            {
                label1.Text = result + "cm";

                this.chart1.Series[1].Points.AddXY(minimum + 5, i); // Actual Dis-
tance value
                row = dataGridView1.Rows.Add();

                this.chart1.Series[0].Points.AddXY(minimum + 5, result); //Measured
Distance value

                dataGridView1.Rows[row].Cells[0].Value = i ;   //Put actual distance
to tabular form
                dataGridView1.Rows[row].Cells[1].Value = Convert.ToDouble(result);
//Put measured distance to tabular form
                dataGridView1.Rows[row].Cells[2].Value = Convert.ToDouble(result) -
i; //Put difference to tabular form

                if ((Convert.ToDouble(result) - i) < 0)
                {
                    dataGridView1.Rows[row].Cells[2].Value = (Convert.ToDouble(re-
sult) - i) * -1;
                }
```

```
                dataGridView1.Rows[row].Cells[3].Value = (((Convert.ToDouble(result)
- i)/i)*100); // Put error to datagrid

                if ((((Convert.ToDouble(result) - i) / i) * 100) < 0)
                {
                    dataGridView1.Rows[row].Cells[3].Value = ((((Convert.ToDou-
ble(result) - i) / i) * 100) * -1);
                }

                this.chart2.Series[0].Points.AddXY(minimum + 5, ((Convert.ToDou-
ble(result) - i) / i) * 100); // Put error to chart2

                if ((((Convert.ToDouble(result) - i) / i) * 100) < 0)
                {
                    this.chart2.Series[0].Points.AddXY(minimum + 5, (((Con-
vert.ToDouble(result) - i) / i) * 100)*-1);
                }

                rowNum++;
                i++;
                row++;

                maximum++;
                minimum++;

            }
            serialPort1.DiscardInBuffer();     // Stop the communication between se-
rial port and application

        }

        private void button3_Click(object sender, EventArgs e)
        {
            string pathImage1 = Environment.CurrentDirectory + "\\Chart1.png";
            chart1.SaveImage(pathImage1, System.Windows.Forms.DataVisualiza-
tion.Charting.ChartImageFormat.Png);

            string pathImage2 = Environment.CurrentDirectory + "\\Chart2.png";
            chart2.SaveImage(pathImage2, System.Windows.Forms.DataVisualiza-
tion.Charting.ChartImageFormat.Png);

            MessageBox.Show("Chart is saved !!!");
        }


    }
}
```

APPENDIX D

```
// Sharp IR GP2Y0A41SK0F Distance Test for Accuracy Purposes

#define sensor A0 // IR's analog data cable

void setup() {
  Serial.begin(9600); // start the serial port
}

void loop() {


  float volts = analogRead(sensor)*0.0048828125;  // value from sensor *
(5/1024) to convert voltage level to digital
  int distance = 13*pow(volts, -1); // worked out from datasheet graph
  delay(1000); // slow down serial port

  if(Serial.Read()=="1"){
            Serial.println(distance);
  }

}
```

APPENDIX E

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace WindowsFormsApp4
{
    public partial class Form1 : Form
    {

        long maximum = 30;  //Chart zoom rate - max
        long minimum = 0;   //Chart zoom rate - min
        int i = 4;          // will be used for actual distance value rep-
resentation
        string result;      //will be used for storing measured distance
data
        int row = 1;
        int rowNum = 1;


        public Form1()
        {
            InitializeComponent();          //Initialize serialport element
            serialPort1.PortName = "COM4";  //Microcontroller's connection
to PC
            serialPort1.BaudRate = 9600;    //Serial communication channel
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Do nothing
        }

        private void button1_Click(object sender, EventArgs e)
        {
            serialPort1.Open();             //Start serial communication
            timer1.Start();                 //Start timer element
            button1.Enabled = false;

        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();        //Stop serial communication
            timer1.Stop();              //Stop  timer element
            button1.Enabled = true;
```

```
        }


        private void timer1_Tick(object sender, EventArgs e)
        {
            // Chart specifications
            chart1.ChartAreas[0].AxisY.Minimum = 0;   //Measured Value chart
- Minimum Y axis value is 0
            chart1.ChartAreas[0].AxisY.Maximum = 30;  //Measured Value chart
- Maximum Y axis value is 30
            chart1.ChartAreas[0].AxisY.Interval = 0.5;      //Measured Value
chart - Interval of Y axis value is 0.5
            chart1.ChartAreas[0].AxisX.Interval = 1;  //Measured Value chart
- Interval of X axis value is 1

            chart2.ChartAreas[0].AxisY.Minimum = 0;         //Error chart -
Minimum Y axis value is 0
            chart2.ChartAreas[0].AxisY.Maximum = 4;         //Error chart -
Maximum Y axis value is 4
            chart2.ChartAreas[0].AxisY.Interval = 0.5;      //Error chart -
Interval of Y axis value is 0.5
            chart2.ChartAreas[0].AxisX.Interval = 1;  //Error chart - Inter-
val of X axis value is 1

            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
      // Measured Value chart - zoomable chart
            chart1.ChartAreas[0].AxisX.ScaleView.Zoom(minimum, maximum);
//MEasured Value chart - zoomable between max and min values
            chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            chart1.ChartAreas[0].AxisY.ScaleView.Zoom(minimum,maximum);

            chart2.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
      //Error chart - zoomable chart
            chart2.ChartAreas[0].AxisX.ScaleView.Zoom(minimum, maximum);
      //Error chart - zoomable between max and min values
            chart2.ChartAreas[0].AxisY.ScaleView.Zoomable = true;

            chart2.ChartAreas[0].AxisY.ScaleView.Zoom(minimum, maximum);

            dataGridView1.FirstDisplayedScrollingRowIndex = data-
GridView1.RowCount - 1;

            // Main work is done here !!!!
            // This part allows user to prompt each measured distance value
at a time
            serialPort1.Write("1");                         //Write serial
port the value "1", So Arduino can measure the distance
            result = serialPort1.ReadLine();

                  if ((result != null))
            {
                label1.Text = result + "cm";

                this.chart1.Series[1].Points.AddXY(minimum + 5, i); // Ac-
tual Distance value
```

```
                row = dataGridView1.Rows.Add();

                this.chart1.Series[0].Points.AddXY(minimum + 5, result);
//Measured Distance value

                dataGridView1.Rows[row].Cells[0].Value = Math.Abs(i) ;
//Put actual distance to tabular form
                dataGridView1.Rows[row].Cells[1].Value = Math.Abs(Con-
vert.ToDouble(result);) //Put measured distance to tabular form
                dataGridView1.Rows[row].Cells[2].Value = Math.Abs(Con-
vert.ToDouble(result) - i); //Put difference to tabular form
                dataGridView1.Rows[row].Cells[3].Value = Math.Abs((((Con-
vert.ToDouble(result) - i)/i)*100)); // Put error to datagrid

                this.chart2.Series[0].Points.AddXY(minimum + 5,
Math.Abs(((Convert.ToDouble(result) - i) / i) * 100)); // Put error to
chart2

                rowNum++; //increase row number
                i++;
                row++;

                maximum++;
                minimum++;

            }
            serialPort1.DiscardInBuffer();      // Stop the communication
between serial port and application

        }

        private void chart1_Click(object sender, EventArgs e)
        {
            // Saving chart images as .png files !!!
            // for .png files look for form's main bin !!!
            string pathImage1 = Environment.CurrentDirectory +
"\\Chart1.png";
            chart1.SaveImage(pathImage1, System.Windows.Forms.DataVisuali-
zation.Charting.ChartImageFormat.Png);

            string pathImage2 = Environment.CurrentDirectory +
"\\Chart2.png";
            chart2.SaveImage(pathImage2, System.Windows.Forms.DataVisuali-
zation.Charting.ChartImageFormat.Png);

            MessageBox.Show("Chart is saved !!!"); //saving process is done
        }


    }
}
```