HARDWARE AND SOFTWARE DEVELOPMENT FOR ROBOCUP SSL ROBOTS

by

Huzeyfe Esen

B.S, in Electrical and Electronics Engineering, Pamukkale University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical & Electronics Engineering
Boğaziçi University

2013

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Mehmet Akar for his invaluable support, encouragement, supervision and useful suggestions throughout this thesis work.

I would also like to thank to Prof. Ömer Cerid for letting us benefit from his broad knowledge of electronics.

I would like to present my thanks to Mehmet Öğüt for his support and help in designing the main board.

I am also grateful to Necati Barış for his help in designing the new kicker circuit.

I would also like to extend my appreciation to Ali Haseltalab for his assistance in the studies on Kalman Filter.

I would also like to express my special thanks to Onur Cihan and Ö. Feyza Erkan for their support, sharing their knowledge, guidance and encouragement at each stage of this thesis study.

I am as ever, especially indebted to my parents for their love and support throughout my life. I also wish to thank my brothers and sister for their support during my study.

ABSTRACT

HARDWARE AND SOFTWARE DEVELOPMENT FOR ROBOCUP SSL ROBOTS

In Small Size League (SSL) which is one of the most dynamic and fastest competitions of RoboCup, a soccer game is played between two autonomous teams of 6 robots on a field of size 6050 $mm \times 4050$ mm. Because of this challenging feature of SSL, the robots should be controlled precisely so that they can react in real-time with intelligent soccer-like moves. In this thesis, we not only focus on the electronic board design of the robots, but also introduce new low and high level control algorithms that enable the robots to compete in SSL successfully.

In the new electronic board that controls the main functions of the robot, the microcontroller MC9S08DZ128 and encoders with higher resolution have been utilized for precise control and the features of the motor driver MC33035 have been adapted accordingly. Morover, a new kicker circuit is designed so that the capacitors can be charged in a shorter amount of time.

The algorithmic contributions of this thesis include not only a new low level control algorithm to drive brushless DC motors, but also implementation of the extended rapidly exploring random trees (ERRT) algorithm in our system for path planning and obstacle avoidance. Morover, new formation control algorithms have been developed for coordination of the robots. In this context, a Kalman filter is implemented to compansate for vision based system delay and least squares techniques are used to predict the ball position.

ÖZET

ROBOCUP KÜÇÜK BOY LİGİ ROBOTLARININ DONANIM VE YAZILIM GİLİŞTİRMESİ

RoboCup'ın en hızlı ve hareketli yarışmalarından biri olan Küçük Boy Robot Liginde, 2 otonom takım tarafından 6'ya 6 robotlarla 6050 mm'ye 4050 mm boyutlu bir sahanın üzerinde futbol maçı oynanmaktadır. Ligin bu mücadeleci özelliğinden dolayı, robotlar hassas bir şekilde kontrol edilmelidirler ki gerçek zamanlı olarak tepki verebilsinler ve zekice futbol hareketleri sergileyebilsinler. Bu tezde, sadece robotların elektronik devre tasarımlarının üzerinde durulmamış olunup, yeni düşük ve yüksek seviye kontrol algoritmaları da tanıtılmıştır.

Robotun temel fonksiyonlarını kontrol eden yeni elektronik devrede, hassas kontrol yapabilmek için Freescale MC9S08DZ128 mikrodenetleyici ve yüksek çözünürlükte optik algılayıcılar kullanılmıştır. MC33035 motor sürücü yongaya ait özellikler uygun bir şekilde yeni devreye adapte edilmiştir. Ek olarak, yeni şut çekme devresi tasarlanılarak, kapasitörlerin çok daha kısa bir sürede şarj edilmeleri sağlanılmıştır.

Bu tezin algoritmik katkıları sadece fırçasız doğru akım motorlarını sürmek için gerekli olan düşük seviye kontrol algoritmalarını değil, robotların yol planlama ve engelden kaçınma eylemlerini gerçekleştirebilmelerini sağlayan ERRT algoritmasının sistemimize uyarlanmasını da kapsamaktadır. Buna ek olarak, robotların koordinasyonu için yeni dizilim kontrol algoritmaları geliştirilmiştir. Bu katkılara ek olarak, görüntü tabanlı sistem gecikmesini minimize etmek için Kalman süzgeci kullanılmıştır ve en küçük kareler yöntemiyle de top konumunun tahmini sağlanılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

$ballposition[n]$	The ball position buffered n frames before
$B(x, y)$	Position of the ball
C_{L1}	First value obtained using Least squares solution
C_{L2}	Second value obtained using Least squares solution
C_n	n^{th} center point on goal line
d	Distance between 2 points
D	The frame delay in the system
$D(x, y)$	Desired position of the robot
$D_n(x, y)$	Desired position of the n^{th} robot
$G(x, y)$	The point which is on the goal line
k	The iteration number of RRT simulations
K_d	Derivative gain constant for the motor controller
K_p	Proportional gain constant for the motor controller
l_n	n^{th} line which is depicted in figures in Chapter 5
p	The probability of choosing the goal point as target point in RRT algorithm
pwm	The value which determines the motor's speed
$pwmmax$	The value which pwm cannot equal above
$pwmmin$	The value which pwm cannot equal below
P_n	n^{th} point which is found as a result for desired destination of robots or desired score point
r	The probability of choosing a waypoint as target point in ERRT algorithm
$R(x, y)$	Position of the robot
$R_n(x, y)$	Position of the n^{th} robot
$R56$	Current sense resistor
$R57$	One of the voltage divider resistors for current sensing
$R58$	The other voltage divider resistor for current sensing
S	Free zone on the goal line for scoring

$s(t)$	The state of the robot at time k
S_a	The first channel of Hall sensors of the motor
S_b	The second channel of Hall sensors of the motor
S_c	The third channel of Hall sensors of the motor
SD	Sub-destination point of the robot
$s_\alpha(t)$	The state vector of the augmented system
T	Sampling period in the system
T_n	n^{th} tangent point on the circle which represents the robot
$u(t)$	The control signal at time k
v_{r_x}	The velocity of robot in its origin coordinate in the direction of x
V_{r_x}	The velocity of the robot in the direction of x
v_{r_y}	The velocity of robot in its origin coordinate in the direction of y
V_{r_y}	The velocity of the robot in the direction of y
v_{r_α}	The velocity of robot in its origin coordinate in the direction of θ
v_x	The speed of the robot in the direction of x at time t
v_y	The speed of the robot in the direction of y at time t
x_{r_n}	The horizontal distance of the n^{th} wheel from robot's center
$x(t)$	The position of the robot on x coordinate at time t
y_{r_n}	The vertical distance of the n^{th} wheel from robot's center
$y(t)$	The position of the robot on y coordinate at time t
α	Angle which is used for localization computations
α_n	The orientation angle of the n^{th} wheel of the robot
β	Angle which is used for localization computations
$\omega(t)$	The rotational speed of the robot at time t
ω_n	The velocity for the n^{th} wheel of the robot
β	Angle which is used for localization computations
$\theta(t)$	The orientation angle of the robot at time t

LIST OF ACRONYMS/ABBREVIATIONS

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
BCD	Binary Coded Decimal
DC	Direct Current
ERRT	Extended Rapidly-Exploring Random Tree
FPGA	Field Programmable Gate Array
IBM	International Business Machines
IC	Integrated Circuit
ID	Identity of the Robot
IEEE	The Institute of Electrical and Electronics Engineers
IO	Input/Output
IR	Infra Red
ISM	Industrial Scientific Medical
Li-Po	Lithium Polymer
MC	Microcontroller
MRS	Multi-Robot Systems
NMOS	N-type Metal-Oxide-Semiconductor
PCB	Printed Circuit Board
PID	Proportional, Integral, Derivative
PMOS	P-type Metal-Oxide-Semiconductor
RF	Radio Frequency
RRT	Rapidly-Exploring Random Tree
SCI	Serial Communications Interface
SPI	Serial Peripheral Interface
SSL	Small Size League
TPM	Timer Pulse-Width Modulator
UART	Universal Asynchronous Receiver/Transmitter

1. INTRODUCTION

Technology is developing in a tremendous speed. The flight adventure set a great example for this argument. Until 1903 while there was not any airplane; the Wright brothers invented and built the world's first successful airplane [4]. Later in 1969 for the first time human beings landed on the moon with Apollo 11 flight [5]. In short, human beings managed to travel from planet to planet, although the first airplane had been invented only 60 years before. The inventions which are known worldwide are not limited only with the flight adventure. For artificial intelligence and robotics, the year 1997 is a time at which impressive events occurred. Firstly in May, the human world champion in chess was defeated by IBM Deep Blue [6]. It is just after 50 years from the date in which the first electronic general-purpose computer was announced to the public (1946) [7]. AI community finally achieved success in their scope. Secondly in July, the first autonomous robotics system, Sojourner, was used on the surface of Mars [8]. These are important examples in showing the advancement of technology in a short amount of time.

Artificial intelligence and robotics are the research areas which contribute to technology. In order to encourage people to study in these areas and to make these studies more exciting, some promoting ways are being utilized and Robocup Organization is one of them [9]. The ultimate goal of the RoboCup is that in 2050 a fully autonomous humanoid robot team will play the winner of the World Cup in a soccer game [10]. Even the dream of this creates sufficient energy and motivation in order for people to be lured into Robocup. It is known that in 50 years there can be many improvements in many areas related to technology and some of them are given as examples above. This is actually why this goal is more than just a dream and why it continues to attract people's attention.

Although Robocup seems to lure people to build a soccer robot team, it promotes AI and robotics and drives research in many areas. Controlling the movements of the robots, establishing communication between them, localization of the robots and

detecting of the locations of them contributes to make improvements in control algorithms, machine vision, real-time distributed computing, real-time ad hoc networking, mechanical design, machine learning and autonomous multiagent systems are some of them.

1.1. History and Organization of Robocup

The first attempt for the robots playing soccer was made by Proffessor Alan Mackworth via his paper in 1992 [11]. Later in Tokyo a Workshop on Grand Challenges in Artificial Intelligence was organized and the contributions of the game of soccer to the technology and science were discussed and the robots playing soccer was concluded as feasible and desirable [11]. After many conferences were held and various ideas were come up with about robot soccer, and finally the first official Robocup games were held in 1997, in Japan with the participation of over 40 teams [11]. Since 2003, Turkey participates in Robocup and Turkey held the organization in 2011. Different leagues are available for teams to participate in. The leagues are listed under 4 main categories [12]:

- Robocop Junior
- Robocop Rescue
- Robocup @Home
- Robocup Soccer
  - Simulation League
  - Small Size League (SSL)
  - Middle-Size League
  - Standard Platform League
  - Humanoid League

Since our thesis is completely related to soccer part of this organization, sub leagues of Robocup Soccer will be briefly explained.

1.1.1. Simulation League

This league encourages teams develop artificial intelligence and game strategy. It is not expected for teams to build real robots. The games are played on a virtual field. It can be said that being successful in this league only depends on the achievement in your software.

1.1.2. Small Size League (SSL)

In this league, electronics and mechanics are crucial research areas as much as software. 6 by 6 robots are challenging on a field with size of 6050 by 4050 millimeters. The game is played with an orange golf ball. There are two cameras located over the field for observing the robots. Two teams competing on the field are informed with the location information of the ball and all robots. The control of the robots is centralized based. As our thesis is based on this platform, it will be explained in detail in Section 1.1.2.

1.1.3. Middle-Size League

In this league, robots are notably larger when compared to SSL ones. The robots are not controlled centrally as is the case with SSL. Each robot has its own sensors and camera. Omnidirectional vision is allowed. Each one has the information of environment around itself, and the robots can communicate with each other via wireless network. The games are played with a real soccer ball. The teams should design and fabricate their own robots by themselves; there is not still standardization in robots.

1.1.4. Standard Platform League

This league differs from the other ones in their robots. It is based on Aldebaran's Nao humanoids. All teams use standard robots, each robot is identical. Omnidirectional vision is not allowed. The robots should make decisions by sharing the information of their local environment with each other.

1.1.5. Humanoid League

Human-like robots are competing with each other. Dynamic walking of the robots is in itself a serious research issue in this league. Self-localization and kicking the ball without disrupting the balance are the other research areas investigated for this league.

1.2. Small Size League

In this thesis, we are concerned with designing the electrical part of SSL robots and developing new artificial intelligence (AI) algorithms. The components of the overall system are described below.

1.2.1. Robots

SSL match is not as a simulation league game where the artificial intelligence is enough for your team to be able to compete. You need to have a system that can interpret your AI and adapt it to real life. If you do not have well-equipped robots, your AI will be useless and vice versa. The robots consist of two main parts: electrical and mechanical. Each part must be compatible with each other. If a mechanical design is made, your electrical system should fit this design and should be able to drive it. Actually this is the point where SSL differs from the other leauges. You should handle with software, mechanics and electronics all by yourself; there is not standardization in these parts. Each team should design their mechanics, electronic and program software on their own. If you are to compete in SSL, you will need robots besides AI. Firstly your robots must satisfy some criteria [13]. They cannot exceed 15 cm height and should fit inside a 18 cm diameter cylinder. In general, robots consists of 4 wheels, each rounded by small O-rings which help robots move omni directional, that is, robots can move at each angle in accordance with the direction of force created by the motors. A typical SSL robot can reach a speed of 3 m/s. The robots should have 2 different types of kicking mechanisms: chip kick and flat kick. There is only one limitation on kicking that the speed of the ball cannot exceed the speed of 8 m/s after it is kicked. Furthermore robots should have the ability of dribbling the ball. This is not

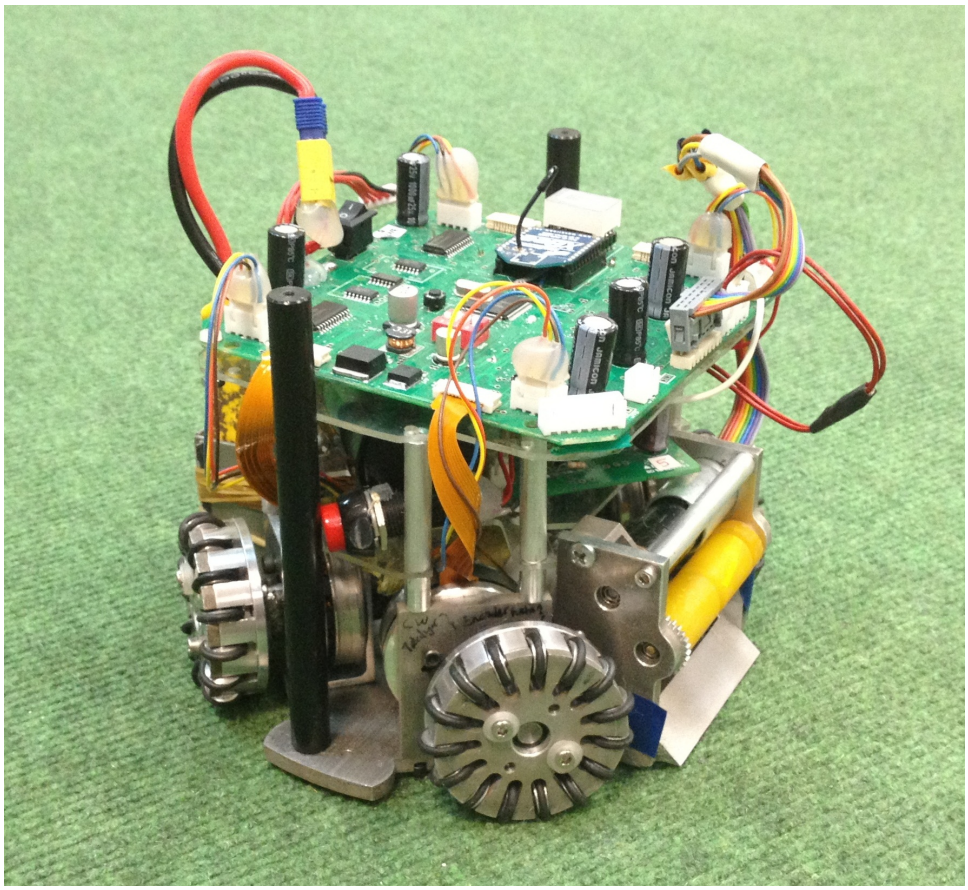

Figure 1.1. The autonomous omni-directional robot

a rule but it is a key to competing successfully. However, there is a restriction related to dribbling. The dribbler system should not cover more than 20% of the ball. The robots should have an electronic board which can receive data which is transmitted from main computer and turn this data into meaningful signals for driving motors and for running the other systems of the robot such as kicking and dribbling. Our robot is depicted in Figure 1.1.

1.2.2. Cameras

Over each SSL field, there are 2 cameras which cover the whole field and they are attached to a bar 4 m above this area. As the size of the field (6.05 m $\times$ 4.05 m) is not suitable for one camera, it is required to use a second camera to cover the field. Actually doubling the size of the field is on to do list of the organization community, and if this proposal is accepted, the number of the cameras will be increased to 4. The

cameras are used for transmitting the vision data which contain the robots and the ball inside, to the computer which is supplied for each field by the organization. As the cameras have the feature of taking approximately 60 frames per second, you have the chance of controlling the robots smoothly in every change of environment.

1.2.3. SSL Vison

Each field is equipped with two computers provided by the organization itself. One of them is for SSL vision and the other one is for Referee Box. Before standardization of the vision system, each team was obliged to program their own image processing software. One computer is assigned just for this software. After the video is transmitted from the cameras to related computer, it is processed to detect the position information of the robots and the ball. The detection of the robots depends on the patterns over the robots. Each robot has a unique pattern. The pattern consists of 5 colorful cylindric papers. One of them locates in the middle of the pattern and the others make a circle around it. While the one in the middle refers to the team, the others refer to the ID of the robot. After the vision data has been processed by the SSL vision software, the positions of the robots and the ball are detected, and this information is transmitted to the computers of each team.

1.2.4. Referee Box

The other computer which is supplied by the organization for each field is for Referee Box software. In an SSL match, the game is played according to the rules as is the case with real football matches. There are many similarities between real soccer games and SSL games. Indirect free kick, throw in, corner kick are the examples of the similarities. The game is ruled by a human referee as in the real soccer game as well. However, there is also a software named as RefereeBox which helps the game to be played properly. The human referee checks the game and gives the instructions to the Referee Box expert in order for him to give the related commands to each team by using Referee Box. For example if the game needs to be stopped, this command is transmitted to the main computers of each team which utilize the information to

position the robots on the field.

1.2.5. AI Module

In order to be able to play an SSL game according to the rules and compete successfully, an artificial intelligence module should be developed properly. This module is the brain of a SSL team. All game strategies, path planning and obstacle avoidance algorithms, formation control algorithms, player states, game states should be implemented inside this module so that teams can challenge to each other in SSL.

1.3. The Organization of the Thesis

This thesis has 7 chapters. While Chapter 1 is the introduction that gives background information on the Robocup and the SSL competition, the rest is organized as follows:

In Chapter 2, the circuit design of SSL robots is provided. Detailed information on the wireless module, the microcontroller, the motor driver, the voltage regulators, the encoders and other fundamentals is given. Additionally new kicker circuit is designed. PCB and schematics designs are also provided in this chapter.

In Chapter 3, a new low level control algorithm is introduced. Some problems that arise during the implementation of this algorithm and proposed fixes are also discussed in detail.

In Chapter 4, path planning and obstacle avoidance algorithms are proposed. Different type of RRT algorthms, some of which are also implemented on our system, are discussed and compared with each other.

Chapter 5 is on formation control. Specifically, game and player states which are required for playing a SSL match properly are discussed. The relation between formation control and these states are explained. New approaches are adopted to

formation and localization issues in accordance with SSL matches.

In Chapter 6, a Kalman filter is proposed to compensate for network induced vision delay. Simulation and experimental results are also given to verify intense performance of the proposed estimator.

Finally, Chapter 7 provides the concluding remarks.

2. ELECTRONIC BOARD HARDWARE DESIGN

2.1. Hardware Design Issues

In order to enable the robots to play soccer properly and the user to have control over the robots, an electronic board has to be designed. The criteria that should be taken into consideration in such a design are listed below.

- The circuit size must obey the SSL (Small Size League) rules. Firstly, an SSL robot cannot exceed the size limits of 15 cm height and 18 cm width [13]. The electronic board should be designed in such a way that the robots should not violate this rule. Apart from this limitation, there are other restrictions which must be taken into consideration, e.g., ball dribbling distance and kicking velocity cannot exceed 500 mm and 8 m/s [13], respectively.
- The board should react fast enough so that robots will be able to continue without any stoppage. Wireless module data transfer rate, encoder data capturing rate and control algorithm update rate are some of the time dependent processes.
- Some feedback mechanisms should be incorporated into the design in order for the user to be in control of the robots. Setting the robot identity, detecting the error signals, controlling battery voltage level are some of these mechanisms.
- In the electronic board design, the components should be identified and obtained before final design so that no problems in board production are encountered.

The general architecture of the main board that is designed based on the above design criteria, is depicted in Figure 2.1. Our design consists of the following subparts which are detailed in the rest of the chapter:

- Communication Unit
- Microcontroller Unit
- Motor Driver Unit
- Voltage Regulator Unit

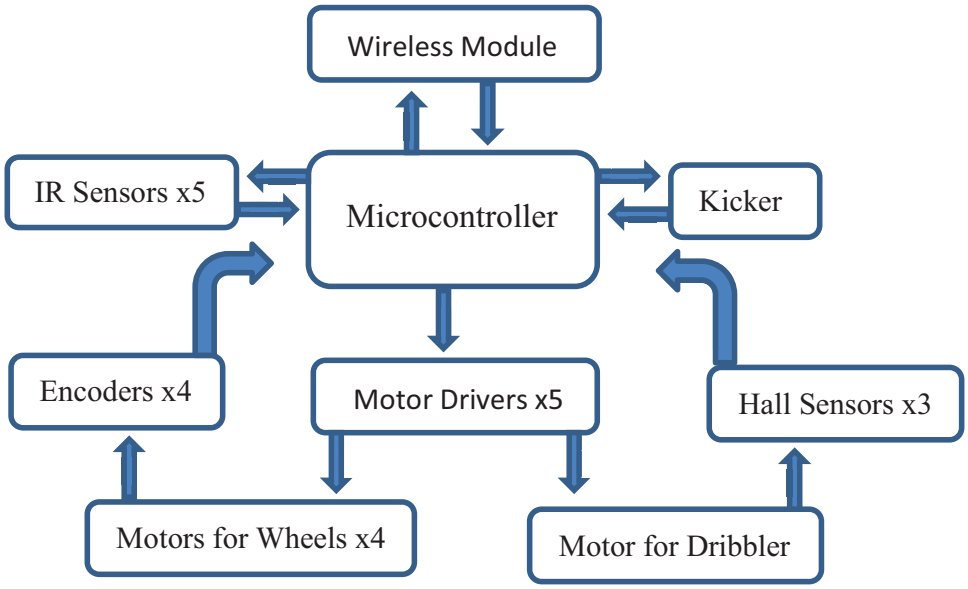

Figure 2.1. General structure of the new circuit design.

- Encoder Unit
- Kicker Unit
- Other Functionality

2.2. Communication Unit

The communication between the main computer which runs the artificial intelligence algorithm and the robots is provided by a wireless module. The data transmission is bi-directional. While the robots transmit some data such as the kicking capacitor and the battery voltage levels, the main computer transmits the motor speed values of the robot and the kick data. Occasionally, it may be required to transmit actual speed values of wheels to the main computer for debugging purposes in addition to the capacitor and the battery voltage levels. These 3 cases are explained below:

- The kick capacitor voltage level: The capacitor voltage level refers to the voltage which will be discharged through the solenoids. This capacitor level is transmitted to the main computer for the purpose of observing the functionality of the electrical part that is related to kicking mechanism.
- The battery voltage level: Battery voltage level is transmitted to the main com-

puter to monitor the stamina of the robots. Due to the nature of Li-Po batteries this is needed since it is highly possible that the battery will not be able to be used again if the voltage level drops below a certain value.

- Wheel speeds: In order to debug the actual and desired speed values of the wheels, sometimes actual speed values can be transmitted to the main computer.

During the game, battery and capacitor voltage levels are solely transmitted at a rate of 1 Hz. There is no need to receive this information more frequently because these data are just used for observing the health of the robots. However, as the cameras have 60 Hz update rate, the new wheel speed data are updated 60 times in one second and are transmitted to the robots.

The communication data packet between the main computer and the robots is directly related to the number of robots. First of all, the start byte is included in the package in order to indicate that the start of the transmission. Following the start byte, the package consists of 5 bytes. While 4 bytes refer to new wheel speed data, the other byte refers to the kicking and dribbling information. As 6 robots represent the team on the field, the space of 31 bytes is required for the robots (see Figures 2.2 and 2.3). At each frame, these bytes are transmitted to the robots. As there are 60 frames in 1 one second, it means that 1860 bytes, or equivalently 14880 bits. The module which will be used in the project must satisfy this data rate.

XBee Series 1 RF [1] shown in Figure 2.4 is used as wireless module in our design for meeting our demands. This module is engineered to meet IEEE 802.15.4 standard. It requires minimal power and provides reliable delivery of data between devices. It operates within the ISM 2.4 GHz frequency band. It can communicate with other modules via the UART communication protocol. The RF data rate is 250000 bits per second, much higher than what we need.

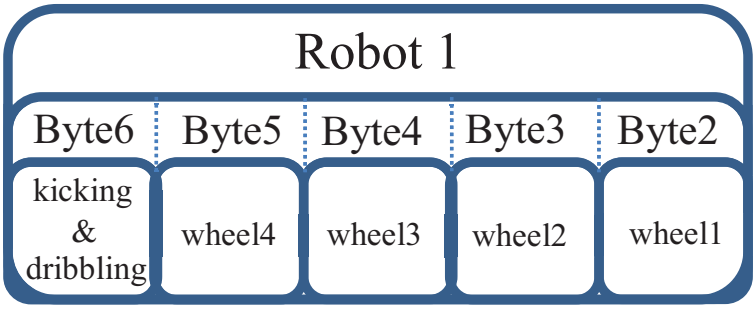

Figure 2.2. Wireless Data Package for 1 Robot.

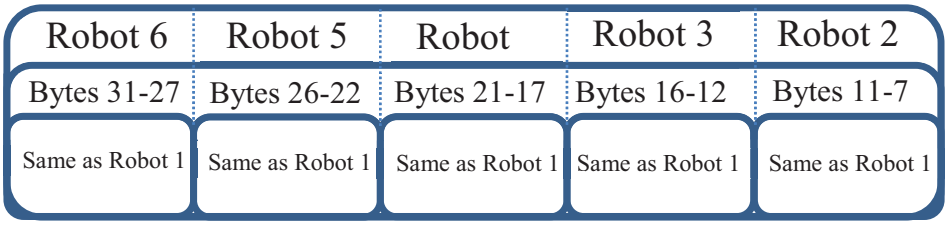

Figure 2.3. Wireless Data Package for Other Robots.

Figure 2.4. XBee Series 1 [1].

2.3. Microcontroller Unit

The microcontroller is the most important part of the design due to its several responsibilities for controlling every unit in the robot. On board, there are many units which the microcontroller must take care of. Firstly the communication between the main computer and robots is satisfied via XBee wireless modules. The microcontroller should have the ability of receiving data from this module and transmitting data to this module. Since XBee modules can communicate via UART protocol, the microcontroller should utilize the SCI module. After processing these data, MC (Microcontroller) should control all the motors in the robot based on the received data. There are 5 motors in one robot: 4 of them are for the wheels and the last one for dribbling. To control these motors, the MC should have enough TPM (Timer Pulse-Width Modulator) channels. The MC should have one TPM pin for sending PWM (Pulse Width Modulation) signal to each motor which is created by these TPM channels.

The MC should have the ability of capturing encoder and Hall Sensor signals. One encoder is connected to each wheel and these are used for identifying the speed data. Each motor has Hall sensors but the Hall sensors of the dribbling motor are only used for detecting the speed of motor.

An IR sensor circuit which is attached to our robot for detecting the ball position is also used by the MC. There are 5 IR transmitters and 4 IR receivers on this IR circuit. The output signals of receivers should be processed by analog to digital converters. As the microcontrollers can include ADCs (Analog to Digital Converter) inside, it is not required to use different ICs for processing these signals. The MC should have the ability of converting analog signals to digital data. It is noted that each receiver requires one ADC channel.

Additionally, the MC should control the kicking mechanisms. It has enough IO and ADC pins for it. In order to control the chip and flat kick mechanisms, it should have 2 IO pins and one ADC pin to measure the voltage level of the capacitors. For

setting the robot ID (Identity) and for indicating this ID, the MC should have enough IO pins. For debugging purposes extra IO, TPM, ADC, SCI, SPI pins are required as well.

In the previous mainboard design, MC9S08DV32 [14] microcontroller was chosen. As the MC9S08DV32 could not satisfy the above requirements by itself, two MCs were used in the design. While the majority of the requirements can be satisfied by one MC, there are not enough TPM channels. While one MC was responsible for capturing the encoder data, the other MC was sending PWM signals to the motors. This was how the issue of TPM channels was resolved. On the other hand, sharing the responsibilities between two MCs was not an efficient solution in terms of synchronization and latency. The other problem was that programming and debugging two MCs simultaneously were not possible. Each one must be programmed and debugged separately. Whilst programming the two one by one was not causing serious difficulties, debugging them was causing some problems. When debugging, the user needed to have the whole control over the all threads which were being run. Besides, two MCs were taking up plenty of space on the PCB (Printed Circuit Board). When all drawbacks are taken into consideration, it is clearly noted that one MC should be preferred over using two MCs. This is why MC9S08DZ128 [2] is chosen in our new design.

Although the DV32 and DZ128 have the same CPU (40 MHz HCSO8) inside, DZ128 has more capacities than DV32 in terms of general-purpose IO and peripherals. While DV32 especially lack in TPM channels (8), there is abundant channels (12) in DZ128. DZ128 has 24 ADC channels, 2 SCI modules, 2 SPI modules, 87 general-purpose IO pins, while DV32 has 16 ADC channels, 2 SCI modules, 1 SPI module, 53 general-purpose IO pins. Pins and peripheral capabilities are shown in Figure 2.5.

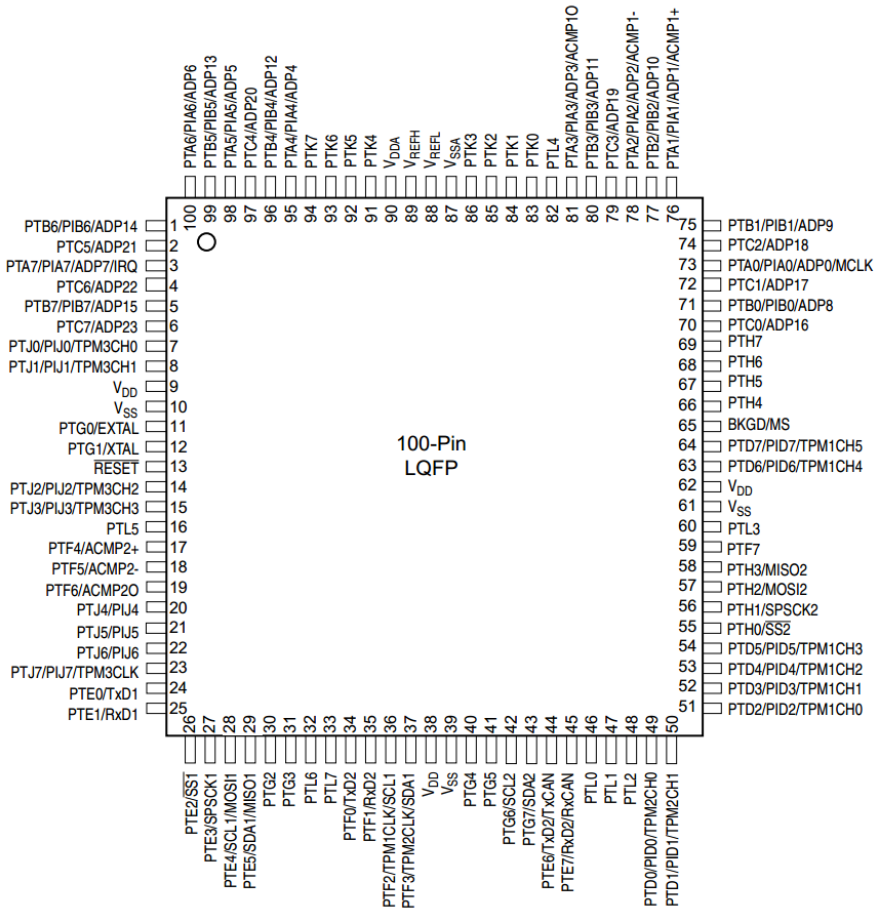

Figure 2.5. Pin Connections of MC9S08DZ128 [2].

2.4. Motor Driver Unit

There are 5 motors in each robot: 4 of them are Maxon EC 45 [15] for the wheels and the last one is Maxon EC 16 [16] for dribbling. They all are brushless DC motors with 3 phases. These motors should be driven properly before they can be controlled. There are many ways of driving these motors but the optimal way should be determined for our design since controlling three phases will cause complexity and consume more space on board. When driving a motor with 3 phases is compared to 1 phase one, the 3 phases motor is obviously far more complicated than the other one due to the synchronization issue. That is not a concern in a 1 phase DC motor. In order to drive 3 phase motors, it is required to drive each phase in a certain sequence. This automatically requires 3 TPM channels of the MC. If it is considered that we have 5 motors, 15 TPM channels will be used just for sending the PWM signals. This is not possible just with one MC or even with two MCs. In our design a special IC MC33035 [3] is used to prevent this complexity.

MC33035 is a brushless DC motor controller containing all of the active functions required to implement a full featured open loop three phase motor control system. Using this IC, the motors can be driven and controlled by MC as if one phase is being controlled, because sending speed information to MC33035 from MC is sufficient for driving motors. With only speed information, MC33035 will drive 3 phases by itself. The speed information from MC is sent as a PWM signal. That gets through a Low Pass Filter so that the PWM signal is converted into a reference DC voltage value. MC33035 drives the motor according to the DC voltage value, not to the PWM signal. This voltage signal is connected to pin 11 as shown in Figure 2.6.

The sequence information for driving 3 phases is obtained by observing the Hall sensor outputs of the motor. MC33035 receives these sensor outputs and controls the motor based on the sequence of Hall sensors. These sensor outputs are connected to pins 4-6 of MC33035 (S_A, S_B, S_C). There are 3 Hall sensor outputs for each motor (see Figure 2.7).

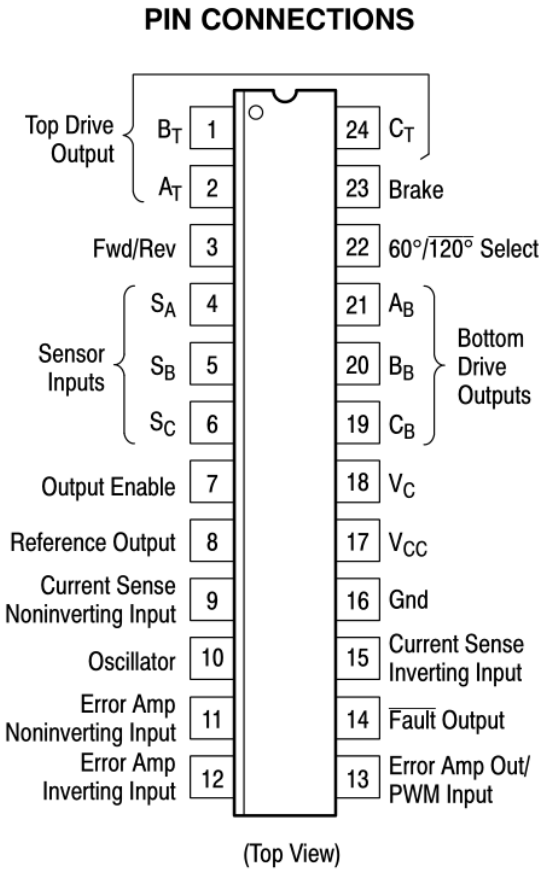

Figure 2.6. Pin connections of MC33035 [3].

While these sensor signals are being processed, MC33305 controls the bottom and top drives simultaneously. There are 6 outputs of this IC for contributing to driving motors. These outputs are connected to the H-bridge via resistors. The H-bridge consists of 3 NMOS and 3 PMOS. These mosfets exist in our design in dual format. FDD8424 is an IC which includes PMOS and NMOS in this way. Thus there are 3 dual mosfets for each motor on the mainboard. The bottom drive outputs of MC33035 are connected to the gate of N-channel mosfets, and the top drive outputs are connected to the gate of P-channel mosfets. The circuit driving one phase is depicted in Figure 2.8.

In order to protect the circuit and the motor, MC33035 has many features [3] such as under voltage lockout, cycle-by-cycle current limiting with a selectable time delayed latched shutdown mode, internal thermal shutdown, and fault output that

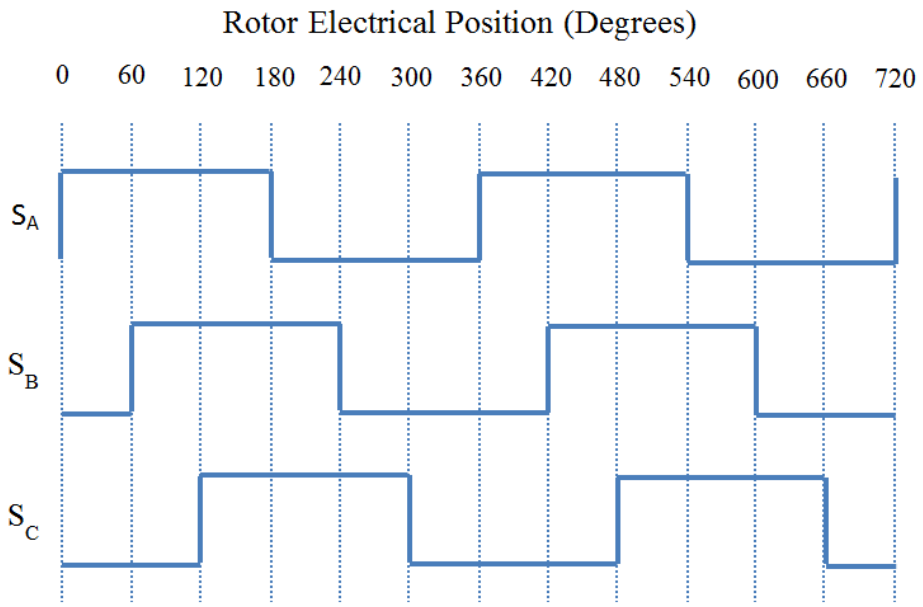

Figure 2.7. The sequence of Hall Sensors [3].

shows whether there is something wrong or not. The most important feature that we utilize is the overcurrent protection.

A motor draws current according to the load on it. Sometimes, something is stuck in the connection between the wheel and the motor, or directly inside the motor. Under such circumstances, the motor will draw full load current. If this lasts for a long time, the circuit and the motor can be damaged. MC33035 has a perfect solution for this problem. Figure 2.9 shows the connections for sensing the current. One resistor ($R56$) is connected to the path leading ground, that is the current which flows through the motor. The most important thing is that the resistor must be at a low value so that the voltage dropout over the resistor will be tolerable. In our circuit, the value of $R56$ is chosen 0.05 Ohm. When the current drawn by the motor flows through the resistor, a voltage dropout over it is created. This voltage can be proportionally divided over two resistors ($R57, R58$), which is connected to the Current Sense Non-inverting pin 9 of MC33035. If the voltage value on pin 9 exceeds 100 $mVolt$, the IC shuts itself down in order to prevent the over current from damaging the circuit and the motor.

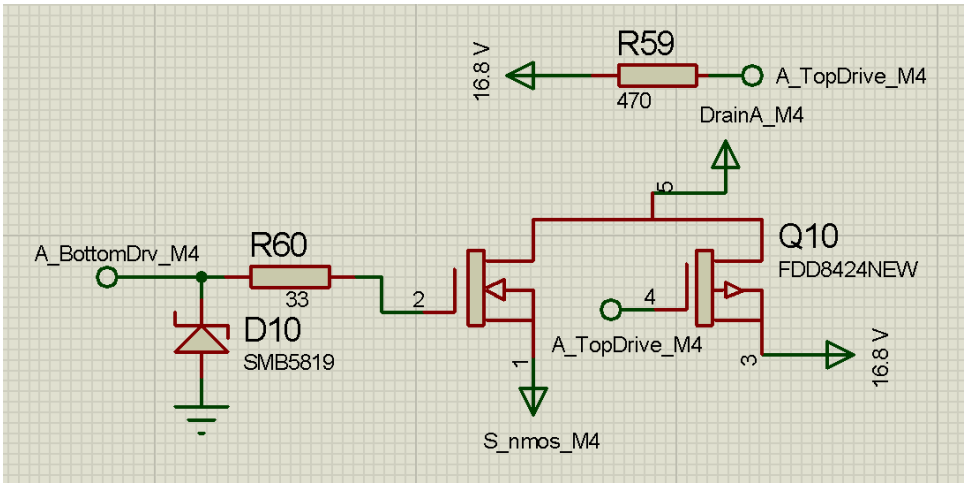

Figure 2.8. The Schematic of driving one phase.

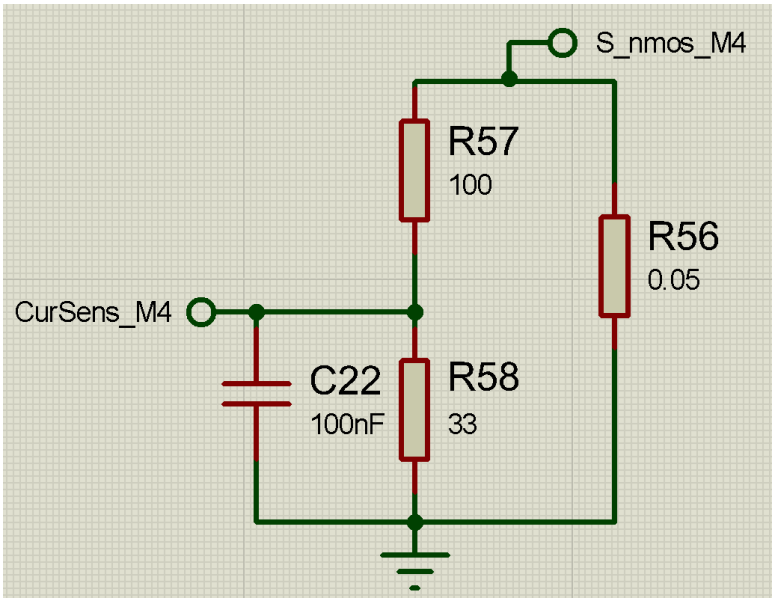

Figure 2.9. The schematic of current sensing.

2.5. Voltage Regulators

In our design, we are using several units for controlling our robot; one XBee communication module, one MC9S08DZ128 microcontroller, one 74HC86 XOR gate, 2 74HC74 positive edge triggered D flip-flops, 4 Encoders, 5 Motors, 5 MC33035 Motor Drivers, 5×3 Hall sensors. Each unit requires different voltage levels. Our motors and MC33035 motor drivers are the units which run at 16.8 V. The Xbee module runs at 3.3 V, and all other units mentioned above function at 5 V.

First of all the type of battery that will be used in our system for powering all units has to be decided on. Since our robots may consume all of the capacity in a short duration, we need a battery which can be charged quickly. This is why a 16.8 V, $4S$ 1750 mAh $20C$ series Li-Po battery is used as our main voltage source.

While some of the units require to be powered by 16.8 V, some require to be powered by 5 V or 3.3 V that are obtained by voltage regulators. In order to provide higher power conversion efficiency, eliminate overheating and reduce power dissipation, switched-mode regulator is preferred to linear voltage regulator for 5 V in our design. Since converting 5 V into 3.3 V does not create any problem, linear regulator is used for 3.3 V.

One of the voltage regulators that is used for obtaining 3.3 V is LM1117 [17]. The other one is LM2575 [18] for obtaining 5 V. As mentioned above, LM2575 has a different structure when compared to LM1117. They are called switched-mode voltage regulators whose working principle is based on switching the circuit using a transistor by enabling and preventing the current flow through the inductor as shown in Figure 2.10.

As long as the transistor is in the ON-mode, the capacitor is charged by the inductor current. When it is in the OFF-mode, the connection between the power supply and inductor is cut-off. The current which is already over inductor flows through the load and completes its path via passing through the diode. The charged capacitor

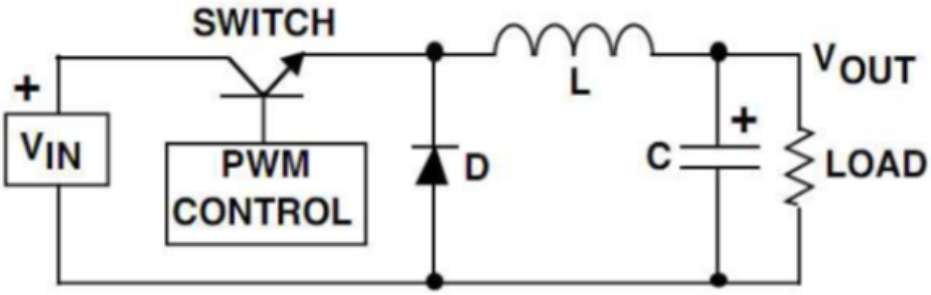

Figure 2.10. The basic principle switched-mode voltage regulators.

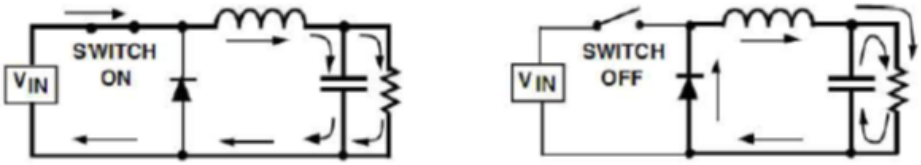

Figure 2.11. Switching phases of switched-mode regulators.

is discharged through the load. The current over the inductor increases and decreases as long as switching mode changes from ON to OFF and from OFF to ON. Stable DC voltage is obtained by setting the ON and OFF durations precisely. The functionality of the circuit is shown in Figure 2.11.

The application of LM2575 in our design is depicted in Figure 2.12.

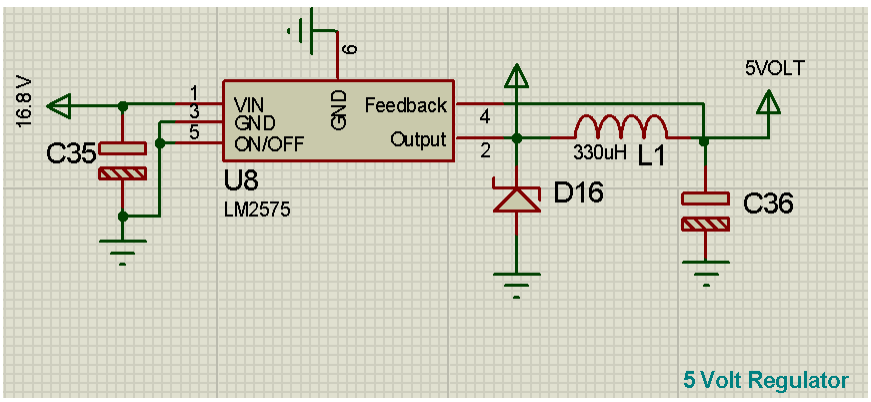

Figure 2.12. The schematic of 5V voltage regulator.

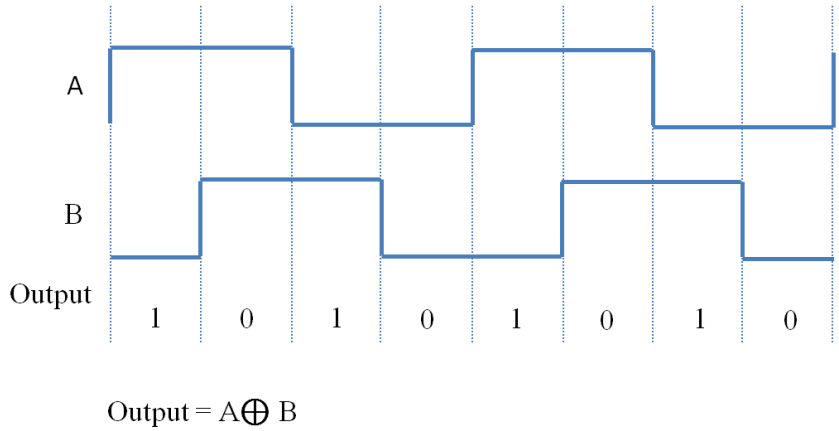

Figure 2.13. Encoder channels are XORed in order to increase the resolution of pulses.

2.6. Encoder Unit

In our previous design, Hall Sensors were used in order to detect the actual speed of the motors by creating 48 pulses per revolution. Each pulse refers to 7.7 degrees, i.e., one pulse may correspond to the wheel turning 1 or 7 degrees. Encoders are used instead of Hall sensors in our new design for detecting speed value of wheels in order to get higher resolution. US Digital E4P [19] encoder that provides 360 cycles per revolution is used in our new design. The encoder has two channels and each channel is shifted by 90 degrees. When these channels are decoded by XOR gate (74HC86) [20], 720 cycles are obtained per revolution. When both the positive and negative edges are used, 1440 pulses are obtained. Figure 2.13 depicts how the resolution is doubled when the channels are decoded. The detailed analysis and comparison of Encoders and Hall sensors is discussed in Chapter 3.

Motor rotation direction is an other issue on which we focused in our new design. When controlling a motor, it is important to know the direction of rotation. The previous control algorithm did not depend on the actual rotation of the motor, i.e., whether the motor was turning at the desired direction or not, the algorithm was assuming that the motor was turning at the desired direction. In the new design, the direction of rotation is detected in real time by utilizing the information in the two channels inside the encoders. As these channels have 90 degrees phase difference, the

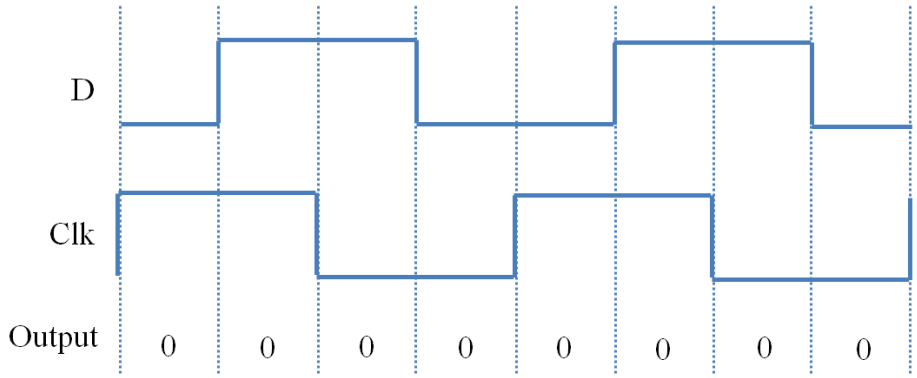

Figure 2.14. Channel A is connected to the input of D Flip-Flop, Channel B is connected to clock. Channel B is following Channel A.

outputs of the channels differ when the motor moves in counterclockwise or clockwise, i.e., the sequence of their outputs changes according to the direction of the movement. While channel A is following channel B in one direction, channel B follows the channel A in the other direction. In order to take advantage of this situation, a positive edge triggered D flip-flop is utilized in our circuit. While the output of one channel is connected to the input of D flip-flop, the output of the other channel is connected to the clock input of D flip-flop. Assume that channel A is connected to the data input and channel B is connected to the clock input. If channel B follows channel A, the output of D flip-flop will be logic 0 on the positive edge of the channel B (see Figure 2.14). If channel A follows channel B, the output will be logic 1 on the positive edge of the channel B (see Figure 2.15).

The connections related to Encoder are shown in Figure 2.16.

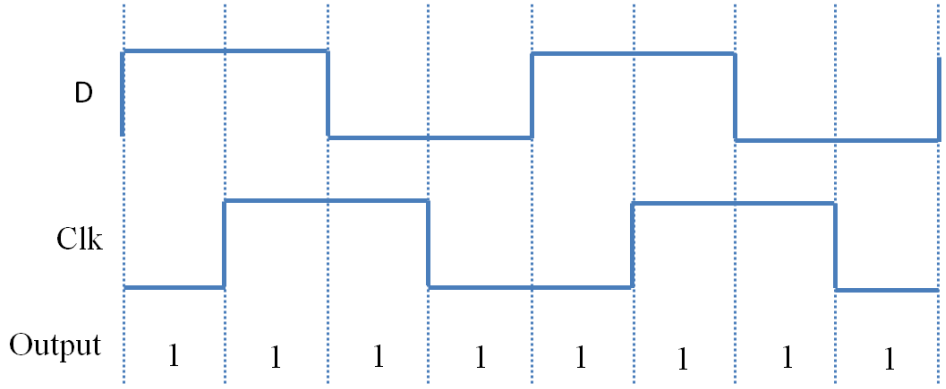

Figure 2.15. Channel A is connected to the input of D Flip-Flop, Channel B is connected to clock. Channel A is following Channel B.

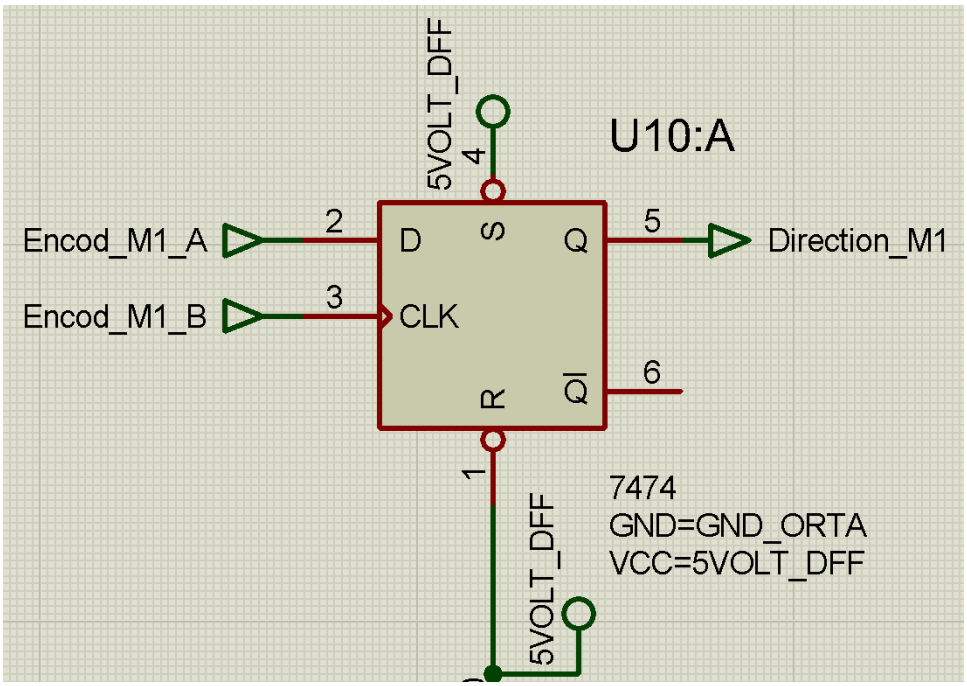

Figure 2.16. Schematics of circuit design related to detecting motor rotation direction using D Flip-Flop.

2.7. Other Functionality

In order for the robots to react properly, they have to get the correct data intended for them. This data is stored in the MC after being received by XBee on board and then transmitted to MC. By processing the data which is available inside the MC, the robot should detect the related data to itself. Each data packet transmitted from the main computer includes new desired speed values for wheels and kicking information for each robot (see Figures 2.2 and 2.3). In order for the robot to pick out the necessary information, an identity number must be given to each robot. Hence, once the main computer sends the data according to the patches over the robots, each robot picks the necessary part out from the whole data packet based on its own identity. The patches can be considered as jerseys of the robots. They have 5 colored circles on them. One of them (yellow or blue) refers to the color of team; the others (green and pink) refer to the robot identity. Assume that there is only one robot on the field, and this robot number is 4 according to the patch over itself. The main computer transmits data for the robot as it is robot number 4. However the robot electronically must know that it is robot number 4. While the patch shows that the robot is number 4, it should not be number 3 electronically for proper communication. This is why setting ID (identity) electrically for each robot is very important. Morover, observing the IDs electrically is as much important as setting them. If the robot IDs are not observable easily, it is possible that the IDs of patches and the IDs of robots cannot be matched, which leads to communication problems between the main computer and the robots. In our previous designs, the robot ID was set by programming the microcontroller. There was no any indicator showing the robot ID. As there was no any indicator above the circuit, when which ID belongs to which robot was forgotten, the microcontroller was being programmed from the scratch.

In our new design, the robot ID is set directly by the switches which are located on the mainboard instead of setting it by programming the microcontroller. There are 4 switches directly connected to the MC for setting the robot ID and one 7 segment display for observing the robot ID. By changing the positions of the switches, the robot ID is set according to the code inside microcontroller. The ID information is sent to

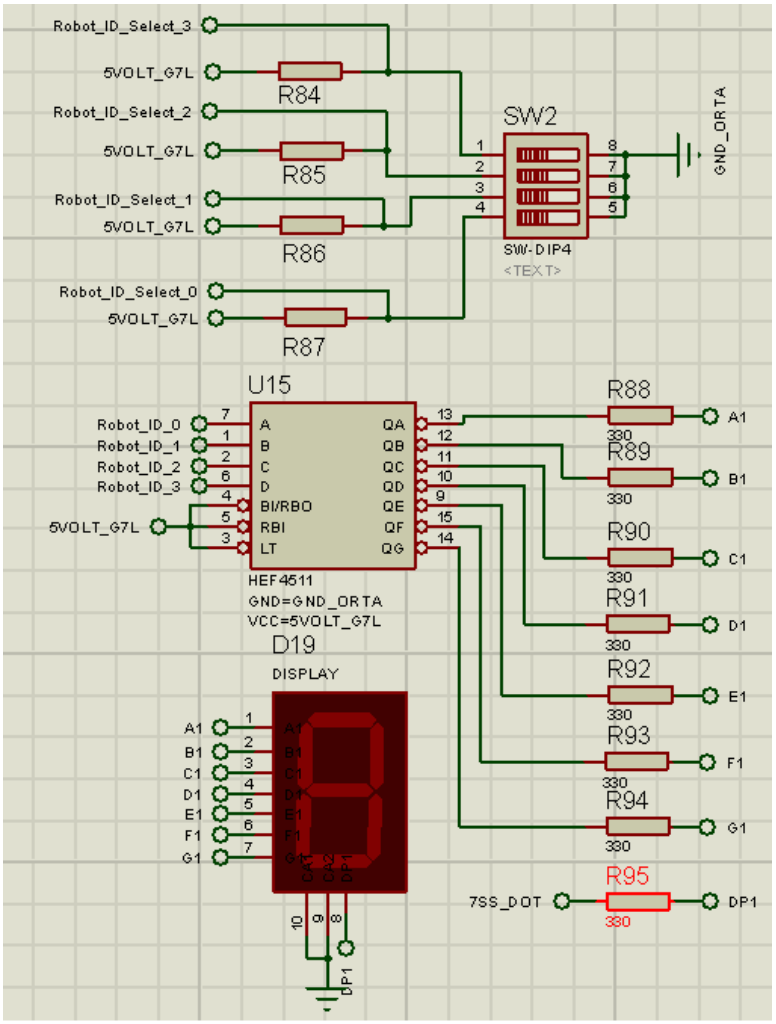

Figure 2.17. Schematics of circuit design related to setting and detecting robot identity.

the BCD-seven segment decoder and later the outputs of the decoder let the display show the robot ID. The decoder may be connected to the switches directly instead of connecting to the microcontroller. However this is not an efficient solution. When this happens, the display only shows the positions of the switches but it does not show the ID which microcontroller understands. The circuit schematics related to IDs is shown in Figure 2.17.

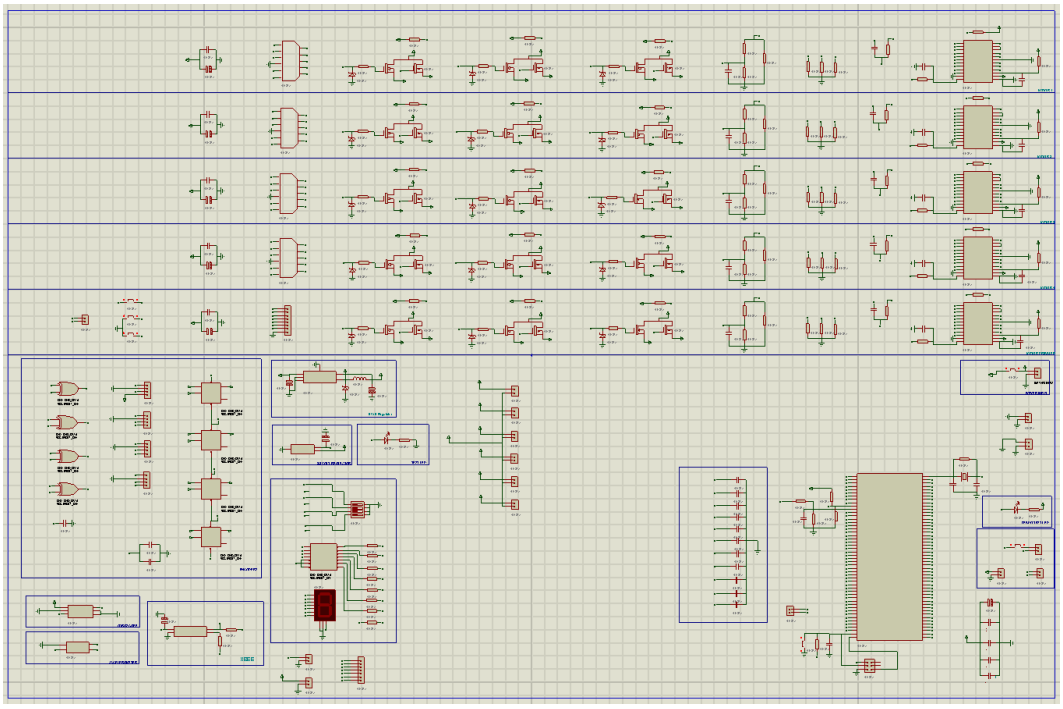

Figure 2.18. Schematics of the circuit.

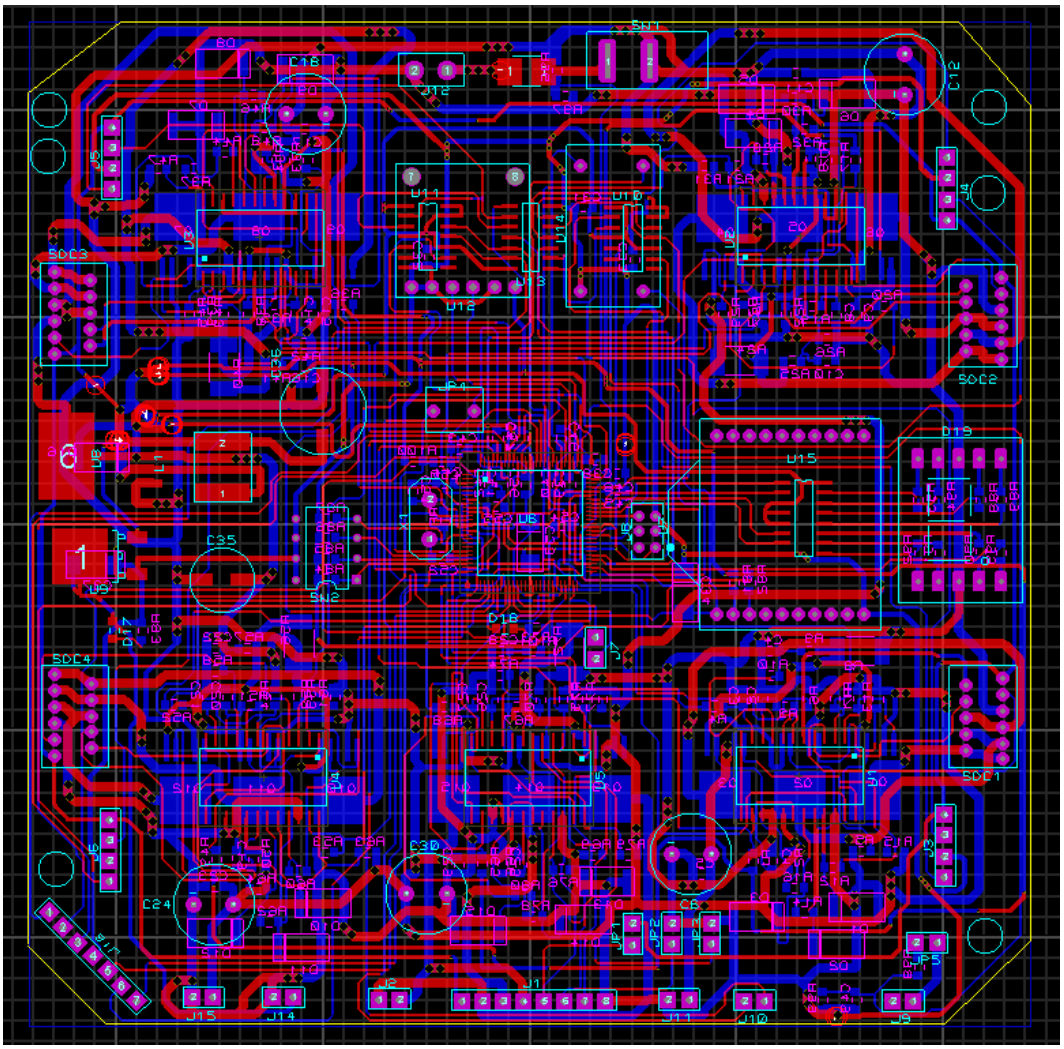

Figure 2.19. PCB design of the circuit.

2.8. Kicker Unit

In our robots, we are using 2 types of kicker mechanism. One of them is for flat kicking, the other one is for chip kicking. Even though they seem to be different from each other, they both use solenoids that have plungers inside. When the shoot is commanded, the voltage at high level is discharged through the desired solenoid. Afterwards, the plunger inside the solenoid moves in the direction of the magnetic area. This is how one robot can kick a ball in general.

In order for the robot to kick, a voltage at 200 V should be discharged through the solenoid. We obtain this voltage level by using another circuit since we do not have any voltage supply unit to power it. As shown in Figure 2.20, this circuit uses a boost (step-up) converter connected to the capacitors with high capacitance. The working principle is similar to the switched-mode voltage regulators discussed in Section 2.5.

In our previous design we used a special IC NCP1054 [21] for boosting the voltage that did not require a PWM signal for boosting and controlling the voltage level. On the other hand, it could charge the capacitors approximately only in 1 minute. Related schematics and PCB designs are depicted in Figures 2.21 and 2.24. The schematics in Figure 2.21 is simplified in order to focus in boosting voltage level. The full version of it is shown in Figure 2.22.

In the new design, IC NCP1054 is not used for boosting the voltage level. Creating a pwm signal and controlling the voltage feedback are accomplished as in Figure 2.20. In this case, the capacitors are charged in 4 seconds. Related schematics and PCB designs are depicted in Figures 2.23 and 2.25.

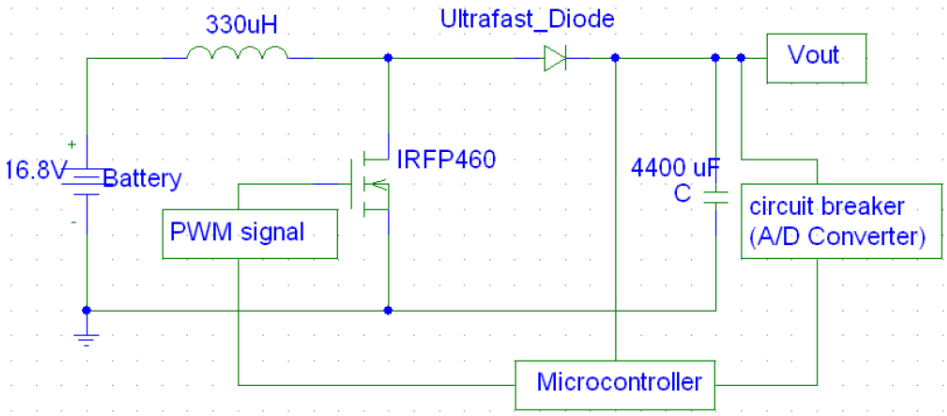

Figure 2.20. General form of the boost converter.

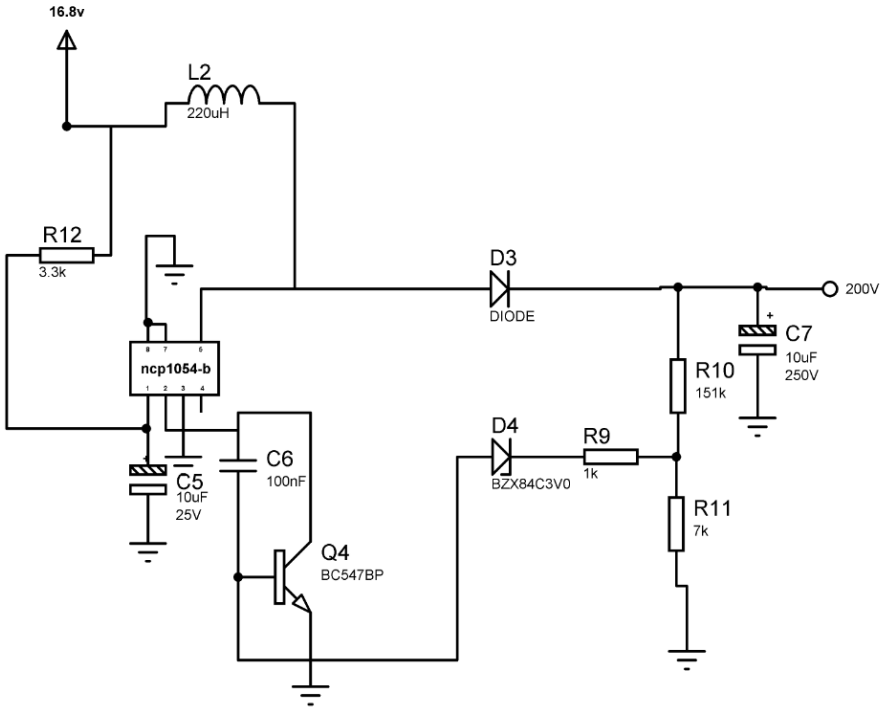

Figure 2.21. Schematics of previous kicker circuit design which is related to boosting part.

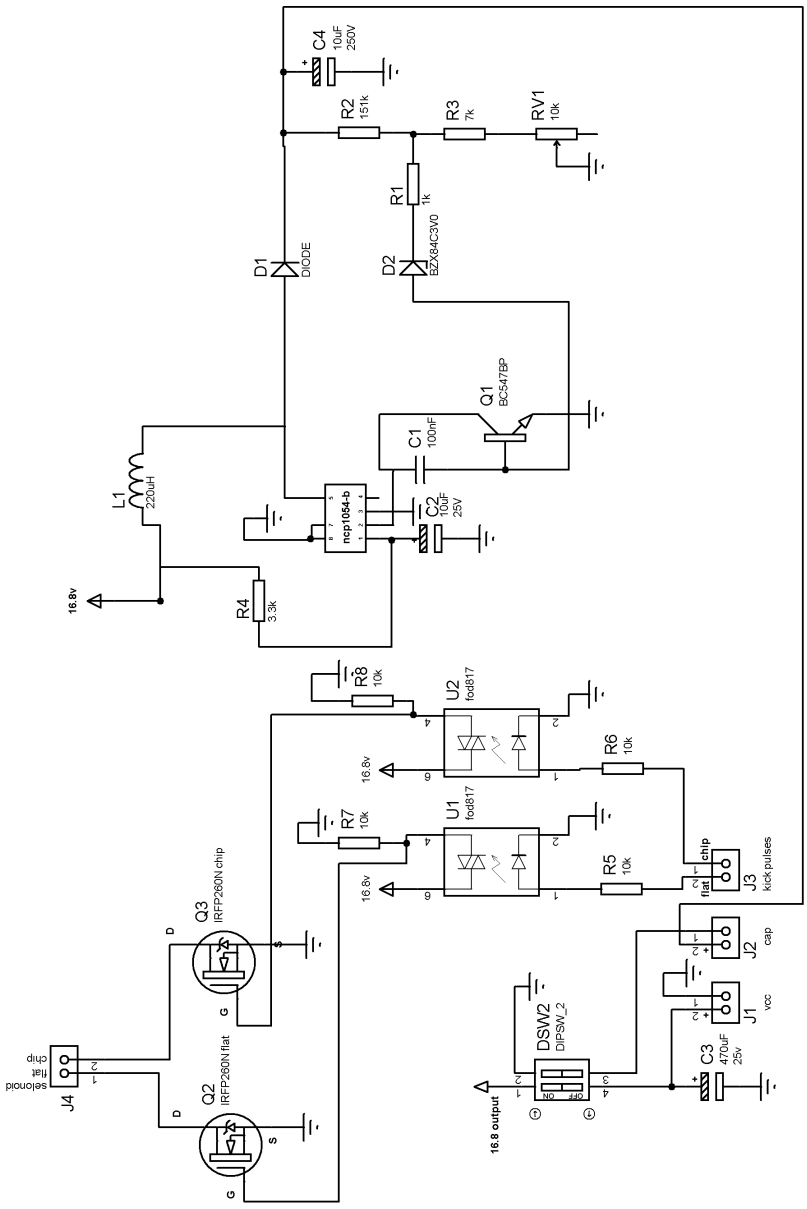

Figure 2.22. Schematics of the previous kicker circuit design.

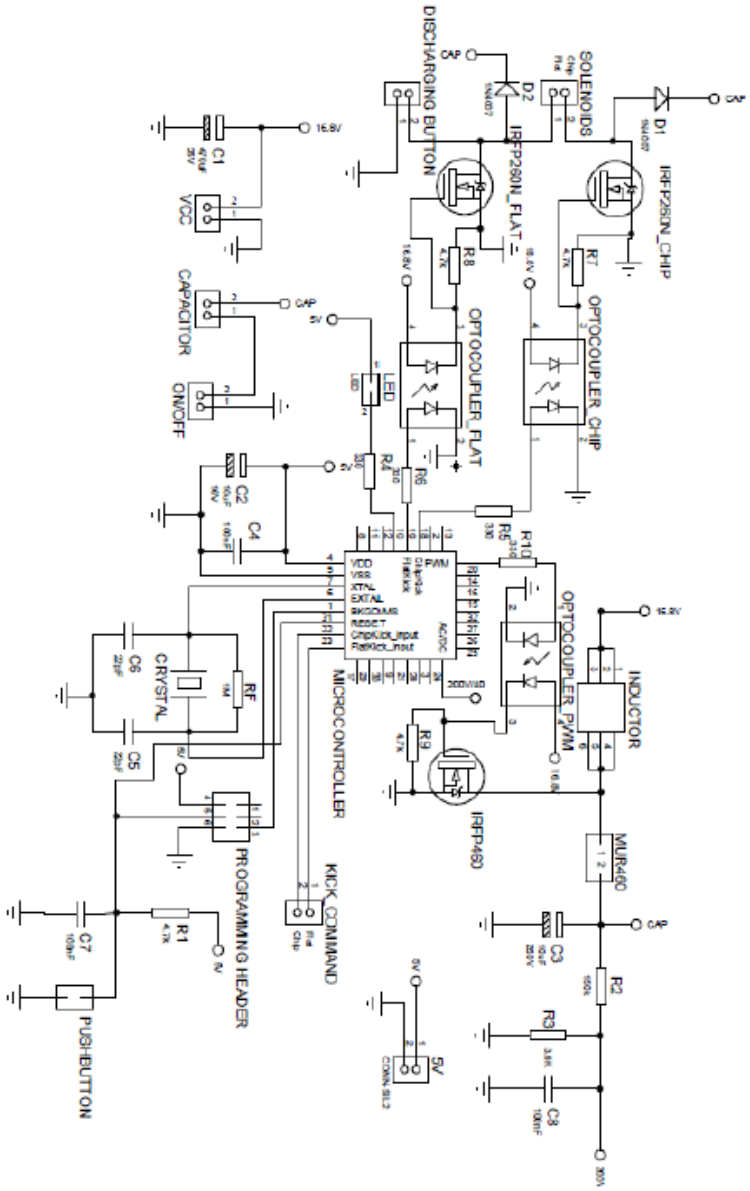

Figure 2.23. Schematics of the new kicker circuit design.

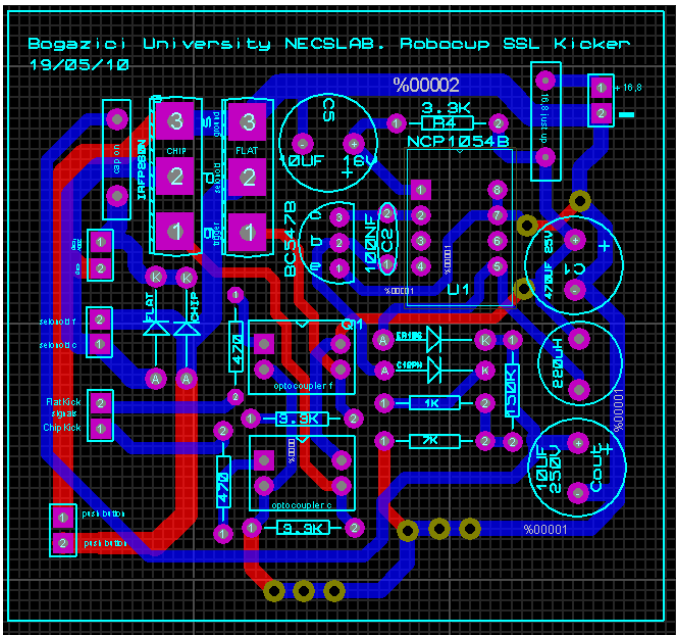

Figure 2.24. PCB design of the previous kicker circuit.

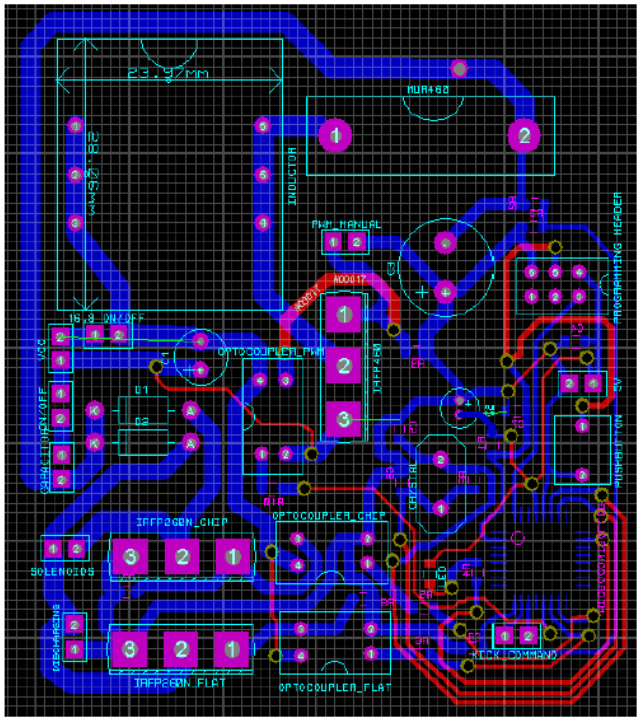

Figure 2.25. PCB design of the new kicker circuit.

2.9. Summary of the Chapter and Concluding Remarks

This chapter covers the hardware details for our new SSL Robots. While 2 MCs were used in the previous design, one MC9S08DZ128 is used in the new one. The transition from Hall sensors to E4P encoders and its effect on higher resolution in control is discussed in detail. Motor driver IC MC3305 is used for eliminating the complexity of driving 3 phases. A switch-mode regulator is implemented in addition to a linear regulator in our new design. Furthermore, the identity of the robot can be set and observed easily without programming the MC from scratch. A new kicker circuit is designed in addition to the main board. While IC NCP1054 was used in the previous design, the boost converter is not controlled by this IC in the new circuit. Thus the charging time has decreased from 1.5 minute to 4 seconds.

3. LOW LEVEL CONTROL

In this chapter, we will discuss our transition from using Hall sensors to encoders with higher resolution. After the method for detecting speed is explained in detail, a new control algorithm is introduced and the differences between the previous one are stated. The bug related to deceleration is noted and a solution to this problem is provided.

3.1. Measuring Speed

In our previous designs, the speed was being measured by using Hall sensor data consisting of 8 pulses per revolution. Although we have increased the number of pulses to 48 with some modifications, this still was not enough for a precise control, and we made a transition from Hall sensors to encoders.

3.1.1. Hall Sensors

Hall sensors are implemented in our motors by default. Each motor consists of 3 Hall sensors. They are located with a certain phase difference inside the motor. Each Hall sensor provides 8 cycles per revolution. Thanks to the phase difference between Hall sensors, each sensor data differ from the others. It does not mean that 2 Hall sensors data will not be used because they are redundant. As each Hall sensor has a phase difference between the other sensors, data created by sensors refer to different motor positions. That is to say, the resolution will be multiplied by 3 by utilizing 3 Hall sensors data. Each signal is in square wave signal format. While using one Hall sensor creates 8 cycles, using three Hall sensors creates 24 cycles. It is not required to observe 3 Hall sensors data separately. If XOR gate is used for combining these signals, it can be easily converted into one signal including 24 cycles (see Figure 3.1). In other words, decoding these 3 different Hall sensors data, 24 positions data per revolution are provided. If negative edge is used for detecting speed in addition to the positive edge, the Hall sensors create 48 position data per revolution. This gives a unique position

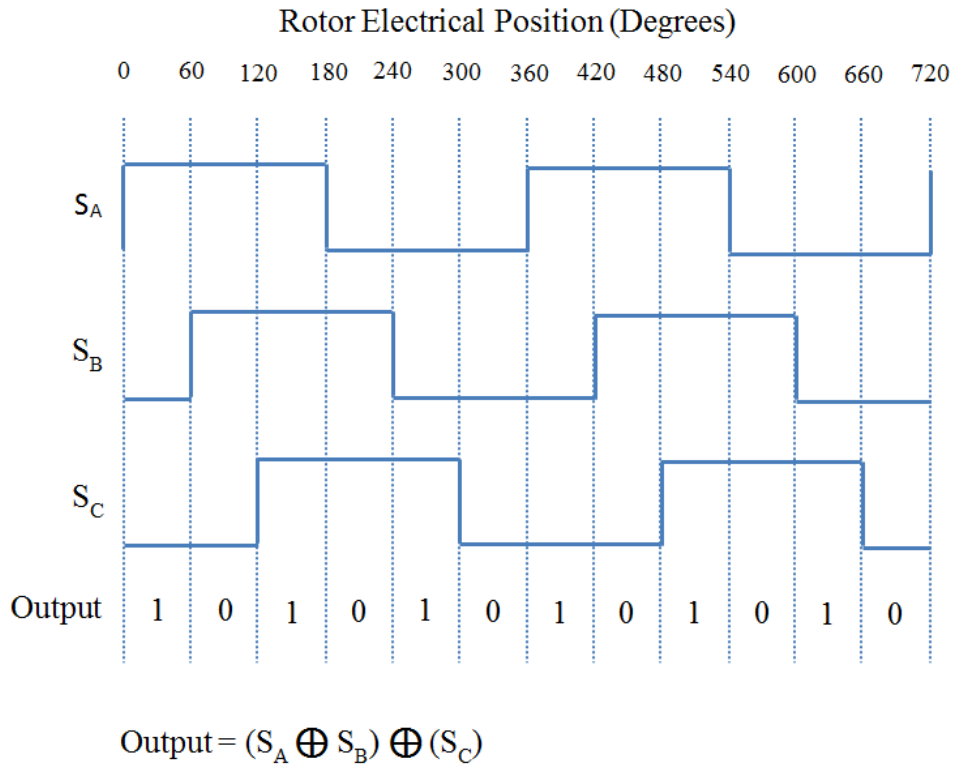

Figure 3.1. Hall Sensors are XORed in order to increase the resolution of pulses.

for every $7,5$ degrees. As the motor does not let the Hall sensors create new data until it moves 7,5 degrees more, an efficient solution could not be found in order to measure the speed of the motor precisely. Note that controlling the motor properly is highly related to the number of acquired pulses.

3.1.2. Encoders

In our new design, encoders with higher resolution replace Hall sensors for detecting the speed value of the wheels. US Digital E4P encoder that provides 360 cycles per revolution is used in the new scheme. Since the encoder has two channels and each channel is shifted by 90 degrees, these channels are decoded by XOR gate (74HC86) as is the case with the Hall sensors. Subsequently 720 cycles are obtained per revolution (see Figure 3.2). If both positive and negative edges are captured, 1440 pulses per revolution are provided, i.e., in every 0.25 degree movement of the motor, the encoder

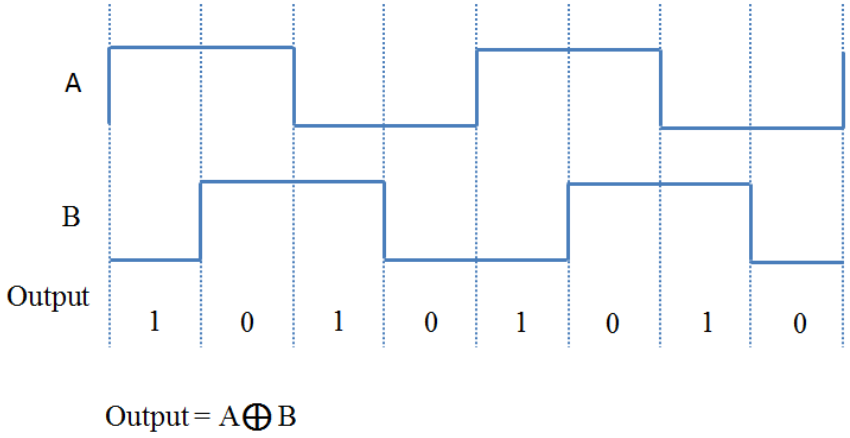

Figure 3.2. Encoder channels are XORed in order to increase the resolution of pulses.

creates new pulse. As a result, encoders E4P provide 30 times higher resolution when compared to the Hall sensors.

3.2. Motor Control

3.2.1. Previous Motor Control Algorithm

In the control of our motors in our previous design, the speed of the motor was being calculated by measuring the duration between two position data. We mentioned in Section 3.1 that 48 position data (pulses) per revolution can be obtained using Hall sensors. Yet, in our first design, only one Hall sensor was being used and only the rising edge of the sensor output was being captured [22]. In order to detect the duration between two position data, we were using our microcontroller's counter function. Our microcontroller has a counter with 16-bits and it counts at each clock cycle of related Timer/PWM (TPM) Module. As soon as the position pulse was captured by the MC, the current counter value was being buffered in an address. Later, when the other Hall sensor pulse was captured, this time the current counter value was being buffered in another address. Since the clock source which drives our TPM Module is set and controlled by us, we know the frequency of the related clock signal. Thus, the difference between two counter values is easily calculated in time domain. After the

duration between two counter values had been found, we were converting this value into the frequency (speed) of the motor. Subsequently, this speed value has being used in our PID algorithm. However, this type of measurement was causing some problems.

First of all, since our speed detection is based on the duration between two position data, the speed was considered as constant until new position pulse arrived. This was leading our algorithm to consider that our motor was moving even when it stopped. When the motor is not turning, the Hall sensors will not create any data and thus new position data will not be included in the control algorithm.

Secondly, since we were using Hall sensor's data, although we were not updating the control algorithm at a constant period, it was not causing serious problems. Our detection of the duration between two pulses was based on the counter values buffered that are directly related to clock source of TPM module. We were using one Hall sensor with this algorithm, and we were receiving 8 pulses per revolution. Since we do not turn the motor more than 3000 rpm, there would not be less than 2.5 ms between two pulses. That is to say, we were not updating our control value less than 2.5 ms. This update rate was reasonable for the MC and PID algorithm. With the new encoders, we receive 1440 pulses per revolution. If we turn the motor at 3000 rpm, the update rate of the control algorithm has to be 13.88 μs. However, the MC will not able to run this algorithm properly since our MC is not fast enough. Additionally our control algorithm cannot response stably at this update rate.

Thirdly, the other related issue was the detection of the speed. For Hall Sensor, our maximum update rate was 2.5 ms as mentioned above. When scaling this value to frequency, even if the counter values are buffered with error 1, it was not changing too much in calculations, because the MC is ticking in micro seconds. Even if we buffer one counter value one clock later or before, this will change nothing. However, for encoders, if we measure the duration between two pulses 13.0 μs instead of 13.88 μs, it will equal 3205 rpm instead of 3000 rpm. This error rate will cause catastrophic problems for controlling the motor.

We are forced to modify the control algorithm because of the above reasons. In the new algorithm, we directly measure the speed of the motor at a constant period. By using this method, we are able to tackle 3 problems mentioned above.

3.2.2. Modified Low-level Control Algorithm

function *MotorControl*(pwmmax,pwmmin,Kp,Kd,desiredspeed,actualspeed)
previouserror $\Leftarrow$ error;
error $\Leftarrow$ $desiredspeed - actualspeed$;
pwm $\Leftarrow$ $pwm + (error \times Kp) + ((error - previouserror) \times Kd)$;
if pwm $>$ pwmmax **then**
  pwm $\Leftarrow$ pwmmax;
else if Pwm $<$ pwmminlevel **then**
  pwm $\Leftarrow$ pwmmin;
else
  pwm $\Leftarrow$ pwm;
end if
MOTORPWM $\Leftarrow$ pwm;
return MOTORPWM;

Figure 3.3. Motor Control Algorithm.

In the previous control algorithm, as soon as new position data were received, the algorithm was updating the control value. The speed of the motor was being extracted from using the counter values buffered at each captured position data. In the modified algorithm, we measure the speed of the motor at a sampling period of 1 ms, and at the same time the control value is updated based on it. At each position data, a counter is running and at the end of 1ms the value is buffered as a scaled version of the speed of the motor. Later this counter is reset and starts running again. At the end of 1ms, the speed of the motor is used in the algorithm depicted in Figure 3.3. In our code, *pwm* changes from 0 to 600. 0 refers to a pulse with %0 width and 600 refers to a pulse with %100 width. If we want to turn the motor at the highest speed, we send PWM signal with %100 width and if we do not want our motor to be powered, we send PWM signal with %0 width. For other speed values, we change this percentage and we set

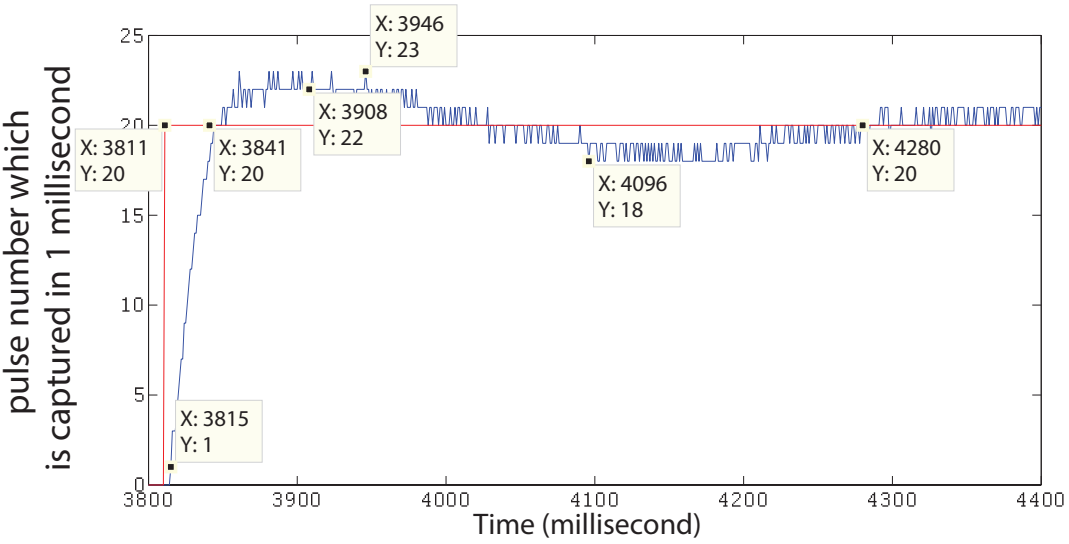

Figure 3.4. Closed Loop Step Response of Our Motor.

our *pwm* variable between 0 and 600 according to the proportion mentioned above. However, we also limit the value of *pwm* by *pwmmax* and *pwmmin*. By *pwmmax* we prevent the motor from being damaged by restricting its speed. By *pwmmin*, we assure that the *pwm* value never drops below the value which is required for the motor to start moving. It is known that the motor is required to be applied more torque for accelerating when it is at hall because the inertia should be overcome. If *pwm* drops below *pwmmin*, from the dropped value to the value *pwmmin*, the motor will not be powered by enough torque in order to be moved. This will effect the control of the motor in adverse manner.

Even though everything seems to be flawless with this modified algorithm after speed detection part of the algorithm has changed, there is one critical issue which has escaped our notice initially. In the algorithm, if the motor needs to be decelerated, the motor sends less pwm value than the last value. However, this is not forcing the motor to make its speed slower, just letting the motor slow down by itself. Even if you stop powering an already accelerated motor, it will still continue to turn for a moment. This problem was causing the motor to settle down at the desired speed seconds later. We have solved this problem by forcing the motor to brake until it reaches down a desired value. Figures 3.5 and 3.6 depict the experimental results that the settling time has

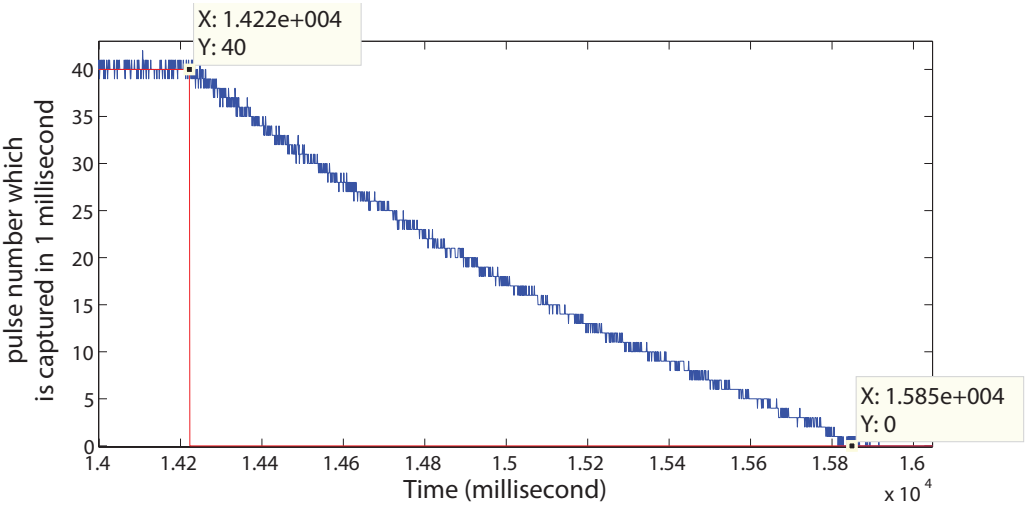

Figure 3.5. The motor settles down in almost 2 seconds without braking.

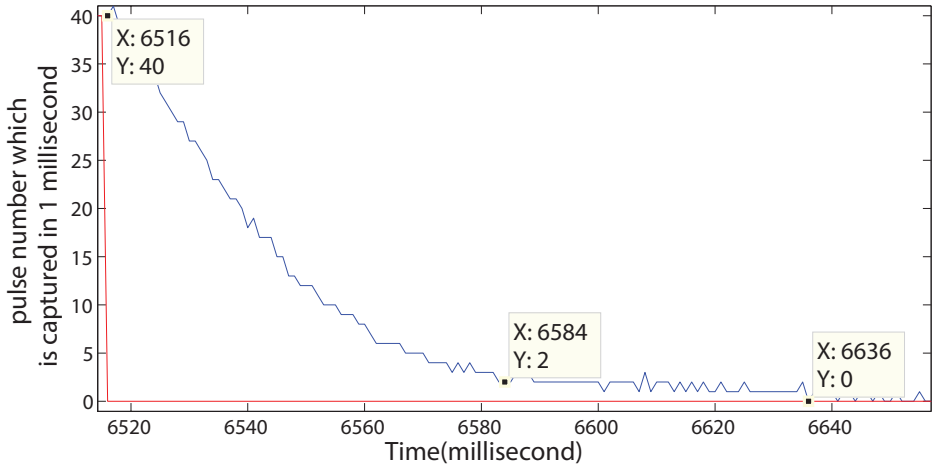

Figure 3.6. The motor settles down in 100 ms with braking.

decreased from seconds to milliseconds by braking the motor when decelerating.

3.3. Summary of the Chapter and Concluding Remarks

In this chapter, the use of Hall sensors and encoders in motor speed control are compared to each other. After discussing potential problems that have been encountered when using the previous control algorithm, a modified low-level control algorithm has been proposed and explained in detail. The problem related to deceleration has been remedied by utilizing the function of braking.

4. PATH PLANNING

Path planning is an important issue in multi-robot navigation in dynamic environments. Although this problem has been studied extensively, there is still not a universally accepted solution. Especially, in complex, fast evolving environments such as RoboCup, path planning is an inevitable part of the platforms. In challenging platforms like RoboCup, there are two main goals of each robot both are related to path planning. The first is to reach the desired point in minimum time whereas the second is to be able to avoid the obstacles while trying to reach that point.

Path planning has to be implemented in our testbed of soccer playing robots. As robots are continuously moving in this environment, path planning turns out to be a complex problem rather than a trivial one. If the environment is not dynamic, and if there is no time limitations for finding a path, solving this problem may be easier. Since computation time and the dynamic environment are at concern, not every algorithm can be used as a solution to path planning in our problem. A new algorithm has been proposed for this kind of environments in which robots should make optimal and quick decisions in a specified time [13]. Rapidly-exploring random trees (RRTs) is the algorithm for tackling the problem of real-time path planning. In RRT implementation randomization is used to explore large states spaces efficiently by RRTs. In this chapter we introduce different versions of RRT algorithms and their implementation on our testbed.

4.1. Related Work

Path planning is an issue on which many studies have been carried out and some of them will be mentioned briefly below. Exploiting randomization is one method for this issue. Randomized potential field algorithm may be declared as the one of the first popular researches on path planning using randomization [23]. Besides, probabilistic roadmap algorithm is another method which has been proposed as randomized approach [24, 25].

Lavalle introduced a new approach for path planning which includes randomized data structure [26]. In their work [26], the path is planned starting from an initial point by extending itself towards random points. The possible paths are not formed by adding each random point to each other directly. After a random point is chosen, the previously founded points are observed and a new point is determined towards the nearest previously founded point. This concept is called Rapidly-exploring Random Tree (RRT).

In [27], an extension has been made to the classical RRT. It is based on bidirectional search algorithms [28]. Not only one tree is grown, but also another tree is grown. While first tree expands itself starting from the initial point, the other one expands from the goal point.

J. Bruce and M. Veloso has introduced an effective extension to RRT which firstly Lavalle proposed. In their work [29], classical RRT algorithm has been improved and implemented in mobile multi robots. While the tree was growing, it was not only based on random points out of the environment, but also on the goal point and waypoint caches. Waypoint caches are the points which are nodes from the previous successful found path. For each iteration, the algorithm needs to decide which point will be used when expanding the tree. This decision is made by a pre-set probability.

4.2. RRT Algorithm

RRT is based on expanding a tree in order to determine a complete path from initial point to the goal point. The initial point of the robot is the root point of the tree. The algorithm firstly builds up the tree by extending the root point. The goal position refers to the point to which the robot is required to reach. Before the path is completed, many points which lead to the goal point have to be created in the tree. These points refer to the nodes of the RRT.

Figure 4.1. Expanding the tree initially towards target point.

4.2.1. Basic RRT Algorithm

function *RRT*(initial,goal,threshold,Δt)
nearest ⇐ initial;
Addnode(tree,initial);
while *Distance*(nearest,goal) < threshold **do**
  target ⇐ *RandomPoint*();
  nearest ⇐ *Nearest*(tree,target);
  possiblenode ⇐ *Findnode*(nearest,target,Δt);
  if possiblenode ∈ CollisionFree **then**
    Addnode(tree,possiblenode);
  end if
  return tree;
end while

Figure 4.2. Basic RRT Algorithm [26].

The initial operation of the basic RRT algorithm is illustrated in Figure 4.1. It is known that expanding the tree starts from the initial node. The extension is made towards a target point. This target point is chosen randomly from the environment in the basic RRT algorithm. A node is found between the initial node and the target point. This node is at a specified distance from the initial node. After this process, the tree consists of 2 nodes as shown in Figure 4.1.

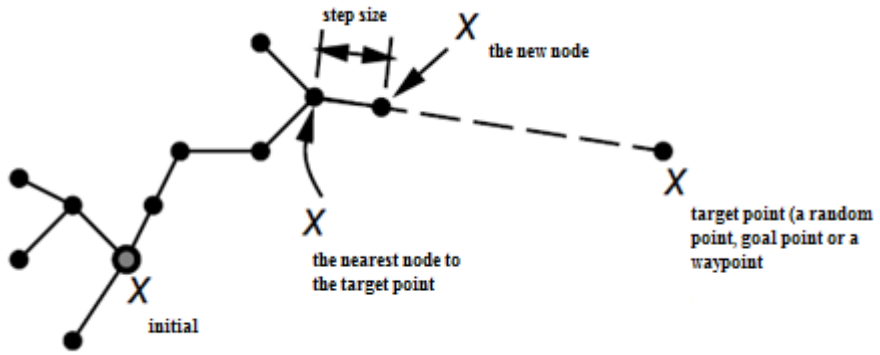

Figure 4.3. Expanding the tree towards target point based on the closest node.

In the next iteration, a random point is chosen again as a target point as shown in Figure 4.3. Then the node which is closest to this target point is discovered. As is the case with extension of the initial node, a point is found between the closest node and the target point. This point is chosen at a specified distance from the closest node as shown in Figure 4.3. This specified distance is set by the user. There is only one important thing which must be taken into consideration when extending the nodes. The new node should not conflict with any obstacle. If a collision occurs, the algorithm repeats itself from scratch until finding a collision free point. Figure 4.2 shows the algorithm of Basic RRT.

Until discovering a node at a desired distance from the goal point, the algorithm continues to extend the nodes towards the target points. When a node is discovered at a desired distance from the goal point, the path is found by following the nodes starting from the goal point to the initial point as depicted in Figure 4.4. Figures 4.5 and 4.6 depicts how the tree looks like when the basic RRT algoritm is used. Observation of these figure will be made in details next section.

Figure 4.4. Extracting the path from the tree.

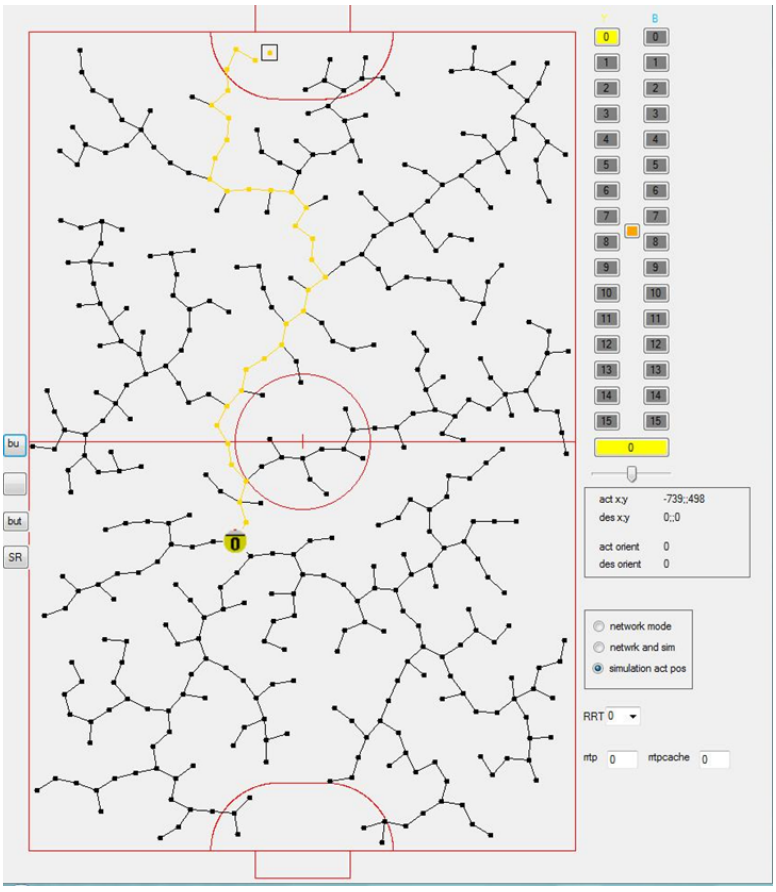

Figure 4.5. Expanded tree without obstacles when a random point is chosen as target point.

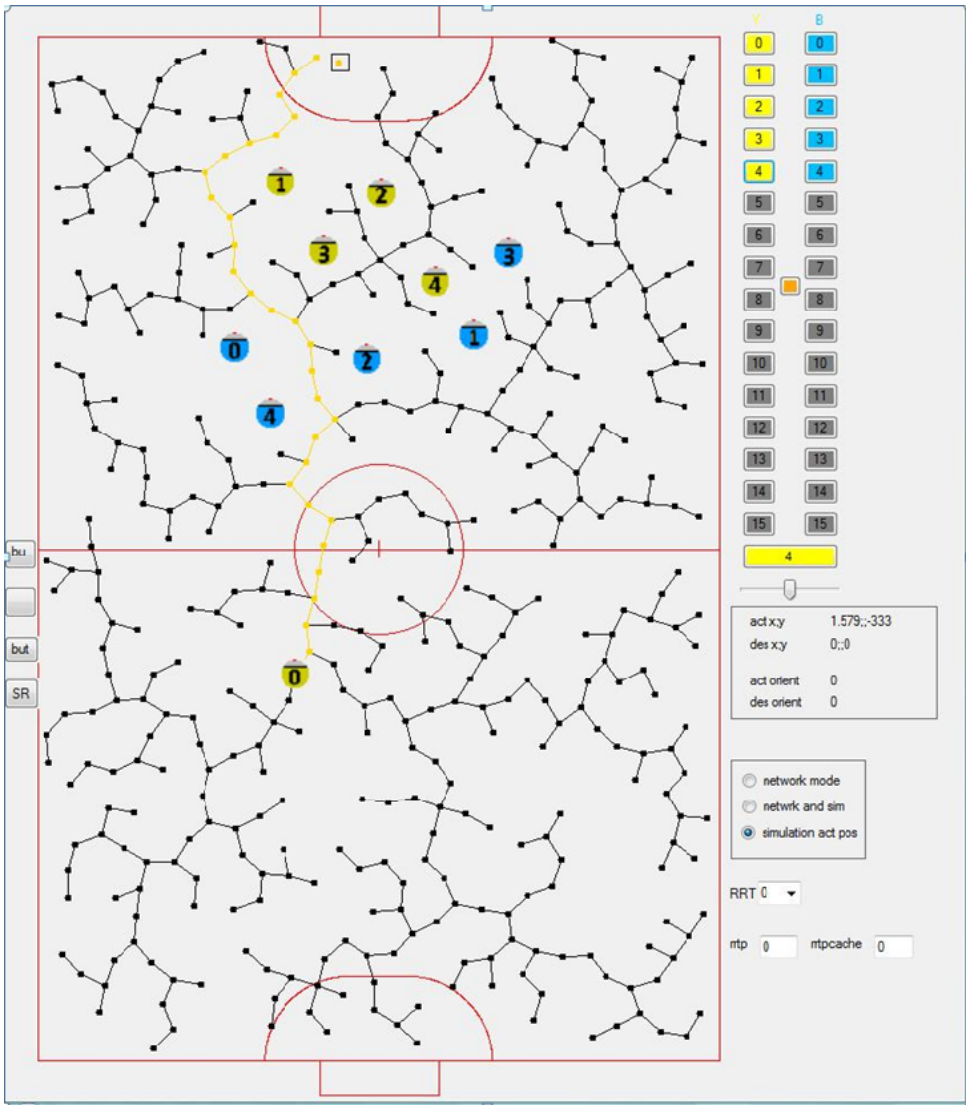

Figure 4.6. Expanded tree with obstacles when a random point is chosen as target point.

While implementing the basic RRT algorithm in robot, there are some main problems that are encountered. Some of these problemles are as follows:

- The path which is found by the algorithm may be nearly optimal but it may not be in a proper structure. A different path which is longer than the previous one may be more efficient for movement of the robot smoothly.
- The algorithm may not be fast enough. In our problem, the cameras above the field have the ability of transmitting the vision data every 16 ms. The robots must make decisions in this duration and take action. In our setting, the main computer runs the artificial intelligence algorithm and transmits the related data to the robots. Decisions of the robots actually refer to the decision which is made by the main computer. The most important decision is the direction the robots should move. As new vision data are transmitted to the main computer every 16 ms, if the decision is not made in 16 ms, the new vision data of the field will be transmitted to the main computer. As the previous data have not been processed yet, the algorithm will still be running for the previous vision frame. Thus this will not be a real time solution.
- The algorithm may be fast and it may find a complete path to the goal sooner or later, but in real time problems there may not be enough time to expand the complete path to the goal.
- Each time, the algorithm starts to run from scratch which leads to inefficient solutions. The new and previous found paths may not be related.

4.2.2. Modified RRT Algorithm

In the basic RRT algorithm, firstly a random point is chosen as a target point. Then the closest node to this target point is determined. A new node is discovered from the closest node towards target point at a specified distance. This algorithm has the potential to generate a complete path to the goal sooner or later. However, using only the random points as the target points creates serious timing issues. Figures 4.5 and 4.6 illustrate the potential problem that finding an available path may require creating

```
function RRT(initial,goal,threshold,Δt)
nearest ⇐ initial;
Addnode(tree,initial);
while Distance(nearest,goal) < threshold do
  target ⇐ DetermineTarget();
  nearest ⇐ Nearest(tree,target);
  possiblenode ⇐ Findnode(nearest,target,Δt);
  if possiblenode ∈ CollisionFree then
    Addnode(tree,possiblenode);
  end if
  return tree;
end while
function DetermineTarget(goal)
p ⇐ UniformRandom [0.0 .. 1.0];
if p < PresetProbGoal then
  return goal;
else
  return RandomPoint();
end if
```

Figure 4.7. Modified RRT Algorithm [29].

many nodes. Thus it will take a lot of time to find an available path. While Figure 4.6 shows the expansion of the tree when the obstacles are available on the field, Figure 4.5 depicts the expansion of the tree on the field without obstacles. In order to speed the algorithm up, a method which is described below has been developed.

While discovering a new node, two points are used as target points. In the above algorithm only random points have been used as target points. Yet in this modified approach, the goal point is used as the target point in addition to the random points. The algorithm creates new nodes by not only extending the tree towards random points, but also extending towards the goal point. At this point the algorithm must make a decision. Will a random point be used as a target point or will the goal point be used as a target point? This decision is made according to the probability which is set by the user as default. While the extension towards a random point has a probability of p, the extension towards the goal point has a probability of $1 - p$. Let us suppose that the algorithm makes a decision that the goal point is to be used as a target point. The scenario will be the same as is the case with using a random point. The closest node to the target point is found and the extension is made by discovering a new node between the closest node and the target point (goal point). The algorithm selects the target point from two alternatives at each iteration: a random point or the goal point. For a collision free environment, Figures 4.8 and 4.9 illustrate how the tree will look like when the goal point is used alone as target points and how the tree will look like when the goal point and a random point are used together as target points, respectively. Besides, when Figures 4.6 and 4.10 are compared, the number of created nodes have been dramatically decreased when goal point has been used as target point as well.

It is known that the goal point cannot be used alone as a target point. When environment has obstacles, an available path cannot be found from initial point towards the goal point. Both the goal point and a random point should be used together as target points (see Figure 4.10). Figure 4.7 shows the algorithm of Modified RRT.

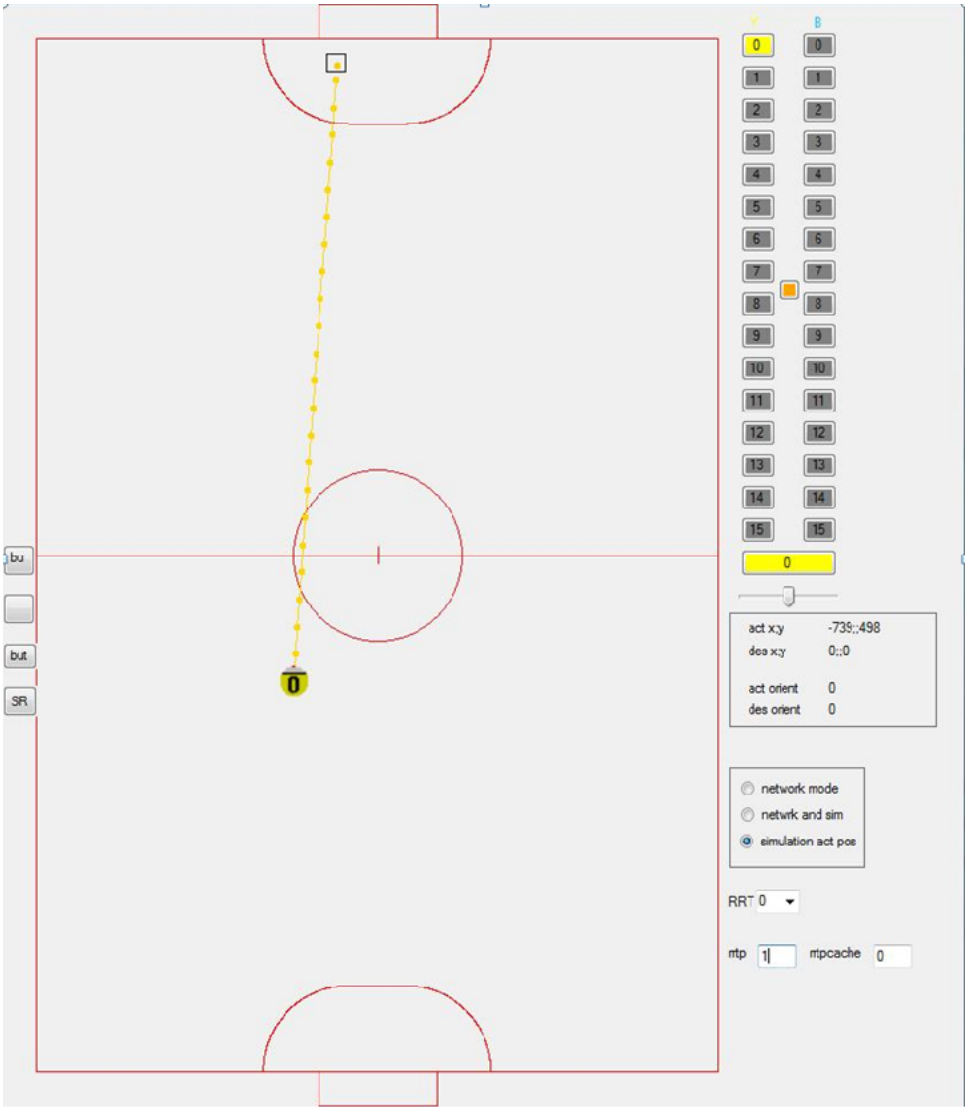

Figure 4.8. Expanded tree without obstacles when goal point is chosen as target point.

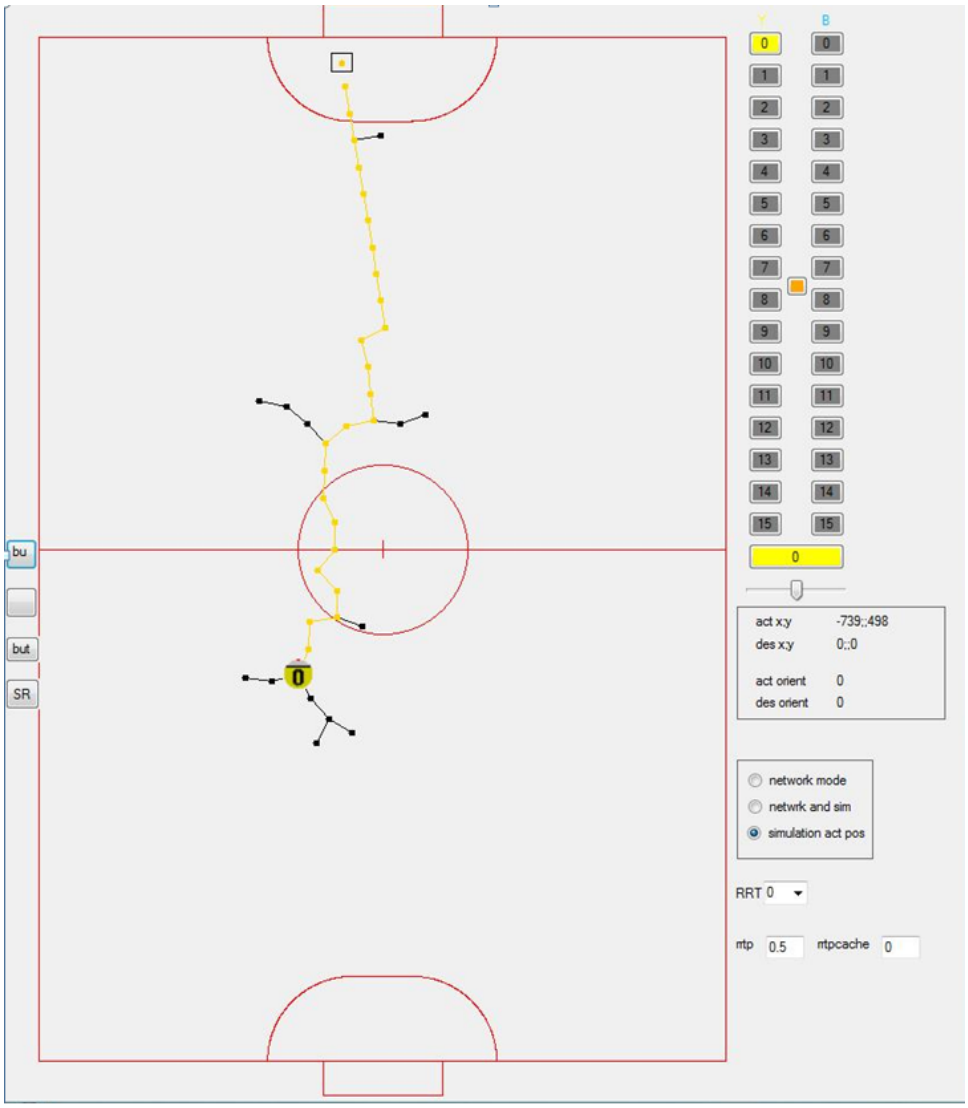

Figure 4.9. Expanded tree without obstacles when a random point and goal point are chosen as target point with equal probabilities.

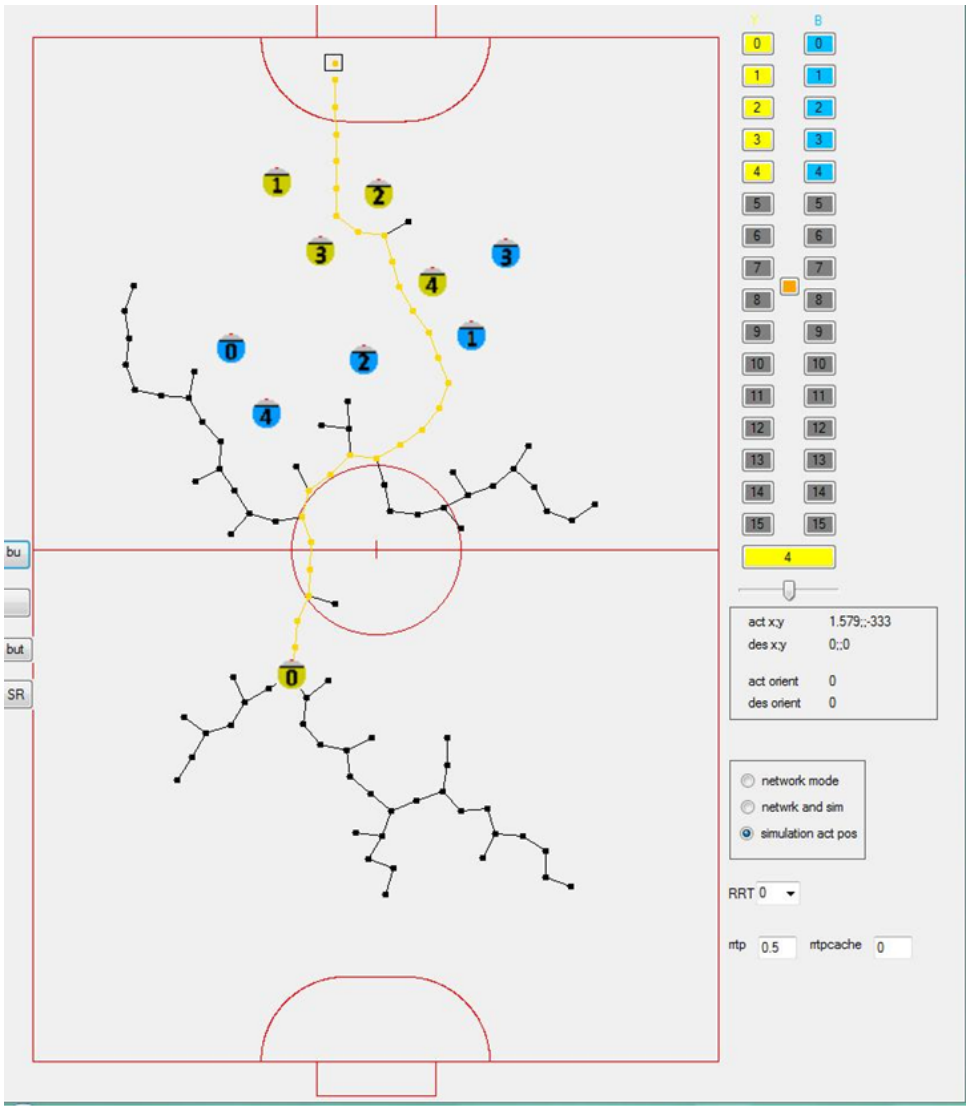

Figure 4.10. Expanded tree with obstacles when a random point and goal point are chosen as target point with equal probabilities.

4.2.3. Extended RRT Algorithm

```
function DetermineTarget(goal)
p ⇐ UniformRandom [0.0 .. 1.0];
i ⇐ UniformRandom [0 .. NumWaycachePoints-1];
if  p < PresetProbGoal  then
  return goal;
else if  PresetProbGoal < p < PresetProbWaypoint  then
  return Waypointcache[i];
else
  return RandomPoint();
end if
```

Figure 4.11. Extended RRT Algorithm chooses target point out of goal point, random point and waypoint caches [29].

Even though picking out the target point from two different points is more efficient than the basic RRT algorithm as illustrated in the previous section, it cannot be a solution to an important issue. It is known that the vision update rate is 16 ms and every vision frame is different from each other as the environment has a dynamic feature. Hence, it does not make sense to rely on a previously found path in order to avoid obstacles until reaching the goal point. This is why the algorithm has to be executed at each new frame in order to find a new path according to the new environment. This constraint can cause new problems such as creating irrelevant paths and time issues. As a new path is to be determined at each frame, the currently and the previously paths found may not be related to each other. This highly plausible scenario may lead the robot to move in an unstable fashion as seen in Figures 4.12 and 4.13. There is a method named Extended RRT (ERRT) for tackling such problems.

Although the environment is highly dynamic, it may be assumed to be static to some extent in the time duration of consecutive frames. As the environment does not change completely, the previously found paths can be useful in finding a new path. This does not mean that the new path will be found only by using only previously found paths; however it does not mean either that the previously found paths cannot

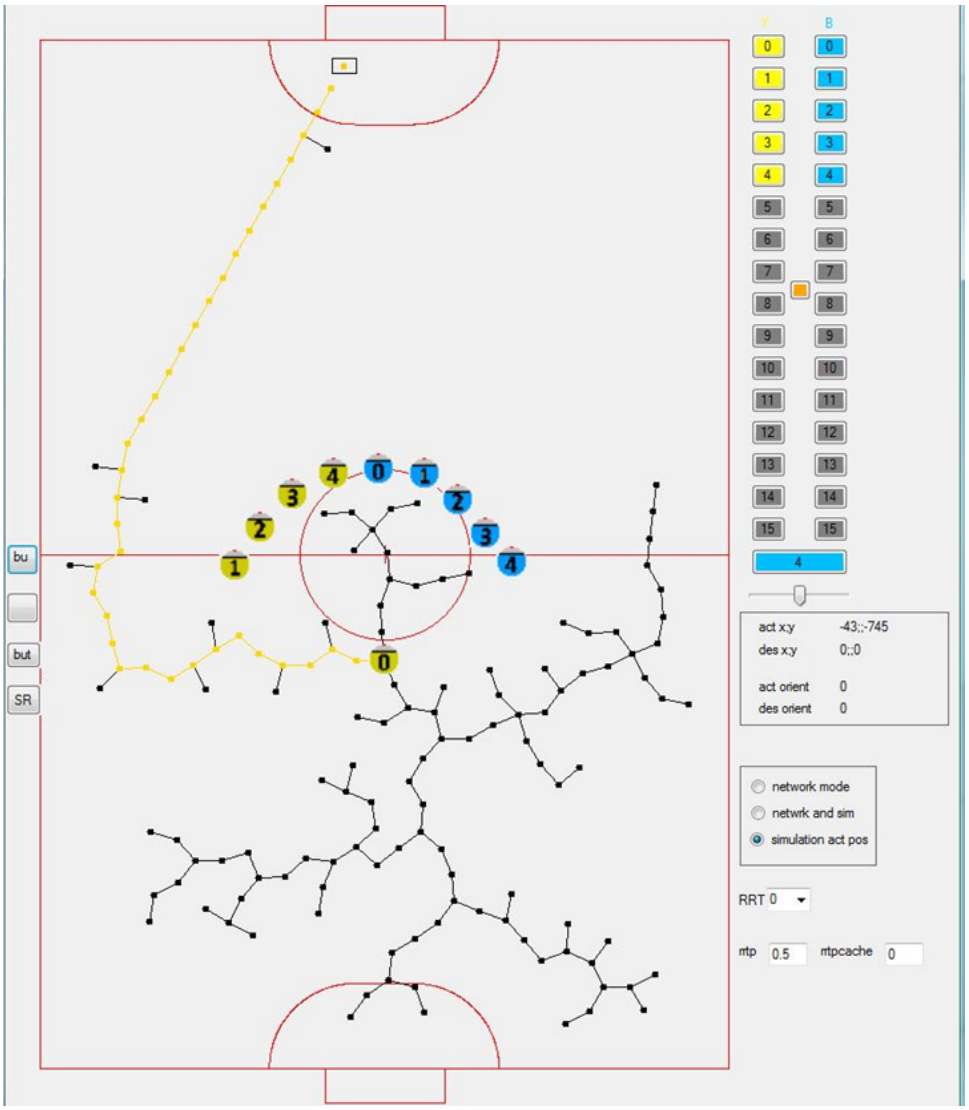

Figure 4.12. Expanded tree and found path at iteration k when a random point and goal point are chosen as target point with equal probabilities.

be a part of the new path at all. We mentioned that while creating new nodes in RRT, two different points are utilized: a random point and the goal point. In ERRT, there is an extension to this approach. A new point is used as a target point and this point will be selected from the waypoint caches which refer to the nodes in the previous successful path. Figure 4.11 depicts the extension in ERRT. As the environment is expected to change slightly, the previous created nodes may be used so that the new found paths will be more stable and will be more related to the previous paths. The target point is picked out from these 3 points according to the probabilities which have been set initially by user (see Figure 4.14). While the path is being planned, the target point is

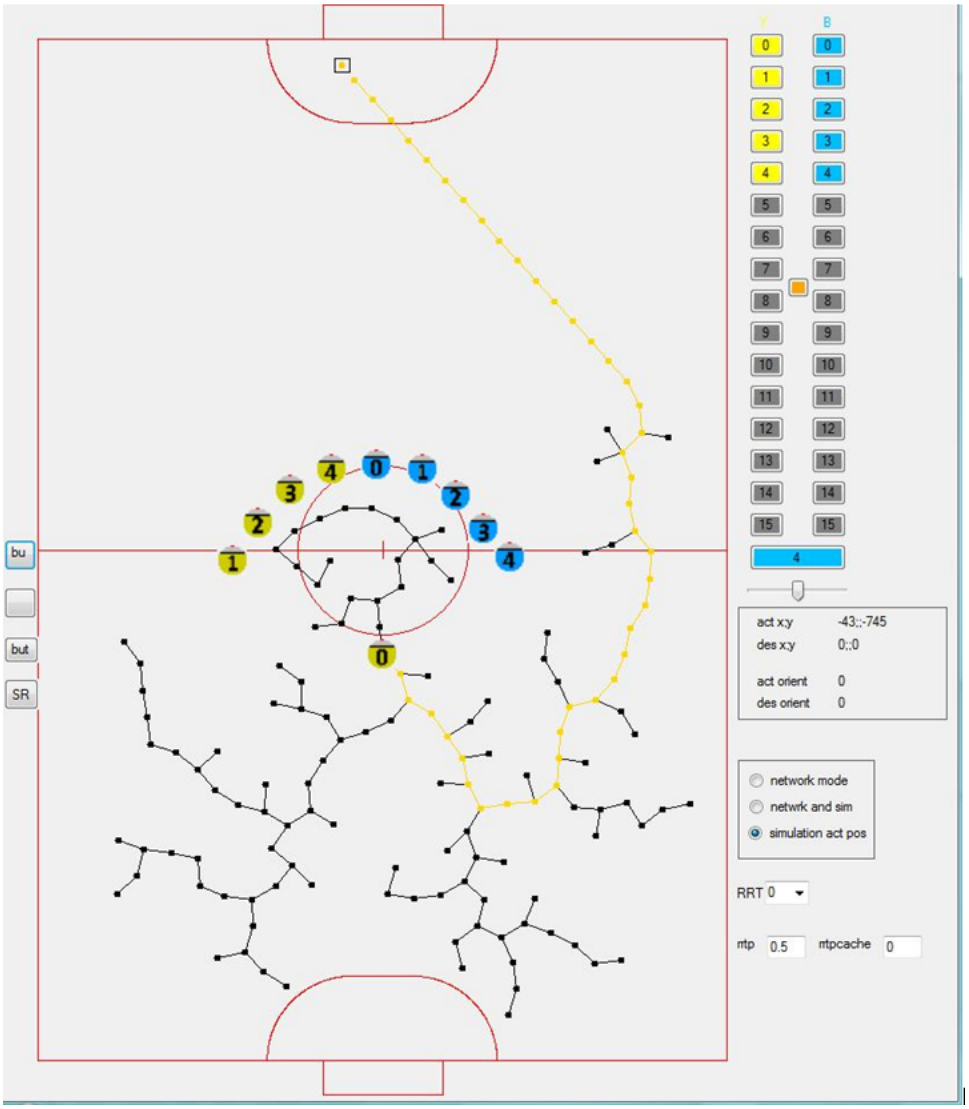

Figure 4.13. Expanded tree and found path at iteration $k+1$ when a random point and goal point are chosen as target point with equal probabilities.

chosen based on these probabilities at each iteration. The nodes which belong to the previous successful path are buffered randomly into an array with predefined size. If the new target point is chosen as waypoint caches, this target point is selected from this array. The new node is created as usual by using this target point.

When cached waypoints are taken into consideration, paths that are found consecutively are similar to each other. Moreover, the number of nodes which create the tree will be decreased. Thus computational time is minimized. This may be observed in Figures 4.15 and 4.16. Goal point, a random point and waypoint are used as target

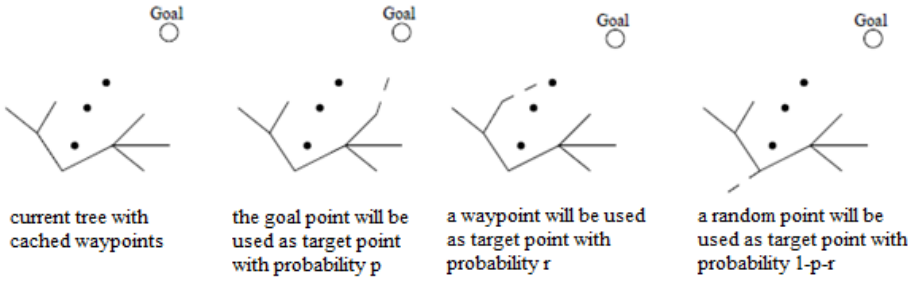

Figure 4.14. Different scenarios for creating nodes.

points with probability 0.2, 0.1, 0.7 respectively in these figures. For a better understanding, we should observe the results of two different RRT approaches. Firstly we can examine the results of the RRT with using 2 different points as target points. While Figure 4.12 depicts the expansion of the tree at iteration k, Figure 4.13 shows at iteration $k+1$. Although these figures belong to 2 consecutive iteration, they are not related to each other. However, using waypoint caches as target point as well, the paths found consecutively at iteration k and $k+1$ are more related to each other. Figures 4.15 and 4.16 illustrate this. In addition to this property, using waypoint caches has decreased the number of nodes which are required to form the tree. Thus finding an available path from the initial point to the goal point costs less computational time.

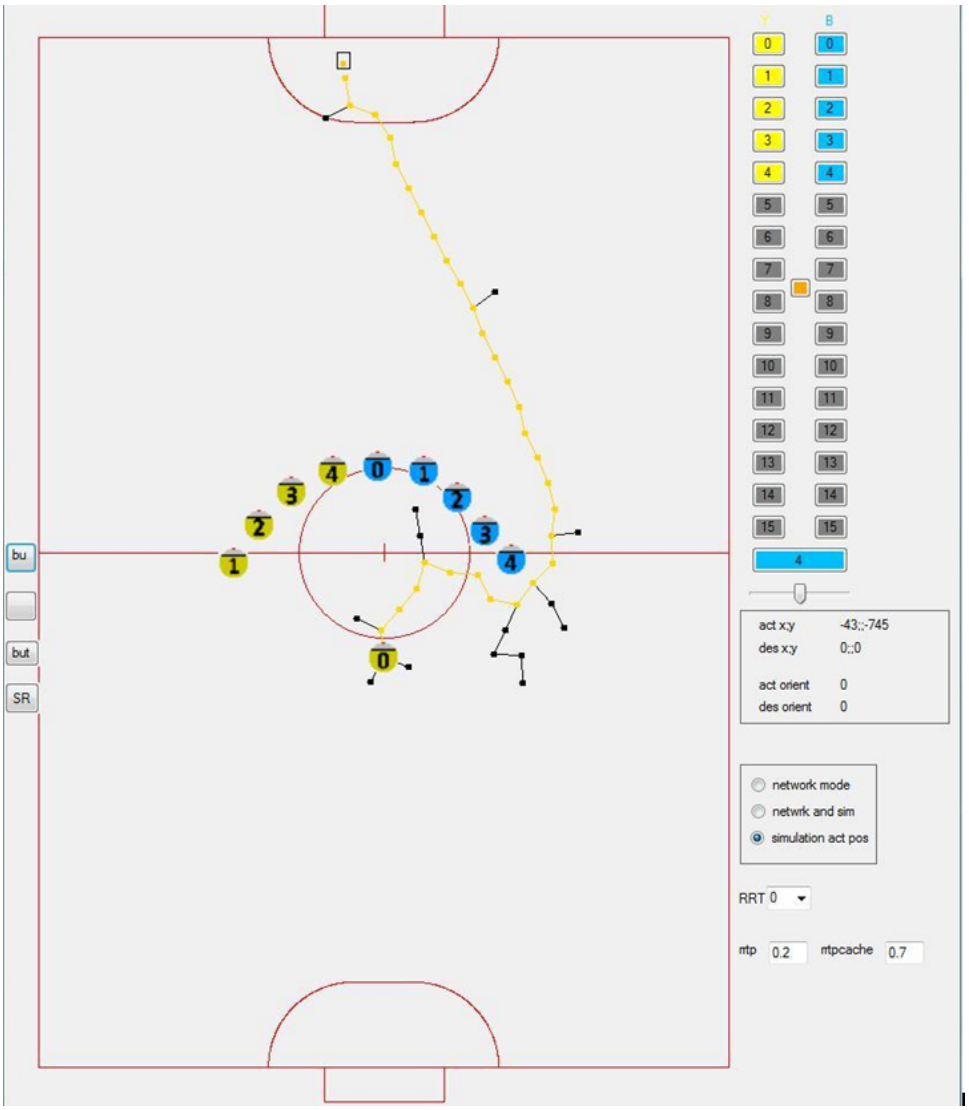

Figure 4.15. Expanded tree and found path at iteration k when goal point, a random point and cached waypoint are chosen as target point with probability 0.2, 0.1, 0.7 respectively.

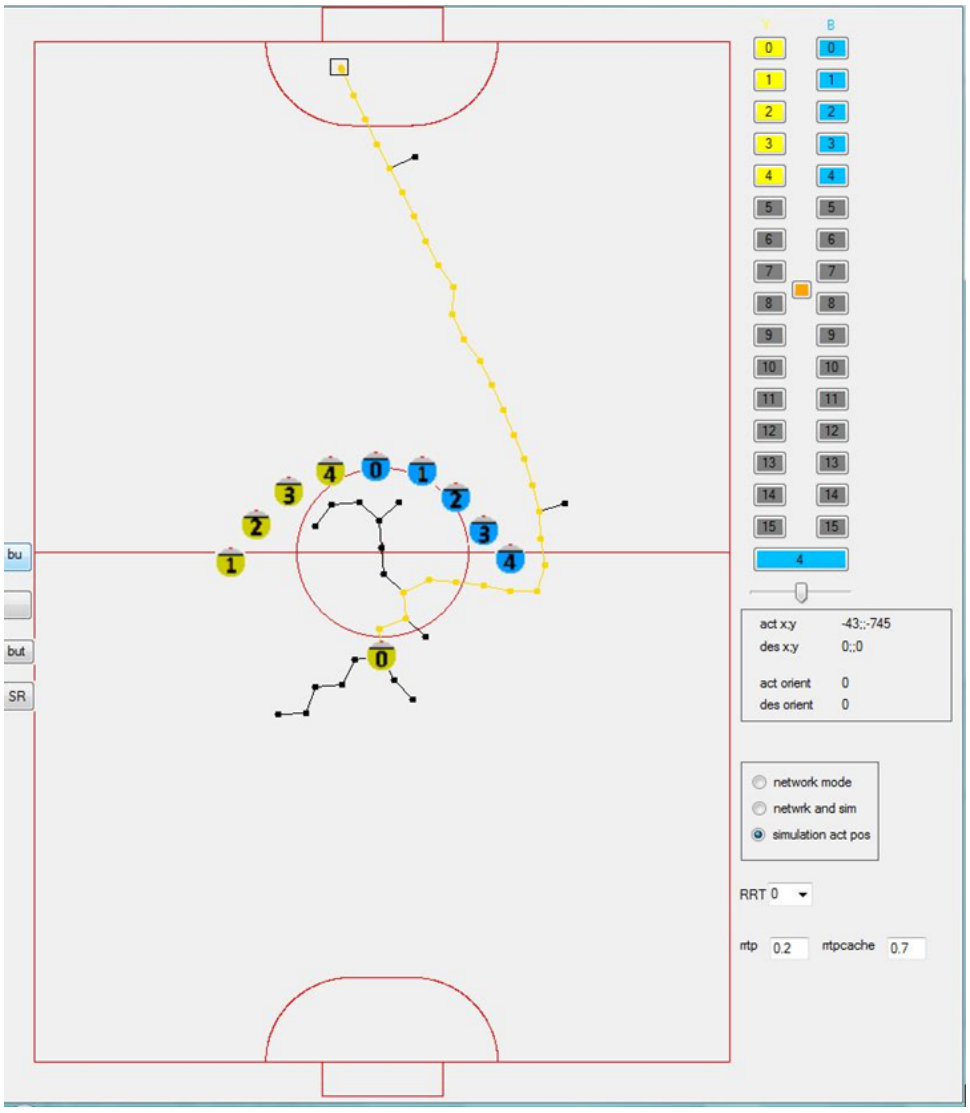

Figure 4.16. Expanded tree and found path at iteration $k+1$ when goal point, a random point and cached waypoint are chosen as target point with probability 0.2, 0.1, 0.7 respectively.

4.3. Summary of the Chapter and Concluding Remarks

In this chapter we discussed path planning and obstacle avoidance issue. Different versions of the RRT algorithms are observed. Each one is tested on our system and results are depicted in our simulation program. Subsequently, the results are compared with each others and which algorithm fits in our system best is discovered. ERRT is adapted in our system with respect to these results.

5. FORMATION CONTROL

In this chapter, we discuss the details of higher level strategy planning software. After a brief review of literature on formation control, game and player states which are required for playing an SSL match properly will be described. The relation between formation control and these states will be explained in detail. New approaches will be adopted to formation and localization issues in accordance with SSL matches.

5.1. Literature Review on Formation Control

As far as Multi-Robot Systems (MRS) are concerned, formation control is an important task in building a properly functioning system. In many applications, it is required for the agents to track a predefined trajectory while keeping a formation with a certain shape. In military missions, a group of autonomous vehicles need to be located in such a formation that the optimum area can be covered. In satellite formation, reducing fuel consumption and increasing sensing capabilities are satisfied [30]. This kind of examples can be given in many areas for formation control. In our design, the robots are required to maintain specific formations in different conditions. The researches in the area of formation control with MRS, first classification is made in terms of the number of the robots. The system may be composed of limited number of robots or more than a certain number of robots. The group with more than a certain number of robots refers to swarm control. Our system is involved in the other group. The other classification is made according to the structures of the robots. The structures may differ in terms of hardware and software. If the robots are not identical, they are called heterogenous. If the robots are identical to each other, they are called homogenous. In our system, all robots are identical to each other. The type of controllers is another classification category for researchers. The robots can be controlled in a centralized or decentralized way. In centralized control, the robots are controlled from one point which has the whole information related to the environment. All decisions are made by that point. In decentralized control, each robot has its own controller and makes decisions by itself. In our system, even though the decisions are

made in a decentralized way to some extent, the critical and important decisions are made in a centralized way.

As for the approaches for formation control, there are mainly three approaches: leader follower strategy [31], virtual structure approach [32], and behaviour-based method [33].

In behaviour based approaches, behaviours are designed and assigned to each robot. Behaviors are processes that reach or sustain goals, i.e., avoiding obstacles, tracking a trajectory. In [34], the formation behaviours are combined with planning path behaviours. The robots are aimed to reach some goal points, avoid obstacles and concurrently keep their formation with a certain shape.

In virtual structures, the formation is considered as a whole instead of particles. Lewis and Tan [35] proposed a virtual structure approach using formation feedback. In their method [35], the virtual structure is adapted in some predefined direction, and the locations of the robots are updated in each iteration.

In the leader following strategy, while one or more robots lead the formation, the other ones move based on the leader. The authors of [36] set an example for this approach. Formations are controlled by using information that is obtained from sensors locally. Feedback linearization is utilized to keep the distance and orientation of the follower in steady.

5.2. General Overview of SSL Game States

In order to play an SSL game properly, the game must be divided into states as the robots cannot play same in each part of the game. Mostly each command from RefereeBox requires a new game state. For instance, the state of A penalty kick differs from the state of kickoff in terms of the rules. While penalty kick requires the robots to be located in certain positions obeying certain rules, a kickoff state requires the robots to be located in different positions obeying different rules. Naturally each game

state requires different player states. Some game states and player states are described below.

- Game States
  - Normal Start: this state is active after "normal start" command is transmitted to each team. This command follows certain commands such as "kickoff" and "penalty kick".
  - Kickoff: after a team scores, "kick off" command is sent and transition to this state occurs.
  - Penalty kick: when "penalty kick" command is sent, this state is active.
  - Indirect kick for opponent: when a throw in or goal kick has occurred, "indirect kick" command is sent and triggers this game state.
  - Force start: when a decision is not for the start of one team, "force start" command is sent and the game starts to be played quickly by each team.
  - Active game: this state refers to the state in which the game can be played by each robot actively.
- Player States
  - Goal Keeper: "goal keeper" protects the goal.
  - Defender Left: when the game state requires 3 defenders, "defender left" protects the left part of the 3 defenders when they are positioned around the defense area as a circle.
  - Defender Middle: "defender middle" protects the middle of the circle created by the defenders.
  - Xcm away Middle: some game states require the robots to be located at least X *cm* away from the ball, and "Xcm away middle" is located in the middle of the ball blockers. Usually 3 robots block the ball. They create a circle so that they are all closest to the ball.
  - Xcm away Right: X *cm* away right blocks the right part of the circle which is created by the ball blockers.
  - Ball Follower: ball follower state is assigned to the robot which is the closest to the ball. It has shooting and passing capabilities.

– Ball Waiter: ball waiter is located in a position where a dangerous attack can be created in the case of obtaining the ball
– Dangerous Robot Blocker: this state is assigned to a player in order to block an opponent player and prevent a likely attack.

In order to make the robots obey the rules and make them play according to the assigned states, we have developed several algorithms which are geometry based. These algorithms will be discussed next.

5.3. Blocking the Ball

Depending on the game states, a player state is assigned to each player. In the "ball blocking" state, the main task is to protect the goal besides defenders and goal keeper. As shown in Figure 5.1, the ball blocking robot is located on the point D between the ball B and the point G which is to be protected. From Figure 5.2 the equations are extracted and the point D is found as follows:

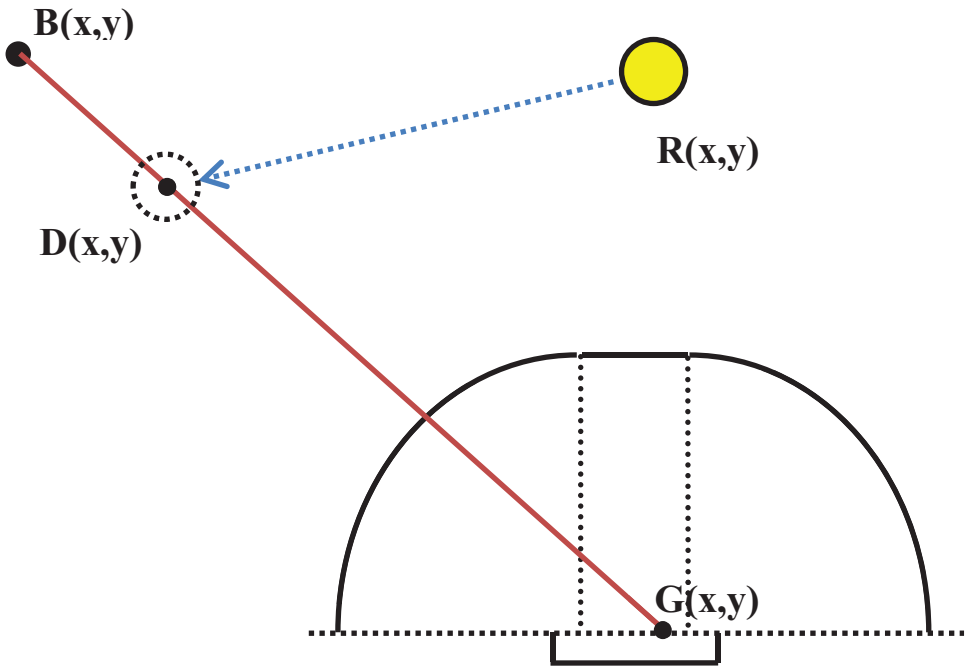

Figure 5.1. Ball Blocking with One Robot.

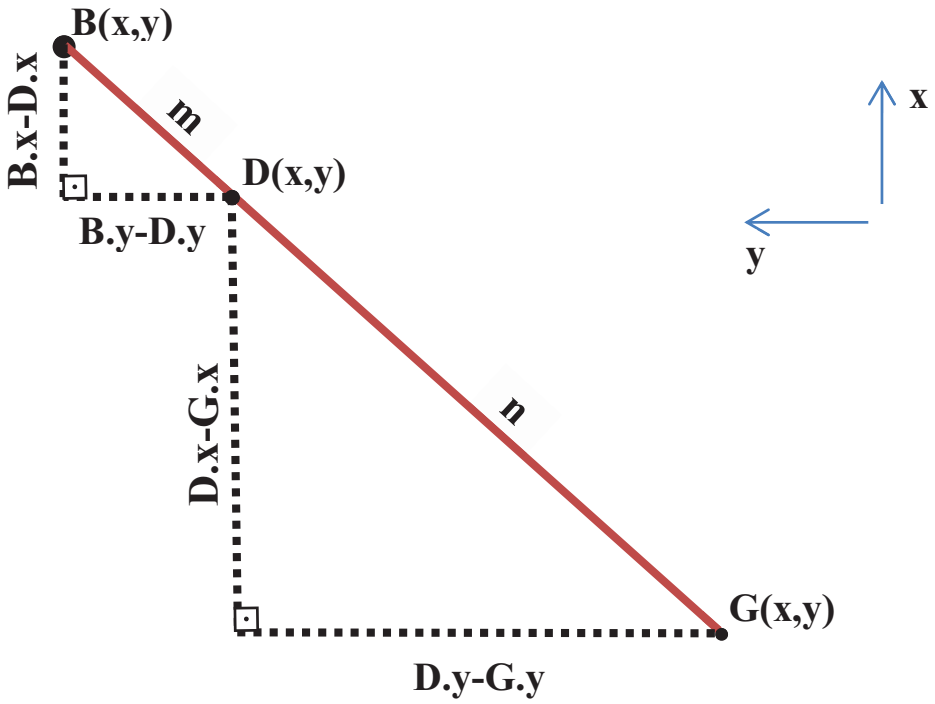

Figure 5.2. How to Block the Ball with One Robot.

$$n = |BG| - m \tag{5.1}$$

$$\frac{m}{n} = \frac{B.y - D.y}{D.y - G.y} \tag{5.2}$$

$$\frac{m}{n} = \frac{B.x - D.x}{D.x - G.x} \tag{5.3}$$

In some cases ball blocking with one robot is not sufficient in order to make good protection of the goal. For this reason, mostly 3 ball blocking states are assigned to the robots. Figure 5.3 depicts this scenario. In order to cover the most area, the robots are located in circle form, each at the same distance from the ball. This means that a circle which the ball B located on the center is drawn and the robots are positioned on

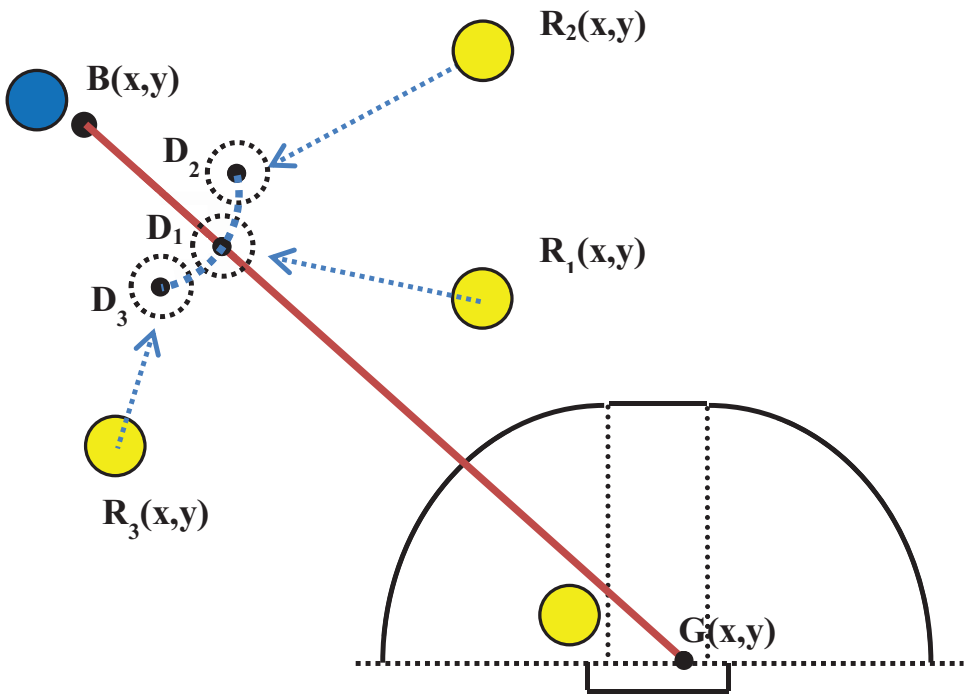

Figure 5.3. Ball Blocking with Three Robots.

this circle. This method is very useful because in many game states this localization is used. The main reason is that the players cannot come close to the ball less than 50 *cm* according to the rules for some game states.

In Figure 5.5, we simulated our algorithm and the robots are located in circle form. The red line is drawn from the ball to the point G, which is mentioned above and it intersects with the centre of the robot in the middle.

From Figure 5.4, we obtain the related equations. The point D_1 between the ball B and the protected point G can be calculated according to Equations 5.1 - 5.3. It is noted that R_2 is located on the left of R_1 and R_3 is located on the right of R_1. After the point D_1 has been found, the desired points for the other blocking robots, D_2 and D_3, can be found as follows:

$$\beta = atan2((D_1.y - B.y), (D_1.x – B.x)) \tag{5.4}$$

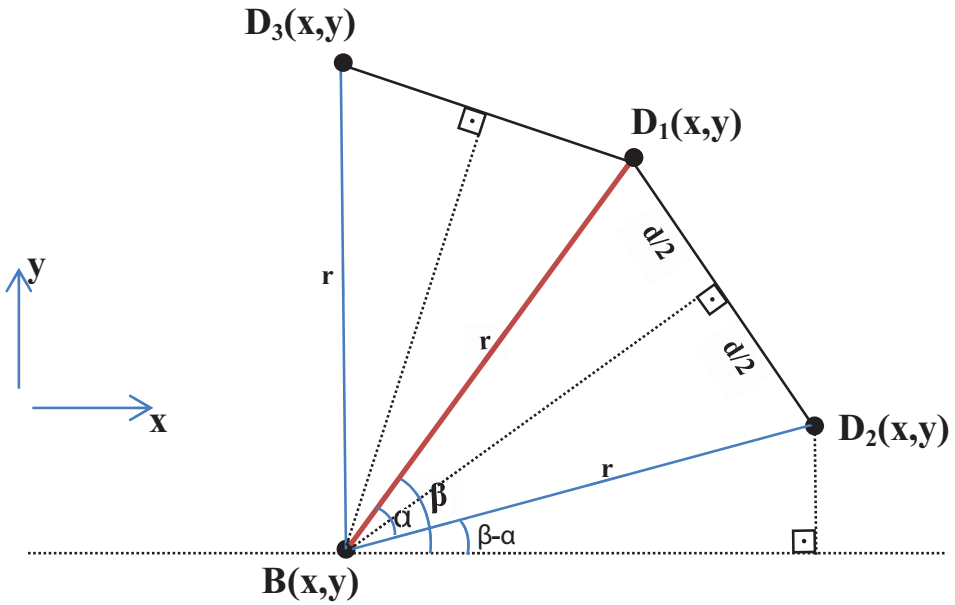

Figure 5.4. How to Block the Ball with Three Robot.

$$\alpha = 2 \times \arcsin(^{\frac{d}{2}}/_{r}) \tag{5.5}$$

$$angle2 = \beta - \alpha \tag{5.6}$$

$$D_2.x = (r \times \cos(angle2)) + B.x \tag{5.7}$$

$$D_2.y = (r \times \sin(angle2)) + B.y \tag{5.8}$$

$$angle3 = \beta + \alpha \tag{5.9}$$

$$D_3.x = (r \times \cos(angle3)) + B.x \tag{5.10}$$

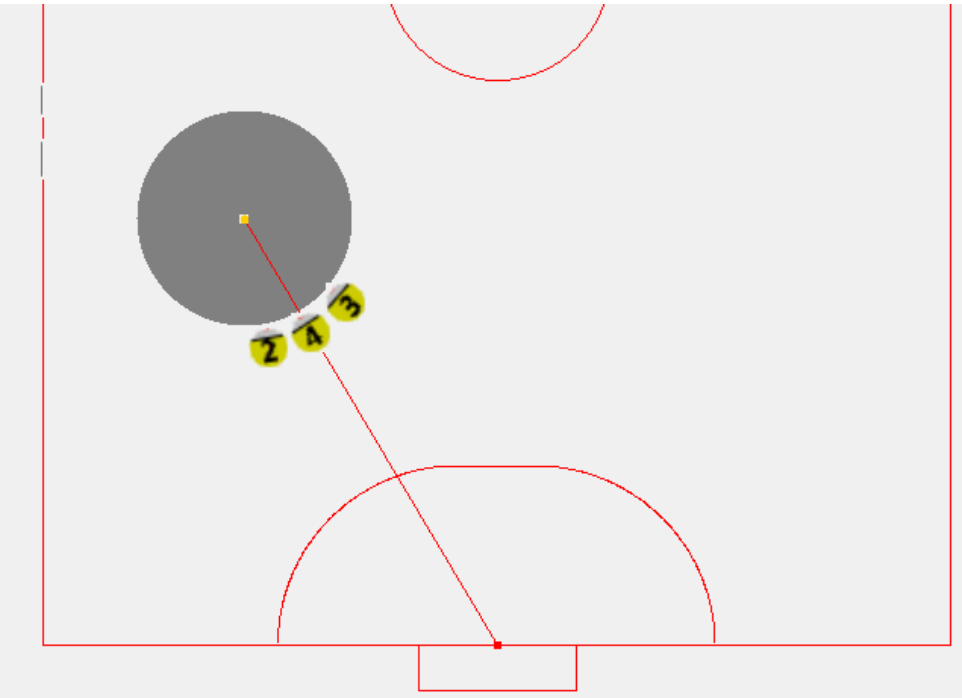

Figure 5.5. Simulation of Ball Blocking.

$$D_3.y = (r \times \sin(angle3)) + B.y \tag{5.11}$$

5.4. Defenders Localization

For good defending, the robots are located around the defense area in a circle form. This localization is similar to the ball blocking localization. But there is a subtle problem in this localization. When positioning the robots in a circle format, one center point must be found according to which the circle will be drawn and the robots will be located on. If the protected point G is considered as a center, the defenders will be located as shown in Figure 5.6.

Let us assume that the protected point is the middle of the goal. As the defense area is not in the shape of a circle, the localization will not be obtained efficiently. Actually the defense area consists of 2 quarter circles with different centers, C_1 and C_3, and a rectangle as illustrated in Figure 5.7.

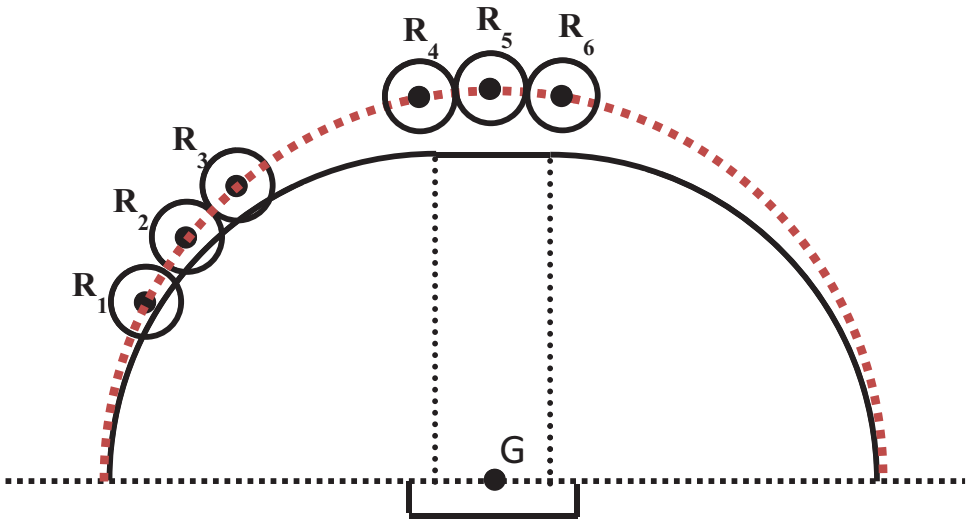

Figure 5.6. Defending the defense area based on one centre point.

In order to make a proper localization, firstly defense area is divided into 3 parts by the l_1 and l_2 lines. The angles between the l_3 line and these two lines are set by the user. Usually these two lines are attempted to be placed on the end of the circles consisting the defense area. A line is drawn from the protected point G to the ball B. This line is named GB. The orientation angle of this line is found using "atan2" function. As the orientations of the l_1 and l_2 lines are already known, it can be decided whether GB line falls in the region below l_1, falls above l_2 or falls between l_1 and l_2. This information is very important for localization. If it falls in the region below l_1, the point C_1 is used as a center of the circle which forms the positioning of the robots; if it falls between l_1 and l_2, the point C_2 is used as a center of the circle; if it falls above the l_2 line, the point C_3 is used as a center of the circle.

Let us assume that the ball is positioned on B_1. If the center and the P_1 points are found, D_2 and D_3 can be found as in the case of blocking ball. The main problem is how to find the P_1 point. Firstly, the point I_1 that is the intersection of the defense area and GB line is found. The related equations are extracted from Figure 5.8. It should be noted that all equations stated below are only for the case depicted in Figure 5.8. If we consider that our robots are blocking the left side of the defense area in Figure 5.8, in order to protect the right side of it some equations should change. In order to protect the other defense area, some equations should change as well. We are here only

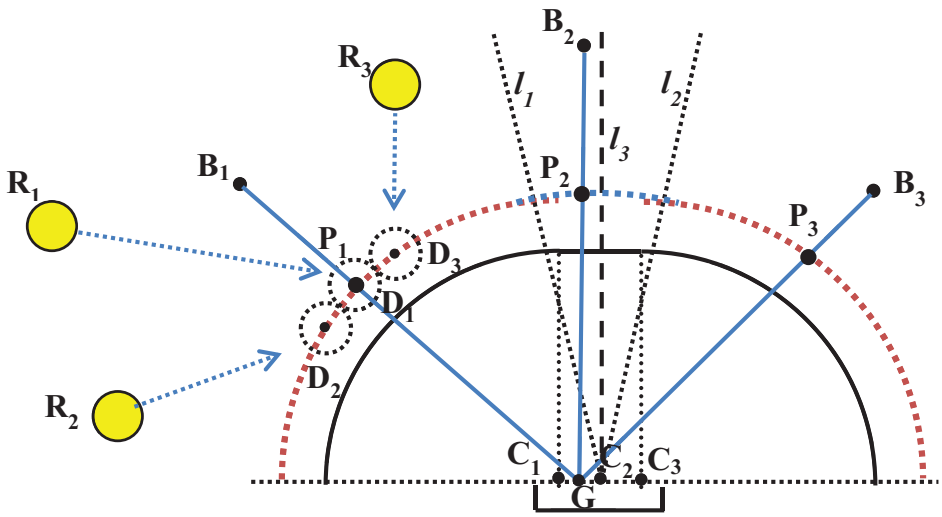

Figure 5.7. Defending the defense area based on three centre point.

focusing on the case in Figure 5.8.

$$a_1 = C_1.y - G.y \tag{5.12}$$

$$\alpha =^{\pi}/_{2} - atan2((B_1.y - G.y), (B_1.x – G.x)) \tag{5.13}$$

$$h_1 = a_1 \times \sin(\alpha) \tag{5.14}$$

$$\theta = \arcsin(^{h_1}/_{r}) \tag{5.15}$$

$$h_2 = r \times \sin(\alpha + \theta) \tag{5.16}$$

$$a_2 = r \times \cos(\alpha + \theta) \tag{5.17}$$

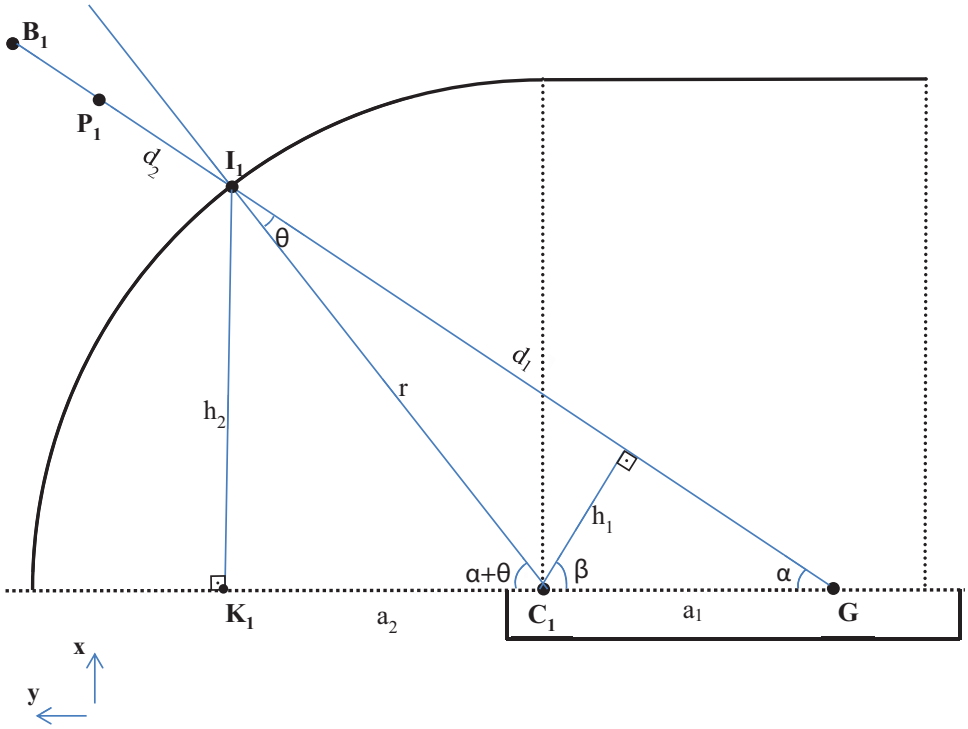

Figure 5.8. How to defend the defense area.

$$I_1.x = G.x + h_2 \tag{5.18}$$

$$I_1.y = C_1.y + a_2 \tag{5.19}$$

After the point I_1 is found, P_1 can be found easily using Equations 5.1-5.3 . After P_1 is found, D_2 and D_3 is found using Equations 5.4-5.11. Figure 5.9 depicts our simulation result. We should note that the robot in the middle is located on D_1, and this point is found based on point G. The other robots are located on a circle with the center point C_1 and with the radius $|C_1D_1|$.

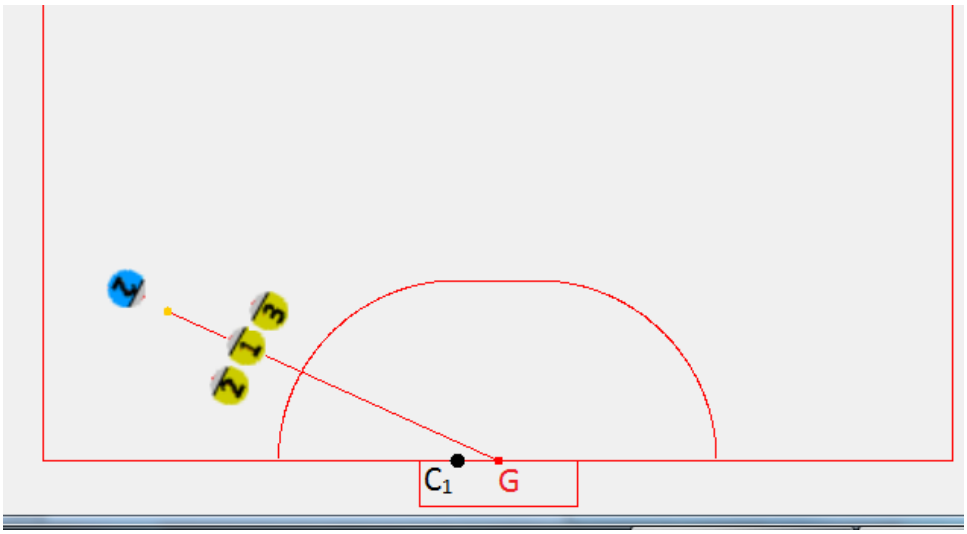

Figure 5.9. Simulation of defending the defense area.

5.5. Scoring

There is not another way except scoring in order to win an SSL game. To score a goal, firstly an empty space must be found in the opponent goal. Empty space refers to the region that if a robot orientates and kicks the ball towards that region, the ball does not hit any obstacle and it is scored as seen in Figure 5.10. The yellow colored space refers to free zones and the red colored space refers to blind zones in goal. After determining this space (free zone), the robot orients itself in the direction of the ball and the middle point of this space S. The robot is positioned back to the ball so that when it kicks the ball, a goal is scored since the ball passes through the free zone without colliding with any obstacle.

In order score, it is required to find the free zones in the goal. The simplest way of doing this is to find the blind zones initially that are related to the ball and the obstacles on the field. First of all, the x coordinates of all obstacles are checked in order to determine whether the obstacle is positioned between the ball and the opponent goal or not. If so, two tangent lines from the ball to the obstacle are drawn. There will be two points P_1 and P_2 which intersect with the tangents and the goal line of the opponent as seen in Figure 5.11. The area which falls between these two points is the blind zone. This procedure is applied to all obstacles. By eliminating the blind zones,

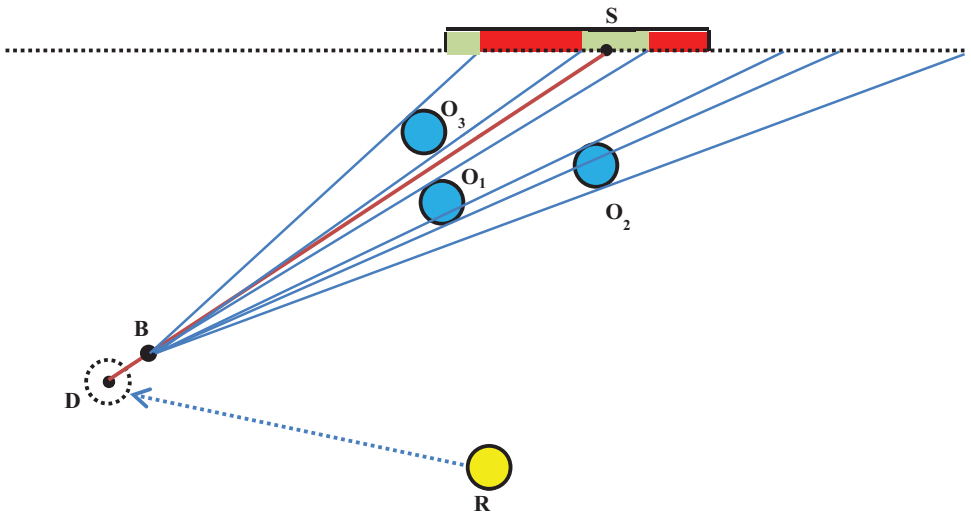

Figure 5.10. Shooting in free zone.

free zones are detected. The shooting robot should orient itself towards the middle of the largest free zone (S) in case there are free zones more than one. In Figure 5.12, our simulation results are presented. The black lines are obtained in order to detect blind zones. If the obstacles are not posing a threat, these lines are omitted. They are tangent lines mentioned above. The red line is drawn from ball to score point which is in free zone.

The equations are stated below in order to find P_1 and P_2 points. The x coordinates are already known, y coordinates are extracted.

$$\alpha = \arctan(\frac{O.y–B.y}{O.x - B.x}) \tag{5.20}$$

$$t^2 = (O.y - B.y)^2 + (O.x - B.x)^2 \tag{5.21}$$

$$\beta = \arcsin(^{|OT_2|}/_t) \tag{5.22}$$

$$angle1 = \alpha + \beta \tag{5.23}$$

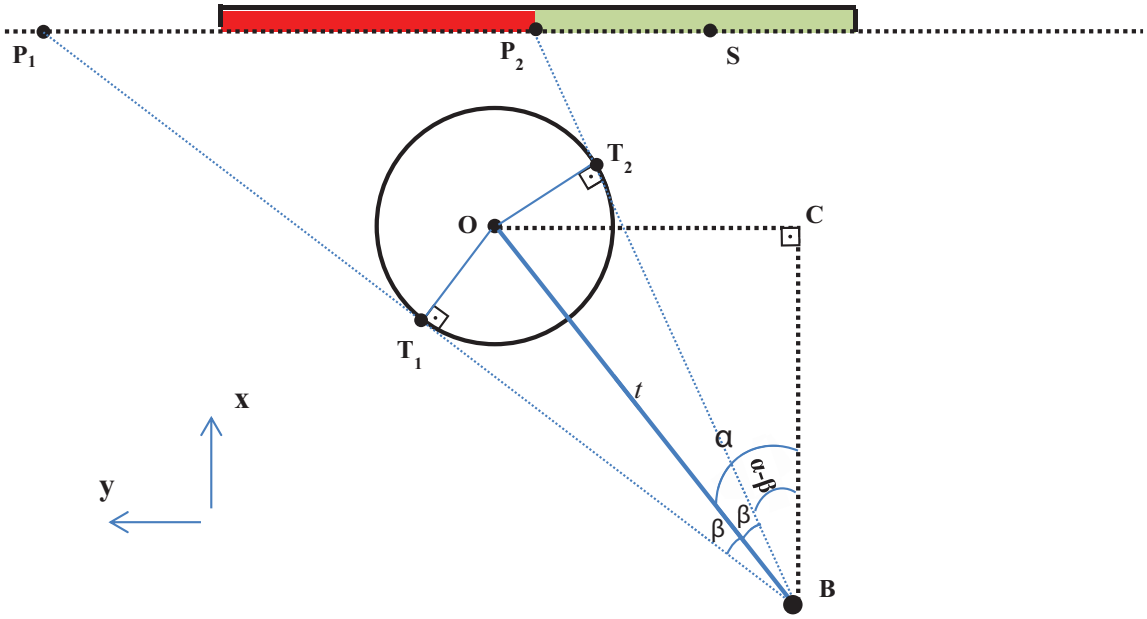

Figure 5.11. How to score.

$$angle2 = \alpha – \beta \tag{5.24}$$

$$P_1.y = (P_1.x \times \tan(angle1)) + (B.y – (\tan(angle1) \times B.x)) \tag{5.25}$$

$$P_2.y = (P_2.x \times \tan(angle2)) + (B.y – (\tan(angle2) \times B.x)) \tag{5.26}$$

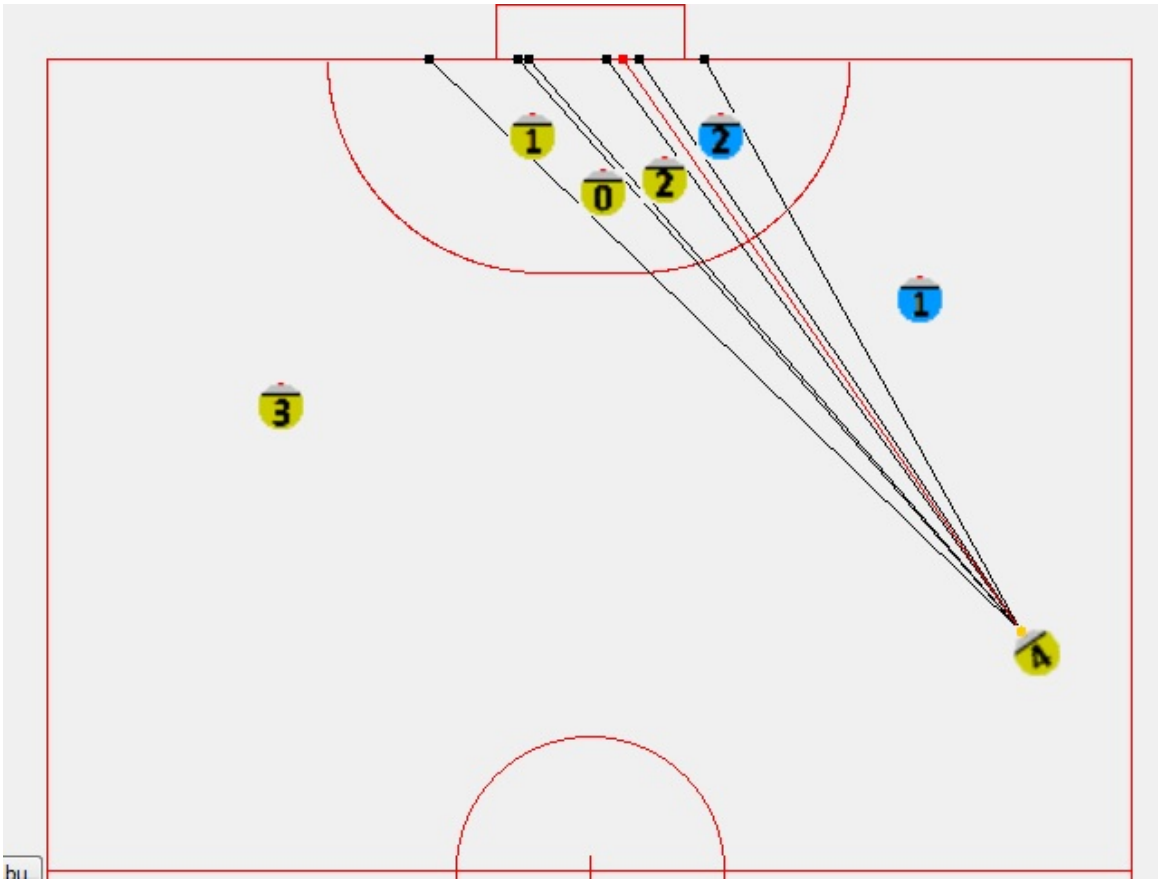

Figure 5.12. Simulation of scoring.

5.6. Ball Position Prediction for Goalie

Goalie should protect the goal by covering the area which poses the biggest threat. As the environment changes frequently, the dangerous area changes as well. Here only the state in which the ball is moving towards the goal will be discussed. While the ball is moving, the goalie should orient itself towards the ball without leaving the goal line. In order to detect the direction of the ball clearly, linear least squares algorithm is used. The last 4 frames are used to predict the future position of the ball. As the SSL vision locates the ball position with an error, it is not effective to depend only on the last two locations of the ball. This is why linear least square technique has been adapted. Using the linear least squares equations, a straight line equation will be extracted. Subsequently, the possible dangerous point on the goal line will be obtained from this equation. The linear least square equation and other related equations are listed below.

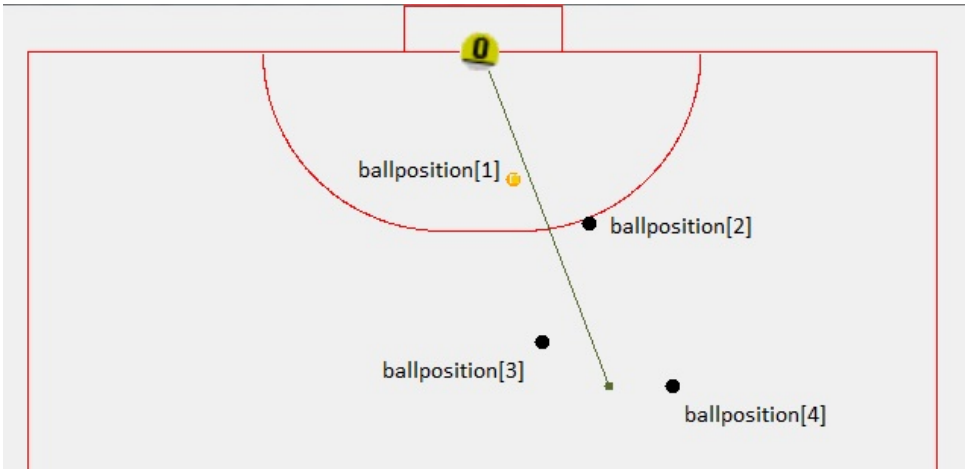

Figure 5.13. Simulation of Linear Least Solution and Goalie Localization.

$$C_L = (X^\top X)^{-1} X^\top Y \tag{5.27}$$

where $C_L = [C_{L1}, C_{L2}]^T$, $X = \begin{bmatrix} 1 & ballposition[4].x \\ 1 & ballposition[3].x \\ 1 & ballposition[2].x \\ 1 & ballposition[1].x \end{bmatrix}$, the last 4 ball position's x coordinates and $Y = \begin{bmatrix} ballposition[4].y \\ ballposition[3].y \\ ballposition[2].y \\ ballposition[1].y \end{bmatrix}$, the last 4 ball position's y coordinates.

Equation 5.27 is solved for each frame and C_L is found newly so that the ball's position can be predicted using Equation 5.28.

$$y = C_{L1} + C_{L2} x \tag{5.28}$$

where x and y refers to the possible future coordinates of the ball.

The implementation of the least squares technique and the detection of the possible future location of the ball on the goal line are depicted in Figure 5.13. The goalie is located on the predicted point whose x coordinate is on the goal line. The line that is seen in this figure from next to *ballposition*[4] towards goalie is the Linear Least Squares Solution.

5.7. Avoiding Defense Area

In SSL, it is a rule that there cannot be a second robot apart from the goalie in the defense area. Otherwise your team is penalized by penalty kick for opponent team. In order not to violate this rule, avoiding defense area is a crucial issue. Even though ERRT is used for obstacle avoidance and path planning, this method gives opportunities of moving smoother and faster if the path is collision free.

There are two different cases to be considered as shown in Figure 5.14. The first one is that the robot, at point R_{11} needs to arrive at D_1. Firstly, we draw a tangent l_1 to the defense area from the robot and the destination point is checked whether it is under the tangent line or not. If so, a line is drawn from the center point C_1 towards the tangent point and SD_1 point is found. Later it is expected for the robot move towards SD_1. After the robot is directed to that point, after 16 ms a new vision data is transmitted to the main computer and there is a new chance for the robot to make new decision about reaching at the desired point D_1. Now the robot is at the point R_{12}. Again the same procedure is rerun and as D_1 point is above the tangent line D_2, the robot is directed towards desired point, because the obstacle has been already avoided.

In Figure 5.14, it is seen that there is also another case for R_{21}. This time scenario is a little bit different than the previous case. The x coordinate of the destination is below the x coordinate of the robot. Actually there may be many different cases for avoiding defense area. Different approaches should be made based on each case. These cases all depend on the robot's and ball's position. For instance the avoiding defense area event may be occurred on the other defense area of the field. This time

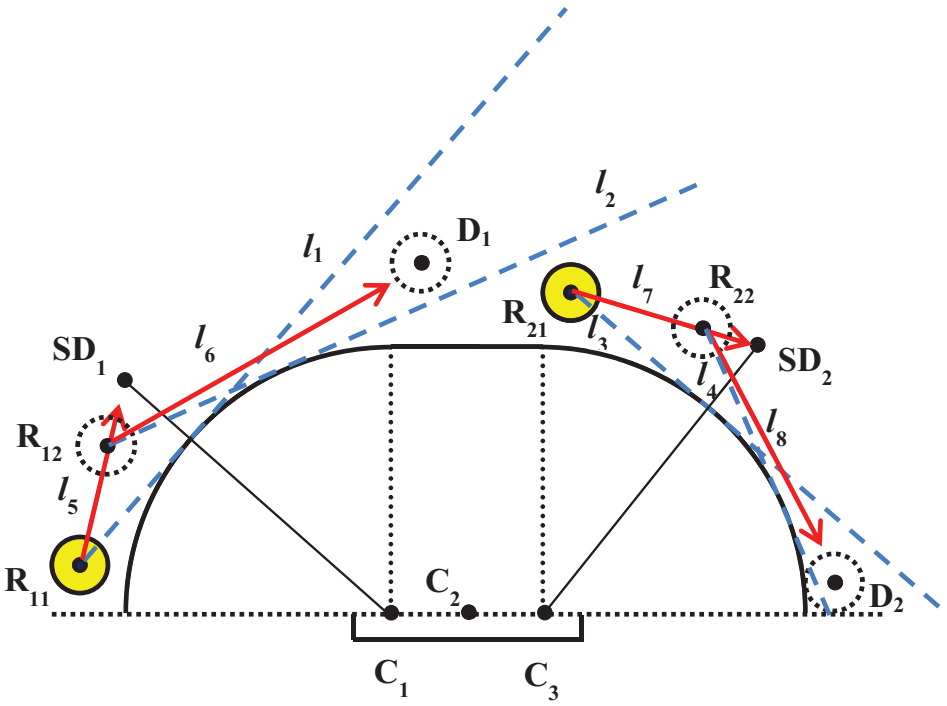

Figure 5.14. Avoiding the defense area.

requires different equations. We will here only focus on the case depicted in Figure 5.15. The equations are listed below for obtaining point P_1. After the point P_1 is found, sub destination point T_1 can be found easily using the equations Equations 5.1-5.3. It is noted that two tangents can be drawn from our robot to defense area. After obtaining two tangent points, the related one is detected as checking the related zone analytically. In our case since the robot should direct towards T_1 point, we will be interested in finding P_1 instead of finding P_2.

In Figures 5.16, 5.17 and 5.18, we simulated our algorithm for different cases. We wanted our robot go to the ball and it tracked the black trajectory in order to avoid penalty area. Red dot refers to our robot's initial point.

$$d = |RC_1| \tag{5.29}$$

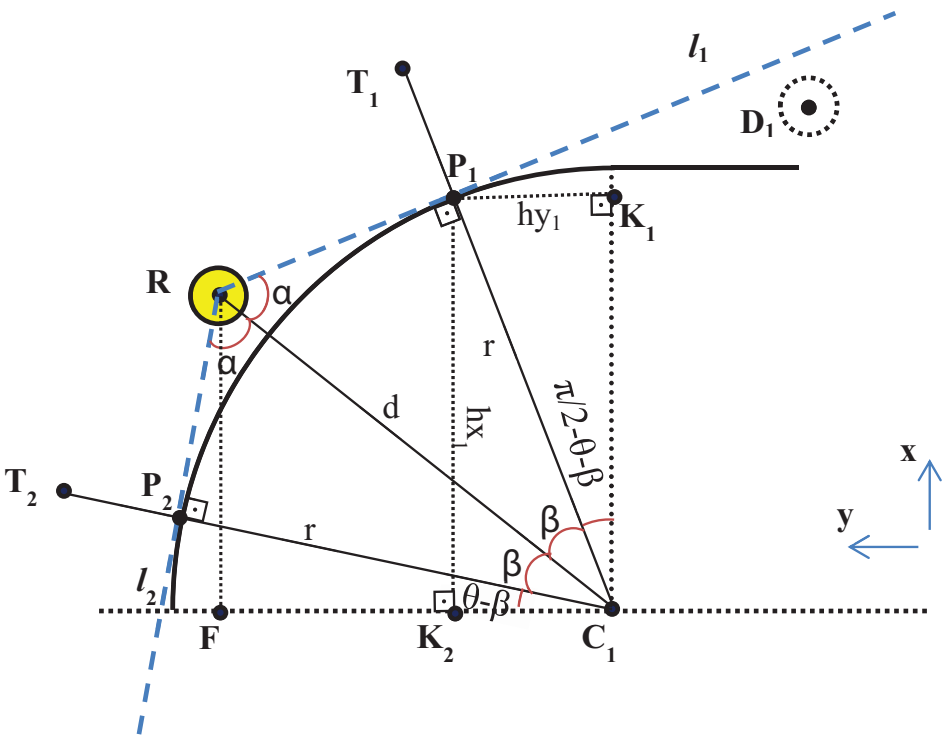

Figure 5.15. How to avoid the defense area.

$$\beta = \arccos({}^{R}/_{d}); \tag{5.30}$$

$$\theta = \arcsin({}^{|RF|}/_{d}); \tag{5.31}$$

$$hx_1 = \sin(\beta + \theta) \times r \tag{5.32}$$

$$hy_1 = \sin({}^{\pi}/_{2} - \theta - \beta) \times r \tag{5.33}$$

$$P_1.x = C_1.x + hx_1 \tag{5.34}$$

$$P_1.y = C_1.y + hy_1 \tag{5.35}$$

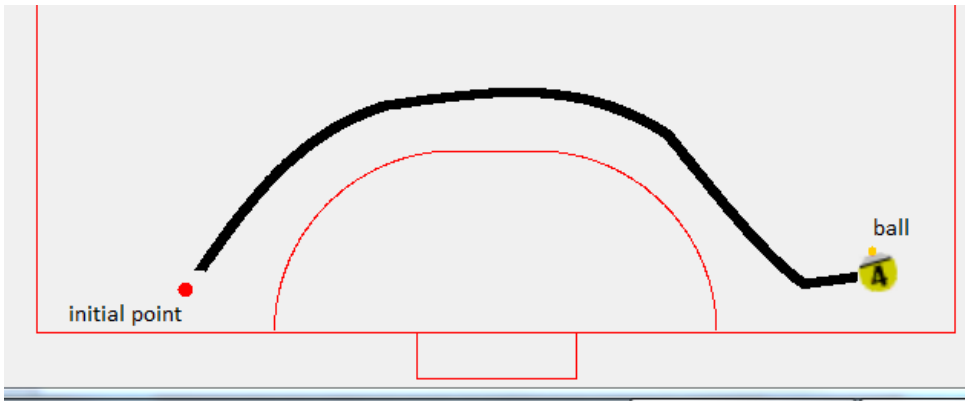

Figure 5.16. Simulation of avoiding the defense area 1.

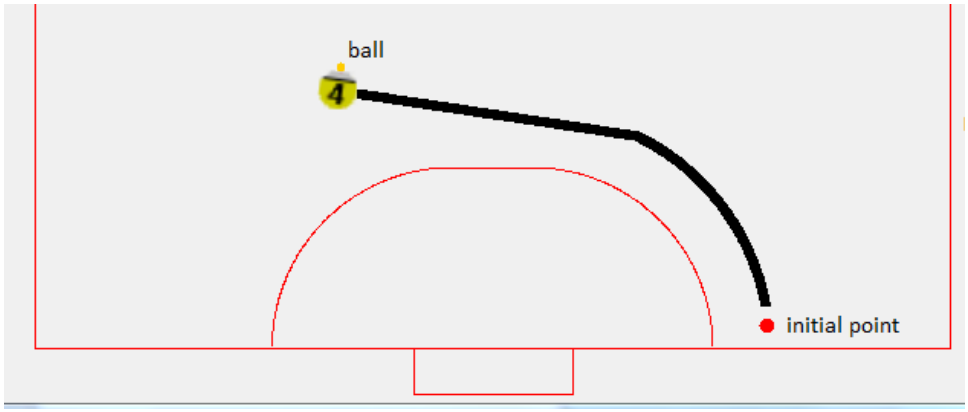

Figure 5.17. Simulation of avoiding the defense area 2.

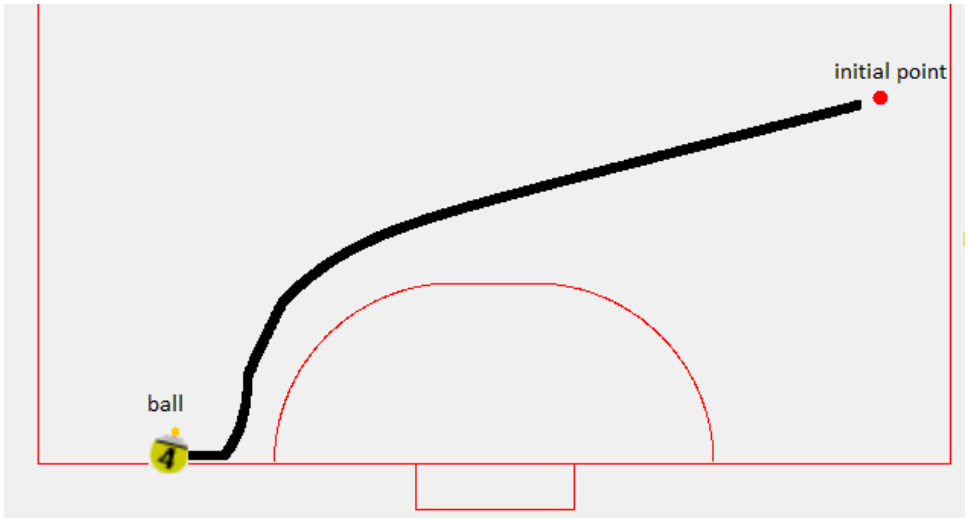

Figure 5.18. Simulation of avoiding the defense area 3.

5.8. Summary of the Chapter

Localization and formation issues have been discussed and new solutions have been proposed. Different scenarios which require formations have been illustrated. For different game states, different localization algorithms have been presented on blocking ball. For defender robots, new localization technique has been introduced for tackling some encountered problems. Desired coordinates and orientation directions have been calculated mathematically in order to be able to score. Ball position prediction algorithm has been introduced for especially protecting the goal. For avoiding the defense area, how a trajectory tracking algorithm may be used instead of ERRT has been presented.

6. NETWORK DELAY COMPENSATION VIA STATE ESTIMATION

In most of control systems, delay is an inherent phenomena. Accordingly, there is no way to eradicate it totally. Delay has many destructive effects on control systems performance such as lowering the stability and accuracy, data loss and slowing down the response of system. It has different origins and sources in control systems. Delay in data transmission, delay in sensors, delay in data processing and delay in actuators are significant kinds of delay in systems.

Almost in every robotic application, accuracy, stability and fast response are important factors. In our case, these three features are so vital for intense performance. Our robots are omni-directional autonomous robots which are fortified and prepared to participate in Small Size Football League (SSL). Thus, accurate path tracking is an important task. Delay is a serious reason for lowering execution quality of this task. Considerable delay may cause collision with obstacles and other robots. It decreases robot's control over ball and moreover, it will bring imprecision in passing and shooting. Therefore, designing a filter in order to decrease the delay's influences can bring more robustness for the robot.

For delay compensation, we should have an approximate information about its amount. We measure velocity response delay of the robot in its own coordinate (V_{r_x} and V_{r_y}) in order to collect the overall delay. As shown in Figure 6.1, the robot velocity is triggered in the direction of V_{r_x} and then, by considering the response signal, approximate amount of delay is achieved which is about 128 millisecond or about 8 frames.

For the delay compensation problem, we offer a Kalman filter which filters the output of the system that is coordinates of the robot and by predicting future coordinates using measured velocity, tries to approximate the current coordinates and

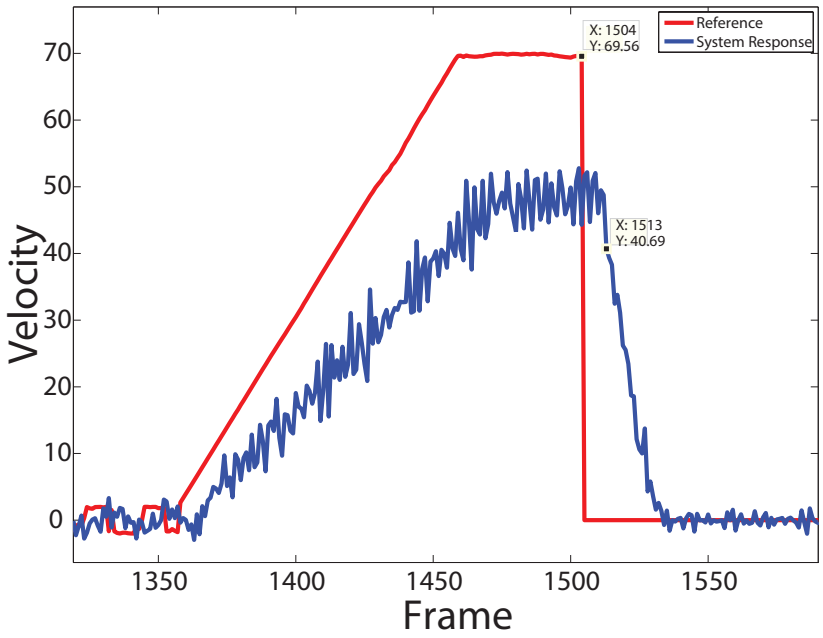

Figure 6.1. Delayed response of the system

therefore, effects of the delay will be lowered. The properties of this method lie on simplicity, low processing delay and accuracy.

This chapter is organized as follows. In Section 6.1, we briefly review some relevant methods that can be used for delay compensation. In Section 6.3 a mathematical modeling of the robot will be covered. In Section 6.4 an augmented system which is responsible for capturing delayed measured dynamics will be represented. Subsequently in Section 6.5, the Kalman filter is explicated and simulation and experimental results are provided in Section 6.6 to verify intense performance of the estimator.

6.1. Related Work

Different approaches have been investigated for delay compensation but one of the most favorite ones is using prediction methods to extract future response of system and compensate delay effects. Some related studies are listed and described superficially below.

The filter which was applied in our case was first introduced by Rudolf E. Kalman in 1960. Even though astronomer Thorvald Nicolai Thiele and radar theoretician Peter Swerling presented alike approach, this filter is named as Kalman Filter [37]. In 1960, A New Approach to Linear Filtering and Prediction Problems was presented [38]. Formulations and solutions to the Wiener problem were proposed in their work. Wiener problem can be listed as below [39]:

- Estimation of random signals.
- Discrimination of random signals from random noise.
- Spotting of some signals such as sinusoidal and wave form signals when random noise is included.

In their studies [40], acclimating the results on memoryless state feedback which were obtained in previous works to the predictor-based case with changing delays.

In their work [41], the systems with inconstant delays are focused. Robustness of the estimator and the depletion of output interferences and calculation noise has been handled by H_∞ performance.

In [42], recursive method was introduced to design predictors for linear, block-feedforward systems with delays in the input and in the states.

In this paper [43], an incomplex continuous-time adaptive controller was presented for time-delay systems. The relative degree of the systems were kept under three. The predictor was inspired by the Smith Controller [44].

The optimal filtering for systems with immediate and constant delayed measurements was studied in another work [45]. Re-organized innovation approach [46] was applied for acquiring a simple approach. The study is based on Kalman Filter.

6.2. Description of the Setup

The overall system consists of different parts and components such as network components, central controller (high level controller), low level controllers, vision system and cameras and the robot itself [47]. All of these systems work together in order to change position of the robot from one place to another. Reference command from the game strategy planner or the user is being sent to the system, then high level control will calculate reference speed of each wheel and then low level control will adjust the speed of its corresponding wheel, therefore the robot moves. In this way, cameras shoot movement of the robot frame by frame. Therefore, period of sampling (T) in the system equals to one frame. The vision system and its software are responsible for detecting and computing position and velocity of the robot in the field coordinates and send it to the high level control to obtain the error. Apparently, data processing and transmission between these components causes delay in the system response. Also, robot's actuators and sensors produce their own response delays too. That is how, the whole system, as a complex system, faces a considerable delay which can bring many defects to the system. Figure 6.2 illustrates the general working loop of the system.

In our system, two overhead cameras that can provide feedback on the robot positions and a host computer that acts as a supervisor are utilized. The host computer receives/processes the vision data, and sends control commands to the robots accordingly. Our vision system consists of two 60 fps digital cameras which provide the visual feedback to the controlling computer. The SSL–Vision software provides the coordinates of the robots and the ball location via a graphical interface once color and field calibrations are done properly based on the light intensity of the field [47]. Also, for data transmission between robot and central computer, XBee modules are exerted.

Our robots consist of four custom–built omniwheels, each of which is driven by a 30 $watt$, 4370 rpm brushless DC motor [47]. The microcontroller is used to estimate the motor speeds and a controller logic is implemented on the microprocessor for precise speed control. The structure of the robot and the wheels is adjusted in a way that there is no maneuverability constraint exist for the robot movement. For gaining this goal,

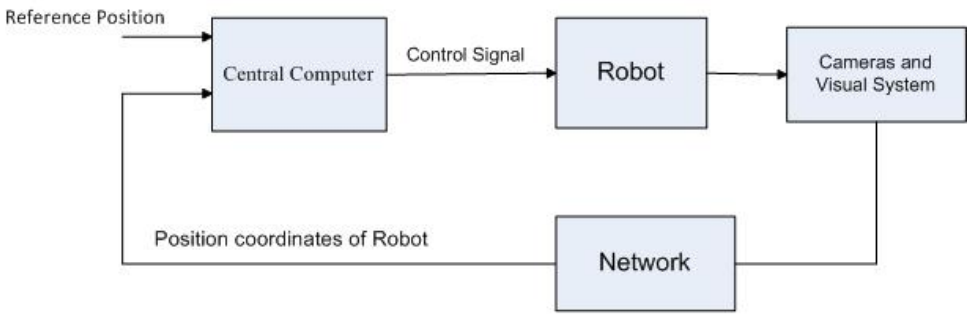

Figure 6.2. System's main components working loop

wheels need to have special orientation corresponding to robot's coordinates. These orientations are $\alpha_1 = 21.46°$, $\alpha_2 = 158.54°$, $\alpha_3 = 222.48°$ and $\alpha_4 = 317.56°$.

As mentioned above, the amount of delay in our system is approximated about 8 frames. Note that each frame is about 16.7 millisecond. Now, assume that our robot moves in an arbitrary direction with speed of 2.5 m/s. By a simple calculation, the amount of error in term of position measurement is obtained 33.4 cm. Obviously, this considerable amount of error has destructive influences on the system behavior. Thus, our aim is compensating this much error and its effects. The Figure 6.3 shows that by increasing the delay, path tracking ability will be worse by comparing the robot trajectory with different amount of delay from the point (0,0) to (1000,1000) and in Figure 6.3(d) we can see that the robot can not follow the trajectory to the desired point, which is (1000,1000), due to high amount of delay.

6.3. Mathematical System Modeling

In this section, we start by introducing a simple model which describes dynamics of the autonomous robot. As far as the high level strategy controller is concerned, the state of the robot can be summarized by:

$$s(t) = [x(t),\ y(t),\ \theta(t)]^T \tag{6.1}$$

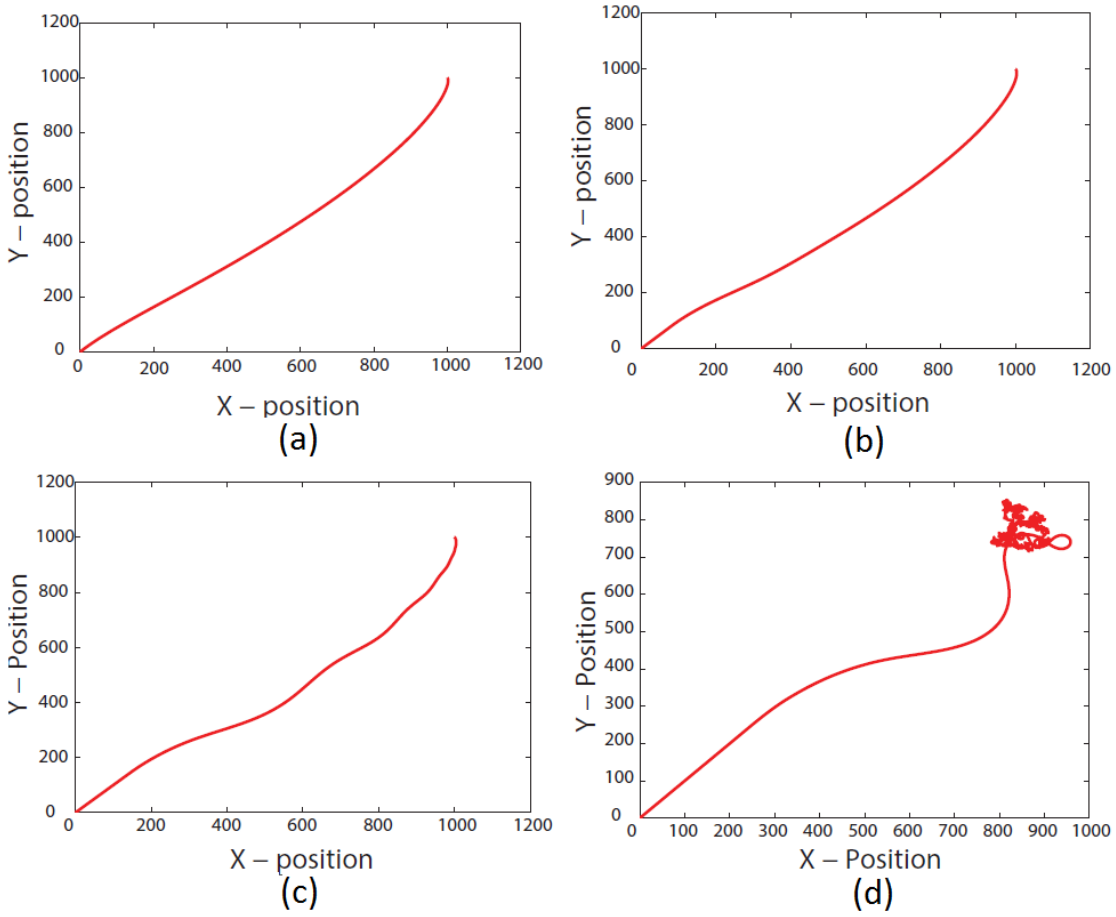

Figure 6.3. Robot's trajectory (a) without delay, (b) with 4 frames delay, (c) with 8 frames delay and (d) with 15 frames delay

where $x(t)$ and $y(t)$ represent the position of the robot at time t and $\theta(t)$ shows heading of the robot. Taking into account all actuator effects, the low-level state controller of the robot can be roughly characterized by:

$$s(t+1) = s(t) + Tu(t) \tag{6.2}$$

where $u(t)$ is control signal and equals to:

$$u(t) = [v_x(t),\ v_y(t),\ \omega(t)]^T \tag{6.3}$$

that $v_x(t)$ and $v_y(t)$ are speed of the robot in direction of x and y respectively and $\omega(t)$ is rotational speed of the robot. The objective is to adjust the control signal such that

the robot can follow the desired trajectory to the reference point which is:

$$s_r = [x_r,\ y_r,\ \theta_r]^T \tag{6.4}$$

In order to meet this objective, we need to compute desired speed of each wheel so that robot can achieve the desired coordinate. We use the following equation to compute the desired velocity:

$$\omega_n = -v_{r_x}\cos(\alpha_n) + v_{r_y}\sin(\alpha_n) + (x_{r_n}\sin(\alpha_n) - y_{r_n}\cos(\alpha_n))v_{r_\alpha} \tag{6.5}$$

where n is number of wheel, v_{r_x}, v_{r_y} and v_{r_α} are velocity of robot in robot's origin coordinate in directions of x, y and θ respectively and x_{r_n} and y_{r_n} are horizontal and vertical distances of the n^{th} wheel from robot's center. Also, a digital PD controller is used for controlling each wheel's motor [47].

After computing wheels velocities and robot velocity, the velocity of the robot in the field coordinate can be obtained using below equation:

$$v = r(\theta)v_r = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & K_{p\theta} \end{bmatrix} (s_r - s)/T \tag{6.6}$$

As mentioned before, the system response has considerable delay due to different reasons. So position which the vision system measures and sends to the high level control belongs to D frame before. Hence, the measured output is:

$$s(t-D) = [x(t-D)\ y(t-D)\ \theta(t-D)]^T \tag{6.7}$$

In order to decrease damaging effects of this delay, we implement an augmented system

to minimize these effects by recording delayed dynamics of the system.

6.4. Augmented System Representation

In this section, we will describe the augmented system state equations, which captures the dynamics associated with delayed measurements. To this end, we define $s_a(t)$, as below:

$$s_a(t) = [s^T(t),\ s^T(t-1),\ ...,\ s^T(t-D)]^T \tag{6.8}$$

where $s_a(t)$ is state vector of the augmented system. Note that, $s_a(t)$ is a vector with $3(D+1)$ elements by defining

$$A = \begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}_{m\times m} \tag{6.9}$$

$$B = [I,\ 0,\ \cdots,\ 0]^T_{m\times l} \tag{6.10}$$

$$C = [0,\ 0,\ \cdots,\ I]_{l\times m} \tag{6.11}$$

we construct augmented system's state space representation as:

$$s_a(t+1) = As_a(t) + Bu_a(t) \tag{6.12}$$

$$z_a(t) = Cs_a(t) \tag{6.13}$$

where m equals to $3(D+1)$, $l=3$, I is the 3×3 identity matrix and z_a is output of the estimator.

6.5. Kalman Estimator Implementation

Based on the augmented system described in Section 6.4, we can implement the observer. In this section, we will briefly describe Kalman filter implementation. The aim of the Kalman estimator is to provide the optimal solution for the above estimation problem. Therefore, considering Equation 6.12 and 6.13, we rewrite them as below by adding white process noise (ξ) and white measurement noise (η).

$$s_a(t+1) = As_a(t) + Bu_a(t) + G\xi \tag{6.14}$$

$$z_a(t) = Cs_a(t) + H\xi + \eta \tag{6.15}$$

With known inputs u, white process noise and measurement noise satisfying $E(\xi) = E(\eta) = 0$, $E(\xi\xi^T) = Q$, $E(\eta\eta^T) = R$ and $E(\xi\eta^T) = N$, where $E(\chi)$ computes expected value of χ, construct a state estimate $\bar{s}$ that minimizes the steady state error covariance which is:

$$P = \lim_{t\to\infty} E(\{s_a - \bar{s}\}\{s_a - \bar{s}\}^T) \tag{6.16}$$

The optimal solution is the Kalman filter with the following equation:

$$\bar{s}(t) = A\bar{s}(t) + Bu(t) + L(z_a(t) - C\bar{s}(t)) \tag{6.17}$$

$$\bar{z} = C\bar{s} \tag{6.18}$$

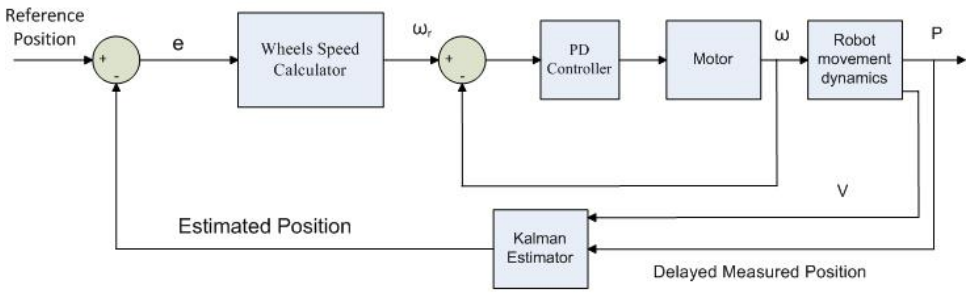

Figure 6.4. Closed loop control system with delay compensation

The filter gain is calculated using an algebraic Riccati equation, which is:

$$L = (PC^T + \bar{N})\bar{R}^{-1} \tag{6.19}$$

where

$$\bar{R} = R + HN + N^T H^T + HQH^T \tag{6.20}$$

$$\bar{N} = G(QH^T + N) \tag{6.21}$$

and P solves the corresponding algebraic Riccati equation. Also, $\bar{z}$ estimates the true system output. The Figure 6.4 shows the control loop of the overall system with Kalman estimator.

6.6. Numerical Analysis

6.6.1. Simulation Results

In this part, we demonstrate that the overall system has better performance using Kalman filter by showing the simulation results using MATLAB software. The amount of delay is considered 8 frames and the Kalman estimator applied to the system as shown in figure 4. The success of the method lies on filtering the output data and processing it in less than one frame period which is meant 18 millisecond. Figure

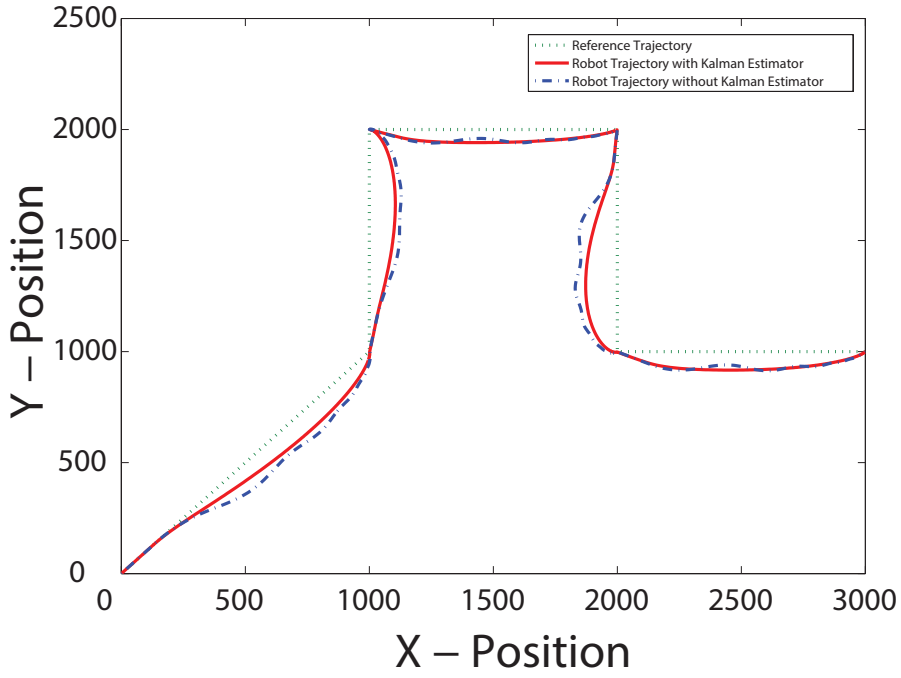

Figure 6.5. Simulation results: path trajectory comparison of the robot with and without applying Kalman estimator

6.5 illustrates path tracking of the robot with and without Kalman estimator. As presented, by applying Kalman estimator, robot movement is much more accurate and oscillations in movement are almost disappeared.

6.6.2. Experimental Verification

In this section, we evaluate the performance of the proposed closed-loop system (Figure 6.4) in real experiments by applying the Kalman filter to the robot. First, we made our robot to move in a trajectory and later we observed the coordination data and orientation of robot with Kalman filter and without implementing it. As we measured the delay time before which was 8 frames, we based all our experiment on this amount and we adjusted the augmented system and Kalman estimator using this amount of delay.

Figure 6.6 depicts error of the data between real-time and system output with and without applying Kalman filter. Figure 6.7 shows orientation response of the robot

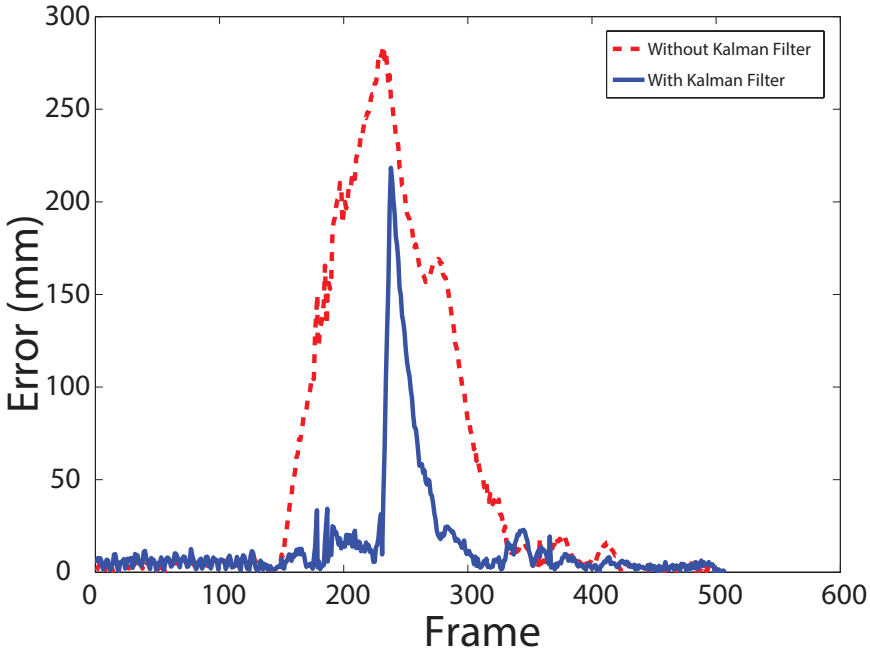

Figure 6.6. Amount of error with and without Kalman Filter

depending on time with and without Kalman filter. Figure 6.8 and Figure 6.9 represent x and y coordinates of the robot with and without implementing the Kalman filter. It is shown that Kalman filter has good performance in decreasing effects of the delay and also, it can predict future responses intensely.

6.7. Summary of the Chapter and Concluding Remarks

In this chapter, we discussed the problem of delay and its effects on our autonomous robots. Then, a Kalman filter which minimizes these effects using an augmented system was proposed. At the end, the simulation and experimental results were demonstrated which verified the improved performance of the overall closed-loop control system using this filter. Adopting this filter brings more accuracy and faster response for the robot. Morover, the control of the robot over ball will be better. Therefore, quality of shooting and passing will increase and game play of the team will upgrade.

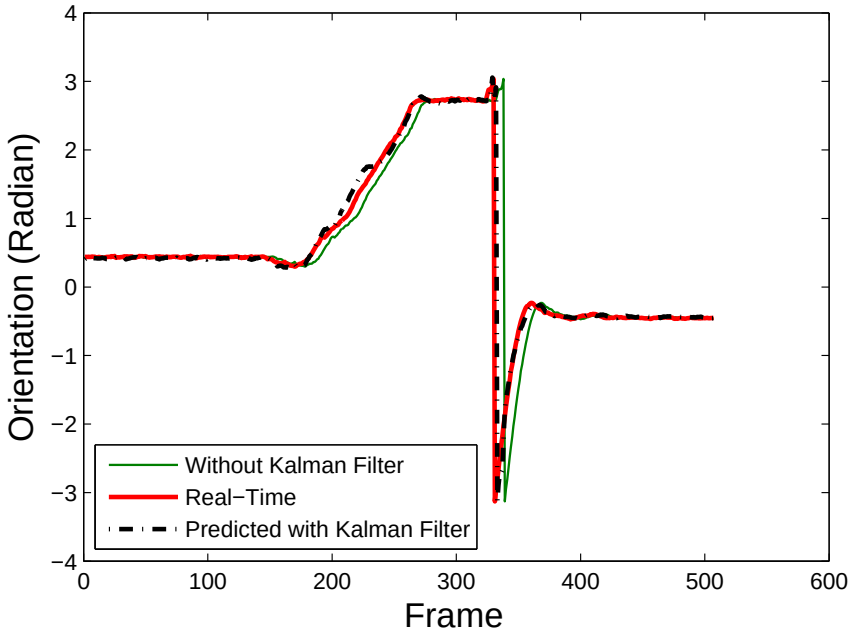

Figure 6.7. Orientation response of the robot

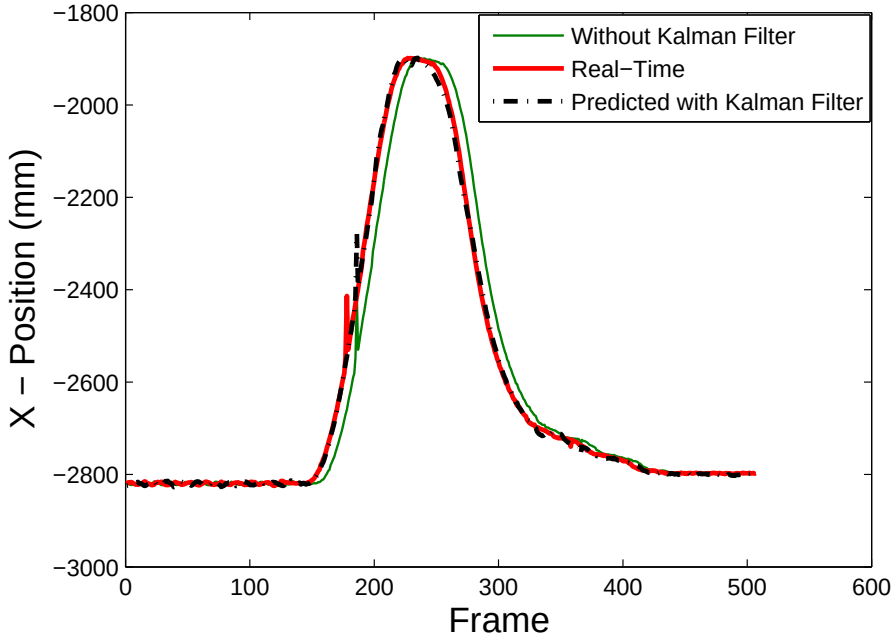

Figure 6.8. Trajectory following of the robot over x coordinate

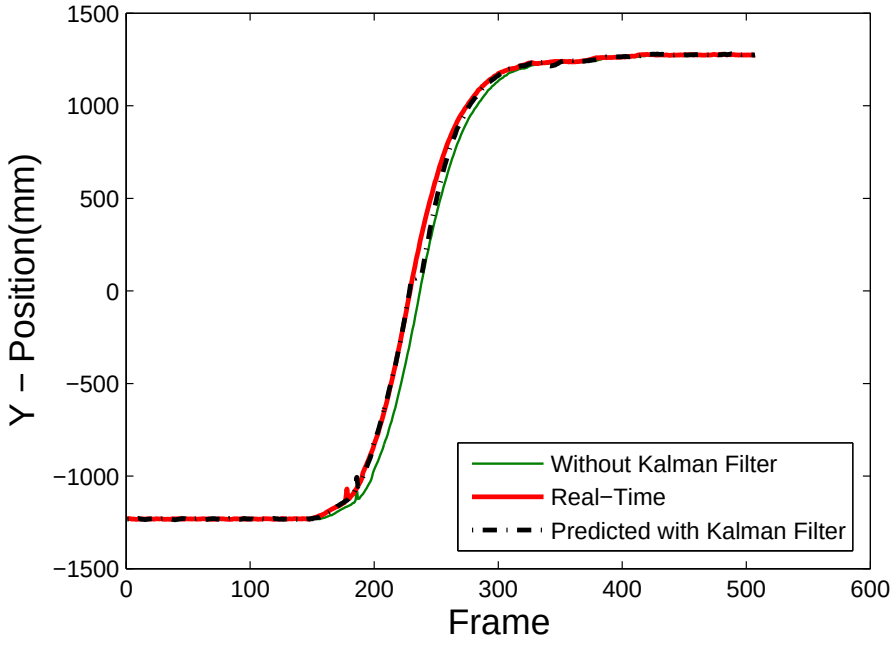

Figure 6.9. Trajectory following of the robot over y coordinate

7. CONCLUSION

SSL is a challenging platform which allows people carry out not only theoretical but also practical multidisciplinary research. In this thesis, we have discussed the contributions made to the SSL team BRocks in terms of hardware and software.

In Chapter 2, the new electronic board design is explained in detail. The new circuit has many sub units including a communication unit, an MC, a motor driver unit, an encoder unit and voltage regulators. While two MCs were being used to process a number of threads in our previous design, one MC in the new design is able to handle with all by itself. MC33035 is still used as the motor driver. However, many problems have been solved by using it such as driving 3 phases and sensing the current. Encoder E4P is used for detecting the actual speed of the motor instead of Hall sensors. By this transition, the resolution has been increased and the control of the motor is more stable and accurate. A new concept that increases efficiency in the new design is the switched-mode regulator in addition to a linear voltage regulator. Our main board has also other functionalities such as setting and observing identity of the robot easily. Additionally a daughter board is designed for the kicking mechanism.

In Chapter 3, new algorithm has been introduced in order to control our DC motors. A bug that occurs when the actual speed of the motor is forced to decrease by the algorithm is fixed by forcing the motor to brake until the speed of it reaches at a specific level. In this way, the settling time has decreased from seconds to milliseconds.

In Chapters 4 and 5, the implementation of ERRT as obstacle avoidance and path planning algorithm for dynamic multi-robot systems has been discussed. For defense area on SSL field a different obstacle avoidance algorithm has been coded; different formation control algorithms have been developed for the coordination of the robots and implemented on the real system; ball position prediction algorithm has been developed; new game states and player states have been created as part of the strategy planning module. All algorithms are first tested in our simulation program

before implementing them on the real system.

In Chapter 6, a remedy for vision based system delay has been in focus. As the vision system is default for each team in SSL, and it has been already reached at a good level in terms of optimization, there is not much to do for improving it. However, it is necessary to compensate for the network induced delay. In this thesis, we have compensated this delay using the Kalman Filter technique and implemented in our system.

Even though many important improvements have been made in terms of hardware and software in order to compete in SSL when compared to previous years, there are still issues which need to be optimized.

In hardware, the current circuit needs to be revised to have optimum performance in robots. The delay which is caused by RC low pass filter when transforming PWM signal into reference voltage level need to be reduced as much as possible. This can be met by driving motor in a different way such as driving directly by PWM signal without using an IC. Besides, the phases of the motors need to be driven by one by contrary to the current method. At present, the motors are driven by microcontroller as if they consist of only one phase; the IC MC33035 drives the phases individually. The whole control of driving motors can be done individually instead of letting MC33035 do it. For synchronization of driving the phases, FPGA may be a good solution because of its ability of parallel processing stemming from its nature.

In software, there are also a few parts which need to be taken care of. Obstacle avoidance and path planning algorithms need to be optimized. Path smoothing algorithms can be developed and alternative solution can be developed for different environments. ERRT algorithm is used as path planning and obstacle avoidance in our system but in some cases different methods can provide better solutions. For instance for avoiding defense area we come up with a different solution and for different cases and environments, various methods can be utilized. New strategies should be added in our system in order for our robots to play better.

REFERENCES

1. *XBee Wireless Module DataSheet*, http://ftp1.digi.com/support/documentation/90000982_B.pdf.

2. *MC9S08DZ128*, http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08DZ128.pdf.

3. *MC33035 DataSheet*, http://www.onsemi.com/pub_link/Collateral/MC33035-D.PDF.

4. *The First Successful Airplane*, http://airandspace.si.edu/exhibitions/wright-brothers/online/fly/1903/index.cfm.

5. *July 20, 1969: One Giant Leap for Mankind*, http://www.nasa.gov/mission_pages/apollo/apollo11_40th.html.

6. *Deep Blue*, http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/.

7. *ENIAC*, http://www.britannica.com/EBchecked/topic/183842/ENIAC.

8. *Mars Pathfinder*, http://www.bbc.co.uk/science/space/solarsystem/space_missions/mars_pathfinder.

9. *RoboCup Official Website*, http://www.robocup.org/.

10. *Objective of RoboCup*, http://www.robocup.org/about-robocup/objective/.

11. *A Brief History of RoboCup*, http://www.robocup.org/about-robocup/a-brief-history-of-robocup/.

12. *Regulations & Rules*, http://www.robocup.org/about-robocup/

regulations-rules/.

13. *Laws of the RoboCup Small Size League 2013*, http://robocupssl.cpe.ku.ac.th/_media/rules:ssl-rules-2013-2.pdf.

14. *MC9S08DV32 DataSheet*, http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08DV60.pdf.

15. *Maxon Motor E45 30W DataSheet*, http://www.electromate.com/ftp/public/Maxon%20Product%20Catalogs/Maxon%20Catalog%202009-2010/195.pdf.

16. *Maxon Motor E16 15W DataSheet*, http://www.electromate.com/ftp/public/Maxon%20Product%20Catalogs/Maxon%20Catalog%202009-2010/147.pdf.

17. *LM1117 DataSheet*, http://www.ti.com/lit/ds/symlink/lm1117-n.pdf.

18. *LM2575 DataSheet*, http://www.ti.com/lit/ds/symlink/lm1575.pdf.

19. *E4P Encoder*, http://www.usdigital.com/products/e4p.

20. *74HC86 DataSheet*, http://www.onsemi.com/pub_link/Collateral/74HC86.REV1.PDF.

21. *NCP1054 DataSheet*, http://www.onsemi.com/pub_link/Collateral/NCP1050-D.PDF.

22. Karaoğuz, H., M. Usta and M. Akar, "Design and Implementation of a Hierarchical Hybrid Controller for Holonomic Robot Formations", *10th International Conference on Control, Automation, Robotics and Vision*, pp. 1993–1998, 2008.

23. Barraquand, J. and J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach", *International Journal of Robotics Research*, Vol. 10, pp. 628–649, 1991.

24. Amato, N. M. and Y. Wu, "A Randomized Roadmap Method for Path and Manipulation Planning", *IEEE International Conference on Robotics and Automation*, pp. 113–120, 1996.

25. Kavraki, L. E., P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces", *IEEE Transactions on Robotics and Automation*, Vol. 12, pp. 566–580, 1996.

26. Lavalle, S. M., "Rapidly-exploring Random Trees: A New Tool for Path Planning", *Technical Report No. 98-11*, 1998.

27. Valle, S. M. L. and J. J. J. Kuffner, "Randomized Kinodynamic Planning", *International Journal of Robotics Research*, Vol. 20, pp. 378–400, 2001.

28. Pohl, I., "Bi-directional and Heuristic Search in Path Problems", *Technical report, Stanford Linear Accelerator Centor*, 1969.

29. Bruce, J. R. and M. Veloso, "Real-Time Randomized Path Planning for Robot Navigation", *Intelligent Robots and Systems, IEEE/RSJ Intl. Conf. on.*, Vol. 3, pp. 2383–2388, 2002.

30. Martin, S. K. M., P. Klupar and J. Winter, "TechSat 21 and Revolutionizing Space Missions Using Microsatellites", *In Proc. of USU/AIAA Conf., Logan, UT,*, 2001.

31. Cowan, N., O. Shakerina, R. Vidal and S. Sastry, "Vision-based Follow-the-leader", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1796–1801, 2003.

32. Ren, W. and R. W. Beard, "A Decentralized Scheme for Spacecraft Formation Flying via The Virtual Structure Approach", *Proceedings of the 2003 American Control Conference*, pp. 1746–1751, 2003.

33. Scharf, D. P., F. Y. Hadaegh and S. R. Ploen, "A Survey of Spacecraft Forma-

tion Flying Guidance and Control (Part 2)", *Proceedings of the American Control Conference*, 2004.

34. Balch, T. and R. C. Arkin, "Behavior-based Formation Control for Multirobot Teams", *IEEE Trans. on Robotics and Automation*, Vol. 14, pp. 1–15, 1998.

35. Lewis, M. A. and K. H. Tan, "High Precision Formation Control of Mobile Robots Using Virtual Structures", *Autonomous Robots*, Vol. 4, pp. 387–403, 1997.

36. Desai, J. P., J. Ostrowski and V. Kumar, "Controlling Formations of Multiple Mobile Robots", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 2864–2869, 1998.

37. *Kalman Filter*, `http://en.wikipedia.org/wiki/Kalman_filter`.

38. Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems", *Transactions of the ASME, Series D, Journal of Basic Engineering*, Vol. 82, pp. 35–45, 1960.

39. Wiener, N., "The Extrapolation, Interpolation and Smoothing of Stationary Time Series", *John Wiley & Sons, Inc., New York*, 1949.

40. Gonzalez, A., A. Sala and P. Albertos, "Predictor-based Stabilization of Discrete Time-varying Input-delay Systems", *Automatica*, Vol. 48, pp. 454–457, 2012.

41. Sanchis, R., I. Pearrocha and P. Albertos, "Design of Robust Output Predictors under Scarce Measurements with Time-varying Delays", *Automatica*, Vol. 43, pp. 281–289, 2007.

42. Jankovic, M., "Recursive Predictor Design for State and Output Feedback Controllers for Linear Time Delay Systems", *Automatica*, Vol. 46, pp. 510–517, 2010.

43. Niculescu, S. I. and A. M. Annaswamy, "An Adaptive Smith-controller for Time-delay Systems with Relative Degree n$\leq$2", *Systems & Control Letters*, Vol. 49, pp.

347–358, 2003.

44. Smith, O. J. M., "A Controller to Overcome Dead Time", *ISA Transactions*, Vol. 6, pp. 28–33, 1959.

45. Lu, X., H. Zhang, W. Wang and K. L. Teo, "Kalman Filtering for Multiple Time-delay Systems", *Automatica*, Vol. 41, pp. 1455–1461, 2005.

46. Zhang, H., L. Xie and Y. C. Soh, "A Unified Approach to Linear Estimation for Discrete-time Systems—Part I: H2 Estimation", *Proceedings of 41th IEEE conference on decision and control*, 2001.

47. Varol, O. F., O. Cihan, H. Esen and M. Akar, *BRocks Team Description Paper*, 2012.