A NEW GENERATION EMBEDDED SYSTEMS DESIGN FOR ROBOCUP SSL ROBOTS

by

Günay Turan

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2015

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 and Electronics Engineering
Boğaziçi University
2018

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisor Prof. Dr. Yasemin Kahya and co-advisor Prof. Dr. Mehmet Akar for their guidance during my thesis work. Besides, I would like to thank my evaluation committee members Asst. Prof. Faik Başkaya and Asst. Prof. İpek Şen.

I sincerely thank to my colleagues in NECS-Lab, Özlem Feyza Erkan, Ümit Develer, Halil Yiğit Öksüz, and Ceren Çoker for their constant support and friendship throughout years. I have to mention many people for their contribution on this thesis work; I sincerely would like to thank Erman Kömürcü for his support in firmware development, Yalçın Erdoğan for sharing his deep knowledge in motor control and hardware design, Esen Yel for her overseas help on PI control, Timur Altınsoy for 3D modelling everything I need, and Alperen Zengin for coaching and supporting me in the most difficult times. Finally, I would like to thank Halil Samed Çıldır specially, for his contributions to both high and low level software development and debug processes.

I also thank to TÜBİTAK for their financial support during my Master's studies with 2210-A scholarship and would like to thank Boğaziçi University BAP (Scientific Research Projects) for supporting this project and enabling us to implement hardware design.

Last but not the least, I am more than grateful to my dear family for leading me to everything I have and I am.

ABSTRACT

A NEW GENERATION EMBEDDED SYSTEMS DESIGN FOR ROBOCUP SSL ROBOTS

RoboCup Small Size League (SSL) is one of the most competitive divisions in the RoboCup where international universities compete with their state of the art robotic hardware and software designs. In the SSL, teams build their own robots which move by four omniwheels; therefore, balance of the robot is trivial compared to humanoid competitions. This feature of the SSL enables teams to build faster robots and focus on more advanced AI development strategies. However, it is essential to have a robot pool that is robust enough to comply with AI directives. In this thesis, we have developed a new generation of robotic hardware and low level firmware as well as low level control algorithms for improvements in precise robot movement. Moreover, a fully modular system design is of importance due to highly evolving nature of the competition. This thesis focuses on the work and methods used to overcome the problems in previous robotic designs and improvements upon them. A robust, modular, and improvable robot pool is aimed to be developed.

The robotic hardware is split into two main parts; motor driver circuit and main board. An integrated FET, Brushless DC (BLDC) motor driver IC is used separately from the main board to reduce noise and interference and to save space. In order to increase control performance and modularity of the design, two main logic units are used in the main board. The DSP chip used in the previous design is replaced with FPGA and MCU chips. Firmware development is done for both chips working simultaneously during the game. Tasks are divided among them according to their strengths. Finally, a PI control algorithm, developed on the FPGA in logic gate level, is discussed.

ÖZET

ROBOCUP SSL ROBOTLARI İÇİN YENİ NESİL GÖMÜLÜ SİSTEM TASARIMI

Uluslararası üniversitelerin robotik donanım ve yazılımları ile yarıştığı RoboCup organizasyonun en rekabetçi ve zorlu liglerinden biri olan RoboCup SSL'de takımlar dört adet heryönlü tekerlek ile hareket eden robotlar ile futbol karşılaşması yapmaktadırlar. Bu ligde, insansı liglerin aksine, robot dengesi sorun olmadığından, takımlar çok daha hızlı robotlar tasarlayabilir ve yapay zeka üzerine yoğunlaşabilirler. Yapay zeka gelişiminin yüksek olmasına karşın, robotların yapay zeka tarafından gelen emirleri başarılı bir şekilde uygulayabilmesi zaruridir. Bu tez çalışmasında yeni nesil robotların gömülü sistem tasarımı ve uygulaması yapılmıştır. Gömülü sistem tasarımı donanım, alt seviye yazılım ve kontrol algoritması geliştirme kısımlarından oluşmaktadır. Turnuvanın sürekli değişen ve gelişen yapısına uyum sağlayabilmesi için tümüyle birimsel bir tasarım amaçlanmıştır. Bu tez geçmiş nesillerdeki problemlerin çözümü ve robotların iyileştirilmesi odaklıdır. Gürbüz, birimsel ve geliştirilebilir bir robot havuzu oluşturulması amaçlanmıştır.

Robotların donanımı iki ana birimden oluşmaktadır: ana kart ve motor sürücü kartları. Tümleşik FET BLDC motor sürücü yongası ile tasarlanan motor sürücü kartları ana karttan ayrılarak elektriksel gürültü ve girişimler azaltılmış ayrıca robotun alan kullanımı daha verimli hale getirilmiştir. Robotların kontrol performansını artırmak ve sistemin ilerleyen gelişimlere açılması amacıyla ana kart üzerinde iki farklı sayısal birim kullanılmıştır. Geçmiş nesillerde kullanılan DSP yongalarının yerine kullanılan FPGA ve mikrodenetleyici yongaları ile bunların güçlü yönleri birleştirilmiş, maç süresince ortak çalışmaları sağlanmıştır. Son olarak FPGA üzerinde kontrol algoritmalarının sayısal kapılar seviyesinde tasarımı işlenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$e(t)$	Control Variable Error
e_A	BEMF of Motor A Phase
e_B	BEMF of Motor B Phase
e_C	BEMF of Motor C Phase
e_q	Quantization Error
f_{PI}	PI Controller Clock Frequency
f_{SYS}	FPGA System Clock Frequency
G_C	Step Response of the Controller
i_A	Current of Motor A Phase
i_B	Current of Motor B Phase
i_C	Current of Motor C Phase
I_{out}	Linear Regulator Output Current
K_I	Coefficient of Integral Term
K_P	Coefficient of Proportional Term
P_{Loss}	Power Loss on Linear Regulator
$r(t)$	Reference Signal
T_{PI}	PI Controller Loop Period
$u(t)$	Controller Output
V_A	Voltage of Motor A Phase
V_B	Voltage of Motor B Phase
V_C	Voltage of Motor C Phase
V_F	Floating Node Voltage
V_{in}	Linear Regulator Input Voltage
V_M	Motor Supply Voltage
V_n	Voltage of Motor Center Tap
V_{out}	Linear Regulator Output Voltage
$y(t)$	Control Variable

LIST OF ACRONYMS/ABBREVIATIONS

3D	Three Dimensional
ADC	Analog to Digital Converter
AI	Artificial Intelligence
BCD	Binary Coded Decimal
BEMF	Back Electro-Motive Force
BGA	Ball Grid Array
BLDC	Brushless Direct Current
BLDCM	Brushless Direct Current Motor
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
D	Derivative
DIP	Dual In-Line Package
DC	Direct Current
DMOS	Double-Diffused Metal-Oxide-Semiconductor
DSC	Digital Signal Controller
DSP	Digital Signal Processor
EMF	Electro-Motive Force
ESD	Electrostatic Discharge
FC	Ferrite - Capacitor
FET	Field Effect Transistor
FIFA	International Federation of Football Associacion
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPIO	General Purpose Input Output
HF	High - Frequency
I	Integral
I/O	Input/Output
I^2C	Inter-Integrated Circuit

IC	Integrated Circuit
ID	Identity
IPD	Initial Position Detection
ISD	Initial Speed Detection
ISM	Industrial Scientific Medical
LED	Light-Emitting Diode
Li-Po	Lithium-Polymer
LQFP	Low Profile Quad Flat Pack
MCU	Microcontroller Unit
MEMS	Micro Electro-Mechanical Systems
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
MSB	Most Significant Bit
NECS	Networked and Embedded Control Systems (Laboratory)
NMOS	N-Channel Metal-Oxide-Semiconductor
NVIC	Nested Vectored Interrupt Controller
P	Proportional
PCB	Printed Circuit Board
PD	Proportional - Derivative
PI	Proportional - Integral
PID	Proportional - Integral - Derivative
PLL	Phase Lock Loop
PMOS	P-Channel Metal-Oxide-Semiconductor
PWM	Pulse Width Modulation
RAM	Random Access Memory
RC	Resistor - Capacitor
RFI	Radio-Frequency Interference
RPM	Revolutions per Minute
SPI	Serial Peripheral Interface
SSL	Small Size League
TVS	Transient Voltage Suppressor
UART	Universal Asynchronous Receiver Transmitter

USART	Universal Synchronous Asynchronous Receiver Transmitter
VCO	Voltage Controlled Oscillator
VHDL	Very High-Speed Integrated Circuit Hardware Description Language
XOR	Exclusive - OR

1. INTRODUCTION

Robotics technology has been of great importance in the industry for decades and still continues to evolve with increasing speed. Facilitating robotics and AI in manufacturing has brought many industries to an advanced level. In recent years, we have been experiencing the state of art robotics and AI technologies in our daily life. From autonomous call centers and personal assistance in mobile phones to autonomous cars and even space travel, we are more and more closely interacting with the robotics technology and artificial intelligence. RoboCup has been contributing to this area for twenty years. With its subdivisions, many real life problems and robotics challenges have been overcome throughout the years. Since the first RoboCup event in 1997, the competition has evolved to a multidisciplinary organization with four main leagues and a RoboCup Junior league. These consist of RoboCup Soccer, RoboCup Rescue, RoboCup @Home, and RoboCup Industrial. Among the four leagues, RoboCup Soccer is the main interest of the tournament.

The main objective of the event is to win an official soccer game against the most recent FIFA World Cup winner with humanoid robots by 2050 [1]. Even though this goal can be thought as a far-fetched dream, RoboCup and its participants are motivated by the great developments in human history. As it took fifty years to complete Apollo mission, and it took fifty years from the first computer to Deep Blue to beat the World Chess Champion, researchers believe that with focused effort this goal can be achieved [1]. RoboCup Soccer is often taken lightly as it is seen as a leisure activity. However, the highly dynamic environment of soccer creates many difficulties which replicate a number of real life problems to be solved. For instance, in Humanoid League and Standard Platform League robots have arms and legs similar to human body form and the main focus is to keep the robot in balance while executing difficult tasks. This challenge represents the problems faced in walk assistance robots and exoskeleton studies for disabled people. In the middle size league, a fast machine vision development is crucial since all robots and the ball on the field move very fast,

therefore findings in this league contribute to localization and autonomous path finding studies.

In the SSL, which is one of the oldest divisions of RoboCup, teams develop their own robots according to size, shape, and feature limitations. Robots then autonomously play a soccer game by teams of six with most of the FIFA rules. In contrary to humanoid leagues, it is relatively a trivial challenge to balance and move the robots and kick the ball in the Small Size League; therefore, the competition has mostly focused on AI development. The main area of interest in the SSL is intelligent multi-agent consensus and centralized control in dynamic environments [2].

The objective of this thesis is to build a new generation of robots that can compete in RoboCup SSL competitions. Due to some design flaws, the current version of electronic boards on the existing robots limits the team's performance in the competitions. The aim of this thesis is to eliminate these problems in the previous designs and develop modular/robust electronic units that will provide not only satisfactory performance but also ease of implementation.

1.1. Robocup SSL

RoboCup SSL is a subdivision of the international tournament RoboCup where teams from universities compete in different fields with various concepts. In the RoboCup SSL, two teams of six (the number is planned to be increased to eight in 2018 and eleven in 2019 or 2020) autonomous robots compete in a soccer game in which most of the rules of a regular game are applied [3].

RoboCup SSL robots are designed and built by teams independently; however, the robots must conform to some physical limitations such as dimensions, maximum travel and kick speeds, and so on. Each team designs its own artificial intelligence module running on a main computer which sends the necessary information to the robots using wireless communication during the game.

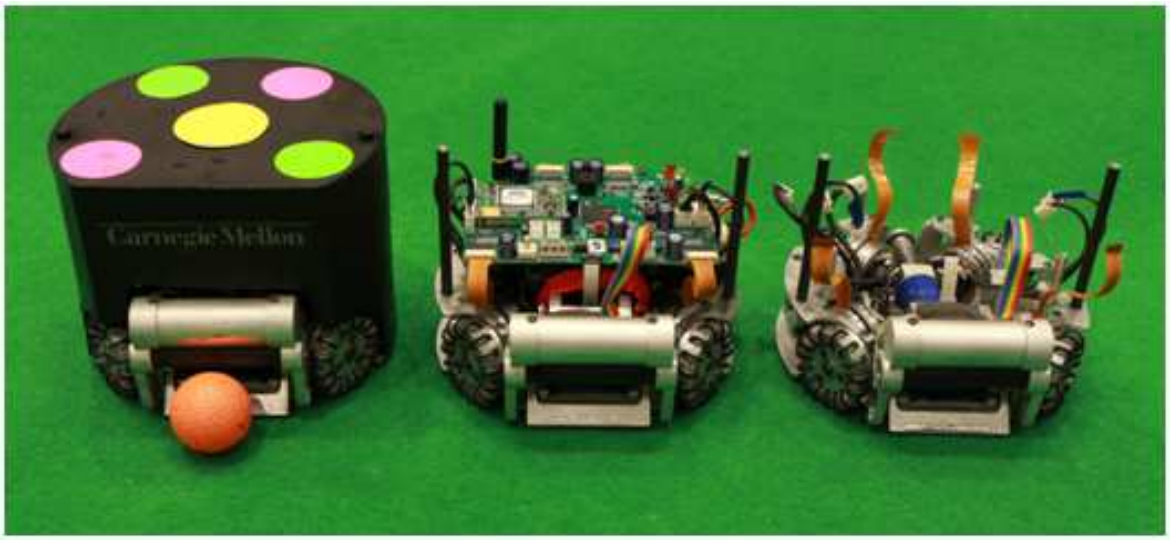

Figure 1.1. RoboCup SSL robots.

As depicted in Figure 1.1 [3, 4], the SSL robots have colored patches on top of them. This is the standard pattern by which teams and robot IDs can be recognized from the centralized vision system. The robots must fit in a cylinder of 180 mm diameter and 150 mm height described in F180 rules [5].

1.1.1. Game Setup

A standard RoboCup SSL game is played in a rectangular field as shown in Figure 1.2 [6] with a standard orange colored golf ball. The field is made of 6 meters wide and 9 meters long green carpet or mat with 70 cm extension to the game area in each direction. The extension is divided into two parts with a wooden bar, in order to create a safe zone for referees and to allow robots to leave the field while keeping the ball in. 10 mm wide, white lines are used to show boundaries such as defence area, goal boundary, and middle line.

The referees are responsible for applying the rules and giving directions to referee box computer as well as placing the ball on the field whenever game pauses and needs to be restarted in cases like throw-ins or goal kicks. The referee box computer keeps

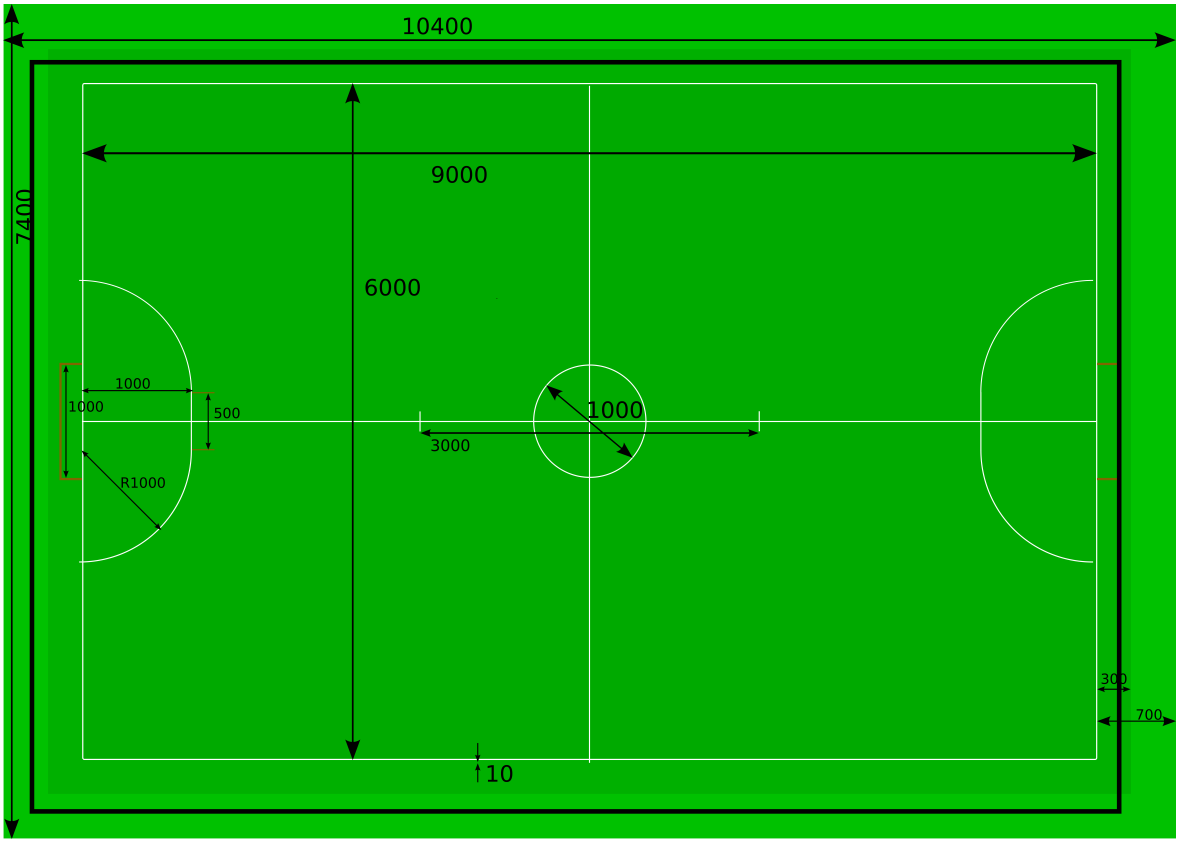

Figure 1.2. The field dimensions (mm).

the record of the game and transmit signals to the teams' AI computers. These signals include, game start and end, direct and indirect kicks, emergency stop (halt), and so on. Since 2016, an auto referee software has been used. This software takes the image stream from the central vision system and processes it to decide every rule call during the game. The software is currently being used as a supporting tool for the referees but is planned to be used as the sole decision entity in the near future. As of 2018, auto ball placement will be mandatory for each team in the first division, which will remove the need for human referees eventually.

In order to increase the efforts on robotics and AI development as well as to reduce the effort to setup cameras during the tournament for each team, a global vision system has been used since 2010. This system was developed collaboratively by the participating teams and published as an open-source tool for every team in the league. This global vision system shown in Figures 1.3 [5] and 1.4 [7] facilitates four

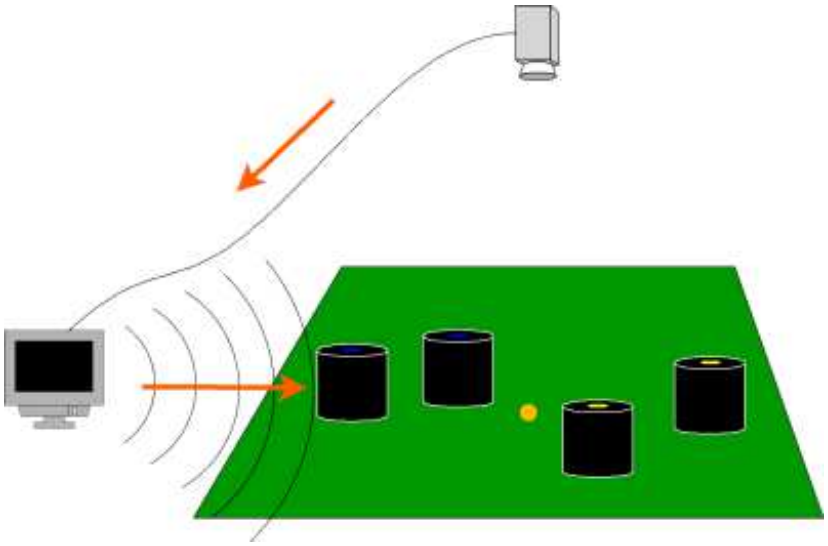

Figure 1.3. RoboCup SSL global vision system dataflow.

cameras above the field and processes the obtained image. After processing the image, the positions and orientations of all robots and the position of the ball are transmitted to AI computers. The image data is also used to determine ball speed, the last robot to touch the ball, and so on, in order to feed the referee box for decision making.

Figure 1.5 shows the standard pattern used in the SSL. The pattern consists of five circular color marks on top of each robot. The middle color mark, either blue or yellow, indicates the team each robot belongs to. The remaining marks, either green or magenta, provide the basis for the robot ID information to the global vision system. This data is also sent to AI computers so that teams can easily keep track of their robots and assign duties to them.

1.1.2. Hardware Requirements

The robots in RoboCup SSL must conform to several physical limitations, also known as F180 Rules [6]. These include robot shape, wheels, standard pattern, maximum speed, and so on. The robots must fit in a cylinder of 180 mm diameter and 150 mm height. Their top surface must carry the standard pattern shown in Figure 1.5. The robots must have wheels that will not damage the game field. Velcro and metal

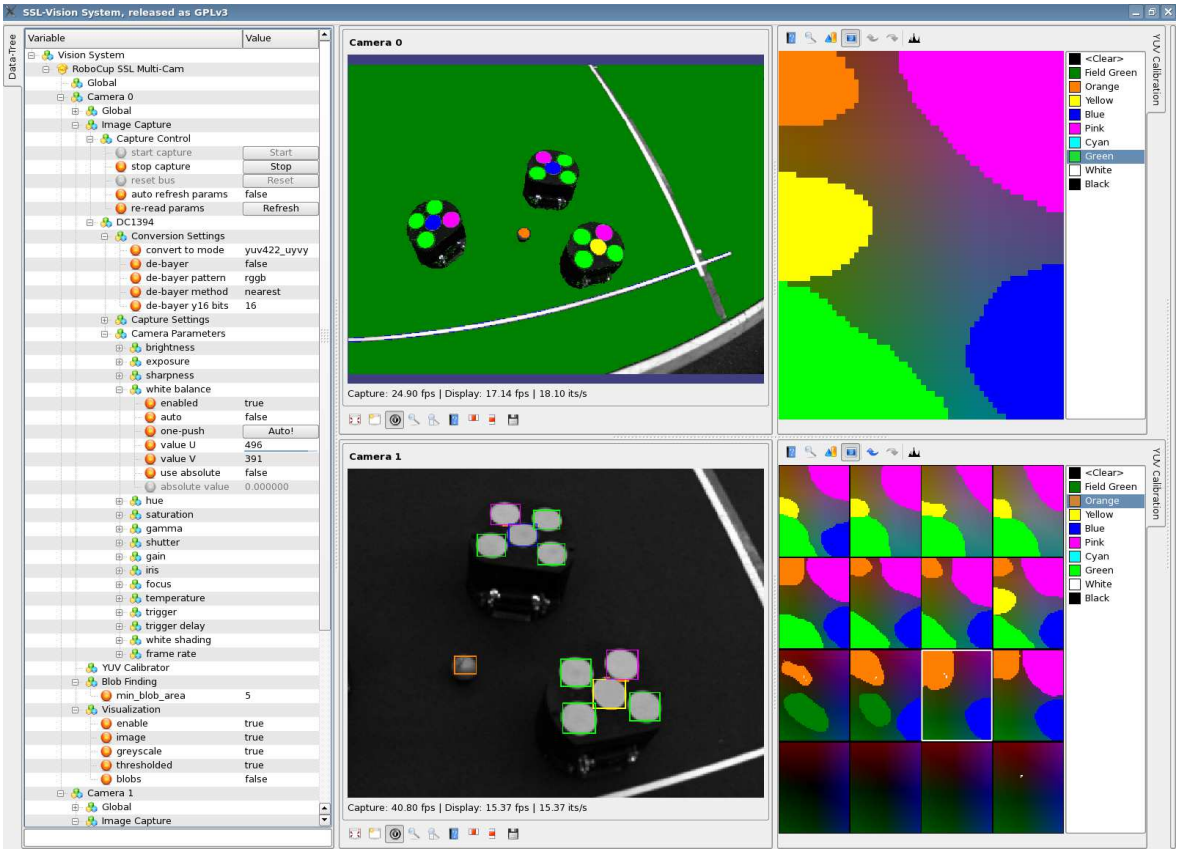

Figure 1.4. RoboCup SSL global vision system client.

spikes are strictly forbidden to be used in wheels. There are also speed limits for the robots and the ball. During a game, while the ball is out of play the robots cannot exceed the speed limit of 1.5 m/s and kicks above 8 m/s speeds are not allowed to score a goal. Moreover, high speed ball kick that can damage the other robots is not allowed. All teams must add a collision avoidance algorithm in their AI software such that any collision committed by the faster robot is considered as foul.

All these rules bring the need for a well designed AI software as well as a robust robot design that can follow the orders from the main computer. In order to make a robot compatible with the rules, there are several design considerations for the hardware design. The robots must have a precise speed control over the locomotive motors. This requires a design which is immune to noise and crosstalk. The global vision system sends data with a rate of 60 frames per second. This creates a need for a robust communication system such that the robots do not miss any data package due to the

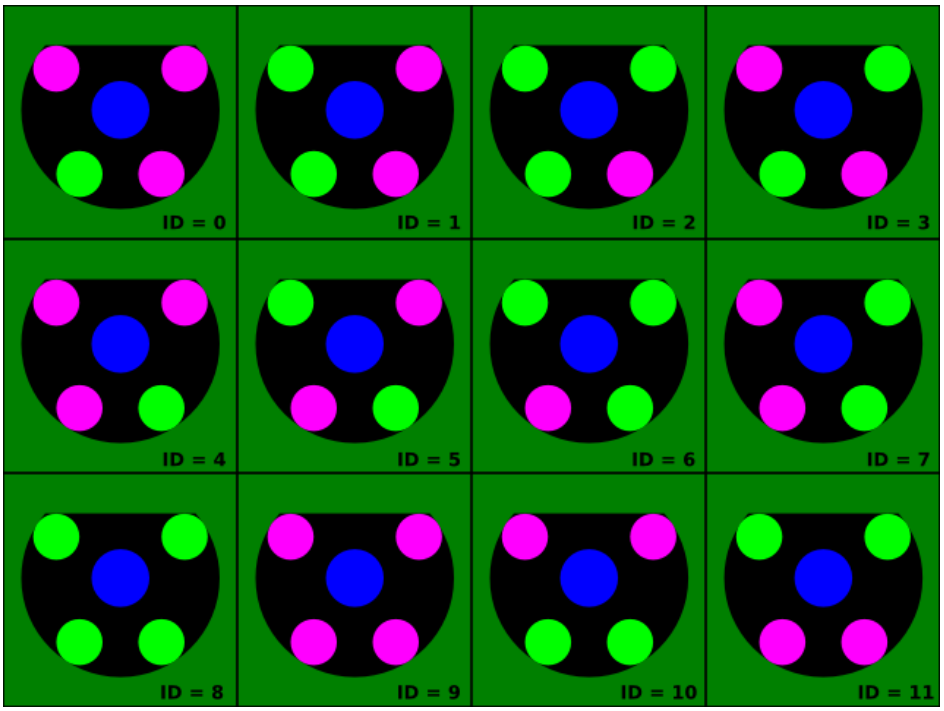

Figure 1.5. RoboCup SSL standard pattern.

fact that the fast pace of the game can cause in collisions and reduced performance. Another essential design consideration is that the control algorithms must be very fast in order to complete their cycles several times to reach steady state before another data package arrives in 16 ms. Finally, the robot design must be modular and open to improvements because the rules of the tournament are constantly evolving. A modular design that can adapt to these changes easily is essential for teams so that the robot pool would not be required to be redesigned from scratch.

1.2. Literature Review

In the NECS lab, two main generations of robots were designed previously. While these designs have enabled our team to qualify and compete in various tournaments, they have some electrical design flaws that have led the robots fall behind other teams even though the main AI software is sufficient to compete in the tournament. Below, these former designs will be explained part by part also in comparison to rival team solutions.

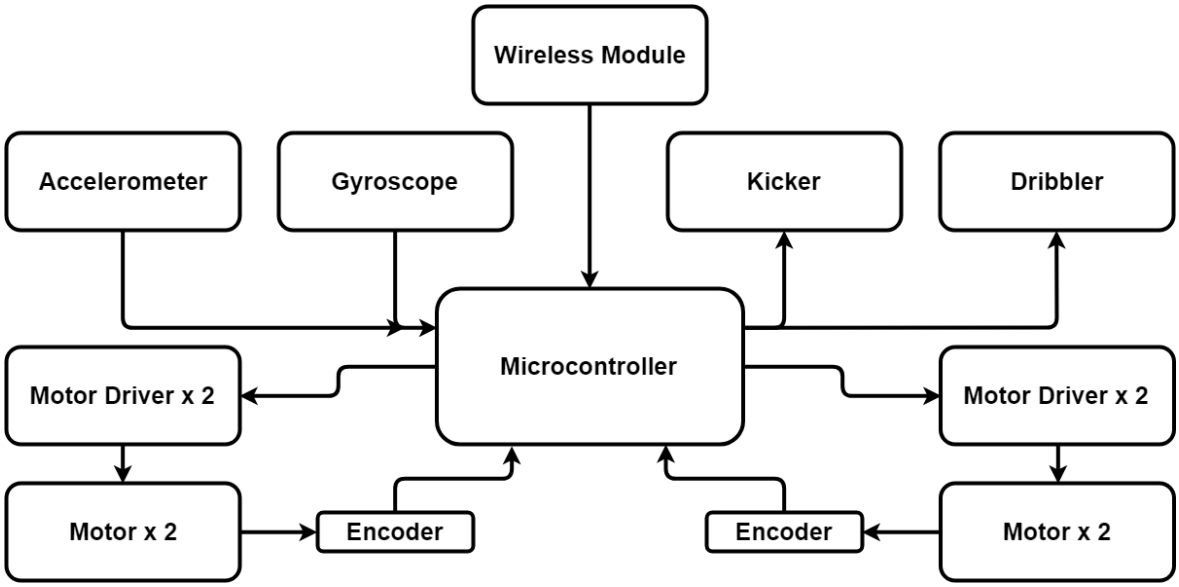

Figure 1.6. Previous main board design block diagram.

The main board is one of the two most important elements of the robots. Our previous designs use one single board to handle data processing and motor drive operations. This method is used by most of the teams in the tournament. Figure 1.6 shows the block diagram of BRocks (NECS Lab RoboCup SSL Team) main board [8].

This method raises the problems of motor driver noise in the processing unit and problematic driver failures. Due to the dynamic nature of the game, it is not uncommon to have driver circuit failures such as burning of MOSFETs or motor drive controller ICs. In such a case, it becomes necessary to repair or replace the main board which is difficult, time consuming, and costly. This also brings a competitive disadvantage during the tournaments. This problem is overcome by designing daughter boards by several teams. The Iranian team MRL uses such a design approach as shown in Figure 1.7 [9].

Our team developed and implemented the idea of separating the driver and processing boards in the competition of 2016 as depicted in Figure 1.8 [10]. This design did not resolve the issues with the noise while also raising a size problem. The reasons for failure are the common power transmission in both boards and unprotected power

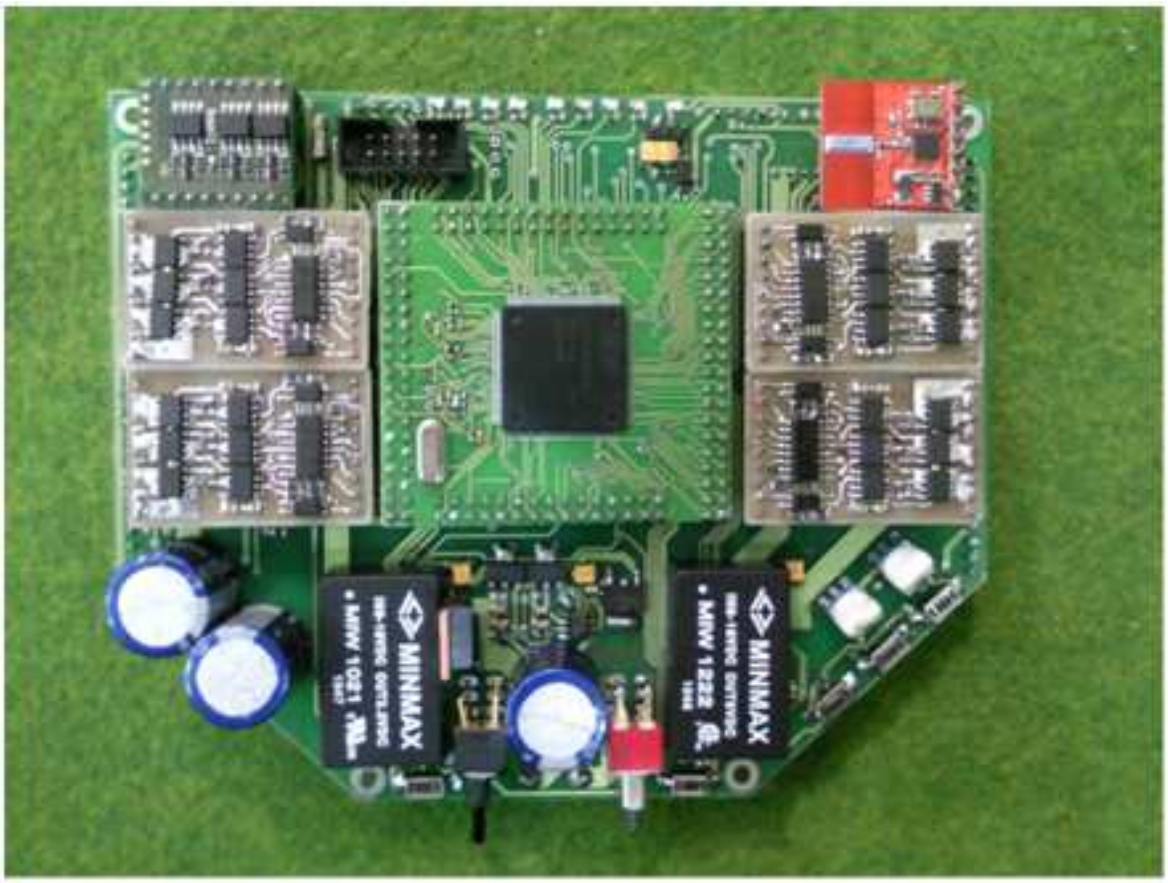

Figure 1.7. MRL Team main board and daughter boards.

and signal lines on the PCB.

Apart from the PCB design, the design methods for BLDC motor drive also differ among teams. The two main approaches rely on the use of BLDC Drive Control ICs and Direct Drive. The Boğaziçi University team BRocks have used MC33035 motor driver IC in both former designs [8]. Since BLDC motors have three phases which are switched in a sequence, 15 PWM channels are needed to drive the motors directly which is unfeasible for most MCU or DSP units. Teams, who use FPGA as their processing unit, can use the direct drive method by using power MOSFETs and controlling them with the help of FPGA's parallel processing power. In this method, the Hall sensor and encoder data are fed into the FPGA to determine the motor's position and six power MOSFETs for each motor are switched in order to control stator currents [11].

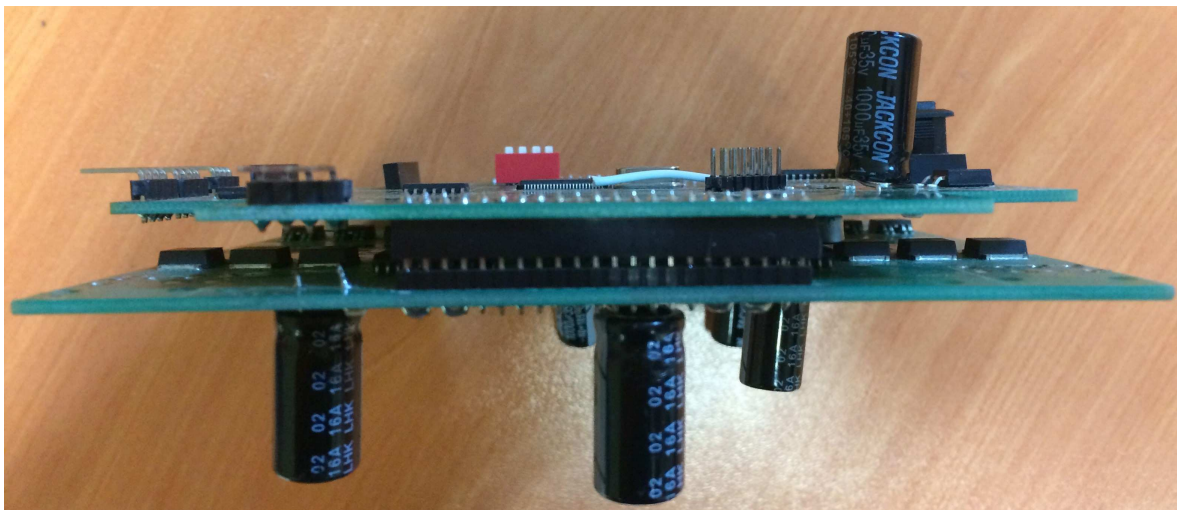

Figure 1.8. BRocks second generation main board.

Another point of design where teams differ is the processing unit. While some teams use FPGAs with parallel processing capability, others use MCUs or DSPs with FPU processing capability. However, the state of the art is evolving into using both chips on one board in order to combine their powerful aspects.

Running control algorithms on FPGA has some advantages and disadvantages. As multiplication, use of non-integer values, and mixed signed operations are very difficult to implement at logic gate level without dedicated hardware, the size of the circuit increases and makes it difficult to find chips with reasonable size and pricing. This nature of the FPGA chips elongates the algorithm design process dramatically. However, once the algorithm is implemented successfully, it makes it possible to run P, I, and D blocks of the PID controller in parallel [12]. Moreover, the results will be as close as possible to simulated results with high performance digital tools such as Matlab and Simulink [13].

In the light of these, the system in Figure 1.9 is proposed. In this thesis, hardware design of the motor driver and main board circuits as well as firmware development for FPGA and MCU chips will be explained in detail.

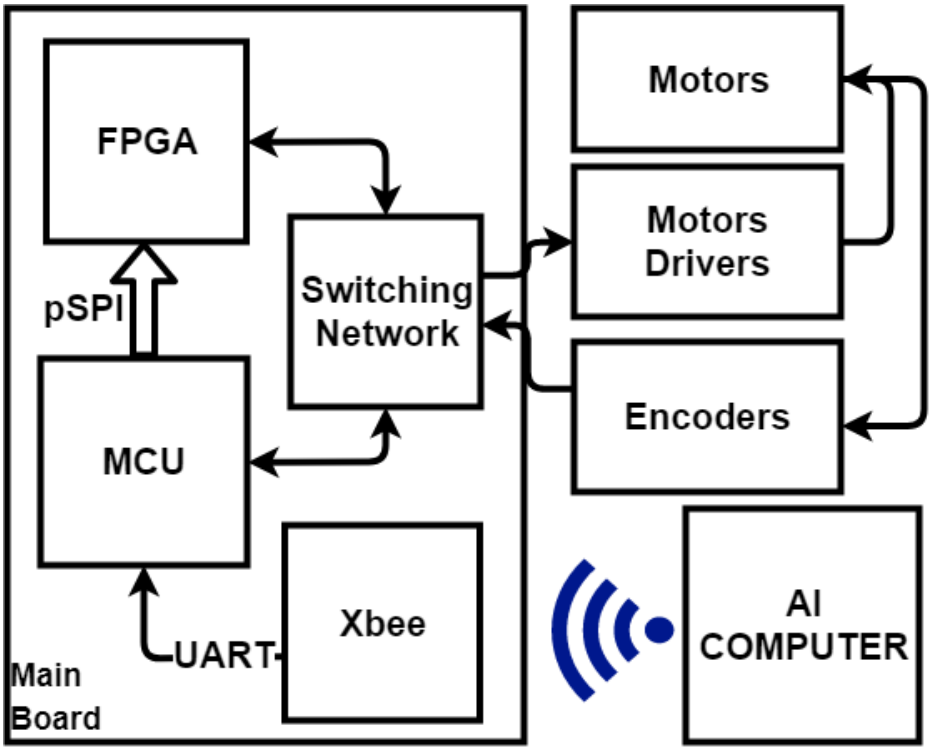

Figure 1.9. Proposed system's simplified block diagram.

1.3. The Organization of the Thesis

This thesis consists of six chapters. The first chapter, Introduction, gives elementary background information about the RoboCup SSL and its requirements as well as a review of state of the art in the competition.

Chapter 2 gives detailed information on the motor driver circuit design and improvements upon previous circuit designs

In Chapter 3, the Main Board design is explained. This chapter includes logic units, power regulation, communication, and other design features.

Chapter 4 covers the low level firmware design. This chapter can be divided into two main parts: MCU firmware and FPGA firmware. The main algorithms are explained in the chapter.

Chapter 5 is dedicated to the low level control algorithm implemented on the FPGA chip. The chapter includes low level design, control algorithm, and test results.

Finally, in Chapter 6, the thesis is concluded with a summary and possible future improvements.

2. MOTOR DRIVER

Motor drive is the most crucial part of the design for RoboCup SSL robots. Motors are very noisy components and yet their performance is vulnerable to external noise. The noise generated from the motors can interfere with logical signals in the main board and, more importantly, the reference analog signal that controls the motor speed. Moreover, in some cases such as sudden loads (e.g. an object stuck to the wheel) or brake, huge voltage spikes are generated from the motor and travel to the power supply or the controller circuitry. These spikes are usually around five - ten times of the supply voltage which can damage the circuit permanently. These problematic issues make it of great importance to have a good design for motor driver circuitry.

In this chapter, a general overview on BLDC motors and drive techniques will be given. In Section 2.2, the previous motor driver designs in our lab will be examined and their design flaws will be stated. Finally, design of the new generation motor driver circuit will be demonstrated followed by conclusion of the chapter.

2.1. BLDC Motors

In our robots we use "Maxon EC 45 Flat" BLDC motors which have 50 W power rating. DC motors have great advantage on high and dynamic speed applications due to their linear torque vs. speed relationship [14] and very high maximum RPM values. BLDC motors are better off compared to brushed DC motors due to several advantages. First of all, removal of the brush in the motor increases the durability of the motor while reducing the need for maintenance. The brushes also have tendency to create arcs which in turn creates RFI. Moreover, BLDC motors have higher speed and torque performance in relatively smaller sizes. The main disadvantage of BLDC motors, on the other hand, is the difficulty of speed control due to the required voltage switching among phases [15]. Figure 2.1 shows the equivalent circuit for a BLDCM as well as its driver circuit [16].

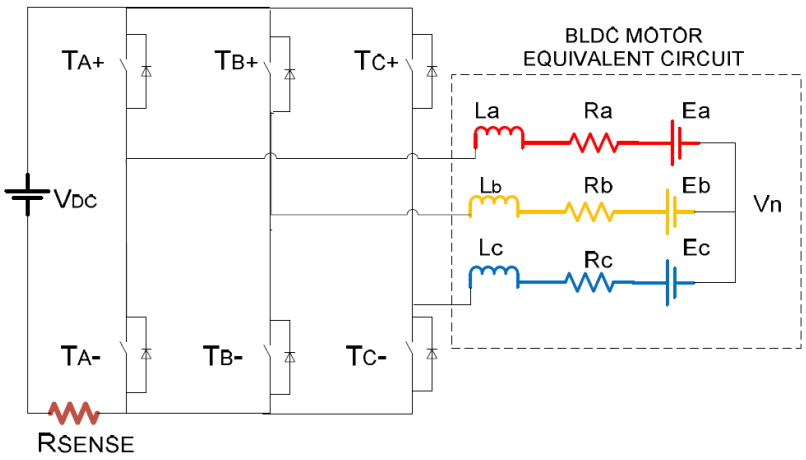

Figure 2.1. Brushless DC motor equivalent circuit with inverter drive.

2.1.1. Basic Principles

Brushless DC motors are based on brushed DC motors which have permanent magnets (electromagnets in high torque applications) in the stator and windings in the rotor as shown in Figure 2.2 [17]. The need for brush emerges from two main reasons. Firstly, the current direction must be altered in each half of the rotation. This can be achieved by connecting alternating ends of the coil to the brushes while the motor rotates. The second reason is the fact that the windings are revolving; therefore, power cables cannot be directly connected to the wound cables. This type of motors have many disadvantages stated in Section 2.1. A BLDC motor has the same main principle but the placement of the core components are inside out. The windings are in the stator and the permanent magnets are in the rotor. The duty of brushes are achieved by inverter circuits (mostly solid state) which alternates the current direction in each cycle by external switching control [18].

2.1.2. Drive Methods

Stator windings of brushless DC motors must be switched consecutively with precise timing since the rotor magnet can only be attracted to a single coil for continuous operation. Otherwise the motor can stall, oscillate, or revolve changeably. Therefore it

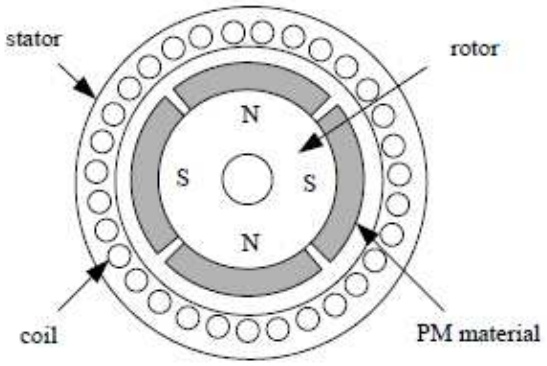

Figure 2.2. Brushless DC motor cross-section.

is essential to know the position of the rotor magnet with respect to the stator windings. Only then a correct operation of the BLDC motor can be achieved by applying a correct sequence of phase switching. This requires a dedicated hardware to decide the rotor position and consecutively switch the coil currents. There are two main methods for rotor position detection which are based on back electro-motive force (BEMF) or magnetic position sensors.

Position-sensorless BLDC rotor position detection method facilitates back EMF generated on the stator coils since the magnetic rotor creates a varying BEMF on each winding with a 120° phase difference. Equation (2.1) gives the voltage of phase coils (V) in terms of phase current (i), winding resistance (R) and inductance (L), BEMF (e), and neutral node (center tap) voltage (V_n) [19]:

$$\begin{aligned} V_A &= Ri_A + L\frac{di_A}{dt} + e_A + V_n \\ V_B &= Ri_B + L\frac{di_B}{dt} + e_B + V_n \\ V_C &= Ri_C + L\frac{di_C}{dt} + e_C + V_n \end{aligned} \tag{2.1}$$

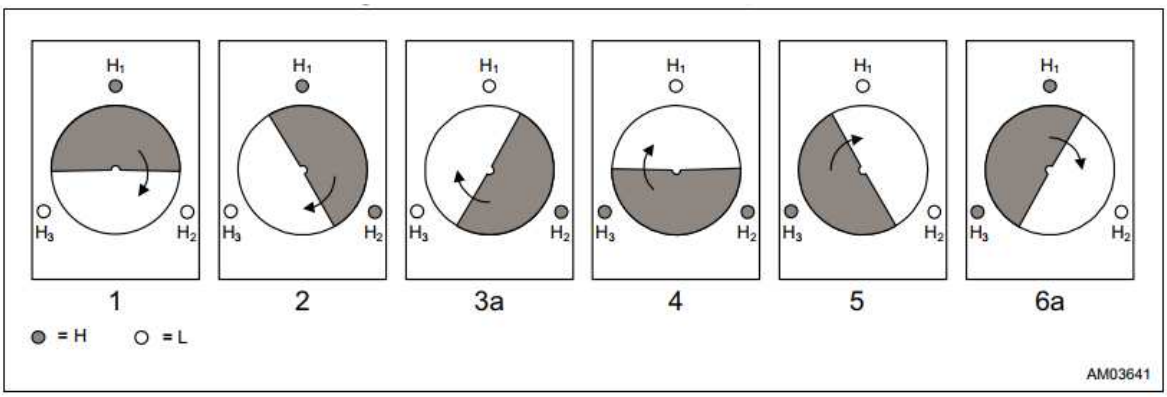

Figure 2.3. Hall sensor sequence.

given that in any position of the rotor, two coils are inversely polarized and one coil is left floating. The following equation for floating node voltage in BEMF zero crossing point is derived by Kirchhoff's Laws with the assumption of identical phase resistances

$$V_F = \frac{V_M}{2} \tag{2.2}$$

where V_F is the floating node voltage and V_M is the motor supply voltage.

With this method, phase voltages can be measured in order to determine the position of the rotors so that a correct switching sequence can be started by the controller logic. However, in order to detect zero crossing, it is necessary to have an initial rotation of the motor. This can be achieved by starting the sequence from a random point and repeating it until a zero crossing measurement is received, which reduces the start-up performance of the motor.

Another method to detect rotor position is to use magnetic position sensors, most commonly Hall effect sensors. Even though this method increases motor size and price, it is more robust compared to sensorless detection and does not have the problem of initial measurement as stated above. In this method, three Hall sensors are placed next to each stator winding to determine the rotor position. The sensor reading is mostly digital which eases the driver circuit design dramatically.

Figure 2.3 shows the valid (possible) Hall sensor reading combinations [20]. With these three bits of information, the controller can decide the best start-up switch positions at initial start case.

2.2. Previous Designs

In our lab, there are two previous versions of electrical circuitry for the robots. Both designs have the same motor driver configuration, while their positioning with respect to the main board is different. These designs are based on ON Semiconductor MC33035 Brushless DC Motor Controller IC [8]. In this section, this motor driver circuit design will be examined in terms of design considerations. Then, a failed design method, which we used in this thesis work, will be explained with its reasons to failure.

2.2.1. Former Generations' Motor Driver Designs

MC33035 Brushless DC Motor (BLDCM) Driver IC is a motor driver controller, which does not have integrated power output stage. Therefore, the designer must add three PMOS - NMOS transistor pairs as phase inverters at the output stage. These transistors must be able to carry high currents under high voltages, leading them to be bulky components. This feature creates a sizing problem for the driver circuit. The Ic requires two analog voltage inputs to operate. These are used for speed control and current limitation [21]. These analog signals are highly susceptible to motor noise and must be protected thoroughly.

Figure 2.4 shows the schematics for one motor's driver circuitry [8]. Notice that, all signals share the same ground reference point. This unprotected grounding scheme requires delicate noise handling in the PCB design level. However, as shown in Figure 2.5 in both generations of the circuit, there are some grounding problems which were tried to be solved by additional ground wirings and removing some traces [8,10]. These problems are mainly caused by several reasons. First, both circuits do not have ground planes which prevents some return path signals, creating electromagnetic noise in turn

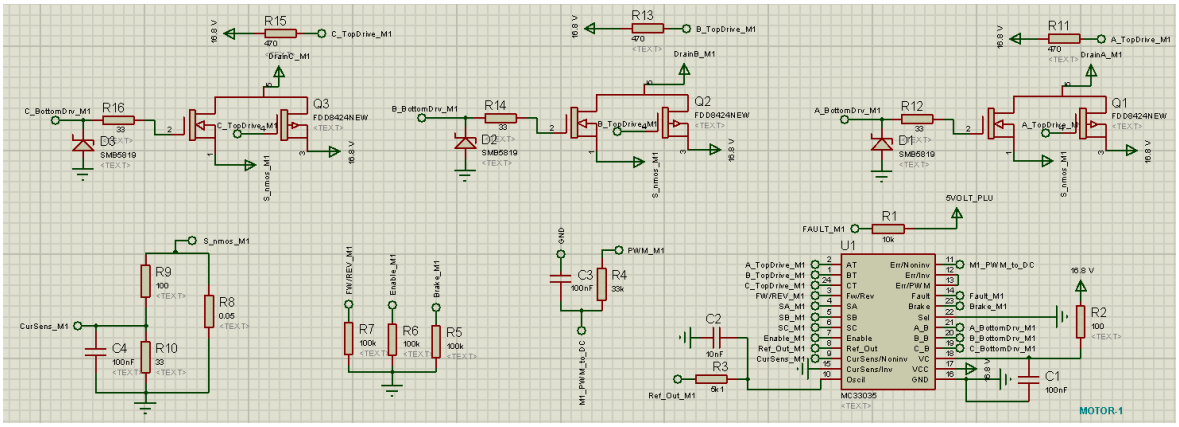

Figure 2.4. Circuit diagram for motor drive from previous generations.

due to return path loops. Second, critical ground traces (power lines and analog lines) are overlapping thus the noisy motor power signals affect analog signal dramatically. Lastly, ground signals do not have a common connection point in the PCB, which leads to ground reference differences on the same IC.

The first generation had problems with crosstalk among motors due to bad grounding practice. Since there is no isolation between motor drivers, it was a frequent problem that as one motor rotated, the others started to move undesirably. Although the control performance was very good in this design with one wheel, this crosstalk created performance issues during high-paced games.

In the second generation, this problem was attempted to be solved by separating power components (motor drivers) and logic components (DSP, XBee, and small power ICs) into two boards that connect to each other as shown in Figure 1.8. However, this change did not solve the crosstalk and noise problems, and the tracing error on the power PCB caused even a bigger failure mode. Since the motor power ground passed through over-current sense input pin, the motor could not reach even medium speeds. As the motor sped up, the noise level on the ground trace of the sense pin could go upto 200 mV which is two times the overcurrent threshold voltage of the sense pin. Therefore, in mid-range speeds, the noise level on this pin increased above the threshold level, which stops the operation of the MC33035 IC.

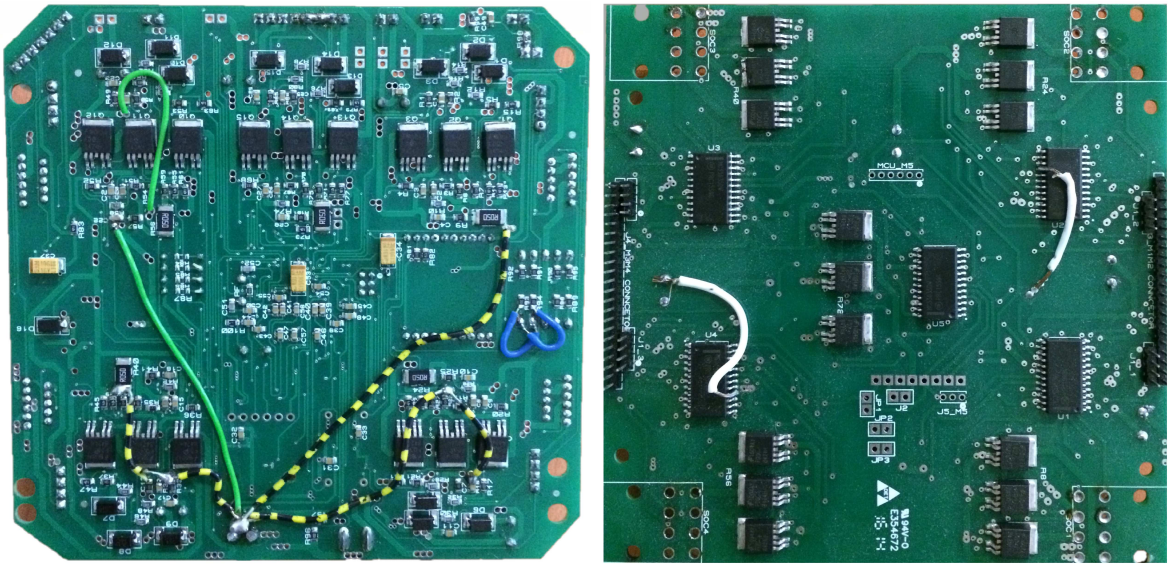

Figure 2.5. Previous PCBs and additional ground wirings.

2.2.2. Failed Design Attempt

As the circuits in the old generations had problems with size and electrical noise, a new circuit design was necessary to solve these issues. For this purpose, a low cost, low size, and low external component count BLDC motor driver IC was selected from Texas Instrument and its design was implemented on PCB. The TI DRV10983 driver IC has integrated power MOSFETs and can drive motors upto 2 A of current rating. Additional to MC30355, it can also be driven by digital logic thanks to its configuration and speed registers [22]. This feature could have solved the problem with motor noise interference in the analog speed control signal.

However, DRV10983 uses sensorless speed control as was explained in Section 2.1.2. This type of control has startup performance issues due to the necessity of initial movement. Since it is expected from the IC to obtain correct switching sequence under one revolution, which would have a negligible effect on robots' performance, DRV10983 needed high speed continuous revolution to be able to control motor speed. Internal closed loop control system of the IC required a threshold of hundreds of revolutions per minute to gain control of the system according to our tests. This performance was not acceptable for our application since it would create significant instability of

the robots at low speeds. Therefore this design became obsolete and a new method is implemented as it will be explained in the next section.

2.3. New Motor Driver Circuit

In the previous section, a number of design failures and possible improvement points were discussed. In the light of this knowledge, a new method of motor drive is proposed and implemented. This design should have small size, easy maintenance during games, low noise effect on its control signals and low noise effect on the main board. This section discusses the design steps and considerations in detail.

2.3.1. BLDCM Driver IC

Based on our previous experiences on the topic, a correct selection of the IC is essential for the circuit's performance. Therefore, several critical features are to be existing on the selected IC. These can be listed as, low total area on the PCB (including the sufficient external components for the IC), high current capability (2.8 A for our existing motors), correct drive method (Hall sensor tracking), and system compatible control logic. ST L6235PD Brushless DC Motor Driver covers all the needed features. Moreover, it separates DMOS power circuitry from the CMOS logic enabling it to handle high currents while reducing the noise effect on the logic pins. Current sense circuitry is also placed on the power side of the circuit which removes the problems with the ground reference differences on overcurrent detection. Figure 2.6 shows the typical application circuit of the IC [20].

2.3.2. Circuit Design

In the new motor driver circuit design, the typical application circuit, shown in Figure 2.6, is followed in general. Several changes and additions are made in order to gain better performance. In the first generation of the motor driver circuit, crosstalk was an important problematic issue. Moreover, the circuit did not have BEMF pro-

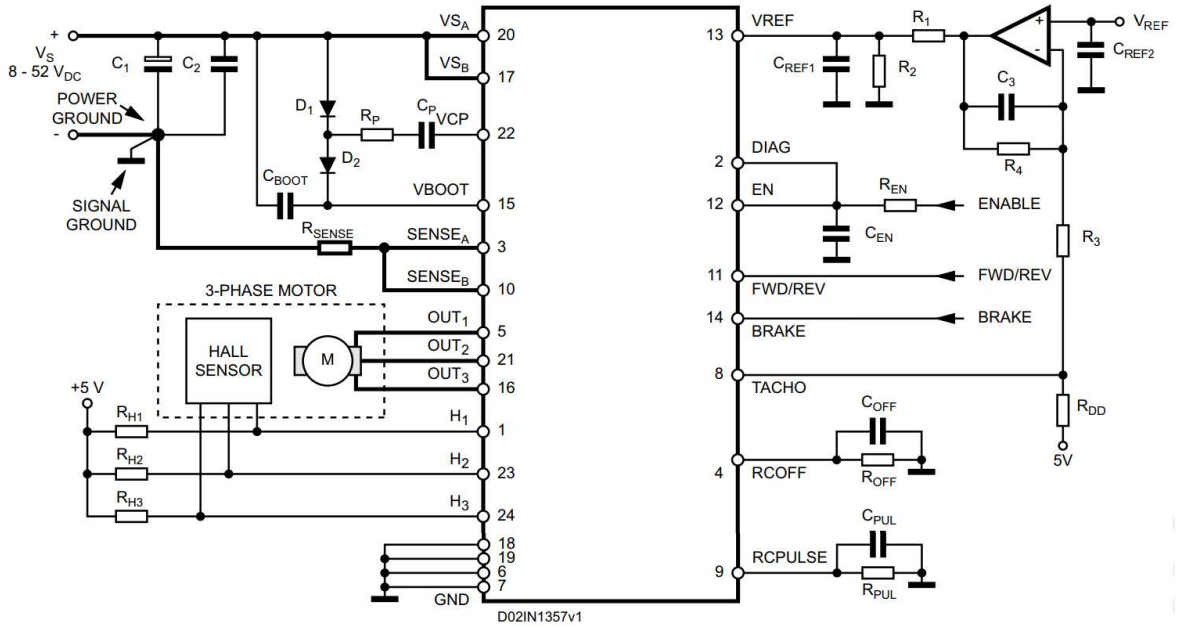

Figure 2.6. STM L6235 typical application circuit.

tection, therefore, if a motor sent a pulse towards the motor driver, this pulse would travel through the main board, causing a number of circuits to break. This type of short-circuit behavior is difficult to handle, especially during competitive games.

In order to solve these problems due to motor noise and BEMF propagation, several measures were taken as depicted in Figure 2.8. First, a series Schottky diode was placed at the power input of the driver circuit, which would stop BEMF pulses coming from the motor while having fast turn-on time and very low forward bias voltage. Then a ferrite bead, followed by 100 μF and 100 nF capacitors, is placed as low pass filter in order to reduce motor noise. Ferrite - capacitor (FC) filter provides better performance compared to RC low pass filter, due to increasing impedance of ferrite as the frequency increases. Moreover, in current carrying lines, resistance in the RC filter must be very low in order to avoid voltage drop due to filter, which increases the capacitance value for the same cutoff frequency. Figure 2.7 shows the performance difference of FC and RC filters with same DC resistance and capacitance values. Ferrite - capacitor filter is followed by a TVS diode in the protection circuit. This component is crucial for the operation of the circuit because when a BEMF pulse travels to the protection circuit,

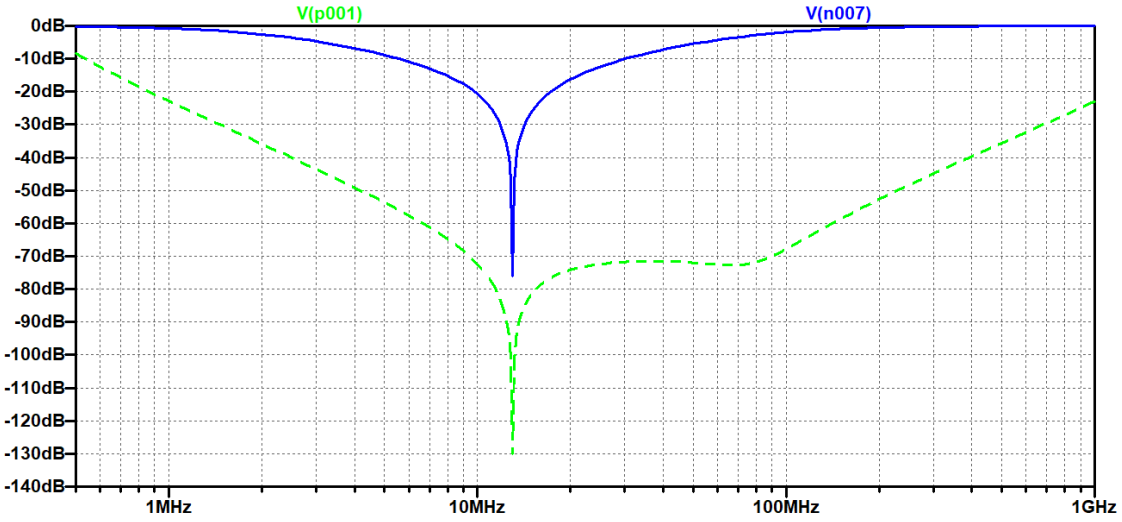

Figure 2.7. FC filter (dashed) vs. RC filter (solid).

it cannot pass through the Schottky diode, therefore input capacitors must suppress this pulse. However, this requires capacitors with voltage ratings at least 120 V. As the voltage rating of the capacitor increases, its size increases dramatically. Therefore, a Transient Voltage Suppressor (TVS) diode becomes necessary to reduce the spikes and capacitor size. TVS diode we used in the application crimps the pulses above 24 V to 38 V, therefore a 50 V rated capacitor is sufficient for proper operation of the circuit. TVS diode has the structure of zener diode with greater zener current and breakdown voltage. They provide high performance ESD and voltage spike performance.

Apart from power input protection, several measures were taken to reduce noise on signal lines and main board. First of all, each signal line to the main board is filtered

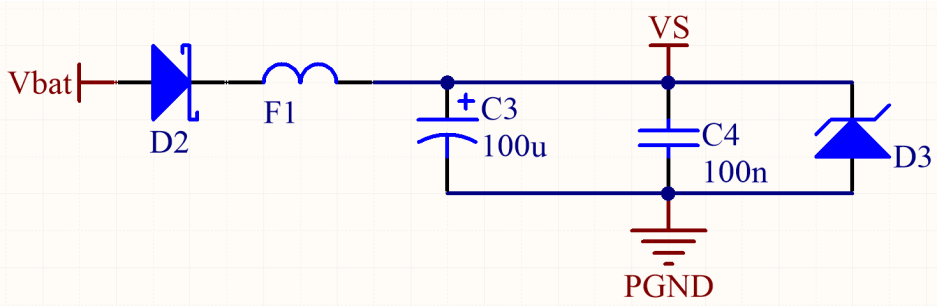

Figure 2.8. Input protection circuit for the motor driver.

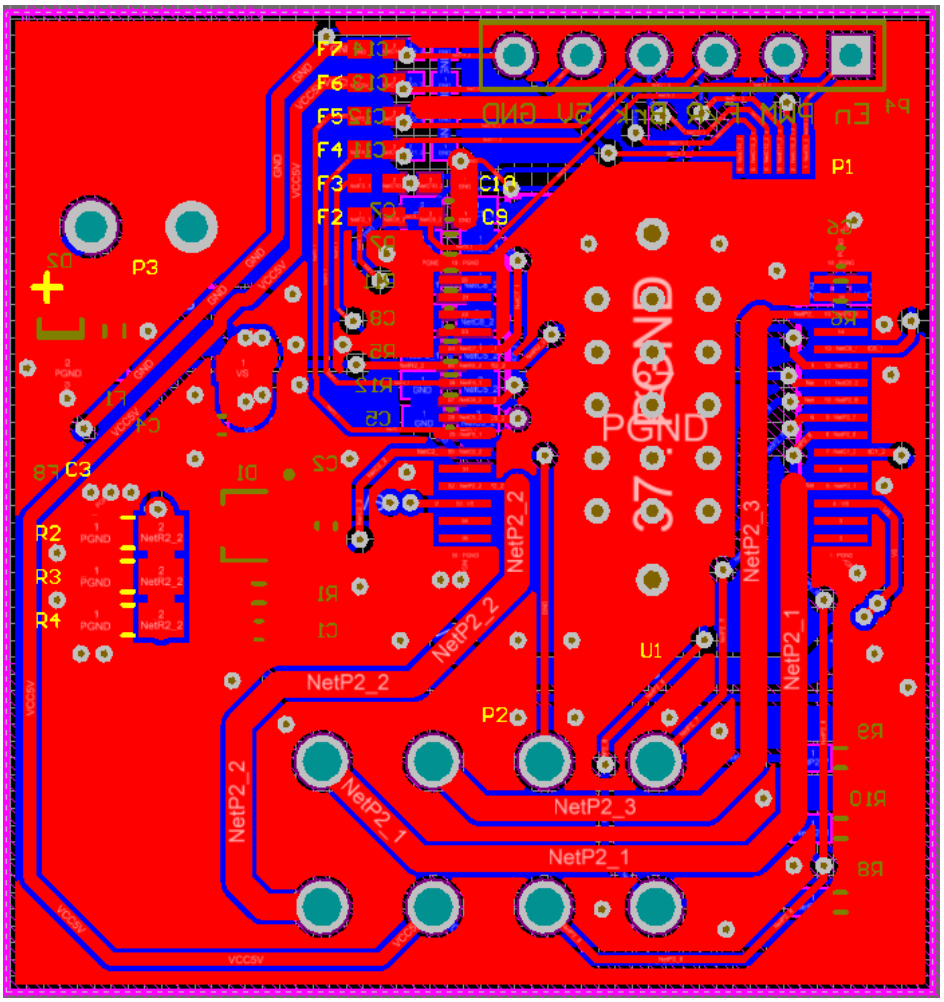

Figure 2.9. Motor driver PCB layout.

with FC filters. One of these channels is PWM signal that controls the motor speed after being converted to analog voltage. This channel must be noise and ripple free for steady motor operation. The PWM frequency is 50 kHz, which requires better low pass filtering than suggested in the IC typical application. The typical application circuit suggests RC filter with 1.8 kΩ resistor and 5.6 nF capacitor. This filter is changed to 10 kΩ - 100 nF filter. This change reduced the ripple at the reference point of the IC from 1.43 V to 15.2 mV. Moreover, in order to reduce motor noise on logic and analog signals, power ground and signal ground are separated by a ferrite bead. This reduces the noise while keeping grounds at the same DC level.

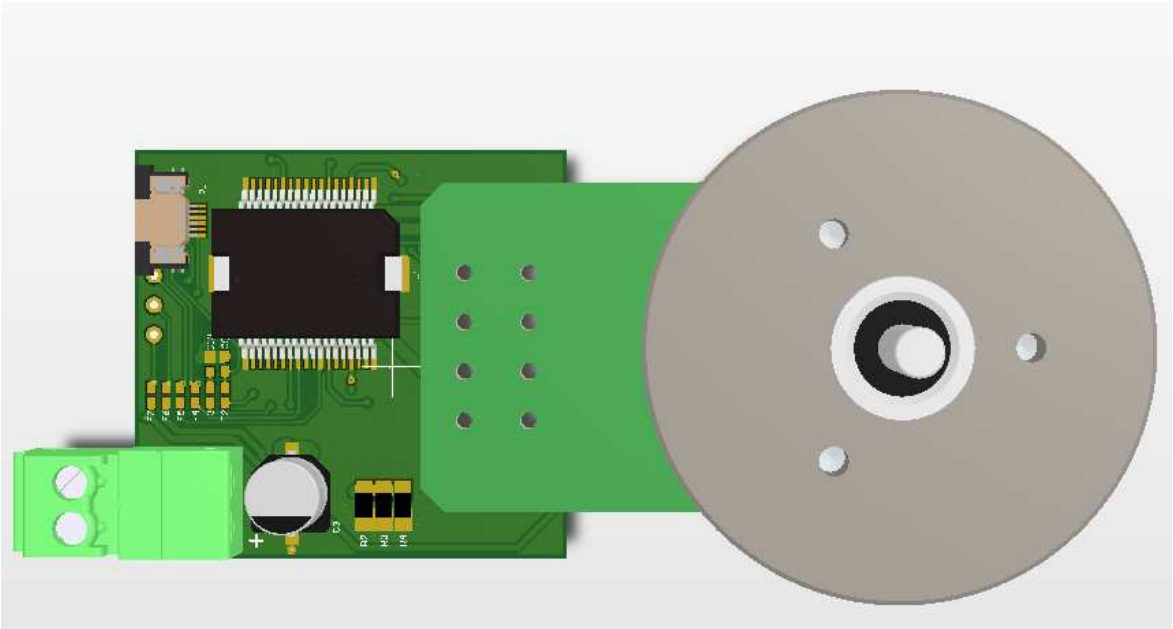

Figure 2.10. 3D model of the driver circuit mounted on the motor.

Figure 2.9 shows the PCB design of the circuit. There are several critical design considerations that must be handled carefully. In this design, no vias are used on motor signals. Thick traces are used in power lines in order to reduce trace resistance. In input power trace, it was necessary to switch layers; in such switch points, multiple vias are used to reduce via resistance. The board is poured with power ground planes and many ground vias are placed at critical points. This way, motor noise is reduced due to better return paths and heat dissipation of the motor driver IC is increased. Finally, all signal lines are physically separated from the power lines. Low pass filter of the PWM line is placed right next to the IC reducing the noise coupling on the trace. These signals' ground point is connected to the power ground at a signal point next to negative terminal of the 100 μF capacitor.

2.4. Summary and Concluding Remarks

In this chapter, advantages and driver circuit design challenges of BLDC motors were explained. BLDC motors provide high performance and durability while requiring more sophisticated driver circuit design. In Section 2.2, the importance of grounding

and isolation is shown with current designs' flaws and failures. Based on experiences from our previous designs, a set of design considerations and constraints are determined and applied to the new generation of BLDCM driver circuit.

In conclusion, a well protected, low noise, small, and modular motor driver is designed and implemented. Due to its modularity, easier maintenance of the robots is achieved. Results on motor performance will be discussed in Chapter 5 along with the control algorithm development.

3. MAIN BOARD

Main board is a unit of the robot which handles all logic operations, communication, and, in many cases in the SSL, motor driver circuits. The main board must be able to communicate with the AI computer and take instructions. It is the main board's duty to decode the AI instructions and implement them precisely and in real time. Main board must also take feedback from wheel encoders to achieve the desired speed with closed loop control. The closed loop control algorithms must run on the main board processing unit because sending the encoder data of all four wheels would put a huge burden on the communication line, which can result in unstable operation due to data loss or corruption. Therefore, a robust main board circuit which can communicate with the AI computer, run control algorithms as fast as possible without losing precision, and allow the robot to evolve with the changing nature of the RoboCup SSL, must be designed and implemented.

In this chapter, necessity of a new design will be demonstrated with examples from our previous designs. After creation of the design considerations, new design of the circuit and its implementation will be explained in detail.

3.1. Previous Designs

Our previous robot designs operated with a single board that could handle many operations at once. This board can communicate with the AI computer via low power Zigbee modules and use this communication data in order to control four BLDC motors for wheels and one BLDC for dribbler. It also can send control signals for kicker circuit [23]. This caused overpopulation on the board and resulted in performance drops in some cases. In this section, design features of the previous two generations' main boards are going to be explained with a focus on logical operations.

3.1.1. Processing Unit

The previous designs use Freescale MCU chips MC9S08DZ128 and MC56F84789 respectively [10]. The former is an 8-bit MCU which has hardware multiplication and division units [24]. This feature enables us to develop control algorithms faster and more precise without spending huge amount of clock cycles. The latter, on the other hand, combines features of DSP and MCU features in a single DSC chip which provides better FPU performance and has same amount of peripherals for additional functionality such as analog reading and UART communication [25]. Both chips can provide great performance for motor control algorithms, PWM generation, and encoder reading. Our first design shows that all these operations can be done in under 16 ms which is the interval for AI communication data packages. However, the second generation main board, even though it has a higher performance processing unit, could not meet the first generation's performance. This is caused by a design flaw on circuit design. In the first generation, encoder reading is done by timer peripherals without interrupting the operation in each encoder pulse whereas the latter does not use these dedicated timer pins for encoders. Therefore, in this main board, each encoder pulse generates an interrupt to the code flow. In the normal operation conditions of the robots, the number of interrupts can easily go up to 400 thousands per second. Since all wheels' encoders have the same priority, this problem caused false readings of wheel data. Since the communication interrupt, which takes the most clock cycles to be completed, has the topmost priority, number of missing encoder data is further increased. In the 2016 RoboCup event in Leipzig, Germany, this problem caused our robots to lose control on the field. In the competition, we increased encoder priority for better control over wheels, however, the missing communication data led us to lose control of the robots.

3.1.2. Power Regulation and Distribution

Both previous designs of our robots use the same two stage power regulation for the logic ICs and distribute the power to the boards over the main board PCB. These designs use 16.8 V and 18.4 V Li-Po batteries respectively to power all the motors and

circuitry.

Stepping down the battery voltage to 5 V and 3.3 V is necessary for the integrated circuits on the main board to operate. Both our previous generations use LM2575 Fixed Output Buck converter to step down the battery voltage to 5 V and a linear regulator to further step down to 3.3 V with 5 V input. Considering almost all IC's except encoders and Hall sensors use 3.3 V, most power to the main board is provided by linear regulator. Taking into account that our board consumes an average of 300 mA for the main board, 0.51 W power is wasted and dissipated as heat on the IC (See (3.1)).

$$P_{Loss} = (V_{in} - V_{out}) \times I_{out} \tag{3.1}$$

Power distribution is another problematic issue of the main board design. Since motor driver IC's are distributed over the PCB for symmetry and location with respect to motors, battery voltage power lines cover the main board in previous designs. These traces carry up to 5.6 A of current in high speed - high load situations. Therefore, they impose a tremendous noise to the logic circuitry and cause crosstalk due to daisy-chain trace topology. The effect of grounding is explained in Section 2.2 in detail. Those effects apply to logic circuitry as well. Figure 3.1 shows the PCB design of the first generation robots. The second generation of the main board distributed the logic board and power board, however, the power board's supply traces travel across the main board which still creates high level of noise on the logic ICs.

3.1.3. PCB Design

There are several PCB design features of the previous main boards that should be avoided. First of all, power traces to the motors carry high switching currents which are prone to create radiated emission of noise. In the previous boards, daisy chain topology was used to connect these traces. Among three main routing topologies (star,

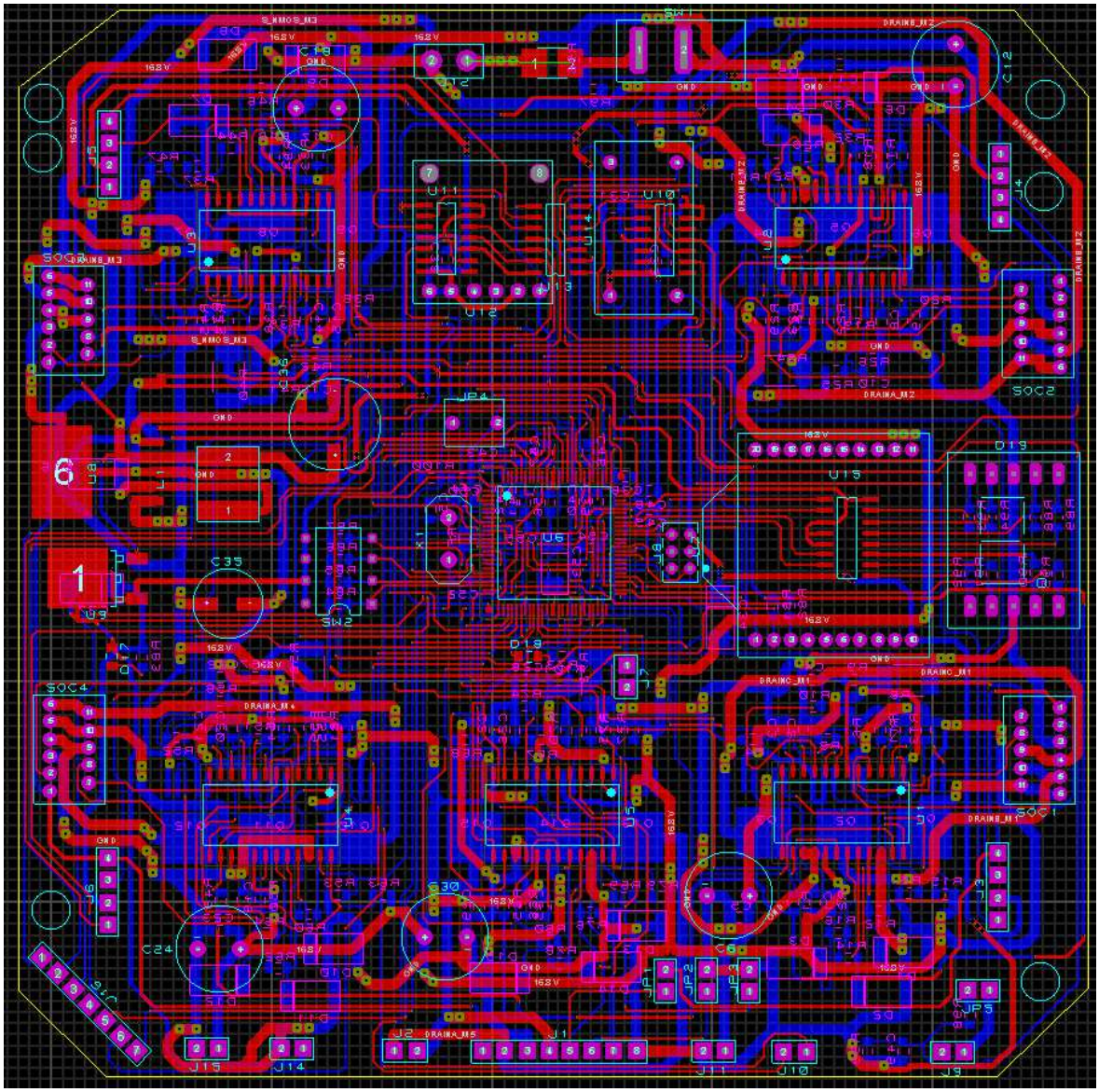

Figure 3.1. The first generation main board design.

tree, and daisy chain), daisy chain is the most noisy option due to longer signal paths, while star connection provides with the lowest radiated emission [26].

Ground traces must be as low-impedance as possible, otherwise noise coupling occurs between circuits as the return signal would follow the lowest impedance path [27]. Therefore, ground traces must be replaced with ground planes with proper return path handling.

Noisy traces below vulnerable components and vulnerable traces below noisy components must be avoided. In the previous design, there are traces below the buck converter inductor, which creates a perpendicular magnetic field to the PCB. When the feedback of the buck converter passes below the inductor, it reduces the control performance of the converter. Moreover, output or ground trace below the inductor would distribute this magnetically coupled noise to be distributed all over the main board. On the other hand, noisy signal traces such as PWM signal to the motors should not be routed below sensitive components. In the previous boards, PWM trace passes below crystal oscillator, which can reduce the clock accuracy of the MCU. Since our control algorithms are done discretely with a specific time interval, this accuracy loss can affect control performance of the wheels.

3.2. Design Considerations

In light of the previous section and the literature review, the requirements for the new main board can be listed as follows:

(i) The processing unit must be able to parallel process the motor control without interruption and also should be able to process analog signals and communicate with the wireless module easily. Therefore, a dual processing unit with an FPGA and an MCU should be used.

(ii) The most crucial part of the electronic circuitry, motor control, has to be controlled by any of the processing units if necessary. This switching of duties must be easy to implement and should not create any noise or interference on the circuit.

(iii) Data transfer between these units must be fast and efficient in order to increase control algorithm and communication performance. System delays can reduce the reaction time of the robots.

(iv) Power regulation and distribution must be power efficient and noise free. Battery life should be enough for a robot to complete a game of 20 minutes. Crosstalk and control signal noise must be avoided strictly.

(v) The main board has to be adaptable and improvable for the changes in RoboCup SSL requirements. This includes additional sensor networks, communication modules, and analog and digital input/outputs.

(vi) The PCB should be able to be mounted and demounted easily and also be clear of any mechanical load. This brings the chance for quick maintenance during games and also protects the circuit from external damages.

(vii) Proper signal and power routing as well as grounding scheme is crucial for circuit performance. Sensitive components and dangerous traces must be handled with an extra focus.

(viii) Sufficient ports must be reserved for easier debugging and firmware development.

(ix) Due to dangerous nature of the Li-Po batteries, emergency switch must be used and placed in a way that can be reached easily.

(x) Component selection should make the design cost effective, easy to populate and maintain, sustainable for future manufacturing, and size limited for additional circuitry.

3.3. New Main Board Circuit

The new generation main board design is aimed to be efficient, modular, and improvable. In this section design processes of the main board circuit will be shown in detail.

3.3.1. Power Regulation and Distribution

In order to achieve noise and crosstalk free and power efficient power regulation, several measures were taken. First of all, as depicted in Figure 3.2, battery voltage is fed through the circuit with an emergency rocker switch. This voltage is distributed to six different ports for five motors (four wheels and a dribbler) and kicker circuit. These six ports are HF noise protected via two ferrite beads on both supply and ground rails. This way, crosstalk and motor noise effect on the main board is planned to be eliminated. Furthermore, battery input to the main board circuit is filtered by a

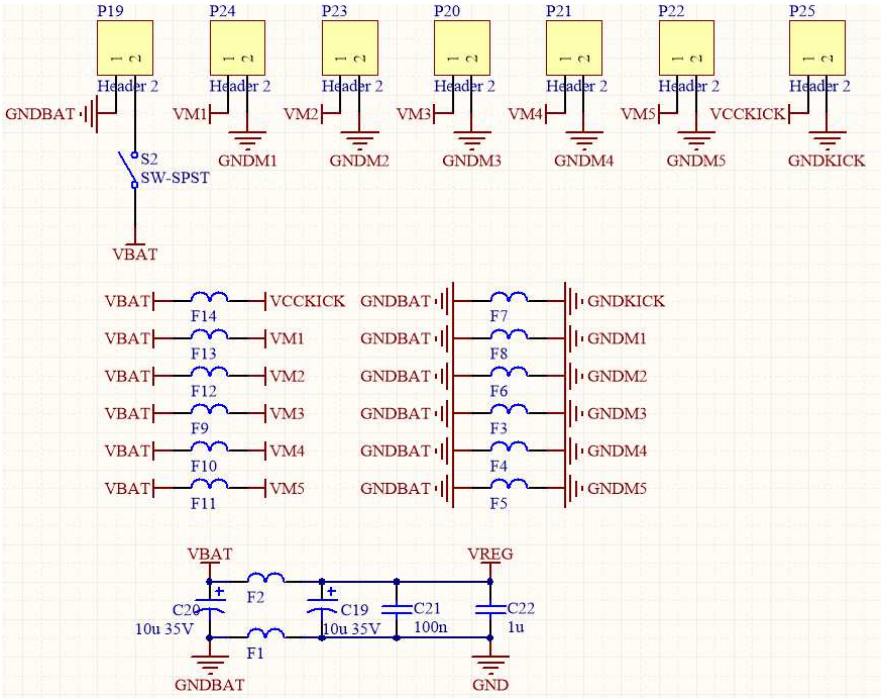

Figure 3.2. The new generation power input regulation and distribution.

differential mode filter for additional noise protection due to motors.

For the sake of increasing stepping down power efficiency, two buck converter ICs (Semtech TS30011) are used in parallel. These converters generate 5 V and 3.3 V supplies separately from the battery voltage. Both converters can supply 1 A load current and have high input voltage capacity of 24 V; therefore, the power supplies can work with higher-cell batteries and also can support extensional circuitry such as sensor networks. Additionally, the FPGA chip requires 1.2 V supply for its internal logic supply apart from auxiliary ports and output drivers. Since this supply consumes small currents as low as 4 mA, a linear regulator is used to reduce the voltage supply from 3.3 V to 1.2 V [28].

3.3.1.1. PCB Layout. Figure 3.3 demonstrates the power regulation and distribution network without ground planes for better visibility. The six terminal blocks (P20 to P25) are the connections for the motors and the kicker. Each of them has ferrite beads in their inputs for noise and crosstalk isolation. They are also positioned in a way that

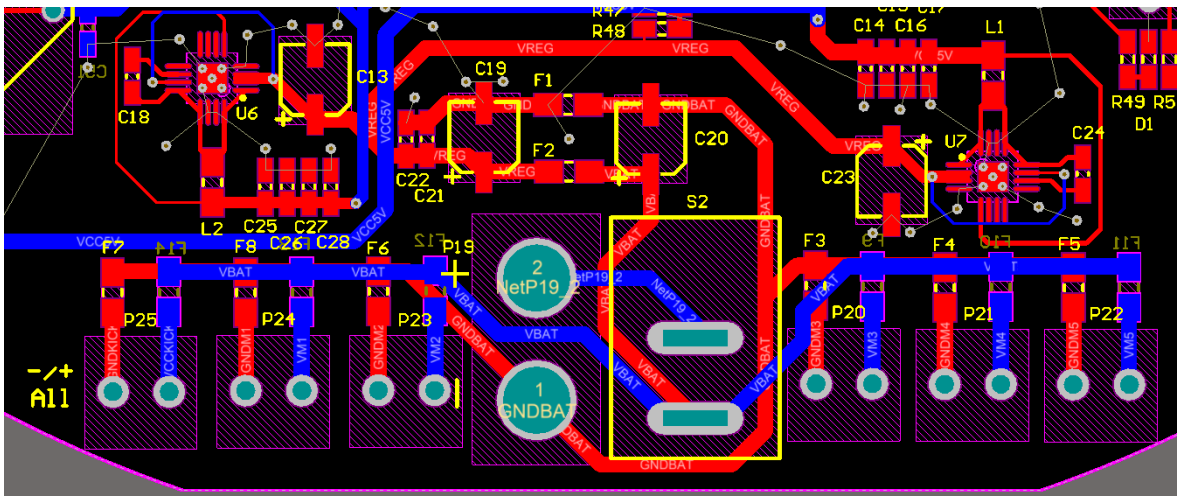

Figure 3.3. Power regulation section of the main board.

their traces do not travel across or around any logic circuit and very short paths are used for lower impedance and travel distance.

Right above the battery input, P19, a differential mode filter is placed and its output supplies both buck converter ICs, U6 and U7. Notice that there is no trace below or very close to inductors which create a great amount of magnetic field and can impose noise on the nearby traces. The feedback traces do not have any vias for better level sense and voltage regulation. There are also many ground vias especially close to ground pins of the capacitors for low impedance for ground return paths. Finally, all current carrying traces are drawn as thick as possible to reduce transmission impedance.

3.3.2. Processing Units

3.3.2.1. FPGA Chip. The main processing need of the robots is handled by the FPGA chip due to its parallel processing capability. Contrary to CPUs, FPGA chips operate concurrently by means of dedicated hardware blocks and look-up tables; therefore, defined circuit blocks do not necessarily depend on each other. This way, we can develop a motor control block and use four of them simultaneously for each wheel. In micro-controller units, only one thread of code can run at any given time. Since our application has to fulfill many different operations at the same time continuously, using

an MCU causes interruptions; hence, an FPGA chip is decided to be used for critical operations such as motor control and encoder reading.

Selection of the FPGA chip is done according to some considerations. First, the chip must be easy to be placed to and removed from the main board by hand so that maintenance of the main board can be done without professional PCB assembly equipment. The chip must also have enough I/O count such that all motors, their encoders, MCU communication, and future improvements can be done in parallel. Therefore a LQFP package chip is selected over BGA type packages for easy mounting/demounting, and 144 pin version is used due to its pin count. Moreover, the chip must be general application type since their price increase as their grade increases and their import gets more and more difficult because FPGAs have mostly military applications. In light of these limitations and requirements, a Xilinx XC6SLX9-2TQG144C Spartan-6 chip is selected.

3.3.2.2. MCU Chip. Even though FPGA chips have many advantages over MCU chips, they also have a number of drawbacks. These disadvantages can be covered by an MCU very efficiently. First of all, FPGAs do not have non-volatile memory so that they cannot store programming data after power down and must be re-programmed in every power up. They are also completely digital circuits; they need external ICs to read or create analog signals. Moreover, most FPGA chips do not have FPU operation capability, so floating number operations must be handled manually, which is difficult to implement and costly for FPGA capacity. Finally, they do not include hardware peripherals such as UART, timers, and I^2C. These types of peripherals can be used as free-IP modules but they reduce the gate capacity for the rest of the design. A micro-controller unit that can fulfill all these needs must be used on the main board.

The selected chip must be able to have enough flash memory to keep its own program and FPGA's program as well. It should be able to communicate with Zigbee module easily, measure analog signals such as battery voltage and kicker charge level, read encoder signals without program interruption, and have enough pins to take all

duties from FPGA, program it, and communicate with it. The chip should also have high clock frequency to complete periodical duties in short time so that additional functions in the future does not affect the software architecture. An ST Microelectronics ARM Cortex M4 MCU, STM32F407VGT6 with LQFP 100 pin packaging, is selected for the design.

3.3.2.3. Cooperation and Flexibility. In the designed dual processing unit system, communication between the chips is of crucial importance for fast and efficient cooperation between chips and duty sharing. In the current design, a six bytes of data package from the MCU to the FPGA is sufficient. In order to transfer this data from one chip to the other, a parallel six-byte SPI module is designed on both chips with an additional channel for future. This module is placed in a single GPIO port of the MCU so that the communication can be done in lowest clock cycles for fast data transfer between chips. Details of this module will be explained in Chapter 4.

Software duties such as mathematical operations or timer counts can be shared and swapped among the chips with the data transfer over the SPI bus, easily; however, there are also hardware duties such as controlling logic signals to the motor driver ICs. This type of changes have effect on the hardware. For instance, if we decide to replace the PWM generation for the motor speed reference from FPGA to the MCU, there should be a predesigned route from MCU to the motor driver. Moreover, these signals must be somehow isolated for proper operation. This can be done by setting unused pins as high impedance, using additional circuitry to nullify the former controller's effect, or connecting and disconnecting traces. Among these options, the most efficient and risk-free method is to switch traces with jumpers. Figure 3.4 shows how the signals are isolated from each controller. In each important feature of the robot that may change its source in the future, the trace is separated by 0 Ω resistors. By assembling only one of the resistors, one of the routes will be completed while the other is disconnected. SMT resistors are used in order to reduce the contact noise compared to plug-in jumpers and cut-traces are avoided for reusability of the same PCB for further changes.

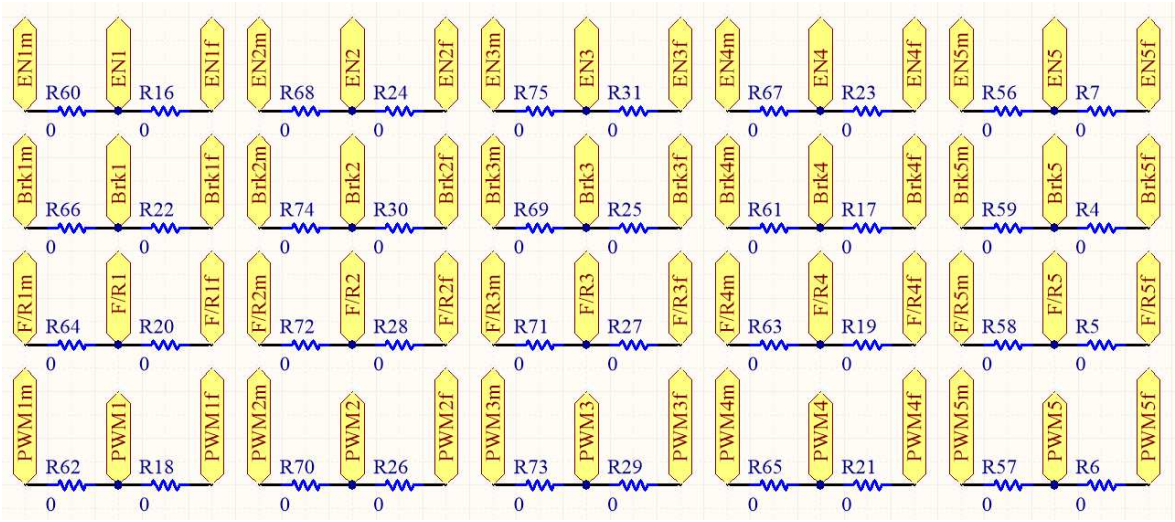

Figure 3.4. Trace swapping jumper resistors.

3.3.2.4. Improvability. RoboCup SSL robots must be able to adapt changes considering that the rules of the game are changing by time in order to increase challenge and contribution to the robotics. Moreover, in the nature of the competition, teams add features to gain advantage. The robots should be able to evolve easily to catch the other teams and/or implement game changing improvements. This can be achieved by designing a new electrical board or building a system that can integrate additional circuitry with the current main board. In this main board design, we added many ports that can be beneficial for future improvements. Table 3.1 shows additional ports in the main board, which can create many possibilities to enhance the robots. The values in the parentheses show number of ports that can be achieved by cancelling some other features.

These additional ports can be used for many applications. These include a sensor network over I^2C, communication modules over UART, analog sensor reading from ADC, and many different ICs with 3.3 or 5 V compatibility.

3.3.2.5. PCB Design. Figure 3.5 shows the PCB layout of the main board. In this board, several design points were considered. First of all, the components are placed to facilitate connection of the cables to motor drivers and kicker as well as to make routing

Table 3.1. Numbers of extension ports.

	FPGA	**MCU**	**Supply**
GPIO	26	2(10)	-
Analog Input	-	2	-
UART	-	2	-
I^2**C**	-	1	-
5 V	-	-	2
3.3 V	-	-	6

easier and via-optimized. In order to achieve that, motor logic signals are placed as close as possible to the motor drivers. FPGA (U2) traces are routed mainly on the top layer (red traces) and their signal integrity is prioritized. Power connection terminals are placed at the back of the board where battery supply is connected. This way, travel distance of the power traces are reduced to minimum. Mounting of the PCB is also important for maintenance of the robots during the games. J3 and J4 are holes in the PCB through which two aluminum columns pass. The columns are used to fixate the plastic cover of the robots. With the help of two screw holes on J1 and J2, and the columns through J3 and J4, the PCB is fixed to the chassis of the robot and free from any mechanical load from collisions and robot's movement.

Trace symmetry is another design factor that can affect the performance of the motor control algorithm. PCB layout is as symmetrical as possible, especially for motor reference signal, so that a single PI control loop design would work on each wheel with the same performance. Traces are also designed according to their current loads. All supply traces are routed with thick layers with star topology, in order to provide the ICs with noise-free and equipotential power supply. The FPGA and the MCU chips have their power supply distributed from their centers in the bottom layer. The additional 1.2 V power supply of the FPGA is placed close to the chip in order to reduce impedance and noise on the supply. Clock crystal of the MCU and MEMS oscillator of the FPGA

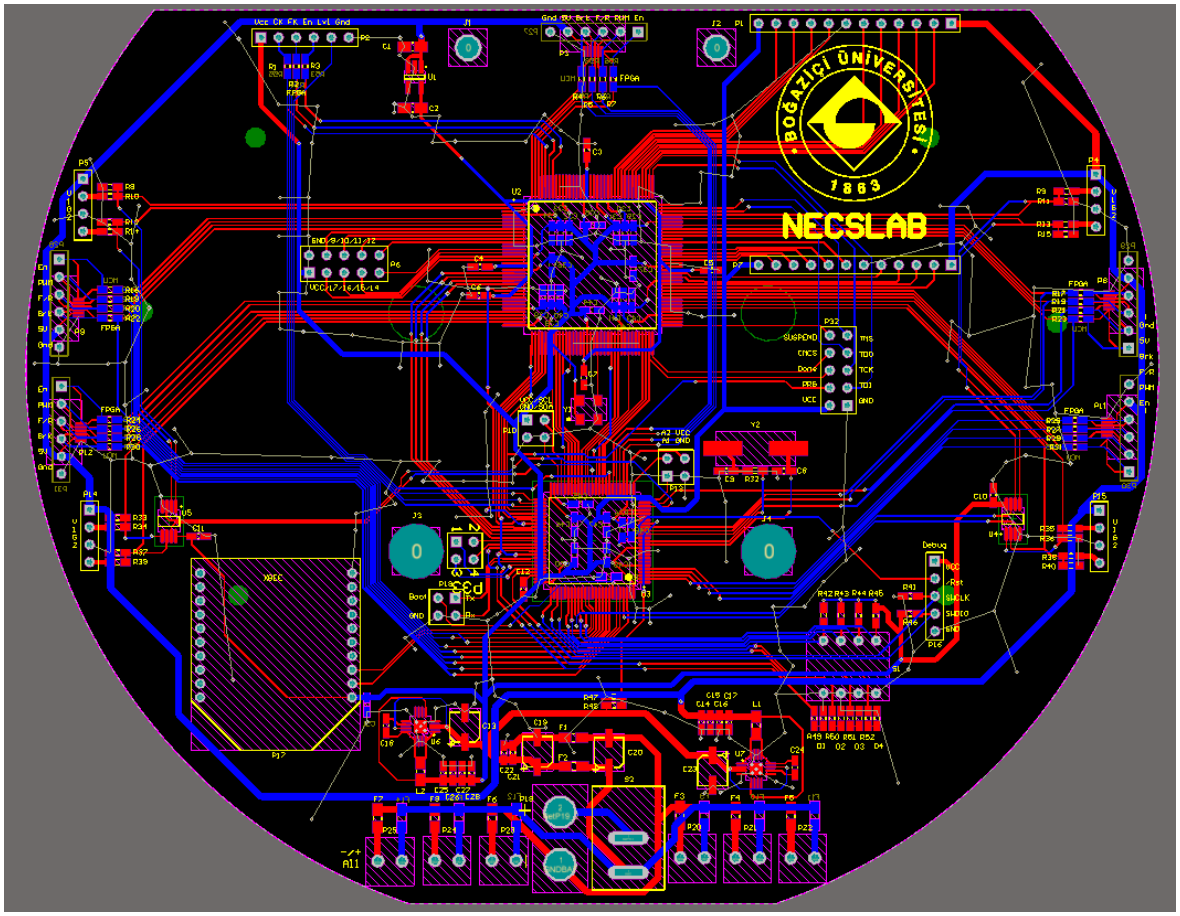

Figure 3.5. Main board PCB layout.

are also placed close to the chips and no trace is routed under them because these components are highly susceptible to external noise and any decrease in their accuracy immediately affect the chips' performance.

There are many rectangular header connections on the main board, these are extension ports reserved for future improvements. They can be type of daughter boards or wired to another board mounted on the robot. The features of these ports are shown in Table 3.1.

Omitted in Figure 3.5, both routing layers are poured with ground planes. These planes are connected with extra vias at trace intersections to provide the shortest ground return paths.

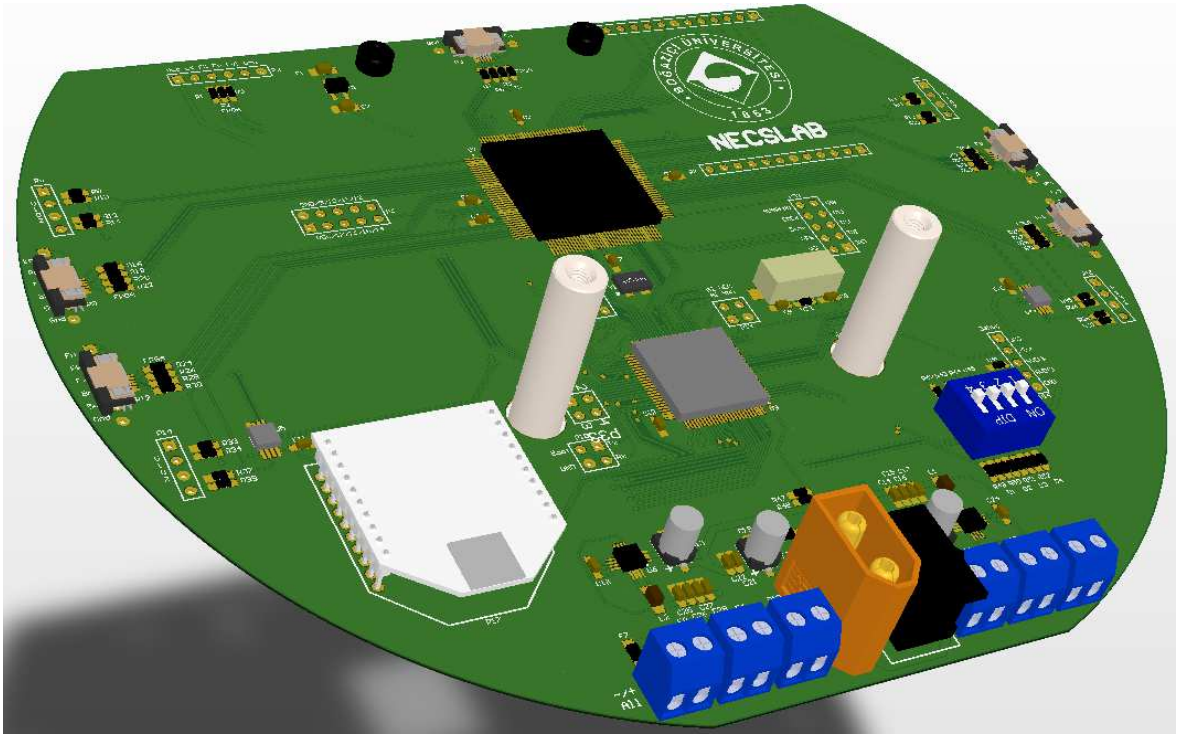

Figure 3.6. Main board PCB.

3.4. Summary and Concluding Remarks

In the beginning of this chapter, problems in the previous versions of the main boards are reviewed. The importance of processing unit selection, power regulation and distribution and PCB layout are stated with examples from previous generations' failures. Based on this knowledge and expectations from the RoboCup SSL, a set of improvements and corrections on the previous generations are proposed. Main considerations of the main board design can be summarized as high-performance processing unit(s), good power design in terms of efficiency and noise effects, easy manufacturing and usage, facilitation of improvement and modification for future developments, and good PCB layout that is noise-free and able to perform as desired.

Then, design and implementation of the new generation main board is explained in detail. The figure of merit for this board is to run control algorithms fast and accurately, communicate without error with the AI computer, and run the motors with precision. The results for these will be given in Chapters 4 and 5.

4. LOW LEVEL FIRMWARE

Once the robotic hardware is designed and implemented, the second part is to develop low level firmware for the processing units. The firmware must be time efficient, robust, and improvable. This firmware includes communication, security, motor control, sensor readings, and so on. The firmware design can be divided into two main parts which are MCU and FPGA programs. The duties to run the robots smoothly and in a robust manner must be divided between the MCU and the FPGA chips according to their strengths. In this chapter, the firmware development and implementation will be explained as well as the additional software tools to facilitate development process.

4.1. Microcontroller Unit Firmware

In this thesis work, an ST Microelectronics STM32F407VGT6, ARM Cortex-M4 microcontroller unit is used. This chip has FPU processing capability and many important hardware peripherals such as GPIO, USART, timers, analog to digital converters, and I^2C. Moreover, the chip has 1 MB non-volatile flash memory which is used to store the firmware as well as FPGA configuration data. Based on these features, the chip is mainly responsible for AI computer communication, FPGA communication, FPGA programming, security checks, battery and kicker level measurements. Figure 4.1 shows the main program flow of the MCU. In this section, main features of this firmware will be explained. These are system level initializations, FPGA programming, user I/O, communication with the AI computer via XBee module, FPGA communication, and security measures.

4.1.1. System Initialization

In order for the MCU chip to perform certain tasks, its core and peripherals must be configured accordingly. The following subsections give details about the MCU configuration.

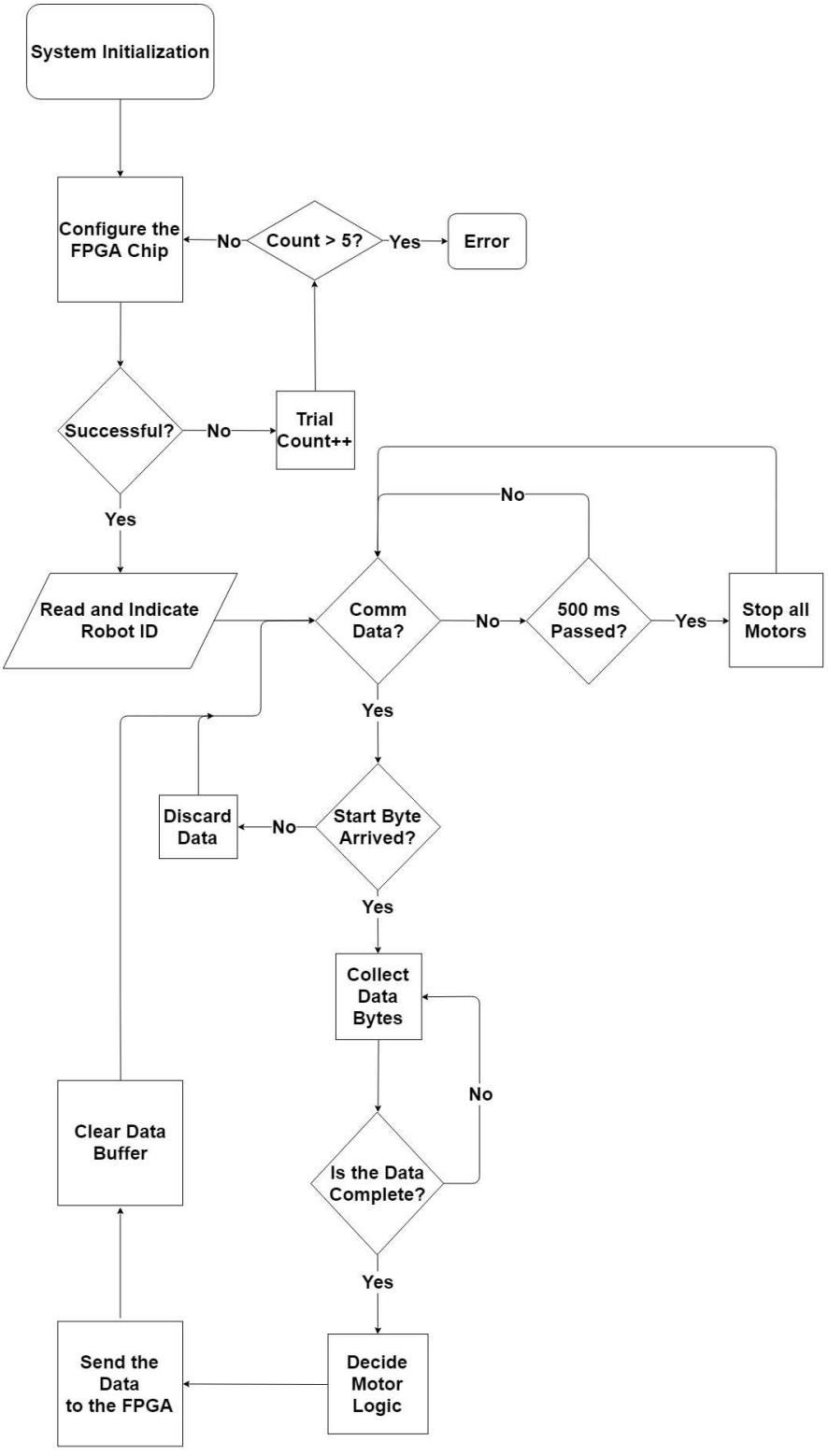

Figure 4.1. Microcontroller unit firmware flowchart.

4.1.1.1. Core Initialization. There are two main configurations for the MCU core, clock and interrupts. The system clock can be fed from either internal VCO or external clock crystal. For high speed applications, external crystal must be used and configured accordingly. In the main board, an 8 MHz crystal oscillator is used. This clock source does not give logic clock signal but an accurate low voltage oscillation. Therefore, the signal must be multiplied to the desired clock frequency and amplified via phase lock loop. The PLL is configured such that the system clock runs at the chip maximum of 168 MHz. Moreover, the chip features clock control for peripherals to reduce power consumption. The clock signals for the used peripherals are also enabled in this configuration.

Secondly, NVIC registers must be configured for the system to be able to receive interrupts. Interrupts are very important in multitasking operations since the MCU core can handle only one thread of code. Moreover, it is more speed efficient than polling in multi-interrupt firmware because it takes fewer clock cycles to handle the interrupt in the worst case scenario. There are two interrupts configured in this design which are sysTick and UART receive interrupts. The sysTick interrupt is a fixed period interrupt mainly used for periodic operations such as sensor checks and time controlled operations. In this application, sysTick interrupt is used for periodic analog readings and measurement of the time between consecutive data packages. The second interrupt configured for the application is the UART receive interrupt. This way, when a byte of data arrives to the MCU from the AI computer via XBee module, the main code operation stops and the received data package is handled. This interrupt is also configured to have the highest priority for security reasons because its data is used to control all the wheels. Any data from the AI computer must be handled immediately to avoid collisions and undesired behaviors.

4.1.1.2. GPIO Initialization. The most used peripheral of the MCU is General Purpose Input Output. This peripheral handles user interface (buttons and LEDs) as well as FPGA configuration and communication. These interfacing pins must be configured as inputs or outputs before high and low logic level signals are set or read on them.

Moreover, there are several additional configuration options which must be considered for proper operation. For example, pull-up and pull-down resistors at the pins must be selected by writing into the corresponding register. These are important for floating inputs and pins with external pull-up or pull-down resistors. Pin speed is another feature of GPIO configuration. As the I/O toggle speed is increased, its power consumption increases as well. Therefore, speed requirements for the pins must be calculated and the most power efficient pin speed configuration must be selected accordingly.

4.1.1.3. UART Initialization. AI computer communication is of crucial importance for the robots to comply with the decisions made by the strategy algorithms. The UART peripheral takes the orders from the AI computer via an XBee wireless module that outputs the data over UART. UART communication interface has many configuration options for the peripheral and the GPIO pins they are connected to must be configured as UART pins. GPIO configuration requires alternate function selection as UART transmitter and receiver and the pin speed must be configured as the highest setting. In this application, the UART peripheral is configured with 57600 Baud rate, 8 bit data, 1 stop bit, no parity, no flow control, and 16x oversampling. Oversampling is selected as 16 in order to reduce communication errors without going under the burden of using the hardware flow control pins. Finally, UART receiver interrupt is enabled via the configuration registers and the peripheral is enabled to work in a free-running manner.

4.1.2. FPGA Programming

One of the main duties of the microcontroller unit is to store the FPGA configuration data and program the FPGA chip on every power on. This duty is required because the FPGA chip is not able to store its own program due to the lack of enough non-volatile memory. There are several methods to configure the FPGA chip which can be either master or slave, and serial or parallel. In this application, we used parallel-slave configuration method. In the master configuration, the FPGA chip controls an external memory IC which was unnecessary due to the sufficient size of the MCU flash

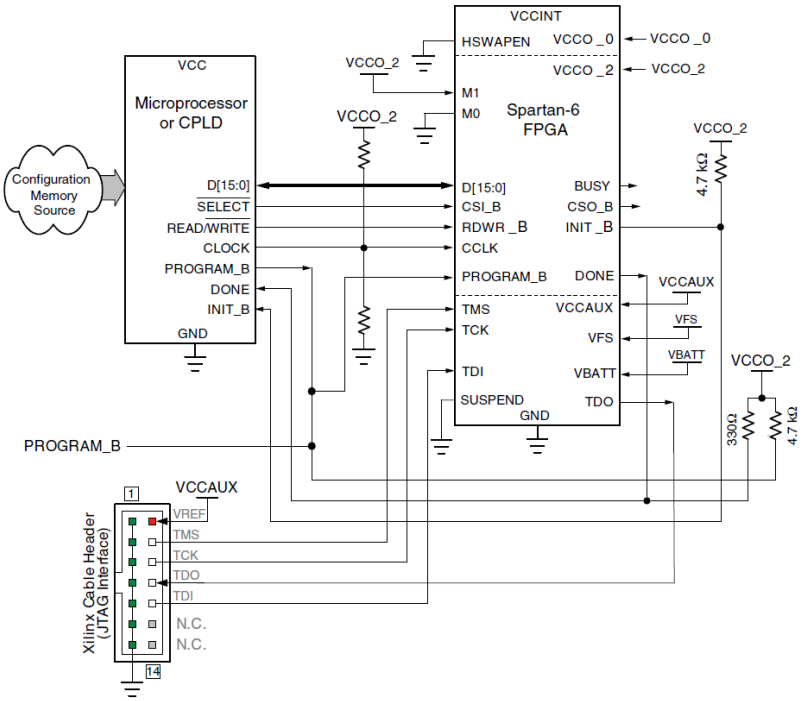

Figure 4.2. FPGA configuration schematics for slave operation.

memory. Therefore, the FPGA must be the slave in the configuration stage. Parallel configuration with 8-bits is selected in order to reduce the programming time by 8. In this way, system initialization process is done as fast as possible such that the robots are ready to operate quickly after power-on or reset. Figure 4.2 shows the necessary connections for the aforementioned configuration method [29].

Once the necessary hardware connections are made on the main board, several configuration, data, and clock pins must be set or reset in a predefined sequence with timing requirements as shown in the Figure 4.3 [29]. Figure 4.4 depicts the algorithm for the FPGA programming function. When the FPGA is configured properly, the chip releases the open drain output of DONE pin to indicate successful configuration. The MCU controls this pin once the data is sent completely. If the configuration is complete without any errors, the main program starts to run; otherwise, the MCU restarts the configuration sequence up to five trials. If all the trials are unsuccessful, the MCU goes to an error loop. In this loop all four LEDs on the main board flash continuously in order to indicate the robot handler that there is an error on the sequence so that the

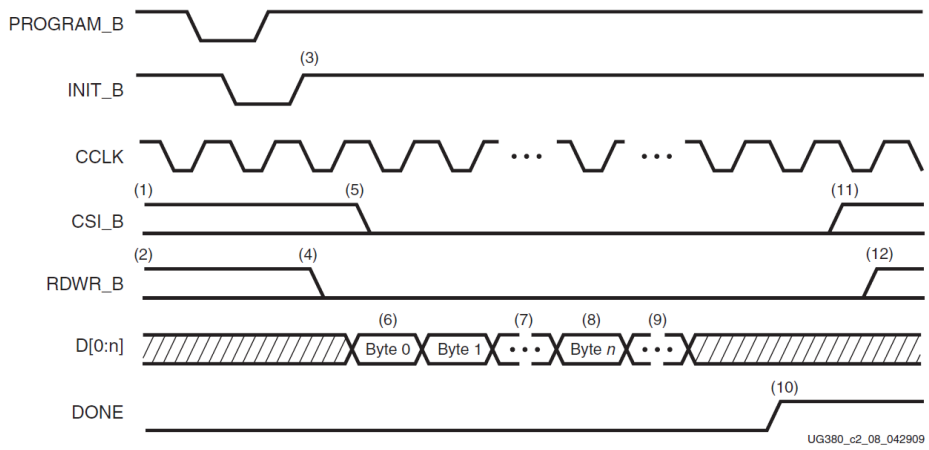

Figure 4.3. FPGA configuration sequence.

main board must be restarted or maintained for hardware errors. In order to increase the speed of the FPGA programming, there are no delays between data bus assignment and clock edges.

4.1.3. User Interface

The robots are identical in terms of hardware and firmware, however, they need to be differentiated in order to assign them different tasks during the game as well as control them separately. This can be done either hard coding a unique ID to the robots in the firmware or setting the ID with a user interface. Hard coding the ID raises several problems. First of all, in order to substitute robots during the game or interchange their positions, each robot has to be programmed with the MCU firmware with a small change of code. This loss of time is highly undesirable during the competitive games. Therefore, a user interface must be added to the main boards, which can easily communicate with the MCU chip and give feedback to the robot handler. In this design, we used a four-pin DIP switch which sends BCD data to the microcontroller. These four stationary switches are used to set a unique robot ID so that the robots can be differentiated in terms of firmware. Since there are eleven robots on the field at most, four bits are enough and necessary in order to identify each robot with a different ID.

```
trialCount = 0
codeIndex = 0
for trialCount < 5 do
  set PROGRAM_B
  set INIT_B
  reset PROGRAM_B
  wait 1 ms
  reset INIT_B
  wait 1 ms
  set PROGRAM_B
  wait 1 ms
  set INIT_B
  wait 1 ms
  for codeIndex < codeSize do
    reset CCLK
    D[0 : 7] = fpgaCodeArray[codeIndex]
    set CCLK
    codeIndex++
  end for
  if fpgaDone then
    break
  end if
  trialCount++
end for
if trialCount > 5 then
  ERROR
else
  return
end if
```

Figure 4.4. FPGA programming algorithm.

Table 4.1. LED indications.

LED STATUS	POWER-ON	OPERATION
Stationary	Successful Programming	Robot ID in BCD
One LED Flashing	Programming is started	Low battery
Two LEDs Flashing	Programming is at 50%	No communication data
Three LEDs Flashing	Unsuccessful attempt	Kicker failure
Four LEDs Flashing	Programming error	Kicker overcharge

Apart from sending data to the MCU, it is also important to read data from the chip. This must be done in a quick and efficient way. In our robots we used four LEDs to visually read crucial robot signals. During power-on FPGA programming sequence, the LEDs show the status of the FPGA programming and during normal operation of the code, they indicate robot status. Table 4.1 shows these indications and their meanings. Due to the number of LEDs, upto 15 error messages can be given in flashing form since the stationary zero is reserved for the robot ID zero.

4.1.4. AI Communication

The artificial intelligence computer processes the global vision system data, decides strategies, and assigns duties to the robots on the fields. This output is in the form of wheel speed reference and additional commands. Data from the AI computer must arrive to the MCU completely, and the MCU must process the data without delay for proper robot operation. In our robots we used XBee RF modules that can communicate in pairs or broadcast wireless data in 2.4 GHz ISM band. The XBee module is preprogrammed with an adapter and a dedicated software (it can also be done via the MCU) in order to set UART communication speed, wireless channel, and the network among the modules. In our application, an XBee module, connected to the AI computer, broadcasts a complete set of data package for all robots on the field 60 times per second. In Figure 4.5, contents of the data package are shown.

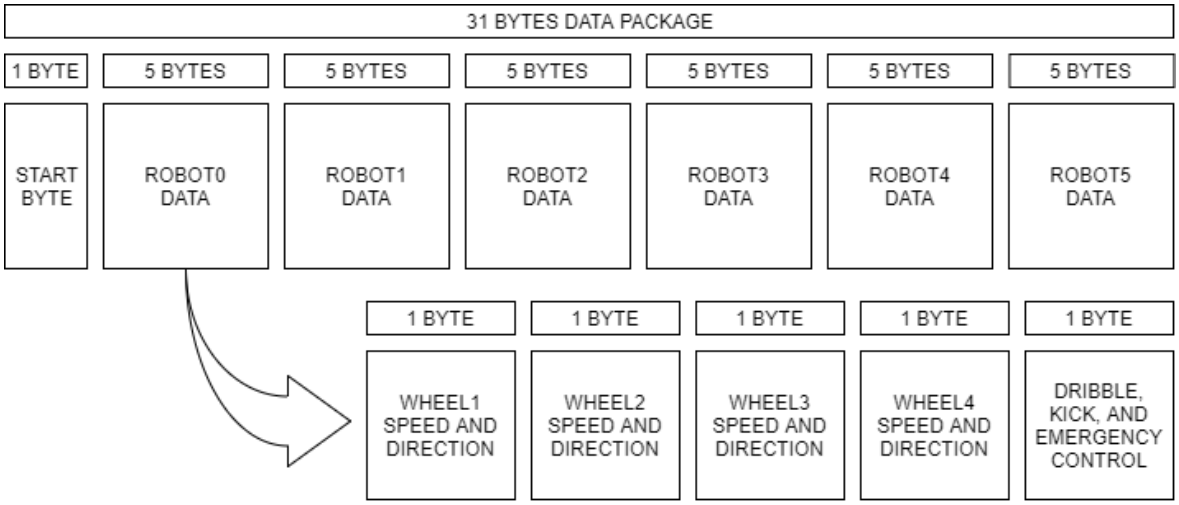

Figure 4.5. XBee communication package.

The data packages are identical for every robot on the field, therefore, each robot must take dedicated data for themselves and process it to convert into a meaningful form for the FPGA chip, which is the final recipient of the AI messages. The 31 bytes of data package consists of one start byte and 5 bytes of robot data for the six robots on the field. They are in ascending order with the robot ID so that it is easy to parse the desired data for the robots. The start byte, 0x4D (decimal 77) indicates that the communication sequence is started. Any package which does not start with this byte is discarded for security reasons. Then each robot parses the five bytes of data addressed to themselves. These bytes contain each wheel's speed and direction, kick and dribble commands, and an emergency brake signal for all four wheels. The first four bytes are dedicated for wheels one through four respectively. The MSB of the byte indicates the motor direction. The remaining seven bits contain the desired wheel speed between 0 and 127, represented in terms of encoder pulses per control loop interval duration. The details of the control loop will be given in Chapter 5. The last byte of the package is dedicated for the kick, dribble, and brake commands. The MSB is the emergency brake signal, the lesser significant bit enables and disables the dribbling, and the sixth bit indicates the type of kick, chip or flat. The remaining five bits show the duration for the selected kick method, which, in turn, controls the speed of the kick so that accurate shots and passes can be achieved.

Once the communication data is sent to the FPGA, the chip runs accordingly, until another package arrives. Because of this slave manner of the FPGA, if communication data stops due to an error on the software or the hardware, the robot will lose control. In order to prevent this, a time counter based on sysTick interrupt is set. This counter resets after a data package is received and a flag is set if there is no communication data for 500 ms. The MCU firmware checks this flag in a polling manner and if the flag is set the MCU signals the FPGA to brake all the motors until the communication is active again. During this time, the error message shown in the Table 4.1 will be given from the LEDs.

4.1.5. FPGA Communication

The previous subsection explained the decoding method for the AI communication signals. Once these signals are transformed into meaningful data in the MCU chip, they should be sent to the FPGA chip immediately to take action on the AI orders. In order to achieve this, a parallel SPI communication is designed and implemented on both MCU and FPGA chips. In this communication method, the MCU is the master and the FPGA is the slave. In other words, the MCU chips decides to initiate the communication and synchronize it whenever necessary. For the six bytes of robot data, a six pin bus is created between the processing units and a clock signal is also connected to control data flow among them. The MCU assigns the bus with the MSB of each data byte and sends a rising edge from the clock pin. Then, the data is shifted on the bus seven times, each followed by a rising and falling edge on the clock pin.

This design has some advantages and disadvantages over other possible communication methods. The data could have been sent serially over a 2-bit interface or byte by byte over a 9-bit interface (eight data bits and one clock) or asynchronous communication could have been implemented. The synchronous communication has the disadvantage of additional clock connection however it reduces the design effort and increases reliability drastically. Among serial synchronous communication methods, the two bit interface provides the most efficient hardware design however increases

the communication time by far. Seven and nine bit communication methods, on the other hand, have similar hardware complexity and communication speed. The reason for choosing the seven bit interface is to reduce the design effort on the FPGA side given that they have no significant difference in the performance and MCU firmware complexity.

4.1.6. Additional Software

The FPGA codes are formatted as .bit or .hex files. These raw data forms must be converted into C language in order to add them into code and store in the flash memory. For this purpose, a Python script is developed, which reads raw FPGA data and outputs a .c file, that can be added as a source document to the MCU firmware project. Figure 4.6 shows the algorithm for the script. This algorithm is based on the fact that the FPGA configuration code is fixed size for device families independent from the complexity or content of the code itself. The code array is therefore created in C format and it is stored in the flash memory. In order to notify the compiler to store it into the flash memory, the array is defined as *const* char array because constant variables are stored in the flash memory while the others are stored in the RAM.

Moreover, in order to record wheel data and display it on the computer, an Arduino script along with a compatible Verilog module are designed. This data is then used for accurate PI tuning of the motors. This script reads the encoder reading before each cycle of the PI control loop and stores them in the Arduino chip's RAM. Then it transmits the data over serial port to the computer. This script is written in a custom slave SPI manner, which gets its data and clock from the FPGA chip where a module that transmits the encoder data, synchronous to the PI loop clock is designed and implemented.

```
codeSize = 340604
Read .bit file
Open .c file
Print "#include <definitions.h>"
Print "const char fpgaCode[codeSize] = {"
for index < (codeSize * 2) do
    Print "0x"
    Read two half bytes
    Print the half bytes
    Print ", "
end for
Print "};"
```

Figure 4.6. Python script to convert .hex file to C code.

4.2. FPGA Firmware

Field Programmable Gate Array (FPGA) chips consist of programmable logic blocks that form hardware blocks inside the chip to run desired logic operations defined by the developer. Instead of traditional programming languages, FPGAs are programmed with hardware description languages such as Verilog and VHDL. Contrary to sequential programming languages (e.g. C, Java, Python), HDLs are consecutive languages due to their effect on the chip. An HDL defines blocks of hardware that run continuously without waiting other instructions or works to be completed. This feature of the FPGA chips allows them to perform many tasks in parallel, increasing the time efficiency of the system. In this section, design and implementation of FPGA firmware development with Verilog HDL will be explained in detail.

4.2.1. Overall Design

Figure 4.8 shows the main blocks of the FPGA design for the robots. Four identical controller blocks are dedicated to run the motors at the desired speed with

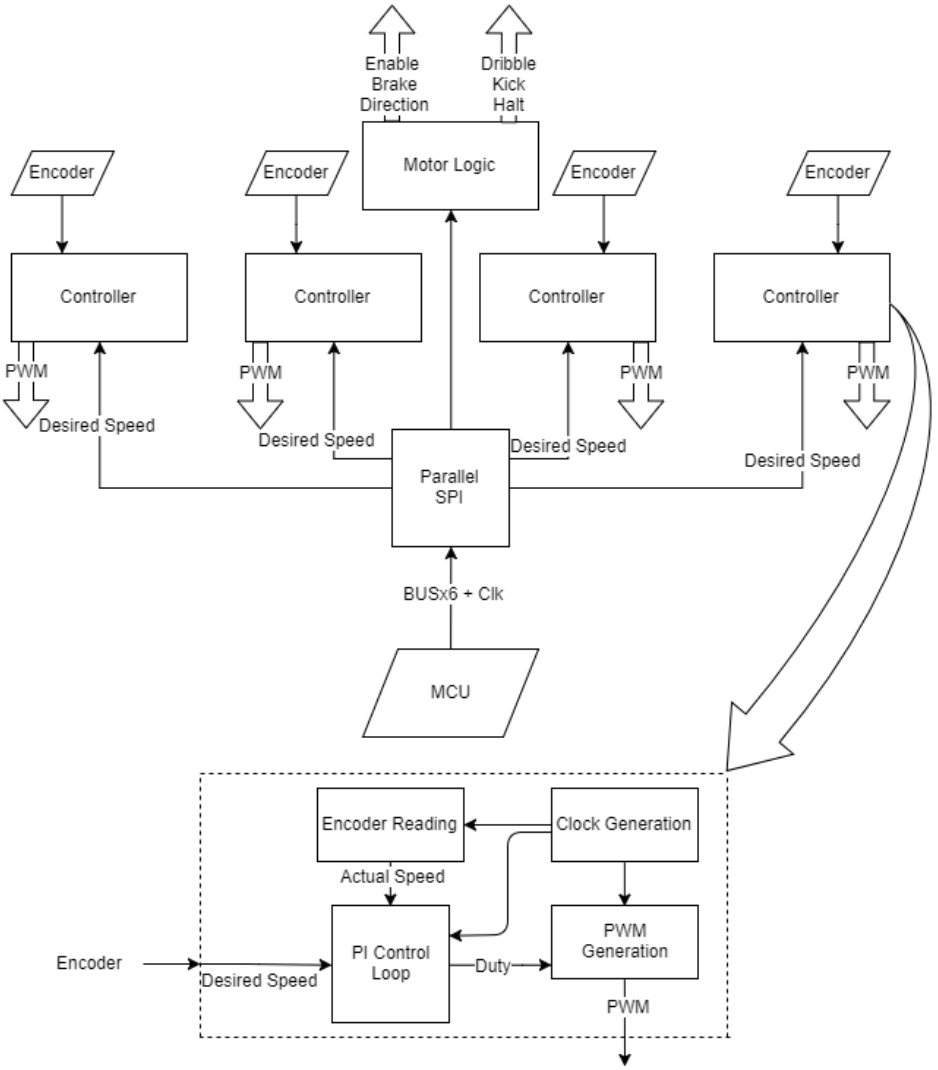

Figure 4.7. FPGA firmware block diagram.

the feedback from encoders connected to the wheels. The parallel SPI block takes the critical data from the MCU and sends it to the corresponding units. There is also a motor logic block which is responsible for sending digital data to five motors and the kicker circuit. Details of the blocks will be given in the following sections.

4.2.2. Parallel SPI Interface

In Section 4.1.5, microcontroller side of the SPI communication was explained. There are a six-pin data bus and one clock pin for the communication. The reason for transmitting the data in serial bits of each byte instead of series of parallel bytes is to

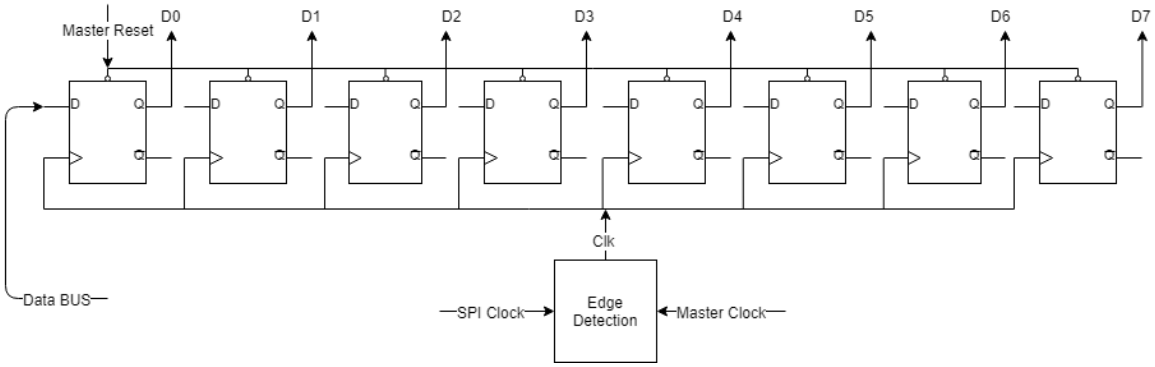

Figure 4.8. Shift register design for the parallel SPI.

reduce the design effort and complexity of implementation on the FPGA side. In this design, six 8-bit shift registers are designed. The shift registers are clocked by the SPI clock connection between the MCU and the FPGA. Since the critical data arrives to the MCU from the artificial intelligence computer, it is only logical to assign the MCU as the master of the communication interface for time efficiency.

Since each pin takes a serial byte data, the design is based on shift registers assigned to each bus pin. On each rising edge of the SPI clock, the bits on the bus are shifted into the registers, storing the bytes in each register so that the rest of the system can use this data until it is updated with the new communication data after 16 ms. One problem with this design is that external clock signals are not allowed by the Verilog. Therefore, the transition on the SPI clock must be sensed by an intermediate block that is clocked by the FPGA system clock. Therefore, the SPI clock must be slowed so that it can be detected from the FPGA. In order to achieve this a one-bit data is stored in the clock detector, that updates its value in every FPGA system clock and compares its value to the previous data. Then, the rising transitions are used to shift the bits through the register. In the MCU code, a small delay is added to the rising edges of the SPI clock in order to reduce clock frequency. Data transfer sequence takes 8 μs to be completed.

4.2.3. Clock Generation

The FPGA chip is clocked by an external 50 MHz MEMS oscillator as suggested in various applications. This is used as a master clock signal in the hardware, however, for certain tasks, different clock frequencies are necessary. First of all, the PI control loop must be completed at least 10 - 20 times for a given data package so that the desired speed value can be reached to its steady state without error. Therefore, a clock signal with a period of close to a millisecond must be generated from the master clock. A 16 bit up counter is implemented for this purpose that divides the system clock as given by

$$\begin{aligned} f_{PI} &= \frac{f_{sys}}{2^{16}} = 762.94Hz \\ T_{PI} &= 1.31ms \end{aligned} \tag{4.1}$$

Similarly, a clock for PWM output must be generated. For 50 kHz PWM frequency with a resolution of 1000 (10 bit PWM), the system clock can be fed into PWM generator, because each increment in the PWM pulse has 20 ns period for the given design requirement. Finally, an encoder count clear signal must be created. This clock signal is a rising edge pulse comes right after PI loop starts and finishes before an encoder signal can occur. This is done with a binary comparison operation.

4.2.4. Motor and Kick Logic

Apart from PWM speed reference, the motor driver also requires three logic signals for motor enable, brake, and direction. These values are decided in the MCU code and sent to the FPGA from the 6^{th} pin of the SPI bus. From the received data, the FPGA sends the necessary logic values to each motor driver. The kicker control block is also added to this block. The AI computer decides the type and speed of the kick and sends it to the FPGA over the MCU. The FPGA creates a pulse to the kicker MOSFETs to discharge the capacitors through kicker solenoids for passing and shooting.

4.2.5. Encoder Reading

One of the main reasons that the control of the motors are implemented on an FPGA is the difficulty of interruptive encoder reading. The motor feedback for the wheels are received from US Digital E4P optical encoder kits. The kit gives 360 cycles per revolution in two phases which results in 720 pulses and 1440 rising and falling edges per revolution. In a double-edge interrupt case, for the maximum speed of 128 pulses/ms, the four wheels of the robot has to process 512000 interrupts per second, which is impossible for any standard MCU. Therefore, this duty is taken from the MCU and implemented on the FPGA.

In order to design the encoder reading block, first of all, the two phases of the encoder are operated with an XOR gate to reduce them into a one pulse. Then, this pulse is used to clock an up-counter block with the limit of 255 in order to operate in a overflow-safe region. The counter value is registered in each cycle and the register value is outputted to the controller block. Therefore, this block needs to be synchronized with the PI controller module. This is achieved by the encoder clear count which is explained in the Clock Generation section. This way, after the PI controller module stores the data in the encoder counter, the value is cleared and ready to count the next cycle's actual speed value.

4.2.6. PI Controller

The most crucial part of the FPGA firmware design is the PI controller block. Although the theoretical background and actual design of the controller will be explained in Chapter 5, this section will show the design of the basic building blocks of the controller loop. The PI controller calculates the difference between the desired speed and the actual speed (error), and the integral of the error. Then, these are multiplied with K_I and K_P values and collected to give the system output in terms of duty cycle. These require signed and unsigned addition, multiplication, memory, and limiting blocks.

First of all, signed operations in FPGA chips are not trivial, especially when it is not certain whether all operands are signed or not. For example, desired speed and actual speed values are always positive and have fixed width. In this case, using a signed subtractor requires an additional sign bit for both operands. Moreover, the output of this subtraction (i.e. error) does not have to be positive. Therefore, this operation is done with unsigned 7-bit operands and 8-bit signed output where the signed output is to be used as operands in signed adders and subtractors. Similarly, duty cycle is always non-negative but its derivative ($dDuty$) can be either positive or negative. This part of the controller is designed in behavioral style with with sequential features in order to ensure accuracy.

Another important tool in the controller design is memory blocks because the controller algorithm does not only depend on the current error value but also on the previous loop's error and output values. Therefore, in order to store the parameters whose values affect the system, a memory block is designed. This block updates its content with the current value after every PI control cycle. Therefore, in the following loop, the stored value can be used as previous value which forms the basis of the concept of derivative in discrete time. The memory module is designed in behavioral style and based on a D-Flip-Flop based parallel load register.

Finally, limiter modules are designed for parameters that have to be positive and/or can overflow during the operation. For example, the duty cycle output is added to its previous value in each control cycle. Therefore, in a case where error stays positive when the duty cycle reaches its saturation, the next cycle will overflow the duty cycle into a value which is close to zero. This would cause oscillations in the motor control behavior. Similarly, negative overflow is also possible. Therefore, such values must be conditioned with a lower boundary limiter module. In this application, two limiter circuits are designed. The first limits the duty cycle while it is operated as signed value, and the latter limits the output of the block with minimum and maximum PWM values.

5. PI CONTROL ALGORITHM ON FPGA

Motor speed for each wheel is the main communication between the robots and the AI computer and it is crucial for the robots to comply with these orders. Closed loop motor control is an absolute necessity for the robots to operate properly because the motor load and therefore the motors' response to the reference input is rapidly changing. This dynamic feature of the system makes it impossible to use open loop control. It is only possible with feedback control to make sure that the wheel speeds are exactly at the desired level. In this chapter, theoretical basis of PI control and its tuning will be explained. Then, the FPGA implementation of the algorithm will be shown in detail.

5.1. PI(D) Control Basics

PID based control algorithms, despite the rapid development of automatic control, have usage over 90% in industry [30] mainly due to its ease of implementation especially on low level hardware and software. A PID controller consists of three parts, proportional, integral, and derivative. Each of these terms has certain effects and contributions on the plants' transient and steady-state behavior. In our robots, a PI controller is used for motor speed control. A PI and/or PID controller uses the process variable $y(t)$ (motor speed in our case) error ($e(t)$) with respect to the reference signal $r(t)$, and outputs the necessary controller output ($u(t)$) in order for the system to reach to the desired state. Figure 5.1 shows the block diagram of a closed loop control system. The controller transfer function denoted by $G_C(s)$ is given by

$$G_C(s) = K_P + \frac{K_I}{s} + K_D s \tag{5.1}$$

for a PID controller. The input to the plant can be written in time domain as

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{5.2}$$

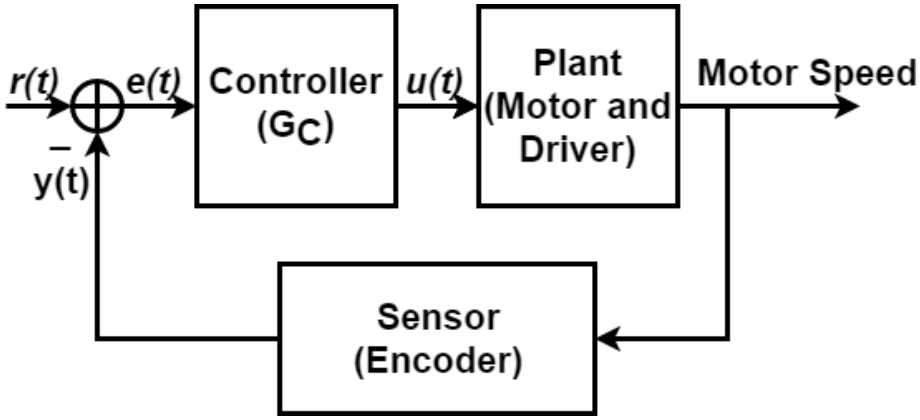

Figure 5.1. A generic closed loop control system.

where

$$e(t) = r(t) - y(t) \tag{5.3}$$

is the tracking error.

The first term of the PID controller, proportional term (P), outputs a certain multiple (K_P) of the error as the controller output. This is similar to on-off control method, where the control output varies between its maximum and minimum value depending on the direction of the error [31]. This type of controllers inherently gives steady state error and oscillation. In order to eliminate the steady state error, the integral term is necessary. The integral term (I), is the integral of the error (e) with a constant multiple (K_I). Since the integral of the error can have a nonzero value while the error is zero, the integral action can maintain the zero error state. As the output of the P controller is equal to zero when the tracking error is zero, the zero steady state error behavior is impossible with this type of controller unless the plant has a 1/s term, which itself is an integration action [32].

The derivative term, D, has the potential of great stability improvements on the plant. It anticipates the rate of change in the error and acts as a damper to the rapid changes output. Therefore, it can reduce the overshoot while allowing higher rise time

of the system response [32]. However, this term creates certain issues in practical cases especially when the reading of the sensing element is noisy. The derivative action amplifies the noise on the error with a rate proportional to the frequency of the noise [31]. This problem forces many practical applications to use PI control instead of PID for safer operation of the system and immunity against noise [30].

In our robots, the sensor reading of the motor speed is done via digital encoders, which means it is immune to electrical noise. On the other hand, the resolution of the encoder, however high it is, creates a quantization error and this can be considered as a white noise on the process variable reading [33]. The quantization error for motor speed (e_q) is calculated in (5.4). This error is caused by the 0.25° of uncertainty in the motor angle reading due to the resolution of the encoder.

$$\begin{aligned} e_q &= \frac{1}{T_{PI}(\text{ms})} \frac{\text{revolution}}{\text{pulse}} \frac{\text{ms}}{\text{minute}} \\ e_q &= \frac{1}{1.3} \frac{1}{1440} 60000 \approx 32\text{RPM} \end{aligned} \tag{5.4}$$

In the case of RoboCup SSL robots, the most important features of the control loop output are zero steady-state error, low rise time, and low overshoot. Therefore, a PI controller is decided to be used in the robots since it can provide the required features and avoids the instability caused by quantization noise calculated above.

Finally, as the PI algorithm is chosen, the following equation governs the controller to be implemented:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau \tag{5.5}$$

Since the FPGA cannot process the loop in continuous time, (5.5) can be converted to discrete time as,

$$u(k) = K_P e(k) + K_I T \sum_{n=0}^{k} e(n) \tag{5.6}$$

and this equation can be reformulated as,

$$u(k) - u(k-1) = K_P(e(k) - e(k-1)) + K_I T e(k) \tag{5.7}$$

which will be useful for the FPGA implementation of the algorithm.

5.2. FPGA Implementation of the PI Algorithm

Once the PI algorithm is decided to be used, it has to be implemented on the FPGA chip in a fast and robust manner. In the previous chapters, the reasons for using the FPGA for the control loop have been explained in detail. In this section, design process and considerations will be detailed.

Equation (5.7) forms the basis of the algorithm. In this algorithm, the output of the system depends on the previous values of the output and the error, current error, and PI parameters K_P and K_I. Therefore, in order to implement this equation, the main blocks required are memory for previous values and the four basic arithmetic operations. This equation can be implemented in one clock cycle, however, it would have a greater propagation delay than the sampling time of the PWM generation. Therefore, for one cycle of the PWM generation, the duty cycle of the reference PWM signal may take a random value resulting in peaks at the motor speed. These peaks can cause rapid changes in the error and change the steady state significantly. In order to avoid any glitches in the controller loop, a sequential controller design is implemented.

The controller loop is divided into six stages where different operations, which do not directly affect each other, are handled. In order to achieve this, a train of clock

pulses are generated with a period of 1.3 ms as calculated in (4.1). Each pulse follows each other and activates a different part of the process. The sequence is as follows:

(i) Last values of the error and the output are stored.

(ii) Error is calculated from desired and actual values.

(iii) The derivative of the error (error - previous error) is calculated.

(iv) Difference of the output is calculated with K_P, K_I, error, and the derivative of the error.

(v) The calculated change of output is added to the last output value.

(vi) The output value is limited to avoid overflow.

Moreover, a similar glitch avoid design is implemented on the PWM generation block with the help of a sampling stop signal generated in the clock generation module. In the clock generation module a signal whose value is zero during the PI control loop is generated and sent to the PWM generator. In this module, the sample stop signal is multiplied ("and") with the PWM clock signal so that the sampling of the register type duty cycle cannot occur during the PI control loop. The final precaution for glitches on the output is taken on the SPI communication block with the MCU. This block is added with a second stage buffer where the data is transferred only after the data transmission from the MCU is over. Therefore, the possibility of false data reading from the SPI due to sampling during the shifting of the bytes is avoided even though it is a very uncommon problem given that the transmission takes only 8 μs and it is done 60 times per second.

Finally, a master SPI module is designed for external data transmission. This module reads the encoder value before each PI loop and outputs it in a serial manner. In order to achieve synchronization, clock and data start signals are also added to the module. This module's output is read by an Arduino which transmits the data to a computer over the serial communication port. The encoder data reading is crucial to determine the K_P and K_I paramters of the controller.

5.3. PI Tuning

The PI control algorithm's operation and performance is dependent on its parameters, K_P and K_I. There are several methods for determining those parameters mathematically or practically. Mathematical methods are not feasible in many practical cases, because they require an accurate modeling of the plant and, more importantly, the noise over the measurement and the reference output. Therefore, even though the system gives great response on simulation, in the practical case, the parameters will need to be adjusted by trial and error. For such cases where the mathematical modelling of the system is unknown or overcomplicated to be obtained, there are experimental methods, such as Ziegler - Nichols rules, which are based on on-site data collection and fine tuning the parameters after a certain acceptable performance of the system is reached [32].

In this thesis, Good Gain Method is used in order to determine the controller parameters. This is a cookbook approach to the PI(D) tuning based on Ziegler - Nichols rules. The method aims to improve the stability of controller compared to the Ziegler - Nichols method. Moreover, it does not require the system to oscillate for data collection [34].

The method is applied to our system as follows:

(i) An average value of motor speed is chosen for tuning (60 pulses/T_{PI})

(ii) K_I is set as zero and K_P is set as 1.

(iii) K_P is increased until a small overshoot is observed along with the start of an undershoot.

(iv) The time difference between the overshoot and the undershoot is multiplied by 1.5 and set as integral time.

(v) Due to the introduction of the integral term, K_P is reduced to 0.8 of its original value.

(vi) Fine tuning is applied on the parameters until the desired system performance is reached.

The results of the PI tuning and system performance will be given in the next section.

Moreover, Ziegler - Nichols method was also used for tuning in order to compare its performance against Good Gain Method. However, due to the low inflection point value, the algorithm led to a very low K_I value (0.017). Therefore, the rise time of the motor speed increased dramatically compared to the results found from Good Gain Method.

5.4. Experimental Results

Once the PI control algorithm is tuned, K_I and K_P values are found as 0.21875 and 4.0000 respectively. Figures 5.4 to 5.5 show the results for encoder data from the wheels. In the recorded data, glitches due to mechanical problems are removed.

Due to mismatches in the mechanical bearings of the robot wheels, the load on the motor is not uniform during each revolution. Therefore, there are sudden increases or decreases in the wheel speed with the current mechanical production. This glitches cause the control algorithm to deviate from steady state and cause small oscillations in the steady state as it can be seen in Figures 5.4 to 5.5.

From the recorded data, the results in Table 5.1 are obtained for control algorithm performance.

Table 5.1. Control algorithm performance.

STEP INPUT	**RISE TIME**	**OVERSHOOT**	**SETTLING TIME (%5)**
120	31.2 ms	15.8%	97.5 ms
60	35.1 ms	16.7%	79.3 ms
30	35.1 ms	16.7%	88.4 ms

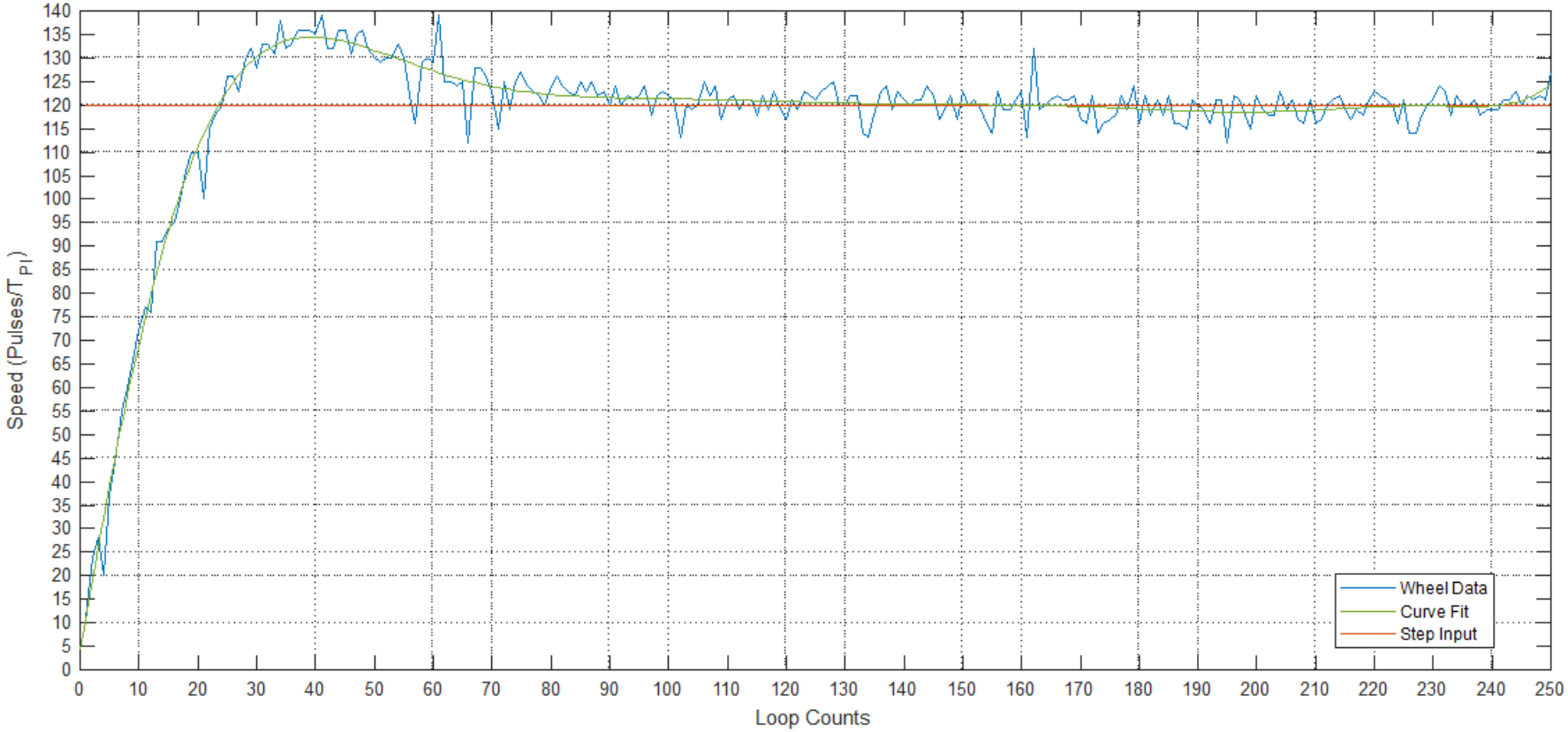

Figure 5.2. Wheel data for 120 pulses/cycle speed step input.

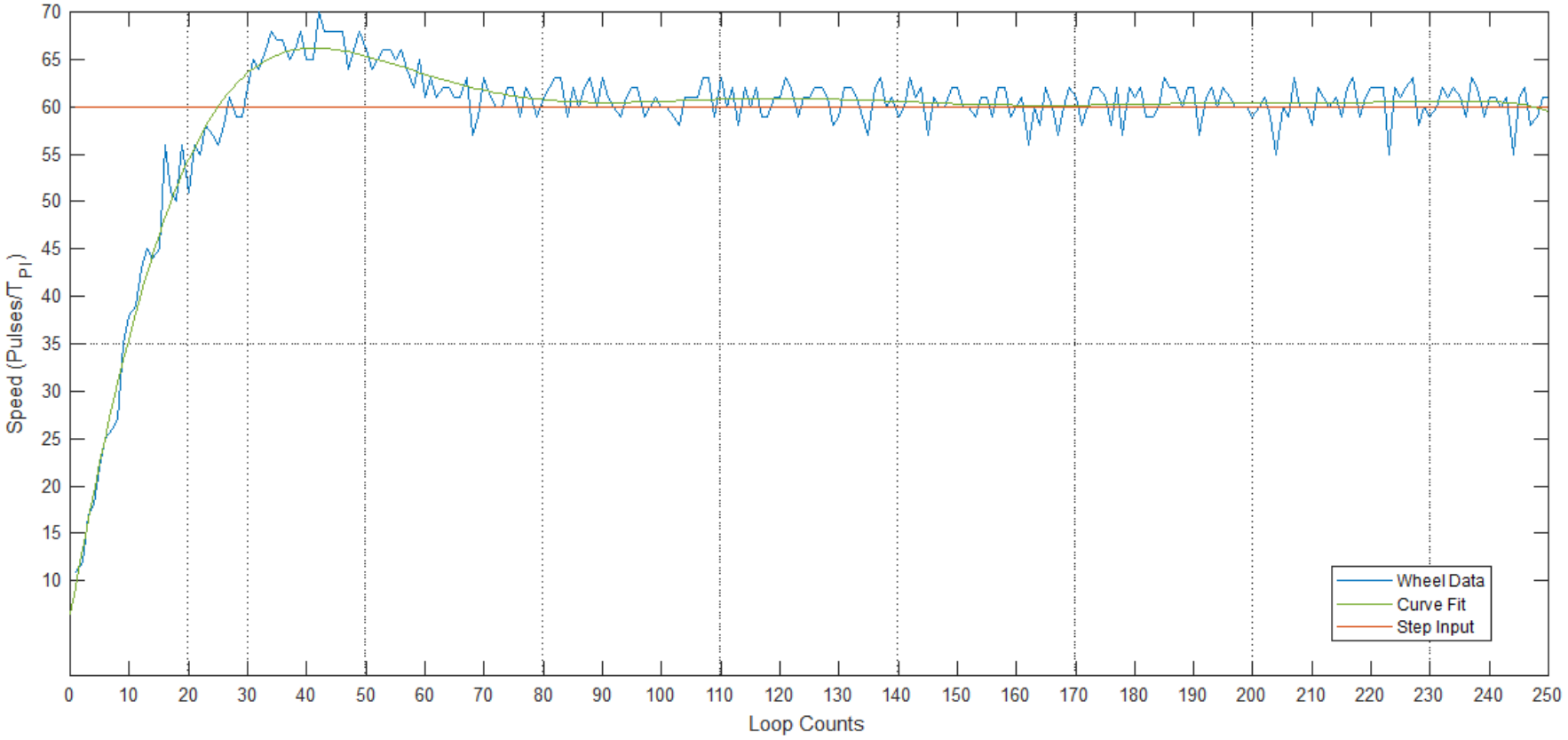

Figure 5.3. Wheel data for 60 pulses/cycle speed step input.

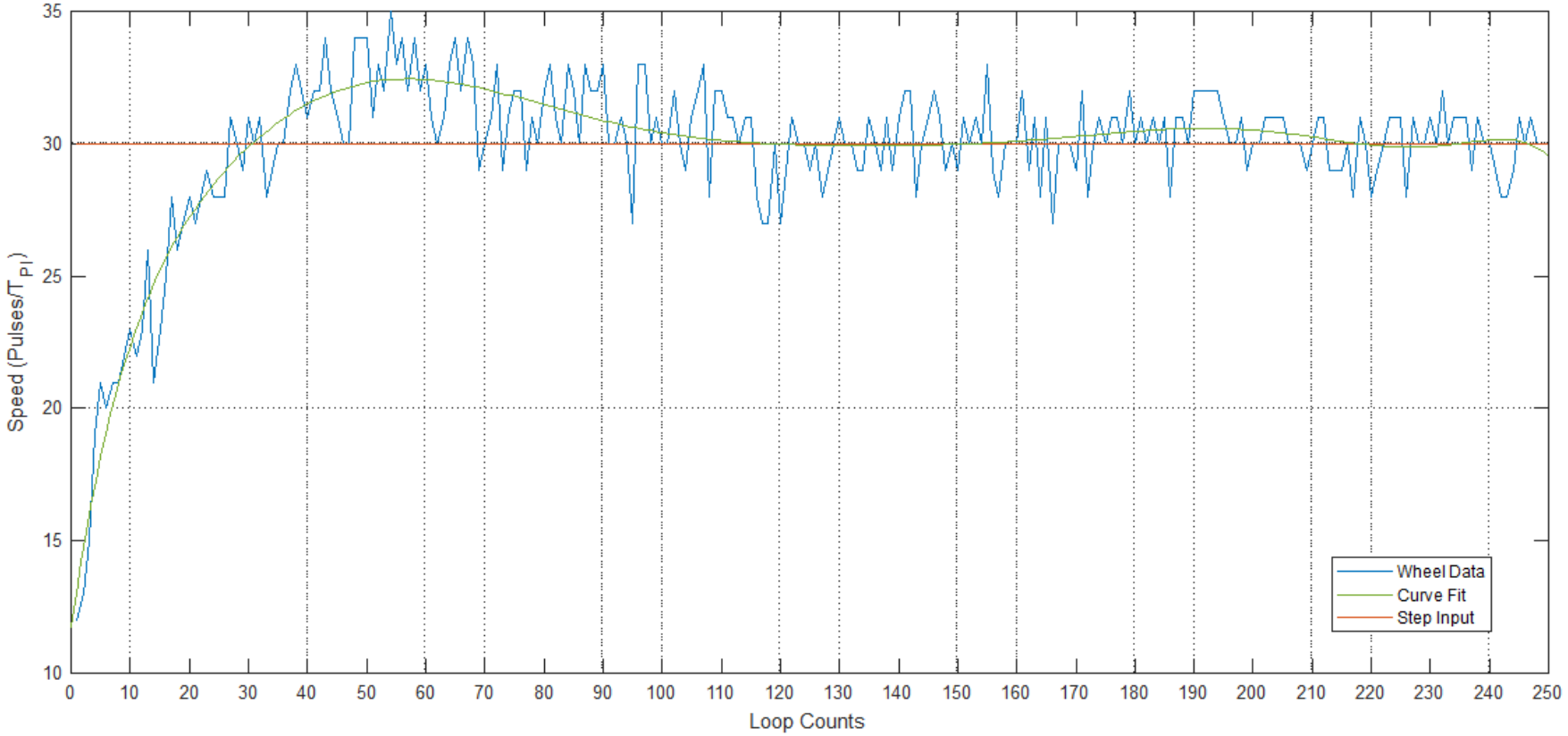

Figure 5.4. Wheel data for 30 pulses/cycle speed step input.

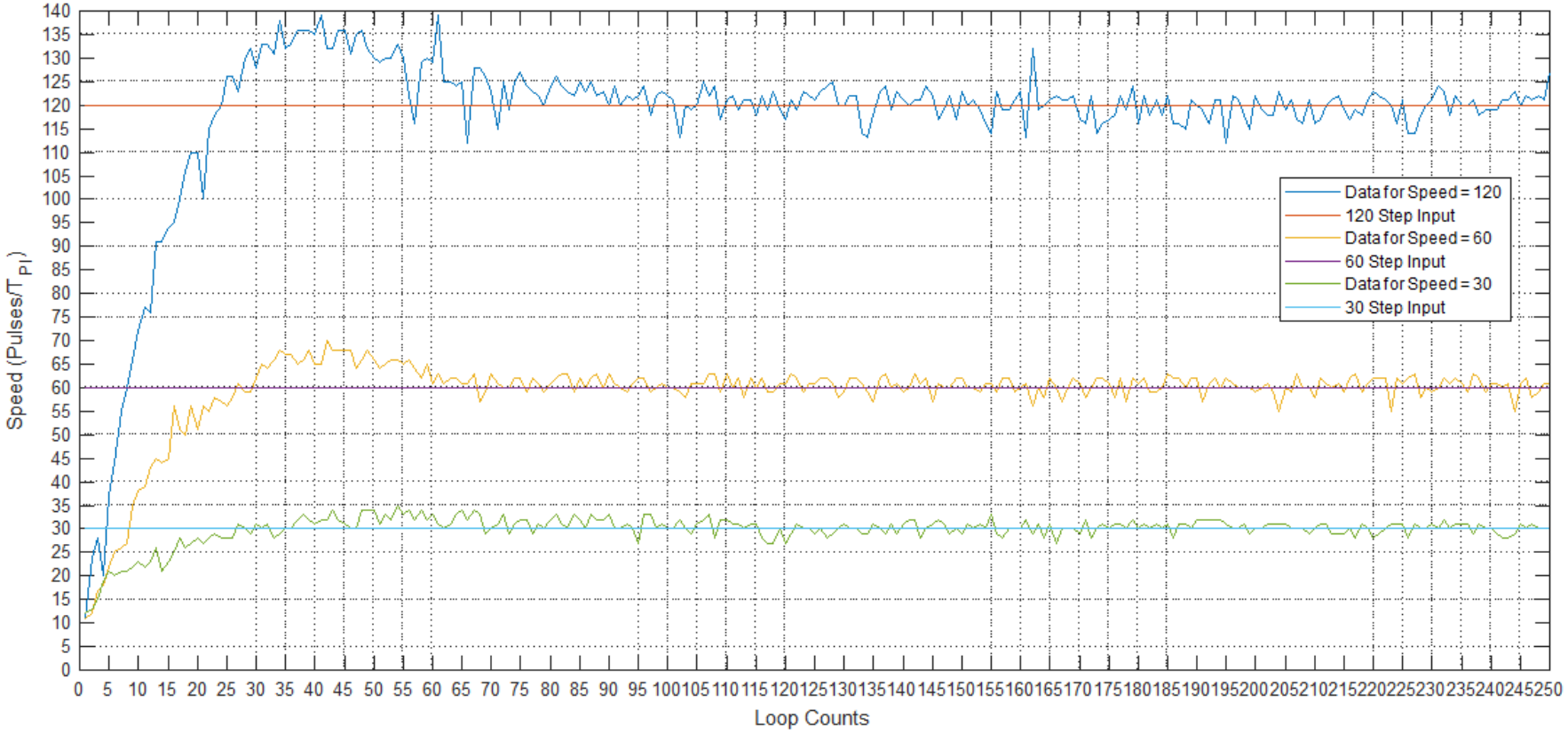

Figure 5.5. Wheel data for three different step inputs.

6. CONCLUSION

In this thesis work, RoboCup Small Size League and its requirements are introduced. The improvements in the state of art is presented with an emphasis on hardware and motor control methods. Then the need for a new set of robotic pool is explained along with the problems in the previous generations of robots. These problems include hardware design failures, which affect the motor performance especially due to crosstalk and control signal noise, and limitations of the processing units, which cause false readings of the encoders and AI computer signals as well as slow response time of the system.

In order to solve these problems and improve upon the previous generations, two main alterations on the system are proposed. These are separated motor driver circuit and double processing unit system which is based on cooperation of FPGA and MCU. Moreover, a modular design is suggested that can be improved to meet the continuously evolving requirements of the RoboCup SSL.

The hardware performance is enhanced by separating noisy motor drivers from the main board with an extra effort on noise reduction and crosstalk prevention. The motor driver circuits are supplied individually from the battery through noise isolated lines. A better grounding scheme is also implemented to reduce the noise on the control signals. Modular circuits that are mounted directly on the motors also increased the efficiency of space usage. This feature is of great importance since the robots are strictly size limited [6].

Another improvement on the hardware design is the FPGA - MCU cooperation. In this thesis work, advantages and disadvantages of both processing units are explained and a double - processing unit system design is proposed. In this design, parallel processing power of FPGAs is combined with the analog operation and peripheral support of microcontroller units. The duties such as communication, encoder reading,

and motor control are distributed between the FPGA and the MCU chips according to their stronger aspects. In this design, the MCU's main duty is communicating with the AI computer and interpreting its orders while these orders are immediately transmitted to the FPGA chip where encoder reading and control algorithm run in parallel for each motor. The results of the control algorithm are given in Chapter 5.

Moreover, a fully modular and improvable design is implemented for the robots. In this design, there are options for further developments of the main board which support daughter boards with the capabilities of FPGA module interfaces, analog readings, sensor communications via I^2C, UART communication, and so on. Therefore, the robots are able to be altered in order to catch the changes in the state of art of the RoboCup SSL robots.

Finally, development of low level firmware and motor control algorithm is presented. The firmware on both FPGA and MCU chips are designed to be fast, precise, and safe. Therefore, the performance loss due to delay in the process of implementing control algorithms starting with AI computer data reception is aimed to be reduced. Moreover, in order to prevent collisions of the robots and undesired behavior, several safety measures are taken such as emergency stop in case of data transmission failure and glitch removal modules on FPGA implementation.

To sum up, a new generation of robots is designed and implemented in this thesis work. These robots are compatible with the RoboCup SSL rules and they offer higher performance with respect to its predecessors. The robots also provide possibility of further development to keep up with the state of art.

REFERENCES

1. "RoboCup Official Website", `www.robocup.org/objective`, accessed at October 2017.

2. "RoboCup SSL Official Website", `http://www.robocup.org/leagues/7`, accessed at October 2017.

3. Weitzenfeld, A., J. Biswas, M. Akar and K. Sukvichai, "Robocup small-size league: past, present and future", *Robot Soccer World Cup*, pp. 611–623, Springer, 2014.

4. Biswas, J., J. P. Mendoza, D. Zhu, S. Klee and M. Veloso, "CMDragons 2014 Extended Team Description", *Proceedings of the 18th RoboCup Symposium, Brazil*, 2014.

5. "RoboCup SSL Wiki Page", `http://wiki.robocup.org/Small_Size_League`, accessed at October 2017.

6. Committee, S. S. L. T., "Laws of the RoboCup Small Size League 2017", `http://wiki.robocup.org/Small_Size_League`, accessed at October 2017.

7. Zickler, S., T. Laue, O. Birbach, M. Wongphati and M. Veloso, "SSL-vision: The shared vision system for the RoboCup Small Size League", *Robot Soccer World Cup*, pp. 425–436, Springer, 2009.

8. Esen, H., *Hardware and software development for RoboCup SSL robots*, Master's Thesis, Boğaziçi University, 2013.

9. Sharbafi, M. A., A. Azidehak, M. Hoshyari, O. Bakhshandeh, A. A.-M. Babarsad, A. Zareian, D. Esmaeely, A. Ganjali, S. Esmaeelpourfard, S. Ziadloo *et al.*, "MRL Extended Team Description 2011", *Proceedings of the 15th international RoboCup symposium, Istanbul, Turkey*, pp. 1–29, 2011.

10. Develer, Ü., H. Oksüz, G. Turan and M. Akar, "BRocks 2016 Team Description", *Proceedings of the 20th RoboCup Symposium, Leipzig, Germany*, 2016.

11. Niknejad, M. R., S. A. S. Neyshabouri, S. A. GhaziMirSaeed, E. Kamali, M. H. Fazeli, Y. Piran and M. Khouzani, "Immortals 2011 Team Description Paper", *Iran University of Science and Technology*, 2011.

12. Watjanathepin, N., N. Eawsakul, M. Puangpool, A. Namahoot and S. Yimman, "Implementation of PI Controllers with the FPGA", *ProcICCAS2003, October*, Vol. 22, p. 25, 2003.

13. Mahmood, S. S., M. S. Croock and M. Y. Hassan, "Design of FPGA Based P/PI/PD/PID Controllerfor Industrial Applications FPGA P/PI/PD/PID", *Journal of Engineering and Sustainable Development*, Vol. 10, No. 3, pp. 135–147, 2006.

14. Wilamowski, B. and J. Irwin, *Control and Mechatronics*, ENGnetBASE 2015, CRC Press, 2016.

15. Miller, T., *Brushless permanent-magnet and reluctance motor drives*, Monographs in electrical and electronic engineering, Clarendon Press, 1989.

16. Pillai, N. S., V. A. M and R. Radhakrishnan, "Analysis and simulation studies for position sensorless BLDC motor drive with initial rotor position estimation", *2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE)*, pp. 1–6, Jan 2015.

17. Xia, C., *Permanent Magnet Brushless DC Motor Drives and Controls*, Wiley, 2012.

18. Sen, P., *Principles of Electric Machines and Power Electronics, 3rd Edition: Third Edition*, Wiley Global Education, 2013.

19. Aiguo, Z., Y. Lang and P. Qiangbiao, "Rotor position detection and start-up methods of a sensorless BLDC motor", *2015 IEEE Advanced Information Tech-*

nology, Electronic and Automation Control Conference (IAEAC), pp. 1092–1096, Dec 2015.

20. ST Microelectronics, *DMOS driver for 3-phase brushless DC motor*, 10 2014, rev. 3.

21. ON Semiconductor, *Brushless DC Motor Controller*, 5 2014, rev. 9.

22. Texas Instruments, *12 V to 24 V, Three-Phase, Sensorless BLDC Motor Driver*, 7 2014, rev. E.

23. Varol, Ö., O. Cihan, H. Esen, A. Haseltalab and M. Akar, "BRocks 2013 Team Description", *Proceedings of the 17th RoboCup Symposium, Eindhoven, The Netherlands*, 2013.

24. Freescale Semiconductor, *MC9S08DZ128 Series Datasheet*, 7 2015, rev. 2.

25. Freescale Semiconductor, *MC56F847XX Data Sheet*, 6 2014, rev. 3.1.

26. Moongilan, D., "PCB to chassis grounding graphic architectures for minimizing radiated emissions", *2006 IEEE International Symposium on Electromagnetic Compatibility, 2006. EMC 2006.*, Vol. 1, pp. 47–52, Aug 2006.

27. Brander, T. and W. E. eiSos GmbH & Co. KG (Waldenburg), *Trilogy of Magnetics: Design Guide for EMI Filter Design, SMPS & RF Circuits*, Würth Elektronik GmbH & Company KG, 2009.

28. Xilinx, *Spartan-6 FPGA Data Sheet*, 1 2015, v3.1.1.

29. Xilinx, *Spartan-6 FPGA Configuration, UG380*, 3 2017, v2.10.

30. Ang, K. H., G. Chong and Y. Li, "PID control system analysis, design, and technology", *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 4, pp.

559–576, July 2005.

31. Visioli, A., *Practical PID Control*, Advances in Industrial Control, Springer London, 2006.

32. Ogata, K., *Modern Control Engineering*, Instrumentation and controls series, Prentice Hall, 2010.

33. Balakrishnan, A., *Quantization Noise Analysis of a Closed-Loop PWM Controller That Includes* Σ-Δ *Modulation*, Master's Thesis, Missouri University of Science and Technology, 2013.

34. Haugen, F., "The Good Gain method for simple experimental tuning of PI controllers", *Modeling, Identification and Control*, Vol. 33, No. 4, pp. 141–152, 2012.