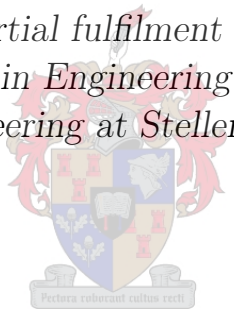
Robocup Small Size League: Active ball handling system

by

Daniël Gideon Hugo Smit

Thesis presented in partial fulfilment of the requirements for the degree of Master in Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University



Department Mechanical and Mechatronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602.

Supervisors:

Prof. K. Schreve
Dr. M. Blanckenberg

March 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2013/12/01 .....

Abstract

The RoboCup offers a research platform to advance robotics and multi-robot cooperation in dynamic environments. This project builds on work previously done to develop a research platform for multi-robot cooperation at Stellenbosch University. This thesis describes the development of an active ball handling system for a robot in the RoboCup Small Size League (SSL). This was achieved by building on the work done in the previous projects.

The hardware for the kicker and dribbler mechanisms on the robot were implemented and tested to characterise their capabilities. The kicker was characterised to control the speed at which a ball is kicked and the dribbler for optimal control over a ball. More accurate movement was required and the Proportional Integral and Derivative (PID) controllers for translational and rotational movement on the robot were improved. The test results show an improvement in straight line trajectory tracking when compared to those of the previous controllers. Dribble control sensors were implemented on the robot for successful dribbling by the robot. This resulted in a significant improvement to the dribbling ability of the robot when these sensors are used. This dribbling ability was compared to the dribbling ability of the robot when no feedback was received from the sensors. Lastly a proposed curved trajectory tracking algorithm was tested by combining translational and rotational movement of the robot. This algorithm showed the capabilities of the robot to follow a curved trajectory with the improved translational and rotational controllers.

Uittreksel

Die RoboCup bied 'n navorsingsplatvorm om robotika en multi-robot samewerking in 'n dinamiese omgewing te bevorder. Hierdie projek bou voort op werk wat reeds gedoen is om 'n navorsingsplatvorm vir multi-robot samewerking aan die Universiteit van Stellenbosch te ontwikkel. Hierdie tesis beskryf die ontwikkeling van 'n aktiewe balhanteringsstelsel vir 'n robot in die RoboCup Klein Liga (KL). Dit is bereik deur voort te bou op die werk wat in vorige projekte gedoen is.

Die hardeware vir die skopper- en dribbelmeganismes is geïmplementeer en getoets om hulle vermoëns te karakteriseer. Die skopper is gekenmerk deur die spoed waarteen 'n bal geskop word en die dribbler vir optimale beheer oor 'n bal. Meer akkurate beweging was nodig en die PID-beheerders vir translasie- en rotasiebeweging in die robot is verbeter. Die resultate van die toetse toon 'n verbetering in reguitlynbeweging in vergelyking met dié van die vorige beheerders. Dribbelbeheersensors is in die robot geïmplementeer vir suksesvolle dribbelbeweging deur die robot. Gevolglik is daar 'n aansienlike verbetering in die dribbelvermoë van die robot wanneer hierdie sensors gebruik word. Hierdie dribbelvermoë is vergelyk met die dribbelvermoë wanneer die robot geen terugvoer van die sensors ontvang nie. Laastens is 'n voorgestelde algoritme vir die robot om 'n geboë trajek te volg, getoets. Dit is bereik deur die translasie- en die rotasiebeweging van die robot te kombineer. Hierdie algoritme het die vermoë van die robot om 'n geboë baan te laat volg deur gebruik te maak van die verbeterde translasie- en rotasiebeheerders.

Acknowledgements

I would like to express my sincere gratitude to the following people and organizations:

- The CSIR for providing funding for this wonderful research opportunity.
- Lorita Jacobs and David Holtzhausen for assisting me with understanding the work they had done on the project.
- Mr. Ferdie Zietsman and the workshop personnel for their advice and assistance during the project.
- Reynaldo Rodriguez who was always willing to help and provide me with technical assistance.
- My supervisors, Prof. Kristiaan Schreve and Dr. Mike Blanckenberg for their support throughout this project.
- Michelle Marias for supporting me throughout this year and keeping me motivated up until the end.
- Lastly I would like to thank my parents and two sisters for their full support and motivation throughout my postgraduate studies.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CO	Controller Output
I^{2}C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
IR	Infrared
GPIO	General Purpose Input/Output
GPL	General Public License
GUI	Graphical User Interface
HLC	High Level Controller
FPGA	Field Programmable Gate Array
KF	Kalman Filter
KL	Klein Liga
LLC	Low Level Controller
MARIE	Autonomous Robotics Integration Environment
MS	Measured Speed
MSL	Middle Size League
OFC	Off-field Computer
PI	Proportional and Integral

PID	Proportional Integral and Derivative
PWM	Pulse-width Modulation
ROS	Robot Operating System
SBC	Single Board Computer
SP	Set Point
SPI	Serial Peripheral Interface Bus
SSL	Small Size League
XDR	External Data Representation

1. Introduction

1.1. Background

Robotics is the design and development of a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer (Oxford, 2012). Furthermore robotics forms a very big part of modern technology and includes, but is not limited to, manufacturing of appliances, assembling production cars and robots that are used in space. Robotic systems require intensive research in order to improve and make it increasingly automated.

This project builds on previous work done in developing a mobile robotics research platform at Stellenbosch University. The goal of this research is to create a team of robots that can manoeuvre in a dynamic environment and therefore to develop a team of robots that would be able to participate in a division of the RoboCup.

The RoboCup is a competition in which Artificial Intelligence (AI) and advanced robotics are researched in a friendly competition environment. The competition is based on the rules of soccer and the related research is aimed at multi-robot and multi-agent cooperation in a very dynamic environment.

The robots (players) should be fully autonomous and no human interaction is allowed during the match (RoboCup, 2012*b*). Due to the quick nature of the game it is required to have real time sensing together with proficient and immediate decision making to perform the actions. This motivates why robot soccer is an excellent research platform. A closer look at all the individual actions performed in soccer, like moving towards a ball, dribbling a ball and scoring a goal, is just some of the research objectives. Other actions would be to reach an end destination accurately, with the correct orientation. If a robot is to direct a ball along a desired path it must be able to approach the ball with the correct orientation.

1.2. RoboCup Leagues

The RoboCup Soccer competitions have different leagues that consider different aspects of robotics in order to achieve the research goals. The research

goals include cooperative multi-robot and multi-agent systems in a dynamic environment where all the robots are fully autonomous (RoboCup, 2012*b*). The research goals are achieved throughout the five different RoboCup soccer leagues. These leagues include the Simulation League, Small Size League SSL, Middle Size League (MSL), Standard Platform and the Humanoid League. The research done in this project only focus on the SSL. More information on the other leagues can be found on the Robocup website (RoboCup, 2012*b*).

1.3. SSL League

The SSL consists of two teams, each with six players and every player should fit within a cylinder of 180 mm in diameter and 150 mm in height. Each individual team on the field is assigned a colour, either yellow or blue. This is done by round marker stickers in the centre, on top of each robot in the team. Each robot then has four round markers around the centre marker with different colours as shown in Figure 1.1, in order to give each robot its own identity. Figure 1.1 shows all the different colour assignments for the blue team.

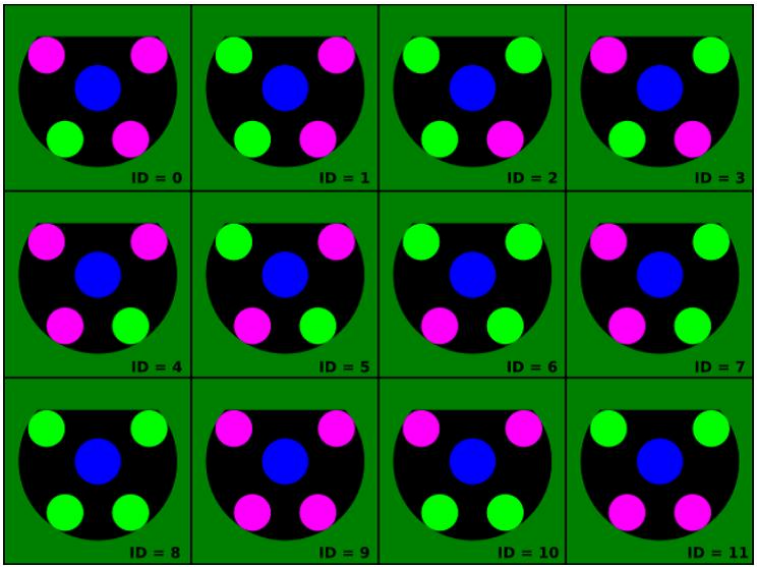


Figure 1.1: Standard colour assignments to robots (RoboCup, 2012*a*)

An orange golf ball is used as the match ball. The dimensions of the field are 6.05 m by 4.05 m. The robots make use of wireless communication and are tracked using two overhead cameras, each covering the view of one half of the field. A team has its own Off-field Computer (OFC) used for high level control and a referee box is used to send its commands to each of the teams' off-field computers. This configuration is shown in Figure 1.2.

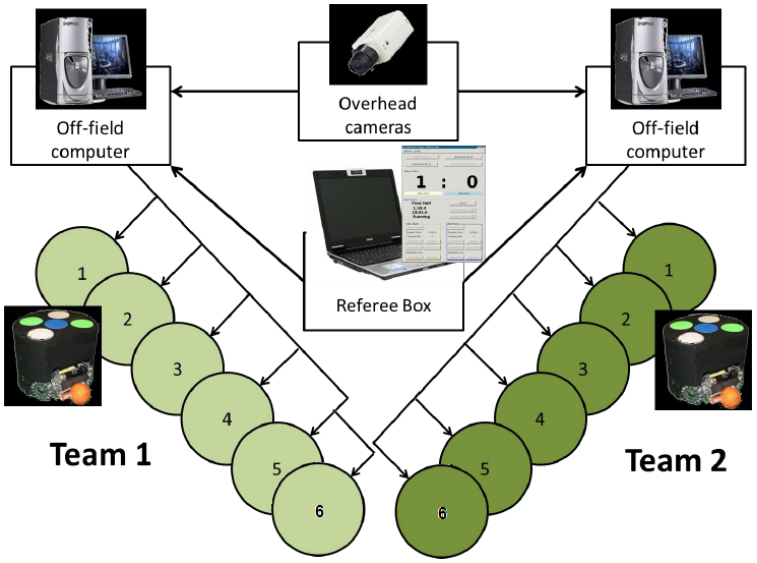


Figure 1.2: Setup of centralised control

The robots are allowed to be equipped with dribbling devices that constantly exert backspin on the ball under certain conditions. According to the rules, the force should be exerted perpendicular to the plane of the field and it is also not allowed to have vertical or side dribblers. The rule of the dribbling device is indicated in Figure 1.3.

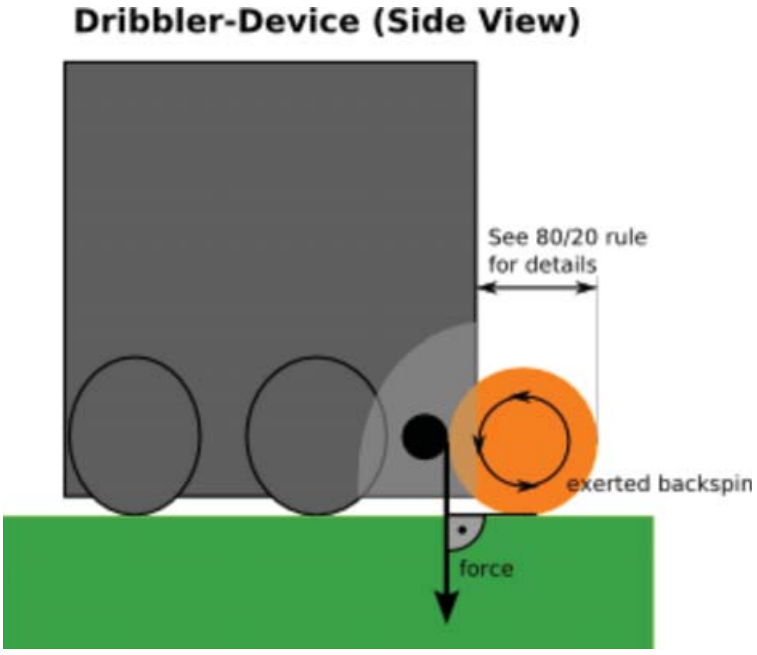


Figure 1.3: The side view of the allowed dribbler device (RoboCup, 2012*a*)

Another rule that should be considered when approaching the current project is that the robot having control of the ball, is not allowed to "swallow" the ball. According to the RoboCup rules, the robot in control of the ball is not allowed to constrain all the degrees of freedom of the ball. Other players should be able to remove the ball from the robot in control of the ball and when viewed from above, 80% of the area of the ball should be outside the convex hull around the robot (RoboCup, 2012*a*). This is shown in Figure 1.4.

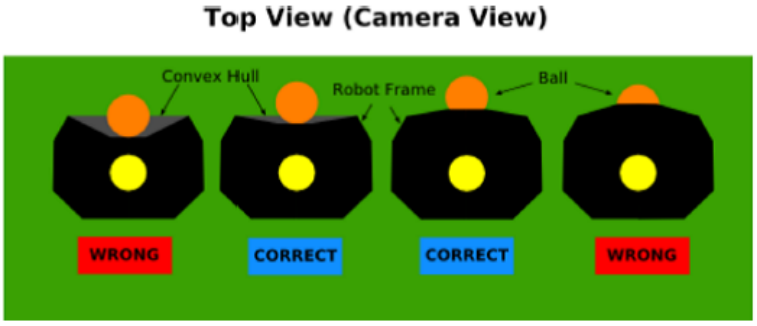


Figure 1.4: The correct rules for covering and holding the ball (RoboCup, 2012*a*)

The main focus of the SSL league is on the problem of multi-robot/agent cooperation and control of the robots in a highly dynamic environment with some sort of centralised or distributed control system (RoboCup, 2012*a*). Distributed control is a combination of centralised and decentralised control. The SSL robots usually have more centralised control meaning that the robots have little processing power and most of the calculations are performed on the OFC on which commands the robots rely (RoboCup, 2012*a*). The reason for this is mainly due to size limitations. The robots in the SSL league typically have a main controller, motor controller, kicker and dribbling mechanism on its chassis. The main controller is used to communicate with the server computer and motor controller. The motor controller receives commands from the main controller to control the wheels. The kicker and dribbling mechanisms are used to perform the play actions and the chassis to attach the wheels and the rest of the robot.

The flow of data between the cameras, the OFC and the robots is shown in Figure 1.5. Both teams make use of these two cameras by using an open source vision system (SSL Vision) that provides the information of all objects on the field to the OFC of each team. Merging of data of the individual cameras focussing on each half has to be done by the teams to compile a view of the field as a whole.

The soccer game is fully autonomous except for the referee and assistant referee inputs to the referee computer. The software together with the hardware of the robot is used to perform the individual actions in the game. Decisions

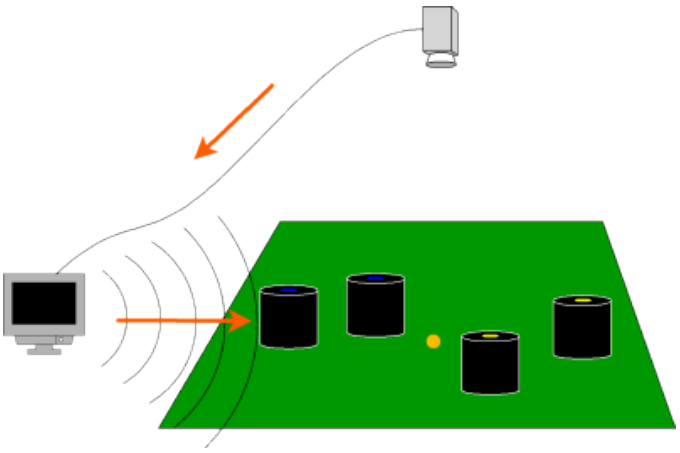


Figure 1.5: Flow of data from the overhead cameras (RoboCup, 2012*a*)

on which actions to perform are controlled with the AI on the server computer of each team. A good understanding of the hardware and the limitations is needed to perform these game play actions in a successful manner.

1.4. SSL Robots at Stellenbosch University

The history of the SSL robots at Stellenbosch University started with Smit (2011) who developed the first iteration robot. This robot had four omnidirectional wheels with a motor controller for each wheel. Smit (2011) chose the Roboard as main controller for the SSL robots. The robot of Smit (2011) was however not within the size limitations of the RoboCup rules and a second iteration robot was designed by Reynaldo Rodriguez, a technician who did the mechanical design to comply with the rules of the competition.

Jacobs (2011) continued with the second iteration robot and designed and manufactured a new single motor controller circuit to control the four individual wheels on the robot. Holtzhausen (2012) built on the work of Jacobs (2011) and improved the robot control in the x- and y-direction by making use of distributed control.

All the previous work done was mainly focussed on robot movement and control in x- and y-direction. The other mechanics like the dribbler and kicker device on the robot was not yet implemented or tested. The previous work done only focussed on straight line movement and never looked at curved movement by the robot.

1.5. Objectives

This project involves the need to advance AI and robotics. The focus of this research will be mainly on the control of the ball when dribbling and kicking.

This project only focuses on effectively controlling one robot for now and later assembling a group of robots to play as a team. The following objectives will have to be met in order to reach the goal of this research:

1. The current distributed control system with the SSL-Vision and the combined on-board sensors for trajectory control has to be tested.
2. The distributed control system for accurate movement on the field needs to be improved.
3. The dribbling device on the robot has to be assembled and the dribbler device should be tested and modified in order to effectively gain control of the ball whilst obeying all the rules.
4. The dribbling device should be characterised to know the capabilities of controlling a ball and the material used on the dribbling bar should be chosen for the best friction force on the ball for backspin.
5. The kicker device should be assembled and implemented on the robot.
6. The kicker should be tested and modified in order to effectively kick the ball and the kicker device should be characterised to know the capabilities and limitations when kicking the ball on the field. So too the force with which the ball is kicked should be characterised so as to have control over a kick.
7. Sensors on the robot must be implemented for more accurate control when dribbling and kicking the ball. Then, test the capabilities of the robot and compare the results with the tests with no sensors for the dribbling and kicking of the ball.
8. Rotational movement control of the robot should be tested to combine translational and rotational movement. This will give the robot the ability to follow curved paths that was not possible till now.

1.6. Motivation

This project was started by Smit (2011) in 2010 and this is the fourth master's thesis on this project. Every thesis focussed on a different part of the bigger project building towards a fully autonomous team of robots that can compete in the RoboCup SSL league. The research and work done in this project is needed to continue working towards the end of the bigger project.

The aim of this project is to have a robot successfully gain control over the ball and to dribble it. The objectives mentioned should be accomplished to get a single robot to effectively perform the actions needed by soccer robot.

This forms an essential part in putting together a team of players. The characterisation and capabilities of a robot should be tested. This will help the AI system to make the correct decisions and give the correct commands to the right robot.

1.7. Document Outline

This document gives a report of the work done in this project to achieve the mentioned objectives. The document has the following structure. Chapter 2 gives background information about the robots in the SSL league and the literature needed to achieve the objectives of this project. Chapter 3 gives a hardware description of the robot. This chapter is divided into a mechanical and electronic description. Chapter 4 describes the tests and results of a test to characterise the kicker and a test to characterise the dribbler mechanism.

Chapter 5 discusses the software that was developed to perform the tests in this project. Thereafter, Chapter 6 describes the procedure that was followed to perform the tests. This chapter also gives the results and a discussion thereof. The research is summarised and concluded in chapter 7. Recommendations on possible future work is also given in chapter 7. The appendices follow after this. In appendix A the statistical data of the tests done in chapter 6 is given. Appendix B explains the coordinate transformations between the different axis system in the environment.

2. Background Information

2.1. Introduction

This chapter will discuss the literature related in achieving the objectives of this project. The robots used in the RoboCup SSL league make use of holonomic movement. The mechanical design and the way the wheels are fit onto the robot give it this holonomic characteristic. This allows a robot to combine translational and rotational movement with the aim of tracking curved trajectories. Holonomic movement will be discussed and also the equations used to control the movements of the robot.

After understanding holonomic movement and how it is used on robots, more complex movements can be achieved. One of the main uses is when a robot should reach an end destination while avoiding obstacles along the way. Trajectory tracking and path following will be discussed in this chapter.

The movements and actions performed by the robot are done with actuators and hardware on the robot. Communication with these actuators and hardware is done between the OFC and the robot through a central receiver in the robot. The OFC should communicate with this central receiver through middelware which provide the OFC with access to the hardware of the robot. Middelware and also the preferred middelware, Player, will be discussed in this chapter.

2.2. Holonomic Movement

In the robot soccer game it is essential for the robots to be manoeuvrable and move accurately. This can be achieved by making use of holonomic movement and it is also the preference for robots in the SSL league. Holonomic movement allows the robot to move in any direction without having to turn first. This is achieved by driving and controlling each wheel individually. It also allows a robot to move along a path by combining translational and rotational movement, which gives the robot three degrees of freedom, identified as two for translation motion and one for rotational motion. This is achieved by using three or more omnidirectional wheels on the circumference of the robot. Each wheel can move the robot forward and since they are located on the peripheral

of the robot, each wheel also contributes to rotating the robot about the center of the robot.

The desired velocities of each individual wheel needs to be calculated in order to control the x- and y-velocities, as well as the rotational speed of the robot. The axis of the robot and the arrangement of a holonomic robot with n wheels and the force distribution is shown in Figure 2.1. Each wheel can rotate with the use of a motor in the positive or negative direction of the force and translate freely in the perpendicular direction with the use of smaller wheels on the peripheral which rotate freely. Refer to section 3.2 for a more detailed description of the hardware design. The angle of each individual wheel is measured relative to the x-axis. The positive driving direction of the i-th wheel is $\theta_i + \pi/2$ and is also the positive force direction for each wheel. Although it is possible to achieve the desired movement with only three wheels, most of the current competing teams use four omnidirectional wheels for locomotion. More wheels mean more force that can move the robot and also more traction. Some robots use three wheels that are symmetrically arranged with a spacing of 120 degrees of intersection of the axis. For these robots the kinematic control equation is complex, because the translational and rotational motion driven by the three wheels is not decoupled, which requires complicated transformations for controlling the robot in each degree of freedom (Asama *et al.*, 1995). Robots that use three wheels only have three contact points on the ground, which make them less stable in dynamic motion due to the center of gravity being higher (Asama *et al.*, 1995). Robots that use four wheels for locomotion have one excess degree of freedom. This redundancy forces the control to be in the two dimensional space and coherent control of four wheels is required (Asama *et al.*, 1995). The robot used in this project use four omnidirectional wheels.

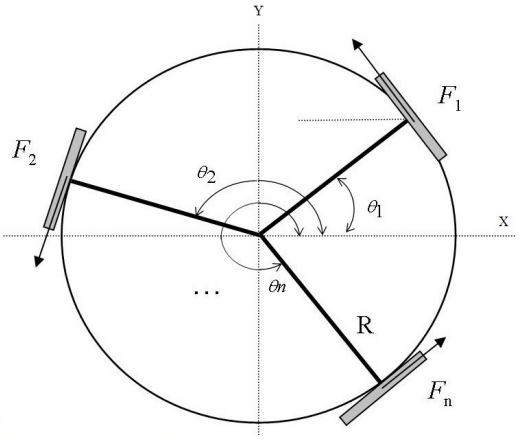


Figure 2.1: Holonomic robot with n wheels and force distribution (Rojas and Förster, 2006)

It is desirable to compute the speed of each individual wheel having the

translation and rotation speed of the robot in its coordinate frame. Let the translation speed of the robot in the coordinate frame of the robot be v_x and v_y and the rotational speed ω be counter-clockwise according to the axis in Figure 2.1. If the robot is required to translate 1 m/s in the positive x-direction then the speed of wheel one can be calculated as $-\sin(\theta_1)$ by looking at Figure 2.2. The same approach can be used when a desired forward speed of 1 m/s ($v_y = 1$) is required, the speed of wheel one will then be $\cos(\theta_1)$. The contribution of each wheel for the rotation speed of the robot is simply the required rotation speed multiplied by the radius of the robot.

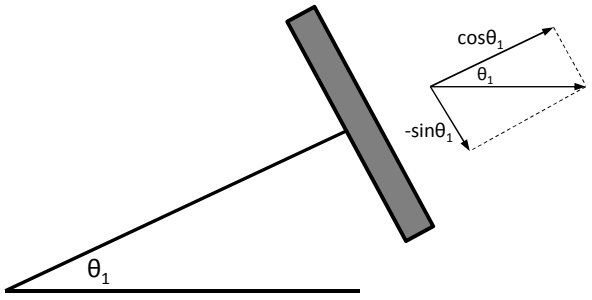


Figure 2.2: Speed of wheel one associated with 1 m/s in the robot x-direction

The individual wheel speed for an omnidirectional robot, with n wheels is related to the robot velocities, using the geometry of the robot and shown in equation 2.1,

$$\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} -\sin\theta_1 & \cos\theta_1 & R_r \\ \vdots & \vdots & \vdots \\ -\sin\theta_n & \cos\theta_n & R_r \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_r \end{bmatrix} \tag{2.1}$$

where v_1 to v_n are the wheel velocities and v_x, v_y and ω_r are the robot velocities.

2.3. Trajectory Tracking and Path Following for an Omnidirectional Mobile Robot

Mobile robots with omnidirectional capabilities have a significant advantage over non-holonomic mobile robots since the translational and rotational movement is controlled independently and give them the advantage to move in any direction without reorientation. Soetanto *et al.* (2003) addressed three main problems in the literature namely point stabilisation, trajectory tracking and path following.

In point stabilisation the goal is to drive a robot to a certain desired point with the desired orientation. In trajectory tracking the robot should track

a time parametrised reference path. With trajectory tracking it is assumed that there is perfect velocity tracking and the angular rotation of the robot is controlled to follow the time parametrised path. The robot should start on the desired path since the trajectory tracking algorithm does not account for errors in the initial conditions.

With path following the robot should converge to and follow a path (Soetanto *et al.*, 2003). In path following the vehicle tracks a desired velocity profile while the controller acts on the vehicle orientation to drive it to the path (Soetanto *et al.*, 2003). Smoother convergence to the path is achieved with this type of controller when compared to that of trajectory tracking (Soetanto *et al.*, 2003). A path following controller should look at the distance from the robot to the reference path, the angle between the robot's velocity vector and the tangent to the path and reduce both to zero without consideration in temporal specifications (Kanjanawanishkul and Zell, 2009), (Soetanto *et al.*, 2003). In path following the desired geometric path $p_d(s)$ is parametrised by the curvilinear abscissa s instead of time, which is used in the trajectory tracking problem (Kanjanawanishkul and Zell, 2009). This allows one to select a temporal specification for $s(t)$ which can be considered as an additional control input (Kanjanawanishkul and Zell, 2009). The stringent initial condition constrains present in trajectory control can be overcome by controlling the rate of progression ($\dot{s}$) of a "virtual robot" that is tracked by the "real robot" (Soetanto *et al.*, 2003), (Kanjanawanishkul and Zell, 2009), (Lapierre *et al.*, 2007). The errors in distance and orientation between the "virtual robot" and the "real robot" are then reduced to zero by the controller. Singularities occur when the robots current location is at the center of curvature of the path "virtual robot" (closest point not unique) but global convergence to the path is achieved with the "virtual robot" approach discussed by the authors above (Aicardi *et al.*, 1995).

Egerstedt *et al.* (2001) discuss a strategy that only requires control in the lateral direction, i.e. rotational control, while keeping the longitudinal velocity constant value. This strategy is also based on the "virtual robot" approach, where the motion of the reference point is governed by a differential equation containing error feedback (Egerstedt *et al.*, 2001).

2.4. Middleware

The robots used in the SSL league need to communicate with the server computer and receive commands from the server computer. This is done over a wireless network. The robots also have different sensors and actuators on them to perform the tasks needed in the game of soccer. These sensors and actuators have different communication mechanisms and middleware is used to provide a common communication platform. Middleware should support the coupling of subsystems on the robot. Smit (2011) discussed the middleware

in much more depth and also elaborated on the choice of middleware used for this project. Smit (2011) discussed the Robot Operating System (ROS) and also Autonomous Robotics Integration Environment (MARIE) as well as The Player Project. Smit (2011) motivates why The Player Project was the best choice of middleware for this project. Holtzhausen (2012) researched more middleware solutions based on research done by Kramer and Scheutz (2007). Holtzhausen (2012) explained that the choice of The Player Project is still the best choice of middleware for this project.

2.5. Player

Programming robots is time-consuming and can sometimes be complicated especially when working with multiple robots on the same network (Gerkey *et al.*, 2003*b*). The Player/Stage Project provides Open Source tools that simplify controller development for multiple robots with distributed control (Gerkey *et al.*, 2003*b*).

2.5.1. Player/Stage

The Player/Stage Project began in 1999 and has since been adopted, improved and extended with the newest version Player 3.0.2 released on 28 June 2010. This is the same version that Smit (2011) started with when the first iteration robot of Stellenbosch University was developed and also the version used with the current robots of Stellenbosch University. The Player/Stage Project consists of two parts namely the Player server and the Stage multiple robot simulator in a 2-D bitmapped environment.

Player is a network server running on the robot and provides a simple interface to the sensors and actuators of the robot over the IP network (Gerkey *et al.*, 2003*a*). Player has existing device drivers for most of the popular robot hardware that can be used (Collett *et al.*, 2005). Player runs on Linux and is also installed on the central controller (OFC) that uses a client to communicate with each individual robot that has the Player drivers on it. The client program communicates with Player over a TCP socket to send commands and read data from the robots that are connected to the network. The choice of programming language (that supports TCP sockets) is up to the designer and client-side utilities available in C++, Tcl, Java and Python (Gerkey *et al.*, 2003*a*). Player also allows any client to connect with another client to read sensor data or give commands to the other client (Gerkey *et al.*, 2003*a*). Before sensor readings can be done the client must open the appropriate device with read access and before controlling an actuator the client must open the appropriate device with write access (Collett *et al.*, 2005).

The Player library is divided into the core layer and the transport layer (Collett *et al.*, 2005). The core layer then consists out of the core library

and the driver library and provides the core Application Programming Interface (API) and functionality of the Player system (Collett *et al.*, 2005). The core system is a queue-based message passing system where every driver has a single incoming message queue and can publish messages to the incoming queue of other drivers and to the clients that have requests (Collett *et al.*, 2005). The core library coordinates the passing of these messages and defines the message syntax. The structure of these messages consists of four values: host, robot, interface and index which allow inter-server subscriptions (Collett *et al.*, 2005). In Player the TCP layer consists of a library that handles the TCP communication and an External Data Representation (XDR) library that handles data representation (Holtzhausen, 2012). The TCP library moves the messages between TCP sockets and Player device queues. The XDR specifies a platform-independent encoding for commonly used data types, as well as integers and floating point values (Collett *et al.*, 2005).

The player architecture was chosen by Holtzhausen (2012) and is shown in Figure 2.3. The client is used by the central controller to communicate with the robots which have the Player drivers on them (Holtzhausen, 2012). This is done with the Player proxies. All the robots are connected to the same central controller and offers a TCP socket interface between them. Referring to Figure 2.3, the device interfaces specify how to interact with a device and the type of messages the driver can receive, whereas the driver execute the command that is given by the client. Probably the most commonly used Player interface is the `position2d` interface that is used to control ground based robots. This interface specifies the format in which a velocity and/or position target should be sent. Multiple drivers can support the same interface and a single driver can support multiple interfaces (Collett *et al.*, 2005).

Stage is a simulator that simulates a population of robots with the choice of sensors and actuators on each robot. It is possible to write robot control algorithms as 'clients' to the Player 'server' which is equivalent to the real robot devices (Gerkey *et al.*, 2003*a*). The Player clients developed using Stage will work with little or no modification with the real robots (Playerstage, 2012).

2.5.2. Player Drivers

Player has two different types of drivers namely Static and Plugin. Static drivers are statically linked to the Player server and distributed with Player code and are used for standard sensors or actuators (Petersen and Fonseca, 2006). Plugin drivers are loaded into the Player server at runtime and are custom drivers which offer more flexibility compared to static drivers (Petersen and Fonseca, 2006).

The messages in Player is queue-based, and all the messages received by the driver is put in a queue, categorised and then processed (Holtzhausen, 2012). Message handling is part of the three successive processes that repeat in a loop, shown in Figure 2.4.

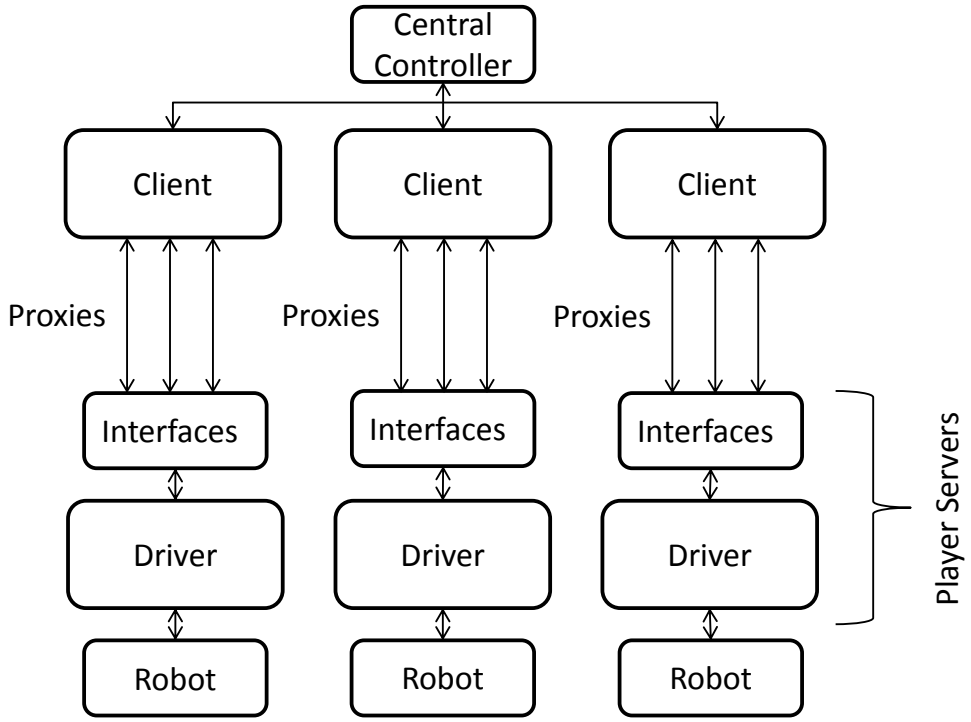


Figure 2.3: Player architecture configuration (Holtzhausen, 2012)

The first step in the loop is the *ProcessMessage* step where all incoming messages are categorised according to their type. There are five message types in Player of which the most important types are Data, Command and Request Messages. Data Messages usually contain information about the state of a robot and are published asynchronously. Command Messages usually contain desired position or velocity information and are also sent asynchronously. Request Messages are published synchronously and for every request message sent a response is returned, either Acknowledged (ACK) or Negatively Acknowledged (NACK) (Holtzhausen, 2012). Every new message that is received is processed by the *ProcessMessage* process to check with which interface it relates. Next, it checks to see if it is a Request or Command message, and finally the process is executed. The second step in the loop is the *ReadSensors* step where the sensors on the robot is read and information is stored in the variables. This information can then be used in calculations or in the *PublishData* step. All the calculations and algorithms performed on the robot is done in the *ReadSensors* step (Holtzhausen, 2012).

The interfaces is a specification of data, command and configuration format to interact with a certain device (Gerkey *et al.*, 2003*b*). The interfaces define the set of messages that is supported. Player does not have all the interfaces required by the soccer robots, but support most of it. There is for example no interface for the kicker mechanism in the Player Codebase (Holtzhausen, 2012). Interfaces are however not limited to be used for their intended purpose

only and can be used for other purposes originally intended for. For instance the `position2d` interface support amongst others the function to send desired position or velocity commands to the robot. The velocity command in this interface sends one attribute namely player_pos2d which contains three values *px*, *py* and *pa* relating to the x and y velocity and the angular speed. The client can be programmed to send the intensity of a kick in variable *px* and the angle of the kick in variable *py* and nothing in *pa*. The driver is then programmed to interpret the message in this way and the command can be executed.

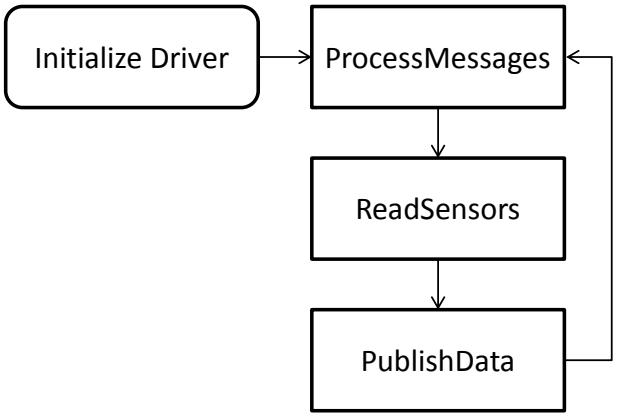


Figure 2.4: The main flow of the Player driver (Holtzhausen, 2012)

2.5.3. Player Client

In the latest rules of the SSL robot soccer each team consists of six robots. Each robot at Stellenbosch University will run the Player server which consists of a driver with multiple interfaces. A `PlayerClient` is an object that is used to connect to a Player server on a robot over a TCP socket (Holtzhausen, 2012). The client program contains one of these objects for every robot it wishes to connect to. When the client is connected to each robot, the Player proxies can be connected to the interfaces to control the robots (Holtzhausen, 2012). The messages are sent and received through these proxies. For instance the `position2d` proxy corresponds to the `position2d` interface.

2.6. Summary

This chapter took a look at the literature needed to achieve the objectives of this project. Holonomic movement was discussed and how it allows a robot to translate in any direction without changing its orientation. The calculations used in holonomic movement was also discussed.

It was shown how the holonomic characteristics can be used to combine translational and rotational movement and how it can be used in trajectory tracking and path following algorithms. Trajectory tracking and path following research was discussed and the state of the art algorithms briefly discussed.

Middelware is used to provide the OFC with access to the actuators and hardware on the robot over a wireless network. This is used to control the robot and perform the actions required in robot soccer. A discussion on middelware was given in this chapter. The Player Project was the best choice for middelware and was dicussed in depth.

3. Hardware Description

3.1. Introduction

This chapter describes the hardware on the robot used for this project. The first project on the soccer robots of Stellenbosch University was to develop an omnidirectional robot for RoboCup SSL (Smit, 2011). The second project was the redesign of the motor controllers with the use of encoder feedback from the motors (Jacobs, 2011). The next project after that was to develop a distributed control system for SSL robots (Holtzhausen, 2012). This project continues on these mentioned projects. Most of the hardware decisions and designs were done by Smit (2011) and Jacobs (2011) and will be discussed in this chapter. Hardware was added by Holtzhausen (2012) and will be discussed. The focus of this project was not the design of the hardware, but rather the implementation and testing of additional hardware that was not added by the mentioned authors. The focus of this project also included the implementation of sensors and software to control the hardware to meet the objectives of this project. As mentioned these include finding and dribbling a ball with a robot having full control and performing controlled kicks. This chapter is divided into the mechanical design and the electronic design and will be discussed separately.

3.2. Mechanical Design

The first iteration of the SSL robot was designed by Smit (2011) at Stellenbosch University. The design done by Smit (2011) was not within the required size limit and did not have a kicking or dribbling mechanisms (Holtzhausen, 2012). A second iteration of the robot was designed by Reynaldo Rodriguez, a technician at Stellenbosch University which was within the size specification. The technician also manufactured the kicker and dribbler mechanism. The kicker and dribbler mechanism was however never implemented or tested before and this was done for the first time in this project and can be seen in Figure 3.1.

It is now known that the SSL robots are omnidirectional, which means the robot can translate in any direction without having to turn first and the

robot is also able to make a full 360 degrees rotation on a single spot. The mechanical design of the robot has certain defects and irregularities which influence the robot while driving, therefore software is needed to compensate for these unpredictable behaviour (Holtzhausen, 2012). The motor unit of the robot is responsible for these irregularities. The main mechanical units of the robot is the omnidirectional wheels, motor unit, kicker unit, chipper unit and the dribbling unit. These will be discussed in the subsections below.

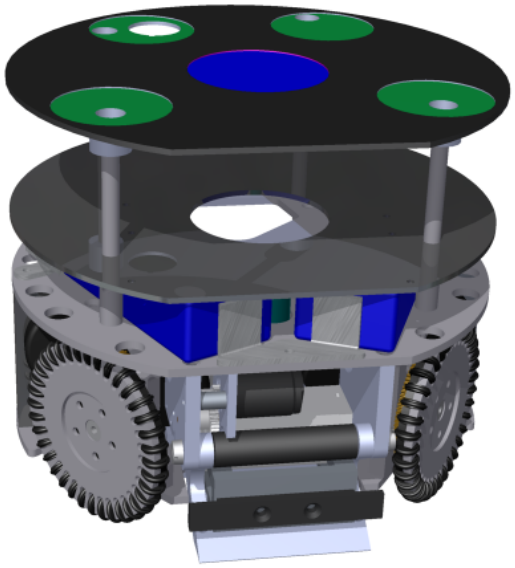


Figure 3.1: CAD design of second iteration soccer robot

3.2.1. Omnidirectional Wheels

Omnidirectional wheels are composed of a big wheel with grooves in order to make space for the 36 small wheels to rotate freely inside the grooves. Four of these omnidirectional wheels mounted on the robot give it three degrees of freedom, rotational, x- and y-direction. The omnidirectional wheels of the second iteration robot of Stellenbosch University is shown in Figure 3.2.

3.2.2. Motor Unit

The robot design consists of four motor units each driving one of the individual wheels of the robot. The motor unit consists out of the motor, gears and wheels. The motors that are used to drive the wheels are Faulhaber 2342 012 CR brushed DC motors. Most of the teams, including the current top three teams, Zickler *et al.* (2010), Watugala *et al.* (2005) and Chaiso and Sukvichai

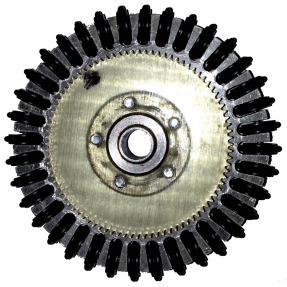


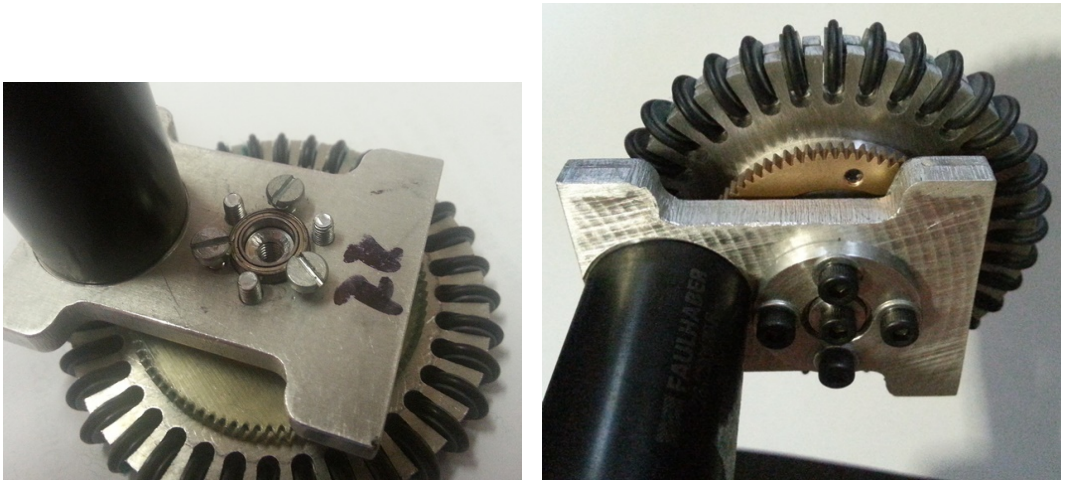
Figure 3.2: Omnidirectional wheel of second iteration soccer robot

(2011), use brushless DC motors which are more powerful than brushed motors. There are however teams that use brushed motors for instance Fukumoto *et al.* (2011) and Smit (2011). The brushed motors of this robot operate at a nominal speed of 12 V, a maximum output power of 17 W, a maximum speed of 7000 rpm and a maximum torque of up to 16 mNm (Faulhaber, 2012). The motors come with Faulhaber IE2-512 magnetic encoders, with 512 lines per revolution, which produce quadrature signal output (Holtzhausen, 2012).

The gear ratio of the second iteration robot was chosen by Jacobs (2011) as 9 to ensure that the wheels receive the adequate torque. The gear ratio of 9 was large. This required custom gears that was manufactured using a 3D printer. These gears were however not as strong as required when high torque transmission was needed. During the second year of this project a new motor unit design was researched and manufactured by Mouton (2013), a final year student from Stellenbosch University. This motor unit used the same motors but the gear ratio was changed by Mouton (2013) to 5. These gears were wire cut from phosphor bronze which is much more accurate and much stronger than the 3D printed gears (Mouton, 2013). The second iteration gears and the third iteration gears are shown in Figure 3.3. There was no significant change in weight since the old motor unit weighed 192 g and the new motor unit weighs 198 g. These new gears was tested and compared to the previous gears to determine if there is an improvement to the robot. These tests were performed and are discussed in section 6.

3.2.3. Kicker Unit

The kicker mechanism is used to shoot the match ball. According to the rules of RoboCup, the robot is not allowed to have (in its construction) anything that damages the field or the ball (RoboCup, 2012*a*). The different possible mechanisms that could be used is a spring, pneumatics or a solenoid. The kicker mechanism that was chosen for this project was designed and manufactured by the technician at Stellenbosch University and it was decided to use a solenoid, as it had the best performance in this application. It was only implemented on the robot during the course of this project. The kicker mechanism



(a) Motor unit of second iteration gearbox (b) Motor unit of third iteration gearbox

Figure 3.3: Motor unit showing the difference between the old and new gears

consists of a solenoid made from copper windings with an iron core inside that is propelled when current goes through the solenoid coil. The resistance of the solenoid wires is 1.6 Ω. The iron core hits against the kicking mechanism which then in turn hits the ball. The mechanical assembly of the kicker unit with the solenoid is shown in Figure 3.4. Very high current is needed to achieve the desired force from the iron core inside the solenoid and the electronic part of the kicker unit is responsible for this (discussed in section 3.3.4).

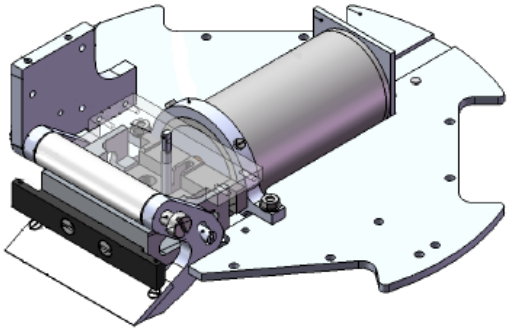
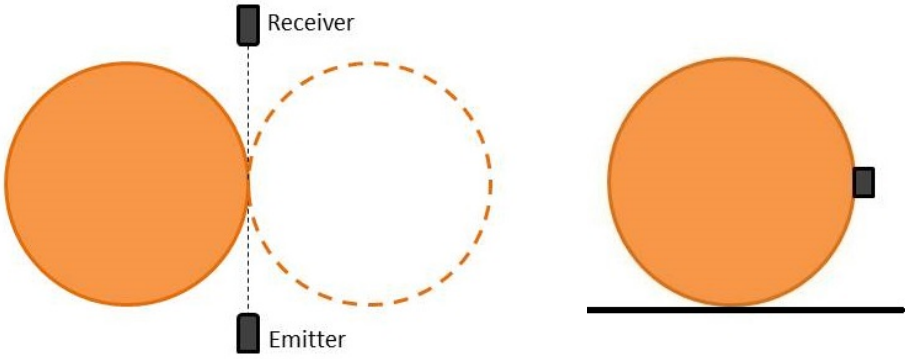


Figure 3.4: CAD assembly showing kicker unit

The kicker system was implemented on the robot to test the capabilities of the kicker. According to the rules of RoboCup, the maximum allowable speed that a robot is permitted to kick the ball is 10 m/s. The kicker was tested and the results are discussed in section 4.2.

3.2.4. Kicker Test Unit

A device was needed to measure the speed of the ball when kicked by the kicker plate. Different methods are available for example Akhter *et al.* (2013) who made use of acoustic measurements to determine the ball velocity at the different time durations of voltage flowing through the coil. Akhter *et al.* (2013) recorded two sound peaks, the first when the kicker touches the ball and the second when the ball hits a target at a known distance where the average velocity of the ball could then be calculated over that known distance. Pead (2013) made use of an LED and a photodiode to measure the speed of the ball. Research was done on ways to measure the ball speed and it was decided to use a light emitter and receiver pair. A structure was designed and manufactured to keep the emitter and receiver pair in place at the same height as the center of the ball. The emitter and receiver were carefully placed across from each other to create a beam at the same height as the center of the ball. The beam should be broken the instance the center of the ball passes through the beam and should be repaired the moment the center of the ball is pass the beam. An illustration of this design is shown in Figure 3.5. The electronic design is discussed in section 3.3.5.



(a) Top view showing the ball measuring device where the ball is breaking the beam and restoring the beam (dashed ball)

(b) Side view showing the height of the emitter receiver pair

Figure 3.5: Emitter receiver pair for ball speed measurement

The kicker mechanism discussed in section 3.2.3 was tested and characterised with the use of the ball speed measurement device shown in Figure 3.6. The mechanical design was done to ensure a dark environment for the light source and the phototransistor and to have them right across from each other and at the same height as the center of the ball.

The ball measurement unit was constructed using 1 mm mild steel that was laser cut and bended. The holes were also laser cut in the mild steel for

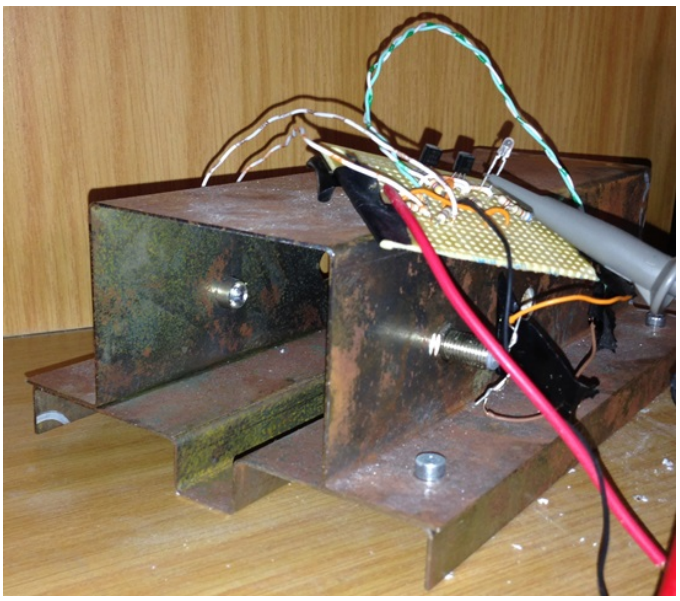


Figure 3.6: Ball speed measurement assembly with electronic circuit

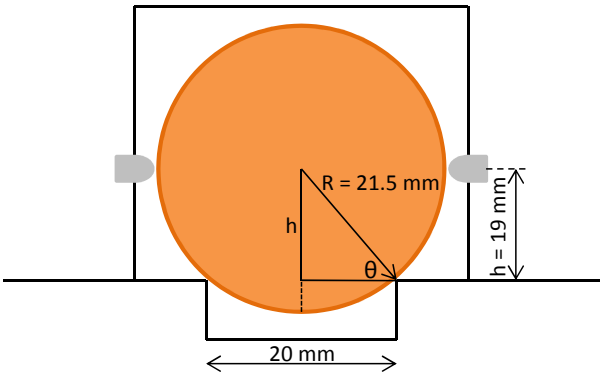


Figure 3.7: Ball speed measurement assembly showing sensor height

accuracy. The height was simply determined with the use of Pythagoras, as the ball diameter is known as 43 mm and the dimensions of the steel structure is known. Referring to Figure 3.7 the height is determined as in equation 3.1 and found as h = 0.019 m.

$$h = \sqrt{0.01^2 + R^2}. \tag{3.1}$$

3.2.5. Chipper Unit

The chipper unit is used in the game of robot soccer to chip the ball, lifting it from the field to avoid collision with obstacles, making it available for another team player in a better position, to continue game play. The chipper mecha-

nism was designed by the technician as Stellenbosch University, but was not used in this project as it was not included in the scope.

3.2.6. Dribbler Unit

The dribbler unit is located in the front of the robot as seen in Figure 3.1. The dribbler is essential for controlling the ball in a game match. The dribbler unit consists of a dribbler motor, gears and a dribbler bar. The dribbler motor is a Faulhaber 1724012SR brushed DC motor. The motor operate at a nominal voltage of 12 VDC with a maximum output of 2.12 W a maximum speed of 7900 rpm and a stall torque of 10.5 mNm (Faulhaber, 2012). The gears consist of two gears with a gear reduction ratio of 2. The dribbler bar has a length of 48 mm and is covered with silicon tube to give it a total diameter of 12 mm.

The dribbler bar is powered by the dribbler motor through the gear and when the bar makes contact with the ball it exerts backspin on the ball, keeping the ball in contact with the robot. According to RoboCup "Law 4 - The Robotic Equipment", the force exerted on the ball must be perpendicular to the plane of the field. Vertical or partially vertical dribbling bars, also known as side dribblers, are not permitted (RoboCup, 2012*a*). The dribbler bar is also not allowed to have a change in diameter. Before this rule was implemented, designs where the dribbler bar is tapered to the centre of the bar reducing the diameter was used to force the ball to the middle of the dribbler bar. The capabilities of the dribbling mechanism was tested and the results are discussed in section 4.3.

3.3. Electronic Design

The individual electronic subsystems that are used in the robot will be discussed in this section. These subsystems discussed are: the main controller, the motor controller, the translation and rotation sensors, the kicker/chipper board and the dribble control sensors.

3.3.1. Main Controller

The main controller on the robot is the central part of the robot that controls the rest of the electronic hardware on the robot. All the high level calculations are done on the main controller. Smit (2011) found that most of the SSL teams use Field Programmable Gate Array (FPGA) as main controllers. It was stated by Smit (2011) that the use of a Single Board Computer (SBC) is a good alternative. A SBC is a small computer that has the most of the functionality of a normal size computer, such as USB and ethernet interfaces all on one electronic board (Holtzhausen, 2012).

It was decided by Smit (2011) to use a SBC for the soccer robots of Stellenbosch University. The reason for this being that the SBC provides higher processing power, the use of standard networking communication and allows the ability to program using high-level programming languages such as Java, Python or C++ (Smit, 2011). Smit (2011) considered five different SBC's which are all Linux compatible and have microSD storage capability. The operating operating voltages of these SBC's are all within the desired range of 5 - 12 V. The chosen SBC will need to communicate with the other electronic subsystems through Serial Peripheral Interface Bus (SPI) and Inter-Integrated Circuit (I^2C) communication interfaces (Smit, 2011). Smit (2011) motivated why it was the best choice to use the Roboard RB-100 for the soccer robots of Stellenbosch University. The Roboard RB-100 is shown in Figure 3.8. The Roboard RB-100 is very competitive with a 1 GHz processor and 256 MB RAM and is specifically designed for robotic applications (Roboard, 2012). Roboard released the Roboard RB-110 after the choice for the RB-100 was made, but the only major difference is that the RB-110 has High-Speed serial communication of up to 12 Mbps and the Rb-100 has SPI. For the soccer robots of Stellenbosch University SPI is required and thus the RB-100 was still the preferred choice. The Roboard can be linked to the local network through either the Ethernet port or the Wi-Fi device. The Roboard can be accessed remotely from another computer that is also connected to the network, through remote desktop or secure shell (SSH) (Smit, 2011). The Roboard runs the operating system together with the control software from the 8 GB microSD card on the Roboard.

Figure 3.8: Roboard RB-100 used for the main controller (Roboard, 2012)

The Roboard uses IEEE 802.11 (Wi-Fi) for the communication between the central controller (server computer) and the main controller. The Roboard uses a Mini-PCI-WLAN card connected to the miniPCI slot. Communication between the central controller and the robot's main controller is a very important aspect of game play and this is another aspect where the soccer robots of Stellenbosch University differ from other SSL teams. Communication between the main controller and the motor controller used to control the translation

and rotation of the robot is done through SPI. The Roboard also has 24 Pulse-width Modulation (PWM) channels that can also be configured to be used as General Purpose Input/Output (GPIO) pins. Four of the PWM pins are configured as GPIO pins. One of them is used for the kicker board and used as an output pin. The other three are used as input pins to read the state of the IR sensors that are used for ball control. The board outline is shown in 3.9.

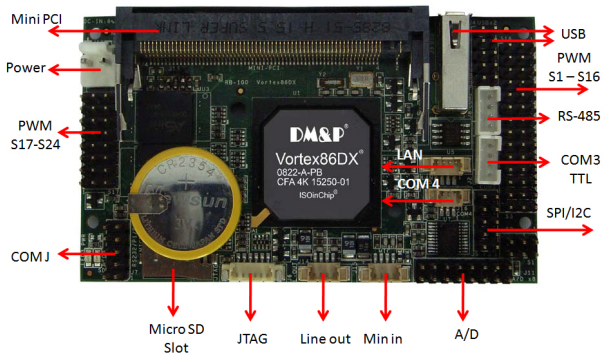


Figure 3.9: Roboard RB-100 board outline (Roboard, 2012)

3.3.2. Motor Controller

The motor controller is used to drive the individual motors at their desired speeds to perform the desired translation and rotation of the robot. The first iteration of the motor controller was done by Smit (2011) where each motor for driving the robot had its own controller. This means that four separate motor controllers were built. Each used closed-loop control to drive the motor at the desired speed (Smit, 2011). Feedback from the motors are needed for the closed-loop control and the encoders on the motors were used as feedback. The motor controller board of Smit (2011) used a PIC 18F2431 microcontroller and VNH2SP30 H-bridge. Smit (2011) motivates the choice for the microcontroller for its on-board quadrature encoder module, and high clock frequency and SPI interface that connect to the main controller through the SPI bus. The H-bridge is connected to the microcontroller to drive the motors directly from the battery. The magnetic encoders of the motors are connected to the microcontroller to provide feedback of the odometry readings of each motors for better controllability. The design of Smit (2011) experienced the problem that the motors caused the PIC on the motor controller to reset.

A new design for the motor controllers was made by Jacobs (2011). This new design used the same components as the design by Smit (2011). It contained the motor controller for each motor on a single board; furthermore it

had more connectors for more sensors. The I^2C connection of the Roboard are connected to the motor controller to be used by the sensors that require the I^2C interface. The new motor controller design by Jacobs (2011) can be seen in Figure 3.10.

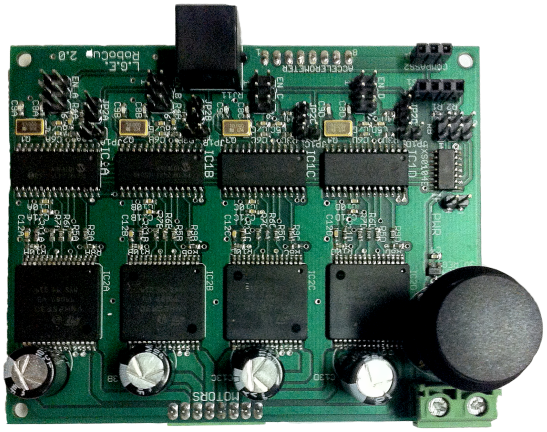


Figure 3.10: Motor controller circuit

3.3.3. Translation and Rotation Sensors

The robots receive position and orientation information from the OFC that receives the information from the SSL-Vision software. On-board sensors on the robots make them less dependent on camera data. Various sensors have been used in the RoboCup SSL of which rotary encoders that determine the rotational speed of the motors are the most common (Holtzhausen, 2012). Holtzhausen (2012) focussed on achieving accurate motion control of the soccer robots at Stellenbosch University. The translation and rotation of a device can be measured using a combination of accelerometers, gyroscopes, electronic compasses and mouse sensors (Holtzhausen, 2012). Holtzhausen (2012) motivates why a mouse sensor was not used for in-plane velocity of the robot. Holtzhausen (2012) also considered accelerometers for translational measurements of the robots. Accelerometers measure in-plane acceleration, which can be integrated over time to determine velocity and displacement of the robots. Holtzhausen (2012) decided to use accelerometers for the soccer robots of Stellenbosch University.

Smit (2011) suggested that a compass be used to determine the orientation of the robot, but Holtzhausen (2012) motivated that it could not be used since it is very sensitive to electromagnetic interference. Holtzhausen (2012) again motivated why the encoders on the motors cannot be used to determine the rotation of the robot due to wheel slippage. Holtzhausen (2012) motivated the choice of a gyroscope to determine the orientation of the robot.

Holtzhausen (2012) chose an Inertial Measurement Unit (IMU) which is a sensor that consists of accelerometers, gyroscopes and/or a compass. The IMU chosen by Holtzhausen (2012) have a three-axis accelerometer and a three-axis gyroscope. The IMU is installed in the centre of the robot with the axis of the IMU aligning with the axis of the robot. Two of the axis of the accelerometer were used to measure translational movement and the third axis was used to calibrate the IMU for slight misplacement. The IMU makes use of the I^2C interface of the Roboard. The IMU used by Holtzhausen (2012) is shown in Figure 3.11.

Figure 3.11: 6 degrees of freedom IMU used by Holtzhausen (2012)

3.3.4. Kicker Board

The hardware of the kicker mechanism is discussed in section 3.2.3. The solenoid needs a very high current in order to perform fast kicks of up to 10 m/s. This is done by charging four 250 V 1500 μF capacitors. These four capacitors are coupled in parallel to get a 250 V 6000 μF capacitor. The control of the voltage level to which the capacitors are charged and discharged through the coil of the solenoid is done with a kicker/chipper board that was designed by Pead (2013). This circuit board and the capacitors operate at very high voltages and are extremely dangerous. Full knowledge of how the circuit works, was required before it could be built and implemented on the Stellenbosch robots. Pead (2013) suggested that a discharge circuit be designed and implemented with the kicker/chipper board. This was done in this project before the kicker system was fully implemented. The kicker board design of Pead (2013) was built together with the discharge circuit before installation on the robot and is shown in Figure 3.12.

The circuit that was designed needs to fully discharge the capacitors in the case of an emergency or when the battery gets disconnected. This is done by dissipating the energy inside the capacitors through high power resistors. A relay and a combination of resistors and a LED is used between the kicker/chipper board and the charging capacitors and is shown in Figure 3.13. The relay works in a normally closed way in which the capacitors are connected to the

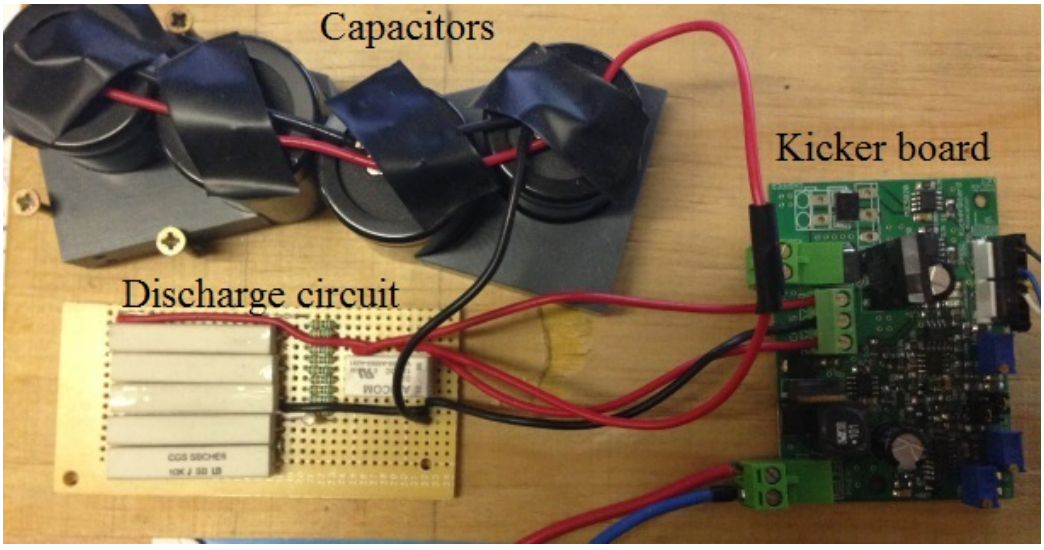


Figure 3.12: Kicker board with capacitors and discharge circuit

discharging resistors when no power is supplied to the kicker/chipper board. Only when voltage is supplied to the kicker/chipper board will the relay open to connect the charging capacitors to the kicker/chipper board. The LED is connected to the resisters to indicate when the energy inside the capacitors is fully dissipated. The circuit is designed to discharge the capacitors from 200 V. Resistor R1 is a combination of 9 0.6 W 110 KΩ resistors in parallel to give the correct voltage to the LED which operates at 2 V and resistor R2 is a combination of five 7 W 10 KΩ resistors in parallel used for most of the energy dissipation.

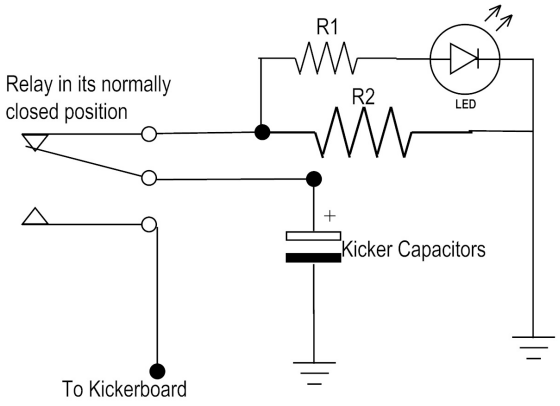


Figure 3.13: Kicker board with capacitors and discharge circuit

The total resistance seen by the 250 V 6000 μF capacitor is 2075 Ω and the total discharge time from 200 V to 50 V is approximately 17 seconds. This is considered enough time, since the capacitors will rarely get charged to that

high voltage. Also by referring to Figure 3.14 most of the energy is dissipated right at the very beginning and as time goes on, the change in voltage slows down. A faster discharge time would require much more discharge resistors. Due to space limitations it was not possible to fit more resistors onto the robot.

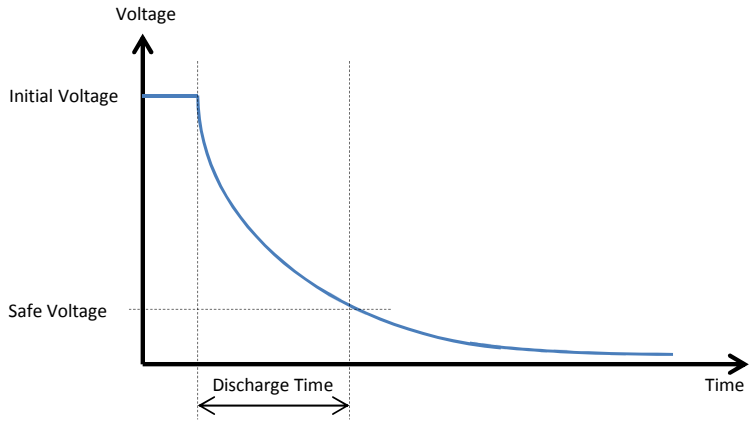


Figure 3.14: Typical capacitor discharge graph

The voltage to which the capacitors are charged is adjustable on the kicker/chipper circuit. The time that the current flows through the coil of the solenoid is adjustable by controlling the time that the kick pin is set logic high and this allows the IGBT to let the current flow from the capacitors to the solenoid coil. The voltage level and also the time that the kick pin is set high has an influence on the force of the kick. The GPIO pins of the Roboard are used to control the time that the kick pin is set high. The test results that were obtained from the kicker mechanism will be discussed in section 4.2.

3.3.5. Ball Speed Measurement Circuit

The kicker unit is used to propel the ball. The mechanical part of the kicker unit is discussed in section 3.2.3 and the electronic circuit of the kicker is discussed in section 3.3.4. In the game of soccer it is necessary to have control over the force a ball is kicked with. A design is needed to characterise the kicker unit in order to kick with different forces. The mechanical part kicks the ball and the electronic part controls the voltage passing through the solenoid and by varying the voltage, the force at which the ball is kicked, can be controlled.

A design was needed to determine the speed of the ball at the instance it leaves the kicker unit. The design choice was to use a light source with a phototransistor. Research was done on how to develop a simple, yet sufficient circuit to measure the speed of the ball. A circuit was used from Circuit (2012) that was already designed for a dark sensor using a phototransistor and is shown in Figure 3.15. The circuit uses two transistors, T1 and T2, a LED and

a phototransistor. The LED indicates when transistor T2 is switched on. The phototransistor used is a NPN phototransistor with a clear lens manufactured by SUNLED and is the light sensor of the circuit. The phototransistor uses a p-n junction in the reverse bias operation where the PN junction is the collector-base diode of a bipolar transistor and the light-induced current effectively replaces the *base* current (Coilgun, 2012). When light is concentrated on the phototransistor there is a change in base current high enough to increase the *collector* current. When current passes through the phototransistor, there is current flowing into the *base* of transistor T1 and this lets current flow through the *collector* to the *emitter* of T1. When current is flowing through resistor R1, there is a small amount of current flowing to the *base* of transistor T2 which lets current pass from the *collector* to the *emitter* of T2 and this switches the indicating LED on.

The resistor R5 was added to the existing design to provide the light source with the correct current. The light source used is a super white LED with a directivity angle of 15°, which is very desirable for this application. The LED is 5 mm in diameter, the same as the phototransistor and operates at 5 VDC.

An oscilloscope was used together with this circuit to determine the time it took the ball to pass the sensor. The oscilloscope was connected between R2 and the indicating LED and ground. The tests and results are discussed in section 4.2.

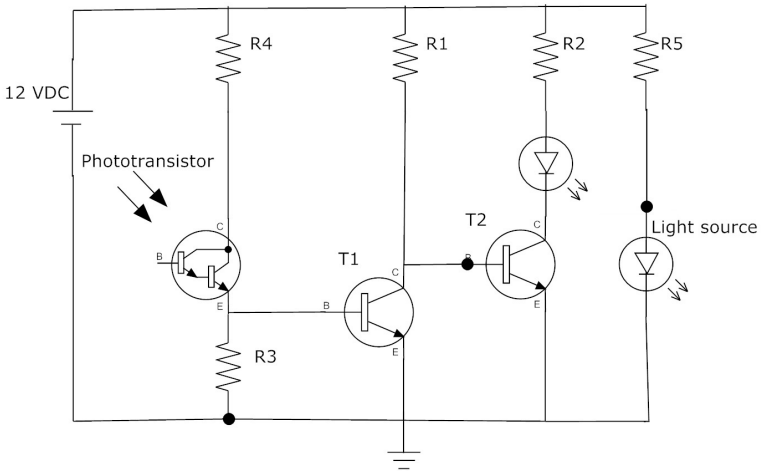


Figure 3.15: Light sensor used to measure the speed of the ball

3.3.6. Dribbler Speed Control Circuit

The speed of the dribbler bar was controlled with a simple yet sufficient circuit that could control the voltage seen by the dribbler motor. This was necessary to perform the test to characterise the dribbler that is discussed in section 4.3.

A LM317 adjustable voltage regulator was used to adjust the voltage seen by the dribbler motor. This adjustable voltage regulator is capable of supplying voltages between 1.2 V and 37 V and up to 1.5 A current. The standard application circuit shown in Figure 3.16 was found on the data sheet of the LM317 voltage regulator (Onsemi, 2013).

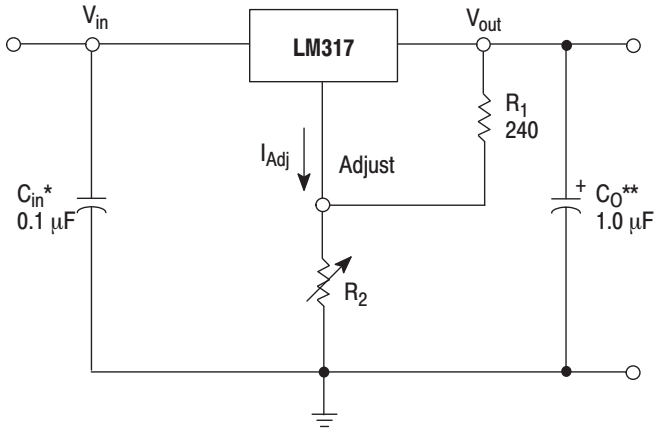


Figure 3.16: Standard application circuit for LM317 (Onsemi, 2013)

3.3.7. Dribble Control Sensors

The dribbler is used to keep control of the ball, but this can only be achieved if the ball stays on the dribbler bar. According to the rules of RoboCup the dribbler bar may not have any change in diameter to force the ball to the center of the bar. A design of some sort is needed to know when the ball is on the dribbler bar and if the position of the ball on the dribbler bar is known, this could help with ball control.

Watugala *et al.* (2005) used a small camera in conjunction with their FPGA to detect the position of the ball on their dribbler bar. Ishikawa *et al.* (2011) uses three IR LEDs to detect the ball. Goto *et al.* (2013) detect whether the ball is in front of the robot by using IR LED. Palmera *et al.* (2012) uses a set of four horizontally mounted lasers below the dribbler bar which detect the position of the ball on the dribbler bar.

The robots of Stellenbosch University have very limited space in the area where the mechanical part of dribbler is fitted. The mechanical design did not take into account that some sort of sensor(s) would have to be implemented onto the robot. The design criteria was that it should be small and fit onto the robot, indicate if the ball is on the dribbler and give the position of the ball on the dribbler bar. Taking a look at Figure 3.1 it would not be possible to implement a vision system to detect the ball and this would also require image processing that could take time to give accurate position information.

The other option of IR sensors was considered next and also to find what type of IR sensors are used by the teams that do use them. No information on the specific type of IR sensors used by other teams were found, but an off the shelf distance sensor manufactured by Sharp was found and is shown in Figure 3.17. This sensor has a range of 5 cm with a digital output. When the sensor detects an object within 5 cm, the output of the sensor is logic low and when no object is detected within 5 cm, the output of the sensor is logic high. A design choice was to use three of these distance sensors next to each other, above the dribbler bar, to provide the main controller with ball position information. Carefully choosing the placement of these three sensors could tell the main controller if the ball is on the dribbler and also estimate the position of the ball on the dribbler.

Figure 3.17: Sharp distance sensor GP2Y0D805Z0F (Sharp, 2013)

The total length of the dribbler bar is 48 mm and the diameter of the ball used is 43 mm. If the ball is on the dribbler bar it will be in front of at least one of the sensors. There exist six cases of ball positioning as seen in Figure 5.4. The first case is when the ball is completely in the middle of the dribbler bar (ideal case) and only the middle sensor's output is low. As soon as the ball drifts to the left, both the left and middle sensor's output will be low and similarly if the ball drifts to the right, the middle and right sensors' output will be low. When the ball drifts further to the left and almost off the bar, only the left sensor's output will be low and similarly if the ball drifts further to the right almost off the bar where only the right sensor's output will be low. The last case is when the ball drifts completely off the bar either to the left or right where none of the sensors' output will be low. The output of the sensor is 5 V for a logic high and 0 V for a logic low. This can be directly connected to the GPIO pins of the Roboard to be read in the software used for control of the robot. The circuit diagram on the data sheet is shown in Figure 3.18. The Roboard output pins are 12 V which is reduced to 5 V through a 5 V voltage regulator for the distance sensor. In the circuit V_{cc} gets 5 V from the voltage regulator and V_o is the output of the distance sensor which is connected to the input pin on the Roboard.

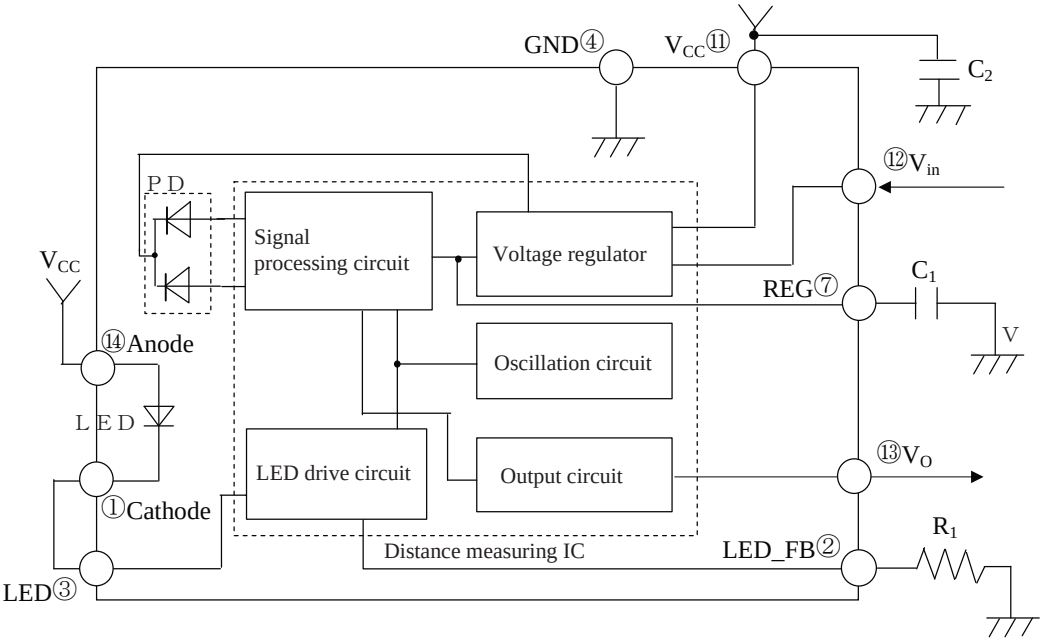


Figure 3.18: Sharp distance sensor circuit (Sharp, 2013)

3.4. Summary

This chapter mentioned the work done in previous projects on the soccer robots of Stellenbosch University. It also stated the hardware used in this project to meet the objectives of this project. This hardware was discussed in two main sections namely the mechanical part and the electronic part. The design choices made in previous projects as well as the design choices in this project was discussed. The main hardware addition in this project on the robot was the dribbler and kicker mechanism as well as the IR sensors used for dribble control.

4. Hardware Characterisation

4.1. Introduction

The kicker and the dribbler mechanism was implemented on the robots of Stellenbosch University during this project. The hardware was designed before this project, but never implemented or tested. Tests were performed to determine the characteristics and limitations of the kicker and the dribbler. On the kicker side it is needed to determine the forces of the solenoid to perform controlled kicks in a game. On the dribbler side it is desired to know the how well the dribbler works and under what conditions the dribble control is at its best. Data was obtained during the testing and will be discussed in this chapter.

4.2. Kicker Mechanism

The kicker mechanism is fitted onto the chassis of the robot and makes use of an iron core inside a solenoid to propel the ball in the soccer game. Ball control in the act of kicking is essential when passing the ball to another team member and also when kicking into the goal net. In the game of soccer the force at which the ball is kicked is critical and depends on the action that needs to be performed. When passing to a team member nearby would require a soft kick whereas a pass to the other side of the field may require a stronger kick. When aiming towards the goal it may be desirable to kick at a certain speed towards the goal. The mechanical design of the kicker unit is discussed in section 3.2.3 and the electronic design in section 3.3.4.

The factors that influence the force with which the ball is kicked include the voltage passing through the coil of the solenoid as well as the time duration of the voltage passing through the coil. Another possible factor could be the starting distance of the iron core with respect to the centre of the coil. The kicker board was used to control the voltage and the time duration of the voltage passing through the coil of the solenoid. The GPIO pins of the Roboard were used to accurately control the time duration of the voltage passing through the coil by giving the "Kick" pin a logic high for the desired time. The time the output pin was set high was verified with the oscilloscope.

The variable that needs to be controlled is the speed of the ball at the instance it is kicked and this is also the variable that is measured at the time instance when it leaves the kicker plate.

The output of the receiver was connected to an oscilloscope and a very small time/division to capture the time that the ball was breaking the beam as in Figure 3.5. The emitter/receiver pair is placed on the same height as the center of the ball so that the total distance that the ball travels when passing through the beam is equal to the diameter of the ball. If the time is captured and the diameter of the ball is known then the average speed of the ball over a very small distance can be calculated. This is the speed that was captured for comparison.

The first test was done by charging the capacitors to 100 V and using the Roboard to set the "Kick" pin high on the kicker board for a certain time period and it is also the period the IGBT lets current flow through the coils. The time period was varied from 0 to 11 ms with increments of 0.5 ms and the time the beam was broken was observed on the oscilloscope. This test was repeated in the same manner, except the capacitors had been charged to 200 V and the average of the results are shown in Figure 4.1. As the current flows through the coils, the iron core is pulled towards the center of the solenoid. From the figure it can be seen that when the on-time of the IGBT's is very low (below 1.5 ms for the 100 V test and below 0.5 ms for the 200 V test) the measured ball speed is zero. This happens because the force induced on the iron core is too small to even hit the kicker plate. As the on-time is increased, the ball speed increases fast and then reaches a plateau. After this the ball speed starts to decrease a little as the on-time is increased. As the on-time is increased from zero, the duration of the force on the iron core also increases. When the iron core is past the center of the solenoid, the force is still acting on the iron core and the force again pulls the iron core towards the center of the solenoid. This is what happens after 7 ms for the 100 V test and after 6 ms for the 200 V test.

It is also evident from Figure 4.1 that there is at some on-times almost a doubling of ball speed for the same on-time as for the 200 V test. The peak ball velocity is also achieved much faster for the 200 V test compared to the 100 V test. Although faster kicks can be achieved with the 200 V, it would be much easier to control ball speeds in the 0 to 6 m/s range when charging the capacitors to 100 V.

Another interesting test is to see the effect of the starting position of the iron core relative to the center of the solenoid. The three distances of the core with respect to the solenoid that were investigated is shown in Figure 4.2. The test was done 20 times at each distance and the ball speed was recorded. The capacitors were charged to 100 V and the on-time was fixed at 7 ms as this was the fastest ball speed referring to Figure 4.1. The result is shown in Figure 4.3. All the tests in Figure 4.1 were done with the core position at 36 cm.

The result was as expected, that is that the ball speed will be higher when

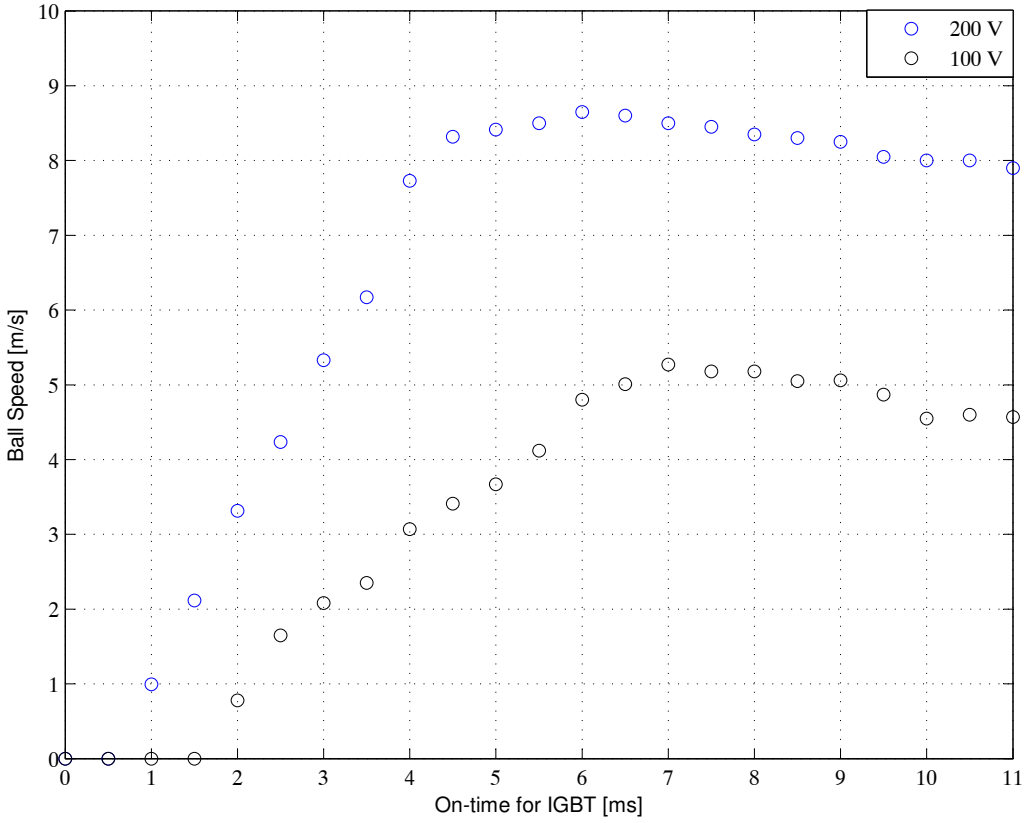


Figure 4.1: The ball speed recorded with the kicker test unit

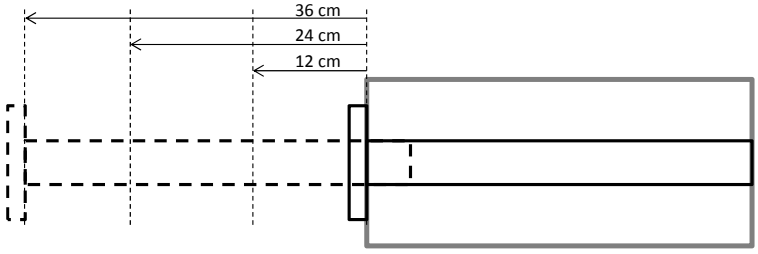


Figure 4.2: The position of the iron core with respect to the solenoid

the iron core is placed further away from the center of the solenoid. The ball speed varies almost linearly with the change in distance of the iron core relative to the solenoid. The furthest distance of 36 cm is the maximum distance that the iron core can be placed when the robot cover is over the robot.

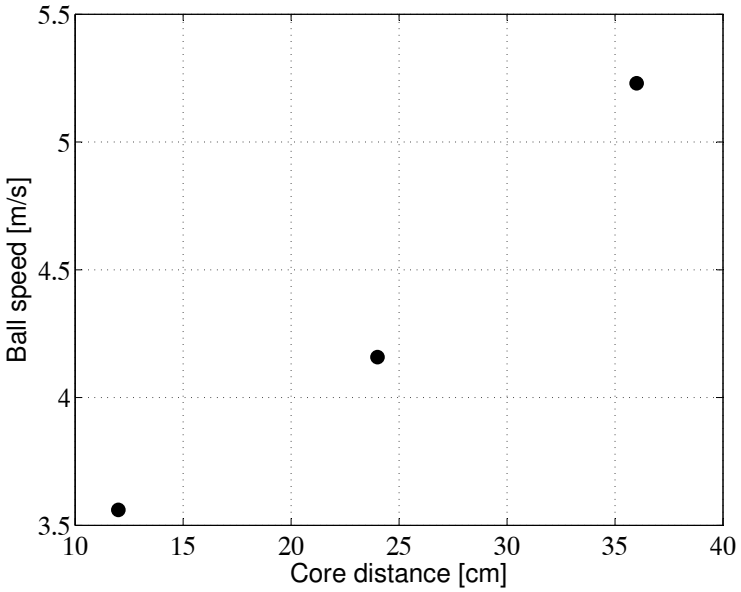


Figure 4.3: The ball speed with respect to the distance relative to the solenoid

4.3. Dribbler Mechanism

The dribbler mechanism is an essential part of the robot to be able to be competitive in robot soccer. The mechanical design is discussed in section 3.2.6 and the dribbler mechanism can be seen in Figure 3.1. The dribbler bar is rotated with a brushed DC motor by two gears and enough power needs to be transmitted to the dribbler bar to have sufficient control over the ball. The material properties of the dribbler bar must be appropriate to have high friction to transmit the backspin force to the ball. The chosen material is silicon tube that forms part of the dribbler bar and provide the sufficient friction needed. The performance of the dribbler was tested to determine the capabilities of the dribbler mechanism. The rest of the section will focus on the test setup and results and is similar to the work done by Ruiz and Weitzenfeld (2006).

Ruiz and Weitzenfeld (2006) defines dribbler control as "the ability of the dribbler to catch the ball, dribble it (hold it) and set it free when so desired". Ruiz and Weitzenfeld (2006) identified three types of responses: "catch with dribble", "catch with no dribble" and "no catch".

- The catch with dribble scenario is when the dribbler bar stops the ball without bouncing off and stays on the dribbler bar due to the dribbler rotation.

- The catch with no dribble scenario is when the ball hits against the dribbler bar without bouncing off, but the ball does not stay on the dribbler bar because the bar is not touching the ball. In other words the

robot would be able to gain full control over the ball by just moving a little forward in the current facing direction.

- The no catch scenario is when the ball hits against the dribbler bar and bounces off with the robot not having any control over the ball.

The objective in the test is similar to that of Ruiz and Weitzenfeld (2006) where it is desired to quantify the effect of the dribbler bar by varying the speed and approach angle of the ball impacting the dribbler bar. The experiment needs to be designed statistically in order to obtain unambiguous results (Hahn, 1977). The important part of planning an experiment is to identify the variables which can affect the performance and deciding what to do about them (Hahn, 1977). The dependent variables can be modified in order to see what the outcome is by changing each individual variable, one at a time.

The experimental setup consists of a launch pad directing the ball to the dribbler bar when manually dropped in a similar sense as Ruiz and Weitzenfeld (2006) did. Ball friction between the ball and the ramp is neglected, since it is kept constant throughout the experiment by using the same golf ball of mass 46 gr. The ball is simply dropped from a certain height to vary the speed. The experimental setup is shown in Figure 4.4.

The experimental setup with the dependent variables are shown in Figure 4.5. Three variables were identified, namely the ramp height, the approach angle of the ball to the dribbler bar and also the rotation speed of the dribbler bar. Ruiz and Weitzenfeld (2006) changed the ramp height, the approach angle of the ball to the dribbler bar and also the distance from the bottom of the ramp to the robot. The reason for choosing to change the rotation speed of the dribbler bar instead of the distance as done in Ruiz and Weitzenfeld (2006) is because changing the distance has a similar effect as changing the height h from which the ball is dropped. By changing the distance from the bottom of the ramp to the dribbler bar would only give the ball a longer distance to roll on the field. This would just decrease the speed of the ball due to the longer distance. By changing the voltage over the dribbler motor, the speed of the dribbler bar is also changed proportionally. Ruiz and Weitzenfeld (2006) kept the voltage constant at 7.4 V. In this experiment the distance from the ramp bottom to the dribbler was kept constant at 30 cm. The variables that are changed is shown in table 4.1. There are three levels for each variable: -1 for a low value, 0 for an medium value and +1 for a high value.

By changing the levels for each of the three variables shown in table 4.1 there exist 27 different combinations for the experiment. These different combinations, together with the results obtained from the experiment are shown in table 4.2. For every combination the ball was dropped from the corresponding height h, a total of 30 times.

It is evident from the experimental results shown in table 4.2 that combinations 1, 4, 7, 8, 10, 13 and 16 was very successful. From table 4.2 one can

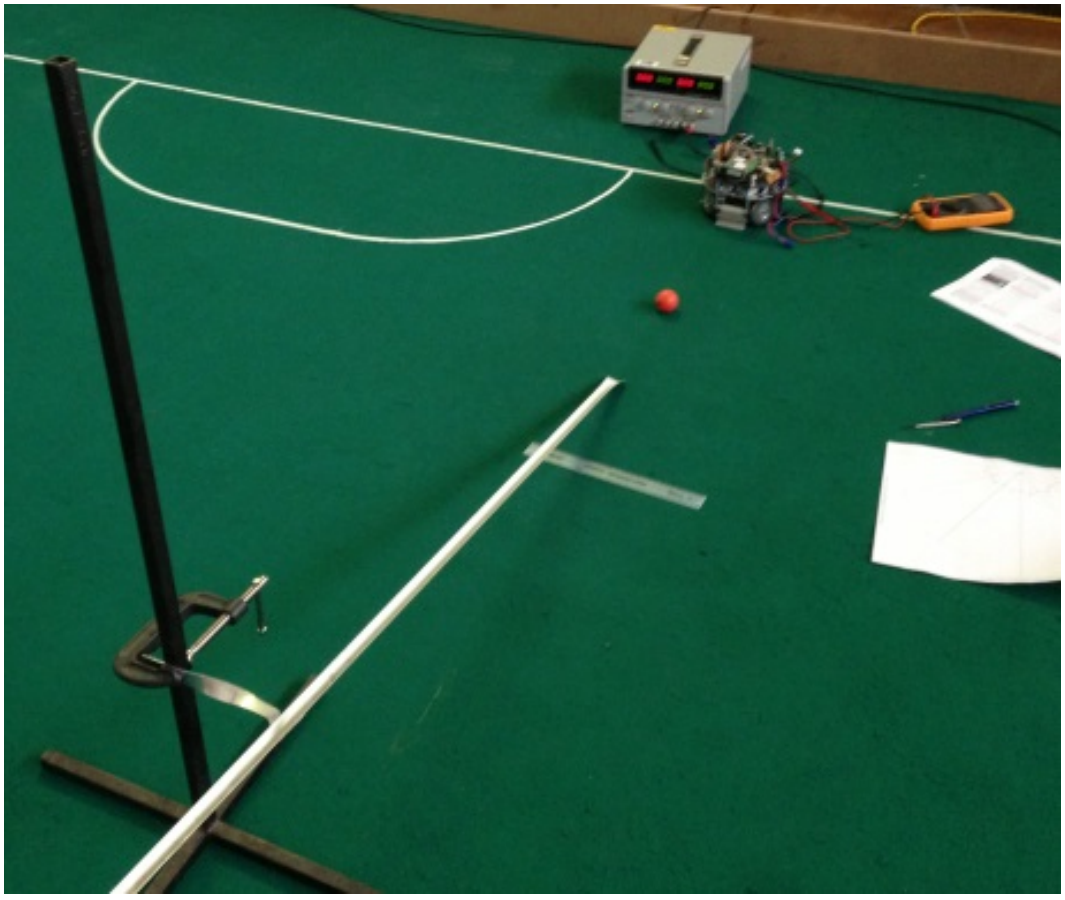


Figure 4.4: Experimental setup for dribbler testing consisting of the launch pad and the Stellenbosch soccer robot

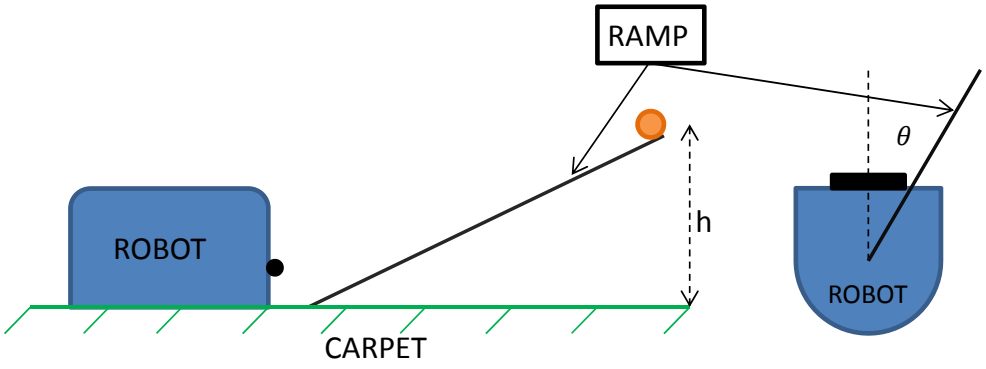
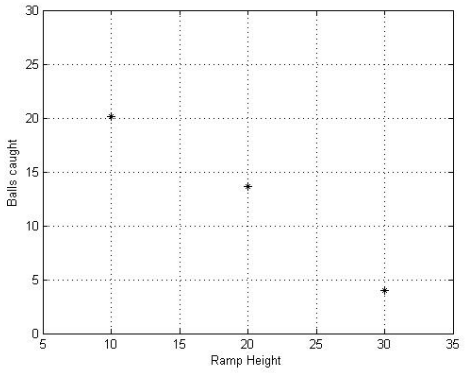


Figure 4.5: Side view and top view of the experimental setup showing the controlled ramp height h and controlled rotation angle θ.

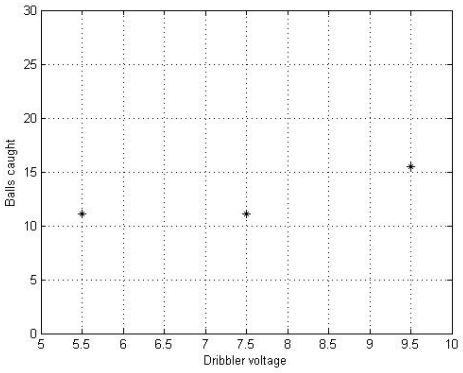
Table 4.1: Changing values in dribbler experiment

Factors	Notation	Levels		
		-1	0	1
Height	h (cm)	10	20	30
Dribbler Voltage	V (Volt)	5.5	7.5	9.5
Angle	θ (deg)	0°	10°	20°

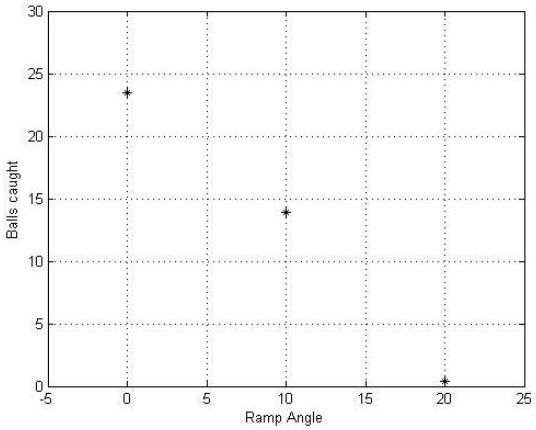
see that the angle θ has a significant effect on the performance of the dribbler. When θ is 0° all the tests were good except when the height is high and the dribbler speed is low. The dribbler performed at its best when the height h was kept low, the dribbler speed height and the angle θ kept low. The balls caught (with dribble) are shown for each of the different dependent values to give an indication of the effect of change for each variable and is shown in Figure 4.6.



(a) Ramp height



(b) Dribbler voltage



(c) Ramp angle

Figure 4.6: The number of balls caught for the different controlled values

Table 4.2: Dribbler experiment formulation and results

Formulation	Factor			Catch and dribble	Catch no dribble	No catch
	h	V	θ			
1	-1	-1	-1	30	0	0
2	0	-1	-1	26	4	0
3	+1	-1	-1	6	7	17
4	-1	0	-1	30	0	0
5	0	0	-1	29	1	0
6	+1	0	-1	5	25	0
7	-1	+1	-1	30	0	0
8	0	+1	-1	30	0	0
9	+1	+1	-1	25	5	0
10	-1	-1	0	29	1	0
11	0	-1	0	9	16	5
12	+1	-1	0	0	0	30
13	-1	0	0	30	0	0
14	0	0	0	3	16	11
15	+1	0	0	0	0	30
16	-1	+1	0	29	0	1
17	0	+1	0	25	2	3
18	+1	+1	0	0	3	27
19	-1	-1	+1	0	0	30
20	0	-1	+1	0	0	30
21	+1	-1	+1	0	0	30
22	-1	0	+1	2	0	28
23	0	0	+1	1	2	27
24	+1	0	+1	0	0	30
25	-1	+1	+1	1	0	29
26	0	+1	+1	0	0	30
27	+1	+1	+1	0	0	30

It is clear from Figure 4.6 that the dribbler is very sensitive to the angle of approach of the ball to the dribbler. The dribbler voltage, and directly related the dribbler speed also has a significant effect on ball control.

4.4. Summary

This chapter discussed two tests that was done to characterise the kicker mechanism and the dribbler mechanism. It is critical that the kicker should be able to propel the ball at different speeds. The voltage through the coil of the solenoid and the time duration of the current through the coil was varied to control the speed the ball is kicked at. Test data was obtained and found that it is possible to control the speed a ball is kicked at by changing the charging voltage of the capacitors and the time duration current flows through the coil of the solenoid.

The second test was to characterise the dribbler mechanism. The dribbler exerts backspin on the ball to keep the ball in control. The speed of the dribbler bar was varied and also the speed and angle with which the stationary robot should catch the ball. Test data was obtained for this test to show the capabilities of the dribbling mechanism. It was clear from the tests that best performance is achieved when the speed of the dribbler is high, the approaching speed of the ball is low and also the approach angle of the ball kept small. These results can be used in the AI system to calculate with what confidence level a robot can catch a moving ball.

5. Software Design

5.1. Introduction

This chapter is used to discuss the software used in this project to control the robot in meeting the objectives stated. The Player Project is used as middelware in this project to provide the OFC with access to the actuators and hardware on the robot. Player supports C++ and the software was the choice of programming language. The software builds on the work Holtzhausen (2012) did to control the robot in meeting the objectives of that project. Programming for the driver and client side is needed and will be discussed in this chapter. This chapter will also discuss the software used to perform the individual tests performed in this project. The tests and the data obtained from these tests will be discussed in chapter 6.

5.2. Motor Control

The robot used in this project has four identical brushed motors, all driven by the same motor controller circuit, that are used to manoeuvre the robot. The motor controller circuit consists of four individual motor controllers and a main controller which receives motor speed commands for each individual motor from the Roboard. Jacobs (2011) designed the current motor controller circuit to operate in a daisy chain where all the motor controllers are connected in series. When the command is received by the main controller, it is sent to the first motor controller and then gets sent to the next one and so forth until it reaches the fourth motor controller and then back to the main controller. The reason why this was the design choice is because of the reduction of connections between the motor controllers, but it has the disadvantage that there is a time difference at which each motor controller receives its commands due to the daisy chain action. Each motor controller has its own PID controller to ensure that each wheel turns at the desired rotational speed. The feedback to the PID of each motor controller is the shaft encoders on each motor.

5.3. Motion Control

This section discusses the control of the omni-directional robots used in the RoboCup SSL. The robots are required to move around in a dynamic environment without colliding with other moving objects or obstacles. This project only focusses on the manoeuvrability of one robot and obstacle avoidance was not considered. Holtzhausen (2012) developed a velocity profile based on the work of Purwin and D'Andrea (2006).

5.3.1. Velocity Profile

The robots need to manoeuvre on the field without the wheel slippage on the field. This can be achieved by limiting the maximum acceleration of the robots. Velocity profiles are generated before movement to calculate the velocities of each wheel without exceeding the maximum accelerations given. An algorithm developed by Purwin and D'Andrea (2006) was used by Holtzhausen (2012) to generate the velocity profiles for the robots of Stellenbosch University. This algorithm only computes the translational velocities and not the rotational velocity speeds of the robot. The rotational movement of the robot was not in the scope of the project of Holtzhausen (2012), but in the scope of this project.

The goal of the algorithm of Purwin and D'Andrea (2006) is to compute the velocity of the robot in the x- and y-direction by knowing the desired distance to be travelled in the x- and y-direction. The algorithm takes in the initial velocity and maximum allowed acceleration, deceleration and velocity to compute the velocity profile in the x- and y-direction. The algorithm takes into account that the distance in the x- and y-direction may be different so it computes the velocity profiles that both moving directions with take the same time to execute. The typical velocity profile generated for either the x- or y-direction with an initial velocity of $0\,\mathrm{m/s}$ is shown in Figure 5.1. The first slope is the acceleration, between t_1 and t_2 is the maximum velocity and after t_2 is the deceleration of the robot. For each velocity profile there are six variables calculated, that is three for the slopes (m_1, m_2 and m_3) and three for the intercepts (c_1, c_2 and c_3). The desired velocity of the robot at a current time t is then calculated by looking where on the velocity profile the current time is.

5.4. Driver Program

The driver program is used to provide the central controller with access to control the robot. The Player program and the drivers available are discussed in section 2.5. The messages that are sent between the client and the driver provide the central controller with access to the robot.

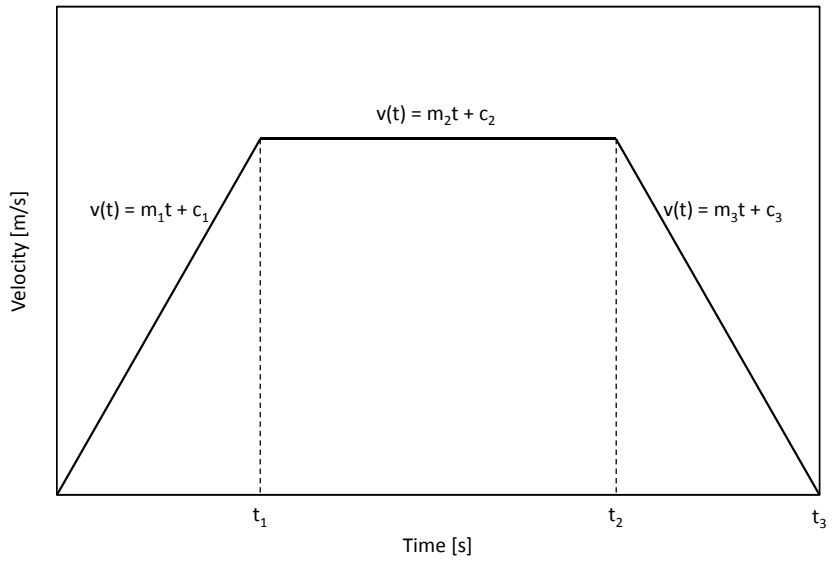


Figure 5.1: Velocity profile used for translational movement

5.4.1. Interfaces Used in Driver Program

Interfaces were discussed in section 2.5 and this section explains the interfaces that were used to control the robots. Only one of the request messages of the `position2d` interface is used and that is the odometry request that enables new clients connecting to the robot to establish the position and orientation of the robot. The command messages of the `position2d` interface that is mostly used is the position and velocity commands. It depends on how the message content is used on the driver side, but in most cases the position command can be used to send a desired end position and orientation of the robot and the velocity command can be used to send the desired speed at which the robot should move. Every interface has only a limited amount of messages it can send, but certain types of messages of other interfaces may be used to overcome this limitation.

The $\texttt{position2d}_\texttt{m}$ interface is used to control the robot. The $\texttt{position2d}_\texttt{k}$ will be used mainly for the kicker actions and some for robot. The $\texttt{position2d}_\texttt{b}$ will be used for sending information about the ball. The GoTo message type of $\texttt{position2d}_\texttt{m}$ contains a velocity and position message that is used to send the desired end position of the robot and the current velocity of the robot obtained from the vision system. The position data of the robot obtained from the vision system should also be sent to the robot, but since the position and velocity message types of the $\texttt{position2d}_\texttt{m}$ interface is already used another message is needed to send this information. Of the $\texttt{position2d}_\texttt{k}$ interface only the position message type is used for commands regarding the kicker and the velocity message type is still available. The velocity message type of the $\texttt{position2d}_\texttt{k}$ interface is then used to send the most recent position informa-

tion obtained from the vision system to the robot. It could have been possible to make use of the data message in `position2d_m`, but data messages require an ACK message to be sent back and slows down the process. Command messages are faster and some loss in transmission is acceptable for the position information of the robot.

Table 5.1: Interface command messages

Interface	Command	Variables	Purpose
`position2d_m`	velocity	$\dot{x}\ \dot{y}\ \dot{\theta}$	Send a translational and rotational velocity command to the robot
`position2d_m`	position	$x\ y\ \theta$	Send a desired destination command to the robot
`position2d_k`	velocity	$x\ y\ \theta$	Send the camera data to the robot
`position2d_k`	position	$x\ y\ \theta$	Send the kicker commands to the robot
`position2d_b`	velocity	$x\ y\ \theta$	Send the camera data of ball position to the robot

A setspeed velocity message is used from the `position2d_b` interface to send the position data of the ball obtained from the vision system. It does not mean that if a velocity message is sent that it should be interpreted that way on the driver side. The driver is written as such that it will use the values of vx and vy as the x and y position of the ball and the third element in the array, va is ignored. The `setposition` message contains two 3x1 arrays, one for position and one for velocity whereas the `setspeed` only contains one 3x1 array of which only two elements is needed. This is just the way the Player message types are constructed and should be chosen to send only the amount of data needed.

5.4.2. Classes Used in Driver Program

Certain classes were written by Holtzhausen (2012) in C++ that were required to achieve the distributed control architecture. Most of these classes were used exactly like Holtzhausen (2012) wrote them with changes made inside these classes, where necessary.

The `RoboDriver` class communicates with the motor controller over the SPI of the Roboard to send velocity commands to each individual motor. The encoder data from each motor can also be accessed with the `RoboDriver` class. The `IMU` class is used to initialize the I^2C communication with the IMU sensor which provides acceleration and rotational data measured by the sensor (Holtzhausen, 2012). The `AngleKalman` and `Controller` classes use a Kalman

Filter (KF) algorithm implemented by Holtzhausen (2012) to estimate the position and velocity information at a faster frequency than the camera data information of the robot is available. The estimator is updated every time new IMU or camera data is made available to the robot. The `MotionPlan` class generates a velocity profile that Holtzhausen (2012) implemented from Purwin and D'Andrea (2006) and is discussed in section 5.3.1. The `PID` class implements a trajectory tracking control and was improved from the work of Holtzhausen (2012). The `DTime` class is used to keep track of time while the robot is executing a command. It is also used with sampling times for certain algorithms as well as with the kicker to adjust the time which current is flowing through the solenoid of the kicker.

Table 5.2: Custom classes used in the Player drivers (Holtzhausen, 2012)

Class	**Instances**	**Purpose**
`RoboDriver`	mtrCntrl	Communicates with the motor controller circuit. Sends velocity commands to the wheels
`IMU`	imu	Initializes the IMU sensor and reads all the IMU data
`Controller`	xcntrl	Implements the state estimator in the x direction
	ycntrl	Implements the state estimator in the y direction
`AngleKalman`	acntrl	Implements the angular state estimator
`PID`	xPID	Implements the trajectory tracking control in the x direction
	yPID	Implements the trajectory tracking control in the y direction
	anglePID	Implements angular trajectory tracking control
`DTime`	driveTime	Determines the sampling time for the estimator algorithms
`MotionPlan`	mp	Generates the minimum time velocity profile

5.4.3. Main Function in Driver Program

The `ProcessPos2dPosCmd()` is the main function of the driver used for the distributed control that was further developed from Holtzhausen (2012). This function receives a desired end position relative to the current position of the robot from the central controller. A basic flow chart of this function is shown in Figure 5.2. In the function there is a *while* loop with condition *while* (move == 1) and inside that *while* loop it sets move equal to 0 when the desired

manoeuvre is completed. This loop time is approximately 20 ms, but may wary with a few milliseconds depending on all the calculations made during this loop.

This function starts off with initializing the PID class with the PID constants. The calculation of the derivative gain was only implemented during this project due to time limitations during the project of Holtzhausen (2012). The PID controller was optimized and the results are discussed in section 6.2.

The next step in the function is where the KFs are initialized for the translational and rotational predictions. There were no improvements made to these parameters and they were used from Holtzhausen (2012). The next step is to initialise the GPIO pins of the Roboard that will be used to communicate with the IR sensors for ball control during dribbling. The velocity profiles are next generated with the `MotionPlan` class to determine the desired global velocities of the robot during the execution of the translational manoeuvre. The *move* integer is set to 1 and the function enters the *while* loop. The first step in the *while* loop is to determine the desired robot velocities in the global frame based on the velocity profile generated. There is a check to see if the desired robot velocities of the robot is zero, which means the robot should stop, otherwise the command is ignored. The acceleration data is received from the IMU sensors on the robot and this data is in the robot axis frame. This data is transformed to the global axis frame based on the current orientation of the robot relative the global axis frame. Camera data of the position, velocity and orientation of the robot (in the global axis frame) together with the IMU data of the robot (now in the global axis frame) is used to update the estimates of the position, velocity, orientation and angular velocity of the robot (in the global axis frame). The frequency at which camera data is available is much slower than that of the IMU data, but the camera data is much more accurate than the IMU data and that is why the KF was implemented by Holtzhausen (2012). The next step in Figure 5.2 is used in ball control while dribbling and will be discussed later in this section.

The next step is to calculate the errors in translational velocities and the rotational velocity based on the KF estimates and the desired velocities calculated from the velocity profile. These errors are sent to the `PID` class to calculate the output of the PID controller for each of the three errors. The output from the PID controller is the current desired velocities of the robot in the global axis frame. These velocities need to be transformed to the robot axis frame based on the current orientation of the robot relative to the global axis frame. This transformation was however never done by Holtzhausen (2012) and resulted in a system that was impossible to control in all circumstances. The transformation matrices are discussed in appendix B. Holtzhausen (2012) tuned the PID parameters to get the robot to follow a trajectory, but this was never as good as it should be and the system had random behaviour in some directions when a command was issued. When this transformation of the velocities back to the robot frame was done, the controllability was initially even

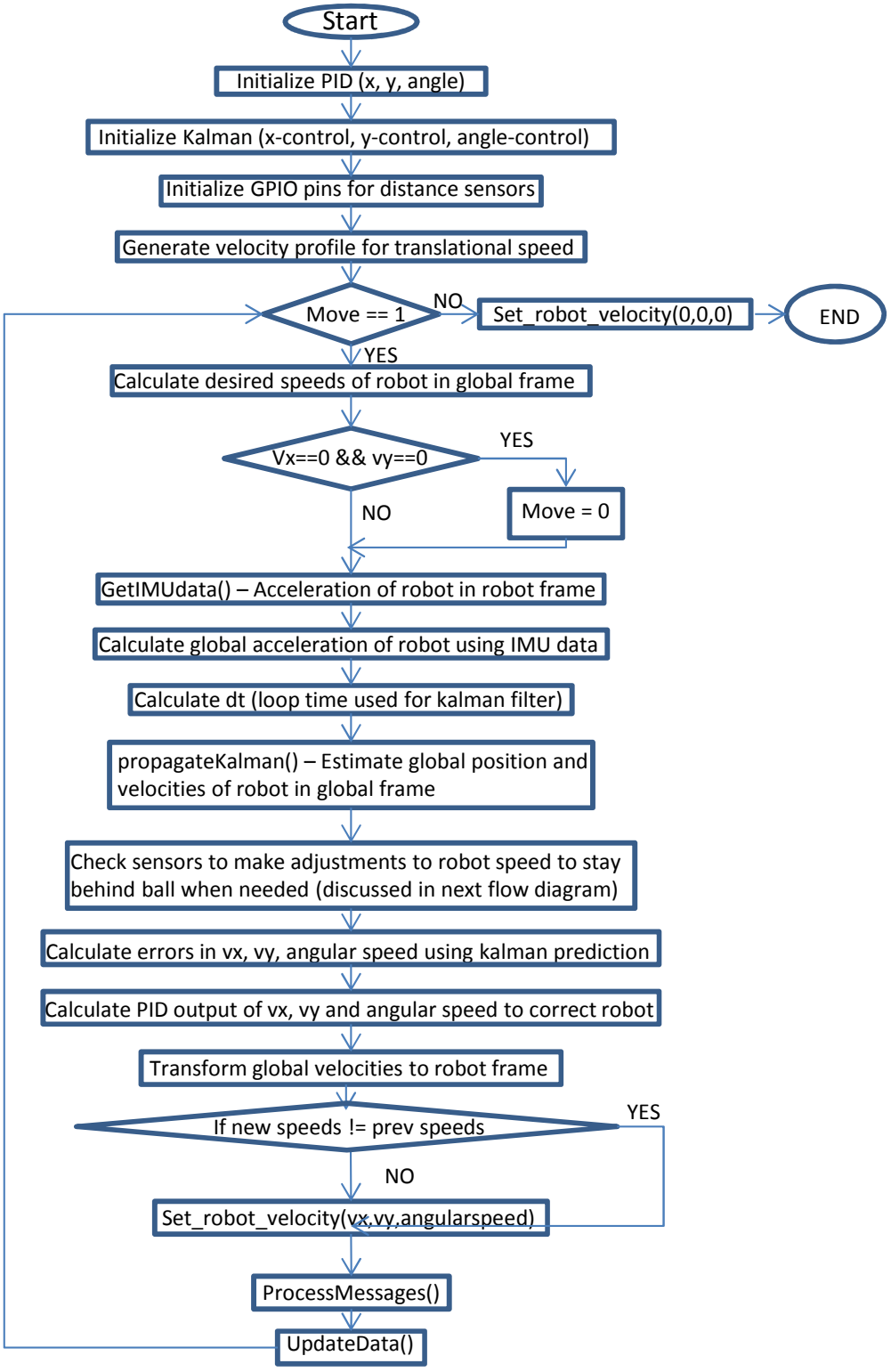


Figure 5.2: Flowchart of `ProcessPos2dPosCmd()` function

worse. This was due to control parameters being completely incorrect. The tuning of the control parameters was started from scratch and is discussed in section 6.2. The next step in the flow diagram is to determine if the current velocity of the robot (in the robot axis frame) is the same as the previous velocity. If this is the case then there is no need to send it again to the motor controller. If the new speed is not the same as the previous speed then the new speed is sent to the motor controller with the set_robot_velocity function. The last step in this flow chart is the `ProcessMessages()` that checks for new incoming messages that should be executed.

5.4.4. Dribble Control Algorithm

The main part of this project was to implement a control algorithm that would ensure the robot can dribble the ball without losing control of the ball. It is a very important aspect to have full control of the ball at all times when dribbling. A basic flowchart of this algorithm is shown in Figure 5.3 and this algorithm is called inside the *while* loop in the `ProcessPos2dPosCmd` function when needed. The part in the source code where this algorithm is used is explained in Appendix C.

There are three distance sensors like the one described in section 3.3.7 that detect the position of the ball on the dribbler. The ideal is that the ball stays in the center of the dribbler bar at all times during dribbling. It is however not the worst case if the ball moves slightly to the left or right of the center of the dribbler bar, as long as the ball stays on the dribbler bar while dribbling the ball. In Figure 5.4 the top view of the dribbler with the three IR dribble sensors (black) and the five possible positions of the ball on the dribbler (dotted orange circles) are shown. The dotted black lines show the line of sight of each dribbler sensor and whenever the ball is within the line of sight of a sensor it is read on the GPIO pins of the Roboard.

Three pins are initialised in the beginning of the `ProcessPos2dPosCmd()` function and read either logic 0 or logic 1. The IR sensors, in their line of sight, give a logic zero output when the ball is within 5 cm of the sensor and a logic 1 output when there is nothing or the object is further than 5 cm in the line of sight of the sensor. There are three integer values, *leftsensor*, *middlesensor* and *rightsensor* that store the output of each of the sensors right before the start of the algorithm shown in Figure 5.3. Each of these integer values are set either TRUE if the ball is in front of them or FALSE if no ball is detected in font of a sensor. These integers are used in the algorithm, shown in Figure 5.3.

The sensors are placed above the dribbler bar in such a way that the ball will always be detectable by one (in some cases by two) of the sensors when the ball is on the dribbler bar. The first condition in the algorithm establish if any of the three sensors detect a ball otherwise the algorithm cannot be continued before the ball is found. When it is known that the ball is indeed somewhere on the dribbler bar the next step is to establish the location of the ball on

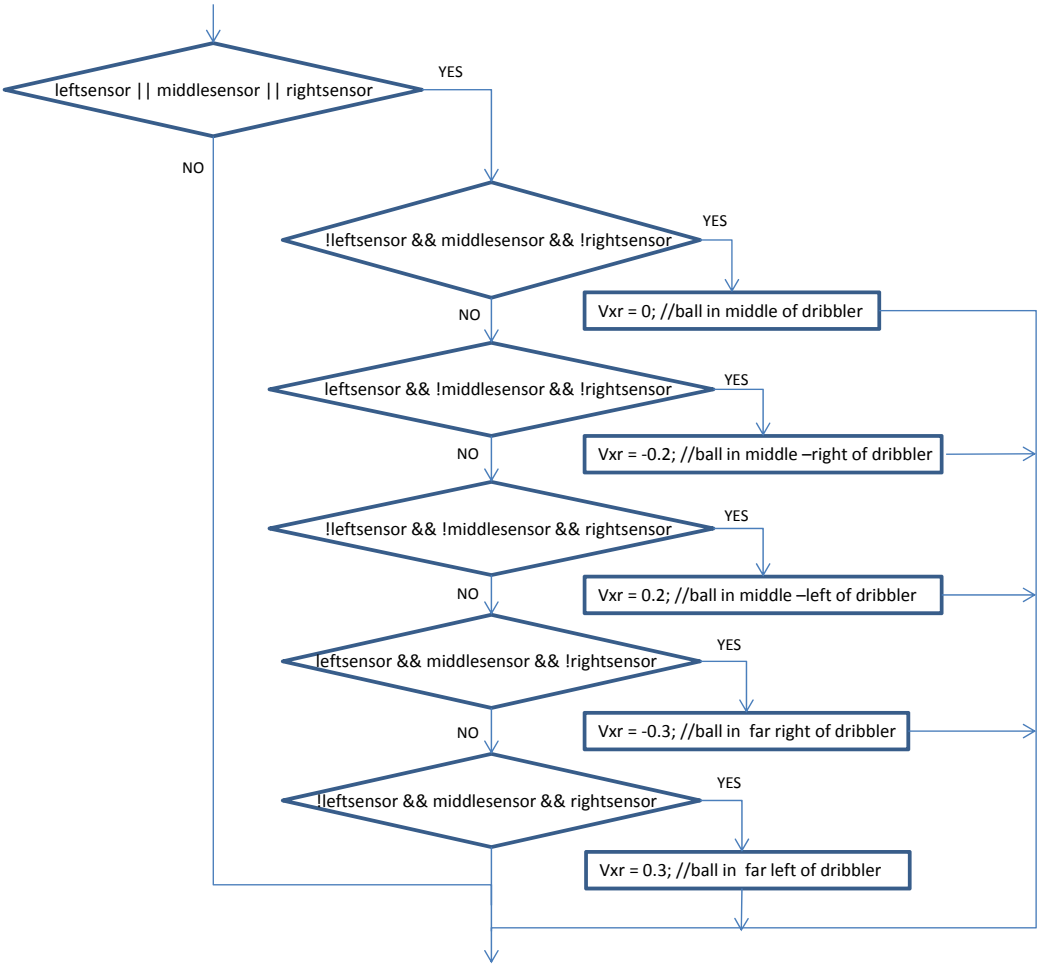


Figure 5.3: Ball control flow diagram

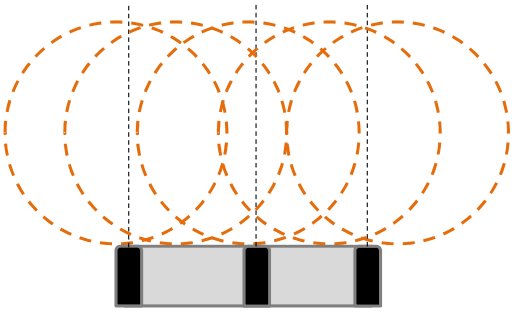


Figure 5.4: Top view of dribbler bar with IR sensors

the dribbler bar. When dribbling the ball, the ideal is that the ball should be in the middle of the dribbler bar and if not the robot should compensate accordingly to ensure the ball stays in the middle of the bar.

To find out where on the dribbler bar the ball is it is first established if *middlesensor* is TRUE and *leftsensor* and *rightsensor* is FALSE. This will be the case when the ball is right in the middle and the robot should continue the intended trajectory. If this was not the case it then determines if *middlesensor* and *rightsensor* are TRUE and leftsensor is FALSE. This will be the case when the ball is only slightly to the right of the center of the dribbler bar. Only little compensation is needed and the robot moves slightly to its right. This movement is perpendicular to the current facing orientation of the robot. The opposite is done in the next condition statement when the ball is only slightly to the left of the center of the dribbler. When the ball is to the far right of the dribbler bar, only *rightsensor* will be TRUE and the robot has to compensate more by moving more to the right and the opposite should happen when only *leftsensor* is TRUE. If none of the sensors picks up the ball, the ball is lost and the robot has to rely on the vision system to tell the robot the position of the ball.

To the authors knowledge there are no algorithms available on how dribble control is performed by other robots. It is only known that most of the teams use an IR sensor of a combination of them for dribble control. This algorithm was developed by playing around on the field and moving the robot by hand while the dribbler bar is switched on and the ball dribbled. The robot was moved forward by hand and compensated by hand when the ball was not in the center of the dribbler bar. Initially it was thought that an angle compensation by the robot according to the position of the ball would force the ball to the center of the dribbler bar. It was very difficult to keep the ball on the dribbler using this method and the robot would end up completely missing the end destination due to this angle compensation.

Another approach was then considered, again moving the robot by hand and trying to keep the ball as close to the center of the dribbler bar as possible. This was to move the robot in the sideways direction when the ball would move off center of the dribbler bar. This was a much easier method and always kept the robot orientation the same as the starting orientation. It was seen that this method would result in some error in the end position, but way better than the previous approach. This is the motivation for doing some research on trajectory tracking and path following and hoping to get the robot to dribble a ball and end at the desired destination with full ball control.

5.4.5. Ball Find Algorithm

When the robot is at a known location and the ball is at another known location, the robot could be commanded move to the ball. See the algorithm used to find the ball in Figure 5.5. This manoeuvre is done in two steps. Firstly,

the robot completes a rotational movement to the desired end orientation and then do the translational movement.

The global position and orientation of the robot and the position of the ball is obtained from the vision system. Secondly the x- and y-distance of the ball relative to the position of the robot in the global system is calculated and used to calculate the angle the ball makes with the robot and also the distance the robot should travel. The robot should turn to the angle where the dribbler bar is facing the ball. This angle is calculated with the *atan2(y,x)* function in C++ which returns the principle value of the arc tangent of y/x in radians.

When the robot is now "facing" the ball, the next command is for the robot to translate in a straight line towards the ball in the current orientation.

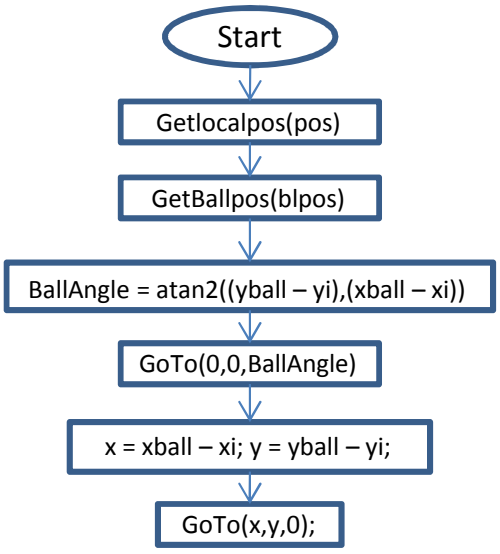


Figure 5.5: Flow diagram of algorithm used to find the ball

5.4.6. Trajectory Tracking on a Curve

In section 2.3 a brief discussion is given of trajectory tracking and path following algorithms used for mobile robots. A very basic trajectory tracking algorithm was proposed for the robots of Stellenbosch University where the "virtual robot" approach is not yet used. The proposed algorithm is a curvature steering method where the robot forward velocity is kept constant and the rotational velocity of the robot adapted stepwise to follow the desired trajectory.

This algorithm was proposed to test the capabilities of robot control in a combined translational and rotational movement. The PID controllers for translational velocity and rotational velocity are discussed in section 5.4.3.

This trajectory tracking algorithm can then be further developed to the state of the art algorithms ("virtual robot" approach) used by the authors discussed in section 2.3.

For this algorithm, the path length for every step needs to be determined to calculate the orientation adjustment to stay on the path. The duration of the *while* loop in the main function in Figure 5.2 is 20 ms. When the robot is moving at a constant velocity, the orientation adjustment ($\Delta\theta$) for every step should be calculated such that the robot arrives at the next point on the trajectory after 20 ms. Refer to Figure 5.6 of an example of a robot following a trajectory by adjusting the orientation with ($\Delta\theta$) between the current point and the next point. With the time, $\Delta t = 20$ ms and the constant velocity v chosen, the distance $\Delta s = v\Delta t$ travelled in one step can be calculated. This distance will be fixed for every step on the trajectory for a constant velocity chosen. To calculate the orientation adjustment, the next point on the trajectory where the distance along the path is equal to Δs is to be determined.

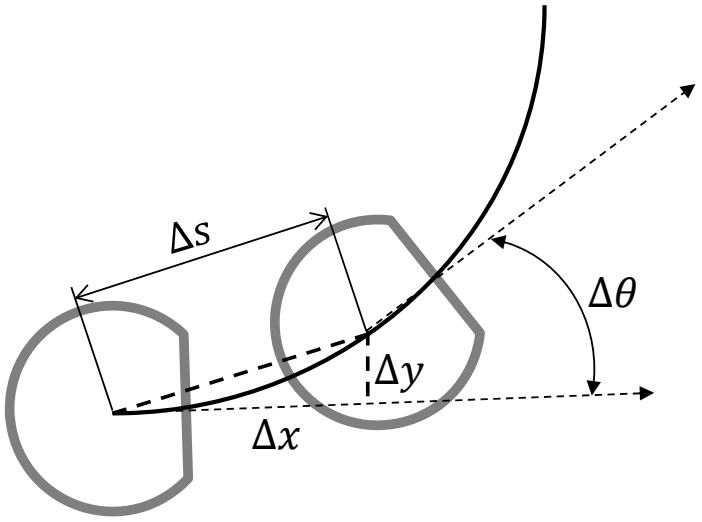


Figure 5.6: Trajectory tracking along a curved path

Consider a real function *f(x)* with derivative *f'(x)* = dy/dx describing a continuous trajectory. The distance Δs between two points, a and b on the trajectory, can be calculated with the use of Pythagoras (see Figure 5.6). The derivation of the path length between points a and b is shown in equation 5.1.

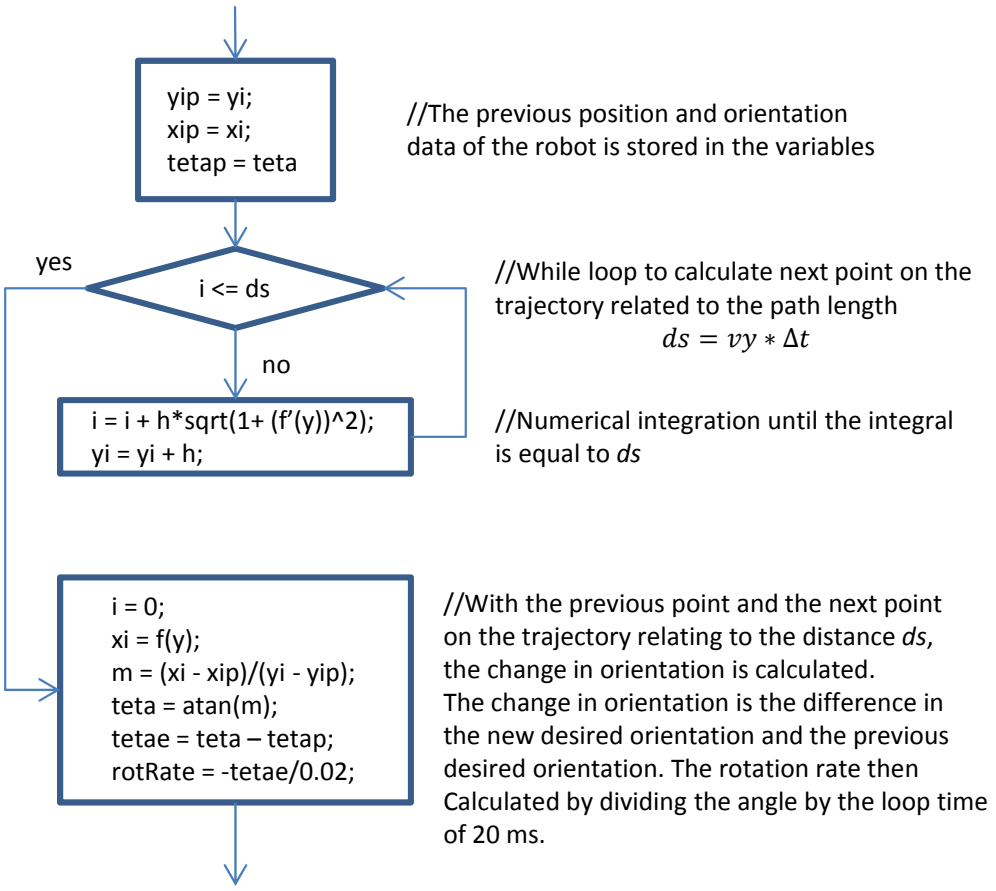


Figure 5.7: Flow diagram of the proposed algorithm for curved trajectory tracking

$$
\begin{aligned}
ds^2 &= dx^2 + dy^2 \\
\frac{ds^2}{dx^2} &= 1 + \frac{dy^2}{dx^2} \\
ds &= \sqrt{1 + \frac{dy^2}{dx^2}}\,\mathrm{d}x \\
s &= \int_a^b \sqrt{1 + \frac{dy^2}{dx^2}}\,\mathrm{d}x \\
s &= \int_a^b \sqrt{1 + (f'(x))^2}\,\mathrm{d}x
\end{aligned}
\tag{5.1}
$$

The proposed algorithm for curved trajectory tracking is shown in Figure 5.7 and inside the *while* loop in Figure 5.2, before the errors in vx, vy and angular speed gets calculated. This algorithm is executed every 20 ms.

The desired robot velocity in the x-direction is vx=0 and in the y-direction is the constant chosen velocity vy. With the proposed algorithm an error in the angle from the previous orientation to the new orientation is calculated. This angle error is used to determine the new desired angular speed of the robot, given the duration of the *while* loop of 20 ms. The algorithm is simulated and tested and will be discussed in Section 6.4.

5.5. Client Program

The central controller (OFC) connects through the Player proxies to the robot with the client program and was discussed in section 2.5.2. The messages is sent and received with the use of this client program to control the robots on the field. The main classes used in the client program is `Control`, `Robot` and `GLSoccerview`. The SSL-Vision software contains a Graphical User Interface (GUI) that is used to display the field and the robots with the ball on the field. This software was developed under the GNU General Public License (GPL) v3 and distributed. The `main` function is called from this GUI which in turn creates the `Vision`, `Control` and `Robot` objects used to control each individual robot. The main function receives the information captured by the vision system and calls the functions needed in the `Control` class. The control class then calls the functions needed in the `Robot` class from where the `Robot` class sends the Player messages through the proxies to the robots. This project did not require a full soccer match to be played, but rather the testing of individual actions that will be used in a game of robot soccer. The client program is still basic in the sense that the `main` function is only called by double clicking on the field in the SSL-Vision GUI. The `main` program was modified to send a series of fixed commands to the robot for the individual tests that were performed. The AI system is not yet implemented and it is up to the AI system to decide which commands should be sent at what time to which robot.

5.6. Summary

This chapter discussed the software used to control the robot on the field. The software developed by Holtzhausen (2012) and the software developed in this project were discussed. This included algorithms developed in this project to perform the tests to achieve the objectives stated for this project. The drivers that was used to access the hardware on the robots were discussed. The software was written to use these drivers to send the commands to the robot to perform the actions for the tests that will be discussed in chapter 6.

6. Robot Control and Testing

6.1. Introduction

This chapter will discuss the tests performed in order to achieve the objectives stated. The tests performed were all done on the robot soccer field of Stellenbosch University with the robot utilised in the current project. The robot used, with the kicker, dribbler and dribble sensors implemented is shown in Figure 6.1. Holtzhausen (2012) developed distributed control architecture that consists of a HLC on the robot. A KF was implemented on the robot that used camera data from the vision system and acceleration data obtained from sensors on the robot to predict the position, orientation and speed of the robot on the field. These predictions were used as input for the controllers on the robot. The HLC was tested with the robot by commanding the robot to make straight line movements on the field. This HLC was improved for more accurate movement by the robot. The tests performed by Holtzhausen (2012) was repeated with the improved control and compared. A new stronger gearbox was developed by Mouton (2013) and fitted onto the robot with the improved control and the tests again repeated to determine if the new gearbox improved the accuracy of the robot.

Also discussed herein, is the dribble control tests performed. It was done with the proposed algorithm and the IR sensors implemented above the dribbler bar. The aim of the project was to get an active ball handling system, again, done by improving the HLC on the robot and by implementing three IR sensors on the robot. Ball handling was tested with and without feedback from the IR sensors. The data was obtained for both tests and compared.

An algorithm for trajectory tracking of a curved path was proposed in section 5.4.6. The aim of the proposed algorithm was to correct the position error of the robot, due to the compensation of ball handling. Test data was obtained and will be discussed in more detail, later in this chapter.

The Player Project is used on the OFC together with the robot to communicate with each other and perform the tasks for the tests that will be discussed hereinafter.

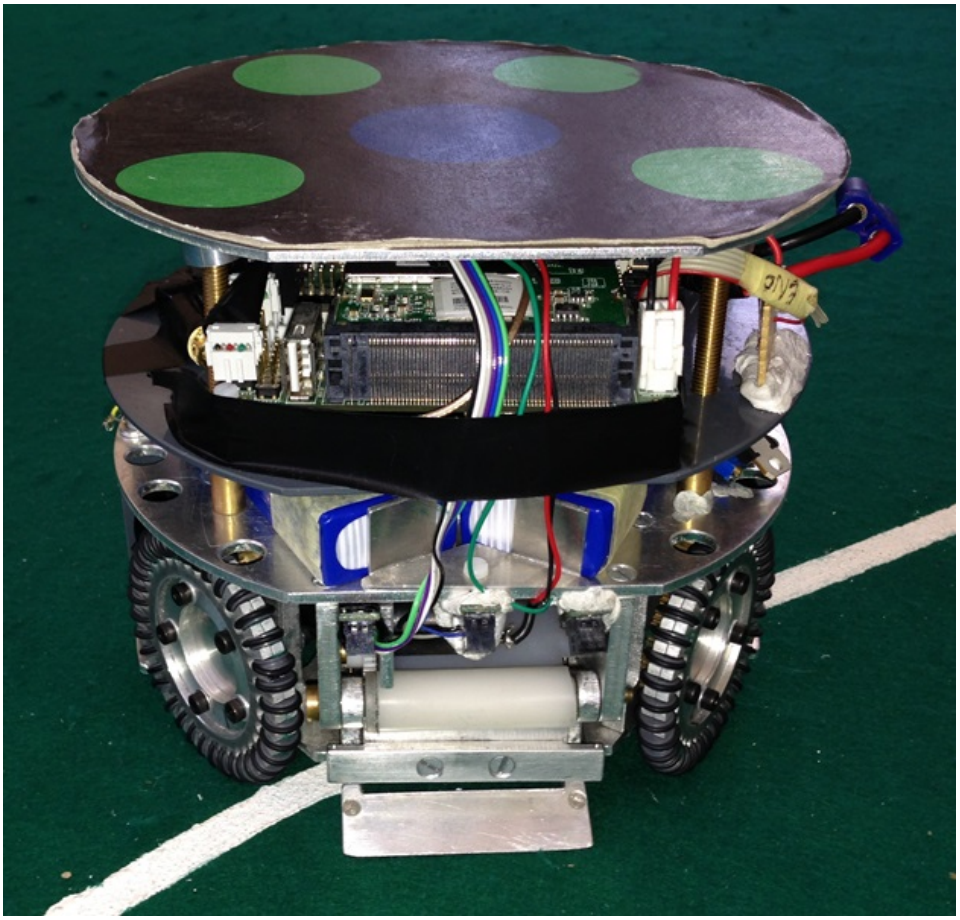


Figure 6.1: Soccer robot of Stellenbosch University

6.2. Improvements of High Level Controller

The HLC was designed by Holtzhausen (2012) which consisted of three seperate Proportional and Integral (PI) controllers. Two of these controllers were used for translational movement in the x- and y-directions and the other controller was used to control rotational movement of the robot. A KF was implemented by Holtzhausen (2012) to give the velocity estimates of the robot's x- and y-direction as well as the rotational speed of the robot by making use of the IMU on the robot and the position data obtained from the OFC. Holtzhausen (2012) showed the accuracy of the KF and it is assumed that the velocity estimates given by the KF is 100% correct. The current velocity estimate will be used to indicate the robot's measured velocity during the tests.

The position accuracy of the robot when moving was not ideal when the project was started and had to be improved in order to continue the tests performed in this project, mainly due to the PI parameters not optimally tuned. The behaviour of the translational PI controller of Holtzhausen (2012) is shown in Figure 6.2 where the velocity Set Point (SP), Controller Output

(CO) and the Measured Speed (MS) is shown. The velocity SP is only reached after approximately 1.4 seconds and since this is a velocity vs. time plot the integral over the time (the area under the graph) will give the desired distance and travelled distance of the robot. Explaining why the desired distance is not travelled and also why the robot is always short of the desired distance.

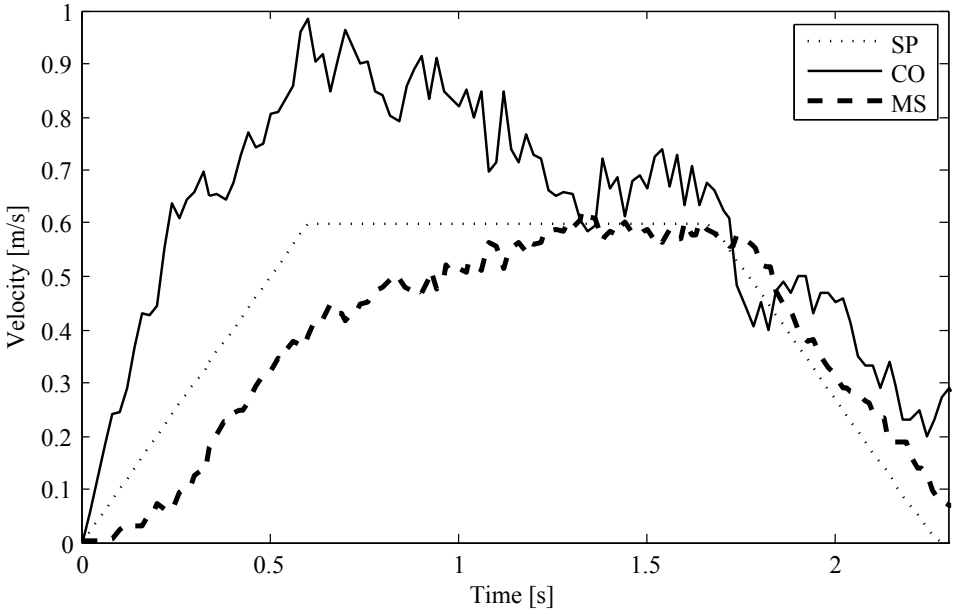


Figure 6.2: Behaviour of translational PI controller of Holtzhausen (2012)

The procedure was to improve the control parameters to get the robot to follow the speed profile more accurately, so that the robot will reach the desired position more accurately. It was also decided to include differential control to help meet the SP. The method for PID tuning that was used is the Zigler Nichols method. This method could however not be used for optimally tuning the control parameters. The Zigler Nichols method was only used as a starting point for tuning these control parameters. The optimal parameters was obtained through a series of trial and error tests. There is a lot of noise in the velocity estimate from the Kalman filter and this made the tuning of the PID parameters difficult. A pole was added in the control loop with the hope to reduce the effect of noise, but it had no significant effect. The derivative gain parameter was made extremely small since derivative control is very sensitive to measurement noise. The behaviour of the translational PID controller is shown in Figure 6.3 where it can be seen that the desired velocities are followed more accurately, thus improving the desired distance travelled by the robot.

Holtzhausen (2012) performed two types of tests to show the control of the robot by recording the position data of the robot while moving. The first test was to command the robot to move in a square, 1 m by 1 m, without rotating. The second test was to command the robot to move 1 m in a polar coordinate

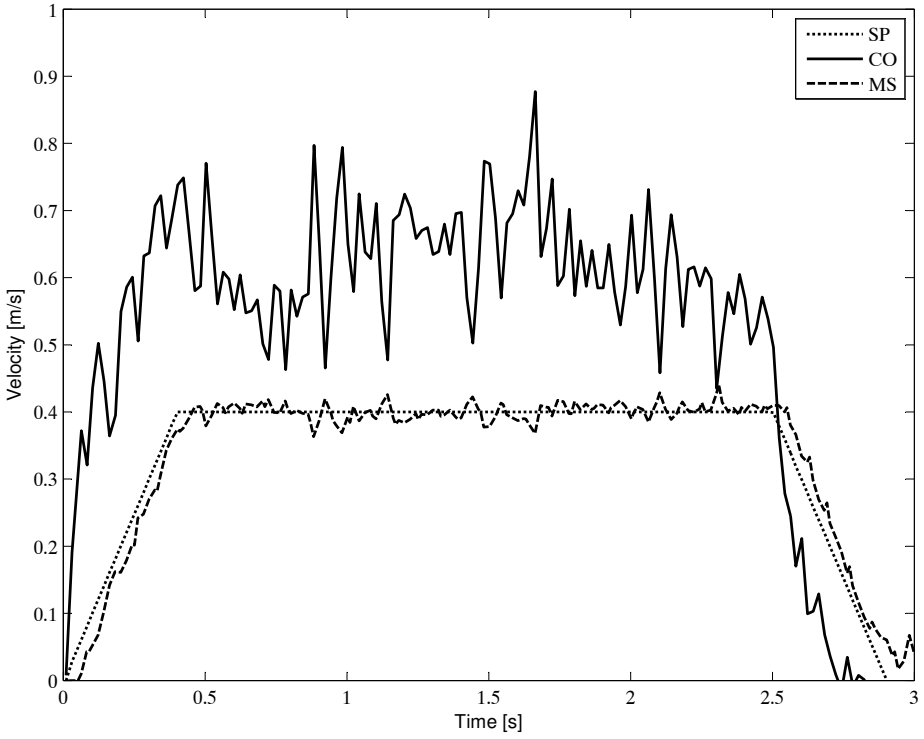


Figure 6.3: Behaviour of translational PID controller

from 0° to 315°, with incrementing the angle with 45°. Both of these test were done with the improved PID controllers to show improvements made to the previous system.

6.2.1. Straight Line Trajectory Tracking

The first test of Holtzhausen (2012) is shown in Figure 6.4 where the 1 m by 1 m square test was performed. The robot was commanded to travel 1 m in four directions to complete a square and desirably finish on the same place it started. The robot received distance commands and not destination commands which can result in an error accumulation. The same test was done with the second iteration gears (3D printed) and the new gears (phosphor bronze wire cut) both with the improved PID controllers of this project, see Figure 6.5 and Figure 6.6. These plots show the position of the robot during test where the position data is recorded from the KF data.

It is evident by comparing Figures 6.4 and 6.5 that the improved PID controllers are far more position accurate than the old ones used by Holtzhausen (2012). The second iteration gears were used by Holtzhausen (2012) as well as for the test shown in Figure 6.5. The second iteration gears were not very strong and durable and mechanical imperfection had an effect on the results of the tests. The third iteration gears (discussed in section 3.2.2) were designed

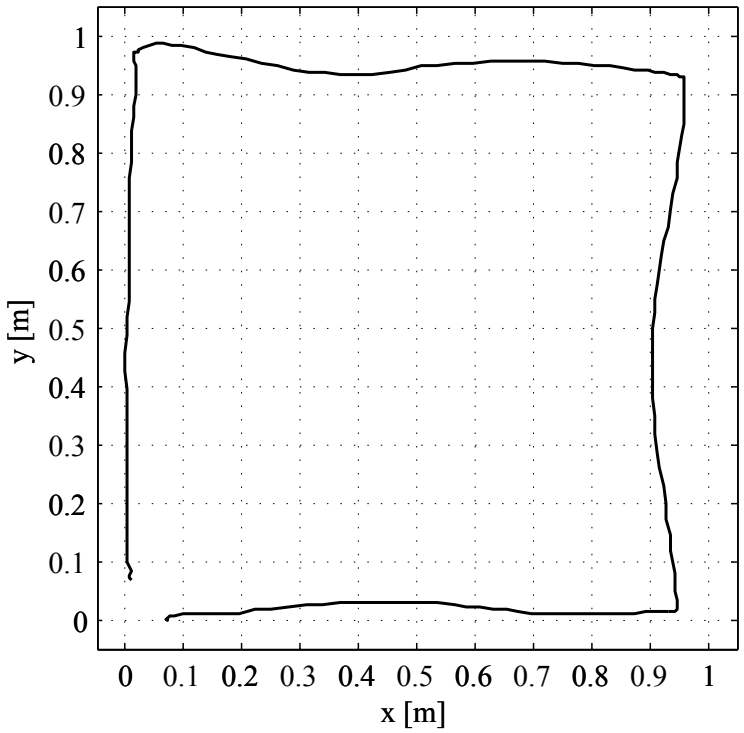


Figure 6.4: Square command tracking of Holtzhausen (2012)

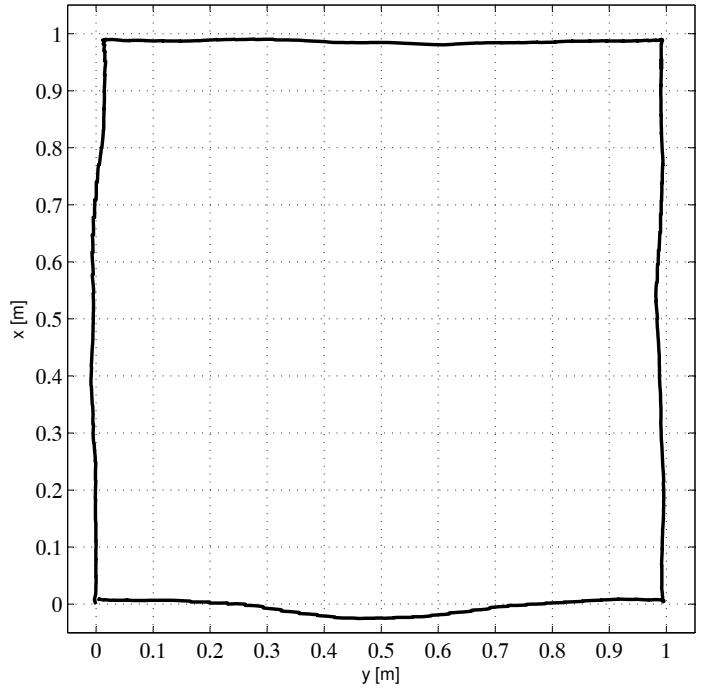


Figure 6.5: Square command tracking with the second iteration gears

and manufactured Mouton (2013) and these were implemented on the robot. The same test was repeated with these new gears and the same improved PID controllers. The result is shown in Figure 6.6 and it is seen that there is a slight improvement of the trajectory tracking.

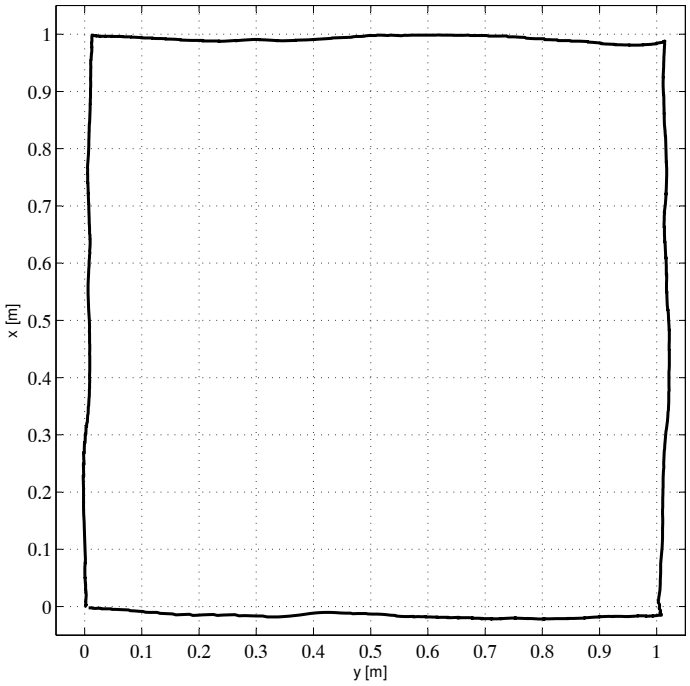


Figure 6.6: Square command tracking with the new gears

6.2.2. Robot Direction and Radial Deviation

The second test performed by Holtzhausen (2012) was to show the robot direction and radial deviation when commanded to follow a specified trajectory. Holtzhausen (2012) repeated a test done by Smit (2011) to show the improvements when using a HLC. The test was done with the robot and commanding it to move 1 m in a radial distance from 0° to 315° with increments of 45°. With this test the robot orientation was in the 90° direction as in Figure 6.7. The robot orientation was kept the same for all directions. The robot's position was recorded with the use of the KF data over time. The test performed by Holtzhausen (2012) is shown in Figure 6.7 and each direction was done 15 times.

The results obtained by Holtzhausen (2012) show the directional deviation in table A.1 as well as the radial distances travelled for each direction and is shown in table A.4. The directional error is normally distributed and can be analysed with the use of t-distribution and the upper and lower bounds were

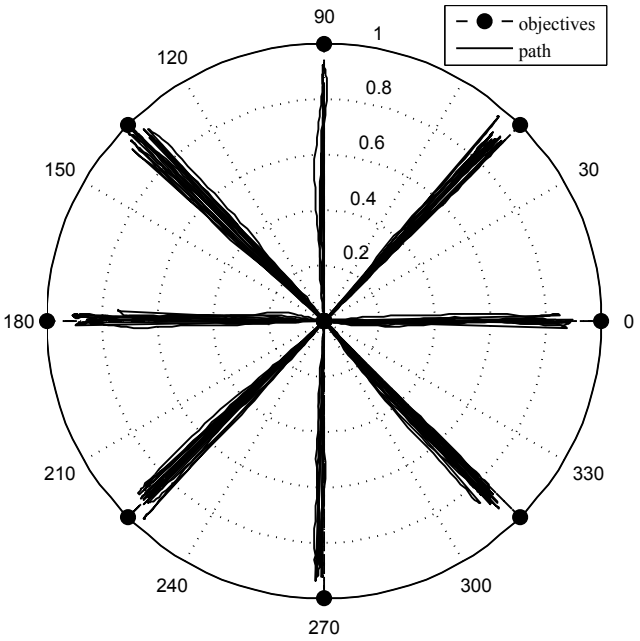


Figure 6.7: Directional deviation with HLC (Holtzhausen, 2012)

determined with 95% confidence (Holtzhausen, 2012). This test was repeated with the improved HLC and shown in Figure 6.8. The directional deviation is shown in table A.2 and the radial distance travelled is shown in table A.5. By comparing the radial deviation results in table A.1 and table A.2, it is clear that there is an improvement in the standard deviation in each direction except for the 225° direction. By looking at the average standard deviation in table A.1 and table A.2 there is a 30% reduction with the new HLC. The smallest standard deviation in table A.1 is 0.97 degrees and is reduced to 0.54 degrees by looking at table A.2 which is a 44% improvement. There is a 95% confidence that the directional deviation will never be larger than 1.83 degrees which is a 29.6% improvement to the 2.6 degrees of Holtzhausen (2012).

The test was again repeated with the third iteration gearbox that was implemented on the robot and the same improved HLC. There was a further improvement to the directional deviation when looking at table A.2 and table A.3. The average standard deviation was improved by 15.7% from 0.902 degrees to 0.76 degrees with the new gearbox and the biggest error was reduced from 1.83 degrees to 0.67 degrees which is a 63% improvement from the old gears to the new gears.

The data of the directional deviation for the three separate tests are all put together in a Box-and-Whisker Plot to get an overall picture of the magnitude of improvement and is shown in Figure 6.10. It is much easier to see the improvements on a single plot.

The radial distance travelled with the command of 1 m of Holtzhausen

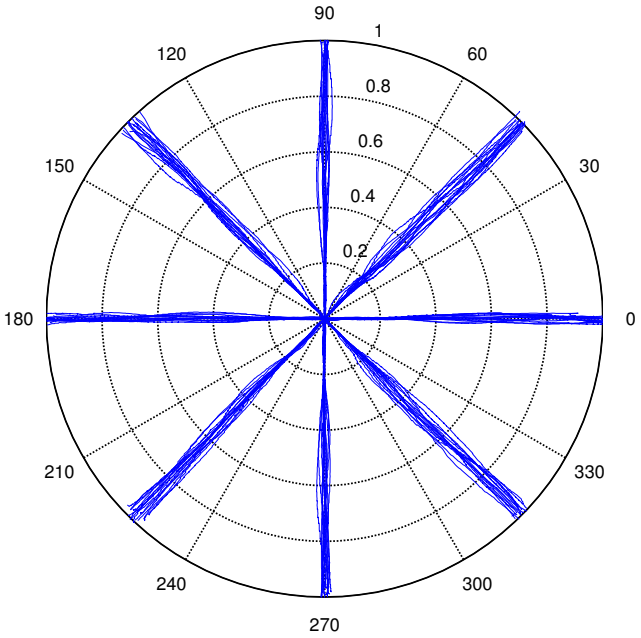


Figure 6.8: Directional deviation with new HLC and second iteration gearbox

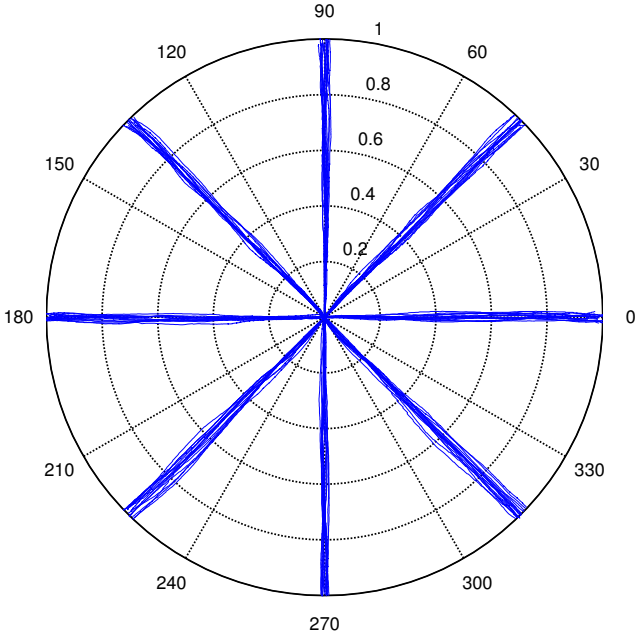


Figure 6.9: Directional deviation with new HLC and third iteration gearbox

(2012) is shown in table A.4 and the radial distance travelled with the improved HLC and second iteration gearbox is shown in table A.5. By comparing the mean error of these two tests it is clear that there is a significant improvement with the improved HLC. This gave an improvement of 90.8% of average mean error and an improvement of 60.5% in average standard deviation.

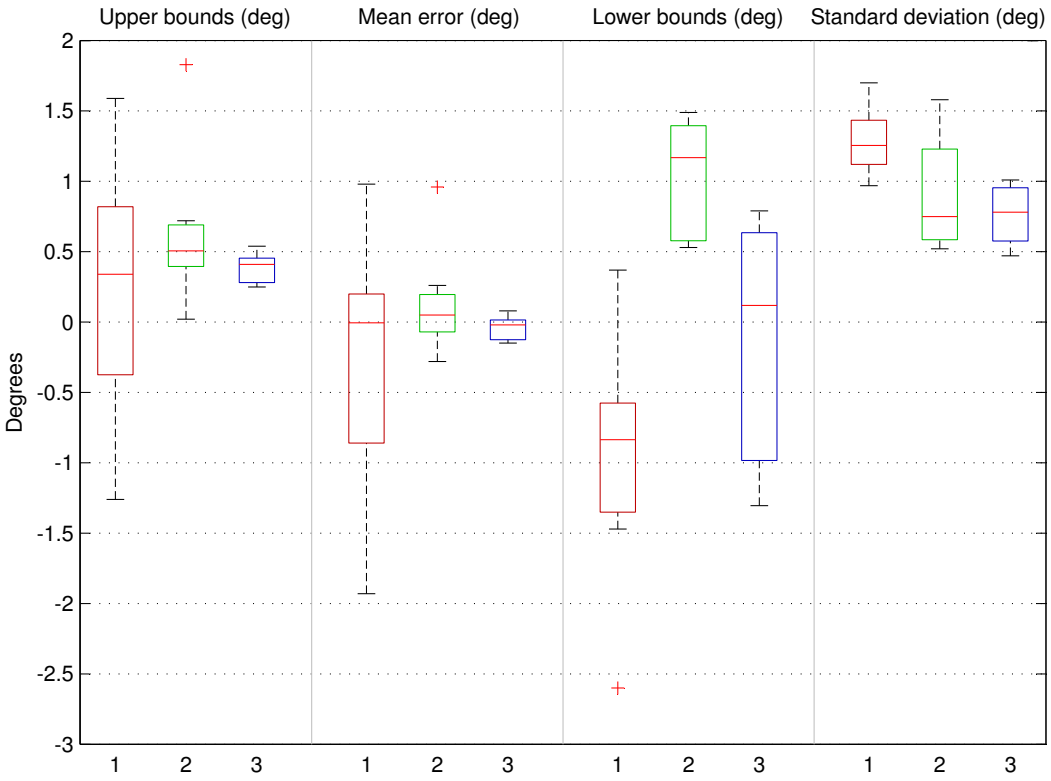


Figure 6.10: Directional deviation of the three tests in a Box-and-Whisker Plot

The results of the radial distance travelled with the improved HLC and the third iteration gearbox is shown in table A.6. When comparing table A.5 with table A.6 there is an improvement in some directions with the third iteration gearbox. There is a 32% improvement in the average mean error and a 34% improvement in the average standard deviation when comparing the third iteration gearbox with the second iteration robot. These three seperate tests are again plotted in a Box-and-Whisker Plot to easily compare the results and is shown in Figure 6.11.

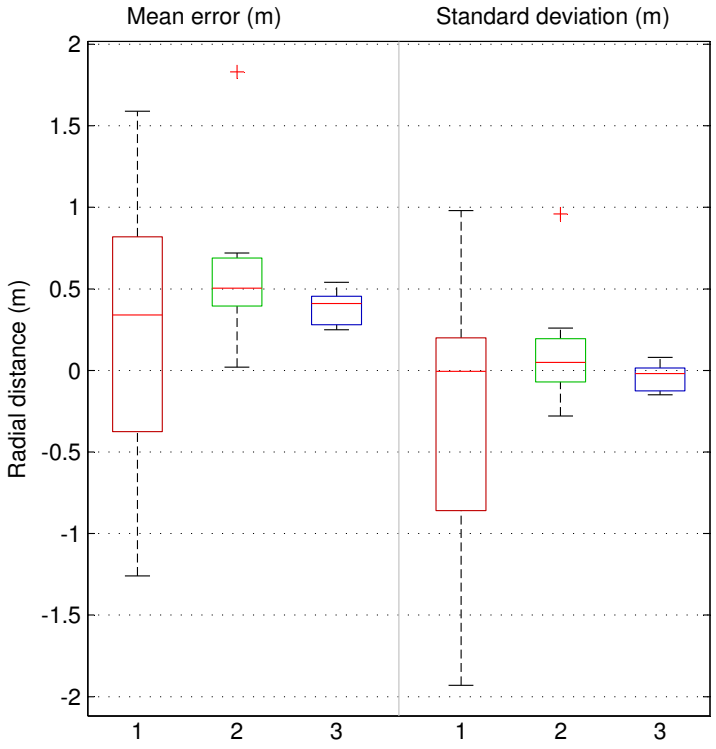


Figure 6.11: Error in radial distances achieved in the three tests in Box-and-Whisker Plot

6.3. Testing of Ball Handling System

Robot control in a straight line was tested in the previous section and the capabilities of the robots shown. This was necessary to ensure the robot can accurately follow a straight line. With unreliable straight line movement no ball control can even be considered for implementation. The dribbler mechanism was tested in section 4.3 and the information used to get the best performance from the dribbler. It was seen from the results that the dribbler bar performed at its best when the speed of the dribbler bar was high. During these tests the voltage of the dribbler motor was on 9.5 V resulting in a very fast rotation speed. In this section the implementation of the active ball handling system is implemented and tested.

6.3.1. Ball Handling without Sensor Feedback

In this subsection the tests that were done to test the dribbling ability of the robot without feedback from the dribbling sensors will be discussed. The sensors were implemented and activated, but not used for feedback and rather used to determine at what time the ball was lost in order to determine the distance travelled by the robot correlating to that time. The robot was placed on the field with the dribbler activated and the ball was placed on the dribbler bar. The orientation of the robot was determined by the camera and then the robot was commanded to move 3 m in the current orientation, thus moving in a straight line. The moment the ball was out of the line of sight of all the sensors, the ball was lost. The moment this happened the time was recorded to determine the distance the robot travelled with the ball. This was done 30 times and the distances recorded and shown in the histogram in Figure 6.12. A summary of the data is shown in table 6.1.

Table 6.1: Distance travelled with no sensor feedback

Maximum distance (m)	1.6231
Mean distance (m)	0.8996
Minimum distance (m)	0.3780
Standard deviation (m)	0.3128

From the histogram in Figure 6.12 it is seen that most of the occurrences were where the robot lost the ball before dribbling 1 m. The confidence of accurate dribbling is too low for a game of soccer and the ball can be lost when dribbling. This should be improved to have more confidence that the robot will keep control of the ball when commanded to dribble to a certain location on the field.

6.3.2. Ball Handling with Sensor Feedback

The test discussed in section 6.3.1 found that the ability of the robot to dribble with the ball is not satisfactory when no feedback is given about the position of the ball on the dribbler bar. The next test is again to move in a straight line, but this time with feedback from the dribble sensors and the algorithm described in flow diagram of Figure 5.3.

The robot was commanded to move 3 m in the starting orientation and the algorithm should correct the robot when the ball is sliding to the left or right of the center of the dribbler bar. The dribbler bar is activated and the ball placed on the dribbler bar before moving the 3 m.

This was done 30 times to get a significant data set. For all the 30 tests the robot had full control and not once was the ball lost. The robot moves 3 m in its y-direction and the velocity of the robot in its x-direction is adapted according to the position of the ball sensed by the dribble sensors. In other words the robot will move sideways when performing this dribble compensation. After a 3 m travel there is some error in the x-distance travelled since the robot had to correct to stay behind the ball. This error is shown in a histogram plot in Figure 6.13. A summary of the error data is shown in table 6.2.

This is a significant improvement in dribbling control compared to the test when the dribbling sensors were not used for feedback and there is 100% confidence for this data set that the robot could dribble the 3 m. Due to the correction of the ball the error in the robot x-direction is not that small. This is the error in the desired end position of the robot. This error is over a distance of 3 m and this would rarely be the distance needed to dribble

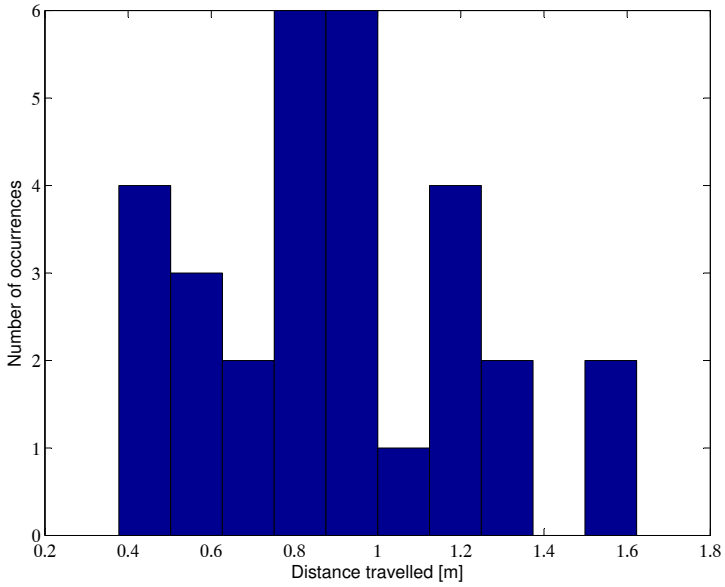


Figure 6.12: Histogram of distance dribbled with no sensor feedback

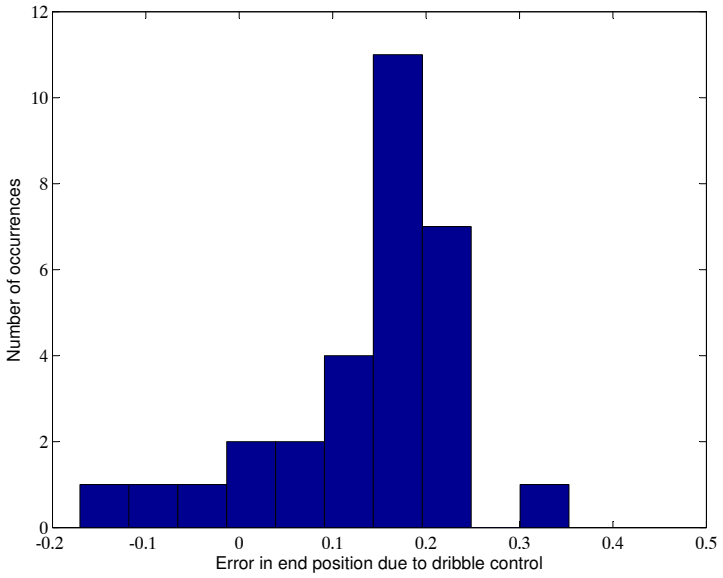


Figure 6.13: Histogram of error in robot x-direction due to dribble correction

Table 6.2: Error in x-distance due to dribble correction

Maximum error (m)	0.353
Mean error (m)	0.138
Minimum error (m)	-0.171
Standard deviation (m)	0.107

by a robot in a soccer match. This was observed from videos of previous RoboCup tournaments. This error gets smaller as the required dribble distance is reduced. This error was acceptable given that the robot is now able to dribble with full confidence.

6.4. Trajectory Tracking of a Curved Path

A proposed algorithm was explained in section 5.4.6 for the omnidirectional robots to follow a curved trajectory. This algorithm was tested on the robots of Stellenbosch University. This was done to see the capabilities of the robot to follow a curved path. The dribble test in section 6.3.2 was very good except for a small error in the robot x-direction due to the correction of robot position for dribbling. This error was about 0.2 m most of the time.

A sinusoidal path was chosen as a trajectory to be followed by the robot. The reason being that a sine wave is very well defined and easy to differentiate. The robot should correct the error in its x-direction while being behind the ball with the dribbler spinning all the time. The orientation of the robot should be the same at the start and finish of the dribble correction.

A simulation was done to test the algorithm explained in section 5.4.6. The error in x-direction to be corrected was chosen as 0.2 m. The length of the path should be long enough so that the orientation adjustment is not too large for every step. This aggressive steering will result in losing the ball. The proposed path will be half a cycle of a sine wave in which the 0.2 m error needs to be corrected. The equation to define the path will have the form shown in equation 6.1.

$$x = A\cos(B \cdot y) + c \tag{6.1}$$

where A is the amplitude, B the angular frequency and c the vertical shift on the x-axis.

The amplitude A is then 0.1 m (error in x-direction divided by two) and B should be calculated. It is chosen that the error in the x-direction should be corrected within 0.8 m. Due to this the proposed path should be half a sine wave cycle, B is calculated as 3.927. This gives the equation of the proposed path as in equation 6.2.

$$x = -0.1\cos(3.927y) + 0.1 \tag{6.2}$$

The derivative of equation 6.2 is shown in equation 6.3.

$$\frac{dx}{dy} = -0.3927\sin(3.927y) \tag{6.3}$$

Equations 6.2 and 6.3 is used in the algorithm of Figure 5.7 to generate the simulation shown in Figure 6.14. The blue line indicates the plot of equation 6.2 and the red line shows how the simulation follows the real equation.

The simulation in Figure 6.14 gave a very satisfying result and the algorithm was implemented on the robot. The robot was commanded to drive in the current orientation until the top speed of the velocity profile was reached and then the robot was given the constant velocity and the rotation speed as calculated in the algorithm in Figure 5.7. This test was done 30 times and

the position of the robot recorded from the Kalman filter on the robot as done with all the previous tests in this chapter. The recorded positions are relative to the starting position of the robot in the camera axis frame and the results are shown in Figure 6.15. Here the black line indicates the desired path that should be followed by the robot and the blue lines indicate the recorded positions of the robot. This test was for now not intended to be done while dribbling, but rather to test the capabilities of the robot following a path. This algorithm would have been implemented together with the dribble control sensors to ensure the robot ending at the desired end destination with full control of the ball. Due to time limitations this could not be done.

The statistical data of the 30 tests are shown in table 6.3. This data relates to the errors in the x-direction assuming an initial error of 0.2 m with the robot trying to reduce the position error to zero.

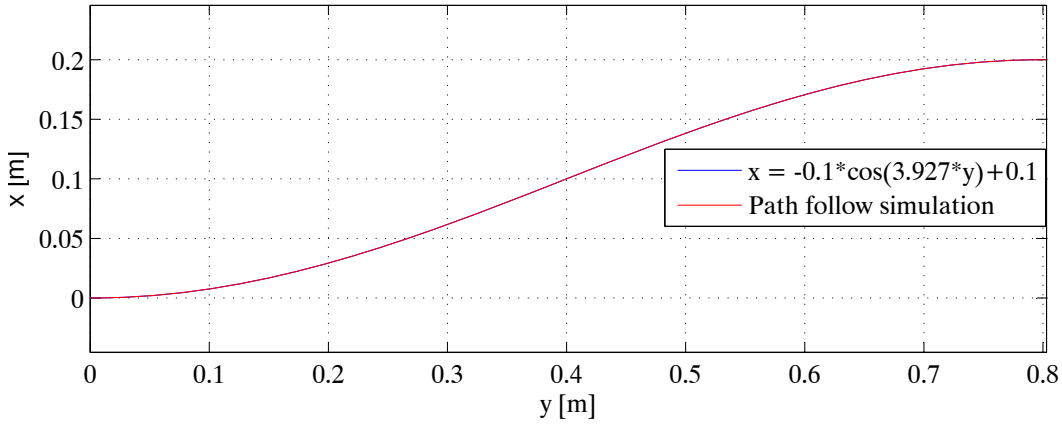


Figure 6.14: Simulation of path

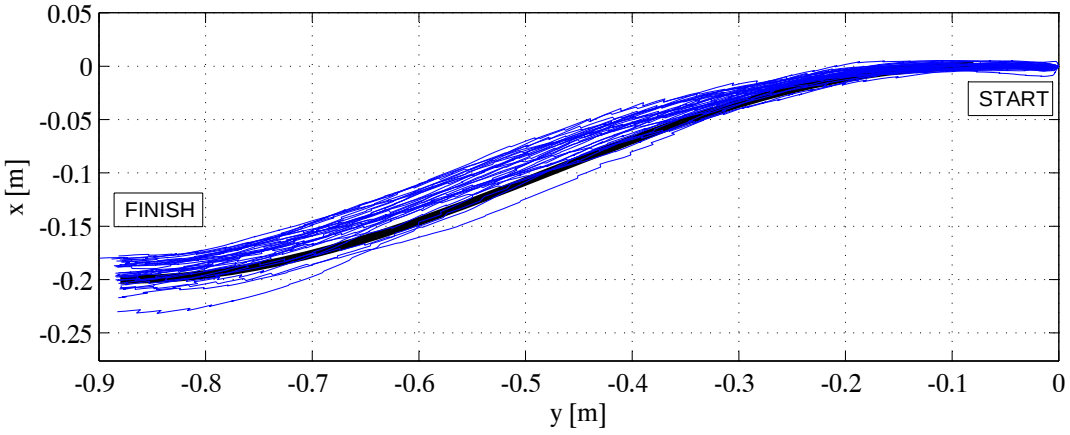


Figure 6.15: Results of path

Table 6.3: Error in x-distance after path following correction

Maximum error (m)	0.0221
Mean error (m)	0.00543
Minimum error (m)	-0.0301
Standard deviation (m)	0.01207

6.5. Summary

This chapter discussed the work done by Holtzhausen (2012) and the performance of the distributed control architecture. The improvements made to the HLC of Holtzhausen (2012) and the results were discussed to validate the significance. Furthermore the new gearbox design of Mouton (2013) was compared to the old gearbox by using the same improved control system and changing the gearboxes. This indicated that there was a slight improvement in accuracy of robot movement on the field with the new gearbox.

The IR sensors used for dribble control were implemented on the robot. Dribbling tests were done by the robot with and without feedback from the sensors. The capability of a robot to dribble a ball without sensor feedback was very poor and unreliable. With feedback from the IR sensors when dribbling there was a significant improvement with very good reliability.

The proposed curved trajectory tracking algorithm was tested on the robot and the results were discussed. This was the first time translational and rotational movement were combined to achieve curved paths with the robots of Stellenbosch University. The results were good and the statistical data was discussed.

7. Conclusion and Recommendations

7.1. Conclusion

Chapter 1 gave a brief introduction of the RoboCup soccer and more specifically the SSL league. The primary objective was stated in this chapter: to develop an active ball handling system for a robot in the SSL league. This project builds on previous work done by Smit (2011), Jacobs (2011) and Holtzhausen (2012). Smit (2011) designed and built the first iteration robot that was used as a robotic platform. The motor controller was developed by Smit (2011) and further improved by Jacobs (2011). The first iteration robot was able to execute basic movements with the use of a Low Level Controller (LLC) (encoder feedback from the motors). The first iteration robot was not within the size limits of the SSL rules. A second iteration robot was developed and built by a technician and was used by Holtzhausen (2012) to develop a distributed control system on the robot. This distributed control was developed to lower the amount of commands to be sent by the OFC and perform more calculations on the robot itself. Holtzhausen (2012) implemented a KF on the robot to estimate the position, velocity and orientation of the robot. This was achieved by using the data from the vision system and also the acceleration data of the robot from the IMU sensors on the robot. The estimator output was then used as input for the HLC on the robot.

To achieve the primary objective of this project, the project was divided into eight sub objectives. These objectives and how they were achieved will be discussed. Before any of these objectives could be achieved, the Player Project together with the drivers and the software written by authors mentioned above had to be fully understood. The Player Project together with the drivers and classes used to control the robot over a wireless network was discussed in section 2.4. With this knowledge the software could be developed further to achieve the objectives.

The first objective was to test the distributed control system developed by Holtzhausen (2012). This was done and it was observed that the robot movement was not ideal every time. This was mainly due to a calculation

error and the PI controllers not being optimally tuned. Accurate movement is required to develop a robot with an active ball handling system. The second objective was to correct this error and to improve the controllers on the robot. This was done and tested in chapter 6. The results were satisfying and showed a significant improvement in movement accuracy on the field. The robot is able to perform the needed accurate straight line movements needed when dribbling the ball.

The third objective was to implement the dribbler on the robot and test it. Implementation and testing was done for the first time at Stellenbosch University. The dribbler was switched on and a ball was placed in front of the dribbler. The electronic circuit of the dribbler motor used in this test was also discussed. The fourth objective was to characterise the dribbler. A series of tests were done and discussed in section 4.3. It showed that a higher rotation speed of the dribbler bar showed better controllability. When catching a ball, higher approach speeds of the ball resulted in a lowering of controllability and an increase in the approach angle of the ball also lead to less controllability. These results were used to set up the dribbler to give the robot the best control over the ball when dribbling.

The fifth objective was to implement the kicker mechanism on the robot. This included the mechanical parts and electronic circuits. This was also the first time the kicker mechanism was implemented. The kicker was tested and characterised after a series of tests which was discussed in section 4.2 as stated by the sixth objective. This showed that the kicker can be controlled to vary the speed with which a ball is kicked by changing the voltage of the discharge capacitors and the time current is flowing through the coil of the solenoid. These results will be used by the AI system in the game when the ball should be kicked at different speeds.

Objective seven stated that sensors should be implemented onto the robot to improve control over the ball when dribbling. Tests were done and discussed in section 6.3.1 that showed the limited ability of the robot to dribble a ball without sensor feedback. This motivated the need to implement sensors to provide information about the ball position when dribbling. The choice was to use three IR sensors above the dribbler bar to detect the position of the ball on the dribbler bar. A proposed algorithm to ensure accurate dribbling was implemented and tested. The improvements of the robot's dribbling ability were motivated with another set of tests discussed in section 6.3.2 that was compared to the test data when the no sensor feedback is received. There was a significant improvement to the robot's ability to dribble a ball over a distance in a straight line. There was however some error in final position of the robot due to compensation by the robot to keep control of the ball that was discussed.

The last objective was satisfied with the implementation of a proposed algorithm for curved trajectory tracking. Trajectory tracking and path following were discussed in section 2.3 and the proposed algorithm discussed in

section 5.4.6. The proposed algorithm is a type of curvature steering method where the robot's forward translational speed is kept constant and rotational speed calculated stepwise to adapt the orientation of the robot to stay on the curved trajectory. This was the first time the robots of Stellenbosch University attempted to combine translational and rotational movement with the objective to perform curved paths. This algorithm was tested and discussed in section 6.4. The results obtained showed the desired path and the actual robot position. The error in distance between the robot and the desired path was below 3 cm which is a very satisfactory result. This trajectory tracking algorithm can be used to correct the error in final position due to dribble correction. It can also be further developed in obstacle avoidance algorithms where the robot makes use of curved paths to avoid obstacles on the field.

To conclude the objectives were all achieved to develop an active ball handling system to meet the requirements of this project.

7.2. Recommendations

In this section a discussion will be given about the recommendations for continuation of this project towards the goal of developing a team of robots to compete in the RoboCup SSL league.

The trajectory tracking implemented in this research gave the robot the ability to follow a curved path. However with trajectory tracking an error in position or orientation could lead to an accumulation of errors with the robot not arriving at the desired end position. Therefore it is suggested to look at path following algorithms that drive the robot towards the path based on the current location and orientation of the robot. The "virtual robot" approach is a very popular approach in path following and should be further researched for better results of curved paths.

Obstacle avoidance was not considered in this project as it was not part of the scope. There exist algorithms for combined path following and obstacle avoidance, for instance Lapierre *et al.* (2007). This is necessary in environments of multi-robots cooperation. It is recommended that research should be done on this field when attempting to continue the work of this project.

The SSL-vision was set up on the field with all the required hardware. The system was however only set up for one half of the field and the other half was not yet needed. The camera data for the other half of the field should be merged to get the information on the field as a whole. This will be required in a game where the whole field is used.

The cameras that were used with the SSL vision software at Stellenbosch University used ambient light in the laboratory. This is sufficient when the weather conditions are sunny and no change in light occurs. The cameras are calibrated for certain ambient light conditions and tests should then be done in these conditions for optimal results. In the winter time when it is cloudy it

is difficult to calibrate the cameras for a specific light condition since there is constantly a change in light on the field. It is recommended that these tests are performed in a dark room and that a constant light source with LED's is implemented to ensure the same light conditions during every test.

Finally it is essential to have a robot that can perform accurate movement on the field. This research showed that it is also necessary to have dribbler sensors on the robot that provide feedback of ball position on the dribbler bar to perform accurate dribbling and remain in contact with the ball.

A. Test Statistics

A.1. Directional Deviation of HLC

The directional deviation refers to the tests that was discussed in section 6 which compared the difference between the old PID controllers and the new PID controllers and also the difference the third iteration gearbox had on the movement of the robot.

Table A.1: Directional deviation of HLC Holtzhausen (2012)

Desired direction	Upper bounds (deg)	Mean error (deg)	Lower bounds (deg)	Standard deviation (deg)
0	0.45	-0.05	-0.55	1.08
45	1.59	0.98	0.37	1.51
90	0.23	-0.32	-0.86	1.16
135	0.88	0.04	-0.81	1.70
180	0.76	0.08	-0.60	1.36
225	-0.22	-0.72	-1.23	1.20
270	-0.53	-1.00	-1.47	0.97
315	-1.26	-1.93	-2.60	1.31
Average	0.24	-0.36	-0.97	1.29

Table A.2: Directional deviation of HLC with new controller and second iteration gearbox

Desired direction	Upper bounds (deg)	Mean error (deg)	Lower bounds (deg)	Standard deviation (deg)
0	0.02	-0.28	-0.58	0.54
45	0.46	-0.02	-0.50	0.87
90	0.55	0.26	-0.02	0.52
135	0.72	-0.12	-0.95	1.51
180	0.35	0.0	-0.35	0.63
225	1.83	0.96	0.09	1.58
270	0.44	0.10	-0.25	0.63
315	0.66	0.13	-0.39	0.95
Average	0.63	0.13	-0.37	0.902

Table A.3: Directional deviation of HLC with new controller and third iteration gearbox

Desired direction	Upper bounds (deg)	Mean error (deg)	Lower bounds (deg)	Standard deviation (deg)
0	0.25	-0.15	-0.55	0.73
45	0.54	0.01	-0.52	0.96
90	0.45	0.08	-0.28	0.65
135	0.42	-0.04	-0.50	0.83
180	0.28	0.02	-0.25	0.47
225	0.46	-0.11	-0.67	1.01
270	0.28	0.0	-0.28	0.502
315	0.40	-0.14	-0.67	0.95
Average	0.38	-0.04	-0.46	0.76

A.2. Radial Distance Achieved by HLC

The radial distance achieved by the robot refers to the tests that were discussed in section 6. These results show the difference of the old PID controllers and the new PID controllers and also the effect of the third iteration gearbox.

Table A.4: Radial distance achieved by HLC of Holtzhausen (2012)

Desired direction	Mean error (m)	Standard deviation (m)	Mean error (%)
0	0.132	0.020	13.25
45	0.080	0.041	8.01
90	0.103	0.043	10.30
135	0.079	0.027	7.86
180	0.124	0.017	12.44
225	0.081	0.055	8.06
270	0.111	0.049	11.09
315	0.129	0.025	12.91
Average	0.105	0.035	10.49

Table A.5: Radial distance achieved by new HLC and second iteration gears

Desired direction	Mean error (m)	Standard deviation (m)	Mean error (%)
0	0.0022	0.0155	0.22
45	0.0026	0.0120	0.26
90	0.0102	0.0132	1.02
135	0.0104	0.0124	1.04
180	0.0072	0.0138	0.72
225	0.0197	0.0177	1.97
270	0.0107	0.0122	1.07
315	0.0141	0.0136	1.41
Average	0.0096	0.0138	0.96

Table A.6: Radial distance achieved by new HLC and third iteration gears

Desired direction	Mean error (m)	Standard deviation (m)	Mean error (%)
0	0.0098	0.0103	0.98
45	0.0061	0.0084	0.61
90	0.0083	0.0082	0.83
135	0.0096	0.0126	0.96
180	0.0071	0.0111	0.71
225	0.0052	0.0086	0.52
270	0.0053	0.0105	0.53
315	0.0003	0.0028	0.03
Average	0.0065	0.0091	0.65

B. Coordinate Transformation

There are three different coordinate frames when controlling a robot on the field, which is the coordinate frame of the IMU, the robot frame and the camera coordinate frame. Rotation of coordinate frames is done by using a rotation matrix described by the roll pitch and yaw angles about the reference frame. The roll (ϕ) is the rotation around the z-axis, pitch (β) is the rotation around the y-axis and yaw (ψ) is the rotation around the x-axis. The rotation matrix is shown in equation B.1.

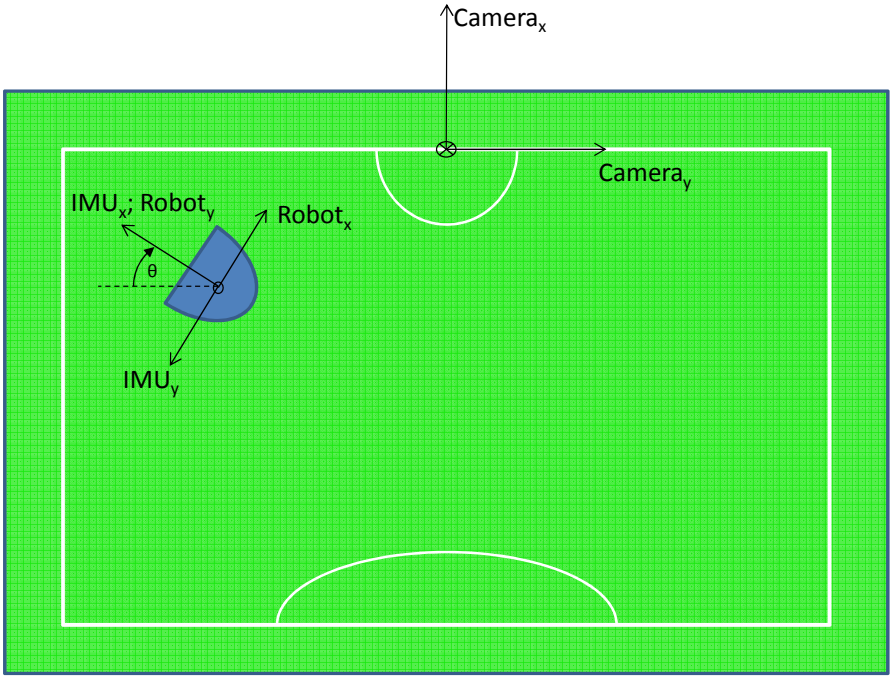


Figure B.1: Coordinates on the soccer field

$$
\begin{aligned}
R &= R_{z,\phi} \cdot R_{y,\beta} \cdot R_{x,\psi} \\
R &= \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}
\end{aligned} \tag{B.1}
$$

The rotation matrix in equation B.1 will be used to transform the IMU data to the coordinate frame of the camera. This is done by rotating the IMU data by $\pi/2+\theta$ around the z-axis and then rotating by an angle of π around the y-axis and no rotation around the x-axis. The angle θ is the rotation of the robot around its z-axis. In this case $\phi = \pi/2+\theta$, $\beta = \pi$ and $\psi = 0$ and the rotation matrix shown in equation B.2.

$$
\begin{aligned}
R &= \begin{bmatrix} \cos(\theta+\pi/2) & -\sin(\theta+\pi/2) & 0 \\ \sin(\theta+\pi/2) & \cos(\theta+\pi/2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
R &= \begin{bmatrix} -\cos(\theta+\pi/2) & -\sin(\theta+\pi/2) & 0 \\ -\sin(\theta+\pi/2) & \cos(\theta+\pi/2) & 0 \\ 0 & 0 & -1 \end{bmatrix}
\end{aligned} \tag{B.2}
$$

The global velocities received from the PID controllers need to be transformed to the coordinate frame of the robot for the motor controller to move the robot. This is done with another rotation matrix like the one in equation B.1. To go from the camera coordinate frame to that of the robot the camera coordinates are rotated by an angle θ around the z-axis, none around the y-axis and an angle π around the x-axis. Thus $\phi = \theta$, $\beta = 0$ and $\psi = \pi$ and shown in equation B.3.

$$
\begin{aligned}
R &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \\
R &= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & -1 \end{bmatrix}
\end{aligned} \tag{B.3}
$$

C. Source Code

C.1. GoTo

This section describes the C++ code that was written for the ball control algorithm. The velocity profile is generated for the robot speeds in the x- and y-directions in the global frame (vx and vy in the code). The speed of the robot in its x-direction is then adapted according to the ball position. This speed of the robot should then be transformed to the global axis frame and added to the original calculated desired global velocities (line 44 and 45).

```
1
2     //The code for ball control inside the main loop that gets
          executed
3     //every 20 ms
4     LS = rcservo_InPin(leftsensor);
5          MS = rcservo_InPin(middelsensor);
6          RS = rcservo_InPin(rightsensor);
7          if(!LS || !MS || !RS)
8          {
9               if(LS && !MS && RS)
10              {
11                   vxr = 0;
12                   printf("middlesensor\n");
13              }
14              else if (LS && !MS && !RS)
15              {
16                   //Robot moves to the right of its own axis system
17                   vxr = -0.2; //Speed of robot in its own axis
                         system
18                   printf("MR\n");
19              }
20              else if (!LS && !MS && RS)
21              {
22                   vxr = 0.2;
23                   printf("ML\n");
24              }
25              else if (LS && MS && !RS)
26              {
27                   vxr = -0.3;
28                   printf("rightsensor\n");
29              }
```

```
30              else //(!LS && MS && RS)
31              {
32                  vxr = 0.3;
33                  printf("leftsensor\n");
34              }
35          }
36          else
37          {
38              //ball not on dribbler
39              printf("Time ball was lost is %f\n", sampleTime);
40              vxr = 0;
41          }
42
43      //For ball control the desired robot x velocity should be
            transformed to global
44          vx = vx - vxr*cos(angle); //vx is the desired speed of
                robot in the global frame x-direction
45          vy = vy - vxr*sin(angle); //vy is the desired speed of
                robot in the global frame y-direction
```

ballcontrol.cc

List of References

Aicardi, M., Casalino, G., Bicchi, A. and Balestrino, A. (1995). Closed loop steering of unicycle like vehicles via Lyapunov techniques. *Robotics & Automation Magazine, IEEE*, vol. 2, no. 1, pp. 27–35.

Akhter, A., Butt, U.T., Azhar, M., Raza, S.M., Ahmad, T., Zia, G., Bhatti, M.S., Sohail, J., Arshad, M. and Imran, Z. (2013). Emenents 2013 Team Description.

Asama, H., Sato, M., Bogoni, L., Kaetsu, H., Mitsumoto, A. and Endo, I. (1995). Development of an omni-directional mobile robot with 3 DOF decoupling drive mechanism. In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2, pp. 1925–1930. IEEE.

Chaiso, K. and Sukvichai, K. (2011). Skuba Extended Team Description Paper.

Circuit, B. (2012 October). Dark sensor using transistor and phototransistor. Available at: `http://www.buildcircuit.com`

Coilgun (2012 October). How phototransistor operate. Available at: `http://www.coilgun.info/theory/phototransistors.htm`

Collett, T.H., MacDonald, B.A. and Gerkey, B.P. (2005). Player 2.0: Toward a practical robot programming framework. In: *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*.

Egerstedt, M., Hu, X. and Stotsky, A. (2001). Control of mobile platforms using a virtual vehicle approach. *Automatic Control, IEEE Transactions on*, vol. 46, no. 11, pp. 1777–1782.

Faulhaber (2012 June). Dc-motors series 1724. Available at: `http://www.faulhaber.com/`

Fukumoto, Y., Nakajima, M. and Masutani, Y. (2011). Odens 2011 Team Description. Tech. Rep., Osaka Electro-Communication University.

Gerkey, B., Vaughan, R., Howard, A. and Koenig, N. (2003*a*). The Player/Stage Project. *hosted at http://playerstage. sourceforge. net.*

Gerkey, B., Vaughan, R.T. and Howard, A. (2003*b*). The Player/Stage Project: Tools for multi-robot and distributed sensor systems. In: *Proceedings of the 11th international conference on advanced robotics*, vol. 1, pp. 317–323.

Goto, R., Asakura, T., Fujii, N., Matsuoka, K. and Mizuno, M. (2013). Kiks 2013 Extended Team Description.

Hahn, G.J. (1977). Some things engineers should know about experimental design. *Journal of Quality Technology*, vol. 9, no. 1, pp. 13–20.

Holtzhausen, D. (2012). *Devolpment of distributed control system for SSL soccer robots*. Master's thesis, Stellenbosch University.

Ishikawa, A., Yasui, K., Inagaki, T., Sawaguchi, H., Yasui, T., Murakami, K. and Naruse, T. (2011). RoboDragons 2011 Extended Team Description.

Jacobs, L. (2011). *RoboCup Small Size League: Motor Control and Sensory Feedback*. Master's thesis, Reutlingen University.

Kanjanawanishkul, K. and Zell, A. (2009). Path following for an omnidirectional mobile robot based on model predictive control. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3341–3346. IEEE.

Kramer, J. and Scheutz, M. (2007). Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, vol. 22, no. 2, pp. 101–132.

Lapierre, L., Zapata, R. and Lepinay, P. (2007). Combined path-following and obstacle avoidance control of a wheeled robot. *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 361–375.

Mouton, C. (2013). Transmissie ratkas ontwerp vir 'n sokker robot. Tech. Rep., Stellenbosch University.

Onsemi (2013 May). Lm317 adjustable output positive voltage regulator. Available at: `http://www.onsemi.com/pub_link/Collateral/LM317-D`

Oxford (2012 October). Definition of robot. Available at: `http://www.oxforddictionaries.com/definition/english/robot`

Palmera, A., Balankoa, J., Headb, C., Fraserc, J. and Luma, S. (2012). 2012 team description paper: Ubc thunderbots.

Pead, J. (2013). *Research, Design and Construction of a team of Small Size League Soccer Robots for RoboCup Soccer*. Master's thesis, University of Cape Town.

Petersen, B. and Fonseca, J. (2006). Player/stage. Tech. Rep., Department of Computer Science University of Copenhagen.

Playerstage (2012 July). The Player Project. Available at: `http://www.playerstage.sourceforge.net`

Purwin, O. and D'Andrea, R. (2006). Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, vol. 54, no. 1, pp. 13–22.

Roboard (2012 July). Roboard RB-100.
Available at: `http://www.roboard.com/RB-100.htm`

RoboCup (2012 October*a*). Laws of the robocup small size league 2012.
Available at: `http://www.robocup.org/robocup-soccer/small-size/`

RoboCup (2012 February*b*). Robocup soccer.
Available at: `http://www.robocup.org/robocup-soccer/`

Rojas, R. and Förster, A.G. (2006). Holonomic control of a robot with an omnidirectional drive. *KI-Künstliche Intelligenz*, vol. 20, no. 2, pp. 12–17.

Ruiz, M.S. and Weitzenfeld, A. (2006). Soccer dribbler design for the eagle knights robocup small size robot. In: *Robotics Symposium, 2006. LARS'06. IEEE 3rd Latin American*, pp. 34–40. IEEE.

Sharp (2013). Distance measuring sensor unit digital output (50 mm) type.
Available at: `http://docs-europe.electrocomponents.com/webdocs/0d1b/0900766b80d1bdd4.pdf`

Smit, A. (2011). *Development of an omni-directional robot for RoboCup Small Size League, designing for a distributed control architecture.* Master's thesis, Stellenbosch University.

Soetanto, D., Lapierre, L. and Pascoal, A. (2003). Adaptive, non-singular path-following control of dynamic wheeled robots. In: *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 2, pp. 1765–1770. IEEE.

Watugala, K., Sherback, M., McCabe, M., Byrne, G. and Lupashin, S. (2005). Cornell Big Red 2005 Team Description.

Zickler, S., Biswas, J., Luo, K. and Veloso, M. (2010). CMDragons 2010 Team Description.

-