# Cornell University RoboCup: Big Red Bots

# Final Design Document Team Brazil 1999

**Advisors:**

**Prof. Raffaello D'Andrea**

**Dr. Jin-Woo Lee**

**Volume I**

**May 1999**

## Cornell University RoboCup Team: Big Red Bots

226 Upson Hall, Mechanical & Aerospace Engineering,

Cornell University,

Ithaca, NY 14850,

USA

Tel: (607) 255 - 8424

Fax: (607) 255 - 1222

If you have any question about this documentation, please contact Dr. Jin-Woo Lee(jl206@cornell.edu), Prof. Raffaello D'Andrea(rd28@cornell.edu), or visit our web site, http://www.mae.cornell.edu/RoboCup.

## Section 1 Executive Summary

In competition the primary goal is to be number one. How one goes about reaching the coveted number one spot is what makes a team special. Utilizing a disciplined systems engineering design process, Team Brazil has constructed an excellent team of robotic soccer players. Throughout the entire academic year, Team Brazil focused on three primary goals: 1) To win RoboCup 1999, 2) to win future RoboCup competitions, and 3) to win the honor of representing Cornell University. Achieving the first step, Team Brazil has won the honor of representing Cornell at the international competition in Stockholm, Sweden.

The following pages capture the culmination of Team Brazil's efforts over the span of two full semesters. Through many man-hours of hard work, Team Brazil has constructed a fully autonomous system of soccer playing robots. Drawing from everyone's experiences and knowledge base, Team Brazil has cooperatively designed the current system.

The early weeks of the first semester were spent gaining an understanding of the complexities of the project, including the rules and regulations. Once the framing of the problem had taken shape, decisions were made based on brainstorming, initial analyses and many discussions of the system trade-offs. During subsequent meetings tasks were divided among team members who were interested in the subject matter and those who had experience. The extra time and thought put into our designs through the design process made the construction of our robots much less problematic. The most difficult aspect of the first semester was the fact that we didn't work with any real products.

Finally settling on the design early in the second semester, everyone set out to execute the team's plans. The electrical engineering team produced the hardware for the local control from their schematics and laid out the boards by hand. They also coded the microcontroller for local control and wireless communication. The mechanical engineering team manufactured the chassis, assembled the drivetrain, and assembled the entire robot. The artificial intelligence team refined and redefined their solutions to the problem of playing a soccer game.

Tackling the challenging problem of building a team of autonomous robot soccer players, Team Brazil has accomplished much and learned a great deal. The foundation is there to build upon. Continued success is the next step on the road to number one.

# Section 2 Table of Contents

## Section 3 List of Figures

## Section 4 Design Team

### Team Manager

Dennis Huang

### Artificial Intelligence

Salman Qureshi

Mike Smullens

### Communications/Local Control

Jason Oversmith

Alexander Lau

### Vision

Thomas Karpati

### Mechanical Design

Charles Poon

Alex Sepulveda

William Hegarty

Syaril Hussin

### Research and Development

Aris Samad-Yahaya

Christopher Lau

Aaron Delfausse

## Section 5 Context (Statement of Objectives Document)

### Goals

Maximize learning

Win RoboCup 1999

Win RoboCup 2000 and beyond

Team Brazil wins CU competition

Make Scientific Contribution

Simple, 3 people set-up for Cornell's "soccer team"

Minimize maintenance

Allow for interchangeability of robots/components

Allow for simulation prototyping

Keep costs below $20,000

### Functions

Ball Control

Soccer Strategy

Communications

Visual Sensing

Adaptable Strategy

Endurance

Maintainability

Model System Functions

### Objectives

Pass, kick, dribble, intercept ball

Kick frequently and very soon after collecting the ball

Deflect ball with high reliability and accuracy

Fast, accurate robot movements

Maintain ball control during acceleration

Fast, accurate velocity feedback control

Provide enough power to finish games (20 minutes)

Robust, real time, accurate visual system for AI system

Noise tolerant, color distinguishing, occlusion handling vision system

Communicate quickly with robots

Robots respond correctly to global commands

Avoid obstacles

Global does most of the processing

Global sensory inputs

To make robots perform as team rather than independent agents.

Intelligent algorithm that works with robots.

Logical reasoning from AI

Effective distribution of information from AI

Accurately model physical attributes

Create team of 4 players and 1 specialized goalie

Goalie – accurate ball control, fast specialized movement

Reliably locate our own team and the ball

Track the location and velocity of the ball with the highest accuracy possible.

Design, build structure that withstands numerous, high velocity impacts

Design, build robust, modular, re-usable players

Structurally rigid and stable robots

Simulate robots, communication system, AI & vision interactions

Modularize designs

Achieve high level of excellence in designing/building the robot soccer team

Ensure high level of "corporate memory" (documentation times $\infty$)

Run an efficient operation

Reliable system

Modular, upgradeable system designs

Easily testable subsystems

Utilize opponent prediction to adapt strategy

Easily transported

Contain readable and easily understood code for current/future CU RoboCup team

# Section 6 Design Requirements

## 6.1 Environment

Field size is 152.5 cm by 274 cm, corners contain 3 cm by 3 cm triangular blocks with green strips of width 1 cm painted on the edges of the diagonal block.

The field will be dark green, with slight differences in color between tables. (It will meet the International Table Tennis Federation Standards, which are fairly loose specifications according to RoboCup)

The field border walls are white, and 10 cm tall. Lines and borders are drawn in white.

A 1cm thick line will be painted across the center of the table, with a circle 25 cm in diameter in the center.

The defense zone will be 22.5 cm from the goal line and 100 cm wide, marked by a 1cm thick white line.

Goals are 50 cm wide, and 18 cm deep. There is no safety net over the goal. The wall and area behind the goal line will be painted either dark blue or dark yellow, depending on which side of the table you are on. Robots may use the area behind the goal line.

The ball will be a standard orange golf ball.

## 6.2 Robot Restrictions

The team shall consist of no more than five robots.

The total floor area of a robot must be no more than 180 square cm, and the maximum diagonal of the body shall be no more than 18 cm. Robot height for teams using global vision is restricted to less than 15 cm, and less than 22.5 cm for teams not using a global vision system. Measurements are to be taken at the robot’s maximum size.

The robots will be examined by the referee before the game to ensure that they fulfill the size requirements.

The goalkeeper may hold and manipulate the ball for up to 10 seconds within the defensive zone.

Two ping-pong balls must be mounted on 1-2 mm diameter spindles of equal height according the rules specified in the vision section below.

The ball may be lifted during play, but lifting the ball must not endanger spectators, referees, or team members.

Another robot should be able to take the ball from another player. In general, 80% of the ball should be outside the concave hull around the robot.

## 6.3 Vision

The use of a global vision system and an external distributed vision system is allowed. The height of the mounting beam for global vision equipment will be at least 2 meters above the table. The placement of the camera is on a game by game basis, and a coin toss is used to determine which team places their camera first.

Each team will be given a certain amount of time to fine tune their robots for the actual field settings on a day before the competition.

Robots will be marked with color ping-pong balls mounted on the top surface so they do not touch. Unless the shape of the robot does not allow it, the ping pong balls must lie along the axis of movement. If they are not capable of being lined up, this must be advertised before the competition. The color can be different from one robot to the next. One of the markers will be specified as either yellow or blue. The other ping-pong ball may be any color provided the team has registered the colors to be used before the start of the competition.

RoboCup officials should be notified of the use of any global vision system at the time of registration, and detailed arrangements shall be discussed with the RoboCup committee.

The local organizer will inform the participants of the camera attachments required.

The lighting for the playing field is currently specified as 700-1,000 LUX.

## 6.4 Wireless Communications

Teams are required to specify the method of wireless communication, power, and frequency at the time of registration (by the first of May). The tournament committee should be notified as soon as possible if there are any changes after registration.

To avoid interference, teams should be able to select two carrier frequencies before the match.

The transmission bandwidth is not limited.

The type of wireless communication must follow legal regulations of Sweden.

## 6.5 Game Competition Rules

### 6.5.1 General Rules:

The game consists of three 10 minute periods, the first half, break, and second half.

Only one robot from each team may enter the defense zone. Accidental entry is permitted, but intentional entry or staying is prohibited.

The goal keeper may hold and manipulate the ball for up to 10 seconds within the defensive zone. After releasing, the goal keeper may not recapture the ball until it is touched by any opponent or team member outside of the defensive zone.

If the goal keeper releases the ball and it reaches the half way line without another robot contacting it, the other team is given an indirect free kick positioned anywhere on the half way line.

If the goal keeper has hold of the ball, the opponent must leave the defense zone.

In the defensive zone, the opposing team's robot cannot intentionally interfere with the movement of the defender/goalkeeper robot.

A robot is said to be within the defense zone if any part of it is within the zone.

The offside rule is not adopted.

If the playing field or ball is modified in any way, the game is stopped and the appropriate restoration is done immediately.

Resolution of dispute and interpretation of ambiguity of rules shall be made by three officials who will act as umpires, designated prior to the match. The umpires may consult with RoboCup tournament officials to aid in resolving conflicts. Ambiguities shall be resolved by referring to FIFA official regulations where appropriate.

Special modifications to rules may be agreed to at the time of the competition, provided a majority of contestants agree.

### 6.5.2 Stoppage of Play:

Substitutions cannot occur during play, unless a robot is damaged. If damaged, the team has no more than 5 minutes to repair or replace the damaged robot.

The ball has to go forward at a kick off or restart.

In general, hand placement of the robots is not allowed during stoppage of play if your team is using a global vision system. If a team does not use global vision, the teams' robots may be hand-positioned at kickoffs and restarts.

The ball may be lifted during play, but lifting the ball must not endanger spectators, referees, or team members.

The referee will call verbally or use a whistle to signal kickoffs, restarts, and stoppages of play. The operator of the team can then send signals to robots. The signal can be entered through a keyboard attached to a server being used on the sideline.

On kickoff, all robots shall be positioned on their side of the field. Only one robot from the side that will kickoff shall be in the center circle. Restarts after a goal shall take the form of a kickoff.

Only the goalkeeper may be in the defensive zone during penalty kicks. All other robots must be located at least 15 cm behind the robot kicking the ball. Only the goalie and the kicker may move until the kicker touches the ball.

Currently all free kicks will take the form of kickoffs. If a team has been fouled, they will be awarded a free kick, otherwise the team that last touched the ball will have the kickoff.

A ball that exits the playing field will be immediately returned and positioned 5 cm from the inside of the wall where the ball left the playing field. A robot from the team that did not push the ball out of play shall kick the ball. No other robots shall be within 15 cm from the ball. Time counting continues, and the robot may continue to move.

If the referee deems that the game has stopped due to a lack of progress, a free kick is awarded to the team that last touched the ball. A game is considered stopped if the ball has not been touched by a robot for 30 seconds, and it appears that no robots are likely to hit the ball.

All players must be halted prior to kick off or restart of the game. The referee checks or adjusts the placement of the players and declares the completion of adjustment 5 seconds before indicating a kick off or restart action. During these 5 seconds, the players cannot move.

### 6.5.3 Fouls:

When more than one robot on the defensive team enters the defense zone and substantially affects the game, a foul will be called and penalty kick declared.

Only goal keepers in the penalty area may hold the ball. The referee will judge whether or not someone is holding the ball. Generally, another robot should be able to remove the ball from another player.  If a player is deemed to be holding the ball, then a free kick will be declared.  If this occurs in the defense zone, a penalty kick will be declared.

Players must not attack each other unless they are trying to fight for the ball.  If the umpire clearly observes an opponent attacking for any other reason, the responsible robot is removed from the playing field (red card).  In general it is unacceptable to hit another robot from behind, multiple robots to charge another robot, or to push another player along the table.  Exact interpretation is left to the referee.

If a player utilizes a device or action whose purpose appears to be to damage other robots, the robot must be removed and the problem must be corrected before the player may return to the game (yellow card).  If the same type of foul is repeated, the player will be ejected (red card).

# Section 7 Conceptual Design

## 7.1 Concepts (Features and Wish List)

### 7.1.1 Vision (Global Vision)

The global vision system used for the RoboCup team will be a color system capable of tracking the positions of the ball, the five opponents, and the five Team Brazil robots. The vision system must be real-time, reliable, robust, give accurate positional information, and cannot be susceptible to ball occlusions and loss of objects during tracking. The vision system must also be able to keep track of the goals scored during the game. The positional data and the classes of each object recognized will be sent to the AI system for updating the current internal state of the game and further processing. For ease and speed of setup, the vision system should be able to automatically calibrate itself upon system setup mapping a specific pixel value into a physical distance measure from the origin of the playing field.

#### *Hardware (Camera, Framegrabber and DSP)*

The global vision system will employ either a single or multiple camera system to capture the mechanical state of the game and send this data to the computer for processing. The visual processing will be done on a dedicated DSP board with framegrabber, which will enable the host processor and bus to be free for artificial intelligence processing, planning, and communication. The DSP board will be optimized for image acquisition and processing and will be able to process a 640x480 image at 60 frames per second.

#### *Software (Tracking Module)*

The tracking algorithm employed by the vision system will be able to differentiate between the following five colors reserved by the Robocup Federation : dark green, orange, yellow, dark blue, white, and also another color(s) used for orientation of the robots. The color histogram backprojection with blob aggregation algorithm, an edge based detection algorithm, and a k-means neural network algorithm with thresholding are currently the algorithms under consideration for Team Brazil. The positional information of the eleven moving objects, the orientation, and the class of each object are sent to the prediction module and the artificial intelligence system. The prediction module will return a predicted location of the eleven objects to the vision system. The vision system will use prediction to reduce the amount of time spent on visual search for any particular object. Prediction will enable the algorithm to localize the search in areas of interest instead of searching the entire image. This type of prediction can be done because of the constraint of spatial proximity of objects through the visual field. This simply states that objects localized in time must also be localized in space

### 7.1.2 Wireless Communication

Fast transmission

A way to give information to each robot

Robust, good error handling

Back-up action for lost packets or communication link

(i.e. stop after time T or continue until hit a wall.)

Low power consumption

Light weight

At least 10 meters range

Easily connected to main computer (RS-232)

Simple MPC connection

Modular and detachable from main circuit for upgrade or replacement

Strategy independent control/ ability to changes the data transmitted

Effective coding strategy to maximize information flow to robots. (i.e. compression, decrease settling time by sending large patches etc.)

### 7.1.3 On-Board Electronics

*Microcontroller (MCU)*

Able to unpack data and interpret data quickly

Enough signal control for local devices

Adequate sensor inputs ports

Integral with motor control

Fast interrupt handling

Enough memory for program and registers

Low power consumption

Easy to add functionality

Easy to load main program into memory from PC

Upgradable

Local program stability

Good coding to handle the process efficiently

*Local sensing*

IR sensors for shaft sensing

Current spike sensing if the motor stops spinning due to malfunction or a object is blocking path (optional)

Battery condition sensing through a A/D port on MPC

Kicker sweet spot sensor (optional)

Goal ball sensor (optional)

*Power conditioning*

Stable +5 volt power for digital circuits

Separate analog and digital power

Rechargeable

Ability to handle power surges

Recharging unit

Good weight/power ratio

Low battery indicator and/or condition of battery

Memory-less batteries or ability to discharge before charging

AC/DC power for transmitter

Turn-on power surge protection

Local memory will not be erased if the batteries are removed

*Motor controller*

Good power output

Accurate and fast response to the MPC

Input control is such that it does not take a lot of MPC time

Change or maintain velocity quickly and accurately

### 7.1.4 Artificial Intelligence

The artificial intelligence system receives tracking information from the vision system consisting of the position and velocity of the ball, the position of opponents and our robots, and the orientation of our robots. The AI predicts what the tracking information will be in one cycle into the future, using the commands given to our robots and Kalman filtering. The AI then feeds the prediction information back into vision to facilitate the search for the objects being tracked. The prediction information is again used in the strategy module, since the predicted locations are a better approximation of where the objects will be by the time the commands issued are carried out.

On the high level, our strategy will be to utilize a state-based design. Each *state* has an associated evaluation function and a strategy-execution module. The *evaluation function* characterizes the game with respect to the current state. Based on the evaluation, a *module* will be run. A module contains the high-level strategy to execute. The modules will be developed through intuition about the game and experience gained from simulation. The modules will call lower level functions to carry out the strategy. Lower-level functions include obstacle avoidance, robot motion planning, and ball handling routines. The evaluation functions will begin as simple heuristics taking into account relevant prediction information, but we will improve them slowly, perhaps incorporating learning, searches, lookup tables, a combination of these strategies, or simply better heuristics selected by simulation.

The four levels of states are:

Long-term global strategy: These will govern team behaviors over longer periods of time in the game. For example, these states will consider the relative scores and the success of recent plays to determine the best general strategy for the team against the current opponent, in terms of aggressiveness or defensiveness, and which formation the 4 field players take.

Short-term global strategy: These strategies operate from play to play, determining such things as current ball possession and risk of scoring from either side. These states are weighted by the long-term strategy, and in turn, they weigh or demand the states individual robots may take on.

Long-term local strategy: These are gradual shifts in the strategy of individual robots, such as detecting failed communication and trying to get them back online, adapting to the robots position in the selected formation, and trying to defend against specific robots.

Short-term local strategy: These are the most specific states, covering tactics such as shooting, passing, intercepting, blocking, and dribbling. These states are weighed and may be superceded based on all the higher level states.

### 7.1.5 Attacker/Defenders

The attackers and defenders for the Brazilian Cornell RoboCup team will be identical. The overall goal for Team Brazil is to have robots that are faster and stronger than the opponents' even if that means they are not as intelligent. To accomplish this, each of Team Brazil's robots will be equipped with two electric motors and a kicking mechanism. The robots will receive instructions from the global computer system and carry them out. The instructions should not be any more complicated than trajectory vectors to follow and kicking commands.

#### *Frame and propulsion*

The basic frame of the robot will consist of a chassis with two wheels mounted in parallel along the same axis. Small casters or pads will mitigate scraping and tipping. The electric motors will drive one wheel apiece, giving the robot the ability to turn by spinning one wheel faster than the other and the ability to move forward and backwards. The robots will be "ambidextrous", with the ability to move forward and backward at the same speed with the same agility. (i.e. 2 control surfaces)

#### *Kicking mechanism*

The kicking mechanism has not been decided at this point. The kicking system should be operable from both the front and back of the robot. At this time the device powering the kicking mechanism is still undetermined. There are several methods of energy storage under consideration: (i) compressed CO2 in a cartridge triggered by a solenoid valve, and (ii) compressed spring mechanism. The advantages and disadvantages of each concept are discussed in the analysis section.

### 7.1.6 Goalie

The goalie will be different from the other robots on the field. Its main purpose will be to stop balls from entering the goal, so it will be optimized for acceleration and motion along one axis. The rules permit the goalie to hold the ball, so the goalie will be fitted with a device that allows it to scoop up the ball. The details of this device have not been worked out at this time, but the goalie will certainly have the ability to grasp balls headed for the corners of the net using its sides, and possibly in the front as well. Holding the ball from the front allows formations to be created on the field (the rules allow the goalie to hold the ball for 10 seconds).

For mobility, the goalie will have two wheels, mounted in a fashion similar to the other robots. The gearing on the goalie will be much lower to accommodate the higher acceleration.

### 7.1.7 Simulation

The purpose of simulation is to provide the Strategy group early access to accurate information with which to begin testing and coding certain basic and/or high level functions. In concurrent engineering, many of the necessary details for AI coding exist in the early stages, but no hardware interfaces are available to test the viability of the codes developed.

*Simulation should provide the following features:*
Accurate movement of robots and ball in a 2D environment

Accurate environmental features (e.g. correct field, ball and goal sizes)

Ability to test various noise conditions

Ability to test various strategies concurrently

Ability to simulate system interaction/performance

Ability to provide quick feedback on possible design considerations

## 7.2 Analysis

### 7.2.1 Wireless Communication

| **Off the shelf vs. "in-house design"** | |
|---|---|
| Off the shelf | "in-house design" |
| • Slower<br>• lower risk<br>• less control with inputs and outputs | • more control over design<br>• more complex<br>• high cost in time<br>• high risk<br>• faster |

| **2 way vs. 1 way** | |
|---|---|
| 2-way | 1 way |
| • more complex<br>• better control<br>• slow MPC time<br>• robot to robot communication<br>• more robust in returning OK signals | • cheaper<br>• faster implementation<br>• fast execution speed<br>• simpler network protocols<br>• faster communication |

| **High speed vs. Low speed** | |
|---|---|
| High speed | Low speed |

| | |
|---|---|
| • more data<br>• higher error rates | • lower power<br>• more delay in transmissions<br>• easier to get and set-up<br>• easier to test operation |

### 7.2.2 On board electronics

*Micro-controller*

MC68HC11 requires 2 cycles at 8MHz so 1 microsecond per instruction and up to 0.5 microseconds per instruction.

| **CPU vs. MPC** | |
|---|---|
| CPU | MPC |
| • Faster<br>• More control ability<br>• more discreet components<br>• more complicated | • more analog connections<br>• more built in functions<br>• lower power<br>• interface to program<br>• built-in clocking and memory |

*Sensors*

Shaft encoders typically are 32, 48, or 64 segments. The more segments the greater the resolution but it is limited by the field of view of the photo-reflector. For a 64 segment the formula would be radius*tan (360/64). Therefore a wheel of 3cm would have an accuracy of 0.29cm.

| **Local ball sensing vs. Global** | |
|---|---|
| Local ball sensing | Global |
| • better "kick" position sensing<br>• more accurate ball handling | • better strategy planing<br>• more accurate ball location |

*Power*

The battery requirements will be clear after some parts are decided but to give some rough figures:

| **Alkaline** | **Li-Oxyhalide** |
|---|---|
| D 1.5V 8000mAh @ 100mA | D 1.5V 10500mAh @ 100mA |

About 0.5-1A will be needed for the digital circuits and based on the mechanical calculations, the robot will need about 7W of power per motor. That is 1A at 7 volts (P=IV).

| **Rechargeable vs. Non-rechargeable** | |
|---|---|
| Rechargeable | Non-rechargeable |
| • lower cost<br>• more options | • more related power output over time<br>• more power |

| **Separate Analog and Digital power conditioning vs. One system** | |
|---|---|
| Separate A, D | One system |
| • more robust<br>• less reactive to power surges | • easier to implement<br>• lighter/ less components<br>• problems with noise |

*<u>Motor controller</u>*

| Pulse modulated vs. Register loaded | |
|---|---|
| Pulse modulated | Register Loaded |
| • the MPC will control the feed back loop<br>• easier to implement for motors | • Less inputs needed for MPC<br>• easier on the MPC processing time |

### 7.2.3 AI

The state-based design was chosen because it is highly flexible and easily expansible. We can add new states or change evaluation functions without having to change anything else in our code, and we are free to implement the evaluation functions and state modules in any ways. For these reasons we felt the state-based paradigm was a good place to start thinking about our high level design.

There are tradeoffs in the potential designs for our evaluation functions. If we use a neural net developed in simulation we will be at risk of developing a strategy that is highly specific to certain styles of play which may or may not appear in actual matches. Separating those aspects of gameplay which are constant and those which are idiosyncratic to our simulation conditions will be difficult. We would have to develop our algorithms quickly and dedicate substantial processor time to simulation for the rest of the year. Lookup tables would do somewhat better in terms of covering more general styles of play, but they will still require substantial simulator time. We would need large amounts of RAM on our strategy computer, and even still, our tables will be quite granular due to all the factors which can come into play. Within this granularity there could be room for radically different optimal strategies. A search approach is difficult because there are so many potential moves. It is a computationally intractable problem that makes chess look like tic-tac-toe. We can never even be entirely sure about the range of motion of our opponents. Because of these significant tradeoffs, we may be forced to implement some form of heuristic, but we can blend it with any or all of these approaches to some extent.

High end personal computers are an order of magnitude cheaper than our total budget, so for standard PCs, performance is the prime factor. Specialized computers with greater performance have a lower performance to cost ratio and may not be needed to satisfy our specifications. The hardware needed for AI, distinct from those from vision and communication, is standard.

### 7.2.4 Players

The two wheeled, kicking design was determined to have distinct advantages over the other designs we considered. Two wheels were chosen because they could allow us to build a tight-turning and agile robot. Using two wheels would also be lightweight. Rather than using a steering motor and a drive motor, both motors would be used to drive the robot.

| | 2 Wheels | 3/4 Wheels | Tank tracks | trackball |
|---|---|---|---|---|
| Steering/ Agility | 2 | 2 | 1 | 4 |
| Speed | 3 | 4 | 1 | 2 |
| traction (shoving) | 2 | 3 | 4 | 1 |
| ability to model | 4 | 3 | 1 | 2 |
| ease of getting feedback | 4 | 3 | 1 | 2 |
| ease of construction | 4 | 2 | 3 | 1 |
| totals: | 19 | 17 | 11 | 12 |

The advantages of the two-wheeled system can be seen in this chart, where each aspect of each system received a relative ranking. Furthermore, the two-wheeled design was proven to work quite well in the Mirosot competition and previous RoboCup competitions.

## *Calculations*

Rough estimations of power required for each motor was performed. The procedure was as follows:

| | |
|---|---|
| Knowns: | |
| Coefficient of friction | gear box efficiency |
| Mass of vehicle | engine speed(rpm) |
| min. top speed | want: |
| min. acceleration | gear ratio |
| wheel radius | torque at motor |
| axle radius | |

First we want to find the gear ratio that would provide the desired top speed. The top speed is directly related to the wheel radius, desired operating engine speed, and gear ratio. From there, we obtain the torque that is required of the motor given the minimum acceleration we want for our robot. This method is useful for both the players and goalie if the required torque and gear box we need are commercially available and meet the electrical power constraints(initial calculations show that this will be the case).

## *Procedure:*

$$GearRatio = \frac{2 * \pi * WheelRadius * EngineSpeed}{60 * RobotVelocity}$$

$$TorquePerMotor = \frac{Acceleration * MassOfRobot * AxleRadius}{2 * GearboxEfficiency * GearRatio}$$

*Sample calculations:*

| | |
|---|---|
| coefficient of static friction | 0.43 |
| Mass of robot (kg) | 2 |
| axle radius (m) | 0.025 |
| Wheel radius(m) | 0.03 |
| minimum acceleration (m/s^2) | 2 |
| minimum force required at wheel(N) | |
| Gear box efficiency | 0.7 |
| max torque available at wheel (Nm) | 4.214 |
| Min. velocity (m/s) | 2 |
| **engine speed available (rpm)** | 4000 |
| | |
| Gear ratio required | 6.283185 |
| | |
| Torque required at motor (Nm) | .011368 |

## ***Kicking***

Kicking mechanisms are still under investigation. The team decided it was preferable to include a kicking mechanism since it would be consistent with the “speed” philosophy we have adopted. The mechanism should be able to provide a high powered kick in one direction. Passing will be accomplished by simple bounces.

Two designs for kicking were discarded as being too complex:

A turnable kicker was thought to be too complicated and also difficult to package in the 180 $cm^2$ size constraint.

A rotating shell was also considered, but dropped when it was considered too difficult to control by the global vision system, and that it could lead to dynamic instability.

Also, solenoid kicking was deemed unsuitable because of the high power required to impart a satisfactory acceleration onto the ball.

Several kicking mechanisms under consideration are summarized below:

| Energy Storage | Advantages | Disadvantages |
|---|---|---|
| <u>Compressed Linear/Torsional Spring</u><br><br>A linear or torsional spring will be used through a system of levers and gears to push a firing pin that will strike the ball. The spring will be compressed by a servomotor. | • Mechanical system can be designed to be robust<br>• Wide selection of linear and torsional springs<br>• Possible to simulate with Working Model | • Requires additional servo motor<br>• Requires latching mechanism that can be triggered electronically |
| <u>Compressed $CO_2$</u><br>Compressed $CO_2$ is stored in a tank, and released to impart force to the ball. The gas will be released in a piston system which will push a striking pin onto the ball. Piston will be designed to impart maximum momentum to the ball. | • Cheap, efficient way of storing energy.<br>• 9oz paintball $CO_2$ tanks are rated at 2000-2500 psi. | • Limited availability of small, high-pressure $CO_2$ tanks.<br>• Designing and fabricating valve-piston system might be difficult |

***Ball-control***

There are several ball-control concepts under consideration. A final decision requires more analysis and hands-on investigation:

| | **Advantages** | **Disadvantages** |
|---|---|---|
| **Roller system**<br>A roller system will impart backspin to the ball, keeping it close to the robot's side and keeping the ball from bouncing off when it contacts the robot. This system would be mounted on the front and rear of the robot. | • Ball keeps in contact with the robot<br>• Increased ability to maneuver with the ball<br>• Rollers might be used in reverse to accelerate the ball for passing | • Complicated system will require a motor and gearing system for the rollers<br>• More than one roller is required for smooth handling of the ball. From rough estimates, three rollers at different vertical heights is ideal. |
| **Curved/triangular surface with retainers**<br>A parabolic surface will automatically keep the ball at the center when the robot is pushing the ball forward. | • Robust, no moving parts.<br>• Ball is kept in the center when the robot accelerates. | • Possible difficulty in manufacturing curved surface to exact specifications.<br>• Curved surface is more mathematically taxing for the AI system to anticipate ball behavior during collision. |
| **Flat surface with retainers**<br>A flat surface will have extrusions at the end to keep the ball from falling away from the sides when the robot turns. | • Robust, no moving parts<br>• Simple to build<br>• Ball collisions are easier to predict. | • Limited flexibility.<br>• Ball is not automatically centered when the robot moves forward |

### 7.2.5 Goalie

Power and calculations for the goalie are the same as for the players.

### 7.2.6 Simulation

The robots themselves must be accurately simulated and then the necessary data passed to the respective functions (i.e. AI). The accurate simulation of the robots consists of providing the correct artificial forces of gravity to account for friction. The data then must be passed through the AI using similar inputs as the competition. In this year's case, the inputs will be only the vision system's camera. In future years, our robots may be able to send information back to the global system. However, this possibility is not being utilized this year in the interest of simplicity and time.

## 7.3 Design Specifications

### 7.3.1 Vision

#### Hardware

The specification for the vision system hardware currently is 640x480x24-bit resolution at 60 frames per second

#### Software

The specification for the software component of the vision system is that the system be able to process images in real-time and reliably. There must also be a prediction module that would reduce the amount of time for visual search and tracking.

### 7.3.2 Wireless Communication

- 40Kbps with 100% over head and 3-5ms setting time
- 5 volt power for the wireless system

### 7.3.3 On-Board Electronics

These are the specifications used in the assumptions for our calculations and do not necessarily reflect the actual design.

MC68HC11 requires 2 cycles at 8MHz so 1microsecond/instruction and up to 20.5microsecond/instruction.

<u>Microcontroller MC68MC16Z1</u>

- modular and easily upgradable
- 16-bit architecture
- watchdog timer, clock monitor, and bus monitor
- PLL clocking system
- two 8-bit Dual function Ports
- One 7-bit Dual Function Port
- fast interrupt response time
- 1MB of program memory and 1MB of data memory
- three 16-bit index registers
- programmable chip-select outputs
- 8 ADC
- two 16-bit free running counters with prescalers
- sensors
- IR motor control position sensors
- 256 divisions for accurate speed measurements

- power of batter into A/D port on MPC
- every X execution test batters
- Two pulse modulated outputs for the motor control

Power

- 7805 power regulator
- up to 1A for digital
- Analog unregulated
- On-off switch
- Caps to control surges into digital and analog

Motor controller

- MPC1710A
  - 1A
  - pulse with modulation
- LM629
  - register controlled

### 7.3.4 AI

- 60 Hz cycle time
- 400 MHz Pentium II or better
- High speed LAN with tracking
- 64M RAM

### 7.3.5 Players

| | |
|---|---|
| Top speed | 2 m/s in one quarter of the length of the field<br><br>This speed was realistic and attainable after watching the Mirosot competition as well as last years Robocup and after consultation with Jin Woo. The robots in Robocup were definitely slower than the robots in Mirosot. If we could attain the speed of the Mirosot robots, we would have a distinct advantage. |
| Turning radius | 0 |
| Kicking mechanism | Able to propel the ball to 2 m/s |
| Acceleration | Top speed in ¼ of the field |

### 7.3.6 Goalie

Top acceleration is more important than top speed. The minimum acceleration that is required for the goalie is 5 $m/s^2$. This number is based on the minimum acceleration required to cover the entire length of the goal, and successfully intercept a ball approaching at 2 m/s to the opposite end of the goal.

# Section 8 Preliminary Design

## 8.1 Analysis

### 8.1.1 Wireless Communication

***RF communication:***

The robots will be required to receive the wheel velocities and kicking information at every 60Hz. There are two main transceivers available that will fit our requirements the FM-RPC-XXX and the FMBiM-XXX-F. The FM-RPC-XXX handles all of the packing and unpacking of the transmission and can send a 1-27byte signal. For a full 27byte (216bits) signal to be sent requires 13.8ms (72.5Hz). Every time you send a packet you pay a 5ms settling time penalty so you want to use the full 27byte packets so you don't pay this penalty often. The FMBiM-XXX-F does not handle the encoding and decoding so the MCU will have to do more work. You could tweak the FMBiM-XXX-F to get a little better throughput than the FM-RPC-XXX but the engineering effort would not be worth it. Especially since the same company will be coming out with a 160Kbit/s transponder in 1999 that can be used next year.

There are a lot of different coding schemes to send the data to the robots. The simplest is sending the velocity wheel 1, velocity wheel 2, and action command. (See table below)

| | Velocity wheel 1 | Velocity wheel 2 | Kicking/action | Total bits |
|---|---|---|---|---|
| Bits | 9 | 9 | 2 | 20 |

The total data send out be 20*5=100bits

We don't need to send the robot number because the FM-RPC-XXX calls an interrupt line on our MCU indicating that it has a complete packet. When the MCU is ready, it will download the full code from the FM-RPC-XXX. This action will take less than 1ms. Once the packet for all the robots are received, robot number one will take the first 20 bits. Robot number 2 will shift left 20 bits and take the new first 20 bits and so on.

To determine an adequate resolution for the velocities of the wheel uses this formula. (See table below)

$$\max velocity \pm 2m/s \Big/ 2^{\#\,of\ bits} = resolution$$

| # of bits | posiblities | resolution cm/s |
|---|---|---|
| 2 | 4 | 100.00000 |
| 3 | 8 | 50.00000 |
| 4 | 16 | 25.00000 |
| 5 | 32 | 12.50000 |
| 6 | 64 | 6.25000 |
| 7 | 128 | 3.12500 |
| 8 | 256 | 1.56250 |
| 9 | 512 | 0.78125 |
| 10 | 1024 | 0.39063 |

We will be using a wire antenna on the robots. This antenna design will give us the best signal and we can make it out of a copper wire.

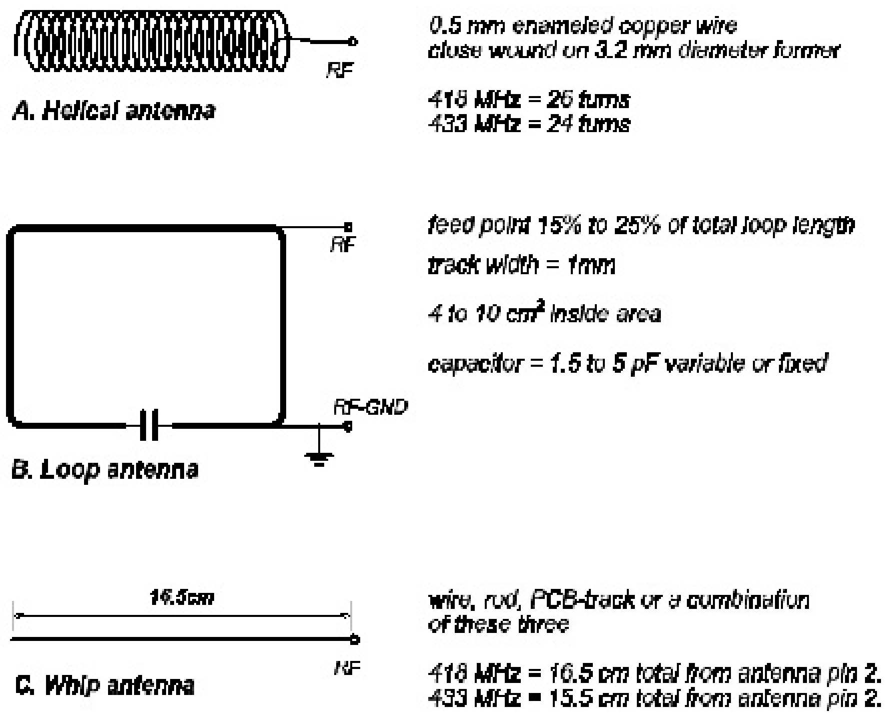

RF
A. Helical antenna
0.5 mm enameled copper wire
close wound on 3.2 mm diameter former
418 MHz = 26 turns
433 MHz = 24 turns
RF
RF-GND
B. Loop antenna
feed point 15% to 25% of total loop length
track width = 1mm
4 to 10 cm² inside area
capacitor = 1.5 to 5 pF variable or fixed
16.5cm
RF
C. Whip antenna
wire, rod, PCB-track or a combination
of these three
418 MHz = 16.5 cm total from antenna pin 2.
433 MHz = 15.5 cm total from antenna pin 2.

**Figure 1.** **Antenna configurations**

The transmitter will be the FM-RPC-XXX evaluation kit. The evaluation kit has an RS-232 connection and the power supply. This would simplify the design of the transmitter.

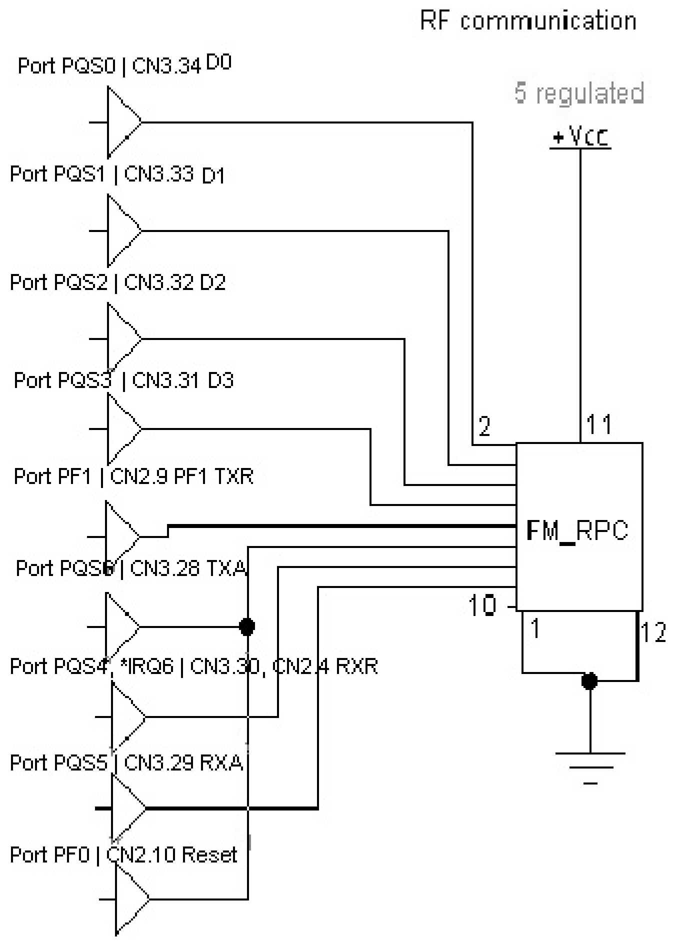


**Figure 2.** **Connection between the transceiver to the MCU**

### 8.1.2 On-Board Electronics

### Electrical robot design:

The electrical design will be split into two separate areas: analog and digital. They will be on separate breadboards connected together by connectors. There are reliability issues with using connectors. This issue is outweighed by the fact that we can change the shape of the electrical layout and can quickly test and replace failed systems. This will also allow next years robots to use sections of the design and upgrade where needed and wanted.

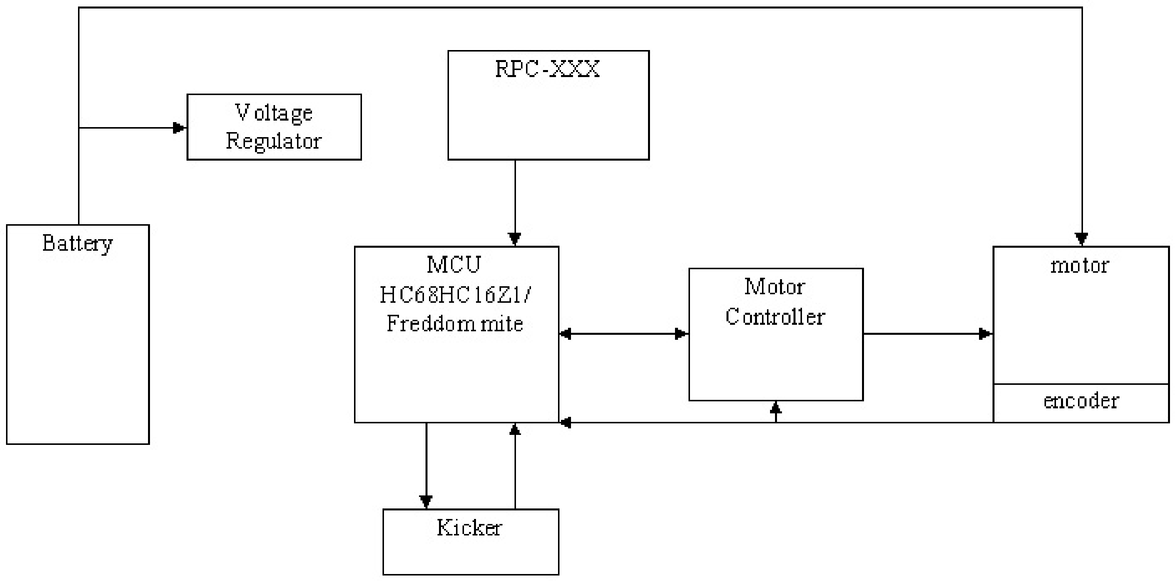


**Figure 3. Overview of electrical design**

We want a simple electrical design for good reliability, low cost, and ease of maintenance. For this reason the design for the robots and the goalie are the same. For the design we will use as many off the shelf and standard parts as possible.

### Micro-controller

*MCU:*

We will be using the MC68HC16 MCU from Motorola. This family of MCU's was chosen because they have a long history in robot design and there are lots of prewritten software algorithms and circuit designs for it. Motorola has been supporting this chip for many years and has upgraded the technology while keeping backward compatibility. This means that the MCU improvements in performance and features will not require a redesign to incorporate into the robots. This MCU features:

- 96K bytes memory + 128K SRAM + 128K EEPROM
- Phase lock looping
- 30 input/output ports
- An 8 channel 8 bit ADC
- Two PWM outputs
- One pulse accumulator register

There are a couple of pre-made design boards in the market. We decided to use the "freedom mite" board because of its small size; the "freedom mite" will greatly reduce the design effort of the chip layout. The pre-made board will handle the entire interface with the computer and include a lot of prewritten C functions to run the MCU.

### Motor Controller:

We have two options for the configuration of the motor controls.

We can have the MCU in the feedback loop and send a pulse width modulation (PWM) signal to a motor controller.

*Advantage*: We will be able to "see" if there are any errors with the execution of our desired velocity. We can also detect and differentiate between wheel 1 and wheel 2 and correct for it. We can write custom software depending on our motor control requirements. This will give us more flexibility in the control of the robots, which is very important in the first attempt at getting the robots to work well.

*Disadvantage*: The MCU will be using more computing power because it only one has accumulator register. The other encoder will have to be connected to a standard input port. This will require an interrupt to be called for every pulse of the motor. Since the MCU handles interrupts very quickly, this should not be a concern.

Use the MCU to "download" (i.e. either through a register or a PWM signal) a velocity to a PID. The PID will handle the feedback loop.

*Advantage*: Frees up the MCU from controlling the motor speed.

*Disadvantage*: We will not have as much control over the feedback loop and the motor control algorithm.

Solution:

The advantage of using the MCU greatly outweighs the disadvantage. It will also reduce the cost and decrease the complexity of the electrical design satisfying our initial desire of simplicity.

The LMD18201 will give us up to 3A and 5.5 volts through an H-bridge. This is more than enough to satisfy our needs. The inputs/outputs to this chip are Direction, Brake, PWM, and thermal warning. The problem with using this chip is that it requires 12 volts for the digital logic to work. Since we will be using a 9.6 voltage power supply this will not work for the design.

The L298 is another motor controller that can deliver up to 4A and the logic will run at 5 volts. This control is a little harder to use because it has less built in features.

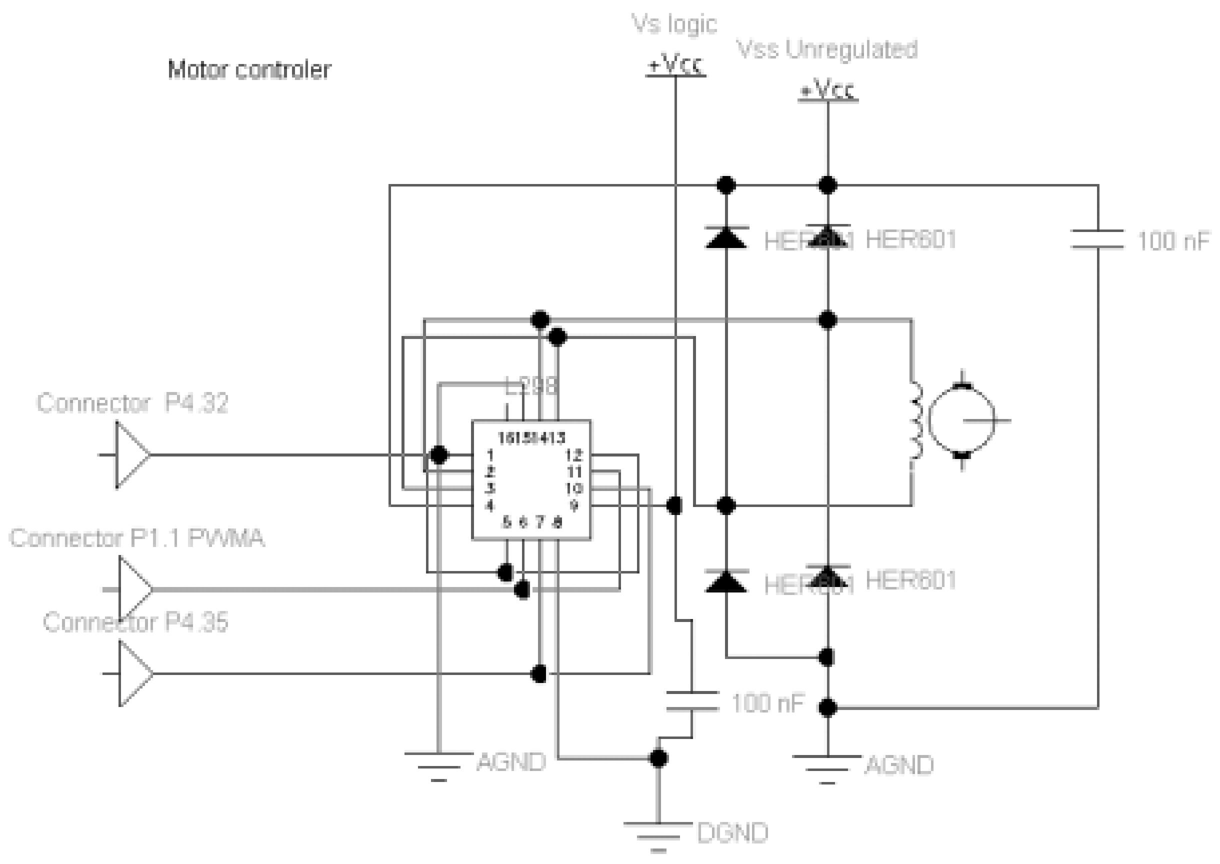


**Figure 4.** **Motor controller**

## Batteries:

The motors will require 6 volts at up to 3.4A and the digital circuits will require 0.45A at 5 volts. Eight batteries at 1.2 volts(each) will give the correct voltage for our design. We will be changing the batteries at halftime so we only need 10 minutes of playtime per battery pack. The Ah's of the batteries need a discharge rate over 5 hours. The total Ah's discharged at a different rate is non-linearly related. For this reason, we have to buy batteries with higher Ah's than what is needed. We will be needing a batteries with a minimum of 0.87Ah's per cell and a total of 8 cells (see

| **Batteries** | | | | |
|---|---|---|---|---|
| time: | minutes | hour | | |
| | **30** | 0.5 | | |
| unregulated power: | mim | max | | |
| | 7.2 | 9.6 | | |
| volts per cell: | mim | max | | |
| | 0.9 | 1.2 | | |
| number of batters: | **8** | | | |
| | | | | |
| **component** | **watts** | **voltage** | **current** | **Ah** |
| motor average | 11.30 | 9.6 | 1.18 | 0.59 |
| kicker | 1.00 | 9.6 | 0.10 | 0.05 |
| electrical | 4.32 | 5 | 0.45 | 0.23 |
| **total** | 16.62 | 9.6 | 1.73 | **0.87** |
| *total I max* | *4.34* | | | |
| *motor MAX* | *24.00* | | | |
| *I mim* | *2.50* | | | |
| *I max* | *3.33* | *<- for motor controler* | | |

chart below)

There are a lot of batteries on the market that will meet our power requirements. Nevertheless, we want rechargeable batteries with a very high current discharge rate to meet the motors' requirements. When we take those factors into account, the selection of batteries is greatly reduced. There are two main batteries type that meet the requirements. They are NiCd and NiMH. When you compare different size batteries in both NiCd and NiMH you will find that the size of 4/5A has the best energy density and energy volume. NiCd and NiMH are very close in energy density, cost, and recharge time. NiCd's have a lower internal resistance and therefore can deliver a high current to the motors. This will be very important to get the desired performance out of the motors. It is even more critical for the goalie since it designed for higher accelerations.

We want to use high performance connectors for the battery packs because we want to make sure that the batteries do not become disconnected during play.  Another reason is that we want to reduce the power loss inherent to poor electrical connections.

### Voltage regulation:

We will only be regulating the voltage for the digital circuits. The motors will be getting unregulated power because we can control motor speed by using our feedback loop. Regulating power consumes energy (P=(Vin-Vout)*I) and should not be used unless absolutely necessary. We have designed all of the digital circuits to run on 5 volts and are expecting our digital circuits to draw no more than 0.45A. When the batteries are at the end of their life they will be delivering 7.2 volts. Most voltage regulators have a voltage drop of 2 volts to maintain regulation. When the motors are drawling full current there will be a drop in battery output level. For this reason we have chosen a voltage regulator with a low voltage drop. The LM2940CT5 regulator has a drop of 0.5 volts, so it will maintain an output of 5 volts when the input is as low as 5.5 volts. We will also place capacitors at the output of the regulator so that if the voltage drops below 5.5 for a given moment, the digital circuits will not be affected. Another positive thing about the LM2940CT5 is that it does not need a heat sink if it operates below 0.5A.

### Battery level sensing:

We will be using an ADC port on the MCU to monitor the voltage of the battery pack.  This will be done by using a voltage divider on the output of the battery to normalize the voltage to the range of the MCU's ADC.  Since the ADC on the MCU has an input range of 0-5 volts and the battery  range will be 7.2-9.6 volts, we want the 10.5 volts to be at 5 volts on the ADC. When choosing R1 and R2 you want them large so that you don't waste energy but small enough so the internal resistance of the MCU is negligible. Good values for R1 and R2 are 5K ohms and 6.67K ohms, respectively (see equation below).

$$V_{ADC} = {R_1}/{R_1 + R_2} \times V_{battery}$$

We will want the resistor to be 1% so we can get an accurate reading on the battery level. We will be using through holes so the resistors will be easy to change.

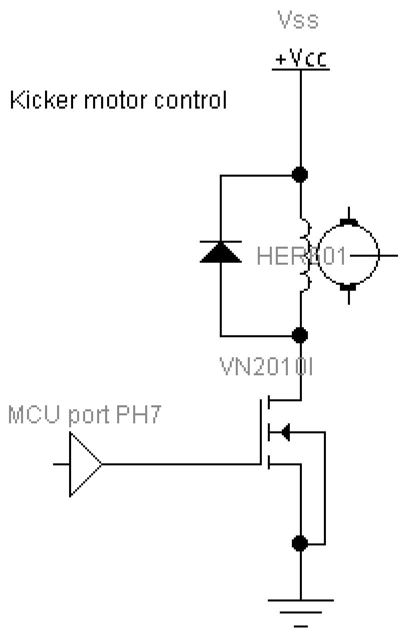


**Figure 5. Kicker Controller**

### Kicker Control:

The kicker will consist of a motor compressing springs. When the springs are fully compressed a button will be hit indicating to the MCU through an interrupt line to stop the motor. When the MCU gets the signal to kick, it will accelerate the wheel motors and turn the kicker back on. This will reset the switch and start the cycle all over. Since it is critical that this process does not go haywire and randomly kick, the software will be monitoring to see that the stop signal comes in on time. If it does not, it will shut down the kicker and turn on an indication light.

### User input and output:

We will be adding four mini size, low current LED's (green, yellow, red, red) to the output of the MCU. They will be connected to output ports on the MCP and have a 2.2Kohm resistor to ground. The reason for having them is to help debug and indicate the status of all the systems. Four LED's will give us a total of 16 possible states of the robot. We will have a look up table to decode the light combinations.

Possible uses:

Low battery indicator

Received a signal for computer

MCP is powered up and the software has booted

Wheels not turning

Kick misbehaving so turned off

In test mode

Indicate robot number in the software 1-5

Etc…

There will also be dip-switch inputs to the MCU. These will allow the user to put the robots into different modes for testing and debugging. The robot number will also be controlled by the dip-switches so that during the game, we can switch any robots without having to download the software to the MCU.

Possible uses:

Normal operation

Test motors and kicker

Run a test pattern

Robot number 1-5

### 8.1.3 Robots

### Goalie

#### Drivetrain

The same calculations as for the players were used to determine our goalie. The calculations for gearhead selection were the same as for the players. A custom gear pair was selected for minimizing drive train space, interchangeability, and cost reduction. Wheel selection for the goalie was done the same way as for the players. The calculations are as follows:

| **ME Characteristics** | |
|---|---|
| Maker | **Maxon** |
| Series/model | 2332-960-12-216-200 |
| | |
| Desired top speed (m/s) | 2 |
| Distance to achieve top speed (m) | 0.68 |
| Friction coefficient | 0.6 |
| Stall torque (oz.in) | 12.616 |
| no load speed(rpm) | 5810 |
| min. acceleration (m/s^2) | 5 |
| Time to reach min velocity (s) | 0.4 |
| Mass of robot (kg) | 2 |
| Wheel radius (m) | 0.03 |
| Max frictional force (N) | 11.76 |
| Force required at wheels (N) | 10 |
| Torque required per axle (Nm) | 0.15 |
| Torque required at axle (oz.in) | 21.24947 |
| 30% of stall torque (oz.in) | 3.7848 |
| Predicted gear head efficiency | 0.7 |
| min. gear ratio required | 8.020605 |

| | |
|---|---|
| 90% of no load speed(rpm) | 5229 |
| Gear head available | 8 |
| Custom gear reduction | 1 |
| max velocity obtained (m/s) | 2.053423 |
| Actual gearhead efficiency | 0.7 |
| Actual ouput torque (oz.in) | 21.19488 |
| Actual acceleration (m/s^2) | 4.989628 |
| | |
| **EE Characteristics** | |
| Torque constant (mNm/A) | 9.43 |
| Counter EMF constant (mV/rpm) | |
| Armature resistance (ohms) | |
| Line voltage (V) | 6 |
| Max. Armature current (A) | 2.834586 |
| Power required (W) | 17.00752 |
| Torque available | |

## Player (Attacker/Defender)

### Drivetrain

The method for selecting motor components is as follows:

Given desired top speed, minimum acceleration and physical assumptions of the table surface and robotic players stated above a motor was chosen. The desired operating range of the motor was selected as 30% of the stall torque and no more than 90% of its no-load speed. This range was recommended by the manufacturers as giving optimum motor efficiency, lowest electrical noise, acceptable temperature range, and maximum motor life. The motor was chosen such that the desired initial power requirements were within this range. This was a trial and error process until the manufacturer's specifications matched our requirements. The selected motor is shown in the following pages with the corresponding calculation sheet:

$$MinimumAcceleration = \frac{DesiredTopSpeed}{2 * Dis\tan ceForTopSpeed}$$

$$= \frac{V_{\max}}{2 * \Delta X}$$

$$MaximumFrictionalForce = FrictionCoeficient * MassOfRobot * GravitationalAcceleration$$

$$= \mu * m_{robot} * g$$

$$Force\,\mathrm{Re}\,quiredAtWheels = MinimumAcceleration * MassOfRobot$$

$$= a_{\min} * m_{robot}$$

$$Torque\,\mathrm{Re}\,quiredAtAxle = \frac{Force\,\mathrm{Re}\,quiredAtWheels * WheelRadius}{2}$$

$$= \frac{F_{wheel} * R_{wheel}}{2}$$

$$GearRatio\,\mathrm{Re}\,quirement = \frac{Torque\,\mathrm{Re}\,quiredAtAxle}{30\% StallTorque * GearEfficiency}$$

$$= \frac{\mathrm{T}_{axle}}{0.30 * \mathrm{T}_{stall} * \eta_{gear}}$$

$$MaximumArmatureCurrent = \frac{30\% StallTorque}{TorqueCons\,\tan t}$$

$$= \frac{0.30 * \mathrm{T}_{stall}}{K_{torque}}$$

| **ME Characteristics** | |
|---|---|
| Maker | **Maxon** |
| series/model | 2332-960-12-216-200 |
| | |
| Desired top speed (m/s) | 2 |
| Distance to achieve top speed (m) | 0.68 |
| Friction coefficient | 0.6 |
| stall torque (oz.in) | 12.616 |
| no load speed(rpm) | 5810 |
| min. acceleration (m/s^2) | 2.941176 |
| time to reach min velocity (s) | 0.68 |
| mass of robot (kg) | 2 |
| wheel radius (m) | 0.03 |
| max frictional force (N) | 11.76 |
| force required at wheels (N) | 5.882353 |
| torque required per axle (Nm) | 0.088235 |
| torque required at axle (oz.in) | 12.49969 |

| | |
|---|---|
| 30% of stall torque (oz.in) | 3.7848 |
| predicted gear head efficiency | 0.7 |
| min. gear ratio required | 4.718003 |
| 90% of no load speed(rpm) | 5229 |
| gear head available | 5 |
| custom gear reduction | 1 |
| max velocity obtained (m/s) | 3.285478 |
| actual gearhead efficiency | 0.8 |
| actual ouput torque (oz.in) | 15.1392 |
| actual acceleration (m/s^2) | 3.56402 |
| | |
| **EE Characteristics** | |
| Torque constant (mNm/A) | 9.43 |
| Counter EMF constant (mV/rpm) | |
| Armature resistance (ohms) | |
| Line voltage (V) | 6 |
| Max. Armature current (A) | 2.834586 |
| Power required (W) | 17.00752 |
| Torque available | |

The method for selecting gearhead components is as follows:

With our selected motor, a desired gearhead that would meet our speed and torque requirements had to be selected. The process started out by selecting components made by the motor manufacturer since they would work well with the motors. Our requirements called for gearhead ratios that were difficult to match with the available ones. The next best match could be selected for this purpose but this would mean that our actual velocity and acceleration would change. In an effort to get a better match, a decision to custom build a gear reduction was made. The advantages to a custom gearbox are smaller size, cheaper overall drive train, smaller weight, and more interchangeability over an off-the-shelf gearbox. This decision also gave us a lot more flexibility in the structural design of the robot, especially for the kicking mechanisms (see kicking section).

The method for selecting wheel components is as follows:

The wheel radius was initially set to a reasonable size based on previous designs so that initial power calculations could be performed. The size was then refined to optimize motor/gearhead combinations. The optimum motor/gearhead combination is the smallest motor with the lowest gear ratio for highest drive train power efficiency. Two available options are available for the wheels: custom machined or off-the-shelf. A custom-machined part would require extra engineering and manufacturing time, which added together may cost more than off-the-shelf components. Off-the-shelf parts may come in the form of radio controlled car (R/C) racing wheels. These wheels are designed for light weight, high strength, and low moment of inertia. The cost is relatively cheap, approximately five to ten dollars per pair. These wheels come in a variety of materials, shapes, and sizes that will fit our needs. At this point R/C wheels will be used. Various tire materials are available for the same R/C wheels. The materials will be chosen after extensive experimentation with the actual ping-pong table.

### Kicking Mechanism

The factor that drives the calculations for this design is the speed we wish to obtain from the ball. It is expected that the robot will be accelerating as it kicks, so the calculations have been made using the assumption that the ball will be pushed rather than struck.

This analysis uses energy conservation. The energy must go into propelling the ball and the kicking plates. The energy of the ball is:

$$KE = \frac{1}{2}mv^2 + \frac{1}{2}I\omega^2$$

The energy of the kicker plate is also ½ I w $^2$, since it is a hinged component. At the end of its travel, the bottom edge of the kicking plate must have the same velocity of the ball, 2 m/s

Therefore, the total energy to be provided on each side is:

$$E = \frac{1}{2}(.04593Kg)(2m/s)^2 + \frac{1}{2}\left[{}^{2}\!/_{5}(.04593Kg)\left({}^{.04267}\!/_{2}\right)^2\right]\left({}^{2}\!/_{.0213}\right)^2 + \frac{1}{2}(1.169x10^{-4})(19.93^2) = .153J$$

In this case, we have taken the mass of the ball as .04539 Kg, the diameter of the ball as .04265 meters, and modeled the kicker as a plate with density 1 g / cubic centimeter. The moment if inertia of a sphere is (2/5)mr$^2$, and the moment of inertia of the plate when hinged at the end is mh$^2$/3. [m – mass of object, r – radius of sphere, h – height of plate]

This energy must be stored as the spring is compressed. Using the dimensions of the robot frame and the RoboCup rules as a guide, the allowable kicker extension is .01689 m per side. Using the energy contained in a spring as ½ k x $^2$, the spring constant of the spring used must be 1073 N/m. We intend to use two springs per side, making the spring constant per spring equal to 536 N/m.

This design is intended to retract the kicking plates by wrapping the fishing line over an angle of 150 degrees. Therefore, the radius of the spool must be:

$$x = r\theta$$

$$.01689 = r * 150 * \frac{\pi}{180}$$

or r = .00645 m, or 6.45 mm.

The compressed springs exert a torque on the kicker, which is exerted on the worm gear and all the parts leading to the motor. The torque exerted is

$$T = F * r$$
$$= 4 * Kx * r$$
$$= 4 * 536 * .01689 * .00645$$
$$= .2335 N - m$$

In the ideal case, this torque is reduced by the worm gearing, which has a 120:1 reduction. Thus the torque transferred to the motor that is needed to directly turn the spool is

Torque to spool = .2335/120 = .001946 N-m.

Unfortunately, real worm gears have a lot of friction that must be accounted for. According to Shigley and Mishke, *Mechanical Engineering Design, 5th ed*. the most conservative coefficient of friction for worm gears is .1. The lead angle of these worm gears is only 1.8 degrees, so the friction encountered by the worm, and therefore the motor is:

$$Torque = Contactforce * \mu * pitchradiusofworm$$

The contact force is the torque from the T-bar / radius of the worm gear, so the equation comes to:

$$Contactforce = \frac{.2335N-m}{.0238m} = 9.81N$$

$$Torque = .1*9.81N*.00635m = .006229N-m$$

However, there are other sources of friction in the system. The worm must ride on a thrust bearing, which experiences the same contact force as the worm. If the thrust bearing is assumed to have a coefficient of friction of .05, and rub against the worm at a radius .00635 m as well, the torque required to overcome this friction is:

$$\mathrm{Torque} = 9.81*.05*.00635 = .003115\ \mathrm{N-m}$$

Summing all these torques, we find the motor must provide .01129 N-m of torque.

Once the motor and springs are specified, the weakest component in the kicking system is the worm gear. The forces exerted on the worm gear are high, and the teeth are fairly small, since this worm gear is 64 pitch. Using Shigley and Mishke’s formulae for bronze worm gear stress, we find:

$$Tg = \frac{\vartheta B d_g^2 F_g \cos(\psi)}{1.5N}$$

The allowable bending stress for bronze (sigma) is 48.2 Mpa. d is the pitch diameter of the gear, .047625 m, F is the face width of the gear, .00476 m, and Ψ is the lead angle of the worm, 1.78 degrees. N is the number of teeth on the gear.

With this data used in this formula, the allowable torque on the gear is 2.89 N-m, which is far above our expected torque of .2335. The gears are by far the weakest component in this system. The aluminum braces are theoretically subject to zero stress because of the symmetry of the system. The T-bar will be made of .25 inch steel bar stock, which, in torsion has:

$$Shearstress = \frac{Tc}{J} = \frac{.2335N-m*.003175m}{.5\pi.003175^4} = 4.6MPa$$

Steel has a yield stress of approximately 200 *Mega* Pascals, which means this T-bar is definitely strong enough. The arms of the T-bar can be analyzed for bending stress

$$stress = \frac{MY}{I} = \frac{.2335\ N-m*.003175\ m}{.25\pi*.003175^4} = 9.2Mpa$$

this is well below the normal stress limit of approximately 400 Mpa for steel. All other parts of the kicker are subject to less stress, and should have even better safety factors.

### Available Parts & Performance

Most of the parts for the kicker will be fabricated by hand in the machine shop. Some parts have been specified in catalogs for purchase. The Maxon GM20 gear motor is ideal for this application. The motor contains a 55.1:1 gear reduction inside its casing that creates very high torque for its small size. This motor is specified for operation at 30 mNm continuously, with a stall torque of 117 mNm. As shown by the equations above, the torque required of the motor will be 11.3 mNm, which is below the maximum torque rating of the motor, and is well below the stall torque. Unfortunately, the high gear reduction makes this motor slower than others, but a lengthy warm up time is acceptable. Other motors from Maxon and MicroMo were investigated, but none could offer the low power consumption, small size, and high torque of this Maxon motor. In addition to these great credentials, the Maxon G20 is one of the cheapest motors

The Maxon G20 runs at 6 volts, so it is compatible with the electrical system we wish to use. It has a torque constant of 169 mNm/A. Since we expect to reach a maximum torque of 11.3 mNm, the peak current draw will be .06 A, or an energy draw of 0.3 Watts at maximum current draw. Since the kicker will not be in constant operation, its energy requirement over the course of a match will depend on usage.

The helical spiral will be made of teflon impregnated delrin composite. This material was chosen because it is lightweight, strong, easy to machine, and low friction.

The worm gears were chosen based on their small size and high gear reduction. They are 64 pitch, 120 tooth, single thread worm gears with matching worms from SDP-SI.

The T-bar will be made of steel. Even though aluminum would be strong enough, steel is required since it will be easy to weld. The T-bar will be fabricated as two parts that will be welded together. It is too difficult to make two round aluminum bars mate at a 90 degree angle.

The motor mount and braces will be machined from aluminum because of its lightweight and easy machinability.

The springs have been selected from the McMaster catalog. They are precision metric springs, with a spring constant of 510 N/m. This is slightly lower than the actual spring constant required (536 N/m), but should be adequate, and is the safe choice especially when one considers that the motor is already going to operate above its designed torque limit.

Using the parts specified above, the energy stored in the springs will be .14549 J, this energy will be used to accelerate the plate and the ball. Solving for the ball speed, using the equation:

$$.14549 = \frac{1}{2}mv^2 + \frac{1}{2}Iw^2 + \frac{1}{2}Iw^2$$

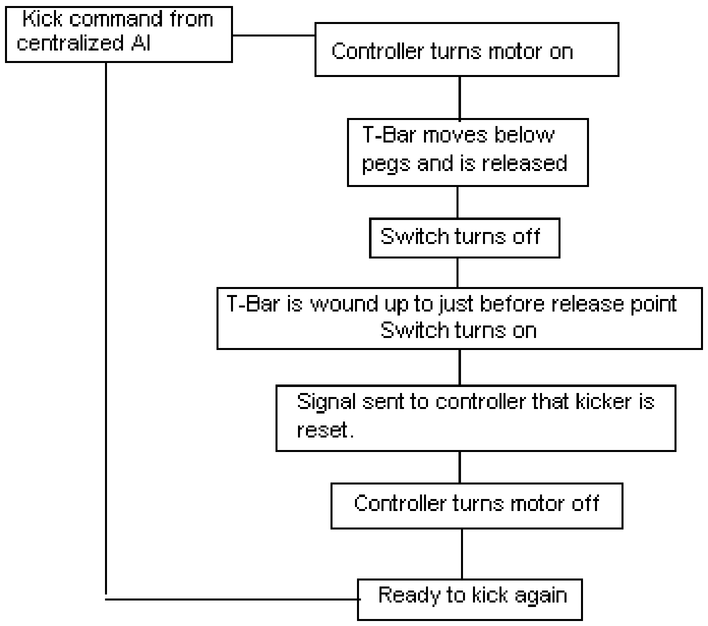
$$.14549 =$$

$$.5 * .04593\ v^2 + .5(8.363e-6)(v/.0213)^2 + .5(1.169e-4)(v/.10033)^2$$

we find the kicker will be capable of kicking at v= 1.957 m/s.

The kicking force is imparted to the ball through two hinged plates, or flaps, on the front and back of the robot. These plates are hinged on top, near the electronic boards on the robot, and compress against springs on the lower chassis. Each is pulled inward by fishing line that is wrapped around a spool located in the center of the robot, above the batteries. This spool is located at the bottom of a steel T-shaped bar. The arms at the top of the T-shaped bar ride two helical grooves machined into a cylindrical piece of delrin-teflon composite. In this arrangement, as the T-bar, and thus the spool, swivel, they move up and down. A worm gear is mounted axially above the T-bar/ helical pivot system. Two pegs protrude from the worm gear that can engage and disengage with the arms of the T. As the worm gear is turned, the pegs will engage the T-bar and will be able to make it turn in its helical slot over a certain range. Eventually, the T-bar will be guided downwards in the helical slot far enough so the pegs on the worm gear can no longer touch it. At this point, the T-bar will be released, allowing the kicking plates to spring outwards. The worm that drives the worm gear will be mounted to a Maxon G20 Gear Motor, which adds further gear reduction. The worm will be supported by thrust bearings to prevent the high axial stresses from ruining the motor bearings.

This design interfaces well with the robot as can be seen in the following state diagram:


Kick command from centralized AI
Controller turns motor on
T-Bar moves below pegs and is released
Switch turns off
T-Bar is wound up to just before release point Switch turns on
Signal sent to controller that kicker is reset.
Controller turns motor off
Ready to kick again

The mechanical parts of the kicker are "smart" enough to reset themselves after a kick without relying on the microcontroller. The microcontroller simply starts the motor when told to kick by the centralized AI computer. The kicker is released, but the motor is not shut off. The pegs on the worm gear engage with the released T-bar and begin to draw in the kicking plates. When the springs are fully compressed a signal indicates to the microcontroller that the kicker has reset and is ready to be activated again. The microcontroller shuts off the motor until it is time to kick. The kicker does not unwind because of the low worm lead angle (approximately 2 degrees) prevents back driving.

## 8.2 Specifications

### 8.2.1 Wireless Communication

Fast transmission

A way to give information to each robot

Robust, good error handling

Back-up action for lost packets or communication link (i.e. stop after time T or continue until hit a wall.)

Low power consumption

Light weight

At least 10 meters range

Easily connected to main computer (RS-232)

Simple MPC connection

Modular and detachable from main circuit for upgrade or replacement

Strategy independent control/ ability to changes the data transmitted

Effective coding strategy to maximize information flow to robots. (i.e. compression, decrease settling time by sending large patches etc.

40Kbps with 100% over head and 3-5ms setting time

5 volt power for the wireless system

### 8.2.2 On-Board Electronics

#### Micro-controller

*MCU:*

| MCU part no. | Voltage | Input/output por | IRQ | A/D Ports | Price |
|---|---|---|---|---|---|
| Freedom Lite | 5V | 30 | 7 | 16(MCX) | $129.00 |

*Motor Controller:*

LM629N-8 (PID controller)

Programmable derivative sampling interval

8-bit sign-magnitude PWM output data (LM629)

Velocity, target position, and filter parameters may be changed during motion

Position and velocity modes of operation

LMD18201 3A, 55V H-Bridge

Delivers up to 3A continuous output

Operates at supply voltages up to 55V

Thermal warning flag output at 145C

TTL and CMOS compatible inputs

*Batteries:*

| Batteries | Type | voltage | power (Ah) | weight (g) | Whr/kg | Diameter (mm) | height (mm) | cost |
|---|---|---|---|---|---|---|---|---|
| Ni-Cd | AA | 1.20 | 0.50 | 21 | 38 | 14.50 | 48.30 | $3.00 |
| Ni-Cd High | AA | 1.20 | 1.00 | 23 |  | 14.50 | 48.30 | $4.00 |
|  |  |  |  |  |  |  |  |  |
| NiMH | AA | 1.30 | 1.10 | 25 | 57 | 14.50 | 48.30 | $4.00 |
|  |  |  |  |  |  |  |  |  |
| Lithium | AA | 3.00 | 2.10 |  | 300 | 14.50 | 50.80 | $8.00 |
| Lithium | C | 3.00 | 5.00 | 42 | 300 | 25.30 | 49.30 | $15.00 |

*Voltage regulators:*

| Part | Max Current (A) | Vin min | Vin max | Price |
|---|---|---|---|---|
| MC78T05ACT | 3.00 | 7.30 | 35.00 | $0.70 |
| LM7805CK | 1.00 | 7.00 | 35.00 | $0.70 |
| LM2904T-5.0 | 1.00 | 5.50 | 26.00 | $4.00 |

### 8.2.3 AI/Strategy

Artificial Intelligence can be subdivided into several layers, each building upon the layer immediately below it and providing services to the layer immediately above it.

### Interface with communication

Fine-tuning of the format of the protocol as well as exploration of other protocols is an activity that we expect to continue through the prototyping phase. This layered approach allows us to keep things modular. We change protocol implementations and formats and keep the interface to upper layer unchanged. For the sake of modularity, the communication layer provides sendv1(value), sendv2(value), kick/action and robot number services to the upper layer ONLY.

### The path planning layer

The next layer is the path-planning layer. Robot motion planning is a complicated task. Two approaches that we have considered for highly dynamic environments are field vectors and Bezier curves.

Field vectors have been used for path planning in robotic soccer. The particular implementation that we are considering is to start off with a current location and a target. All obstacles are then considered attractors. Field vectors emanate from the target. The current location itself is an attractor. Now we simply reverse the direction of all the field vectors. The current location of the robot always intersects some field vector and it follows the field vector until it reaches the destination. In the basic case, we would reach the destination in one time interval, but in the real world the field would change over time and the robots would modify their motion accordingly.

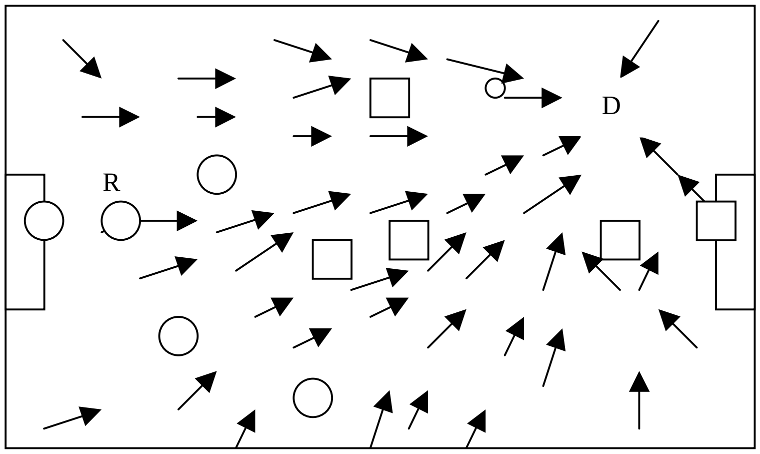


**Figure A**: Field vectors for Robot R and destination D

Bezier curves are a simpler solution. Here we simply draw a straight 'band' between the target and the source, and intervening objects force the band to be 'curved' in both directions by assigning control points further and further from the straight path. When a band exists where there is no obstacle, the robot moves along that band until further instructions. However, Bezier curves' robustness in highly dynamic environments needs to be studied on the simulator.

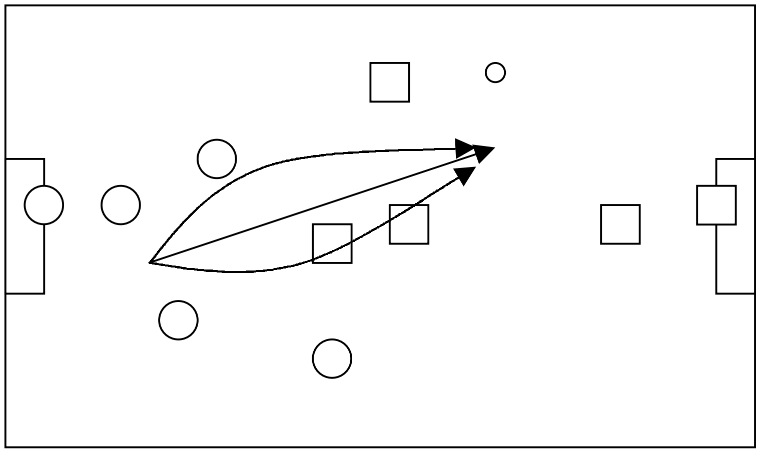


**Figure B**: Path planning using Bezier curves. The straight-line path is obstructed. Hence we try to introduce curves that have two twists, the first evades the points of obstruction, and the second reestablishes the desired final orientation of the path. The lower curve is the correct solution here.

Higher layers may wish the robot to be at a certain orientation on reaching the target. With both the field and Bezier curve approaches this is easily accomplished by ensuring that the field vectors point in the target's direction, or the end of the band is oriented a certain way.

Both approaches even allow a robot to approach other robots closely for 'marking' purposes as well. The limit of their proximity would be determined by the resolution of the field grid in the first approach and the band size in the second approach. We shall be exploring these and other approaches on the simulator.

Collaboration with the mechanical and electrical subgroups is underway to write modules that will determine the correct difference to apply to the wheel velocities to achieve the turning effect. This calculation can only be done once the path is determined and the system knows what the radius of the turn should be.

The path planning layer provides moveTo(location Y, Velocity v) functionality to its upper layer and sends the appropriate wheel velocities to communication channel.

### Low level behavior

The layer above path planning is low level behavior or 'local state'. Like path planning, this layer is local to each robot's software module. The layer can assume that the robot will find the most efficient path to the target. This layer determines the target and the velocity the robot needs to reach it on time. The field is assumed, to have 'hot spots' or regions where the robot can place itself. The ball and the goal area can be included.

The current state of the robot determines its next move. This layer is told the states of the other robots. It then deduces which of the hot spots the other robots are considering and, hence which hot spot the current robot should go to. A hotspot that is easily reachable might not be considered because, for example, it may be too far away from the goal in a dangerous play, or because another robot is closer to it. It may, on the other hand be considered if other robots are in a better position to defend the dangerous play.

This judgement is critical. It will require tests on the simulator and the actual teams.

This layer will accept a hotspot list. This is similar to real soccer where a player doesn't really know the state of another player, but assumes that if the other player is in a reasonably critical location, it will do whatever it can to respond to it.

### Global Strategy

The highest layer is the global strategy layer. This layer manages the running of the five separate modules that correspond to the five robots and keeps track of the location of the ball and the opponents. It also keeps a list of formations and decides on the best formation in any given situation. It assigns hot spots based upon its knowledge of formations and the global state of the game. It receives data from vision and the prediction module to perform these tasks. It also keeps track of the opponent's robots.

A good example of the use of a hotspot is when a team wishes to play a 'man-to-man' marking game. A robot can simply be told to position itself near an opponent's robot and obstruct its shot at the goal. This can be done by assigning a hotspot for each robot that corresponds to a location near every opponent. If no opponent robot can position itself such that it can propel the ball in the direction of our goal, the defense is successful! This strategy is easy to implement with our approach.

The highest two layers are also the most flexible. They can involve aspects from various branches of AI. They can, therefore be hybrids of various learning techniques. Role assignment, however, is an important component of our AI, because it allows a large number of potential scenarios to be handled by a relatively simple behavioral model.

### Tracking Interface

Because of the scale and complexity of tracking, it belongs to a subsystem of its own. It falls into a category between Vision and AI. We expect to use some form of Kalman filtering because it is a robust prediction technique. Since inputs to and outputs from this module are clearly defined. The module need only predict until the next time slot i.e. 1/60th of a second for a 60 Hz update. It provides the predicted locations of all the objects on the field to both Vision and AI. AI needs both the current and predicted paths in order to refine it's estimate of where the system is 'tending' and to assign states accordingly.

### 8.2.4 Robots

### Player (Attacker/Defender)

***Drivetrain***

| | Manufacturer | Maxon 2332-960 |
|---|---|---|
| | acceleration (m/s^2) | 3.56 |
| | speed (m/s) | 3.26 |
| **MOTOR** | Torque (oz. in.) | 3.79 |
| | Current (A) | 2.83 |
| | Power (W) | 17.01 |
| **GEARHEAD** | | 5:1 |
| **WHEEL** | radius (cm) | 3 |
| NOTE: options are custom or off the shelf R/C racing wheels | | |

Procedure:

$$GearRatio = \frac{2 * \pi * WheelRadius * EngineSpeed}{60 * RobotVelocity}$$

$$TorquePerMotor = \frac{Acceleration * MassOfRobot * AxleRadius}{2 * GearboxEfficiency * GearRatio}$$

Sample calculations:

| | |
|---|---|
| Coefficient of static friction | 0.43 |
| Mass of robot (kg) | 2 |
| axle radius (m) | 0.025 |
| Wheel radius(m) | 0.03 |
| Minimum acceleration (m/s^2) | 2 |
| Minimum force required at wheel(N) | |
| Gear box efficiency | 0.7 |
| Max torque available at wheel (Nm) | 4.214 |
| Min. velocity (m/s) | 2 |
| **Engine speed available (rpm)** | 4000 |
| | |
| Gear ratio required | 6.283185 |
| | |
| Torque required at motor (Nm) | .011368 |

Having the basic constraints for the motion of the robots we set up a spreadsheet to help us determine the motors that met our requirements. The design started from the assumptions stated above combined with the assumed mass of the robot, friction coefficient, wheel radius, and motor. With these assumptions a required gear ratio was obtained. An available gearhead was selected to meet the required gear ratio. The actual acceleration, output torque, and maximum velocity obtained were rechecked to meet the required specifications. From the characteristics of the chosen motor, max current draw and continuous power were calculated. This data was given to the electrical design team.

The robot design started out with several features that were thought to be very helpful in the effort to have a very competitive robot team. These basic features were discussed among all the subdivisions in the team to see how the components would interact with each other. After agreeing on the desired features for the robots, some target numbers such as maximum speed, acceleration, kick speed, were decided upon. The speed of the robot was decided with input from Dr. Jin-Woo Lee as to what was desirable based on his experience in the Mirosot competition. The basic equations of motion were used to determine the required acceleration to obtain the desired maximum speed in the specified distance (1/4 of the field).

The procedure was as follows:

<u>Knowns:</u>

| | |
|---|---|
| Coefficient of friction | Gear box efficiency |
| Mass of vehicle | Engine speed(rpm) |
| Min. top speed | Want: |
| min. acceleration | Gear ratio |
| wheel radius | Torque at motor |
| axle radius | |

First we want to find the gear ratio that would provide the desired top speed. The top speed is directly related to the wheel radius, desired operating engine speed, and gear ratio. From there, we can obtain the torque that is required of the motor given the minimum acceleration we want for our robot.

### Kicker

There were several goals we wished to accomplish in designing the kicking mechanism for the robotic field players. From viewing the videotapes of past RoboCup competitions, it was decided that the ability to kick the ball at 2 meters per second relative to the robot's velocity would give Cornell's team a significant advantage over its opponents. In light of the fact that the robots will not be able to trap the ball very well, it was decided a powerful kicking mechanism should only be used for shots on goal. For simplicity, the robots will only be able to kick along one axis, forward and backward. With that in mind, our attention turned to making a kicker that would work well with the other components in the robot. This was to be accomplished in several ways:

The kicker should be energy efficient. Since the kicker will be used at times when the robot is likely to be performing strenuous maneuvers, it could not sap a lot of power from the batteries when used.

The kicker should be simple mechanically. The kicker should not be made of many fragile parts, and it should be lightweight.

The kicker must be "intelligent." It would be preferable if the kicker did not rely heavily on the on-board microcontroller for operation. A clockwork design that could maintain itself mechanically would be preferable, though it could interfere with objective number 2.

The kicker must be able to fit in the robot, so small size and tolerance to misalignment is essential.

The kicker should be ambidextrous. We desired a kicker that could kick from the front and back of the robot. Ideally, the kicker should be able to use most of the same hardware to kick in both directions.

A spring loaded kicking mechanism was chosen for our design. This design has outstanding energy efficiency because it can wind up during the rest periods a robot will encounter between kicks. Our design is also "intelligent", in that it only requires one switch to describe its state to the microcontroller. This design is also fairly small, using worm gears and a tiny gearmotor; more importantly, its linkage system makes it tolerant of misalignments, so it can be packaged easily in the robot.

## Goalie

### *Drivetrain*

|  | Manufacturer | Maxon 2332-960 |
|---|---|---|
|  | Acceleration (m/s^2) | 4.99 |
|  | speed (m/s) | 2.05 |
| **MOTOR** | Torque (oz. in.) | 3.78 |
|  | Current (A) | 2.83 |
|  | Power (W) | 17.01 |
| **GEARHEAD** |  | 8:1 |
| **WHEEL** | Radius (cm) | 3 |

NOTE: options are custom or off the shelf R/C racing wheels

The design analysis for the goalie was similar to the players except for a few key areas. Acceleration had priority over top speed. This was due to the specialized tasks that the goalie has to perform, mainly going from one side to the other in order to block the ball. A bi-wheeled arrangement was chosen for its high maneuverability characteristics. The wheel arrangement will be parallel to the goal for optimal back and forth motion. The same basic assumptions such as mass of the robot, friction coefficient, wheel radius, and motor along with acceleration requirements were chosen to obtain initial design specifications. Calculations performed for selecting motors were similar to the players.

It was decided that the mechanical engineering group would construct a very safe goalie, one that would not have kicking or sophisticated trapping abilities. The reasons behind that decision are listed below.

The R&D group will design an advanced technology, high-risk goalie. Therefore, the mechanical engineering group will be working on a low risk design that can be easily implemented if something happens to theirs. Scooping methods of gripping the ball were considered to be too risky for the mechanical design team to undertake.

The rules prohibit the goalie from kicking the ball past half field, so the goalie would not be able to use a powerful kicker like the players. Also, the goalie is designed for high acceleration, so its "bump passing" abilities should be better than the other players'.

The current mechanical engineering goalie design utilizes the same motors as the field players. The goalie will be geared lower for higher acceleration. The layout of the goalie will also be similar. The body will look like a box with the length being close to 18cm. This maximizes the blocking surface so that the actual travel to stop a ball is minimized along with the angles of shots to the goal.

# Section 9 Detailed Design

## 9.1 System Overview

The overall goal for Team Brazil was a robust implementation for RoboCup, with built-in flexibililty for future modifications or additions. The vision system was deemed to be a major expense and a major commitment in terms of locking in the technology for an extended amount of time. Therefore, the vision system was jointly chosen and purchased. Once the vision system was decided upon the other sub-systems, artificial intelligence (AI) , communications, local processing, and robot structure and drivetrain, were designed to utilize the speed and processing power of the vision system.

The vision system consisted of a high-end Sony camera with a zoom lens, and a digital signal processing (DSP) board hosted on a Pentium® II computer. The AI was designed as a Role-based model where a Role would get assigned to a robot if the situation were “right.” The Role-based model allows for a fairly fixed interface to allow for modularity and easier debugging. For concurrent engineering, the AI was designed with the assistance of a physics-based simulation of Team Brazil’s robots playing a soccer game, while the robots were being manufactured. The AI executables were hosted on a Pentium® II computer as well. For the Cornell University competition the AI and vision were hosted on separate computers, but we hope to host both on one Pentium® II computer for the international competition. The AI then passed out target points to a trajectory generation algorithm. The trajectory algorithm planned out the path that the robot should take and sent out the packetized commands through the wireless communications hardware to the robot. The packets were received on the robot by a transceiver and sent to the local microcontroller. The microcontoller then handled interpreting the commands, and controlling the motors. The construction of the robot is such that the electronics board can be swung upward for removal of internal component or for testing purposes. The robot is also symmetric to allow for easier orientation of the robot during gameplay. The drivetrain of the robot is strong enough to achieve speeds close to three meters per second.

The overall system runs at 60 Hertz, with vision currently outputting new data at a rate of 30 Hz. The system has the capability of running at 60 Hz with more optimized code. The following sections will delve into the details of each sub-component of the system, with vision contained in a separate document. (The attached flow diagram describes the stages needed to provide an accurate account of data flow throughout out the entire system.)

## 9.2 On-board Electronics

The electronics are separated into four circuit boards. First, the pre-made Freedom16-mite MCU board is integrated into the main board similar to a removable socket device. Since the pin layout is set in the Freedom board (See *Freedom16-mite board layout* section 5.3.3.1.1.), the main board pin setting is fixed for the MCU and we have to design from there.

The main board contains all of the user interface, power regulation, motor direction indicators, and connections of the RF transmitter. This board is placed on the top of the robots and is easily accessible. The motor controller boards are placed right above the motors and use an aluminum mount as a heat sink. This location is chosen to minimize the distance of the high current wires. The motor control boards have no connectors and can be considered as a disposable unit should they malfunction.

The designs of each part of the electronics and software are discussed in the sections below.

### 9.2.1 MCU

The Freedom16-mite is built with the Motorola 16 bit MC68HC16Z1 chip. The 68HC16Z1 has specific features such as pulse counting, high speed inputs and outputs, 2 pulse width modulation (PWM) outputs, 39 digital I/O ports, and 8 channels of 10-bit A/D ports. On top of the 96K on chip memory, the F16-mite includes 128K of SRAM and 128K flash EPPROM. Other features of the Freedom16-mite board are a RS232 port, debug ports, and ten interrupts. With a board size only 5.2cm X 5.2cm, it is a good fit for our application.

The connector and port functions of the Freedom16-mite board are listed below.

CN1 – Debug Port

One of the features that sets the MC68HC16 chip set apart from other controllers is the monitor/debug capabilities which are built-in on the chip. This allows the debugging software to take control of the process and command all the processor instructions from this port. This means we do not have to load special debugging software that will take up program space. The ICD cable that comes with the evaluation kit has all the configuration set up which user intervention is not needed.

CN2 – General Purpose Digital I/O Pads & Interrupt Request

There are 15 general-purpose digital I/O channels in the HC68HC16 chip (Port E & F). Each pin can be programmed as an input or output. Seven of the I/O channels (Port F [7..1]) can be programmed as interrupts that are called when the input is pulled low. In our design, we use these ports to handle the RF interrupt and other general I/O functions (*See pin connection*).

CN3 – Expansion Connector

This port has 8 data, 3 address, 9 chip select lines and 7 Queued Serial Peripheral Interface (QSPI) lines, which can facilitate off board expansion or drive components that could be mounted on the prototyping area. The six chip select lines can be programmed as digital outputs and the 7 QSPI lines can be programmed as digital I/O ports. We used port 3 to communicate with the LCD and handle the data transfer to the Radiometrix transceiver.

CN4 – General Purpose Timer Port

This port offers a number of timer related channels. There are two 8-bit Pulse Width Modulation channels (PWM) suitable to drive our motor controller. This port is used to time the falling edges for the encoder signals from the left and right motor. Setting up the system clock divide function can control the 16-bit resolution of the timer. We can obtain a 17-bit resolution by using the carry interrupt bit on the timer.

CN5 – RS232 Serial Port

This 2x5 header port offers RS232 level Rx and Tx signals, along with a ground reference. This port can be used to communicate with the PC or used it as general purpose I/O.

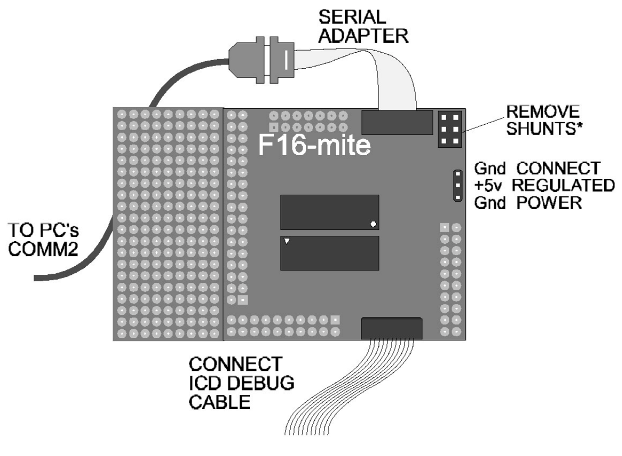
CN6 – Power Supply

The board required +5volts supplied through this port to operate.

CN7 -- 8 Analog Inputs

We use only one of the eight A/D ports, for checking the voltage level of the power supply. Since no other device needs to use any of the A/D ports, we use the other 4 channels as the input signals of our dip switch.

The ports are selected to use the Freedom-mite board and the MCU to its fullest potential by leaving ports that might have future value open. For example, the ADC ports are used as inputs for the dip switch instead of a general purpose I/O digital port. This is done because there seems to be little use for 7 ADC ports but a digital port is more useful incase we want to use a PID to control the motor in the future instead of software control. There are still 3 ADC ports open. Care is also taken to make the software programming of the ports simple and logical. Where possible we tried to place logic functions together. Below is a listing of the final  connections to the Freedom-mite board. Please refer to the *Freedom16-mite board layout and connector position* for location of the connector.



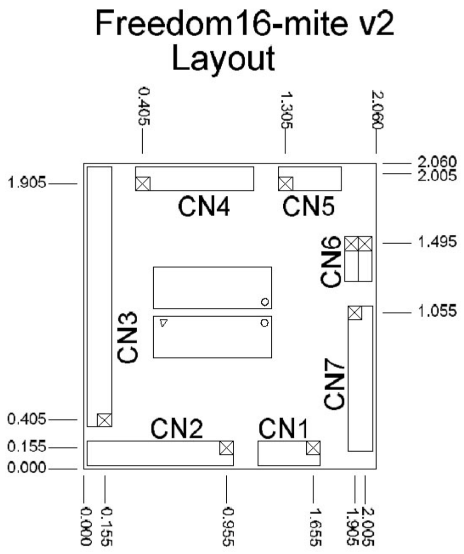


**Figure 6. Freedom16-mite board layout and connector position**

The ports and pin locations on the MCU listed for easy access (See *Ports and pin layout of the MC68HC16Z1).*

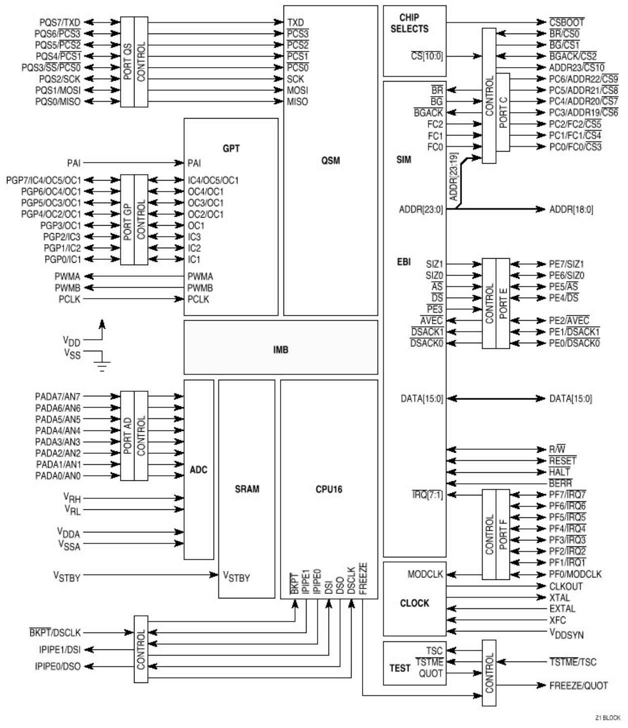
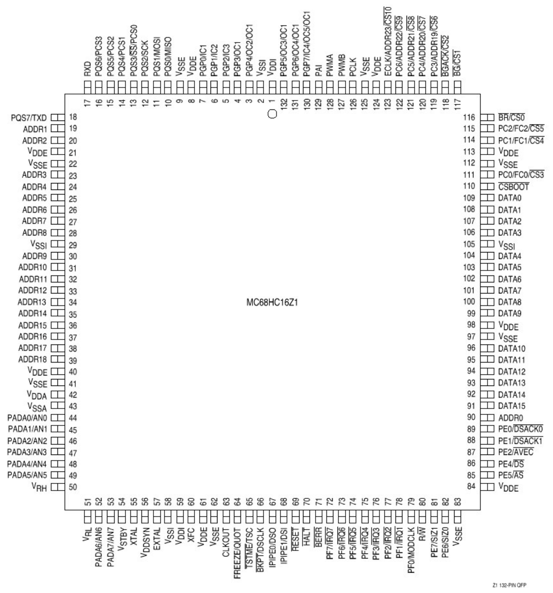


**Figure 7. Ports and pin layout of the MC68HC16Z1**



## 9.2.2 MCU Software Design

We have decided that we are going to program all behavior functions of the MCU in C. The P&E ICD debugger software is used to download the code into the MCU. The code is very well commented and can be found at "`Y:\electrical\software\68hc16\softmc\main.c`". The full list of functions and how they are called is in the table bellow.

| Function | Called by |
|---|---|
| Main | Start of program |
| Port_init | main |
| Kick | port_init, main, test_robot_motor |
| Speed_control | pit_isr |
| Test_batteries | main |
| Test_robot_motor | main |
| User_output_set | main |
| Decode_rf | irq6isr |
| Write_port | all functions |
| Limit_range | speed_control |
| Test_rf | irq6isr |
| rf_encoder_decode | irq6isr |
| rf_send_data | test_rf |

| | |
|---|---|
| irq6isr | Interrupt called |
| ic1_isr | Interrupt called |
| ic2_isr | Interrupt called |
| ic3_isr | Interrupt called |
| Tov_isr | Interrupt called |
| pit_isr | Interrupt called |

A description of the code and the functions in it are found below.

The program is divided into six sections and can be found be searching on these lines:

```
/*program organization:*/


/* INCLUDES */
/* DEFINES */
/* GLOBAL VARIABLES */
/* FUNCTION DEFINITION */
//------------- MAIN ---------------------------------------------
//------------- SUBROUTINES --------------------------------------
//------------- SUB-SUBROUTINES ----------------------------------
//------------- INTERRUPTS ---------------------------------------
//------------- FUNCTIONS DEFINITIONS ----------------------------
//------------- TEMPLATES ----------------------------------------
```

There are print statements in a lot of places in the code to print out variables. This can be very helpful when debugging the code on the computer. To turn on and off the print statement set the line

```
#define DEBUG_ON 0      //debug variable to turn on debug functions
```

equal to 1 instead of 0.

Main( )

This function controls the state that the robots are in and calls functions as required. It first calls port_init( ) to set-up the ports and initial all the variables to known states. When port_init( ) is complete the main( ) gets the status of the dip switches and goes into test mode or play depending on the dip switch settings. It also determines the robot number by receiving signals from the first three pins of the dip switches The main loop is just in a while(1) and cycles through continuously reading the dip switches, testing the batteries, and displaying information to the LCD and LEDs.

port_init( )

This function sets-up all of the ports to the correct I/O or interrupt. It also handles initializing all of the internal registers. The ordering of this function is very important. For example you cannot turn on the interrupts without first setting up the ports to the correct I/O values. The Radiometrix transceiver is also reset during this function.

This function also tests that the kick is fully retracted and turns it on if it is not. This would allow the kicker to move back to its ready position.

Kick( )

The kicker function sets port E pin 7 to high and sets a counter variable to current time.

speed_control( )

The speed_control handles the feedback loop for the wheel. The software feedback control is done by a P.I. controller. The basic idea of a control loop is to take the desired velocity command, send that command to the motors, and see how fast the motors actually spin by getting the data from the encoder. The speed of the motors is compared to the commanded speed. The difference is the error signal and it can be either positive or negative. The control loop will provide a signal to the motors proportional to the difference between the actual and desired velocities, and proportional to the integral of the error signal over time. The gain of the loop can be tune by changing the constants of the P.I. loop. We have used a rather low-gain at this point since it will save battery power and it’s already good enough for the Vision system to control the robot.

This function also houses the watchdog timer and system timer variables. The loop checks to see if the kicker has timed out. It will turn off the kicker if ic3_isr was not called with in 15 seconds after turning on the kicker. The speed control loop also gets the current value of the buttons.

test_batteries( )

The test_batteries function tests the voltage at the ADC port PADA7. It then returns it the caller.

test_robot_motor( )

The function takes in a test number and runs that test. The numbers are gotten by the button input.

user_output_set( )

This function sets up the LEDs and LCD output depending on the value of the status bit.

decode_rf( )

Once the robot receives the RF message, irq6isr calls this function. It decodes all of the information based on the number of the robot and then the information will be stored in global variables.

write_port( )

This function is used to write a binary value to a port without destroying the information that is already there.

limit_range( )

This function limits the range of a value passed to it. It is currently used for the PWM value since it can not be larger then 256.

test_rf( )

This is called by the irq6isr when the robot is in RF digital test mode. First, the global computer transmits a 32 bit packet. This packet contains two matching 16 bit numbers. The robot acknowledges proper receipt of this packet by transmitting the 16 bit number back to the global computer. A copy of this number is stored on-board in the robot. If the global computer never receives a response, it will rebroadcast the packet. Upon receiving the confirmation from the robot, the global computer begins sending a stream of packets to the robot. These packets hold values starting at test length - 1 and count down to zero. For example, a robot running a test of length 50 would send the packets 50|49, 50|48, 50|47…50|0. The robot has its own counter that counts down by one every time it gets a packet. If the counter on the robot is not the same as the test data then the robot missed packets. The number of packets it missed is equal to the difference between the counter and the test data number. The robot then sets its counter to the test data to resynchronize with the global computer. When the robot gets a test data of zero it returns the number of errors it got to the global computer. If the global computer does not get the packets of number of errors from the robot it resends the zero test packet.

rf_encoder_decode( )

This program can be used to characterizing the robots. It is designed to get a test size, mode, and wheel velocities from the global computer and then send back the encoder values. It is also to demonstrate the type of information that can be returned with the telemetry systems.

rf_send_data( )

This function in used to send data back to the global computer. To use it the user loads the rf_data_out[] array and calls rf_send_data to tell it the number of packets to return.

irq6isr( )

This interrupt is called by the Radiometrix transponder. It down-loads the data from the transponder and stores it in an array, rf_data_in[]. It then calls the appropriate function depending on the robot mode.

ic1_isr( ) and ic2_isr( )

These interrupts are called by the encoder signals of the left and right motors respectively. The function stores the difference between the current value and the last value of a free running counter. The difference can be directly related to time because we set the speed of the free running counter.

By knowing that the motor encoders have 100 division per turn, the gear ratio is 1:7.4, the clock speed is 25.1MHz, and the robot wheels are 0.076m in diameter we can calculate the speed for a given number of counts in the free running timer. By setting to clock divide by 8 we get the table below relating the speed of the robot in m/s to the number of clock cycle that we got from the interrupt ic1_isr.

| Clocks | m/s |
|---|---|
| 253.075 | 4.000000 |
| 506.15 | 2.000000 |
| 1012.3 | 1.000000 |
| 2024.6 | 0.500000 |
| 4049.2 | 0.250000 |
| 8098.4 | 0.125000 |
| 16196.8 | 0.062500 |
| 32393.6 | 0.031250 |
| 64787.2 | 0.015625 |
| 129574.4 | 0.007813 |

ic3_isr( )

This interrupt is the input to the Hall effect sensor. This turns off the kicker when it is called by setting port E pin 7 to low.

tov_isr( )

This is the overflow of the free running counter and is used to get a 17-bit resolution on the wheel speed.

pit_isr( )

This interrupt is the periodic interrupt timer and calls the speed_control function. The time interval that it is called in is set by PIT period equal to [clk_div*4]/32.768KHz*(speed), where clk_div is 1,2,4,8,16, or 32 and speed is 1 or 512. We set this to PIT period equal to [6*4]/32.768KHz*(1)=0.00073242187 second, about (1/1200)Hz.

The LEDs functions are described bellow:

Play mode:

Green: This indicates that you are in play mode and that the MPC has booted correctly.

Yellow: This indicates that the batteries are low. It will light even if you just replaced the 9volt since it is mainly for the NiCd.

Red 1: The kicker has timed out. This comes on when the Hall effect sensor has not detected anything in the last 15 seconds.

Red 2: Nothing

In test mode:

Green light: always off

| Yellow | Red 1 | Red 2 | Funciton |
|---|---|---|---|
| On | On | On | motor test mode |

| | | | |
|---|---|---|---|
| On | On | Off | digital RF test, ready |
| On | Off | Off | Digital RF is running |
| On | Off | On | Digital test dropped a packet |
| Off | Off | On | Joystick test mode |

The full list of functions of the dip switches is found bellow.

| Dip 1 | Dip 2 | Dip 3 | Dip 4 | Mode | Robot # | Function |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | play | 0 | robot 1 |
| 0 | 1 | 0 | 0 | play | 1 | robot 2 |
| 0 | 0 | 1 | 0 | play | 2 | robot 3 |
| 0 | 1 | 1 | 0 | play | 3 | robot 4 |
| 0 | 0 | 0 | 1 | play | 4 | robot 5 |
| 0 | 1 | 0 | 1 | play | 5 | none |
| 0 | 0 | 1 | 1 | play | 6 | none |
| 0 | 1 | 1 | 1 | play | 7 | none |
| 1 | 0 | 0 | 0 | test | 0 | motor test |
| 1 | 1 | 0 | 0 | test | 1 | RF digital |
| 1 | 0 | 1 | 0 | test | 2 | encoder |
| 1 | 1 | 1 | 0 | test | 3 | none |
| 1 | 0 | 0 | 1 | test | 4 | none |
| 1 | 1 | 0 | 1 | test | 5 | none |
| 1 | 0 | 1 | 1 | test | 6 | none |
| 1 | 1 | 1 | 1 | test | 7 | joystick |

Dip switch states

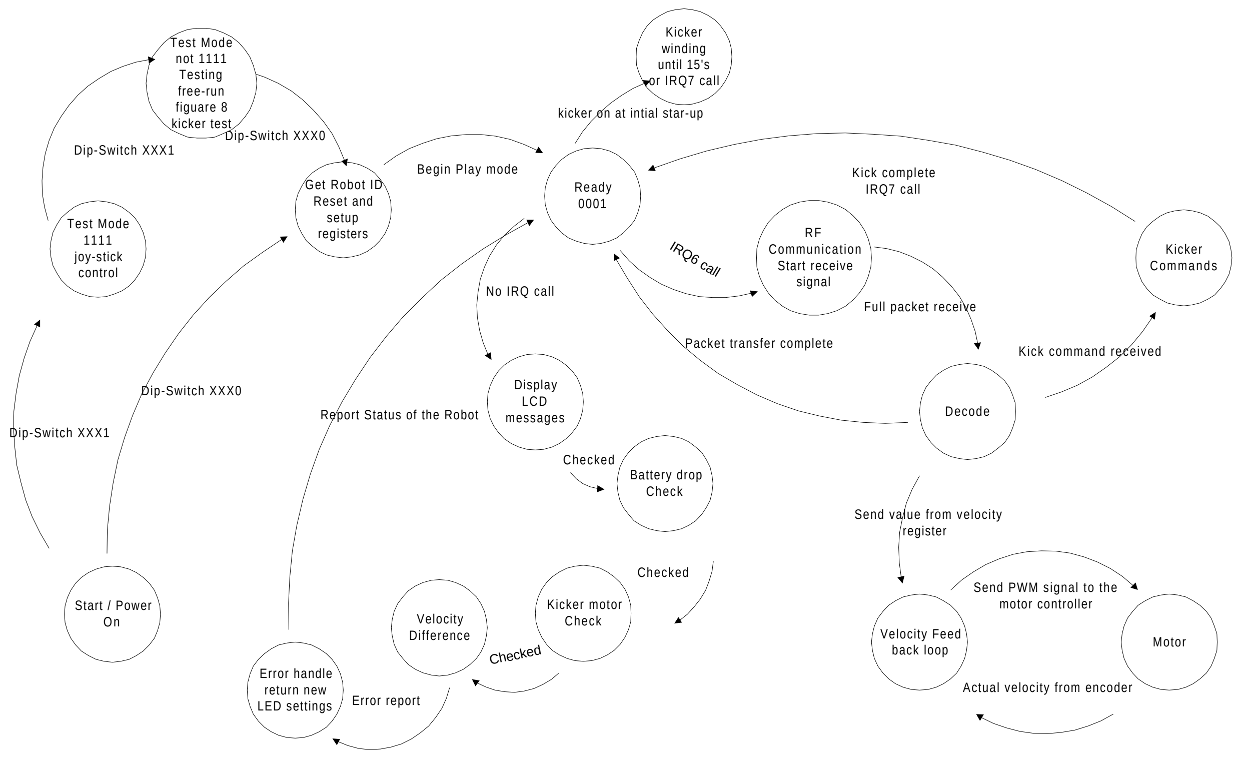

Test Mode not 1111 Testing free-run figuare 8 kicker test
Kicker winding until 15's or IRQ7 call
kicker on at intial star-up
Dip-Switch XXX1
Dip-Switch XXX0
Begin Play mode
Get Robot ID Reset and setup registers
Ready 0001
Kick complete IRQ7 call
Test Mode 1111 joy-stick control
IRQ6 call
RF Communication Start receive signal
Kicker Commands
No IRQ call
Full packet receive
Packet transfer complete
Kick command received
Dip-Switch XXX0
Display LCD messages
Report Status of the Robot
Decode
Dip-Switch XXX1
Checked
Battery drop Check
Send value from velocity register
Checked
Start / Power On
Velocity Difference
Kicker motor Check
Send PWM signal to the motor controller
Velocity Feed back loop
Motor
Checked
Error handle return new LED settings
Error report
Actual velocity from encoder

**Figure 8. State diagram**

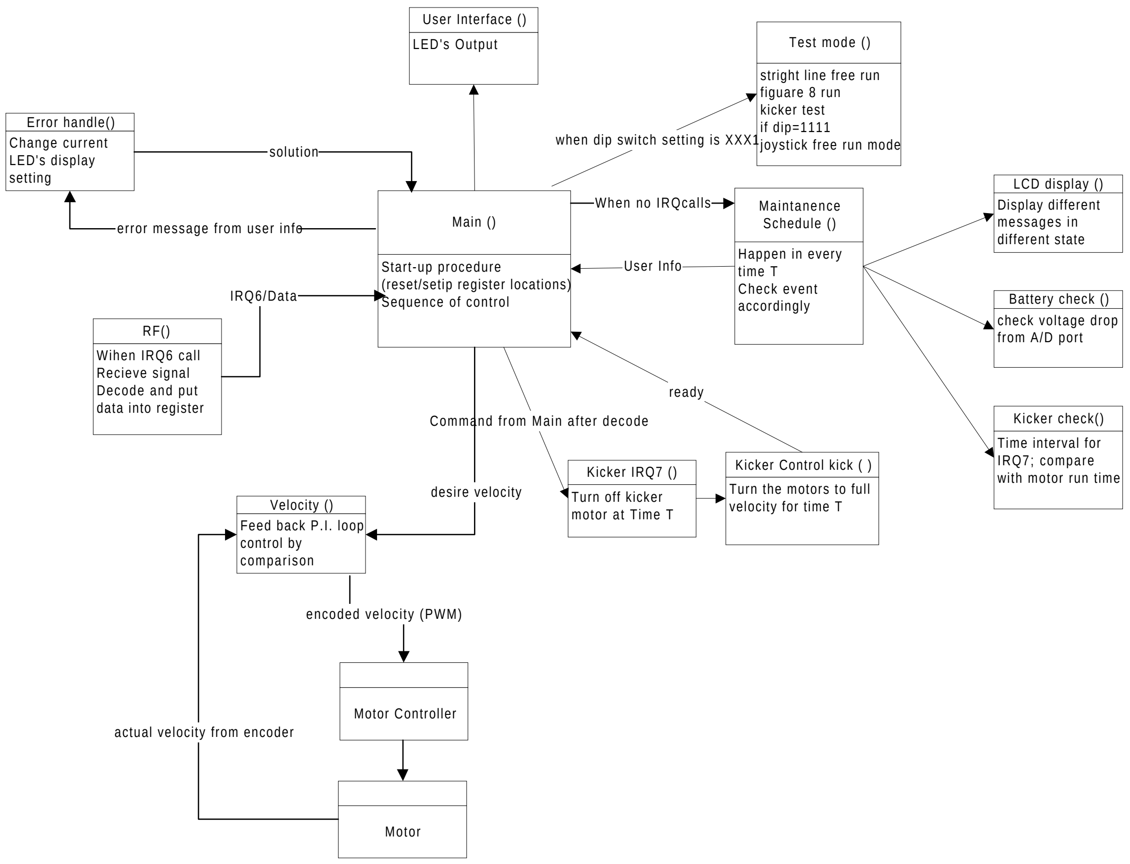

User Interface ()
LED's Output
Test mode ()
stright line free run
figuare 8 run
kicker test
if dip=1111
joystick free run mode
when dip switch setting is XXX1
Error handle()
Change current LED's display setting
solution
error message from user info
Main ()
Start-up procedure (reset/setip register locations) Sequence of control
When no IRQcalls
Maintanence Schedule ()
Happen in every time T
Check event accordingly
User Info
LCD display ()
Display different messages in different state
Battery check ()
check voltage drop from A/D port
Kicker check()
Time interval for IRQ7; compare with motor run time
IRQ6/Data
RF()
Wihen IRQ6 call
Recieve signal
Decode and put data into register
ready
Command from Main after decode
Kicker IRQ7 ()
Turn off kicker motor at Time T
Kicker Control kick ( )
Turn the motors to full velocity for time T
desire velocity
Velocity ()
Feed back P.I. loop control by comparison
encoded velocity (PWM)
Motor Controller
actual velocity from encoder
Motor

**Figure 9.** **Function layout**

### 9.2.3 Kicker

There are two simple components in the kicker electronics, the hall-effect sensor and a BJT darlington transistor. The hall-effect sensor is used to detect the location of the kicker and the transistor is used to provide power to the kicker motor. It is very important that the MCU responds quickly to turn the motor off or else the kicker will kick when it should be reset. For this reason an interrupt port was used to detect the Hall effect sensor. The Hall effect switch is chosen because it does not bounce and is very reliable. We could have used a schmitt trigger but this would add complexity and reliability problems. The motor is driven with a darlington transistor directly with the output of port PE7 on the MCU.

The motor will shut off once the magnet located on the top surface of the kicker gear passes under the hall-effect sensor. The hall-effect sensor will signal to the MCU via interrupt once it detects a magnetic field. When the kicker motor is switched-off, it will not turn back on until a kick command is received from the global computer. Upon receiving a kick command, the kicker motor is turned on. This causes the kicking plates to release and wind back up though a "clockwork" mechanism. The motor will shut off again just before the mechanisms release point when the hall-effect sensor detects another strategically positioned magnet.

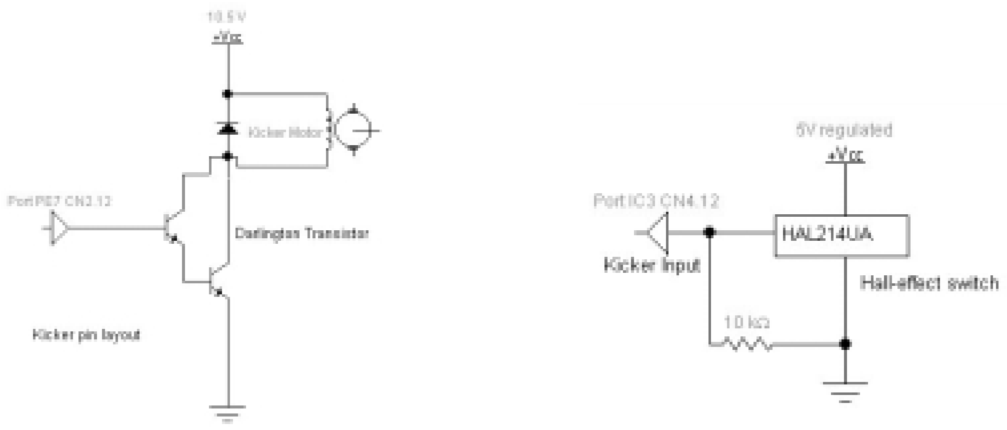


**Figure 10. →Kicker pin layout**

### 9.2.4 User interface

Since there will only be two to three persons helping during the competition, we would like to design our user interface for the on-board electronics to be as simple as possible. As our robots are identical, it is easy for our robots to be able to inter-changeable. If our program for the MCU is robot dependent, then each time there is a need to change a robot the user would have to download the appropriate program from a PC. In order to make it easier to handle, we decided that the robot number would be determined by a dip switch that the user could set by themselves. Also, the dip switch would be used to determine program mode so that the user can test the robot easily. The dip switch is readily accessible to the user with fewer confusion than using jumper configuration. There are 4-buttons lay-out on the main board in which less are designed to be used with the LCD screen. The user can make selections by pushing the buttons.

We decided to use four LED lights to indicate the state of the robot. By doing so, the user can understand the state of the robot easily. They also provide very useful information for debugging and error detection. The LED lights light up for different states of the robot such as green for ready state, yellow for battery low, and red for other error.

The LEDs are low current and can be driven directly with the MCU. The LED draws 2mA of current and when they are all 'on' the power lost is only 0.04W. This power cost is justified because it saves time in debug. The LED's are also designed to be on when the pin is in high. This will help to make the code easier to write and easier to understand (See *LED output pin layout*).

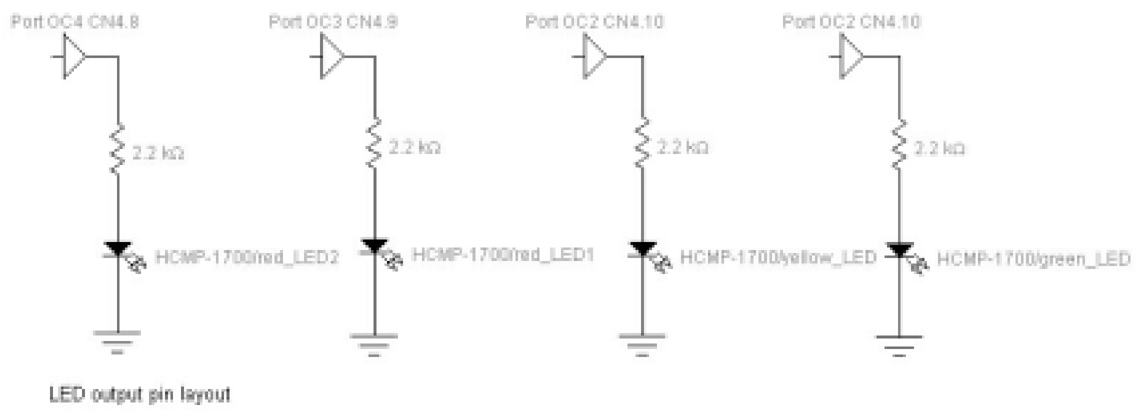


**Figure 11. LED output pin layout**

•

The dip switches are pulled high by a resistor (See *User input pin layout*).

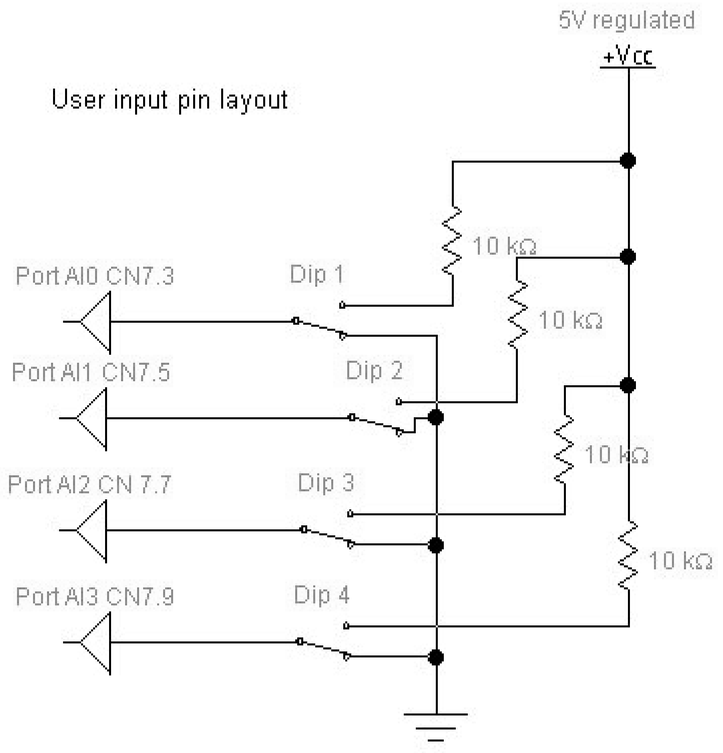


**Figure 12. User input pin layout**

In addition to the LED lights, we have added a 20x4 LCD panel to make the robot more user friendly. The idea of the LCD panel is to help debug the robot during testing in the lab. As of now, the LCD will provide information such as the voltage level of the battery, robot name, number and menu options. The LCD is not intended to be used during the competition. The buttons in conjunction with the LCD panel can give the user control of many possible functions by using a menu system.

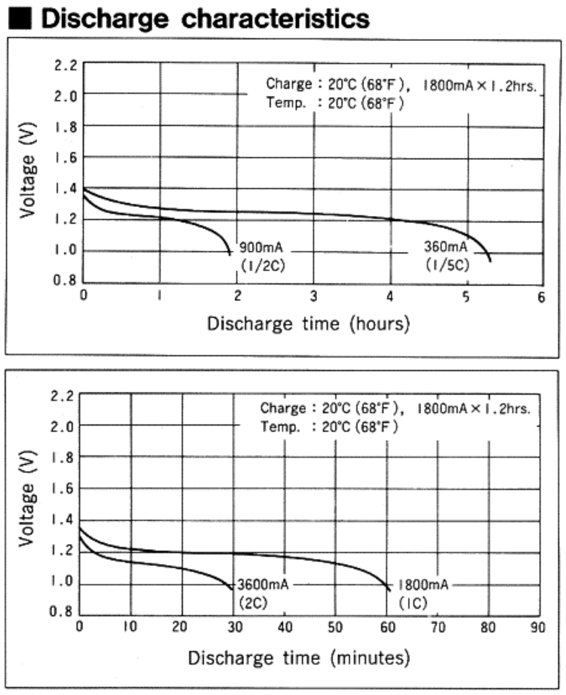
### 9.2.5 Power

#### Battery power supply

Two batteries seem to meet the power requirement for the robot, NiCd High and NiMH. First, we want to know how much power a robot requires during the game. The robot gets new batteries at half time. This means that robots run for 10-15 minutes on one battery pack. Looking at the energy required for average motor speeds and the kicker, we need 0.3Ah's of capacity. (See *Total continuous current*). When selecting a battery size one has to understand that batteries are rated for much lower discharge rates. So drawing all that power at once requires a much larger battery (See *Discharge curve*). For this reason we want a battery with an energy density of at least 1.0Ah.

| **Current Regulated** | | | **Current Unregulated** | |
|---|---|---|---|---|
| **Part** | **morn** | | **Part** | **Max** |
| Freedom-mite | 300 | | Wheel Motor | 3.33 |
| L298HM 4x | 96 | | Kicker | 0.5 |
| LED's | 7.8 | | **Total** | 3.83 |
| FM-RPC-XXX | 20 | | | |
| **Total** | **423.8** | | | |

Total continuous current



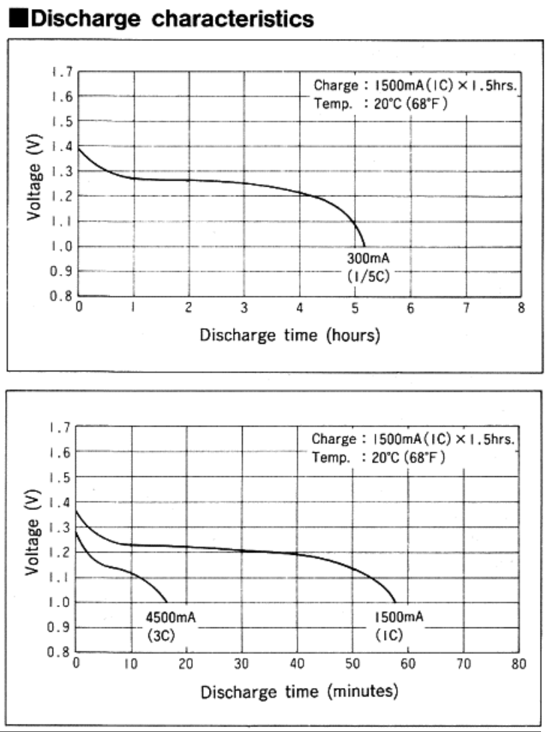


**Figure 13. Discharge curve for NiMH Discharge curve for NiCd**

Next, we want to make a decision on which battery to use. We make a comparison of size and composition of several batteries (See *Comparison*). Looking at the chart, we find that the size 4/5A has the best energy-weight ratio for both NiCd High and NiMH. This size satifies the energy requirments for the robots with room to spare. Both are easy to recharge and require about the same time to do so. NiCd high has a lower internal resistance and can deliver more instantaneous current then NiMH. This advantage out weighs the slightly larger energy destiny of NiMH.

| | | | | | | mm | mm | grams | grams | Want max # | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Types** | **voltage** | **size** | **Ah** | **cost** | **Total Cost** | **Height** | **Diameter** | **Weight** | **T Weight** | **Wh/kg** | **Wh/volumn** |
| NiCd high | 1.2 | 1/3AA | 0.18 | $3.09 | $24.72 | 16.9 | 14.6 | 7 | 56 | 30.86 | 9.55E-06 |
| NiCd high | 1.2 | AAA | 0.22 | $3.26 | $26.08 | 44.8 | 10.5 | 10 | 80 | 26.40 | 8.51E-06 |
| NiCd high | 1.2 | 2/3AA | 0.40 | $2.20 | $17.60 | 30.7 | 14.6 | 12 | 96 | 40.00 | 1.17E-05 |
| NiCd high | 1.2 | AA | 1.00 | $4.08 | $32.64 | 50.0 | 14.5 | 23 | 184 | 52.17 | 1.82E-05 |
| NiCd high | 1.2 | 5/4AA | 1.20 | $5.29 | $42.32 | 65.0 | 14.5 | 31 | 248 | 46.45 | 1.68E-05 |
| **NiCd high** | **1.2** | **4/5Af** | **1.50** | **$4.96** | **$39.68** | **43.0** | **17.0** | **27** | **216** | **66.67** | **2.31E-05** |
| NiCd high | 1.2 | Af | 1.60 | $4.72 | $37.76 | 50.0 | 17.0 | 33 | 264 | 58.18 | 2.12E-05 |
| NiCd high | 1.2 | SC | 1.80 | $5.86 | $46.88 | 42.5 | 22.5 | 47 | 376 | 45.96 | 1.60E-05 |
| NiCd high | 1.2 | 7/5Af | 2.00 | $7.22 | $57.76 | 66.2 | 17.2 | 43 | 344 | 55.81 | 1.95E-05 |
| | | | | | | | | | | | |
| NIMH | 1.2 | AAA | 0.6 | | | 44.50 | 10.50 | 12 | 96 | 60.00 | 2.34E-05 |
| NIMH | 1.2 | AA | 1.1 | 4.17 | $33.36 | 50.50 | 14.50 | 25 | 200 | 52.80 | 1.98E-05 |
| **NIMH** | **1.2** | **4/5A** | **2** | | | **42.00** | **17.00** | **32** | **256** | **75.00** | **3.15E-05** |
| NIMH | 1.2 | C | 2.6 | 12.92 | $103.36 | 49.00 | 26.00 | 70 | 560 | 44.57 | 1.50E-05 |

Comparison of the different size and types of batteries

### Power On-board

For the power we are going to have one battery for the digital and analog circuits with a second switching in only when needed. In the design we have selected a low voltage drop voltage regulator. With a .5V voltage drop and a minimum input voltage of 7.4 volts. Even if the wheel motors drain the batteries to less than 5.5 volts we are going to buffer the output of the voltage regulator to prevent the drop from reaching our digital circuits. Also, there is a 9V battery supply installed on-board as a back-up power supply. Once the power on the main battery drops below the threshold voltage, the 9V will supply current. This will prevent the robot from restarting during the game due to huge power drain from the motor. If only the 9V-battery is installed in the robot, the circuit will only apply the voltage to the digital circuit and the analog circuit will be shut off.

There will be a 10A fuse for the analogy circuit and a 0.8mA fuse protecting the robot from shorts. There is a dual mode on/off switch for only the digital circuits or the whole robot including the motors. This will be useful during debugging stage as the robot will not move around unintentionally (See *Power control pin layout*).

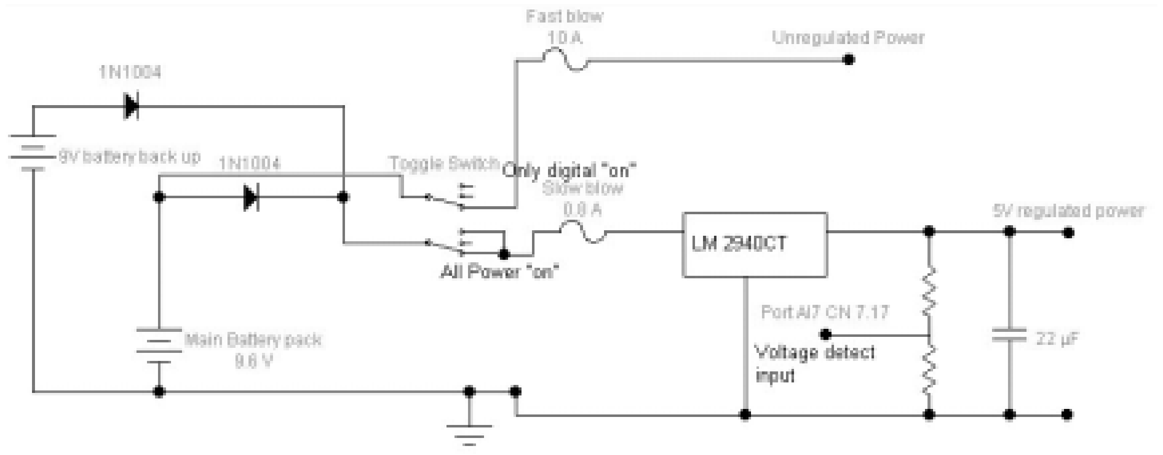


**Figure 14. Power control pin layout**

### 9.2.6 Motor controller

The motor controller is the SGS-Thomson's L298. It can handle a peak current of 6Amps when the 2 H-bridges are tied together. The motor stall current is 9.450 Amps (See table), since one L298 will not handle that peak current. We decided to use two L298 for each motor to handle the huge power drain.

Each H-bridge in the L298 will be tied together to increase the maximum power output and then two L298 will be tied together to increase the power output even more. See *Internal wiring and pin connection for L298* for the pin connections and internal operation.

| | | | | | Max Current output | |
|---|---|---|---|---|---|---|
| **Motor specification** | | | | **number of L298:** | 1 | 2 |
| motor resistance | 0.635 | Ohm | | Non repetive (t=100micro sec) | 6A | 12A |
| stall current | 9.450 | Amp | | Repetive (20% off-80% on) | 5A | 10A |
| no load current | 0.110 | Amp | | DC operation | 4A | 8A |

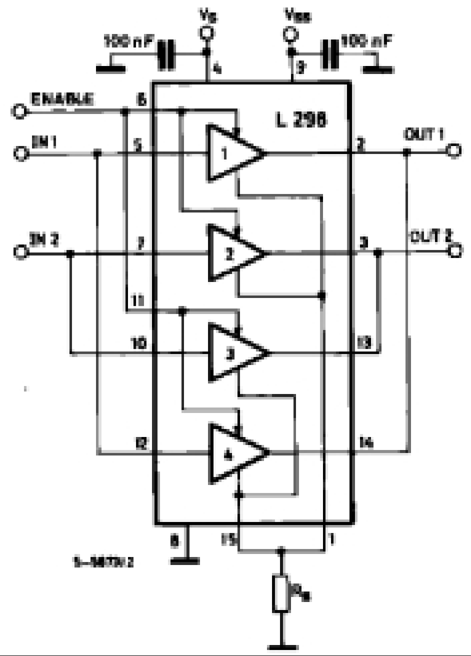


**Figure 15. Internal wiring and pin connection for L298**

The L298 is protected form the inductive spike with 4 6A fast recover (Vrr = 60ns).

The L298 enables are connected to the PWM output of the MPC. This provides the correct speed of the motor based on the duty cycle of the PWM. Input In 1 and In 2 are connected to digital outputs on the MCU. See the motor controller table for the control functions of the motor.

| Motor controller | | | |
|---|---|---|---|
| **Enable** | **In 1** | **In 2** | **function** |
| 1 | 1 | 0 | turn right |
| 1 | 0 | 1 | turn left |
| 1 | X | X | brake |
| 0 | X | X | run free |

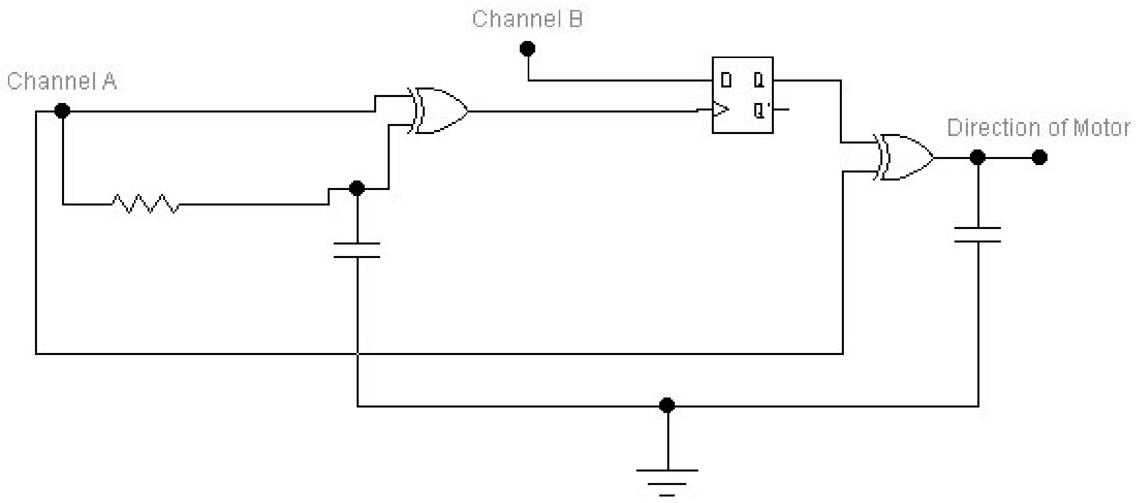


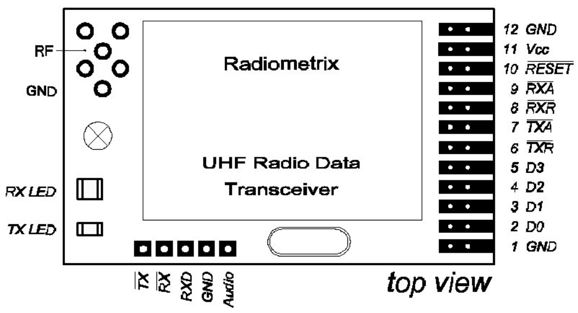
**Figure 16. Direction detection from Motor Encoder**

The motor encoders are digital outputs and can be directly input into the MCU. Both motor encoders can call in an interrupt through the general purpose timer (GPT). Channels A and channel B are used for direction and a small circuit is laid out on the main board to indicate the direction (See *appendix for the pin layout*).

### 9.2.7 Wireless Communication

For the transmitter we are using the development kit for the RPC-XXX-F. The kit comes with software and the cables to connect the transmitter to the computer. The kit can be used to test and debug our RF communications. The only thing missing from the kit is a 7 to 12 V 40mA power supply.

The RPC-XXX-F can handle the entire packet encoding and decoding. If a packet arrives it will indicate to the MCU via the interrupt line 6 by putting RXR high, and then it will receive the packet. The MCU will then respond, stating it's ready to start receiving by setting RXA high. The RPC-XXX-F will start downloading the entire packet. The download from the RPC-XXX-F to the MCU is a 4 bit parallel load into an I/O port on the MCU (See *RPC-XXX-F wiring to Freedom connectors*).



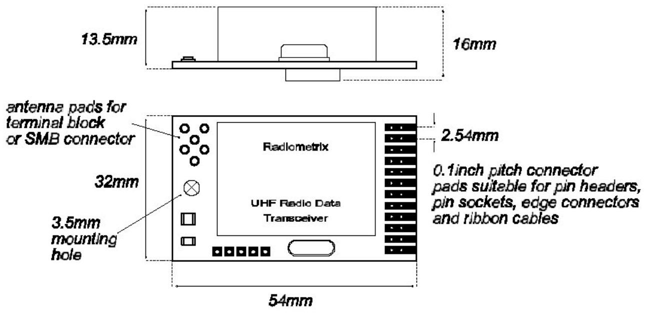

13.5mm
16mm
antenna pads for terminal block or SMB connector
2.54mm
Radiometrix
UHF Radio Data Transceiver
0.1inch pitch connector pads suitable for pin headers, pin sockets, edge connectors and ribbon cables
32mm
3.5mm mounting hole
54mm

**Figure 17. RPC-XXX-F pin layout and physical dimensions**

Since the cost for connecting the transmit lines (TXA and TXR) to MCU is low, we decide to connect them. This will give us the ability for two-way communication. The reset line connects to the MPC so that we can reset the RF on start up or if a failure is detected.

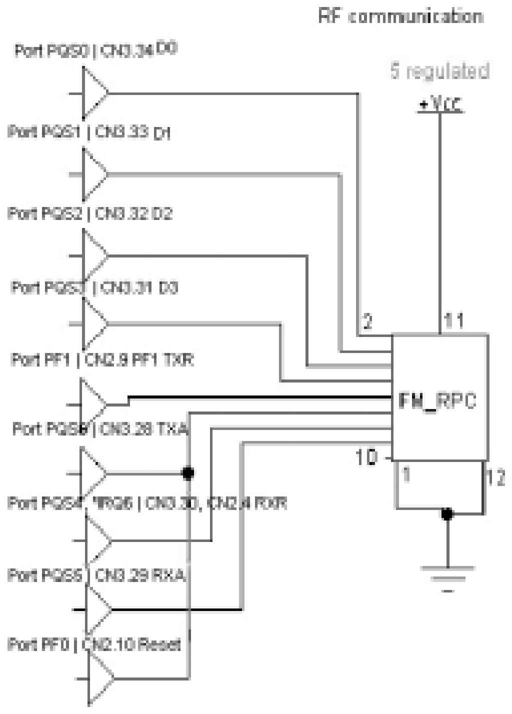


**Figure 18. RPC-XXX-F wiring to Freedom connectors**

## Robot packet design

The robots require wheel velocities and action commands. This data can be transmitted for all five robots in 100bits. The RPC-XXX-F can transmit 216bits at 72.5Hz. Since there is a 5ms second penalty for each packet sent a 100bit packet can transmit at 110.2Hz. The minimum requirement for the robots is 100bit at 60Hz. So we are doing better then required. The transmitter will continuously transmit the wheel velocities and action even if the velocities have not changed. In this way if a packet is lost, the robot can get the next packet even if the global computer has not updated the velocities. For the kick command, the global computer can transmit several times. This will not change the function of the robot kicker because the kicker motor will be 'on' anyway.

At every 60Hz, 25 packets / 100 bits of data will be sent to the robots. Each individual robot has its own 5 x 4 = 20 bits of data to decode. The data packet setup for one robot is as follow:

| | | | | | |
|---|---|---|---|---|---|
| 4 bit $V_{LH}$ | 4 bit $V_{LL}$ | 4 bit $V_{RH}$ | 4 bit $V_{RL}$ | 2 bit dir | 2 bit action |

The first eight bits is the left wheel velocity, followed by the right wheel velocity, left and right directional and the two bits action command where ‘1’ indicates to kick. Each robot can decode their own packet by their robot number in which robot ‘0’ will decode the first 5 packets and robot ‘1’ will decode packets 6-10 and so on.

## 9.3 AI Detailed Design

### 9.3.1 Environment

#### Language/Development Environment Notes

The Team Brazil Artificial Intelligence subsystem was implemented using Microsoft Visual C++ version 6.0. The code base is not heavily object-oriented; most of the functions in the overall system are global in scope, rather than member functions. The code utilizes the C++ memory-handling model, and the roles (discussed below) use object-oriented ideas. The code uses the C stdio functions, string manipulation functions, and the Win32 console output routines, rather than C++ I/O streams and string type. A small portion of the code (the vision-AI networking code) is implemented entirely in C, rather than C++.

There are a number of compile-time options that are toggled by setting compile flags. These are noted in section 1.1.3.

Note that any references in this documentation and in the source code referring to “RPC” refer to the “Radio Packet Controller”, NOT to “Remote Procedure Calls”. Note also the potential namespace conflicts that can occur with variables and files named “RPC” when compiling with Win32 headers.

Always refer to source code comments for details and clarification. Source code comments reflect the most recent changes and additions to the AI subsystem.

#### Workspace Notes & Projects

The Team Brazil AI subsystem is implemented as the Visual C++ workspace “Brazil Master Program”. The project is also called “Brazil Master Program,” and generates an executable called “Brazil Master Program.exe”. The following documentation covers the code structure, compile options, and high-level design of the project.

#### Compiler Notes (Configurations)

There are two development environments for the AI code: Simulation, and the real system. The core AI code has been designed so that it can be compiled instantly to either environment. That is, the code can be compiled to either an executable .EXE file or to a MEX function .DLL callable from Matlab.

The core AI code is a function called `RoleBrazilCall`. This is the same function that is called for either the simulation build or the system build. Two separate wrappers above `RoleBrazilCall` handle the conversions necessary to run `RoleBrazilCall` from either simulation or real system.

To compile the AI for the real system environment, simply compile and link the “Brazil Master Program” project. The files "`Brazil Master Program.cpp`" and "`wrapper.cpp`" are the two functions that interface between the user/vision and `RoleBrazilCall.`

To compile the AI for the simulation environment, run the “`make.m`” script file from Matlab, which will compile and link the necessary files for the simulation environment. The matlab file `AI_Brazil.m` and the functions `mexFunction` handle the interface between Matlab/Working Model and `RoleBrazilCall.`

Set compile-time options using the following compiler flags. Some options may be incompatible with one another. Compiler flags should be set in the "project settings" dialog for the appropriate build configuration; they should *not* be #defined in the source code itself.

Debugging flags: (Note that all debugging output files are written to the `c:\temp` directory.)

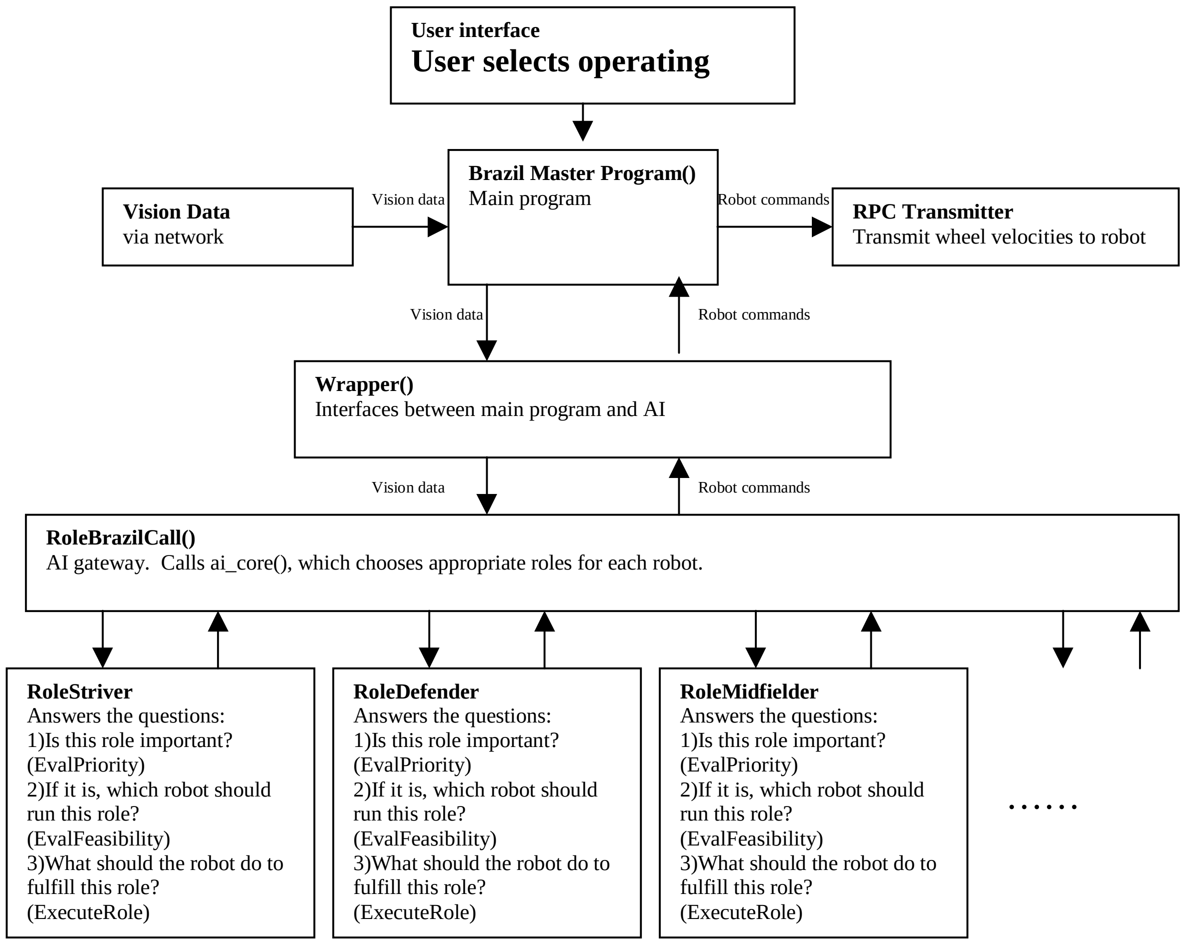
NTCDEBUG – Suppress polynomial trajectory control debugging output.

NO_MEX_COMPILE – Suppress linking with MEX-file functions in SIMULATION builds. Useful for "test compiles" in the workspace before performing a final compile using the make.m MATLAB script.

### 9.3.2 Code Structure

#### System Build

For the real system environment, the following block diagram illustrates the structure of the code. .



User interface
User selects operating
Brazil Master Program()
Main program
Vision Data
via network
Vision data
Robot commands
RPC Transmitter
Transmit wheel velocities to robot
Vision data
Robot commands
Wrapper()
Interfaces between main program and AI
Vision data
Robot commands
RoleBrazilCall()
AI gateway. Calls ai_core(), which chooses appropriate roles for each robot.
RoleStriver
Answers the questions:
1)Is this role important? (EvalPriority)
2)If it is, which robot should run this role? (EvalFeasibility)
3)What should the robot do to fulfill this role? (ExecuteRole)
RoleDefender
Answers the questions:
1)Is this role important? (EvalPriority)
2)If it is, which robot should run this role? (EvalFeasibility)
3)What should the robot do to fulfill this role? (ExecuteRole)
RoleMidfielder
Answers the questions:
1)Is this role important? (EvalPriority)
2)If it is, which robot should run this role? (EvalFeasibility)
3)What should the robot do to fulfill this role? (ExecuteRole)
……

Brazil Master Program – The main "Master" Program that interfaces with the rest of the RoboCup System: vision networking, wireless/RPC control, user interface, system initialization, and the Aritificial Intelligence code.

Wrapper – The interface between "Brazil Master Program" and the AI gateway RoleBrazilCall.

MexFunction – The interface between matlab (AI_Brazil.m) and the AI gateway RoleBrazilCall

RoleBrazilCall – The gateway to the AI code. Calls the functions for performing high-level AI functions: role initialization, priority ordering, feasibility evaluation and role assignment and role execution. The actual high-level functions are in `ai_core.cpp.`

Roles – The implementation of individual roles. Each role defines at least 3 functions: `EvalPriority`, `EvalFeasibility`, and `ExecuteRole`. In addition, a role may also define `DoRole`.

### Simulation Build

Compiling the code for the simulation requires no changes to the actual code. The only thing that must be updated is the `make.m` MATLAB script file. It should be updated to include any new roles that have been implemented. Note that this is done automatically using the Project Manager of Visual C++.

The difference between running the AI for simulation and running the AI for the real system is that in the real system AI, the Brazil Master Program runs as a continuous loop and calls `RoleBrazilCall` via wrapper once every cycle. For the simulation AI, the simulation environment calls `RoleBrazilCall` once every cycle.

### 9.3.3 High-Level AI Design

### The Role-Based AI

The high-level AI design is based on the notion of robot roles. At each cycle, the high-level AI assigns each robot a role based on the notions of role priority and role feasibility. Once the robot is assigned a role, it executes the role in two stages: a target state for the robot is selected, then this target point is converted into a robot command (wheel velocities and kicker command) by the trajectory controller.

The high-level AI is implemented in the file `ai_core.cpp`.

All roles are derived from the base role class. Roles determine their own priority, feasibility, and execution (they are implemented as the member functions `EvalPriority`, `EvalFeasibility`, and `ExecuteRole`, respectively.) Refer to the "Role" subsection below for additional notes on implementing and adding roles to the AI subsystem.

The primary data structures for the AI subsystem are declared as global variables. The following two global data structures are used for role management:

roleList – An array containing pointers to all of the instantiated roles available to the high-level AI.

roleAssigns – An array containing role priority and feasibility information.

roleAssignsCopy – An array that contains the same information as roleAssigns, but with only the highest roles in each category remaining. All others slots are replaced by NullRoles. This array is then updated to contain role-robot assignments.

The following three data structures store state information and robot commands, and are the interface between the AI and the rest of the subsystem:

fs – contains field state information (locations and velocities) for robots on the field, the ball, field dimensions, etc. This data is retrieved from the vision system every cycle, and is read (but not modified) by the high-level AI.

gps – the gameplay state, contains game state information: the current game score, the game status (time-out, normal game play, kickoff setup, etc.) This data is retrieved from the vision system every cycle, and is read (but not modified) by the high-level AI.

commands[ ] – contains robot commands generated by the high-level AI. This is copied into the fs variable (`fs.frobot[i].lwv` and `fs.frobot[i].rwv`).

Each role's member functions should refer to the game state through the member variable currState, a pointer to the current global field state variable fs. This pointer is set in AI initialization.

### Mutual Exclusion

Mutual exclusion defines categories that roles may be put into. No two roles in the same category may be active at one time. This allows more control over the activation of roles. There is a default category where roles that do not participate in mutual exclusion may be put. Hence, all roles go into *some* category.

The current list of categories is present as an enumerable in the header file `role.h` called `e_categories`. The number of categories should be no more than the number of roles. In the worst case, ever role can then assume it's own category. This is analogous to having all the roles in the default category. Currently we have categories for ball-handling roles and roles that extract the robot from walls and other robots. As can be seen, categories are useful only in special situations. `RL_CAT_DEFAULT` is the category for all roles that do not participate in mutual exclusion. A role need only initialize its category member variable to one of the predefined categories, preferably in the constructor, to be included. By default all roles are part of `RL_CAT_DEFAULT`.

Categories are implemented with the help of the role `NullRole`. This is just a role that returns 0 for all priorities and feasibilities. The function `discard_roles` performs mutual exclusion. It is called once the priorities of all roles has been found by the function `evalPriorities`.

`discard_roles` reserves an array location for every category and replaces the role offset in this category with a new role if the new role has a higher priority. Hence, the greatest priorities in each category remain. These roles are copied into the final list of roles. The empty slots in this list are filled in by NullRoles.

`discard_roles` can be replaced by code that copies over the list of roles into the final list directly, and mutual exclusion will effectively be turned off. This code is present, but commented out, just before `discard_roles` is called.

Ball-handling roles, that are mutually exclusive:

RolePoacher,

RoleLinearDribbler,

RoleDeflector, and

RoleJamShoot

RoleShootMovingBall

This is because it is not desirable for more than one friendly robot to compete for the ball at any one time.

Special roles that are mutually exclusive:

RoleUnstickFromWall, and

RoleRemoveWallBall

In execution order, `discard_roles` is followed by the function `sortRoles`, which sorts the roles by priority, and then `assignRoles` and `executeRoles`.

### Role Assignment

Roles are assigned using the following procedure:

Order roles based on their priorities, given the current game state. Role priorities are real values between 0 and 1, returned by the `EvalPriority` member function. Roles that are assigned priority 0 are *never* assigned to robots. Ordering of roles that have the same priority value is arbitrary.

Perform mutual exclusion. Refer to the Mutual Exclusion section for details on this. Mutual exclusion may also be bypassed without affecting other functions.

For each role, the feasibility of each not-yet-assigned robot executing the role is evaluated by the `EvalFeasibility` function. The highest-priority role is evaluated first. The robot with the highest feasibility value is assigned the role, and is taken out of the pool of available robots. Robots with a feasibility of 0 will never be assigned the role. If all robots have a feasibility of 0, the role is not assigned to any robot, and the next highest priority role is evaluated.

Step 2 repeats for each role, in decreasing priority order, until either the pool of available robots or instantiated roles is exhausted. If the list of roles is exhausted before the list of robots, the remaining unassigned robots are given stop commands (0 wheel velocities and kicker off.)

As the role assignment proceeds, the `roleAssignsCopy[]` array is updated. `EvalFeasibility` functions can use the information in `roleUsed[]` to implement mutual exclusion between roles.

An example of robot-role assignments in a given game state is illustrated in Appendix B.

### Role Execution

Once role assignment is complete, roles are executed for each robot. The high-level AI calls the `DoRole` function for each role. `DoRole` executes in two stages:

The `ExecuteRole` function returns a target state for the robot, consisting of a target location, target orientation, time to target, velocity at target, and kicker activation bit.

The target state is passed to a trajectory controller, which converts this target state into a robot command for the next cycle.

The polynomial trajectory control (see below) generates a set of wheel velocities for several cycles in the future. For this trajectory controller, wheel velocities are best generated every 5 or so cycles, and they are therefore buffered within the role. The goalie trajectory controllers only generate wheel velocities for the next cycle. Refer to the following section for details on individual role execution and the types of trajectory controllers.

## Roles

Each role extends the base role class and is implemented in a source file with the same name as the role. The following steps are necessary to implement a role.

Create a class derived from the base role class.

Override the virtual `EvalPriority` function to return a priority coordinated with the roles it will interact with. Priorities are normalized between 0 and 1. See the previous section for details.

Override the virtual `EvalFeasibility` function to evaluate the feasibility of each robot executing the role. Note that if all robots have 0 feasibility, the role will not be assigned.

Override the virtual `ExecuteRole` function to return a desired target state given a current game state.

Optionally, override the `DoRole` function if the standard trajectory controller should not be used.

If there are any new member variables that should be reset, override the constructor, destructor, and the `Reset` functions. Override the `category` value from the default category, in the role's constructor.

Override the `ReturnRoleName` function.

Add a new constant to the `e_roles` enumeration in `ai_core.cpp`.

Include the new role header file at the head of `ai_core.cpp`.

Add a new line instantiating the new role in `AllocateRoles` in `ai_core.cpp`.

Add the cpp and h files into the project.

Recompile the project.

Refer to the source code for further details.

### RoleMidfielder

The job of the midfielder role is to keep a robot near the center of the field. This is especially important when another robot shoots the ball to score, but fails and the ball bounces back near the midfielder. The midfielder will then be in an ideal situation to take control of the ball and attack.

The Midfielder role has a constant medium priority. This priority is lower than a striver role so that it may switch to a striver when appropriate.

The Midfielder robot stays close to the middle line, known as the midfielder line. There are two possible midfielder lines, slightly offset from each other. The role chooses the line that is least obstructed by other robots. This allows the midfielder to avoid needless collisions.

The Midfielder robot stays on either the right side or the left side of the field depending on which side the ball is.

#### RoleCornerBlock

This role is a defensive role. It attempts to block and add clutter to our corner when the ball is there. This prevents the other team from easily getting to the ball and hitting it in from the corner.

It has a high priority only when the ball is near the corner. Otherwise, its priority is 0.

The CornerBlock robot simply moves to cover the corner. Since this robot is not allowed in our defense zone, it stays just outside. It takes extra care to make sure that it doesn't approach the corner in a way that might hit the ball into our own goal.

#### RoleFormation

This role is activated only in a kick-off. It checks the gps state variable to see which team is doing the kick-off, and directs the individual robots into their starting formation.

#### RoleJamShoot

This role is an effective offensive role. It is rarely activated but has a high success rate when it is activated. This role checks to see if there is a robot that is in a very opportune position of scoring. Specifically, it checks to see if there is a robot that is close to and facing the goal. If there is, it then checks to see if the ball is directly in front, and if there are no obstacles directly in front. If all these checks pass, it then directly sends a very high wheel velocity to the robot and activates the kicker for a fixed number of cycles to shoot the ball into the goal.

#### RoleShootMovingBall

This role is an offensive role and attempts to shoot a ball. The name "Moving Ball" is a bit of a misnomer in that it also shoots stopped balls, but the name came out because it originated from a role that handled only a ball that is stopped.

This Role returns a constant high priority only when conditions are ideal for shooting, that is, when the ball is in zone in front of the enemy goal. Other offensive roles such as dribbler or deflector adjust their priority to be higher than shooter when the situation calls for it.

This role chooses only robots that are behind the ball.

It works as follows. If the robot is far away, it gives a target point that is behind the ball by about 20 cm. When the robot is close, it then directly controls the wheel speeds to turn the robot so that it hits the ball facing the correct target. This role currently works excellently in simulation but not so in the real system. This is probably due to the latency in the vision data.

#### RoleUnstickFromWall

This role checks to see if any robot appears stuck to the wall for a certain amount of time. Being stuck is defined as a robot that is very close to the wall, is facing the wall +- 30 degrees, and has been so for about a second.

If a robot is defined as stuck, it then returns a high priority and selects the stuck robot. It then manually controls the robot wheel speeds to reverse and spin it out of the wall.

Note that it does NOT unstick a defender robot, as this is the only robot that might be required to stay close to the wall for a long amount of time.

#### RoleRemoveWallBall

We often find the ball close to the wall where it is hard to get to. An ideal strategy in this case would be to try to offensively move the ball without actually going behind it. This is what this role does.

It returns a high priority when the ball is close to the wall. It first heads towards the ball. When the ball is close, the robot spins quickly (i.e. controls the wheel speeds directly) in the correct direction to bang the ball along the wall towards the opponent side.

#### RoleScoreCornerBall

This role is very similar to `RoleRemoveWallBall`, but handles the case where the ball is on the opponent wall. It returns a high priority when the ball is on the opponent wall. It sends the robot to the ball, and then spins the robot quickly to bang the wall to the goal. Since this ball is coming from the side, it is very effective against certain teams that don't defend well against that.

#### RoleStriver

This role attempt to place the robot behind the ball so that it may then dribble, shoot or deflect the ball. It contains a few heuristics to try to improve performance.

The role returns a constant high priority (but can be superceded by the other offensive roles when necessary) when the ball is not close to a wall.

The striver makes sure that if the striver robot is in front of the ball, it doesn't bang the ball backwards in trying to go behind it. Instead, it sends such a robot around the ball.

If the robot is far away, it sets a target to a point behind the line from the ball to the goal. As it gets close, the target gets closer and closer to the ball. This allows the striver to angle itself strategically behind the ball and facing the target by the time it gets close.

The robot also slows down as it gets close to the ball so that it doesn't knock it in the wrong direction.

#### RoleDefender

The defender role uses almost the same logic as the goalie role except that the defending line is near the quarter line, and it defends the entire field width as opposed to simply the goal width. Refer to the Goalie Role below.

#### Goalie Role

The goalie AI is not placed inside the role architecture, but instead directly controls robot number 0. This is because it was created before the role architecture was put into place, and we saw no reason to move the goalie AI inside the role architecture, since it never switches with other robots.

The goalie AI works as follows.  It moves the goalie laterally along the goal-line with half the body inside the goal and half the body outside the goal.  This is optimal to protect the goal from both frontal attacks as well as side attacks.

When the ball is very far away, it checks to see if the goalie is on the goal line.  If it is very far from the goal line, it runs the regular trajectory controller to bring the goalie robot onto the goal line.

When the ball is closer but still more than a quarter field away, it again checks to see if it is on the goal line.  If not, it attempts to wiggle itself back into the goal, and it does this while blocking the correct goal position and without changing direction too much.

When the ball is closer than a quarter field away, it attempts to cover the goal in the most optimal way.  That is, when the ball is fast, it covers the direct path of the ball, but when the ball is slow, it covers the y position of the ball.

In general, this role directly controls the wheel speeds, except when using the trajectory controller as described above.  It controls the wheel speeds with two variables: forward velocity v and differential velocity diff.  v is determined by how fast the goalie should move, and diff is determined by how much the goalie needs to turn to align itself with the goal line.

#### RoleLinearDribbler

This role attempts to roll the ball in a straight line towards the goal. Activation criteria for this role are very strict. If there is a straight line that intersects the robot, ball and the opponent's goal and the ball is just in front of the robot, then this role is activated.

The wheel velocities of the robot are controlled directly. They are 'ramped up' linearly to allow smooth acceleration to the target velocity. The target velocity itself is just slightly greater than the velocity of the ball to push the ball in the desired direction. The values of the target velocity are obtained through actual trials. The target location of the robot, is actually the ball's current location, since it must be moving in as linear a path as possible anyway.

The purpose of the role is to propel the ball towards the enemy goal while keeping the ball under control. Bypassing trajectory control also bypasses collision avoidance. This is acceptable because the robot must move in a straight line to keep control of the ball.

The role's priority is set to be just greater than that of `RoleStriver` when the ball is close enough. However, if the path to the goal is obstructed it is below `RoleDeflector's` priority. `RoleJamShoot` will have the highest priority of all if the ball is in the correct spot and near the goal.

#### RoleDeflector

The deflector role attempts to deflect the ball into enemy's zone. This role is most appropriate when the ball needs to be cleared from our own half to the enemy's half, without losing possession.

This role is activated in defensive and midfield positions. Also, the ball has to be reasonably close to the robot we wish to label as the 'passer'. If these are not satisfied, the priority drops to zero.

Within these restriction the role has two priorities. The higher priority is active when there is a friendly robot upfield, with a clear path to it. In this case, the role also sets the upfield robot as the 'receiver' and the robot with the ball as the passer. If there is no such robot upfield, the role sets its priority to an intermediate level and targets the center of the goal.

The actual passing algorithm is as follows. We draw a line from the passer's current position to the target, say L. We then calculate when the ball will intersect L. If the ball's trajectory is such that it will intersect L out-of-bounds, or not at all, the robot just heads for the ball, hoping to improve it's chances for deflecting the ball in the future. If the ball is heading in the correct direction, we set:

the final position of the robot to the extrapolated position of the ball intersecting L,

the final orientation of the robot such that it is along L,

the final velocity to the maximum velocity the robot can achieve in that time, and

the target time to the number of cycles we expect the ball to take to get to the intersection point.

The lower the final velocity of the robot, the more the ball 'skids' across its surface. Since we wish the ball to be deflected to a trajectory as close to the final trajectory of the robot as possible, we increase the robot's final speed as much possible.

Accounting for noise in the motion of the ball, we cannot hope to reliably orient the robot at it's final point such that the ball is deflected in a desired direction. Indeed, at high speeds and sharp angles, the ball may pass by the robot altogether. However, in most cases, especially when the ball is near, the angle of deflection is within 30 degrees. This, in addition to the fact that the path to the opponent robot is clear, leads to fairly reliable role behaviour.

### RolePoacher

The poacher's role is to wait near the enemy goal for a stray ball, and then attempt to score from that situation. The poacher, in order to be effective must satisfy two criteria:

It should not wait in a region where it is unlikely to receive the ball.

It should not obstruct friendly robots from a shot at the goal.

There is no situation that forces the poacher to take on a priority of zero. This is because we wish to play an attacking game. For a defensive approach, the poacher's priority can be set to drop to zero if the ball is in our defensive zone. We set the poacher's priority to 0.6 in the best case scenario, which is enough to take over any 'unoccupied' robots.

The poacher positions itself in the side of the goal with fewer enemy robots. Its primary region is just off to the side from the front of the goal, to leave a clear path for any advancing forwards. It waits in a region in front of the left or right goalpost depending upon the concentration of enemy robots. Also, the exact region is determined within a random radius to cover the maximum area. The radius is currently fixed at 15 cm, but ideally it should be dynamically adjusted depending upon the enemy robot's strategy and robot characteristics.

As can be seen, we only approximate requirements 1. and 2. We randomize our waiting region to maximize our chances of receiving the ball in a dynamic situation, and we stay to the side of the goal at least 25 cm away from it to allow robots with the ball a clearer shot if they wish to attempt one.

### 9.3.4 Trajectory Control

There is one built in trajectory control available for converting target states generated by ExecuteRole() into robot commands. It is a polynomial-based trajectory generation scheme, which generates a planned trajectory and a set of wheel velocities for the entire trajectory at every evaluation.

The target point is specified by the following values: target location, robot orientation at the target, time to reach target, and velocity at target.

By default, the standard trajectory controller is used by `DoRole`. Override the `DoRole` function in any role that should use a different trajectory controller, i.e. for controlling wheel speeds directly.

### Polynomial Trajectory Generation

The polynomial-based trajectory generation scheme generates a polynomial-based path to the specified target point. This polynomial path is then discretized and converted into wheel velocity pairs along the path, constrained by wheel acceleration limits and target state values. Note that the polynomial trajectory generation internally uses units of centimeters and cycles.

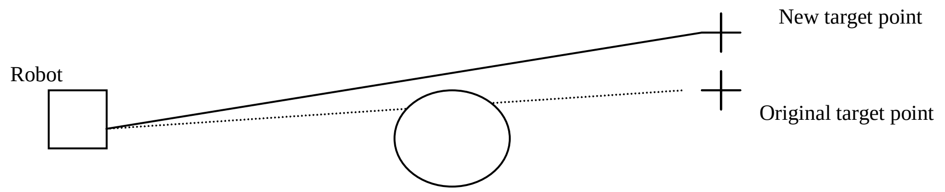
Although the polynomial trajectory control generates good paths, particularly with respect to target orientation, some conceptual problems prevent it from being very robust. These problems will be examined over summer. The polynomial trajectory generation contains obstacle avoidance and basic wall avoidance.

Refer to the appendix at the end of this section for technical details of the polynomial trajectory generation.

The files implementing the polynomial trajectory control are `trajcontrol.cpp`, `tc_pathgen.cpp`, `tc_polytraj.cpp`, `tc_wvgen.cpp`.

#### Obstacle Avoidance

The obstacle avoidance algorithm works as follows. The algorithm assumes that the robot will traverse a straight line from source to target. If this line is blocked by an obstacle, the trajectory generator will look for a new target point to the left or to the right of the obstacle, depending on which results in less deviation. If this new line from source to new target point is then blocked by another obstacle that is closer than the first obstacle, the trajectory generator will again look for a new target point to the left or to the right of both obstacles, and choose one which results in less deviation. This process continues until a target point is found that is not blocked by an obstacle. The robot will then follow this new target until a direct path to the original target is found.



Obstacle

#### Wall Avoidance

The wall avoidance algorithm simply checks the target point and ensures that it is inside the field. If the target point is outside the field, it projects the target point inside the field. Note that if the target point it close to the wall, the robots might hit the wall and get stuck. It is then up to the `UnstickFromWall` role to takeover and remove the robot from the wall. For further information consult Appendix A.

### 9.3.5 System Interfaces

#### System Initialization

Initialization of each system component, the field and game state data structures, robot PID acceleration gains, and the high-level AI is accomplished by the `InitSystem` function in `system_utils.cpp.`

#### System Timing

The AI subsystem main loop is not synchronized to the vision system. However, the RPC unit is only capable of transmitting approximately 95 packets per second. When the `send_data` function is called, the function call will hang until the previously uploaded packet has completed transmission. Therefore, the AI main loop is limited by the RPC packet rate.

We would like to run the AI at 60 Hz. This helps ensure that wheel speed commands are sent uniformly to the robots as is assumed by the trajectory controller. Without any pause in the AI loop, it runs at 95 Hz (RPC limit). We pause the loop at the end of each cycle to achieve the 60 Hz rate. Because the AI doesn't run at exactly the same speed every time due to changes in the AI or computer system, we've implemented a self-calibrating system that automatically changes the pause time so that the cycle rate is 60 Hz.

Refer to the following subsections and the EE documentation for further detail.

#### Vision Data

The AI subsystem receives data from the vision system over a UDP connection as requested. The AI subsystem therefore runs asynchronously from the vision system. If necessary, field state information is interpolated by the vision system as necessary. Refer to the vision system documentation for further detail. Vision system networking code is contained in `networkFunctions.c`.

The vision data is stored in the network field state data structures. This data is converted to the AI subsystem data structures. frobots are considered valid (found by the vision system) or invalid based on the bit field `fNumber`. `oNumber` specifies the number of valid opponent robots. Note that this means that `orobot` identities are not necessarily consistent from frame to frame.

Values are copied with the following modifications:

Vision gives location in meters, but the AI works in cm. The vision data is converted appropriately.

The orientation marker for all the robots is set perpendicular to the axis of motion, and is converted to be in line with the axis of motion when orientation information is copied from vision.

Orientation sines and cosines are calculated accordingly.

Robot wheel velocities are fed back in an open loop. Rather than computing robot velocities from vision-fed velocity data, the previously sent wheel velocities are fed back directly to the appropriate data structure fields.

The final copied and converted values are stored in the global fs and gps variables.

### Wireless Communications (RPC)

The wireless communications subsystem sends the final computed robot commands to the robots. These commands are generated by the roles and are stored in the fs variable.

The wheel velocities are computed by the AI in units of cm per cycle, but must be converted to meters per second and is done through a simple multiplication prior to transmission. The speed is also multiplied by a fudge factor.

The wireless communications is run synchronously with the AI subsystem.

Refer to the EE documentation for notes on packet encoding, etc.

The wireless communications functions are implemented in rf_control.cpp.

### User Interface

The user interface of the AI subsystem uses the standard console i/o functions.

User input is handled directly by the main loop of the Brazil Master Program.

When the Brazil Master Program is first run, it first asks if the user wishes to run the normal AI or run system tests.

i) If normal AI is chosen, the user is presented with a menu of available AI modes, which are:

1) Normal

2) Kickoff - us

3) Kickoff - them

4) AI `Go-Point` role test

The main AI loop will then run. When the AI is running, it will present a snapshot of the vision data it receives once every second to indicate normal status.

The user can pause the main AI loop with a keypress. It will then present the above available AI modes.

ii) For System Tests, the user can send robot number 0 to any coordinate on the field. This is useful to ensure that the system is working.

The user interface only uses the `printf` and `scanf` functions. Such console I/O functions, when excessively used, can cause a slowdown in the overall system.

Refer to the User's Guide for further details.

## 9.4 Player

### 9.4.1 Design Philosophy

The design of the mechanical system was based upon the top-down systems engineering approach. Once the overall team and system goals were established, the mechanics were designed to support those team goals. To fulfill those requirements the mechanical design team opted for modularity and part interchangeability over custom, complicated, single-piece parts. For example, the robot structure is comprised of more than a dozen parts screwed and bolted together instead of a single custom-machined part. However, many of the parts are identical and are easily interchangeable both between the robots and within a single robot. The parts themselves are easily machined using standard machining operations such as drilling, milling and lathing.

In a nutshell, our design philosophy is to be simple, efficient and robust. A robust design was very important considering that the mechanical team members had no prior experience in robot soccer design. The requirements on performance and reliability were very ambitious (as compared to previous years) so large factors of safety were incorporated into the design. Optimization was performed on various levels such as weight and strength to acceleration performance. Efficiency was emphasized especially in the design moving subassemblies.

The robot was also designed to be modular. Subassemblies could be manufactured concurrently and tested for mating and operation. This lowered development and manufacturing time of the entire design. Also many parts are interchangeable or close to identical. The players and goalie share many identical parts and upon failure of a player or the goalie, either one can essentially replace the other. Also, almost all of the fasteners are of the same size (4-40 screws) of different lengths to ease maintenance and stocking of spare parts.

Design for Manufacturing

The design of each part in the robots was done with several key issues in mind:

Each part must be moderately simple so they can be fabricated within acceptable tolerance.

Each part that needs to be fabricated must not be too complicated given the experience and skills of the people responsible for machining and assembling them. The limited processes available to the team at Emerson Lab and the limited budget and time constraints or part sourcing, manufacturing and finishing must be accommodated. Questions that were often asked during the physical design process were:

Can this be machined in a machining lab such as Emerson?

Would I be willing to spend the time and machine the part?

If not, can we make it simpler?

The reasoning behind this decision is that by using many similar parts several benefits can be achieved:

A large amount of resources is not frozen in a complex, custom mold.

Robots can be disassembled and fewer parts need to be replaced if a robot is damaged.

There are tradeoffs that need to be considered when approaching a mechanical design from this prospective. A well thought out part is often maximized for functionality. However each time a new function is assigned to a part, the complexity of its design also increases. A balance was struck in our reasoning so that the number of parts in the entire robot was minimized while not producing overly complex parts that were beyond the capabilities of the team to manufacture.

By lowering the part count of the robot assembly, a large pool of spare parts can be easily manufactured and readied for substitution into practically any robot. The parts are not robot specific and approximately 90% of the parts that constitute the goalie are identical to a player.

In keeping with the modular design principle, the mechanical design was divided into subassemblies. These subassemblies are the power train, kicker/kicking plate surface, battery pack assembly, and chassis/side control surfaces.

### 9.4.2 Power train

The components that make up the power train assembly include:

motor/encoder

pinion

spur gear

hub

wheel

tires

wheel bearings

mounting hardware

The robot design started out with several features that were thought to be very helpful in the effort to have a top level robot team. These basic features were discussed among all the subdivisions in the team to gather input about them and to see how the components would interact with each other. After laying down the desired features for the robots, some target numbers such as maximum speed, acceleration, kick speed, were decided upon. The speed of the robot was decided with inputs from Jin Woo as to what was desirable based on his experience in the Mirosot competition. A bi-wheeled arrangement was chosen for its high maneuverability characteristics. The basic equations of motion were used to determine the required acceleration to obtain the desired maximum speed in the specified distance (1/4 of the field).

The procedure was as follows:

Knowns: Want:

Coefficient of friction
Mass of vehicle
min. top speed
min. acceleration
wheel radius
axle radius
gear box efficiency
engine speed(rpm)

gear ratio
torque at motor

The method for selecting motor components is as follows:

Given desired top speed, minimum acceleration and physical assumptions of the table surface and robotic players stated above a motor was chosen. The desired operating range of the motor was selected as 30% of the stall torque and no more than 90% of its no-load speed. This range was recommended by the manufacturers as giving optimum motor efficiency, lowest electrical noise, acceptable temperature range, and maximum motor life. The motor was chosen such that the desired initial power requirements were within this range. This was a trial and error process until the manufacturer's specifications matched our requirements.

$$MinimumAcceleration = \frac{DesiredTopSpeed}{2 * Dis\tan ceForTopSpeed}$$

$$= \frac{V_{\max}}{2 * \Delta X}$$

$$MaximumFrictionalForce = FrictionCoeficient * MassOfRobot * GravitationalAcceleration$$

$$= \mu * m_{robot} * g$$

$$Force\,\mathrm{Re}\,quiredAtWheels = MinimumAcceleration * MassOfRobot$$

$$= a_{\min} * m_{robot}$$

$$Torque\,\mathrm{Re}\,quiredAtAxle = \frac{Force\,\mathrm{Re}\,quiredAtWheels * WheelRadius}{2}$$

$$= \frac{F_{wheel} * R_{wheel}}{2}$$

$$GearRatio\,\mathrm{Re}\,quirement = \frac{Torque\,\mathrm{Re}\,quiredAtAxle}{30\%StallTorque * GearEfficiency}$$

$$= \frac{\mathrm{T}_{axle}}{0.30 * \mathrm{T}_{stall} * \eta_{gear}}$$

$$MaximumArmatureCurrent = \frac{30\%StallTorque}{TorqueCons\tan t}$$

$$= \frac{0.30 * \mathrm{T}_{stall}}{K_{torque}}$$

First, we want to find the gear ratio that would provide the desired top speed. The top speed is directly related to the wheel radius, desired operating engine speed, and gear ratio. From there, we obtain the torque that is required of the motor given the min. acceleration we want for our robot. This method is useful for both the players and goalie if the required torque and gearbox we need are commercially available and meet the electrical power constraints (initial calculations show that this will be the case).

***<u>Procedure:</u>***

$$GearRatio = \frac{2 * \pi * WheelRadius * EngineSpeed}{60 * RobotVelocity}$$

$$TorquePerMotor = \frac{Acceleration * MassOfRobot * AxleRadius}{2 * GearboxEfficiency * GearRatio}$$

Having the basic constraints for the motion of the robots we set up a spreadsheet to help us determine the motors that met our requirements. The design started from the assumptions stated above combined with the assumed mass of the robot, friction coefficient, wheel radius, and motor. With these assumptions a required gear ratio was obtained. An available gearhead was selected to meet the required gear ratio. The actual acceleration, output torque, and maximum velocity obtained were rechecked to meet required specifications. From the characteristics of the chosen motor maximum current draw and continuous power was calculated. This data was given to the electrical design team.

The results of these calculations are shown in the following table:

| **ME Characteristics** | |
|---|---|
| Maker | **Maxon** |
| series/model | |
| price/piece ($) | 137.35 |
| Desired top speed (m/s) | 2 |
| Distance to achieve top speed (m) | 0.68 |
| Friction coefficient | 0.6 |
| stall torque (oz.in) | 12.62218 |
| no load speed(rpm) | 5810 |
| min. acceleration (m/s^2) | 2.941176 |
| time to reach min velocity (s) | 0.68 |
| mass of robot (kg) | 1.75 |
| wheel radius (m) | 0.038 |
| max frictional force (N) | 10.29 |
| force required at wheels (N) | 5.147059 |
| torque required per axle (Nm) | 0.097794 |
| torque required at axle (oz.in) | 13.85382 |
| 20% of stall torque (oz.in) | 2.524437 |
| Predicted gear head efficiency | 0.7 |
| min. gear ratio required | 7.839836 |
| 90% of no load speed(rpm) | 5229 |
| gear head available | 7.4 |
| custom gear reduction | 1 |
| max velocity obtained (m/s) | 2.811895 |
| actual gearhead efficiency | 0.8 |
| actual ouput torque (oz.in) | 14.94467 |
| actual acceleration (m/s^2) | 3.174337 |
| | |
| **EE Characteristics** | |
| Torque constant (mNm/A) | 9.43 |
| Line voltage (V) | 6 |
| Max. Armature current (A) | 1.890651 |
| Power required (W) | 11.3439 |
| Torque available | 2.524437 |

Maxon was chosen from other motor manufacturers (such as MicroMo and Pittman) because of their excellent performance characteristics at a low price. We wanted encoder resolution at the wheels to be no more than 1 degree at the wheels. This number was obtained from the approximate backlash of the gear train. Therefore, there would be no need to have resolution greater than this. Available encoder options gave us a 0.72 degrees of resolution at the wheels which was the closest match.

The method for selecting gearhead components is as follows:

With our selected motor, a desired gearhead that would meet our speed and torque requirements had to be selected. The process started out by selecting components made by the motor manufacturer to ensure proper functioning. Our requirements called for gearhead ratios that were difficult to match with the available ones. The next best match would be selected for this purpose but this meant that our actual velocity and acceleration would change. In an effort to get a better match, a decision to custom build a gear reduction was made.

The advantages to a custom gearbox are smaller size, cheaper overall drive train, smaller weight, more interchangeability over an off the shelf gearbox. This decision gave us a lot more flexibility in the structural design of the robot, especially for the kicking mechanisms (see kicking section).

The materials for the pinion and the spur were chosen after doing tooth strength and surface strength calculations using various materials. There are two modes of failure that are attributed to gear teeth. The first is fatigue fracture, which occurs when oscillating bending stresses at the root of the tooth are present. The other is surface fatigue or pitting of the tooth due to material contact when gears mesh. Both failure modes were checked using the AGMA bending stress equations (AGMA Standard 2001-B88) with these assumptions:

The contact ratio is between 1 and 2 (required for a conservative estimate of an indeterminate problem)

There is no interference between the tip and the root of mating teeth

The teeth on the gears are not pointed

A positive amount of backlash exists

The roots of the gears are assumed to be standard

The frictional forces are negligible

Given the forces applied at the gears as determined by the weight of the robot and the accelerations it will experience, plastic gears (ABS) will be sufficient to withstand both fatigue and pitting with a reliability of 99.999% per gear set. However, in practice one of the gears in the pair should be made of harder material than the other so to reduce wear on both gears. The spur will be made of ABS and the pinion will be made of aluminum. The materials were chosen in this combination based on availability and low cost.

With this setup, the gear ratio of the robot can be tuned very easily for maximum system performance.

The wheel and spur will rotate with the same angular speed and will both be mounted on an aluminum hub. Aluminum was chosen for its light weight and high strength. The diameter will be determined by the inner diameter of the ball bearings which the wheel and the spur will rotate on. The factor of safety obtained given the loads that the hub will bear (determined by the weight of the robot and the desired acceleration) was calculated to be 61 using max shear stress theory. Using the distortion energy theory tends to give a higher estimate because the max shear stress method is a more conservative estimate of equivalent stress experienced by the element. The hub will be rigidly mounted to the motor mounting plate using epoxy.

The method for selecting wheel components is as follows:

The wheel radius was initially set to a reasonable size based on previous designs so that initial power calculations could be performed. The size was then refined to optimize motor/gearhead combinations. The optimum motor/gearhead combination is the smallest motor with the lowest gear ratio for highest drive train power efficiency. Two options are available for the wheels: custom machined or off the shelf. Custom machined wheels would require engineering and manufacturing time which added together cost more than off the shelf. Off the shelf may come in the form of R/C racing wheels. These wheels are designed for lightweight, high strength, and low moment of inertia. The cost is relatively cheap, approximately five dollars per pair. These wheels come in a variety of materials, shapes, and sizes that will fit our needs. At this point R/C wheels will be used.

To reduce friction between the wheels and the hub, the wheels will ride on stainless steel ball bearings as will the spur gear. The wheels and spur gear will be locked into place between locators on the hub and a screw with a washer at the end of the hub.

The tire material chosen is material found in mousepads made by Dell. Experiments with different materials showed that rubber compounds used in mousepads provides superior traction in “tug of war” tests. The material is readily available and the cost are very cheap if not free. The tire material consists of two layers of different mouse pad material. The first layer is the stiffer of the two which provides support to the traction layer. The traction layer contacts the playing surface and is more pliant. This tendency to bend under stress provides excellent traction characteristics in accelerations. The tire material is mounted onto the rims using low viscosity super glue.

Wheel bearings will be sealed stainless steel ball bearings that have been found to be compatible with the mounting holes of the R/C wheels. Bearings have significant performance gains at high speeds and accelerations over bushings. Unlike bushings, bearings do not require external lubrication and are low maintenance. The performance gains justify their cost ($8.00/robot).

Standard mounting hardware will be used to mount the motor to the motor plate as well as the pinion set screw and gear train.

### 9.4.3 Kicker and Kicking plate surface

#### Kicker Design Goals

There were several goals we wished to accomplish in designing the kicking mechanism for the robotic field players. From viewing the videotapes of past RoboCup competitions, it was decided that the ability to kick the ball at 2 meters per second relative to the robot’s velocity would give Cornell’s team a significant advantage over its opponents. In light of the fact that the robots will not be able to trap the ball very well, it was decided a powerful kicking mechanism should only be used for shots on goal. For simplicity, the robots will only be able to kick along one axis, forward and backward. With that in mind, our attention turned to making a kicker that would work well with the other components in the robot. This was to be accomplished in several ways:

The kicker should be energy efficient. Since the kicker will be used at times when the robot is likely to be performing energy draining maneuvers, it could not sap a lot of power from the batteries when in used.

The kicker should be simple mechanically. The kicker should not be made of many parts that can break, and it should be lightweight.

The kicker must be "intelligent." It would be preferable if the kicker did not rely very much on the on-board microcontroller for operation. A clockwork design that could maintain itself mechanically would be preferable, though it could interfere with objective number 2

The kicker must be able to fit in the robot, so small size and tolerance to misalignment is essential.

The kicker should be ambidextrous. We desired a kicker that could kick from the front and back of the robot. Ideally, the kicker should be able to use most of the same hardware to kick in both directions.

Several ideas were discussed and researched for the kicker, though they were eventually dismissed for not having the desired qualities. The inferior designs were:

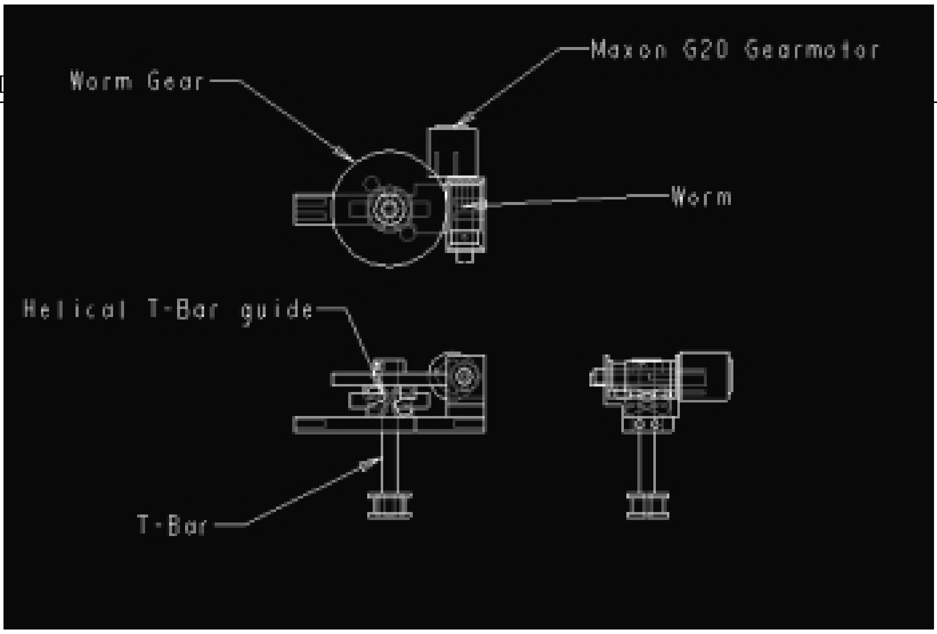
### ***<u>Solenoids</u>***

Solenoids were dismissed because powerful ones are fairly heavy, and they rob power from the drive motors when activated. On the other hand, they are very simple for the motor controller to operate and are not prone to breaking.

### ***<u>Servomotors</u>***

Servomotors were dismissed because they could not provide enough power to propel the ball at 2 m/s, and they would also rob the drive motors of power during kicking.

### ***<u>Spring Loaded Kicker (Best Design)</u>***

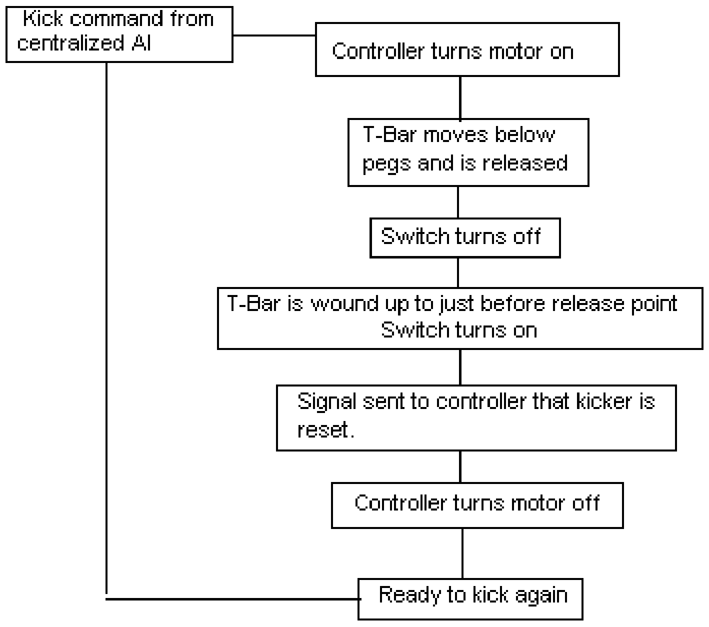


The spring loaded kicking mechanism was chosen for our design. This design has outstanding energy efficiency because it can wind up during the rest periods a robot will encounter between kicks. Our design is also "intelligent", in that it only requires one switch to describe its state to the microcontroller. This design is also fairly small, using worm gears and a tiny gearmotor; more importantly, its linkage system makes it tolerant of misalignments, so it can be packaged easily in the robot.

### Kicker Design

The kicking force is imparted to the ball through two hinged plates, or flaps on the front and back of the robot. These plates are hinged on top, near the electronic boards on the robot, and compress against springs on the lower chassis. Each is pulled inward by fishing line that is wrapped around a spool located in the center of the robot, above the batteries. This spool is located at the bottom of a steel T-shaped bar. The arms at the top of the T-shaped bar ride two helical grooves machined into a cylindrical piece of delrin-teflon composite. In this arrangement, as the T-bar, and thus the spool, swivel, they move up and down. A worm gear is mounted axially above the T-bar/ helical pivot system. Two pegs protrude from the worm gear that can engage and disengage with the arms of the T. As the worm gear is turned, the pegs will engage the T-bar and will be able to make it turn in its helical slot over a certain range. Eventually, the T-bar will be guided downwards in the helical slot far enough so the pegs on the worm gear can no longer touch it. At this point, the T-bar will be released, allowing the kicking plates to spring outwards. The worm that drives the worm gear will be mounted to a Maxon G20 Gear Motor, which adds further gear reduction. The worm will be supported by thrust bearings to prevent the high axial stresses from ruining the motor bearings.

This design interfaces well with the robot as can be seen in the following state diagram:


Kick command from centralized AI
Controller turns motor on
T-Bar moves below pegs and is released
Switch turns off
T-Bar is wound up to just before release point
Switch turns on
Signal sent to controller that kicker is reset.
Controller turns motor off
Ready to kick again

The mechanical parts of the kicker are "smart" enough to reset themselves after a kick without relying on the microcontroller. The microcontroller simply starts the motor when told to kick by the centralized AI computer. The kicker is released, but the motor is not shut off. The pegs on the worm gear engage with the released T-bar and begin to draw in the kicking plates. When the springs are fully compressed (the kicking plates are drawn in all the way) a Hall-effect switch is thrown. This signal indicates to the microcontroller that the kicker has reset and is ready to be activated again. The microcontroller shuts off the motor until it is time to kick. The kicker does not unwind because of the low worm lead angle (approximately 2 degrees) prevents back driving.

Initial analysis of the energy that the kicker would require revealed that the springs would need to be fairly strong. We also wanted the motor to be fairly small, and slowness was not an issue. Immediately we realized a large gear reduction would be ideal for this design. Worm gears are the best way to obtain high gear reduction ratios, and have the added benefit of being a one-way drive. The motor cannot be turned by activating the kicking plates externally. This was another benefit we wished to exploit. The difficulty lay in finding the proper way to engage and disengage the motor from the spring loaded plates. The helical path traveled by the T-Bar allows the motor to simply and easily be engaged and disengaged. Best of all, the process is completely automatic.

## Kicker Design Analysis

### Spring Selection

The springs were selected using an energy balance, namely, Energy stored in spring = Energy of ball + energy of plate

The initial goal for the kicker was $v = 2$ m/s. The energy for each of these components is:

Energy in spring = .5 K x^2

Energy in ball = .5 m v^2 + .5 I w^2

Energy in plate = .5 I w^2

The ball is a regular golf ball, with:

mass = .04593 Kg

diameter = .04267 m

velocity = 2 m/s

The inertia of the ball is

I = (2/5) M r^2 = 8.363e-6 Kg-m^2

w = v/r

so Eball = .128 J when it is rolling at 2 m/s

The energy of the plate is:

E plate = .5 I w^2

I for a plate like this is I = mh^2 / 3. If the plate is assumed to be approximately 1 g / cm^3, the moment of inertia is:

I = 5.8e-5

The bottom of the plate is moving at 2 m/s. If the plate is .10 m tall, the angular velocity of the plate will be

w = v/r = 21 rad/sec

The energy in the plate is:

.5 I w^2 = .02322 J

Therefore, the total energy stored in the springs must be .14 J per side.

If the displacement of each kicking plate is to be 1cm (remember, the size limitation prevents the plates from coming out too far) then the spring constant must be 1.4 N/mm or 8.096 lbs/inch (actually, multiply these values by 2. We are using two springs per side). This analysis was simplified using an Excel spreadsheet.

### Motor Selection

The motor selection process was a lengthy process that required enumerating all of the friction losses in the system. This was critical since the torque losses due to the worm gear was so high. In the end, more energy is wasted in friction than in kicking the ball. The friction was modeled as coulomb friction, which increased as the kicker wound. The increase in friction was easy to model since the springs are assumed to be linear, so all the torques, and thus forces on the mechanism increase linearly with angle or displacement. The calculations were performed on a spreadsheet as follows:

energy required: =.14 J per side

angle of rotation of spool = 120deg = 2.09 radians

Spring forces transmitted to motor:

Needed spool radius for this spring constant = .00477 m

Torque on T bar due to the springs = spring constant * spool radius^2 * theta (in radians)
=.1290* theta

Torque after the 120:1 worm gear reduction = .001075 * theta

Torque after the 55:1 gear reduction = 1.954e-5 * theta

Friction in helix transmitted to motor:

Force on helix at contact point = 27.16 * theta
(This was calculated by knowing the torque in the shaft and the approximate distance it contacts the helix)

Frictional force from helix = 27.16 * theta * .1 *sin 20
This assumes the coefficient of friction is .1 and uses the fact that the helix is at 20 degrees. = .93 * theta

Torque from helix on T bar = .00427 * theta

| | |
|---|---|
| Torque after 120:1 worm gear reduction | = 3.5611e-5 * theta |
| Torque after 55:1 gear reduction | = 6.475e-7 * theta |

Friction in the gearing

| | |
|---|---|
| Gear tooth contact force | = 5.431 * theta |
| frictional force | = .5431 * theta |

This was calculated knowing the radius of the worm gear and the torque in the shaft.

| | |
|---|---|
| Torque on worm | = .00345 * theta |

This value is calculated by knowing the radius of the worm. These gears are only cut at about 1.7 degrees, so the angle of the gear tooth was neglected.

| | |
|---|---|
| Torque after motor 55:1 gear reduction | = 6.27e-5 * theta |

Similar forces are acting on the thrust bearing ( indeed, it is the surface against which the contact force on the worm reacts!)

| | |
|---|---|
| Torque due to Thrust Bearing | = .00345 * theta |
| Torque After 55:1 Gear Reduction | = 6.27e-5 * theta |

The totals are:

| | |
|---|---|
| Total of torques on the T bar | = .1335 * theta |
| Total of torques on motor shaft | = .008008 * theta |
| Total of torques on motor armature | = 1.45e-4 * theta |

The first issue is selecting a motor with the appropriate output torque. The total of the torques on the motor shaft will be .008008 * (Theta =2.094 radians) = .01677 N-m at maximum displacement. The Maxon GM20 gearmotor has a maximum rated continuous torque of .030 N-m, a safety factor of approximately 1.78

Maxon motor publishes speed / torque curves for its motors. This makes it possible to set up a nonhomogeneous differential equation to find out how long it will take the motor to wind since all of the terms in this equation are functions of theta or theta'. The equation is:

Torque applied to motor (theta) = Stall torque - torque/speed gradient * theta'

The values given on the spec sheet for the motor are:

stall torque = .117 N-m

speed/torque gradient = 6400 rpm / mNm

However, the stall torque is stated for the motor shaft, while the speed / torque gradient is given for the motor after its internal 55:1 gear reduction. Therefore, to make this equation refer to a consistent angle, everything was adjusted to match the internal armature revolutions.

Over the course of winding the kicker:

T bar spins 120deg = 2.09439 radians

worm spins 120 * T bar angle = 251.32 radians

armature spins 55.1 * worm revs = 13848 radians over one kick.

The stall torque at the armature is .117 / 55.1 = .002123 N-m

The speed/ torque gradient is still 6400 rpm / mNm. expressed in radians/sec / N-m, this is 670206 (rad/sec) / N-m.

In general, theta of the T shaft, which was used to calculate all of the friction torques, is related to the angle of the motor armature as:

theta_armature = theta_shaft * 120 * 55.1 = theta_shaft*6612

Therefore the differential equation giving the motion of the kicker is:

(1 / 670206)*theta_armature' + (1.45e-4 / 6612)*theta_armature = .002123 N-m

simplifying, we get:

theta_armature' = -.014698 * theta_armature + 1422.92

The solution to this ODE is:

theta_armature(t) = -96810* exp(-.014698 t) + 96810

Note that we are looking for an angle of approximately 14000 radians. This is around 12 seconds, which is acceptable by our standards.

Once the motor and springs are specified, the weakest component in the kicking system is the worm gear. The forces exerted on the worm gear are high, and the teeth are fairly small, since this worm gear is 64 pitch. Using Shigley and Mishke's formulae for bronze worm gear stress, we find:

$$Tg = \frac{\vartheta B d_g^2 F_g \cos(\psi)}{1.5N}$$

The allowable bending stress for bronze (sigma) is 48.2 Mpa. d is the pitch diameter of the gear, .047625 m, F is the face width of the gear, .00476 m, and psi is the lead angle of the worm, 1.78 degrees. N is the number of teeth on the gear.

With this data used in this formula, the allowable torque on the gear is 2.89 N-m, which is far above our expected torque of .279. The gears are by far the weakest component in this system. The aluminum braces are theoretically subject to zero stress because of the symmetry of the system. The T-bar will be made of .25 inch steel bar stock, which, in torsion has:

$$Shearstress = \frac{Tc}{J} = \frac{.279N - m * .003175m}{.5\pi .003175^4} = 5.5MPa$$

Steel has a yield stress of approximately 200 *Mega* Pascals, which means this T-bar is definitely strong enough. The arms of the T-bar can be analyzed for bending stress

$$stress = \frac{MY}{I} = \frac{.279\ N - m * .003175\ m}{.25\pi * .003175^4} = 110.9Mpa$$

this is well below the normal stress limit of approximately 400 Mpa for steel. All other parts of the kicker are subject to less stress, and should have even better safety factors.

### 9.4.4 Available Parts & Performance

Most of the parts for the kicker will be fabricated by hand in the machine shop. Some parts have been specified in catalogs. The Maxon GM20 gear motor is ideal for this application. The motor contains a 55.1:1 gear reduction inside its casing that creates very high torque for its small size. This motor is specified for operation at 30 mNm, with a stall torque of 117 mNm. As shown by the equations above, the torque required of the motor will be 16.7 mNm, which below the rated torque of the motor. Other motors from Maxon and MicroMo were investigated, but none could offer the low power consumption, small size, and high torque of this Maxon motor. In addition to these great credentials, the Maxon G20 is one of the cheapest motors

The Maxon G20 runs at 6 volts, but our electrical system is designed for 9 volts. The motor should still work with this system, especially since it will not be in continuous operation. The higher voltage will actually give us the opportunity to draw the plates in farther or have the kicker wind faster based on spool size.

The helical spiral will be made of teflon impregnated delrin composite. This material was chosen because it is lightweight, strong, easy to machine, and low friction.

The worm gears were chosen based on their small size and high gear reduction. They are 64 pitch, 120 tooth, single thread worm gears with matching worms from SDP-SI.

The T-bar will be made of steel. Even though aluminum would be strong enough, steel is required since it will be easy to weld. The T-bar will be fabricated as two parts that will be welded together. It is too difficult to make two round aluminum bars mate at a 90 degree angle.

The motor mount and braces will be machined from aluminum because of its light weight and easy machinability.

The springs have been selected from the SDP-SI catalog. Since no spring met our needs exactly, two springs will be used in series at each corner. The effective spring constant is 8.34 lbs / in. They are part no's, S78CSY-018024050 and S78CSY-024038150.

Using the parts specified above, the energy stored in the springs will be .14 J, this will be used to accelerate the plate and the ball. Solving for the ball speed, using the equation:

$$.14 = \frac{1}{2}mv^2 + \frac{1}{2}Iw^2 + \frac{1}{2}Iw^2$$

$$.14 =$$

$$.5 * .04593\ v^2 + .5(8.363e-6)(v/.0213)^2 + .5(1.169e-4)(v/.10033)^2$$

we find the kicker will be capable of kicking at v= 1.957 m/s.

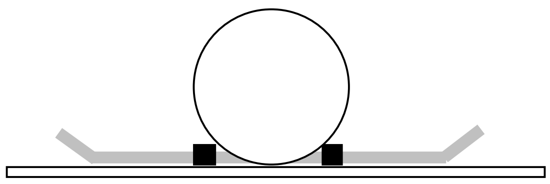
### 9.4.5 Ball control surface

The ball control surface has several objectives. It must enable the robot to intercept an incoming golf ball and successfully redirect it accurately towards another direction. In addition, the robot must also be able to stop the ball if necessary.

### Design considerations

The control surface must have contact with the ball at a minimum of three points. This will allow friction forces to stop the ball from rotating and translating sideways. It must also keep the ball within the length of the kicking plate, even though the robot is spinning. This is not too difficult to achieve if the robot is rotating while moving forward.

The control surface was designed so that the ball will be automatically centered in front of the player as the player moves forward. This is important so that the player can shoot the ball straight ahead after moving forward a short distance.



On the simulation, the protrusions (which have been assigned the term 'nubbies') were simulated with both solid, non-elastic, fixed objects and also flexible protrusions with elasticity. Although in theory, having a system of elastic protrusions mounted on springs is ideal, the performance penalty of having fixed nubbies is negligible. Furthermore, because the decision was made for the kicking plate to be made out of carbon fiber, fixed nubbies will be incorporated in the design, and they will be made out of 1/8" cubes of neoprene rubber to reduce wear and tear.

It was also later decided that the nubbies themselves would not suffice in keeping the ball on the kicking plate. Thus, an extra ball controlling surface (in gray) was added to the kicking plate.

### Design Process

Several designs were simulated in working model. The general design chosen consisted of a number of nubbies (square protrusions on the kicking plate made of rubber) which will hold the ball in place. The parameters considered were material selection, surface roughness, number, size and placement of the nubbies and end-guards to stop the ball from laterally sliding off the kicking plate.

The final design consists of two nubbies spaced 1 inch from each other. The size of the nubbies (1/8" x 1/8" x 1/8") is the maximum as can be allowed by robot size restrictions. The placement allows the golf ball to be touched at exactly three points.

In addition to the nubbies, a small horizontal strip bent at the ends is glued onto the kicking plate right below the nubbies for several reasons:

Tests show that having another contact point on the ball below the vertical center will keep the ball from spinning and stabilize it. Although we were hoping that it would make the ball spin backwards and make it stick onto the kicking plate, it does not exert enough force to achieve that result.

Having the edges bent on the edge of the horizontal strip keeps the ball from sliding off horizontally when the robot turns.

### Material Selection

The kicking plate is made out of carbon fiber. This is strong enough to withstand bending moments caused by the kicking mechanism, and absorbs vibrations well. Large black carbon fiber sheets were cut into size, and the delrin hinges were glued and screwed in place onto them.

The material required for the nubbies satisfy several requirements:

The material must be able to absorb shock upon impact so the ball does not bounce off the kicking plate.

High friction coefficient to transfer more lateral force to the ball to stop it from spinning.

Neoprene rubber was selected for the final design. 1/8" thick rubber sheets were cut into 1/8" cubes and were glued onto the kicking plates. Tests show that if the robot is accelerating even minimally, the ball will move towards and stay at the center.

### Future design ideas

The ball control surface can be greatly improved. The texture of the kicking surface can be made rougher by adding a layer of rough material or by cutting grooves into the kicking plate. The corners of the cplate should be rounded off so that the cplate can help keep the ball on the kicking plate. One design that has been tried already is putting a thin sheet of metal with protruding corners on the face of the plate to "cup" the ball.

### 9.4.6 Chassis

A modular approach was taken in the design of the chassis assembly components. Components were designed into specific subassemblies within the robot geometrical limitations. The lowest level subassembly is the main chassis itself which is the platform that all other mechanical components and electrical components will be attached. This platform is shared between all players including the goalie. This strategy produced the least number of spare parts between the player and goalie designs. The chassis is essentially made up of two main motor mounting plates and structural cross members that attach the two parallel plates. The cross members were positioned as far apart as possible to maximize its structural rigidity under linear and torsional loading. The material used was 6061 Al which provides excellent strength to weight ratios as compared to steel. Also this material has excellent machining properties which was considered important because the chassis components were to be custom machined. Aluminum also has a high heat conductivity which is needed to dissipate heat from the driving motors which are directly attached to the two parallel motor mounting plates on either side of the chassis.

Using this approach, the other robot subassemblies such as the kicker can be manufactured and tested separate from the chassis and other components. This allows for concurrent design, manufacturing and testing of major components.

Side plates were also added to the design to cover up the wheels. The material chosen was lexan or clear polycarbonate. The material is resistant to heat, fracture and is light and cheap and easy to manufacture. Also the material can be painted so that the color of the wheels will not interfere with the operation of computer vision. The sideplates serve two main purposes. First, they provide an additional control surface on the robot so that the ball can be deflected with precision. Second, the sideplates also serve as wheel protectors to protect the wheels from impact and rubbing along the sidewalls of the playing field.

### 9.4.7 Battery tray

The battery subassembly was designed in such a way as to place the batteries as close to the ground as possible and also be easy to replace. This has two important consequences. First, by placing the batteries as low as possible, the height of the center of mass is minimized. The lower center of mass the less weight is transferred during high accelerations and decelerations. Also having a low center of mass makes it very difficult for the robot to be tipped by other robots.

Having the battery in an accessible spot on the robot makes it very easy for the batteries to be changed in emergencies such as battery changeovers during timeouts and half times. The current design allows the battery pack to be changed by loosening two screws on the bottom of the robot. The tray is then slid back to the larger holes to clear the screw heads and battery is removed by unhooking the battery plugs. The reverse of the procedures is used to replace the battery pack. The battery changeover time has been tested to be approximately 30 seconds which would be sufficient to change the battery packs of all robots during a timeout if it is required.

## Section 10 Schedule

*Note: A copy of the Team Brazil's schedule and a copy of the shared schedule with Team Italy can be found in the Appendix.*

The schedule is divided into several components: AI, Vision, Environment, Mechanical Design, Electrical Design and Research and Development. After the initial brainstorming phase, enough concepts were presented in order to determine a high-level task list. These tasks were linked by their dependencies and assigned in parallel when possible.

The scheduling for Team Brazil attempted to account for the inherent complexity of a project of this large magnitude. The schedule for the first semester basically entailed getting the various sub-groups to commit hard deadlines to decide upon the final specifications for the team. The biggest milestones in the first semester were the design reviews. The difficulty in the first semester can be attributed to the novelty of the problem to all the students.

With no one having experience with the particular problem presented in RoboCup, the students set out to research past proceedings, to seek out solutions to various related problems, and to acquaint themselves with the RoboCup rules and regulations. After the conceptual design review, each high-level task was detailed to a level which ownership could also be assigned. Ownership was assigned based upon past experience as well as the desire to learn. During the winter break the team began full-scale construction of the robots and on-board electronics. Once classes began, the team was already close to 100% operating capacity in terms of resources and time spent. The fact that this occurred so early in the semester was an indication of the motivation and dedication of *all* the team members. The unfortunate side-effect of this was that the rate of productive work was very difficult to maintain throughout the semester and took its toll on the students. A possible solution would be to schedule a break to occur at some point in the 2nd semester. (The break can be rotated for different groups so that work would continue for the overall project). Once the major components of the system were manufactured, the team operated in debugging mode for the rest of the semester, trying to get all the robots to work as desired (without the need for a redesign).

Given the shared tasks between the two Cornell RoboCup Teams, two schedules were created. The first schedule included the shared vision system, environment development and simulation. The second schedule, unique to the Brazil RoboCup Team, included the artificial intelligence development, mechanical design of the robots, and all electrical components. Instead of linking the two schedules, it was easier to maintain separate schedules while keeping in mind how each subsystem impacted the other tasks.

The shared schedule proved to be useful through the first semester. Instead of using it to drive the project, it was used as a guide. Since a single student was developing the vision system, it was important to understand the impact of the vision system if it was not completed on time. On the contrary, the simulation was completed in a minimal amount of time in order to give the artificial intelligence time to test their system. The amount of time spent on the simulation was minimal compared to what it offered to the AI team. What was underestimated was the development of the environment. The estimated task durations were incorrect and impacted the development of the vision system. In result, the shared schedule is not representative of actual task durations; rather estimated durations.

# Section 11 Environment

## 11.1 Overview

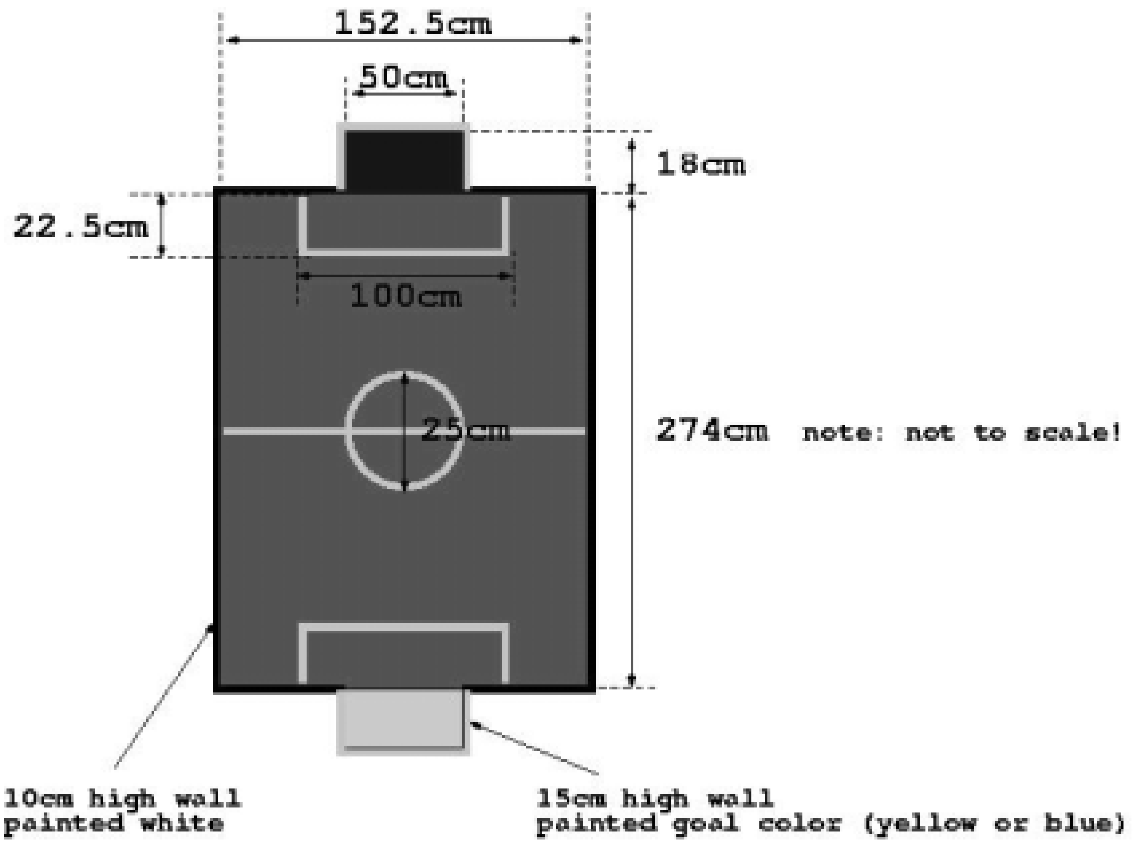
The overall goal of the environment was to accurately reproduce the playing conditions of the international competition. In order to achieve this goal, the rules outlined by the RoboCup organization were followed. The designs also attempted to incorporate the uncertainties in the rules. The focus of the designs was to keep it modular and simple to build and maintain.

The environment was broken into two major structures: the playing field and the lighting and camera mounts. In designing the playing field, the table and the support structure were designed to be separate for the sake of modularity. For the lighting structure, Item MB Kit Systems was used due to the fact that the extruded aluminum was flexible and an off-the-shelf solution. The intention was that efforts be directed on the actual design instead of on the manufacture of the structure.

## 11.2 RoboCup Playing Field Rules

### 11.2.1 Surface

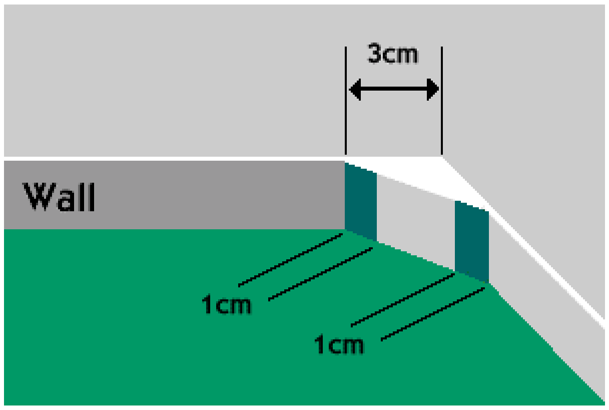
A table tennis table is the official surface for matches. The size and color of the table is defined as the International Table Tennis Federation (ITTF) standard. ITTF regulations concerning the height of the table surface above the floor do not apply. We will provide an appropriate height for the RoboCup competition, which will be chosen to aid the global vision systems. Dimensions are 152.5cm by 274cm; the color is matte green. Every effort shall be made to ensure that the table is flat, however, it is up to individual teams to design their robots to cope with slight curvatures of the surface. (please refer to ITTF Regulations for more details on the table).



### 11.2.2 Walls

Walls shall be placed all around the field, including behind the goals. The walls shall be painted white and shall be 10cm high. Behind the goals walls will be 15cm high and painted one of the two appropriate goal colors.

Four small panels are positioned in the corners to avoid balls getting stuck. As shown in the figure below, they are located 3cm from the corner for each axis. Green stripes 1cm wide are painted on the corners to aid robots in visual identification of the edge of the panel.



### 11.2.3 Goals

The width of each goal is 50 cm, (approximately 1/3 of the width of the shorter end of the field). The goal is 18 centimeters deep. The wall continues behind the goal but increases to a height of

15cm. There is no safety net over the goal, nor is there a horizontal goal bar. Robots can not fall off the playing area because of the wall. The wall and area behind the goal line will be painted either yellow or blue. It should be noted that a robot may use the area behind the goal.

### 11.2.4 Defense Zone

A defense zone is created around each of the goals. It extends from the front of the goal to 22.5cm into the field. The zone is 100 cm wide. Only one robot from each team may enter this area. Brief passing and accidental entry of other robots is permitted, but intentional entry and stay is prohibited.

Once a defending robot (goal keeper) has hold of the ball or is facing and in contact with the ball then all attacking robots must leave the area. The attacking robot can not interfere with the goal keeper. Given the size of the defense zone a robot is said to be in the defense zone if any part of it is within the area.

Also, an attack robot can not intentionally interfere with the movement of the defenders robot in the defense zone. A robot can not be used to block the movement of the goal keeper.

### 11.2.5 Table markings/colors

i. The field shall be dark green. ITTF's color regulation is flexible, there may be slight differences in the color of the table. Robot designers should take this fact into consideration.

ii. Walls are white.

iii. A 1 centimeter thick white line will be painted across the table (the center line), with a center circle 25 centimeters in diameter placed at the center.

iv. The border of the defense zone will be painted in white, with a width of 1cm.

v. The area behind the goal is either dark blue or yellow (one end is dark blue and the other yellow).

## 11.3 Playing Field

The playing field is designed to be modular. As to whether it was built as modular as desired is another thing. The idea behind the design is that the table structure and the outside support should be decoupled providing for the possibility of swapping out either of the two. In my feeble attempt at this decoupling, I have created a product that is not of great quality.

### 11.3.1 Table Structure

The design philosophy for the table structure was to produce a very stiff, long lasting structure. Warping of the ping-pong table is a major concern. The shape of the underlying table structure is a box with a middle beam for extra support.

The materials for the table structure were 2x4 wood pieces cut to appropriate lengths, 2 inch long wood screws and wood glue. The 2x4’s were wood glued to the ping-pong table. Two wood screws at each joint attached the wood pieces to each other.

The inner support is for the table only. This structure was built with the intention of retaining the table's shape so that in the future it wouldn't warp or bend any further.

### 11.3.2 Support Structure

The materials used were 2x4 wood pieces cut to the appropriate lengths, 2 L-brackets, wood glue and a set of four round metal "feet". The outside support runs along the outer edge of each half of the table, with the walls hinged to the support so that easy removal of the walls can take place. The outside support has four feet per half table for each half to be leveled.

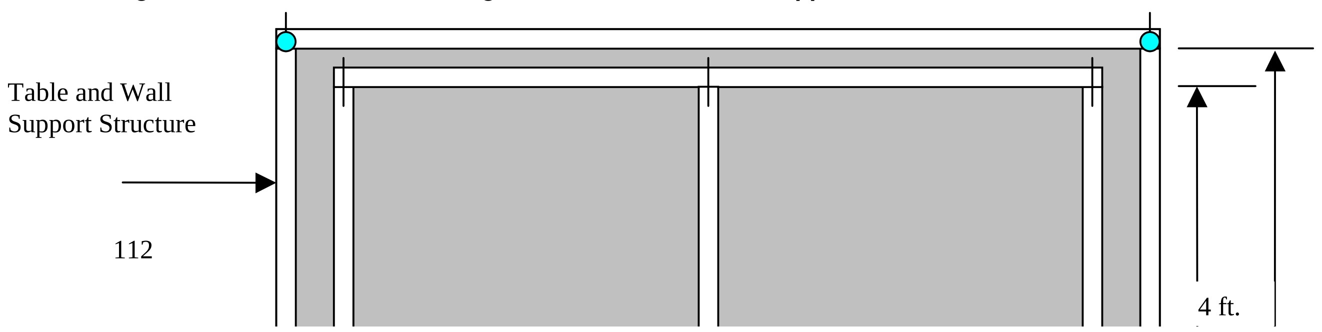
The feet were bought from a hardware store and followed the directions on the package. Drilled 3/8" holes to accommodate for a plastic insert. The foot was then screwed into the plastic insert.

The walls are attached to the support structure via hinges on the underside. The walls are held up by the corner fillets and their supports. The walls were painted with 2 coats of semi-gloss paint. Strips of metal with screws support the corner fillets. The goals were basically an afterthought. Initially the intent was to have removable goals so that they could be moved more easily from place to place. However, this idea was never realized and we attached them to the walls with screws. The goals are made of the same wood as the walls of the field. The support of the goal is a 4x4 with a "foot" for leveling purposes.

All 2x4's actually are 1.5 inches x 3.5 inches. Each 2x4 in the drawing is 1.5 inches wide.

An underlying problem with the support is that the table is assumed to be flat. Because the table became warped during the table structure construction stage, a fix was designed. The fix was to drill out holes in the table structure while holding the table flat. Then the holes would be filled with wooden dowels. The dowels would hold the table flat. To hold the table halves together, we drilled two more holes and used bolts with wingnuts.

**Figure 19. Drawing of Table Structure and Support Structure**



# Section 12 Research and Development

## 12.1 Conceptual

### 12.1.1 Objectives

The objectives of Research and Development are twofold. In the early phases, the R&D group will assist other subgroups (especially mechanical design and simulation) to conceptualize and design robots. This includes but is not limited to concept generation, research, analysis, decision-making, and detailed design work.

As the basic design of this year's team approaches finalization, the R&D's task will be to identify areas for innovation and improvement. The group will then propose, research and test solutions. It is in this second phase when R&D will function more as an individual entity in designing and testing concepts for future competitions.

### 12.1.2 Current and Future Roles

Currently, Research and Development is still in phase I. The focus has been in essentially two areas: ball control and kicking. Ball control concepts that have been developed are backspin rollers and other purely mechanical controls, vibrating plates, active damping, and utilization of multidirectional wheels. Kicking mechanisms that have been conceptualized are solenoids, springs, stiff plates and $CO_2$ cartridges. Some of these ideas have been adopted by the ME group.

Phase II will begin when a more detailed design of the robot team has been established. The beginning of phase II does not signal the end of phase I. Phase II will run concurrently with phase I. Phase II will consist of further development of ideas that are decided to be non-feasible for this year's robot competition. Concepts to be developed will also be drawn from ideas generated by members of other divisions as they come up with ideas, which can not be implemented for this coming year's competition.

The following sections detail some of the designs that R&D has investigated.

### 12.1.3 Concepts for Development

#### Ball Control

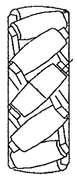
##### Omni-directional wheels

Omni-directional robots hold great promise in robot soccer. The ability to go in any direction at any time has many advantages, including:

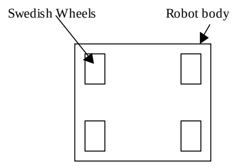
Ability to intersect ball at any angle without loss of time required in turning – quicker paths to reach the ball since no turning is required.

Ability to use 4 different sides (of the robot) which have different functions.

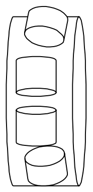
The traditional wheel of choice is the Swedish wheel, shown in Fig 7.a. The arrangement of these wheels on the robot body is shown in Fig 7.b. However, these wheels are not available commercially and are too complex to build, so we will investigate the use of the commercially available wheels shown in Fig. 7.c, which are arranged differently on the robot body as shown in Fig. 7.d. This arrangement permits omni-directional travel, as long as the four wheels are individually controllable.



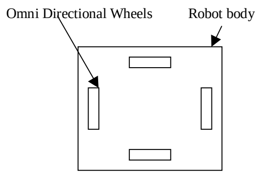
Swedish wheel - Ideal wheel but hard to build



Arrangement of Swedish wheel on robot body



Commercially available wheel

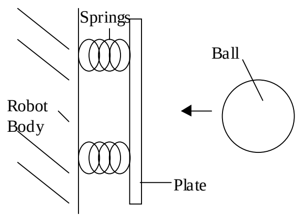


Arrangement of commercially-available wheels on robot body

### Active damping ball receiving system

A major difficulty for a robot to dribble a ball is stopping the ball upon receipt of a pass. A golf ball has a tendency to simply bounce off most surfaces, even soft surfaces. The objective of the active damping system is to bring the golf ball to a controlled, predictable stop upon receipt.

Consider a spring-cushioned plate as shown in Fig. 7.e. A ball hitting this plate will de-accelerate to a velocity of zero as the spring cushion is compressed. However, the spring will then re-extend, accelerating the ball away from the robot which is undesired. If the plate can be halted right when the velocity of the plate-ball system is 0, the ball will not bounce away. Refer to the ball velocity graph shown in Fig. 7.f. Thus, active damping freezes the plate at the right time to prevent ball bounce. The active damper consists of a distance sensor that is highly accurate over a short range to measure plate deflection, an electronics system to detect when the plate has been deflected and is at velocity 0, and a electronically-activated clamp to freeze the plate at the desired time.



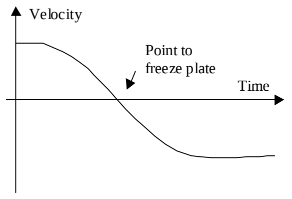


**Figure 20. Top View of Spring-Cushioned Plate Ball Velocity Upon Hitting Plate**

### Kicking

The kicking mechanism is a crucial factor in the Cornell strategy to success. The ability to accelerate the ball quickly is essential to defeating goalies. It is only by accelerating the ball to a speed such that an opponent's goalie cannot react quickly enough that a goal can be scored.

The kicker must adhere to a number of physical and performance-related constraints. Three constraints are of particular importance and are used as guides in designing kicking mechanisms. First is limited space within the robot. Real estate is expensive – the space occupied by the all components of the kicking mechanism (dedicated motors, springs, travel area) must be minimized. Second, power considerations and battery drain must be taken into account. Finally, the kicker must meet minimum performance requirements, such as how fast the ball will be launched.

### Solenoid

A linear solenoid was an obvious choice for a kicker. This could be accomplished either by using the solenoid itself to launch the ball or to use the solenoid to reset a spring-loaded device that would launch the ball. A solenoid can provide large forces over short lengths and there are a large variety of designs and configurations that are readily available. However, solenoids are extremely expensive in terms of power. Considering battery limitations, solenoids are probably not an ideal solution to the kicking problem.

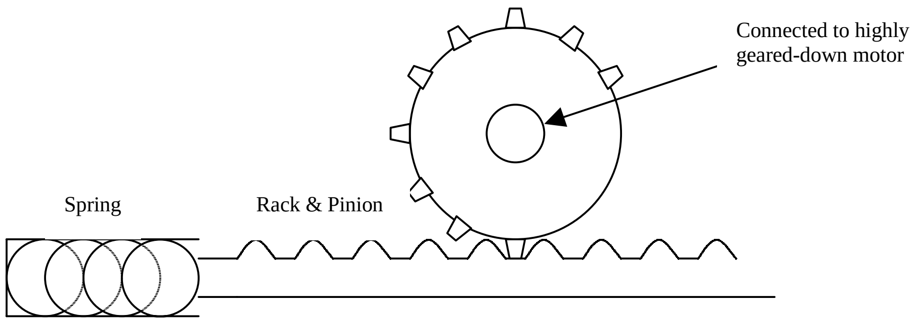
### K-Kickers

The use of a potential energy device reduces the power demands on batteries by distributing work and energy over time. Rather than requiring a sudden high power burst to actuate a kicking device, a steady flow of power would be used to cock or reset a kicker connected to a spring or a stiff beam. The largest drawback of these designs is the “down-time” between kicks when the mechanisms are being reset.

Two ideas for k-kickers are currently being explored. Both involved using a highly geared-down motor to pull back an energy storage device:

The first uses a geared down motor to turn a rack and pinion. The rack and pinion resets a spring. However, notice that the gear has a section without teeth. A wheel encoder would let the robot know when the gear is at its last tooth and the spring is fully reset. Another turn of the gear would release the spring and activate the kicker.

Exactly the same design as the previous except a beam is used as the energy storage device. The beam is pulled back and bent to store energy (as opposed compressing the spring).



### $CO_2$

Carbon dioxide tanks are a small, light, and cheap. Withstanding pressures of over 2000 psi, they have the potential to be the core technology for Cornell’s kicking mechanism. The largest advantage of $CO_2$ kickers is the fact they the do not use battery power to kick the ball (though power may be needed open and close the valve controlling the pressure). Further investigations is required to understand how powerful a $CO_2$ kicker can be and how many kicks can be derived from a full tank of $CO_2$.

## 12.2Preliminary

There are two phases of work for the R&D sub-group. Phase I involves R&D team assisting in the design and construction of the present robot team. The R&D group will assist other subgroups (especially mechanical design and simulation) to conceptualize, design, and build this year's models. Phase II involves identifying areas of need or of high potential, researching possible solutions, and testing the feasibility of those solutions.

Within phase I, the focus has been in essentially two areas: ball control and kicking. Ball control concepts that have been developed are backspin rollers, vibrating plates, active damping, purely mechanical control, and utilization of multidirectional wheels. Kicking mechanisms that have been conceptualized are solenoids, stiff plates and $CO_2$ cartridges.

When detailed design of the robot team has been established, phase II will run concurrently with phase I. Phase II consists of further development of ideas that are decided to be non-feasible for this year's robot competition. Concepts to be developed will also be drawn from ideas generated by members of other divisions as they come up with ideas, which can not be implemented for this coming year's competition. R&D, however, will continue to support the other divisions in their design of this year's robot team.

There are two major designs that R&D is currently working on:

An innovative goalie design using a sliding bumper to take full advantage of the maximum length restrictions and to maximize effective blocking length.

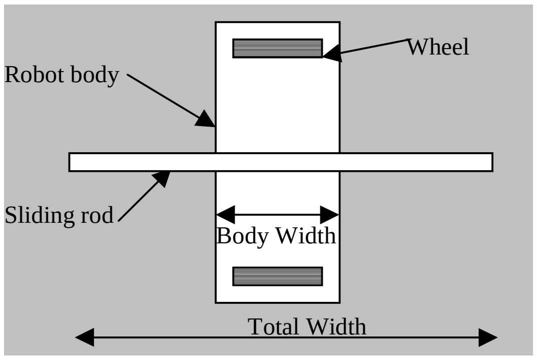
Active Damping, a ball-receive system that does not allow the ball to rebound, allowing a robot to dribble the ball more effectively.

### 12.2.1 CONCEPTS

#### Sliding Bumper Goalie

The philosophy behind the R&D Goalie is very simple: to expand the effective coverage area of the goalie. First, the wider a goalie, the less distance it has to travel to stop a shot on goal. Second, a wider goalie limits the range of angles at which an opponent can shoot.

It seems that the longest legal total robot width is 18cm. By using a sliding bumper which moves from one side of the goalie to the other side, the R&D Goalie is able to block a length longer than the 18cm size limit. The R&D Goalie, while only 18 cm long at full extension, has an effective length of 26.6 cm. The diagram below shows the essential components of R&D Goalie.



## Special Considerations

- Minimizing the body width to maximize effective coverage area by the rod will be increased.
- Being able to withstand impact at the furthest tip of the bumper without being torqued enough to allow the ball to pass into the goal.
- Being able to deal with a fast ball hitting the robot dead-on. The robot should be able to hold its ground and not be forced backward.
- Being able to withstand the impact of a fast ball hitting without toppling. We don't anticipate this to be a problem because balls will hit the goalie at a low center of gravity.
- Being able to withstand impact of other robots, especially since global vision systems may not be able to see the bumper.
- Ensuring that the robot dimensions conform to the rules of RoboCup.
- Being prepared for shoving matches with opponents. An opposing robot may dribble the ball into bumper. However, the goalie is positioned to only move laterally and cannot counter the forward motion of the ball and opponent as they push against the bumper. The robot will almost certainly be torqued. However, we anticipate this problem to be minimal because an opponent dribbling the ball will have low maneuverability (with the current "ball holding" rules, it is very difficult for a robot to execute sharp turns without losing the ball). The goalie ought to be fast enough to intercept the ball using its full body if the opponent attempts to dribble the ball in. The rod is primarily for blocking high-speed balls shot at sharp angles.

## Overall Design

### Acceleration

In the goalie, acceleration takes precedence over maximum velocity because it is unlikely that the goalie will ever reach its maximum velocity. The motors are chosen for their torque characteristics.

### Kicking and Pre-emptive blocking

At the very least, the goalie must be able to make goal kicks and put a stopped ball into motion. By turning quickly (spinning the wheels in opposite directions), the goalie can use its rod to bat the ball away in the general direction of a teammate.

If the global vision anticipates a high speed shot on goal, the goalie can take pre-emptive action and use the same batting to stop a high speed ball in manner that prevents the goalie from being torqued backwards.

### Structural rigidity

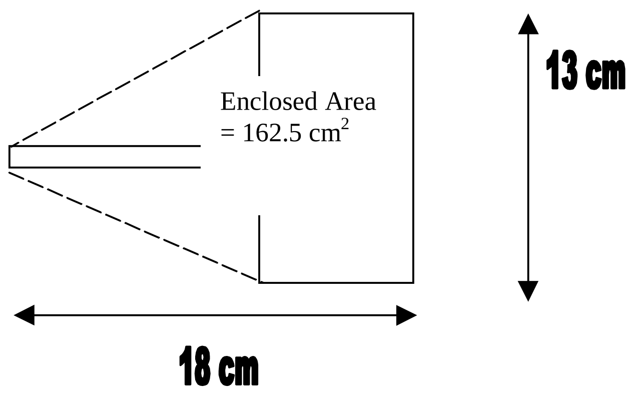
Analyzing videos of Mirosot, it is apparent that a goalie must be able to withstand a significant amount of abuse. Penalties for charging the goalie are common. One can anticipate this being a problem compounded with the R&D Goalie design. It is very likely that the bumper will not be noticeable to many global vision systems, making the R&D Goalie susceptible to charging at the sides. It is a necessity to have a goalie that can withstand the impacts from opponents and high speed balls and be able to hold its ground in pushing matched with opponents.

### Dimensions

In accordance with the rules, the maximum length in any single direction is 18 cm and the maximum area covered is 180 cm$^2$. It is optimal to minimize the width of the body of the robot. The thinner the robot, the more travel the bumper has and the longer the effective coverage length of the goalie.

Accommodation of parts is the most significant restriction on the robot's width. The parts expected to occupy the most space are the two drive motors and the batteries.

To resist torqueing, the goalie should have a long wheelbase. A longer robot provides a longer moment arm through which the friction at the wheel can act and prevent undesired torqueing. Our design dimensions are shown below:



Sliding Bumper Goalie Dimesions

It is also necessary to check for compliance with the maximum enclosed area rule. The figure above shows that enclosed area is 162.5 cm$^2$, less than 180 cm$^2$ limit.

## Bumper and Sliding Mechanism

### Acceleration

The bumper must accelerate quickly enough so that it effectively covers both sides of the goalie. The bumper should move quickly enough from one end of the robot to the other to block a ball traveling at 4 m/s shot from one-quarter of the field. In order to accomplish this, the rod must accelerate at 22m/s$^2$.

### Rod control

A decision must be made between having a rod with binary control (only Full Right and Full Left positions) and having a rod with a range of positions (a gradient of positions). We examined the trade-offs and decided on a binary control. Refer to the trade-off table below:

| | Binary Rod Control (Full Left/ Full Right) | Fine Rod Control (Range of Positions) |
|---|---|---|
| Advantages | a) No electrical de-acceleration required. Motor accelerates to full speed in one direction. A trigger switches off motor, and a short spring cushions impact. | a) A centralized rod position allows for more effective coverage for a ball that's coming in from the center. |
| Disadvantages | a) Less flexible<br>b) Greater wear of rack & pinion | a) Careful control of rod acceleration and de-acceleration required.<br>b) Certain positions are inherently ineffective, i.e. a rod length less than the diameter of the ball will not be effective.<br>c) Complex local control |

### Local vs Global Control

Control of the rod will be left to global systems. Using ball positions from global vision, the central AI can determine the location of the rod. Local sensing would only be necessary if the goalie obscures the ball from the camera during play (it would be very bad to lose the ball close to your own goal). However, we have found that this is unlikely (see calculations regarding Camera Loss of Ball in the Analysis section).

## Collision Analysis

It is the goal of this section to quantify ball-goalie collisions, and to show that these collisions will not result in a goal being scored. We analyze collisions in two cases: Dead-on collision, where the ball directly hits the front of the robot, and rod collision, where the ball hits the rod near the end.

Assumptions will be made about the mass of the robot and ball, and the coefficients of static and kinetic friction. The assumptions are:

Mass of robot $m_r$= 1.5 kg

Mass of ball $m_b$ = .046 kg

Coefficient of static friction $\mu_s$ = 0.6

Coefficient of kinetic friction $\mu_k$ = 0.5

Robot body and ball are inelastically rigid

Further assumptions are made in the analysis. These assumptions do degrade the accuracy of the calculations to a certain degree. However, because we want ballpark figures these assumptions are ok.

Dead-On Collision

Ball hits the robot directly in front

No rotational torque exists around the wheels.

*PROBLEM 1*: Find velocity of ball ($v_{slip}$) that just causes the robot to slip.

*Assumptions*: After ball hits the goalie, the ball comes to a complete stop. This change of momentum exerts constant force $F_{ave}$ on the robot.

Calculations:

Impulse on robot,

$I = \Delta p$

$I = m_b v_{slip}$

$m_b v_{slip} = F_{ave} t$

$$\mathbf{F}_{ave} = \frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}}$$

For the robot to be just in slip,

$F_{ave} = F_s$

$$\frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}} = \mu_s \mathbf{N}$$

By assuming a safe value of t = 0.01s, we can solve for $v_{slip}$:

$$v_{slip} = \frac{9.8 m_r \mu_s t}{m_b}$$

$$v_{slip} = 1.8 \text{ ms}^{-1}$$

We do not expect many balls to be of a velocity of more than 1.8 m/s. Thus, we do not expect the goalie to slip many times.

*PROBLEM 2*: If the ball IS going at faster than $v_{slip}$, find approximately the distance the goalie will slip.

*Assumptions:* See problem 1. A ball with velocity $v_{slip}$ heading dead-on will just cause a ball to slip.

Calculations:

According to assumptions, a ball with kinetic energy:

$$\mathbf{E}_{slip} = \frac{\mathbf{m}_b \mathbf{v}_{slip}^2}{\mathbf{2}}$$

will just cause the robot to slip. A ball with more kinetic energy, however, will cause the robot to slide distance d. The energy transfer required to make the robot move is:

$E_{move} = E_{total} - E_{slip}$

$= (1/2)( m_b v^2 – m_b v_{slip}^2)$

This energy is then absorbed by friction as the robot moves a distance d.

$E_{move} = F_k d$

Solving for d:

$$d = \frac{\frac{1}{2}(\mathbf{m}_b \mathbf{v}^2 - \mathbf{m}_b \mathbf{v}_{slip}^2)}{\mathbf{9.8m}_r \mu_k}$$

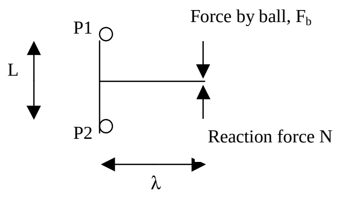
Using the previously calculated $v_{slip}$, we find the slip distance of the robot if an incoming ball has velocity $v = 2.5\ ms^{-1}$:

d = 1.0 cm

This is a safe distance to slide -- a distance of more than 10 cm is needed before a goal is scored. We conclude that the goalie design will not suffer from slip problems in a head-on collision.

### Ball-Bumper Collision and Torque Analysis

There are concerns that if the ball hits near the end of the extended bumper, torqueing will cause the robot to spin and let the ball through to the goal. When the ball hits the end of the rod, there are two possible pivot points corresponding to the wheels, labeled P1 and P2 in the diagram below.



The figure shows possible pivot points P1 and P2 corresponding to the two wheels, as well as force exerted by ball near edge of rod.

The torque exerted by Fb will be the same on P1 and P2. Thus, which pivot point will be the actual pivot point will depend on the weight on P1 or P2. For our analysis, we assume that the mass of robot $m_r$ is equally distributed over P1 and P2, but the pivot is P2 (due to the weight of the rod system on P2).

*PROBLEM 3*: Find $v_{slip}$, velocity of ball that just causes the robot to slip and spin on pivot P2.

*Assumptions:* Contact time between ball and bumper t = 0.01 s.

Calculations:

$\Delta p = F_b t$

$$\mathbf{F}_b = \frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}}$$

For the situation where robot is about to slip, sum of torques around P2 is 0:

$$\frac{\mathbf{m}_b \mathbf{v}_{slip} \lambda}{\mathbf{t}} = \frac{\mu_s \mathbf{NL}}{\mathbf{2}}$$

Solve for $v_{slip}$:

$V_{slip}$ = $9.8 m_r \mu_s L t$

$2 m_b \lambda$

Using a safe assumption of t = 0.01s, and L=14 cm and λ=14cm, we find:

$v_{slip} = 1.2\ ms^{-1}$

We note that in the torqueing situation, a smaller ball velocity can result in slip. This is a potentially dangerous situation. To curb this, we will try to maximize wheelbase length L, maximize coefficient of static friction, maximize contact time t by soft-padding the rod, and use pre-emptive motion of the goalie.

## Stress and Deflection Analysis of Bumper

The shape and material of the R&D Goalie's bumper was analyzed for possible yielding and for excessive deflection. Yielding of the bumper would severely limit the performance of the goalie and could result in near-catastrophic failure if the bent bumper jams the sliding mechanism. An extremely flexible bumper is undesirable because a ball colliding at the tip of the bumper could pass into the goal.

The velocity of the ball to be blocked is assumed to be 4 m/s. A ball traveling at that speed has .6 J of kinetic energy (this calculation of energy is high to provide conservative numbers). It is assumed that, upon collision with the bumper, the ball comes to a complete stop and all energy is transferred to the bumper. The impulse is easily calculated as:

$$I = m\Delta V = m(V_f - V_i) = mV_f$$

I = .05kg * 4 m/s = .2 (kg*m/s)

To derive force, the time duration of the impulse is needed:

$$F = \frac{I}{\Delta t}$$

However, we do not know the exact amount of time the ball is in contact with the bumper. We can estimate this to be .01 seconds. F = .2/.01 = 20 N.

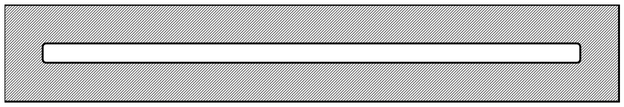
The maximum flexural stress in a beam is calculated as:

$$\sigma_{\max} = \frac{M * c}{I} = \frac{F * L * c}{I}$$

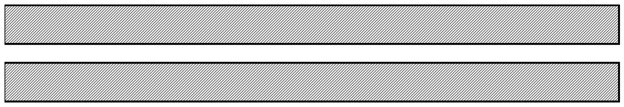
where F is the applied force, L is the length of the moment arm, c is the furthest fiber from the neutral axis, and I is the second moment of inertia.

At this point, the geometry of the bumper must be taken into consideration. Both I and c vary with geometry. When designing for stiffness, I-beams immediately come to mind. A T-shape, a channel, and a hollow cylinder are also possibilities. However, there are constraints on the design of the bumper. Its cross-section must be fit within the allotted space of .05m x .01 m. Also, the bumper must be rigidly mounted to the 13/32" wide rack in the sliding mechanism. Considering the small size of these parts, it would be difficult to mount I-beam's and other shapes to the rack. It was decided that a simple solid beam would suffice. The weight gained by using a solid beam is not significant and the solid beam design facilitates manufacturing and maintenance.

Choosing a beam of depth .5 cm and height of 1 cm, $\sigma_{max}$ = 3.84 MPa. This is only a fraction of the yield strength of aluminum 6061. However, this analysis does not take into account the milled channel in the bumper:



The milled channel not only removes material but also causes stress concentrations. To do a detailed analysis, FEA is required. However, to get a ballpark number on stress, the bumper can be modeled as two separate thin beams.



Taking the bumper to be two thin beams gives a $\sigma_{max}$ = 154 MPa. This is 57% of the yield strength of aluminum. This is a liberal estimate -- areas of stress concentrations would see stresses even higher. However, this is not a source of concern because of slip; the goalie would slip at the wheels and rotate before stresses reached dangerous levels.

## Main Drive-Motor Selection

For a goalie, speed characteristics can be sacrificed for greater acceleration. The goalie will never go in a single direction long enough to reach top speed. High acceleration motors will be necessary for the goalie.

## Rod-Motor Selection

The primary objective considered in selecting a motor for the sliding rod mechanism was the quickness with which the rod would move into either of its two positions, full-left or full-right. It was determined that to accomplish this the motor should be one that delivers high torque and therefore high acceleration. Because the sliding rod moves a maximum of 0.08 meters, the motor will never reach top speed. The target acceleration was determined by assuming a ball traveling 4 m/s is redirected 0.34 meters from the front of the goal (a quarter the length of the field) towards the goal. This is the quickest desired reaction for the rod. So the rod must go from full-right to full-left (or vise-versa) in:

$$\frac{0.34m}{4m/s} = 0.085\sec$$

So if this is to be the time that rod has to travel the distance of 0.08 meters and it is assumed that the rod will be accelerating at a constant rate (i.e. it will not reach top speed). Then this acceleration needed can be found.

$$x = x_0 + v_0 t + \tfrac{1}{2} a t^2$$

With $x_0$, $v_0 = 0$, this simply becomes

$$x = \tfrac{1}{2} a t^2$$

Which can be rearranged to solve for a

$$a = \frac{2x}{t^2} = \frac{2 * 0.08m}{(0.085s)^2} \approx 22.15 m/s^2$$

With this acceleration the force needed to move the rod can be found by using a rod of mass 0.05 kg and no friction resisting motion:

$$F = ma = 0.05kg * 22.15 m/s^2 = 1.11N$$

Now a motor can be selected. A rack and pinion system has been chosen, and the pitch diameter is 3.2 mm. If this is so then the torque required to generate this force is.

$$T = F * \ell = 01.11N * 0.0016m = 1.78mNm$$

However, we need to consider the rotor inertia as well. The motor we have selected has a rotor inertia of 0.298 gcm^2.

Rotor inertia = I = $2.98 \times 10^{-8}$ kgm$^2$

$$\alpha = \frac{a}{r} = \frac{22.15ms^{-2}}{0.0016m}$$

$$\alpha = 13844 rad/s^2$$

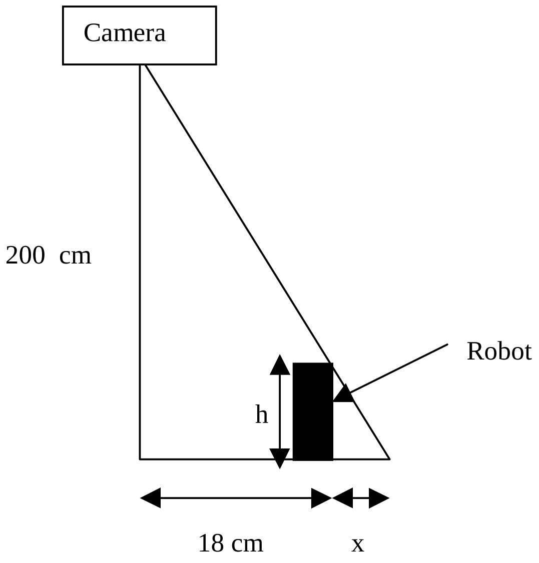
$$\mathrm{T} = I\alpha = 0.41mNm$$

Thus, we have found the total torque that needs to be generated by the motor:

Total torque = 0.41 + 1.78 = 2.2 mNm.

### Camera Loss of Ball

In the worst case scenario, the camera will only be mounted 200 cm above the field. At the far end of the goal, the center of the robot will be 18cm away from the center of the field. The area obscured from the camera is a function of the robots height.



$$\frac{h}{x} = \frac{200}{18 + x}$$

Substituting 18 cm in for h, we find that the obscured area is 1.78 cm. Over half the ball will always be visible to the goalie. This should be enough for global vision to track the ball. R&D Goalie will forgo local sensing and rely on global.

## Budget for R&D Goalie

**RoboRod Goalie Additional Cost Analysis**
Items unique to Goalie Design

| Item | Vendor | Part # | Cost per piece | Quantity | Total Cost |
|---|---|---|---|---|---|
| Rod Motor | Maxxon Inc. | 118628 | $100.00 | 1 | $100.00 |
| Rack (brass) | SDP/SI metric components | A1B12MY04B300 | $20.00 | 2 | $40.00 |
| Pinion (delrin) | SDP/SI metric components | A1B12MY04008 | $25.00 | 2 | $50.00 |
| Chassis | | | $50.00 | 1 | $50.00 |
| Bumper | | | $10.00 | 1 | $10.00 |
| | | | | Total: | $250.00 |

## THE ACTIVE DAMPING BALL RECEIVE SYSTEM

The major obstacle for a robot to dribble a ball is to actually stop the ball upon receipt of a pass. The golf ball has a tendency to simply bounce off most surfaces including soft surfaces. The objective of the active damping ball receive system is to bring the golf ball to a controlled, predictable stop upon receipt.

Consider a spring-cushioned plate as shown in Fig. 1. A ball hitting this plate will de-accelerate to a velocity of zero as the spring cushion is compressed. However, the spring will then re-extend, accelerating the ball away from the robot which is undesired. If the plate can be frozen right when the velocity of the plate-ball system is 0, the ball will not bounce away. Refer to the ball velocity graph shown in Fig. 2. The objective of active damping is to freeze the plate at the right time to prevent ball bounce. The active damper consists of a distance sensor to measure the plate deflection, an electronics system to detect when the plate has been deflected and is at velocity 0, and an electronically-activated clamp to freeze the plate at the desired time.

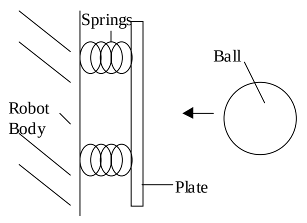


**Figure 21. Top view of a spring-cushioned Plate attached to robot body**

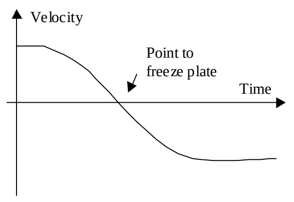


**Figure 22. Bvelocity graph upon hitting plate**

We note here that it is possible to do all of the above with a purely mechanical system. Such a system would have the plate connected to a rod that can be pushed back only in one direction. Thus, when the ball hits the plate, it will compress the springs to the maximum point, and in doing so pushes the rod, but since the rod cannot be pulled back, the system freezes. The rod and plate can then be released when appropriate. However, such a mechanical system has the drawback of wear and tear. The rod inherently works on friction everytime it is pushed back. Therefore, we concentrate here on an electronic-intensive system, with minimum mechanical systems.

The overall system is as follows:

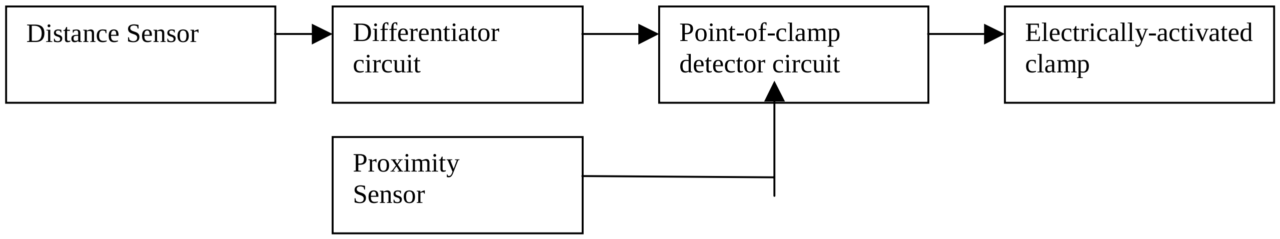


**Figure 23. Block diagram of Active Damping system**

A brief description of the individual blocks:

The distance sensor measures the distance of the ball-receive plate and outputs a voltage that is proportional to distance.

The differentiator circuit takes the distance voltage and differentiates it, producing a voltage proportional to velocity of the ball-receive plate.

The point-of-clamp detector circuit detects the point when the ball has compressed the ball-plate-spring system to the maximum point, and is about to rebound.

The electrically-activated clamp freezes the ball-plate-spring system on electrical command.

The proximity sensor will reset the point-of-clamp detector when the ball is no longer within close range, thus releasing the compressed ball-plate-spring system.

Now we shall discuss the various components of the active damper. Each component will be presented with one or several solutions, and the trade-offs of each will be examined.

## The Distance Sensor Block

The goal of the distance sensor block is to measure plate deflection and output an analog waveform that is proportional to deflection. That is:

$$V_{output} = A + (B \times (D_{deflection})^n)$$

where:

$V_{output}$ = Output voltage of sensor

A = Any constant

B = Any constant

n = Any constant

$D_{deflection}$ = Deflection distance of plate

As can be seen from the above, and non-linearities, DC offsets and multiplication factors in the sensor is not a problem. The only requirement is that $V_{output} \propto D_{deflection}$. The reason for this is we are interested only to find out when the first derivative of $V_{output}$ is 0, which is when the velocity of the plate is 0:

$$\frac{dV_{output}}{dt} = (V_{plate})(nB)(D_{deflection})^{n-1}$$

We note from the above equation that when $V_{plate} = 0$, the derivative of $V_{output} = 0$.

There are several ways we can implement a distance sensor that satisfies the above requirements.

### Laser/IR range finding device

This is the most obvious solution to the distance sensor problem. A laser/IR range finding device works by sending an output pulse, and by measuring the time and phase difference of the returning pulse, distance can be determined. Consider the layout shown below in Fig. 4:

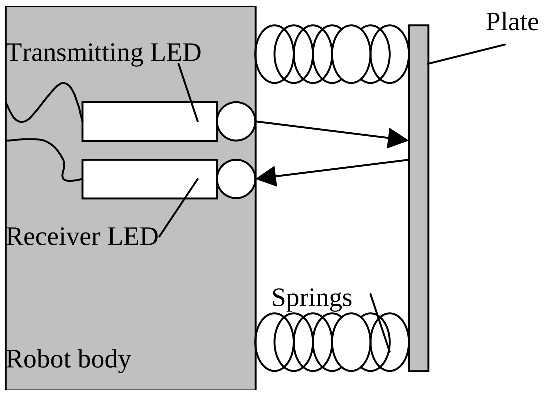


**Figure 24. Using a Laser/IR range finding system**

However, this solution has several drawbacks. In order for the system to be effective, we need a sensor with very high accuracy ( $\pm$ 0.1mm) over a very short range ( $\approx$ 1.5 cm). Sensors in this category are very hard to find and are expensive. Therefore, we present alternative solutions to the distance sensing problem.

### LED and Phototransistor system

Here we use a similar setup as the laser/IR device. Refer to Figure 5.

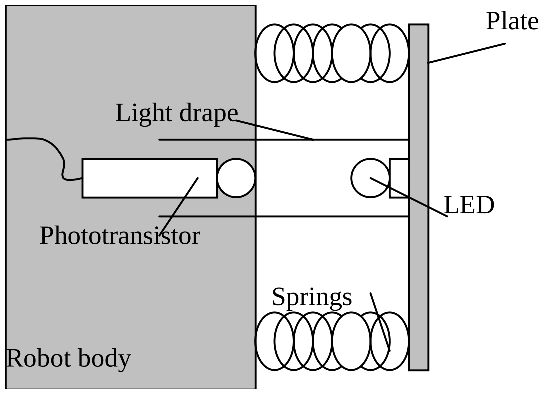


**Figure 25. LED and Phototransistor system to determine distance**

In this setup, we shine an LED continuously opposite a phototransistor in a closed area (i.e. no external light). The phototransistor is sensitive to small variations in light. When the plate is deflected, the LED gets closer to the phototransistor, and this will modulate the output voltage. This setup is preferred to the Laser/IR range finding system because it is a cheaper and simpler system. However, accuracy is an issue and will not be determined until further testing is done.

### Magnetic Induction method

This method is different from the above methods in that the Magnetic Induction method directly outputs a waveform that is proportional to velocity.

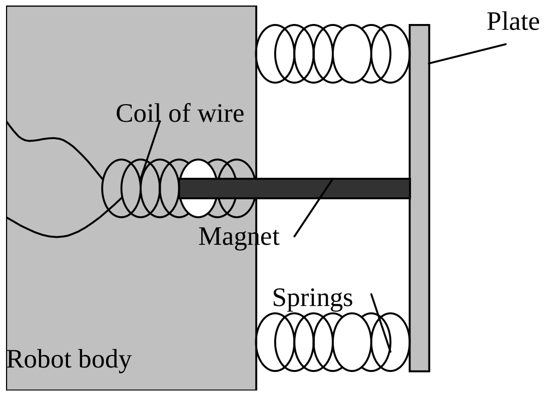


**Figure 26. Magnetic Induction method to directly measure plate velocity**

Here, a magnet is arranged to thrust into a coil of wire as the plate as deflected. Faraday's Law states that the induced EMF in a coil of wire is proportional to the change in magnetic flux through it:

$$|\varepsilon| = \frac{d\phi_m}{dt}$$

Magnetic flux is defined as:

$\phi_m = BA$

where

$B$ = Magnetic force

$A$ = Area covered by coil

Thus, we will find that:

$\varepsilon_{output} \propto Velocity_{plate}$

With this method, no further differentiation of the output signal is required, as the waveform is a measure of the velocity of the plate, and will be zero when plate velocity is zero.

Because this method is simple and requires no differentiator circuit, it appears as a viable solution. However, this is sensitive to magnetic noise, which is inherent in a robot with motors. The actual sensitivity will be studied during prototyping. If the noise is found to be too high, magnetic shielding could be used to surround the coil of wire.

### LVDT (Linear Variable Distance Transducer)

This is a method we have just learnt of, and no analyses have yet been done. But it seems a very viable method of measuring distances. It uses a central coil will AC running through it, and two secondary coils spaced equally apart from the central coil. The secondary coils move with distance, and the reluctance of the two coils then change. This can be detected by measuring the voltage differences in the two secondary coils.

The advantages of this system is it is very accurate to distance changes, and is almost linear in the region of interest. However, the circuity involved is far more complex, as DC-AC converters and AC-DC converters are needed.

### The Differentiator Block

After getting a voltage that varies with distance, we need to differentiate the output signal of the distance sensor because we are interested in the plate velocity. Refer to the differentiator circuit block in Figure 8. $V_{out}$ is:

$$V_{out} = -R1C1 * \frac{dv_{in}}{dt}$$

This differentiator has problems at high frequencies, but we don’t expect high frequencies with this system, i.e. the ball hitting the plate and getting deflected occurs at no more than about 5 Hz.

We choose R1C1 to be about equal to the period of the input signal for a well-defined signal.

Note again that we do not need this differentiator block if we use the Magnetic Induction method above.

### The Point-Of-Clamp Detector Block

We want to be able to detect when the plate is at zero velocity, but only after a deflection has occurred from its stable point. A sample velocity waveform output from the Differentiator Block is shown below in Figure 7.

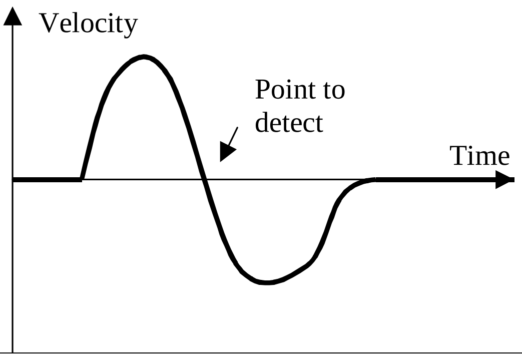


**Figure 27. Plate velocity as a function of time during ball contact**

As we can see, the velocity of the plate quickly increases when the ball hits the plate. The springs bring the plate to a stop at the zero-crossing point, and reaccelerates it out. The objective of the point-of-clamp detector block is to detect the zero-crossing after plate deflection occurs. We can do this using a Schmitt trigger, Active Peak Follower, and Voltage subtractor. Refer to the circuit in Figure 8.

Before any activity from the differentiator, the Schmitt trigger is active low. When the voltage of the differentiator goes up beyond $V_{hi}$, the Schmitt trigger switches to active high. When the voltage of the differentiator approaches zero and crosses $V_{lo}$, the Schmitt trigger switches back to active low.

The peak follower close follows the output of the Schmitt trigger, but remains at active high when the Schmitt trigger switches back to active low. Refer to Figure 9 for a simultaneous view of the outputs of the differentiator, Schmitt trigger and peak follower.

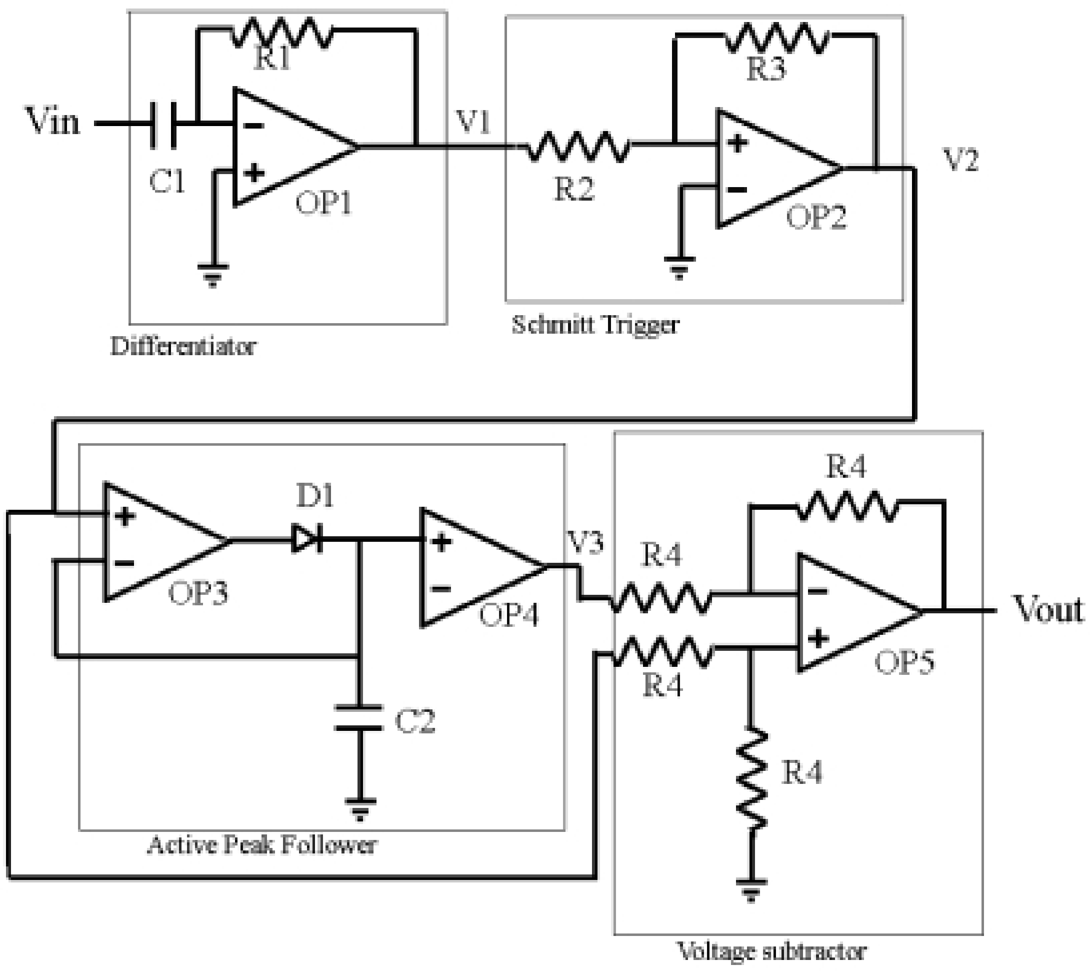


**Figure 28.** **Overall circuit diagram of differentiator block and point-of-clamp detector block, which comprises the Schmitt trigger, active peak follower, and voltage subtractor.**

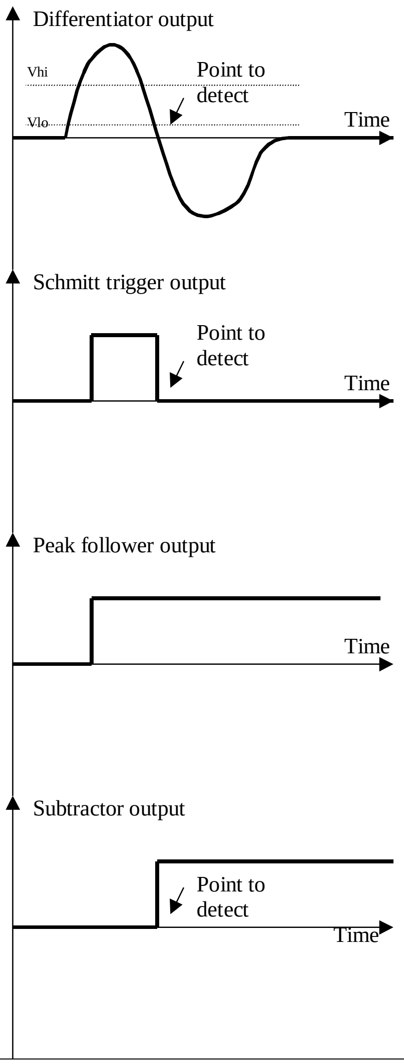


**Figure 29. Simultaneous outputs of differentiator, Schmitt trigger, peak follower and voltage subtractor**

We now note that the voltage difference between the Schmitt trigger output and the peak follower ouput will give an active high voltage at the right point. This is achieved by the voltage subtractor block. Bottom graph shows the output of the voltage subtractor. This output is then directly connected to an electronic switch which powers the clamp.

### Resetting

After losing the ball, the robot will have to reset the peak follower output to low so that the clamp will be released. This is achieved using a proximity sensor to sense when the ball is far away. When the ball is far away, a switch shorts out the capacitor C2, which resets the peak detector.

### Possible problems

The op-amps do not have infinite slew rate. This can complicate matters because the peak follower might now follow the peak of the Schmitt trigger exactly, and the output of the voltage subtractor might not be all zero before switching on. However, we are dealing with outputs of a mechanical system. This has inherently low frequencies involved, and we thus expect no problems.

## The Electronically-Activated Clamp

An electromagnetic force will be applied to clamp down on a rod connected to the ball-receive plate. A rachet design will be used.

## Active-Damping-System Budget

We will attempt to first use the magnetic induction method above. The materials for such a system is thus:

| | |
|---|---|
| 5 Op-Amps | 0 (Already in lab) |
| Resistors | 0 (Already in lab) |
| Capacitors | 0 (Already in lab) |
| Wire | 0 (Already in lab) |
| Strong magnet | \$20 |
| Springs, etc | \$30 |
| | ----- |
| Total | \$5 |

## 12.3 Detail Design

It has been the goal of the R&D subgroup to provide information, ideas, and tools that will benefit future teams. Early on the R&D group decided that contributions in ball control would give the Cornell RoboCup team a significant advantage in future years. To this end the group has brainstormed and developed a number of interesting and potentially useful ideas. Two were pursued: an Active Damping Ball Receiving System that prevents the ball from bouncing off our robots' face upon receiving a pass or intercepting the ball and a Sliding Bumper Goalie that takes full advantage of space restrictions. Only the Sliding Bumper Goalie was designed in detail.

### 12.3.1 Sliding Bumper Goalie (R&D Goalie)

The goal box for the medium-sized robot competition, in which we are competing, is 50 cm in width. With a maximum robot width 18cm, there are 32 cm of open goal for an opponent to score. The objective of the Sliding Bumper Goalie is to reduce the amount of open goal while working within the rules of RoboCup. This is accomplished by using a sliding bumper which slides from one side of the goalie to the other side.

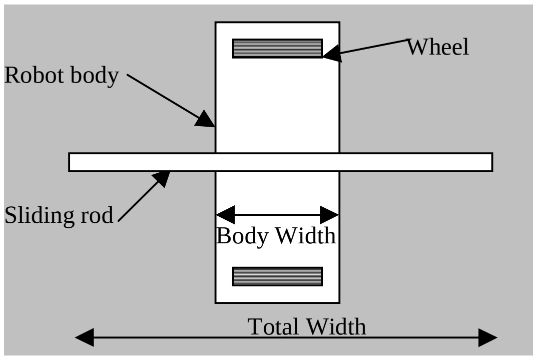


Figure 30. Sliding Bumper Goalie Illustration

To maximize the effectiveness of this design, the body of the goalie must be as thin as posibble. The minimum body width is limited by the space required to house the wheel motors, the sliding mechanism motor, and the electronics. With the current design, the R&D goalie effectively covers 26.6 cm of length at a time.

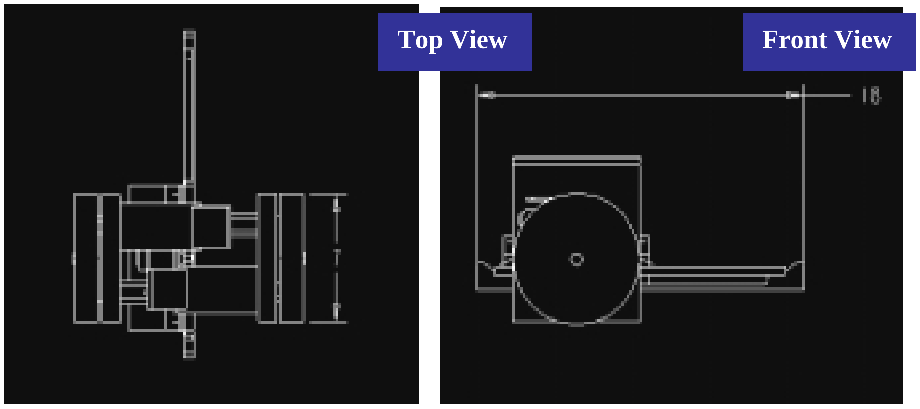


Figure 31. Sliding Bumper Goalie

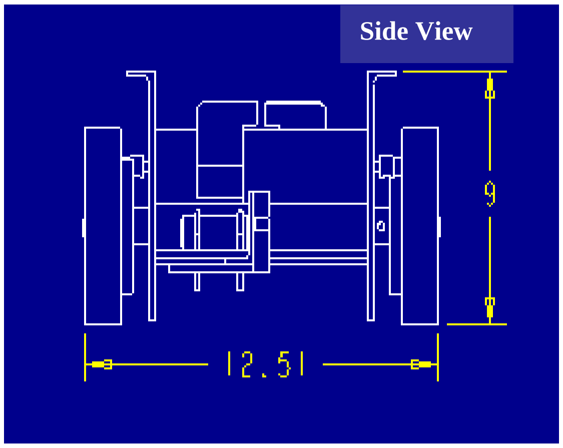


Figure 32. Sliding Bumper Goalie

### 12.3.2 Compatibility

In order to reduce inventory, lower costs, increase modularity, and improve serviceability, the R&D team attempted to keep the R&D goalie as compatible with the regular field players as possible. To accomplish this, the R&D goalie uses the same basic components as the regular players as possible, including 4-40 screws, wheels, and batteries. In the command calls given by the central computer to the field players, the "kick bit" is used to control the direction of the sliding bumper – eliminating the need for specialized code or additional overhead to run the R&D goalie. Of course, the sliding mechanism is completely unique to the R&D robot.

### 12.3.3 Sliding Bumper Mechanism

#### Motor Selection

The primary objective considered in selecting a motor for the sliding rod mechanism was the quickness with which the rod would move into either of its two positions, full-left or full-right. It was determined that to accomplish this the motor should be one that delivers high torque and therefore high acceleration. Because the sliding rod moves a maximum of 0.08 meters, the motor will never reach top speed. The target acceleration was determined by assuming a ball traveling 4 m/s is redirected 0.34 meters from the front of the goal (a quarter the length of the field) towards the goal. This is the quickest desired reaction for the rod. So the rod must go from full-right to full-left (or vise-versa) in:

$$\frac{0.34m}{4m/s} = 0.085\sec$$

If this is to be the time that rod has to travel the distance of 0.08 meters and it is assumed that the rod will be accelerating at a constant rate (i.e. it will not reach top speed). Then acceleration can be found using:

$$x = x_0 + v_0 t + \tfrac{1}{2} a t^2$$

With $x_0 = v_0 = 0$, this simply becomes

$$x = \frac{1}{2}at^2$$

Which can be rearranged to solve for a

$$a = \frac{2x}{t^2} = \frac{2 * 0.08m}{(0.085s)^2} \approx 22.15m/s^2$$

With this acceleration the force needed to move the rod can be found by using a rod of mass 0.05 kg and no friction resisting motion:

$$F = ma = 0.05kg * 22.15m/s^2 = 1.11N$$

Now a motor can be selected. A rack and pinion system has been chosen, and the pitch diameter is 3.2 mm. If this is so then the torque required to generate this force is.

$$T = F * \ell = 01.11N * 0.0016m = 1.78mNm$$

However, we need to consider the rotor inertia as well. The motor we have selected has a rotor inertia of 0.298 gcm^2.

Rotor inertia = I = 2.98 x $10^{-8}$ kgm$^2$

$$\alpha = \frac{a}{r} = \frac{22.15ms^{-2}}{0.0016m}$$

$$\alpha = 13844rad/s^2$$

$$\mathrm{T} = I\alpha = 0.41mNm$$

Thus, we have found the total torque that needs to be generated by the motor:

Total torque = 0.41 + 1.78 = 2.2 mNm.

| Manufacturer | Maxon Motors |
|---|---|
| Model | 118527 |
| Price | $59.10 |
| Desired Torque (mNm) | 2.2 |
| Stall Torque (mNm) | 3.2 |

**Figure 33. Sliding Motor Chart**

## Rack and Pinion

The rack and pinion chosen have a module of .4. The primary constraint was finding a rack and pinion set with a pinion bore small enough to be press fitted onto the sliding motor shaft. This severly limited our options. The biggest problem encountered within the limited selection was the lack of choices in material.

Since the pinion is fixed and the rack is replaceable, it is desireable to have a rack of softer material than the pinion. The replaceable rack will fatigue rather then the pinion mounted onto the motor shaft. The R&D group was unable to find a properly meshing rack of softer material that the available pinions. The R&D team settled for a brass on brass rack and pinion set.

### Bumper Control

The bumper control is directed by the global AI. The global AI was chosen over local control because any local control at the robot level would solely be based on wheel velocities. The global AI has information about the ball, the players, the opponents, and the respective velocities. The global system has the data to make more intelligent choices about the position of the rod.

It was debated whether the bumper should have a gradient of positions or whether it should only have two positions (full left and full right). Allowing the rod to have a range of position allows for more complicated strategy. However, the defense advantage gained from a gradient of positions is only marginal. In most scenarios it advantageous to have the bumper in the full left or full right position. The marginal gains from a gradient of positions require a non-trivial amount of complexity. The mechanical system would require an additional wheel encoder. The electronics would need to process the additional information. Also, the communcation system would be complicated; there would need to be a specific protocol for letting the goalie know how far to move the bumper. The standard on/off kick bit would not suffice. Thus, it was decided to only have full left and full right positions for the bumper.

Though very little code has been written, the strategy of the goalie has been discussed in depth. The robot should track the ball as if it were a goalie of small width. The bumper provides a large safety factor in blocking the ball.

Most goals come from the side or corners of the field. The robots strong motors and bumper should easily defend against attacks from the sides. However, since the robot has no traction or force to counter a robot pushing toward the goal, the possibility of a robot dribbling the ball into the bumper and past the goalie is very likely. But any robot dribbling the ball has very little manuverability due to the rules of ball-holding and concavity. Thus, the AI should be able to predict the path of the ball reliably. A ball being dribbled in should be easily stopped the body of the goalie. A last minute shot near the goal would be blocked the bumper.

### Bumper Deflection

Yielding of the bumper would severely limit the performance of the goalie and could result in near-catastrophic failure if the bent bumper jams the sliding mechanism. An extremely flexible bumper is undesirable since a ball colliding at the tip of the bumper could pass into the goal.

The velocity of the ball to be blocked is assumed to be 4 m/s. A ball traveling at that speed has .6 J of kinetic energy (this calculation of energy is high to provide conservative numbers). It is assumed that, upon collision with the bumper, the ball comes to a complete stop and all energy is transferred to the bumper. The impulse is easily calculated as:

$$I = m\Delta V = m(V_f - V_i) = mV_f$$

I = .05kg * 4 m/s = .2 (kg*m/s)

To derive force, the time duration of the impulse is needed:

$$F = \frac{I}{\Delta t}$$

However, we do not know the exact amount of time the ball is in contact with the bumper. We can estimate this to be .01 seconds. F = .2/.01 = 20 N.

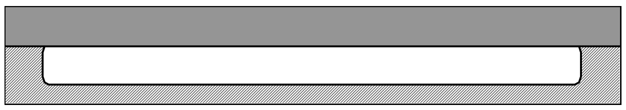
The maximum flexural stress in a beam is calculated as:

$$\sigma_{\max} = \frac{M * c}{I} = \frac{F * L * c}{I}$$

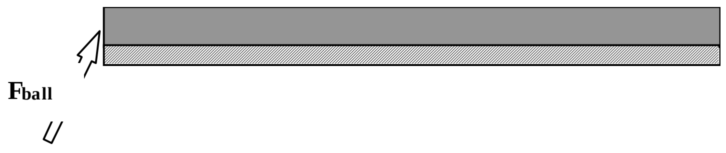
where F is the applied force, L is the length of the moment arm, c is the furthest fiber from the neutral axis, and I is the second moment of inertia.

At this point, the geometry of the bumper must be taken into consideration. Both I and c vary with geometry. When designing for stiffness, I-beams immediately come to mind. A T-shape, a channel, and a hollow cylinder are also possibilities. However, there are constraints on the design of the bumper. Its cross-section must be fit within the allotted space of .05m x .01 m. Also, the bumper must be rigidly mounted to the 13/32" wide rack in the sliding mechanism. Considering the small size of these parts, it would be difficult to mount I-beam's and other shapes to the rack. It was decided that a simple solid beam would suffice. The weight gained by using a solid beam is not significant and the solid beam design facilitates manufacturing and maintenance.

Choosing a beam of depth .48 cm and height of .4 cm, $\sigma_{\max}$ = 86 MPa. However, this analysis does not take into account the true geometry and materials of the bumper. The above dimensions only account for the rack. The bumper actually consists of a brass rack connected to the top of an aluminum U-mount - forming a channel.



The is bumper consists of varying geometries, mixed materials, has obvious stress concentrations. To do a detailed analysis, FEA is required. However, to get a ballpark number on stress, the bumper can be modeled as a composite beam, a layer of brass rigidly attached to a layer of aluminum:



Both the brass and aluminum sections are .48 cm deep. The brass section is .4 cm high while the aluminum section is .27 cm high. The ball force acts 8 cm away from a rigid constraint. From a force balance:

$$M = -\int_{areabrass} \sigma_b * t * dA - \int_{areaAl} \sigma_{Alb} * t * dA$$

where M is the moment on the beam, σ represents stress, and t represents the distance from the neutral axis.

Applying Hooke's law ( $\sigma = E\varepsilon$ ) and the assumption that plane sections will remain plane $(\varepsilon = \frac{t}{\rho})$ , we see that

$$M = \frac{-1}{\rho} * (E_B \int_{areabrass} t^2 dA + E_{Al} \int_{areaAl} t^2 dA)$$

$$Mt = \varepsilon(E_B I_B + E_{Al} I_{Al}) = \frac{\sigma_x}{E_x}(E_B I_B + E_{Al} I_{Al})$$

where I is the second moment of area.

Now we can solve for the stresses within the two materials:

$$\sigma_x = \frac{M * t * E_x}{(E_B I_B + E_{Al} I_{Al})}$$

where x represents the material in question.

We find that $\sigma_{brass}$ = 75.8 MPa and that $\sigma_{Al}$ = 53 MPa . The yield strengths of these two materials are 100 MPa and 270 MPa, respectively. The factor of safety for the brass component is 1.3 and the factor of safety for the aluminum is 5.1.

It would be advisable to increase the safety factor for the brass – but failure is unlikely because of slippage; the goalie would slip at the wheels and rotate before the stresses in the bumper ever reached critical levels.

## Collision Analysis

It is the goal of this section to quantify ball-goalie collisions, and to show that these collisions will not result in a goal being scored. We analyze collisions in two cases: Dead-on collision, where the ball directly hits the front of the robot, and rod collision, where the ball hits the rod near the end.

Assumptions will be made about the mass of the robot and ball, and the coefficients of static and kinetic friction. The assumptions are:

Mass of robot $m_r$= 1.5 kg
Mass of ball $m_b$ = .046 kg
Coefficient of static friction $\mu_s$ = 0.6
Coefficient of kinetic friction $\mu_k$ = 0.5
Robot body and ball are inelastically rigid

Further assumptions are made in the analysis. These assumptions do degrade the accuracy of the calculations to a certain degree. However, because we want ballpark figures these assumptions are ok.

## Dead-On Collision

- Ball hits the robot directly in front
- No rotational torque exists around the wheels.

PROBLEM 1: Find velocity of ball ($v_{slip}$) that just causes the robot to slip.
ASSUMPTIONS: After ball hits the goalie, the ball comes to a complete stop. This change of momentum exerts constant force $F_{ave}$ on the robot.

## CALCULATIONS:

Impulse on robot,

$$I = \Delta p$$
$$I = m_b v_{slip}$$
$$m_b v_{slip} = F_{ave} t$$
$$\mathbf{F}_{ave} = \frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}}$$

For the robot to be just in slip,

$$F_{ave} = F_s$$

$$\frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}} = \mu_s \mathbf{N}$$

By assuming a safe value of t = 0.01s, we can solve for $v_{slip}$:

$$v_{slip} = \frac{9.8 m_r \mu_s t}{m_b}$$
$$v_{slip} = 1.8\ ms^{-1}$$

We do not expect many balls to be of a velocity of more than 1.8 m/s. Thus, we do not expect the goalie to slip many times.

PROBLEM 2: If the ball IS going at faster than $v_{slip}$, find approximately the distance the goalie will slip.
ASSUMPTIONS: See problem 1. A ball with velocity $v_{slip}$ heading dead-on will just cause a ball to slip.

## CALCULATIONS:

According to assumptions, a ball with kinetic energy:

$$\mathbf{E}_{slip} = \frac{\mathbf{m}_b \mathbf{v}^2_{slip}}{\mathbf{2}}$$

will just cause the robot to slip. A ball with more kinetic energy, however, will cause the robot to slide distance d. The energy transfer required to make the robot move is:

$$E_{move} = E_{total} - E_{slip}$$

$$= (1/2)(m_b v^2 - m_b v_{slip}^2)$$

This energy is then absorbed by friction as the robot moves a distance d.

$$E_{move} = F_k d$$

Solving for d:

$$d = \frac{\frac{1}{2}(\mathbf{m}_b \mathbf{v}^2 - \mathbf{m}_b \mathbf{v}_{slip}^2)}{\mathbf{9.8m}_r \mu_k}$$

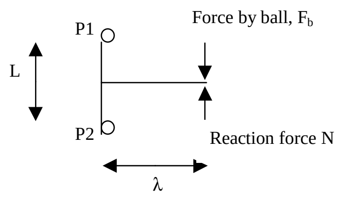
Using the previously calculated $v_{slip}$, we find the slip distance of the robot if an incoming ball has velocity v = 2.5 ms$^{-1}$:

$$d = 1.0 \text{ cm}$$

This is a safe distance to slide -- a distance of more than 10 cm is needed before a goal is scored. We conclude that the goalie design will not suffer from slip problems in a head-on collision.

## Ball-Bumper Collision and Torque Analysis

There are concerns that if the ball hits near the end of the extended bumper, torqueing will cause the robot to spin and let the ball through to the goal. When the ball hits the end of the rod, there are two possible pivot points corresponding to the wheels, labeled P1 and P2 in the diagram below.



The figure shows possible pivot points P1 and P2 corresponding to the two wheels, as well as force exerted by ball near edge of rod.

The torque exerted by Fb will be the same on P1 and P2. Thus, which pivot point will be the actual pivot point will depend on the weight on P1 or P2. For our analysis, we assume that the mass of robot $m_r$ is equally distributed over P1 and P2, but the pivot is P2 (due to the weight of the rod system on P2).

PROBLEM 3: Find $v_{slip}$, velocity of ball that just causes the robot to slip and spin on pivot P2.
ASSUMPTIONS: Contact time between ball and bumper t = 0.01 s.

CALCULATIONS:

$$\Delta p = F_b t$$

$$\mathbf{F}_b = \frac{\mathbf{m}_b \mathbf{v}_{slip}}{\mathbf{t}}$$

For the situation where robot is about to slip, sum of torques around P2 is 0:

$$\frac{\mathbf{m}_b \mathbf{v}_{slip} \lambda}{\mathbf{t}} = \frac{\mu_s \mathbf{NL}}{\mathbf{2}}$$

Solve for $v_{slip}$:

$$V_{slip} = \frac{9.8 m_r \mu_s L t}{2 m_b \lambda}$$

Using a safe assumption of t = 0.01s, and L=12.5 cm and λ=8 cm, we find:

$$v_{slip} = 1.4\ ms^{-1}$$

We note that in the torqueing situation, a smaller ball velocity can result in slip. This is a potentially dangerous situation. To curb this, we will try to maximize wheelbase length L, maximize coefficient of static friction, maximize contact time t by soft-padding the rod, and use pre-emptive motion of the goalie.

### 12.3.4 Drive Train

#### Motor Selection

Other parts of the R&D goalie that are different from those in the field payers include the drive motors. Based on the concept of traveling a minimum amount, the R&D goalie did not require motors capable of reaching a top speed of 2m/s. However, the ability to accelerate quickly was necessary. Hence the need for high-torque motors that are different from those used in the field players. The calculations are very similar to those done for the sliding mechanism and the field player motors:

| | |
|---|---|
| Manufacturer | Maxon Motors |
| Model | 232693352236200 |
| Price | $132.75 |
| **Desired acceleration ($m/s^2$)** | **5** |
| Mass of robot (kg) | 1.5 |
| Wheel radius (m) | 0.03 |
| Force required at wheels (N) | 7.5 |
| Torque required per axle (Nm) | 0.1125 |
| Gear ratio | 7 |
| Predicted gearhead efficiency | 0.7 |
| Stall Torque (mNm) | 42.5 |
| **Actual Acceleration ($m/s^2$)** | **6.6** |

**Figure 34. Drive Motor Calculation Highlights**

#### Gearing

A gear ratio of 7 was chosen give to the proper acceleration. The pinions and spurs chosen are brass and polyethylene, respectively. Brass, the harder material of the two materials, was chosen for the pinion because the pinion must be permenantly and rigidly mounted. It is the polyethylene spur that will wear and fatigue. While a failure at the spur gear in competition would require replacing the entire wheel assembly, this is a relatively easy process if spare wheels have been assembled. Replacing the pinion would not be possible at competition. The R&D goalie uses wheel identical to those used by the field players.

### 12.3.5 Appendices

A) Parts and Manufacturers List

B) Recommendations for the Future

C) Machine Drawings

# Section 13 Appendix:

## 13.1 Electrical

### 13.1.1 Price list

### 13.1.2 Node list for the Freedom16-mite board

The text in bold are the modules that are used.

| device 1 | Module | Optional fuction | I/O | Port.pin | device 2 | Port name | Pin |
|---|---|---|---|---|---|---|---|
| Freedom16-mite | ***DS** | PE6 | | CN1.1 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | ***BERR** | | | CN1.2 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **GND** | | | CN1.3 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **DSCLK** | | | CN1.4 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **GND** | | | CN1.5 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **FREEZE** | | | CN1.6 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | ***RESET** | | | CN1.7 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **DS1** | IPIPE1 | | CN1.8 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **+5V** | | | CN1.9 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | **DSO** | IPIPE0 | | CN1.10 | COMPUTER | DEBUG CABLE | |
| Freedom16-mite | +5V | | | CN2.1 | N/C | | |
| Freedom16-mite | GND | | | CN2.2 | N/C | | |
| Freedom16-mite | PF7 | *IRQ7 | | CN2.3 | N/C | | |
| Freedom16-mite | PF6 | ***IRQ6** | I | CN2.4 | FM-RPC-XXX | *RX-request | 8 |
| Freedom16-mite | **PF5** | *IRQ5 | I | CN2.5 | button 4 | | |
| Freedom16-mite | **PF4** | *IRQ4 | I | CN2.6 | button 3 | | |
| Freedom16-mite | **PF3** | *IRQ3 | I | CN2.7 | button 2 | | |
| Freedom16-mite | **PF2** | *IRQ2 | I | CN2.8 | button 1 | | |
| Freedom16-mite | **PF1** | *IRQ1 | O | CN2.9 | FM-RPC-XXX | *TX-request | 6 |
| Freedom16-mite | **PF0** | MODCLK | O | CN2.10 | FM-RPC-XXX | *RESET | 10 |
| Freedom16-mite | GND | | | CN2.11 | N/C | | |
| Freedom16-mite | **PE7** | SIZ1 | O | CN2.12 | TRAN1 | KICKER | |
| Freedom16-mite | **PE6** | SIZ0 | I | CN2.13 | MC14070B | Direction L | 3 |
| Freedom16-mite | **PE5** | *AS | I | CN2.14 | MC14070B | Direction R | 11 |
| Freedom16-mite | **PE4** | *DS | O | CN2.15 | L298 right | In 1 | 5,12 |
| Freedom16-mite | **PE2** | *AVEC | O | CN2.16 | L298 right | In 2 | 7,10 |
| Freedom16-mite | **PE1** | *DSACK1 | O | CN2.17 | L298 left | In 1 | 5,12 |
| Freedom16-mite | **PE0** | *DSACK0 | O | CN2.18 | L298 left | In 2 | 7,10 |
| Freedom16-mite | +5V | | | CN2.19 | N/C | | |
| Freedom16-mite | GND | | | CN2.20 | N/C | | |
| Freedom16-mite | W/*R | | | CN3.1 | N/C | | |
| Freedom16-mite | R/*W | | | CN3.2 | N/C | | |
| Freedom16-mite | *RESET | | | CN3.3 | N/C | | |
| Freedom16-mite | A4 | | | CN3.4 | N/C | | |
| Freedom16-mite | A3 | | | CN3.5 | N/C | | |
| Freedom16-mite | A2 | | | CN3.6 | N/C | | |
| Freedom16-mite | A1 | | | CN3.7 | N/C | | |
| Freedom16-mite | A0 | | | CN3.8 | N/C | | |
| Freedom16-mite | D15 | | | CN3.9 | N/C | | |
| Freedom16-mite | D14 | | | CN3.10 | N/C | | |
| Freedom16-mite | D13 | | | CN3.11 | N/C | | |
| Freedom16-mite | D12 | | | CN3.12 | N/C | | |
| Freedom16-mite | D11 | | | CN3.13 | N/C | | |
| Freedom16-mite | D10 | | | CN3.14 | N/C | | |
| Freedom16-mite | D9 | | | CN3.15 | N/C | | |
| Freedom16-mite | D8 | | | CN3.16 | N/C | | |
| Freedom16-mite | *CS10 | ECLK | | CN3.17 | N/C | | |
| Freedom16-mite | *CS2 | *BGACK | | CN3.18 | N/C | | |
| Freedom16-mite | *CS3 | PC0 | | CN3.19 | N/C | | |
| Freedom16-mite | **+5V** | | | CN3.20 | LCD1002-20x4 | Power +5 | 2,5 |

| device 1 | Module | Optional fuction | I/O | Port.pin | device 2 | Port name | Pin |
|---|---|---|---|---|---|---|---|
| Freedom16-mite | **GND** | | | CN3.21 | LCD1002-20x4 | GND,R/W | 1,5 |
| Freedom16-mite | *CS4 | PC1/**LCD_RS** | | CN3.22 | LCD1002-20x4 | Reset | 4 |
| Freedom16-mite | *CS5 | PC2/**LCD_E** | | CN3.23 | LCD1002-20x4 | Enable | 6 |
| Freedom16-mite | *CS6 | PC3/**LCD_D1**/KPO | | CN3.24 | LCD1002-20x4 | DB5 | 12 |
| Freedom16-mite | *CS7 | PC4/**LCD_D0**/KP1 | | CN3.25 | LCD1002-20x4 | DB4 | 11 |
| Freedom16-mite | *CS8 | PC5/**LCD_D3**/KP2 | | CN3.26 | LCD1002-20x4 | DB7 | 14 |
| Freedom16-mite | *CS9 | PC6/**LCD_D2**/KP3 | | CN3.27 | LCD1002-20x4 | DB6 | 13 |
| Freedom16-mite | PCS3 | **PQS6**/KP4 | I | CN3.28 | FM-RPC-XXX | *TX-accept | 7 |
| Freedom16-mite | PCS2 | **PQS5**/KP5 | O | CN3.29 | FM-RPC-XXX | *RX-accept | 9 |
| Freedom16-mite | PCS1 | **PQS4**/KP6 | I | CN3.30 | FM-RPC-XXX | *RX-request | 8 |
| Freedom16-mite | PCS0 | *SS/**PQS3**/KP7 | I/O | CN3.31 | FM-RPC-XXX | DATA bit 3 | 5 |
| Freedom16-mite | SCK | **PQS2** | I/O | CN3.32 | FM-RPC-XXX | DATA bit 2 | 4 |
| Freedom16-mite | MOSI | **PQS1** | I/O | CN3.33 | FM-RPC-XXX | DATA bit 1 | 3 |
| Freedom16-mite | MISO | **PQS0**/PWS0 | I/O | CN3.34 | FM-RPC-XXX | DATA bit 0 | 2 |
| Freedom16-mite | +5V | | | CN4.1 | N/C | | |
| Freedom16-mite | GND | | | CN4.2 | N/C | | |
| Freedom16-mite | PCLK | | | CN4.3 | N/C | | |
| Freedom16-mite | **PWMB** | | O | CN4.4 | L298 right | enable | 6,11 |
| Freedom16-mite | **PWMA** | | O | CN4.5 | L298 left | enable | 6,11 |
| Freedom16-mite | PAI | | | CN4.6 | N/C | | |
| Freedom16-mite | IC4/OC5 | PGP7 | | CN4.7 | N/C | | |
| Freedom16-mite | OC4 | **PGP6** | 0 | CN4.8 | LED RED2 | | |
| Freedom16-mite | OC3 | **PGP5** | 0 | CN4.9 | LED RED1 | | |
| Freedom16-mite | OC2 | **PGP4** | 0 | CN4.10 | LED YELLOW | | |
| Freedom16-mite | OC1 | **PGP3** | 0 | CN4.11 | LED GREEN | | |
| Freedom16-mite | **IC3** | PGP2 | I | CN4.12 | HALL | | 3 |
| Freedom16-mite | **IC2** | PGP1 | I | CN4.13 | ENCODER right | | 3 |
| Freedom16-mite | **IC1** | PGP0 | I | CN4.14 | ENCODER left | | 3 |
| Freedom16-mite | HC16_TX | | | CN5.1 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | N/C | | | CN5.2 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | 232_Tx | | | CN5.3 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | PC_RTS | SER_INT | | CN5.4 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | 232_Rx | | | CN5.5 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | PC_CTS | SER_ACK | | CN5.6 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | PC_DTR | RESET | | CN5.7 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | N/C | | | CN5.8 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | GND | | | CN5.9 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | HC16_RX | | | CN5.10 | COMPUTER | SERIAL CABLE | |
| Freedom16-mite | **GND** | | I | CN6.1 | LM2940 | GND | 2 |
| Freedom16-mite | **+5V** | | I | CN6.2 | LM2940 | +5V | 3 |
| Freedom16-mite | **GND** | | I | CN6.3 | LM2940 | GND | 2 |
| Freedom16-mite | +5V | | | CN7.1 | N/C | | |
| Freedom16-mite | GND | | | CN7.2 | N/C | | |
| Freedom16-mite | AI0 | **PADA0** | I | CN7.3 | DIP 1 | | |
| Freedom16-mite | AGND | | | CN7.4 | N/C | | |
| Freedom16-mite | AI1 | **PADA1** | I | CN7.5 | DIP 2 | | |
| Freedom16-mite | AGND | | | CN7.6 | N/C | | |
| Freedom16-mite | AI2 | **PADA2** | I | CN7.7 | DIP 3 | | |
| Freedom16-mite | AGND | | | CN7.8 | N/C | | |
| Freedom16-mite | AI3 | **PADA3** | I | CN7.9 | DIP 4 | | |
| Freedom16-mite | AGND | | | CN7.10 | N/C | | |
| Freedom16-mite | AI4 | PADA4 | | CN7.11 | N/C | | |
| Freedom16-mite | AGND | | | CN7.12 | N/C | | |
| Freedom16-mite | AI5 | PADA5 | | CN7.13 | N/C | | |
| Freedom16-mite | AGND | | | CN7.14 | N/C | | |
| Freedom16-mite | AI6 | PADA6 | | CN7.15 | N/C | | |
| Freedom16-mite | AGND | | | CN7.16 | N/C | | |
| Freedom16-mite | **AI7** | PADA7 | I | CN7.17 | RESISTOR DIVIDER FROM BATTERY | | |
| Freedom16-mite | AGND | | | CN7.18 | RESISTOR DIVIDER GND | | |
| Freedom16-mite | +5V_ANALOG | | | CN7.19 | N/C | | |
| | | | | | | | |

### 13.1.3 Radiometrix Transponder connections

| device 1 | Port 1 name | Pin | device 2 | Port 2 name | Port.pin |
|---|---|---|---|---|---|
| FM-RPC-XXX | GND | 1 | DGND | | |
| FM-RPC-XXX | D0 | 2 | Freedom16-mite | PQS0 | CN3.34 |
| FM-RPC-XXX | D1 | 3 | Freedom16-mite | PQS1 | CN3.33 |
| FM-RPC-XXX | D2 | 4 | Freedom16-mite | PQS2 | CN3.32 |
| FM-RPC-XXX | D3 | 5 | Freedom16-mite | PQS3 | CN3.31 |
| FM-RPC-XXX | *TX-request | 6 | Freedom16-mite | PF1 | CN2.9 |
| FM-RPC-XXX | *TX-accept | 7 | Freedom16-mite | PQS6 | CN3.28 |
| FM-RPC-XXX | *RX-request | 8 | Freedom16-mite | PQS4, *IRQ6 | CN3.30, CN2.4 |
| FM-RPC-XXX | *RX-accept | 9 | Freedom16-mite | PQS5 | CN3.29 |
| FM-RPC-XXX | *RESET | 10 | Freedom16-mite | PF0 | CN2.10 |
| FM-RPC-XXX | Vcc | 11 | +5v | | |
| FM-RPC-XXX | GND | 12 | DGND | | |

### 13.1.4 Motor encoders pin connections

| device 1 | Port 1 name | color | Pin | device 2 | Port 2 name | Pin |
|---|---|---|---|---|---|---|
| Motor encoder right | GND | Brown | 1 | GND | | |
| Motor encoder right | Index | none | 2 | N/C | | |
| Motor encoder right | Channel A | Red | 3 | Freedom16-mite | IC2 | CN4.13 |
| Motor encoder right | Channel A | Red | 3 | MC14070B | | 2,4 |
| Motor encoder right | Vcc | Orange | 4 | +5V | | |
| Motor encoder right | Channel B | Yellow | 5 | CD4013BE | | 5 |
| Motor encoder left | GND | Brown | 1 | GND | | |
| Motor encoder left | Index | none | 2 | N/C | | |
| Motor encoder left | Channel A | Red | 3 | Freedom16-mite | IC1 | CN4.14 |
| Motor encoder left | Channel A | Red | 3 | MC14070B | | 9,13 |
| Motor encoder left | Vcc | Orange | 4 | +5V | | |
| Motor encoder left | Channel B | Yellow | 5 | CD4013BE | | 9 |

### 13.1.5 Motor controllers pin layout

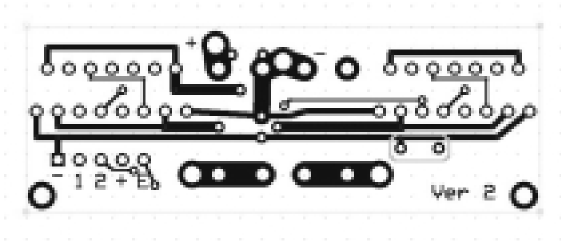
| device 1 | Port 1 name | Pin | device 2 | Port 2 name | Pin | device 3 | Port 3 name | Pin |
|---|---|---|---|---|---|---|---|---|
| L298 right | Current sensing A | 1 | | | Analog GND | | | |
| L298 right | output 1 | 2 | L298 right | Output 4 | 14 | | | |
| L298 right | output 2 | 3 | L298 right | Output 3 | 13 | | | |
| L298 right | Supply Vs | 4 | | unregulated battery and a non-inducting 100nF Cap to AGND | | | | |
| L298 right | Input 1 | 5 | L298 right | Input 4 | 12 | Freedom16-mite | PE4 | CN2.15 |
| L298 right | Enable A | 6 | L298 right | Enable B | 11 | Freedom16-mite | PWMB | CN4.4 |
| L298 right | Input 2 | 7 | L298 right | Input 3 | 10 | Freedom16-mite | PE2 | CN2.16 |
| L298 right | GND & tab | 8 | | | Digital GND | | | |
| L298 right | logic Vss | 9 | | 5 and a 100nF Cap to DGND | | | | |
| L298 right | Input 3 | 10 | L298 right | Input 2 | 7 | Freedom16-mite | PE2 | CN2.16 |
| L298 right | Enable B | 11 | L298 right | Enable A | 6 | Freedom16-mite | PWMB | CN4.4 |
| L298 right | Input 4 | 12 | L298 right | Input 1 | 5 | Freedom16-mite | PE4 | CN2.15 |
| L298 right | Output 3 | 13 | L298 right | output 2 | 3 | | | |
| L298 right | Output 4 | 14 | L298 right | output 1 | 2 | | | |
| L298 right | Current sensing B | 15 | | | Analog GND | | | |

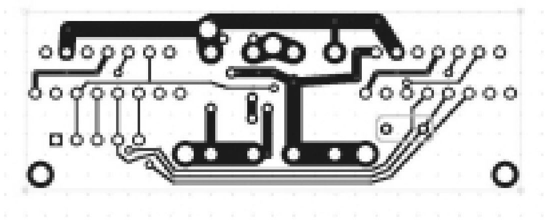
| device 1 | Port 1 name | Pin | device 2 | Port 2 name | Pin | device 3 | Port 3 name | Pin |
|---|---|---|---|---|---|---|---|---|
| L298 left | Current sensing A | 1 | | | Analog GND | | | |
| L298 left | output 1 | 2 | L298 left | Output 4 | 14 | | | |
| L298 left | output 2 | 3 | L298 left | Output 3 | 13 | | | |
| L298 left | Supply Vs | 4 | | unregulated battery and a non-inducting 100nF Cap to AGND | | | | |
| L298 left | Input 1 | 5 | L298 left | Input 4 | 12 | Freedom16-mite | PE1 | CN2.17 |
| L298 left | Enable A | 6 | L298 left | Enable B | 11 | Freedom16-mite | PWMA | CN4.5 |
| L298 left | Input 2 | 7 | L298 left | Input 3 | 10 | Freedom16-mite | PE0 | CN2.18 |
| L298 left | GND & tab | 8 | | | Digital GND | | | |
| L298 left | logic Vss | 9 | | 5 and a 100nF Cap to DGND | | | | |
| L298 left | Input 3 | 10 | L298 left | Input 2 | 7 | Freedom16-mite | PE0 | CN2.18 |
| L298 left | Enable B | 11 | L298 left | Enable A | 6 | Freedom16-mite | PWMA | CN4.5 |
| L298 left | Input 4 | 12 | L298 left | Input 1 | 5 | Freedom16-mite | PE1 | CN2.17 |
| L298 left | Output 3 | 13 | L298 left | output 2 | 3 | | | |
| L298 left | Output 4 | 14 | L298 left | output 1 | 2 | | | |
| L298 left | Current sensing B | 15 | | | Analog GND | | | |

### 13.1.6 Board layouts

**Motor controllers:**

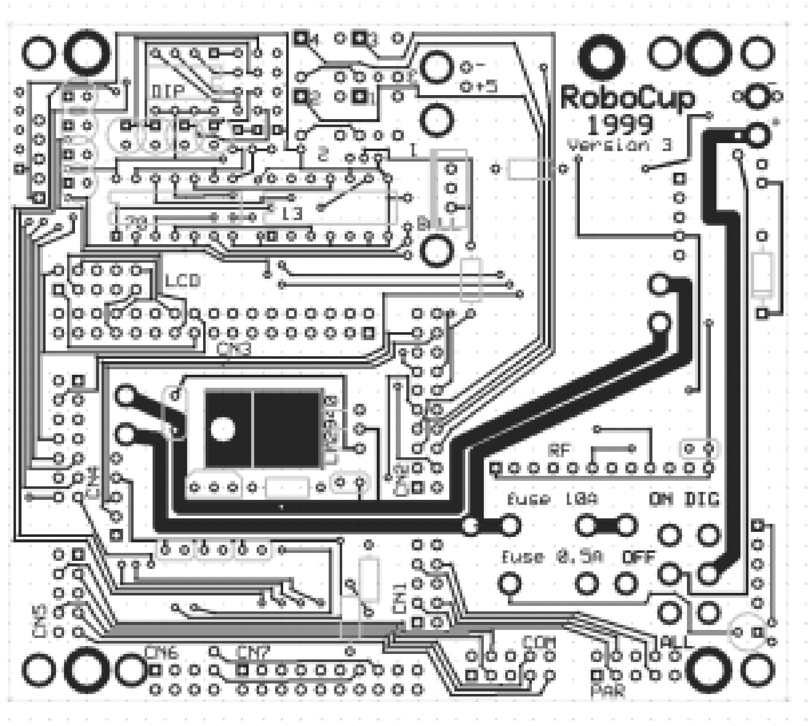
Top of board



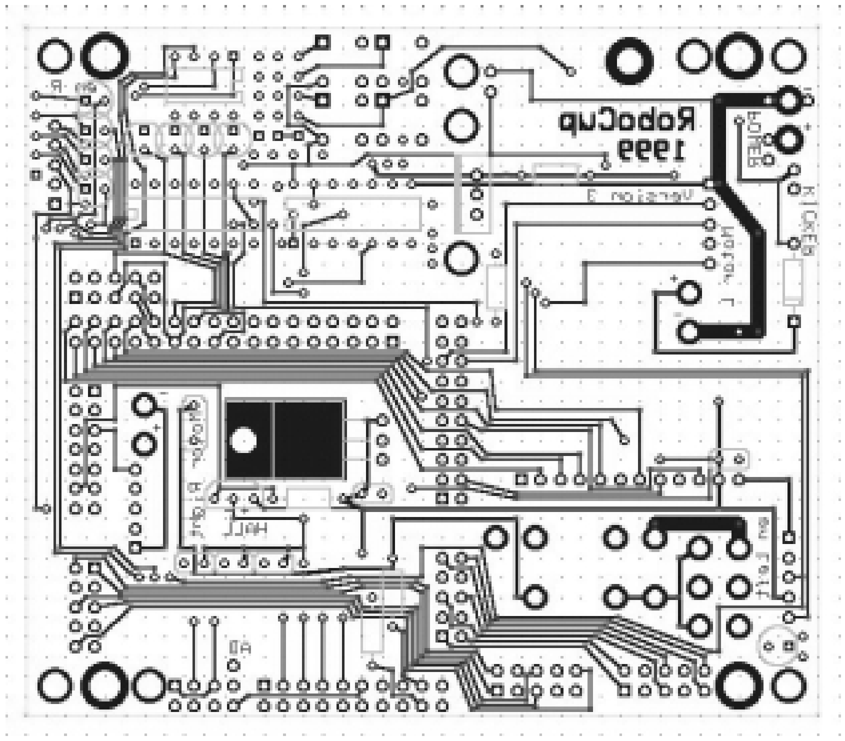


Bottom of board

**Main Board**



Top of board



Bottom of board