Indiana University-Purdue University Fort Wayne
Department of Engineering

ENGR 410-ENGR411
Capstone Senior Design Project
Report #2

Project Title: Kicking Mechanism for the Pioneer 3-DX

Team Members: Mr. Charles Craft (EE)
               Mr. Mark Macki (ME)
               Mr. Francois Mikobi (EE)
               Mr. Landry Nzudie (CmpE, EE)
               Mr. Miguel Tostado (ME)

Faculty Advisors: Dr. Yanfei Liu (EE)
                  Dr. Jiaxin Zhao (ME)

Date: April 18, 2008
0. TABLE OF CONTENTS

0. Table of Contents ........................................................................................................... 1
1. Acknowledgements ......................................................................................................... 4
2. Abstract .......................................................................................................................... 4
3. DETAILED FINAL DESIGN ......................................................................................... 6
   3.1. KICKING MECHANISM ......................................................................................... 6
       3.1.1 DYNAMIC ANALYSIS ................................................................................. 8
   3.2 PNEUMATIC SYSTEM ............................................................................................. 10
       3.2.1. PNEUMATIC CYLINDER .......................................................................... 11
       3.2.2. SOLENOID VALVE .................................................................................. 11
       3.2.3. TANK ANALYSIS ..................................................................................... 11
       3.2.4. COMPRESSIBLE FLOW ............................................................................. 14
       3.2.5. TUBING ..................................................................................................... 15
       3.2.6. Cylinder Response ....................................................................................... 16
   3.3 STRESS ANALYSIS ................................................................................................. 18
       3.3.1 KICKING PLATE ........................................................................................... 18
       3.3.2 MAXIMUM AXIAL FORCE ON PISTON ROD ............................................ 20
       3.3.3 MAXIMUM LATERAL FORCE ON EXTENDED PISTON ROD ................. 20
       3.3.4 CYLINDER BOLTS ....................................................................................... 21
       3.3.5 STRESS CONCENTRATION ON PIONEER 3-DX .................................... 22
       3.3.6 TANK ............................................................................................................ 23
       3.3.7 TANK BRACKETS ......................................................................................... 24
   3.4 SOLENOID DRIVER ................................................................................................. 25
       3.4.1 LIMITATIONS ............................................................................................... 25
       3.4.2 THERMAL ANALYSIS AND POWER CALCULATIONS ............................ 25
       3.4.3 ADDITIONAL ANALYSIS ............................................................................. 26
5.3. Mechanical testing.............................................................................................. 63
  5.3.1. MECHANICAL TESTING PROCEDURE ................................................... 63
  5.3.2. Mechanical components........................................................................... 63
5.4. Software procedure ............................................................................................ 69
  5.4.1. System testing procedure ........................................................................ 70
6. EVALUATION AND RECOMMENDATION .......................................................... 71
  6.1. EVALUATION .............................................................................................. 71
  6.2. RECOMMENDATION .................................................................................. 72
7. CONCLUSION ....................................................................................................... 73
8. REFERENCES ........................................................................................................ 74
9. List of figures ........................................................................................................... 75
10. list of tables .......................................................................................................... 77
11. APPENDICES ...................................................................................................... 78
  11.1. Test_Serial.cpp .......................................................................................... 78
  11.2. serialIntrf.c .............................................................................................. 79
  11.3. FOLLOW BALL and KICK ........................................................................ 84
  11.3. Pipe Flow Program.ees ............................................................................ 90
  11.4. Tank_analysis.m ....................................................................................... 92
  11.5. Cylinder Response ..................................................................................... 94
  11.6. CAD DRAWINGS ..................................................................................... 96
    11.6.1. KiCKER ................................................................................................ 96
    11.6.2. Tank Brackets .................................................................................... 97
1. ACKNOWLEDGEMENTS

We wish to thank our advisors Dr. Liu and Dr. Zhao for their comments and recommendations. We would also like to thank Bob, the PCB board technician in the Engineering Technology and Computer Science department for cutting our board and assisting us in some repairs and John Mitchell for machining the kicker. Finally, we would like to thank our sponsor Raytheon for providing the Pioneer 3-DX base as well as the funding for the kicking mechanism.

2. ABSTRACT

Raytheon has provided a Pioneer 3-DX robot base for the development of a RoboCup Middle Size League robot soccer player. RoboCup is an international organization putting two robotic soccer teams against each other. Our team’s mission is to design and build a kicking mechanism and a microcontroller based unit to interface and control the kicker with the provided base robot. When completed, the final robot should be programmed to locate an orange, size 5 FIFA ball on a green field, approach the ball, orient the ball towards the goal, and finally kick it in that direction. Several ideas of designs were initially generated; each of them was evaluated in order to generate the most optimal final design. The mechanical kicker of our final design will consist of two pneumatic cylinders utilizing compressed nitrogen, which will be engaged and controlled by a PIC microcontroller based interface unit. The robot should meet the RoboCup rules applicable for a single Middle Size robot soccer player.
Table 2.1: List of Variables Used

<table>
<thead>
<tr>
<th><strong>Thermodynamic Variables</strong></th>
<th><strong>Fluid Mechanics Variables</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_p$ specific heat at constant pressure</td>
<td>$c$ speed of sound</td>
</tr>
<tr>
<td>$C_v$ specific heat at constant volume</td>
<td>$M$ Mach number</td>
</tr>
<tr>
<td>$k$ ratio of specific heats</td>
<td>$P_0$ Stagnation Pressure</td>
</tr>
<tr>
<td>$m$ mass</td>
<td>$Q$ volumetric flow rate</td>
</tr>
<tr>
<td>$P$ pressure</td>
<td>$T_0$ Stagnation Temperature</td>
</tr>
<tr>
<td>$P_c$ critical pressure</td>
<td>$P$ density</td>
</tr>
<tr>
<td>$P_r$ reduced pressure</td>
<td></td>
</tr>
<tr>
<td>$R$ gas constant</td>
<td></td>
</tr>
<tr>
<td>$T$ temperature</td>
<td></td>
</tr>
<tr>
<td>$T_c$ critical temperature</td>
<td></td>
</tr>
<tr>
<td>$T_r$ reduced temperature</td>
<td></td>
</tr>
<tr>
<td>$V$ volume</td>
<td></td>
</tr>
<tr>
<td>$x$ quality</td>
<td></td>
</tr>
<tr>
<td>$Z$ compressibility factor</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dynamics, Kinematics, and Mechanics of Materials Variables</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$ acceleration</td>
</tr>
<tr>
<td>$A_c$ cross sectional area</td>
</tr>
<tr>
<td>$E$ modulus of elasticity</td>
</tr>
<tr>
<td>$F$ force</td>
</tr>
<tr>
<td>$I$ moment of inertia</td>
</tr>
<tr>
<td>$k$ spring constant</td>
</tr>
<tr>
<td>$L$ length</td>
</tr>
<tr>
<td>$M$ moment</td>
</tr>
<tr>
<td>$r$ radius</td>
</tr>
<tr>
<td>$S_y$ yield strength</td>
</tr>
<tr>
<td>$T$ torque</td>
</tr>
<tr>
<td>$t$ time</td>
</tr>
<tr>
<td>$t$ thickness</td>
</tr>
<tr>
<td>$v$ velocity</td>
</tr>
<tr>
<td>$V$ velocity</td>
</tr>
<tr>
<td>$x$ displacement</td>
</tr>
<tr>
<td>$\alpha$ angular acceleration</td>
</tr>
<tr>
<td>$\theta$ angle</td>
</tr>
<tr>
<td>$\sigma$ normal stress</td>
</tr>
<tr>
<td>$\tau$ shear stress</td>
</tr>
<tr>
<td>$\omega$ angular velocity</td>
</tr>
</tbody>
</table>
3. DETAILED FINAL DESIGN

This section discusses a kicking mechanism for the Pioneer 3-DX robot. The kicker and associated interface were designed to conform to the RoboCup Middle Size League rules. The kicker is powered with twin pneumatic cylinders. The kicking mechanism will act as a plunger when both cylinders are actuated simultaneously. The kicker will act as a flipper when a single cylinder is fired, directing the ball away from the force. The pneumatic cylinders are controlled via a pair of 3/2 solenoid valves.

The electrical design of the control is divided in three main parts; the microcontroller interface, the driving circuit interface, and the power circuit. The microcontroller interface expands from the serial connection with the SH2 microcontroller of the robot—using a male to male DB9 cable—to the driving circuit for the actuators of the kicking mechanism; the Driving circuit drives that input from the microcontroller to the actuators of the kicker; and, the power circuit will take the power source from the robot and feed to all the electrical and mechanical elements of the kicker that will need to be powered.

3.1. KICKING MECHANISM

The most logical place to begin discussions is with the kicker itself. Several possibilities existed for the design of a kicker including: spring, motor, solenoid and pneumatic. After comparing pros and cons of these designs, we settled on implementing a pneumatic system. In addition, to having a system that can propel a ball forward in a consistently uniform manner we wanted to devise some way of adjusting the kick. We have designed a dual cylinder system that can not only kick the ball forward, in order to score, but also kick it at an angle, perhaps to pass to a teammate.

The two pneumatic cylinders can be fired individually or as a synchronized pair. The kicking plate will be attached via a pin to one of the cylinder and via a slider to the other cylinder, see Figure 4.1. The simple linkage shown, allows for a left kick when the right cylinder fires and a right kick when, alternatively, the left cylinder fires. The figure gives an approximation of the linkage before a kick and then after the left cylinder has been activated. The cylinders will be controlled via 3/2 solenoid valves. 3/2 solenoid valves have 3 ports and 2 operating positions. One port comes from the supply line, one port goes to the cylinder and one goes to exhaust. The two positions determine which output line is active. When kicking, the output to the cylinder is active, when idle the position is switched and air is allowed to exhaust.
The kicking mechanism allows for 2 degrees of freedom. In the figure the joints are numbered and circled, the links are numbered without circles. Gruebblar’s equation for calculating degrees of freedom is: $M=3(L-1)-2J$. Analyzing the linkage shows two degrees of freedom in the linkage: $M=3(4-1)-2(3.5) = 2$. In practice we would anticipate only 1 degree of freedom on any single kick, either both cylinders firing concurrently or 1 cylinder firing while the other holds.

Table 3.1: Link/Joint Description

<table>
<thead>
<tr>
<th>Links</th>
<th>Desc</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Left Plunger</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Right Plunger</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Kicking Plate</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Joints</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pin Joint</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Sliding full joint</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Sliding full joint</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Pin in slot, half joint</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Figure 3.1: Cylinders and kicking plate. Right diagram shows side kick.
3.1.1 DYNAMIC ANALYSIS

The aluminum kicking plate will measure 16.51 cm wide by 10.16 cm high with a thickness of 0.635 cm. At a density of 2700 kg/m³ the total mass of the plate (including brackets and screws) is 0.4 kg. The ball’s mass is 0.43 kg for a total mass of 0.83 kg. The cylinders being used create a force of 270 N at a pressure of 620 kPa apiece and they deliver a force of approximately 540 N when fired as a pair.

Analyzing a single firing cylinder we can predict the kinematics for the resulting kick. The kicking plate can be modeled as a link in pure rotation; Figure 3.2 displays the free body diagram.

![Figure 3.2: Free body diagram](image)

We can set up 3 equations summing the forces in the x, y direction and the moments around the secured pin.

\[
\sum F_y = m a_{gy} = F_{appl} - \mu F_{appl} \cos \theta \sin \theta
\]

\[
\sum F_x = m a_{gx} = \mu F_{appl} \cos^2 \theta
\]

\[
\sum M_\alpha = I \alpha = m R_x a_{gx} + m R_y a_{gy}
\]

In both equations for force there is a frictional force. This is caused by the contact in the cutout of the kicking plate. The plate will be made of still and the pin will be made of steel, which has a coefficient of friction, \( \mu \), approximately 0.6. While the angle of rotation is small the impact of friction is minimal. As the angle increases the magnitude of the frictional force also increases. The coefficient of friction can be reduced by utilizing a rolling pin, \( \mu = 0.001 \).

Moment of inertia for a thin plate is: \( I = \frac{1}{3} mL^3 \). The moment arms, \( R_x \) and \( R_y \), are also dependent on the angle. Plugging in the known values we can evaluate the 3 unknowns: \( \alpha, a_{gy}, a_{gx} \), so that they are dependent on the angle only.
When the angular acceleration is calculated it is possible to determine the instantaneous angular velocity via the following relationship:

\[
\alpha = \frac{d\omega}{dt}, \quad \omega = \frac{d\theta}{dt}, \quad \int \alpha d\omega = \int \alpha d\theta
\]

Integrating the equation above allows us to calculate the angular velocity of the kicking plate and cylinder at any point. Since the kicker is in full rotation we can calculate the instantaneous linear velocity at any point on the plate \( v = r \times \omega \). When analyzing the dynamics of the plate and ball, assumptions have to be made. The pair can be viewed as connected traveling together and the ball releasing at the end of the cylinder extension or the velocity of the plate can be evaluated and then calculating the energy transferred as a collision. We will assume that the plate travels alone and experiences an inelastic collision with the ball standing still when the cylinder is extended 3 of the 5 cm. Substituting into the equations we get a linear velocity at the midpoint of the kicker of 13.7 m/s. In the equations below \( M_1 \) and \( M_2 \) on the left are post-collision masses, on the right side they are pre-collision masses.

\[
M_1 V_{1f} + M_2 V_{2f} = (M_1 V_1 + M_2 V_2)
\]
\[
-V_{1f} + V_{2f} = C_R (V_1 - V_2)
\]

\( M_1 \) and \( M_2 \) represent the masses of the plate and ball respectively. The velocities displayed are initial and final velocities, with \( V_2 = 0 \). The coefficient of restitution describes the inelastic nature of a particular collision. This value can be found experimentally for a variety of objects. For a soccer ball we will use 0.64. Substituting the known values we obtain final speeds of the plate as -1.43 m/s and the ball as 6.7 m/s. The negative velocity of the plate indicates the recoil experienced.
3.2 PNEUMATIC SYSTEM

Figure 3.3: Schematic of Pneumatic System

Figure 3.4: CAD Drawing of Final Design
3.2.1. PNEUMATIC CYLINDER

The pneumatic cylinder will provide the force to propel the ball five meters. The force of the cylinder is

\[ F_{cylinder} = PA - F_{spring} - F_{friction} \]

The cylinders selected for the design are Parker P1A-S 025 SS-0050. They have a bore diameter of 25 mm. The cylinders have a maximum operating pressure of 10 bars. The spring constant is 285 N/m.

3.2.2. SOLENOID VALVE

The solenoid valve that was selected for this design is the Parker 753130122B. We will use two 3/2 valves. The flow coefficient for the valve is 0.11. The diameter of the orifice is 1/16 in.

3.2.3. TANK ANALYSIS

The tank chosen for the final design is a 48 cubic inch paint ball tank. It can be filled with either 3000 psi of air or nitrogen.

<table>
<thead>
<tr>
<th>Fluid</th>
<th>Air</th>
<th>Nitrogen</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.287</td>
<td>0.2968</td>
</tr>
<tr>
<td>(C_p)</td>
<td>1.004</td>
<td>1.042</td>
</tr>
<tr>
<td>(C_v)</td>
<td>0.717</td>
<td>0.745</td>
</tr>
<tr>
<td>(k)</td>
<td>1.400</td>
<td>1.399</td>
</tr>
</tbody>
</table>

In order to determine the volume of the tank needed, an analysis was performed using the ideal gas law.

\[ PV = mRT \]

For a real gas the above equation involves a compressibility factor, \(Z\)

\[ PV = mZRT \]
<table>
<thead>
<tr>
<th>m</th>
<th>mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>pressure</td>
</tr>
<tr>
<td>R</td>
<td>gas constant</td>
</tr>
<tr>
<td>T</td>
<td>temperature</td>
</tr>
<tr>
<td>V</td>
<td>volume</td>
</tr>
<tr>
<td>Z</td>
<td>compressibility factor</td>
</tr>
</tbody>
</table>

In the case of an ideal gas, the compressibility factor, $Z$, is unity. At a temperature of 300 K and for pressures below 8 MPa (1160 psi), the compressibility factor of nitrogen is unity. Using Figure 5 (Sontag, pg. 65) the compressibility factor reaches 1.2 at a pressure of 20.7 MPa (3000 psi).

![Figure 3.5: Compressibility Chart for Nitrogen](image)

The pressure regulator will can adjust the pressure supplied to the cylinders. The minimum number of times that the cylinders can kick the ball is determined by using a kicking pressure of 90 psi. If the tank pressure drops below 90 psi, the available tank pressure is used to kick the ball.

The initial mass of fluid in the tank is determined by the ideal gas law. The mass of fluid that is needed to kick the ball is

$$m_{\text{exhausted}} = \frac{Z P_{\text{kick}} V_{\text{exhausted}}}{RT}$$

The mass of the gas in the tank after each kick is
The pressure in the tank is then recalculated with the ideal gas law using the new mass.

The figure below displays the tank pressure in the tank if both cylinders are actuated at 90 psi until the tank is empty.

![Figure 3.6: Tank pressure after each kick for air](image-url)
Using a 48 cubic inch tank filled with nitrogen at an initial pressure 3000 psi of the ball can be kicked straight 160 times by actuating both cylinders at the same time at a pressure of 90 psi. If only one cylinder is actuated at a time the tank will supply enough pressure for about 340 kicks.

3.2.4. COMPRESSIBLE FLOW

An aspect that is important in the design of the pneumatic system is the velocity of the fluid. The Mach number, M, is the ratio of the actual velocity, \( V \), and the velocity of sound, \( c \).

\[
M = \frac{V}{c}
\]

The sonic velocity is,

\[
c = \sqrt{kRT}
\]

The critical pressure ratio is,

\[
\frac{P^*}{P_0} = \left( \frac{2}{k + 1} \right)^{\frac{k}{k - 1}}
\]

Where \( k \) is the ratio of specific heats.

During adiabatic isentropic flow the stagnation temperature, \( T_0 \) is constant. The stagnation temperature, \( T_0 \), and stagnation pressure, \( P_0 \), are calculated with the following equations.

\[
\frac{T_0}{T} = 1 + \frac{k - 1}{2} M^2
\]

\[
\frac{P_0}{P} = \left( 1 + \frac{k - 1}{2} M^2 \right)^{\frac{k}{k - 1}}
\]

If air enters a converging valve or nozzle and the initial back pressure ratio, \( P_b/P \), is less than the critical pressure ratio the initial flow is choked. The back pressure at the solenoid valve is atmospheric pressure. At the instant the ball is kicked the back pressure ratio is less than the critical pressure ratio and the velocity of the fluid at that point is Mach 1. Since the velocity at the throat of the valve is now known the mass flow rate can be computed. The mass flow rate is approximately 3.3 g/s at a gauge pressure of 620 kPa.
3.2.5. TUBING

In analyzing the flow through the tubing the following assumptions were made.

1. One dimensional flow
2. Steady uniform flow
3. Fully developed flow

The flow can be considered to be incompressible if the velocity is less than Mach 0.3. The velocity of the gas throughout the pipe system is less than Mach 0.3 except in the throat of the valve. The tubing consists of drawn rubber and brass. The absolute roughness, $e$, for drawn tubing is $1.5 \times 10^{-6}$ m. The Reynolds’ number, $Re$, is calculated using the following equation.

$$Re = \frac{\rho V D}{\mu}$$

Where $\rho$ is the density

$V$ is the velocity

$D$ is the diameter

$\mu$ is the dynamic viscosity

The friction factor, $f$, was calculated using the Colebrook equation.

$$\frac{1}{\sqrt{f}} = -2 \log \left( \frac{e}{3.7D} + \frac{2.51}{Re \sqrt{f}} \right)$$

The major head losses, $h_l$, were calculated using the following equation.

$$h_l = f \frac{L V^2}{D^2}$$

Where $L$ is the length.

The pressure drop was calculated using Bernoulli’s equation.

$$\frac{P_1}{\rho g} + z_1 + \frac{v_1^2}{2g} = \frac{P_2}{\rho g} + z_2 + \frac{v_2^2}{2g} + h_l + h_{leak}$$
Table 3.3: Pressure in pipe system

<table>
<thead>
<tr>
<th></th>
<th>Pressure (kPa abs)</th>
<th>Pressure (kPa gage)</th>
<th>Pressure (psig)</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regulator</td>
<td>722</td>
<td>621</td>
<td>90</td>
<td>0.01</td>
</tr>
<tr>
<td>Valve</td>
<td>708</td>
<td>607</td>
<td>88</td>
<td>1</td>
</tr>
</tbody>
</table>

The table above displays the pressure at the solenoid valve if the regulator is set to 90 psi if 3/8” nominal diameter pipe is used. There will only be a modest pressure loss in the tubing.

### 3.2.6. CYLINDER RESPONSE

In order to determine the response of the pneumatic cylinders two equations of motion were used. The first EOM is for the air flow.

\[
RC \frac{dp_o}{dt} + p_o = p_i
\]

Where 
- \( R \) is the flow resistance
- \( C \) is the capacitance of the pneumatic cylinder
- \( p_o \) is the pressure at the cylinder
- \( p_i \) is the pressure at the pressure regulator

The average resistance is calculated using the following equation.

\[
R = \frac{\overline{p_i} - \overline{p_o}}{\overline{m}}
\]

Where \( \overline{m} \) is the mass flow rate

The capacitance of the pressure vessel is calculated using the following equation.

\[
C = \frac{V}{nRT}
\]

Where
- \( V \) is the volume of the pneumatic cylinder
- \( n \) is a constant (1.4 for and adiabatic process using nitrogen)

The second EOM is for the ball.

\[
m\ddot{x} + b\dot{x} + kx = p_o A
\]

Where
- \( m \) is the mass of the ball, kicking plate, and piston
- \( b \) is the damping coefficient
- \( k \) is the spring constant
- \( A \) is the cross-sectional area of the cylinder
The sources of Coulomb damping have been neglected in the above equation. The only damping force that is considered is the viscous damping inside the cylinder. The damping coefficient was estimated from the velocities of the ball obtained during testing.

In order to determine a relationship between the input pressure and the displacement transfer functions of the two EOM were taken.

\[
\frac{P_2(s)}{P_1(s)} = \frac{1}{RCs + 1}
\]

\[
\frac{X(s)}{P_2(s)} = \frac{A}{ms^2 + bs + k}
\]

The two transfer functions were then multiplied together to eliminate the output pressure and determine the relationship between the displacement and the input pressure. The forcing function is assumed to be a step input.

\[
X(s) = \frac{P_2(s)}{s(0)(RCs + 1)(ms^2 + bs + k)}
\]

The displacement, velocity, and acceleration of the cylinder are shown below assuming an input pressure of 620 kPa (90 psig) with all the initial conditions equal to zero. At this pressure the valve will need to be open a minimum of 33 milliseconds for the cylinder to extend the entire 50 mm stroke. The solution to the displacement, velocity, and acceleration are included in the appendix.
Figure 3.7: Displacement, velocity, and acceleration of pneumatic cylinder. (b=150kg/s)

3.3 STRESS ANALYSIS

3.3.1 KICKING PLATE

Stresses can be calculated at any point in the kick as a function of the angle. Maximum bending stress occurs when the kick just starts and maximum normal stress occurs at the end of the kick.

\[
\text{Bending stress: } \sigma = \frac{Mc}{I} \quad \text{Normal stress: } \sigma = \frac{P}{A}
\]
The normal axial stress is in tension; however, it is small when compared to the bending stress. The maximum tensile stress occurs at the beginning of the kick where the total stress is 17.8 MPa. This is well below the yield strength for aluminum allow, 400 MPa and also below the allowable yield stress of 60% $S_y$. We can also predict the number of cycles until failure, where $S^f_c$ is the fatigue life 0.75 $S_u$ and $S_e$ is the endurance limit 0.45 $S_u$. For aluminum the ultimate strength, $S_u$, is 454 MPa.

$$b_x = \frac{1}{S_x} \log \frac{S^f_c}{S_e}$$

$$C = \log \left( \frac{S^f_c}{S_e} \right)$$

$$S^f_c = 10^{C \times N^f_c}$$

This predicts a life of $N=1.9 \times 10^{20}$ cycles.
The piston rod material is stainless steel. The modulus of elasticity for stainless steel is 193 GPa and a yield strength of 207 MPa. The rod diameter is 10 mm and the stroke is 50 mm.

The critical buckling load, $F_{cr}$, for the cylinder is computed using the following equation.

$$F_{cr} = \frac{\pi^2 EI}{(L')^2}$$

Where $E$ is the modulus of elasticity

$I$ is the moment of inertia

The piston rod has one fixed end and one free end. The effective length is $L' = 2L$. The buckling load is calculated to be 93 kN.

The maximum axial force was also checked using the following equation.

$$\sigma = \frac{P}{A}$$

The maximum axial force assuming a uniaxial load using the above equation is 16 kN. The maximum axial force on the piston rod is then 16 kN. The rod will not fail from an axial load.

3.3.3 MAXIMUM LATERAL FORCE ON EXTENDED PISTON ROD

Figure 3.9: Extended Cylinder Piston Rod
The maximum deflection, $\delta_{\text{max}}$, of a fixed-free rod shown in the figure above is

$$\delta_{\text{max}} = \frac{FL^3}{3EI}$$

The maximum stress due to $F$ occurs at the fixed end and is

$$\sigma = \frac{Mc}{l} = \frac{FLr}{\frac{\pi}{4}r^4}$$

Using the yield strength for stainless steel, the maximum lateral load that can be applied to the piston rod is 406 N. The maximum deflection is 0.166 mm.

---

### 3.3.4 CYLINDER BOLTS

The pneumatic cylinders are attached to the Pioneer-3DX with two brackets. Each bracket has two holes for bolts. The bolts holding the cylinders must be able to withstand the shear stress. In order to avoid failure from shear stress, $\tau_{\text{max}}$, the shear stress must be less than 40% of the yield strength.

$$\tau = \frac{4F}{\pi d_c^2} < 0.4S_y$$

Where $d_c$ is the crest diameter

The brackets used to attach the cylinders that have holes with a diameter of 6.5 mm. The brackets have a thickness of 5 mm. 1/4” UNF SAE grade 5 bolts have a proof strength of 85 ksi (580 MPa). The maximum allowable shear stress is 232 MPa. If the pneumatic cylinder is actuated at 1.0 MPa a force of about 500 N develops. Each bolt will need to withstand a shear force of 125 N. A 1/4” UNF bolt will experience a shear stress of 5.5 MPa and has a safety factor of 16.
3.3.5 STRESS CONCENTRATION ON PIONEER 3-DX

The holes on the cylinder brackets are spaced 40 mm apart. Using ¼” bolts will require eight ¼” diameter holes to be drilled into the Pioneer 3-DX. The Pioneer 3-DX manual states that the body material is aluminum. Pure aluminum has a yield strength of 17 MPa. Using the figure above b and c are both 20 mm. d in 6 mm. The stress concentration factor is 3.5. The nominal stress is

\[ \sigma_{\text{nom}} = K_c \frac{F}{A} = K_c \frac{F}{(b-d)h} \]

The minimum thickness of the Pioneer 3-DX needs to be at least 1.8 mm. The thickness of the Pioneer 3-DX was measured to be 0.075 in or 1.905 mm. The preceding analysis assumed that the body material is pure aluminum. More than likely the material is an aluminum alloy. If it is assumed to be 6061 aluminum the minimum thickness will only need to be 0.35 mm.

The paintball tank being used has a maximum operating pressure of 3000 psi. It is made of 6351-T6 aluminum as described by DOT specification 3AL cylinders. The internal volume of the tank is 48 cubic inches. The outer diameter of the tank is 2.8 in. Based on the inner volume, the inner diameter was estimated to be 1.2 inches. The stress of the tank was calculated assuming the only stress on the tank was from the internal pressure. The axial, radial, and tangential stresses were calculated using the following equations.

\[ \sigma_a = \frac{P_t}{r_t^2 - r_i^2} \]
The axial, radial, and tangential stresses are 1350, -3000, and 5700 psi assuming the internal pressure is 3000 psi.

The safety factor was calculated using the distortion energy theory and the maximum shear stress theory. The distortion energy theory predicts yielding will occur whenever the Von Mises stress exceeds the yield strength. \( \sigma_\text{dist} \geq \sigma_2 \geq \sigma_\text{m} \)

\[
\sqrt{\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}} \geq \sigma_\text{m}
\]

The maximum shear stress theory states that yielding will occur when the maximum shear stress equals the maximum shear stress in a tension test specimen.

\[
\tau_{\text{max}} = \frac{\sigma_1 - \sigma_2}{2} \geq \sigma_\text{m}
\]

The distortion energy and maximum shear stress theory predict safety factors of 5.4 and 4.7 respectively.

The pressure inside the tank will not cause the tank to yield since it is equipped with a rupture disc. The rupture disc will allow gas to vent if the internal pressure exceeds 5000 psi. The tank must also be hydrotested every five years to ensure that the tank is structurally sound. The date the tank must be hydrotested is stamped on the tank.

The paintball tank being used has a maximum operating pressure of 3000 psi. It is made of 6351-T6 aluminum as described by DOT specification 3AL cylinders. The internal volume of the tank is 48 cubic inches. The outer diameter of the tank is 2.8 in. Based on the inner volume, the inner diameter was estimated to be 1.2 inches. The stress of the tank was calculated assuming the only stress on the tank was from the internal pressure. The axial, radial, and tangential stresses were calculated using the following equations.

\[
\sigma_\text{a} = -P_t \frac{r_t^2}{r_0^2 - r_t^2}
\]

\[
\sigma_\text{r} = -P_t
\]

\[
\sigma_\text{t} = P_t \frac{r_0^2}{r_0^2 - r_t^2}
\]

The axial, radial, and tangential stresses are 1350, -3000, and 5700 psi assuming the internal pressure is 3000 psi.
The safety factor was calculated using the distortion energy theory and the maximum shear stress theory. The distortion energy theory predicts yielding will occur whenever the Von Mises stress exceeds the yield strength. \[ σ_1 ≥ σ_2 ≥ σ_3 \]

\[ \sqrt{(σ_1 - σ_2)^2 + (σ_2 - σ_3)^2 + (σ_3 - σ_1)^2} ≥ S_y \]

The maximum shear stress theory states that yielding will occur when the maximum shear stress equals the maximum shear stress in a tension test specimen.

\[ τ_{max} = \frac{σ_1 - σ_2}{2} ≥ S_y \]

The distortion energy and maximum shear stress theory predict safety factors of 5.4 and 4.7 respectively.

The pressure inside the tank will not cause the tank to yield since it is equipped with a rupture disc. The rupture disc will allow gas to vent if the internal pressure exceeds 5000 psi. The tank must also be hydrotested every five years to ensure that the tank is structurally sound. The date the tank must be hydrotested is stamped on the tank.

### 3.3.7 TANK BRACKETS

The tank brackets have been designed so that the tank would not be subjected to external stress. The brackets also will contain the tank in the event that the tank valve was removed from the tank, which is the main safety risk of the highly pressurized tank. The force that the tank would exert on the bracket can be calculated using the principle of impulse and momentum. The reaction force is calculated using the following equation.

\[ m_{tank}a_{tank} = P_eA_e - P_{atm}A_e + \dot{m}_e \dot{v}_e \]

Where
- \( m \) is the mass of the tank
- \( a \) is the acceleration of the tank
- \( P_e \) is the absolute pressure to the gas exiting the tank
- \( A_e \) is the area of the valve
- \( \dot{m} \) is the mass flow rate
- \( \dot{v}_e \) is the velocity of the gas exiting the tank

If the tank valve should detach the exit diameter is about 0.125 inches. If the tank is filled to the maximum absolute pressure of 3014.7 psi, the back pressure ratio will be less than the critical pressure ratio, and the gas will exit at a velocity of Mach 1. The tank brackets will need to resist a force of 210 N. If the tank was not constrained its initial acceleration would be 92 m/s\(^2\).
3.4 SOLENOID DRIVER

Two of the DRV104 manufactured by Texas Instruments will be used to engage the solenoids of the pneumatic valves actuated system. The DRV104 will act as a two state driver to open and close our solenoid. The chip has an internal pulse width modulation (PWM) feature that is used to make the chip more reliable by reducing power consumption and heat produced by the chip and to generate a modulated output current.

There is a delay that occurs at the beginning of each of the DRV104 PWM cycle which determines the amount of time the DRV104 will run at 100% duty cycle before it is switched to its PWM mode. It is important to have the solenoid completely closed before using the chip’s internal PWM mode so the solenoid will run at the best efficiency. This will ensure that the valve will be completely open the whole duration the solenoid is powered. The minimum output current of the solenoid driver will be 1.2A and a typical output voltage of 0.45 V.

The timing constraint in the PWM cycle of the DRV104 forces us to rely entirely on the PWM cycle of the TTL input voltage from the PIC18F458 microcontroller, which can be programmed and adjusted as desired, in order for the DRV104 to generate a modulated current needed to actuate the kicker with various speed.

3.4.1 LIMITATIONS

In order to turn the output of the chip ON the TTL input voltage must be between +2.2V and +5.5V. In order to turn the output of the chip OFF the TTL input voltage must be between 0V and +1.2V. The input voltage coming from the PIC18F458 processor is a PWM signal which will create a PWM output signal. This received PWM signal will have to be larger than the internal DMOS transistor turn-on time. A typical DMOS transistor turn on time is around 50nS. The DRV104 will shut down if the junction temperature reaches +140 °C but will perform the best at +125 °C or below. There still might be the need for heat sinking. We will have to find out how long the chip will be turned on to power our solenoid in order to know if heat sinking will be needed. The 3A Schottky diode will act as a flyback diode in this circuit. The flyback diode needs to be used at the output of our chip to eliminate any current from returning to the chip.

3.4.2 THERMAL ANALYSIS AND POWER CALCULATIONS

The power consumption of the Solenoid is 6 watts. The solenoid is powered by 12 volts meaning that it requires 500mA current to power this solenoid.

\[ P_d = R \times I_0 \]
The power consumed in this chip depends on the output current and the output DMOS transistor on-resistance provided by the datasheet ($R_o=0.45$ ohms) multiplied by the output current ($I_o=500mA$). The power dissipated is 225 mW. When calculating how much of a difference there is between the ambient temperature and the junction temperature, we find that the junction temperature equals the ambient temperature plus the total power dissipated ($PD=225$ mW) multiplied by the thermal resistance provided by the datasheet ($\theta_{JA}=37.5$C/W).

$$T_J = TA + PD \times \theta_{JA}$$

We find that the ambient temperature is only 8.4375 °C smaller than the junction temperature which is possible to get as hot as 85 °C as described in the datasheet.

$$P_i = Vi \times ii$$

The input power ($P_i$) is input voltage ($V_i=12V$) multiplied by the input voltage ($I_i=1mA$). The input power is 12mW.

**3.4.3 ADDITIONAL ANALYSIS**

All of this analysis was done through different data tables. We decided to run the solenoid at 100 percent duty cycle. To ensure that the duty cycle is 100 percent we set the delay time as long as possible. The solenoid is only going to be powered for a short amount of time and we want to close the solenoid quickly to ensure the valve opening happens as quickly as possible. The delay capacitance is 10 $\mu$F. Since we are going to keep the DRV104’s duty cycle at 100 percent, there is no need for the chip to use its internal PWM. We set the oscillator of the DRV104’s PWM to the lowest frequency to minimize the EMI with of the rest of the circuits. The DRV104 has an internal oscillator. The resistor required for the lowest oscillation frequency is 9.76MΩ. We also set the duty cycle of the chip’s internal PWM to 90 percent. The resistor value that will create this duty cycle is 1050 kΩ. This is also to ensure the valve opening happens as quickly as possible. The current flyback diode is 3A this data was obtained through the application circuit figure 4.11. This flyback diode will be used as a protective circuit stopping any current from returning to the DRV104. The boot—pin5 is attached to a 470pF capacitance and the output pins—6, 7 like figure 11 shows. There are two outputs available for driving two different solenoids using the same input from the PIC18F458 processor. We are planning to use two different solenoid drivers but they need to be activated at different times requiring us to have two different inputs from the PIC18F458.
3.4.4 PIN CONFIGURATIONS

**Figure 3.11: DRV104 Chip interface circuit**

- **PIN1** = Duty Cycle Adjust
- **PIN2** = Delay Adjust
- **PIN3** = Oscillator Frequency Adjust
- **PIN4** = Master
- **PIN5** = BOOT
- **PIN6, PIN7** = OUT1, OUT2
- **PIN8, PIN9** = Vps1, Vps2
- **PIN10** = +Vs
- **PIN11** = GND
- **PIN12** = SYNC
- **PIN13** = Status KO Flag
3.5 SENSORS

We decided to mount two proximity sensors in the kicking mechanism to help us locate the presence of the ball closer to the kicker. When the ball gets closer to the kicker, the sensor connected to the kicker will send a signal to the PIC18F458 to notify the presence of an object in the range of the sensor.

The sensing range varies from one type of sensor to another. The sensor that we will use for our project is a proximity sensor CT1-AP-1A. The sensor that will be used in our project is manufactured by AUTOMATIONDIRECT (see www.automationdirect.com).

Capacitive proximity sensor will sense metal as well as nonmetallic materials such as paper, glass, liquids, and cloth. When an object is within 15 mm of the sensor, the sensor will send a signal to the PIC18F458 to activate the kicker.

![Capacitive proximity sensor](image)

Figure 3.12: Capacitive proximity sensor
Figure 3.13: Capacitive proximity sensor mounted on the 3-DX robot

The sensing range: 2-15 mm
Power supply: 10-30 DC
No-load supply current: 8mA
Power consumption: 96mW
Switching frequency: 100Hz
Dimension (length, diameter): 100mm × 30mm

The CT1-AP-1A sensor will come with a shield housing which will restrict it from detecting material that surround the sensor. It will only detect object pass through it sensing face. There is a 2 m cable that comes with our sensor. We will need to purchase
separately some mounting brackets (ST30A metal axial bracket or ST30C right-angle bracket) to help us fix our sensors. Referring to mechanical design, we may use two sensors for better result.

The output type of the sensor is PNP. The CT1-AP-1A sensor has three wires: one wire is connected to 12V DC in the 3-DX power board, the second wire is connected to ground and the third wire is connected to pin 19 or 20-RD0/PSP0C1IN+ or RD1/PSP1C1IN- of our PIC processor. From Figure 6.15, there are three wires that come out of the box which is internal to the sensor. The top wire BN (Brown) will be connected to 12V, the middle wire BK (Black) is the signal thus when the sensor detect an object it will send 12V to the PIC processor, and the lower wire BU (Blue) is the ground wire.

![Figure 3.14: PNP output wiring diagrams](image)

### 3.6 POWER ANALYSIS

The primary power source we will use for our design is going to be provided by the Pioneer 3-DX robot. The robot has a power board that provides several power ports—5V and 12V—that could be used for any additional accessories or user applications. This power will be provided by the three 12V- lead acid batteries that come with the robot. However, drawing a certain amount of power may affect the robot’s behavior, which explains the purpose of the following power analysis that was conducted.

This analysis basically consists of taking the sum of the power consumed by every electrical or electro-mechanical element that is going to receive power supply from the Pioneer 3-DX batteries and comparing it to the power generated by the 3 batteries of the robot, in order to decide how long and well the robot would perform.
Table 3.4: Voltage needed to power different components

<table>
<thead>
<tr>
<th>Description</th>
<th>Operating Voltage (V)</th>
<th>Max Supply current (mA)</th>
<th>Power consumption (W)</th>
<th>Power supplied by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solenoid Valve</td>
<td>12</td>
<td>375</td>
<td>4.5</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>Solenoid Valve</td>
<td>12</td>
<td>375</td>
<td>4.5</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>PIC18F458</td>
<td>5</td>
<td>50</td>
<td>0.5</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>PIC18F458</td>
<td>5</td>
<td>50</td>
<td>0.5</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>Solenoid driver chip</td>
<td>12</td>
<td>0.6</td>
<td>0.0072</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>Solenoid driver chip</td>
<td>12</td>
<td>0.6</td>
<td>0.0072</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>MAX232</td>
<td>5</td>
<td>0.01</td>
<td>0.00005</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>Proximity sensor</td>
<td>12</td>
<td>8</td>
<td>0.096</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td>Proximity sensor</td>
<td>12</td>
<td>8</td>
<td>0.096</td>
<td>3-DX Power board</td>
</tr>
<tr>
<td><strong>Total Power</strong></td>
<td></td>
<td></td>
<td><strong>10.2</strong></td>
<td></td>
</tr>
</tbody>
</table>

All the components in our design use 12V DC or 5V DC. Since these specific voltages can be provided by 3-DX microcontroller pins there is no need for us to build voltage amplifier circuit to achieve different voltage.

Maximum supply currents at specific voltage are displayed at the table above to help us compute the power consumption of every component that will be used in our design.

The total power consumption for all component used in the design is 10.2 W. When the robot is fully charge it has 252 Watt-hours which is large enough for all our components. However we need to take into account the power consume by the camera, the 3-DX PC, the 3-DX microcontroller, and other electronic component included in the 3-DX robot. All internal component of the 3-DX robot consume about 84 W. combining the power consumption of robot itself with that of component used in our design give us a total of 94.2 Watts which is small compare to 252 Watt, thus our robot is expected to run for more than two hours if fully charged.

3.7 PIC 18F458 MINROCONTROLLER INTERFACE DESIGN

We will interface a female DB9 serial connector on our control unit board, and using a male to male DB9 cable we will connect it to the female serial connector of the SH2 controller of the robot. The main reason why we chose to interface a female serial connector is because it more durable and reliable, as oppose to male connector that could easily get damaged when their internal pins—as shown on the figure below—break or get bent. It would be much easier to replace the cable if the pins of the male connectors get damaged than it would be to replace it from the circuit board of the control unit.
The PIC18F458 is the microcontroller that was primarily chosen for our design. The PIC microcontroller pins operate mostly at a TTL, as opposed to the Serial connectors of the Robot that is RS232 level compatible; thus, a transceiver or driver is needed to be interfaced with the PIC18F458 and convert the voltage levels from the serial connectors to the microcontroller and vice versa. One of the most popular drivers used for this purpose with PIC microcontrollers is the MAX232 from Maxim Inc, which we selected to use for our design. The figure above shows the interface circuit with the PIC18F458. The bits 6 (Pin 26: TX/RC6) and 7 (Pin 25, RX/RC7) of Port C will be programmed to Transmit and Receive signals between the PIC microcontroller and the SH2 Pioneer 3-DX microcontroller. Figure 4.16 shows the interface circuitry and pins used.
The 20MHz CA 20.000M-C from EPSON was chosen as recommended by the Microchip in the PIC18F458 datasheet; however this is specific part is very rare and difficult to find, so we will use the H01588-01W from BCCOMP, which is a similar 20 MHz parallel cut crystal also popular for PIC microcontroller application. In addition, 2 ceramic Disc 22 pF 100V capacitors will be used for the crystal oscillator circuit interface as sown in Figure 4.17.

![Crystal oscillator interface circuit with the PIC18F458](image)

**Figure 3.17: Crystal oscillator interface circuit with the PIC18F458**

### 3.7.2 POWER-ON RESET CIRCUIT

The PIC18F458 could be interfaced and used with various kinds of RESET modes. We will be using the Power on Reset (POR), which consists of starting the reset of the chip every time it is being powered. The pin1—MCLR of the PIC18F458 requires a minimum rise rate of 0.05V/ms the VCC in order for it to reset the microcontroller; therefore, the circuitry shown on the Figure 6.19 below is recommended by Microchip(from the Datasheet) for slow rise time. The diode used in the circuit helps the capacitor discharge quickly if when VCC powers down.
3.7.3 PIC18F458 I/O PORT USAGE

PWM output ports—Pneumatic Valves

Finally, the digital ports of the PIC microcontroller will be used for two purposes; the first one is to provide output voltages to the driving circuit that will be used to drive the 2 actuators of the kicker. Because we are trying to actuate pneumatic valves, a Pulse width Modulated (PWM) output will be preferable to allow a variation of the kicker’s actuation speed. The PIC18F458 has 2 types of Pulse width Modulation modules; a Capture Compare PWM (CCP) that provides 1 PWM output and an Enhanced Capture Compare PWM (ECCP) module that allows up to 2 PWM output channels active at the time with user selectable polarity; and, which could be operated in four different output mode configurations—Single output, Half-bridge Output, Full Bridge Output, forward mode and Full bridge Output, reverse mode. The module’s output mode and polarity will be configured by setting the EPWM1M1:EPWM1M0 and ECCP1M3:ECCP1M0 bits of the ECCP1CON register—ECCP1CON<7:6> and ECCP1CON<3:0>, respectively.

Our kicking mechanism will requires the 2 pneumatic valves to be actuated synchronously in order for it to kick the ball in a straight path; therefore, we need two PWM output ports from the PIC18F458 that could operate synchronously. Although the ECCP module provides 4 outputs, none of its output mode’s configurations could provide to PMW output ports operating as desired—synchronously modulated high active. So the solution to this problem would be to use both CCP and ECCP modules that we can configure to operate synchronously and provide to PWM outputs as desired as illustrated on figure 11. In order to do that, the ECCP module will be configured for a standard...
single PWM output mode, and the period and the duty cycle of the PWM output will be set by writing to the PR2 and the CCP1RL register, respectively. The PWM period and duty cycle are computed using the following equations:

- **PWM Period** = \(((\text{PR2}) + 1) \cdot 4 \cdot \text{TOSC} \cdot (\text{TMR2 Prescale Value})

- **PWM Duty Cycle** = \((\text{CCPR1L:CCP1CON<5:4}> \cdot \text{TOSC} \cdot (\text{TMR2 Prescale Value})

Where \(\text{TOSC}\) is the clock oscillator period and \(\text{TMR2}\) is an 8-bit period register for the Timer 2.

Finally, bit 2 (Pin 17: RC2/CCP1) of Port C and bit 4 (Pin 27: RD4/PS4/ECPP1/P1A) of port D will be programmed as PWM outputs of the CCP and ECCP modules, respectively. When the PIC18F458 powers up, the I/O pins are all in a high impedance state; therefore a 2-15KΩ pull-up resistors will be connected to the PWM pins chosen in order to keep the power switch devices in an off state until the microcontroller activates the PWM outputs.

Digital Input ports-Sensors

The second purpose will be to connect the two proximity sensors that will be used to send a signal to alert the robot when the ball gets in its proximity. Pin 19, and 20—RD0/PSP0C1IN+ and RD1/PSP1C1IN- will be used and bit 0 and 1 of Port D will be programmed as a digital input (RD0 and RD1).
In order to design our PCB layout, we first generated the schematic of our control unit in Multisim with all the correct components that were needed, and then we used the Ultiboard program to create the PCB layout shown on the figure below. Finally, the ultiboard file was converted into a Gerber file which was used by a computer controlled machine to cut the board.

Figure 3.21: Final Schematic of the Control Unit PCB
4. BUILDING PROCESS

4.1. MECHANICAL KICKER

Pneumatic cylinders

The twin pneumatic cylinders were tested individually with a compressed air source prior to be connected to any power. The cylinders were both tested repeatedly to ensure that they would be robust enough for live operations. Next the cylinders were connected to the solenoid valves and power was derived from an external source (not the robot). Finally, the cylinders were tested with the tank compressed air source and connected to the kicking plate via the wooden mock-up, this will be described later. While connected to the wooden prototype both cylinders were operated individually and in tandem.

Kicking plate

The kicking plate was fabricated with 6061 aluminum and bonded to the aluminum bracket. The kicking plate was secured to the cylinders on the prototype and operated manually be pulling on the either side of the kicking plate. This ensured that the operation would be smooth and not likely to damage the cylinder during full normal operation.
4.1.1. MODIFICATIONS

The original mechanical design called for a single machined kicking plate. The available resources were not able to machine it as a single component so the bracket, which attached to the cylinders, and kicking plate had to be connected via an alternative method. It was determined that there were epoxies available that would meet the forces exerted by the cylinders. However, as testing commenced silicon spray was used to reduce friction at the pins. It is believed that the spray may have compromised the epoxy, because twice after applying the spray the bracket and kicking plate failed. Finally, the epoxy was replaced by brackets that were connected with counter-sunk bolts.

4.1.2. PROTOTYPE KICKING MECHANISM

Wooden Prototype

A wooden prototype was constructed, to match the critical dimensions of the robot base. The dimensions critical to construction were, ground clearance, location of cylinders on prototype underbody and kicking plate clearance, both from the robot base and from the ground. Once the pistons and valves were secured to the prototype and connected with tubes and brackets the full prototype was tested. Both left and right kicks were tested as well as a straight kick. The kicks were repeated multiple times and in various sequences to ensure robustness. The process was repeated until the tank was emptied.

Figure 4.1: Wooden Prototype
4.1.3. FINAL KICKING MECHANISM

At the conclusion of prototype testing the components were attached to the robot base body. Cylinders were bolted to the robot underbody. The tank was attached to the top of the robot with brackets. A clevis was used to attach the kicker to the cylinders. A camera tower was constructed to improve ball location and meet the RoboCup Middle Size League height requirements. After initial testing it was determined that the camera tower should allow the camera to angle down more severely to improve ball location. The camera tower was angled down approximately 30°. The proximity sensor was also added to the front of the robot and attached to the camera tower. The proximity sensor was tested first by rolling the ball slow to the metal plate multiple times for indicated left, right and center kicks.
Figure 4.3: Cylinders Bolted to Under Pioneer 3-DX

Figure 4.4: Front View of Kicker
The building process of our control unit consisted of constructing a printed circuit board (PCB) for the circuit illustrated on the schematic above and programming the PIC18F458 microcontroller as desired. This process was divided into two main phases; the first phase was committed to building a prototype control unit board and doing preliminary testing on it, then the second phase focused on building and testing a final control unit PCB.

4.2.1. MODIFICATIONS

**PIC18F458-Sensor circuitry Interface**

In the beginning of the semester, we thought about using a voltage regulator to convert the sensor output from a 12V to 5V needed in the PIC18F458. But after testing the voltage regulator, we find that the output current of the voltage regulator was larger than the input current range allow for the PIC18F458. Next we designed a transistor circuit to convert a 12V to a 5V needed for the PIC processor. This transistor circuit had: a NPN transistor, two resistors, an inverter chip, and power supplies. To test this product, we connected a 12V power supply at the base and 5V at the collector. The collector-emitter voltage was inverted using the inverter chip. The output of the transistor circuitry was sent at the PIC18F458 microcontroller. In addition, we used one sensor in our final
design instead of two because during testing we realized that one sensor was enough to get the ball detected that was needed.

Figure 4.6: Final PCB of the Control Unit

**PIC18F458-Solenoid driver Interface**

Originally, the pins 17 and 27 of PIC microcontroller were going to be configured as PWM (pulse width modulated) output which would feed a varying voltage to the DRV104 solenoid driver chips in order to control the power of the kicks. But after preliminary testing of the solenoid valves and confirming with the application engineers for the solenoid driver, we found that the valves would only open and close instantaneously. Thus, using a PWM voltage would not have been necessary anymore. For the final design we just configured pins 24 and 27 as digital outputs sending a 5V signal to the solenoid drivers.

**Serial Interface with the Pioneer 3-DX**

Initially, we intended to establish communication between the onboard computer of the robot and the PIC18F458 through the serial port on the user panel of the robot. However, after initial testing and deeper understanding of the purpose of that serial port on the user panel, we found that such communication between the onboard computer and the control unit board could not be possible as we expected. The internal serial host that allows communication between the SH2 microcontroller of the robot and its onboard computer is shared with the serial connector on the user panel. This connector is primarily intended to allow a client PC to connect to the SH2 controller and control the robot; it is used to run maintenance software for the SH2 controller. The solution we found for our problem was to use USB-to-serial converter connector that will allow connecting to the USB port of the onboard computer of the robot and making it act as a serial port. We simply installed the driver for the connector we purchased, which created a virtual serial port COM5 that represents the USB connection. We then repeated our primary test using HyperTerminal tool and were successfully able to confirm the communication to the port.
4.2.2. PIC18F458 PROGRAMMING

We used the MPLAB ICD2 In-circuit debugger to program the PIC18458. We first built the required interface circuit between the ICD2 and the microcontroller in order to be able to start the programming and debugging process. The two figures below describe the connections for this interface.

In addition, we used the MPLAB programming environment software that allowed us to write, compile and download our codes onto the PIC microcontroller. We developed our code in C language and compiled it using the MPLAB-C18 compiler. Here are the key steps that we followed to program the PIC microcontroller.

Step1: Install MPLAB on your computer

Step2: Install the desired compiler on the computer (C18 or else)

Step 3: Connect the ICD2 Debugger to the computer through USB or Serial.
Step 4: Connect the circuit board to the ICD2 through the phone jack connector.

Note: Also power the ICD2 with a 9V supply if it will be powering the circuit board.

Step 5: Open MPLAB and go to Project wizard under the project menu. Follow the steps to create and configure a new project.

Note: when creating a new project, remember to choose the correct PIC microcontroller that will be programmed and the desired compiler.

Step 6: Add source files, header files, and linker files needed for the project.

Step 7: Under the programmer menu, go to choose programmer and select MPLAB ICD2

Step 8: a self test and the connection will be automatically engaged. Verify that everything is successful. One of the window will give you log messages about the status of every operation taken.

Note: If the ICD2 is not automatically connected or if the connection fails, retry it by clicking “connect” under the programmer menu.

Step 9: Build the project by going under the project menu.

Step 10: download the built program onto the microcontroller from the programmer menu.

In the beginning, we programmed to microcontroller to do basic task such as configuring some of its I/O ports as input and output and verifying that they were properly configured by connecting LEDs at the outputs and verifying that they would light up when expected.

The final code we wrote mainly consisted of tracking the sensor’s input and the kick command received from the robot through its serial ports and engaging the kicker by sending digital output voltages through its I/O ports. The final copy of the code is attached to the appendix.

4.2.3. PROTOTYPE

Due to the fact that we did not have any Dip packages for the DRV104 chips and the current flyback diodes, we initially built the entire circuit on the breadboard except for the driving circuit interface. First, we build the small programming interface circuit to establish a communication with the PIC microcontroller and setup the programming environment. Next, we built the sensor interface and verified that the PIC could successfully read the sensor input and lighting some LED’s. Furthermore, we built the serial interface circuit and modify the PIC program to receive a signal from the serial port first, then receive the sensor’s input, and finally it will light up the LEDS. At this stage of
the building process, the LED’s represented the driving circuits for the solenoid drivers of the kicker, since they each required approximately 5 V to be lit.

By the time we successfully constructed and tested the interfaces mentioned above, the design of our PCB board layout was also completed, which allowed us to build the prototype PCB board including the driving circuit. The picture below shows the prototype PCB with all repairs made after testing it.

![Prototype PCB board of the control Unit](image)

Figure 4.9: Programming Prototype PCB board of the control Unit

4.2.4. **FINAL BOARD**

After making several repairs and successfully testing the prototype PCB, we implemented the necessary changes that we made to fix the prototype in the design of the final PCB of the control unit. In addition, we added four holes on the edges of the board to mount it in a plastic box, and finally we replaced the crimp connectors with screw connectors which we evaluated to hold a stronger connection.
4.3. SOFTWARE DEVELOPMENT

This part of the building process consisted of developing an application program (AP) that will control the robot as desired. The goal of the AP as we mentioned in the problem statement is to instruct the robot to find an orange ball, find the blue goal, move to the ball, position itself behind ball and facing the goal, and finally send the kick command to the kicker. In order to develop such program, we used the ARIA (Advanced Robotics Interface Application) application program interface (API) that the Pioneer 3-DX robot supports. ARIA allows us to communicate with the internal SH2 microcontroller of the robot and control its motion and the accessories of the robot such as the camera, the sensors, odometer, and others. We used C++ programming language to develop our AP.

During this process, we developed 2 main codes. The first one instructs the robot to find the ball and follow it; whenever it loses the ball, it will rotate for 360 degrees and wander around until it sees it again and repeat the process. This code was obtained by modifying an example code provided with the robot. The program structure basically consists of several classes that are interacting with each other according to the hierarchy established by ARIA. Some classes called action classes are used to implement behaviors such as “kick” or “go to ball”. These action classes require a priority parameter that determines the importance their execution.

With a similar structure, we developed the second code that allows the robot to find the ball, calculate its distance from the ball using calibration that was done before, calculate the angles of rotation and the distance it needs to travel to a kick point where it will be behind the ball and facing the goal, and finally move towards the ball to allow the ball to be seen by the proximity sensor.

In addition, we trained the robot to recognize an orange ball and the blue goal using the ACTS software package. The training is saved as a configuration that should be running
on the robot while the final AP is running in order for the robot to see the ball and behave accordingly.

Finally, here is a brief procedure to follow to run the programs developed.

Step 1: Start ACTS and open the correct training configuration.
Step 2: Open IP through which allows communication to the robot
Step 3: compile the program and run it

4.4. DIFFICULTIES ENCOUNTERED

The main difficulties we encountered throughout the project originated from the Pioneer 3-DX robot.

First, we blew up a fuse in the robot after drawing power from the robot using wires connected to its power board of the robot. One of the wires that supplies 12 V accidentally touched the metal top of the robot when it was powered which caused a short circuit in the robot and blew up the fuse that prevented the electrical boards in the robot from being destroyed. After few hours of troubleshooting and support from Mobile Robots, we found the blown fuse and replaced it.

The second important difficulty we encountered was when the onboard computer was not starting up. In order to mount the kicker at the bottom of the robot, we were required to take out the motherboard of the onboard computer of the robot, in this process we removed the memory card and did not properly sat it back in place. Because of this reason, the onboard computer would not start. It’s only after several hours of troubleshooting that we were able to solve the problem.

Third, we have had and are still having several difficulties with connecting to the robot wirelessly/remote and being able to run all the software packages on the robot’s computer easily. For example every time we stop running a program that was using ACTS (ActiveMedia Color Tracking System), we have to manually restart and sometimes re-open the TCP connection in order to successfully operate the robot.

Finally, we encountered several issues with our prototype PCB. Because of some of the resistors that we used, the PCB at one point of time would not work as expected unless we made contact at a special point of the board using a piece of wire. After several testing and further analysis, we figured out the appropriate values that were needed to make the board work well and consistently.

5. TESTING PROCESS

The purpose of this section is to perform testing of our kicking mechanism. To perform the testing, we tested every electrical and mechanical component separately. Next we
grouped these components and test them together. Finally, we tested the entire system, recorded the data and analyzed these results collected.

5.1. DESIGN PARAMETERS TESTED

The goal of this report is to identify some parameters that need to be determined as well as the method that will be used during the testing process in order to successfully verify the operation of the prototype and validate the original design. During the testing process, we will make sure the main components of the design work as expected independently, which will also help us to better troubleshoot any failure that might occur. These main components fall into three categories: the electrical, the mechanical and the software. We will conclude by defining the parameters of the entire system—final kicking mechanism attached to the Pioneer 3-DX—that we will test in order to ensure that it fulfills all requirements we set at the beginning of the project.

I. Electrical parameters

The electrical parameters of our design that will be covered during the testing mainly consist of the following three interfaces: The PIC18F458-sensor circuitry interface, the PIC-Solenoid driver circuitry, and the PIC-serial interface. However, individual components will also be tested during the building process.

A. PIC18F458 and sensor circuitry interface:

The combination of the PIC microcontroller and the sensor circuitry will be tested using the program downloaded in the PIC microcontroller. We will program the PIC processor to receive a signal from the proximity sensor on pin 16 corresponding to I/O port D. We will read the data received on port D and verify that the appropriate bit is set to 1 when a 5 V is received or it is set to 0 when nothing is sent. The following individual components of this interface will also be tested during the building process as described below:

Proximity sensor: CT1-AP-1A sensor has three wires. The top wire BN (Brown) will be connected to 12V, the middle wire BK (Black) is the signal thus when the sensor detect an object it will send 12V to the PIC processor, and the lower wire BU (Blue) is the ground wire. During the testing part of the sensor we will connect the Brown wire to a 12V power supply and the Blue wire will be connected to ground. Next will bring the ball closer to the face of the sensor and measure what voltage we are getting at the Black wire using a Multimeter.
**Transistor circuit:** This circuit is used to convert a 12V to a 5V needed for the PIC processor. This transistor circuit will have: a NPN transistor, two resistors, an inverter chip, and power supplies. To test this product, we will connect a 12V power supply at the base and 5V at the collector. The collector-emitter voltage will be inverted using the inverter chip.

**Sensor circuitry:** The combination of proximity sensor and transistor circuitry will need to be tested to verify the exact voltage signal that will be sent to the microcontroller. We are expecting to measure about 5V at the output of the combination. The multimeter and power supply will be used for this testing. Another important measurement that will be tested is the current at the sensor circuitry. We will verify that the maximum output current of the signal sent by the sensor is below the maximum current sunk by the I/O pins of the PIC microcontroller.

B. **PIC18F458 and solenoid driver interface:**

The processor and solenoid driver interface will be tested to make sure that a digital input voltage (approximately 5V) from the PIC18F458 would be converted to a 12V and 0.8A current or more needed to close or open the solenoid valves, respectively.

Microcontroller: The PIC18F458 will be programmed to send a digital voltage signal to pins 24 and 27 which will be connected to an oscilloscope to verify that the code running is allowing the PIC microcontroller to send the right voltage signal (0-5V).

Solenoid Driver: Pin 14 of the DRV104 chip will receive the digital voltage from the PIC18F458 and output current at pins 6 and 7. These two pins are shorted together making one output which will be connected to an ammeter to verify that a 0 or at least 0.8 A is sent out to the valves to close and open them, respectively. Finally, we will visually make sure that the solenoid valves close when a 0 current is measured at the output of the drivers, and open when a 1.2 A is measured.
Figure 5.2: Pin diagram of PIC18F458

C. PIC18F458 Serial Interface with the Pioneer 3-DX

We will verify that this interface allows a signal sent through the serial host of the Pioneer 3-DX robot to be successfully received by the PIC microcontroller and vice versa. In order to test that, we will first, connect the PIC18F458 to the serial host of the robot; then, we will write a simple code in C/C++ to send and receive signals through the internal serial host of the robot; finally, we will read the data received on Port C to verify that the appropriate bits are set to 1. Similarly, we will verify that a transmitted signal from the PIC microcontroller through the serial host is successfully received.

II. Mechanical parameters

In order to ensure the kicking mechanism is operating properly we initially tested these components independently from the remainder of the system. The components to test are as follows:

Pneumatic cylinder: Two pneumatic cylinders will be used to transfer energy of a compressed fluid to kinetic energy of the ball. In the final system they are attached to the undercarriage of the robot assembly. Two solenoid valves will control whether the cylinders are activated or at rest.

Kicking plate: The kicking plate and kicker has the ability to adjust the kicking direction. The range of kick expected can be calculated based on the distances between the cylinders (8 cm) and plunger length of the cylinder (5 cm). The theoretical range of kicking direction is \(-/+ \tan^{-1}(5/8)\) or -32° to +32°. The minimum desired kicking distance, including roll, will be 5 m.
Initially, the kicker was built separately from the robot assembly. The cylinders were secured and via the pressure regulator will be connected to a compressed air source. In separate trials we fed compressed air to each independent cylinder to attempt to kick at the maximum negative and positive kicking angles. Then, equal pressure was pushed through both cylinders simultaneously to model a straight kick. For each kick the distance traveled in flight was measured, time to travel 5 m and angle of kick. The average velocity of the ball was calculated based on the time to travel the 5 m, if reached. The angle of kick and flight distance was estimated by observation.

To determine the impact of the kicking pressure on subsequent kicks the above tests will be repeated at varying pressures. This can be controlled via the pressure regulator.

III. Software parameters

The software parameters will be tested to verify that application programs developed will move and control the robot as described initially and without the kicker attached. They include making sure that the robot could recognize an orange ball and a blue goal, move towards the ball and positions itself with respect to the ball in order to kick it in the direction of the goal. The codes developed throughout the design will be tested to achieve the following tasks separately:

The robot will find the ball and chase it, then find a blue goal and move towards it.

The robot will be able to calculate world coordinates of an object that it sees and recognizes

The robot will move to a location specified by the x and y world coordinate of the location

In the end, the application program will command the robot to find the ball and the goal, calculate the world coordinate of a location where the robot will be behind the ball and facing the goal, and finally move the robot to the computed location.

Once we attached the kicker is attached, the application program tested above will be improved to send signals to the PIC microcontroller when the robot is in certain states of execution such as “ready-kick” state or “kick-complete” state for example. Finally, we will verify that those signals are sent and received using the method described in the PIC serial interface section above.

IV. System parameters and requirements
The testing will be conducted in a classroom with a minimum kicking area of 8m square, to ensure adequate space. Additionally, it is important to have the room well lit for subsequent testing with the camera and proximity sensors.

After testing has been completed on the components it is necessary to repeat the testing procedures for the final assembled unit.

Robot mobility: Identify any noticeable delays in the robot during location of the ball and directing itself towards the ball. We are not testing the speed of the robot per se but the game suitability of the system. Additionally, how quickly does the robot orient itself such that the ball is kicked in the direction of the goal?

Kicking motion: Repeat the component tests earlier. Determine any impact the completed robot assembly has on the results. For example: Does the fact that the robot is on wheels adversely affect kick angle, trajectory or kick velocity? This can be repeated while robot recoil is hindered and while the robot is free to roll.

Repeat tests to determine the impact of battery loss or pressure loss on the quality of the kicks. Again, as mentioned earlier, the pressure loss can be simulated by adjusting the pressure regulator.

Measure battery power throughout the tests to determine the relationship between power consumption and kick frequency.

5.2. ELECTRICAL TESTING

Every electrical component was tested separately to verify that every component is working properly. Then we combined these electrical components and tested them together.

5.2.1. PIC18F458 AND SENSOR CIRCUITRY INTERFACE

This section describes in details the procedures that were followed in order to test the PIC18F458-Sensor circuitry interface.

Sensor

The sensor has three wires (Brown, Black and Blue). To test the sensor, connect the brown wire to 12V pin of the power supply and the blue wire to the ground pin of the same power supply. Next connect the black wire to the positive pin of the Multimeter and the blue wire to the ground of the multimeter. Every time you bring the ball closer to the front face of the sensor, a green light will display at the surface of the sensor and 12V reading will be seen at the Multimeter display.
We measured the output voltage $V_{out}$ at the black wire when an object is present at the face of the sensor. $V_{out}$ is reported while the input voltage $V_{ps}$ at the brown wire is varying from 0 to 20V. We also reported the output voltage $V_{no}$ at the black wire when no object is present at the face of the sensor.
Table 5.1: Sensor Testing Result

<table>
<thead>
<tr>
<th>Vps</th>
<th>Vout</th>
<th>Vno</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.007</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.532</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>1.014</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>2.096</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>2.99</td>
<td>0.002</td>
<td>0.004</td>
</tr>
<tr>
<td>4.08</td>
<td>0.004</td>
<td>0.004</td>
</tr>
<tr>
<td>5.07</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>6.09</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>6.53</td>
<td>0.007</td>
<td>0.009</td>
</tr>
<tr>
<td>6.97</td>
<td>5.66</td>
<td>0.008</td>
</tr>
<tr>
<td>7</td>
<td>5.77</td>
<td>0.009</td>
</tr>
<tr>
<td>7.06</td>
<td>6.22</td>
<td>0.009</td>
</tr>
<tr>
<td>7.51</td>
<td>7.47</td>
<td>0.009</td>
</tr>
<tr>
<td>8.01</td>
<td>7.98</td>
<td>0.009</td>
</tr>
<tr>
<td>8.5</td>
<td>8.45</td>
<td>0.009</td>
</tr>
<tr>
<td>9.02</td>
<td>8.97</td>
<td>0.009</td>
</tr>
<tr>
<td>9.5</td>
<td>9.45</td>
<td>0.01</td>
</tr>
<tr>
<td>9.99</td>
<td>9.94</td>
<td>0.01</td>
</tr>
<tr>
<td>10.49</td>
<td>10.43</td>
<td>0.011</td>
</tr>
<tr>
<td>11.02</td>
<td>10.97</td>
<td>0.011</td>
</tr>
<tr>
<td>12</td>
<td>11.96</td>
<td>0.013</td>
</tr>
<tr>
<td>13.02</td>
<td>12.99</td>
<td>0.013</td>
</tr>
<tr>
<td>14.03</td>
<td>13.99</td>
<td>0.015</td>
</tr>
<tr>
<td>15.01</td>
<td>14.96</td>
<td>0.016</td>
</tr>
<tr>
<td>16</td>
<td>15.95</td>
<td>0.018</td>
</tr>
<tr>
<td>17.02</td>
<td>16.97</td>
<td>0.018</td>
</tr>
<tr>
<td>18.02</td>
<td>17.98</td>
<td>0.02</td>
</tr>
<tr>
<td>19.06</td>
<td>19.02</td>
<td>0.022</td>
</tr>
<tr>
<td>19.99</td>
<td>19.95</td>
<td>0.023</td>
</tr>
</tbody>
</table>
From the graph above, $V_{\text{no}}$ represents the output voltage when no object is detected by the sensor, and $V_{\text{out}}$ is the output signal when an object is detected by the sensor. From the graph, we observe that when no object is detected by the sensor, no matter what input voltage is placed at the brown wire, the output voltage $V_{\text{no}}$ will be about zero. From the graph, we also observe that when an object is present at the face of the sensor, the sensor will read a value greater than zero only when the input voltage is about 7V, and the output voltage is proportional to the input voltage starting at about 7V.

**Transistor circuitry**

The transistor circuitry is constructed based on the circuitry below.

The base of the transistor is connected to 12V through a 1M$\Omega$ resistor, and the collector is connected to 5V through a 47K$\Omega$. These values were obtained by calculating the saturation of the transistor. The collector-emitter voltage is passed through an inverted, and the output of that inverted is measured using the Multimeter. During testing every
time we have 12V at the base, the Multimeter should read about 5V, and when we have about 0V at the base the multimeter should read 0V.

Here we reported the output voltage Vo out of the inverter chip while the voltage at the base VBB is varying from 0 to 20V.

Table 5.2: Transistor circuit testing

<table>
<thead>
<tr>
<th>VBB</th>
<th>Vo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.002</td>
<td>0.061</td>
</tr>
<tr>
<td>1.029</td>
<td>0.061</td>
</tr>
<tr>
<td>2.022</td>
<td>0.061</td>
</tr>
<tr>
<td>3.01</td>
<td>0.06</td>
</tr>
<tr>
<td>4.05</td>
<td>0.243</td>
</tr>
<tr>
<td>4.09</td>
<td>0.368</td>
</tr>
<tr>
<td>4.49</td>
<td>2.928</td>
</tr>
<tr>
<td>5</td>
<td>3.73</td>
</tr>
<tr>
<td>6.01</td>
<td>4.09</td>
</tr>
<tr>
<td>7</td>
<td>4.09</td>
</tr>
<tr>
<td>8.04</td>
<td>4.09</td>
</tr>
<tr>
<td>9.01</td>
<td>4.09</td>
</tr>
<tr>
<td>10.07</td>
<td>4.09</td>
</tr>
<tr>
<td>11.18</td>
<td>4.09</td>
</tr>
<tr>
<td>12</td>
<td>4.09</td>
</tr>
<tr>
<td>13.09</td>
<td>4.09</td>
</tr>
<tr>
<td>14.05</td>
<td>4.09</td>
</tr>
<tr>
<td>15.1</td>
<td>4.09</td>
</tr>
<tr>
<td>16.05</td>
<td>4.09</td>
</tr>
<tr>
<td>17.09</td>
<td>4.09</td>
</tr>
<tr>
<td>18</td>
<td>4.09</td>
</tr>
<tr>
<td>19.01</td>
<td>4.09</td>
</tr>
<tr>
<td>20.06</td>
<td>4.09</td>
</tr>
</tbody>
</table>
The output voltage from the transistor circuit is fed into the PIC18F458. The graph above show that when the voltage at the base of the transistor circuit is greater than 5V the output voltage sent to the PIC18F458 will be about 4V.

**Sensor circuitry**

To test the entire sensor circuitry, we combine the sensor and the transistor circuit. Thus instead of connecting the base of the transistor circuit to 12V power supply, we connect the transistor base to the 12V coming out of the black wire of the sensor.
PIC18F458 and sensor circuitry interface

The PIC18F458 and sensor circuitry interface was tested using a C program downloaded in the PIC18F458. Pin 16 in the PIC18F458 microcontroller was configured to receive the output voltage from the sensor circuitry, and Pins 24 and 27 were connected to LED to verify that whenever 4 or 5V was received at Pin 16 LED connected to Pin 24 and 27 will light up.

This part of the testing consisted of observing that the LED lighting up when an object is detected by the sensor and staying off when no object is detected by the sensor.

After combining the sensor and the transistor circuitry, we observe that whenever the sensor is powered with at least 7V if an object is detected by the sensor, the voltage that will be sent to the PIC18F458 controller will be about 4V.

5.2.2. PIC18F458 AND SOLENOID DRIVER INTERFACE

Solenoid driver circuit

The driver circuit was tested on a prototype board and ended up getting hot and burning out. So we had to attach two potentiometers in place of the duty cycle resistor and the oscillator resistor. Through this testing we obtained the resistor results of 552 KΩ for the duty cycle and 940kΩ for the oscillation. This makes the duty cycle about 20 percent and the oscillator 5 KHz. The output voltage of the solenoid driver is 11.46 V and has a current output of 1.24 A.
As mentioned earlier, the driving interface will be tested only once the DRV104 chips are mounted to the PCB board. First, we will connect the board to a laptop through the serial connectors, and then we will connect the sensor to the board and finally power everything up with 12V and 5V supplies. Next, we will use the HyperTerminal and the sensor as described above to force the PIC to send out 5V digital outputs to the driver chips and verify the signal sent is effectively 5V using a multimeter. Finally, we will verify that we get a 12 volt output voltage from the driver chips by connecting it to an oscilloscope.

Once we verify that the output is 12V, we will now connect the drivers' outputs to the solenoid valves and follow the same procedure except that this time we will verify that we can hear the valves opening up and extend the pneumatic cylinders if a pressure tank is connected.

5.2.3. PIC18F458 SERIAL INTERFACE WITH PIONEER 3-DX

PIC18F458 Serial Interface with the Pioneer 3-DX

The testing of this interface was done in 3 stages. We first used the HyperTerminal tool from windows to verify we can send and receive signal to that serial port of the robot.

Here are the basic steps to follow to set up this test environment.

Step 1: we connect a USB- to-Serial converter cable to the USB port of the robot.

Step 2: Open HyperTerminal

From the windows menu, HyperTerminal is accessed by going to ‘Communications’ in the Accessories menu. Then, we name the connection; we choose the COM port; and set the parameters for the serial communications as desired. The screenshot in the figure
below shows some of the parameters that need to be set. For our testing we restore the
default values and changed the baud rate to 19200.

Step 3: Loop the transmit signal back to the receive signal

In order to do this, we first soldered two wires on the “transmit” and the “receive” pins of
a DB9 female connector; then, we connected it to the DB9 male end of the USB to Serial
converter cable. Finally, we connected the two wires together which creates an echo of
what is sent to the port.

Step 4: type characters on the keyboard and verify that they are displayed in the
HyperTerminal window.

The second stage of testing the serial interface consisted in testing the serial interface
circuit between the PIC18F458 and the MAX232E driver. Similarly to the previous stage,
we open the virtual COM port using HyperTerminal, but then instead of connecting the
TX and RX pins of the of DB9 connector directly, we connected the transmit (T1IN) and
receive (R1OUT) pins of the Maxim 232E driver together. We then powered the
MAX232E chip and verified that any character that we type is displayed in the
HyperTerminal window, which then confirms that the interface circuit works properly
and the information is being sent and received correctly.
The final stage consists of writing a C++/C code that will send a character to the serial port of the computer and read it back. The code developed and used in this stage of the testing is attached to the appendix. The environment for this test was set up identically as in the first stage. Then, we run the code and verify that the character data sent is the one received and displayed in the execution window.

**PIC18F458 Serial Interface with the Pioneer 3-DX**
Testing the serial interface did not require to record much data; however, in the primary test experiment which consisted of looping the transmitted signal back to the receiver pin, whenever the character’s ‘s’, ‘r’, or ‘l’ is sent, the exact same character should be displayed in the HyperTerminal window. When testing the interface with the system, after any command character is sent to the PIC microcontroller, the PIC continuously sends back the character ‘f’ which is displayed until the kick is made, after which he sends the ‘p’ character which is displayed as well. The figure below shows a screen shot of the HyperTerminal window during the kicking test process.

5.3. MECHANICAL TESTING

This section discusses the testing of the mechanical components.

5.3.1. MECHANICAL TESTING PROCEDURE

The kicking device was first tested by setting the regulator at 90 psi and the initial pressure of the tank was recorded. The kicker was set to kick left, right and then straight alternatively. The distance the ball traveled, time and angle the ball traveled for each kick were recorded. The ball was kicked about 26 times at 90 psi. When the tank pressure dropped to 1700 psi a series of sixty-five straight kicks were performed to illustrate the tank pressure drop over time. The tank pressure was periodically recorded throughout the testing. Once the tank pressure dropped to 1500 psi the regulator was set to about 20 psi. Five left, right and straight kicks were performed. This process of 5 left, right, straight kicks was repeated for incrementally increasing regulator pressure until the regulator pressure reached 130 psi. The total testing time was recorded. Additionally, the temperature of the PCB board was also checked at the conclusion of testing.

5.3.2. MECHANICAL COMPONENTS

The table below indicates the average kicking angle for left and right kicks. The conclusion reached is that the angle directed by left and right kicks is approximately equal. This data was taken over approximately 40 left and right kicks.

<table>
<thead>
<tr>
<th>Kick Angle</th>
<th>Avg. Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>11.98</td>
</tr>
<tr>
<td>Right</td>
<td>11.68</td>
</tr>
</tbody>
</table>
The table below indicates the average velocity and distance traveled for each permutation taken of kick direction and regulator pressure.

Table 5.4: Average Speed and distance for regulator pressure and kick direction

<table>
<thead>
<tr>
<th>Pressure</th>
<th>Direction</th>
<th>Avg Speed (m/s)</th>
<th>Distance (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>70 Left</td>
<td></td>
<td>1.02</td>
<td>4.8</td>
</tr>
<tr>
<td>80 Left</td>
<td></td>
<td>0.84</td>
<td>4.3</td>
</tr>
<tr>
<td>90 Left</td>
<td></td>
<td>1.08</td>
<td>4.8</td>
</tr>
<tr>
<td>100 Left</td>
<td></td>
<td>1.21</td>
<td>5.0</td>
</tr>
<tr>
<td>110 Left</td>
<td></td>
<td>1.47</td>
<td>5.0</td>
</tr>
<tr>
<td>120 Left</td>
<td></td>
<td>1.39</td>
<td>5.0</td>
</tr>
<tr>
<td>130 Left</td>
<td></td>
<td>1.63</td>
<td>5.0</td>
</tr>
<tr>
<td>70 Right</td>
<td></td>
<td>0.57</td>
<td>2.9</td>
</tr>
<tr>
<td>80 Right</td>
<td></td>
<td>0.80</td>
<td>4.6</td>
</tr>
<tr>
<td>90 Right</td>
<td></td>
<td>0.58</td>
<td>3.4</td>
</tr>
<tr>
<td>110 Right</td>
<td></td>
<td>0.93</td>
<td>4.8</td>
</tr>
<tr>
<td>110 Right</td>
<td></td>
<td>0.88</td>
<td>4.5</td>
</tr>
<tr>
<td>120 Right</td>
<td></td>
<td>1.02</td>
<td>5.0</td>
</tr>
<tr>
<td>130 Right</td>
<td></td>
<td>1.05</td>
<td>5.0</td>
</tr>
<tr>
<td>24 Straight</td>
<td></td>
<td>0.34</td>
<td>1.0</td>
</tr>
<tr>
<td>41 Straight</td>
<td></td>
<td>0.51</td>
<td>2.9</td>
</tr>
<tr>
<td>50 Straight</td>
<td></td>
<td>0.53</td>
<td>3.4</td>
</tr>
<tr>
<td>60 Straight</td>
<td></td>
<td>0.59</td>
<td>3.9</td>
</tr>
<tr>
<td>70 Straight</td>
<td></td>
<td>0.66</td>
<td>4.8</td>
</tr>
<tr>
<td>80 Straight</td>
<td></td>
<td>0.87</td>
<td>4.8</td>
</tr>
<tr>
<td>90 Straight</td>
<td></td>
<td>0.78</td>
<td>5.0</td>
</tr>
<tr>
<td>92 Straight</td>
<td></td>
<td>0.95</td>
<td>5.0</td>
</tr>
<tr>
<td>100 Straight</td>
<td></td>
<td>1.07</td>
<td>5.0</td>
</tr>
<tr>
<td>110 Straight</td>
<td></td>
<td>1.18</td>
<td>5.0</td>
</tr>
<tr>
<td>120 Straight</td>
<td></td>
<td>1.11</td>
<td>5.0</td>
</tr>
<tr>
<td>130 Straight</td>
<td></td>
<td>1.26</td>
<td>5.0</td>
</tr>
</tbody>
</table>
The figure above indicates kick angle vs. regulator pressure. It appears based on the sporadic plotting that there is no correlation between pressure and kick angle. By observation, it appeared that kick angle was more dependent on where the ball struck the kicking plate. The average kick angle as mentioned in the previous table was approximately 12 degrees.
The figure above is a plot of the kicking distance vs. regulator pressure. It shows a linear relationship until the distance reached is 5 m. In the initial system design the robot was designed to kick a minimum of 5 meters. Once this distance was measured the time was recorded. No kicking distance greater than 5 m was needed for the test. Each plot on the graph represents approximately 5 kicks. Based on the plot, the pressure regulator should be set at 90 psi to consistently reach the 5 m requirement.
Figure 5.9: Kick Velocity vs. Pressure

The figure above displays the average velocity of the ball vs. the pressure regulator setting. The left kick consistently showed the greatest velocity at each setting while the right kick was the weakest. This is most likely due to increased friction during right kicks. The increased friction is due to a defect machining process. One side of the pinned joints is not machined as smooth as the other side which results in increased friction during a right kick.
The plot above displays the pressure in the tank obtained during a series of straight kicks between 1700 and 1500 psi. The tank provided 35 kicks during that time interval. The theoretical number of kicks using the ideal gas law is only 13. This is most likely due to the compressibility of the gas. At high pressures the compressibility is greater than one as shown in Figure 3.5. The ideal gas law assumes a compressibility of unity. Since the gas was assumed to be an ideal gas and the actual compressibility is greater than one the theoretical analysis underestimates the actual number of kicks.

During the testing the ball was kicked about 181 times with an initial tank pressure of 2100 psi to a final pressure of 1000 psi. The robot was on for two hours. The PCB board did not heat up and remained at room temperature.
5.4. SOFTWARE PROCEDURE

As mentioned previously, 3 main different codes were developed throughout the design to command the robot to achieve individual tasks which include localizing the ball and following it, localizing the goal, and calculating and moving to a kick point when both goal and ball are seen. Each of these codes was tested individually by simply running in the appropriate environment. To test the first code that commands the robot to track and follow an orange ball, we trained the robot to recognize an orange ball using the ACTS and we ran the code on the robot in a room well lit. Finally we verified that the robot consistently tracks the ball and behaves as expected by the code.

Similarly, we tested the second and the third codes by running them on the robot with a blue object—which represents the goal—and the orange ball placed appropriately in room well lit. In order to test the third code that commands the robot to compute to kick point and move to it, we placed the robot at a chosen initial position 3 meters straight ahead from the center of the goal. Then we placed the ball anywhere both the ball and the robot could be seen by the camera, and finally we ran the robot and verified that the robot moves to the expected location.

The final part of the software testing consisted of verifying that the final code developed for the entire system (the robot with the kicker) will command the robot to perform the same tasks as the individual codes described before and kick the ball when instructed to.

Initialization Procedure:

Below is basic procedure that must be taken in order to run a code on the robot.

a. Power up the robot

b. Start the onboard computer and run the appropriate ACTS configuration

c. Remotely access the robot onboard computer over the wireless network using TightVNC

d. Run the code
The testing process of the entire system focuses on various aspects which include the robot-system mobility, the kicking motion, the battery-pressure relationship, the battery life and the performance of the overall system. Below are some test cases that we used.

**System Mobility:**

Step 1: Initialization procedure

Step 2: verify that the robot consistently tracks the ball and moves to the ball as instructed by the code running.

Step 3: Repeat step 2 five times.

**Kicking Motion:**

Step 1: Initialization process

Step 2: verify on the user interface that the kick command ('s' for straight kick) is sent as soon as the ball is seen

Step 3: Verify that robot approaches the ball until it is seen by the proximity sensors. A red led on top of the sensors lights up when it sees an object.

Step 4: Verify that both cylinders of the kicker are actuated simultaneously as soon as the red led is seen on the sensor.

Step 5: repeat steps 2 to 4 two more times for the ‘l’ (left) and ‘r’ (right) commands. In step 4 verify that only one cylinder is actuated at the time.

**Battery-Pressure Relationship:**

In order to determine how much the battery level affects the pressure that is given out for each kick, we will execute several kicks and measure the speed and the distance of each of them, then we will check if overtime we observe a decrease in the speed or the distance traveled by the ball.

**Battery Life:**
We will test the battery life of the robot with the following procedure:
Step 1: Fully charge the robot
Step 2: Start the robot and a stop watch
Step 3: Engage the robot and the kicker
Step3: wait for the power led of the robot starts blinking and stop the watch.

System Overall Performance:
Step 1: Initialization procedure
Step 2: Track the number of miss hits
Step 3: track the kick failures.
Step 4: track the serial connection failure by looking at the log data on the user interface window.
Step 5: check the accuracy of the kicks; how straight, left or right are they?

6. EVALUATION AND RECOMMENDATION

6.1. EVALUATION

At the beginning of the project, we committed our team to designing and building a kicking mechanism for the Pioneer 3-DX robot, and then program it to find the ball, orient itself to the goal, and finally kick the ball in the goal’s direction. We successfully built a kicker which kicks the ball at least 5 m away as we committed to; we programmed the robot to localize an orange ball and move towards it until it senses the ball and kick it. Furthermore, with our collaboration with the software engineering team, we could command the robot to find an orange ball and a blue goal, and then calculate the distances and angles to a kick point where the robot will be facing the goal move the robot to the kick point, and finally kick the ball in the direction of the goal.

Overall, our system works consistently as long as the robot stays connected to the access point and all the connections ports—serial, TCP, IP through—of the robot stay open.
Throughout the project, we made several decisions and choices; some of which were the best according to the constraints that we have, others could have been better. We encountered some difficulties and learned more about what was needed in order to reach our goal. In this section of the report we will point some aspect of the projects that we believe could improve the result of our project or help the next team improve our designs.

First, a smaller sensor that has color capability could be used instead. The one chosen was a little larger than necessary and affected the presentation of the robot. A smaller size sensor will provide future project flexibility in where to place it. Also the current sensor can detect any object that is within its sensing range; however, being able to find a sensor that can detect only the ball or the color of the ball will enhance future design.

In addition, the PCB board could be professionally made to ensure it is built properly. It would be ideal to have different voltage layers, a ground layer, at least two copper layers, and a silk screen layer so the screw terminals can be labeled so the wires will not be attached to the wrong parts. Sending it out to be professionally made will also eliminate the need to debug the traces because the company will already debug it for you.

Furthermore, the use of an electrically controllable pressure regulator will allow a variation of the kicking power, which could be needed for the final development of robots for the competition.

Finally, if robot could not be built in its entirety, the base robot that is purchased must be faster and poster triangular wheels emplacement allowing the robot to move in various direction without having to turn around. It should also have a shorter length and more ground clearance, so that a more powerful kicking mechanism with a longer stroke could be attached. The base robot must have some sort of vision capability that could be easily enhanced to omnivision by adding some mirrors and achieving a better vision over the field of play.
7. CONCLUSION

The main purpose of our project was to design a kicking mechanism and a microcontroller based unit to interface and control the kicker with the provided Pioneer 3-DX (P3-DX) base robot. Several conceptual designs were first generated and evaluated based on several criteria. The design selected during the first phase of our project consisted of two pneumatic cylinders utilizing compressed nitrogen and controlled by a PIC18F458 microcontroller based interface unit. The pneumatic cylinders bolted underneath the P3-DX and the nitrogen tank mounted on top of the robot. Two proximity sensors mounted in the front of the robot to aid in the detection of the soccer ball.

During the second and final stage of our design, we built the kicking mechanism that was designed during the first stage of the project. During the building process several modification were made to our design in order to make our final product more efficient and reduce cost. The kicking mechanism built consisted of two pneumatic cylinders using nitrogen gas and controlled by a PIC18F458 microcontroller. The pneumatic cylinders were bolted underneath the P3-DX and the nitrogen tank mounted on top of the robot. One proximity sensor mounted in the front of the robot to aid in the detection of the soccer ball.
8. REFERENCES

Automation Direct, automationdirect.com

http://www.copper.org/applications/plumbing/techref/cth/cth_main.htm


Matweb.com


Pneumatic Cylinder Data Sheet,
http://www.parker.com/literature/Literature%20Files/actuator/Cat/english/0900P1A.pdf
http://www.roymech.co.uk/Useful_Tables/Form/Weld_strength.html


Solenoid Valve Data Sheet,


9. LIST OF FIGURES

Figure 3.1: Cylinders and kicking plate. Right diagram shows side kick.................... 7
Figure 3.2: Free body diagram ................................................................................. 8
Figure 3.3: Schematic of Pneumatic System ............................................................ 10
Figure 3.4: CAD Drawing of Final Design ............................................................. 10
Figure 3.5: Compressibility Chart for Nitrogen ....................................................... 12
Figure 3.6: Tank pressure after each kick for air.................................................... 13
Figure 3.7: Displacement, velocity, and acceleration of pneumatic cylinder. (b=150kg/s) ............................................................................................................. 18
Figure 3.8: Stress on kicking plate .......................................................................... 19
Figure 3.9: Extended Cylinder Piston Rod ............................................................ 0
Figure 3.10: Stress concentration factors for rectangular plate with central hole (Source: Hamrock) ............................................................................................................. 22
Figure 3.11: DRV104 Chip interface circuit .......................................................... 27
Figure 3.12: Capacitive proximity sensor .............................................................. 28
Figure 3.13: Capacitive proximity sensor mounted on the 3-DX robot ................ 29
Figure 3.14: PNP output wiring diagrams ............................................................... 30
Figure 3.15: DB9 male and female end connectors ............................................... 32
Figure 3.16: RS232/Serial Interface circuit ............................................................ 32
Figure 3.17: Crystal oscillator interface circuit with the PIC18F458 .................... 33
Figure 3.18: Power-On Reset Circuit—For Slow VCC Power up ......................... 34
Figure 3.19: PWM output cycle.............................................................................. 0
Figure 3.20: Old Schematic of the Control Unit Interface ...................................... 0
Figure 3.21: Final Schematic of the Control Unit PCB ............................................ 37
Figure 3.22: Final PCB Board Design (Designed by Charles Craft II) ................. 38
Figure 4.1: Wooden Prototype ........................................................................................................ 39
Figure 4.2: Kicking plate attached to prototype ......................................................................... 40
Figure 4.3: Cylinders Bolted to Under Pioneer 3-DX ............................................................... 41
Figure 4.4: Front View of Kicker ................................................................................................ 41
Figure 4.5: Final Design .............................................................................................................. 42
Figure 4.6: MPLAB ICD 2 Connections to the PIC18F458 and Pin Numbering for the Phone Jack Connector ......................................................................................................................... 44
Figure 4.7: Programming Prototype PCB board of the control Unit ........................................ 46
Figure 4.8: Final PCB of the Control Unit .................................................................................. 47
Figure 5.1: PNP output wiring diagrams .................................................................................... 50
Figure 5.2: Pin diagram of PIC18F458 ..................................................................................... 51
Figure 5.3: Sensor testing diagram ............................................................................................ 54
Figure 5.4: Transistor circuit testing ........................................................................................... 58
Figure 5.5: Solenoid Driver Final Circuit (Designed by Charles Craft II) ............................... 60
Figure 5.6: Screen shot of the HyperTerminal during Kicker Testing ....................................... 62
Figure 5.7: Kicking angle vs. Pressure ....................................................................................... 65
Figure 5.8: Distance Traveled vs. Pressure ............................................................................... 66
Figure 5.9: Kick Velocity vs. Pressure ....................................................................................... 67
Figure 5.10: Theoretical and Experimental Tank pressure ......................................................... 68
10. LIST OF TABLES

Table 2.1: List of Variables Used ................................................................. 5
Table 3.1: Link/Joint Description ................................................................. 7
Table 3.2: Fluid Properties (Source: Sontag) .............................................. 11
Table 3.3: Pressure in pipe system .............................................................. 16
Table 3.4: Voltage needed to power different components ...................... 31
Table 5.1: Sensor Testing Result ................................................................. 55
Table 5.2: Transistor circuit testing ............................................................ 57
Table 5.3: Average Kicking Angle .............................................................. 63
Table 5.4: Average Speed and distance for regulator pressure and kick direction........ 64
This code was run in Visual Studio on a laptop to test the serial communication.

```c
#include <Aria.h>

int main(int ac, char **av)
{
    ArSerialConnection serial;
    char inbuf[1024];
    int rv;

    serial.setBaud(19200);
    rv = serial.open("COM4");
    printf("Open returned %i\n", rv);
    printf("Enter 'q' to quit.\n");

    while(true) {
        printf("Input> ");
        fscanf(stdin, "%s", inbuf);

        if(inbuf[0] == 'q') return 0;

        printf("Writing %s\n", inbuf);
        rv = serial.write(inbuf, 1);
        printf("Write returned %i\n", rv);
        memset(inbuf, 0, sizeof(inbuf));

        printf("Reading...\n");
        rv = serial.read(&inbuf[0], sizeof(inbuf));
        printf("Read returned %i\nRead data: %s", rv, inbuf);
    }
}
```
This code was downloaded onto the PIC18F458

//==================================================================================================
// Filename: serialIntrf.c
//==================================================================================================
// Author:   Landry Nzudie
// Company:  Purdue University, Fort Wayne- Engineering Department
// Revision: 1.00
// Date: 03/04/2008
// Description:
//==================================================================================================
// Compiled using MPLAB-C18

#include <p18f458.h>
#include <timers.h>
#include "usart.h"

#pragma config WDT = OFF
#pragma config OSC = HS

/******************************************************************************
 * Function Name: delay
 * Return Value: None
 *
 * Parameters: None
 * Description: This routine causes a delay using the For loop
 */
void delay(int time)
{
    int i;

    for (i = 0; i < time; i++) ;
}

/******************************************************************************
 * Function Name: ReadUSART
 * Return Value: char: received data
 *
 * Parameters: void
 * Description: This routine reads the data from the USART and records the status flags for that byte in USART_Status (Framing and Overrun).
 */
char ReadUSART(void) //this function can be removed by macro
#define ReadUSART RCREG
{
    char data; // Holds received data
    USART_Status.val &= 0xf2; // Clear previous status flags
    if(RCSTAbits.RX9) // If 9-bit mode
    {
        USART_Status.RX_NINE = 0; // Clear the receive bit 9 for USART
        if(RCSTAbits.RX9D) // according to the RX9D bit
            USART_Status.RX_NINE = 1;
    }
    if(RCSTAbits.FERR) // If a framing error occurred
        USART_Status.FRAME_ERROR = 1; // Set the status bit
    if(RCSTAbits.OERR) // If an overrun error occurred
        USART_Status.OVERRUN_ERROR = 1; // Set the status bit
    data = RCREG; // Read data
    return (data); // Return the received data
}

/********************************************************************
* Function Name:  WriteUSART
* Return Value:   none
* Parameters:     data: data to transmit
* Description:    This routine transmits a byte out the USART.
********************************************************************/
void WriteUSART(char data)
{
    if(TXSTAbits.TX9) // 9-bit mode?
    {
        TXSTAbits.TX9D = 0; // Set the TX9D bit according to the
        if(USART_Status.TX_NINE) // USART Tx 9th bit in status reg
            TXSTAbits.TX9D = 1;
    }
    TXREG = data; // Write the data byte to the USART
}

/********************************************************************
* Function Name:  OpenUSART
* Return Value:   void
* Parameters:     config: bit definitions to configure USART
* spbrg: baudrate value for register SPBRG1
* or for SPBRGH:SPBRG for 16-bit baud rate generation for applicable parts
* Description:    This routine first resets the USART regs to the POR state. It then configures the
*                 USART for interrupts, synch/async, 8/9-bit, sync slave-master and single/cont. rx.
* Notes:          The bit definitions for config can be found
********************************************************************/
void OpenUSART( unsigned char config, unsigned int spbrg)  
{  
    TXSTA = 0; // Reset USART registers to POR state
    RCSTA = 0;

    if(config&0x01) // Sync or async operation
        TXSTAbits.SYNC = 1;

    if(config&0x02) // 8- or 9-bit mode
    {  
        TXSTAbits.TX9 = 1;
        RCSTAbits.RX9 = 1;
    }

    if(config&0x04) // Master or Slave (sync only)
        TXSTAbits.CSRC = 1;

    if(config&0x08) // Continuous or single reception
        RCSTAbits.CREN = 1;
    else
        RCSTAbits.SREN = 1;

    if(config&0x10) // Baud rate select (asynchronous mode only)
        TXSTAbits.BRGH = 1;
    else
        TXSTAbits.BRGH = 0;

    PIR1bits.TXIF = 0;
    if(config&0x40) // Interrupt on receipt
        PIE1bits.RCIE = 1;
    else
        PIE1bits.RCIE = 0;

    PIR1bits.RCIF = 0;
    if(config&0x80) // Interrupt on transmission
        PIE1bits.TXIE = 1;
    else
        PIE1bits.TXIE = 0;

    SPBRG = spbrg; // Write baudrate to SPBRG1
    TXSTAbits.TXEN = 1; // Enable transmitter
    RCSTAbits.SPEN = 1; // Enable receiver

    TRISCbits.TRISC6 = 0;TRISCbits.TRISC7 = 1;
    if(TXSTAbits.SYNC && !TXSTAbits.CSRC) //synchronous slave
        TRISCbits.TRISC6 = 1;
}

void main (void)
{  
    char command;
int kick = 0;
//configure USART
OpenUSART( USART_TX_INT_OFF &
          USART_RX_INT_OFF &
          USART_ASYNCH_MODE &
          USART_EIGHT_BIT &
          USART_CONT_RX &
          USART_BRGH_HIGH, 64 );

//Set I/O pins for the Kicker driver as output
TRISCbits.TRISC5 = 0;
TRISDbits.TRISD4 = 0;

//Set I/O pins for the Sensors Input
TRISCbits.TRISC1 = 1;
TRISCbits.TRISC2 = 1;

while (1)
{
    if (kick == 0) {
        while (!DataRdyUSART()); //wait for the data to 
        be received
        command = ReadUSART();
        if (kick == 0) { //entering kicking 
            state
                if (command == 's' )
                    kick = 1;
                else if (command == 'l' )
                    kick = 2;
                else if (command == 'r' )
                    kick = 3;
                else
                    kick = 0;
            }
        }
        if (kick == 1 && PORTCbits.RC1 == 1){
            PORTCbits.RC5 = 1;
            PORTDbits.RD4 = 1;
            if(!BusyUSART()){  //Check if the USART is
                ready for transmission
                WriteUSART('p');  //send confirmation
                message back
                while (BusyUSART());  //wait for the data to
                be transmitted
            }
        }
        delay(5000);
        PORTCbits.RC5 = 0;
PORTDbits.RD4 = 0;
kick = 0;
//CloseUSART();
}
else if (kick == 2 && PORTCbits.RC1 == 1){
    PORTCbits.RC5 = 0;
    PORTDbits.RD4 = 1;
    if(!BusyUSART()){  //Check if the USART is ready for transmission
        WriteUSART('p');  //send confirmation message back
        while (BusyUSART()); //wait for the data to be transmitted
    }
    delay(5000);
    PORTCbits.RC5 = 0;
    PORTDbits.RD4 = 0;
    kick = 0;
    //CloseUSART();
}
else if (kick == 3 && PORTCbits.RC1 == 1){
    PORTCbits.RC5 = 1;
    PORTDbits.RD4 = 0;
    if(!BusyUSART()){  //Check if the USART is ready for transmission
        WriteUSART('p');  //send confirmation message back
        while (BusyUSART()); //wait for the data to be transmitted
    }
    delay(5000);
    PORTCbits.RC5 = 0;
    PORTDbits.RD4 = 0;
    }
    }
else
    }
}
This code runs on the onboard computer of the Pioneer 3-DX robot.

```c
#include "Aria.h"

/* Constants -- see the class for explanations of what these are */
#define TIMESTAMP_EXPIRATION 25
#define CHANNEL 1
#define MAX_DELTA_HEADING 60
#define VELOCITY 300
#define LOST 1
#define FOLLOW 2
#define GUESS 3

/* Version numbers don't belong in class names... */
typedef ArACTS_1_2 ArACTS;

/* The width of the captured image in pixels (Used to calculate
heading) */
const int width = 160;

//serial connection
ArSerialConnection serial;

char kickresponse[1024];
char kickcommand[1024];
int rv ;
int rv1 ;
int count = 0;

class GotoBall: public ArAction
{
public:
    GotoBall(ArACTS *acts, ArVCC4 *camera);

    ~GotoBall();

    /* Moves toward the ball. If it isn't visible, uses the last
position it saw */
    ArActionDesired *fire(ArActionDesired currentDesired);

protected:
    ArActionDesired desired;
    ArACTS *acts;
    ArVCC4 *camera;

    /* The relative position on the screen where the ball was last
seen */
    double lastPosition;
    /* The time (relative to this->time) that the ball was last seen */
    int lastPositionTimestamp;
    /* The current 'time' (incremented each time fire is called) */
    int time;
}```
/* How many ticks until the ball is considered 'lost' */
const int timestampExpiration;

/* The ACTS channel that will recognize the ball */
const int channel;

/* The most the robot will attempt to turn in a single fire call */
const int maxDeltaHeading;

/* The speed the robot will move */
const int velocity;

/* Finds the index of the largest blob in view (-1 if none are in view) */
gotoBlob();
char KickBall(char kickdirect[1024]);

GotoBall::GotoBall(ArACTS *acts, ArVCC4 *camera):
    acts(acts),
camera(camera),
timestampExpiration(TIMESTAMP_EXPIRATION),
channel(CHANNEL),
maxDeltaHeading(MAX_DELTA_HEADING),
velocity(VELOCITY),
ArAction("GotoBall", "Goes to the ball")
{
    time = 0;
    lastPosition = 0;
    lastPositionTimestamp = -1;
}

GotoBall::~GotoBall() { }

int GotoBall::getBlob()
{
    ArACTSBlob blob;
    int numBlobs;
    int aMax = 75, iMax = -1;

    static int lastNumBlobs = -2;

    numBlobs = acts->getNumBlobs(channel);
    if(numBlobs != lastNumBlobs) {
        ArLog::log(ArLog::Normal, "NumBlobs: %i", numBlobs);
        lastNumBlobs = numBlobs;
    }

    if(numBlobs > 100) {
        return -1;
    }

    for(int i = 0 ; i < numBlobs; i++) {
        acts->getBlob(channel, i+1, &blob);
        ArLog::log(ArLog::Normal, "Area: %i", blob.getArea());
    }
if(blob.getArea() > aMax) {
    iMax = i + 1;
    aMax = blob.getArea();
}

return iMax;
}

char GotoBall::KickBall(char kickdirect[1024])
{
    //rv =-1;
    //kickresponse[0] = 'f';
    //kickcommand[0] = 's';
    ArLog::log(ArLog::Normal, "Sending the signal %s to kick the ball.",kickdirect);
    //while (rv == -1){
    rv = serial.write(kickdirect, 1);
    ArLog::log(ArLog::Normal, "Write returned %i", rv);
    //}

    //memset(kickdirect, 0, sizeof(kickdirect));

    //while (rv1 == -1){
    rv1 = serial.read(&kickresponse[0], sizeof(kickresponse));
    ArLog::log(ArLog::Normal, "Read returned %i Read data: %s", rv1, kickresponse);
    return kickresponse[0];
    //}
}

ArActionDesired *GotoBall::fire(ArActionDesired currentDesired)
{
    ArACTSBlob blob;
    int whichBlob = getBlob();
    double xRel;
    static int state = 0;

    /* Reset desired action */
    desired.reset();

    /* Increment the timer */
    time++;

    /* No blob is found */
    if(whichBlob == -1) {
        if( (time - lastPositionTimestamp) > timestampExpiration) {
            if(state != LOST) {
                ArLog::log(ArLog::Normal, "Ball lost. (Set to wander mode?)");
            }
            state = LOST;
        }
    } /* Move slowly in random directions */
    desired.setVel(0);  
    if(lastPosition < 0)
        desired.setDeltaHeading(30.0);
else    desired.setDeltaHeading(-30.0);
    return &desired;
}
if(state != GUESS) {
    ArLog::log(ArLog::Normal, "Ball not found; using last known position:%f", lastPosition);
    state = GUESS;
}
    xRel = lastPosition;
} else {
    if(state != FOLLOW) {
        ArLog::log(ArLog::Normal, "Ball is visible");
        state = FOLLOW;
    }
    acts->getBlob(channel, whichBlob, &blob);
    /* Get the position of the ball's center of gravity relative to the center of the 'screen' */
    xRel = (blob.getXCG() - width / 2.0) / width;
    /* Update the last known position and timestamp */
    lastPosition = xRel;
    lastPositionTimestamp = time;
    //
    if (count == 0) {
        kickcommand[0] = 's';
        count = 1;
    } else if (count == 1) {
        kickcommand[0] = 'l';
        count = 2;
    } else {
        kickcommand[0] = 'r';
        count = 0;
    }
    //Kick
    KickBall(kickcommand);
    /*while(count < 0){
    if (KickBall(kickcommand) == 'p'){
        count++;
        if (kickcommand[0] == 's')
            kickcommand[0] = 'l';
        else if(kickcommand[0] == 'l')
            kickcommand[0] = 'r';
        else
            kickcommand[0] = 's';
    }*/
}
if (ArMath::fabs(-xRel * 10) <= maxDeltaHeading) {
    /* If it's not too far on the edge... */
    desired.setDeltaHeading(-xRel * 10);
} else if (-xRel > 0) {
/* If it is, and it's to the right */
desired.setDeltaHeading(maxDeltaHeading);
} else {
/* If it is, and it's to the left */
desired.setDeltaHeading(-maxDeltaHeading);
}

desired.setVel(velocity);
return &desired;

// Main function
int main(int argc, char** argv)
{
    Aria::init();

    // The robot
    ArRobot robot;

    // A key handler to take input from keyboard
    ArKeyHandler keyHandler;

    // Sonar for basic obstacle avoidance
    ArSonarDevice sonar;

    // The camera (Cannon VC-C4)
    ArVCC4 vcc4 (&robot);

    // ACTS, for tracking blobs of color
    ArACTS_1_2 acts;

    // command line arguments
    ArArgumentParser argParser(&argc, argv);
    argParser.loadDefaultArguments();

    // The simple way to connect to things (takes arguments from argParser)
    ArSimpleConnector simpleConnector(&argParser);

    // Open serial port
    serial.setBaud(19200);
    rv = serial.open("COM5");
    ArLog::log(ArLog::Normal, "Open returned %i\n", rv);

    // Parse the arguments
    if (!Aria::parseArgs())
    {
        Aria::logOptions();
        keyHandler.restore();
        Aria::shutdown();
        return 1;
    }

    // Robot motion limiter actions (if obstacles are detected by sonar)
    ArActionLimiterForwards limiter("speed limiter near", 300, 600, 250);
ArActionLimiterForwards limiterFar("speed limiter far", 300, 1100, 400);
ArActionLimiterBackwards backwardsLimiter;
ArActionConstantVelocity stop("stop", 0);
ArActionConstantVelocity backup("backup", -200);

// The color following action, defined above
GotoBall gotoBall(&acts, &vcc4);

// Let Aria know about the key handler
Aria::setKeyHandler(&keyHandler);

// Add the key handler to the robot
robot.attachKeyHandler(&keyHandler);

// Add the sonar to the robot
robot.addRangeDevice(&sonar);

// Connect to the robot
if (!simpleConnector.connectRobot(&robot))
{
    printf("Could not connect to robot... exiting\n");
    keyHandler.restore();
    Aria::shutdown();
    return 1;
}

// Open a connection to ACTS
acts.openPort(&robot);

// Initialize the camera
vcc4.init();

// Wait a second.....
ArUtil::sleep(1000);

// Artificially keep the robot from going too fast
robot.setAbsoluteMaxTransVel(400);

// Enable the motors
robot.comInt(ArCommands::ENABLE, 1);

// Turn off the amigobot sounds
robot.comInt(ArCommands::SOUNDTOG, 0);

// Wait....
ArUtil::sleep(200);

// Add the actions to the robot in descending order of importance.
robot.addAction(&limiter, 100);
robot.addAction(&limiterFar, 99);
robot.addAction(&backwardsLimiter, 98);
robot.addAction(&gotoBall, 77);
robot.addAction(&backup, 50);
robot.addAction(&stop, 30);

// Run the robot processing cycle until the connection is lost
robot.run(true);
Aria::shutdown();
return 0;
}

11.3. PIPE FLOW PROGRAM EES

FUNCTION velocity(backPressureRatio, criticalPressureRatio, c)
  IF (backPressureRatio < criticalPressureRatio) THEN
    velocity := c
  endif
{This program will calculate the pressure drop and mass flow rate in the pipe system}

P_b = 101

Diameter of orifice = 1/16 * 2.54 / 100
D = .402 * 2.54 / 100
diameter_orifice = 1/16 * 2.54 / 100
"diameter of pipe"

"State 1 Valve Inlet"
T[1] = 20 + 273.15
P[1] = 620.528156 [kPa] + P_atm
rho[1]= Density(fluid$, T=T[1], P=P[1])
A[1]=pi/4*D^2
P_0=P[1]
m_dot = rho[1]*V[1]*A[1]
s[1]= Entropy(fluid$, T=T[1], P=P[1])
h[1] = Enthalpy(fluid$, T=T[1])

"Stagnation Temperature"
T_0=T[1]

"State 2 Valve Throat"
V[2] = velocity(backPressureRatio, criticalPressureRatio, c[2])
"Mach number"
h[2] = Enthalpy(fluid$, T=T[2])
s[2]
"(T_0/T[2])^((k/(k-1))=P_0/[P[2]]
(P_0/P[2] = (1+(k-1)/2*M[2]^2)^((k/(k-1))

Isentropic Flow"
"State 3 Valve Exit"
m_dot = rho[3]*V[3]*A[3]
rho[3]=Density(fluid$,T=T[3],P=P[3])
c[3]= SoundSpeed(fluid$,T=T[3])
h[3] =h[1]
P_0/P[3] = (1+(k-1)/2*M[3]^2)^k/(k-1)
mu[3] = Viscosity(fluid$,T=T[3])
e=0.0015*10^(4*(3-
1/f[3]*(0.5) = -2.0 * log10((e/D)/3.7) + 2.51/(Re[3] * f[3]*(0.5))
L_34 = 2 *12 *2.54/100

"State 4 Pipe"
"Frictional Adiabatic Flow in a constant-area channel"

"State 5 Entrance of cylinder"
D_5 = 1/8 * 2.54 / 100
L_45 = 2 * 2.54 / 100
mu[5] = Viscosity(fluid$,T=T[5])
1/f[5]*(0.5) = -2.0 * log10((e/D_5)/3.7) + 2.51/(Re[5] * f[5]*(0.5))
"Contraction Loss"
h_licont = 0.5 * V[5]^2/2
"90 deg elbow"
"Major Losses"
"Total Losses"
h_[l][5] = h_l5 + h_licont + h_lelbow

"State 6 Cylinder"
A[6] = pi/4*0.025^2
"V[6] = 1.8 "
L_56 = 0.05 [m]
Resistance = (P[1] - P[5]) / m_dot * 1000 [Pa/kPa]


11.4. TANK_ANALYSIS.M

% This program will calculate the air pressure in the tank after each kick
% assuming a constant kicking force is applied. It is assumed that the
% process is isothermal and the fluid is an ideal gas.
clear all; close all; clc;
kicks = 200;       % Number of kicks
% Properties of fluid
R = 0.2968;  % [kJ/kg-K]
T = 293.15; % [K] temperature
k = 1.399;
Z = 1;   % Compressibility factor
cylinders_fired = 2;

% Tank Specs
V_tank = 7.86579072e-4; % [m^3]
p_tank = linspace(0,0,kicks);
p_tank(1) = 3000 * 6.89475729 + 101; % [kPa] initial absolute tank pressure
mass_tank = linspace(0,0,kicks);
mass_tank(1) = p_tank(1) * V_tank / (R*T*Z);  % [kg] initial mass of fluid in tank

% Cylinder Properties
A_c = 0.025^2 * pi/4; % [m^2] Cross Sectional area of piston
length_cylinder = 0.131; % [m]
p_kick = linspace(90 * 6.89475729 + 101, 90 * 6.89475729 + 101,kicks);
mass_cylinder = linspace(0,0,kicks);
mass_cylinder(1) = p_kick(1) * A_c * length_cylinder / (Z*R*T);
pipe_diameter = 1/8 * 2.54 / 100;
pipe_length = 4 * 2.54 / 1000;
vol_exhausted = A_c * length_cylinder + pi/4 * pipe_diameter^2 * pipe_length;

% Pressure in tank
for z = 2:kicks
    if p_kick(z) >= p_tank(z-1)
        p_kick(z) = p_tank(z-1);
    end
    mass_cylinder(z) = p_kick(z) * cylinders_fired* vol_exhausted / (Z*R*T);
    mass_tank(z) = mass_tank(z-1) - mass_cylinder(z-1);
    p_tank(z) = mass_tank(z) * R * T * Z / V_tank;
    if p_tank(z) <= 101
        z   % last kick
        break
    end
end
n = linspace(1,z,kicks);
p_tank_gage = (p_tank -101) * 0.145037738;  % convert kPa to psig
plot(n,p_tank_gage, '.')
title('Tank Pressure vs Number of Kicks'); grid on;
xlabel('Kick'); ylabel('Tank Pressure (psi)');
p_tank_gage = transpose(p_tank_gage);
11.5. CYLINDER RESPONSE

Cylinder Displacement

\[
\left( -b^2 RC + bm + 2 RC km \right) \sinh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) e^{-\frac{1}{2} \frac{tb}{m}} \\
\left( -m + b RC - k R^2 C^2 \right) \sqrt{b^2 - 4 mk} k \\
+ \frac{1}{(-m + b RC - k R^2 C^2) k} \left( -m + b RC - k R^2 C^2 \left( 1 - e^{-\frac{t}{RC}} \right) \right) \\
+ \cosh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) e^{-\frac{1}{2} \frac{tb}{m}} \left( m - b RC \right) \right) P_i A
\]

Cylinder Velocity

\[
\left( \frac{1}{2} \left( -b^2 RC + bm + 2 RC km \right) \cosh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) e^{-\frac{1}{2} \frac{tb}{m}} \\
\frac{1}{m \left( -m + b RC - k R^2 C^2 \right) k} \\
- \frac{1}{2} \left( -b^2 RC + bm + 2 RC km \right) \sinh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) b e^{-\frac{1}{2} \frac{tb}{m}} \right) \\
\left( -m + b RC - k R^2 C^2 \right) m \sqrt{b^2 - 4 mk} k \\
+ \frac{1}{(-m + b RC - k R^2 C^2) k} \left( -k RC e^{-\frac{t}{RC}} \right) \\
+ \frac{1}{2} \sinh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) \sqrt{b^2 - 4 mk} e^{-\frac{1}{2} \frac{tb}{m}} \left( m - b RC \right) \\
- \frac{1}{2} \cosh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 mk}}{m} \right) b e^{-\frac{1}{2} \frac{tb}{m}} \left( m - b RC \right) \right) P_i A
\]
Cylinder Acceleration

\[
\frac{1}{4} \left( \begin{aligned} &\left( -b^2 RC + b m + 2 R C k m \right) \sinh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) \sqrt{b^2 - 4 m k} \ e^{-\frac{t b}{m}} \\
&\frac{1}{2} \left( -b^2 RC + b m + 2 R C k m \right) \cosh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) b \ e^{-\frac{t b}{m}} \\
&\frac{1}{4} \left( -b^2 RC + b m + 2 R C k m \right) \sinh \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) b^2 \ e^{-\frac{t b}{m}} \\
&\frac{1}{4} \left( -b^2 RC + b m + 2 R C k m \right) \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) \left( b^2 - 4 m k \right) \ e^{-\frac{t b}{m}} (m - b RC) \\
&\frac{1}{2} \left( -b^2 RC + b m + 2 R C k m \right) \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) \sqrt{b^2 - 4 m k} \ b \ e^{-\frac{t b}{m}} (m - b RC) \\
&\frac{1}{4} \left( -b^2 RC + b m + 2 R C k m \right) \left( \frac{1}{2} \frac{t \sqrt{b^2 - 4 m k}}{m} \right) b^2 \ e^{-\frac{t b}{m}} (m - b RC) \end{aligned} \right) \right) P_A
\]
All dimensions in inches
11.6.2. TANK BRACKETS

All dimensions in inches
Need two