

Introduction

CHESSTER will play chess against a human opponent using a physical chess board rather than through a graphical user interface (GUI). It will be able to “see” its opponent’s moves using magnetic sensors and respond by moving its own pieces using a robotic arm. I will complete CHESSTER by May 1 and submit it as my course project.

CHESSTER JR. is the mid-semester checkpoint for CHESSTER. I will be demonstrating CHESSTER JR. at the *UML Botfest* technology exhibition on Saturday, Marcy 29, 2003. At that point, CHESSTER JR. will be able to move pieces on the chess board but will not be able to observe moves on the physical board. As a result, human players will play the game using a GUI and CHESSTER JR. will make the corresponding moves on a physical chess board. For demonstration purposes, the software can be configured to play an automatic match, with each move reflected on the physical chess board.

This document is a plan and schedule for CHESSTER JR. only, not the full CHESSTER project.

Hardware Description

CHESSTER JR. will use the Lynx5 Robotic Arm¹, an inexpensive four-axis (plus gripper) robotic arm. The Lynx5 will be controlled by the 8-Servo Controller prototype. A Handy Cricket will interface the servo controller to a Windows PC. Figure 1 shows this arrangement.

The Lynx5 servos will be powered by a 5V regulated power supply. The Handy Cricket battery pack will power both the cricket and the 8-Servo Controller to minimize electrical noise from the servos. Figure 2 is the schematic for the 8-Servo Controller.

The Lynx5 has a maximum reach of 11 inches. If the arm is placed two inches from the center of one side of the board and the board spaces are 1 inch square, then the furthest corner squares will be 10.1 inches away from the arm. So, the arm should be able to reach all positions on the board.

Bill of Materials:

1. Chess board, non-magnetic, 1 inch squares
2. Magnetic chess pieces
3. Lynxmotion Lynx5 Robotic Arm
4. Handy Cricket
5. 8-Servo Controller prototype
 1. MicroChip PIC16F84 microcontroller with 8servo firmware
 2. 8MHz resonator
 3. 0.1 μ F capacitor
 4. 4.7 μ F electrolytic capacitor
 5. 470 Ω resistor
 6. green LED
7. 5V, 1.9A power supply. Autec Power Systems DT25-1004.

¹<http://www.lynxmotion.com/15arm0.htm>

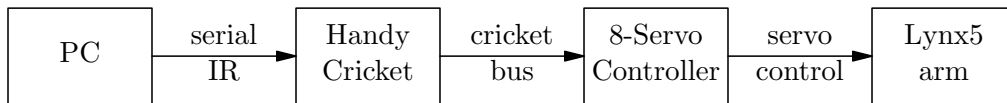


Figure 1: Hardware Block Diagram

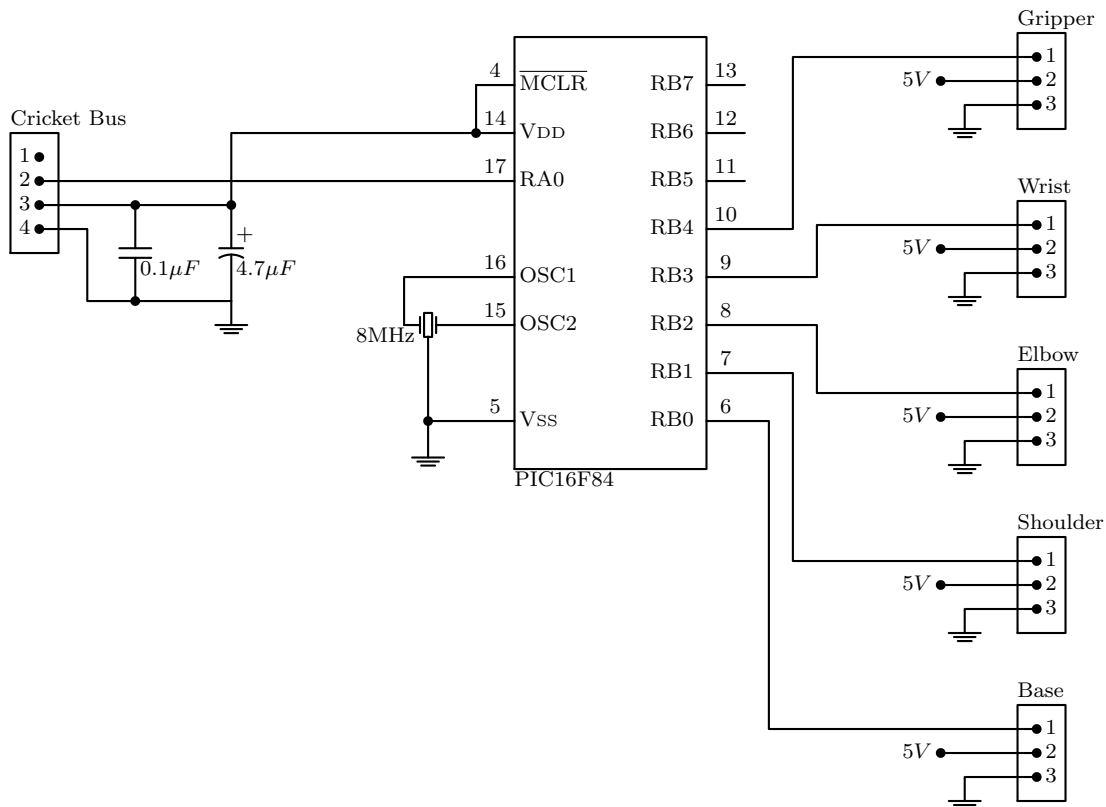


Figure 2: 8-Servo Controller Schematic



Figure 3: User versus GNU Chess

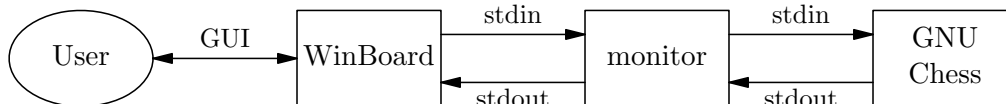


Figure 4: User versus GNU Chess, with monitoring

Software Description

I plan to use open source software for two major components: (1) GNU Chess² will be the chess engine that actually plays the game. Since GNU Chess is an active project in its fifth major release, I expect CHESSTER to play better than an easy exhibition game. (2) WinBoard³ will provide the game GUI and will connect the robotic components to GNU Chess.

GNU Chess has a simple console interface. So, typing “d2d4” to start a game moves the white queen’s pawn forward two spaces; GNU Chess responds with “My move is: d7d5”. This is just the beginning of a defacto standard for chess engines and Internet Chess Servers⁴.

WinBoard runs GNU Chess as a separate child process and uses anonymous pipes to communicate with the GNU Chess console. For example, figure 3 shows the two programs that are running when a person uses WinBoard to play against GNU Chess. This loose coupling makes it easy to insert a monitor program into the match (figure 4).

I have already prototyped the monitor program in Java using the Java Communications API⁵. It uses two threads: one to handle commands from WinBoard to GNU Chess and one to handle commands in the reverse direction. Both threads have an input stream and an output stream and run this algorithm: (1) Read a full line from the input stream. (2) If the line contains a move command, print the move to a separate window. (3) Write the line to the output stream (4) repeat.

The prototype monitor proves feasibility of using GNU Chess and WinBoard. The key development task for CHESSTER JR. is to extend the monitor prototype so that it instructs the robotic arm to move the physical pieces as the game progresses.

I plan to calculate the arm path in Java and send low level commands to the arm controller (such as “move servo 2 to position 110”). The Handy Cricket will simply forward commands from the PC to the 8-Servo Controller. An alternative would be to have the monitor send high-level commands to the arm controller (such as “move d2 to d4”) and calculate the arm path in Cricket Logo. I believe that approach would take longer to develop, so I have decided against it.

²<http://www.gnu.org/software/chess/>

³<http://www.tim-mann.org/xboard.html>

⁴<http://www.tim-mann.org/xboard/engine-intf.html>

⁵<http://java.sun.com/products/javacomm>

Tasks & Schedule

The following table gives the planned and actual completion dates for each of the tasks required to complete CHESSTER JR..

Task Description	Plan	Actual
Prototype a Java process interposed between WinBoard and GNU Chess.	—	3/12
Order the Lynx5 (Fred).	—	3/14
Install development software on a laptop that I can bring to the UML Botfest.	3/17	
Breadboard the 8-Servo Controller.	3/17	
Interface the 8-Servo Controller to the PC through a Handy Cricket.	3/17	
Build a prototype 2-DOF arm using Futaba servos and experiment with sending commands from the PC.	3/20	
Purchase chess board.	3/20	
Pickup Lynx5 from UML (assuming it arrives on time).	3/21	
Build the Lynx5.	3/22	
Mount Lynx5, chess board, servo controller, Cricket on plywood base.	3/23	
Experiment with Lynx5 position commands to get familiar with its capabilities. The greatest risk is that this is the first chance I will have to see if the arm is physically capable.	3/24	
Implement arm paths in Java.	3/27	
Test and debug.	3/28	