Wifi Connected Chess Boards

Project Requirement Specification

Nick Kraus, Kyle Jameson, Maurice Wallace, Mark Mauriello

October 20, 2015

Purpose

The goal of our project is to create a physical implementation of an internet chess game. The practical purpose of this project is to provide entertainment value to users by allowing them to play chess against others through an internet connection with a physical chess set. This project will allow users who have a more tactile style to have the best of both internet and traditional chess.

Definitions

- Embedded System: A computer system that is within and controls a larger mechanical or electrical system.
- Microcontroller: A small computer with programmable inputs and outputs.
- UART: A Universal Asynchronous Receiver Transmitter, a method to serially send data over electrical wires.
- Reed Switch: An electrical switch which triggers in the presence of a magnetic field.
- Limit Switch: A switch that triggers when an object collides with it.
- Gantry: A structure with a moving platform.
- Stepper Motor: An electric motor which divides a full rotation into a number of discrete equal steps.
- API: An application programming interface, which allows different software systems to interface with each other.
- Client Server Architecture: An architecture in which the clients (which run the users application) communicate with each other through a central server.
- Bare Metal: A term referring to software that runs directly on a processor without any supporting operating system.

System Overview

Our project will have four main systems which will integrate together as the final project. These are the electrical, mechanical, client, and server systems. The electronics will be designed to control the mechanical hardware, through an ARM Cortex-m microcontroller. This microcontroller will run the client software as its firmware; with the help of the Cypress peripheral libraries the client software will be able to directly interact with the mechanical system through the electronics. The client will also talk to the server with the help of an ESP8266 WiFi module, allowing the project to connect to the internet over wireless networks. The server software will be used to communicate between either two boards or a board and an PC client, allowing the users to connect and play with each other.

Electrical System

An embedded circuit board will house all the electrical components which will be used to run the client software, communicate with the server, and control the mechanical system. This system will be able to measure each chess space with a reed switch array embedded in the playing surface and decide if a piece is on the above space or not. These reed switches will be in a 8 by 12 array and read by the Cypress CY8C5268AXI microcontroller through its GPIOs. Two Allegro A4988 stepper motor drivers will be used to power the stepper motors which will drive the mechanical system. The electrical system will also contain an auxiliary adjustable power supply for an electromagnet which will be used to move pieces around the board, and a 20 by 4 character LCD screen and rotary encoder with center button for the user interface. The WiFi module also comes in a convenient package for direct connection to the circuit board.

Mechanical System

The mechanical system consists of the moving cartesian gantry below the playing surface that moves the electromagnet in order to move the chess pieces. The mechanical system consists smooth metal rods which will be held together by custom 3D printed plastic brackets, and the gantry will ride on bearings across the rods. Stepper motors, which have the ability to move in discrete lengths, will drive the gantry system with a high accuracy. A GT2 timing belt will be connected between the stepper motors and gantry to transfer the torque of the motors to the moving parts. The timing belt and gantry will be driven in an h-bot configuration, which ensures that no stepper motors will be on a moving piece which helps to increase speed and accuracy. The electromagnet will ride on the gantry and will pull pieces across the playing surface when activated by the embedded system. Two limit switches will be placed on the sides on the gantry system to enable calibration of the motors.

Client Software

The client software is the bare metal firmware on top of the microcontroller that runs the user interface, controls the mechanical system, and connects with the WiFi module. It is written in the C programming language using

libraries and APIs provided by Cypress for their ARM microcontroller family. A custom firmware also runs on the WiFi module, which uses the C++ programming language and open source community libraries that allow integration with the easy to use Arduino IDE. The Cypress libraries abstract away the accessing of registers in order to use the peripherals. The client software will implement many functions to interact with the underlying mechanical system, the WiFi module, and the user interface. To control the mechanical system a MoveGantry() function will be created which will move the gantry a set distance in the X and Y directions. A Home() function will also be implemented which resets the coordinate systems reference point by calibrating the stepper motors to the limit switches. In order to turn on the magnet to move pieces a SetMagnet() function will be implemented which can turn the magnet to either of its two states. A ScanBoard() function will be implemented to scan the reed switch array in order to determine if each piece does or does not have a piece on it. The client software will also implement higher level functions that will facilitate the actual playing of chess. Functions such as IsLegal() and IsCheckmate() will be used to determine if the last move played was a legal chess move, and if the last move played resulted in a checkmate, respectively. A MovePiece() function will be used to move an actual chess piece from one board position to the next, taking care of any logic necessary to either move in between other pieces or move other pieces out of the way, if necessary. This function will use some of the lower level functions to control the actual mechanical system.

Server Software

The server software will allow one chess board to communicate with either another chess board or a PC client. It will be a simple TCP server which gets formatted data sent from a client and responds accordingly. The server software will eventually become an executable that can be run by anyone looking to play a game of chess on their computer, and then connected to from either a board or PC client, by displaying the IP address that the server will be located at. For a basic game of chess the server will wait for a request from one of the boards, which will have a command string followed by the current state of the board at the end of that players turn (the game of chess is completely defined by its current state). The server will then relay this information to the opponent, whose board will replace its current state and make the appropriate moves to get its pieces in the correct place.

Project Features

- Board State: The electronics will scan the reed switch array to discover if a piece is on any given space on the chess board or not. If an error occurs the UI will notify the user to place all pieces onto the center of their space to fix any errors in piece detection. (Basic)
- Opponent Connection: A board will be able to connect to an opponent through the server in order to start a game of chess. If an error occurs in connecting to the server the UI will be able to display troubleshooting information related to the server connection and options to attempt to reconnect. (Basic)

- Opponent Communication: The boards will be able to send their current state to the other board at the end of each turn to progress the chess game. Upon a faulty state being received from the server a board can request that the transmission be repeated until correct. (Basic)
- Move Recognition: The client software can detect if a legal move was made and only send a request to the server when such a move has been made. If an illegal state happens to be sent by a board the opponents board can also check the legality upon arrival of the state. (Basic)
- State Interaction: The chess boards will react to an opponents game state by relocating pieces to their correct position. If it is detected that the final position is not correct the UI can inform the user to place the pieces back into the correct locations. (Basic)
- Game Completion: The boards will be able to communicate with each other that the current game of chess has been finished. If one board does not acknowledge that the game has ended the opponent may continue to send requests until one is received. (Basic)
- Piece Graveyard: There will be a dedicated place for each piece to go when it is removed from the game. This facilitates game modes in which pieces can be placed back onto the board and events such as pawn promotion. (Basic)
- Error Notification: If an error occurs during any point in the gameplay error codes can be sent to the LCD screen so that the user can troubleshoot the problem or reset the board to continue gameplay. (Basic)
- Virtual Interface: A PC client version will be able to play with the WiFi chess board instead of another WiFi chess board. (Desired)
- Checkmate Detection: Each board can detect if a checkmate has occurred and signal that the current game has ended if it has. If one board fails to detect a checkmate has occurred or signal accordingly the state will be checked for checkmates upon arrival by the opponents board.(Desired)
- Stalemate Detection: The boards can detect if a stalemate has occurred and signal that the current game has ended if it has. If one board fails to detect a stalemate has occurred or signal accordingly the state will be checked for stalemates upon arrival by the opponents board. (Desired)
- Game Resume: The server will be able to start a previously played game from mid way through the game. If the server fails to correctly have both boards set to the correct current state it can check each boards state before gameplay commences to make sure it is correct. (Desired)
- State Recording: All the states of a chess game will be recorded by the server as the game is played for record keeping. (Desired)

- Chess Timers: A move timer will be included with the board, and implementations of how they are used integrated with the chess game. The amount of time in between turns will be transmitted to each board at the end of the turn to make sure that data transmission times do not affect the move timer. (Desired)
- Arbitrary Positioning: The board will be able to move all pieces to any correct board set up from any previous board set up. The boards will communicate their state before a game can commence to verify they are correct, and the server can acknowledge, in case an error occurs. (Dream)
- Chess Variants: Multiple chess variation will be playable by the boards, in addition to tradition chess. (Dream)
- AI: The server will provide a basic AI so that a user can play a game on the board even if no other players are available to play a game with. The users board will be able to verify legal moves have been played upon arrival of a new move, in case a faulty play is made. (Dream)
- Game Playback: The server will be able to make a board play through all moves of a pre-recorded chess game. The users board will be able to verify legal moves have been played upon arrival of a new move, in case a faulty state has been sent. (Dream)
- Standardized Recording: All game moves will be recorded on the server in a standardized format for the game of chess, such as algebraic chess notation. (Dream)

System Interfaces

User Interfaces

This project will have two separate parts to its user interface, the actual chess board itself and a LCD screen and rotary encoder. The chess board and pieces will allow the user to control the state of the chess game, just like in a physical version of the game; moving a piece will move that equivalent piece on the opponents board. The LCD and encoder will allow the user to modify options within the game, such as marking the end of a turn or setting up a new game with another player. To interface the hardware with the user interface some high level C functions will be used. A function to write strings to the 4 by 20 character LCD screen will be present, as well as to clear the screen of previous data. A function will also be implemented and connected to an embedded hardware interrupt such that the function will be called whenever the rotary encoders position is changed or button is pressed.

Hardware Interfaces

The client software is the firmware for the microcontroller that is in the embedded system. This microcontroller will get user input and control the mechanical system. The microcontroller has programmable inputs and outputs which will allow the state of the chess board to be read through the reed switch array, the stepper motors to be

controlled through the Allegro A4988 drivers, the electromagnet to be turned on by an FET switch, the user interface to be displayed, and the WiFi module to be communicated with over the UART bus. The stepper motor drivers have a step/direction interface through which the stepper motor will progress one step either clockwise or counterclockwise (dependent on the direction pin) every time the step pin transitions from low to high. Cypress has a LCD controller embedded in the microcontroller which will control all the timing necessary for the LCD screen as well as present an easy to use API to control the display. Cypress also has digital blocks which can detect if a rotary encoder is being used and what direction it is being spun in, which will provide an easy to use API for that component. A UART hardware block will be used to connect the microcontroller to the ESP8266 module, with some simple circuitry to transition the incompatible voltages of the two devices.

Software Interfaces

This project will have two sets of software that will interface with each other: the client software and the server software. The client software will connect to the server through a TCP connection and use formatted commands which the server will recognize and react accordingly to. This command formatting will be decided on ahead of time and documented to ease the process of changing a command in the future.

Communication Interfaces

Communication in this project is split up between communication in the embedded system and communication that connects the client to the server. To connect the client to the server a TCP connection will be used over a WiFi network. This connection will allow the client to reliably connect to the server, and make sure that no packets are lost during data transfers. The server will run a TCP server using python, and the firmware on the microcontroller will interact to the server through the ESP8266 module. The microcontroller will contain functions such as WiFIStart() which will start the wifi module, connect it to the server and send a handshake packet to make sure communications are working. There will also be a WiFiSendState() and WiFiReadState() function which will send and receive the state of the game at the end of each turn. These will be sent as formatted TCP data transmissions to the server, which will then respond accordingly. The embedded system will be interconnected using UART buses. The UART bus will allow bidirectional communication between the two modules and allow the WiFi module to notify the microcontroller when it has packets which are pending from the server.

Dependencies

An internet server application will need to be running on one of the players computers in order for a game to be played. This server will inform the users of the IP address to connect to and allow a game to begin. Because this server runs locally on a users machine no cloud services will be required. At a future time integration with the free chess servers at FreeChess.org would be a useful side project.