CS 362 – Computer Design

University of Illinois at Chicago

Robots Can Play Checkers Too

- Course Project -

Project Members

Name	NetID	Email
Mark Dabler	mdable2	mdable2@uic.edu
Anthony Slas	aslas2	aslas2@uic.edu

Robots Can Play Checkers Too

- Project Abstract -

This project will allow two robotic arms to go head to head in a game of Checkers. To do so, we will make use a few key project components; a game server to perform the game logic, video processing, and provide the user with a GUI to control the various parts of the game, a game camera to get an image of the game, two Arduino's to control the robotic arms, and two robotic arms to perform physical game actions. The Arduino's will communicate with the server via Wi-Fi.

1. Overall Description/Design of the Project (#'s 1&2 in project writeup)

Our project name, Robots can Play Checkers too, describes a good amount of what this project is about. We will be designing a computer system that will allow two robotic arms the ability to play against each other in a game of checkers. These two robotic will have the ability to reach the entire length and width of the game board to do so they will be driven by two Arduino Uno R3 microprocessors. The two Arduino Uno R3's will communicate with a game server (hosted on a desktop computer) via Wi-Fi. The Arduino's will gather the information needed for their robot's next move from the game server then translate the instructions into physical movements for the robot to perform game actions.

The game server will be running the main software for this project. The game server's purpose is to connect with each Arduino via Wi-Fi, determine the next-best move for the robot who's turn is next, then communicate this move to the Arduino's. The game server will also provide a GUI that will allow the user to control some parts of the game such as stopping, starting, pausing the game or server itself. The game server will determine the next move with the help of a game camera positioned directly above (at 90 degrees) the game board, the software running on the game server will be able to grab this image and use a type of image detection system to visualize the game board layout. The game server will then use this information from the game camera to perform the logic behind the next-best move.

When the next move instruction is received at the Arduino microcontroller it will then power the robotic arm to perform its various physical game actions. After the robotic arm is done making its move the Arduino will send a signal to the server letting it know that it has finished with its move. The game server will then go through the motions on getting the next robotic arm to make its move.

The game "Checkers" is a classic board game that is simple. Pieces can only move diagonally on the dark squares; the light squares of the board are never used. A normal move is moving a piece diagonally forward one square. The initial pieces can only move forward diagonally, not backwards. You cannot move onto a square that is occupied by another piece. However, if an opponent piece is on the square diagonally in front of you and the square behind it is empty then you can (and must) jump over it diagonally, thereby killing it. If you land on a square where you can kill another opponent piece you must jump over that piece as well, immediately. One turn can kill many pieces. It is required to jump over pieces whenever you can.

If a piece reaches the end row of the board, on the opponent's side, it becomes a King. Kings can move diagonally forwards and backwards, making them more powerful in jumping over opponent pieces. However, if you jump over a piece to become a King you cannot jump backwards over another piece in the same move, you must wait until the next turn to start moving backwards. Jumping over opponents is required. However, if you have two possible

moves, where one jumps over one opponent and the other jumps over two or more opponents you are not required to take the jump with the most opponents killed, you are just required to take any jump move.

Winning Checkers

To win you don't lose, and game can be lost in two ways:

1. If a player has lost all his pieces, he loses.
2. If a player can't move at all, he loses.

Upon a victory the game server will send instructions to the winning Arduino.

2. Initial Project Design including Expected Inputs/Outputs

We expect multiple inputs and outputs coming from the; game server, two Arduino's, and the game camera:

Game Server

Inputs	Outputs
• "Finished turn" signal from Arduino • Image produced from the game camera.	• Next movement instruction to the correct Arduino. • Game won, pause, stop, start signal to the Arduino. • Other game options pertaining to the robotic arm.

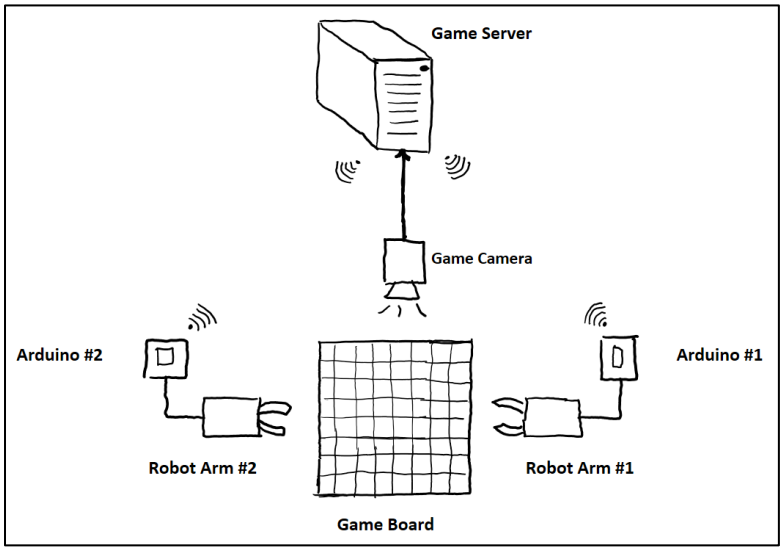
Arduino Uno R3 Microprocessor

Inputs	Outputs
• Current move instruction from the game server. • Game won, pause, stop, start signal from the game server. • Other game options pertaining to the robotic arm	• "Finished turn" signal to the game server.

Game Camera

Inputs	Outputs
• The image of the current game board.	• The image of the current game board.

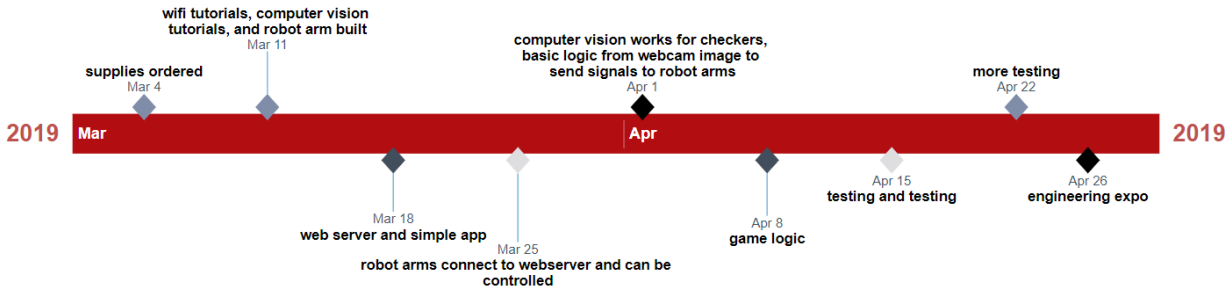
The image below shows a sketch of the project physical layout.



3. Expected Plan for Use and Communication between the multiple Arduinos

We will use Wi-Fi for communication between the game server and the two Arduino microprocessors. The Arduino's will have a special Wi-Fi module wired in to assist with Arduino with this. In a worst-case scenario where Wi-Fi was not available, we will then use serial as communication between the game server and the Arduino's.

4. Project Timeline



5. List of Materials Needed

1. 2 Arduino Uno R3 Microprocessors.
2. 2 Adafruit HUZZAH ESP8266 breakout Wi-Fi modules.
3. 2 ArmUno 2.0 from MicroBotLabs.
4. Checkers 8x8 board & pieces.
5. Desktop/laptop to host the Game Server.
6. Camera and camera stand.
7. Wires to connect everything together.

6. List of References

- Robotic Arms Plays Tic-Tac-Toe
- Robotic Arm Plays Tic-Tac-Toe with controller
- Robotic Arm Plays Checkers
- Robotic Arm Controlled by Android App
- Wi-Fi Introduction

7. Description of Original Work

Most of this project will be done by us. We will be using other references only to help from the code base that will be used in this project. The game server will be written in Python, camera logic will utilize OpenCV, and the Arduino's will be coded in C. We will write our own game logic, make our own webserver, custom computer vision logic, and code to move robotic arms.