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. Project Description

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 an external Wi-Fi chip. 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 board won't be a traditional Checkers board, instead it will have 4 columns and 5 rows, this is because of the size of the robotic arm. Each game piece will be magnetic for the robotic arm to easily manipulate each piece.

## 1.1 Game Server

The game server will be running the main software for this project, its main 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 respective Arduino. To begin, the game server will first accept two clients before starting the game. Once two clients have been connected, the game server will send a random beginning-move to each Arduino, this will officially "start the game". After this initial move, the game server will then begin to calculate the next best move for each Arduino. For the game server to calculate these moves it will always have to know the current state of the game, to do so the game server will utilize a game camera which will be positioned directly above the game board. Once the game server knows the current state of the game it will then use a Checkers algorithm to determine the next move. After the next move has been determined the game server will then communicate this move to the respective Arduino, then wait for the Arduino to finish making the move. The game server will continue this process, for both Arduinos, until a winner has been declared, officially ending the game.

## 1.2 Game Camera

The game camera will be a simple webcam that will feed its raw input into the game server. The game server will in turn process this input using OpenCV, an open source library of functions mainly aimed at real-time computer vision. Once the input from the game camera has been processed with OpenCV, it will when feed the result into a Checkers algorithm. We were able to get the computer vision algorithms to process name, color, and centroid. Unfortunately, we were not able to figure out how to process shapes within shapes (circle pieces within square board positions).

### 1.3 Arduino Microcontrollers

The Arduino microcontrollers will be responsible for receiving its next move from the game server, make the physical game actions provided by the game server, and then communicating back to the game server that it's move has finished. To begin, the Arduino will first power up its respective Wi-Fi card, and then try a connection to a specified game sever. Once connected, the Arduino will move its respective robot arm into a starting position, and then await the first random initial move sent by the game server. The Arduino will then make this initial move once it has received it, and then continue making moves send by the game server until a winner has been declared, officially ending the game.

#### 1.3.1 Robotic Arm

The robotic arm is responsible for making the physical game actions such as; picking up game pieces, moving game pieces, and removing game pieces. Since the game pieces will be magnetic, the robotic arm will be equipped with a electromagnet. This electromagnet can easily pick up and move each piece just by hovering near a selected game piece and then receiving 5 volts to turn on the magnet, which in turn picks up the game piece.

### 1.4 ESP8266 Wi-Fi Card

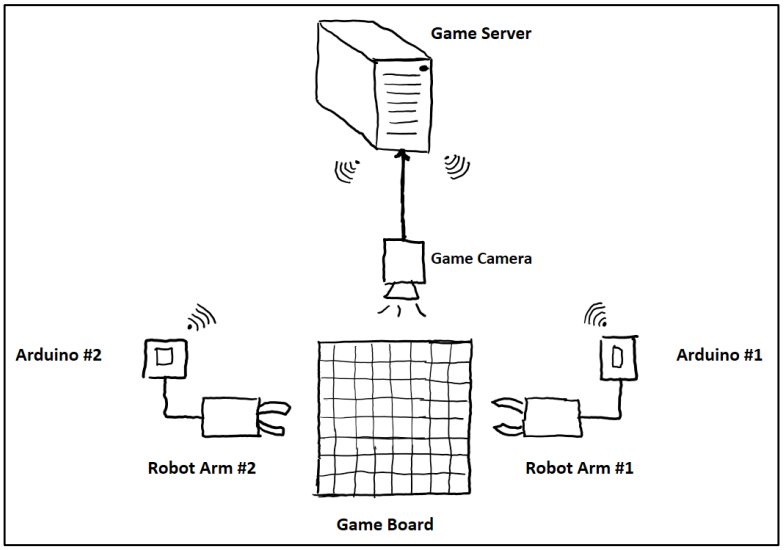
The ESP8266 Wi-Fi card, after being powered on by its respective Arduino microcontroller, will connect to a game server at a port and host specified by its respective Arduino. Upon a successful connection the Wi-Fi card will be responsible for receiving all moves sent to it by the game server and communicating any information back to the game server provided by its respective Arduino. Due to the computer vision aspect not working as well as we had hoped, we wrote a command line interface that allowed us to send commands to the Wi-Fi card which would decode the message and send a move command via serial communication to the respective robotic arm.

## 2. Project Design: Inputs/Outputs

We expect multiple inputs and outputs coming from the; game server, two Arduino's, two ESP8266 Wi-Fi cards, and the game camera:

| Game Server | |
|---|---|
| Inputs | Outputs |
| • "Finished turn" signal from Arduino | • Next movement instruction to the correct Arduino. |

| | |
|---|---|
| • Image produced from the game camera. | • Game won, pause, stop, start signal to the Arduino.<br>• Other game options pertaining to the robotic arm. |




## 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.

| Arduino Uno R3 Microprocessor | |
|---|---|
| Inputs | Outputs |
| • Current move instruction from the game server.<br>• Game won, pause, stop, start signal from the game server.<br>• Other game options pertaining to the robotic arm | • “Finished turn” signal to the game server to the ESP8266 Wi-Fi card. |

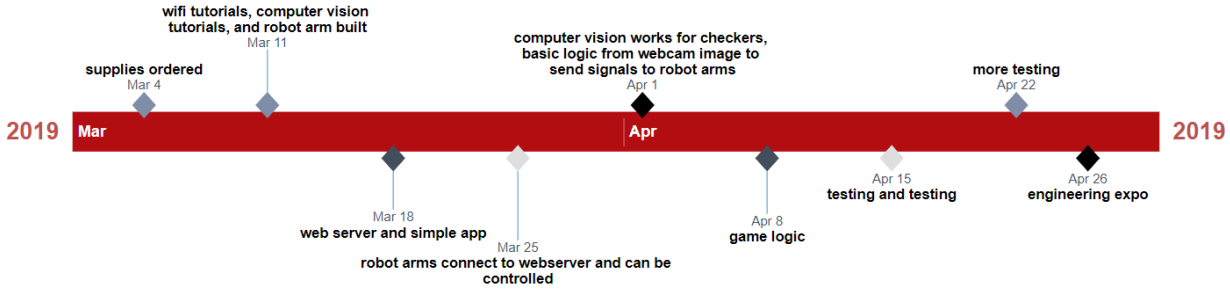
| ESP8266 Wi-Fi Card | |
|---|---|
| Inputs | Outputs |
| • Move instructions from game server.<br>• “Finished turn” signal from the Arduino. | • “Finished turn” signal to the game server. |

## 3. Inner-Arduino Communication

The Arduino and the ESP8266 Wi-Fi card, because the Wi-Fi card is external, will have to communicate to each other using serial (TX/RX). The Arduino will control its respective robot arm and robot arm’s electromagnet via digital pins on the Arduino itself. The ESP8266 Wi-Fi card will communicate with the game server using a TCP socket.

## 4. Project Timeline

Overall:




Development Completed:

- Week 1: Supplies ordered.
- Week 3: Robot Arm Built.
- Week 5: Established communication between Arduino and Wi-Fi card, circuit designed, game server development started both arms can connect to game server and receive a message, electromagnet connected to robot.
- Week 6: Basic move instructions made and configured for robotic arm.
- Week 7: OpenCV and computer vision worked on. Game board made and pieces.
- Week 8: Turn in project and write command line interface to interact with robot arms.

## 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 8x6 board & pieces.
5. Desktop/laptop to host the Game Server.
6. Camera and camera stand.
7. Wires to connect everything together.
8. 2 Electromagnets

## 6. List of References

**Robotics Tutorials**
[Robotic Arm Plays Tic-Tac-Toe]
[Robotic Arm Plays Tic-Tac-Toe with controller]
[Robotic Arm Plays Checkers, repo in video description]
[Robotic Arm Controlled By Android App]

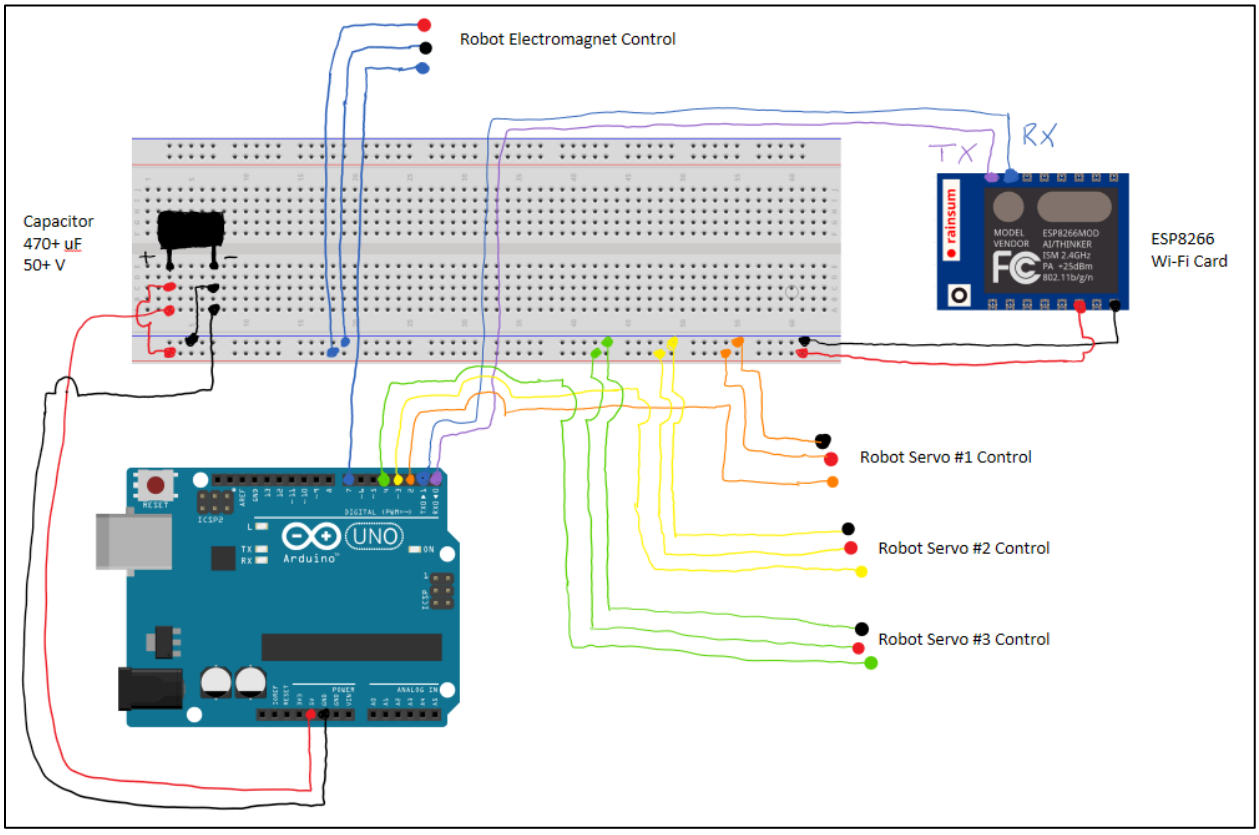
**Wi-Fi Tutorials**
[Wi-Fi Introduction]
[ESP8266 Basic Web Server]
[Python Sockets]

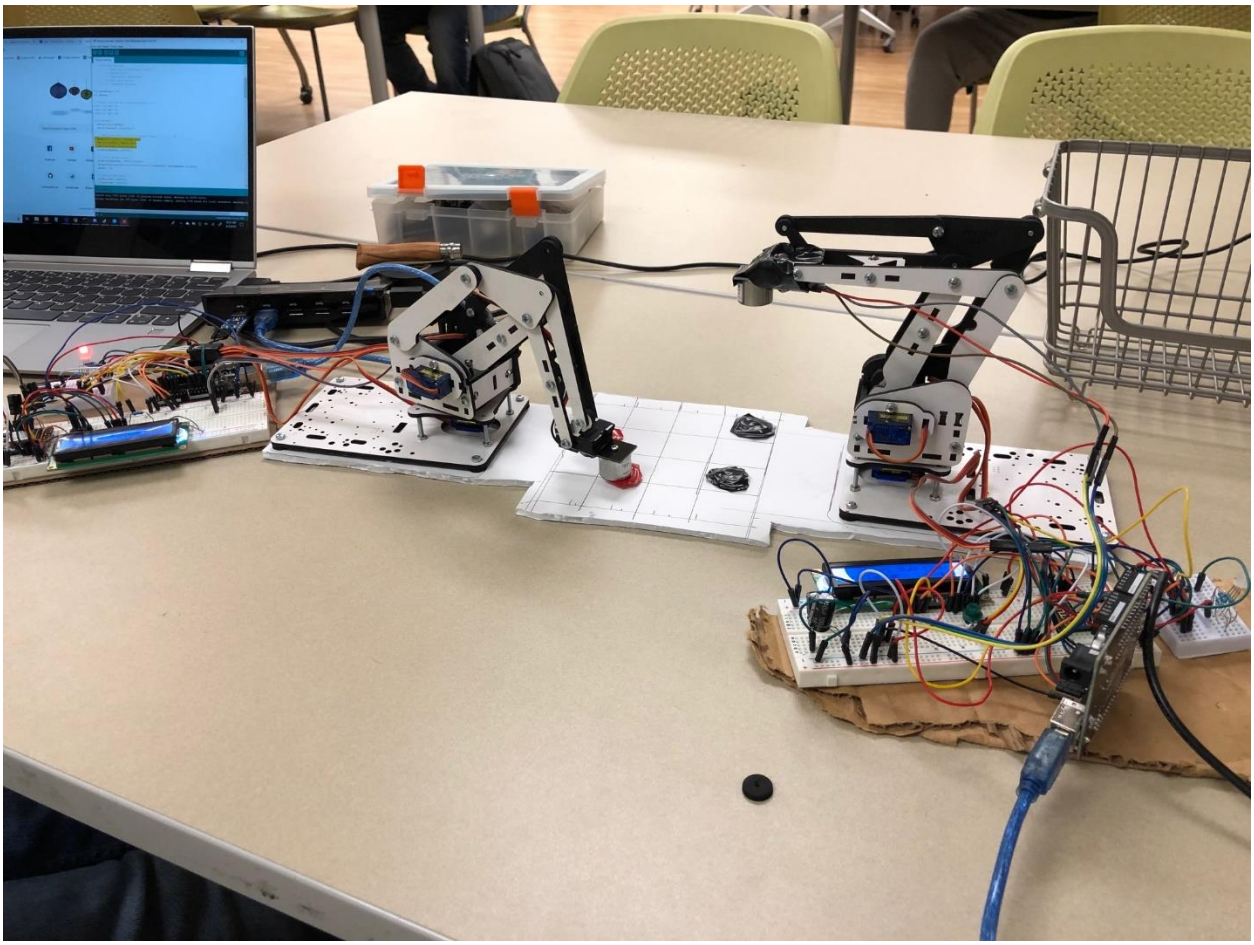
**Robot Arm Kit Tutorials**
[Build Instructions]

**Electromagnet Module**
[Module Wiki]

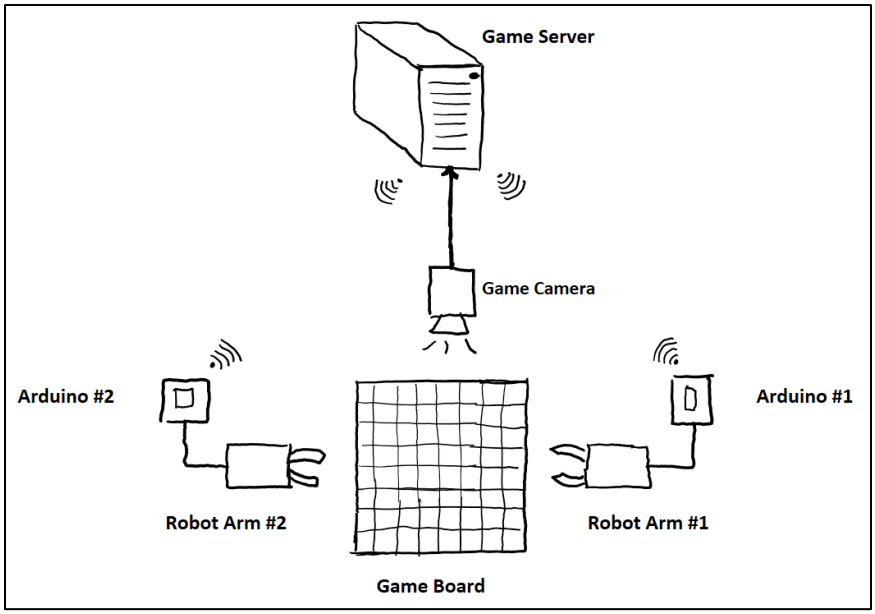
## 7. Design Diagrams

Below is a diagram of the overall low-level circuit for each Arduino, the outputs/inputs are displayed here. The circuit will be the same for both Arduino's.



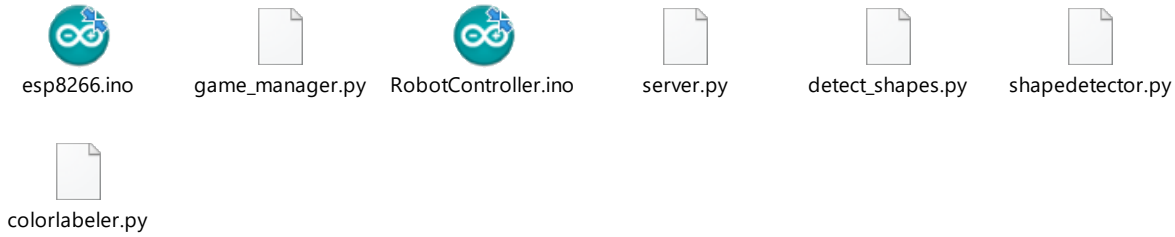


Below is a top-level design of our system.



## 8. Project Code

Our project code, and all of the supporting documents and images can be found at our public GitHub repo: https://github.com/mdable2/robot-arms




## 9. Description of Original Work

Most of this project was done by us. We used other references only to help from the code base that will be used in this project. The game server was written in Python, camera logic utilized OpenCV, and the Arduino's coded in C. We wrote our own game logic, made our own webserver, custom computer vision logic, and code to move robotic arms. At the end, we used a command line program to control the robot arms by sending messages to the Wi-Fi chips.