CSC453/591/791 Final Project Team 10

Combining several IoT techniques to have a robotic manipulator serve as a remote user’s means of interacting with a physical space

### Contributors: [Thomas Batchelder](mailto:tjbatche@ncsu.edu), [Pavel Koprov](mailto:pkoprov@ncsu.edu), [Chaitanya Pawar](mailto:cpawar@ncsu.edu), [Sajal Kaushik](mailto:skaushi2@ncsu.edu), [Rishi Patel](mailto:rapate26@ncsu.edu), [Rachana Kondabala](mailto:rkondab@ncsu.edu)



# Breakdown of Individual Contribution

| Final Project Task | Skills | Expected Amount of Time (for Thomas) **Not correlated to project percentage** | Project Percentage | Assignee |
| --- | --- | --- | --- | --- |
| Setting up Github Repo & file structure | Github | 15 Minutes | 1.31% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Report Introduction | Writing, Google Docs | 45 - 60 minutes | 3% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu) |
| Report Design | Writing, Google Docs | 120 - 150 minutes | 7.59% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Report Implementation | Writing, Google Docs | 120 - 150 minutes | 5% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Report Results & Discussion | Writing, Google Docs | 120 - 150 minutes | 5% | Rachana Kondabala |
| Report Related Work & References | Writing, Google Docs | 45 - 60 minutes | 1.69% | Rachana Kondabala |
| Presentation + Video Demo | Video Recording/Editing, Google Slides | 120 - 180 minutes | 6% | Everyone (1%) |
| Pulling together Final Report | Writing, Google Docs | 45 - 60 minutes | 6% | Everyone (1%) |
| Player\_client: UI/GUI | Python, Console or TKinter | 90 - 120 minutes | 4.76% | Rachana Kondabala |
| Player\_client: Game Logic | Python | 120 - 180 minutes | 8% | Rishi Patel (6.3%), Rachana Kondabala (1.7%) |
| Player\_client: MQTT | Python, MQTT | 120 - 180 minutes | 6.85% | Rishi Patel |
| Camera\_client: OpenCV implementation | Python, OpenCV | 180 - 240 minutes | 13.15% | Sajal Kaushik |
| Camera\_client: MQTT | Python, MQTT | 120 - 180 minutes | 7.65% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu) |
| Robot\_client: Setting up hardware and environment | CAD Modeling | 90 - 120 minutes | 5.85% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Robot\_client: ROS Implementation | Python, ROS | 120 - 180 minutes | 5% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Robot\_client: MQTT | Python, MQTT | 90 - 120 minutes | 8% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Broker Setup | Python | 15 Minutes | 0.15% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Final Demo | Presenting | 90 - 120 minutes | 5% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu)(2.5%), [Thomas Batchelder](mailto:tjbatche@ncsu.edu)(2.5%) |

# Breakdown of each Individual stask

This table is just to help everyone understand what their tasks are without having to dig through the table above.

| Thomas Batchelder | [Pavel Koprov](mailto:pkoprov@ncsu.edu) | [Chaitanya Pawar](mailto:cpawar@ncsu.edu) | Sajal Kaushik | Rachana Kondabala | Rishi Patel |
| --- | --- | --- | --- | --- | --- |
| Presentation + Video Demo | Presentation + Video Demo | Presentation + Video Demo | Presentation + Video Demo | Presentation + Video Demo | Presentation + Video Demo |
| Pulling together Final Report | Pulling together Final Report | Pulling together Final Report | Pulling together Final Report | Pulling together Final Report | Pulling together Final Report |
| Setting up Github Repo & file structure | Report Implementation | Report Introduction | Camera\_client: OpenCV implementation | Report Results & Discussion | Player\_client: MQTT |
| Report Design | Robot\_client: MQTT | Camera\_client: MQTT |  | Report Related Work & References |  |
| Robot\_client: Setting up hardware, camera, and environment | Broker Setup |  |  | Player\_client: UI/GUI |  |
| Robot\_client: ROS Implementation |  |  |  | Player\_client: Game Logic | Player\_client: Game Logic |
| Final Demo |  | Final Demo |  |  |  |

# 

# What’s expected for the demo

Here is a walkthrough on how we are expecting the demo. The design portion of this document contains more detail on how all of the pieces work together.

**Starting Game:** We will start by launching all of the clients and resetting the connect4 board, in order to start a new game. The camera client should detect that a new game has started and should publish a message to reset the player\_client. The robot’s coin container should be loaded with all the red pieces. The robot will always be red and the human player will always be yellow. Additionally, the human player will always go first.

**First Turn:** Someone should be sitting in front of the computer running the player client. The player client should state that it is waiting for the game to start. The human player should place their first coin at this point. The player client will notify the player that it is now their turn and display where the piece was placed. The camera client will determine this. The player client will now specify where it would like to place a piece. The robot client will then place the piece in the corresponding slot. It is now the human players' turn again.

**Continued Play:** The human player and the player client will continue playing one another until one of the following cases occurs: The human player wins, the player client wins, or a tie occurs where the entire board is full of coins, but four in a row never occurs. The robot will respond with an animation when this occurs. Additionally, the camera client should store an image of the final board and the time the game was completed.

**Resetting:** The human player can remove the pieces from the connect4 board, which should trigger a reset by the camera. A new game will start once the human player places the first piece.

# Introduction

### Assumptions for the Game

The human player will always go first. This will reduce some complexity in the long run, as we never have to worry about who will be starting. Once the human player plays the first piece, the player\_client will display where the piece was placed and where they would like to place their first piece.

# Design

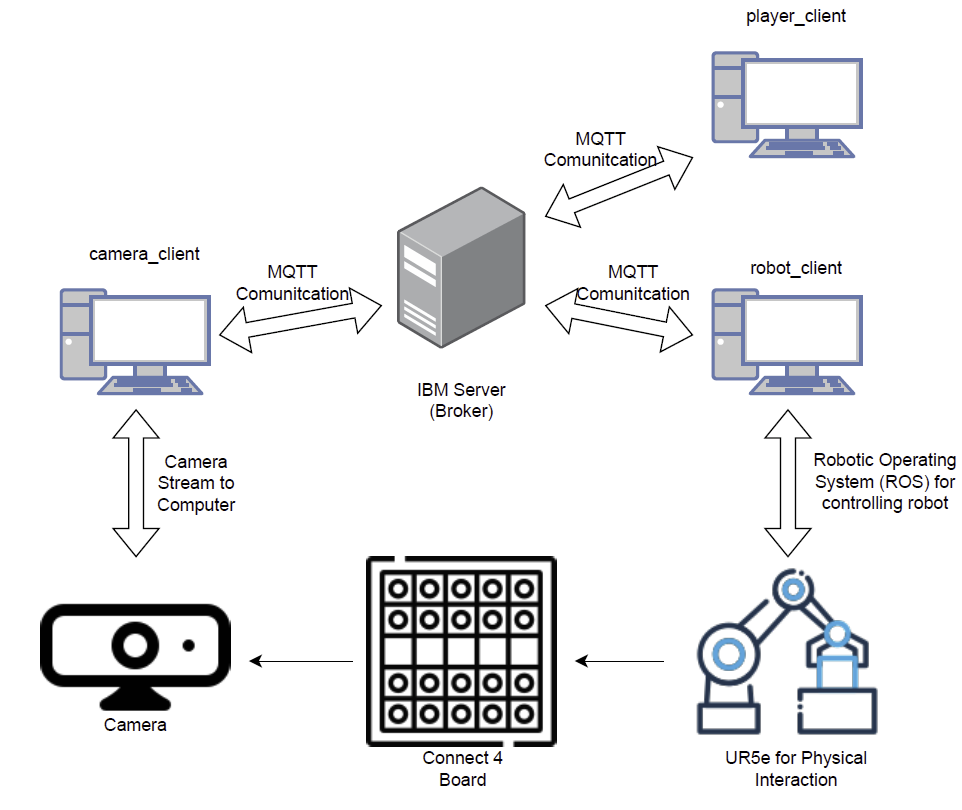


Figure 1 - Overall Architecture of the Project

## 

### Table of all Topics and Messages

| Topic | Messages | Description |
| --- | --- | --- |
| camera\_client/status | “online”, “offline” | Used by the player\_client to determine that the camera is online or offline |
| robot\_client/status | “online”, “offline” | Used by the player\_client to determine that the robot is online or offline |
| camera\_client/board\_data | “#######  #######  #######  #######  #######  #######  ###ox##  #x#xoo#”  “#######  #######  #######  #######  #######  ##xo#o#  ##xoxx#  #xoxoox” | The camera\_client will publish the current state of the board every time an update occurs (A piece being played). The board will be represented as a string of 6x7 ‘#’ symbols. The ‘#’ represents empty space. An ‘o’ will be a yellow piece that has been placed and an ‘x’ will be the red piece. This data will be used by the player\_client to determine the current state of the board. This will be displayed to player\_client, so that can determine where they would place a piece. This should only contain ‘#’ at the start of the game. |
| reset | “true”, “false” | This is published by the camera client and subscribed to by the player\_client. It is published to true whenever the board is reset (all the pieces are removed). This is used by the player to reset the console for a new game. camera\_client/board\_date should contain no pieces at this point and the game\_status should be “incomplete”. This will be set to false once the human player places the first piece. |
| robot\_client/moving | “true”, “false” | Used by the player\_client to determine if the robot is active. If the robot is active, then it cannot receive a new position to place a space. This makes sure that there are not multiple positions published to the robot, which could cause many issues. |
| robot\_client/place\_piece | “0”, “1”, “2”, “3”, “4”, “5”, “6”,”7” | This is used by the player\_client to tell the robot\_client to place a new piece. The number indicated which slot of the piece should be placed in (1 - 7). “0” is used as an acknowledgment by the robot\_client and to keep the robot from placing two pieces in the same column. After the player\_client places tell the robot\_client to place a piece, the robot will publish a “0”. |
| game\_status | “win”, “lose”, “tie”, “incomplete” | This is used by the robot client when the game is complete. The robot will display an animation based on the final state of the game. If neither player has won, “incomplete” will be the payload. If a state other than “incomplete” is published then the camera\_client should save a final image of the game. |

### player\_client Description

The player client the primary system for this project. It will handle all the logic for the game and instruct the other client on what they should do. In order to start the game, the camera “reset” topic must be true and both the camera\_client and the robot\_client must be online. Since the player\_client must always go second when it is started a message will display “Waiting for the game to begin…” Then the player\_client will wait again for its turn. It will get all of the information about the current game from the camera\_client

| Subscribed Topics | Published Topics |
| --- | --- |
| robot\_client/status  robot\_client/moving  robot\_client/place\_piece  camera\_client/status  camera\_client/board\_data  reset  game\_status | robot\_client/place\_piece  game\_status |

### camera\_client Description

| Subscribed Topics | Published Topics |
| --- | --- |
| game\_status | camera\_client/status  camera\_client/board\_data  reset |

### robot\_client Description

| Subscribed Topics | Published Topics |
| --- | --- |
| robot\_client/place\_piece  game\_status | robot\_client/status  robot\_client/moving  robot\_client/place\_piece |

# Implementation

# Results & Discussion

# Related Work & References