CSC 591/791 ECE 592/792: IoT Architecture, Application, and Implementation

Spring 2022

Final Project Report

G10 Members:

Chaitanya Pawar

Thomas Batchelder

[Pavel Koprov](mailto:pkoprov@ncsu.edu)

Rachana Kondabala

Rishi Patel

[Sajal Kaushik](mailto:skaushi2@ncsu.edu)

Breakdown of Individual Contribution

|  |  |  |  |
| --- | --- | --- | --- |
| Final Project Task | Skills | Project Percentage | Assignee |
| Setting up Github Repo & file structure | Github | 1.31% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Report Introduction | Writing, Google Docs | 3% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu) |
| Report Design | Writing, Google Docs | 7.59% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Report Implementation | Writing, Google Docs | 5% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Report Results & Discussion | Writing, Google Docs | 5% | Rachana Kondabala |
| Report Related Work & References | Writing, Google Docs | 1.69% | Rachana Kondabala |
| Presentation + Video Demo | Video Recording/Editing, Google Slides | 6% | Everyone (1%) |
| Pulling together Final Report | Writing, Google Docs | 6% | Everyone (1%) |
| Player\_client: UI/GUI | Python, Console or TKinter | 4.76% | Rachana Kondabala |
| Player\_client: Game Logic | Python | 8% | Rishi Patel (6.3%), Rachana Kondabala (1.7%) |
| Player\_client: MQTT | Python, MQTT | 6.85% | Rishi Patel |
| Camera\_client: OpenCV implementation | Python, OpenCV | 13.15% | Sajal Kaushik |
| Camera\_client: MQTT | Python, MQTT | 7.65% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu) |
| Robot\_client: Setting up hardware and environment | CAD Modeling | 5.85% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Robot\_client: ROS Implementation | Python, ROS | 5% | [Thomas Batchelder](mailto:tjbatche@ncsu.edu) |
| Robot\_client: MQTT | Python, MQTT | 8% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Broker Setup | Python | 0.15% | [Pavel Koprov](mailto:pkoprov@ncsu.edu) |
| Final Demo | Presenting | 5% | [Chaitanya Pawar](mailto:cpawar@ncsu.edu)(2.5%), [Thomas Batchelder](mailto:tjbatche@ncsu.edu)(2.5%) |

Note: The contribution table for our group looks different because we coupled Assignment 4 and final project contribution together. Email was sent on Mar 30, 2022, 3:08 PM under [453/591/751 IoT Project] - Team #1

**1.**   **INTRODUCTION**

The main objective of this project was to implement MQTT, a IoT communication protocol, to allow a remote player to play Connect 4 game with a robotic arm, located in NC State’s Fitts-Woolard Hall. MQTT is a publish/subscribe communication protocol that allows machine to machine communication between multiple devices. In this project, three devices will be communicating with each other on an online broker, to achieve the project’s goal. This project reflects on how factory robots can use IoT technology to communicate and transfer data, how self-driving cars can communicate with each other, how medical robots can be used for surgeries, and many more applications that are transforming the world.

**2.   DESIGN**

**2.1.** **SYSTEM ARCHITECTURE**

**Diagram

Description automatically generated**

**Figure 1: System Diagram**

Figure 1 illustrates all the components in this project and how the information flows from one system to another. The system can be split up into three different subsystems: player client, and robot client. These three subsystems communicate with each other by publishing and subscribing to a topic, on an online MQTT broker. The online broker that was used was hivemq.com The following table explains all the topics that the three clients subscribe and publish to:

**Table 1: Topics and Corresponding Messages**

|  |  |
| --- | --- |
| Topics | Messages |
| camera\_client/status | “Online”/”Offline” |
| robot\_client/status | “Online”/”Offline” |
| camera\_client/board\_data | String of 6x7 “#, o,x” |
| reset | “true”/”false” |
| robot\_client/moving | “true”/”false” |
| robot\_client/place\_piece | “0”, “1”, “2”, “3”, “4”, “5”, “6”,”7” |
| game\_status | “win”, “lose”, “tie”, “incomplete |

The following sections will expand on the topic information from table 1 and describe what each client is subscribing/publishing to and how the information is flowing through the MQTT broker.

**2.2. DESIGN: CAMERA\_CLIENT**

**Table 2: Camera\_Client**

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

From table 2, the camera\_client publishes to camera\_client /status to indicate whether the camera is online or offline. It is also published to camera\_ client/board\_data to indicate the current state of the board every time an update occurs (A piece being played). The board was represented as a string of 6x7 ‘#’ symbols. The ‘#’ represents empty space and ‘o’ represents yellow pieces that were placed, and an ‘x’ were represented by the red pieces. This data was used by the player\_client to determine the current state of the board. This was displayed to player\_client, so that they could determine where they would place a piece.The camera client was publishing the reset topic to “true”  whenever the board was reset (all the pieces were 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 was set to false once the human player placed the first piece. Camera\_Client was subscribing to game\_status which indicated who won the game. If the game\_status published “incomplete”, then the camera\_client saved a final image of the game.

**2.3. DESIGN: PLAYER\_CLIENT**

**Table 3: Player\_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 |

Based on table 3, the player client was the primary system for this project because it handled all the logic for the game and instructed the other client on what they should do. The first step in designing the player client was to develop the connect four game logic. The Connect four game has 4 main functions - initialize board, print board, insert piece, and check for victory. The board was a two-dimensional matrix which holds -1,0, and 1 to identify player 1 piece, empty spot, and player 2 piece. 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 was displayed “Waiting for the game to begin…” Then the player\_client waited again for its turn. It got all the information about the current game from the camera\_client.

**2.4. DESIGN: ROBOT\_CLIENT**

**Table 4: Robot\_Client**

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

Table 4 illustrates that the robot\_client published to topic robot\_client/ status to notify the player\_client if the robot is online or offline. The client then publishes to robot\_client/ moving to notify the player client that the robot is active and that it cannot receive a new position to place a space. This ensured that the robot didn’t get confused with multiple positions published to the robot.The third topic that the robot client published to was robot-client/ place\_piece which lets the player\_client know that a new connect 4 disc has been placed by the robot. The number indicated which slot of the piece would be placed in (1 - 6). “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 tells the robot\_client to place a piece, the robot publishes a “0”. The robot\_client is subscribed to game\_status, to monitor the results of the game. The robot displayed an animation based on the final state of the game.

**2.4. DESIGN: PHYSICAL HARDWARE**

In this project, the following physical components were used to create demo apparatus:

* UR5e Robotic Arm
* 2F-85 Robotic Gripper
* Connect 4 Game
* 3D printed fixtures
* Logitech Webcam
* Computers

The following screenshots illustrate all the major hardware we used for this project.

Close-up of a faucet

Description automatically generated with medium confidence

**Figure 2: UR5E Robotic Arm**

UR5e is a flexible, medium-duty robotic arm used for various applications throughout multiple industries. For this project, this equipment acted as the focal point of playing connect 4 against a robot. Some general specifications of the robot are as follows:

* Max Payload: 5 kg/ 11 lbs.
* Reach: 850 mm/ 33.5 in
* Footprint: Dia 149 mm
* Weight: 20.6 kg/45.4 lbs.

A picture containing icon

Description automatically generated

**Figure 3: Robotiq 2F-85**

2F-85 is a robotic gripper that is integrated with the UR5e robotic arm. The gripper allows the robotic arm to pick the connect 4 chips and place it in the corresponding slot.

**A picture containing indoor

Description automatically generated**

**Figure 4: 3D Printed Fixture**

Figure 4 illustrates the 3D printed fixture that was used to contain the connect 4 chips for the robot. It allowed the robot to sequentially grab a chip from the bottom and place it in the Connect 4 slot.

**3.   IMPLEMENTATION**

The following sections will describe how all the clients were implemented in this project.

**3.1. CAMERA\_CLIENT**

Camera\_client was implemented by taking a live stream which captured 30 frames per second(fps). At every frame, first bilateral filtering was applied to remove the background noise, smoothing the image which in return helps in identifying the edges. After that Canny Edge detection algorithm was applied opencv library was used to find contours function. These functions get the contours and then find the shapes that have a high number of vertices. Then contours that represent circular slots were selected by using conditions on the area, vertices, and polygon’s dimensions.

Then color filtering was applied for identifying red and yellow connect 4 chips. In order to do this, the images were captured to HSV space, which was used to construct a mask from a certain range of hue values. The last step of implementation was iterating over the grid to locate the position of all the red and yellow contours to create a grid. The grid was mapped using a numpy 2d array which indicates “0” for no coin, “1” for red, and “-1” for yellow. Then the MQTT protocol was implemented to the camera\_client by using paho.mqtt.client, cv2, and camera\_processing. Broker.hivemq.com was used as the online broker.

**3.2. PLAYER\_CLIENT**

**3.2.1. CONNECT4 GAME LOGIC**

The first step in designing the player client was to develop the connect four game logic. The Connect four game has 4 main functions - initialize board, print board, insert piece, and check for victory. The board was a two-dimensional matrix which holds -1,0, and 1 to identify player 1 piece, empty spot, and player 2 piece.

**3.2.2 PLAYER\_CLIENT LOGIC**

The second part is the actual player client which utilized mqtt protocol using Paho library to communicate with all the other clients on what steps to take. Player client is responsible for starting the game once all the necessary conditions are met. Player client then takes input from the user of where to place the piece and communicates with the robot client to physically place the piece. Player client then checks the board to see if the robot or the player won the game. The player client only starts the game once the robot and camera client are online, and the reset game is true. Player client also ensures that multiple requests are not sent to the robot while the robot is moving to place a piece.

**3.2.3 USER INTERFACE**

Connect4 GUI is developed using PyGame.

It has the following three functions:

* Draw\_board:
  + uses pygame module to display the game on UI.
* Update\_board :
  + Gets the board data from camera\_client and displays whenever a human player(player1) inserts a piece and shows the updated board.
  + Takes the input from player2 and displays it in the UI. A check is added to prevent the player from placing the coin in a completely filled column
* Display\_message:
  + Let the player know when it’s their turn. After every move, it displays the column number where the piece is placed.
  + Once the game finishes, it displays the result.

**3.3.** **ROBOT\_CLIENT**

**4.   RESULTS & DISCUSSION**

**A picture containing text, indoor, cluttered

Description automatically generated**

**Figure 5: Final Demo**

Based on the in-person demonstration, the project was successful in launching all of the clients and resetting the connect 4 board, in order to start a new game. The camera client successfully detected that a new game had started and published a message to the “reset” topic, letting the player client know that a new game could be started. The robot’s Connect 4 chip container should be loaded with all the red chips, while the human player’s chips were yellow.

The game started when the human player made their first move by indicating where they wanted to place their first chip, through the game's user interface. The player client was notified that it is now their turn and display where the piece was placed. The camera client will determine this. The player client specified where it would like to place a piece. The robot client then placed the piece in the corresponding slot.

The game continued, until the robot won and then the camera\_client took a picture of the result of the game and the time the game was completed. The human player can remove the pieces from the connect4 board, which should trigger a reset by the camera. A new game was started once the human player placed the first piece.

**5.   REFERENCES**

1. Matt Jennings. (2020, June 2). *Connect-Four Computer Vision A.I*. Matt Jennings. Retrieved April 20, 2022, from https://mattjennings.co.uk/portfolio/Connect-Four%20Computer%20Vision%20A.I/
2. Sawant, S. (2021, April 12). *Connect four game in Python*. AskPython. Retrieved April 20, 2022, from https://www.askpython.com/python/examples/connect-four-game
3. Nieves, O. (2021, August 10). *Programming a connect-4 game on Python*. Medium. Retrieved April 20, 2022, from https://oscarnieves100.medium.com/programming-a-connect-4-game-on-python-f0e787a3a0cf
4. *UR5E*. Universal Robots - UR5e. (n.d.). Retrieved April 20, 2022, from https://shop.universal-robots.com/ur5e/
5. *Ur+: 2F-85 adaptive gripper for Universal Robots*. UR+ | 2F-85 Adaptive Gripper for Universal Robots. (n.d.). Retrieved April 20, 2022, from https://www.universal-robots.com/plus/products/robotiq/robotiq-2f-85/
6. *Mqtt Client Library encyclopedia*. HiveMQ Library - Tutorials on how to use the MQTT client. (n.d.). Retrieved April 20, 2022, from https://www.hivemq.com/mqtt-client-library-encyclopedia/