Summary Write-Up

Duke University Pratt School of Engineering

ME 555: Intro to Robotics and Automation
Write-Up Final Project
Homage to the Mechanical Turk

Lilly Chiavetta (lhc24)
Austin Camacho (atc52)
Kaelyn Pieter (kmp101)
Jared Bailey (jlb271)

Professors:
Dr. Siobhan Oca

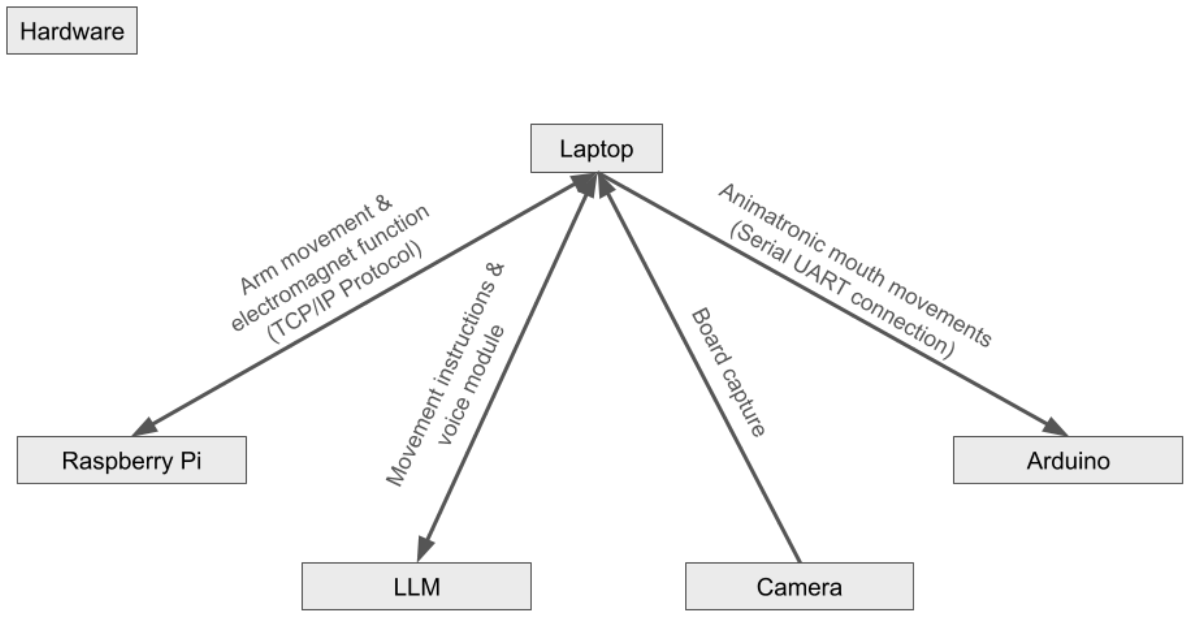
TA:
Vincent Wang

Due Date:
December 10, 2024

Documentation of Robot Actions

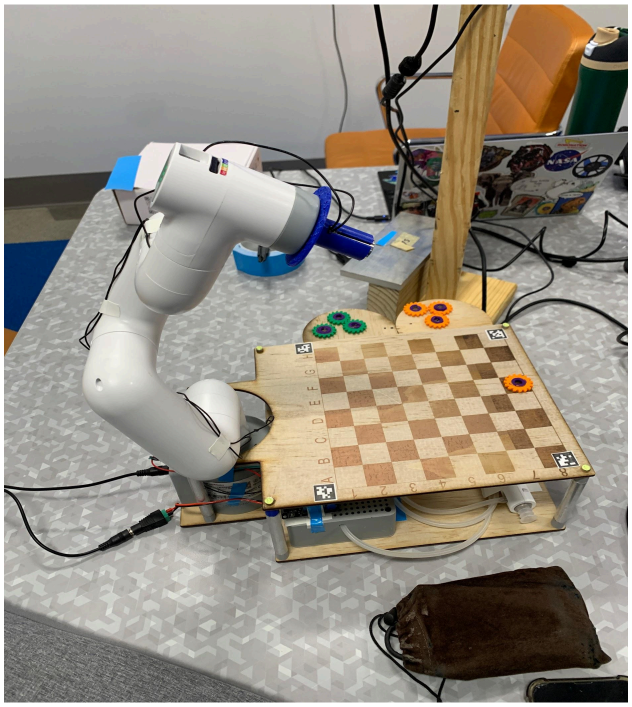
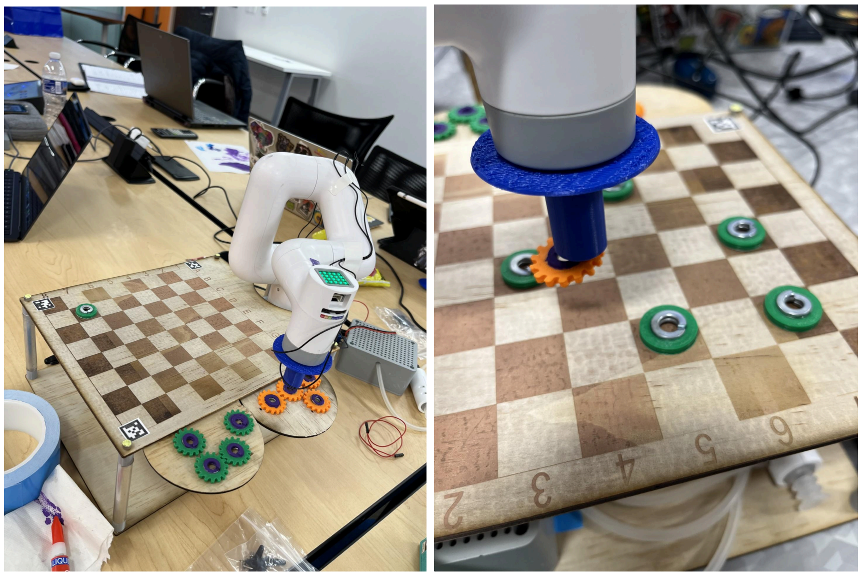
High Level Description
An autonomous robot capable of playing checkers against itself, or a human opponent. Utilizing CV to log and update board state, the robot will move checkers across the board to jump the other player's pieces and crown kings. In the spirit of the Mechanical Turk of old, the robot is able to appear lifelike using animatronics and a voice clone of our class TA Kent Yamamoto.

Systems Diagram


Hardware
Laptop
Arm movement & electromagnet function (TCP/IP Protocol)
Movement instructions & voice module
Board capture
Animatronic mouth movements (Serial UART connection)
Raspberry Pi
LLM
Camera
Arduino

Project Breakdown
We broke this project down into a few sections: Animatronics, Program Robot, Computer Vision, Mechanical Robot, and Simulation Robot. By assigning a few people to each section, we ensured all parts of our project moved forward at a consistent pace. Lilly focused on system integration and reception of human movements, programming the robot movements, the checkers game, and completing the electrical portion of the end effector. Jared focused on Animatronics, CV, robot and electromagnet movement and operation, the checkers game, and robot voice control. Austin focused on robot simulation in RViz, board construction, animatronics, and ensuring the mechanical parts were completed. Kaelyn focused on designing the end effectors, board and piece construction, animatronics, and simulation work in RViz.

Robot Image
Demonstration Videos
https://drive.google.com/drive/folders/1jkflwo8Y7DBBj6jdzRHv-Q7fGoU75lg5?usp=drive_link

Code Explanation

- There are more functions and methods (100+) than can be reasonably posted and described here. As such, we provide high level overviews of code sections. The code is well commented and organized for ease in understanding and use.
    - Connections - Python
        - We integrated the various hardware (laptop, camera, arduino, raspberry pi) and software components using serial communication and TCP/IP protocol, with the primary code running on a laptop.
        - The code here can set up the desired game configuration (whether the robot is on and can be connected, whether smack talk is desired, whether the camera is

connected, if we are using voice commands or manual movements, and if we are playing or if the robot needs to play itself).

        - The code here also allows for custom setups that can allow the user to put the pieces in any configuration on the board and the robot will play with that configuration.
    - Robot movement - Python
        - Operational space - We converted board squares (a2, b3, etc.) into operational space for robot movement. This was done through addition and multiplication of known square spacing. We programmed movements directly above each board space, allowing for controlled movement down to each square. This method proved useful, as we were able to extend this code to operate outside the board to collect kings and remove pieces from the board using off board square (ex. I5, J6)
        - Joint space - Far reaching spaces stretched the robot too far for functional use of operational space. In these cases, we made use of joint space for row 8 on the board.
        - Electromagnet operation - The electromagnet is operated using a power relay of 5V, which provides 8V to the end effector. This code is combined with the robot movement code to allow the robot to lift and place the metal washers within the board squares at will.
    - Computer vision - Python
        - Streaming video - A live video feed is used to capture an image of the board.
        - April Tags - 4 tags were used to locate and orient the board. Since the camera and stand is not affixed to the robot arm, we had to account for varying distances and rotations of the camera in relation to the board. For redundancy, only 3 of the 4 April Tags are needed for locating each board square and providing board orientation.
        - HSV filter - A color filter and contour detection were combined to identify orange, green, and purple objects within the board. Purple washers were painted to signify king status, so that a checker piece could be both a player's color (orange or green) and a king (purple).
        - Integration - Using the combination of the April Tags and HSV filters, we were able to determine each piece's location, closest board square (using manhattan distance), color, and king status. This information was combined with the checker game code in order to allow the human to move pieces, and the robot would update the board state layout in its memory.
    - Animatronics - C++ style arduino code
        - Eyeball operation - The eyeballs are set to move vertically and horizontally at random intervals, to random locations. This is done by operating 2 180 degree servos.
        - Eyelid operation - All four eyelids operate in tandem. They are set to operate at random intervals, to specified locations. Before eyelid operation, the eyeballs first move into a set position in order to lessen the chance of undesired object friction. The eyelid code operates 4 180 degree servos.

        - Mouth operation - The lower jaw is set to move vertically for the time specified by the incoming audio file. The audio file length in seconds (integer) is transferred serially from the laptop to the arduino. The arduino code operates the jaw using 2, 360 degree servos.
    - Voice cloning - Python
        - We performed voice cloning by asking our TA Kent Yamamoto for 30 voice samples. These samples contained a large variety of the phoneme sounds in a variety of word and sentence locations available in the English language. Since these voice samples were unintuitive and difficult to say, we also asked Kent to record natural sounding language. Both sets of recordings were used to create a voice clone through ElevenLabs. We saved over 50 pre-recorded messages for robot use, as well as built an API call to ChatGPT and ElevenLabs in order to provide real-time speech. The code also allows the developer to call a local LLM (Llamafile) to save money. This code was integrated to operate between the piece movements made by the robot arm, and with the mouth movements of the animatronics.
    - Checkers game - Python
        - Overall - The game was built using OOP, containing classes for piece, board, and game. The pieces contained information such as color, location, and king status. The board contained information such as possible moves and all piece locations. The game allowed both human and robot movement of the pieces on the board.
        - Game modes - 5 game modes are functional in the robot: human, random, prioritize jumps, LLM, and minimax. The LLM mode is able to call a locally running LLM on a CPU (Llamafile), or the ChatGPT service. The minimax is able to look ahead a number of moves preselected by the user in order to select the best available next move. Game modes include failovers to simpler methods in the event of errors.
        - Voice control - Allows a user to speak moves aloud, and the robot to respond with the desired movements.

Human Integration

We designed the robotic system with human interaction in mind. The human can either move pieces themselves, or vocally instruct the robot to move to pieces during the human's turn. Since the robot arm is on the smaller side, of low weight, and contains low torque motors, there is low risk that the robot can harm the human. That being said, the robot arm does contain an emergency stop switch. Out of an abundance of caution, we also advise the human paying against the robot to stay aware of the workspace of the robot, the robot's current position, and the timing between human and robot turns.

Challenges

Programming robot behaviors has been a series of challenges throughout the last several weeks. Getting the robot to work was a challenge all on its own because the robot's instruction manual and user guide were both in Chinese. With the help of an out of date user manual, we were able to get the robot up and running.

The next challenge with programming was figuring out and using the unique coordinate system of the robot, being careful not to collide with our custom board. Due to limitations of the coordinate system for the robot, we use joint angles to reach the 8th row of the checkers board.

Writing a minimax function, and integrating an LLM into this project were both challenges for programming. Once those functions were up and running, there was then the challenge of integrating the checkers game with the robot, computer vision code, voice clone, and animatronics. While moving from location to location is simple, interacting with all the various components in this project at the required times and in the desired ways was quite a challenge.

Future Work

This project could be expanded in many ways in the future. Foremost, we could expand to other board games such as chess or connect 4. While the robot is currently mounted to our checkers board, by connecting it to other boards and creating a set of metal pieces, we could utilize this same magnetic end effector and robotic arm to easily play another game. Changing the game would require immense programming in addition to making a new set of pieces or a new board, but many games would be feasible with the current end effector and robot arm.

Additionally, we would move from simulation in RViz, to physics simulation in Gazebo. Due to the nature of this project, we decided physics simulation wasn't necessary in the short term. However, as we expand this project further, it would become helpful.

Another option would be to fully integrate voice control of the robot. We accomplished consistent voice control in quiet environments, but decided to discontinue development for noisy environments due to lack of available time.

Finally, including additional safety systems to stop the robot arm movement if the human was to move within its workspace as detected by computer vision would provide a welcome reduction in risk.

Takeaways

- ROS 2 is hard to learn, and working in various versions is very complex.
    - Utilizing ROS 2 has a variety of challenges, especially due to our limited experience with the operating system. For this project, we were working with RViz in Humble, however we learned how to use RViz in Iron during this semester's coursework. We had trouble

    - transferring our Iron knowledge to Humble. Learning to customize code for the specific version of ROS 2 we are working on is a key takeaway for future robotics projects.
- Individual code pieces on separate systems work well, but are very difficult to properly integrate.
    - Integrating individual code pieces is very difficult. Various parts of code are running on different devices, so inter-device communication is key. In the future, it may be ideal to limit the number of individual devices to avoid potential unnecessary complications.
- Integration between two separate electrical systems is complex.
    - Due to the fact that we chose to build a custom end effector for this project, we needed to integrate our electromagnet into the robot arm. All the wiring and electronics for this robot are neatly integrated inside the arm's plastic shell, leaving our end effector wiring outside. Since a majority of the documentation for this project is in Chinese (and unfinished), it was very difficult to understand which external ports we should be using. After much trial and error, we found ports on the robot's base to utilize, rather than the ports near the end effector of the robot arm.
    - Initially, we built a custom electromagnet. However, this electromagnet proved to have a dangerous amount of amps and was scraped. We then purchased a safer electromagnet. It was difficult to know what type of power draw our custom electromagnet would perform without knowing specifics about the robot's electrical ports, which were listed in Chinese.

Code

See GitHub for all Code
https://github.com/JaredBaileyDuke/checkers-bot

Sources

Robot Arm
https://shop.elephantrobotics.com/collections/mycobot-280

Animatronic Eyes
https://willcogley.notion.site/Will-Cogley-Project-Archive-75a4864d73ab4361ab26cabaadaec33a

3D Teeth Mold
https://makerworld.com/en/models/657234#profileId-584358