MTRN4231 Team Portfolio

David Ceballos Munoz	z5267332
Kerim Tapan	z5627537
Finn Crawford	z5309129
Tony Mei	z5507288

Table of contents

1. ROS2 Architecture

1.1 Node Chart

The system's architecture is built around a ROS2 node-based framework, which enables efficient communication and operation of the robotic system. The node interaction diagram shows how the individual nodes work together to accomplish the goal of playing a xylophone. Each node performs a distinct function, while their integration ensures the overall functionality of the robot.

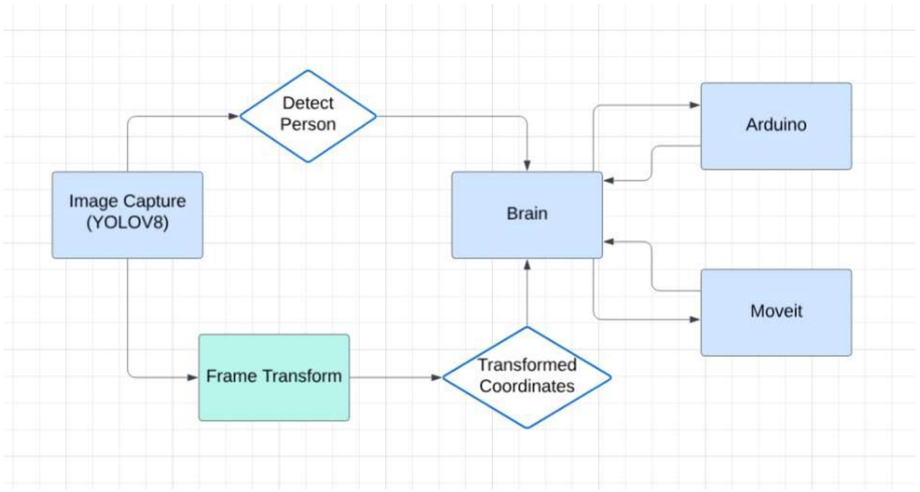


Figure 1 - Node chart

The Image Capture Node (Colour masking/ YOLOv8) is the starting point of the system. It captures and processes visual data from the environment to detect objects (xylophone keys) and people. The image node publishes key coordinates, a **dots_detected** status and a **people_detected** status via the **key_positions** topic. The custom message type **KeyPositionArray** is utilised here, containing an array of **KeyPosition** messages, each with fields such as letter, x, y, and z for key identification and spatial data.

The Brain Node acts as the central processing unit of the system, coordinating the operation of all other nodes. It subscribes to two main topics: **key_positions**, which provides raw data from YOLO, and **transformed_key_positions**, which contains frame-transformed coordinates from the Frame Transform node. Using this data, the Brain node makes decisions and publishes **MoveItCoords** messages to the **moveit_target** topic. This custom message includes fields for target coordinates (x, y, z) and orientation (qx, qy, qz, qw). The Brain node also communicates with the Arduino node via the **trigger_operation** service to initiate the physical striking of keys.

The Frame Transform Node converts the pixel coordinates of detected keys into real-world coordinates in the robot's base frame. It subscribes to the **key_positions** topic and publishes the transformed data to the **transformed_key_positions** topic. This transformation ensures that the robot arm can accurately move to the desired locations.

The **MoveIt** Node is responsible for executing the motion planning commands issued by the Brain node. It subscribes to the **moveit_target** topic to receive precise coordinates and orientation for the robot arm. This integration ensures smooth movement of the arm toward the keys.

The Arduino Node controls the end effector, specifically the mallet for striking the xylophone keys. Upon receiving a service call from the Brain node via **trigger_operation**, it communicates with the Arduino through PySerial and transmits feedback to Brain node which is received from the Arduino, after the successful actuation.

1.2 Brain Close-Loop System

The Brain node operates in a closed-loop manner, ensuring that each operation is synchronised and robust to dynamic changes in the environment. A control loop diagram can be found in appendix 8.3. The close-loop system is depicted in the second diagram and functions as follows:

1. **Set Key Order:** The Brain node iterates through the predefined array of keys to play, ensuring that each key is processed sequentially. After striking a key, it moves on to the next key unless interrupted by a person detection event or an invalid coordinate frame.
2. **Move to Home Pose:** When the system detects people in the environment, the robot arm moves to a predefined safe home position. The Brain node publishes a MoveItCoords message containing the home position coordinates to the MoveIt node.
3. **Detect Person:** The Brain node continuously monitors the people detected flag from YOLOv8. If people are detected, the system halts its operation, redirects to the home pose, and waits for the environment to clear.
4. **Coordinate Frame Validation:** Before initiating any movement, the Brain node validates the coordinates received from the Frame Transform node. If the transformation is invalid, the system waits until valid data is available.
5. **Move to Key:** Once the coordinates are validated, the Brain node sends a MoveItCoords message with the target key's coordinates to the MoveIt node, instructing the arm to move to the desired location.
6. **Wait for MoveIt Completion:** The Brain node waits for confirmation that the robot arm has completed its movement before proceeding to the next step.
7. **Initiate Arduino Function:** Upon successful movement, the Brain node sends a service call to the Arduino node, triggering the physical strike of the key.

1.3 Topics, Services, and Custom Messages

The system effectively utilises ROS2 topics, services, and custom message types to ensure smooth communication between nodes:

Topics:

- **key_positions:** Publishes key detection data including key positions, dot detection and human detection from colour masking system and YOLO to the Brain and Frame Transform nodes.
- **transformed_key_positions:** Publishes transformed key coordinates from the Frame Transform node to the Brain node.
- **moveit_target:** Sends target coordinates and orientation from the Brain node to the **MoveIt** node.

Services:

- **trigger_operation:** Allows the Brain node to communicate with the Arduino node for end-effector control.

Custom Messages:

- **KeyPositionArray:** Contains an array of KeyPosition messages for detected keys, a Boolean for if blue dots are detected, and a Boolean for if humans are detected.
- **KeyPosition:** Includes fields letter, x, y, and z for key identification and position.
- **MoveItCoords:** Contains fields x, y, z, qx, qy, qz, and qw for target position and orientation.

2. Code Progress Overview

2.1 Contribution Summary

Each team member focused on one subsystem throughout the project with clearly defined system boundaries for smooth integration. David was responsible for ROS architecture and brain node implementation with closed-loop operation, Finn has worked on the computer vision for detection of keys positions and human detection, Kerim designed, manufactured and integrated the end-effector to the rest of the system and Tony attempted to implement Moveit and system visualisation to control the robot. The provided table below shows the significant contributions from each member during the last 5 weeks of the project:

Table 1 - Member contribution summary from week 6

Member	Week 6	Week 7	Week 8	Week 9	Week 10
David	Brain Node	Communication method for nodes	Key frame transforms | Closed-loop procedure	Attempted to implement MoveIt	Close loop Finalisation
Finn	Key detection software + custom ROS message type	Localisation method with blue dots + full Computer vision prototype	CV prototype refining to lab conditions + human detection	GUI for colour masking	Presentation slides + help in attempt to implement in MoveIt

Kerim	End Effector assembled	End Effector tested in operation	ROS-Arduino Serial communication implemented	End Effector URDF for system visualisation	Offset calibration
Tony	Background knowledge for robot kinematics	N/A	Trying to test basic robot kinematics using Moveit2	N/A	Preparing demonstration for example Moveit2 simulation

During week 6 & week 7 everyone worked on their own to implement their assigned part of the project. At week 8 it has been noticed by the other members that the Moveit & Visualisation part hadn't had any implemented progress yet. Due to this reason, the rest of the team attempted to implement the missing Moveit & Visualisation nodes needed to complete the whole system and to be able to demonstrate the full functionality of the system on the real robot.

Key Milestones Timeline:

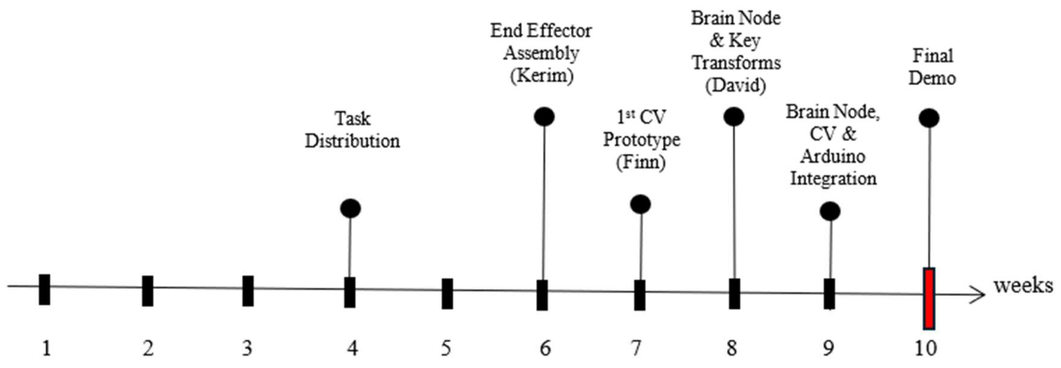


Figure 2 - Key Milestone Timeline

The key milestones can be seen in the above timeline, with each members' key subsystem achievements and the collaborative milestone of integration. Until the end of week 5 the team has focused on the labs and the individual assignment, which was followed by the flexibility week, when everyone started implementing their subsystem.

2.2 Timeline of Contributions

The code contribution and sharing has been done by sharing compressed source files through Microsoft Teams. In the Appendix 8.2 "Team Code Contribution Details" the .zip files sent by each member to contribute to the project can be seen.

To summarise the contributions by each member, Finn implemented the first iteration of the key detection in week 6, while Kerim has designed and manufactured the end-effector. In week 7 the end effector's functionality was tested and the computer vision prototype was built. This was followed by the first draft of the brain node and key transformation nodes by David on week 7 and the closed loop procedure by the beginning of week 8. Kerim implemented the Arduino-ROS communication in week 8, as well as Finn and David refining their parts. In week 9, the Arduino end effector actuation has been integrated into the brain node by David, while Kerim built a system visualisation with a simplified URDF of the end-effector, and Finn built a graphical user interface for the CV part to calibrate the colour masking with ease and matched the pixel size of the image with the real-world distance.

In the last week, David and Kerim calculated the offset of the blue dots to the base of the robot and implemented the coordinate transformation from the pixel coordinates to the coordinates from the robot origin, namely base link. With this progression, the closed-loop procedure was finalised by David. While this happened Finn made the presentation slides. At this point, the team was able to demonstrate detection of precise key positions and accurate transformation of these to robot frame, during the runtime of the brain main loop, as well as the end effector actuation through the brain node.

3. Evidence of Collaboration

3.1 Collaboration Methods

The team relied solely on Microsoft Teams as the primary communication platform throughout the project. Within Teams, a dedicated group chat was established, which served as the central hub for team discussions. This chat was vital to facilitate regular updates on individual components, discussing any challenges or changes to the system, and sharing files necessary for integration. The consistent communication through Teams ensured that all members were informed of the project's progress and could align their tasks accordingly.

In-person collaboration sessions were also integral to the project. These sessions, held during designated lab times, provided an opportunity for team members to work together on integrating and testing the various components of the system. The attending members utilised this time to troubleshoot issues with dependent nodes and validate the brain node's functionality within the overall system. These hands-on sessions were especially valuable for identifying and addressing integration challenges that were otherwise difficult to fix individually.

3.2 Task Division and Tracking

The project tasks were divided among the team members based on their areas of expertise and interest. The Brain node and overall closed-loop functionality of the system were assigned to one member, who was responsible for coordinating the robot's movements and ensuring communication between nodes. Another member focused on frame transformations and MoveIt node integration, handling the alignment of coordinates and robot motion planning. The YOLO-based visualisation and object detection system were managed by a third member, while the Arduino integration and end effector, responsible for striking the keys, was assigned to the fourth member.

Task tracking was conducted primarily through bi-weekly meetings on Microsoft Teams, where each team member provided updates on their progress. These meetings served as checkpoints to ensure alignment and identify potential issues. Additionally, the Teams chat was used for real-time updates, file sharing, and discussions. While this approach worked well for most of the team, the lack of consistent and accurate updates from the member responsible for the MoveIt node and frame transformations created challenges, requiring the team to adjust responsibilities and proceed with assumptions for certain components.

To ensure accountability, progress was monitored through informal check-ins between team members and during integration phases. However, the absence of stricter deadlines for individual contributions made it difficult to identify delays until the later stages of the project, impacting the overall timeline.

3.3 Supporting Evidence

Refer to appendix section for chat logs related to meeting reminders (appendix 8.1) and team code contribution details (appendix 8.2)

4. Team Performance Evaluation

The team exhibited an effective level of communication and collaboration through the project, successfully achieving partial functionality of the xylophone-playing robot. Despite facing challenges with integrating the MoveIt node, the team managed to demonstrate critical aspects of the system, such as object detection via custom colour masking and human detection with YOLO, the successful functioning of the custom end effector, the transformation of coordinates into the robot's base frame, and the Brain node's ability to manage the overall closed-loop system. These achievements were the result of consistent efforts in testing and iterative development through the unexpected setbacks experienced.

The inability to fully utilise the MoveIt node proved to be a major setback for the progression of the project, as it hindered the team's ability to test the safety measures expected during the robot's operation. This left gaps in the interactivity of the Brain node with all its dependent nodes. To address this issue, the team opted to work based on assumptions regarding the expected inputs and outputs of the MoveIt platform. This approach allowed for the implementation of the functionality of each dependent node in a closed-loop process, as described in Section 1.2.

During the presentation, the team focused on showcasing the fully implemented aspects of the system, leveraging its strengths to highlight the progress made. This approach helped mitigate the impact of the incomplete final product on the overall evaluation. If given another opportunity to present, the team would aim to improve preparation by addressing integration challenges earlier in the development process, ensuring a more cohesive and comprehensive demonstration of the system's full functionality.

As a self-assessed grade for the final presentation, the team believes they effectively addressed most aspects of the marking criteria. However, limitations in demonstrating the closed-loop operation and visualising the workspace impacted the overall performance, leading to an estimated grade of 75%. This reflects the strengths and vision of the project while acknowledging the areas where further development was needed.

5. Reflection on Collaboration

This section analyses the team’s communication, task delegation, and project management and identifies strengths and weaknesses in collaboration.

Reflection on Communication

The team was efficient in working remotely on individual tasks and organising Microsoft Teams calls when collaboration was needed. The meetings were held concise and effective, focused on deciding how the tasks should be distributed for the upcoming submissions and general project progression. Additionally, most members were providing short and accurate updates on their progression, which helped with keeping everyone on the track and understand the working principles of the other parts of the system.

Reflection on Task Delegation

We focused on distributing the tasks based on each members strengths and prior knowledge. This helped us to work on the subsystems with maximum efficiency and the fact that the subsystems such as computer vision, end effector and the brain node were built relatively quickly proves the effectiveness of our decision. On the non-technical side, task delegation worked well at moments where time constraints were present, such as the day before the demonstration. While some members were working on trying to make moveit work another member prepared the presentation for the demonstration to save time for the other members to continue their work.

Reflection on Project Management

At the beginning of the project the time management, as well as setting goals for the scope of the project has been done considering the time constraints of the project and difficulty estimation of the tasks. The selection of a xylophone playing robot was made considering the technical challenges such as the ease of detecting different coloured keys, simplicity of required arm movements and simplicity of the end effector hardware design. On the time management side, during weeks where we had more time to meet on campus, we focused on integrating the individual parts and collaborating, while during remote weeks such as the flexible week we focused on working on the individual tasks, which resulted in an efficient workflow.

Increasing In-person Meetings

Although the regular online meetings and working individually was efficient at the early stages, it became necessary to be on site more often to integrate the subsystems. It also became clear that by spending time together in the lab the team was realising issues that haven't been considered before and needed attention. It is also easier to assess and keep up with peer progression during face-to-face meetings. In similar future projects, more in-person meetings can be organised to improve these abovementioned aspects.

Improving Task Ownership and Accountability

While the team had defined roles, there were moments where responsibilities overlapped, or tasks were delayed due to unclear accountability. This experience reinforced the importance of not only assigning tasks but also regularly checking in to ensure milestones are being met. This could have been improved by having regular project update meetings, where each member presents their progression.

Expanding Prototyping Beyond Individual Components

Most of our prototyping focused on individual subsystems, like computer vision or the end effector, rather than how they interacted as part of the whole system. In future projects, we would prioritise creating early prototypes of the complete workflow, even if simplified, to identify integration challenges sooner. This would allow for more informed iterations and fewer last-minute adjustments.

Cross-Component Collaboration

Roles within the team were fairly separated, which limited chances for knowledge sharing and collaborative problem-solving. For instance, while the end effector design was well-executed, others in the team could have contributed ideas or gained experience in hardware design. In future projects, we would:

- Schedule collaborative brainstorming sessions for critical design decisions, ensuring all team members understand and contribute to the overall system architecture.
- Pair team members with complementary strengths (e.g., pairing a software-focused member with someone working on hardware) for more well-rounded skill development.
- Rotate responsibilities for tasks like debugging and integration testing to ensure everyone has hands-on experience with multiple system components.

6. Improvements for Version 2.0

6.1 System Design / Implementation Improvements

Below is a list of potential changes to the design based on what was not completed relative to the original client brief, and further options considered throughout the design process:

MoveIt implementation

This is the most obvious improvement that could be introduced to the system. As the computer vision, end effector action and local transforms were all validated to be working on the arm, the only component needed to meet the original minimum viable product would be to integrate MoveIt so that the end of the mallet can be aligned with the desired key and be played.

Second mallet integration

The initial plan beyond the minimum viable product was to adjust the end effector so it had two mallets attached to it spaced apart from each other and able to each play independently. This would have been good for playing a string of keys in quicker succession, instead of relying on the arm moving the mallet between each key for each note playing. This would likely be implemented by guiding the closest mallet at a given moment to the key to be played but may have been able to use more advanced logic in path planning to optimise the system's speed in playing successive notes.

Aruco marker integration

One clear weakness of the system was that varying lighting conditions affected the computer vision's ability to consistently detect the blue dots that were used for localising the play space to the base of the arm. Aruco markers were explored initially when considering how to detect the individual keys, but it was determined that they were too small to detect when fitted to the keys. However, they would be a suitable replacement for the blue dot system and would allow for more consistent localisation.

More robust automatic adjusting to lighting condition changes

While an automatic white balancing function was introduced to the computer vision, it did not have the desired effect in adjusting to varied lighting conditions. Therefore, a more robust system would be worth developing, potentially relying on an additional object or marker being placed in a known position with a known colour for more reliable colour balancing.

Pose estimation given view obstruction

As the initial proposed customer requirements entailed children interacting with the system, it was assumed they would be entering frame. To deal with this the human detection and stop safety feature was implemented, however in terms of key pose detection, when a human entered frame, the arm would move back and out of the way of the camera view so the key positions could be obtained again. This adds a lot of time to the operation of the system, so a proposed further development would be to figure out multiple levels of redundancy in the key detection, so that pose estimations can be obtained without all the keys being visible. This was implemented to a degree with the system not needing to see all the colours to estimate key position but could be developed a lot further.

Integrated Feedback on Strike Quality

Currently, the system assumes each strike on the xylophone produces a correct note. Adding a feedback mechanism to monitor the sound quality of each strike could enhance the system's reliability. This could involve using a microphone to detect and analyse the frequency of the produced sound and comparing it to the expected frequency for each key. Then it could adjust mallet positioning or force if a discrepancy is detected, which would make it sound better on successive hits.

6.2 Design Process Improvements

Improved Use of Version Control (GitHub Integration)

One of the biggest inefficiencies in our design process was using Teams to share code instead of a proper version control system like GitHub. This led to challenges in tracking changes, managing code integration, and ensuring consistency across team contributions. For future projects, we would:

- Use GitHub as the primary code repository from the outset.
- Set up a branching strategy to ensure work can proceed concurrently without overwriting anyone's progress.
- Regularly review pull requests to improve code quality and work on our collaboration

Regular and Structured Progress Checks

While we maintained an open Teams chat, our progress tracking was informal and irregular. This led to misaligned expectations and delays in identifying bottlenecks, such as the incomplete MoveIt integration. To address this in Version 2.0, we would:

- Schedule weekly progress meetings with clear agendas and action items.
- Use a shared task tracker like Microsoft Planner to assign tasks, set deadlines, and monitor progress.

Earlier and More Iterative Integration Testing

Integration between subsystems like computer vision, localisation, and the end effector was delayed until late in the term. This caused rushed adjustments, such as last-minute changes to the brain node. To improve, we would:

- Define clear integration milestones early in the project timeline, ensuring individual components are tested as part of the larger system well before the final demonstration.
- Begin testing high-priority integrations, like MoveIt and end effector control, as soon as basic functionality is implemented in other components.
- Use a simulation environment like Rviz earlier and more extensively to identify and resolve compatibility issues.

7. Analysis of External Factors for Version 2.0

7.1 Mass Production Considerations

To scale the system for mass production, significant changes would be needed in both hardware and software. Using the UR5e robotic arm was good for prototyping but is cost-prohibitive for large-scale deployment. A simplified, custom-designed robotic arm could be developed with fewer degrees of freedom, focusing only on the movements necessary for xylophone playing. Realistically, as long as it's robust, safe and looks interesting enough for the intended target audience of children to engage with it as an educational STEM tool, a lot of functionality can be taken away.

For software scalability, the system must be streamlined to work with plug-and-play configurations. This includes integrating pre-calibrated settings for instruments, minimising setup time for the educators that would be using it. Further, developing modular components that can be reconfigured for other educational tools (e.g., different instruments or tasks) would broaden the system's appeal in schools and museums, ensuring scalability across all the considered use cases from our client brief.

7.2 Humanitarian and Societal Impacts

The system has the potential to really benefit education by engaging children in STEM concepts through hands-on interaction with robotics through the lens of something they're familiar with like music. This lines up with efforts to make technical fields more accessible and appealing to young learners, particularly in under-resourced areas. Partnering with organisations that promote STEM education could allow us to deploy future versions in schools or public spaces where traditional robotics demonstrations may be inaccessible, such as libraries.

Additionally, the system could be adapted for inclusive educational purposes, such as supporting children with disabilities. For instance, implementing gesture-based melody input or voice commands would allow children with motor impairments to interact with the system. By emphasising adaptability and inclusivity, the system could serve as both a teaching tool and a platform for fostering creativity across diverse audiences.

7.3 Environmental and Sustainability Factors

The current prototype relies on some parts like 3D-printed materials and commercially available robotic arms, which wouldn't be environmentally sustainable at scale. Moving to biodegradable or recycled materials for components like the mallet holder would reduce the system's environmental footprint. Also, ensuring the robot consumes minimal energy by optimising the motion planning algorithm and using efficient servo motors could make the system more sustainable for long-term use in schools or museums.

Also, designing the system for longevity and ease of repair would reduce e-waste. Parts that can be easily replaced or upgraded would make sure the system remains functional over time, reducing the need for complete replacements.

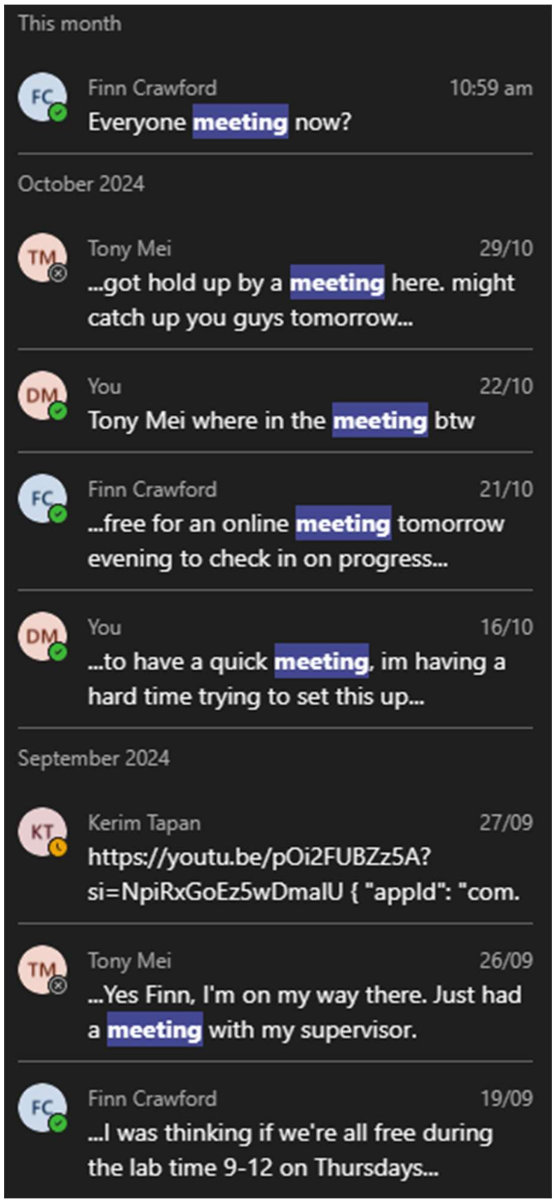
7.4 Ethical Considerations

Safety is a critical ethical consideration, particularly given the system's proximity to children. While the current design includes human detection for halting operations, further steps could be taken to address edge cases where accidental contact might occur. For instance, integrating soft materials into the end effector or using force-limiting actuators could further reduce the risk of injury. Additionally, offering comprehensive safety guidelines and training materials for educators would ensure safe and responsible operation.

Another ethical concern is the system's potential to replace traditional teaching methods or human interaction in educational settings. It could be a great teaching tool, but there's an ever-growing concern that machines have the potential to take jobs in the education space, so it needs to be made clear from the outset that the system will act as an educational aid rather than any kind of autonomous instructor.

8. Appendices

8.1 Chat Logs



This month

Finn Crawford 10:59 am
Everyone meeting now?

October 2024

Tony Mei 29/10
...got hold up by a meeting here. might catch up you guys tomorrow...

You 22/10
Tony Mei where in the meeting btw

Finn Crawford 21/10
...free for an online meeting tomorrow evening to check in on progress...

You 16/10
...to have a quick meeting, im having a hard time trying to set this up...

September 2024

Kerim Tapan 27/09
https://youtu.be/pOi2FUBZz5A?si=NpiRxGoEz5wDmalU { "appId": "com.

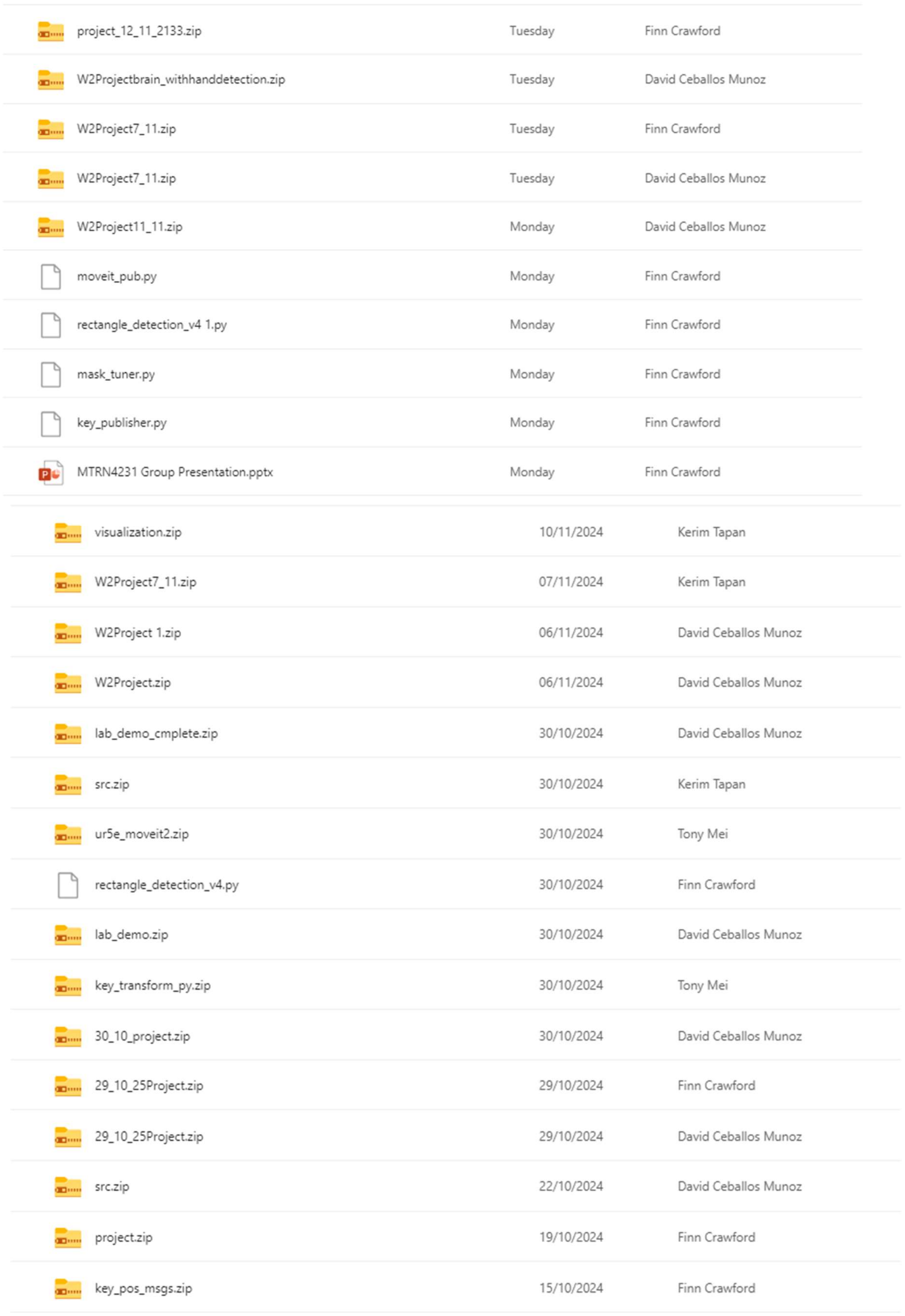
Tony Mei 26/09
...Yes Finn, I'm on my way there. Just had a meeting with my supervisor.

Finn Crawford 19/09
...I was thinking if we're all free during the lab time 9-12 on Thursdays...

Figure 3 - Chat logs

8.2 Team Code Contribution Details

Detailed breakdown of code contributions from the Teams shared files. Some code contributions are not included, such as work that was done on the lab PC but uploaded from David or Finn's accounts despite being contributed to by Kerim, David and Finn.



project_12_11_2133.zip	Tuesday	Finn Crawford
W2Projectbrain_withhanddetection.zip	Tuesday	David Ceballos Munoz
W2Project7_11.zip	Tuesday	Finn Crawford
W2Project7_11.zip	Tuesday	David Ceballos Munoz
W2Project11_11.zip	Monday	David Ceballos Munoz
moveit_pub.py	Monday	Finn Crawford
rectangle_detection_v4 1.py	Monday	Finn Crawford
mask_tuner.py	Monday	Finn Crawford
key_publisher.py	Monday	Finn Crawford
MTRN4231 Group Presentation.pptx	Monday	Finn Crawford

visualization.zip	10/11/2024	Kerim Tapan
W2Project7_11.zip	07/11/2024	Kerim Tapan
W2Project 1.zip	06/11/2024	David Ceballos Munoz
W2Project.zip	06/11/2024	David Ceballos Munoz
lab_demo_cmplete.zip	30/10/2024	David Ceballos Munoz
src.zip	30/10/2024	Kerim Tapan
ur5e_moveit2.zip	30/10/2024	Tony Mei
rectangle_detection_v4.py	30/10/2024	Finn Crawford
lab_demo.zip	30/10/2024	David Ceballos Munoz
key_transform_py.zip	30/10/2024	Tony Mei
30_10_project.zip	30/10/2024	David Ceballos Munoz
29_10_25Project.zip	29/10/2024	Finn Crawford
29_10_25Project.zip	29/10/2024	David Ceballos Munoz
src.zip	22/10/2024	David Ceballos Munoz
project.zip	19/10/2024	Finn Crawford
key_pos_msgs.zip	15/10/2024	Finn Crawford

Figure 4 - Code contributions

8.3 Control Loop Diagram

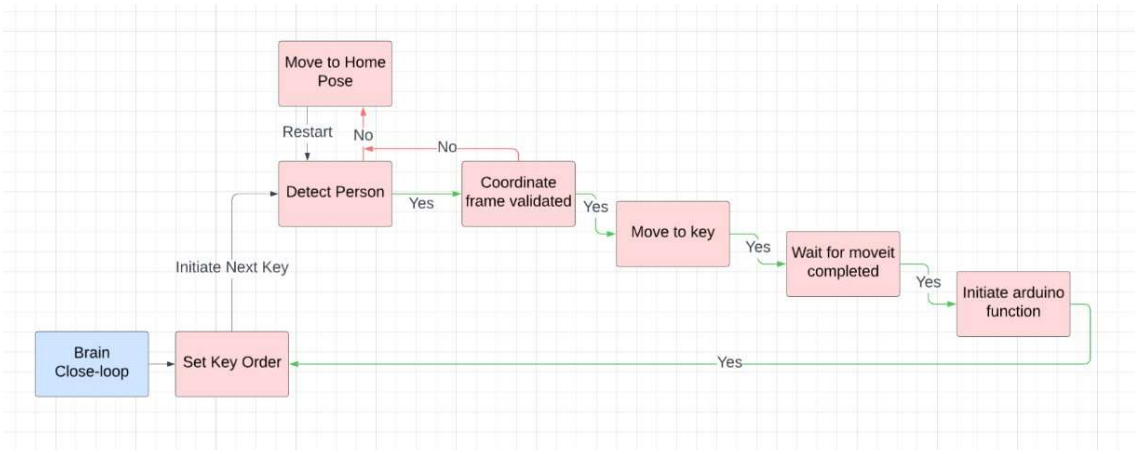


Figure 5 - Control loop logic