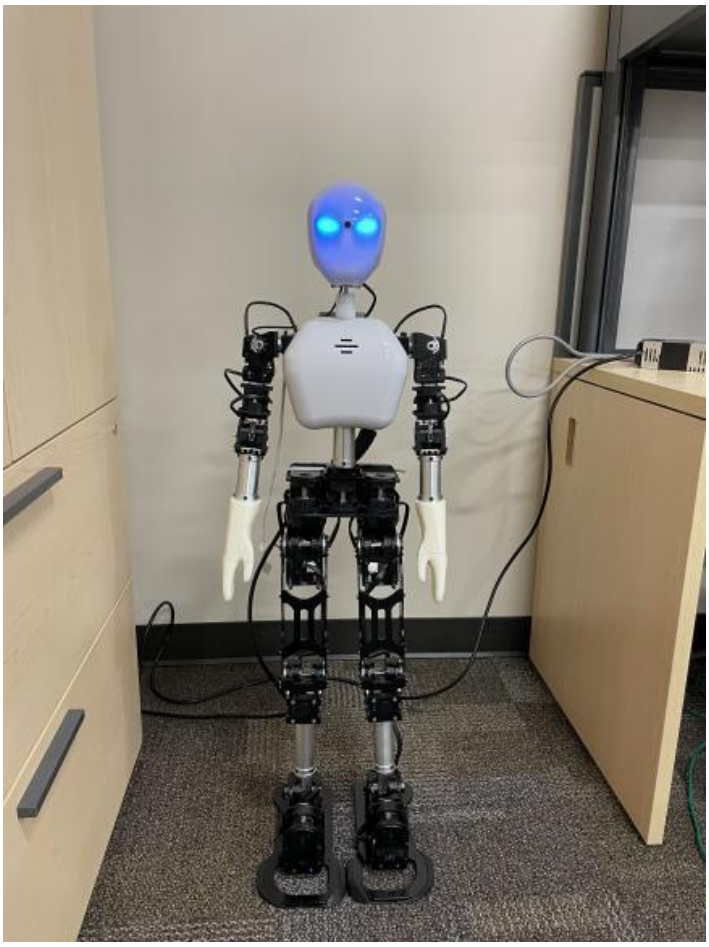
# Robot Control featuring Oculus VR

Collaboration of Ian Bryenton, Vince Giordano, Kazhan Sofy, George Larumbe

Advised by Prof. Sharon Perry

**The UXA-90 robot is a humanoid shaped robot designed to imitate humanlike movement. This project was conducted to utilize the UXA-90's functionalities along with those of the Meta Quest 2 virtual reality (VR) headset to control the UXA-90 through VR and controllers.**



https://sites.google.com/d/10zrvw3t3CBOGyzua_xynrl9vghIrIGCk/p/1C4gsBwiaVkQDDNnAz21LkxoodPPWqys1/edit

# 1 ABSTRACT

Working on the UXA-90 Robot has been a challenge to say the least. None of us had any experience working with a robot or an Oculus VR headset. Despite this limitation, we willingly took on this project topic because we believed it would be a good learning experience.

We were originally planning to program the Aldebaran Nao, but there were difficulties with locating its battery. After nearly a month of searching, our team decided to shift our focus over to the UXA-90 Robot instead, as it was in good shape and had everything needed to get up and running. The Aldebaran Nao, while being relatively simple to control, didn't have as many functionalities as the UXA-90 Robot. This was because individual motors & limbs could be controlled, giving us more options in terms of what we wanted to do with the robot. These additional functionalities were a result of the previous team creating an API to streamline the robot's movement. Before being able to do any robot programming or testing, our group had to contact the previous users of the robot and revive the API that was created for it. We managed to contact a member of the previous group that previously worked with the robot, Andrew. With his help, we were able to revive the API into a new Raspberry Pi. During this period, we came to the agreement that we wanted to incorporate a VR headset in this project.

The goals that we wanted to accomplish with the robot and VR headset were controlling where the robot walked via the headset, streaming what the robot sees to the headset, and finally tracking head and arm movements of the user and relaying them to the robot.

# 2 TABLE OF CONTENTS

# 3 RESEARCH

The RoboBuilder UXA-90 Humanoid Robot is a robot which is designed to imitate the skeletal structure of a human body, making it possible to produce humanlike movements. Various programming sources can be used to control it in many ways. It also contains emotional gestures and mouth expressions with synchronized sounds.

Before being able to roll into our development phase, we needed to discover the functionalities and uncover the technical requirements of the UXA-90 robot. The robot would need to be able to follow motion commands; it would need to be able to process individual motor requests. The functionalities will be discussed in the functionality section

**Phases of Development**

## Phase 1

We decided to split our goals we planned to accomplish with the robot into two phases. Phase 1 consisted of basic programming such as programming the robot to walk using the VR controllers. Phase 2 consisted of more complex developments including streaming the camera to the VR headset & tracking the users head and arm movement.

For phase 1, we needed to figure out how to make API calls to the raspberry PI using the inputs from the VR headset controllers. As a proof of concept, we used Unity to map computer keys to API calls. Once we successfully did that, we worked on receiving inputs from the VR controller.

This is when some of our first challenges began appearing. With the hardware aspect, we noticed that if the robot is left in the standing position for too long, the motor of its left knee can overheat causing the robot to fall over. As a result, we had to turn off the robot in between testing. This causes an increase in the time it takes to test the robot.

With the software aspect, button presses on the VR headset caused multiple API calls, input lag between the button presses and robot movement, and IP address changes in the raspberry Pi. This change of IP addresses in the raspberry pi was an issue that had to be addressed immediately, as we couldn't communicate with the robot from Unity. We contacted Andrew for assistance on automating the IP generation process with a website. For the other software related bugs, we learned how to use a debugger in Unity to solve those problems.

We felt we had completed phase 1 when the robot could move in all cardinal directions (except backwards) with minimal input lag from the VR controllers and moved onto phase 2.

## Phase 2

Phase 2 started with us attempting to access the robot's camara. We did this with the help of Andrew once again. We immediately recognized that the robot's built in camara was going to be an

issue. The video stream was very laggy and borderline unusable. We decided to buy a raspberry pi camara to replace it with. This is currently a work in progress.

We then proceeded to move on to the head and hand tracking portion of this project. For the head, we started by gauging the robot's head rotation limits. Once that was done, we determined the possible range of motion for the head, then compared that to how far a person's head would typically rotate to make it comfortable. We then mapped robot head rotation to human head rotation using a simple slope formula to generate the corresponding angle for the robot. This was necessary due to the headset using values from –1 to 1 and mapping the lowest robot head angle to –1 then scaling the range to the maximum robot head angle to 1. As of now, the robot's head can be fully controlled by the VR headset by simply moving your head.

With that done, we then moved to working on the arm tracking feature. We once again started by mapping out the rotational limits of the two motors located on either shoulder. We started with only two of the four motors to simplify the arm movements we would have to track. This is still a work in progress.

The challenges in this phase were the need to replace the robot's camera, the robot's head rotations controls would sometimes be inverted, the headset's starting position (origin) would change if you took off the headset and put it back on in a different position and trying to figure out how to map complex hand movements. As of right now, we have fixed all the bugs in the head tracking portion of this project.

## Conclusion

There have been many ups and downs while working on this project. From thinking that the robot was broken when its knee motor gave out, to having to create a Meta account, this project has been a rollercoaster and our team faced many challenges. But with hardships comes ease. As a result, we have not only learned a lot about robotics, Unity, and VR technology, but also about working with a dedicated team on a complex project.

As of right now, we have completed roughly 70%-80% of our requirements. The only things we have left to check off are the robot's camera streaming to the VR headset and tracking the user hand movements to the robot. We currently have roughly four weeks left to finish up.

This project was only made possible because of Andrew and his team's work. Their API is the foundation of this project. We want to do the same for the next group that works with the UXA-90 Robot. Ensuring that the next team can build off our work is imperative. We plan on writing a detailed guide for setting up Unity, the VR headset, the API, and our source code.

# 4 Previous Group's Website & Report, and what they did

Blah blah blah

# 5 RESEARCH

Andrew and his team provided us with detailed documentation on the UXA-90 Robot, REST API, and raspberry pi. As a result, this allowed us to completely focus our research on our project requirements. The majority of the research we conducted was on the innerworkings of Unity and C# documentation.

## 5.1 UNITY/C# DOCUMENTATION

Learning Unity was one of the biggest hurdles of this project. None of our team members had ever worked on any game development related work. Thankfully, Unity provides ample documentation on setting up the editor, APIs, assets, etc.

We quickly realized that we wouldn't be able to fully rely on just the Unity documentation. Unity uses the C# scripting language for everything. When it came to non-game-design related programing (.NET framework), we used Microsoft's C# documentation as reference.

# 6 APPLICATION

## 6.1 UNITY/CODE

Unity is a game engine designed to develop video games on different platforms, such as Xbox, PC, and more. It includes extensive libraries developed for assisting game developers in accomplishing various tasks in their projects. However, for our project, we learned that some of these libraries could be used to develop an application for VR headsets that would allow us to send user input to a Raspberry Pi. Additionally, these libraries are universal across VR platforms, so Unity could generate an application for any given VR platform and the Raspberry Pi could still translate the commands.

## 6.2 MOVEMENT

### 6.2.1 Walking

- If the user pushes the joystick in any of the cardinal direction the robot will then take a single step in that direction.
    - An API call is made based on the direction input given. For example, if the joystick is held to the left, the request sent would be **[example of API call]**

### 6.2.2 Head movement

- While the user is wearing the VR headset, any head movements will be tracked and mapped to the robot' head.
    - **[how it's done]**

### 6.2.3 Arm movement

- While the user is holding both controllers, any arm movements will be tracked ad mapped to the robot's arms. (To a certain extent)
    - **[how it's done]**

### 6.3 CAMERA

- The robot's internal camera is streamed to the VR headset, enabling the user to see what the robot sees.
    - **[Work in progress]**

# 7 RASPBERRY PI

We are using a Raspberry Pi 4 with Ubuntu 22.04 LTS installed. Robot Operating System 2.0 version Humble and Node.js LTS version 16.

### 7.1 WHAT WE USE IT FOR

- Making API motor function calls
- Accessing robot's internal camera
- IP generation

# 8 OCULUS VR HEADSET

The VR headset used in this project is called the Oculus Quest 2. It was developed by Reality Labs, a division of Meta. The Oculus Quest 2 accurately tracks the user's head and body movements without the need for any external cameras.

### 8.1 WHAT WE USE IT FOR

- Control where the robot walks using the joysticks on the controller.
- Control functions (sit down, stand up, etc.) using various button inputs from the controller.
- Control the robot's head and arm movements via the tracking capabilities of the Oculus Quest 2.

# 9 SETUP

### 9.1 UNITY

### 9.2 RASPBERRY PI (OUR STUFF + REFERENCE OTHER GROUP)

### 9.3 OCULUS VR HEADSET

### 9.4 UXA-90 (REFERENCE OTHER GROUP)

# 10 CHALLENGES AND ASSUMPTIONS

**(feel free to add any ideas Kaz; ill expand on this) (Ian lmk if this is too whiny)**

Working on a project that involves a robot, Unity, and VR headset is undoubtedly going to pose several challenges for any team. These challenges were only magnified by the fact that none of us had ever worked with any of these technologies. As a result, we had to overcome a massive learning curve throughout the development stage.

Another challenge the team faced was working with a very outdated robot. Working with outdated technology always presents challenges, as compatibility issues with newer software and outdated hardware leads to limited functionality. While the Raspberry Pi that was integrated by the previous team addressed a lot of these issues, we were very limited with what we could program the robot do.

Furthermore, the fact that development could only be done in our assigned room limited our productivity. This restriction limited the team's ability to work remotely or collaborate with other team members in different locations.

Despite these challenges, we were able to learn quickly and adapt to the new technologies. We managed to coordinate everybody's busy schedules during the development phase of the project effectively. Overcoming these challenges was imperative to delivering a successful project, and we believe we did just that.

# 11 VERSION CONTROL (GITHUB)

# 12 TEAM WEBSITE

# 13 C-DAY

# 14 APPENDIX

## 14.1 REFERENCES

## 14.2 GANTT CHART

**Accessing the work space**

The UXA 90 is locked with a combination code and kept safe in a case in room J170. It is also kept under surveillance so that it is not removed from the premises of the campus or used by unauthorized persons. Access to the robot is granted to the team members under the approval of professor parry and administration of KSU.

Additionally, room J170 was our designated work room. Here, team members performed all our project activities including research, testing, and programming throughout the semester. As enforced by KSU administration, assigned student ids were necessary to gain access to the room and a timesheet of members check in and check outs were recorded to ensure safety of room and robot access.

Phase 1

**Safety Handling the UXA 90 and Training Certification**

**Training certification**

Upon learning and mastering the previous group's training certification, our team members ensured that these safety precautions are to be

taken when handling the robot, including creating a safe environment for the robot to operate in.

-When removing the UXA-90 out from its case, lift using the plywood rest and strings attached

-Connect the power supply/adapter to check if the power works before using the battery

-Inspect for damages after every use

-Do not place the robot on an elevated surface such as a table to prevent it from falling, operate it on the floor

-Make sure that the built in fans are not covered to prevent overheating

-Be cautious of movements and motions that may damage the robot

Boot-up and Power-off Positions

To boot up the robot properly, sit it in the squatting position, then plug in the power supply and push the power button for 3 seconds.

While testing the robot, it should be positioned into a squat. Otherwise, if left unattended it should be turned off.

To ensure that the robot is properly turned off, ensure the robot is in the squatting position, hold the power button for 3 seconds and then unplug it from the power supply.

# (UXA 90 Squatting)

## Standing Positions

The UXA 90 has two standing positions. Straight legged and bent knees.

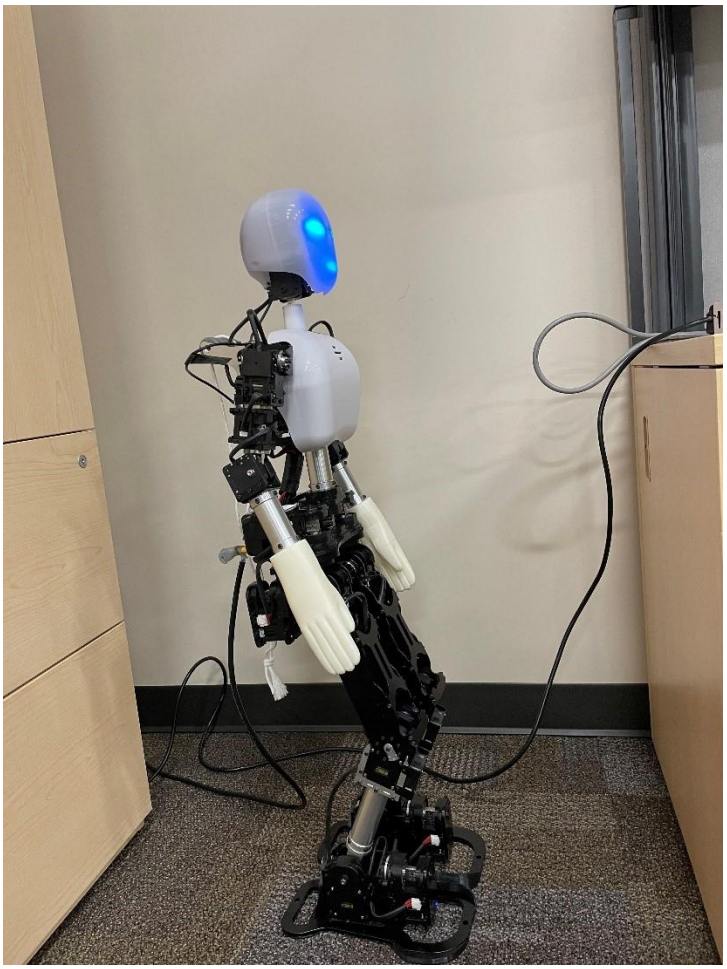
The straight legged position is for longer periods of standing.

The bent knees position is when the robot is prepared to start walking.

Please note, the robot should not be standing in either position for long. If you want it still powered on, but you need to wait before testing it again, have it squat.

(UXA 90 Squatting)

(UXA 90 standing straight legged)



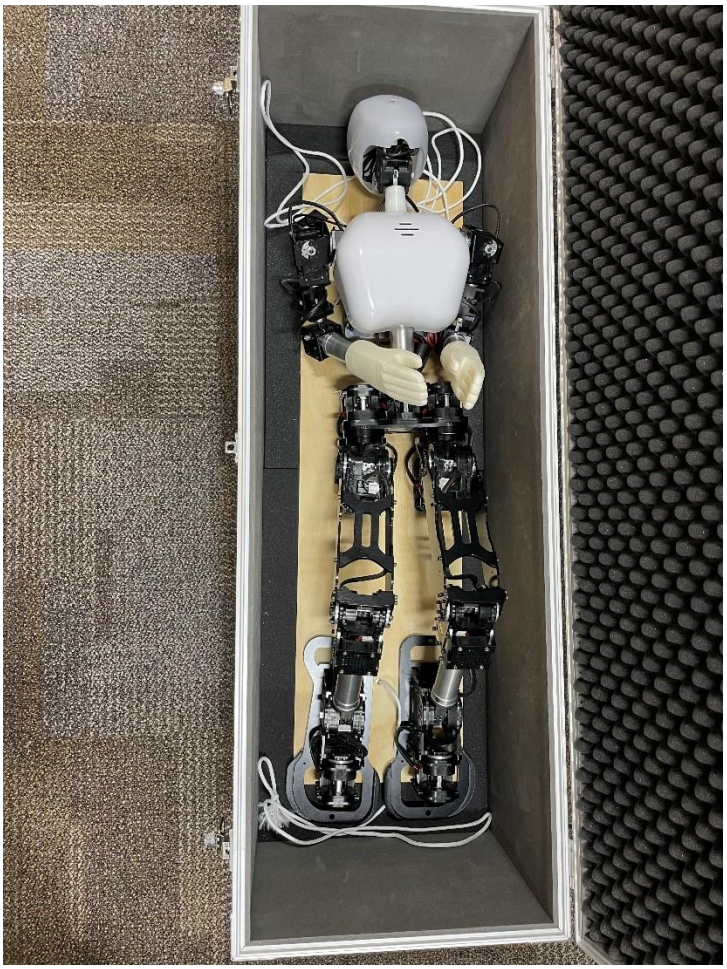
(UXA 90 standing knees bent)

### 4.4.3 Packing Up the UXA 90

After properly shutting down the UXA 90, lay it down on the wooden board outside of the crate. Ensure that the head is placed on the end with the 2 strands of rope and the feet lay on the end with one strand.

(UXA 90 on the wooden board before packing away)

Next, carefully lift the UXA 90 and place it in the crate.



(UXA 90 laying properly in the crate)

Finally, place foam padding under the UXA 90’s head and around the robot as you see fit. Then, fill in the gaps with the miscellaneous equipment.


(How the UXA 90 should look before closing the crate)

Headset and robot- make sure those are safe

## Reviving the API with the previous team

During phase 1, we met with the previous team who worked with the UXA-90 to gain access to the API which was built for the robot.

## Controlling/Mapping movements to the robot using Oculus VR

## Working with Unity

Unity is a 3D/2D cross platform IDE and game engine used by developers to import and assemble assets, write code to interact with your objects, create or import animations for use with an advanced animation system. Our team used Unity to accomplish

### Raspberry pi camera and headset(phase 2)

**Coding**

**Architectural drawings/Models**

## Conclusion

There have been many ups and downs while working on this project. From thinking that the robot was broken when its knee motor gave out, to having to create a Meta account, this project has been a rollercoaster and our team faced many challenges. But with hardships comes ease. As a result, we have not only learned a lot about robotics, Unity, and VR technology, but also about working with a dedicated team on a complex project.

As of right now, we have completed roughly 70%-80% of our requirements. The only things we have left to check off are the robot's camera streaming to the VR headset and tracking the user hand movements to the robot. We currently have roughly four weeks left to finish up.

This project was only made possible because of Andrew and his team's work. Their API is the foundation of this project. We want to do the same for the next group that works with the UXA-90 Robot. Ensuring that the next team can build off our work is imperative. We plan on writing a detailed guide for setting up Unity, the VR headset, the API, and our source code.

2. Abstract / Overview / Executive Summary

3. Table of Contents

4. Background information -

5. Requirements / Analysis / Results / Development – SDLC - depends on the type of project

This is the body of your project report – all the effort, decisions, streams of consciousness:

Requirements derived from working with Team and Project Owner and Faculty Sponsors

Include mockups of screens along with description of tool used to build mockup

Analysis of tech platforms you reviewed before selecting the one to use for your project

Tech platform – describe in detail the tools you are using – use TECHNICAL TERMS

Include high level architecture drawings

Development of software project

Discuss your development effort – what went well, roadblocks, work arounds

Include detailed architecture drawings, data models, etc

Results – where are you with this software

Research

Mobile app –

Web application, desktop application

Project Planning and Management

Discuss collaboration tools and project meeting schedule

Revisit your project plan and include updated Gantt Charts – show you can manage projects

Version Control

Test Plan and Test Report – recall your list of requirements becomes a test plan

Summary / Conclusion

APPENDIX – customize for your report