Controlled Navigation Telepresence Robot (CNTR)

Project Lead: Jack Bluethmann

Members:

Jonathan George, Vishjit Chandramohan, Lucas Ybarra, Avery Althaus, Alex Alvarez, Davi Araujo, Kevin Garcia, Madden Schoener, Krishna Senthil, Indra Thyagaraja, Jacob Yanes

TEXAS A&M UNIVERSITY Engineering

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Intro & Problem Definition

Robots continue to prove useful in the modern day to operate safely and effectively in environments that are too dangerous or inaccessible for people. Environments such as the vacuum of space are incredibly hazardous for humans, where robotic systems can perform inspection, maintenance, and repair tasks without exposing humans to any risk. These robots offer improved safety while also increasing efficiency, as they can work continuously and withstand harsh environments. Because of these advantages, robotics has become an essential tool for modern hazardous environment operations.

Mission Statement

CNTR's mission is to develop a modular humanoid robotic system capable of operating in hazardous or inaccessible environments through precise VR-based control. By integrating a rotating sensor head, a mobile base, and interchangeable task-specific arms, we aim to replicate essential human capabilities while protecting operators from dangerous conditions. Through iterative prototyping, hands-on mechanical and electrical testing, and continuous refinement of our control systems, we seek to demonstrate CNTR's adaptability across a wide range of challenging tasks. Ultimately, the goal is to validate CNTR as a reliable, flexible platform while expanding our team's experience in practical robotic design, development, and real-world deployment.

Mechanical

The CNTR is designed to mimic human motion as closely as possible to ensure an intuitive and seamless transition between human and robot movement. The robot features different subsystems such as

- Camera Module
- 4 DoF detachable arm
- Electronic torso holder
- Turreting base

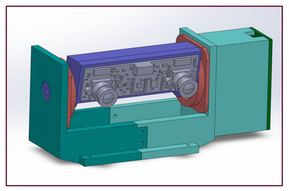


FIgure 1. Camera and Neck Assembly

Mechanical Cont.

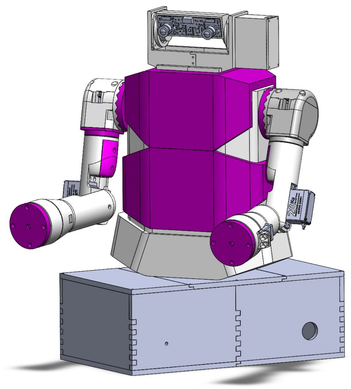
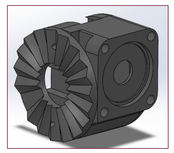


FIgure 2. Full Robot Assembly

The head module was designed so that the camera could move to mimic the movement of the user. The key design characteristics to meet this include:

- Pitch Rotation to look up and down
- Mounting to the Torso Top Plate
- Bearings for smooth camera rotation

The Arm Mechanism has 4 degrees of freedom which is the minimum number of DoF for accurate control of an arm to hit every point in space while having rotation control on the end effector. The arm is composed of 3 main parts, the shoulder, bicep/forearm, and the wrist.



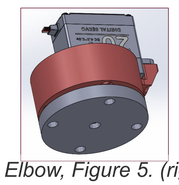
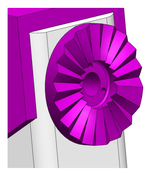
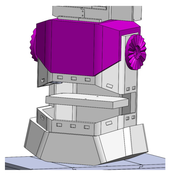


Figure 3. (left) Shoulder, Figure 4. (center) Elbow, Figure 5. (right) Wrist

The torso has three important functions. It holds the servos that pitch the arm, attaches the clocking mechanism. This ensures that whenever a new arm is attached, the servo will always be at the correct degree. The torso also holds all the electronics, including the pwm extender, the Pi 5, and the buck convertors.

The Base uses an FRC system turret. It uses a bearing stackup to constrain the turret and uses a stepper motor to rotate the entire upper body of the robot.





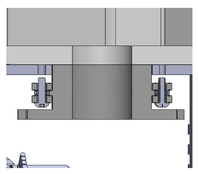


Figure 6. (Left) Shoulder Gear, Figure 7. (Middle) Expanded Torso, Figure 8. (Right) Turret Mechanism

Electrical

The electrical system is comprised of:

- 9x Servos
- 2x Neo 550 Synchronous Motors
- 1x Raspberry Pi 5
- 1x PCA9685 PWM Extender
- 1x Studio Camera
- 2x Buck Converters
  - 12-5 V, 12-6.8 V
- 1x Nema 17 Stepper Motor
- 1x Stepper Motor Controller
- 1x 12.8V LiFePO4 Battery

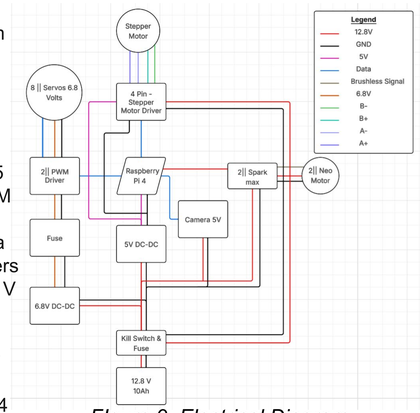


FIgure 9. Electrical Diagram

System Overview:
The robot operates on a 12V electrical system, chosen due to the wide availability of compatible battery options. A Lithium Iron Phosphate battery was selected for its proven reliability and its ability to deliver the high current required by the drive motors.

Control System (VR Interface):
The robot is controlled through a virtual reality interface. Unity periodically transmits CSV files over an internet connection containing positional data from the controllers. The Raspberry Pi receives this data and passes it to the servo control program, which performs the necessary calculations and safety checks before issuing commands to the arm segments.

PWM Extender:
To support multiple actuators while minimizing hardware constraints, a PWM extender is used. This reduces the number of required GPIO pins on the Raspberry Pi, allowing for future expansion of the system's degrees of freedom. The extender communicates with the Raspberry Pi via I2C, making it a fast and low-maintenance addition to the system.

Servo Motors (Actuation):
Each robotic arm is driven by four servo motors, providing three degrees of freedom along with an end-effector capable of precise rotation. These servos receive control signals through the PWM extender, enabling precise movement based on the VR input.

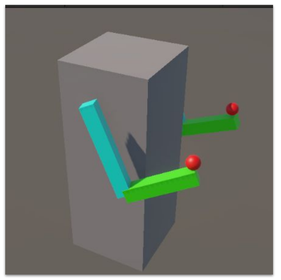


FIgure 10. Unity Simulation

Software

Unity:
VR simulation of the physical robot was created in Unity to test motion paths, catch self-collisions before they reach the hardware, and make sure only safe, accurate commands are sent to the Raspberry Pi. To do this, the robot arm uses a two-bone inverse kinematics solver that calculates the shoulder, upper arm, and elbow angles needed to reach any target point in real time, using the law of cosines to solve the geometry formed by the upper arm, forearm, and the line to the target. If a target is out of reach, it is automatically adjusted to the closest valid position, and the resulting angles are smoothly animated before being exported as servo commands to a CSV file that controls the physical hardware. Collision avoidance works through a trigger zone inside the chest. If the arm's path would cause a collision, the system recalculates and finds the next best route to the target, redirecting the arm without stopping the motion.

TCP:
Servo commands are transmitted from Unity to the Raspberry Pi over a TCP connection. TCP was chosen because it guarantees that every movement command arrives in full and in order. Due to the safety constraints in inverse kinematics movement, decided to make each packet a CSV string specifying servos with their target angle and execution order. This way, Unity can precisely control the sequence in which joints move.

Raspberry Pi:
A dedicated control system on the Pi receives the incoming TCP packages and drives the physical hardware in the command sequence. Servo and motor commands are routed through a PCA9685 PWM extender over I2C, which provides 16 channels of 12-bit PWM output at 50Hz. There is also a two-link IK solver on the Pi as a fallback mode to use without the headset. This computes the shoulder and elbow angles from raw X/Y coordinates using the law of cosines.

Video Stream:
Live video from a stereo camera on the robot is captured and streamed into Unity through the same local network. FFmpeg handles encoding and transmission. In Unity, the decoded frames are loaded directly as a texture so the operator can have a live first-person view of the robot's environment.

Next Steps

- Creating Specialized end-effectors
- Adding a 5th DoF in the arm for wing kinematics
- Modifying the base so that it can move
- Fully integrating VR control with the robot
- Having the power system fully onboard
- Optimize latency