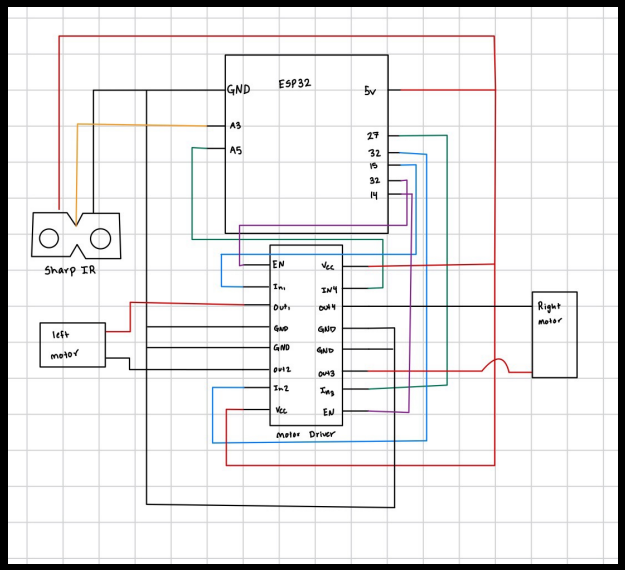
# QUEST-5:PERSONAL ASSISTIVE ROBOT

By Joshua Arrevillaga, Michael Barany, Samuel Kraft

# Project Description

- Designed a wheeled robot controlled by an ESP32 to autonomously navigate through waypoints using precise positioning data from an Optitrack motion capture system.
- Integrated Sharp IR sensor for obstacle detection and avoidance and developed a WASD-based remote control mode for manual operation.
- Built a graphical interface using Sharp IR and Streamlit to display robot positions and waypoints, sourcing real-time data from a Node.js server and TingoDB.

# Circuit Diagram

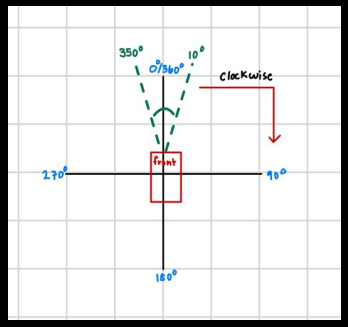


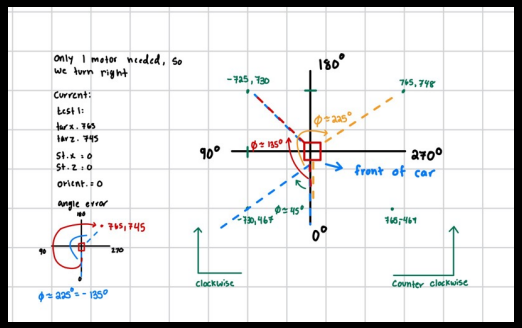
# code/espCode/Car.c

- The code uses positional data from an Optitrack system to calculate angles and distances for guiding the robot through predefined waypoints, dynamically adjusting movement based on real-time calculations.
- A Sharp IR sensor detects obstacles, and the robot switches between autonomous and WASD-based remote control modes for flexibility.
- UDP communication handles real-time data exchange for position updates and control mode changes, ensuring smooth robot operation.

# code/espCode/Car.c

- The error angle is determined by first calculating the target orientation (target_theta) using the arctangent of the negative positional differences (-dx, -dz) between the robot and the target waypoint. This accounts for the robot's coordinate system, ensuring the angle aligns with the correct directional reference. The final error angle is the difference between target_theta and current_theta, normalized to the range [-180°, 180°].



# code/wasd_server/server.js

- Command Sending: Keyboard inputs (w, a, s, d, e, 0, or space) are captured using the keypress module and passed to the sendCommand function, which converts the command to a Buffer and sends it to the ESP32 via a UDP socket.
- UDP Communication: The commands are transmitted to the ESP32's IP address (192.168.0.225) and port (3333) using the dgram module, enabling real-time control of the robot.

# Code/Indoor_robot_tracking/

- Streamlit was used to fetch and display real-time positional data for both robots by integrating with the Node.js server. The data is presented in a live scatter plot showing the robots' current positions within fixed coordinate bounds.
- Historical tracking of both robots was visualized using color-coded scatter plots that show their movements over a configurable time range, enabling analysis of their navigation paths and interaction with the environment.

# Reflections

- Adjusting the angle error computation to account for the different coordinate system was challenging, requiring careful normalization and testing to ensure accurate navigation.
- Balancing the power of each motor for proper calibration involved iterative adjustments to ensure the robot moved straight and turned consistently.
- Combining the WASD control, Optitrack-based autonomous navigation, and sensor-based obstacle detection tasks into a cohesive system required extensive debugging and synchronization.