Rapid Orbit Motion Emulator (ROME)

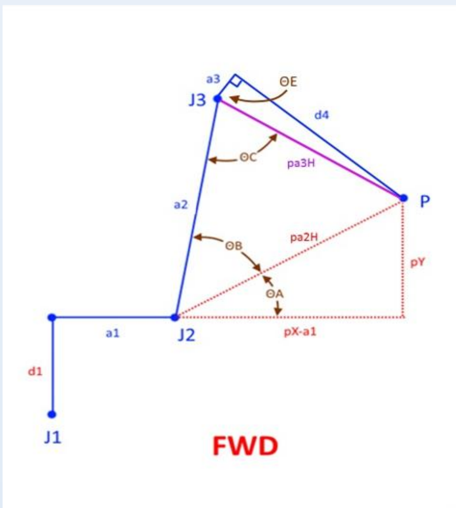
Keanu Brayman, Nathan Eckhoff, Andrew Field, Jacob Llorca, Bastain Weiss, and Tarek A. Elgohary
University of Central Florida, Orlando, FL

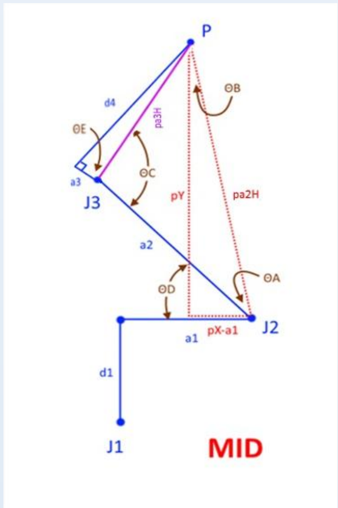
Abstract

- Cost-effective, accurate ways to emulate **orbital motion** and **maneuvers** are needed for **controls testing.**
- The **Rapid Orbit Motion Emulator** (ROME) is a nine-degree-of-freedom system consisting of an **articulated robotic arm** and a **holonomic ground vehicle.**
- Achieving full **closed-loop control** through the use of **encoders**, a **motion tracking** system, and **live telemetry**.

Background

- Software simulation, hardware-in-the-loop simulation (HILS), and **hardware emulation** are all methods engineers have been using and researching to test **spacecraft control algorithms** [1].
- The first iteration of ROME was an **open-loop** system made from a **three-wheeled omnidirectional** ground vehicle and an open-source **3D-printed** robotic arm.
- ROME is controlled by sending motor **positions** and **velocities** to the Teensy through **serial communication**
- These positions are calculated in **MATLAB** by inputting the constraints of the desired **orbit.**
- **Inverse kinematics** are then performed to transform those **positional values** into motor **angles**, and from there, motor **speeds** are calculated.



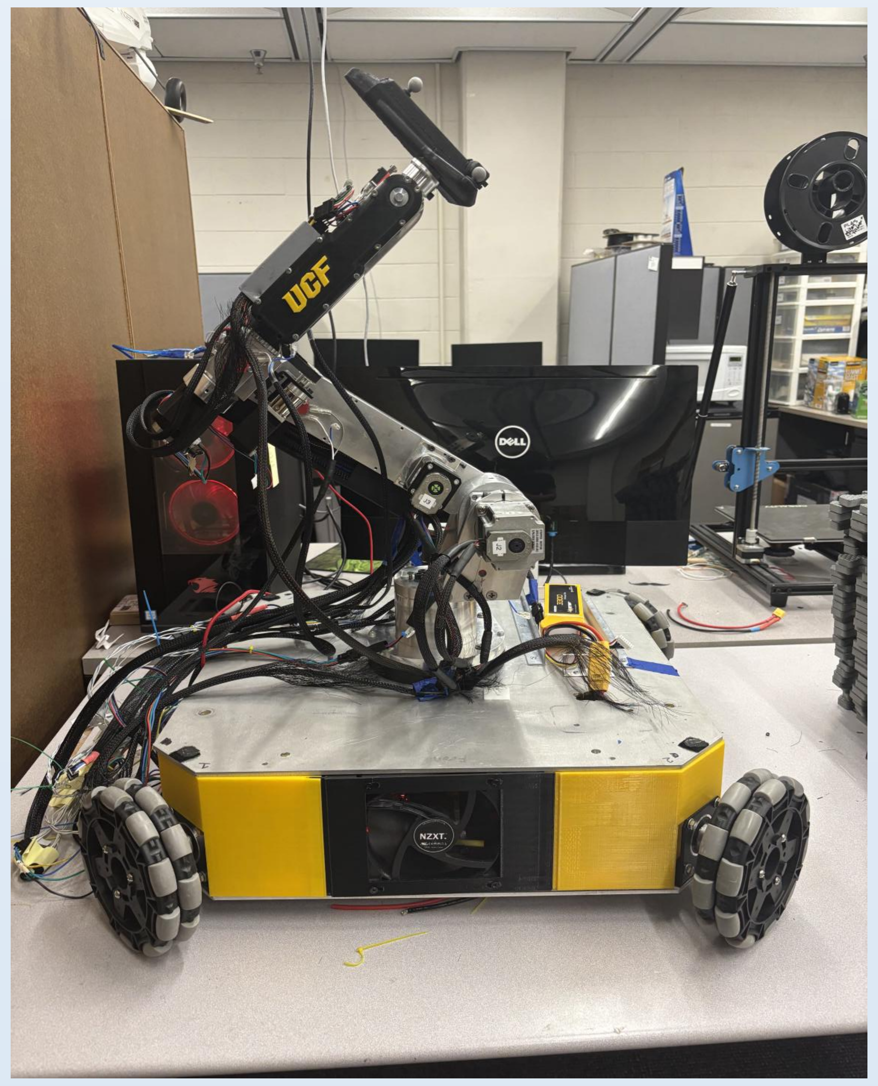


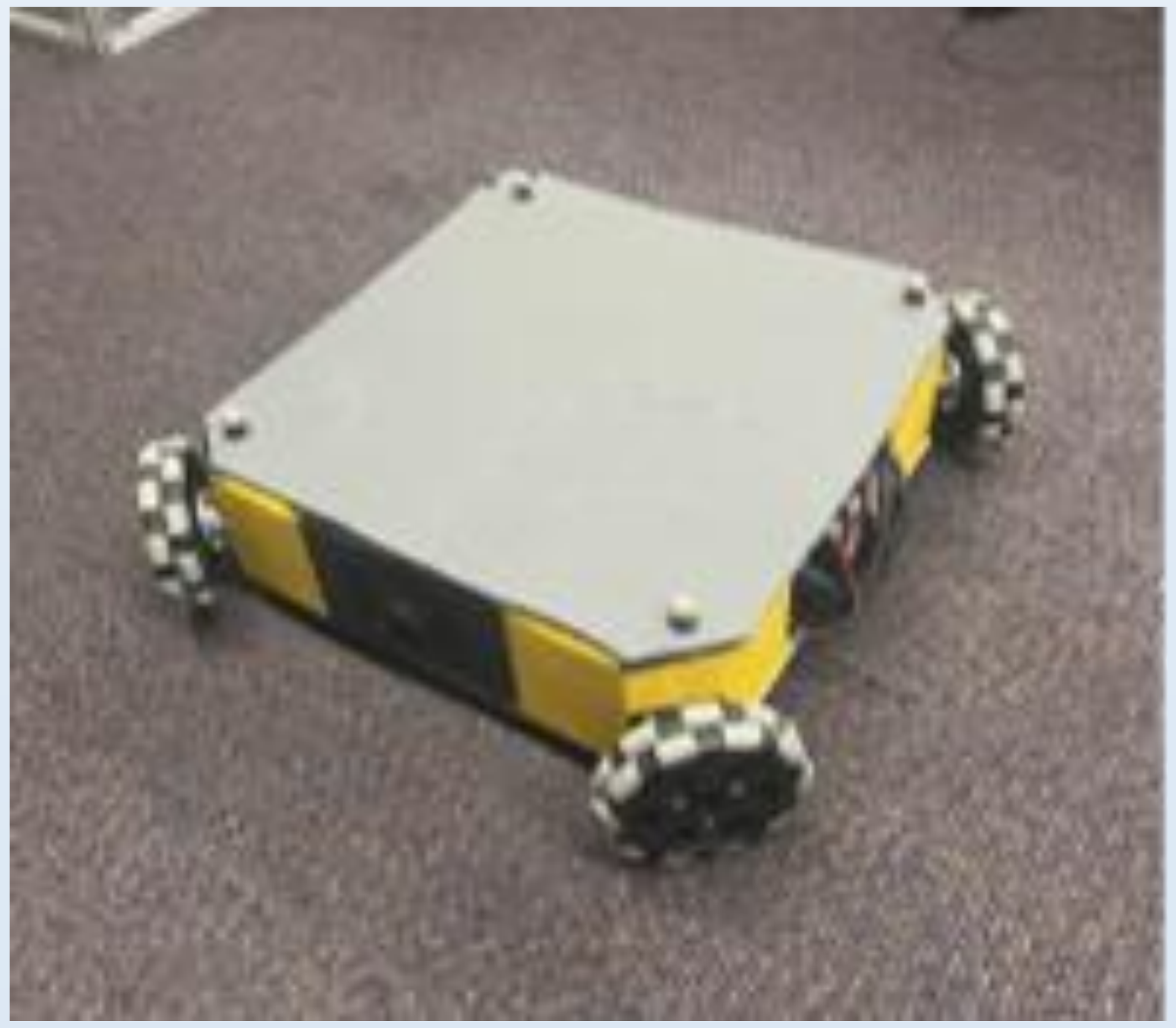
Robotic Emulator

- The robotic arm is a **modified** version of **Annin Robotics'** open-source **AR3** [2].
- It is made up primarily of **aluminum links**, connected by **stepper motors** at each joint, and **belt systems** to turn some joints.
- The arm is controlled by a **Teensy 4.1** microcontroller that is connected to 6 **stepper drivers**, 6 **limit switches**, and 5 **motor encoders** – all necessities to accurately control the robotic arm.
- The system uses MATLAB to send pre-generated **angular positions** and **speeds** to the Teensy, which **updates the position** of each motor accordingly, following a 3000-position orbital path in sixty seconds.
- The **motors** can all be controlled at the same time using this MATLAB setup, but a **calibration algorithm** is being written so that each motor knows its **location** and **limit**.
- Current work on the arm includes **constraining** the **limits** of the system, integrating **ROS 2** to work with the **Teensy**, and cleaning up the **wiring** of the arm.

Ground Vehicle

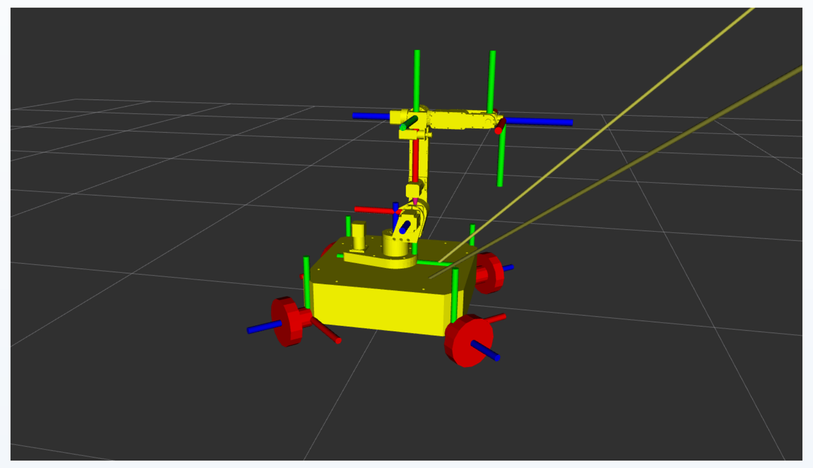
- The base is a **ground vehicle** made with four **omnidirectional** wheels in a **holonomic x-drive** configuration.
- The motors are connected to 4 **ZS-X11H motor controllers**, which are connected to a **relay**, all controlled by an **Arduino Mega 2560** microcontroller and is powered from a **22.2V 6s LiPo Battery.**
- The ground vehicle can currently follow a **pre-generated path** and is picked up by ASRL's **Motion Capture** Cameras.
- The position/velocity data is then **streamed** to MATLAB where the **updated** data points are **accounted** for **each time** the ground vehicle is **commanded** to run.





ROS 2 & Space ROS

- Using the **Denavit-Hartenberg** Convention, a **digital twin** of both the Robotic Emulator and Ground Vehicle is created in a **URDF File**.
- A **custom Omniwheel Controller** was created to **interface** with the Arduino Mega Microcontroller.
- Both the Robotic Emulator's and Ground Vehicle's **movement** is **simulated** inside of **Gazebo** while inside **Space ROS**.



Future Work

- Different **path generation** codes will be tested.
- Attaching **reflective** motion capture balls to ROME allows the **OptiTrack Motion Capture** cameras and **Motive** to track the robot's movement and create **positional data**.
- This positional data will be incorporated into **MATLAB** in real time to implement **full-system closed-loop control** for more accurate **path-following** experiments.
- For the entire system to work with a closed-loop control system, the **Arduino Mega** and the **Teensy 4.1** must be in **communication**.
- This will be made possible using either **Wi-Fi** or **Bluetooth** modules.
- To protect the lab computer from being wired to a moving robot, the computer will **send commands** to ROME's **microcontrollers** through **Wi-Fi** or **Bluetooth**.

References

[1] Seleit Ahmed E., Ketzner, Ryan, Quebedeaux, Hunter and Elgohary, Tarek A., "Rapid Orbit Motion Emulator (ROME): Kinematics" AIAA/AAS Space Flight Mechanics Meeting, January 6-10, 2020, Orlando, FL.

[2] Annin, Chris, "Annin Robotics," https://www.anninrobotics.com/, 2024

Acknowledgements

ASRL | https://mae.ucf.edu/ASRL/

Director: Tarek Elgohary

Dept. of Mechanical & Aerospace Engineering

College of Engineering & Computer Science

University of Central Florida