Autonomous Gyroscopic 2-Wheel Differential Robot

Haruka Kido, James Vrtis, Luke Anderson

Professor Tarek Elderini

EE428: Robotics Fundamentals

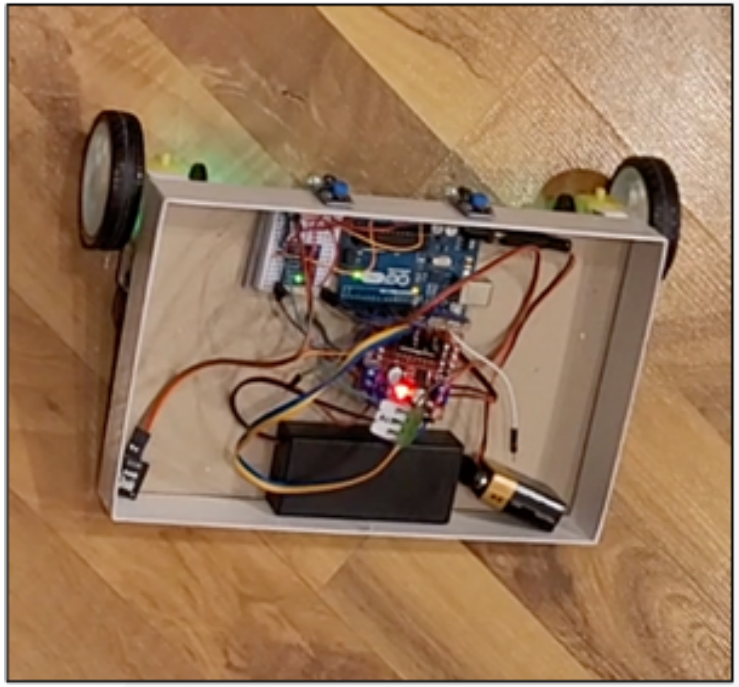
University of North Dakota

Outline

1. Introduction
2. Mathematical Model
3. Simulation Work
4. Experimental Work
5. Discussion/Future Works
6. References

Introduction/Background

This presentation demonstrates the implementation of an autonomous gyroscopic 2-wheel differential robot, including a forward and inverse kinematics simulation in MatLAB, a test hardware robot and programmed demonstration of a simple move forward and spin left motion, and a final configuration of a complete square path based on programming kinematics, gyroscopic speed responses, and remote-control functionality.



Mathematical Model

The forward kinematics of a differential drive robot, the following variables are needed: **L** (length between wheels to center), **r** (diameter of wheels), **x, y, θ** (pose matrix), & $\boldsymbol{\varphi_1}$ / $\boldsymbol{\varphi_2}$ (speed of the right /left wheels) [8]. (The inverse kinematics are explained through the simulation)

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \varphi_1, \varphi_2) \ (\textit{overall speed in global reference frame})$$

$$x_{r1}(\textit{wheel translation speed in reference position}) = \frac{r\varphi_1}{2} \quad (1)$$

$$x_{r2}(\textit{wheel translation speed in reference position}) = \frac{r\varphi_2}{2} \quad (2)$$

$$\omega_1(\textit{right wheel rotation velocity}) = \frac{r\varphi_1}{2*l} \quad (3)$$

$$\omega_2(\textit{left wheel rotoation velocity}) = -\frac{r\varphi_2}{2*l} \quad (4)$$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{r_R}{2} * cos\theta & \frac{r_L}{2} * cos\theta \\ \frac{r_R}{2} * sin\theta & \frac{r_L}{2} * sin\theta \\ \frac{r_R}{2L} & -\frac{r_R}{2L} \end{bmatrix} * \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} \quad (5)$$

Simulation Work

The $\dot{\eta}$ *is* the desired velocity for the center of the robot; containing ***x***, ***y***, and **Θ**, which respectively are the distance and angular displacement of the robot. The ***u*** and ***r*** variables are the forward and angular velocities, respectively [1]. The variable **ρ** in Equation 7 calculate the distance between the desired pose and current pose. Equation 8 & 9 calculate the angular displacement of the robot. Equation 10 calculates the angular displacement **error** between η the existing pose, once the margin of error is below a specified tolerance. Equation 11 is the **Jacobian matrix** vector by angle θ.

$$\dot{\eta} = \begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix} \quad (6)$$

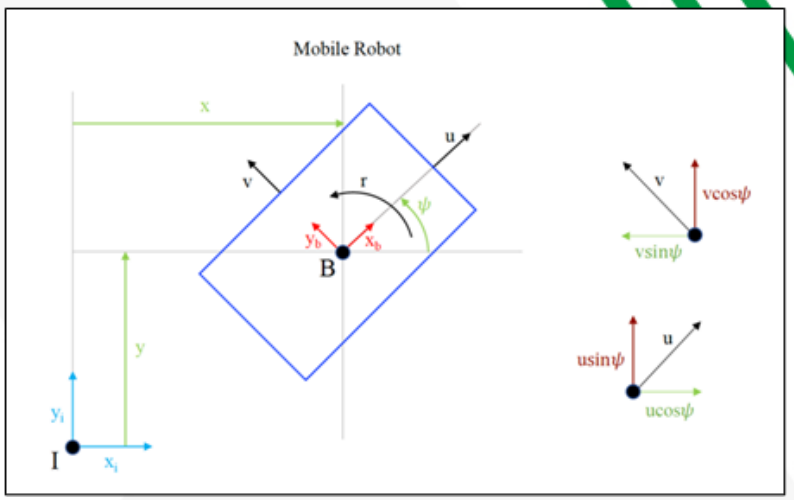
$$\rho = \sqrt{(\dot{\eta}(x) - \eta(x,t))^2 + (\dot{\eta}(y) - \eta(y,t))^2} \quad (7)$$

$$\theta_d = \text{atan}(\frac{y_d - y}{x_d - x})(t) \quad (8)$$

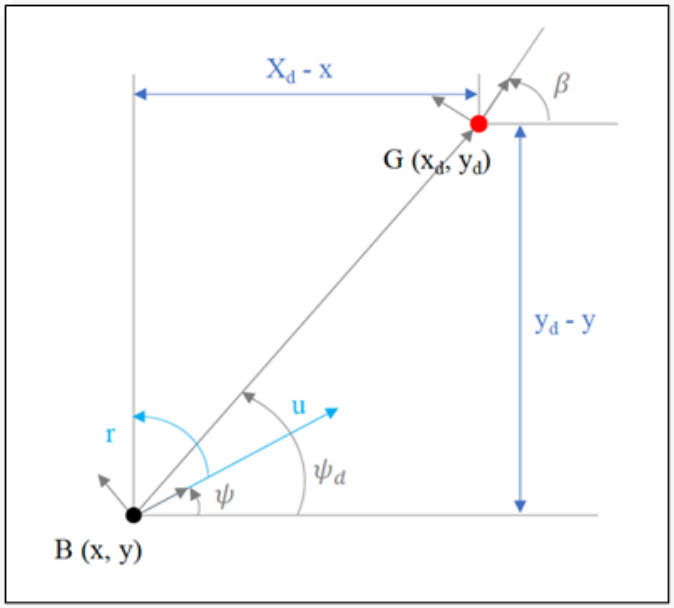
$$e_{error} = \begin{bmatrix} x_d \\ y_d \\ \theta_d(t) \end{bmatrix} \quad (9)$$

$$e_{radius-error} = \dot{\eta} - \eta(t) \quad (10)$$

$$\text{Jacobian Matrix} = \begin{bmatrix} cos\theta(t) & -sin\theta(t) & 0 \\ sin\theta(t) & cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$



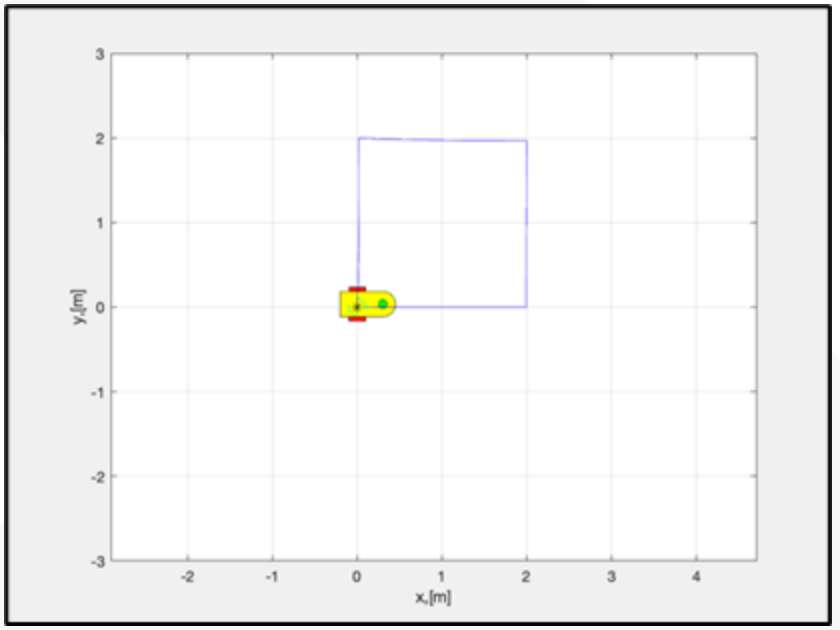
Mobile robot reference frame [1]



Mobile robot line of site reference frame [1]

Simulation Work Cont.

Found in Equation 12.b is where the matrix controller adjusts the forward and angular velocities, and can be tested upon the robot platform parameters in Equation 12.a [1].



MatLAB Simulation

Kinematics Velocity Controller

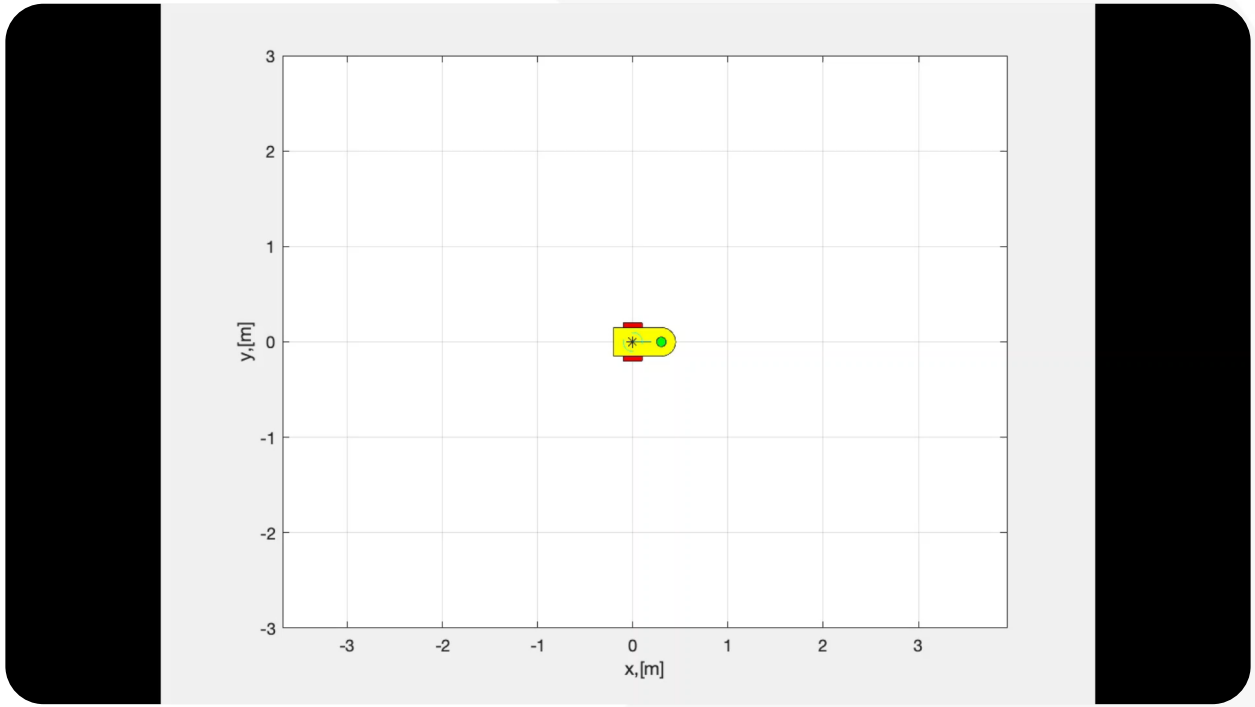
$$\begin{bmatrix} u(t) \\ 0 \\ r(t) \end{bmatrix} = Z = J^{-1} * \left(\begin{bmatrix} K_{vel-x} & 0 & 0 \\ 0 & K_{vel-y} & 0 \\ 0 & 0 & K_{vel-\theta} \end{bmatrix} * \begin{bmatrix} x_d - x(t) \\ y_d - y(t) \\ \theta_d - \theta(t) \end{bmatrix} \right) \quad (12.a)$$

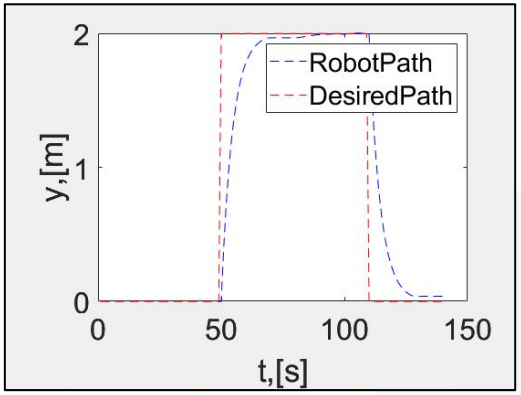
$$\begin{bmatrix} u_z(t) \\ 0 \\ r_z(t) \end{bmatrix} = \left(\begin{bmatrix} a/2 & a/2 \\ 0 & 0 \\ \frac{a}{2*L} & -\frac{a}{2*L} \end{bmatrix} * \begin{bmatrix} u(t) \\ 0 \\ r(t) \end{bmatrix} \right) \quad (12.b)$$

Position Controller

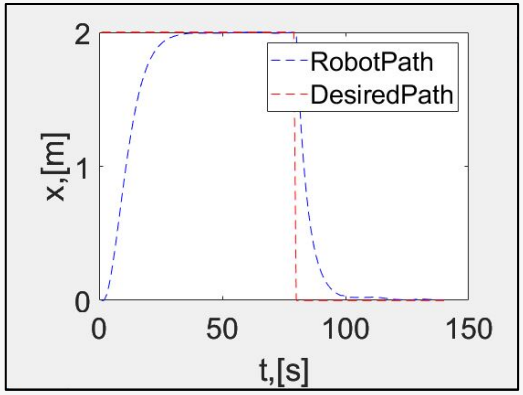
$$\eta(:, t+1) = \eta(t) + \int_t^{\infty} error(t)\, dt = 0 \quad (13)$$

Simulation Work Cont.



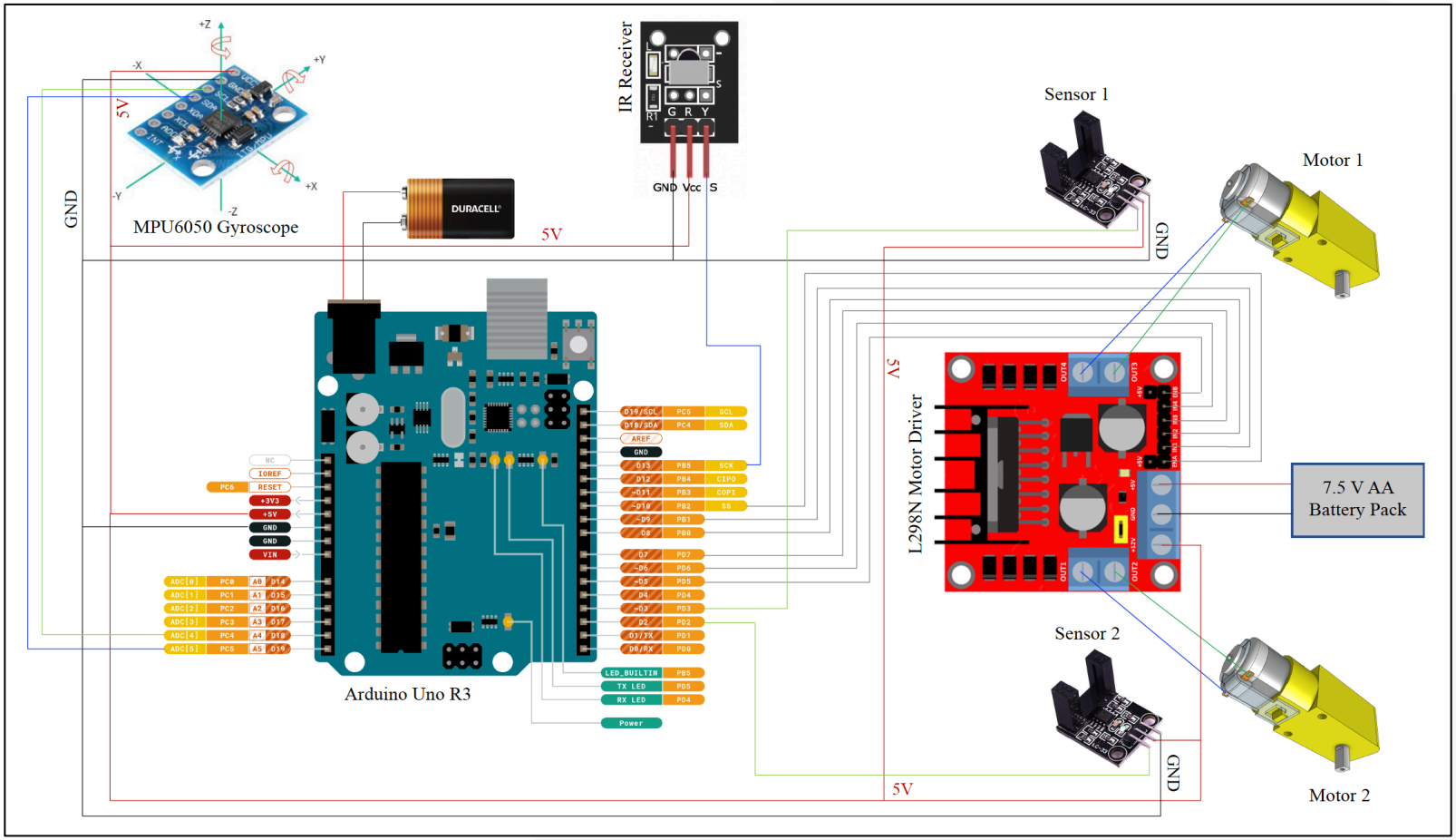


Graphical model of y-axis robot path versus time



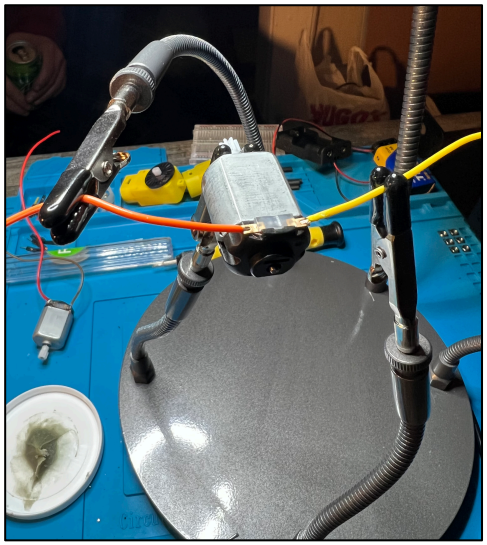
Graphical model of x-axis robot path versus time

Experimental Work:
Circuit Diagram

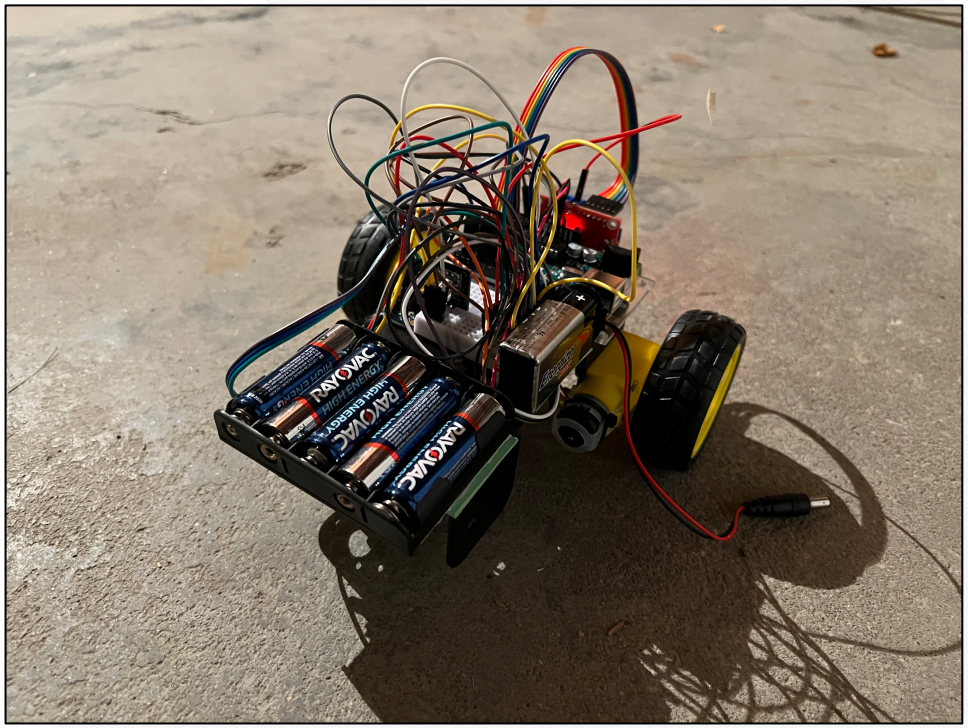


Autonomous Gyroscopic 2-Wheel Differential Drive Hardware Circuit

Experimental Work:
Preliminary Configuration and Testing

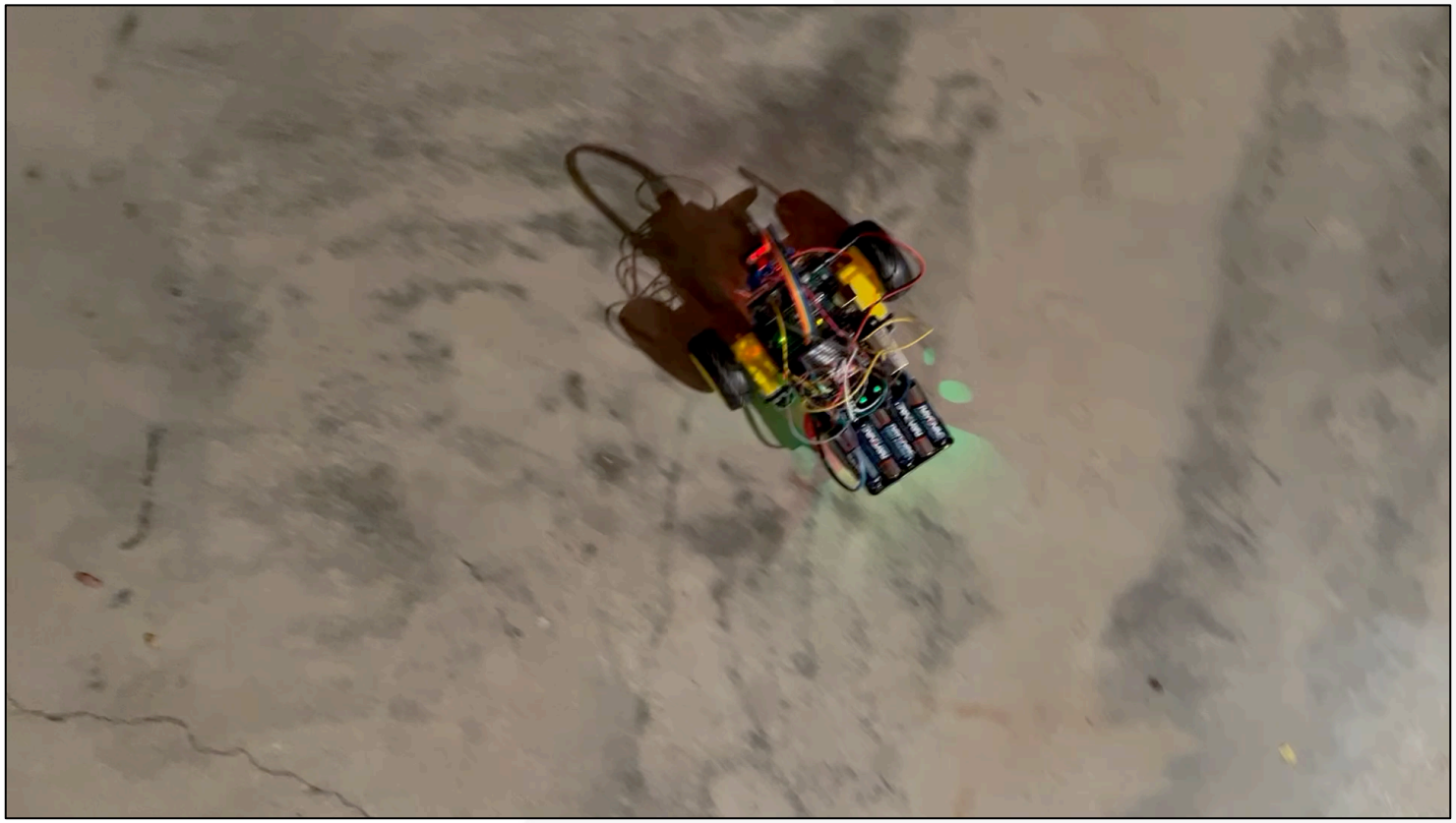


Soldering electrical leads to motor



Preliminary Test Differential Robot with front castor wheel

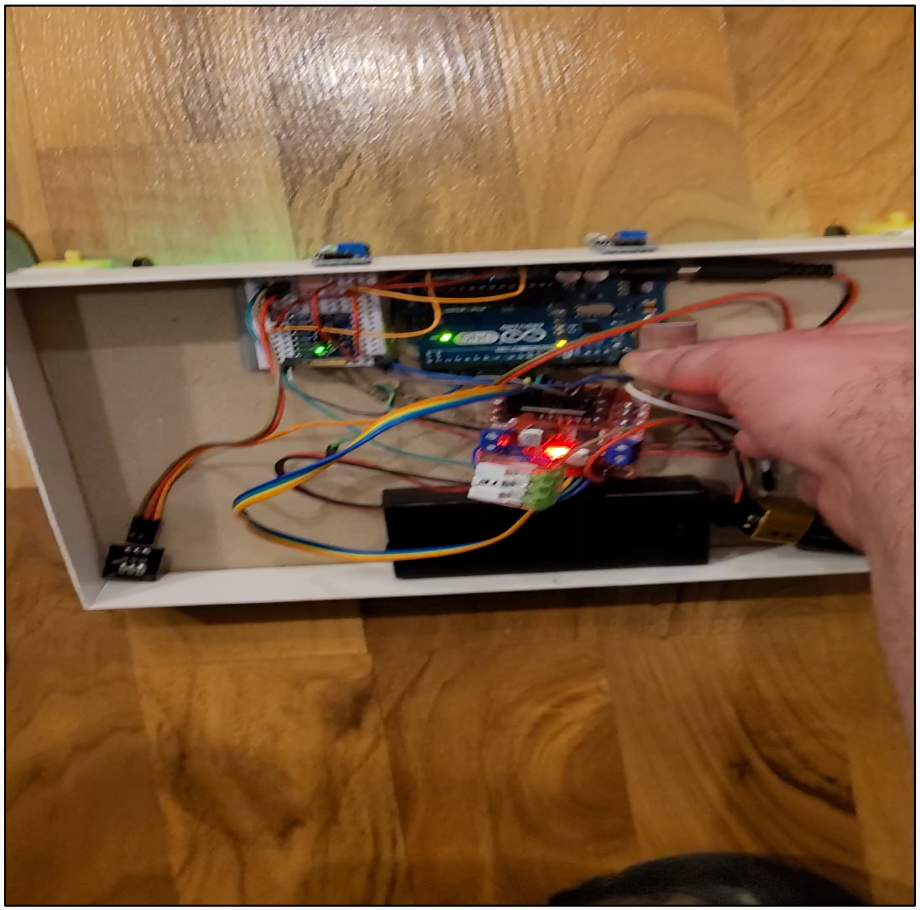
Experimental Work:
Preliminary Configuration and Testing



Preliminary Differential Test Robot in Autonomous Action

Experimental Work

- Testing to see if we could reproduce the square simulation.
- Using the MPU6050 gyro, we could control the turning angle.
- Using the encoders, we could control the distance travel from the wheels.



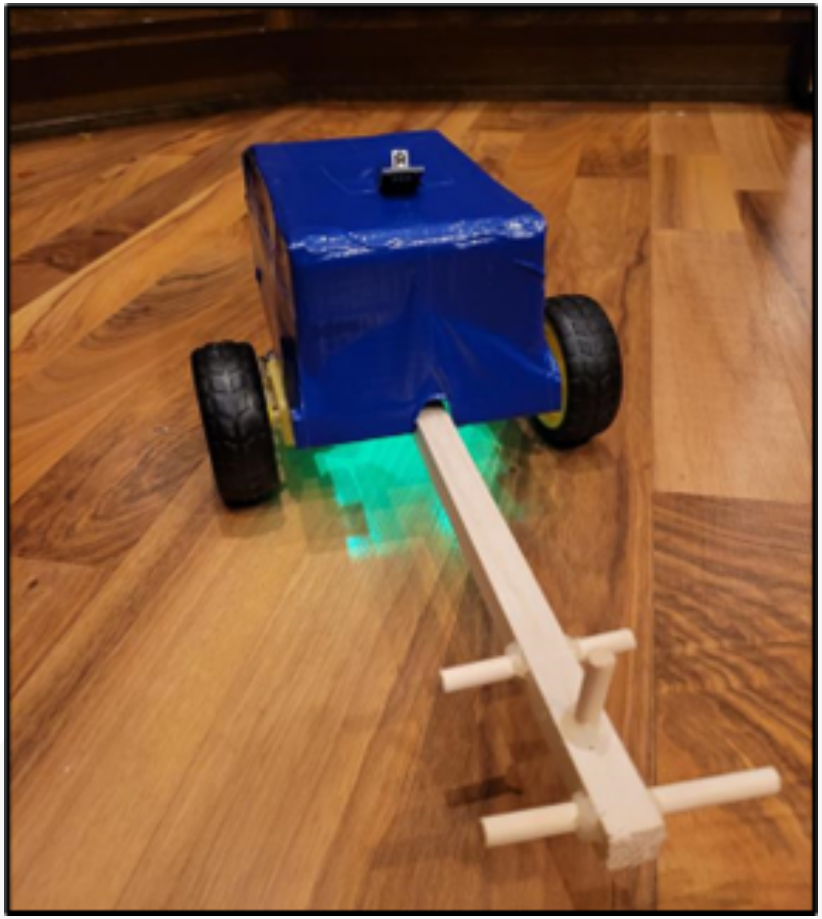
Experimental Robot with Gyro and Encoders: Square Path

Experimental Work Cont.

Combined the two robots to produce the battle bot.

General Specs:

- 2x lithium-ion batteries primary power
- 9V battery for the Arduino
- IR remote control
- Swinging spike arm for weapon



Experimental Battle Bot

Discussion/Future Works

Explore Error Sources:

- Simulations:
  - Error with the line-of-sight method used to perform the autonomous motion: would occur when $\dot{\eta}$ contained a value that was previously used in a preceding desired destination
  - Corrected when the robot aligns itself in the negative $\dot{\theta}$ orientation
  - May be caused by the mathematical line of site method to find the robot's path from the starting location to the endpoint location
- Test Drives:
  - Initial failure possibly due to electrical conductivity insufficiencies, power consumption issues, or motor driver inefficiencies, did not function to continue the motion drive in any direction after the first implementation

Explore Areas of Advancement:

- Adjusting the velocity and computational time for a loop needed for the robot to move from the starting to the ending point for the simulation
- Adding machine vision for image-processing in an autonomous robot
- Physical limitations and design changes of hardware parts (power supplies, board and sensors)
- Higher-level software adaptations and libraries for more complex manipulator systems

References

[1] A. Thondiyath and S. Mohan, "Wheeled Mobile Robots," Indian Institute of Technology, Palakkad.

[2] Siciliano, Bruno, Sciavicco, Lorenzo, Villani, Luigi, Oriolo, Giuseppe, "Robotics Modelling, Planning and Control," Springer, 2009, ISBN: 978-1-84628-642-1

[3] Warren, John-David, Adams, Josh, Molle, Harald, "Arduino Robotics," Technology in Action, 2011

[4] Edwards, Stephen A., "Assembly Instructions for a Motor Robot Car Kit 2WD, L298N Motor driver, HC-SC04 Ultrasonic module, Arduino," July 2018

[5] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, “Robotics Modelling, Planning and Control,” Springer, 2009.

[6] “Arduino UNO R3,” Arduino.cc, https://docs.arduino.cc/hardware/uno-rev3

[7] Muir, Patrick F., Neuman Charles P., “Kinematic Modeling of Wheeled Mobile Robots,” Department of Electrical and Computer Engineering, The Robotics Institute, Carnegie-Mellon University, 1986.

[8] Siegwart, Roland., Nourbakhsh Illah R. “Introduction to Autonomous Mobile Robots” Massachusetts Institute of Technology, 2004.

UNIVERSITY OF NORTH DAKOTA

Thank You! Questions?

UNIVERSITY OF NORTH DAKOTA