Self Balancing Robot

CSE 499 Embedded Controls Project

Andrew Woska, Daniel Maas, Imani Muhammad-Graham

Introduction

The project is a self balancing robot. The robot takes input from the world around it to keep it upright and balanced at all times. As the the robot tilts in one direction, there are conditions set that correct for the movement and keep the robot balanced. This is accomplished by using mounted sensors and a modified chassis in conjuncture to an embedded microcontroller.

Background

Many tools were utilized in the process of creating this system. Thye are defined as:

Control System - a set of devices that run systems using a control loop

Proportional-Integral-Derivative Controller (PID) - an optimized control system utilizing different parameters to impact the system such as angles, acceleration, and speed

Real-Time Operating System (RTOS) - an operating system utilizing real-time applications to process data at schedules times

Finite State Machine (FSM) - a system capable of changing between multiple states that are input dependent and functionally driven

Methods

The system is based off the inverse pendulum model. This is since the system essentially works off a rotating axis attempting to straighten.

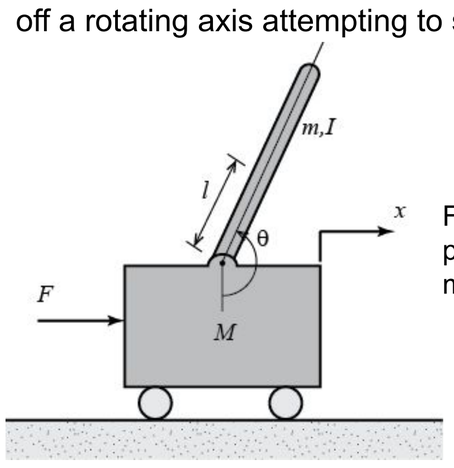


Figure 1: Inverted pendulum system model [1]

This gives us an equation to use with the system to allow changes to the angle. A Laplace transform was required to allow the system to operate as a control system [1].

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \quad \textit{(eq 1)}$$

The system was implemented as a control system with an operator, controller, plant, and feedback elements. An accelerometer and tilt sensors were chosen to act as feedback. Those elements contribute to the application as a closed-loop control system which continuously correct the angle.

Methods

A PID controller was implemented to process the control element in Figure 2.

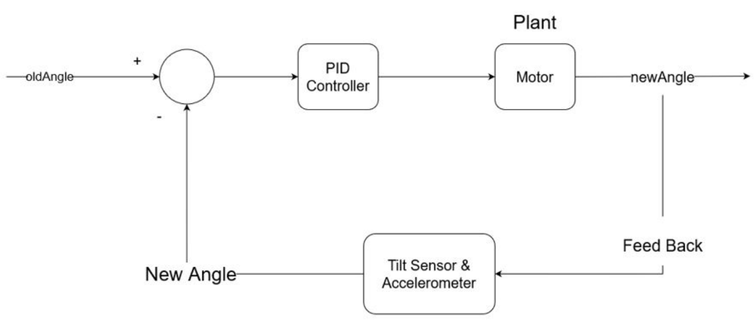
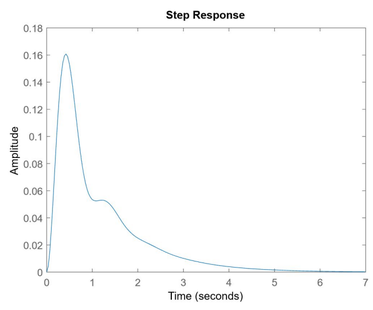


Figure 2: Control system diagram

MATLAB was used to provide a simulation for the stability for the system in Figure 3.



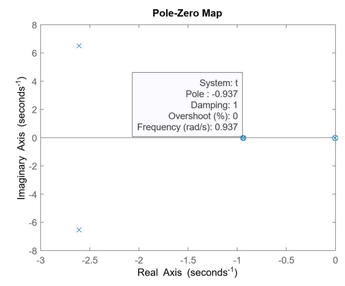


Figure 3.1 (left): Step response for system

Figure 3.2 (right): Pole-zero plot for system

The values and graphs shown show that the system should be stable after a about four seconds.

The PID values used for the control system are:

P = 10 I = 7 D = 0.75

Building the System

The system can be separated into two parts: hardware and software.

Hardware:

- Mounted tilt sensors and the accelerometer in the center of mass for accurate angle measurements
- Modified existing robot-car to fit Nucleo microcontroller, sensors, motors, power supply, and allow room to tilt

Software:

- Programmed on Mbed OS to receive signals from sensors and send signals to motor

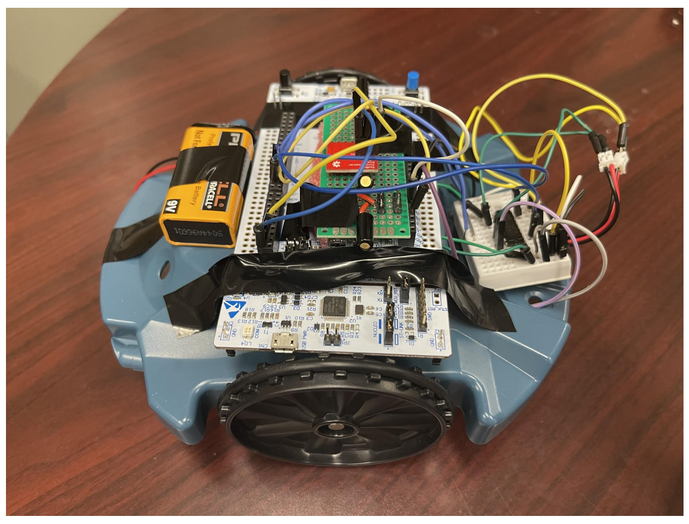


Figure 4: Physical System

Control System vs. FSM

A control system was used instead of a finite state machine (FSM) to analyze and realize the different aspects of a control system. FSMs utilize state in order to function. Control systems utilizes more specialized elements where operations factor into the system from feedback. These operations are utilized by the PID controller to control the plant. FSMs do not need these extra steps as it purely state driven.

Embedded and RTOS

Some of the memory being accessed are global variables. To protect the system from memory leakage and corruption, a mutex has been added to ensure this does not occur. In safety-critical systems, it is important to have features to prevent the system from completely crashing. With a watchdog protecting the system, in the event of partial or total system failure, the watchdog will activate and reset the system to a stable state.

Outcomes and Improvements

Although the system works, there are some concerns regarding it's stability and accuracy. Future cases to address are as follows:

Accuracy:

- Mount tilt sensors in a more effective position so they can work as intended

Software/Efficiency

- Collect data to determine optimal sample rate
- Analyze stability in order to improve functionality of controller
- Account for additional edge cases, i.e. car falls over

Motor

- Research H-bridge malfunction; fix motor to rotate at various speeds

Resources

[1] University at Michigan, "Control Tutorials for MATLAB and Simulink," University at Michigan, 2021. [Online] Available at: https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlPID