Final Project - Self Balancing Robot
EENG 348/CPSC 338 Digital Systems

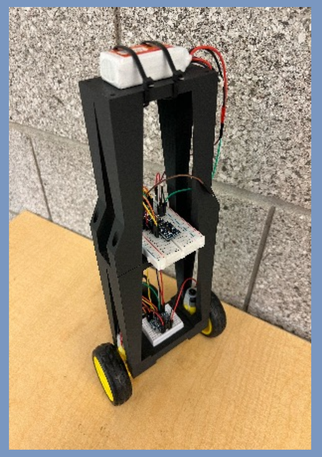
Yale SCHOOL OF ENGINEERING & APPLIED SCIENCE

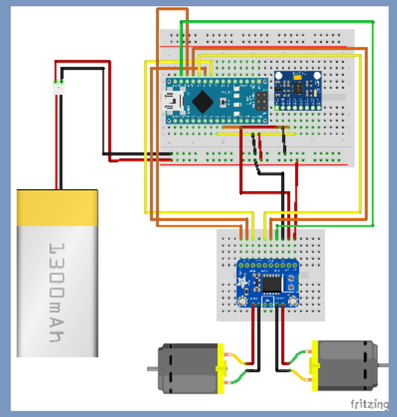
Yale

Henry Demarest[1,2], Esteban Figueroa[3]

[1]Department of Mechanical Engineering and Materials Science, Yale University
[2]Department of Computer Science, Yale University
[3]Department of Electrical Engineering, Yale University

Overview

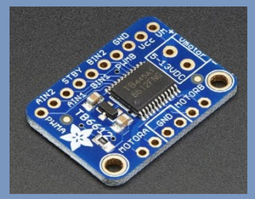


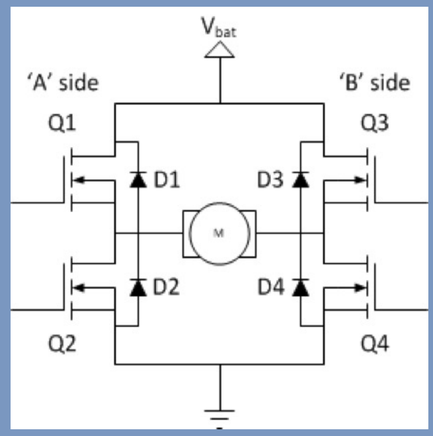


Our project revolves around a popular controls problem where a controller balances a two-wheeled vertical robot. This robot uses an Arduino Nano, MPU-6050 IMU sensor, a TB6612 Motor Driver, a 7.4V LiPo rechargeable battery, and two DC motors. We programmed the Arduino in the Arduino IDE and programmed the controller based on IMU accelerometer data to provide the feedback.

H-Bridge Motor Driver

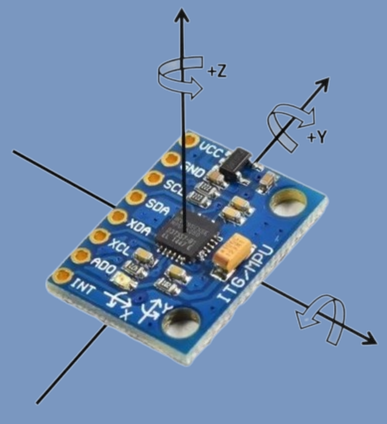
We are using two 3-6VDC motors with a 200 RPM motor with a 1:48 gear ratio to allow the robot to drive and balance. Since we need to the robot to be able to go forwards and backwards, we are using the TB6612 H-Bridge motor driver by Adafruit. The H-Bridge is a transistor network which allows current to run on forward or reverse bias. For example, activating Q1 and Q4 allows for current to run in one direction through the motor, and activating Q3 and Q2 in the reverse direction.

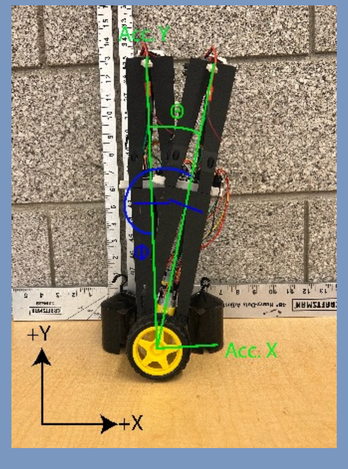




Inertial Measurement Unit

Our MPU-6050 is a 3-axis gyroscope and 3-axis accelerometer which lives on board next to the Arduino Nano on the 2nd stage of the robot. The orientation of our IMU calls for using the roll orientation (Y direction per the IMU graphic to the right), as the acceleration measurement. The symmetry of the robot allows us to arbitrarily set +Y or -Y to be forward or backward on the system. The I^2C protocol is used to communicate with this IMU.





Controller

To process the sensor data and create a motor output, IMU data is first fed through a complementary filter. This combines the accelerometer and gyroscope data to create a more consistent output, using the slowly-drifting gyroscope reading and the jumpy accelerometer readings. These readings are then passed into two PID controllers. One uses an estimated current position of the robot to tilt the robot back towards the center and the other used the approximated angle to guide the robot towards the desired angle. Both required significant tuning.

$$currentAngle = \alpha.(previousAngle + gyroAngle) + (1-\alpha).(accAngle)$$

HPF LPF

$$\alpha = \frac{\tau}{\tau + dt} = \frac{0.75}{0.75 + 0.005} = 0.9934$$

$$Output = Kp.e(t) + Ki\int e(t).dt + Kd.\frac{d}{dt}e(t)$$

where $e(t) = setpoint - input$

Complementary Filter Equation (top) and PID controller equation (bottom)
(Source: https://www.instructables.com/Arduino-Self-Balancing-Robot-1/)