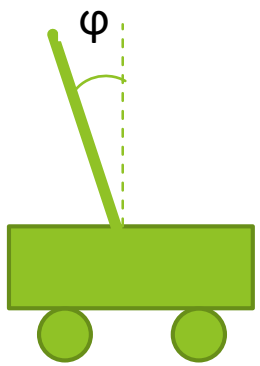
# Self-Balancing Bot

Arduino Based Robot

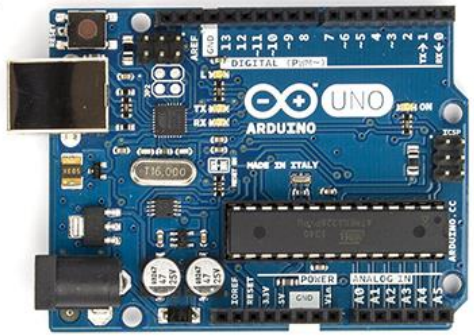
# Introduction

- Problem Statement – Inverted pendulum on a cart, is a classical problem where an inverted pendulum is placed on top of a cart and the cart moves in the direction of tilt to compensate the tilt hence creating balance of the inverted pendulum.

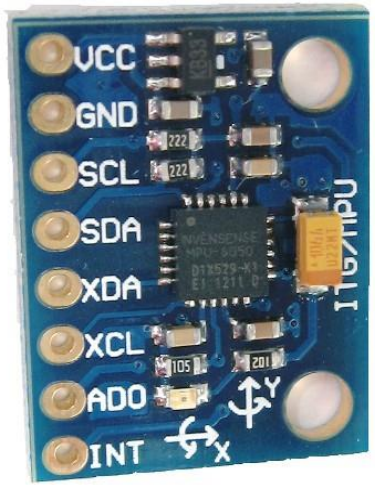


# Hardware Used

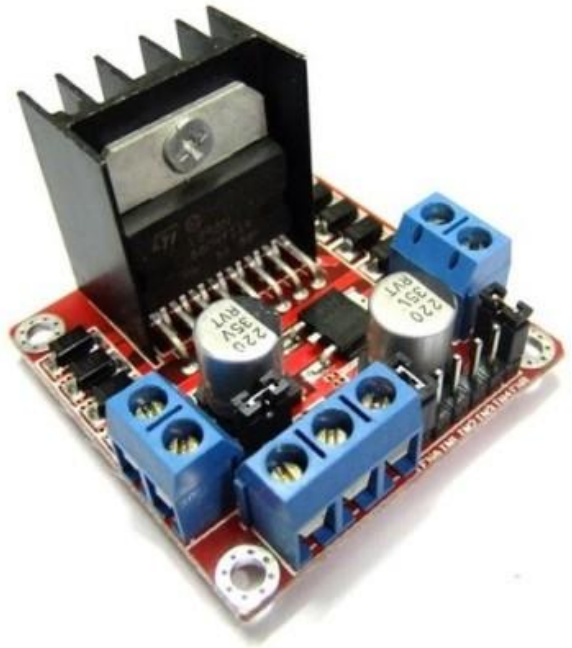
- Arduino Uno R3 - The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.



- MPU6050 - The MPU-6050 sensor contains an accelerometer and a gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino. It also provides filtered readings thus no external filters are to be used.

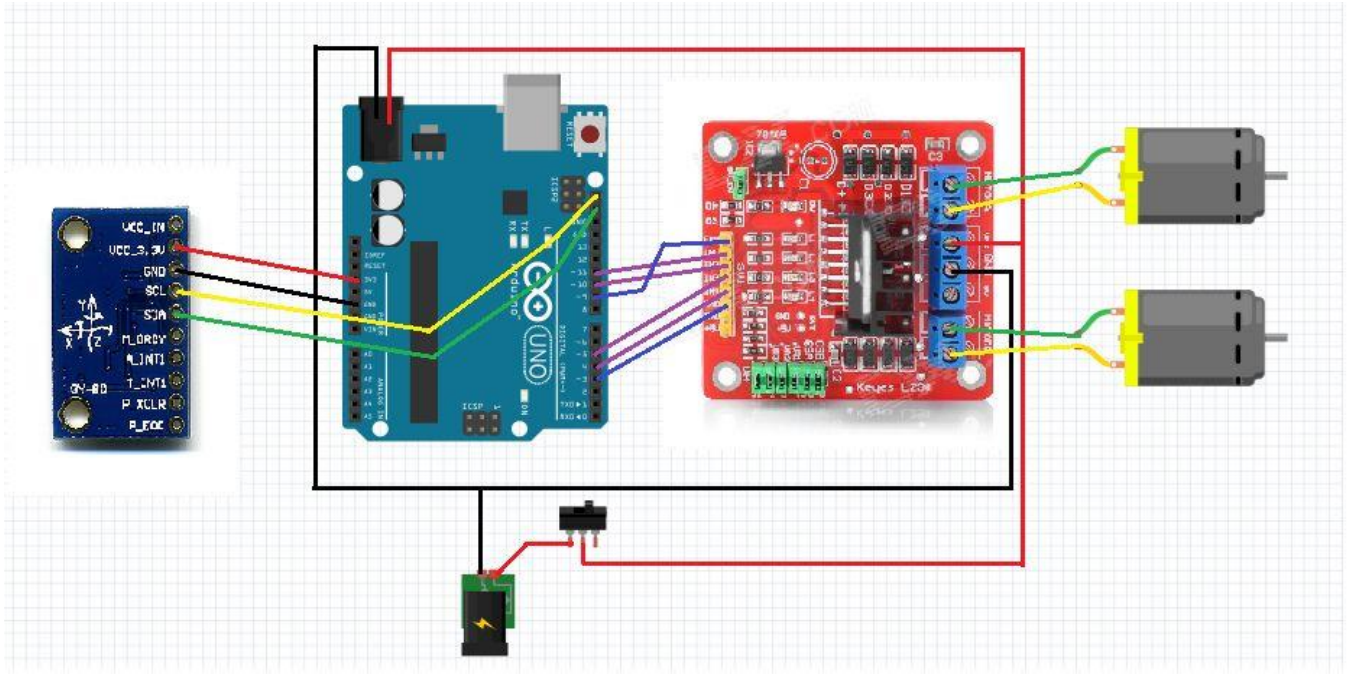


- L289N - The l298n motor driver is a Dual H-Bridge driver that can control 2 motors simultaneously, it uses PWM signals from the Arduino to control the direction and speed of the motors. L298N has a heat sink and hence can control on high currents too and run the motors efficiently.

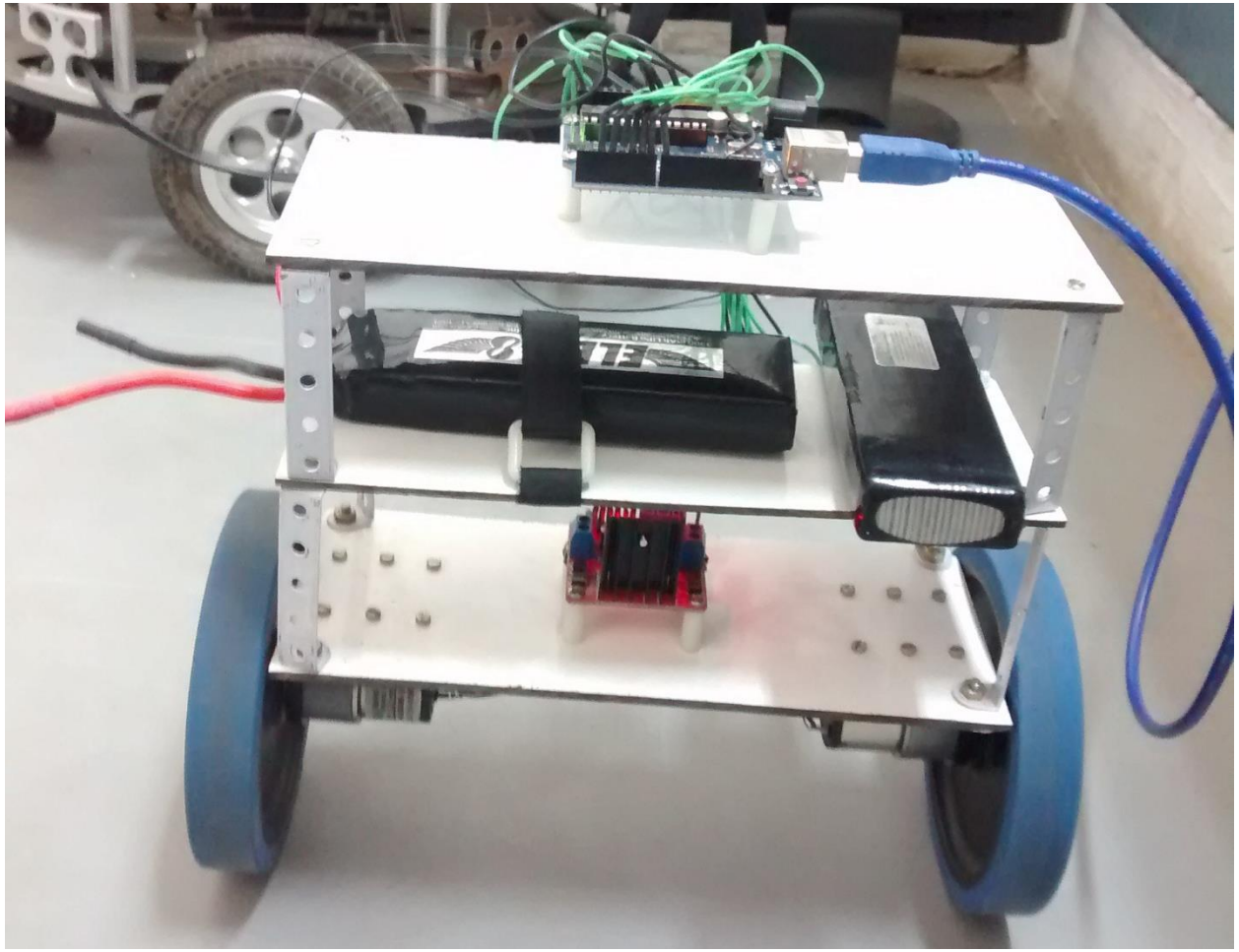


- High torque motors and high traction wheels - Parallax 7.2V Motors, Bracket and Wheel Kit is used in the project as the motors are high torque in nature and hence can make quick movements with the inclination in the system. The wheels are high traction wheels with a 12.4cm diameter.

# The Connection



# The Structure



# PID Controller

A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism (controller) commonly used in industrial control systems. A PID controller continuously calculates an error value as the difference between a desired set point and a measured process variable.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

Where, Kp accounts for current error value

Ki accounts for past error values

Kd accounts for possible future error values

Kp, Ki, and Kd values for our system turned out to be –

Kp – 70, Ki – 240, and Kd - 2

# The System

Take a reference start point

Repeat

- Take IMU reading and calculate tilt
- Send the tilt as input to PID controller
- Generate PWM signal from the controller
- Send the PWM value to the motor driver
- Move the motors

# Working:

# Conclusion

- The result was a complete balancing bot that autonomously balances itself
- The PID controller was implemented and resulted efficient
- The bot was constructed economically in comparison to other available bots

Future Aspects –

- The bot can be equipped with a user control for freely moving
- Controllers can be created using techniques like fuzzy theory and neural networks

# Thank You!