Simple Self Balancing Robot

The main aim of this project is to balance the two wheeled robot with sensor, motor and microcontroller by using PID algorithm.

**Components**

***Mechanical:***

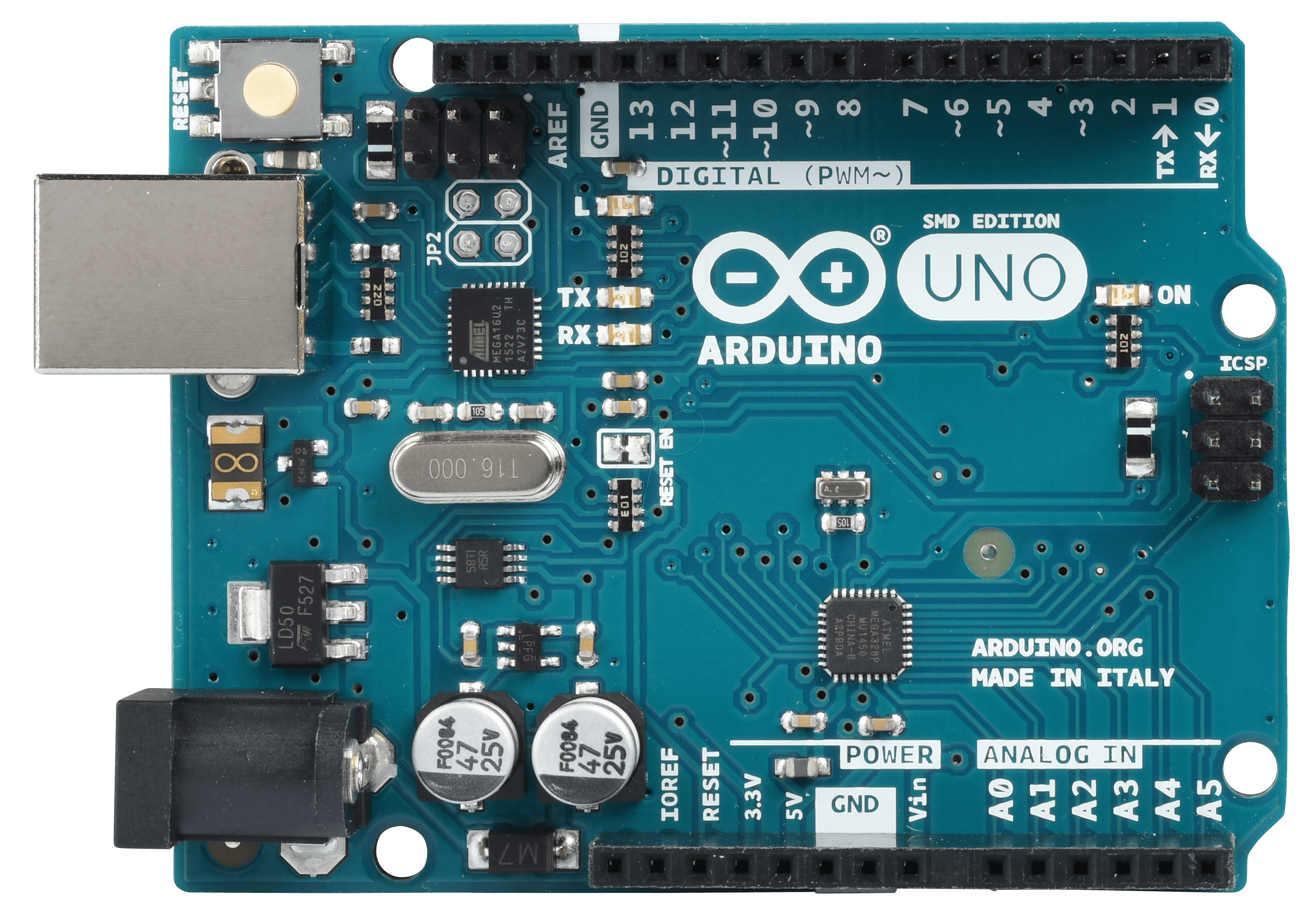
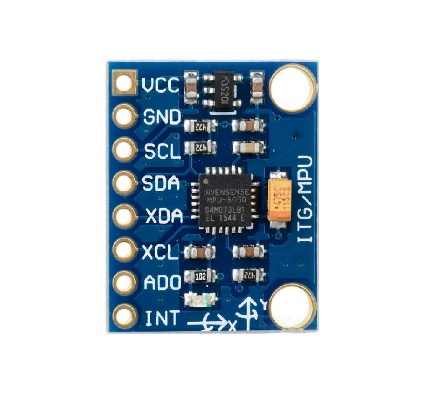
***Hylam Sheet*:** Hylam is a tough and versatile engineering material for electrical and mechanical use.

***Motor Mounting Bracket*:** It used to mount the motor securely.

***Bolts and Nuts***

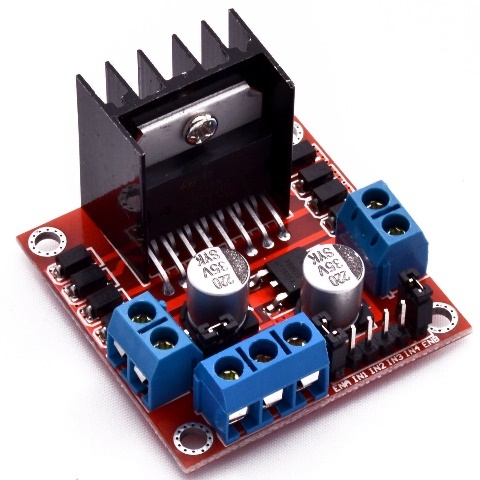
***Electrical:***

***Arduino Uno*:** It is microcontroller board based on the ATmega328P microcontroller.

***Gyro Module*:** It is a six-axis IMU sensor, which means that it gives six values as output: three values from the accelerometer and three from the gyroscope.

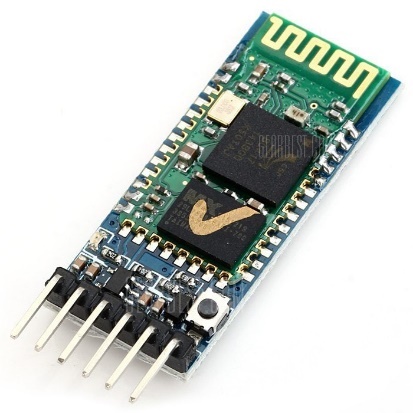
***L298N Driver*:** It is a motor driver which allows speed and direction control of two DC motors at the same time.

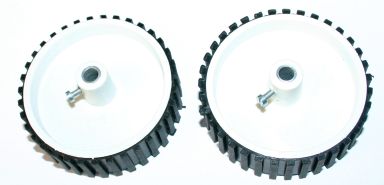
***DC-DC Converter*:** It is an electronic circuit that converts a source of direct current

from one voltage level to another.

***Lithium Ion Battery***with charging circuit or ***Power bank*.**

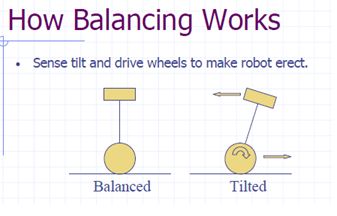
 

***HC05*:** It is a Bluetooth module used for remote communication.

***Motors with Wheels*.**

The ***Self Balancing Robot*** is essentially an inverted pendulum. It can be balanced better if the center of mass is higher relative to wheel axis. A higher center of mass means a higher mass moment of inertia, which corresponds to lower angular acceleration (It can fall slowly).



**Connections**

**Arduino MPU6050 Arduino L298N Arduino HC05**

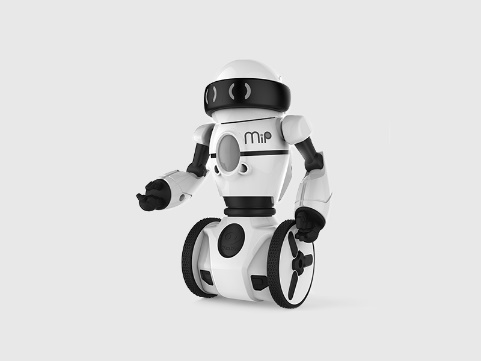
3.3v Vcc 6 IN1 5v Vcc

Gnd Gnd 7 IN2 Gnd Gnd

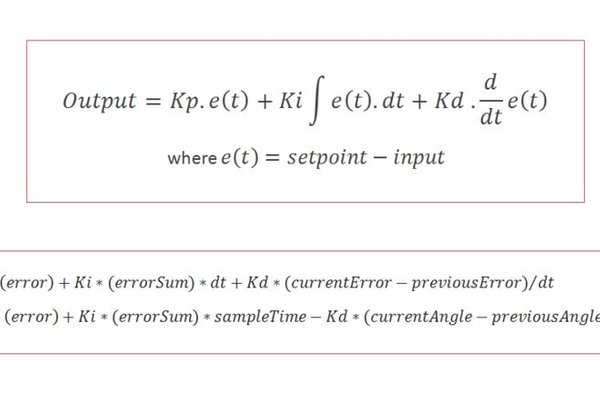
SDA/A4 SDA 5 IN3 RX TX

SCL/A5 SCL 4 IN4 TX RX

**Applications**

* Segway Robots can be used in such a way as to travel forward or backward. 
* They can be even used as vehicles by humans.
* They can be serve as a helper in the shop.
* Nowadays, a short distance can be travelled using self-balanced scooter.  
* Many motorcycle company invented the ride assistant which is self-balanced.

Using gyro module, the angle can be calculated. If there is any correction in angle in degrees, we can set the target angle to the desired angle. Otherwise set the target angle to zero. The error is the difference between the present angle and desired angle. The errorSum is equal to the sum of the errors. The Self Balancing robot needs fast processing of data to calculate the speed of the motor. Timer interrupts allow us to perform a task at very specifically timed intervals regardless of what else is going on in your code. So, we can use the timer interrupt, it can process the calculations repeatedly for every 20ms. The Bluetooth Module (HC05) is used to turn on and off the robot. It can be done by using app in Android. The App can send data to the bot and it can process related to the data.



**Theory**

In control theory, keeping some variable (in this case, the position of the robot) steady needs a special controller called a PID (proportional integral derivative). Each of these parameters has "gains", normally called Kp, Ki, and Kd. PID provides correction between the desired value (or input) and the actual value (or output). The difference between the input and the output is called "error". The PID controller reduces the error to the smallest value possible by continually adjusting the output. In our Arduino self-balancing robot, the input (which is the desired tilt, in degrees) is set by software. The MPU6050 reads the current tilt of the robot and feeds it to the PID algorithm, which performs calculations to control the motor and keep the robot in the upright position. PID requires that the gains Kp, Ki, and Kd values be "tuned" to optimal values. Engineers use software like MATLAB to compute these values automatically. Unfortunately, we can't use MATLAB in our case because it would further complicate the project. We will tune the PID values manually instead.

**Manual Tuning**

We should tune it manually to balance the robot. Here's how to do this:

* 1. Make Kp, Ki, and Kd equal to zero.
  2. Adjust Kp. Too little Kp will make the robot fall over, because there's not enough correction. Too much Kp will make the robot go back and forth wildly. A good enough Kp will make the robot go slightly back and forth (or oscillate a little).
  3. Once the Kp is set, adjust Kd. A good Kd value will lessen the oscillations until the robot is almost steady. Also, the right amount of Kd will keep the robot standing, even if pushed.
  4. Lastly, set the Ki. The robot will oscillate when turned on, even if the Kp and Kd are set, but will stabilize in time. The correct Ki value will shorten the time it takes for the robot to stabilize.

Hence the two wheeled robot is balanced by using PID algorithm with Gyro Module.