Real Time System Department
Fog Computing of Self Balancing Robot

By

(Shoriya Chauhan)

Under the supervision of
MSc. Alexandre Venito
Prof. Gerhard Fohler

(8th May 2020)

1 Motivation

Improvment for the next generation industry is the key idea to implement this project.The traditional robots have dedicated controller and resources to meet the real tme constraints.The main idea here is to be able to do complex funcitonality on fog node meeting the real time contraints, which can not be done via dedicated controller and resourses.Fog computing may provide local data processing along with real time communicatioon mechanism.

2 Project Flow

Here we are implemneting a self balancing robot running on esp32 and the main PID control loop to be implemented on the fog node while meeting the requirement of the robot to be balanced.
Flow of the project:

- To understand the communication protocol of the lego sensors
- Implement communication of lego sensors with Arduino
- Integrate the sensors togther to build self balaning bot
- Change from Arduino to esp32 for the wireless communication
- Structure the robot
- Get exact value of the PID
- Shift the PID function on fog node.

3 Literature Survey

Consider broomstick on the index finger, when trying to balance it we will have to move the finger in direction of the falling broomstick. Similarly the robot would fall either forward or backward which can be controlled by moving the robot either backward or forward. To make a balancing robot we need to control the center of gravity of the robot just at the pivot point.

3.1 Requirement for balancing the robot

- Direction the robot is falling
- Robot tilts

These information can be dedSelfBalancing Robotuced from the accelero-meter and gyroscope in a complimentary filter.

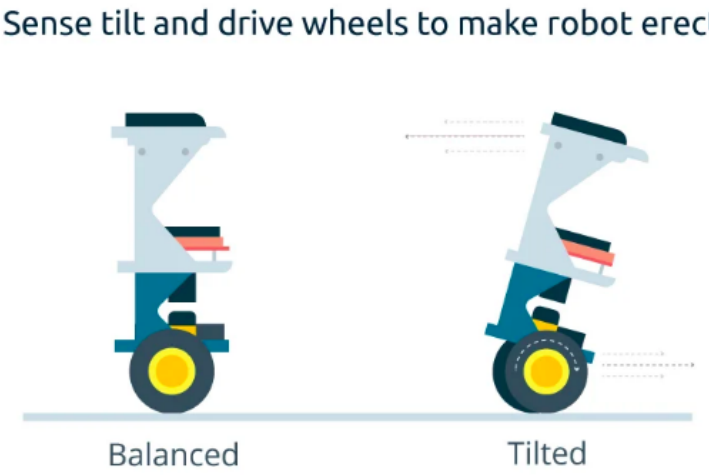

Figure 1: Tilt Angle and Balance

4 Hardware required

- Lego Motors
- Lego Gyroscopic Sensor (replaced it with MPU6050)
- MPU6050
- Ultrasonic Sensor
- Arduino Mega 2560
- L298N H bridge
- ESP32

Table 1: Sensors

Sensors	**Output**	**Unit**
Rotary Encoder	Angle	deg
Ultrasonic Sensor	Distance	cm
Gyro Sensor	Angular Velocity	deg/sec

4.1 Lego Motors

Lego smart motor has built in encoders that can be used to make the robot move around in a very precise manner. It has a resolution of 360 counts per revolution which means it counts 360 when the robot completes one wheel rotation.

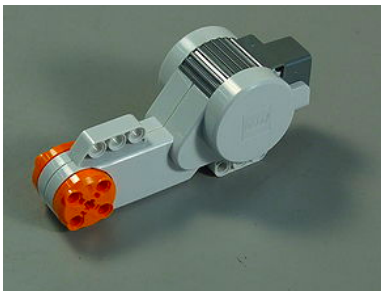

Figure 2: LEGO MOTOR

What are encoders used for? Encoders convert mechanical motion into electronic signal. By reading the encoder signals, the Arduino can figure out in which direction is the wheel turning and how many degrees.

4.2 Gyroscopic Sensor

Gyro offset and Gyro drift have major impact on balancing the robot. Gyro offset is the output when the gyro sensor does not rotate. In our case the gyro offset is 17. Gyro drift is time variation of gyro offset.

4.3 MPU 6050

The MPU 6050 is a 6 DOF (degrees of freedom) or a six-axis IMU sensor, which means that it gives six values as output: three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (micro electro mechanical systems) technology.

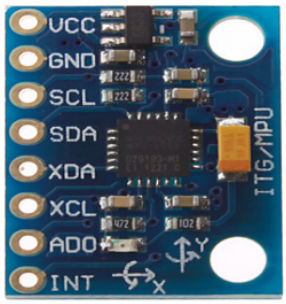

Figure 3: MPU6050

4.4 Ultrasonic Sensor

Ultrasonic sensor here is used to avoid obstacle. The robot would rotate itself when it senses an obstacle within 20 cm. It is done by rotating one wheel faster than the other.

Figure 4: Ultrasonic Sensor

4.5 L298N

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

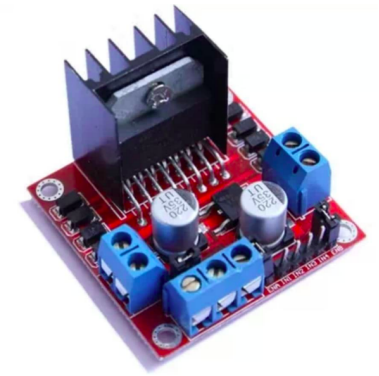

Figure 5: L298N H-Bridge

5 Control System

To keep the robot balanced, the motors must counteract the robot falling. This action requires feedback and correcting elements. The feedback element is the MPU6050 gyroscope + accelerometer, which gives both acceleration and rotation in all three axes. The Arduino uses this to know the current orientation of the robot. The correcting element is the motor and wheel combination.

5.1 Angle Measurement using MPU6050

Gyroscopic sensor was replaced by MPU 6050 as it has both gyro meter and the accelero-meter therefore, it is easily to calculated the angle of inclination accurately using the complimentary filter.

The accelerometer can measure the force of the gravity, and with that information we can obtain the angle of the robot, the problem of the accelerometer is that it can also measure the rest of the forces the vehicle is someted, so it has lot of error and noise. The gyroscope measure the angular velocity, so if we integrate this measure we can obtain the angle the robot is moved, the problem of this measure is that is not perfect and the integration has a deviation, that means that in short time the measure is so good, but for long time terms the angle will deviate much form the real angle.

Those problems can be resolved be the combination of both sensors, that's called sensor fusion, and there are a lot of methods to combine it.

5.2 Complimentary Filter

Complimentary filter used to enhance the quality of accelerometer and gyroscope to give a define angle of inclination.

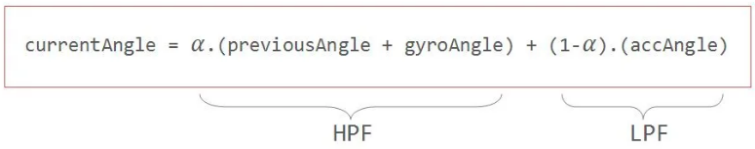

Figure 6: Complimentary Filter

The measurement from accelerometer gets affected by sudden horizontal movements and the measurement from gyroscope gradually drifts away from actual value. In other words, the accelerometer reading gets affected by short duration signals and the gyroscope reading by long duration signals. These readings are, in a way, complementary to each other. Combine them both using a Complementary Filter and we get a stable, accurate measurement of the angle. The complementary filter is essentially a high pass filter acting on the gyroscope and a low pass filter acting on the accelerometer to filter out the drift and noise from the measurement.

currentAngle = 0.9934 * (previousAngle + gyroAngle) + 0.0066 * (accAngle)

0.9934 and 0.0066 are filter coefficients for a filter time constant of 0.75s. The low pass filter allows any signal longer than this duration to pass through it and the high pass filter allows any signal shorter than this duration to pass through. The response of the filter can be tweaked by picking the correct time constant. Lowering the time constant will allow more horizontal acceleration to pass through.

5.3 PID Controller

PID stands for Proportional, Integral and Derivative. In this algorithm, the error signal received is the input. And the following equation is applied onto the error signal

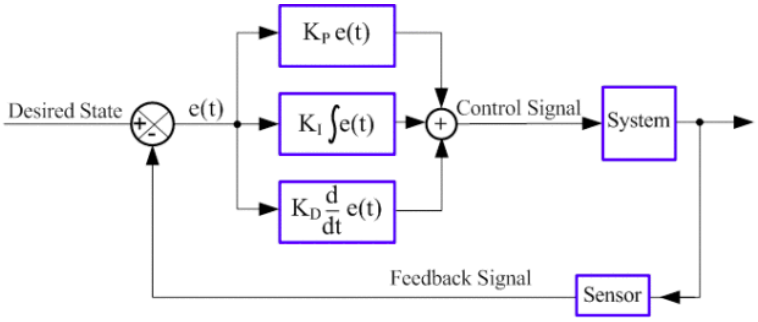

Figure 7: PID Controller

The proportional term (gain) makes a change to the output that is proportional to the current error value. Larger values typically mean faster response since the larger the error, the larger the Proportional term correction signal. However, an excessively large proportional gain will lead to instability and oscillation. The Integral term is proportional to the amount of time the error is present. The integral term accelerates the movement of the process towards the set value and eliminates any residual steady-state error that occurs with a proportional only controller. The contribution from the integral term is dependent both on the magnitude of the error and on the duration of the error. With the derivative term, the controller output is proportional to the rate of change of the measurement or error. The derivative term slows the rate of change of the controller output, most notably near the controller set value. It therefore reduces the magnitude of the overshoot produced by the integral component and improves stability. Larger Kd values decrease overshoot, but slows down transient response and may lead to instability from signal noise amplification from differentiation of the error.

TUNING a PID controller

Tuning: A good Control System will have low rise time, settling time, peak overshoot and steady state error. Therefore, the Kp, Kd, Ki need to be finely tuned to adjust the contribution of the above factors in order to acquire a good Control System.

- SetKp=Ki=Kd= 0.
- Adjust Kp until the system remains in balance, but rapidly oscillates around equilibrium.
- Adjust Kd until the system reaches steady state.•If there is a steady state error, tune Ki. However, be warned that if the system is not actually reaching equilibrium, the integral term can dominate the controller and otherwise have negative effects on your controller.

6 Integrating ESP32

ESP32 is higly integrated solution for IoT Applications with bluetooth, WiFi. Esp32 uses I2C commmunication to communicate with mpu6050.
Pin Connections for self balancing robot

- MPU connection:
  SDA 21 – Pin3
  SCL 22 – Pin4

- Motor Connection

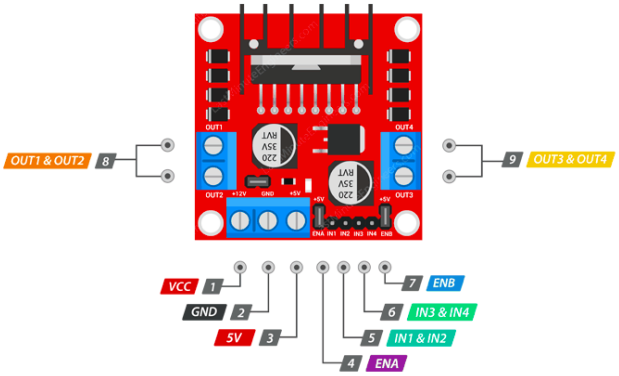

Figure 8: Motor Connector

VCC – Battery 9V
GND – GND Battery

5V – 5V ESP32

ENA – 12
ENB – 14
IN1 – 26
IN2 – 27
IN3 – 18
IN4 – 19

OUT1 – Motor Connector AN IN
OUT2 – Motor Connector GND
OUT3 – Motor Connector AN IN
OUT4 – Motor Connector GND

7 Structure Modelling

The struture is so built that it tries to divides it weight equally and the center of mass is maintained. The base layer hols the L298N motor control and the esp32 and the top is where the mpu6050 is standing verticially. And integrating two batteries one for esp32 and other for L298N motor controller.

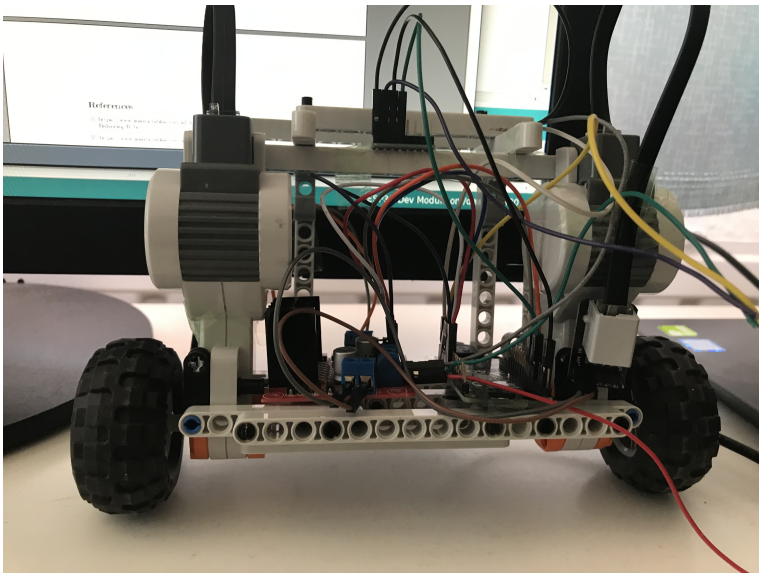

Figure 9: Structure

The struture is able to balance it self with the defined values of PID when the PId is implemented on the controller.

8 Fog Node

Here we using Python socket as our server node (fog node) to implement PID module. The esp32 functions to supply the current angle and the server replies back with the change in motor to be stable. With fog computing technology, all the processing happens on devices physically closer to where the data is collected, instead of sending data to the cloud.

8.1 Exeution time

The first the foremost step was to find the maximum delay that the robot can withstand to still be balanced.This is done by using the millis() or micros() function in Arduino to find the approximate execution time of the program in loop. We find that it takes around 1ms for the code to run and the robot to still be balanced. To this we added a delay funtion to find the minimum number of gyroscope value required in a time frame for it to be balanced. A delay of 20ms was fair enough for the bot to still be balanced with Kp value to be 280.

```
void loop() {
   startTime =millis();
      //Code for Angle Calculations
      //Code for PID Funtion
   delay(20);
   Serial.println(millis()-startTime);

}
```

And after shifting the PID code to the python server the total execution time on ESP32 was re-calculated. It takes around 2050 ms to get the motor speed and the turning direction.

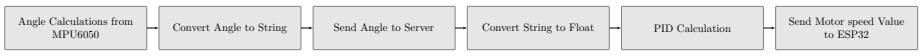

Figure 10: Loop Execution

We have to reduce the the communication time from the esp32 to the fog node and back. Here we introduce OPC-UA (Open Platform Communication United Architecture) which is a data exchange standard for next generation industrial communication

9 Open Platform Communication United Architecture-OPC-UA

OPC-UA is a mechanism for transferring information between server and client which is secure,open and reliable. Here, we will implement the OPC-UA fog server for calculating the PID function and the esp32 will act like an OPC-UA client.

9.1 Steps invloved in configuration of OPC UA

- Server Initial Configuration
- Local Discovery Server LDS - Registration
- Client Configuration
- Local Discovery Server LDS - Discovery
- Connection configuration

9.2 Steps invloved in configuration of OPC UA

10 Problems faced

Problem : Lego Ultrasonic sensor does use simple I2C communication
Solution : Used Separate library for re-start frame I2C communication with arduino

Problem : Either the MPU6050 worked or the Motor- Due to separate ground
Solution : It is necessary when there are groups of circuit need to interface with each other and ground is the reference.

Problem : Shifting code from arduino to ESP32
Solution : ESP32 is not compatible with the libraries for MPU6050 and I2C used in arduino. Hence the code had to be modified for specific integration of gyroscope and accelerometer.

Problem : Getting exact value of the Kp,Ki and Kd Solution : Have to change the Target angle as well as the Kp Ki Kd value on trial and error to get the exact value for the balance

References

[1] https://www.instructables.com/id/Arduino-Self-Balancing-Robot-1/self-Balancing Robot

[2] https://www.instructables.com/id/Self-Balancing-Robot-1/

[3] https://osoyoo.com/2018/08/08/self-balancing-robot-pid-control/

[4] https://www.cs.cmu.edu/ 16311

[5] https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/