**Two-wheel Self-balancing Robot**

**Introduction and motivation**

The main objective of this project is to design and implement an efficient self-balancing algorithm for the two-wheel self-balancing robot using Arduino UNO R3. The two-wheel self-balancing robots have practical use like self-balancing scooters that can bring convenience for people. Due to our interests in self-balancing inverted pendulum and its practical use, we want to achieve a simple implementation of self-balancing robot. In market, many high-end self-balancing products including the self-balancing scooter – the Segway, use either an accelerometer or a gyroscope or even a combination of both to achieve a vertical balance. To combine multiple sensor readings into a single usable value, there are all kinds of software implementations of filters including the Kalman filter and the complementary filter. This project would focus on how to utilize two sensors to get more accurate data and implement useful control algorithm by incorporating a physical model of the robot, IMU, encoders and two motors.

**Learning Outcomes**

1. Physical Model

The three axis of the self-balancing robot are forward direction(x-axis), upward direction (y-axis) and line between two wheels (z-axis). The robot is able to rotate around the z-axis (pitch) by an angle $\theta_p$, with a corresponding angular velocity $\omega_p$. The linear movement of the robot is characterized by the position x and the velocity v. Also, the robot can rotate about the y axis (yaw) by an angle $\alpha$ with a corresponding angular velocity $\dot{\alpha}$ to change the moving direction.

2. Control Algorithm Design

Since the core of this robot is to accurately control the pitch angle $\theta_p$, sensors and corresponding algorithms need be optimized to measure the pitch angle and the rate of change. Therefore, the robot is equipped with MPU 6050 module, which contains an accelerometer and a gyroscope, to obtain the pitch angle. Together with encoders and PID controller, the robot has the ability to move in a vertically balanced state.

2.1 Accelerometer only

Accelerations in the y (upward) and x (forward) directions provide a way to measure the direction of gravity so that it can help calculate the pitch angle according to formula $\theta_p = \tan^{-1}(\frac{a_x}{a_y})$. But when the robot is accelerating in a particular direction, the accelerometer is unable to distinguish between that and the acceleration provided through Earth's gravitational pull. So, the pitch angle cannot be measured correctly and it's not ideal to use accelerometer only for the vertical balance of the robot.

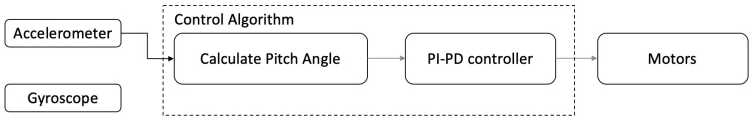


Figure 1. Control algorithm with accelerometer only

2.2 Gyroscope only

A gyroscope measures the angular velocity and the pitch angle can be obtained according to equation $\theta_p(i) = \theta_p(i-1) + \frac{1}{6}(val_{i-3} + 2val_{i-2} + 2val_{i-1} + val_i)$. However, the drawback of the gyroscope is that there exists a small DC bias and low-frequency noise. If the self-balancing robot depends only on the gyroscope, it would be vertical for only a few seconds and eventually fall down because of the drift of zero point.

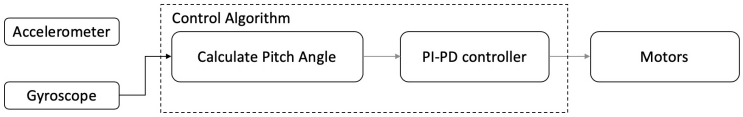


Figure 2. Control algorithm with gyroscope only

2.3 Encoders only

The robot is equipped with two encoders on left and right wheels. They can help measure the distance travelled by the robot. However, if accelerometer or gyroscope is not used to measure the pitch angle, the robot can only move a certain distance without keeping balance.

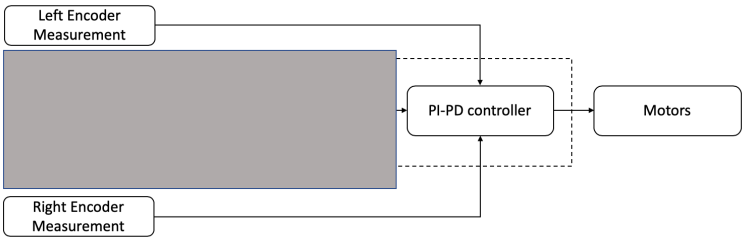


Figure 3. Control algorithm with encoders only

2.4 Filters on outputs from accelerometer and gyroscope

A more accurate measurement of the pitch angle can be obtained by combining the outputs from the accelerometer and the gyroscope through the use of sensor fusion algorithm. A simple algorithm called the complementary filter was adopted to use a small component of the accelerometer output and a large proportion of the gyroscope to offset the noise of the accelerometer and low frequency noise and the drift errors of the gyroscope according to the equation $\theta = K\theta_g + (1-K) * \theta_a$. This algorithm did help obtain more accurate pitch angle. Another algorithm called Kalman filter was also tested to combine the outputs from the accelerometer and the gyroscope. It is an algorithm which uses a series of measurements observed over time, here they are from an accelerometer and a gyroscope. These measurements that contain noise will contribute to the error of the measurement. But Kalman filter will try to estimate the state of the system based on the current and previous states, and tend to be more precise that than the measurements alone. More specifically, it's a set of mathematical equations that provides a way to fuse multiple sensor readings into one by three steps: predict — based on previous knowledge of a robot position and kinematic equations, we predict what should be the position of vehicle after time t+1; measurement (accelerometer and gyroscope) — get readings from sensor regarding position of robot and compare it with prediction; update — update our knowledge about position and state of the robot based on our prediction and sensor readings. Details are not covered here due to page limitation but can be found in my codes.

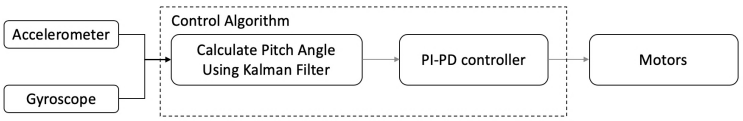


Figure 4. Control algorithm with accelerometer and gyroscope

The conclusion is that these two methods can both minimize the high frequency noise of accelerometer and low frequency noise of gyroscope and then fuses them to give the best accurate pitch angle. We chose the more complicated one with better performance, Kalman filter.

2.5 PI-PD controller

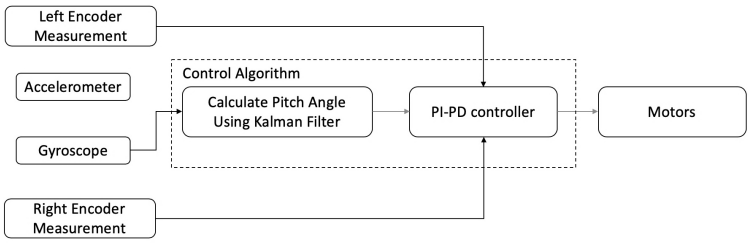


Figure 5. Control algorithm with encoders, accelerometer and gyroscope

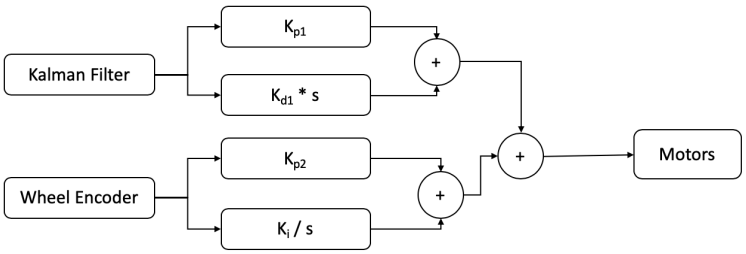


Figure 6. PI-PD control algorithm for encoders, accelerometer and gyroscope

In summary, outputs from encoder, accelerometer and gyroscope need to be combined together for the accurate movement of the robot (rely on encoders) in a balanced state (rely on accelerometer and gyroscope). The choice of proportional, integral and derivative terms on outputs from these sensors is another question. According to Hau-Shiue Juang ‘s paper, PI-PD controller as shown in figure 6 based on filter output and encoders output is better than PID controller based on filter output only. So we apply P and D terms only to pitch angle output from Kalman filter for the balance of the robot and apply P and D terms only to the output of encoders for the correct moving distance of the robot. Although the process of tuning the P, I, D parameters takes a lot of time, the performance of the self-balancing robot is good by applying the PI-PD strategy.

3 Some detailed technical difficulties and solutions during implementation

3.1 The hardware is not always reliable, so many test files to examine each module were written. When weird things happened, these test files made great contributions to find the specific error.

3.2 There is no screen to show data timely, so serial communication is used to view data and some LED lights are adopted for testing and debugging procedure.

3.3 There might be some offsets of the data measured by MPU6050 due to the placement of the module, so certain value was added to eliminate them by doing quantitative experiments.

3.4 The process of tuning PID parameters is too bothering by changing code from the computer. So, potentiometers are used to tune the parameters.

3.5 Interrupts raised by sensors and interrupt to control the robot fired by timer may conflict with each other, so the frequency of the timer is well-designed for less interrupt conflicts and more real-time reponse. The frequency cannot be too high (more interrupt conflicts) or too low (the robot would fall down for late response). Details of the scheduler can be found in Xiang Zhang's report.

**Product**

**General description**

The two-wheel self-balancing robot is controlled by the UNO R3 processor. Both an accelerometer and a gyroscope are connected to the micro-controller to measure the pitch angle. Kalman filter is used to achieve the fusion of sensor data into a single usable value and offset the high-frequency noise of the accelerometer, the low-frequency noise of the gyroscope and the drift errors of the gyroscope. To keep a vertical balance of the robot and make it move according to the instructions, PI-PD controller is applied to control the motors accordingly based on the outputs from Kalman filter and encoders. With the assistance of Bluetooth module and ultrasonic module, the robot can be controlled by the smartphone and achieve many modes like "object-following" and "obstacle avoidance".

**Picture**

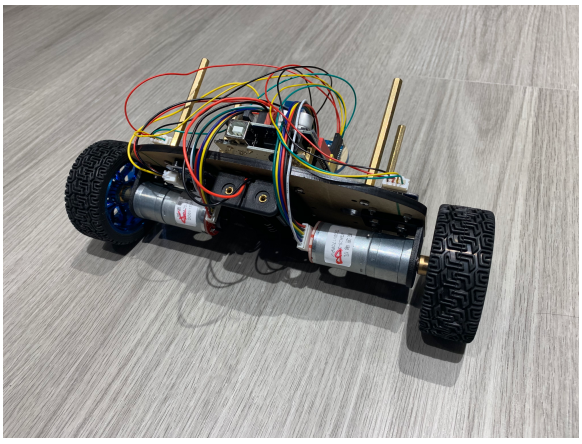

Figure 7. Two-wheel self-balancing robot

**Three modes**

1. Remote control mode via bluetooth.
2. Obstacle avoidance mode by utilizing ultrasonic module.
3. Object-following mode by utilizing ultrasonic module.

**Future work**

1. Due to the shortage of time and equipment, most of the analysis are qualitative. For the better performance of the robot, more quantitative experiments need to be done.
2. Increase the robustness of the robot. After enough quantitative experiments, the self-balancing robot will be more robust (like self-adapted to various surface conditions) and can be controlled more accurately. Then a self-controlled robot will be realizable by utilizing ultrasonic module and camera.

**References**

1. Hau-Shiue Juang, Kai-Yew Lurrr. (2013). Design and control of a two-wheel self-balancing robot using the arduino microcontroller board. 2013 10th IEEE International Conference on Control and Automation (ICCA), 634 - 639. doi: 10.1109/ICCA.2013.6565146.
https://ieeexplore.ieee.org/abstract/document/6565146

2. N. Yu, Y. Li, X. Ruan and C. Wang, "Research on attitude estimation of small self-balancing two-wheeled robot," Proceedings of the 32nd Chinese Control Conference, Xi'an, 2013, pp. 5872-5876.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6640466&isnumber=6639389

3. D. Gong and X. Li, "Dynamics modeling and controller design for a self-balancing unicycle robot," Proceedings of the 32nd Chinese Control Conference, Xi'an, 2013, pp. 3205-3209.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6639973&isnumber=6639389

4. Midhun_s. Arduino Self-Balancing Robot
https://www.instructables.com/id/Arduino-Self-Balancing-Robot-1/

5. Lauszus A practical approach to Kalman filter and how to implement it
http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/

6. Stephen Swift. Self balancing robot: Project 1116 of Engineering Physics Project Laboratory, The University of British Columbia.
https://www.researchgate.net/publication/265227587_SELF_BALANCING_ROBOT#pf18