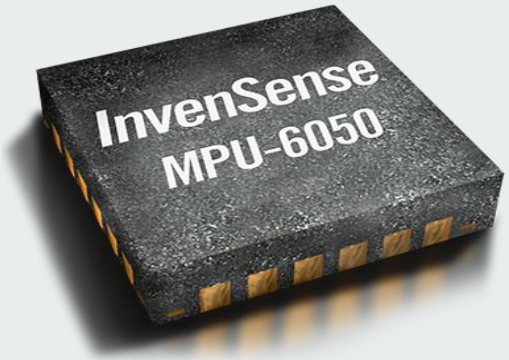
Cảm biến MPU-6050

Cảm biến vận tốc góc và gia tốc

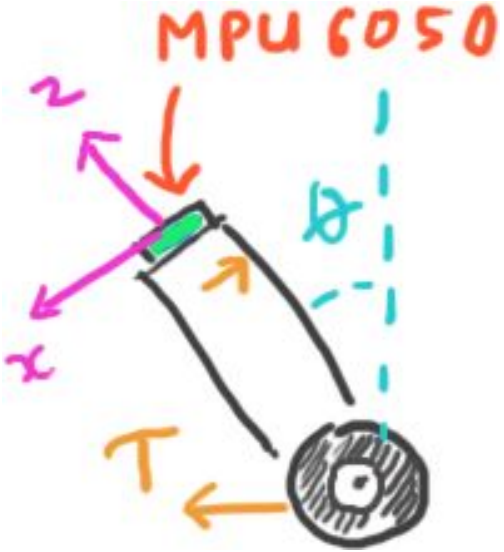
Người thực hiện : Đoàn Hồng Trung
Huỳnh Tấn Bảo
Đặng Hữu Tiến
Nguyễn Văn Bình

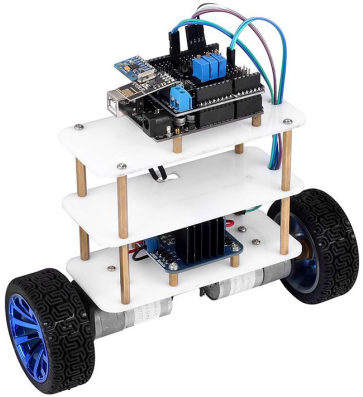


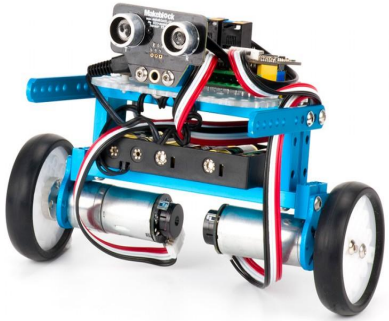
Bộ môn : Hệ thống cơ điện tử
Giảng viên : TS. Lê Hoài Nam

Tại sao phải quan tâm cảm biến này ?

- Vì nó là một phần quan trọng trong việc xây dựng robot 2 bánh tự cân bằng sắp tới


MPU 6050
z
x
T

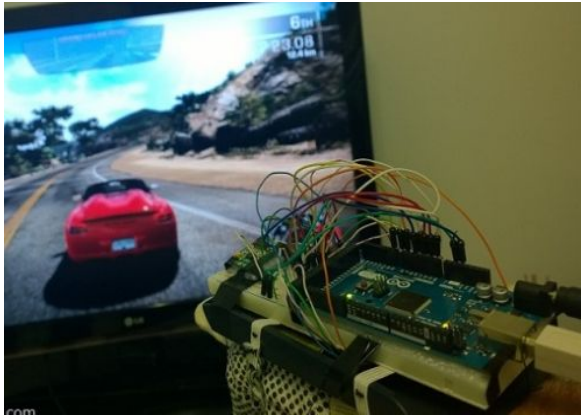





MiP

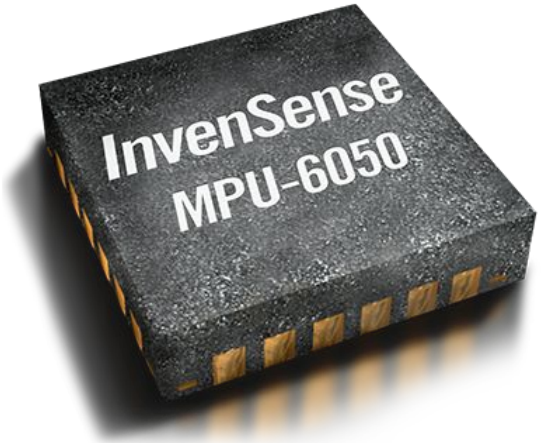
Ứng dụng

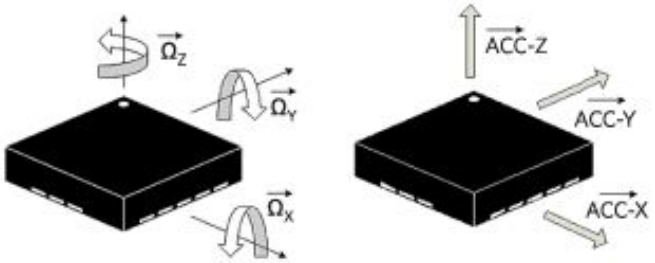





Những thứ đời thường hằng ngày như máy bay, ô tô, xe máy, ... cũng tích hợp những cảm biến tương tự như MPU6050

Chip MPU-6050





Tích hợp 2 hệ vi cơ điện tử :

- Gia tốc kế
- Con quay hồi chuyển

Có 6 bậc tự do

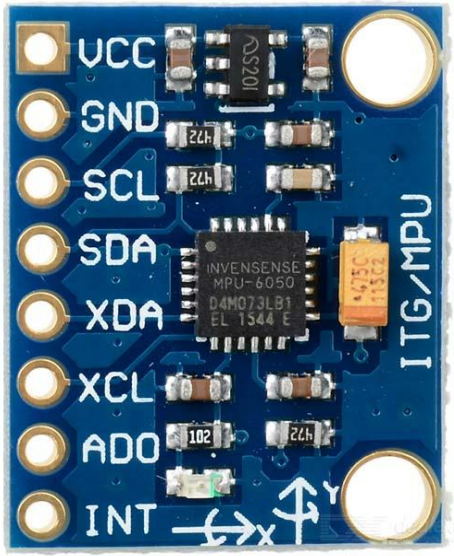
Giới hạn đo :

- Gia tốc : ±2g, ±4g, ±8g and ±16g
- Vận tốc góc : ±250, ±500, ±1000, ±2000°/sec

Ngoài ra còn có :

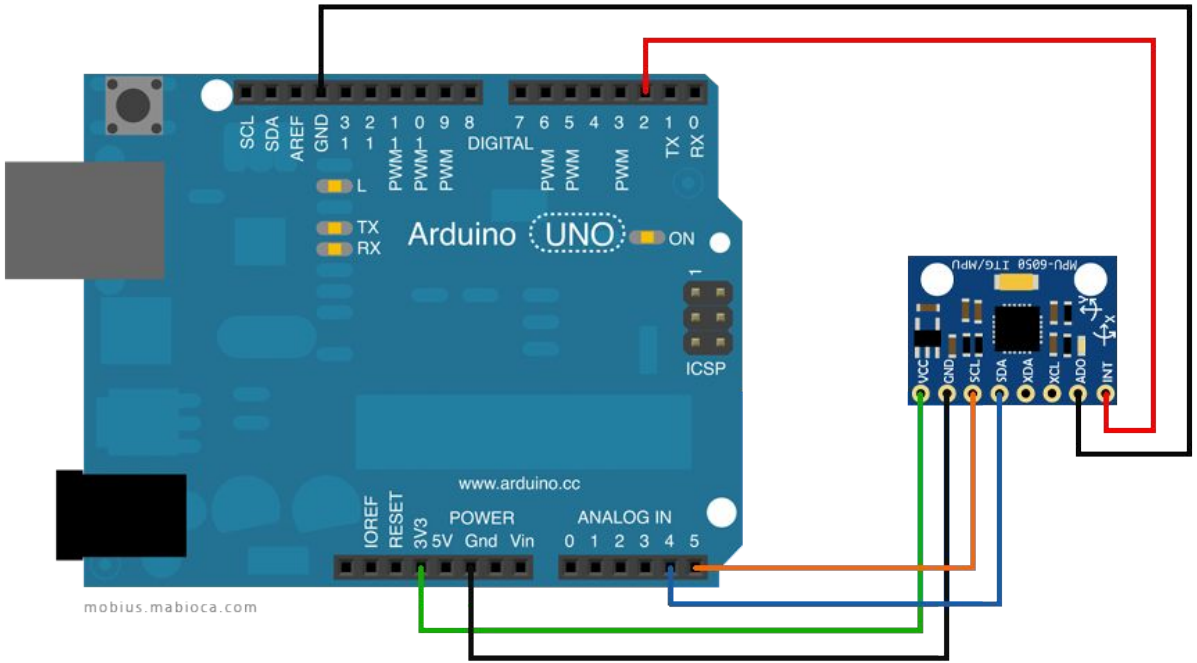
+ Bộ xử lý chuyển động số → Truy cập đến các giá trị cảm biến, sử dụng thuật toán từ vi điều khiển bên ngoài để tính toán, có quyền sử dụng bộ đệm và 1 chân ngắt ngoài.
+ Module I2C → Giao tiếp với bên ngoài
+ Cảm biến nhiệt độ → Tính sai số do nhiệt độ
+ Bộ nhớ đệm FIFO → Tăng thời gian lấy dữ liêu
+ Bộ lọc có thể lập trình → Bộ lọc cho cảm biến gia tốc, gyro, và nhiệt độ
+ ...

Bo mạch GY521



- Rẻ
- Dễ mua.
- Sử dụng thuận tiện

Kết nối arduino



- Nguồn: 5V, 3.3V
- SDA: 4
- SCL: 5
- INT: 2 hoặc 3

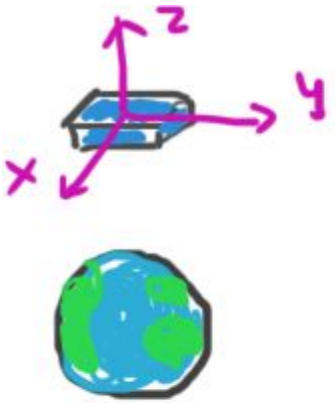
Giá trị đọc được:

- Vận tốc góc theo trục x, y và z
- Gia tốc theo trục x, y và z
- Nhiệt độ bên trong chip

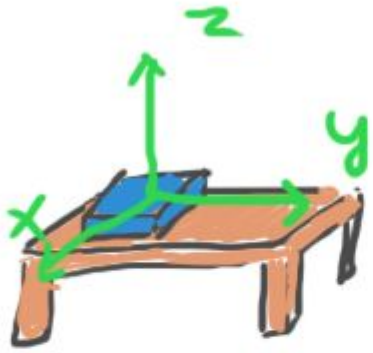
Ví dụ :

```
AcX = 9160  | AcY = -9556 | AcZ = -6648   | Tmp = 26.37 | GyX = 9077  | GyY = 7769    | GyZ = -28687
AcX = 13564| AcY = -2636 | AcZ = -9988   | Tmp = 26.55 | GyX = 9708  | GyY = 23384  | GyZ = -7700
AcX = 3840  | AcY = -1224 | AcZ = -10094 | Tmp = 26.41 | GyX = -2320 | GyY = 17385  | GyZ = -7
AcX = -1724 | AcY = 368    | AcZ = 10428  | Tmp = 26.46 | GyX = 3050  | GyY = -11115 | GyZ = -1357
AcX = 1724  | AcY = -752   | AcZ = 11888  | Tmp = 26.41 | GyX = -101   | GyY = -303     | GyZ = -133
```

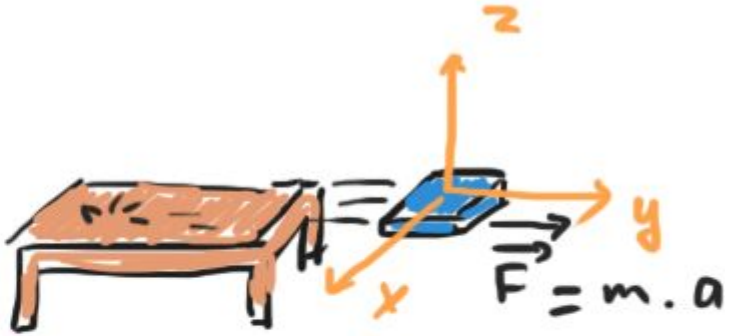
Bản chất giá trị theo trục x, y, z



x = 0
y = 0
z = 0

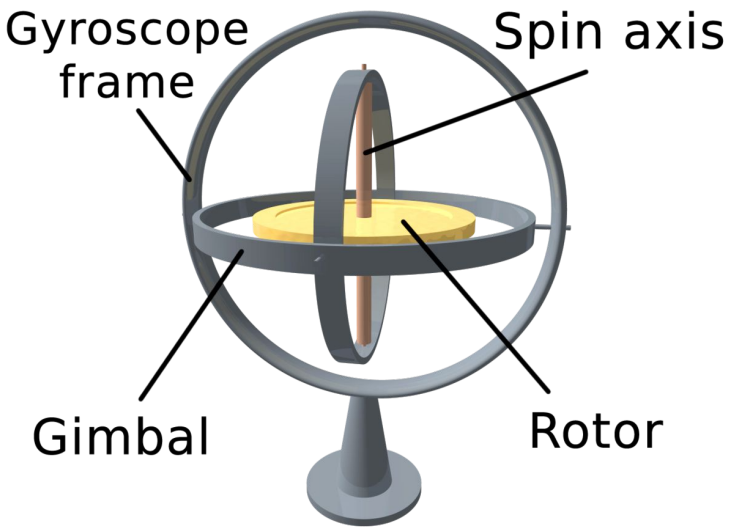


x = 0
y = 0
z = 9,8



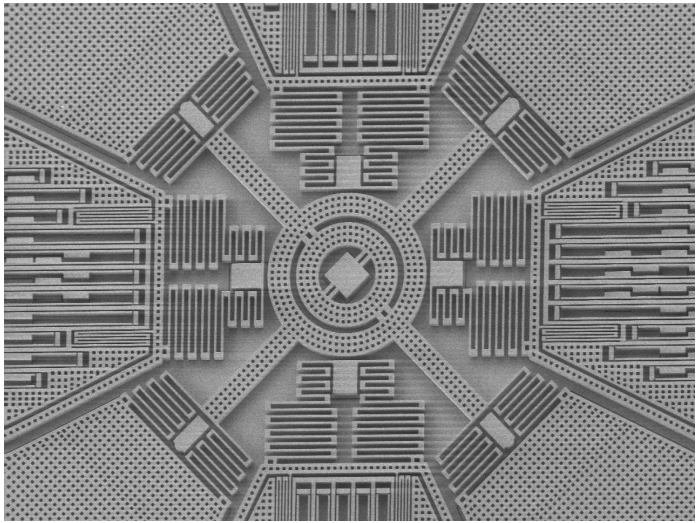
x = 0
y = a
z = 9,8

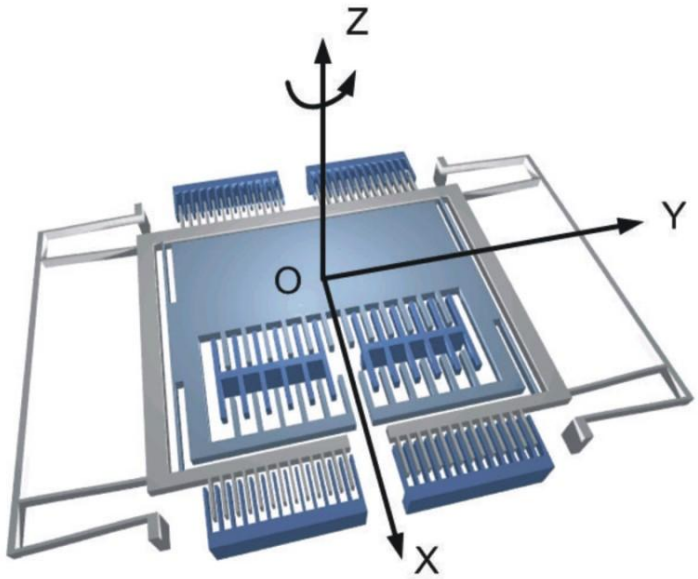
Con quay hồi chuyển là gì ?



- Thiết bị đo đạc và duy trì phương hướng
- Con quay cơ học: khi đĩa quay, hướng của trục không đổi dù hướng khung thay đổi
- Ứng dụng từ xe đạp → tên lửa

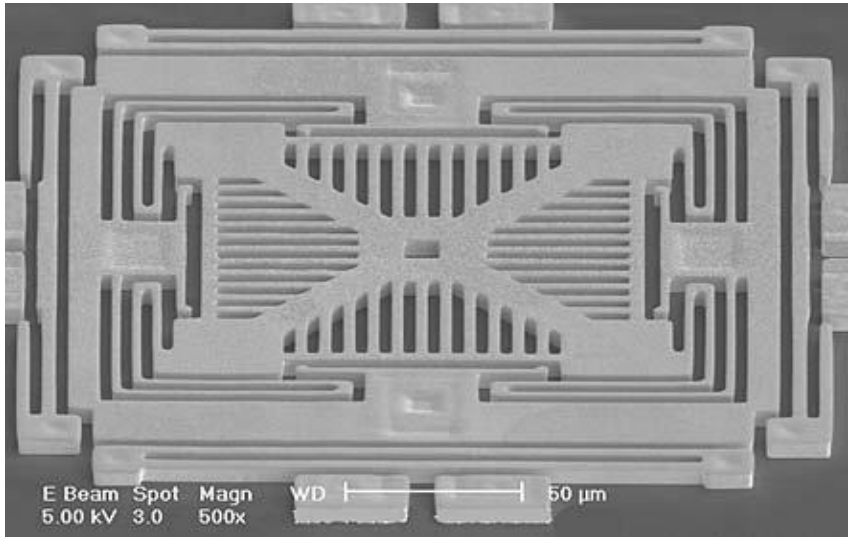
Con quay hồi chuyển vi cơ điện tử

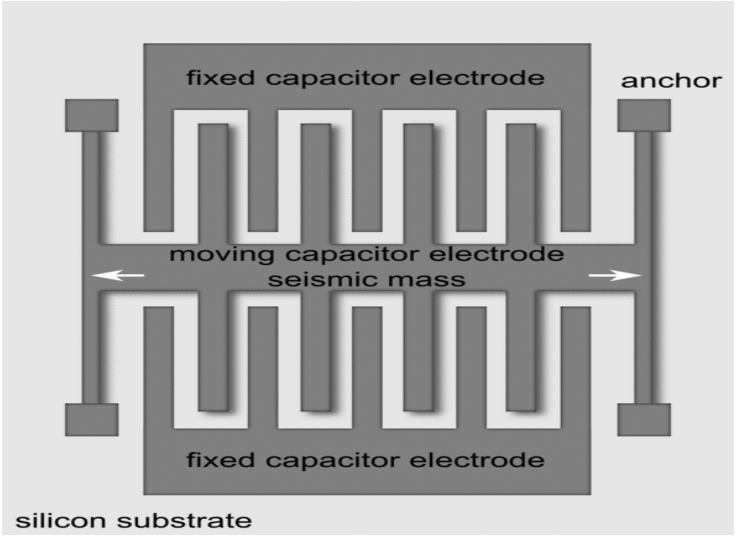




Cực kì nhạy và nhiễu

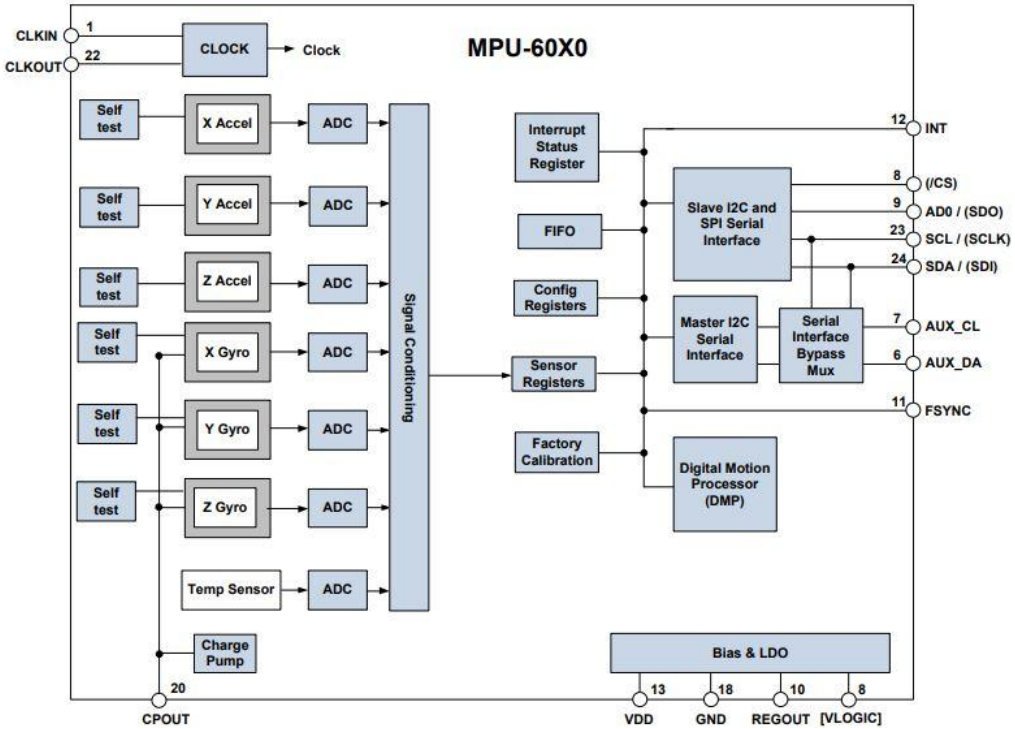
Gia tốc kế vi cơ điện tử





Giá trị đọc được dưới dạng điện tương tự

Sơ đồ khối MPU-6050



- Cảm biến con quay hồi huyển, gia tốc kế, nhiệt độ đọc vào tín hiệu analog cùng 1 lúc → tín hiệu số → bộ rẻ nhánh tín hiệu số → thanh ghi.
- Bộ giao tiếp I2C chính cho phép truy cập đến từng thanh ghi → giá trị gốc
- Bộ xử lý chuyển động số, FIFO, ngắt → giá trị sau lọc được xử lý bên trong MPU-6050

Các bước lập trình

- Kiểm tra kết nối vi điều khiển với MPU6050 trên đường truyền I2C → Gửi tín hiệu đánh thức MPU6050 đang trạng thái ngủ → Cài đặt cấu hình → Gửi yêu cầu lấy dữ liệu ở các thanh ghi lưu trữ giá trị gốc của cảm biến → Đọc giá trị trên đường truyền I2C
- Có 2 cách để có giá trị ổn định:
  - Đọc giá trị gốc từ cảm biến về vi điều khiển. Sử dụng các bộ lọc như trung bình, thông thấp, thông cao, bù, kalman, … để có giá trị ổn định.
  - Điều khiển bộ lọc thông thấp bên trong cảm biến. Khi tính toán xong giá trị thì sẽ có một ngắt xảy ra.

Sử dụng bộ lọc bên trong cảm biến

Ưu điểm : Tiết kiệm thời gian tính toán cho vi điều khiển. Dễ sử dụng hơn khi dùng thư viện arduino.

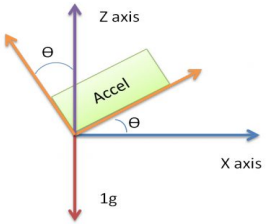
Nhược điểm : Không hẳn chính xác. Tốn 1 chân ngắt ngoài. Lập trình rất khó khi dùng vi điều khiển không có thư viện. Thường xảy ra hiện tượng treo ngắt ngoài.

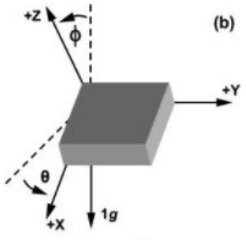
Sử dụng bộ lọc bên trong vi điều khiển

- Bộ lọc thông thấp : Dễ hiểu, dễ sử dụng
- Bộ lọc bù : Dễ sử dụng, chính xác hơn bộ lọc thông thấp
- Bộ lọc Kalman : Khó nhưng loại bỏ nhiễu như một điều thần kì.

Từ giá trị gốc ra góc nghiêng

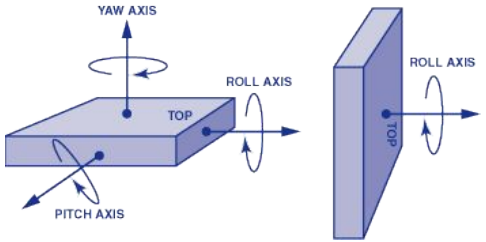
- Có thể tính ra được góc nghiêng của cảm biến từ 1 trong 2 cảm biến gyro và accel
- Độ tin tưởng của 2 cảm biến lại phụ thuộc vào trạng thái chuyển động của cảm biến
- Gia tốc/vận tốc góc = (giá trị gốc – giá trị bù) * hệ số tỉ lệ;
- Hệ số tỉ lệ = giới hạn đo được / giới hạn lưu trữ
- Từ gia tốc (trục y nằm ngang) : góc hiện tại = arctan (-AcX / AcZ) rad
- Từ vận tốc góc (quanh trục y) : góc hiện tại = góc trước đó + Gy * dt độ





$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$



Ví dụ :

- Giới hạn đo được của gia tốc = 2g (cài đặt cảm biến lúc lập trình)
- Giới hạn lưu trữ trong cảm biến = 32767 (bộ lưu trữ 16 bit)
- Giá trị gốc = 1944, giá trị bù = 0

=> Gia tốc = 1944 * (2 / 32767) = 0.118 g

- Gia tốc theo trục X = 0.118g
- Gia tốc theo trục Z = 0.876g
- Chọn Y nằm ngang

=> Góc hiện tại = arctan (-0.118 / 0.876) * 180 / pi = - 15 độ

- Giới hạn vận tốc góc = 250 d/s
- Giá trị gốc = -34

=> Vận tốc góc = -34 * (250 / 32767) = -0.26 độ/giây

- Thời gian giữa 2 lần đọc giá trị từ cảm biến 0.01 s
- Góc trước đó 8 độ => góc hiện tại = 8 + (- 0.26) = 7.74 độ

Tính góc nghiêng sử dụng bộ lọc

- Bộ lọc thông thấp :

  Góc hiện tại = (0.8)*(góc trước đó) + (1 - 0.8)*(gia tốc)

- Bộ lọc bù :

  Góc hiện tại = (0.966)*(góc trước đó + vận tốc góc* 0.0262) + (1 - 0.966)*(gia tốc)

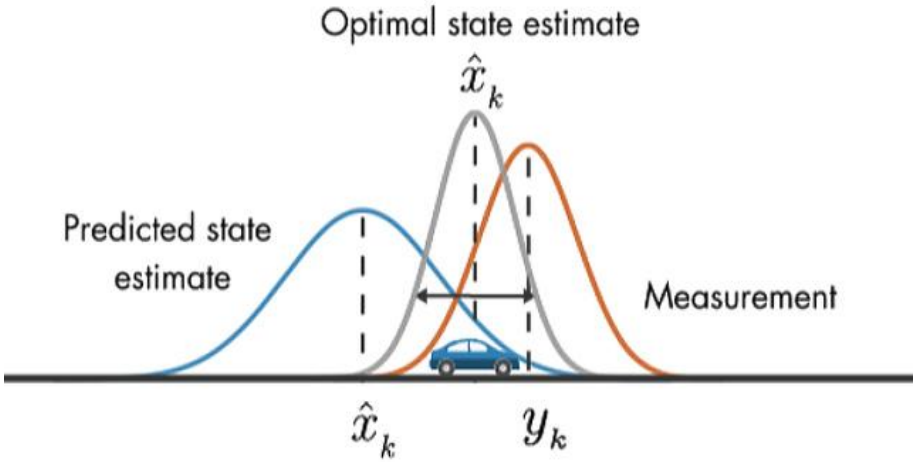
- 0.8 và 0.966 là trọng số được tính dựa vào thời gian đọc mẫu, 0.0262 là thời gian đọc mẫu

  Trọng số = Biên thời gian / (Biên thời gian + thời gian đọc mẫu)

Ví dụ : 0.966 = 0.75 / (0.75 + 0.0262). Trên biên thời gian giá trị cảm biến gia tốc tin cậy hơn, dưới biên thời gian giá trị cảm biến con quay hồi chuyển tin cậy hơn

Bộ lọc Kalman

- Bản chất của bộ lọc Kalman là dựa vào giá trị trước đó để dự đoán giá trị tiếp theo kết hợp với giá trị vừa đo được cho ra kết quả xấp xỉ



Phương trình toán học :

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

- x: biến trạng thái
- u: biến đầu vào
- z: trạng thái đầu vào
- w và v: Nhiễu quá trình và nhiễu đo lường

Lập trình bộ lọc Kalman

Các khái niệm liên quan :

- Trạng thái: x (góc)
- Hiệp phương sai: P (véc tơ)
- Hệ số Kalman: K (véc tơ)
- Phương sai nhiễu quá trình : Q
- Phương sai nhiễu đo lường: R

Đầu vào : Giá trị góc, vận tốc góc vừa đo được và thời gian đọc mẫu
Bộ lọc : Trải qua 7 bước tính toán
Đầu ra : Giá trị góc xấp xỉ

Sử dụng C++. Ta sẽ tạo một đối tượng tên là Kalman. Gồm các thuộc tính Q_angle, Q_bias, R_measure, angle, bias, rate, P[2][2]

Bước 1 :

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

$$\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k|k-1} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k$$

$$= \begin{bmatrix}\theta - \dot{\theta}_b\Delta t \\ \dot{\theta}_b\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k$$

$$= \begin{bmatrix}\theta - \dot{\theta}_b\Delta t + \dot{\theta}\Delta t \\ \dot{\theta}_b\end{bmatrix}$$

$$= \begin{bmatrix}\theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}_b\end{bmatrix}$$

```
rate = newRate - bias;
angle += dt * rate;
```

Bước 2 :

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$

$$\begin{bmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{bmatrix}_{k|k-1} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{bmatrix}_{k-1|k-1}\begin{bmatrix}1 & 0 \\ -\Delta t & 1\end{bmatrix} + \begin{bmatrix}Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b}\end{bmatrix}\Delta t$$

$$= \begin{bmatrix}P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11}\end{bmatrix}_{k-1|k-1}\begin{bmatrix}1 & 0 \\ -\Delta t & 1\end{bmatrix} + \begin{bmatrix}Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b}\end{bmatrix}\Delta t$$

$$= \begin{bmatrix}P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11}\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b}\end{bmatrix}\Delta t$$

$$= \begin{bmatrix}P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta\Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b}\Delta t\end{bmatrix}$$

$$= \begin{bmatrix}P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b}\Delta t\end{bmatrix}$$

```
P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);
P[0][1] -= dt * P[1][1];
P[1][0] -= dt * P[1][1];
P[1][1] += Q_gyroBias * dt;
```

Bước 3

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

$$= \boldsymbol{z}_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1}$$

$$= \boldsymbol{z}_k - \theta_{k|k-1}$$

y = newAngle - angle;

Bước 4

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \boldsymbol{R}$$

$$= P_{00\,k|k-1} + \boldsymbol{R}$$

$$= P_{00\,k|k-1} + var(v)$$

S = P[0][0] + R_measure;

Bước 5

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{S}_k^{-1}$$

$$= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} \boldsymbol{S}_k^{-1}$$

$$= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{\boldsymbol{S}_k}$$

K[0] = P[0][0] / S;
K[1] = P[1][0] / S;

Bước 6

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\,\tilde{\boldsymbol{y}}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\boldsymbol{y}}_k$$

$$= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0\,\tilde{\boldsymbol{y}} \\ K_1\,\tilde{\boldsymbol{y}} \end{bmatrix}_k$$

angle += K[0] * y;
bias += K[1] * y;

Bước 7

$$P_{k|k} = (I - K_k H) P_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

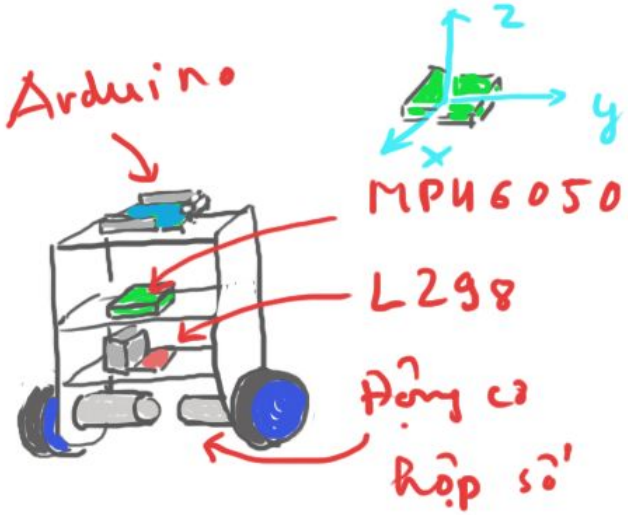
$$= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}$$

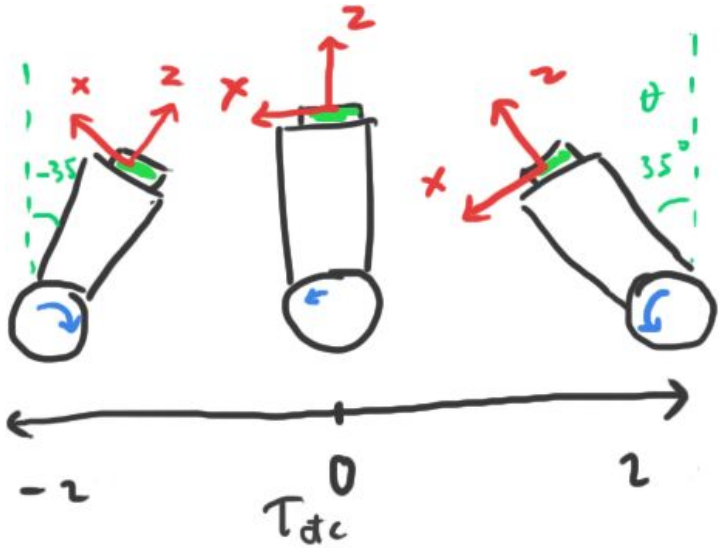
```
float P00_temp = P[0][0];
float P01_temp = P[0][1];

P[0][0] -= K[0] * P00_temp;
P[0][1] -= K[0] * P01_temp;
P[1][0] -= K[1] * P00_temp;
P[1][1] -= K[1] * P01_temp;
```

Đối tượng Kalman có 2 phương thức quan trọng là setAngle và getAngle như sau:

```
float getAngle(float newAngle, float newRate, float dt);  // lấy giá trị sau khi lọc
void setAngle(float angle);                               // truyền giá trị gốc
```

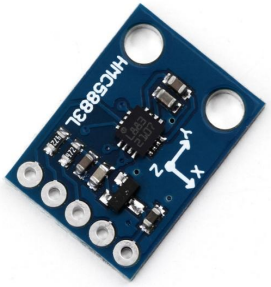
1 chút về PID trong robot cân bằng

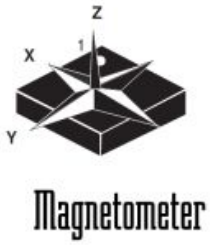




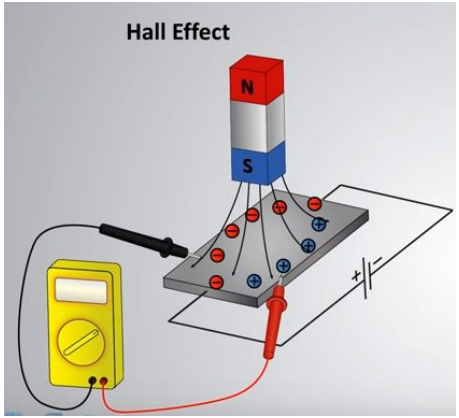
$$PWM_{đc} = \boxed{\text{sai lệch} \times K_p} + \boxed{K_i \times \int \text{sai lệch}} - \boxed{K_d \times \frac{d\ \text{sai lệch}}{dt}}$$

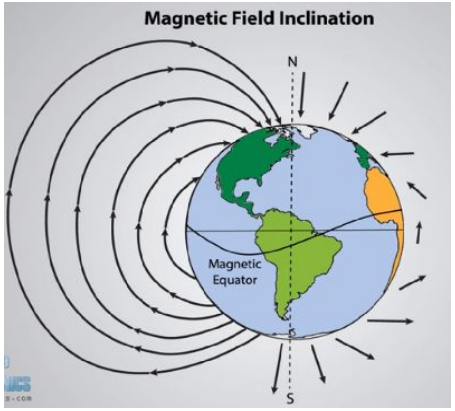
Một chút về cảm biến la bàn

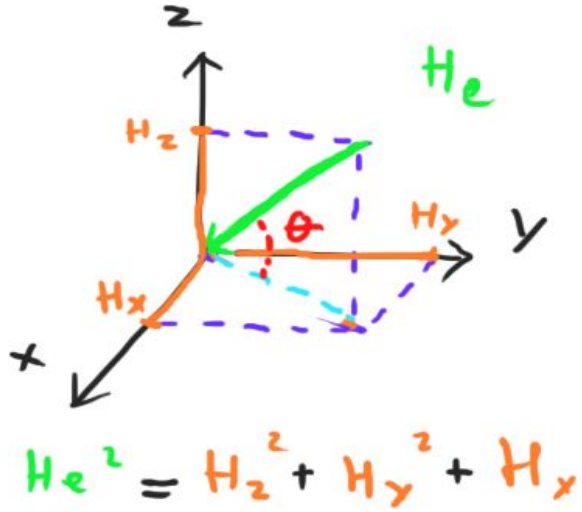




- Kết hợp với MPU6050 → 9 bậc tự do.
- Chịu ảnh hưởng của từ trường trái đất. Xác định phương hướng dựa vào hướng bắc
- Có thể bị nhiễu bởi các vật xung quanh có từ trường







Hướng dẫn code robot 2 bánh tự cân bằng

Thư viện có sẵn dành cho Arduino:

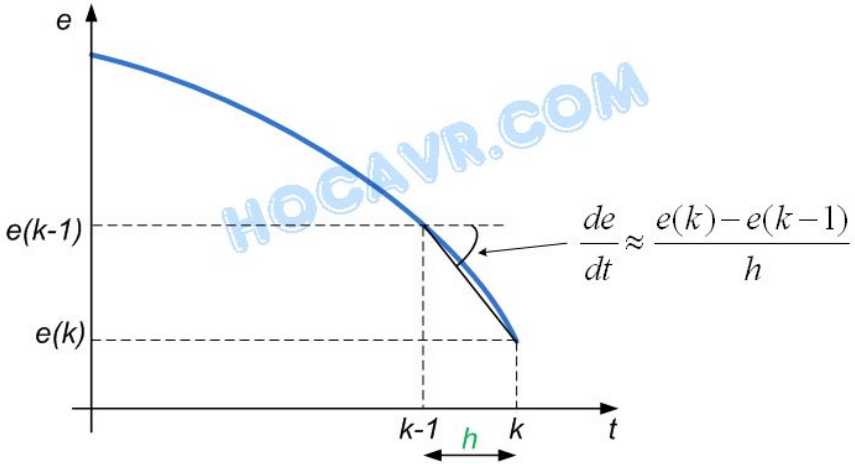
- MPU6050 (Jeff Rowberg) : https://github.com/jrowberg/i2cdevlib
- Kalman Filter Library (TKJ Electronics) : https://github.com/TKJElectronics/KalmanFilter

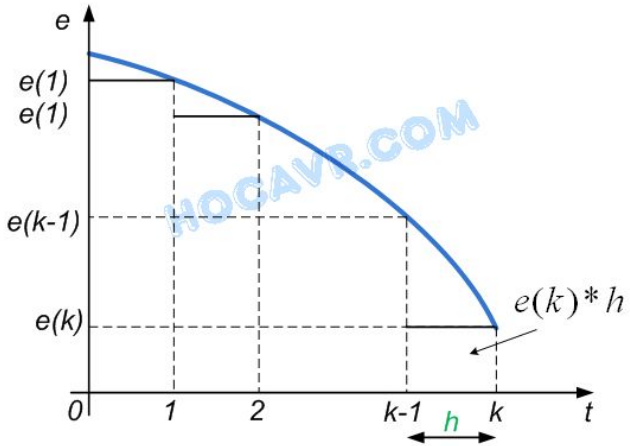
Cài đặt : **Sketch** → **Include Library** → **Manage Libraries** → gõ tên thư viện → **Install**

```
#include <Wire.h>
#include <Kalman.h>
#include "MPU6050.h"

void setup()
{
        KhoiTaoMPU6050();
        KhoiTaoDongCo();
}
```

```
void loop()
{
        DocCamBienMPU6050();
        GiuCanBang();
}
void GiuCanBang()
{
        GiaTriDongCo = TinhToanPID(GocNghiengHienTai);
        HieuChinhGiaTriDongCo();
        ThietLapGiaTriDongCo();
}
```

Lập trình PID






```
float error = RollAngle - BalanceAngle;
IValue += KI * error;
IValue = constrain(IValue, -255, 255);
float dError = error - LastError;
LastError = error;
float output = KP * error + IValue + KD * dError;
```