Robô de Auto Balanceamento

Luiz Felipe Almeida Silva
Marco Louredo Pimenta
Vicente Romeiro de Moraes
Victor Daniel Vieira Maciel

Projeto e Design

Idealização

- DROID
- Aprendizado e Experiência

Desafios

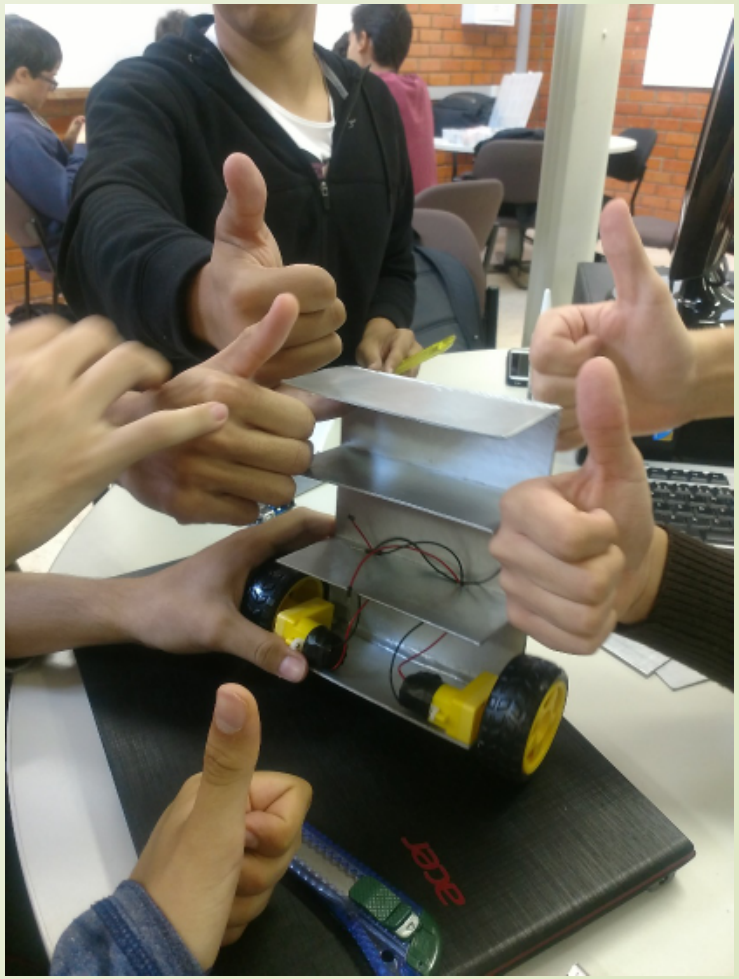
- Conhecimento
- Trabalho e Planejamento

Organização

- Pêndulo Invertido
- Sensores e Programação

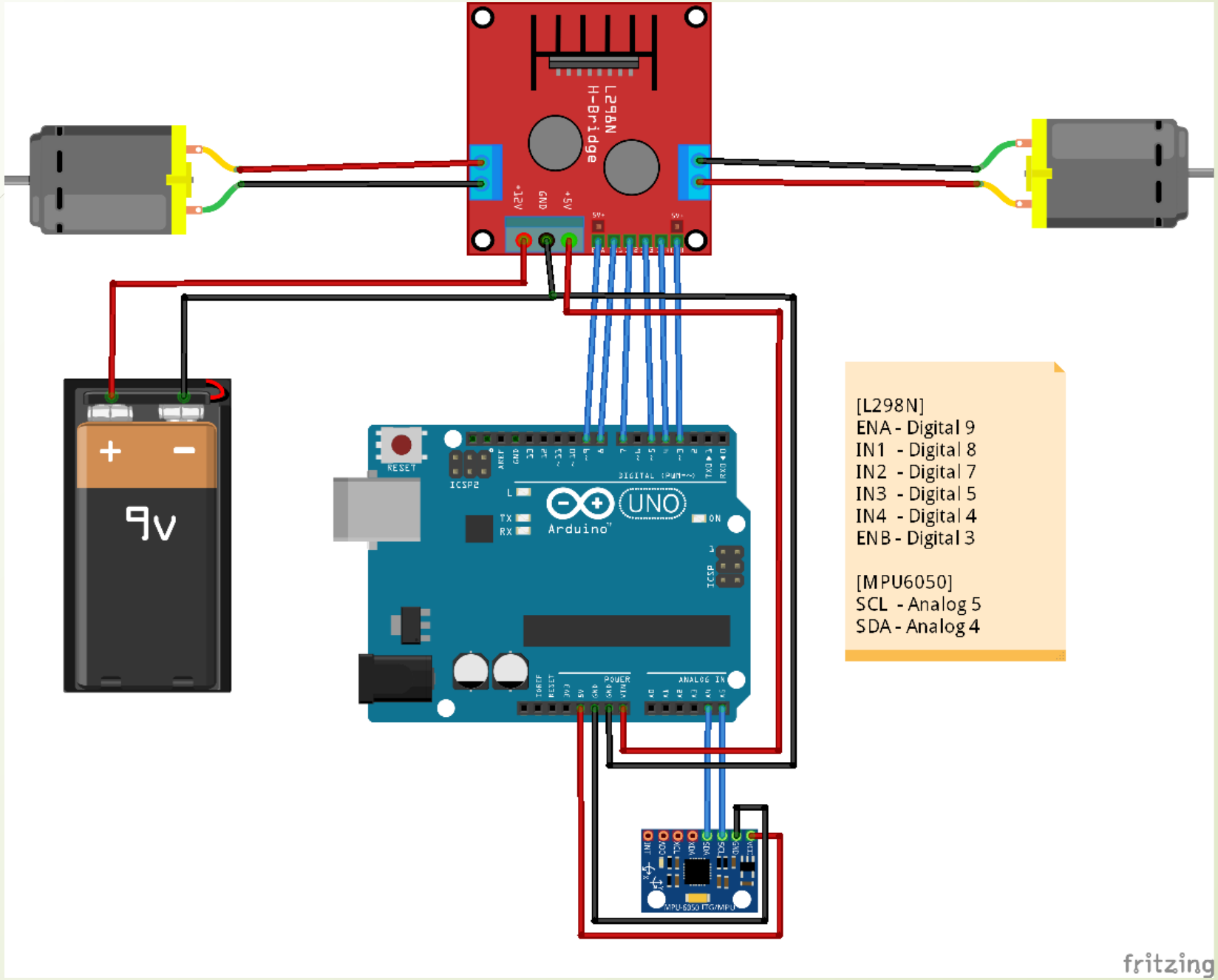
Orçamento e Organização

Arduino Uno
2 Kits Motor 6V com Caixa de Redução – R$ 55,80
Ponte H l298n – R$ 19,90
Acelerômetro / Giroscópio MPU6050 – R$ 15,90
Custos Adicionais (Bateria e Estrutura) – R$ 21,90
Orçamento Total – R$ 113,50

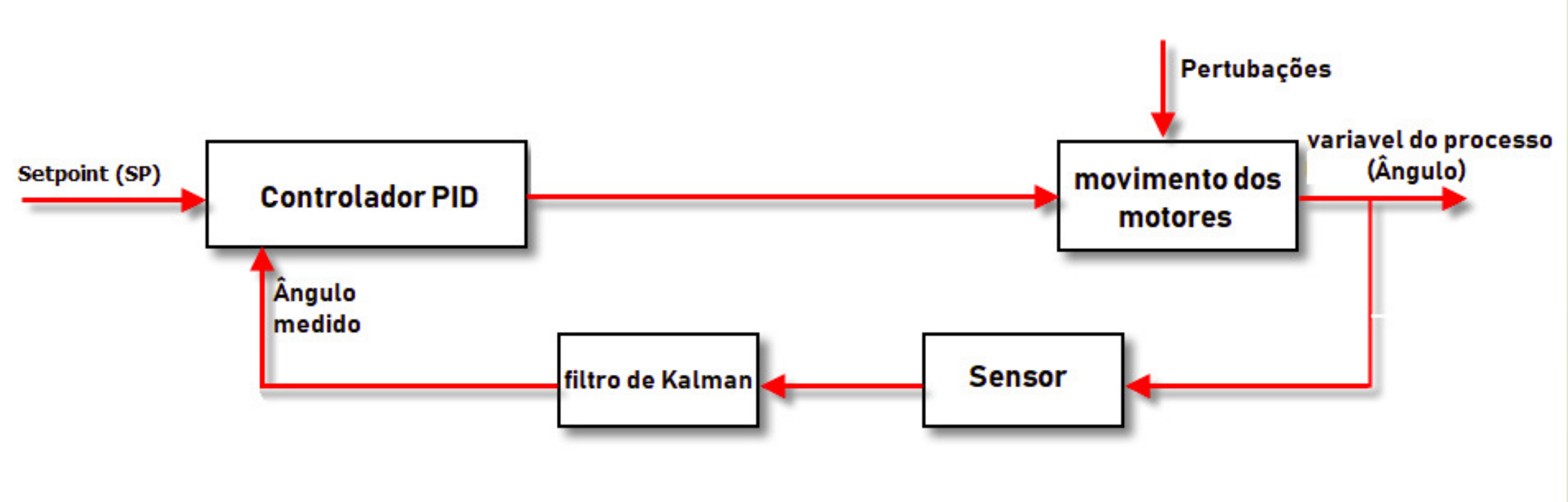


1ª Quinzena 02/04-15/04	2ª Quinzena 16/04-29/04	3ª Quinzena 30/04-13/05	4ª Quinzena 14/05-27/05	5ª Quinzena 28/05-10/06	6ª Quinzena 11/06-24/04	7ª Quinzena 25/06-06/07
Discussão						
	Compra de Materiais					
	Planejamento de Algoritmos	Implementação e Programação	Implementação e Programação	Testes	Testes	
		Montagem	Montagem	Formulação dos Slides	Formulação dos Slides	
						Apresentação

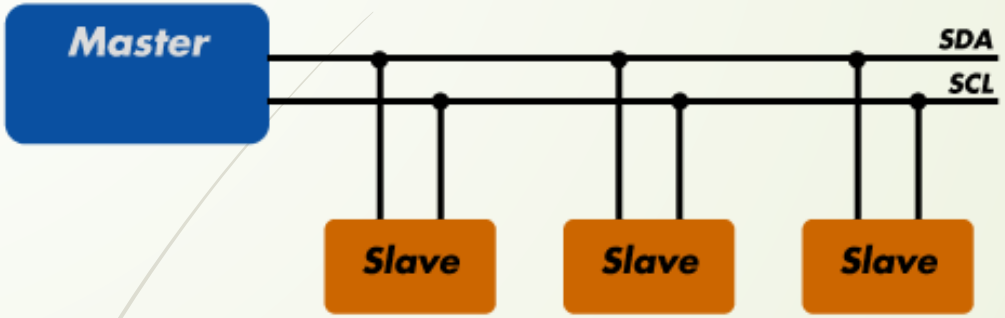
Diagrama


L298N
H-Bridge
+12V
GND
+5V
9v
RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN
[L298N]
ENA - Digital 9
IN1 - Digital 8
IN2 - Digital 7
IN3 - Digital 5
IN4 - Digital 4
ENB - Digital 3
[MPU6050]
SCL - Analog 5
SDA - Analog 4
fritzing

Fluxograma


Pertubações
Setpoint (SP)
Controlador PID
movimento dos motores
variavel do processo (Ângulo)
Ângulo medido
filtro de Kalman
Sensor

Comunicação I2C



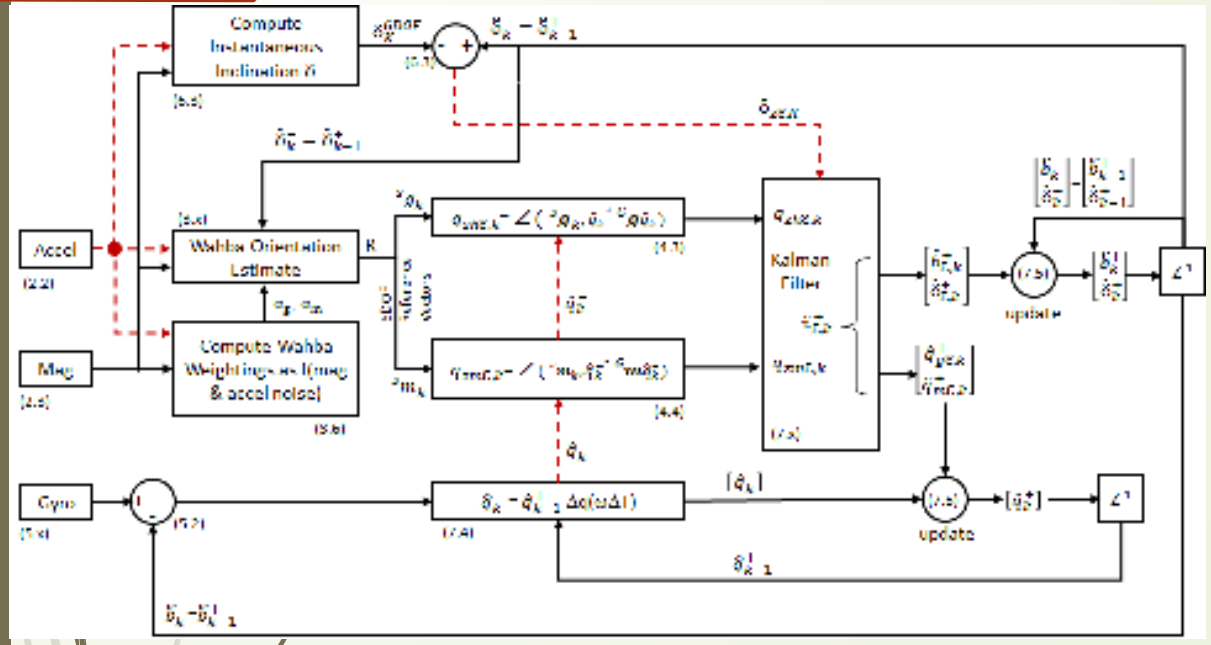
Sistema Master/Slave

MPU

```
#include <Wire.h>
#define MPU 0x68    //define MPU como sinonimo de 0x68

//valores brutos lidos pelo acelerometro
double accX, accY, accZ;
//angulo calculado atraves das aceleraçoes
double accAng;
//valores brutos lidos pelo giroscopio
double gyroX, gyroY, gyroZ;
//angulo calculado atraves dos giros
double gyroAng, gyroBias;
double ultimo_gyroAng;

long temp_anterior;

long dt;

void gyro_Bias(){
  int i;
  for(i=0;i<100;i++){
    LerGyro();
    gyroBias += gyroY;
  }
  gyroBias = gyroBias/100;

}


void inicializar_MPU(){

  //inicializa a biblioteca Wire.h
  Wire.begin();
  //inicia transmissão para o MPU
  Wire.beginTransmission(MPU);
  //reseta o MPU
  Wire.write(0x6B);
  Wire.write(0);
  //finaliza trasmissão e libera o dispositivo
  Wire.endTransmission();

}

void LerAccel(){

  //inicia transmissão para o MPU
```

Algoritmo I2C

Fusão de Sensores e Filtro de Kalman



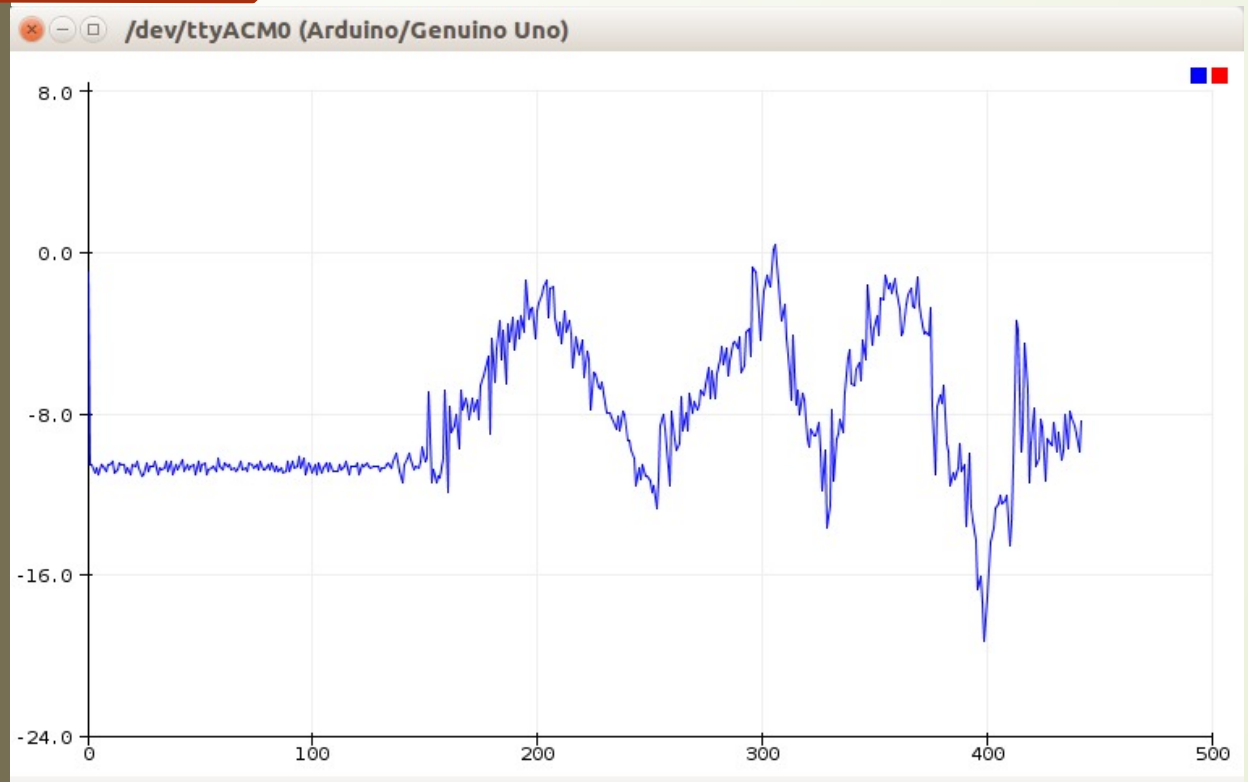
Modelo Matemático

Meu_Kalman §

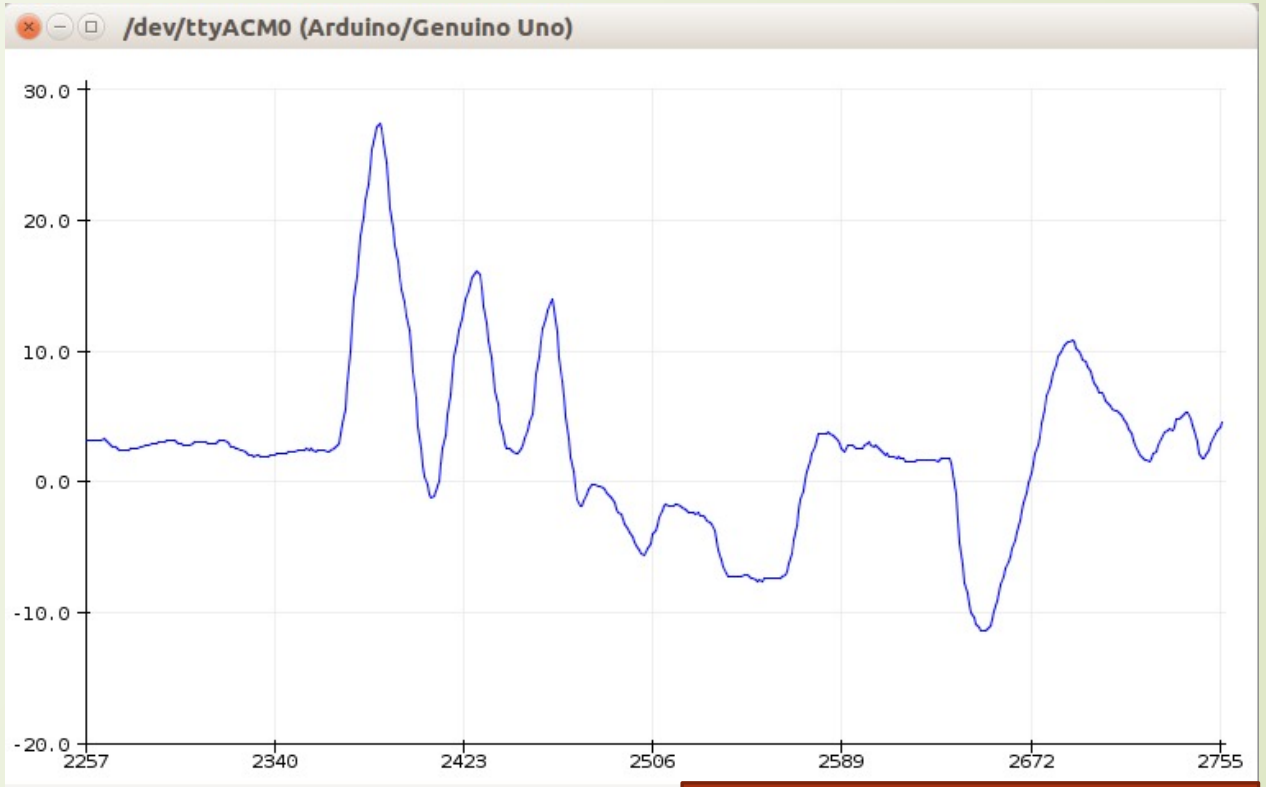
```
double Q_angulo = 0.001;
double Gyro_Q_angulo = 0.003;
double R_angulo = 0.3;

double x_angulo = 0;
double erro_x = 0;
double P00 = 0, P01 = 0, P10 = 0, P11 = 0;
double y, S;
double K, Kpos;

double Kalman(double angulo, double taxa, double tempo){
  double dt = tempo / 1000;
  x_angulo += dt * (taxa - erro_x);
  P00 -= dt * (P10 + P01) + Q_angulo * dt;
  P01 -= dt * P11;
  P10 -= dt * P11;
  P11 += Gyro_Q_angulo * dt;

  y = x_angulo - erro_x;
  S = P00 + R_angulo;
  K = P00 / S;
  Kpos = P10 / S;

  x_angulo += K * y;
  taxa += Kpos * y;
  P00 *= -K;
  P01 *= -K;
  P10 *= -Kpos;
  P11 *= -Kpos;

  return x_angulo;
}
```

Filtro de Kalman Discreto

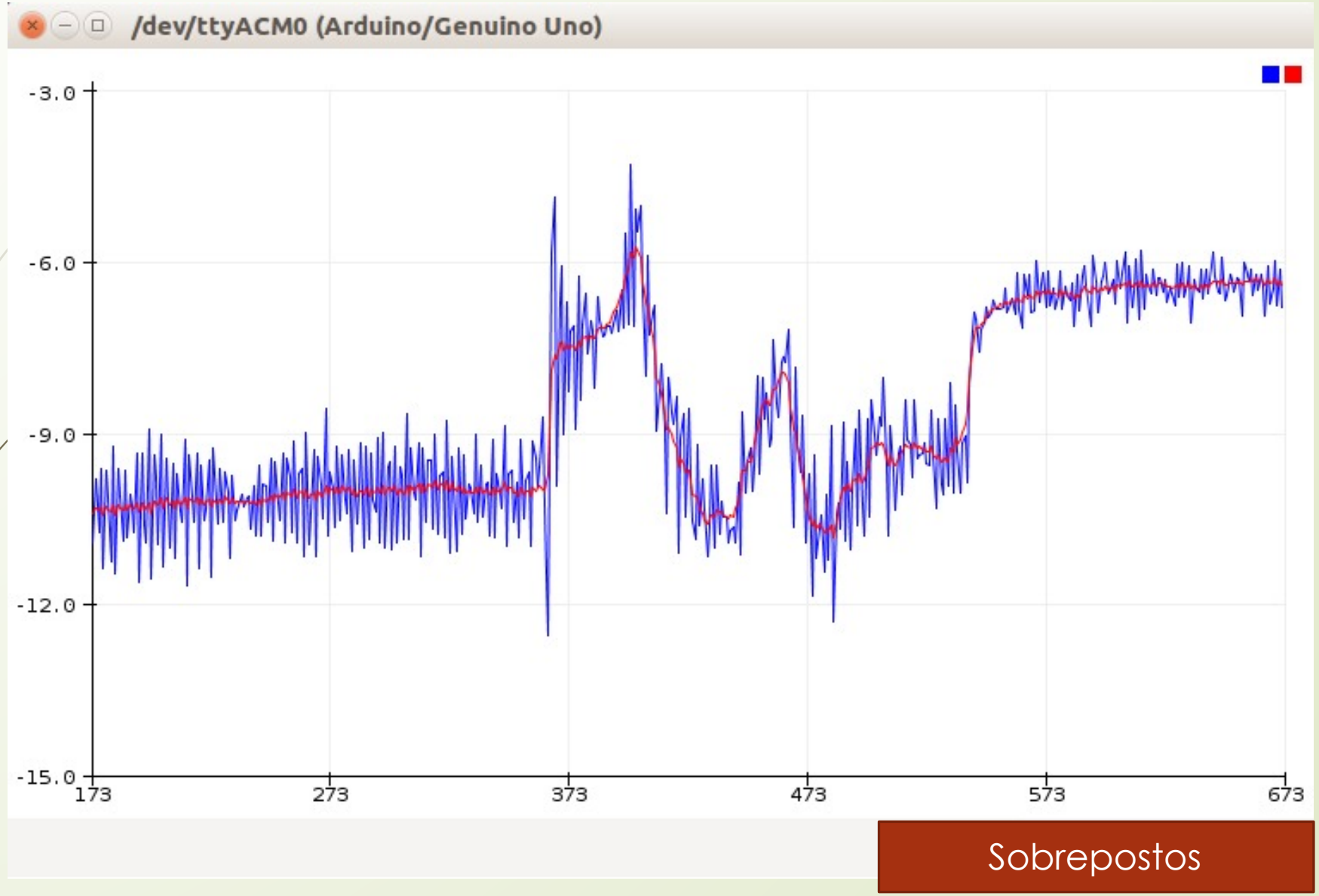
Aplicação do Filtro de Kalman



Medição Bruta



Filtro Aplicado



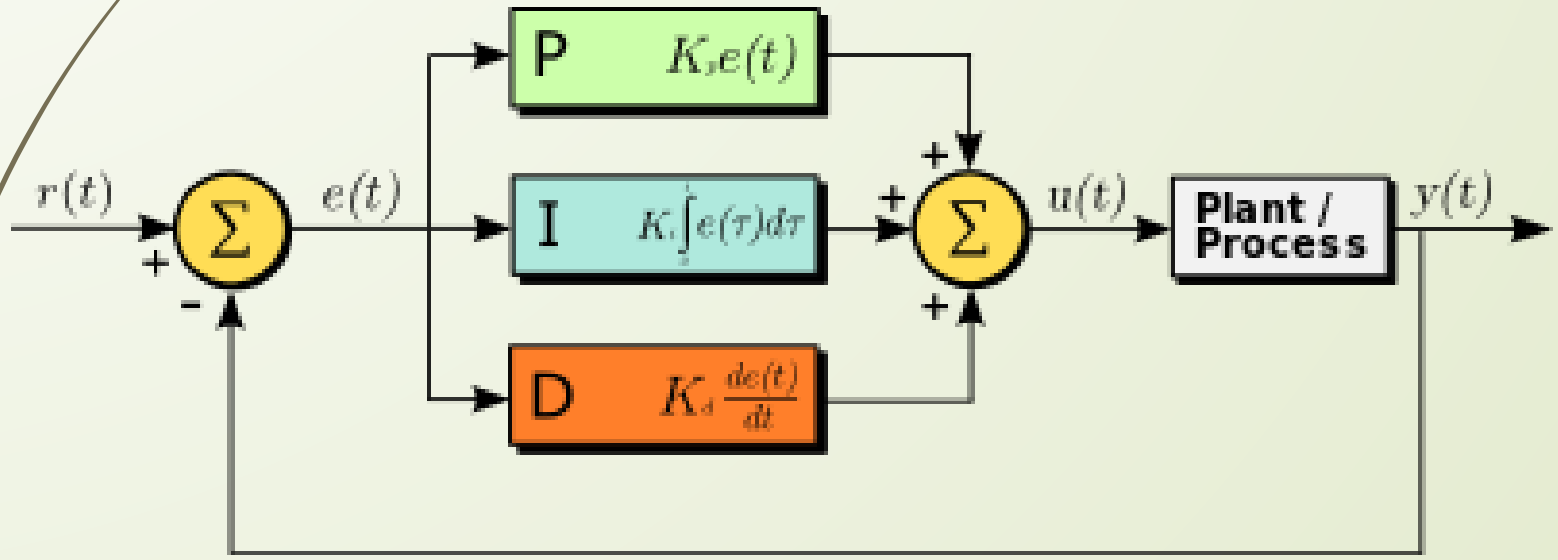
Sobrepostos

Controle PID

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

Output = proportional + integral + derivative

Modelo Matemático




```
PID

double Kp;        //constante de ganho proporcional
double Ki;        //constante de ganho integral
double Kd;        //constante de ganho derivativo

double P;         //termo proporcional
double I = 0;     //termo integral
double D;         //termo derivativo

double PID;       //saída
double erro;
double setPoint; //Angulo ideal

double ultima_entrada = 0;

int outMin = -200;      //saida PWM mínima
int outMax = 200;       //saida PWM máxima

double calcular_PID(double entrada){

  //calculo do erro
  erro = setPoint - entrada;

  //calculo do termo Proporcional
  P = Kp * erro;
  //calculo do termo Integral
  I = I + (Ki * erro);
  //calculo do termo Derivativo
  D = Kd * (entrada - ultima_entrada);

  //mantém o valor do termo integral entre -200 e 200;
  if(I >= outMax) I = outMax;
  if(I <= outMin) I = outMin;

  //calculo da saída
  PID = P + I + D;

  //mantém o valor da saída entre -200 e 200;
  if (PID >= outMax) PID = outMax;
  if (PID <= outMin) PID = outMin;

  return PID;

}


void setSetPoint(double val){
```

Algoritmo PID

Finalização e Revisão

Conclusão

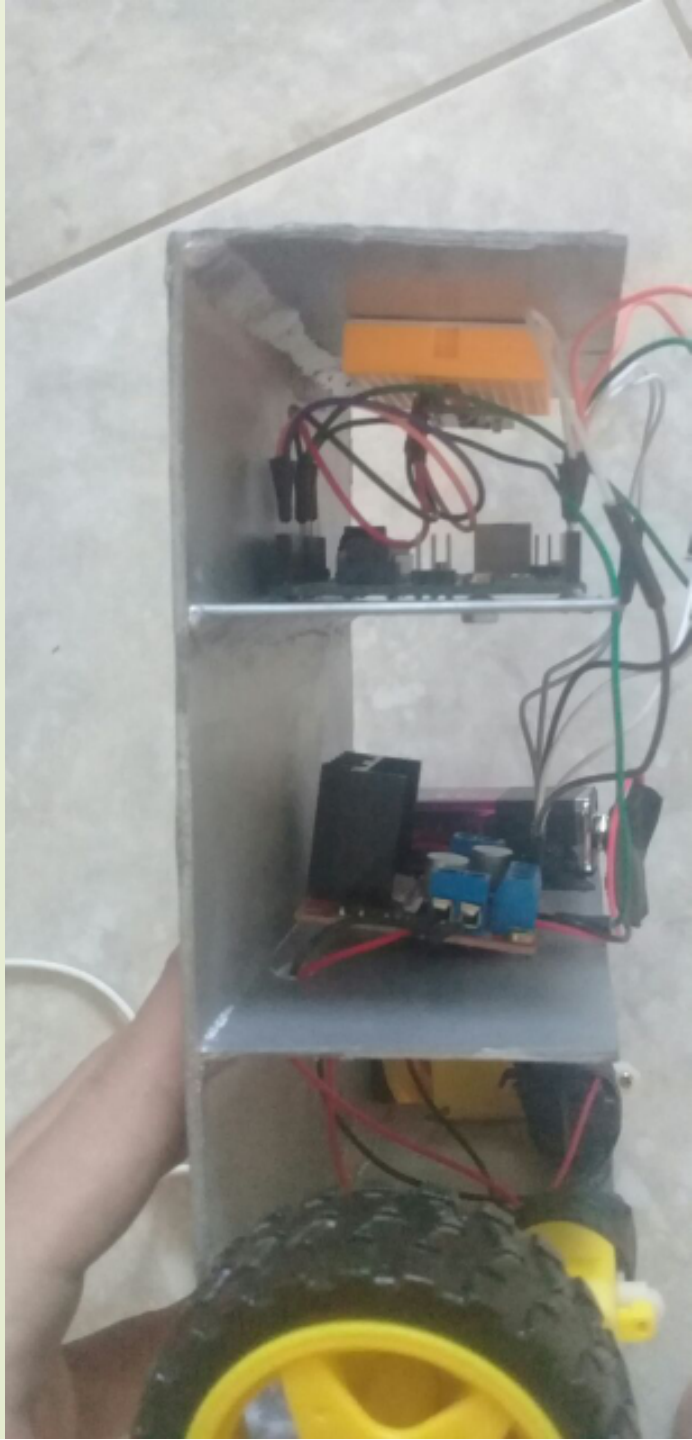
- Depois de cair muito, o robô não cai mais!

Avaliação

- Desenvolvemos conceitos mais avançados no curso.
- Trabalho em equipe.

Revisão e Aplicações

- Sistemas de auto balanceamento e a fusão de sensores se provam cada vez mais essenciais a medida que robôs emulam seres vivos e passam a interagir cada vez mais com o homem.

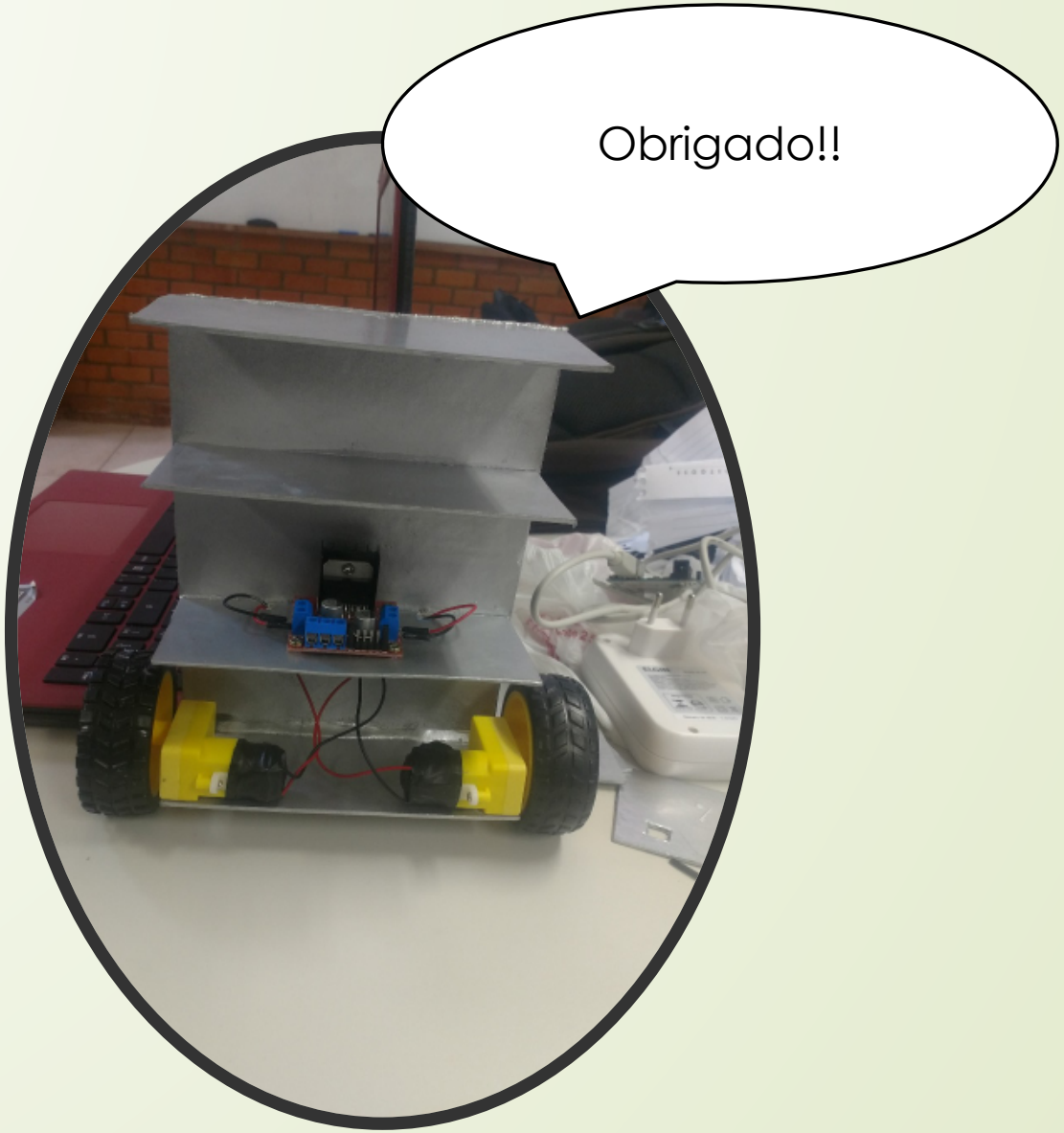


Sugestões

- Trocar motores por motores com Encoders Rotativos ou Motores de Passo
- Controle com servos
- Mais controle em geral

Referências

- https://www.citisystems.com.br/controle-pid/
- http://www.kerrywong.com/2012/03/08/a-self-balancing-robot-i/
- https://github.com/TKJElectronics/KalmanFilter
- http://www.arduinobr.com/arduino/i2c-protocolo-de-comunicacao/
- http://www.pieter-jan.com/node/11
- KIM, Phil. *Kalman Filter for Beginners: With MATLAB Examples. National Rehabilitation Research Institute of Korea, CreateSpace Independent Publishing Platform*, 2011.



Chinelinha