UNIVERSIDADE FEDERAL DE UBERLÂNDIA - UFU

FACULDADE DE ENGENHARIA ELÉTRICA – FEELT

SINAIS E SISTEMAS EM ENGENHARIA BIOMÉDICA

Trabalho Final: Pêndulo Invertido

Alunos:

1. Ítalo Gustavo Sampaio Fernandes - 11511EBI004
2. Paulo Camargos Silva - 11611EBI023

Prof. Dr. Sérgio Ricardo de Jesus Oliveira

Uberlândia, **18** de **dezembro** de 2017

1 – Introdução

Um sistema de pêndulo invertido tem diversas aplicações. O sistema proposto é relativamente simples, porém requer uma engenhosidade na montagem e pode ser tornar complexo. A utilização dos controladores PID nestes sistemas é relativamente comum. Este projeto foi montado utilizando os controladores PID com Kd próximo de 0 devido a complexidade de configuração dos parâmetros Kp, Ki e Kd e de forma que pudéssemos avaliar a influência do controlador D na resposta do sistema.

O sistema de pendulo invertido tem o princípio de manter a haste em um ângulo definido pelo SetPoint, mesmo quando uma força externa é aplicada ao sistema, deslocando a haste para um ângulo diferente. Para isto, foram utilizados motor e potenciômetros para aplicação sinal e detecção do ângulo respectivamente.

O sistema conta ainda com a utilização de de módulo bluetooth para comunicação com o computador e smartphone sem fio. Um aplicativo simples criado para smartphone permite a visualização da posição angular da haste, das constantes Kp e Ki e sua configuração.

2 – Materiais e Métodos

2.1 - Materiais

Os materiais utilizados neste projeto foram:;

- 1 Arduíno Due;
- 1 LED RGB e 3 resistores de 10 kΩ;;
- 1 potenciômetro de 10 kΩ;
- 1 módulo bluetooth HC 05;
- 1 driver para motor L298N;
- 1 motor DC (Pololu Micro GearMotor Low Power - 6V - torque 0.3 Kg.cm - caixa de redução de 30:1 e consumo de 0.36A);
- 1 roda para motor Pololu;
- 2 pilhas de 3.7 V recarregáveis de lítio, montadas em série;
- 1 chave HH - On/Off;
- 6 Jumpers macho-fêmea para conexão da ponte H;
- 2 Jumpers macho-fêmea para conexão do bluetooth;
- 1 palito de picolé montado como seta.

2.2 - Descrição geral de funcionamento

O projeto tem como principal objetivo a criação de um sistema de pêndulo invertido, onde uma posição desejada deverá ser fornecida e a haste do pêndulo deverá se manter naquela posição. Distúrbios externos podem ser aplicados de forma que alterem a posição da haste. Com a utilização dos controladores PID, a haste deverá retornar a posição desejada.

Para a confecção do projeto, foi utilizado um Arduino Due para o sistema de controle e um software foi escrito de forma que permitisse o ajuste das constantes através da porta serial. Um motor DC com caixa de redução foi utilizado para rotacionar a haste. Este motor foi conectado ao módulo LN298 contendo uma ponte H. Este módulo foi conectado ao Arduino.

De forma que o sistema pudesse ser utilizado sem estar conectado ao computador e a alguma fonte de alimentação fixa, foi adicionado ao sistema um módulo bluetooth para compartilhamento e alteração dos parâmetros do sistema e um conjunto de 2 pilhas em série foi utilizado para alimentação de todo o sistema.

Através de um aplicativo simples no smartphone, é possível verificar e alterar os valores de Kp, Ki e Kd, bem como o estado do sistema (ligado/desligado). Estes procedimentos também podem

ser efetuados pelo computador através da porta serial. Foi adicionado também ao sistema, um LED RGB de aviso indicando o erro do sistema. A cor do LED varia com a intensidade do erro. Erro grande, o LED possui cor próxima ao vermelho, enquanto erro pequeno, a cor fica próxima ao verde.

2.3 - Diagrama de blocos do sistema.

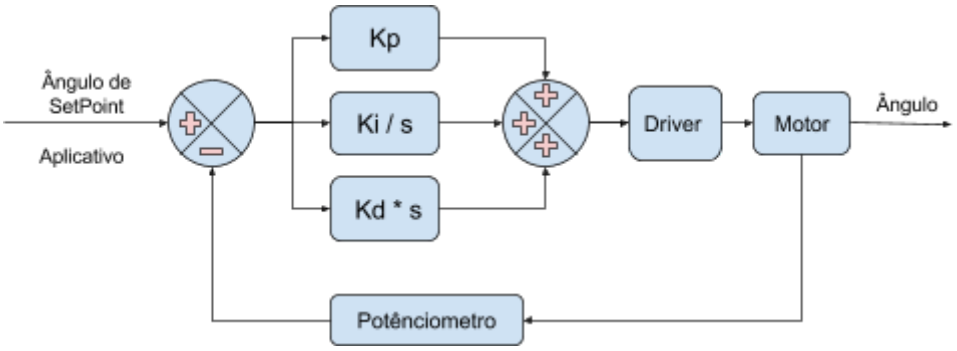

Figura 1: Diagrama de blocos do sistema.

O diagrama de blocos exibido acima monsta como a posição angular de saída (PV) é subtraída da posição angular de entrada (SP). Esta diferença é o erro e então sinal atuante é então calculado utilizando os valores das constantes de cada controlador, Kp, Ki e Kd. Para conversão da posição angular do sistema na unidade de medida utilizada pelo arduino (tensão elétrica), utilizamos um potenciômetro acoplado ao eixo do motor. A tensão no potenciômetro é proporcional ao ângulo do seu eixo. O sinal atuante é então enviado do Arduino para um driver (LN298). O sinal do sinal atuante indicará o sentido da rotação na qual o motor deverá girar.

3 – Resultados e discussões

Inicialmente, o sistema deveria incluir dois motores para a sustentação de um robô em duas rodas. O sistema incluiria o controlador PID e um sensor inercial MPU6050. O sistema foi montado e durantes os teste, algumas dificuldades como calibração dos valores das constantes e velocidade de resposta do sensor inercial. Pudemos avaliar na prática a dificuldade de calibração dos 3 parâmetros do PID e a real necessidade de um sistema rápido, pois a queda do robô pode ocorrer em um intervalo de tempo relativamente pequeno.

Após os testes do primeiro sistema, foi decidido utilizar somente 1 motor para controlar uma haste. Este sistema foi montado e foi necessário a substituição do módulo do sensor inercial por um potenciometro para leitura da posição angular. Pudemos observar que o potenciômetro proporcionou um controle mais fácil e mais preciso que o sensor.

Durante a fase de testes, ocorreram problema com o encaixe do eixo do motor com o potenciômetro de leitura do ângulo, pois o encaixe do potenciômetro é diferente do eixo da caixa de redução do motor. Foram testadas formas diferentes de encaixe. Ao final, foi decidido usar uma pequena roda com o encaixe da caixa de redução e colar junto ao eixo do potenciômetro. Não houve problemas com a transmissão de torque do motor para a carga por ela ser relativamente pequena.

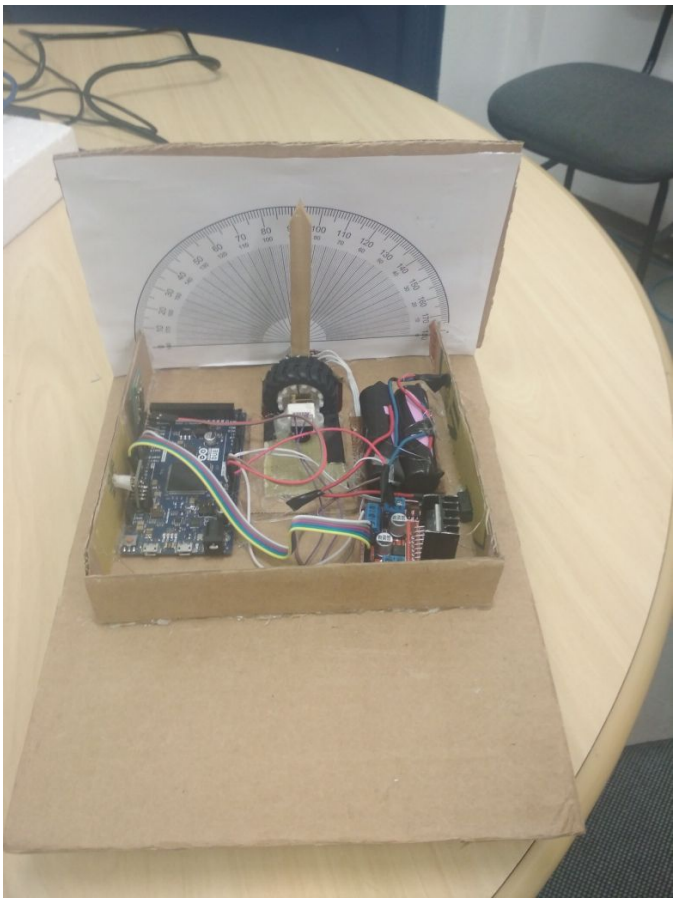

Figura 2: Sistema de pêndulo invertido montado.

Utilizando o código no arduino, foi possível medir o erro do sistema, configurar valores para Kp e Ki, aplicar o sinal e corrigir a posição angular da haste. Foi inserido também meio ciclo trigonométrico para verificação da posição angular como referência da haste.

Ao ligar o sistema, o sistema exige a inicialização também através do software e consequente configuração dos valores de Kp, Ki Kd. Com estes parâmetros zerados no início, a haste mantém a posição inicial em um ângulo qualquer. Ao configurar o primeiro valor de Kp, podemos observar que a haste tende a se mover para a posição desejada (em 90º), porém ao se aproximar (erro aproximadamente 10º), a velocidade se reduz inversamente proporcional ao erro. Isto é esperado, pois no controlador proporcional o sinal atuante é diretamente proporcional ao erro, e com sua redução, o sinal atuante também reduz até que não seja suficiente para mover a haste e ela se fixa em uma posição, diferente da posição desejada. Esta característica é o erro de regime permanente igual a zero.

No instante que o valor de Ki é configurado, observamos a haste mover-se até o setpoint, porém lentamente. Isto também é esperado, pois o controlador integral tem uma ação lenta, porém um bom compromisso com o erro de regime igual a 0. Desta forma, a atuação dos dois controladores nos permite uma velocidade de resposta relativamente boa para a aplicação (apenas visualização) e uma boa precisão da posição angular.

Foi testado também o funcionamento do sistema apenas com o controlador Ki. Foi observado que o sinal atuante é aplicado a todo momento e o sistema não se estabilizou. Isto é devido a própria característica do controlador I, que vai utilizando o erro acumulado para cálculo do sinal atuante. Sem a presença do controlador P, este erro acumulado não é reduzido e o sistema não atinge a estabilidade.

Abaixo são apresentados gráfico plotados com o serial plotter do arduino para os valores de sinal PI e posição angular. A legenda das cores na ordem são:

- Azul escuro: setpoint;
- Vermelho ângulo medido;
- Verde: erro;
- Laranja: resposta P;
- Roxo: resposta I;
- Verde escuro: resposta D;
- Azul claro: resposta PID.

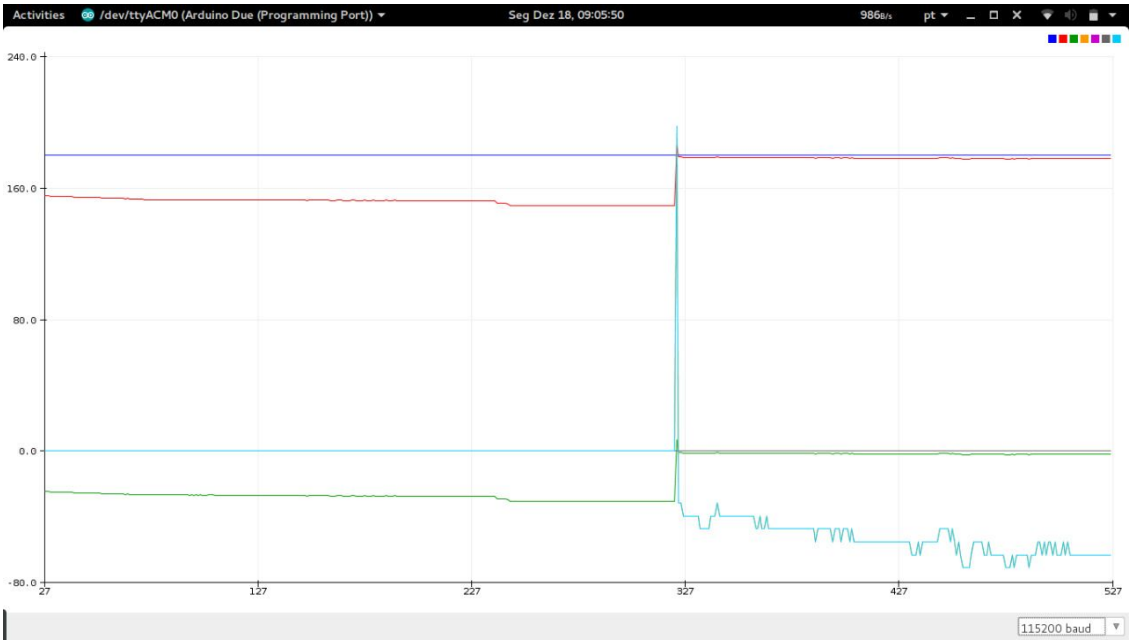

Figura 3: Kp = 30; Ki = 0. Resposta rápida, porém é possível notar a presença do erro de regime permanente.

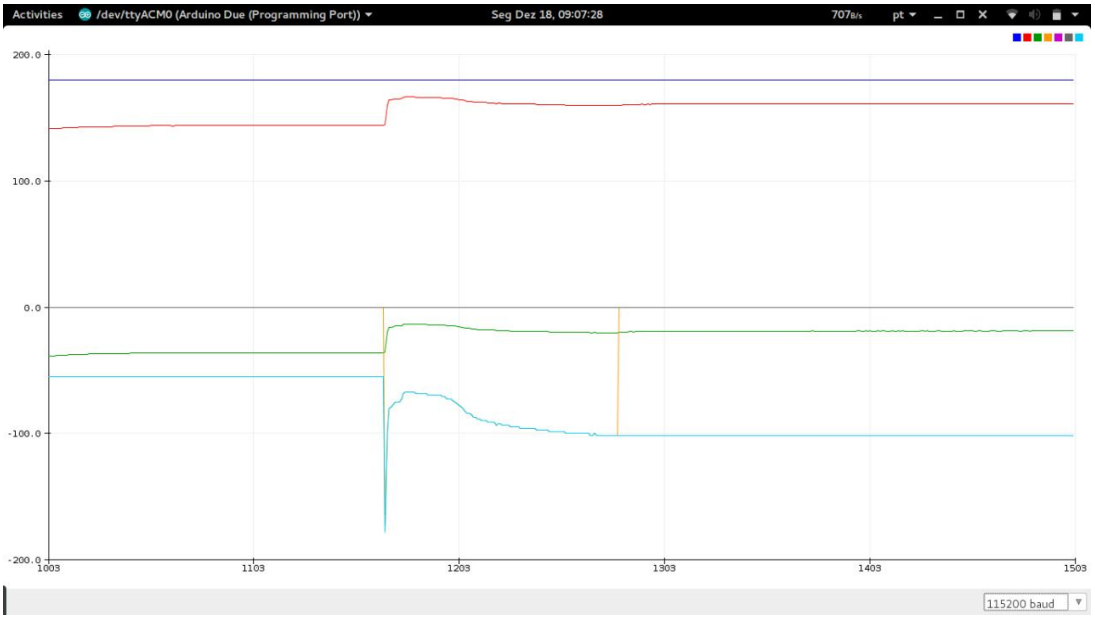

Figura 4: Kp = 5; Ki = 0. Resposta mais lenta que a anterior e com erro de regime permanente maior que o anterior.

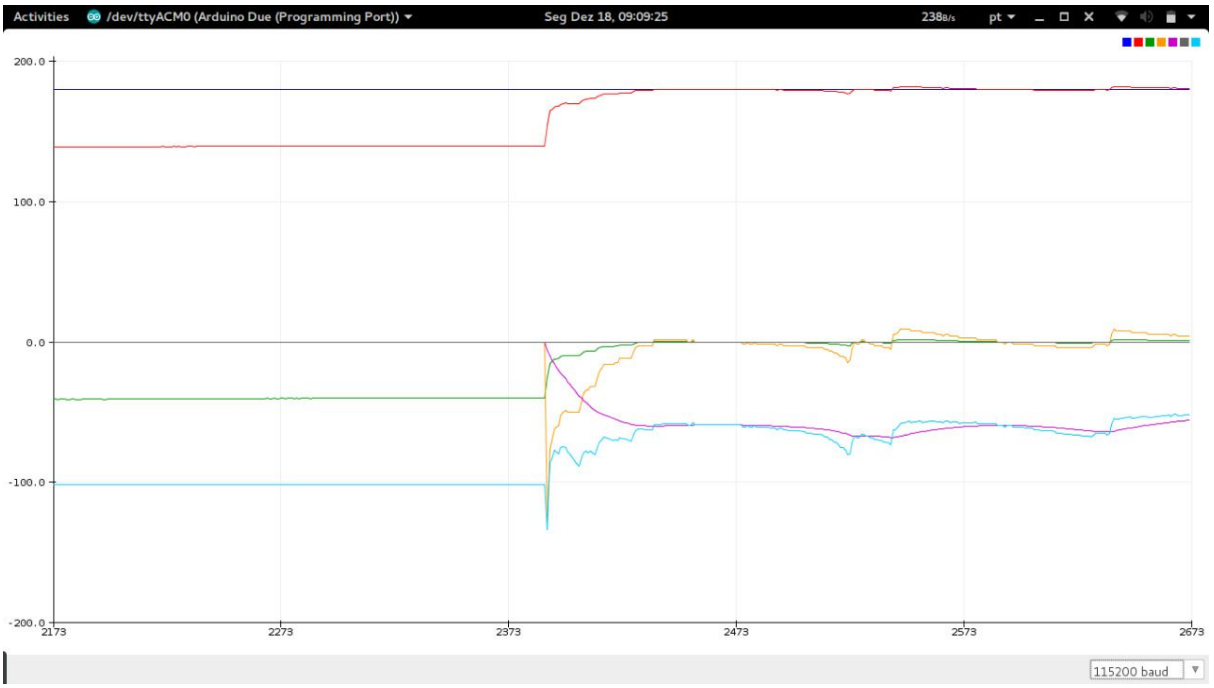

Figura 5: Kp = 5; Ki = 2. resposta relativamente lenta, porém erro de regime permanente próximo de 0.

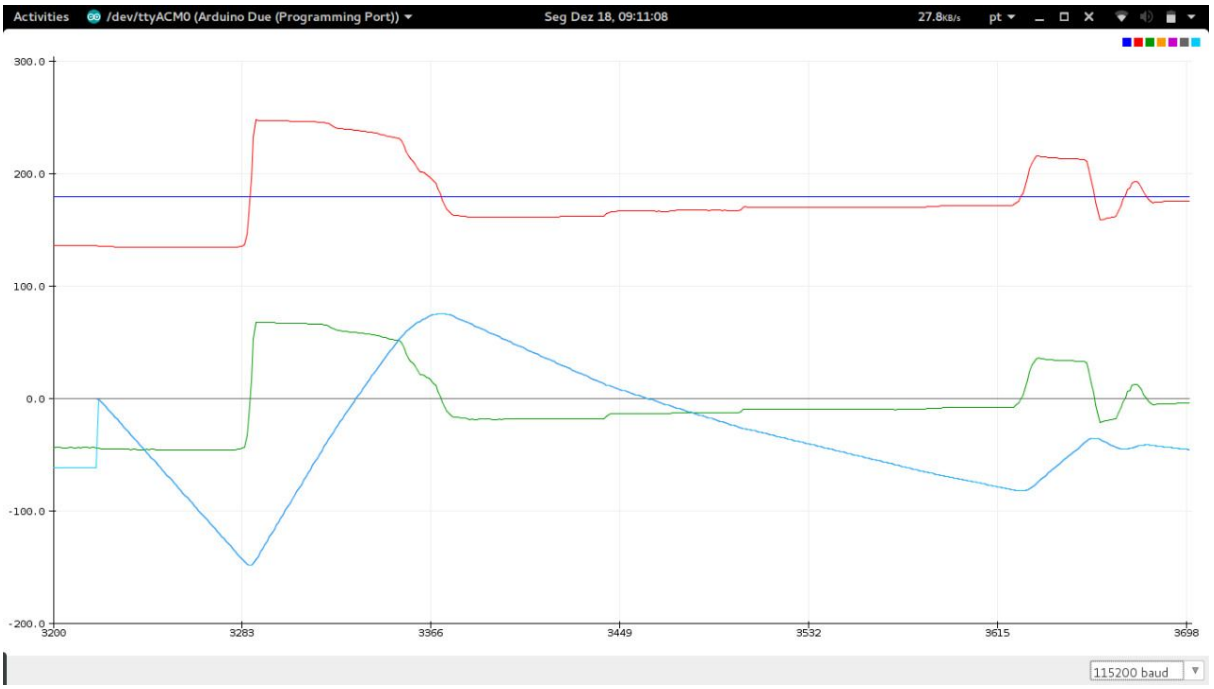

Figura 6: Kp = 0; Ki = 1. O sistema oscilou e não atingiu a estabilidade.

Foi adicionado um valor em Kd igual a 0.01 para observar se a estabilização do sistema seria mais precisa e mais rápida que os controladores PI usados anteriormente, porém, não foi possível observar diferença significativa na resposta. A figura abaixo exibe as informações deste sistema:

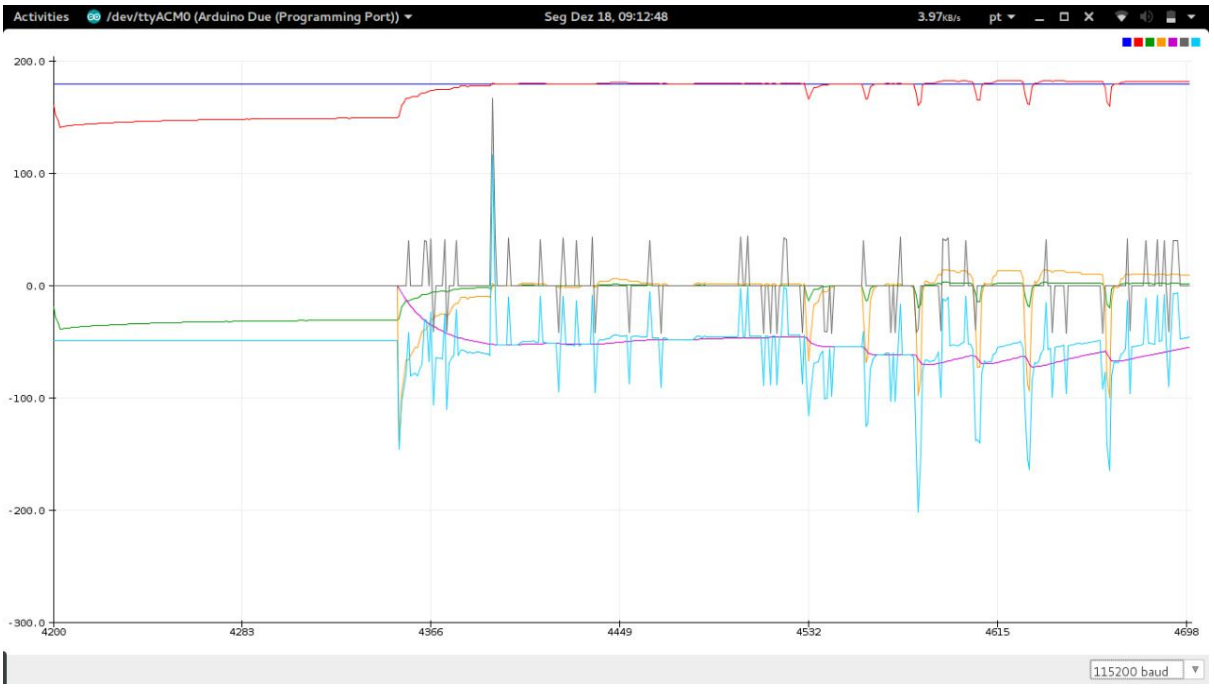

Figura 7: Kp =5; Ki = 2; Kd = 0.01. Inserção do controlador derivativo com este valor de Kd não afetou a resposta do sistema de forma significativa. Resposta igual a do PI mas o derivativo faz com que o sistema responda suporte melhor distúrbios externos (picos dos gráficos mostram oscilação manual do pêndulo, pode-se notar q ele nao se alterou significativamente).

4 – Conclusão

É possível concluir que a utilização do sistema de controle PID é uma importante ferramenta com diversas aplicações tecnológicas. Através dele foi possível implementar um sistema de robusto, capaz de se adequar a diversas aplicações, não somente ao hardware montado. Tal sistema pode ser aplicado ao controle de equilíbrio de robôs, controle de posição em próteses eletrônicas e, recentemente, foi aplicado para o controle de um robô ciclista capaz de andar de bicicleta mantendo seu equilíbrio como um humano.

5 – Referências

1. **Embarcados.** Controlador Proporcional Eletrônico. Disponível em: <https://www.embarcados.com.br/controlador-proporcional-eletronico/>;
2. **Arduino**. Documentação. Disponível em: <https://www.arduino.cc/reference/>;
3. **National Instruments.** PID Control. Disponível em: <http://www.ni.com/white-paper/3782/pt/>
4. **Sebastian O.H. Madgwick**. 2010. An efficient orientation filter for inertial and
5. inertial/magnetic sensor arrays.

Anexo I - CÓDIGO

Disponível em:
https://github.com/italogfernandes/SEB/tree/master/projeto_final_potenciometro_e_motor/projeto_final_potenciometro

```
/* UNIVERSIDADE FEDERAL DE UBERLANDIA
  Biomedical Engineering
  Autors: Ítalo G S Fernandes
  Paulo Camargos Silva
  contact: italogsfernandes@gmail.com
  URLs: https://github.com/italogfernandes/SEB
  Este codigo faz parte da disciplina de sinais e sistemas
  para engenhara biomedica e visa controlar o angulo de
  Os sinais de entrada e saida sao enviados para a interface Serial
  podendo ser visualizado pelo serial plotter.
  Esquema de montagem:
  Arduino - Dispositivo
  A0    - Potenciometro para configurar o setPoint de 0 a 360 graus

  A1    - Potenciometro para configurar o Kp de 0 a 3
  A2    - Potenciometro para configurar o Ki de 0 a 3
  A3    - Potenciometro para configurar o Kd de 0 a 3

  VCC - Potenciometros entrada
  GND - Potenciometros GND

  6 - Pino Enable ou de Velocidade da PonteH
  5 - Pino In1 da PonteH
  4 - Pino In2 da PonteH

  A4 - SCL do MPU66050
  A5 - SDA do MPU6050

  Obs: O sentido do motor
*/
#include<led_rgb.h>
#include<cores_rgb.h>
led_rgb status_led(11, 12, 13); //R_pin,G_pin,B_pin

#define ANALOG_ANGULO A0

#define PINO_MOTOR_A    2
#define PINO_IN1        3
#define PINO_IN2        4
#define PINO_IN3        5
#define PINO_IN4        6
#define PINO_MOTOR_B    7

float set_point; //Angulo desejado,
float angulo_lido; //Angulo lido do sensor de 45º a 315ª no potenciometro
float erro_angulo; //Erro = setpoint - valor_atual
float last_erro;
```

```
float deltaErro; //Erro(t) - Erro(t-1)

float Kp, Ki, Kd;
float res_pid; //Saida o Controlador PID
float res_proporcional; //Saida o Controlador proporcional
float res_integral; //Saida o Controlador integral
float res_derivativo; //Saida o Controlador derivativo
float deltaT;
unsigned long actualTime;
unsigned long lastTime;

unsigned long last_print_time;

bool executando;

int counter_bt;

void setup() {
  Kp = 0;
  Ki = 0;
  Kd = 0;
  set_point =  180;
  executando = false;

  Serial.begin(115200);//Inicia a Serial
  Serial1.begin(9600);//Inicia a Serial
  pinMode(PINO_MOTOR_A, OUTPUT);
  pinMode(PINO_IN1, OUTPUT);
  pinMode(PINO_IN2, OUTPUT);
  pinMode(PINO_MOTOR_B, OUTPUT);
  pinMode(PINO_IN3, OUTPUT);
  pinMode(PINO_IN4, OUTPUT);
  status_led.init();
  setDirection(true);
}


String cmdserial;
void loop() {
  //-----------------------------------------------------
  //Ajusta Valor das constanste Kp,Ki e Kd se necessario

  recebe_comando_serial(); //Atualiza variaveis
  recebe_comando_bluetooth(); //Atualiza variaveis

  //-----------------------------------------------------
  //Realiza leitura do angulo e conversao
  angulo_lido = analogRead(ANALOG_ANGULO) * 270.0f / 1024.0f + 45.0f;; //Angulo lido de 45
ate 315

  //-----------------------------------------------------
  //Calculo do erro, deltaErro e DeltaT
  erro_angulo = angulo_lido - set_point;
  deltaErro = erro_angulo - last_erro;
```

```
  actualTime = micros();
  deltaT = (float) (actualTime - lastTime ) / 1000000.0;

  last_erro = erro_angulo;
  lastTime = actualTime;
  //-------------------------------------------------
  //Executa o controle  PID
  if (executando) {
    controle_pid(); //Executa o controle em si
  } else {
    digitalWrite(PINO_MOTOR_B, 0);
  }

  //-------------------------------------------------
  //Mostra status
  if (millis() - last_print_time >= 100) {
    last_print_time = millis();
    enviar_status();//Mostra o status do sistema
  }
  atualizar_led(); //E muda a cor de um led rgb
}

void recebe_comando_serial() {
  if (Serial.available()) {
    cmdserial = Serial.readStringUntil('\n');
    if (cmdserial.startsWith("?")) {
      mostrar_constantes_serial();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("kp")) {
      Kp = cmdserial.substring(2).toFloat();
      mostrar_constantes_serial();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("ki")) {
      Ki = cmdserial.substring(2).toFloat();
      mostrar_constantes_serial();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("kd")) {
      Kd = cmdserial.substring(2).toFloat();
      mostrar_constantes_serial();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("vai")) {
      executando = true;
      Serial.println("*********EXECUCAO INICIADA********");
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("para")) {
      executando = false;
      Serial.println("*********EXECUCAO INTERROMPIDA********");
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("set")) {
      //-------------------------------------------------
      //Realiza leitura do setPoint
      set_point = cmdserial.substring(3).toFloat();
      Serial.print("Set Point setado para: ");
```

```
      Serial.println(set_point);
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    }
  }
}

void recebe_comando_bluetooth() {
  if (Serial1.available()) {
    cmdserial = Serial1.readStringUntil('\n');
    if (cmdserial.startsWith("?")) {
      mostrar_constantes_bt();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("kp")) {
      Kp = cmdserial.substring(2).toFloat();
      mostrar_constantes_bt();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("ki")) {
      Ki = cmdserial.substring(2).toFloat();
      mostrar_constantes_bt();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("kd")) {
      Kd = cmdserial.substring(2).toFloat();
      mostrar_constantes_bt();
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("vai")) {
      executando = true;
      Serial1.println("*********EXECUCAO INICIADA********");
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("para")) {
      executando = false;
      Serial1.println("*********EXECUCAO INTERROMPIDA********");
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    } else if (cmdserial.startsWith("set")) {
      //-------------------------------------------------
      //Realiza leitura do setPoint
      set_point = cmdserial.substring(3).toFloat();
      Serial1.print("Set Point setado para: ");
      Serial1.println(set_point);
      res_proporcional = 0; res_integral = 0; res_derivativo = 0;
    }
  }
}

void mostrar_constantes_serial() {
  Serial.print("Kp setado para: ") +
  Serial.println(Kp);
  Serial.print("Ki setado para: ") +
  Serial.println(Ki);
  Serial.print("Kd setado para: ") +
  Serial.println(Kd);
}
void mostrar_constantes_bt() {
  Serial1.print("Kp setado para: ") +
  Serial1.println(Kp);
```

```
  Serial1.print("Ki setado para: ") +
  Serial1.println(Ki);
  Serial1.print("Kd setado para: ") +
  Serial1.println(Kd);
}

void atualizar_led() {
  if (abs(erro_angulo)  > 30 ) {
    status_led.acender(0xFF << 16 | 0x00 << 8);
  } else {
    uint16_t erro_porcento = (abs(erro_angulo) * 255.0f / 30.0f);
    uint8_t vermelho = erro_porcento > 255 ? 255 : erro_porcento;
    vermelho = erro_porcento < 0 ? 0 : erro_porcento;
    uint8_t verde = 255 - vermelho;
    status_led.acender(vermelho << 16 | verde << 8);
  }
}

void enviar_status() {
  Serial.println(
    String(set_point, 2) + "\t" +
    String(angulo_lido, 2) + "\t" +
    String(erro_angulo, 2)  + "\t" +
    String(res_proporcional, 2)  + "\t" +
    String(res_integral, 2)  + "\t" +
    String(res_derivativo, 2)  + "\t" +
    String(res_pid, 2)
  );
  ++counter_bt %= 5;
  if (counter_bt == 0) {
    Serial1.println(
      String(set_point, 2) + "\t" +
      String(angulo_lido, 2) + "\t" +
      String(erro_angulo, 2)  + "\t" +
      String(res_proporcional, 2)  + "\t" +
      String(res_integral, 2)  + "\t" +
      String(res_derivativo, 2)  + "\t" +
      String(res_pid, 2)
    );
  }
}

void setDirection(bool is_clockwise) {
  digitalWrite(PINO_IN1, is_clockwise);
  digitalWrite(PINO_IN2, !is_clockwise);
  digitalWrite(PINO_IN3, is_clockwise);
  digitalWrite(PINO_IN4, !is_clockwise);
}

void controle_pid() {

  //------------------------------------------------
  //Calcula as respostas P, I e D
  res_proporcional = Kp * erro_angulo ; //Multiplicaçao
```

```
  res_integral = Ki * erro_angulo * deltaT + res_integral; //Integração
  res_derivativo = Kd * deltaErro / deltaT; //Derivação

  res_pid = res_proporcional + res_integral + res_derivativo;
  res_pid = res_pid < -255 ? -255 : res_pid; //Limite minimo de -255
  res_pid = res_pid > 255 ? 255 : res_pid; //Limite maximo de 255

  //-----------------------------------------------------
  //Joga o sinal na saida do sistema
  setDirection(res_pid < 0);
  analogWrite(PINO_MOTOR_B, (uint8_t) abs(res_pid));

  //-----------------------------------------------------
  //Protecao do sistema..
  if (angulo_lido > 245 && res_pid < 0) {
    analogWrite(PINO_MOTOR_B, 0);
  }

  if (angulo_lido < 130 && res_pid > 0) {
    analogWrite(PINO_MOTOR_B, 0);
  }
}
```