Relatório do Projeto Final de Controle 2

Cristóvão Diniz Trevisan

Engenharia Eletrônica, UTFPR, Curitiba, dezembro de 2015.

1 Introdução

Este trabalho tem por objetivo implementar um sistema controlado digitalmente. O conjunto escolhido é o de um pendulo com um ventilador (motor com hélice) na extremidade, sendo que o fator a ser controlado é o ângulo do pendulo (medido usando um potenciômetro acoplado ao eixo de rotação). Este sistema pode ser visto na Figura 1.

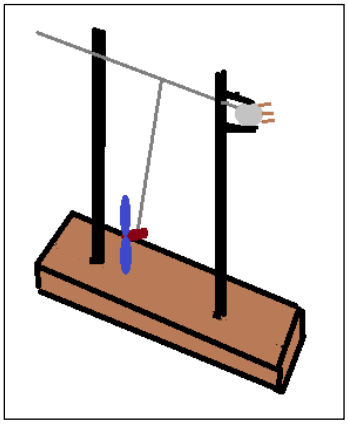

Figura 1: Imagem geral do sistema.

O motor utilizado foi o do quadcóptero Hubsan X4 H107, e depois de ligado a tensão que gera o ângulo de 90 graus é aproximadamente 5 V.

2 Modelo do sistema

Primeiramente o sistema mecânico do sistema foi modelado, para isso foi usado o diagrama da Figura 2, onde Fy é a componente perpendicular ao eixo da gravidade (m.g).

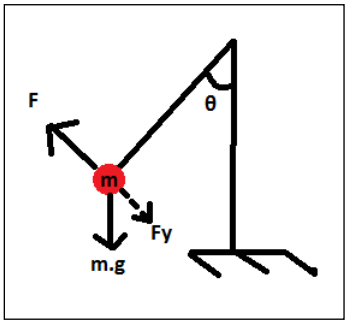

Figura 2: Diagrama de Forças.

Utilizando o digrama é possível obter a transferência entre a força do motor e o ângulo do eixo, dados pela Fórmula 1. Sendo esse sistema uma constante ele pode ser representado por

um ganho, tendo o cuidado de usar o seno da saída do sensor (potenciômetro).

$$\frac{F}{sen(\theta)} = m.g \tag{1}$$

O modelo do motor também deve ser considerado, para isso o sistema foi ligado utilizando um degrau de tensão média (4 V) e a saída do sistema medida no osciloscópio. O resultado, depois de passar por uma função seno, pode ser visto na Figura 3.

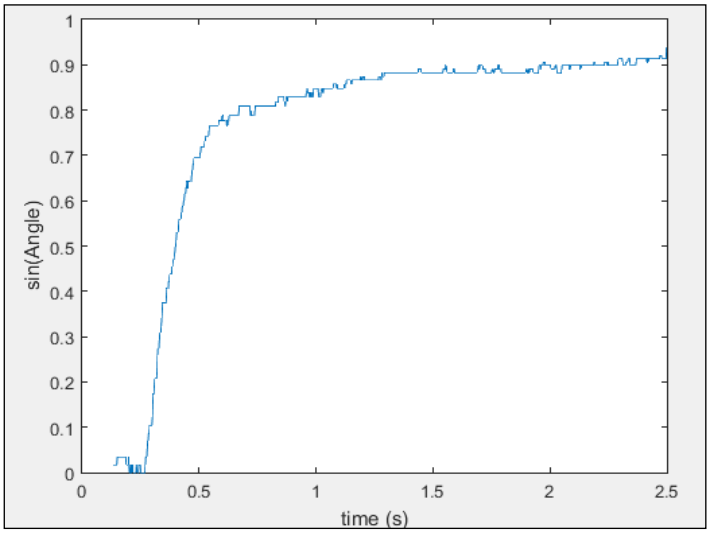

Figura 3: Saída para Entrada Degrau

Este gráfico pode ser associado com um sistema de primeira ordem, onde a equação de transferência pode ser associada com a Fórmula 2, onde "a" é o inverso do tempo onde a resposta atinge 63% do valor final (0,2 s nesse caso).

$$G(s) = \frac{max.a}{s+a} = \frac{0{,}9.5}{s+5} = \frac{4{,}5}{s+5} \tag{2}$$

A equação acima pode ser verificada utilizando a função "step" do Matlab, tendo o resultado da Figura 4, que é muito parecido com a medição.

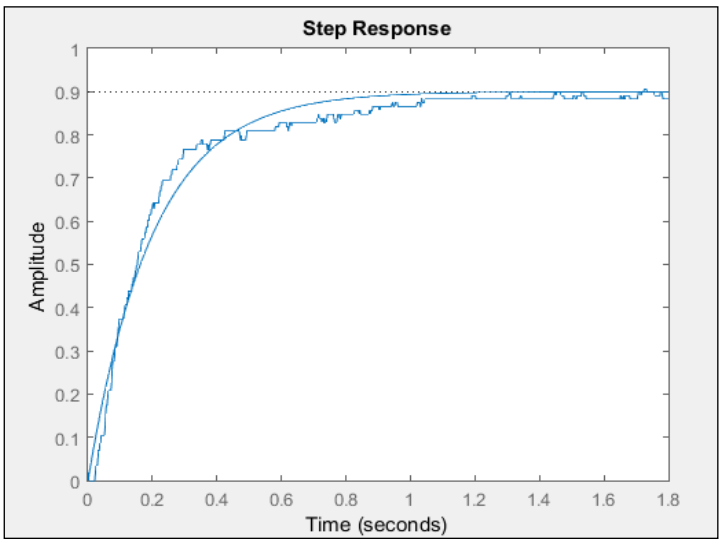

Figura 4: Comparação da equação teórica com resposta medida

3 Implementação

O controlador utilizado é um Arduino Uno, sendo que a correte do motor e fornecida por uma fonte de tensão variável ligada ao circuito integrado L393d (driver de corrente para pwm). O objetivo do controlador é tornar o sistema mais rápido e anular o erro em regime, por isso o escolhido é um controlador do tipo PID (o código pode ser visto no Apêndice A).

Um problema prático para escolher os ganhos deste controlador é que o máximo de tensão fornecida gera o ângulo de 90 graus, assim quanto maior o ângulo maior será o tempo de subida do sistema. Como este valor não passa deste máximo (isso se dá ao fato de que o sistema não tem sensor de fim de curso, mas também não pode passar de 90 graus) outros problemas surgem: os ganhos devem ser pequenos para não ocorrer overflow.

Depois de alguns testes com ajuda da simulação do Matlab vista na Figura 5 (apesar da representação matemática não ter sido tão útil, pois o ventilador tem um resposta não trivial), ganhos satisfatórios foram obtidos.

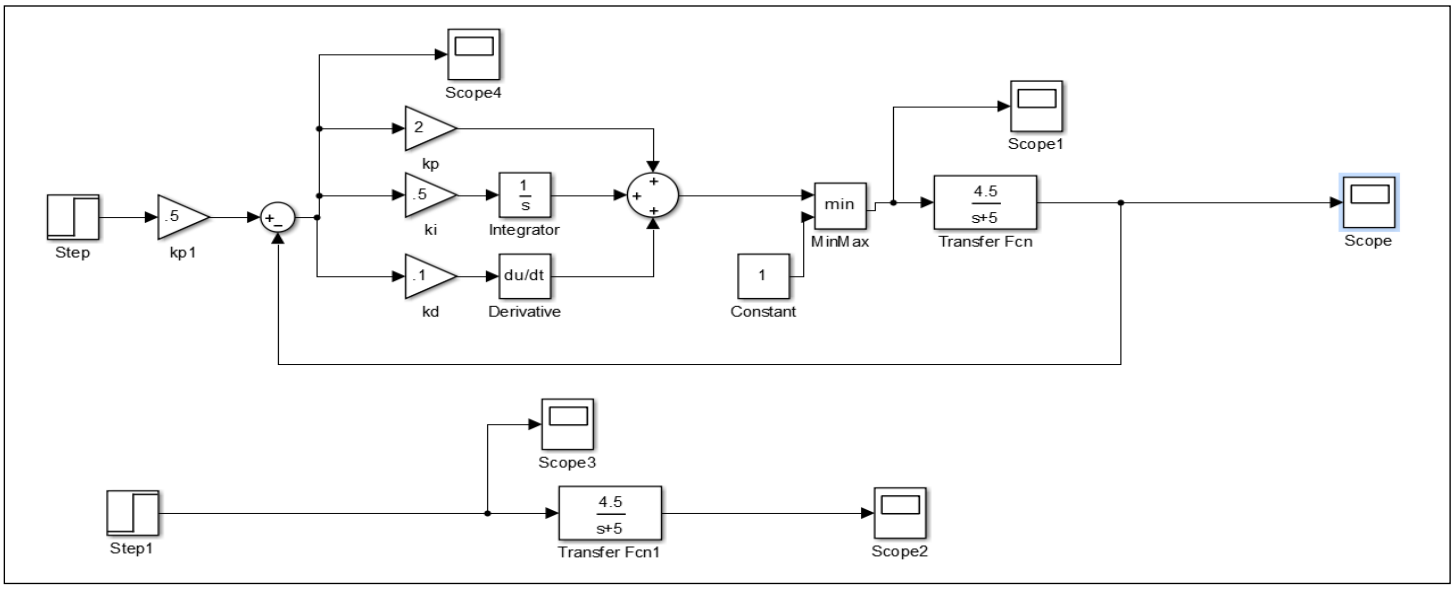

Figura 5: Sistema no Matlab (Simulink)

4 Resultados

Montado o sistema final com o controlador foram então medidos o erro e a saída do sensor (ângulo), utilizando a serial do controlador para uma resposta ao degrau, onde o ângulo objetivo é mediano (60 graus). O erro, que pode ser visto na Figura 6, e é nulo em regime (por causa do fator integrador do controlador), como se desejava no início do projeto.

Na Figura 7 pode ser visto o ângulo em função do tempo para a resposta ao impulso e como realmente ficou mais rápido que o sistema sem controle (tempo de subida é de aproximadamente 0,4 s).

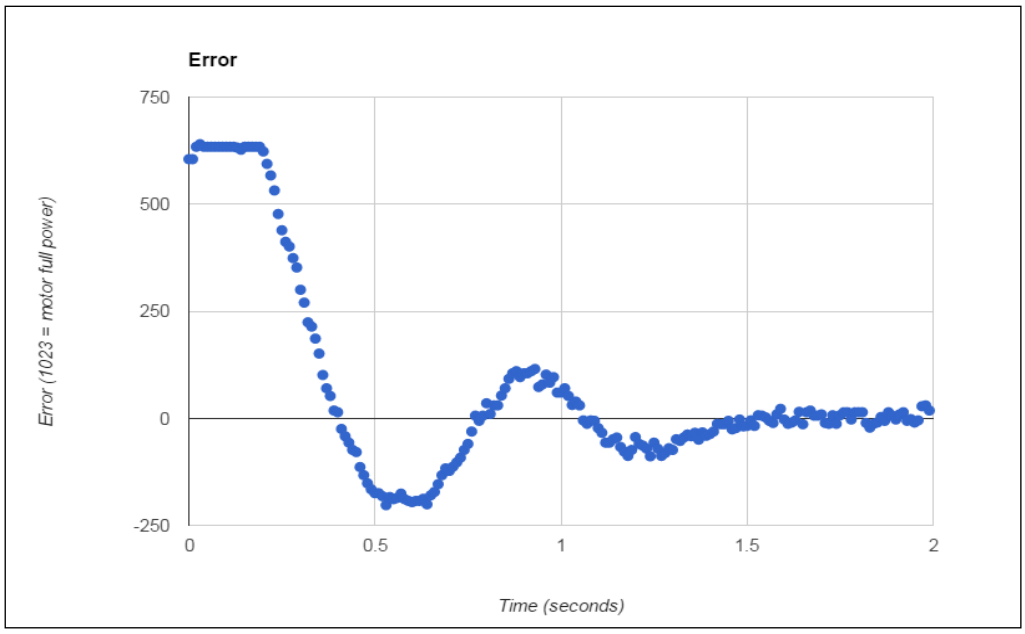

Figura 6: Erro da resposta ao degrau

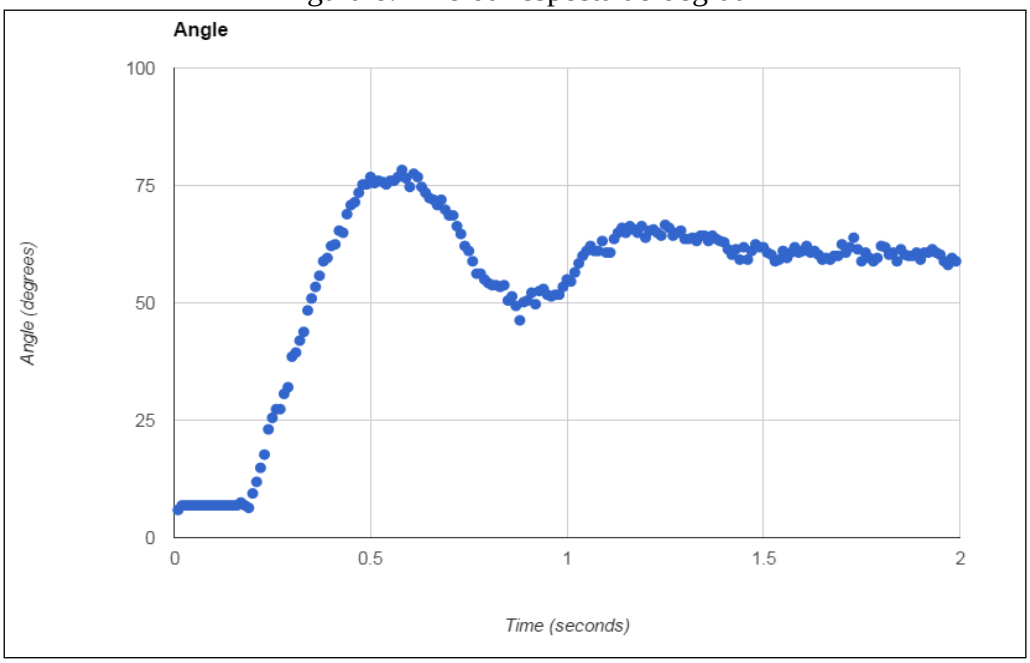

Figura 7: Ângulo da resposta ao degrau

5 Conclusões

O sistema foi controlado com sucesso, mostrando-se difícil implementar tudo perfeitamente pois o sensor e valores de controle do motor apresentam limitações (máximos e resolução), além de que o sistema controlado não se comportou como sendo o de primeira ordem medido (segundo a resposta ao degrau em malha aberta). Também foi visto quão útil é um controlador PID, pois pode ser ajustado de maneira a satisfazer os requisitos desejáveis, mesmo nas adversidades do projeto.

6 Apêndices

5.1 Apêndice A – Código do controlador

```
#include <avr/pgmspace.h>
#define MAX 1023
#define MIN 0
#define MAX_OUT 255
#define MAX_TIME 4294967296

int yPin = 0, setPointPin = 1, pwmPin = 9;
double lastError = 0, inputNow, output, error, integralPartial = 0, derivativePartial;
double kp = 2, ki = .05, kd = .2;
unsigned long startTime, Ts = 10;
int setPoint, readInt;

// this LUT calculates the sin of an angle from (0 input - > 0 degrees) to (1023 input -> 90 degrees)
const int sin_lut[1024] PROGMEM =
{0,2,3,5,6,8,9,11,13,14,16,17,19,20,22,24,25,27,28,30,31,33,35,36,38,39,41,42,44,46,47,49,50,52,53,
55,57,58,60,61,63,64,66,67,69,71,72,74,75,77,78,80,82,83,85,86,88,89,91,93,94,96,97,99,100,102,1
03,105,107,108,110,111,113,114,116,118,119,121,122,124,125,127,128,130,132,133,135,136,138,
139,141,142,144,146,147,149,150,152,153,155,156,158,160,161,163,164,166,167,169,170,172,17
4,175,177,178,180,181,183,184,186,187,189,191,192,194,195,197,198,200,201,203,204,206,207,2
09,211,212,214,215,217,218,220,221,223,224,226,227,229,230,232,234,235,237,238,240,241,243,
244,246,247,249,250,252,253,255,256,258,259,261,262,264,266,267,269,270,272,273,275,276,27
8,279,281,282,284,285,287,288,290,291,293,294,296,297,299,300,302,303,305,306,308,309,311,3
12,314,315,317,318,320,321,323,324,326,327,329,330,332,333,335,336,338,339,341,342,343,345,
346,348,349,351,352,354,355,357,358,360,361,363,364,366,367,369,370,371,373,374,376,377,37
9,380,382,383,385,386,387,389,390,392,393,395,396,398,399,401,402,403,405,406,408,409,411,4
12,414,415,416,418,419,421,422,424,425,426,428,429,431,432,434,435,436,438,439,441,442,443,
445,446,448,449,451,452,453,455,456,458,459,460,462,463,465,466,467,469,470,472,473,474,47
6,477,479,480,481,483,484,485,487,488,490,491,492,494,495,496,498,499,501,502,503,505,506,5
07,509,510,511,513,514,516,517,518,520,521,522,524,525,526,528,529,530,532,533,534,536,537,
538,540,541,542,544,545,546,548,549,550,552,553,554,556,557,558,560,561,562,564,565,566,56
8,569,570,571,573,574,575,577,578,579,581,582,583,584,586,587,588,590,591,592,593,595,596,5
97,599,600,601,602,604,605,606,607,609,610,611,612,614,615,616,617,619,620,621,622,624,625,
626,627,629,630,631,632,634,635,636,637,639,640,641,642,643,645,646,647,648,650,651,652,65
3,654,656,657,658,659,660,662,663,664,665,666,668,669,670,671,672,673,675,676,677,678,679,6
81,682,683,684,685,686,688,689,690,691,692,693,694,696,697,698,699,700,701,703,704,705,706,
707,708,709,710,712,713,714,715,716,717,718,719,721,722,723,724,725,726,727,728,729,731,73
2,733,734,735,736,737,738,739,740,741,743,744,745,746,747,748,749,750,751,752,753,754,755,7
56,757,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,
780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,80
2,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,818,819,820,821,822,823,8
24,825,826,827,828,829,830,831,831,832,833,834,835,836,837,838,839,840,841,841,842,843,844,
845,846,847,848,849,849,850,851,852,853,854,855,855,856,857,858,859,860,861,861,862,863,86
4,865,866,866,867,868,869,870,871,871,872,873,874,875,876,876,877,878,879,880,880,881,882,8
83,884,884,885,886,887,888,888,889,890,891,891,892,893,894,894,895,896,897,897,898,899,900,
```

```
900,901,902,903,903,904,905,906,906,907,908,909,909,910,911,911,912,913,914,914,915,916,91
6,917,918,918,919,920,921,921,922,923,923,924,925,925,926,927,927,928,929,929,930,931,931,9
32,933,933,934,934,935,936,936,937,938,938,939,939,940,941,941,942,943,943,944,944,945,946,
946,947,947,948,949,949,950,950,951,951,952,953,953,954,954,955,955,956,957,957,958,958,95
9,959,960,960,961,961,962,963,963,964,964,965,965,966,966,967,967,968,968,969,969,970,970,9
71,971,972,972,973,973,974,974,975,975,976,976,977,977,977,978,978,979,979,980,980,981,981,
982,982,982,983,983,984,984,985,985,985,986,986,987,987,988,988,988,989,989,990,990,990,99
1,991,992,992,992,993,993,993,994,994,995,995,995,996,996,996,997,997,997,998,998,998,999,9
99,999,1000,1000,1000,1001,1001,1001,1002,1002,1002,1003,1003,1003,1004,1004,1004,1005,1
005,1005,1005,1006,1006,1006,1007,1007,1007,1007,1008,1008,1008,1008,1009,1009,1009,100
9,1010,1010,1010,1010,1011,1011,1011,1011,1012,1012,1012,1012,1013,1013,1013,1013,1013,1
014,1014,1014,1014,1015,1015,1015,1015,1015,1015,1016,1016,1016,1016,1016,1017,1017,101
7,1017,1017,1017,1018,1018,1018,1018,1018,1018,1019,1019,1019,1019,1019,1019,1019,1019,1
020,1020,1020,1020,1020,1020,1020,1020,1021,1021,1021,1021,1021,1021,1021,1021,1021,102
1,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1022,1023,1023,1
023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,1023,102
3,1023};

int values[200], count = 0;

void setup() {
  Serial.begin(9600);
  pinMode(pwmPin, OUTPUT);
  pinMode(yPin, INPUT);
}

void loop() {

  startTime = millis();

  // read the input
  setPoint = analogRead(setPointPin);
  // calculete the sin of the input
  setPoint = pgm_read_word_near(sin_lut+setPoint);

  // read the sensor and set it to wanted the range (*3.6)
  readInt = fixRange(analogRead(yPin)*3.6); // 1023 = 90 degrees
  // calculate the sin of the sensor
  inputNow = pgm_read_word_near(sin_lut+readInt);

  // calculate the error
  error = setPoint - inputNow;

  // calculate integral part
  integralPartial += error;
```

```
  // calculate derivative part
  derivativePartial = error - lastError;

  // output to motor is the sum of parts
  output = kp * error + ki*integralPartial + kd*derivativePartial;

  // write output to pwm
  analogWrite(pwmPin, fixRangeOutput(output/4));

  // save error for derivative part
  lastError  = error;

  // check for error in sample time
  if (millis() -startTime > Ts) {
    Serial.println("too much fs");
  }

  if (count < 200){
    values[count] = inputNow;
    count++;
  }
  else if (count == 200){
    for (int i=0; i<200; i++){
      Serial.print(values[i]);
      Serial.print(", ");
    }
    Serial.println();
    count++;
  }

  startTime = millis();
  // wait to execute next step
  delay(Ts - timeDiff(millis(), startTime));


}

double fixRange(double a){
  if (a>MAX) return MAX;
  if (a<MIN) return MIN;
  return a;
}

double fixRangeOutput(double a){
  if (a>MAX_OUT) return MAX_OUT;
```

```
  if (a<0) return 0;
  return a;
}

unsigned long timeDiff(unsigned long now, unsigned long before){
  if (now>=before) return now - before;
  else return now + MAX_TIME -before;
}
```

Referências

[1] Página de Referencias do Arduino, https://www.arduino.cc/en/Reference/HomePage, acesso em 16/12/2015.

[2] Controle PID em embarcados, http://www.embarcados.com.br/controle-pid-em-sistemas-embarcados, acesso em 16/12/2015.

[3] Datasheed do L293d, http://www.ti.com/lit/ds/symlink/l293.pdf, acesso em 16/12/2015.