

Universidade Federal de São Carlos

Laboratório de Microcontroladores e Aplicações  
Professor Dr. Edilson Kato

## **Relatório 5 - Turma A**

Bruna Zamith (RA: 628093)  
Matheus Vrech (RA: 727349)

11/2018  
São Carlos - SP, Brasil

# **Conteúdo**

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Descrição</b>	<b>1</b>
<b>3</b>	<b>Materiais Utilizados</b>	<b>1</b>
3.1	Arduino . . . . .	2
3.2	Circuito Ponte H . . . . .	2
3.3	PWM . . . . .	4
3.4	PID . . . . .	4
<b>4</b>	<b>Desenvolvimento</b>	<b>5</b>
4.1	Versionamento . . . . .	5
4.2	Página Web . . . . .	5
4.3	Implementação . . . . .	7
<b>5</b>	<b>Resultados</b>	<b>13</b>

# **1 Introdução**

O presente relatório visa detalhar as atividades desenvolvidas ao longo das aulas de Laboratório de Microcontroladores e Aplicações ministradas nos dias 02/11, 16/10 e 23/10 pelo professor Dr. Edilson Kato no Departamento de Computação da Universidade Federal de São Carlos.

O documento está organizado da seguinte forma: A Seção 2 descreve o projeto proposto e seus objetivos principais; a Seção 3 expõe os materiais utilizados para a implementação do projeto; a Seção 4 detalha o seu desenvolvimento; por fim, a Seção 5 expõe e discute os resultados obtidos.

# **2 Descrição**

O projeto a ser detalhado neste relatório objetiva o aprofundamento do conhecimento em motores, Arduino e demais tecnologias.

A proposta era utilizar uma página Web e um Arduino para controlar por PID um motor de corrente contínua. Para tal, tivemos que montar um circuito da ponte H para controlar a corrente e a tensão no motor. Ainda, a velocidade e o sentido de rotação do motor foi feita através de sinais PWM.

Os controles foram feitos através de:

1. Um botão “Reverse” na página Web, muda-se o sentido de rotação do motor;
2. Um botão “Fast” na página Web, faz com que o motor gire na velocidade máxima;
3. Um botão “Medium” na página Web, faz com que o motor gire na velocidade média;
4. Um botão “Slow” na página Web, faz com que o motor gire na velocidade mínima;
5. Um botão “Stop” na página Web, faz com que o motor desacelere e pare;

# **3 Materiais Utilizados**

Para a implementação do projeto descrito na seção anterior, foram utilizados:

- 1 Arduino Mega;
- 1 *Shield Ethernet*;
- 1 Cabo Ethernet;
- 1 Motor CC;
- 4 Transistores TIP122;
- 4 Resistores;
- 2 Capacitores 10  $\mu\text{F}$ ;
- Jumpers;
- Estanho;
- Ferro de solda;
- Placa para realizar as ligações;
- 1 Arduino *Integrated Development Environment* (IDE).

### 3.1 Arduino

O Arduino é uma plataforma *open-source* de *hardware* e *software*, capaz de ler entradas (como sensores e botões) e transformá-las em saídas (como motor e LED). Ele permite a prototipagem eletrônica de *hardware* livre e é projetado com um microcontrolador Atmel AVR<sup>1</sup>. É possível programá-lo através de sua IDE e de sua linguagem de programação própria, sendo esta última semelhante à linguagem C. A Figura 1 exibe um Arduino e a Figura 2, a IDE.

### 3.2 Circuito Ponte H

O circuito de ponte H é responsável por determinar o sentido de corrente e o valor de tensão no controle de um motor CC. Esse circuito é largamente utilizado, sendo montado com transistores e/ou relés. Ainda, existem circuitos integrados ponte H, com todo o circuito interno<sup>2</sup>. A Figura 3 exibe um circuito Ponte H.

---

<sup>1</sup><https://www.arduino.cc/en/Guide/Introduction>

<sup>2</sup><http://engenheirocaicara.com/circuito-ponte-h/>

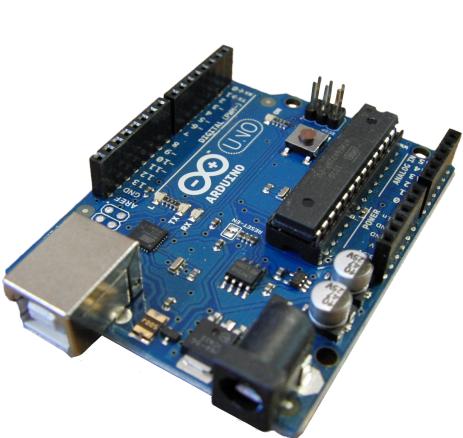


Figura 1: Arduino



Figura 2: IDE do Arduino

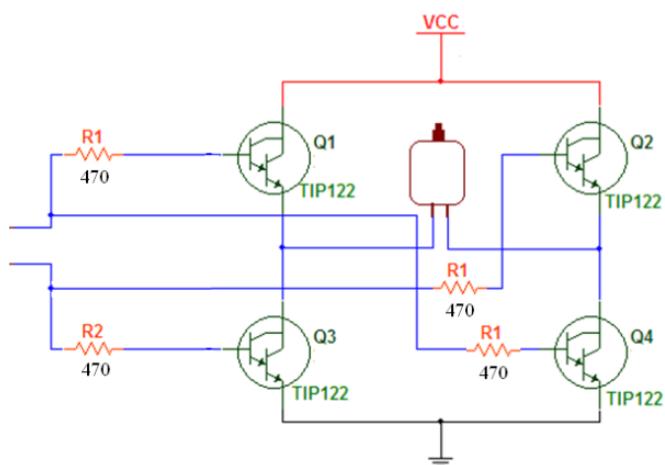


Figura 3: Ponte H. Retirado do material de apoio da disciplina.

### 3.3 PWM

O sinal PWM (do inglês, *Pulse-Width Modulation*) é um sinal digital que permite controlar a tensão de determinado dispositivo de acordo com o *duty cycle* do sinal. Uma vantagem do PWM é que o sinal permanece digital em todo o percurso (do processador até o sistema controlado) e não é necessária nenhuma conversão de digital para analógico. A Figura 4 apresenta o funcionamento de um PWM.

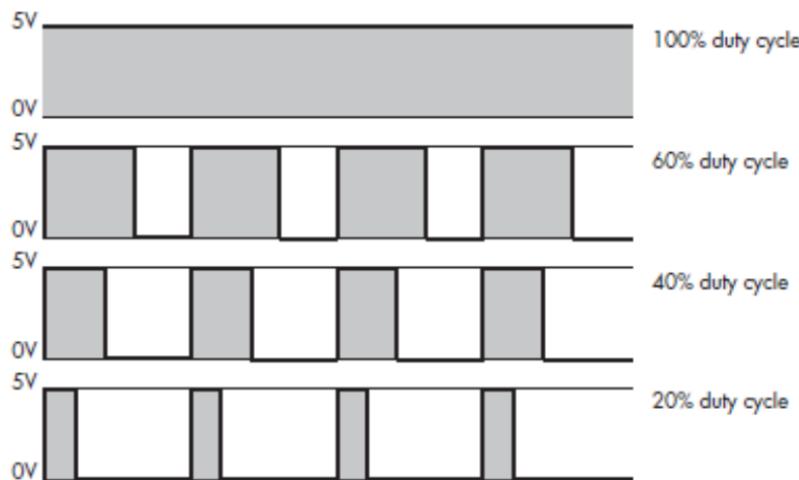


Figura 4: PWM. Retirado do material de apoio da disciplina.

### 3.4 PID

O PID funciona de acordo com 3 fatores:

- Proporcional: Objetiva a redução do tempo de resposta, e com isso afeta consequentemente a velocidade do veículo. Quanto mais longe do alvo, maior o erro, maiores são os incrementos.
- Integrativo: Trabalha em cima do erro acumulado e do erro em regime. Reduz o erro no mesmo degrau. Responsável por diminuir as oscilações quando ocorrem variação bruscas.
- Derivativo: Trabalha em cima do overshooting que ocorre a partir de variações em degrau (como por exemplo a inserção de uma nova velocidade de sistema). Leva em conta a característica da derivada de reduzir a ordem da função. Fazendo um paralelo com o conceito da estatística, poderíamos dizer

que trabalha em cima do erro instantâneo. Evita previamente que o desvio se torne maior.

Os conceitos sublinhados acima são as qualidades do sistema e são os parâmetros de desempenho fundamentais de qualquer sistema de controle. O PID é usado para alterá-los e não para avaliá-los.

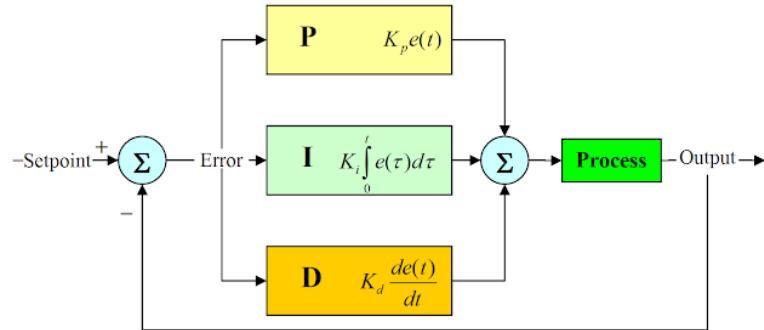


Figura 5: Esquemático de um sistema de controle PID

## 4 Desenvolvimento

### 4.1 Versionamento

O controle de versões do nosso código está sendo feito a partir do GitHub, repositório público “Zavech”<sup>3</sup>. O projeto detalhado neste relatório e todos os outros projetos da disciplina serão incluídos nesse mesmo repositório.

### 4.2 Página Web

A página Web do projeto foi desenvolvida fazendo uso de HTML e Javascript, e das tecnologias Ajax e Bootstrap. É composta de duas páginas principais: “Home”, contendo informações sobre o projeto, lista de tarefas e instruções de como conectar a placa; “Activities”, a qual possuí todas as *features* implementadas.

O site pode ser acessado a partir do link: <https://whoismath.github.io/zavech/>.

As Figuras 6 e 7 exibem as interfaces da página Web. Para correto funcionamento, é preciso apenas que o servidor e o arduino estejam conectados na mesma rede.

---

<sup>3</sup><https://github.com/whoismath/zavech>

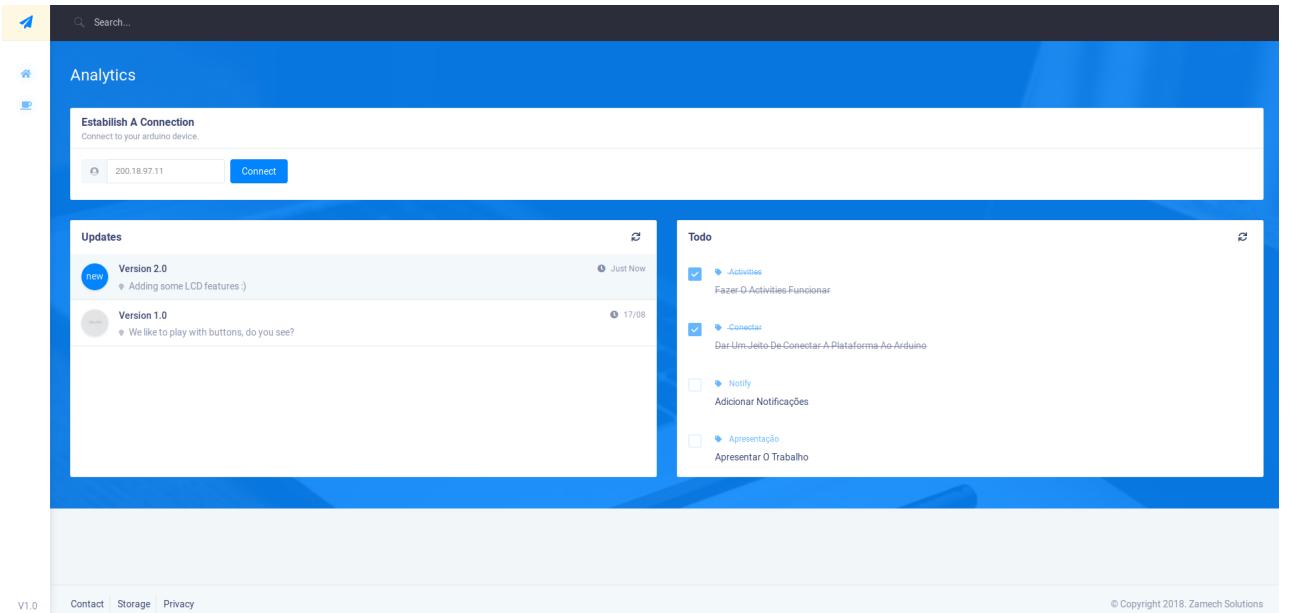


Figura 6: Página “Home”

**PID Motor**  
Run our motor [away](#)

---

**Speed:**

[Slow](#) [Medium](#) [Fast](#)

**Control:**

[Stop](#) [Reverse](#)

Figura 7: Página “Activities”

### 4.3 Implementação

As conexões podem ser visualizadas nas Figuras 8 e 9.

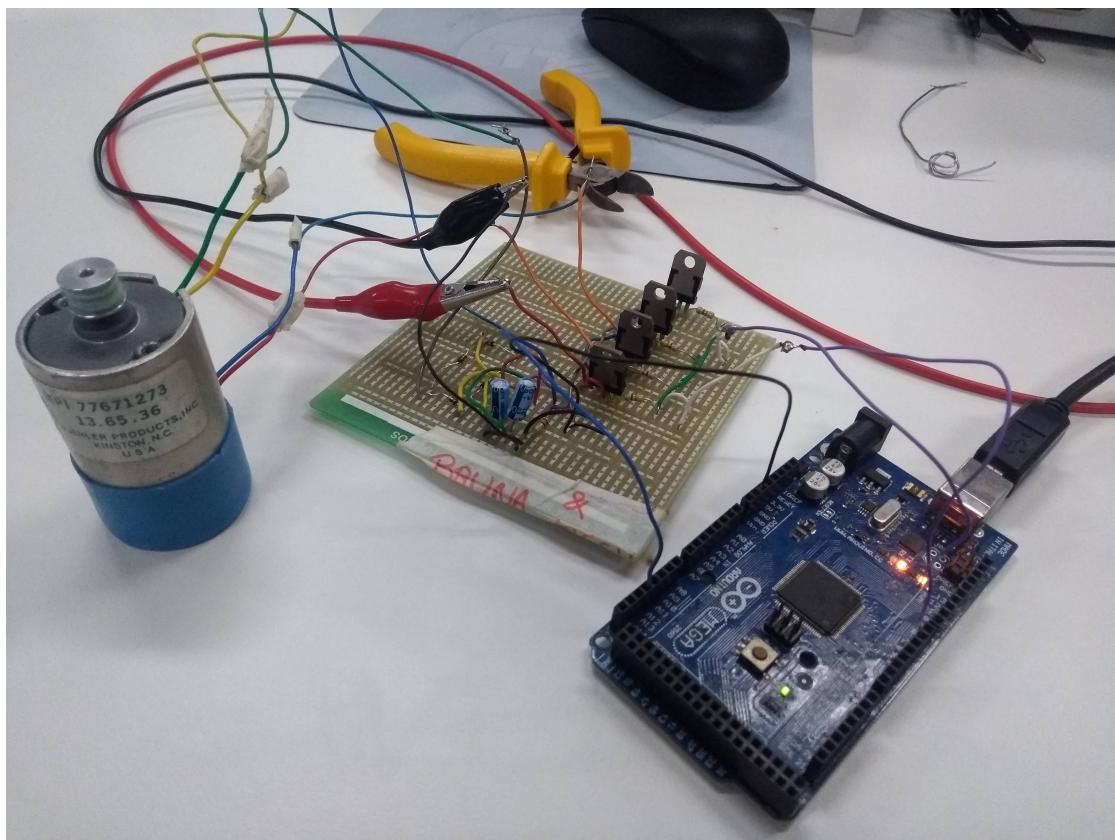


Figura 8: Circuito final

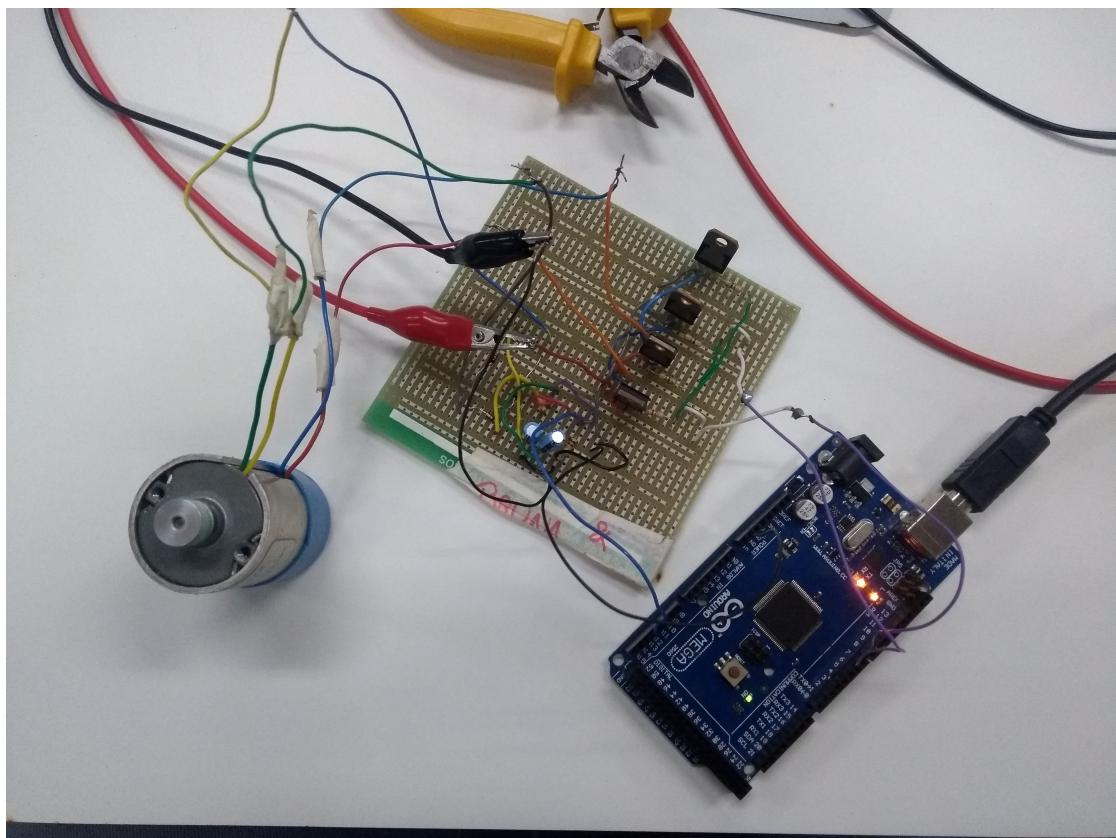


Figura 9: Circuito final

Para a programação, foram primeiramente importadas 3 diferentes bibliotecas:

- <**SPI**>: *Serial Peripheral Interface*, permite a conexão com periféricos através do protocolo SPI<sup>4</sup>;
- <**Wire**>: Permite a comunicação com dispositivos I2C/TWI, como é o caso do LCD<sup>5</sup>;
- <**Stepper**>: Permite o controle de motores de passo unipolares e bipolares<sup>6</sup>;

O código final **.ino** pode ser encontrado em <https://github.com/whoismath/zavech/blob/master/class5.ino> e também exposto a seguir.

```
1  /*
2   * Web Server
3   * A simple web server that shows the value of the analog input pins.
4   * using an Arduino Wiznet Ethernet shield.
5   * Circuit:
6   *   * Ethernet shield attached to pins 10, 11, 12, 13
7   *   * Analog inputs attached to pins Ao through A5 (optional)
8   * created 18 Dec 2009
9   * by David A. Mellis
10  * modified 9 Apr 2012
11  * by Tom Igoe
12  * modified 02 Sept 2015
13  * by Arturo Guadalupi
14  * Besides that, this project was modified by Matheus Vrech and Bruna Zamith
15  * for university classes purpose, enjoy it
16  */
17 /*
18  * Libraries and Definitions
19  *****/
20 #include <SPI.h>
21 #include <Ethernet.h>
22 #include <Wire.h>
23
24 /*
25  * Ethernet Configuration
26  *****/
```

<sup>4</sup><https://www.arduino.cc/en/Reference/SPI>

<sup>5</sup><https://www.arduino.cc/en/Reference/Wire>

<sup>6</sup><https://www.arduino.cc/en/Reference/Stepper>

```

27 byte mac[] = {
28     0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
29 };
30
31 IPAddress ip(200, 18, 97, 11);
32 EthernetServer server(80);
33
34 /*
35 * Motor Configuration
36 *****/
37 #define MOTOR1 5
38 #define MOTOR2 6
39 #define ENCODER Ao
40
41 /*
42 * Variaveis Gerais
43 *****/
44 int _direction = 1,
45     _speed = 0,
46     _selectSpeed = 0,
47     _sensorSpeed = 0,
48     _time = 0,
49     _error, _prevError,
50     KP = 1,
51     KI = 1,
52     KD = 1,
53     _integral, _derivada;
54
55 /*
56 * Setup
57 *****/
58 void setup() {
59     Serial.begin(9600);
60     while (!Serial) {}
61     Ethernet.begin(mac, ip);
62     server.begin();
63     Serial.print("server is at ");
64     Serial.println(Ethernet.localIP());
65
66     pinMode(MOTOR1, OUTPUT);
67     pinMode(MOTOR2, OUTPUT);
68 }

```

```

69
70  /*
71   * Loop
72   * ****
73 void loop() {
74     _time = millis();
75
76     EthernetClient client = server.available();
77     /* Get the http packet */
78     String readString;
79     if (client) {
80       boolean currentLineIsBlank = true;
81       while (client.connected()) {
82         if (client.available()) {
83           char c = client.read();
84
85           Serial.write(c);
86
87           /* Store the entire packet */
88           if (readString.length() < 100) {
89             readString += c;
90           }
91
92           /* Receive GET commands */
93           /* working with our motor right now */
94           else if (readString.indexOf("pid_slow") > 0) {
95             _selectSpeed = 160;
96             client.println("HTTP/1.1 200 OK");
97             break;
98           }
99           else if (readString.indexOf("pid_medium") > 0) {
100             _selectSpeed = 200;
101             client.println("HTTP/1.1 200 OK");
102             break;
103           }
104           else if (readString.indexOf("pid_fast") > 0) {
105             _selectSpeed = 255;
106             client.println("HTTP/1.1 200 OK");
107             break;
108           }
109           else if (readString.indexOf("pid_stop") > 0) {
110             _selectSpeed = 0;
111           }
112         }
113       }
114     }
115   }

```

```

111     client.println("HTTP/1.1 200 OK");
112     break;
113   }
114   else if (readString.indexOf("pid_reverse") > 0) {
115     _direction = _direction ? -1 : 1;
116     client.println("HTTP/1.1 200 OK");
117     break;
118   }
119   /* end of motor functions */
120   if (c == '\n' && currentLineIsBlank) {
121     client.println("HTTP/1.1 200 OK");
122     client.println("Content-Type: text/html");
123     client.println("Connection: keep-alive");
124     //client.println("Refresh: 5"); // refresh the page
125     automatically every 5 sec
126     client.println();
127     client.println("<!DOCTYPE HTML>");
128     client.println("<html>");
129     client.println("<head></head>");
130     /* easter egg for ninjas */
131     client.println("<body>i'm a happy arduino , and you found my
easter egg!<br>curiosity will kill you anyway.</body>");
132     client.println("</html>");
133   }
134   if (c == '\n') {
135     currentLineIsBlank = true;
136   } else if (c != '\r') {
137     currentLineIsBlank = false;
138   }
139 }

140 /* wait for something: nothing at all*/
141 delay(1);
142 //client.stop();
143 //Serial.println("client disconnected");
144 }
145 _sensorSpeed = analogRead(ENCODER);
146 _sensorSpeed = map(_sensorSpeed, 0, 45, 0, 255);
147 _prevError = _error;
148 _error = _speed - _sensorSpeed;
149 _integral = _integral + (_error*_time);

```

```

151 _derivada = (_error - _prevError)/_time;
152 KP = _error*KP;
153 KI = KI*_integral;
154 KD = KD*_derivada;
155 _speed = _selectSpeed + KP + KI + KD;
156 if(_direction == 1){
157     analogWrite(MOTOR1,_speed);
158     analogWrite(MOTOR2,0);
159 }
160 else if(_direction == -1){
161     analogWrite(MOTOR1,0);
162     analogWrite(MOTOR2,_speed);
163 }
164
165 _time = millis() - _time;
166 }
```

## 5 Resultados

Todos os resultados obtidos se mostraram satisfatórios, sendo possível implementar e testar as funcionalidades propostas, incrementais às implementações previamente desenvolvidas.