



**UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
LABORAT. DE APLIC. COM MICROCOMPUTADORES**

**PROJETO 2:
PONG**

Antonio Gabriel Sousa Borralho

**São Luís, MA
5 de julho de 2018**

Antonio Gabriel Sousa Borralho

PROJETO 2: PONG

Relatório referente ao segundo projeto, para obtenção da segunda nota da disciplina Laboratório de Aplic. com Microcomputadores do curso de Engenharia Elétrica da UFMA no período de 2018.1.

Prof. André Borges Cavalcante.

São Luís, MA - Brasil

5 de julho de 2018

Resumo

Este relatório é referente ao segundo projeto do Laboratório de Aplicação com Microcomputadores no período de 2018.1. Foi construído e implementado uma versão do famoso jogo Pong desenvolvido pela Atari em 1972 utilizando a linguagem C para programar, através do software Arduino IDE, o microcontrolador ATMega328p embarcado em uma placa desenvolvida para o projeto. O funcionamento do código desenvolvido foi observado na prática.

Palavras chave: Pong; Linguagem C; microcontrolador.

Sumário

| | | |
|------------|--|-----------|
| 1 | INTRODUÇÃO | 5 |
| 2 | PROCEDIMENTOS EXPERIMENTAIS | 6 |
| 2.1 | Materiais | 6 |
| 2.2 | Métodos | 7 |
| 2.2.1 | Descrição do Funcionamento | 7 |
| 2.2.2 | Simulação | 7 |
| 2.2.3 | Gravação no Microcontrolador | 7 |
| 3 | RESULTADOS E DISCUSSÕES | 8 |
| 3.1 | Fluxograma das Funções Construídas | 8 |
| 3.2 | Configuração de entradas e saídas | 14 |
| 3.3 | Loop Principal | 15 |
| 3.4 | Montagem física do projeto | 16 |
| 4 | CONCLUSÃO | 18 |
| | REFERÊNCIAS | 19 |
| | APÊNDICES | 20 |
| | APÊNDICE A – CÓDIGO FONTE | 21 |
| | APÊNDICE B – LISTA DE MATERIAIS UTILIZADOS E PREÇOS | 31 |

1 Introdução

O software Arduino (IDE) ¹ de código aberto facilita a implementação e a gravação do código em um microcontrolador. Ele é executado no Windows, Mac OS X e Linux. O ambiente é escrito em Java e baseado em linguagem C, baseado também em *Processing*, outro software de código aberto. Este software pode ser usado com qualquer placa Arduino ou que contenha um microcontrolador compatível como na maioria das vezes o ATmega328P da Microchip (utilizado neste trabalho). Ou também, os arquivos gerados pela compilação dos códigos podem ser carregados para o microcontrolador através de um comunicador USB-ASP por exemplo.

O microcontrolador ATmega328P faz parte da popular família de microcontroladores de 8 bits CMOS baseado na arquitetura AVR[®] lançada pela ATMEL[®] ². Este microcontrolador possui alta performance relacionando à sua família, podendo executar instruções com um ciclo de *clock*, fazendo com que o mesmo alcance 1 Milhão de Instruções por Segundo por Mega Hertz, possibilitando ao programador otimizar o projeto combinando consumo de potência versus velocidade de processamento.

A técnica mais importante no projeto da lógica de programas baseada em algoritmos denomina-se programação estruturada ou programação modular (MANZANO, 2010). Fluxogramas são formas gráficas utilizadas para representar uma sequência de passos a serem executados. O motivo da utilização de tais ilustrações é agilizar a codificação da escrita e da programação, facilitar a depuração da leitura, permitira verificação de possíveis falhas apresentadas pelos programas, e facilitar as alterações e atualizações dos programas, bem como o processo de manutenção contribuindo para a fácil leitura de quem tem a necessidade de compreender o código implementado por outra pessoa.

¹ <https://www.arduino.cc/en/Main/Software>

² <https://www.microchip.com/wwwproducts/en/ATmega328P>

2 Procedimentos Experimentais

2.1 Materiais

Para montagem do referido experimento, utilizou-se os seguintes materiais:

- 1 - Software *Arduino IDE*;
- 2 - Módulo com o Microcontrolador *ATMega 328P* da Atmel®;
- 3 - Módulo Pong;
- 4 - Módulo Regulador de Tensão;
- 5 - Matriz de Contatos;
- 6 - Piezoelétrico;
- 7 - Bateria 9V;
- 8 - Gravador USB-ASP (Protocolo ISP);
- 9 - DIP Switch de 4 posições;

A seguir temos as imagens dos módulos feitos para a implementação do projeto:

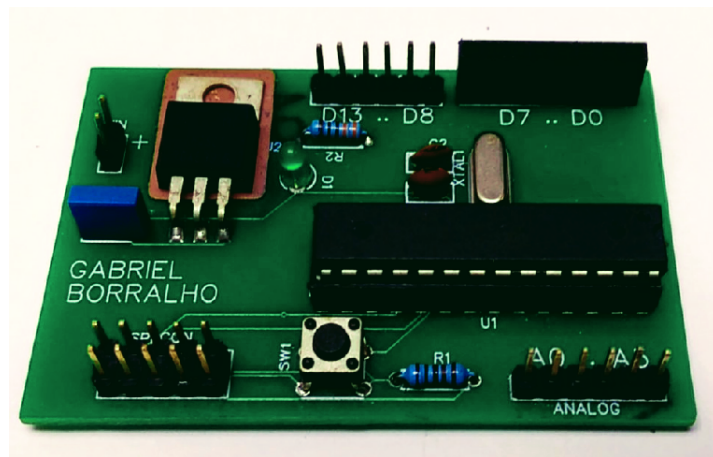


Figura 1 – Módulo com o Microcontrolador *ATMega 328P*

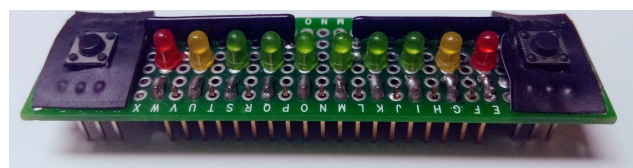


Figura 2 – Módulo Pong

2.2 Métodos

2.2.1 Descrição do Funcionamento

Para rebater a bola, foram utilizados botões (push-button). A bola foi representada por apenas um LED aceso que em sequência deu a ilusão de uma bola em movimento. O campo foi representado por uma sequência de 8 LEDs alinhados. Quando o botão havia sido pressionado na posição correta de rebatimento (LED amarelo), a sequência mudava de sentido em direção ao oponente. A mudança de posição da bola foi representada pelo acendimento e apagamento dos LED nas direções apropriadas. Caso o jogador rebatesse uma bola que estivesse a caminho, todos os LED eram apagados indicando que o jogador oposto marcou um ponto da mesma forma que se o jogador deixasse de rebater a tempo. Além de que o sistema mostrava a pontuação ao fim de cada partida onde cada rodada era completada ao se atingir três pontos, quando isso ocorria uma comemoração indicava o vencedor. Um sistema de seleção de dificuldade foi implementado utilizando um DIP Switch de quatro posições, o sistema permitia ao usuário selecionar dentro de 16 dificuldades diferentes por meio de combinações binárias de 4 Bits, ou seja, $2^4 = 16$ possibilidades, onde uma delas definia um modo automático de incremento de dificuldade deixando todas as posições do DIP SWitch desativadas (0000).

2.2.2 Simulação

Para facilitar a montagem futura e evitar possíveis erros humanos cometidos optou-se pela simulação do projeto em uma plataforma online (Tinkercad da Adobe¹) onde foi possível montar o projeto com o uso de uma matriz de contatos para conexão dos componentes por meio de ligações que simulavam cabos. Além de ser possível testar passo a passo (*debugger*) a lógica que se pretendia-se implementar.

2.2.3 Gravação no Microcontrolador

Todo o código foi implementado no software *Arduíno IDE*. Neste *software* é possível gerar um arquivo .hex que contém o código de máquina correspondente ao programa implementado em Linguagem C. Com o auxílio do *software ProgISP*² foi possível gravar o código no microcontrolador através do gravador USB-ASP que se comunica por meio do protocolo ISP.

¹ <https://www.tinkercad.com/>

² <http://paginapessoal.utfpr.edu.br/demantov/microcontr.-1-el07d-t12/10-gravador-usbasp-phd/progisp172.rar>

3 Resultados e Discussões

3.1 Fluxograma das Funções Construídas

Utilizando o conceito de modularidade ([MANZANO, 2010](#)) pode-se representar através de fluxogramas as funções implementadas no projeto.

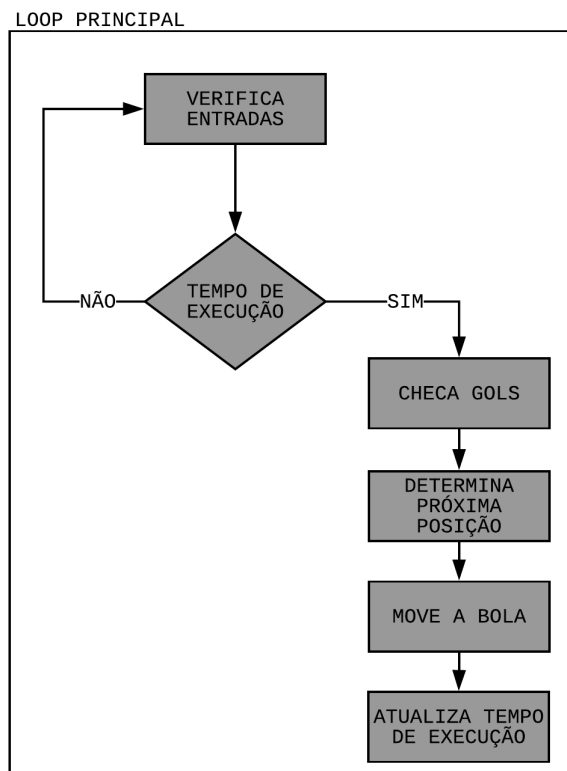


Figura 3 – Função Loop Principal

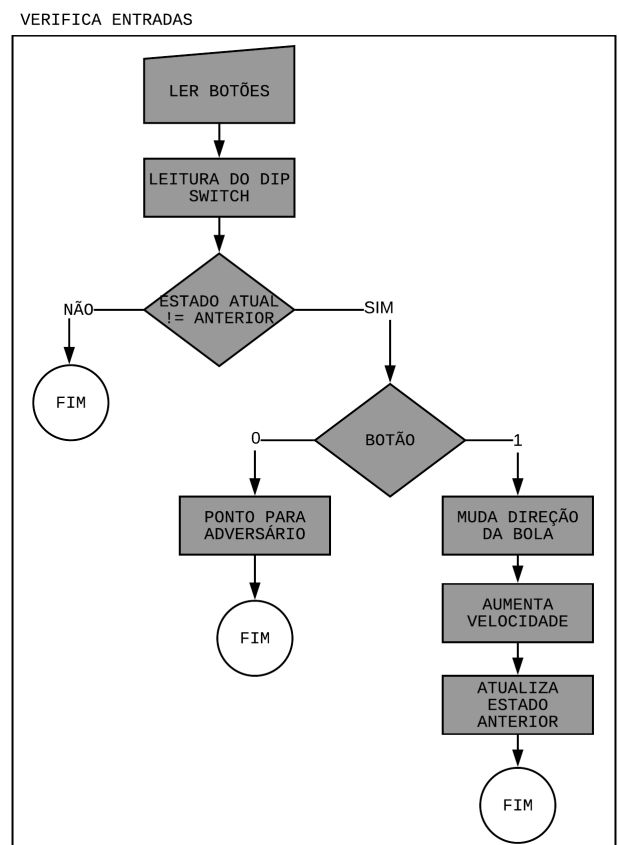


Figura 3 – Função Verifica Entradas

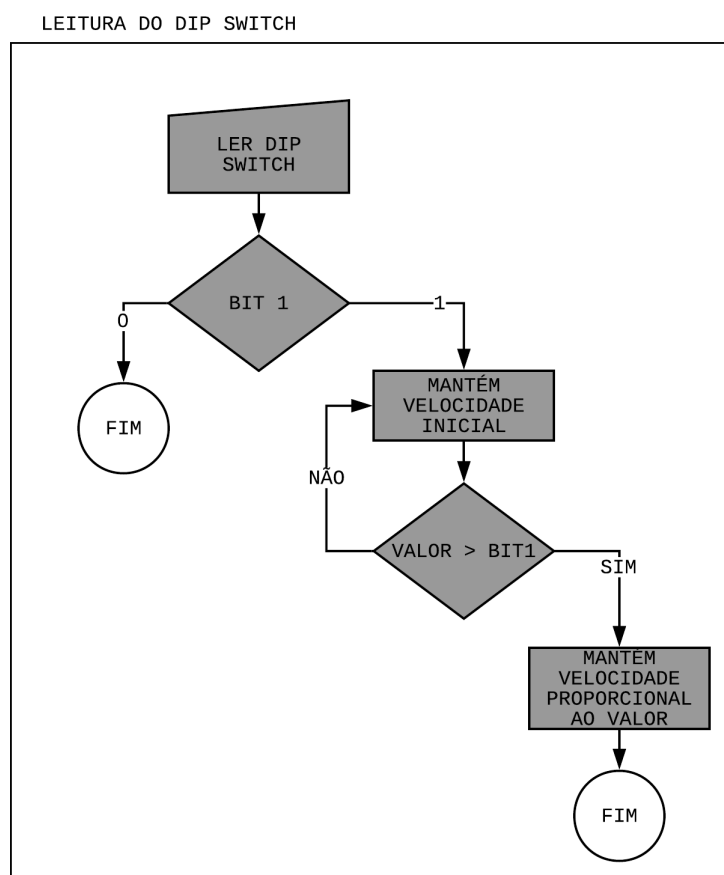


Figura 4 – Função Leitura do Dip Switch

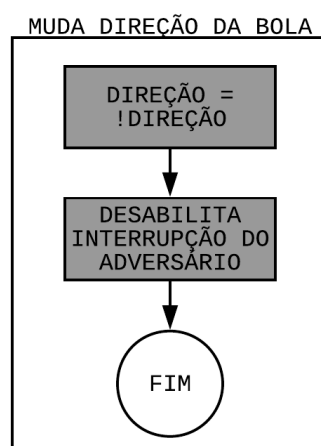


Figura 5 – Função Muda Direção da Bola

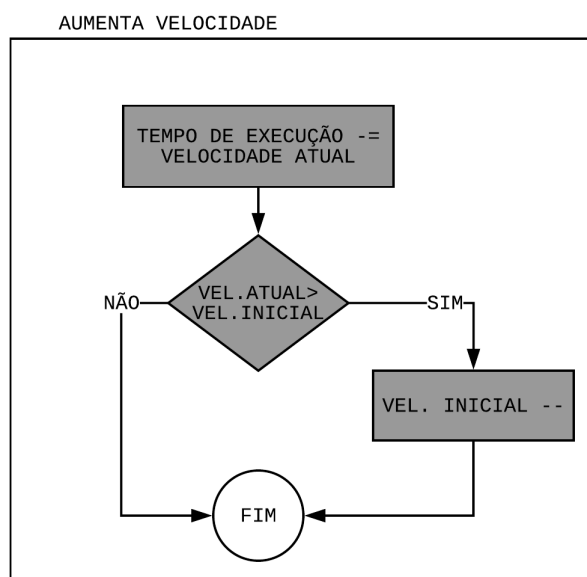


Figura 6 – Função Aumenta Velocidade

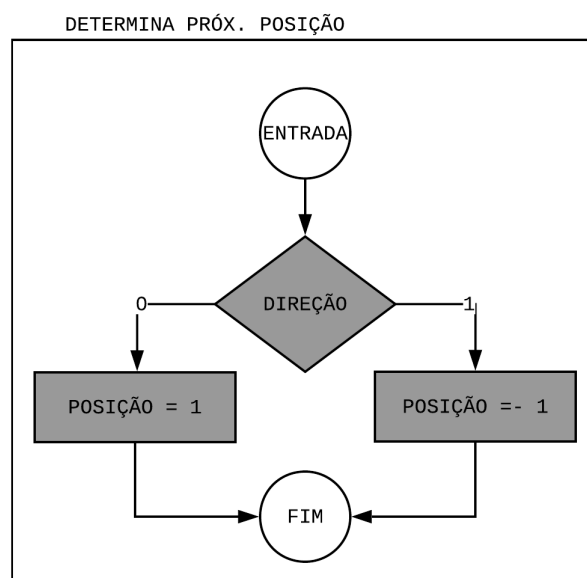


Figura 7 – Função Determina Próxima Posição

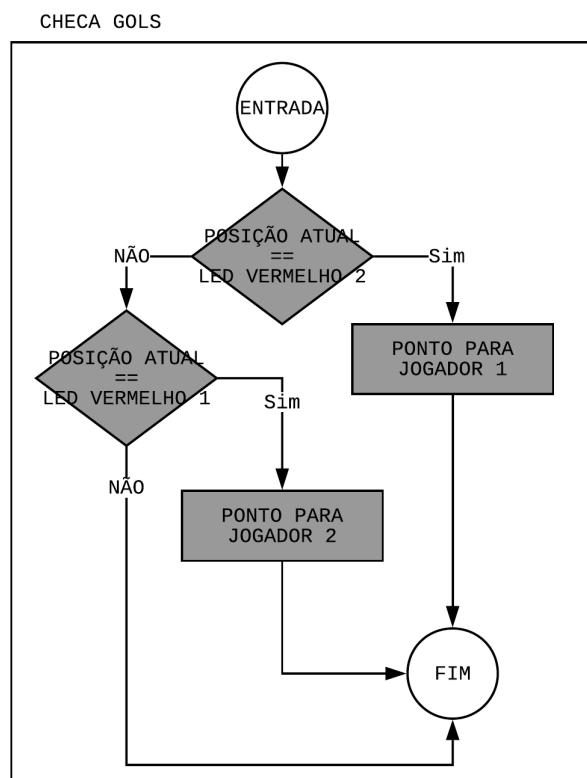


Figura 8 – Função Checa Gols

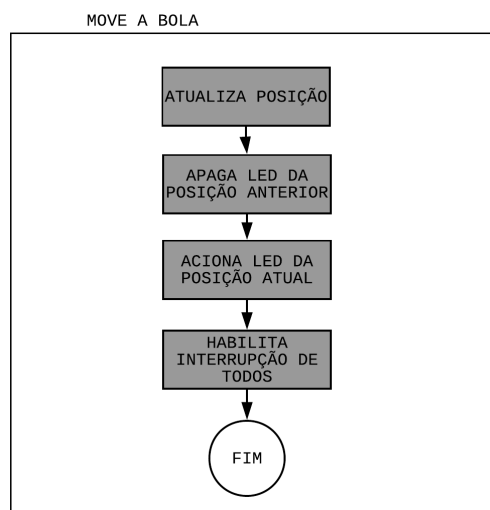


Figura 9 – Função Move a Bola

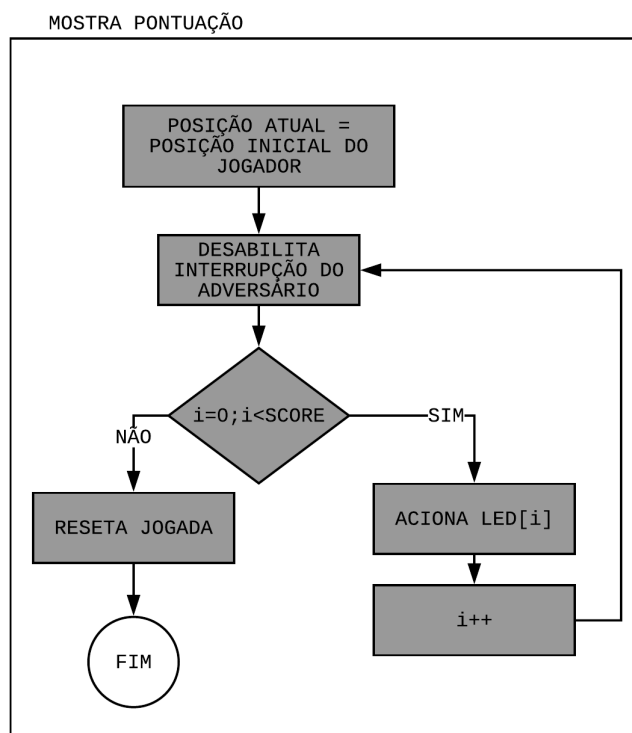


Figura 10 – Função Mostra Pontuação

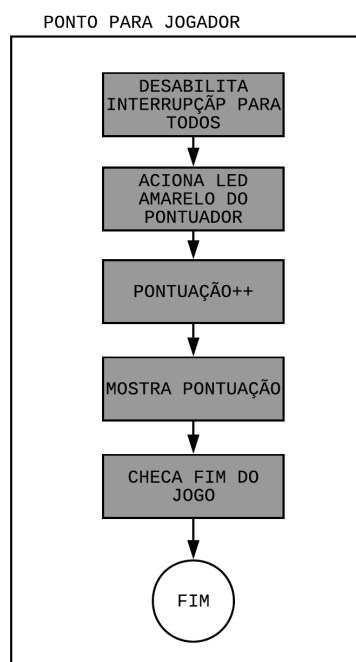


Figura 11 – Função Ponto Para Jogador

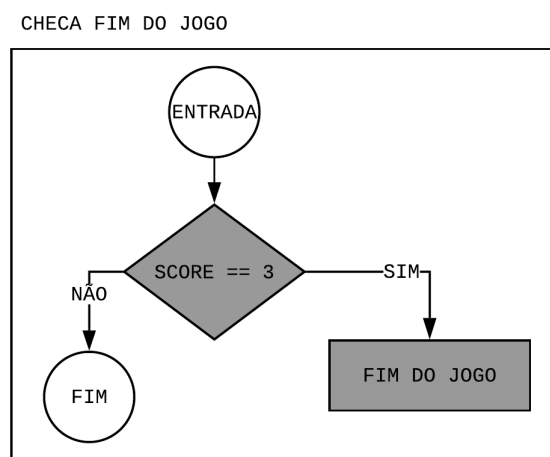


Figura 12 – Função Checa Fim do Jogo

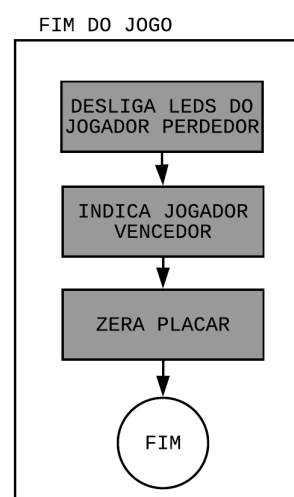


Figura 4 – Função Fim do Jogo

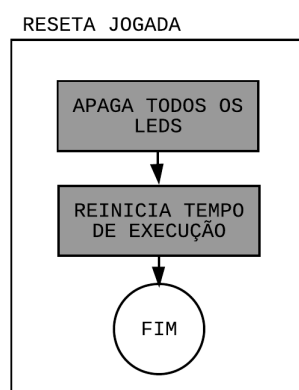


Figura 13 – Função Reseta Jogada

3.2 Configuração de entradas e saídas

A primeira etapa do código foi atribuir às portas do microcontrolador à sua determinada função (Entrada ou Saída). A função a seguir é denominada `void setup()` e ela é executada apenas uma vez antes de todo o resto do código.

Função Setup

```
1 void setup() {  
2   pinMode(Buzzer,OUTPUT);  
3   //Configura Portas 4 a 13 como SAIDAS (LEDS)  
4   for (Porta=4;Porta<=13;Porta++){  
5     pinMode(Porta, OUTPUT);  
6   }  
7   //Configura Portas 2 e 3 como ENTRADAS (botoes)  
8   pinMode(2, INPUT_PULLUP); //(Resistor PullUp ativado)  
9   pinMode(3, INPUT_PULLUP);  
10  //Cerimonia de StartGame  
11  BemVindo(30);  
12  //Aguarda Jogador 1 iniciar  
13  digitalWrite(Jogador1, HIGH);  
14  while (digitalRead(BotaoJogador1)==1){}  
15  delay(500); //Remove buffer do teclado  
16 }
```

Os LEDs são definidos como saídas digitais assim como o piezoelétrico, já os botões são entradas digitais com um resistor interno ativado. O *DIP Switch* é ligado a resistores em um circuito com resistor *Pull-UP*, onde se mantém um nível lógico zero quando uma das chaves do *DIP Switch* não está acionado, ao ser acionado um valor de tensão é enviado para um das portas analógicas do microcontrolador, fazendo com que a leitura destas chaves seja feita de forma analógica.

3.3 Loop Principal

Todo código implementado neste trabalho é baseado em uma unidade de tempo obtida através da função `millis(unsigned int time)`¹(ALANKUS, 2017). Esta função retorna o número de milissegundos desde que o microcontrolador começou a funcionar. Este número irá voltar para zero (*Overflow*), após aproximadamente 50 dias. Desta forma conseguimos ter uma referência de tempo mesmo que existam funções que levem mais tempo para ser executadas que outras. Utilizando variáveis globais é possível manipular os valores sem que os mesmos sofram com o reinício do laço principal. O código seguinte é o Loop Principal do programa, o qual é executado continuamente:

Função Loop

```
1 void loop() {  
2   VerificarEntradas();  
3   tempoAtual = millis();  
4   if (tempoAtual - tempoAnterior >= TempoPorLED) {  
5     LerDIPSW();  
6     tempoDIPSW();  
7  
8     ChecaGols();  
9     DeterminaProximaPosicao();  
10    MoveBola();  
11  
12    tempoAnterior = tempoAtual;  
13  }  
14 }
```

¹ <https://www.arduino.cc/reference/en/language/functions/time/millis/>

3.4 Montagem física do projeto

Para facilitar a montagem optou-se pelo uso de uma matriz de contatos para conexão dos componentes por meio de pequenos cabos. Para evitar falhas nas conexões, também foram desenvolvidos módulos separados com a utilização de placas de circuito impresso padrão. Um módulo contendo 10 LEDs com duas redes de resistores e dois botões. Outro módulo contendo o DIP Switch. E, outro módulo contendo o circuito de alimentação por bateria. Uma placa (Módulo com o Microcontrolador *ATMega 328P*) foi projetada por meio do aplicativo online EASYeda² e fabricada pela empresa JLCPCB³, esta placa foi baseada no circuito mais simples para que se possa ter acesso às portas de entradas e saídas digitais e analógicas, à leitura e gravação do microcontrolador ATMega328P.

O diagrama esquemático do sistema é ilustrada na figura a seguir:

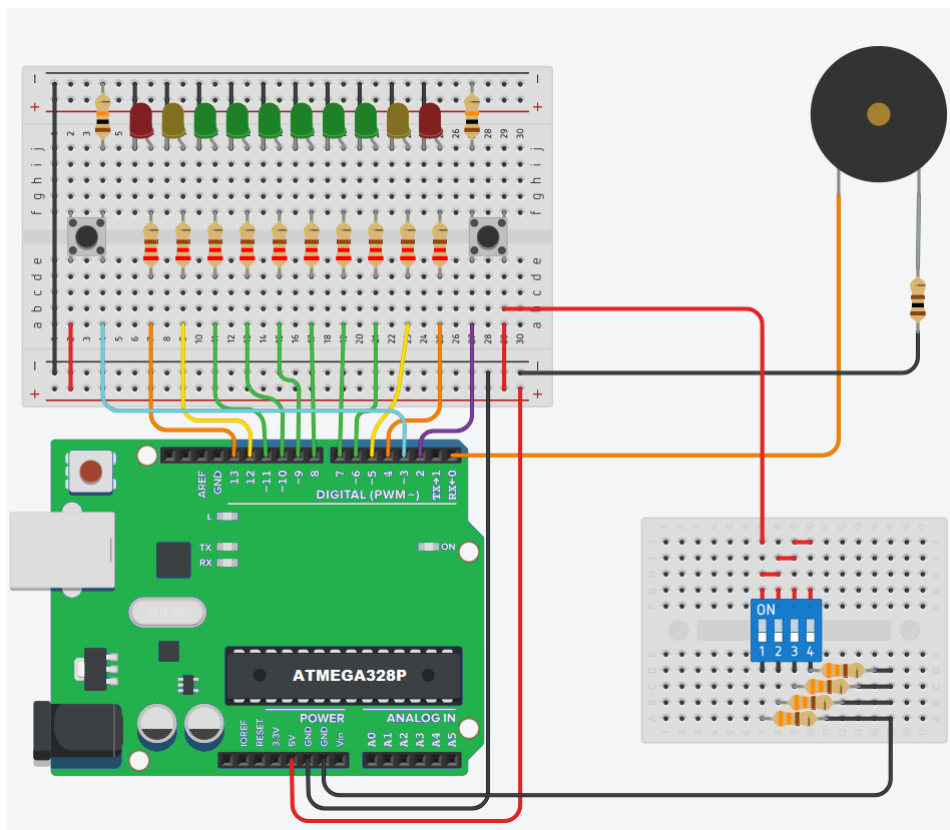


Figura 14 – Diagrama Esquemático
(Produzido no Adobe Tikercad)

² <https://easyeda.com/>

³ <https://jlcpcb.com/>

A figura a seguir mostra o resultado da montagem de todo o projeto:

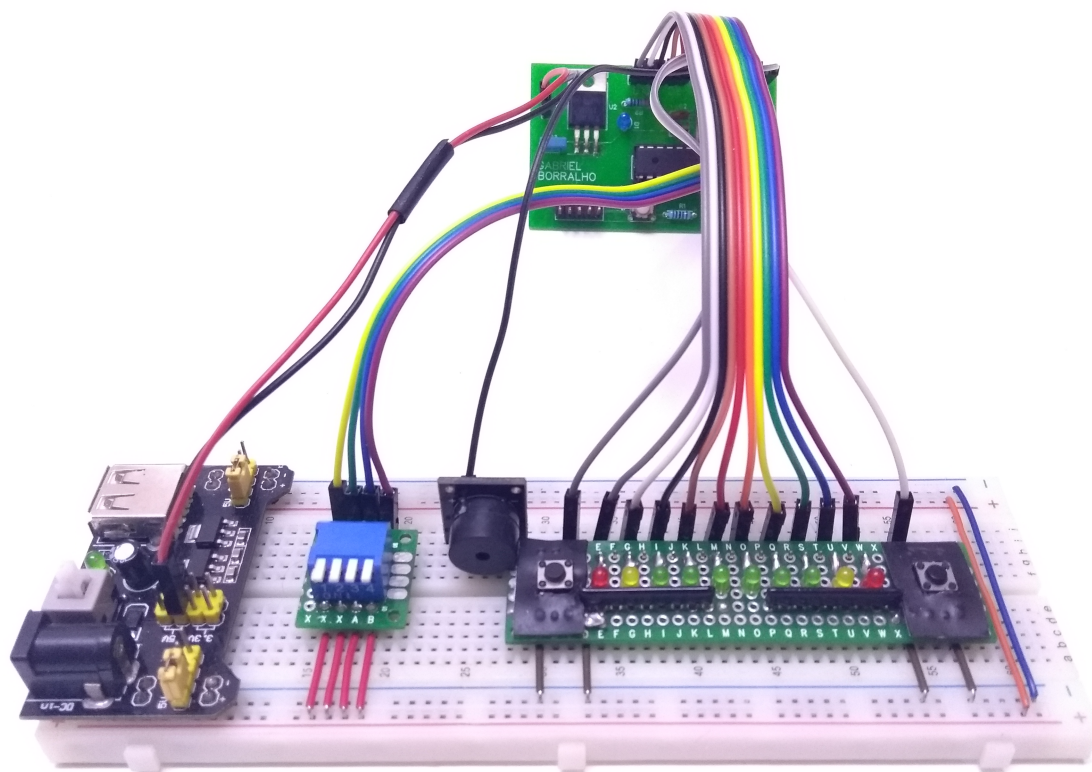


Figura 15 – Montagem física do Jogo

4 Conclusão

Um simples jogo, como o Pong, pode ter uma lógica associada, complexa o suficiente para se tornar cansativa e de difícil implementação. Para isso é necessário inicialmente esboçar as ideias iniciais através de fluxogramas, ferramenta muito útil tanto para compreensão, aprendizagem e leitura de códigos difíceis quanto algoritmos mais simples e ir melhorando essas ideias durante a implementação do código. A lógica por trás desse jogo pode ser complexa para um microcontrolador de apenas um núcleo, pelo motivo de unir muitas funcionalidades e fazer parecer ao usuário que todas estão sendo realizadas simultaneamente. A solução para este problema foi criar uma referência de tempo em que todo código e todas as funções tivessem um mesmo referencial evitando que a funcionalidade do sistema não fosse afetada.

Assim como a lógica necessária para implementar o projeto, a parte física ou *hardware* pode ser complicada, por conta dos erros humanos cometidos durante a montagem. Para resolver esse problema, foi feito com o *hardware* da mesma forma que com o *software*, dividi-lo em partes ou blocos. Foram construídos módulos que pudessem ser montados da forma mais prática e fácil.

O conjunto dessas ideias foram triviais para evitar cansativas tentativas de correção de erros. Desta forma o projeto pode atingir o seu funcionamento esperado de maneira mais rápida.

Referências

ALANKUS, E. *Arduino Soccer Game*. 2017. Disponível em: <<https://github.com/erhanalankus/Arduino-LED-Soccer-Game-with-6-Dollar-Starter-Kit>>. Acesso em: 17 de maio de 2018. Citado na página 15.

MANZANO, J. A. N. G. *ALGORÍTIMOS: lógica para desenvolvimento de programação de computadores*. 24. ed. São Paulo: Érica, 2010. Citado 2 vezes nas páginas 5 e 8.

Todo o projeto pode ser acessado de forma livre no link contido no QR-Code abaixo:



Apêndices

APÊNDICE A – Código Fonte

A seguir temos o código fonte feito neste trabalho:

```

1  /** ##### PROJETO 2 #####
2  **
3  **      Filename   : Pong_1D.ino
4  **      Project    : Projeto_02-LAB. de APLIC. COM MICROCOMPUTADORES
5  **      Professor  : André Borges Cavalcante
6  **      Processor  : Atmega328P
7  **      Version    : 1.00
8  **      Compiler   : Arduino IDE
9  **      Date/Time  : 24/05/2018, 09:24
10 **      Aluno      : Antonio Gabriel Sousa Borralho
11 **      Last Mod   : 28/06/2018 (DIP-SW OK)
12 **
13 ** #####*/
14
15
16 #define NOTE_C6  1047 //Define a frencia para o Buzzer
17
18 //##### VARI VEIS GLOBAIS #####
19 unsigned long tempoAtual = 0;
20 unsigned long tempoAnterior = 0;
21
22 //500 mili segundos para que um led mude de estado
23 const unsigned long inicialTempoPorLED = 500;
24 //Valor decrementado para aumento da velocidade
25 const unsigned long inicialDeltaTempoPorLED = 50;
26 unsigned long TempoPorLED = inicialTempoPorLED;
27 unsigned long deltaTempoPorLED = inicialDeltaTempoPorLED;
28 //Simboliza o zero do valor digital de DIPSW
29 int zero = 100;
30
31 int EstadoAtualBotaoJogador1 = 1;
32 int EstadoAnteriorBotaoJogador1 = 1;
33 int EstadoAtualBotaoJogador2 = 1;
34 int EstadoAnteriorBotaoJogador2 = 1;
35 int Buzzer=0; //Buzzer ligado a porta 0
36 int Porta;
37
38
39 //DIP SWITCH
40 int bit1=0; //Bit 1 do DIP-SW

```

```
41 int bit2=0;
42 int bit3=0;
43 int bit4=0;
44
45 const int Jogador1 = 12; //Led amarelo do Jogador 1
46 const int golJogador1 = 13;
47 /*Se o led vermelho(Porta 13) acender marca ponto para
48 adversário significa que a bola passou do led amarelo*/
49 const int BotaoJogador1 = 3;
50
51 const int Jogador2 = 5; //Led amarelo do Jogador 2
52 const int golJogador2 = 4;
53 /*Se o led vermelho(Porta 4) acender marca ponto para
54 adversário significa que a bola passou do led amarelo*/
55 const int BotaoJogador2 = 2;
56
57 int PosicaoAtual = Jogador1; //Jogador 1 inicia o jogo.
58 int PosicaoAnterior = Jogador1 + 1;
59 int deltaPosicao = 0;
60 //Indica qual o led anterior a ser apagado em PosicaoAnterior.
61
62 int scoreJogador1 = 0;
63 int scoreJogador2 = 0;
64
65 //Sentido Convencional é do Jogador 1 para o 2
66 bool SentidoContrario = true;
67 bool EntradaPermitida = true;
68
69 //##### PROTOTIPOS DAS FUNCOES #####
70 void VerificarEntradas();
71 void MudaDirecaoDaBola();
72 void DeterminaProximaPosicao();
73 void MoveBola();
74 void ChecaGols();
75 void PontuacaoParaJogador(int pontuador);
76 void ChecaFimDoJogo();
77 void MostraScores(int Pontuador);
78 void ResetaJogada();
79 void FimDoJogo(int Vencedor);
80 void ExibeVencedor(int Jogador);
81 void InibeJogador(int Jogador);
82 void PiscaLEDs(int n_vezes, int Jogador);
83 void BemVindo(int n_vezes);
84 void LerDIPSW();
85 void tempoDIPSW();
86
87 //##### LOOP E SETUP #####
```

```

88 void setup() {
89     pinMode(Buzzer,OUTPUT);
90     //Configura Portas 4 a 13 como SA DAS (LEDS)
91     for(Porta=4;Porta<=13;Porta++){
92         pinMode(Porta, OUTPUT);
93     }
94     //Configura Portas 2 e 3 como ENTRADAS (botoes)
95     pinMode(2, INPUT_PULLUP); //(Resistor PullUp ativado)
96     pinMode(3, INPUT_PULLUP);
97     //Cerimônia de StartGame
98     BemVindo(30);
99     //Aguarda Jogador 1 iniciar
100    digitalWrite(Jogador1, HIGH);
101    while(digitalRead(BotaoJogador1)==1){}
102    delay(500); //Remove buffer do teclado
103 }
104
105 void loop(){
106     VerificarEntradas();
107     tempoAtual = millis();
108     if (tempoAtual - tempoAnterior >= TempoPorLED){
109         LerDIPSW();
110         tempoDIPSW();
111
112         ChecaGols();
113         DeterminaProximaPosicao();
114         MoveBola();
115
116         tempoAnterior = tempoAtual;
117     }
118 }
119 //##### FUNCOES #####
120 void VerificarEntradas(){
121     LerDIPSW();
122     EstadoAtualBotaoJogador1 = digitalRead(BotaoJogador1);
123     EstadoAtualBotaoJogador2 = digitalRead(BotaoJogador2);
124     if (EstadoAtualBotaoJogador1 != EstadoAnteriorBotaoJogador1 &&
        EntradaPermitida){
125         if (EstadoAtualBotaoJogador1 == 1){
126             if (PosicaoAtual == Jogador1){
127                 MudaDirecaoDaBola();
128                 AumentaVelocidade();
129             }else{
130                 PontuacaoParaJogador(2);
131             }
132         }
133         EstadoAnteriorBotaoJogador1 = EstadoAtualBotaoJogador1;

```

```

134 }
135 if (EstadoAtualBotaoJogador2 != EstadoAnteriorBotaoJogador2 &&
    EntradaPermitida){
136     if (EstadoAtualBotaoJogador2 == 1){
137         if (PosicaoAtual == Jogador2){
138             MudaDirecaoDaBola();
139             AumentaVelocidade();
140         }else{
141             PontuacaoParaJogador(1);
142         }
143     }
144     EstadoAnteriorBotaoJogador2 = EstadoAtualBotaoJogador2;
145 }
146 }
147 //_____
148 void MudaDirecaoDaBola(){
149     /*Muda a direcao convencional (de Jogador1 para Jogador2) para
150     o SentidoContrario contrário*/
151     SentidoContrario = !SentidoContrario;
152     //Apenas uma mudança de direção por jogada é permitida
153     //para uma jogabilidade consistente
154     EntradaPermitida = false;
155 }
156 //_____
157 void AumentaVelocidade(){
158     TempoPorLED -= deltaTempoPorLED;
159     if (deltaTempoPorLED > 1){
160         /*Quanto menor este valor mais rápido a velocidade aumenta.
161         Isto impede que atinja uma velocidade insana rapidamente.
162         Ajuste ou remova isto se as rodadas ficarem muito longas.*/
163         deltaTempoPorLED -= 1;
164     }
165 }
166 }
167 //_____
168 void MoveBola(){ //Move a bola uma posição
169     PosicaoAnterior = PosicaoAtual;
170     digitalWrite(PosicaoAnterior, 0);
171     PosicaoAtual = PosicaoAtual + deltaPosicao;
172     digitalWrite(PosicaoAtual, 1);
173     EntradaPermitida = true;
174 }
175 //_____
176 void DeterminaProximaPosicao(){
177     //Se a direção estiver no SentidoContrario convencional
178     if (SentidoContrario){
179         deltaPosicao = -1;

```



```
180     }else{
181         deltaPosicao = 1;
182     }
183 }
184 //
185 void ChecaGols(){
186     if (PosicaoAtual == golJogador2){
187         PontuacaoParaJogador(1);
188     }else if (PosicaoAtual == golJogador1){
189         PontuacaoParaJogador(2);
190     }
191 }
192 //
193 void PontuacaoParaJogador(int pontuador){
194     EntradaPermitida = false;
195     PiscaLEDs(2, 0);
196     if (pontuador == 1){
197         scoreJogador1++;
198         MostraScores(1);
199     }else if (pontuador == 2){
200         scoreJogador2++;
201         MostraScores(2);
202     }
203     ChecaFimDoJogo();
204 }
205 //
206 void ChecaFimDoJogo(){
207     if (scoreJogador1 == 3){
208         FimDoJogo(1);
209     }
210     if (scoreJogador2 == 3){
211         FimDoJogo(2);
212     }
213 }
214 //
215 void MostraScores(int Pontuador){
216
217     if (Pontuador == 1){
218         digitalWrite(Jogador1, 1);
219         PosicaoAtual = Jogador1;
220         //Habilita a direção para o adversário
221         SentidoContrario = true;
222     }else if (Pontuador == 2){
223         digitalWrite(Jogador2, 1);
224         PosicaoAtual = Jogador2;
225         //Habilita o SentidoContrario convencional.
226         SentidoContrario = false;
```

```
227 }
228 /*Usamos os seis LEDs no meio para mostrar a pontuação.
229 Cada Jogador tem tres LEDs verdes para mostrar a pontuacao.
230 Com tres gols vence o jogo.*/
231 for (int i = 0; i < scoreJogador1; i++){
232     digitalWrite((11 - i), 1);
233 }
234 for (int i = 0; i < scoreJogador2; i++){
235     digitalWrite((6 + i), 1);
236 }
237 //Tres segundos para que os Jogadores vizualizem a pontuacao
238 delay(3000);
239
240 InibeJogador(1);
241 InibeJogador(2);
242
243 if (Pontuador == 1){
244     ChecaFimDoJogo();
245     digitalWrite(Jogador1, HIGH);
246     while (digitalRead(BotaoJogador1)==1){}
247     delay(800);
248 }else if (Pontuador == 2){
249     ChecaFimDoJogo();
250     digitalWrite(Jogador2, HIGH);
251     while (digitalRead(BotaoJogador2)==1){}
252     delay(800);
253 }
254 ResetaJogada();
255 }
256 //
257 void ResetaJogada(){
258     //Velocidade para o valor inicial de cada rodada
259     TempoPorLED = inicialTempoPorLED;
260     //Velocidade delta para o valor inicial de cada rodada
261     deltaTempoPorLED = inicialDeltaTempoPorLED;
262 }
263 //
264 void FimDoJogo(int Vencedor){
265     PiscaLEDs(10, Vencedor);
266     if(Vencedor==1)
267         InibeJogador(2);
268     else
269         InibeJogador(1);
270     scoreJogador1 = 0;
271     scoreJogador2 = 0;
272 }
273 //
```

```
274 void ExibeVencedor(int Jogador){
275     if (Jogador != 1){
276         digitalWrite(4, HIGH);
277         digitalWrite(5, HIGH);
278         digitalWrite(6, HIGH);
279         digitalWrite(7, HIGH);
280         digitalWrite(8, HIGH);
281     }
282     if (Jogador != 2){
283         digitalWrite(9, HIGH);
284         digitalWrite(10, HIGH);
285         digitalWrite(11, HIGH);
286         digitalWrite(12, HIGH);
287         digitalWrite(13, HIGH);
288     }
289 }
290 //
291 void InibeJogador(int Jogador){
292     if (Jogador != 1){
293         digitalWrite(4, LOW);
294         digitalWrite(5, LOW);
295         digitalWrite(6, LOW);
296         digitalWrite(7, LOW);
297         digitalWrite(8, LOW);
298     }
299     if (Jogador != 2){
300         digitalWrite(9, LOW);
301         digitalWrite(10, LOW);
302         digitalWrite(11, LOW);
303         digitalWrite(12, LOW);
304         digitalWrite(13, LOW);
305     }
306 }
307 //
308 void PiscaLEDs(int n_vezes, int Jogador){
309     for (int i = 0; i < n_vezes; i++){
310         ExibeVencedor(Jogador);
311         tone(Buzzer, NOTE_C6);
312         delay(35);
313         InibeJogador(Jogador);
314         noTone(Buzzer);
315         delay(35);
316     }
317     InibeJogador(1);
318     InibeJogador(2);
319 }
320 //
```

```

321 void BemVindo(int Count){
322     for (int i=0; i < Count; i++){
323         ExibeVencedor(1);
324         ExibeVencedor(2);
325         tone(Buzzer , i*exp(-i/10000)); //Oscilacao para Buzzer
326         delay(35);
327         InibeJogador(1);
328         InibeJogador(2);
329         noTone(Buzzer);
330         delay(35);
331     }
332     InibeJogador(1);
333     InibeJogador(2);
334 }
335 //
336 void LerDIPSW(){
337     bit1 = analogRead(1);
338     bit2 = analogRead(2);
339     bit3 = analogRead(3);
340     bit4 = analogRead(4);
341 }
342 //
343 void tempoDIPSW(){
344     //0001
345     if ((bit1 < zero)&&(bit2 < zero)&&(bit3 < zero)&&(bit4 > zero)){
346         TempoPorLED=initialTempoPorLED;
347         deltaTempoPorLED = inicialDeltaTempoPorLED;
348     }
349     //0010
350     if ((bit1 < zero)&&(bit2 < zero)&&(bit3 > zero)&&(bit4 < zero)){
351         TempoPorLED=initialTempoPorLED-(initialTempoPorLED/15);
352         deltaTempoPorLED = inicialDeltaTempoPorLED-1;
353     }
354     //0011
355     if ((bit1 < zero)&&(bit2 < zero)&&(bit3 > zero)&&(bit4 > zero)){
356         TempoPorLED=initialTempoPorLED-(2*initialTempoPorLED/15);
357         deltaTempoPorLED = inicialDeltaTempoPorLED-2;
358     }
359
360     //0100
361     if ((bit1 < zero)&&(bit2 > zero)&&(bit3 < zero)&&(bit4 < zero)){
362         TempoPorLED=initialTempoPorLED-(3*initialTempoPorLED/15);
363         deltaTempoPorLED = inicialDeltaTempoPorLED-3;
364     }
365     //0101
366     if ((bit1 < zero)&&(bit2 > zero)&&(bit3 < zero)&&(bit4 > zero)){
367         TempoPorLED=initialTempoPorLED-(4*initialTempoPorLED/15);

```

```
368     deltaTempoPorLED = inicialDeltaTempoPorLED-4;
369 }
370
371 //0110
372 if ((bit1 < zero)&&(bit2 > zero)&&(bit3 > zero)&&(bit4 < zero)){
373     TempoPorLED=inicialTempoPorLED-(5*inicialTempoPorLED/15);
374     deltaTempoPorLED = inicialDeltaTempoPorLED-5;
375 }
376
377 //0111
378 if ((bit1 < zero)&&(bit2 > zero)&&(bit3 > zero)&&(bit4 > zero)){
379     TempoPorLED=inicialTempoPorLED-(6*inicialTempoPorLED/15);
380     deltaTempoPorLED = inicialDeltaTempoPorLED-6;
381 }
382
383 //1000
384 if ((bit1 > zero)&&(bit2 < zero)&&(bit3 < zero)&&(bit4 < zero)){
385     TempoPorLED=inicialTempoPorLED-(7*inicialTempoPorLED/15);
386     deltaTempoPorLED = inicialDeltaTempoPorLED-7;
387 }
388
389 //1001
390 if ((bit1 > zero)&&(bit2 < zero)&&(bit3 < zero)&&(bit4 > zero)){
391     TempoPorLED=inicialTempoPorLED-(8*inicialTempoPorLED/15);
392     deltaTempoPorLED = inicialDeltaTempoPorLED-8;
393 }
394
395 //1010
396 if ((bit1 > zero)&&(bit2 < zero)&&(bit3 > zero)&&(bit4 < zero)){
397     TempoPorLED=inicialTempoPorLED-(9*inicialTempoPorLED/15);
398     deltaTempoPorLED = inicialDeltaTempoPorLED-9;
399 }
400
401 //1011
402 if ((bit1 > zero)&&(bit2 < zero)&&(bit3 > zero)&&(bit4 > zero)){
403     TempoPorLED=inicialTempoPorLED-(10*inicialTempoPorLED/15);
404     deltaTempoPorLED = inicialDeltaTempoPorLED-10;
405 }
406
407 //1100
408 if ((bit1 > zero)&&(bit2 > zero)&&(bit3 < zero)&&(bit4 < zero)){
409     TempoPorLED=inicialTempoPorLED-(11*inicialTempoPorLED/15);
410     deltaTempoPorLED = inicialDeltaTempoPorLED-11;
411 }
412
413 //1101
414 if ((bit1 > zero)&&(bit2 > zero)&&(bit3 < zero)&&(bit4 > zero)){
```

```
415     TempoPorLED=inicialTempoPorLED-(12*inicialTempoPorLED/15);
416     deltaTempoPorLED = inicialDeltaTempoPorLED-12;
417 }
418
419 //1110
420 if ((bit1 > zero)&&(bit2 > zero)&&(bit3 > zero)&&(bit4 < zero)){
421     TempoPorLED=inicialTempoPorLED-(13*inicialTempoPorLED/15);
422     deltaTempoPorLED = inicialDeltaTempoPorLED-13;
423 }
424
425 //1111
426 if ((bit1 > zero)&&(bit2 > zero)&&(bit3 > zero)&&(bit4 > zero)){
427     TempoPorLED=inicialTempoPorLED-(14*inicialTempoPorLED/15);
428     deltaTempoPorLED = inicialDeltaTempoPorLED-14;
429 }
430 }
431 //##### FIM #####
```

Listing A.1 – Código fonte

APÊNDICE B – Lista de Materiais Utilizados e Preços

| Quantidade | Componente | Preço |
|------------|--|----------------------|
| 1 | Kit ATmega328P ¹ | R\$ 17,91 |
| 1 | Gravador ATMEL AVR USBasp ² | R\$ 26,55 |
| 1 | Protoboard Padrão ³ | R\$ 26,91 |
| 1 | Fonte Ajustável para Protoboard ⁴ | R\$ 9,90 |
| 1 | Pack de Jumpers ⁵ | R\$ 10,71 |
| 1 | Módulo Buzzer ⁶ | R\$ 6,81 |
| 1 | Regulador de tensão L7805 ⁷ | R\$ 1,16 |
| 1 | DIP Switch 4 vias ⁸ | R\$ 1,99 |
| 1 | Bateria 9V ⁹ | R\$ 19,90 |
| 14 | Resistor 330Ω 5% (1/4W) ¹⁰ | R\$0,08 |
| 10 | LED de alto brilho 3mm ¹¹ | R\$0,20 |
| 2 | Chave Tátil 4 Terminais ¹² | R\$0,68 |
| - | Fabricação da PCB | U\$ 2,00 |
| Total | | R\$126,32 + U\$ 2,00 |

Tabela 1 – Lista de Materiais Utilizados e Preços

Todo o material pode ser encontrado nos respectivos links:

- 1 - <<http://www.baudaeletronica.com.br/kit-arduino-atmega328p-bootloader.html>>
- 2 - <<http://www.baudaeletronica.com.br/gravador-atmel-usb-40zif.html>>
- 3 - <<http://www.baudaeletronica.com.br/protoboard-830-pontos-mp-830a-minipa.html>>
- 4 - <<https://www.filipeflop.com/produto/fonte-ajustavel-protoboard/>>
- 6 - <<http://www.baudaeletronica.com.br/modulo-buzzer-5v-passivo.html>>
- 5 - <<http://www.baudaeletronica.com.br/jumper-premium-40p-x-20cm-macho-macho.html>>
- 7 - <<http://www.baudaeletronica.com.br/regulador-de-tensao-l7805.html>>
- 8 - <<http://www.baudaeletronica.com.br/dip-switch-azul-4-vias-90-graus.html>>
- 9 - <<http://www.baudaeletronica.com.br/bateria-9v-duracell-1006.html>>
- 10 - <<http://www.baudaeletronica.com.br/resistor-330r-5-1-4w.html>>
- 11 - <<http://www.baudaeletronica.com.br/led-de-alto-brilho-3mm-vermelho.html>>
- 12 - <<http://www.baudaeletronica.com.br/chave-tactil-12x12x4-3mm-4-terminais.html>>

O projeto da PCB pode ser acessado e encomendado em <https://easyeda.com/gsborrvalho/ARDUINO_STANDALONE-08d6aada4aad41449d4ec6cce831ee71>