UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
ENGENHARIA DE COMPUTAÇÃO

HENRIQUE CASTRO SANTOS, WELLINGTON MONTAGNINI,
ADRIANNE TOINKO

CUBO DE LEDS 4X4X4

TRABALHO PARA A MATÉRIA DE OFICINA DE INTEGRAÇÃO 1

CURITIBA
2019

HENRIQUE CASTRO SANTOS, WELLINGTON MONTAGNINI,
ADRIANNE TOINKO

CUBO DE LEDS 4X4X4

Trabalho para a matéria de Oficina de Integração 1 apresentado como requisito parcial da grade de Engenharia de Computação.

Orientador: Gustavo Benvenutti Borba, Ronnier Frates Rohrich

CURITIBA
2019

RESUMO

Este é um trabalho feito para a matéria de Oficina de Integração 1 que tem como objetivo integrar conhecimentos de outras matérias, trabalhar em equipe, desenvolver e testar um projeto baseado em um sistema microcontrolado. O projeto desenvolvido pela equipe foi um cubo de LEDs 4x4x4 que é controlado a partir de um Arduino. Com esse projeto, os conhecimentos em montar circuitos, soldar componentes e criação de software foram testados e postos à prova.

Palavras-chaves: cubo, led, microcontrolador, arduino.

ABSTRACT

This is a project made for the discipline called "Oficina de Integração 1" that has as goals, to integrate knowledge from other subjects, teamwork develop and test a project based on a micro-controlled system. The project that was developed here is a 4x4x4 LED cube that is controlled via Arduino. This way, our skills on circuit montage, welding and software development were tested and put to proof. **Key-words**: arduino. led. cube.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 OBJETIVO

O objetivo com o cubo é integrar o mesmo com um software que disponibilize controle sobre qual LED acender, em que momento, por quanto tempo e qual LED acender em seguida. Basicamente ter controle total sobre o cubo e criar animações a partir disso.

1.2 MONTAGEM

Com relação a montagem do cubo, na hora de soldar os LEDs entre si, cada andar está ligado somente com os catodos e os anodos fazem a conexão entre diferentes andares. Cada andar está ligado a um transistor que vai ser responsável por fechar a conexão com o ground. Assim, para acender um LED específico, mandamos um sinal para a coluna desejada e fechamos o transistor da layer designada.

Para controlar cada um dos LEDs, teoricamente é necessário sessenta e quatro portas no Arduino para mandar os sinais. Para simplificar o circuito, utiliza-se um multiplexador que recebe quatro sinais e transforma em dezesseis porém, o multiplexador envia quinze sinais e deixa de mandar para um. Este é exatamente o contrário do necessitado, por isso, foram utilizadas portas NOTs onde cada uma delas recebe seis sinais e os inverte e por haver dezesseis sinais para serem invertidos, três portas NOTs foram utilizadas.

2 HARDWARE

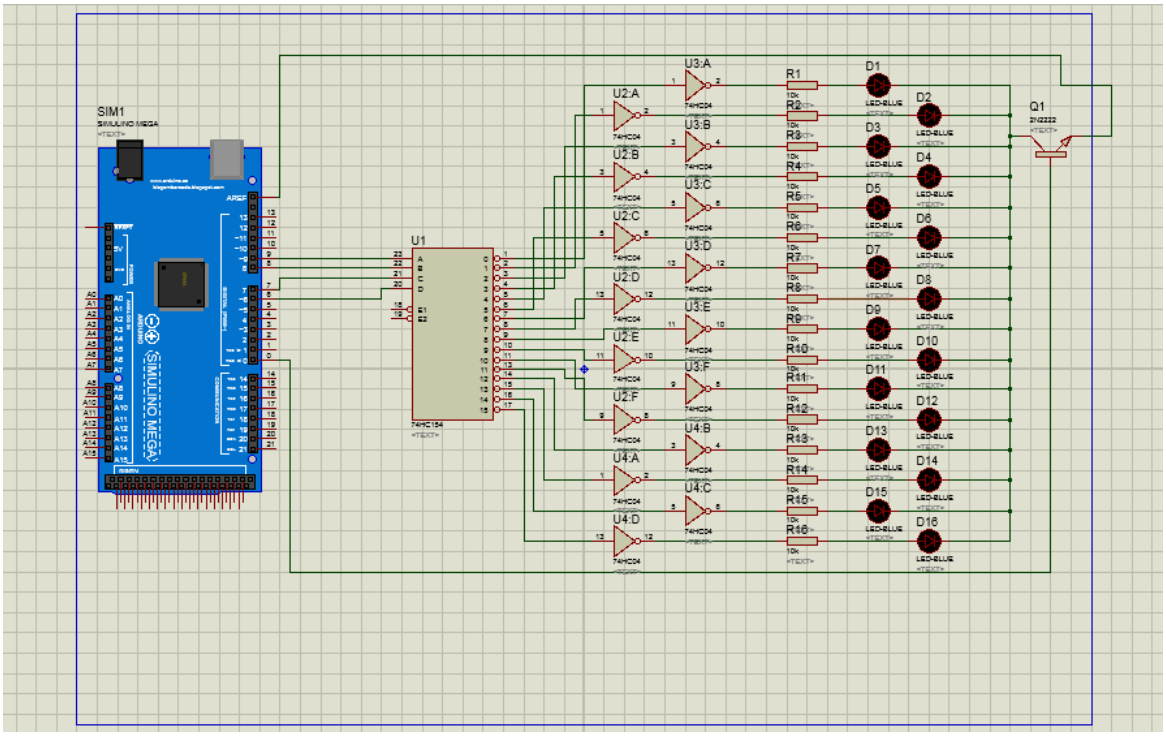

Figura 1 – Circuito Esquemático

2.1 TEORIA

Como é possível ver pela Figura 1, o Arduino manda quatro sinais para o multiplexador 74HC154 que transforma em 16 sinais e conforme o valor mandado pelo Arduino, manda o sinal para a coluna desejada. Saindo do multiplexador, o sinal passa por uma porta NOT que inverte o sinal e chega até os LEDs. No esquemático, há um NOT para cada sinal. Não foi o que ocorreu no projeto físico, foram utilizados 3 NOTs que invertiam 6 sinais cada. Após isso, os sinais chegam até os LEDs que estão todos ligados em um transistor.

No esquemático, estão sendo representados apenas dezesseis LEDs e a única diferença dele para o projeto físico é que existem mais quarenta e oito LEDs e mais três transistores que correspondem aos três andares remanescentes.

Tudo isso está sendo energizado por uma bateria de 9 volts. Além disso, há um switch button para ligar e desligar o fornecimento de bateria. Também existe um push button para alternar entre as animações pré-existentes no código do cubo.

2.2 PRÁTICA

Na execução, a primeira coisa a ser feita era a montagem do cubo. Com cerca de 200 LEDs disponíveis, havia espaço para erro e até mesmo foi cogitada a ideia de construir mais de um cubo. Nenhum dos membros do grupo possuía experiência com a utilização do ferro de solda e por isso o primeiro cubo que foi desenvolvido estava com uma estética um tanto quanto relaxada. No entanto, a ideia inicial era de utilizar ele mesmo para o projeto final.

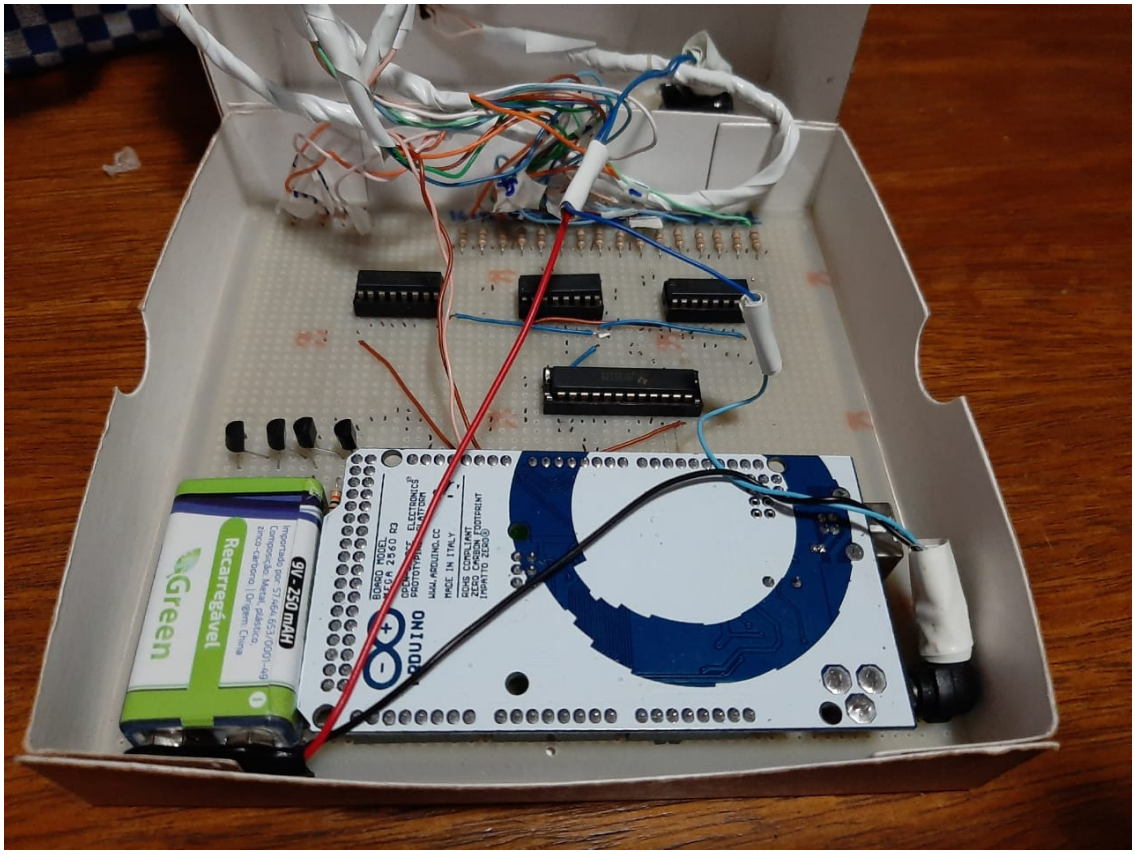

Figura 2 – Circuito na base

Com o cubo pronto, utilizamos uma placa de Fenolite com dimensões de quinze por quinze centímetros para imprimir o circuito. Não houve dificuldade em seguir o esquemático e em pouco dias o mesmo já estava pronto. Logo em seguida, o Arduino, o cubo e o circuito foram integrados entre si sem acabamento algum para a verificação de que tudo estava funcionando como deveria. No momento estava tudo em ordem e os testes com o software começaram a ser desenvolvidos. Porém, uma semana depois, quando a fase da construção da base onde o cubo ficaria estava para iniciar, foi notado que o cubo estava funcionando mesmo sem a parte do aterramento com os transistores estar ligada. Após dias em busca do motivo real para a falha no cubo, foi possível verificar que algumas soldas estavam malfeitas e estavam funcionando como aterramento para algumas layers. Com isso, encontramos pedaços do cubo que poderiam ser responsáveis pelo mal funcionamento. Porém quando um LED era substituído, logo em seguida era encontrado outro apresentando defeito. Com mais experiência com o ferro de solda e

com muitos problemas no cubo, foi decidido que outro seria confeccionado, aproveitando que já existiam LEDs reservas.

Em seguida, foi comprada uma caixa de papelão com as mesmas medidas da placa de fenolite onde se encontra o circuito (Figura 2). Com isso, foram feitos os 16 buracos na sua tampa para colocar os anodos do cubo e assim fazer a base responsável pela estética. Depois disso, foram feitos dois buracos na lateral da caixa para serem encaixados os dois botões (Figura 3) sendo um responsável pela mudança de padrões de sinais e outro pelo fornecimento de energia.

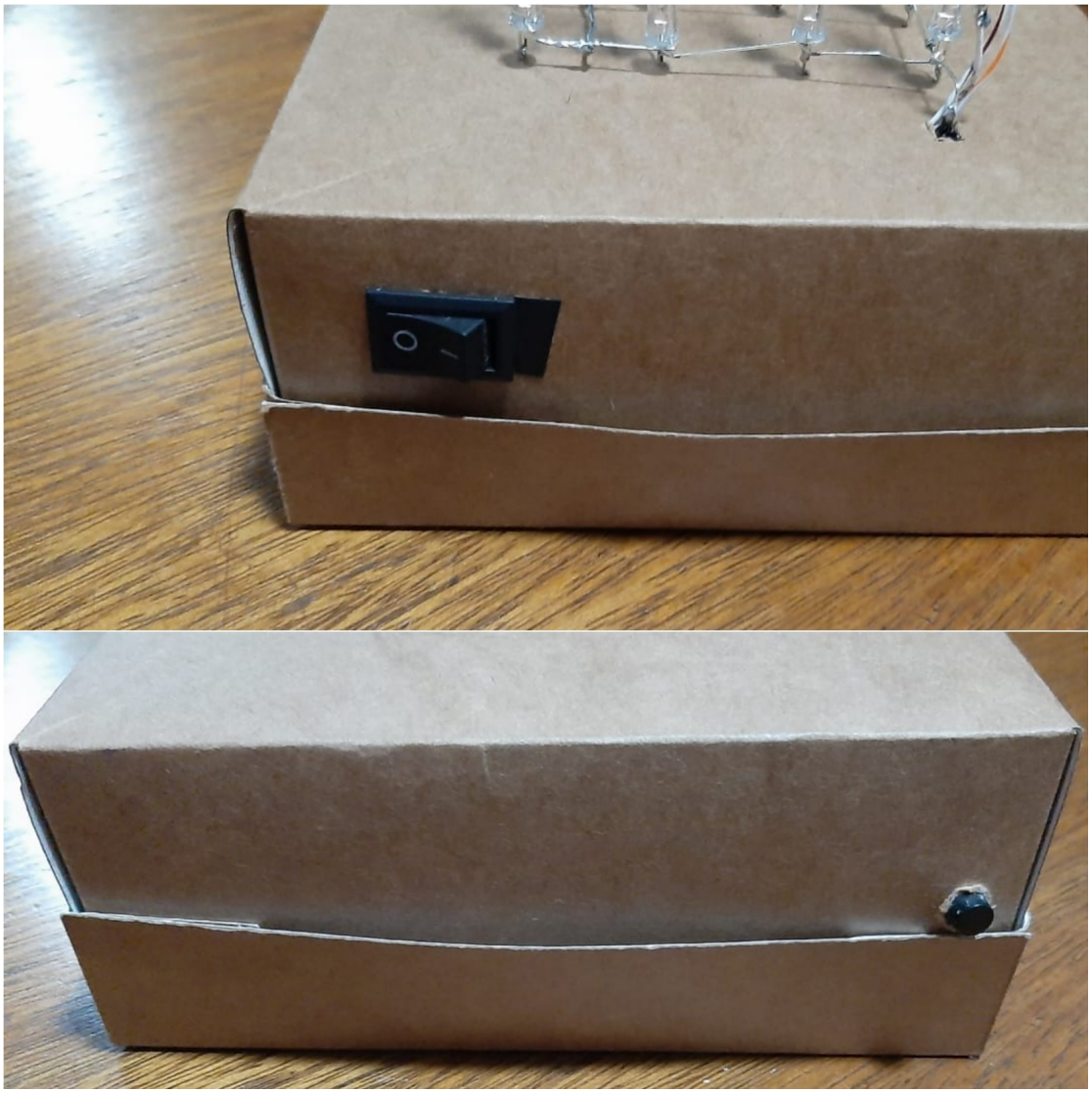

Figura 3 – Detalhes do acabamento

3 SOFTWARE

```
void converteBinario(byte valor, byte andar)
{
    digitalWrite(PORTAS_TRANSISTORES[andar], HIGH);
    for (byte i = 0; i < 4; i++)
    {
      digitalWrite(PORTAS_SAIDAS[i], valor % 2);
      valor /= 2;
    }
    digitalWrite(PORTAS_TRANSISTORES[andar], LOW);

}
```

O código acima foi a primeira função desenvolvida para iniciar o desenvolvimento do trabalho. Como é possível ver, a função recebe um *byte valor* que é um valor entre zero e quinze que representa o número da coluna que se deseja acender. A outra variável a ser recebida é o *byte andar* que é um valor entre zero e três que representa o andar que se deseja acender. Em seguida, há um *digitalWrite* que manda um sinal para o transistor do andar desejado. Com isso, o transistor é fechado e qualquer LED que esteja recebendo sinal dessa camada será aceso. Agora, dentro do for, é feita a conversão do *byte valor* para binário e logo em seguida o sinal é enviado para o multiplexador. Como todos os números recebido vão estar dentro do intervalo de quatro bits e o multiplexador recebe quatro sinais, o loop só é percorrido quatro vezes. No final da função, o sinal dos transistores é interrompido.

Em seguida, um *header* foi criado com todas as variáveis que seriam utilizadas no desenvolver do programa e definindo quais portas do Arduino seriam utilizadas. Segue abaixo o código abaixo:

```
//Define as portas para energizar as colunas
const int PORTAS_SAIDAS[4] = {12, 11, 10, 9};
//Define as portas para os transistores
const int PORTAS_TRANSISTORES[4] = {5, 6, 7, 8};
//Matriz que define o cubo de led
byte *** matrixLed;
//Define porta para entrada do botao de troca de animações
#define PORTA_BOTAO 4
//Valor pré definido para as animações de troca de planos.
#define CONTADOR_PLANOS 600
//Variáveis auxiliares utilizadas nas funções.
```

```
int aux;
int auxFunc;
int contadorPlanos;
int inversor = -20;
byte contadorSegundos;
byte contadorMinutos;
int timerAuxFunc;
int timerFunc;
```

A próxima função a ser comentada é a *acenderCubo* que é a função responsável por fazer a transformação da lógica do multiplexador para uma matriz 4x4x4. Como já foi dito anteriormente, o multiplexador recebe um valor binário entre 0 e 15 e fornece energia para a coluna correspondente ao valor. Para fazer isso funcionar como uma matriz, precisamos fazer uma aritmética básica para assim o multiplexador conseguir receber os valores na forma matricial.

Segue na figura abaixo (Figura 4) que demonstra muito bem a diferença entre a versão matricial (vermelho) e a forma que o multiplexador (preto) recebe os dados:

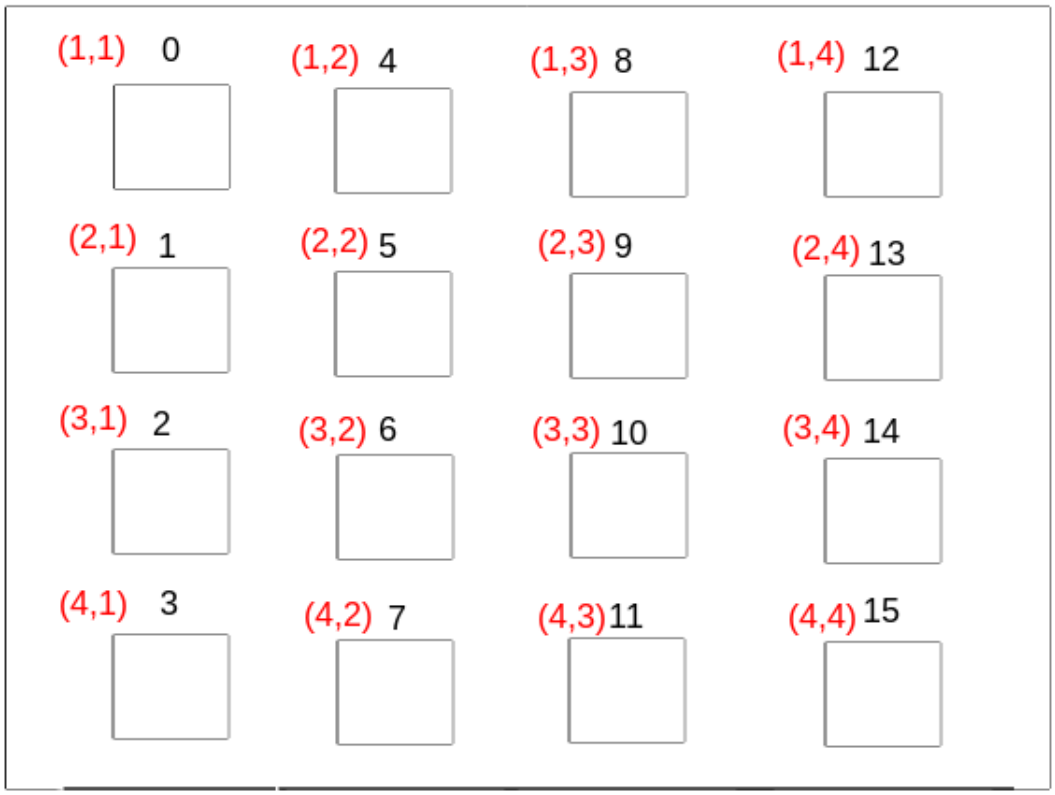

Figura 4 – Cubo na forma matricial vs multiplexador

```
bool acenderCubo(byte *** matrix, int tempo ) {
  int timer, timerAux;
  timerAux = millis();
  do {
    //Percorre toda a matriz
    for (byte i = 0; i < 4; i++) {
      for (byte j = 0; j < 4; j++) {
        for (byte k = 0; k < 4; k++) {
          //Caso o valor seja 1, acende o respectivo led
          if (matrix[i][j][k] == 1) {
            //Ajuste do valor do auxiliar para acender corretamente
            o led
            aux = 4*j;
            //Converte em binário e acende o led
            converteBinario(k + aux, i);
          }
        }
      }
    }
    timer = millis() - timerAux;
    //Verifica se o botão foi pressionado
    if (lerBotao())
      return true;
  } while (timer <= tempo);
  return false;
}
```

Vendo a imagem e o código, é possível concluir que cada coluna precisa ser multiplicada por quatro para ser acessada. Basicamente, se a coluna j for igual a um, o valor do multiplexador correspondente é quatro, se for igual a dois, é oito e assim sucessivamente. Na variável *i*, passamos qual é a camada correspondente ou na forma matricial, o eixo z. Para mexer neste eixo, não é necessária nenhuma aritmética já que o mesmo é equivalente aos quatro transistores e os mesmos são correspondentes a valores entre um e quatro.

Com tudo isso pronto, foram desenvolvidas as animações, que são padrões ordenados de sinais enviados ao cubo que formam a ilusão de continuidade e também a função que disponibiliza ao usuário o controle do LED como uma matriz 4x4x4. Segue abaixo um exemplo de animação e a função de controle respectivamente:

```
void piscaPisca(int tempo, int tempoPiscando) {
  //Zera matriz do cubo
  zeraMatrix   ();
  //Pega o tempo inicial quando a função iniciou
  timerAuxFunc = millis();
  do {
    //Preenche toda a matriz
    for (byte i = 0; i < 4; i++) {
      for (byte j = 0; j < 4; j++) {
        for (byte k = 0; k < 4; k++) {
          matrixLed[i][j][k] = 1;
        }
      }
    }
    //Manda acender as luzes relacionada a matriz
    //Caso retorne true, o botão foi apertado e a função precisa parar.
    if (acenderCubo(matrixLed, tempoPiscando)) {
      return;
    }
    //Zera matriz novamente
    zeraMatrix   ();
    //Manda acender as luzes relacionada a matriz
    //Caso retorne true, o botão foi apertado e a função precisa parar.
    if (acenderCubo(matrixLed, tempoPiscando)) {
      return;
    }
    //Pega o tempo que decorreu desde que a função iniciou
    timerFunc = millis() - timerAuxFunc;
    //Percorre a função até o tempo determinado
  } while (timerFunc <= tempo);
  return;
}

 /**
   Função que o usuário envia um x, y, z e de acordo com essas
   coordenadas acende um led.
   @param x Valor para coordenada x
   @param y Valor para coordenada y
   @param z Valor para coordenada z
*/
```

```
void controlarLed(byte x, byte y, byte z, int tempo) {
  timerAuxFunc = millis();
  zeraMatrix();
  do {
    //Verifica se as coordenadas são válidas
    if (x >= 1 && y >= 1 && z >= 1
        && x <= 4 && y <= 4 && z <= 4) {
      matrixLed[z - 1][y - 1][x - 1] = 1;
      //Manda acender as luzes relacionada a matriz
      //Caso retorne true, o botão foi apertado e a função
      precisa parar.
      if (acenderCubo(matrixLed, 100)) {
        return;
      }
    }
    //Caso as coordenadas não sejam válidas
    else {
      //Pisca 2x significando que deu erro com as coordenadas.
      piscaPisca(1000, 250);
      return;
    }
    timerFunc = millis() - timerAuxFunc;
  } while (timerFunc < tempo);
}
```

4 RESULTADOS

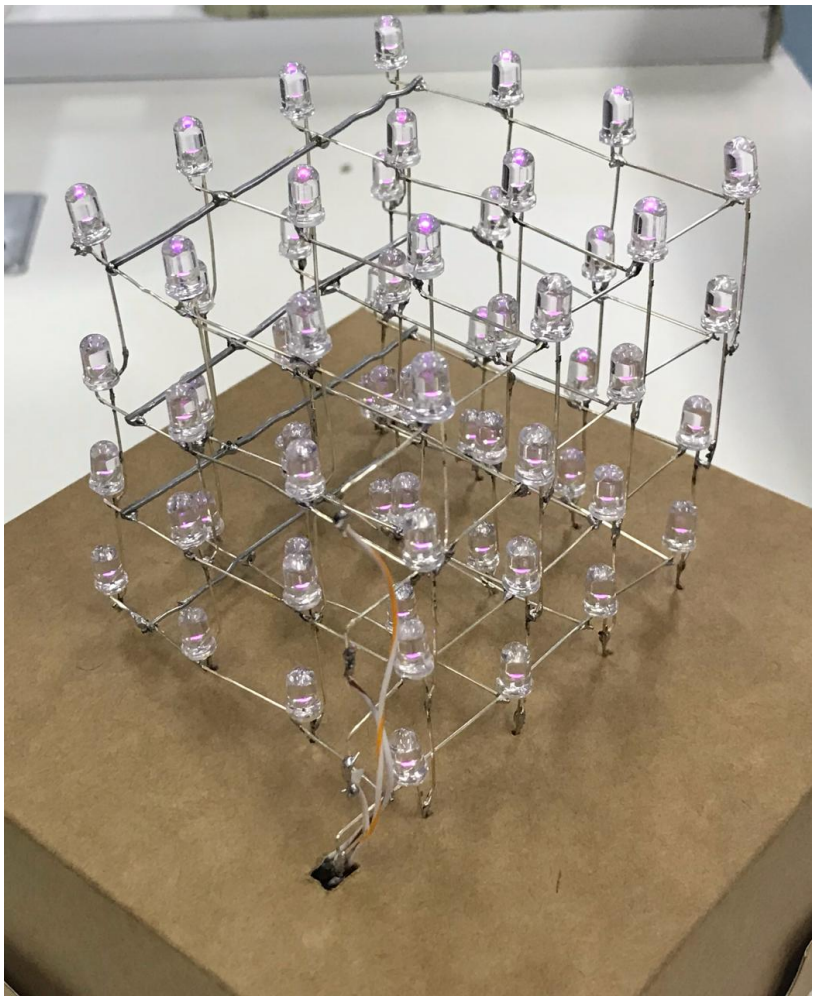

Figura 5 – Resultado Final

O objetivo proposto inicialmente era de, a partir do software, obter o controle total sobre o cubo, podendo acender um LED específico por um tempo determinado, acender outro LED após um determinado intervalo. Com esse objetivo em mente, o cubo pode ser controlado com todos esses parâmetros e além disso, existe um botão que ao ser pressionado pode alternar entre padrões de animação pré-definidos pelo usuário no código. A instalação desses padrões é fácil e intuitiva bastando passar para o programa os LEDs que desejam ser acendidos a partir da matriz 4x4x4.

Com o cubo funcionando praticamente da mesma forma que uma tela, as possibilidades disponíveis agora se tornam infinitas. Jogos podem ser desenvolvidos, integração com softwares de som fazendo com que os padrões de sinais enviados sigam a uma música ou qualquer faixa de som que for tocada. O objetivo inicial foi cumprido e abriram-se portas para mais possibilidades nesse projeto microcontrolado.

5 CONCLUSÃO

Além do objetivo de criar um projeto que fosse dependente de um microcontrolador, o principal objetivo da disciplina foi cumprido. A ideia inicial era desenvolver as habilidades do grupo na hora de integrar os conhecimentos de diversas matérias, trabalhar em equipe e ter um sistema de testes e desenvolvimento concreto. Foi possível aprender que não é simples desenvolver um projeto, mesmo que ele pareça ter um nível de complexidade baixo, do zero. Tanto as habilidades de desenvolver software como hardware dos integrantes do grupo foram desenvolvidas e com certeza todo o conhecimento obtido nesses meses de criação serão úteis para todo o resto do curso.

REFERÊNCIAS

74hc154 codificadores, decodificadores, multiplexadores e desmultiplexadores. <https://br.mouser.com/Search/Refine?Keyword=74HC154>, (accessed Junho, 2019).

Wikipédia. Porta not — wikipédia, a enciclopédia livre. <https://pt.wikipedia.org/w/index.php?title=Porta_NOT&oldid=50317320>, 2017. [Online; accessed Junho-2019].

Wikipédia. Multiplexador — wikipédia, a enciclopédia livre. <https://pt.wikipedia.org/w/index.php?title=Multiplexador&oldid=52851193>, 2018(accessed Junho, 2019.