

**ETEC “PROFª. ANNA DE OLIVEIRA FERRAZ”**

**Técnico em Mecatrônica**

**ARTHUR SCHIMITH  
JORGE FRANCISCO TOMÉ DOS SANTOS  
MURILO AKIRA FELIPE SHIGUEMOTO  
WESLEY DOS SANTOS CARMO**

**ESCAVADEIRA CONTROLADA REMOTAMENTE**

**Araraquara  
2018**

---

**ETEC "PROFª. ANNA DE OLIVEIRA FERRAZ"**

**Técnico em Mecatrônica**

**ARTHUR SCHIMITH  
JORGE FRANCISCO TOMÉ DOS SANTOS  
MURILO AKIRA FELIPE SHIGUEMOTO  
WESLEY DOS SANTOS CARMO**

**ESCAVADEIRA CONTROLADA REMOTAMENTE**

**Araraquara  
2018**



**ARTHUR SCHIMITH  
JORGE FRANCISCO TOMÉ DOS SANTOS  
MURILO AKIRA FELIPE SHIGUEMOTO  
WESLEY DOS SANTOS CARMO**

**ESCAVADEIRA CONTROLADA REMOTAMENTE**

Trabalho de Conclusão de Curso apresentado  
a ETEC "Prof.<sup>a</sup> Anna de Oliveira Ferraz", do  
Centro Estadual de Educação Tecnológica  
Paula Souza, como requisito para a obtenção  
do título de Técnico de Nível Médio em  
Mecatrônica sob a orientação do Professor  
Edson Mello.

**Araraquara  
2018**

## AGRADECIMENTOS

Primeiramente a Deus, por permitir que eu chegasse a esta etapa da minha vida, e a todos os que me apoiaram ao longo do caminho.

A minha família, que sempre me encorajou e me deu o apoio necessário para que eu pudesse enfrentar todos os desafios da vida.

A todos os professores do curso, em especial os professores Edson Filho, por sempre estarem presentes e ajudando-me em todas as dificuldades que eu enfrentei ao longo do curso. E também ao professor Roberto, por sempre estar presente e ajudando-me em todas as dificuldades que eu enfrentei ao longo do curso. E também ao professor Roberto, por sempre estar presente e ajudando-me em todas as dificuldades que eu enfrentei ao longo do curso.

A todos os amigos que me apoiaram e me encorajaram ao longo do curso, especialmente os amigos do curso de Engenharia de Software, por sempre estarem presentes e ajudando-me em todas as dificuldades que eu enfrentei ao longo do curso.

Dedicamos esta obra aos nossos familiares.

## AGRADECIMENTO

Primeiramente a Deus, independente da religião de cada integrante do grupo, pois sem ele nosso trabalho não seria concluído.

A nossos familiares, que sempre nos ofereceram apoio ao longo de nossa jornada, tanto moralmente quanto financeiramente.

A todos os professores de nosso curso, com destaque especial para os professores Edson Mello, por sempre oferecer ajuda no desenvolvimento da parte formal do projeto; ao professor Donizete, por sempre dar conselhos referentes a confecção do projeto; ao professor Jorge, por nos ensinar a utilizar os softwares de programação; ao professor Reinaldo Haddad por nos orientar também como coordenador do curso ao longo dos 4 módulos de ensino.

A todos os colegas que estudaram conosco, que colaboraram para o desenvolvimento do nosso projeto e para manter a boa convivência em sala de aula.



## RESUMO

Este trabalho apresenta a implementação do controle de uma máquina seqüencial a partir de dados de entrada por console remoto e memória que armazenam dados de saída. Os dados de entrada são processados por um programa em linguagem Pascal, que gera os dados de saída e os envia para o computador remoto. Os dados de saída são processados por um programa em linguagem Pascal, que gera os dados de saída e os envia para o computador remoto. Os dados de saída são processados por um programa em linguagem Pascal, que gera os dados de saída e os envia para o computador remoto.

Palavras-chave: Recursão, Contador, Remota, Máquina, Pascal, Programação, Híbrida.

**"Só sei que nada sei, e o fato de saber isso, me coloca em vantagem sobre aqueles que acham que sabem alguma coisa."**

**- Sócrates**



## RESUMO

Este projeto apresenta a automação do controle de uma retroescavadeira, a partir de ondas emitidas por controle remoto e motores que alteram a dimensão de seringas, usadas como pistões hidráulicos, para realizar os movimentos das esteiras e do braço da escavadeira, montados em um protótipo com fins didáticos, aplicando os conceitos de hidráulica, manufatura mecânica, elétrica e linguagens de programação. Foram utilizados para confecção deste, materiais como madeira, alumínio, seringas, motores e componentes microeletrônicos. Detalhes construtivos são apresentados, incluindo esquemático eletrônico e placa de circuito impresso para o protótipo.

**Palavras-chave:** Escavadeira. Controlada. Remotamente. Maquinário. Elétrica. Programação. Hidráulica.

## ABSTRACT

This project presents the automation of the control of a backhoe, from waves emitted by remote control and engines that change the size of syringes, used as hydraulic pistons, to realize the movements of the backhoe's arms, mounted in a prototype with for the purposes of teaching, applying the concepts of hydraulics, mechanical, electrical and programming languages. Materials such as wood, aluminum, syringes, motors and microelectronic components were used to make this. Constructive details are presented, including electronic schematic and printed circuit board for the prototype.

**Keywords:** Backhoe. Controlled. Remotely. Machinery. Electrical. Programming. Hydraulics.

## SUMÁRIO

|                                                                |    |
|----------------------------------------------------------------|----|
| INTRODUÇÃO .....                                               | 10 |
| 1 Estrutura .....                                              | 11 |
| 1.1 Esteira .....                                              | 11 |
| 1.2 “Braço” da escavadeira .....                               | 12 |
| 2 Hidráulica .....                                             | 13 |
| 2.1 Lei de Pascal .....                                        | 13 |
| 2.2 Relação entre as seringas, os parafusos e os motores ..... | 14 |
| 3 Circuito eletrônico .....                                    | 16 |
| 3.1 Circuito transmissor .....                                 | 16 |
| 3.2 Circuito receptor .....                                    | 17 |
| 3.2.1 Ponte H para motores DC .....                            | 18 |
| 3.2.1 Ponte H para motores de passo .....                      | 19 |
| 3.3 Componentes utilizados .....                               | 20 |
| 4 Programação .....                                            | 21 |
| 4.1 Programação do transmissor .....                           | 21 |
| 4.2 Programação do receptor .....                              | 23 |
| 5 Considerações finais ou conclusão .....                      | 30 |
| ANEXOS .....                                                   | 31 |
| REFERÊNCIAS .....                                              | 56 |
| APÊNDICES .....                                                | 57 |



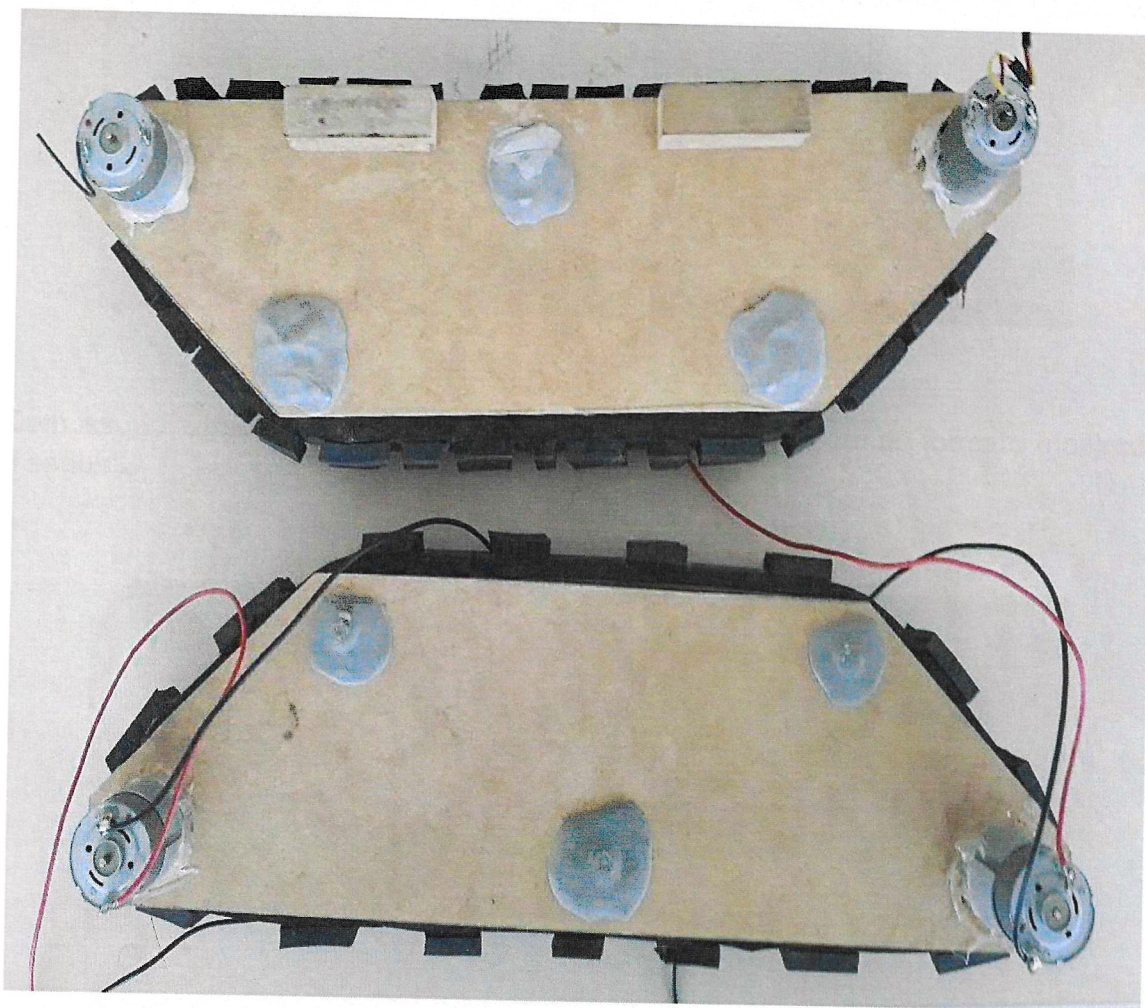
## 1 INTRODUÇÃO

A ideia inicial deste projeto foi obtida a partir de um evento que aconteceu na ETEC no qual os estudantes mostrariam um pouco de seus conhecimentos adquiridos ao longo de seus estudos. Nosso grupo participou e construímos um capacete-móvel, utilizando um capacete como carcaça para transforma-lo em um carrinho de controle por fio. Infelizmente ele não funcionou, mas nos deu a inspiração para tocar para frente nosso trabalho de TCC, à escavadeira Controlada Remotamente. Este projeto tem como o intuito de demonstrar algumas matérias aprendidas durante os quatro (4) módulos do curso, tais como manufatura mecânica, hidráulica, elétrica e linguagens de programação, além de facilitar o controle deste tipo de maquinário. Primeiramente foi utilizado os programas MikroC Pro e ISIS para a elaboração da programação e esquematização dos circuitos elétricos, visando o teste teórico do funcionamento do projeto antes da confecção das placas, na qual foi realizado usando o programa ARES. Com esta parte concluída, passou-se para a próxima etapa, a construção do projeto. Começando com a carcaça, feita a base de madeira MDF, motores DC e de passo e utilização de seringas como êmbolos hidráulicos. A carcaça pode ser dividida em duas partes, superior que consiste no braço hidráulico e os motores que o controlam, e a parte inferior, que seria responsável pela locomoção da base, através de uma esteira lagarta feita com motores DC e tiras de borracha, entre outras coisas. Após a confecção de todas as partes, as mesmas então foram unidas para formar o projeto final.

## Capítulo 1: Estrutura – Esteira; “Braço” da escavadeira

### - Esteira

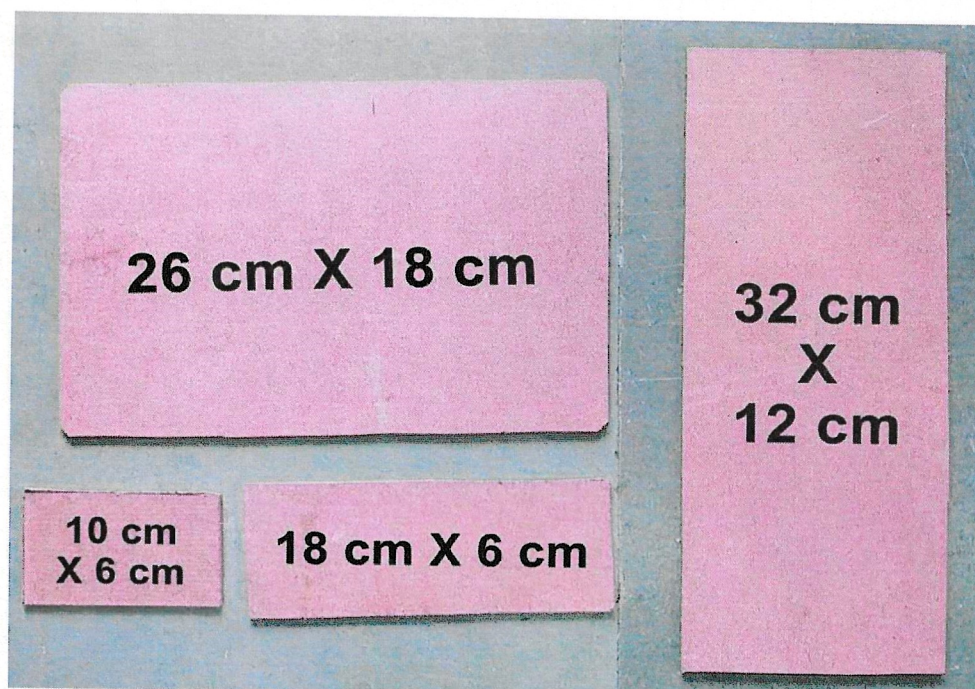
Para a locomoção da estrutura foram feitas duas esteiras “lagarta”. Nelas foram colocados quatro motores DC, dois em cada uma. Desse modo, ao girarem, será gerada força necessária para rodar as tiras de borracha e assim realizar a locomoção.



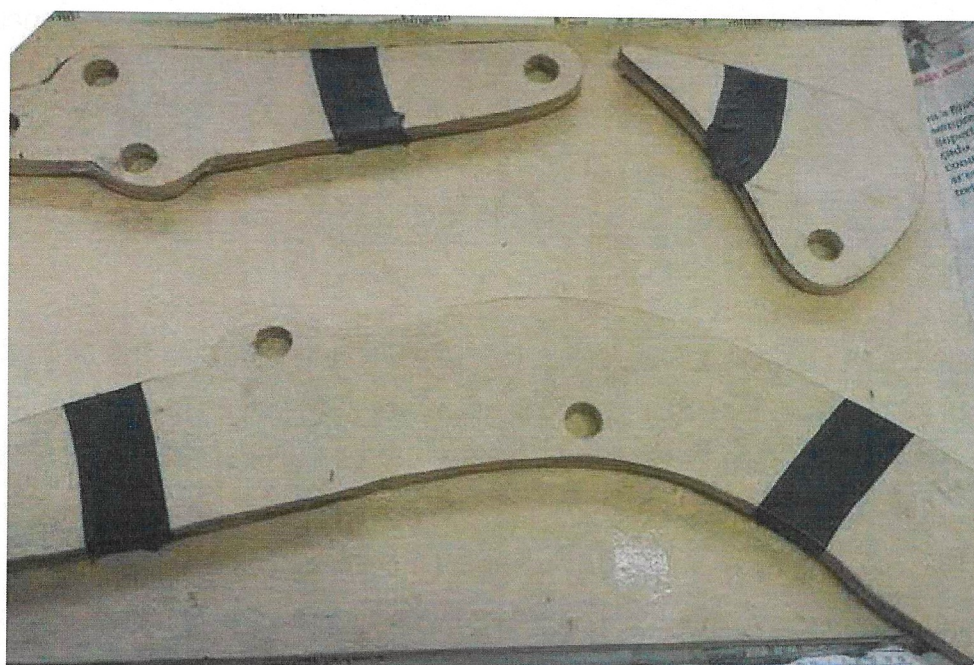
### - “Braço” da escavadeira



Para a base e o braço foram utilizadas placas de **MDF** com as seguintes dimensões:

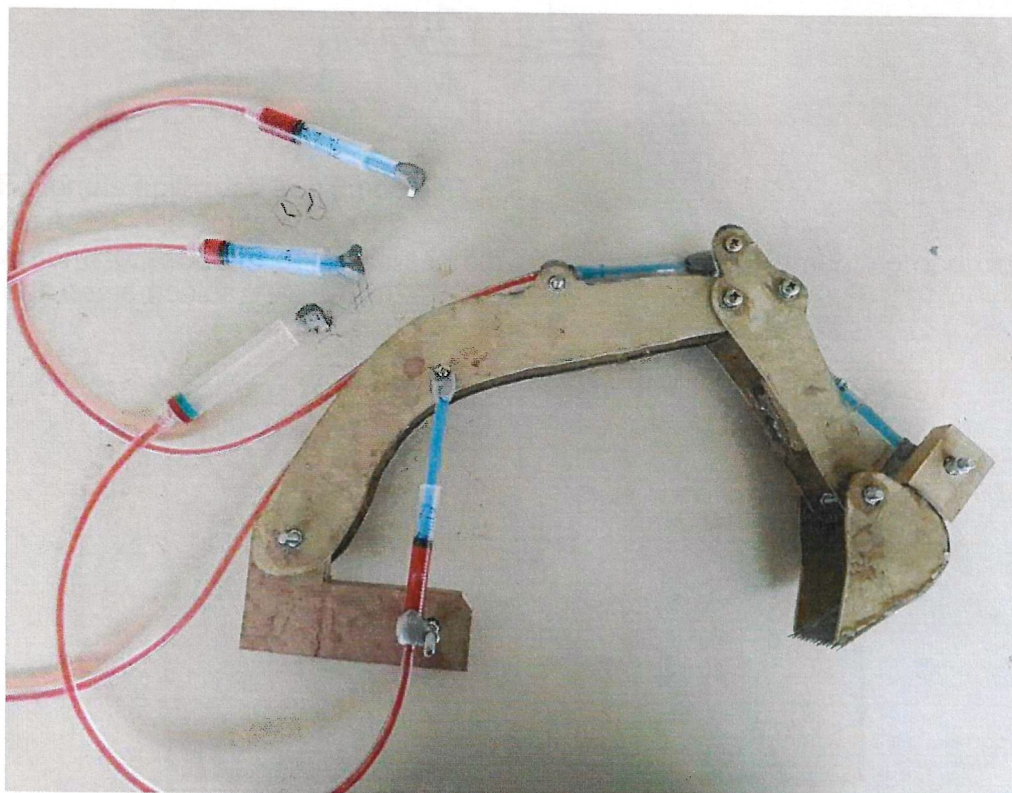


Com isso foi trabalhado nas placas de **MDF** para que ficassem do formato mostrado a seguir:





Segue aqui o braço montado e já conectado às seringas.



## Capítulo 2: Hidráulica — Lei de Pascal; Relação entre as seringas, os parafusos e os motores

### - Lei de Pascal

No projeto foram utilizados conceitos de hidráulica para proporcionar movimento ao “braço” da miniatura de escavadeira, assim como em uma real. Tal ideia foi formulada em 1653 pelo matemático, físico e filósofo francês Blaise Pascal, e em sua homenagem, a mesma foi intitulada de Lei de Pascal.

Seu enunciado é expresso da seguinte maneira: *“O aumento da pressão exercida em um líquido em equilíbrio é transmitido integralmente a todos os pontos do líquido bem como às paredes do recipiente em que ele está contido.”*



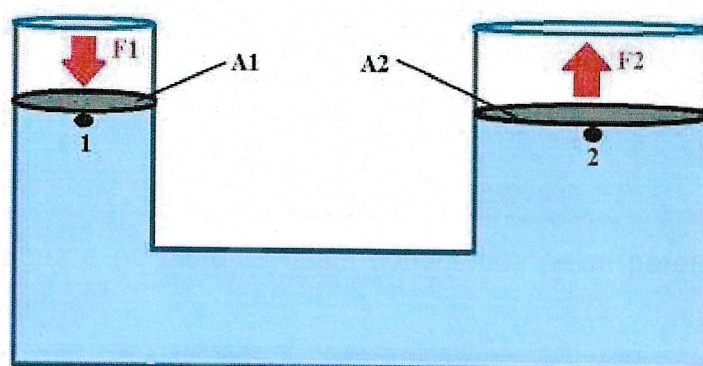
A partir da figura abaixo, a fórmula do Princípio de Pascal é expressa:

$$\frac{F_1}{A_1} = \frac{F_2}{A_2}$$

**$F_1$  e  $F_2$ :** forças aplicadas aos êmbolos 1 e 2

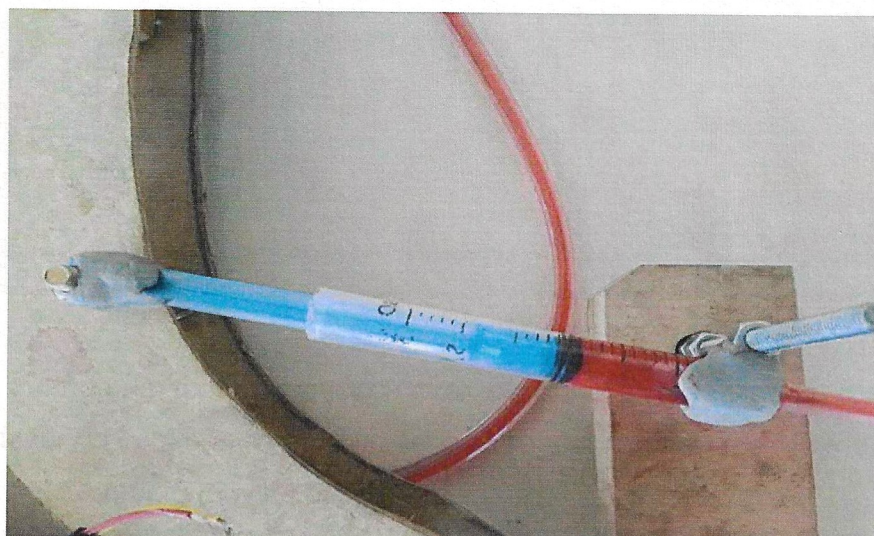
**$A_1$  e  $A_2$ :** áreas dos êmbolos 1 e 2

Nesse sentido, as intensidades das forças aplicadas são diretamente proporcionais às áreas dos êmbolos.



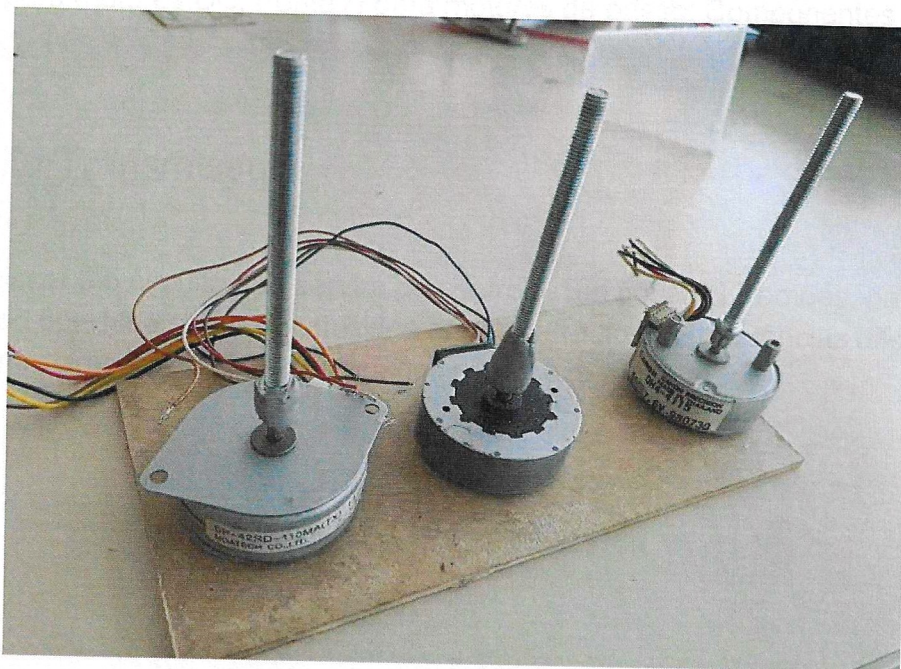
### - Relação entre as seringas, os parafusos e os motores

Para exercer força sobre as seringas, foram colocadas porcas em suas extremidades com Durepoxi, como mostra a figura a seguir.

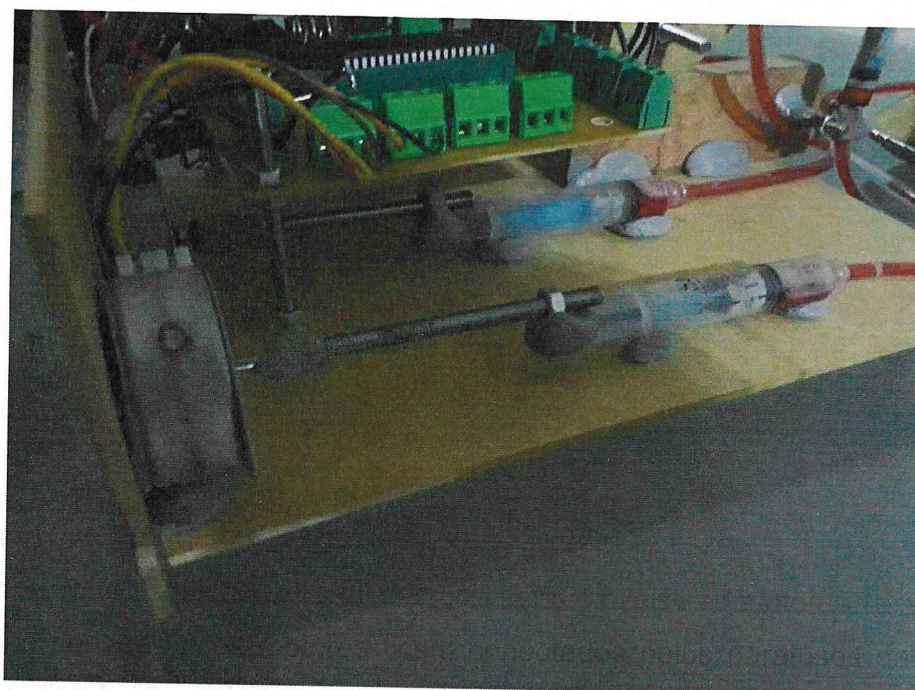




Em seguida, foram colocados parafusos nos eixos móveis dos motores.



E então as seringas e os motores ficam conectados pelos parafusos da seguinte forma.



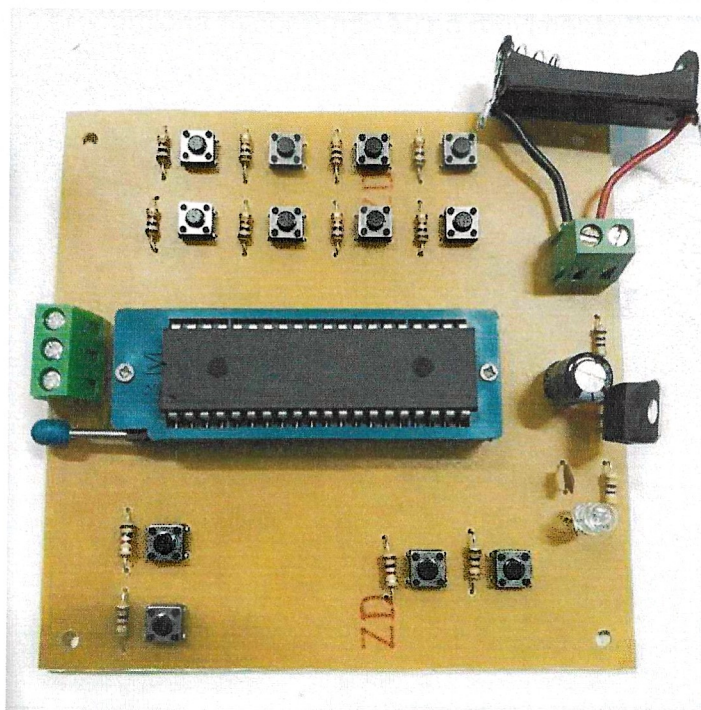
Com isso, quando um motor começar a girar, ele começará a rosquear o parafuso na rosca, fazendo com que a haste da seringa seja puxada ou empurrada, proporcionando então o movimento hidráulico.



### Capítulo 3: Circuito eletrônico — Circuito transmissor; Circuito receptor; Ponte H para motores DC; Ponte H para motores de passo; Componentes utilizados

#### - Circuito Transmissor

O transmissor é o circuito que servirá de controle. Ele possui 12 botões, na qual para cada botão apertado serão transmitidas ondas de rádio em frequências diferentes e assim controlarão os motores conectados no circuito receptor.



Forma final do circuito após montagem

Como mostrado na imagem acima, os botões estão divididos em dois grupos, 8 deles acima do pic 16F887, e 4 abaixo. Os de cima controlarão os movimentos dos motores que estão conectados às seringas e à estrutura superior. Já os de baixo controlarão os motores ligados às rodas da esteira.

Tal distribuição dos botões foi feita com o objetivo de auxiliar o usuário na identificação de qual parte da estrutura cada botão exerce controle.

Note também que os botões inferiores foram dimensionados de forma intuitiva em relação à suas funções de controle da locomoção da estrutura.

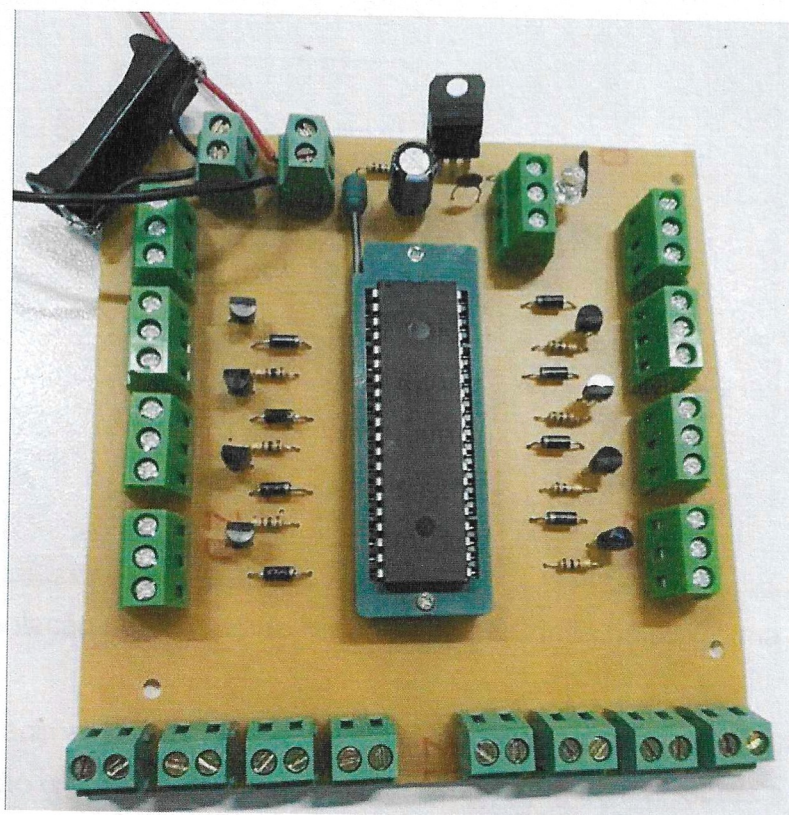


## - Circuito Receptor

O receptor é o circuito que captará as ondas de rádio emitidas pelo transmissor e, para cada um dos 12 formatos de onda que podem ser recebidos, será dado um comando diferente para os 8 motores que nele serão conectados.

Com relação aos motores, 4 deles são motores DC e estarão presentes na esteira para a locomoção e foram escolhidos para este propósito devido a sua maior facilidade de controle, visto que sua finalidade é mais simples.

O restante são motores de passo, na qual apesar de sua maior complexidade de funcionamento e controle, se viu necessária a sua utilização para o controle do braço, visto que o mesmo proporciona maior precisão no movimento.

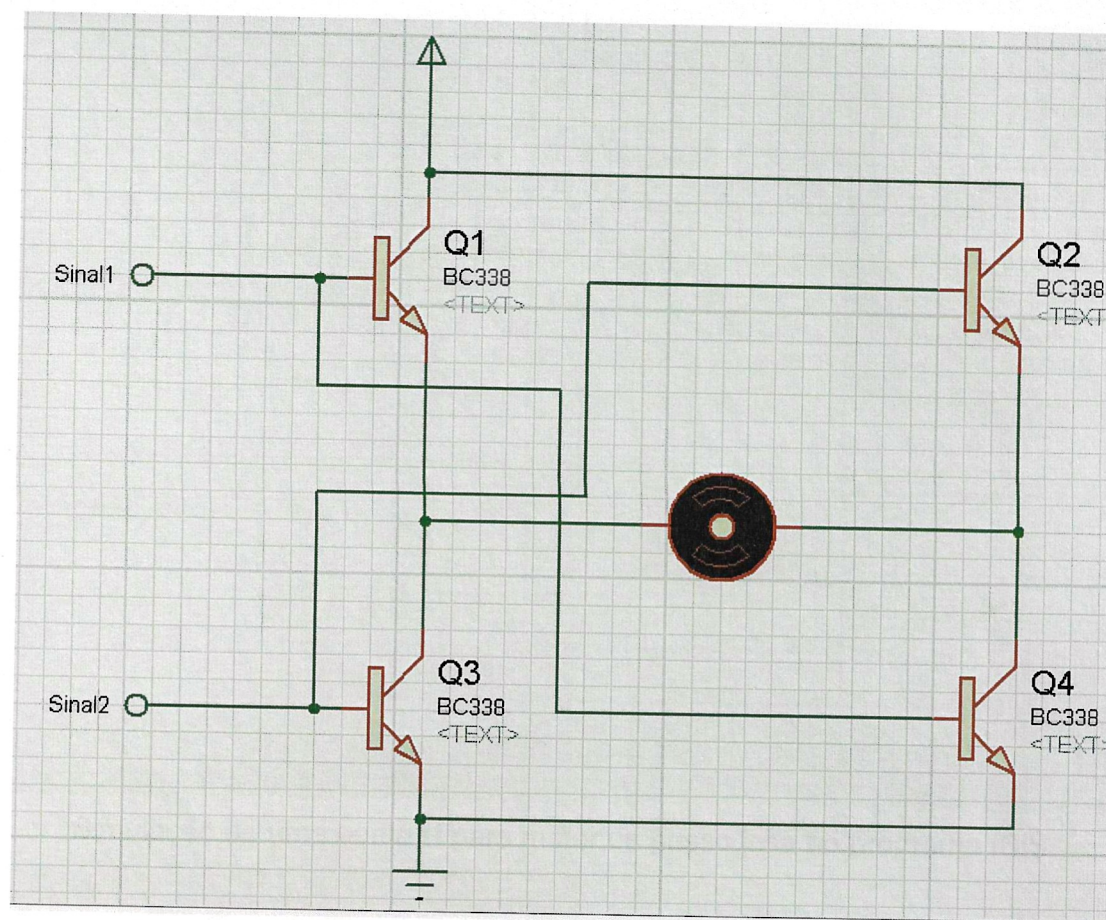


Forma final do circuito principal após montagem



## - Ponte H para motores DC

A “ponte H” é um circuito que tem o objetivo de controlar o sentido de rotação de motores e possui diversas variações. A imagem a seguir mostra uma ponte H para motores DC destinada a circuitos microcontrolados.



Simulação de uma ponte H para motor DC feita no programa ISIS

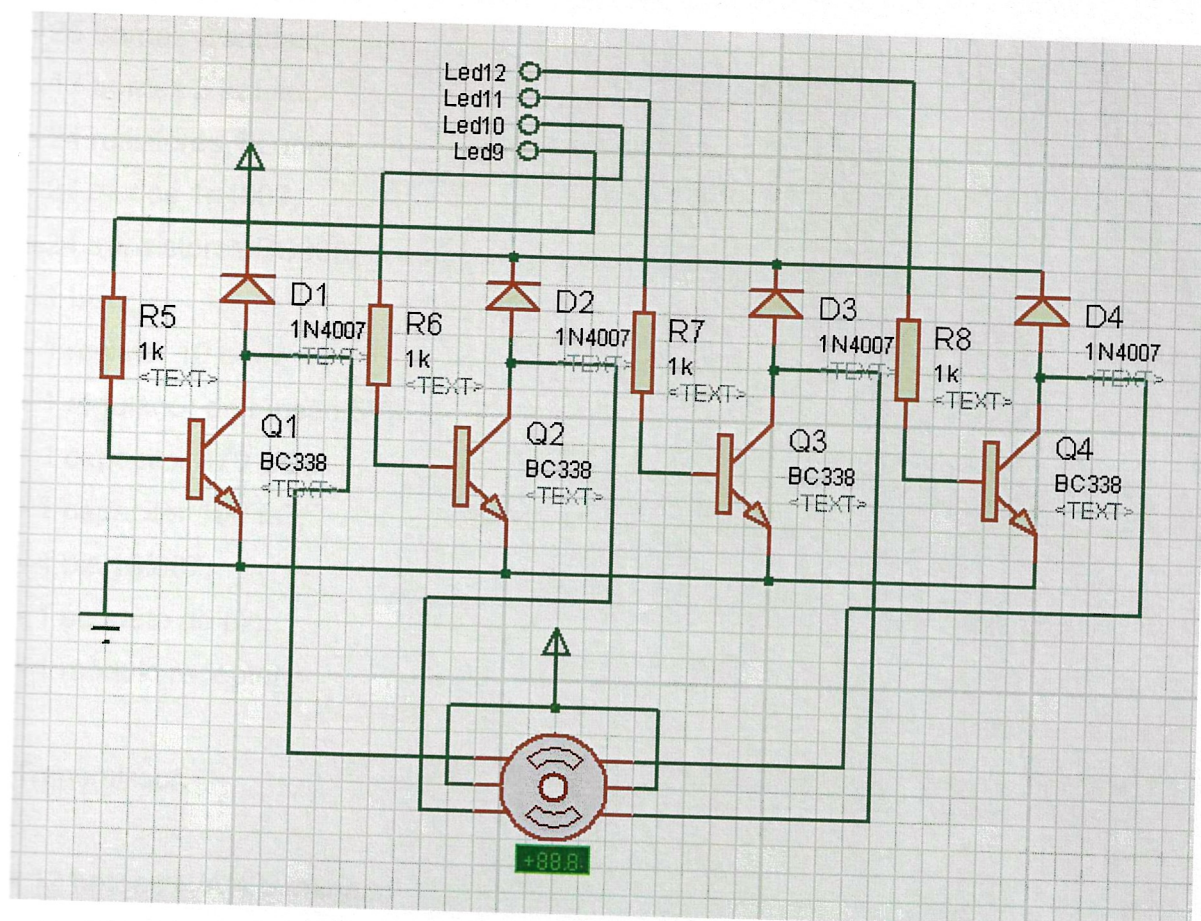
No esquema acima, se a saída “Sinal1” for energizada, as bases dos transistores Q1 e Q4 receberão o pulso necessário para fazer com que os emissores e coletores dos mesmos passem a conduzir corrente. Assim cria-se um “caminho” que conecta o VCC com o Terra, na qual o motor está entre os dois e começará a girar.

Agora se a saída “Sinal2” for energizada, o mesmo processo acontecerá, porém, dessa vez envolvendo os transistores Q2 e Q3, e isso fará com que a corrente passe pelo motor no sentido contrário ao explicado anteriormente, e assim ele girará no sentido oposto.



## - Ponte H para motores de passo

Já este modelo visa o controle de motores de passo.



Simulação de uma ponte H para motor de passo feita no programa ISIS

A ideia do circuito novamente utiliza do transistor com chave condutora, na qual quando energizado o coletor, a liberação da corrente conectará a saída de uma das bobinas do motor para o aterramento, fazendo com que a mesma se energize e então o motor de um “passo”. Assim, se os transistores forem “acionados” na ordem correta, será possível fazer o motor rodar na direção desejada.

Obs.: Após essas explicações, pode-se conferir toda a esquematização do circuito e do design das placas feitos, respectivamente nos programas ISIS e ARES na seção de anexos.

## - Componentes utilizados

### Receptor:

- 25 bornes 2 saídas;
- 17 bornes 3 saídas;
- 24 resistores 1k ohm;
- 24 diodos 1n4007;
- 24 transistores BC338;
- 1 LED;
- 1 resistor 10 ohm;
- 1 regulador de tensão 5V;
- 1 capacitor 470 UC;
- 1 capacitor 100 nC;
- 1 pic 16f887;
- 1 soquete de pic;
- 1 módulo receptor.

### Transmissor:

- 12 botões N/A 4 terminais;
- 1 borne 3 saídas;
- 1 borne 2 saídas;
- 12 resistores 1k ohm;
- 1 resistor 10 ohm;
- 1 regulador de tensão 5V;
- 1 capacitor 470 UC;
- 1 capacitor 100 nC;
- 1 pic 16f887;
- 1 soquete de pic;
- 1 módulo transmissor;
- 1 LED.



## Capítulo 4: Programação — Programação do transmissor; Programação do receptor

Cada um dos circuitos terá uma programação diferente, porém os dois estão voltados para o mesmo fator, o formato da onda.

O receptor por sua vez possui um teste de comparação para determinar se o sinal recebido corresponde a alguma das 12 ondas que o controle transmite, e identificar qual delas é. Depois disso, é dado um comando aos motores correspondente a onda que foi recebida.

### - Programação do transmissor

Na programação do transmissor, a ideia principal é que para cada botão que for pressionado, a saída RC4 do pic pulsará em um padrão específico diferente, e o módulo de transmissão transformará essa frequência de pulso em sinal de rádio.

Segue agora a programação em código C feita no programa MikroC for PIC, porém fragmentada para ser explicada cada parte.

Primeiramente foi estabelecido tudo o que será conectado no microcontrolador e em qual pino.

```
sbit botao1 at rc2_bit;  
sbit botao2 at rc3_bit;  
sbit botao3 at rc0_bit;  
sbit botao4 at rc1_bit;  
sbit sinal_tx at rb7_bit;  
sbit botao5 at rd0_bit;  
sbit botao6 at rd1_bit;  
sbit botao7 at rd2_bit;  
sbit botao8 at rd3_bit;  
sbit botao9 at rd4_bit;  
sbit botao10 at rd5_bit;  
sbit botao11 at rd6_bit;  
sbit botao12 at rd7_bit;
```

A seguir foram criados os blocos de comando “bit\_1()” e “bit\_0()”, sendo eles comandos para duas frequências de nível lógico alto e nulo no pino rb7 (apelidado no programa de “sinal\_tx”), nas quais as duas são opostas umas às outras.

```
void bit_1()
{
    sinal_tx = 1; delay_us(60);
    sinal_tx = 0; delay_us(30);
}

void bit_0()
{
    sinal_tx = 1; delay_us(30);
    sinal_tx = 0; delay_us(60);
}
```

Aqui começa então a parte principal do programa com a abertura do “void main()”, assim como o seu “cabeçalho” para que seja feita uma segunda configuração nos pinos do pic.

```
void main()
{ //1
    intcon=0x00;
    clon_bit = 0; c2on_bit = 0;
    adon_bit = 0; ansel = 0; anselh = 0;
    trisb=0b00000000;
    trisc=0b00001111;
    trisd=0b11111111;
    sinal_tx = 0;
```



Nessa parte define-se uma sequência de bit\_0's e bit\_1's para o caso de algum botão ser acionado, e nisso cada um produzirá uma frequência distinta de pulsações no "sinal\_tx".

```
while(1)
{
    //Esteira
    if(botao1 == 1)
    {
        bit_1();
        bit_0();
        bit_0();
        bit_0();
    }

    if(botao2 == 1)
    {
        bit_1();
        bit_0();
        bit_0();
        bit_1();
    }

    if(botao3 == 1)
    {
        bit_1();
        bit_0();
        bit_1();
        bit_0();
    }

    if(botao4 == 1)
    {
        bit_1();
        bit_0();
        bit_1();
        bit_1();
    }
}
```

## - Programação do receptor

A programação do circuito receptor constitui-se basicamente um teste de comparação para determinar se o sinal recebido corresponde a alguma das 12 ondas que o controle transmite, e identificar qual delas é. Depois disso, é dado um comando aos motores correspondente a onda que foi recebida.

Assim como foi feito anteriormente, será mostrada a programação em código C feita no programa MikroC for PIC de forma dividida em vários trechos para se propor uma explicação clara do que cada parte faz.

Para o início têm-se a definição dos conectivos de cada de cada pino do pic 16F887. Os diversos "LEDs" presentes no código na verdade representam as bobinas dos motores de passo e as saídas dos motores DC.

```

sbit sinal_tx at rb7_bit;
sbit led1 at ra1_bit;
sbit led2 at ra2_bit;
sbit led3 at ra3_bit;
sbit led4 at ra4_bit;
sbit led5 at rb0_bit;
sbit led6 at rb1_bit;
sbit led7 at rb2_bit;
sbit led8 at rb3_bit;

sbit led9 at rc0_bit;
sbit led10 at rc1_bit;
sbit led11 at rc2_bit;
sbit led12 at rc3_bit;
sbit led13 at rc4_bit;
sbit led14 at rc5_bit;
sbit led15 at rc6_bit;
sbit led16 at rc7_bit;
sbit led17 at rd0_bit;
sbit led18 at rd1_bit;
sbit led19 at rd2_bit;
sbit led20 at rd3_bit;
sbit led21 at rd4_bit;
sbit led22 at rd5_bit;
sbit led23 at rd6_bit;
sbit led24 at rd7_bit;

```

Logo em seguida vem a criação das variáveis que serão usadas ao longo do código.

```

char bit_1;
char bit_0;
char start_bit;
char end_bit;
char cont3;
char cont4;
char cont5;
char cont6;

```

Agora vem a parte que pode ser considerada a mais importante e a base para a realização do “teste de comparação” mencionado anteriormente. Neste bloco, denominado de “verificacao\_bit()”, é realizada uma verificação de qual dos dois



formatos de onda transmitidos pelo transmissor está sendo recebido. Se for reconhecido o “bit\_1()”, será dado valor igual a 1 à variável “bit\_1” e 0 à variável “bit\_0”, e o contrário ocorrerá se a onda for avaliada como sendo a “bit\_0()”.

Obs.: Os termos “bit\_1()” e “bit\_0()” correspondem aos mesmos citados na explicação da programação do transmissor.

```
void verificacao_bit()
{
    while(sinal_tx == 0);
    delay_us(15);
    if(sinal_tx == 1)
    {
        delay_us(30);
        if(sinal_tx == 0)
        {
            bit_0 = 1;
            bit_1 = 0;
        }
        if(sinal_tx == 1)
        {
            delay_us(30);
            if(sinal_tx == 0)
            {
                bit_1 = 1;
                bit_0 = 0;
            }
        }
    }
}
```

Aqui inicia-se a parte principal do código e é colocado o “cabeçalho” para que o pic seja configurado devidamente sobre quais pinos são entradas e quais são saídas.

```
void main()
{ //1

    intcon=0x00;
    clon_bit = 0; c2on_bit = 0;
    adon_bit = 0; ansel = 0; anselh = 0;
    trisa=0b00000000;
    trisb=0b10000000;
    trisc=0b00000000;
    trisd=0b00000000;
```

Em seguida é estabelecido nível lógico nulo nas bobinas e saídas dos motores. Essa parte do código ocorre logo após o circuito ser ligado, sendo assim sua tarefa é a de garantir que nada seja ligado sem o devido comando. Também é adicionado às variáveis "cont3", "cont4", "cont5" e "cont6" valor igual a 1.

```
led1 = 0;  
led2 = 0;  
led3 = 0;  
led4 = 0;  
led5 = 0;  
led6 = 0;  
led7 = 0;  
led8 = 0;  
led9 = 0;  
led10 = 0;  
led11 = 0;  
led12 = 0;  
led13 = 0;  
led14 = 0;  
led15 = 0;  
led16 = 0;  
led17 = 0;  
led18 = 0;  
led19 = 0;  
led20 = 0;  
led21 = 0;  
led22 = 0;  
led23 = 0;  
led24 = 0;
```

```
cont3 = 1;  
cont4 = 1;  
cont5 = 1;  
cont6 = 1;
```

Após a entrada no "while(1)", as bobinas e saídas novamente são "desligadas", porém agora com uma função diferente. Fazendo isso nessa parte garante que nenhum motor continuará energizado após o usuário para de apertar os botões no controle.



```
while(1)
{ //2

    led1 = 0;
    led2 = 0;
    led3 = 0;
    led4 = 0;
    led5 = 0;
    led6 = 0;
    led7 = 0;
    led8 = 0;
    led9 = 0;
    led10 = 0;
    led11 = 0;
    led12 = 0;
    led13 = 0;
    led14 = 0;
    led15 = 0;
    led16 = 0;
    led17 = 0;
    led18 = 0;
    led19 = 0;
    led20 = 0;
    led21 = 0;
    led22 = 0;
    led23 = 0;
    led24 = 0;
```

Enfim chega-se ao “teste de comparação”. Tendo em mente a explicação do bloco “verificacao\_bit()”, ao apertar o botão 1 do controle (transmissor), será transmitida a sequência de onda: “bit\_1”, “bit\_0”, “bit\_0”, “bit\_1”. Apertando o botão 2 têm-se: “bit\_1”, “bit\_0”, “bit\_0”, “bit\_0”. No exemplo abaixo, ao receber o sinal, o pic verificará se trata-se de um “bit\_1” ou “bit\_0”, se for um “bit\_1” será feita então uma nova verificação, continuando assim sucessivamente até que a sequência enviada seja completamente reconhecida, e com isso será dado um comando desejado para os motores. A imagem abaixo mostra apenas o reconhecimento de duas ondas, porém o processo se estende muito mais para atender a necessidade de 12 comandos diferentes.

```

verificacao_bit();
if(bit_1 == 1)
{
    verificacao_bit();
    if(bit_0 == 1)
    {
        verificacao_bit();
        if(bit_0 == 1)
        {
            verificacao_bit();
            if(bit_1 == 1)
            {
                //codigo1
                led13 = 1;
                led14 = 0;
                led15 = 1;
                led16 = 0;
            }
            else
            {
                //codigo2
                if(bit_0 == 1)
                {
                    led14 = 1;
                    led13 = 0;
                    led16 = 1;
                    led15 = 0;
                }
            }
        }
    }
}

```

Feito o teste, é estabelecido laços para manter os valores dos "cont's" em um ciclo infinito.

```

if(cont3 > 8)
{
    cont3 = 1;
}

if(cont4 > 8)
{
    cont4 = 1;
}

if(cont3 < 1)
{
    cont3 = 8;
}

if(cont4 < 1)
{
    cont4 = 8;
}

```



No final é estabelecida uma relação entre os valores dos “cont’s” com as bobinas dos motores de passo para que sejam energizadas em sequência à medida q o valor de seu “cont” correspondente aumenta ou diminui seu valor. Para terminar é definida uma espera de 300 milisegundos com o objetivo de eliminar as interferências e os error de identificação de onda.

```
led5=cont3.B0;  
led6=cont3.B1;  
led7=cont3.B2;  
led8=cont3.B3;  
led1=cont4.B0;  
led2=cont4.B1;  
led3=cont4.B2;  
led4=cont4.B3;  
led17=cont5.B0;  
led18=cont5.B1;  
led19=cont5.B2;  
led20=cont5.B3;  
led21=cont6.B0;  
led22=cont6.B1;  
led23=cont6.B2;  
led24=cont6.B3;
```

```
delay_ms(300);  
start_bit = 0;  
end_bit = 0;
```

```
}
```

```
}
```

Obs.: Após essas explicações, pode-se conferir as duas programações na íntegra na seção de anexos.

## CONSIDERAÇÕES FINAIS OU CONCLUSÃO

Apesar dos imprevistos de última hora, a montagem da estrutura foi completada como o esperado.

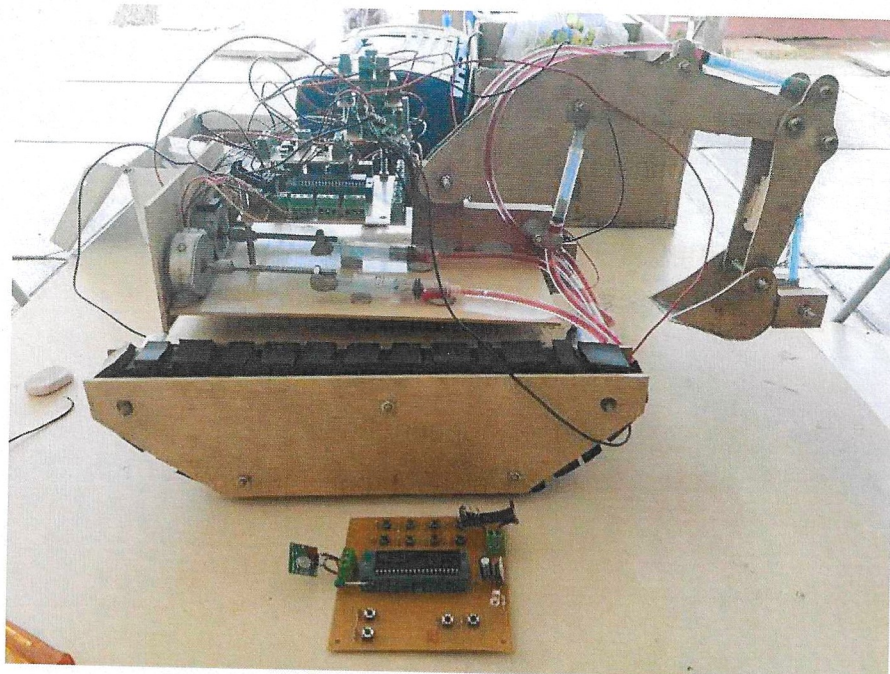
O circuito também segue praticamente sem defeitos, com exceção a um driver de motor de passo presente na placa principal na qual houve má energização. Todavia não houve ameaça quanto a isso, pois na placa foram dimensionados bornes diretamente ligados ao pic, o que possibilita a fácil substituição do defeituoso.

Porém, o projeto apresentou algumas falhas com relação à programação. O teste de comparação muitas vezes confundia uma onda com a outra, gerando comandos não esperados. A transmissão também sofreu interferência de outras frequências presentes no ar, que faziam a onda recebida mudar drasticamente de formato e assim nenhum comando era dado.

Erros de cálculo sobre o peso da estrutura final e o torque necessário nos motores também contribuíram para o mau funcionamento do TCC.

Todavia, mesmo que o mesmo não tenha funcionado, o desenvolvimento do projeto nos proporcionou uma importante experiência a ser levada ao nosso lado profissional. Assim, os diversos erros cometidos por nós, integrantes deste TCC, serão mantidos em nossas mentes a fim de serem evitados em projetos futuros.

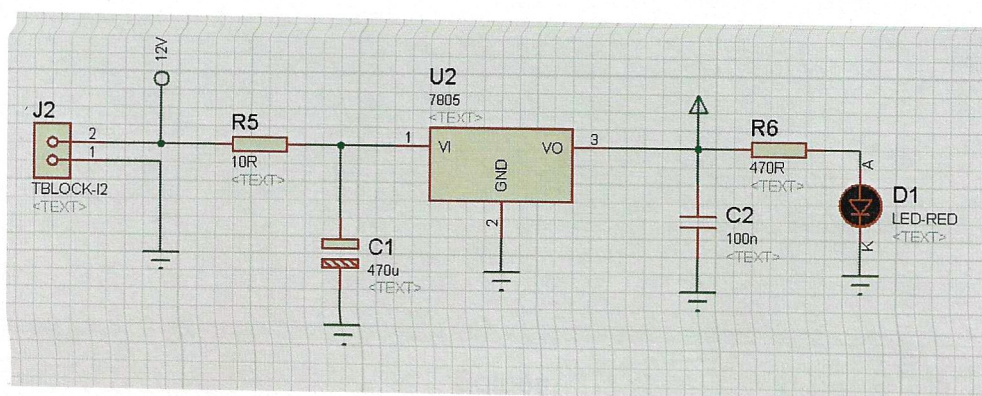
A seguir uma imagem do TCC em sua forma final.



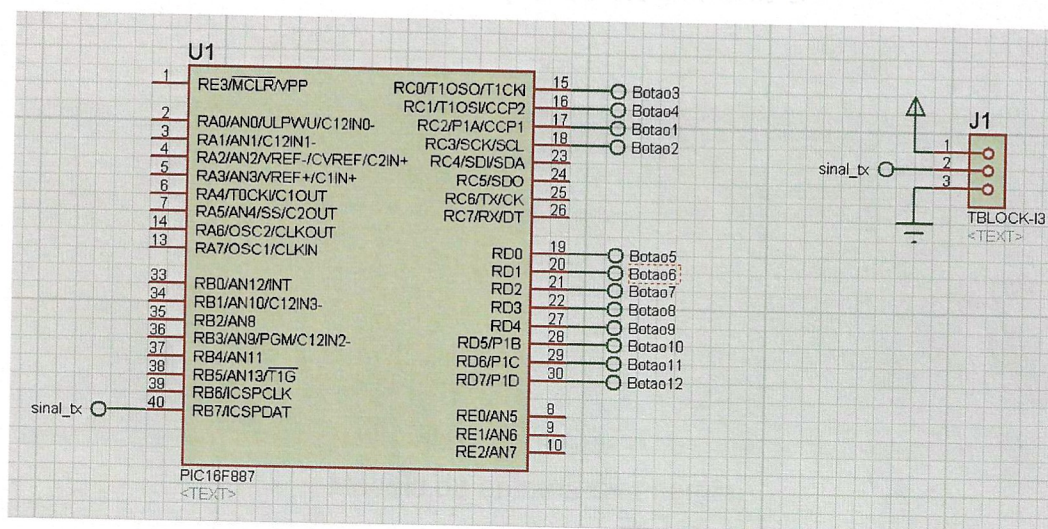


## Anexos

- Imagens da simulação do circuito transmissor no programa ISIS

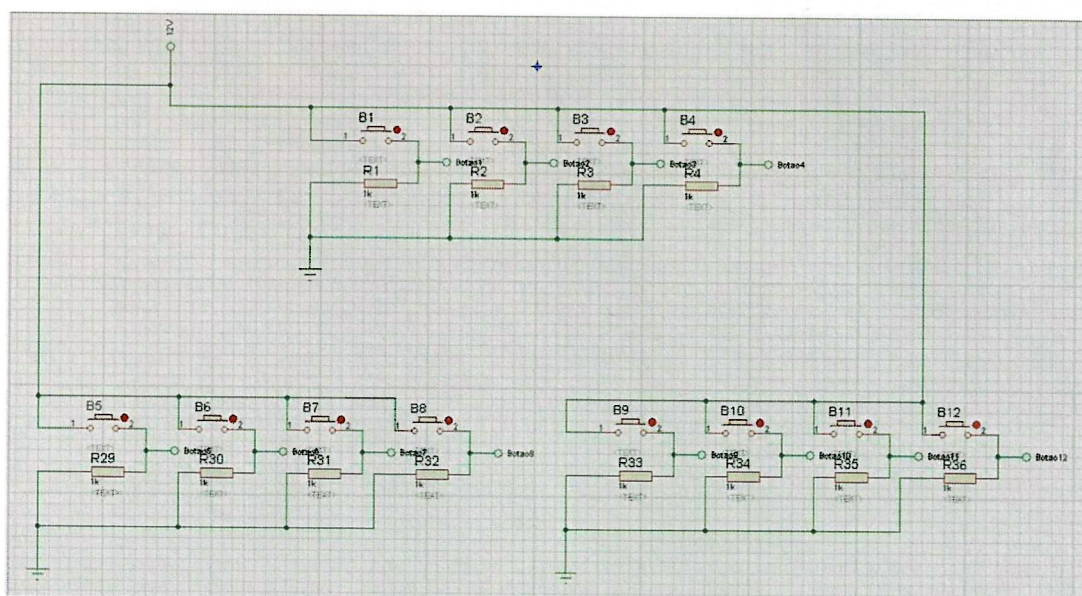


Módulo de entrada de energia.



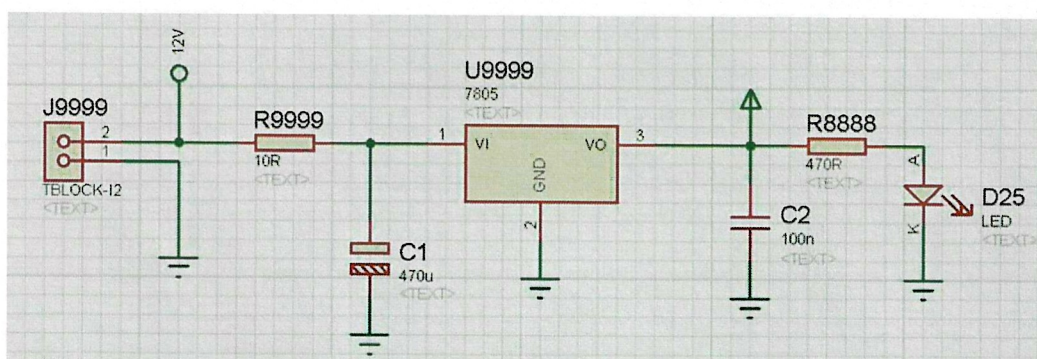
Pic 16F887 e entradas para o módulo de transmissão.





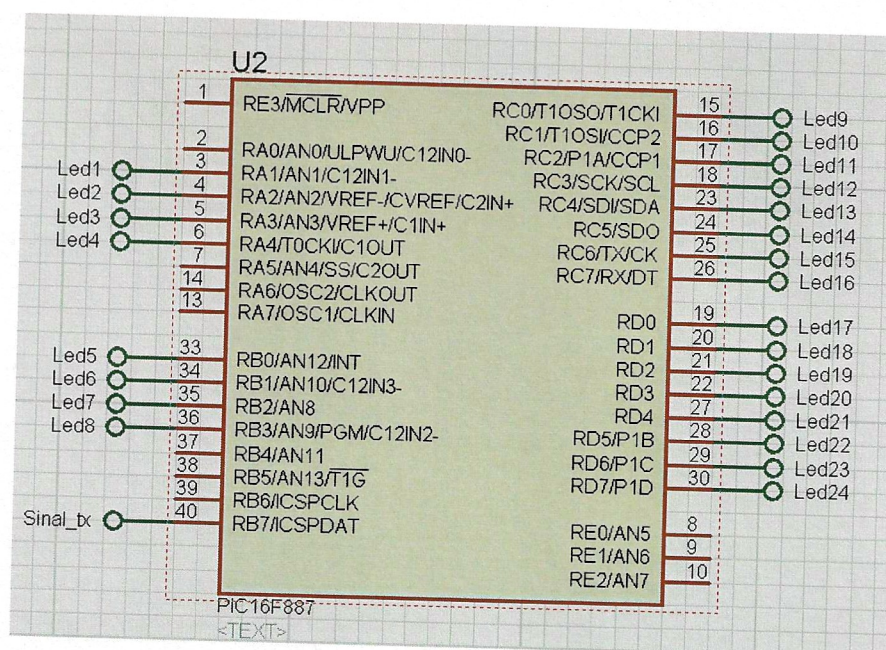
Botões.

- Imagens da simulação do circuito receptor no programa ISIS

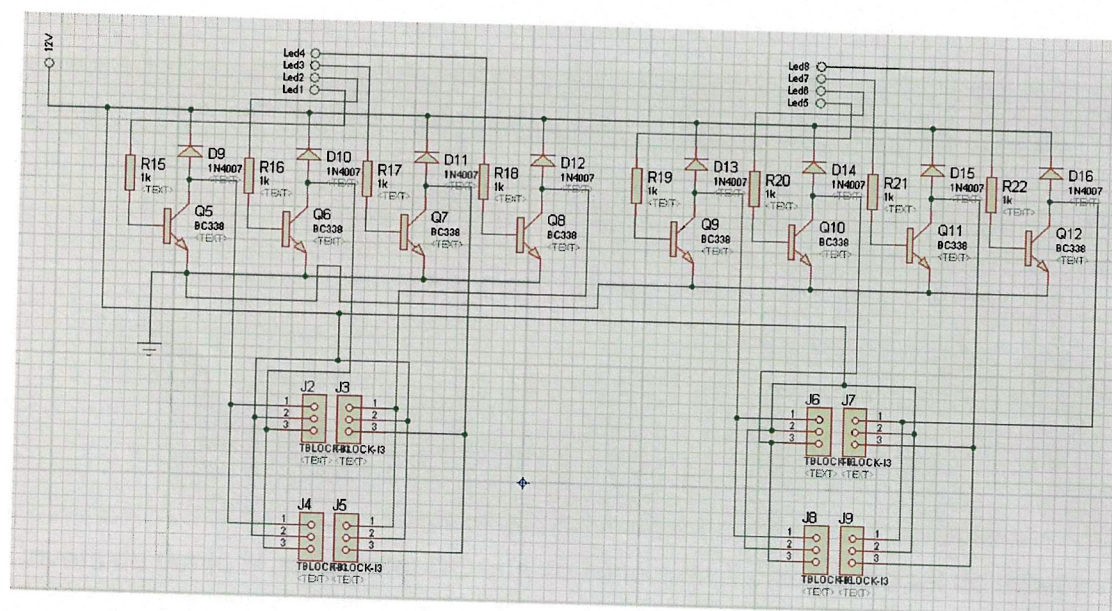


Módulo de entrada de energia.



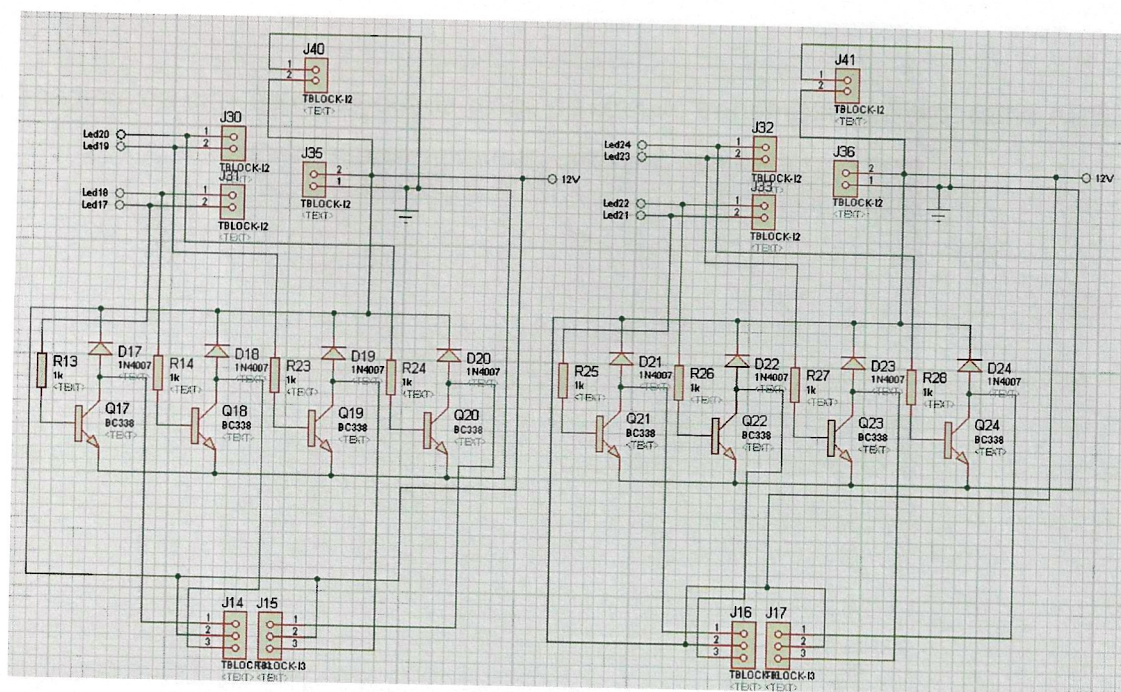


Pic 16F887.

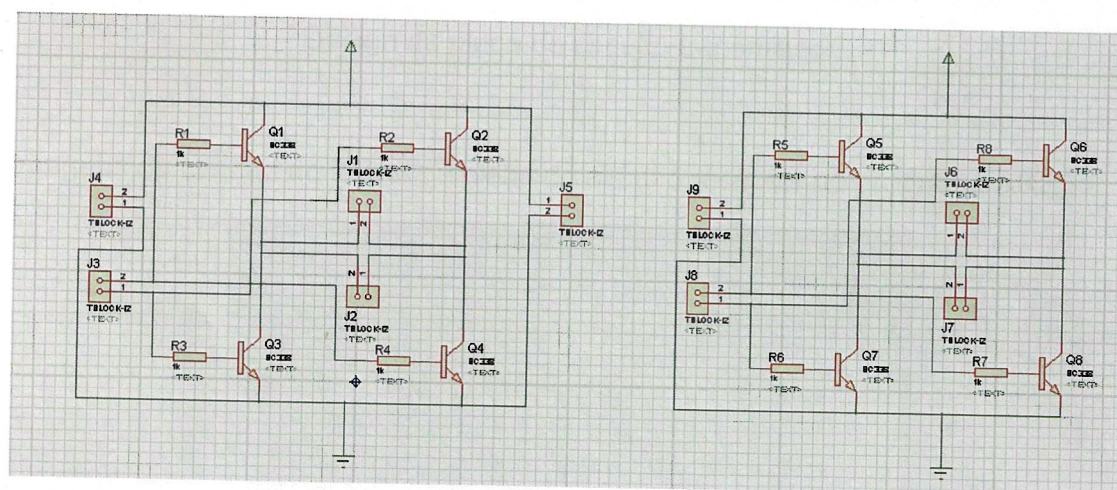


Drivers de motores de passo presentes na placa principal do receptor.





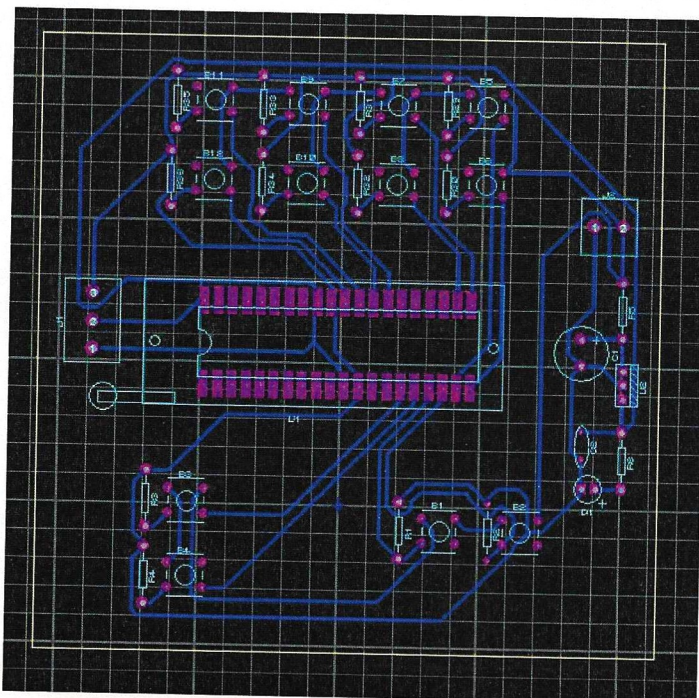
Drivers de motor de passo que se conectam à placa principal do receptor.



Drivers de ponte H para motor DC que se conectam à placa principal do receptor.

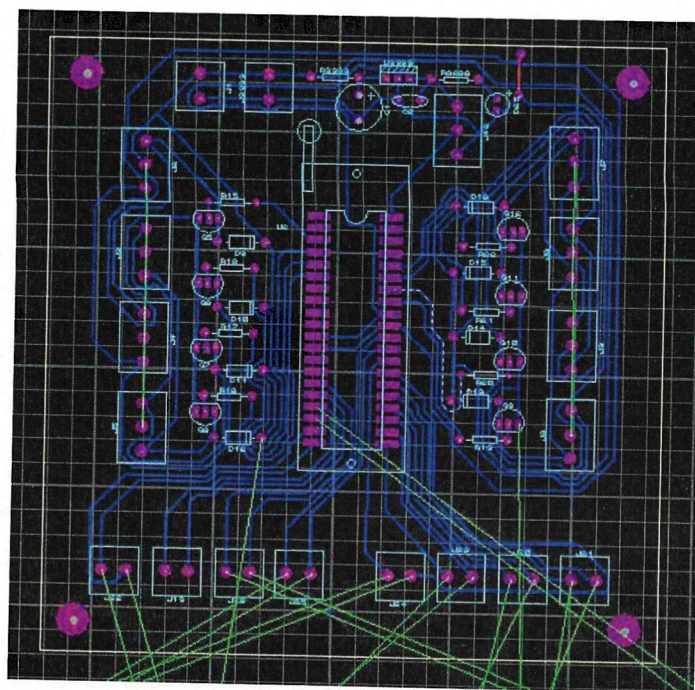


- Imagens da criação do layout da placa do circuito transmissor no programa ARES



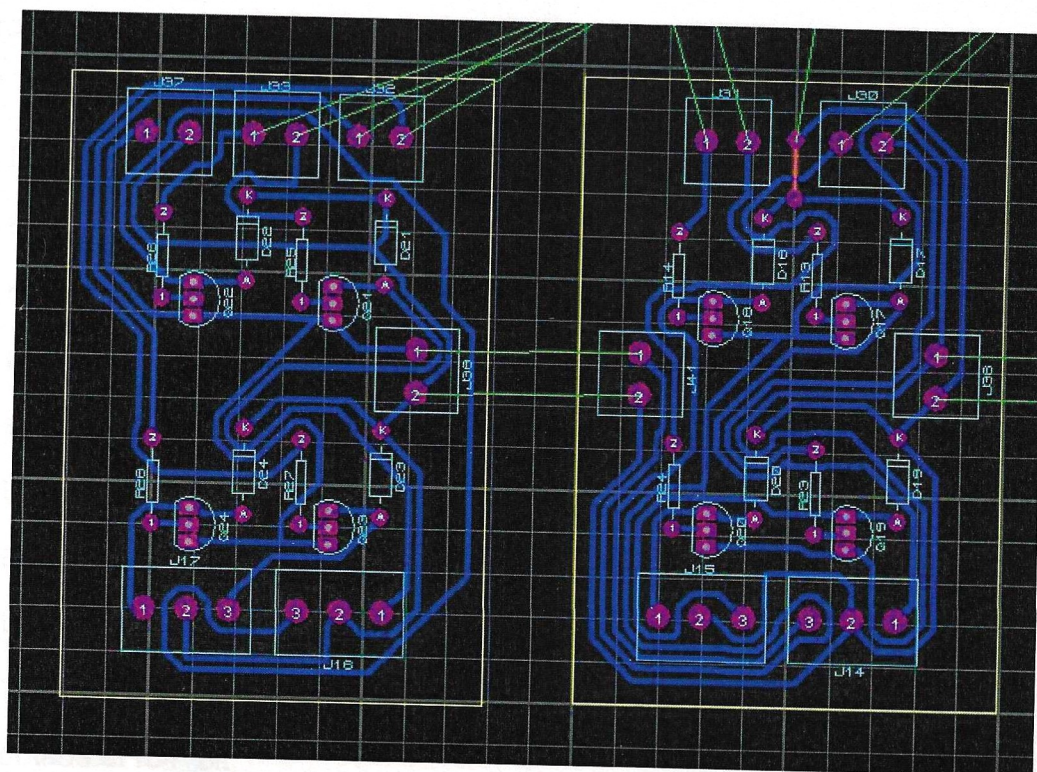
Design da placa do circuito transmissor.

- Imagens da criação do layout da placa do circuito receptor no programa ARES

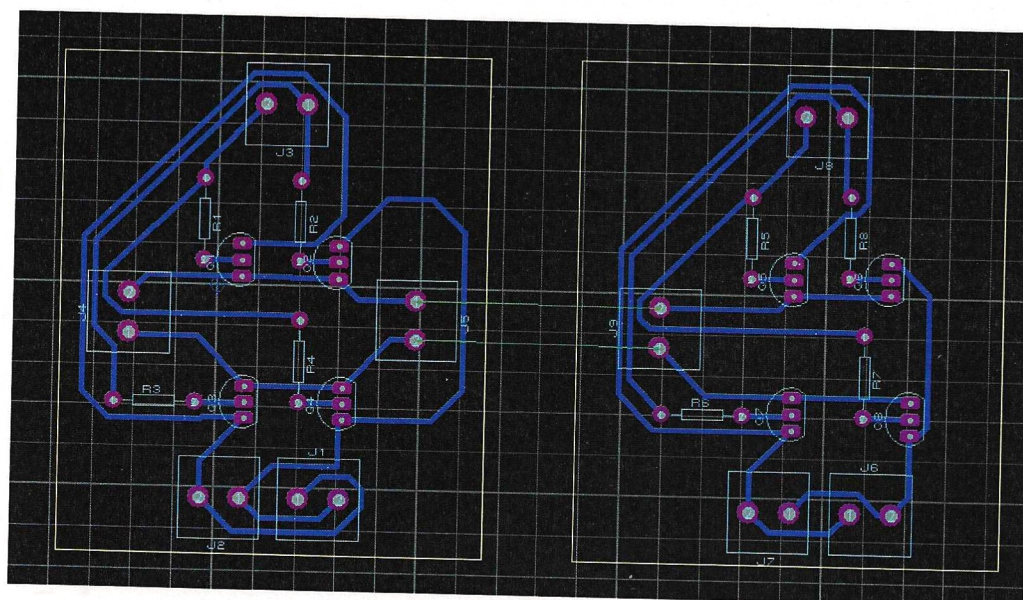


Design da placa principal do circuito receptor.





Design das placas dos drivers de motor de passo que se conectam à placa principal do receptor.



Design das placas de ponte H para motores DC, que também se conectam à placa principal do receptor.



- Programação do circuito transmissor em código C

```
sbit botao1 at rc2_bit;  
sbit botao2 at rc3_bit;  
sbit botao3 at rc0_bit;  
sbit botao4 at rc1_bit;  
sbit sinal_tx at rb7_bit;  
sbit botao5 at rd0_bit;  
sbit botao6 at rd1_bit;  
sbit botao7 at rd2_bit;  
sbit botao8 at rd3_bit;  
sbit botao9 at rd4_bit;  
sbit botao10 at rd5_bit;  
sbit botao11 at rd6_bit;  
sbit botao12 at rd7_bit;
```

```
void bit_1()  
{  
    sinal_tx = 1; delay_us(60);  
    sinal_tx = 0; delay_us(30);  
}
```

```
void bit_0()  
{  
    sinal_tx = 1; delay_us(30);  
    sinal_tx = 0; delay_us(60);  
}
```

```
}
```

```
void start_bit()
```

```
{
```

```
    sinal_tx = 1; delay_us(120);
```

```
    sinal_tx = 0; delay_us(60);
```

```
}
```

```
void end_bit()
```

```
{
```

```
    sinal_tx = 1; delay_us(90);
```

```
    sinal_tx = 0; delay_us(60);
```

```
}
```

```
void main()
```

```
{//1
```

```
    intcon=0x00;
```

```
    c1on_bit = 0; c2on_bit = 0;
```

```
    adon_bit = 0; anse1 = 0; anse1h = 0;
```

```
    trisb=0b00000000;
```

```
    trisc=0b00001111;
```

```
    trisd=0b11111111;
```

```
    sinal_tx = 0;
```

```
while(1)
```

```
{
```



```
//Esteira
```

```
if(botao1 == 1)
```

```
{
```

```
    bit_1();
```

```
    bit_0();
```

```
    bit_0();
```

```
    bit_0();
```

```
}
```

```
if(botao2 == 1)
```

```
{
```

```
    bit_1();
```

```
    bit_0();
```

```
    bit_0();
```

```
    bit_1();
```

```
}
```

```
if(botao3 == 1)
```

```
{
```

```
    bit_1();
```

```
    bit_0();
```

```
    bit_1();
```

```
    bit_0();
```

```
}
```

```
if(botao4 == 1)
```

```
{
```

```
    bit_1();  
    bit_0();  
    bit_1();  
    bit_1();  
}
```

```
//Braço hidráulico
```

```
if(botao5 == 1)
```

```
{  
    bit_1();  
    bit_1();  
    bit_0();  
    bit_0();  
}
```

```
if(botao6 == 1)
```

```
{  
    bit_1();  
    bit_1();  
    bit_0();  
    bit_1();  
}
```

```
if(botao7 == 1)
```

```
{  
    bit_1();  
    bit_1();  
}
```



```
    bit_1();  
    bit_0();  
}
```

```
if(botao8 == 1)  
{  
    bit_1();  
    bit_1();  
    bit_1();  
    bit_1();  
}
```

```
if(botao9 == 1)  
{  
    bit_0();  
    bit_0();  
    bit_0();  
    bit_0();  
}
```

```
if(botao10 == 1)  
{  
    bit_0();  
    bit_0();  
    bit_0();  
    bit_1();  
}
```

```

if(botao11 == 1)
{
    bit_0();
    bit_0();
    bit_1();
    bit_0();
}

```

```

if(botao12 == 1)
{
    bit_0();
    bit_0();
    bit_1();
    bit_1();
}

```

```

}
}

```

- Programação do circuito receptor em linguagem C

```

sbit sinal_tx at rb7_bit;
sbit led1 at ra1_bit;
sbit led2 at ra2_bit;
sbit led3 at ra3_bit;

```



```
sbit led4 at ra4_bit;  
sbit led5 at rb0_bit;  
sbit led6 at rb1_bit;  
sbit led7 at rb2_bit;  
sbit led8 at rb3_bit;
```

```
sbit led9 at rc0_bit;  
sbit led10 at rc1_bit;  
sbit led11 at rc2_bit;  
sbit led12 at rc3_bit;  
sbit led13 at rc4_bit;  
sbit led14 at rc5_bit;  
sbit led15 at rc6_bit;  
sbit led16 at rc7_bit;  
sbit led17 at rd0_bit;  
sbit led18 at rd1_bit;  
sbit led19 at rd2_bit;  
sbit led20 at rd3_bit;  
sbit led21 at rd4_bit;  
sbit led22 at rd5_bit;  
sbit led23 at rd6_bit;  
sbit led24 at rd7_bit;
```

```
char bit_1;  
char bit_0;  
char start_bit;  
char end_bit;
```

```
char cont3;  
char cont4;  
char cont5;  
char cont6;
```

```
void verificacao_bit()
```

```
{  
    while(sinal_tx == 0);  
    delay_us(15);  
    if(sinal_tx == 1)  
    {  
        delay_us(30);  
        if(sinal_tx == 0)  
        {  
            bit_0 = 1;  
            bit_1 = 0;  
        }  
        if(sinal_tx == 1)  
        {  
            delay_us(30);  
            if(sinal_tx == 0)  
            {  
                bit_1 = 1;  
                bit_0 = 0;  
            }  
        }  
    }  
}
```



```
}
```

```
void main()
```

```
{ //1
```

```
    intcon=0x00;
```

```
    c1on_bit = 0; c2on_bit = 0;
```

```
    adon_bit = 0; ansel = 0; anselh = 0;
```

```
    trisa=0b00000000;
```

```
    trisb=0b10000000;
```

```
    trisc=0b00000000;
```

```
    trisd=0b00000000;
```

```
    led1 = 0;
```

```
    led2 = 0;
```

```
    led3 = 0;
```

```
    led4 = 0;
```

```
    led5 = 0;
```

```
    led6 = 0;
```

```
    led7 = 0;
```

```
    led8 = 0;
```

```
    led9 = 0;
```

```
    led10 = 0;
```

```
    led11 = 0;
```

```
    led12 = 0;
```

```
    led13 = 0;
```

```
    led14 = 0;
```

```
    led15 = 0;
```

```
led16 = 0;
```

```
led17 = 0;
```

```
led18 = 0;
```

```
led19 = 0;
```

```
led20 = 0;
```

```
led21 = 0;
```

```
led22 = 0;
```

```
led23 = 0;
```

```
led24 = 0;
```

```
cont3 = 1;
```

```
cont4 = 1;
```

```
cont5 = 1;
```

```
cont6 = 1;
```

```
while(1)
```

```
{ //2
```

```
    led1 = 0;
```

```
    led2 = 0;
```

```
    led3 = 0;
```

```
    led4 = 0;
```

```
    led5 = 0;
```

```
    led6 = 0;
```

```
    led7 = 0;
```

```
    led8 = 0;
```

```
    led9 = 0;
```



```
led10 = 0;
led11 = 0;
led12 = 0;
led13 = 0;
led14 = 0;
led15 = 0;
led16 = 0;
led17 = 0;
led18 = 0;
led19 = 0;
led20 = 0;
led21 = 0;
led22 = 0;
led23 = 0;
led24 = 0;
```

```
verificacao_bit();
if(bit_1 == 1)
{
    verificacao_bit();
    if(bit_0 == 1)
    {
        verificacao_bit();
        if(bit_0 == 1)
        {
            verificacao_bit();
```

```
if(bit_1 == 1)
{
    //codigo1
    led13 = 1;
    led14 = 0;
    led15 = 1;
    led16 = 0;
}
else
{
    //codigo2
    if(bit_0 == 1)
    {
        led14 = 1;
        led13 = 0;
        led16 = 1;
        led15 = 0;
    }
}
}
else
{
    if(bit_1 == 1)
    {
        verificacao_bit();
        if(bit_0 == 1)
        {
```



```
//codigo3
led14 = 1;
led13 = 0;
led15 = 1;
led16 = 0;
}
else
{
    if(bit_1 == 1)
    {
        //codigo4
        led13 = 1;
        led14 = 0;
        led16 = 1;
        led15 = 0;
    }
}
}
}
else
{
    if(bit_1 == 1)
    {
        verificacao_bit();
        if(bit_0 == 1)
        {
```

```
verificacao_bit();  
if(bit_0 == 1)  
{  
    //codigo5  
    cont3 = cont3*2;  
}  
else  
{  
    if(bit_1 == 1)  
    {  
        //codigo6  
        cont3 = cont3/2;  
    }  
}  
}  
else  
{  
    if(bit_1 == 1)  
    {  
        verificacao_bit();  
        if(bit_0 == 1)  
        {  
            //codigo7  
            cont4 = cont4*2;  
        }  
    }  
    else  
    {
```



```
        if(bit_1 == 1)
        {
            //codigo8
            cont4 = cont4/2;
        }
    }
}
}
}
}
}
}
}
else
{
    if(bit_0 = 1)
    {
        verificacao_bit();
        if(bit_0 = 1)
        {
            verificacao_bit();
            if(bit_0 = 1)
            {
                verificacao_bit();
                if(bit_0 = 1)
                {
                    //codigo9
                    cont5 = cont5*2;
                }
            }
        }
    }
}
```

```
else
{
    if(bit_1 = 1)
    {
        //codigo10
        cont5 = cont5/2;
    }
}
}
else
{
    if(bit_1 = 1)
    {
        //codigo11
        verificacao_bit();
        if(bit_0 = 1)
        {
            //codigo11
            cont6 = cont6*2;
        }
    }
    else
    {
        if(bit_1 = 1)
        {
            //codigo12
            cont6 = cont6/2;
        }
    }
}
```



```
    }  
  }  
}  
}  
}
```

```
if(cont3 > 8)  
{  
    cont3 = 1;  
}
```

```
if(cont4 > 8)  
{  
    cont4 = 1;  
}
```

```
if(cont3 < 1)  
{  
    cont3 = 8;  
}
```

```
if(cont4 < 1)  
{  
    cont4 = 8;  
}
```

```
if(cont5 > 8)
```

```
{
```

```
    cont5 = 1;
```

```
}
```

```
if(cont6 > 8)
```

```
{
```

```
    cont6 = 1;
```

```
}
```

```
if(cont5 < 1)
```

```
{
```

```
    cont5 = 8;
```

```
}
```

```
if(cont6 < 1)
```

```
{
```

```
    cont6 = 8;
```

```
}
```

```
led5=cont3.B0;
```

```
led6=cont3.B1;
```

```
led7=cont3.B2;
```

```
led8=cont3.B3;
```

```
led1=cont4.B0;
```

```
led2=cont4.B1;
```



```
led3=cont4.B2;
```

```
led4=cont4.B3;
```

```
led17=cont5.B0;
```

```
led18=cont5.B1;
```

```
led19=cont5.B2;
```

```
led20=cont5.B3;
```

```
led21=cont6.B0;
```

```
led22=cont6.B1;
```

```
led23=cont6.B2;
```

```
led24=cont6.B3;
```

```
delay_ms(300);
```

```
start_bit = 0;
```

```
end_bit = 0;
```

```
}
```

```
}
```

