APLICAÇÃO DE ARDUINO NO EXÉRCITO BRASILEIRO: EMPREGO DO ARDUINO PARA OPERACIONALIDADE DAS COMUNICAÇÕES

GEOVANI DE MORAES TOMAZ, GLEISON LIMA DE SOUZA, JOSE DALVAN DE MOURA SANTOS, MATHEUS CAON STOCHERO, PEDRO VILARIN DE LIMA , MARCOS VINICIUS HOLANDA MARTINS, MATHEUS TAVARES RODRIGUES GUARANY DA SILVA, RAFAEL PORTELLA DA SILVA, RAPHAEL DA SILVA GUIMARÃES E WELLINGTON DE SOUSA SILVA

RESUMO: Desde o início da utilização da radiofrequência no Brasil, a comunicação via rádio continua sendo um dos principais meios de comunicações empregados pela Força Terrestre em campanha. Neste artigo foi apresentado uma atividade pioneira desenvolvida por militares do Curso Avançado de Eletrônica do ano de 2022: a conversão de informação em claro em código morse, utilizando como base o microcontrolador Arduino e o Rádio Harris MPR-9600-MP.

Palavras Chaves: ARDUINO, CÓDIGO MORSE, COMUNICAÇÕES, RADIOFREQUÊNCIA, RÁDIO HARRIS.

1 INTRODUÇÃO

A comunicação no Exército Brasileiro é de suma importância para o seu emprego em diversos terrenos diante da extensão territorial do país. Sendo assim, em algumas ocasiões, a comunicação através do rádio torna-se muito complexa, tendo em vista a dificuldade de acesso ao teatro operacional.

Neste cenário, a fim de se ter uma comunicação com o melhor alcance e com um custo mínimo de bateria, vislumbrou-se a possibilidade de se utilizar o equipamento Rádio Harris MPR 9600 na modulação Continuos Wave (CW).

Sabendo-se que a linguagem em Código Morse é complexa, de difícil aprendizado e que existem poucos militares especializados atuando nesta área de comunicação no Exército Brasileiro e com base no tema proposto no artigo científico, surgiu a possibilidade de se utilizar o Microcontrolador Arduíno acoplado ao equipamento rádio, a fim de permitir a troca de mensagens em Código Morse sem que o operador detenha qualquer conhecimento de telegrafia. Para tanto, foi acoplado ao Microcontrolador Arduíno um display LCD 20x4 e um mini teclado matricial 4x11 que permitem ao operador ler e escrever mensagens que serão enviadas em Código Morse.

Com isso, permitiu-se que o Arduíno automatizasse o processo de codificação e decodificação das mensagens recebidas e enviadas.

2. METODOLOGIA

Para o embasamento teórico foram realizadas diversas pesquisas sobre os temas: estrutura do código morse, modulação CW, transmissão morse via Arduino e o Manual de operação do equipamento rádio Harris MPR-9600MP.

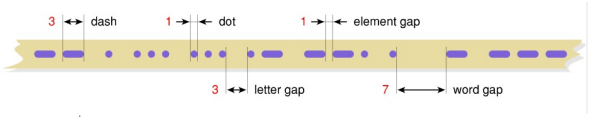
Foram utilizadas diversas bibliografias encontradas sobre o Código Morse e Programação C++ na internet e algumas foram adaptadas ao código do Arduino para a resolução da situação problema. Posteriormente, foram realizados os cálculos do filtro RC, utilizado no circuito de detecção de tom e filtragem das harmônicas.

2.1 INTEGRAÇÃO ARDUINO-RÁDIO

Para desenvolver uma solução de comunicação emergencial que possibilite a transmissão em CW, utilizando o microcontrolador Arduino e o Equipamento Rádio Harris MPR-9600MP, é necessário saber a lógica da comunicação via código morse e

como funciona a modulação CW.
Embora o código morse possa ser transmitido em qualquer velocidade, o tempo relativo entre os vários elementos é fixo. De um modo geral, o código morse consiste em cinco elementos:
a) um ponto (uma unidade);
b) um traço (três unidades);
c) uma lacuna entre elementos (uma unidade);
d) uma lacuna entre letras (três unidades);
e) lacuna entre palavras (7 unidades).

FIGURA 01 - Elementos do Código Morse



Fonte: Crypto Museum, 2016.

No código binário a seguir, o texto “VIR FORTIS” foi usado como exemplo para demonstrar que também é possível representar o código morse por um fluxo constante de bits digitais (uns e zeros). Se definirmos um tom como '1' e um silêncio como '0', o exemplo produziria o seguinte fluxo, conforme TABELA 01 a seguir:

TABELA 01 - Representação da Mensagem Código Morse

Texto	“VIR FORTIS”
Ponto (Dit) e Traço (Dah)	...- .. .-. / ..-. --- .-. - .. ...
Bits	0101011001001001010100100010000001000110010011110101001001010100010010010101010011

Fonte: os autores, 2022.

2.1.1 NA TRANSMISSÃO

Para que seja possível transmitir em “Continuous Wave” (CW) o equipamento rádio deve estar na modulação CW e o botão “Push to Talk” (PTT) deve ser apertado para produzir os pulsos de diferentes comprimentos (pontos e traços) que formarão as mensagens de texto em código Morse.

Para automatizar esse processo de codificação foi confeccionando um cabo para integrar o microcontrolador com o equipamento rádio e fazer com que o Arduino defina corretamente os momentos de fechar os contatos entre os pinos A (terra) e C (PTT) do cabo que vem do rádio.

2.1.2 NA RECEPÇÃO

Já na recepção, para que o Arduino decodifique o sinal vindo pelos pinos A (terra) e B (fone) do cabo que vem do equipamento rádio, é necessário que o sistema consiga interpretar quando o sinal está em nível lógico alto ou nível lógico baixo e a partir daí registrar o tempo que leva em cada estado, para dessa forma conseguir decodificar a mensagem e mostrá-la num display.

2.1.3 DISPOSITIVO DE ENTRADA

Como dispositivo de entrada foi utilizado um teclado adaptado e ligado ao Arduino, por meio de cabos flexíveis, que serve para a devida inserção das mensagens que serão codificadas para linguagem em Código Morse pelo microcontrolador.

2.1.4 DISPOSITIVO DE SAÍDA

Como dispositivo de saída foi utilizado um Display LCD 20x4 ligado ao Arduino, via I2C, que serviu para exibir as mensagens recebidas e transmitidas que foram codificadas e decodificadas para linguagem em Código Morse pelo microcontrolador.

3. RESULTADOS E DISCUSSÃO

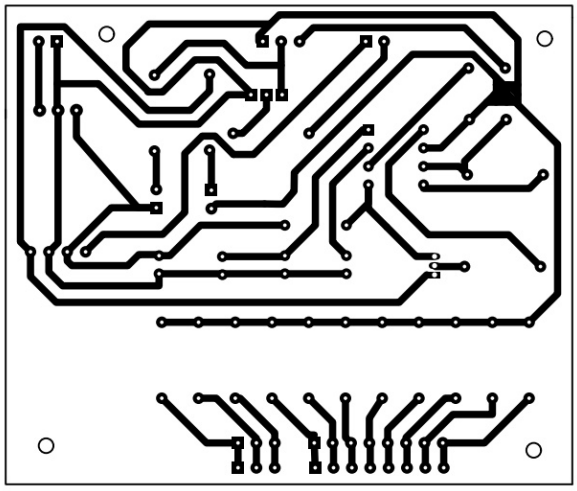
3.1 INTEGRAÇÃO ARDUINO-RÁDIO

A utilização do Arduino como protótipo, que posteriormente se tornou uma placa com os circuitos testados em funcionamento, permitiram que a automaticidade da interpretação da informação na rede rádio ocorresse com sucesso após testes de caixa preta, isto é, teste de funcionalidade, que buscaram garantir que os requisitos funcionais do produto estivessem consistentes.

O projeto da placa para o circuito microcontrolador foi construída com vários componentes, citados na Tabela no Apendice A – Tabela de Custos, deste documento. As

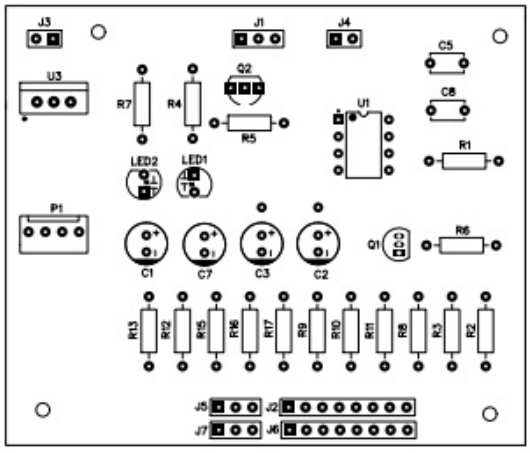
figuras abaixo mostram a Placa PCB projetada para este trabalho com base nos requisitos levantados.

FIGURA 02 – Placa de Circuito Impresso Digital



Fonte: os autores, 2022.

FIGURA 03 - Placa de Circuito Impresso com Componentes

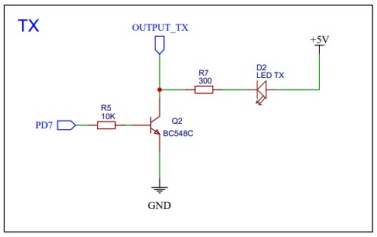


Fonte: os autores, 2022.

O programa usado para projetar o circuito microntrolador idealizado foi o EasyEDA, uma plataforma gratuita, totalmente online e baseado em nuvem.

3.1.1 NA TRANSMISSÃO

FIGURA 04 – Esquema da Transmissão

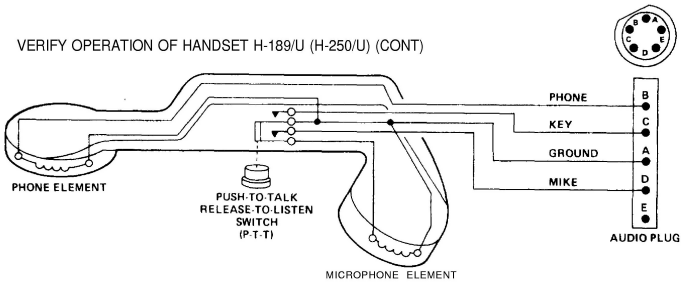


Fonte: os autores, 2022.

Na transmissão temos um circuito alimentado com uma tensão de +5VCC.

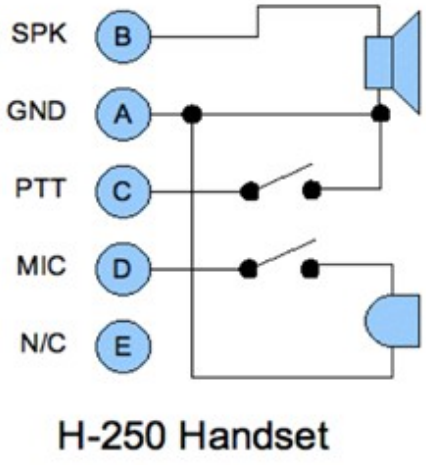
Através de uma programação C++ realizada no Arduino, cada vez que o Push-to-Talk (PTT) é acionado, a corrente passa pelo transistor NPN que funciona como uma chave fechada aterrando o sinal e acendendo o LED. Nessa fase, tendo em vista o Transceptor estar na modulação CW, a portadora é enviada para a fase de recepção do outro equipamento rádio, com que está se realizando o enlace. Vide figura 05 e figura 06.

FIGURA 05 – . Esquema do monofone



Fonte: Radionerds.com, 2014.

FIGURA 06 – Esquema da pinagem do Monofone



Fonte: Radionerds.com, 2014.

3.1.2 NA RECEPÇÃO

O sistema ligado fica na escuta do pino de entrada verificando se há presença de algum sinal elétrico através do conector de áudio do transceptor.

Quando a mensagem em Código Morse chega, o status do pino de entrada muda de

baixo para alto, então entende-se que existe tensão no pino de entrada do Arduino. Quando isso ocorre, o sistema começa a contagem do período em que esse sinal se mantém, determinando dessa forma a duração do pulso recebido.

O ponto (Dit) é identificado pelo sistema quando o pulso dura entre 2/3 de um Dit e 4/3 de um Dit. Se o valor de duração de um Dit for entre 2 vezes o valor de um Dit e 4 vezes o valor de um Dit, o sistema fará a interpretação do pulso como um traço (Dah).

O sistema interpretará a pausa com um espaço quando o pulso elétrico no pino de entrada for interrompido por tempo equivalente a 3 vezes o Dit e esse silêncio não tiver sido associado antes a um espaço.

Dessa forma, se a duração da pausa for superior ao tempo de um ponto e o sistema já tiver reconhecido algum pulso, o sistema compreenderá que o código lido equivale a um caractere, mapeando essa sequência de Dit e Dah.

À medida que a interpretacao de cada pulso elétrico vai ocorrendo e sendo convertida em caracteres, ele vai concatenando, chegando ao final com o texto decodificado que é enviado para o display, a fim de ser visualizado pelo usuário.

Um problema encontrado foi separar o tom de 1000Hz da frequência da portadora. Como se sabe o equipamento rádio HARRIS MPR-9600 MP opera em CW com emissão J2A e possui a configuração CW OFFSET, na qual se tem duas opções, portadora suprimida (CW) ou tom modulado (MCW). Um extrato com as especificações do Manual do Rádio MPR-9600 sobre **CW OFFSET** estão no Apêndice D deste artigo.

Para o desenvolvimento deste projeto a opção escolhida foi **MCW**, na qual, ao fechar os contatos do PTT no rádio transmissor, o rádio receptor em **MCW** obtêm um tom de 1kHz. Caso o rádio receptor esteja na opção **CW**, no lugar do tom de 1kHz ele tem sua portadora suprimida, ou seja, fica um “vazio”, dessa forma seria mais difícil desenvolver o projeto.

TABELA 02 – Funcionamento Configuração do CW OFFSET

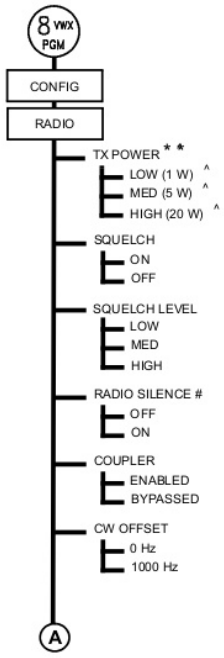
ORD	RÁDIO TX	RÁDIO RX	TOM NA RX
01	CW*	CW	1 KHZ
02	MCW**	MCW	1 KHZ
03	MCW	CW	0 HZ
04	CW	MCW	3,3 KHZ

Fonte: os autores, 2022.

Na Tabela 02 só funciona 01 e 02, pois a configuração é para detectar o sinal de 1 kHz.

Para acessar as configurações de **CW OFFSET** deve ser pressionado a sequência **PGM>CONFIG>RADIO** no menu.

FIGURA 07 – Demonstração Menus Config Harris



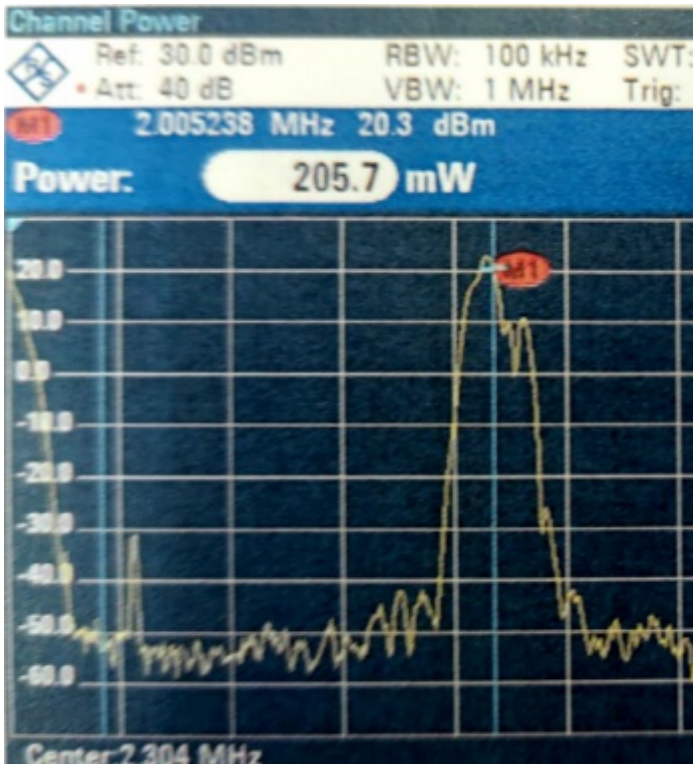
Fonte: os autores, 2022.

3.2 MAIOR ALCANCE E ECONOMIA DE BATERIA DO EQUIPAMENTO RÁDIO

Após a realização de testes em dois Equipamentos Rádios Harris HF MPR-9600-MP transmitindo por meio da modulação em AME e da modulação em Continuos Wave (CW), verificou-se o seguinte resultado:

Na transmissão (TX) da mensagem, com o rádio configurado na modulação AME, a portadora é enviada com as bandas laterais superior e inferior, na mesma faixa de frequência, conforme se demonstra na onda do analisador de espectro da Figura 08.

FIGURA 08 – Análise da Onda na Tx em AME

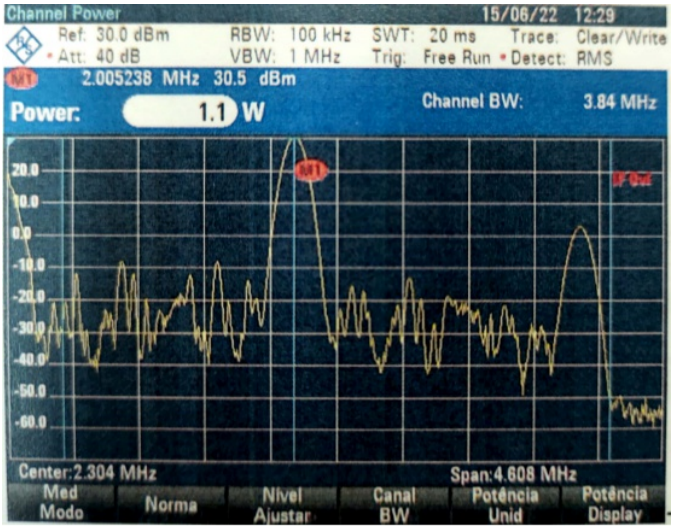


Fonte: os autores, 2022.

Torna-se nítido que a potência neste tipo de transmissão é menos aproveitada, tendo em vista que a portadora carrega todos os dados para o outro equipamento rádio, Sendo assim, exigi-se mais do transceptor quando do envio das mensagens, e consequentemente o consumo de energia armazenada no rádio torna-se elevado, e o alcance da mesma é menor.

Agora, quando a transmissão (TX) é feita com o rádio configurado na modulação Continuos Wave (CW), na mesma faixa de frequência, ocorre o envio da portadora vazia, isto é, sem as bandas laterais superior e inferior, conforme se demonstra na onda do analisador de espectro, Figura 09.

FIGURA 09 - Análise da Onda na TX em Continuous Wave (CW)



Fonte: os autores,2022.

Com isso, temos a mesma potência configurada, porém, mais bem aproveitada, fazendo com que o alcance do rádio, neste tipo de modulação, seja superior, uma vez que ele carrega somente a portadora.

4. CONCLUSÃO

O presente trabalho estudou a aplicação de microcontroladores para solucionar uma problemática real dentro do Exército Brasileiro. Podemos demonstrar que o Arduíno é capaz de contornar uma situação complexa, onde não havia estudos a respeito.

Nos esforços de nosso trabalho neste curto espaço de tempo, podemos perceber que o circuito deve ser bem montado e estável, porque as ondas de rádio, principalemtne HF, sofrem diferentes tipos de interferência ao longo do dia, dificultando a montagem do circuito ideal para o propósito.

O presente trabalho pretende ser um alicerce para futuros estudos na área de integração entre microcontroladores e os equipamentos rádio utilizados pela força, pois as experiencias limitaram-se à utilização do rádio MPR-9600-MP, desconhecendo-se como será a interação do microcontrolador sendo implementado com outros modelos de equipamentos rádios.

5. REFERÊNCIAS

AXTUDO. Recursos decodificador de dados e aplicativos do decodificador de tom ic lm567. Disponível em: <https://www.axtudo.com/recursos-decodificador-de-dados-e-aplicativos-do-decodificador-de-tom-ic-lm567/> Acesso em: 19 de maio de 2022.

BANZI, M., CUARTIELLES, D., IGOE, T., MARTINO, G., MELLIS, D. Página Oficial do Arduino, 2016. Disponível em: <www.arduino.cc>. Acesso em: 24 de maio de 2022.

BRAGA, Newton C. Fevereiro, 2013. CW (Continuous Wave ou Onda Contínua). Disponível em: <https://www.newtoncbraga.com.br/index.php/almanaque/977-cw-continuous-wave-ou-onda-continua.html> Acesso em: 20 de maio de 2022.

CIRIBELLI, Marilda Corrêa. Como elaborar uma dissertação de Mestrado através da pesquisa científica. Marilda Ciribelli Corrêa, Rio de Janeiro: 7 Letras, 2003.

HALÁSZ, Iwan Thomas. Hamlbook do Radioamador. São Paulo: Editora da Universidade de São Paulo, 1993

Harris Corporation. MPR-9600 ADVANCED TACTICAL HF RADIO – Intermediate Maintenance Manual. 2009.

Harris Corporation. MPR-9600 ADVANCED TACTICAL HF RADIO – Operation Manual. 2007.

LAURINDO, F.J.B., ROTONDARO, R.G. Unindo tecnologia da informação e gestão por processos: introdução e objetivos. In: Gestão Integrada de processos e da tecnologia da informação. São Paulo: Atlas, 2006.

Manual C11-1. Manual de Emprego das Comunicações do Exército Brasileiro, 2ª Edição/1997.

Manual C24-6, Exploração em Radiotelegrafia e Telegrafia, 2ª Edição/1979.

MAXIM INTEGRATED. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Califórnia, Estados Unidos, 2015. Disponível em: <http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Acesso em: 11 de maio de 2022.

RODRIGUES, William Costa. Metodologia Científica, 2007. Disponível em:<http://unisc.br/portal/upload/com_arquivo/metodologia_cientifica.pdf>. Acesso em: 10 de maio de 2022.

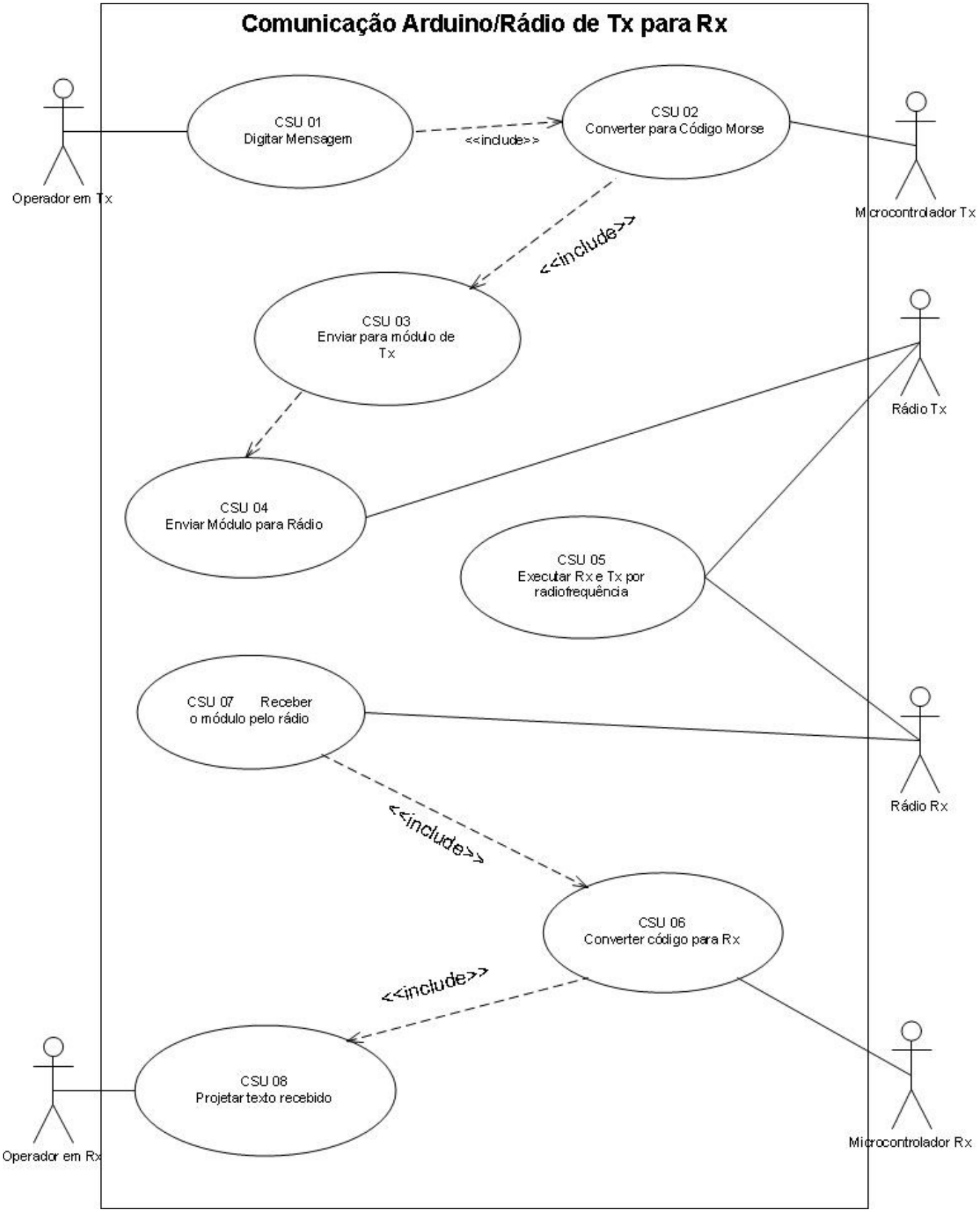
SARMENTO. O que é modulação e que modos são utilizados. Disponível em: <http://www.sarment-o.eng.br/Modulacao.htm>. Acesso em: 14 junho 2022.

STRINGFIXER. Onda Contínua. Disponível em: <https://stringfixer.com/pt/Continuous_wave> Acesso em: 13 de junho de 2022.

APÊNDICE A – TABELA DE CUSTOS

ORD	DESCRIÇÃO	QUANT.	VALOR UN.	VALOR TOTAL
1	Display Cristal Líquido LCD 20x4 – AZ/BR	1	R$ 45,90	R$ 45,90
2	Módulo Serial I2C para Display	1	R$ 8,90	R$ 8,90
3	Mini Teclado Wireless	1	R$ 35,00	R$ 35,00
4	Capacitor Eletrolítico 2,2µF x 50V	1	R$ 0,09	R$ 0,09
5	Capacitor Eletrolítico 100µF x 50V	1	R$ 0,79	R$ 0,79
6	Capacitor Eletrolítico 10µF x 50V	1	R$ 0,12	R$ 0,12
7	Capacitor Eletrolítico 1µF x 50V	2	R$ 0,14	R$ 0,28
8	Capacitor Poliester 100nF	2	R$ 0,59	R$ 1,18
9	Resistor 1kΩ - 1/4W	1	R$ 0,05	R$ 0,05
10	Resistor 300Ω - 1/4W	2	R$ 0,05	R$ 0,10
11	Resistor 10kΩ - 1/4W	1	R$ 0,05	R$ 0,05
12	Resistor 220Ω - 1/4W	11	R$ 0,05	R$ 0,55
13	Transistor PNP - BC558	1	R$ 0,18	R$ 0,18
14	Transistor NPN - BC548	1	R$ 0,16	R$ 0,16
15	LM7805	1	R$ 2,13	R$ 2,13
16	LM567	1	R$ 2,49	R$ 2,49
17	Arduino MEGA 2560	1	R$ 137,50	R$ 137,50
18	Sensor Laser Apontador	1	R$ 3,30	R$ 3,30
19	Fotocélula LDR 5mm	1	R$ 0,90	R$ 0,90
20	LED	2	R$ 0,14	R$ 0,28
21	Chave Gangorra mini 2T	1	R$ 0,69	R$ 0,69
22	Soquete 8 pinos	1	R$ 0,29	R$ 0,29
23	Conector Barra de Pinos	1	R$ 1,21	R$ 1,21
24	Conector MOLEX KK 2T	3	R$ 0,46	R$ 1,38
25	Conector MOLEX KK 3T	1	R$ 0,82	R$ 0,82
26	Conector Circular MIKE 5P	1	R$ 7,79	R$ 7,79
27	Conector USB Fêmea Tipo B	1	R$ 1,69	R$ 1,69
28	Conector Barra de Pinos Fêmea 1x8x11,2 180 graus	2	R$ 0,89	R$ 1,78
29	Conector Barra de Pinos Fêmea 1x3x11,2 180 graus	2	R$ 0,49	R$ 0,98
30	Placa Fenolite 10x20	1	R$ 7,95	R$ 7,95
31	Caixa Plástica Para Circuito	2	R$ 5,55	R$ 11,10
	TOTAL			R$ 275,63

APÊNDICE B – FIGURA DE DIAGRAMA CASOS DE USO



APÊNDICE C – FIGURA DA PARTE STRUCT CW DO CÓDIGO

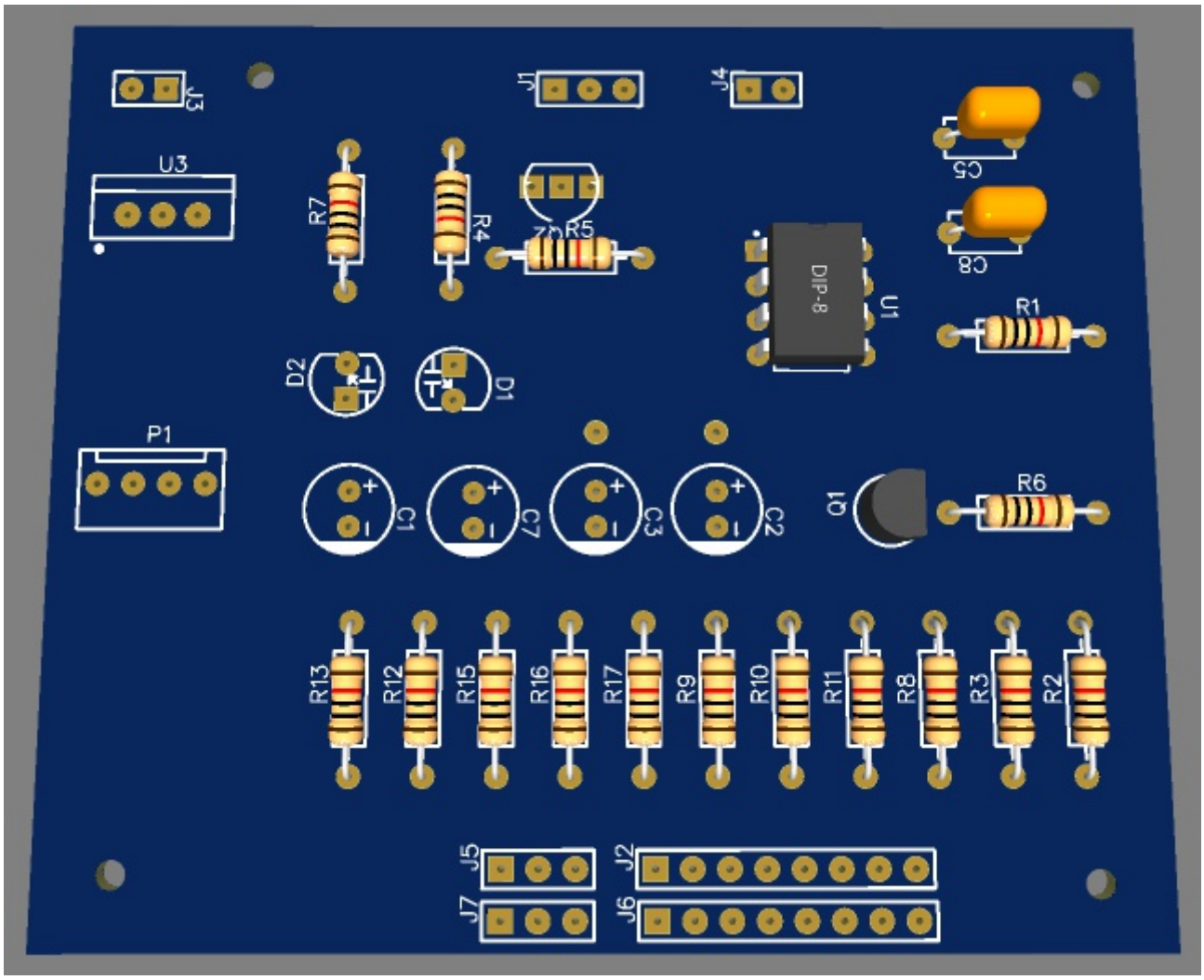
```
struct biblioteca_morse // Cria uma biblioteca para guardar os códigos dos caracte
{
  char character;
  String codigo;
  int linha;
  int coluna;
};

biblioteca_morse matriz[n_caracteres] = {
  {'A', ".-", 3 , 1},    {'B', "-...", 4, 5},   {'C', "-.-.", 4, 3},   {'D', "-.."
  {'E', ".", 2 , 3},     {'F', "..-.", 3, 4},   {'G', "--.", 3, 5},    {'H', "....
  {'I', "..", 2, 8},     {'J', ".---", 3, 7},   {'K', "-.-", 3, 8},    {'L', ".-..
  {'M', "--", 4, 7},     {'N', "-.", 4, 6},     {'O', "---", 2, 9},    {'P', ".--.
  {'Q', "--.-", 2, 1},   {'R', ".-.", 2, 4},    {'S', "...", 3, 2},    {'T', "-",
  {'U', "..-", 2, 7},    {'V', "...-", 4, 4},   {'W', ".--", 2, 2},    {'X', "-..-
  {'Y', "-.--", 2, 6},   {'Z', "--..", 4, 1},   {'0', "-----", 1, 10}, {'1', ".---
  {'2', "..---", 1, 2},  {'3', "...--", 1, 3},  {'4', "....-", 1, 4},  {'5', "....
  {'6', "-....", 1, 6},  {'7', "--...", 1, 7},  {'8', "---..", 1, 8},  {'9', "----
  {'.', ".-.-.-", 4, 9}, {',', "--..--", 4, 8}, {'?', "..--..", 4, 11},{'!', "-.-.
  {' ', "     ", 5, 6},  {';', "-.-.-.", 4, 10},
};
```

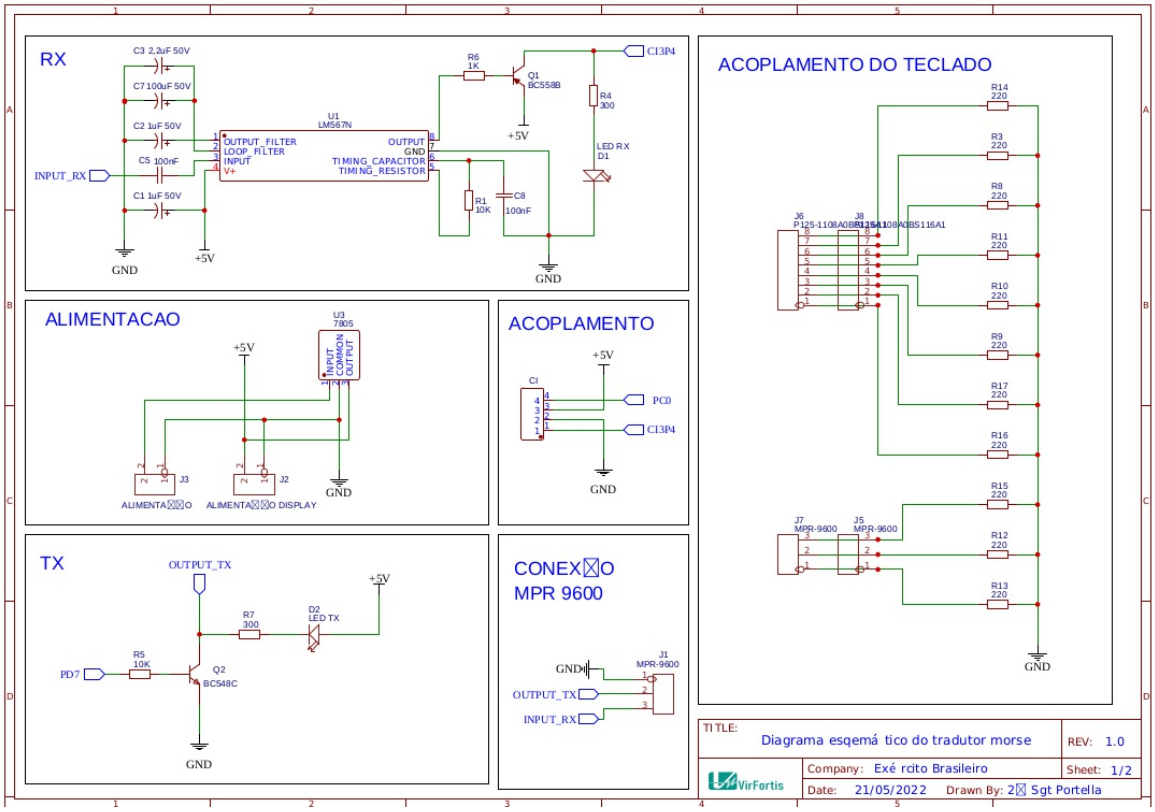
APÊNDICE D – ESPECIFICAÇÕES MANUAL HARRIS CW OFFSET

CW OFFSET	0 Hz 1000 Hz	Frequência central do BFO (clarificador) **0 Hz** – Padrão para CW (Continuous Wave) **1000 Hz** – MCW (Modulated Continuous Wave) com tom de 1000 Hz.

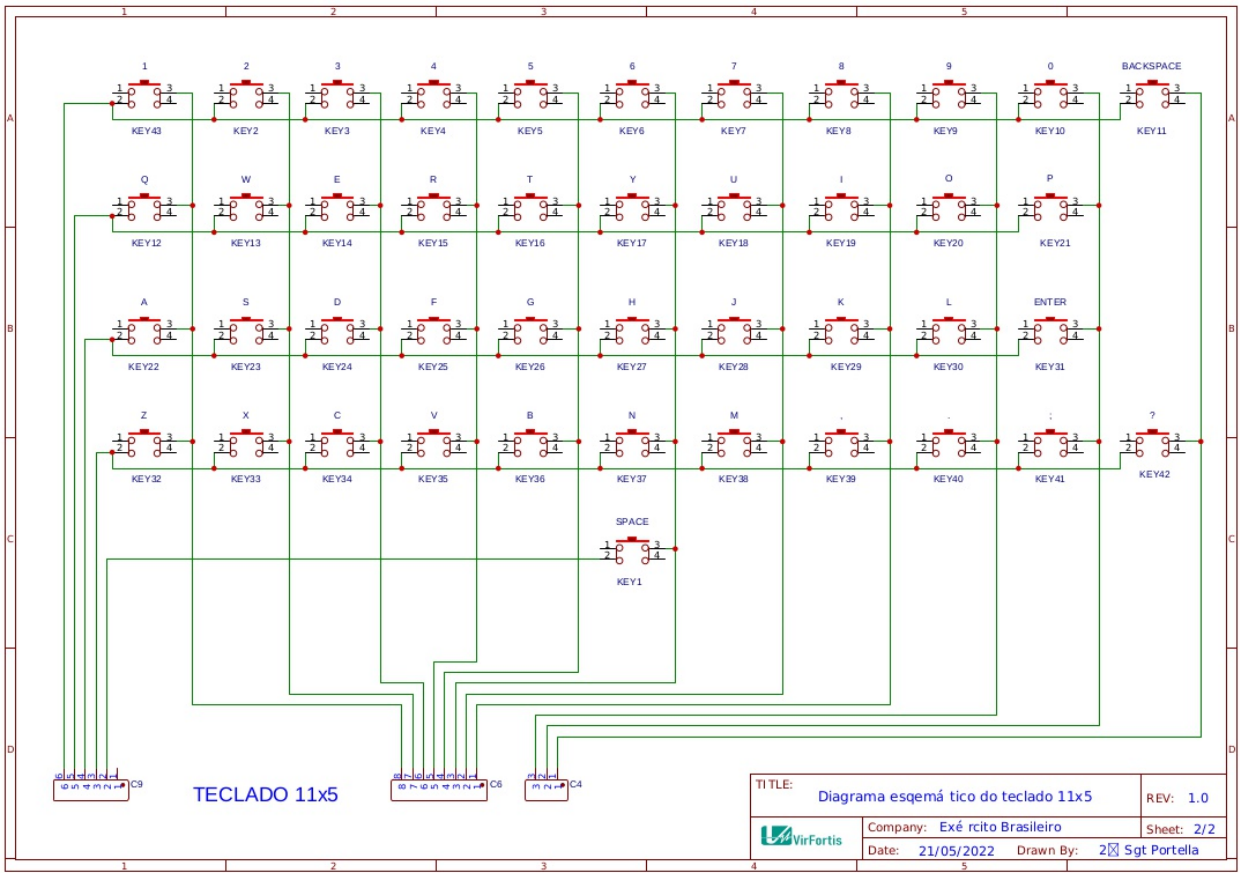
APÊNDICE E – PLACA DE CIRCUITO IMPRESSO COLORIDA



APÊNDICE F – ESQUEMÁTICO DO CIRCUITO DE TRANSMISSÃO/RECEPÇÃO



APÊNDICE G - ESQUEMÁTICO DO MINI TECLADO WIRELESS



APÊNDICE H - CÓDIGO DA PROGRAMAÇÃO NO ARDUÍNO

```
/*****************************************************************************
* IMPORTES, CONSTANTES E VARIAVEIS TECLADO/DISPLAY
*****************************************************************************/
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,20,4);

#define n_caracteres 41

int x = 0;
int y = 0;

int lcd_col = 0;
int lcd_lin = 0;

String mensagem = "";

String mensagens[100]; // salva as mensagens inteiras (permanente)
String linhas[100]; // salva a mensagem dividida em linhas (temporario)
int posUltMsg = 0; // posição da ultima mensagem no arrya de mensagens
int posMsgTela = 0; // posicao da pensagem no array que esta sendo exibida na tela
int poslinha = 0;
int posLinhaExibida = 3;

String fimMsg = " FIM DA MENSAGEM !!!";


int imprimiu = 0;

/*****************************************************************************
* DEFINIÇÕES DE CONSTANTES
*****************************************************************************/
#define pino_saida 7 // Define o pino de saida do MSG em código Morse

#define n_caracteres 41      //Quantidade de caracteres codificados

/*****************************************************************************
* DEFINIÇÕES DE VARIAVEIS
*****************************************************************************/
const byte    pino_entrada = A0;

const int     tempo_do_ponto = 150;  // Define o tempo do ponto

const byte      pinoPWM       = 3;      // Pino com controle PWM
const byte      pinoONOFF     = 4;      // Pino com controle liga/desliga

struct biblioteca_morse // Cria uma biblioteca para guardar os códigos dos caracteres em código
Morse
{
```

```
  char character;
  String codigo;
  int linha;
  int coluna;
};

biblioteca_morse matriz[n_caracteres] = {
  {'A', ".-", 3 , 1},    {'B', "-...", 4, 5},   {'C', "-.-.", 4, 3},   {'D', "-..", 3, 3},
  {'E', ".", 2 , 3},     {'F', "..-.", 3, 4},   {'G', "--.", 3, 5},    {'H', "....", 3, 6},
  {'I', "..", 2, 8},     {'J', ".---", 3, 7},   {'K', "-.-", 3, 8},    {'L', ".-..", 3, 9},
  {'M', "--", 4, 7},     {'N', "-.", 4, 6},     {'O', "---", 2, 9},    {'P', ".--.", 2, 10},
  {'Q', "--.-", 2, 1},   {'R', ".-.", 2, 4},    {'S', "...", 3, 2},    {'T', "-", 2, 5},
  {'U', "..-", 2, 7},    {'V', "...-", 4, 4},   {'W', ".--", 2, 2},    {'X', "-..-", 4, 2},
  {'Y', "-.--", 2, 6},   {'Z', "--..", 4, 1},   {'0', "-----", 1, 10}, {'1', ".----", 1, 1},
  {'2', "..---", 1, 2},  {'3', "...--", 1, 3},  {'4', "....-", 1, 4},  {'5', ".....", 1 , 5},
  {'6', "-....", 1, 6},  {'7', "--...", 1, 7},  {'8', "---..", 1, 8},  {'9', "----.", 1 , 9},
  {'.', ".-.-.-", 4, 9}, {',', "--..--", 4, 8}, {'?', "..--..", 4, 11},{'!', "-.-.--", 5, 1},
  {' ', "     ", 5, 6},
};

/*****************************************************************************
* SETUP
*****************************************************************************/
void setup()
{
  //Serial.begin(115200); // Define o baud rate
  pinMode(pino_saida, OUTPUT); //Define por onde vai sair a MSG
  pinMode(pino_entrada, INPUT);

  pinMode(pinoONOFF, OUTPUT); // Pino com controle liga/desliga
  analogWrite(pinoPWM, 0);    // Pino com controle PWM


  Serial.begin (9600);

  lcd.init();
  lcd.backlight();
  lcd.cursor();
  lcd.blink();

  //Pinos ligados aos pinos 34, 36, 38, 40 e 42 do Arduino - Linhas do botão circular
  pinMode(34, OUTPUT);
  pinMode(36, OUTPUT);
  pinMode(38, OUTPUT);
  pinMode(40, OUTPUT);
  pinMode(42, OUTPUT);

  //Pinos ligados aos pinos 44, 46, 48, 50 e 51 do Arduino - Linhas do teclado
  pinMode(44, OUTPUT);
  pinMode(46, OUTPUT);
```

```
  pinMode(48, OUTPUT);
  pinMode(50, OUTPUT);
  pinMode(52, OUTPUT);

  //Pinos ligados aos pinos 31, 33, 35, 37, 39, 41, 43, 45, 47, 49 e 51 do Arduino - Colunas do
teldado
  pinMode(31, INPUT);
  pinMode(33, INPUT);
  pinMode(35, INPUT);
  pinMode(37, INPUT);
  pinMode(39, INPUT);
  pinMode(41, INPUT);
  pinMode(43, INPUT);
  pinMode(45, INPUT);
  pinMode(47, INPUT);
  pinMode(49, INPUT);
  pinMode(51, INPUT);

  //Pino ligado ao pino 44 do Arduino - Colunas do botão circular
  pinMode(32, INPUT);

  Serial.println("Aguardando acionamento das teclas...");
  Serial.println();
}

/*****************************************************************************
* FUNÇÕES TECLADO/DISPLAY
*****************************************************************************/
void teclado(){
    for (int ti = 34; ti<53; ti = ti+2)
    {
      //Alterna o estado dos pinos das linhas
      digitalWrite(34, LOW); //linha 1 botão circular
      digitalWrite(36, LOW); //linha 2 botão circular
      digitalWrite(38, LOW); //linha 3 botão circular
      digitalWrite(40, LOW); //linha 4 botão circular
      digitalWrite(42, LOW); //linha 5 botão circular

      digitalWrite(44, LOW); //linha 1
      digitalWrite(46, LOW); //linha 2
      digitalWrite(48, LOW); //linha 3
      digitalWrite(50, LOW); //linha 4
      digitalWrite(52, LOW); //linha 5
      digitalWrite(ti, HIGH);

      if (ti == 44){     //troca o número das linhas para 1 a 5
            x = 1;
        }else if(ti == 46){
            x = 2;
        }else if(ti == 48){
```

```
      x = 3;
    }else if(ti == 50){
      x = 4;
    }else if(ti == 52){
      x = 5;
    }else if(ti == 34){
      x = 6;
    }else if(ti == 36){
      x = 7;
    }else if(ti == 38){
      x = 8;
    }else if(ti == 40){
      x = 9;
    }else if(ti == 42){
      x = 10;
    }

//Verifica se alguma tecla da coluna 1 foi pressionada
if (digitalRead(51) == HIGH)
{

  imprime_caracter(x, 1);
  while(digitalRead(51) == HIGH){}
}

//Verifica se alguma tecla da coluna 2 foi pressionada
if (digitalRead(49) == HIGH)  //49
{

  imprime_caracter(x, 2);
  while(digitalRead(49) == HIGH){}; //49
}

//Verifica se alguma tecla da coluna 3 foi pressionada
if (digitalRead(47) == HIGH)
{

  imprime_caracter(x, 3);
  while(digitalRead(47) == HIGH){}
}

//Verifica se alguma tecla da coluna 4 foi pressionada
if (digitalRead(45) == HIGH)
{

  imprime_caracter(x, 4);
  while(digitalRead(45) == HIGH){}
}

//Verifica se alguma tecla da coluna 5 foi pressionada
```

```
if (digitalRead(43) == HIGH)
{

  imprime_caracter(x, 5);
  while(digitalRead(43) == HIGH){}
}

 //Verifica se alguma tecla da coluna 6 foi pressionada
if (digitalRead(41) == HIGH)
{

  imprime_caracter(x, 6);
  while(digitalRead(41) == HIGH){}
}

 //Verifica se alguma tecla da coluna 7 foi pressionada
if (digitalRead(39) == HIGH)
{

  imprime_caracter(x, 7);
  while(digitalRead(39) == HIGH){}
}

//Verifica se alguma tecla da coluna 8 foi pressionada
if (digitalRead(37) == HIGH)
{

  imprime_caracter(x, 8);
  while(digitalRead(37) == HIGH){}
}

//Verifica se alguma tecla da coluna 9 foi pressionada
if (digitalRead(35) == HIGH)
{

  imprime_caracter(x, 9);
  while(digitalRead(35) == HIGH){}
}

//Verifica se alguma tecla da coluna 10 foi pressionada
if (digitalRead(33) == HIGH)
{
  imprime_caracter(x, 10);
  enter (x, 10);
  while(digitalRead(33) == HIGH){}
}

 //Verifica se alguma tecla da coluna 11 foi pressionada
if (digitalRead(31) == HIGH)
{
```

```
        imprime_caracter(x, 11);
        backspace (x, 11);
        while(digitalRead(31) == HIGH){}
      }

      //Verifica se alguma tecla do botão circular foi pressionada
      if (digitalRead(32) == HIGH)
      {

        imprime_caracter(x, 12);
        if(x == 6){
          Serial.println("direita");
          proxMsg();

        }else if(x == 7){
          Serial.println("OK");

        }else if(x == 8){
          Serial.println("desce");
          sobe();

        }else if(x == 9){
          desce();
          Serial.println("sobe");

        }else if(x == 10){
          Serial.println("esquerda");
          msgAnt();
        }

        while(digitalRead(32) == HIGH){}
      }
    }

}

void imprime_caracter(int x, int y)
{
    for (int j = 0; j < n_caracteres ; j++){
        if(matriz[j].linha == x && matriz[j].coluna == y){

          char letra = matriz[j].character;
          Serial.print(letra);

          lcd.cursor();
          lcd.blink();
          lcd_col++;
          lcd.setCursor(lcd_col, 0);

          mensagem = mensagem + letra;
```

```
        lcd.clear();
        lcd.setCursor (0,0);
        lcd.print(mensagem);
        Serial.print (" ");
        Serial.println (mensagem);
      }
    }
}


void backspace (int x, int y){

  if(x == 1  && y == 11){
    if(lcd_col > 0){

      lcd_col--;
      lcd.setCursor(lcd_col, 0);
      lcd.write(' ');
      lcd.setCursor(lcd_col, 0);

      mensagem.remove(mensagem.length()-1);

      Serial.print (" ");
      Serial.println (mensagem);
    }
  }
}

void enter (int x, int y){

  if(x == 3  && y == 10){

      //lcd.home();
      lcd.clear();
      lcd_col = 0;
      lcd.setCursor(lcd_col, 0);
      mensagem = "e" + mensagem; // Coloca um ponto na frente da mensagem para acionar o
modo TX do rádio e evitar perda de dados
      envio_MSG_morse(mensagem); // Função que envia a MSG por código Morse
      mensagem = "";
  }
}

void proxMsg(){
  if(posUltMsg - 1 > posMsgTela){
    posMsgTela++;
    String msgAtual = mensagens[posMsgTela];
    zeraLinhas(); //limpos as linhas
    imprimeLcd(msgAtual);
  }
}
```

```
void msgAnt(){
  if(posMsgTela > 0){
    posMsgTela--;
    String msgAtual = mensagens[posMsgTela];
    zeraLinhas(); //limpos as linhas
    imprimeLcd(msgAtual);
  }
}

void sobe(){
  if(linhas[posLinhaExibida + 1] != ""){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(linhas[posLinhaExibida + 1]);

    lcd.setCursor(0, 1);
    lcd.print(linhas[posLinhaExibida + 2]);

    lcd.setCursor(0, 2);
    lcd.print(linhas[posLinhaExibida + 3]);

    lcd.setCursor(0, 3);
    lcd.print(linhas[posLinhaExibida + 4]);

    posLinhaExibida = posLinhaExibida + 4;
  }
}

void desce(){
  if(posLinhaExibida - 4 >= 0){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(linhas[posLinhaExibida - 7]);

    lcd.setCursor(0, 1);
    lcd.print(linhas[posLinhaExibida - 6]);

    lcd.setCursor(0, 2);
    lcd.print(linhas[posLinhaExibida - 5]);

    lcd.setCursor(0, 3);
    lcd.print(linhas[posLinhaExibida - 4]);

    posLinhaExibida = posLinhaExibida - 4;
  }
}

void imprimeLcd(String texto){
  texto += fimMsg;
```

```
  while(texto.length() > 0){
    lcd.noBlink();
    lcd.noCursor();
    lcd.clear();
    lcd.setCursor(0, 0);
    String msg1 = texto.substring(0,20);
    msg1.trim();
    linhas[poslinha] = msg1;
    poslinha++;
    texto.remove(0,20);
    lcd.print(msg1);

    lcd.setCursor(0, 1);
    String msg2 = texto.substring(0,20);
    msg2.trim();
    linhas[poslinha] = msg2;
    poslinha++;
    texto.remove(0,20);
    lcd.print(msg2);

    lcd.setCursor(0, 2);
    String msg3 = texto.substring(0,20);
    msg3.trim();
    linhas[poslinha] = msg3;
    poslinha++;
    texto.remove(0,20);
    lcd.print(msg3);

    lcd.setCursor(0, 3);
    String msg4 = texto.substring(0,20);
    msg4.trim();
    linhas[poslinha] = msg4;
    posLinhaExibida = poslinha;
    poslinha++;
    texto.remove(0,20);
    lcd.print(msg4);
    delay(1500);
  }
}

void zeraLinhas(){
  poslinha=0; //zera a posição da linha
  int i = 0;
  while (linhas[i] != ""){
    linhas[i] = "";
    i++;
  }
}
/*****************************************************************************
* FUNCÃO DE RECEPÇÃO
*****************************************************************************/
```

```
String RX() {  // Recebe String codificada em Morse

  char         caracter_lido;           // Caractere lido
  String       msg_recebida = "";       // Mensagem recebida
  String       codigo_lido = "";        // Código Morse sendo lido
  boolean      espaco = true;           // Flag de espaço
  unsigned long tempo_anterior = millis();   // Tempo anterior
  unsigned long tempo_do_pulso = 0;             // Tempo do caractere

  while (millis() - tempo_anterior < 6 * tempo_do_ponto) {  // Aguarda mensagem por 7 pontos =
espaço entre palavras
     teclado();
     if (analogRead(pino_entrada)) {

        tempo_do_pulso = millis(); // Inicio de sinal

        while (analogRead(pino_entrada));

           tempo_do_pulso = millis() - tempo_do_pulso;

           if (tempo_do_pulso < tempo_do_ponto + tempo_do_ponto / 3 && tempo_do_pulso > (2
* tempo_do_ponto)/3) {
             codigo_lido += "."; // Ponto
             //Serial.print(".");
           }

         if (tempo_do_pulso > tempo_do_ponto * 2 && tempo_do_pulso < tempo_do_ponto * 4) {
           codigo_lido += "-";// Traço
           //Serial.print("-");
         }

         delay(tempo_do_ponto / 10); // espera para não perder buffer
         tempo_anterior = millis();
     }
     if (millis() - tempo_anterior > tempo_do_ponto * 3 && !espaco) {

       // Espaço
       msg_recebida += " ";

       Serial.print(" / ");
       espaco = true;
     }

     if (millis() - tempo_anterior > tempo_do_ponto && codigo_lido != ""){

        Serial.print (codigo_lido);

        for (int i = 0 ; i < n_caracteres ; i++){
          if(matriz[i].codigo == codigo_lido){
```

```
            caracter_lido = matriz[i].character;
            Serial.print(" " + String(caracter_lido) + " ");
            codigo_lido = "";

            msg_recebida += caracter_lido; // Cria a mensagem a partir de cada caracter recebido
            espaco = false;

          }// end if
        }//end for
      }
  } //END WHILE

  msg_recebida.trim(); //Retira os espaços do início e do final da mensagem recebida
  return msg_recebida;
}// END RX()

/********************************************************************************
* FUNCÃO DE TRANSMISSÃO
********************************************************************************/
void envio_MSG_morse(String texto) // Envia MSG codificada em código Morse
{
  String cod_caracter;
  for (int i = 0; i < texto.length(); i ++) // Percorre o texto para pegar cada um dos caracteres
  {
    char caracter = toupper(texto[i]);// pega o caracter em maiusculo
    for (int j = 0 ; j < n_caracteres ; j++) // For para percorrer o struct (biblioteca_morse)
    {
      if (caracter == matriz[j].character) // Acha o código equivalente ao caracter selecionado
      {
        cod_caracter = matriz[j].codigo;
      }
    }

    Serial.print(caracter); // Imprime o caracter
    cod_caracter.trim();//Remove quaisquer espaços no começo ou final da String
    Serial.print(" " + cod_caracter + " "); // Imprime na saída serial os caracteres em codigo Morse
    for (byte i = 0; i < cod_caracter.length(); i++) // Envia o código morse para o PIN 13 (Rádio)
    {
      switch (cod_caracter[i]) {
        case '.': // Se DIT (ponto) chama a função DIT
          DIT();
          break;
        case '-': // Se DAH (traço) chama a função DAH
          DAH();
          break;
        default:
          vazio(); // espaço entre os DIT ou DAH
        break;
      }
    }
```

```
    vazio(); //espaço entre caracteres
  }
Serial.println ("");
}

void DIT() // Envia um tempo_do_ponto (um ponto)
{
  digitalWrite(pino_saida, HIGH);
  delay(tempo_do_ponto); // Entre cada tempo_do_ponto há uma pausa de mesma duração de
um ponto
  digitalWrite(pino_saida, LOW);
  delay(tempo_do_ponto);
}

void DAH() // Envia dah (traço) que equivale a três pontos
{
  digitalWrite(pino_saida, HIGH);
  delay(tempo_do_ponto * 3);
  digitalWrite(pino_saida, LOW);
  delay(tempo_do_ponto);
}

void vazio() { // Envia intervalo vazio
  delay(tempo_do_ponto * 2 );
}

/**************************************************************************
* LOOP
**************************************************************************/
void loop()
{
  teclado();
  if (Serial.available()) // Se tiver alguma mensangem entra na condicional
  {
    String msg = Serial.readString(); // Lê os caracteres do serial e os armazena em uma string
    msg.trim(); //Remove quaisquer espaços no começo ou final da String
    if (msg != "") // Se a MSG não for vazia ele envia ela por código Morse
    {
      msg = "e" + msg; // Coloca um ponto na frente da mensagem para acionar o modo TX do
rádio e evitar perda de dados
      envio_MSG_morse(msg); // Função que envia a MSG por código Morse
    }
  }

  String msg_recebida = RX();

  if (imprimiu == 0){
    msg_recebida = mensagem1;
  }
```

```
  if (imprimiu == 1){
    msg_recebida = mensagem2;
  }

  if (imprimiu == 2){
    msg_recebida = mensagem3;
  }
  imprimiu++;

  if (msg_recebida != "") {

    mensagens[posUltMsg] = msg_recebida;
    posMsgTela = posUltMsg;
    posUltMsg++;
    zeraLinhas(); //limpos as linhas
    imprimeLcd(msg_recebida);

    Serial.println("\n" + msg_recebida);// Mensagem recebida

    byte espaco = msg_recebida.indexOf(" ");// Efetua tratamento
    String comando = msg_recebida.substring(0, espaco);
    int valor = msg_recebida.substring(espaco + 1).toInt();

    if (comando == "LIGA") {              // Verifica comando

      digitalWrite(pinoONOFF, HIGH);  // Ligar pino
    } else if (comando == "DESLIGA") {    // Desligar pino

      digitalWrite(pinoONOFF, LOW);
    } else if (comando == "PWM") {        // Controle PWM

      analogWrite(pinoPWM, valor);
    }
  }
} // END LOOP
```