Armazenador e emissor de código infravermelho

Gabriel da Silva Soares
Faculdade do Gama
Universidade de Brasília
Gama - DF, Brasil

Ricardo Vieira Borges
Faculdade do Gama
Universidade de Brasília
Gama - DF, Brasil

I. RESUMO

O Projeto proposto consiste em um armazenador de sinal infravermelho qualquer através de um receptor. O projeto também será capaz de emitir esse mesmo sinal armazenado em uma memória do microcontrolador por meio de um LED infravermelho, o projeto será capaz de armazenar múltiplos códigos infravermelhos que foram lidos, respeitando os limites de memória do microcontrolador. Todos os sinais de controle e saídas são controlados pelo microcontrolador MSP-430G2553 da Texas Instruments.

II. INTRODUÇÃO

Muitas vezes necessita-se de vários controles (infravermelho) para diversos eletrodomésticos em uma casa. Nem sempre é cômodo manter todos os controles à mão para usufruir de seus dispositivos. Além disso, alguns equipamentos possuem funções exclusivas somente em seu controle remoto.

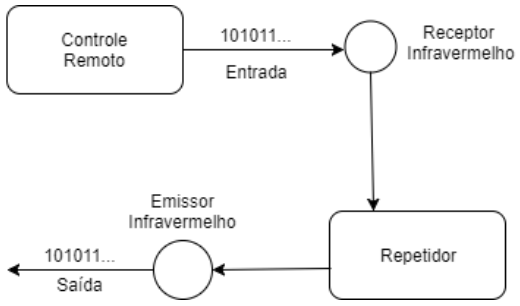

Figura 1 – esquema do projeto.

Um controle remoto infravermelho envia dados por modulação de frequência (no caso dos controles remotos infravermelhos, é utilizada a modulção ASK), ou seja, é definida uma frequência portadora que é reconhecida pelo receptor quando ativada. Os sitemas atuais de emissão e recepção infravermelha possuem apenas uma frequência portadora no sistema, normalmente de 38kHz, mas podendo ser também de 36kHz ou 40kHz. O LED infravermelho envia essa frequência ou não por diferentes intervalos dependendo do modelo do controle remoto.

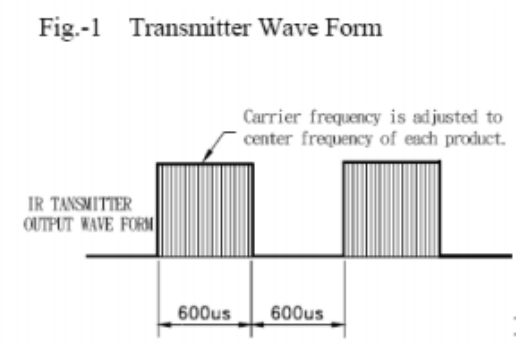

Figura 2 – exemplo de modulação ASK.

Normalmente quando há emissão da frequência portadora, os receptores mais comuns codificam o sinal para nível lógico baixo, quando não há emissão (led apagado) o receptor codifica para nível lógico alto, mas isso pode variar de acordo com o modelo do receptor codificado. A modulação ASK é utilizada pois há muita interferência luminosa no meio em que o aparelho está, o filtro no receptor só permite sinais com frequências próximas da frequência portadora e corta outras frequências.

Em controle remotos comerciais comuns, temos uma códificação de bits por largura de pulsos para indicar se o código irá começar, se o próximo pulso é de dado ou término de envio e etc. A quantidade de bits enviados varia em cada marca, mas raramente passam de 32 bits de dados. Podemos utilizar como exemplo um controle de um aparelho Sony, que gera um código de 12 bits, repetindo o mesmo sinal após

um longo pulso em nível lógico baixo indicando que o código terminou.

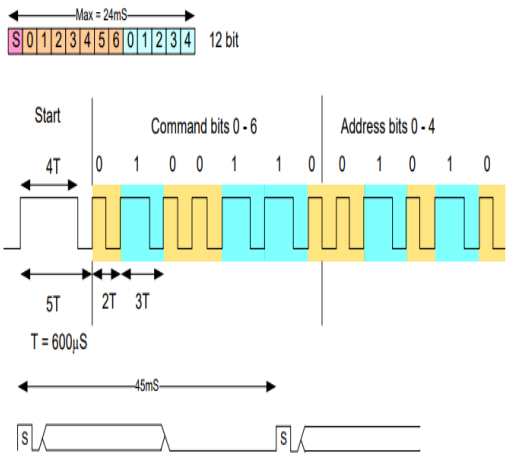

Figura 3 – emissão de dados para um controle comercial sony.

III. DESENVOLVIMENTO

Observando como a modulação ASK funciona e a forma que um controle infravermelho envia o sinal, pensou-se na possibilidade de medir o tempo de duração de cada pulso que o receptor decodificado gerava de acordo com o código do controle a ser copiado, sendo nível lógico alto ou baixo, a fim de reproduzir esses mesmos tempos no LED infravermelho e retransmitir o sinal o mais fielmente possível.

Como o sinal no LED não deve ser simplesmente uma modulação de pulsos, mas sim com a frequência portadora sendo ativada dentro dos pulsos altos, chegou-se a solução de gerar a frequência portadora independente com outro chip MSP-430 e combiná-la com uma máscara (modulação por pulsos) através de uma porta AND.

Inicialmente foi definido a adição de vários botões associados aos sinais a serem copiados para uma maior funcionalidade do controle, mas devido a baixa capacidade de memória do chip MSP-430 a quantidade de botões teve de ser reduzida, mas é algo que pode ser corrigido com chips de memória externa ou outros chips MSP se comunicando entre si.

As funções a ser implementadas no controle são de cópia, que através de um botão seleciona a função, que espera até o botão em que se deseja copiar o sinal seja apertado e o sinal do controle a ser copiado seja emitido, após a cópia do sinal o controle fica em função de enviar, que aguarda o acionamento de algum botão em que se copiou algum sinal para enviar o mesmo sinal para o LED.

Deseja-se implementar todo o circuito e alimentação em uma placa de fenolite, para obter um projeto mais compacto e portátil.

- Descrição de Hardware:

Após a realização de testes para verificar os equipamentos que seriam utilizados no projeto foram feitas pesquisas sobre o assunto e códigos para verificar a funcionalidade de algumas funções com os equipamentos.

O receptor infravermelho utilizado já possui um circuito interno codificador que permite frequências próximas a 38kHz. Para a utilização de um transistor sensível a luz infravermelha, teriam que ser feitos circuitos de filtragem e decodificação para transformar os pulsos em informação simples para o microcontrolador.

Figura 4 – receptor infravermelho codificado modelo LL-M2638.

Após alguns testes utilizado o receptor para receber sinais de um controle qualquer, observou-se que o mesmo possui saída invertida, fica em nível lógico alto quando não recebe pulsos, e nível lógico baixo quando recebe a frequência portadora.

A distância de captação de pulsos depende não apenas do receptor, mas também da intensidade da luz emitida, o LED infravermelho ligado diretamente a saída da AND com alimentação de comum ao MSP-430 (3,3V) oferece uma emissão razoável, mas não satisfatória, por isso foi utilizada uma alimentação de voltagem mais elevada (5V) para a AND que fornece a modulação ASK para o LED, mas todo o circuito funciona com alimentação de 3,3V em todos os

componentes, apenas com perda de brilho no LED.

Figura 5 – LED infravermelho.

Para a alimentação do circuito foram utilizadas duas pilhas AA em série gerando aproximadamente 3,3V para alimentar os microcontroladores e o receptor, uma bateria de 9V ligada a um regulador de tensão para 5V alimentando a porta AND (bateria dispensável com consequência de perda de potência no LED).

Após alguns testes individuais nos componentes, um protótipo funcional com apenas um botão de cópia e um botão de envio foi montado para observar seu comportamento, já que a adição de outros botões de envio são apenas replicações do mesmo botão com endereços diferentes.

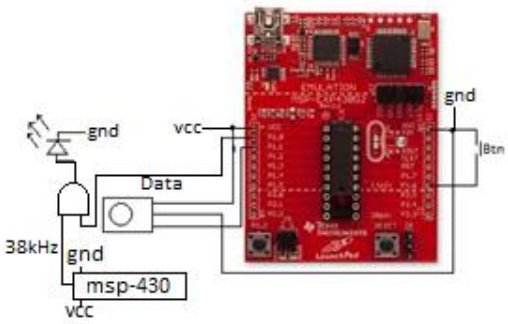

Figura 6 – esquemático funcional, utilizando um botão da placa para selecionar função de cópia.

O protótipo funcional foi transferido para um circuito portátil, sem a placa de programação do MSP e alimentado pelas pilhas e baterias, foram adicionados mais dois botões de envio e um LED para indicar acionamento da função copiar.

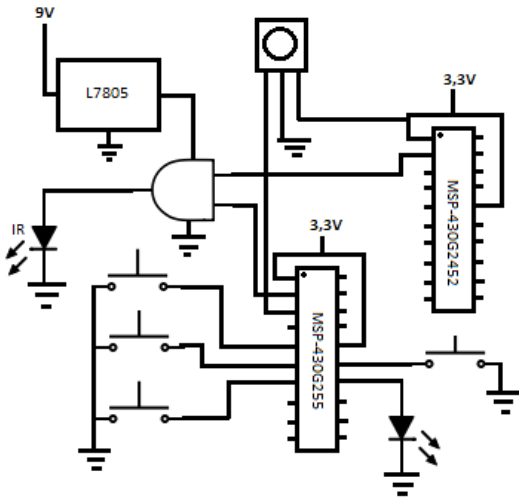

Figura 7 – Esquemático do circuito portátil.

A partir do circuito acima foi gerado um layout de PCB para imprimir o circuito em uma placa de fenolite e soldar os componentes junto com os adaptadores para pilhas e bateria.

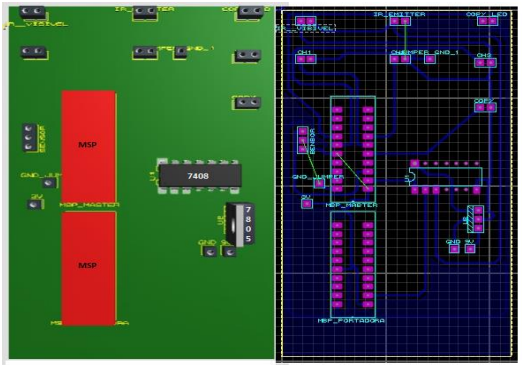

Figura 8 – layouts das PCBs.

Materiais Utilizados:

Componente	Quantidade	Preço
Launchpad MSP430*	1	R$ 40,00
Push-Button	4	R$ 0,10
Porta AND (7408)	1	R$ 1,49
Receptor IR cod.	1	R$ 1,00
LED IR	1	R$ 0,31
Regulador LM7805	1	R$ 0,89
Placa de fenolite	1	R$ 6,00
Pilhas e Bateria	3	R$ 10,00
LED	1	R$ 0,13

*Acompanha 2 chips da família MSP-430G

Descrição de software:

Após constatado o bom funcionamento dos componentes, os códigos para as funções de captura e emissão começaram a ser desenvolvidos, começando pela captura.

A ideia básica para captura foi a de detectar mudanças de nível lógico e contar quanto tempo elas duram. Para isso usou-se o registrador do Timer A com o registrador de comparação/captura no modo de captura.

TA0R recebe SMCLK e é incrementado até o valor máximo de 0xFFFF.

TA0CRR0 recebe o valor de TA0R quando há uma borda de subida ou descida no sinal de entrada (no caso o receptor), este modo é definido como modo de captura.

Após uma interrupção gerada por um botão pré-definido, entramos na função de captura de dados, onde é esperada outra interrupção por borda de subida ou descida que vem do receptor infravermelho, quando há essa interrupção uma flag é ativada (CCIFG) que é usada como condição no código, após a interrupção ocorrer o valor salvo no registrador de captura é salvo em uma posição de um vetor (count[i]), ou seja, o tempo que o pulso durou, após zerar a flag e o timer, é esperada outra interrupção e o processo se repete, salvando os valores em outras posições do vetor.

O tamanho do vetor varia de acordo com o código do controle a ser copiado, mas foi assumido um valor mínimo de 70 posições, para receber códigos de 32-bits por exemplo, que geram 64 posições no vetor.

A parte de emissão também usa o Timer A e o registrador de comparação/captura, agora em modo de comparação.

TA0R utiliza os mesmos parâmetros

TA0CCR0 recebe os valores do vetor onde as capturas de tempo foram salvas, TA0R é incrementado até o valor da posição no vetor, ativando uma flag quando atinge o valor, e reiniciando a contagem, este modo é denominado como modo de comparação.

Após uma interrupção gerada por um botão pré-definido, é ativada a função de envio de dados pelo LED infravermelho, onde a largura dos pulsos em nível alto ou baixo são definidos pelos tempos salvos na parte de captura (count[i]).

Para um melhor entendimento, observe o código em anexo.

Este código é repetido dentro de condições de acordo com a quantidade de botões que podem ser salvos, sem nenhuma alteração em sua base.

A quantidade de botões que podem ser salvos foi limitada a 3 botões, pois a memória RAM do chip utilizado é de 512 bytes, não permitindo o uso de outro botão, ou seja, mais um vetor de no mínimo 70 posições.

IV. RESULTADOS

Foi realizado o debug no código descrito, observando se as transições e interrupções estavam ocorrendo da forma esperada, foram utilizados botões para simular as transições mais rápidas, como a do receptor. Não foram constatados erros no código ou transições indevidas.

Com o circuito real montado, foi realizada a tentativa de salvar nos vetores os tempos de largura de pulso e seus níveis lógicos respectivos. Um controle remoto de um aparelho Sony foi utilizado para a observação de suas transições, a seleção de cópia de sinal foi inicializada por um botão, aguardando apenas a primeira interrupção (pulso) gerada pelo controle remoto, o botão liga/desliga do controle foi apertado e observamos os seguintes valores no vetor de captura (count[i]):

nivel	unsigned int[80]	[0x0000,0x0008,0x0000,0x0008,0x0000...] (Hex)
count	unsigned int[80]	[-30564,2480,478,1286,459...] (Decimal)
[0]	unsigned int	-30564 (Decimal)
[1]	unsigned int	2480 (Decimal)
[2]	unsigned int	478 (Decimal)
[3]	unsigned int	1286 (Decimal)
[4]	unsigned int	459 (Decimal)
[5]	unsigned int	715 (Decimal)
[6]	unsigned int	482 (Decimal)
[7]	unsigned int	1289 (Decimal)
[8]	unsigned int	478 (Decimal)
[9]	unsigned int	689 (Decimal)
[10]	unsigned int	457 (Decimal)
[11]	unsigned int	1312 (Decimal)
[12]	unsigned int	476 (Decimal)
[13]	unsigned int	692 (Decimal)
[14]	unsigned int	482 (Decimal)
[15]	unsigned int	694 (Decimal)
[16]	unsigned int	455 (Decimal)
[17]	unsigned int	1309 (Decimal)
[18]	unsigned int	459 (Decimal)
[19]	unsigned int	715 (Decimal)
[20]	unsigned int	480 (Decimal)
[21]	unsigned int	691 (Decimal)
[22]	unsigned int	460 (Decimal)
[23]	unsigned int	710 (Decimal)
[24]	unsigned int	482 (Decimal)
[25]	unsigned int	692 (Decimal)
[26]	unsigned int	25572 (Decimal)
[27]	unsigned int	2483 (Decimal)
[28]	unsigned int	474 (Decimal)
[29]	unsigned int	1290 (Decimal)
[30]	unsigned int	456 (Decimal)

Figura9 – valores armazenados nos vetores.

No vetor podemos observar quantos ciclos de clk (no caso clk = 1MHz) duraram os pulsos codificados pelo receptor.

Podemos observar as transições, incluindo começo do código (start) com maior tempo (count[1]), pulsos de pré dados e pulsos de dados, observamos também em count[26] um pulso longo em nível lógico baixo e outro pulso de start para a repetição do código, aparentemente uma leitura aceitável considerando que nem o controle remoto original copiado tem alta precisão nos tempos da modulação ASK.

Após conferido o correto funcionamento de recepção e armazenamento do sinal, focou-se na parte de emissão, que no começo não saiu como o esperado, o sinal que era gerado na saída não correspondia a modulação recebida pelo controle, o sinal de saída possuía características totalmente desconhecidas. Depois de uma fina análise no circuito uma interferência do sensor de recepção foi detectada quando a função enviar estava ativada. O pino que recebia seus dados foi desabilitado enquanto a função enviar estiver ativa e o problema foi resolvido.

Após a resolução do problema o programa foi testado copiando o código do controle de um projetor em sala de aula, o vetor de dados foi conferido e o código do controle não ultrapassava as 70 posições de vetor, o LED foi apontado para o aparelho que recebeu o sinal com sucesso.

O projeto foi testado em diversos outros equipamentos, sempre copiando o sinal de algum botão de seu controle e o repetindo perfeitamente, sinal este que era sempre identificado pelo aparelho.

Para controles IR de ar-condicionado, na maioria das vezes, temos uma emissão de 50-bits de dados, totalizando 100 posições no vetor que armazenaria seu código. Para cópia e emissão desse tipo de controle o código do projeto teve de ser alterado, aumentando dois dos vetores de 70 posições para 100 posições e descartando o terceiro botão por questões de limitação de memória.

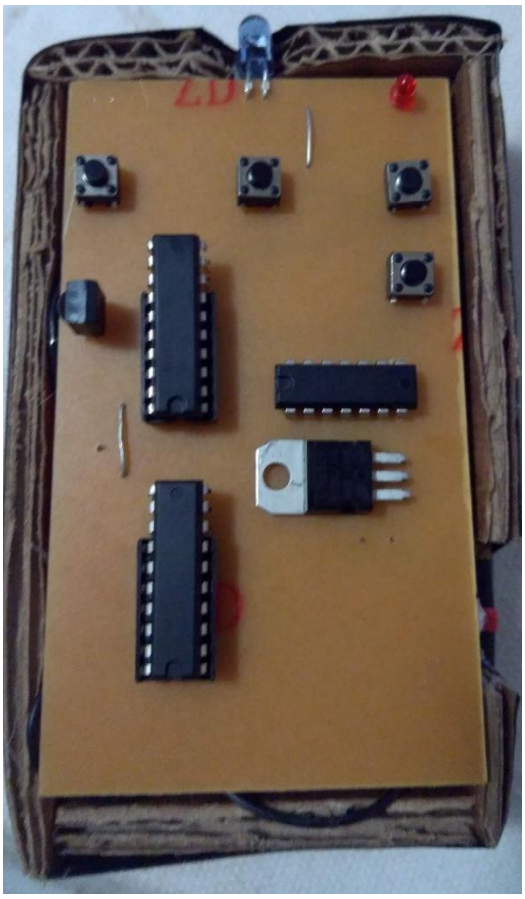

Figura10 – Protótipo final do projeto

V. CONCLUSÃO

Um projeto que copia código infravermelho, armazena e replica o mesmo com perfeição pode se mostrar bastante útil no dia-a-dia, onde há diversos controles para cada aparelho, com funções principais que poderiam ser copiadas para um aparelho só. O projeto proposto possui a funcionalidade para essa prática, pois mostrou que é capaz de controlar qualquer aparelho que aceite código infravermelho, mas não possui a quantidade de botões suficientes, problema esse que poderia ser resolvido com uma memória externa acoplada ao microcontrolador.

As aplicações para o projeto não se limitam apenas a cópia e reprodução de código copiado, também há a possibilidade de gerar seu próprio código com as devidas alterações no programa, e controlar dispositivos feitos para receber este tipo de código criado.

VI. REVISÃO BIBLIOGRÁFICA

Preços dos componentes. Diponível em: http://www.huinfinito.com.br. Acesso em: 29/11/2017

Funcionamento do controle remoto infravermelho. Disponível em: http://www.hpspin.com.br/site1/circuitos/ctremoto/ Acesso em 04/09/2017.

Codificação de controles sony. Disponível em: http://picprojects.org.uk/projects/sirc/sonysirc.pdf Acesso em: 01/11/2017.

Datasheet de receptor infravermelho LL-M2638. Disponível em: http://www.futurlec.com/LED/INFRECMOD.shtml Acesso em: 01/11/2017.

ANEXO

```
#include <msp430g2553.h>
#define RECEPTOR BIT1
#define LEDIR BIT0
#define SELCOPY BIT3
#define BTNA BIT4
#define BTNB BIT5
#define BTNC BIT7
#define LEDCOPY BIT6
#define ARRAY 70

volatile unsigned int i = 0;
volatile unsigned int j = 0;

unsigned int counta[ARRAY];
unsigned int countb[ARRAY];
unsigned int countc[ARRAY];



int main(void)
{

    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

    BCSCTL1 = CALBC1_8MHZ;      // sets de MCLK e SMCLK XMHz
    DCOCTL = CALDCO_8MHZ;

    P1OUT &= 0x0000;
    P1DIR &= ~(RECEPTOR + SELCOPY + BTNA + BTNB + BTNC);        // entradas
    P1SEL |= RECEPTOR;         // pino P1.1 como entrada digital para
periférico, no caso a entrada CCIS do TA0CCTL0
    P1DIR |= LEDIR + LEDCOPY;              // pino P1.0 como saida
    P1REN |= RECEPTOR + SELCOPY + BTNA + BTNB + BTNC;         // habilita
resistores
    P1OUT |= SELCOPY + RECEPTOR + BTNA + BTNB + BTNC;                    //
resistor de pull-up

     while(1){
             j=0;
             P1OUT &= ~LEDIR;
             P1OUT &= ~LEDCOPY;
             if((P1IN&SELCOPY)==0){      //////////////função
copiar//////////////////

                 P1SEL |= RECEPTOR;
                 P1OUT |= LEDCOPY;
                 TA0CTL = TACLR;
                 TA0CTL = TASSEL_2 + ID_3 + MC_2;   // TA recebe SMCLK e /1,
modo up até 0xFFFF
                 TA0CCTL0 = CAP + CM_3 + CCIS_0 + SCS + OUTMOD_0;   //capture
mode (salva valor de TA0R no TA0CCR0 quando detecta borda),

//detectar borda de subida e descida,
                                                                    //sinal
externo (P1.1), síncrono com o clk, outmod desnecessário
                 while(1){
```

```
                if(j==1){
                    break;
                }

                else if((P1IN&BTNA)==0){        //botao a

                    for(i=0;i<ARRAY;i++){
                        counta[i] = 0;
                    }
                    i=0;

                    while(1){

                        if(TA0CCTL0&CCIFG){  //borda detectada ativa flag
                            P1OUT ^= LEDCOPY;
                            TA0CTL = TACLR;
                            TA0CTL = TASSEL_2 + ID_3 + MC_2;   // TA recebe
SMCLK e /1, modo up até 0xFFFF
                            counta[i-1] = TA0CCR0;          //salva
contagens num array
                            TA0CCTL0 &= ~CCIFG;            //zera flag
                            j=1;
                            if(counta[i]>10000){
                                break;
                            }
                            i++;
                            if(i>ARRAY){        //limitando array
                                break;
                            }
                        }
                    }
                }

                else if((P1IN&BTNB)==0){        //botao b

                    for(i=0;i<ARRAY;i++){
                        countb[i] = 0;
                    }
                    i=0;

                    while(1){

                        if(TA0CCTL0&CCIFG){  //borda detectada ativa flag
                            P1OUT ^= LEDCOPY;
                            TA0CTL = TACLR;
                            TA0CTL = TASSEL_2 + ID_3 + MC_2;   // TA recebe
SMCLK e /1, modo up até 0xFFFF
                            countb[i-1] = TA0CCR0;          //salva
contagens num array
                            TA0CCTL0 &= ~CCIFG;            //zera flag
                            j=1;
                            if(countb[i]>10000){
                                break;
                            }
                            i++;
                            if(i>ARRAY){        //limitando array
                                break;
                            }
                        }
```

```
                    }
                }

                else if((P1IN&BTNC)==0){       //botao c

                    for(i=0;i<ARRAY;i++){
                        countc[i] = 0;
                    }
                    i=0;

                    while(1){

                        if(TA0CCTL0&CCIFG){  //borda detectada ativa flag
                            P1OUT ^= LEDCOPY;
                            TA0CTL = TACLR;
                            TA0CTL = TASSEL_2 + ID_3 + MC_2;   // TA recebe
SMCLK e /1, modo up até 0xFFFF
                            countc[i-1] = TA0CCR0;          //salva
contagens num array
                            TA0CCTL0 &= ~CCIFG;            //zera flag
                            j=1;
                            if(countc[i]>10000){
                                break;
                            }
                            i++;
                            if(i>ARRAY){        //limitando array
                                break;
                            }
                        }
                    }
                }
                }
            }


            else if((P1IN&BTNA)==0){   //////////////////////função enviar
a//////////////////////////

                P1SEL &= ~RECEPTOR;
                P1OUT &= ~LEDIR;
                TA0CTL = TACLR;
                TA0CTL = TASSEL_2 + ID_3 + MC_1;
                TA0CTL &= ~TAIFG;
                TA0CCR0 = 0x20;
                i=0;

                while(1){

                    if(TA0CTL&TAIFG){
                        if(i>ARRAY){
                            break;
                        }
                        P1OUT ^= LEDIR;
                        TA0CCR0 = counta[i];     //usa tempo salvo na função
copy

                        TA0CTL &= ~TAIFG;
                        i++;
                    }
```

```
                }
            }

            else if((P1IN&BTNB)==0){   ////////////////////////função enviar
b//////////////////////////

                P1SEL &= ~RECEPTOR;
                P1OUT &= ~LEDIR;
                TA0CTL = TACLR;
                TA0CTL = TASSEL_2 + ID_3 + MC_1;
                TA0CTL &= ~TAIFG;
                TA0CCR0 = 0x20;
                i=0;

                while(1){

                    if(TA0CTL&TAIFG){
                        if(i>ARRAY){
                            break;
                        }
                        P1OUT ^= LEDIR;
                        TA0CCR0 = countb[i];     //usa tempo salvo na função
copy
                        TA0CTL &= ~TAIFG;
                        i++;
                    }
                }
            }

            else if((P1IN&BTNC)==0){   ////////////////////////função enviar
c//////////////////////////

                P1SEL &= ~RECEPTOR;
                P1OUT &= ~LEDIR;
                TA0CTL = TACLR;
                TA0CTL = TASSEL_2 + ID_3 + MC_1;
                TA0CTL &= ~TAIFG;
                TA0CCR0 = 0x20;
                i=0;

                while(1){

                    if(TA0CTL&TAIFG){
                        if(i>ARRAY){
                            break;
                        }
                        P1OUT ^= LEDIR;
                        TA0CCR0 = countc[i];     //usa tempo salvo na função
copy
                        TA0CTL &= ~TAIFG;
                        i++;
                    }
                }
            }
    }
}
```