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 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.

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.

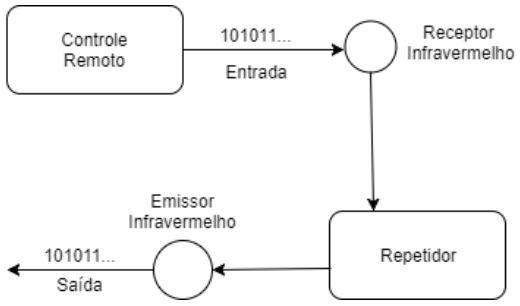


Figura 1 – esquema do projeto

Um controle remoto infravermelho envia dados por modulação ASK, ou seja, é definida uma frequência portadora que é reconhecida como uma informação pelo receptor. 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 períodos dependendo do modelo do controle remoto.

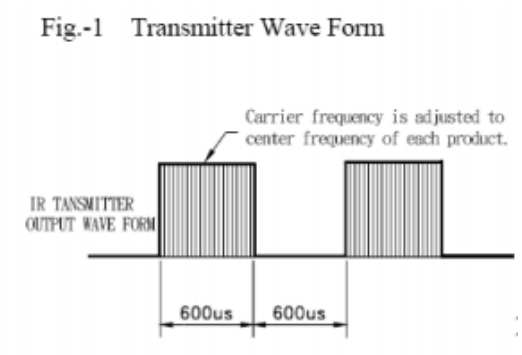


Figura 2 – exemplo de modulação ASK

Normalmente quando há emissão da frequência portadora, o receptor codifica o sinal para nível lógico alto, quando não há emissão (led apagado) o receptor codifica para nível lógico baixo. A modulação é utilizada pois há muita interferência luminosa no meio em que o aparelho está, o filtro no receptor só recebe sinais com a frequência portadora e corta outras frequências.

Em controles remotos comerciais comuns, temos uma codificação 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 não 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.

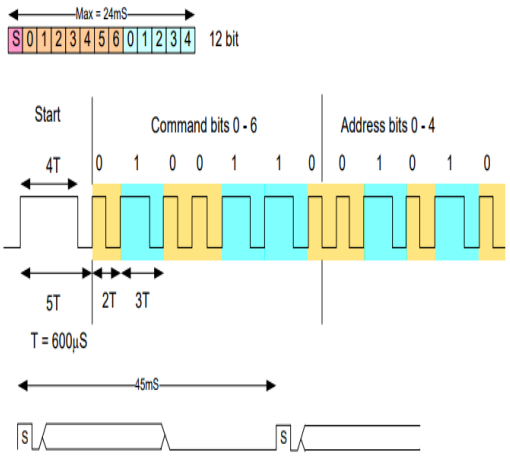


Figura 3 – emissão de dados para um controle comercial sony.

III. DESENVOLVIMENTO

Após a realização de testes para verificar os equipamentos que seriam utilizados no projeto (vide ANEXO III ou arquivo PC2), 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, é necessária uma inversão deste dado por circuito (porta NOT) ou por software.

A distância de captação de pulsos depende não apenas do receptor, mas também da intensidade da luz emitida, o LED infravermelho ligado diretamente a placa oferece uma emissão razoável, mas não satisfatória, por isso foi utilizado um amplificador simples com transistores para ligar o LED.

Figura 5 – LED infravermelho

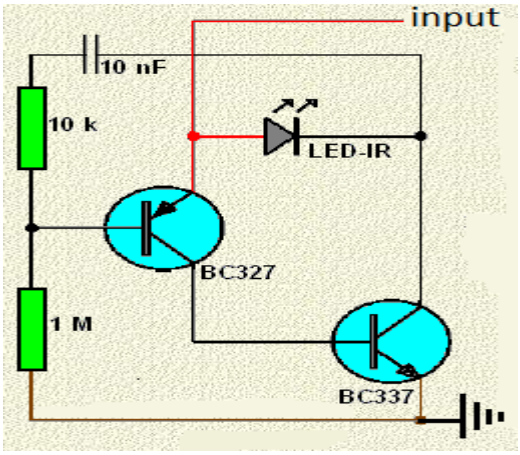


Figura 6 – amplificador para LED com transistores BJT.

O circuito provisório para recepção e emissão de um sinal pode ser observado a seguir:

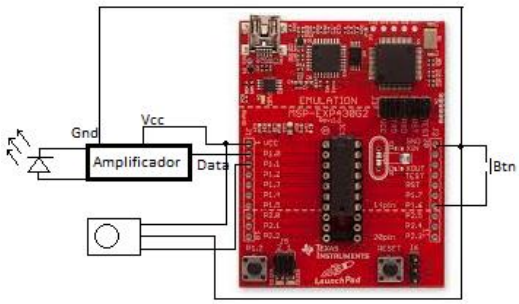


Figura 7 – protótipo do projeto

Após constatado o bom funcionamento dos periféricos, 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, o valor do nível lógico na entrada (receptor) também é salvo na posição de um vetor (nível[i]), após zerar a flag e o timer, é esperada outra interrupção e o processo se repete, salvando os valores em outras posições do 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]), o nível lógico para cada tempo é definido pelo vetor de nível lógico (nível[i]).

Para um melhor entendimento, observe o código em anexo (Anexo I).

Como foi dito na introdução, para o nível lógico alto ser reconhecido pelo receptor do equipamento, devemos mandar uma frequência de 36kHz a 40kHz durante o tempo definido, e não apenas acender o LED. Essa função ainda está sendo trabalhada.

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)

Figura8 – valores armazenados nos vetores.

Observando o vetor nível, observamos os valores de nível lógico invertidos (0x0000 = alto, 0x0008 = baixo), no vetor podemos observar quantos ciclos de clk (no caso clk = 1MHz) durou os pulsos codificados pelo receptor, o primeiro valor count[0] é descartado pois é um valor salvo

aleatoriamente do Timer A quando é gerada a primeira interrupção.

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.

Os valores dos pulsos de pré dado, dado, start e etc deveriam ser iguais em todas as situações em que são ativados, porém o código apresenta baixa precisão na análise dos pulsos devido ao clk 1MHz que não é adequado para a aplicação, o clk será aumentado.

V. REVISÃO BIBLIOGRÁFICA

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 I

Código para captura e emissão do código infravermelho

```
#include <msp430g2553.h>

#define RECEPTOR BIT1
#define LEDIR BIT0
#define SELCOPY BIT3    // botao da placa
#define SELSEND BIT4

volatile unsigned int i=0;
volatile unsigned int nivel[80];
volatile unsigned int count[80];

int main(void)
{

      WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

      BCSCTL1 = CALBC1_1MHZ;      // sets de MCLK e SMCLK 1MHz
      DCOCTL = CALDCO_1MHZ;

      P1OUT &= 0x0000;
      P1DIR &= ~(RECEPTOR + SELSEND + SELCOPY);        // entradas
    P1SEL |= RECEPTOR;         // pino P1.1 como entrada digital para
periférico, no caso a entrada CCIS do TA0CCTL0
      P1DIR |= LEDIR;              // pino P1.0 como saida
      P1REN |= RECEPTOR + SELCOPY + SELSEND;         // habilita resistores
      //P1OUT &= ~RECEPTOR;                  // resistor de pull-down (se
for usar com pulso invertido)
      P1OUT |= SELCOPY + SELSEND + RECEPTOR;                    // resistor
de pull-up

      wait();
}
void wait()
{
    while(1){
        P1OUT &= ~LEDIR;
        if((P1IN&SELCOPY)==0){      //função copiar

            copy();

        }

        else if((P1IN&SELSEND)==0){   //função enviar
            //P1OUT &= ~LEDIR;
            send();

        }
    }
}

void copy()
{

    TA0CTL = TASSEL_2 + ID_0 + 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
    i=0;

    while(1){

        if(TA0CCTL0&CCIFG){  //borda detectada ativa flag
            TA0R = 0x0000;                  //zera TA0R para iniciar nova
contagem
            nivel[i] |= (TA0CCTL0&CCI);     //salva niveis logicos num array
            count[i] = (TA0CCR0);           //salva contagens num array
            TA0CCTL0 &= ~CCIFG;             //zera flag
            TA0CCR0 = 0x0000;               //também zera TA0CCR0 p/ nova
contagem
            i++;
            if(i>79){        //limitando array
                wait();
            }
        }
    }
}

void send()
{

    TA0R = 0x0000;
    P1OUT &= ~LEDIR;       //garantir q LED infra comece em 0
    TA0CCR0 = 0x1000;      //TA0R conta até TA0CCR0 para gerar primeira
interrupção (valor pouco importa)
    TA0CTL = TASSEL_2 + ID_0 + MC_1;   // TA recebe SMCLK e /1, modo up até
TA0CCR0

    TA0CTL &= ~TAIFG;    //zera flag p/ garantir que nao haverá interrupção
antes do tempo
    i=0;

    while(1){

        if(TA0CTL&TAIFG){     //TA0R conta até TA0CCR0 e gera flag
            if(nivel[i]==0x0000){      //definindo nivel logico da saida,
                P1OUT |= LEDIR;        //este caso para nivel do receptor nao
invetido antes de ser recebido pela placa
            }                          // p/ receptor invertido, inverter
if's
            else{
                P1OUT &= ~LEDIR;
            }

            TA0CCR0 = count[i+1];      //começa pegar a partir do count[1]
pois count[0] é lixo;
            TA0CTL &= ~TAIFG;          //zera flag p/ nova interrupção
            i++;
            if(i>79){
                wait();
            }
```

```
        }
    }
}
```

ANEXO II

Informações do PC2

I. DESENVOLVIMENTO

Desta forma, pensou-se em uma maneira de diminuir o número de controles ou de garantir que determinado equipamento não fique inutilizável por causa de um controle. O grupo mantém como objetivo a criação de um controle remoto que não dependa de cadastro de códigos, bastando apontar um controle para ele e associando um botão a essa sequência enviada pelo controle a ser copiado.

Inicialmente foi feito um protótipo funcional do projeto para mostrar a comunicação dos sensores que serão utilizados.

Os materiais utilizados no protótipo foram:

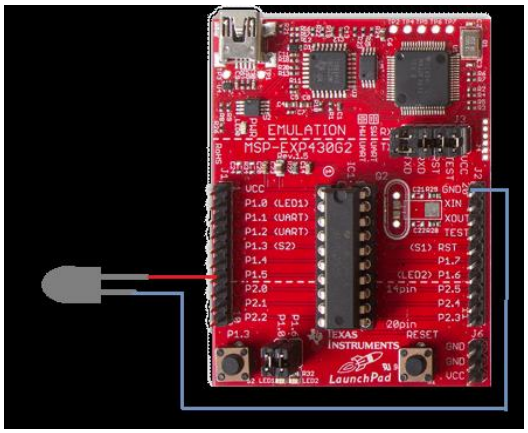
- Duas placas de desenvolvimento com microcontrolador MSP-430;

- Resistor 100Ω;

- LED infravermelho;

- Receptor infravermelho codificado;

Foi montado um pequeno esquema em uma protoboard com a saída P1.5 de um MSP alimentando o LED infravermelho, o outro MSP alimentou o receptor e recebeu como entrada seu pino de dados na porta P1.3

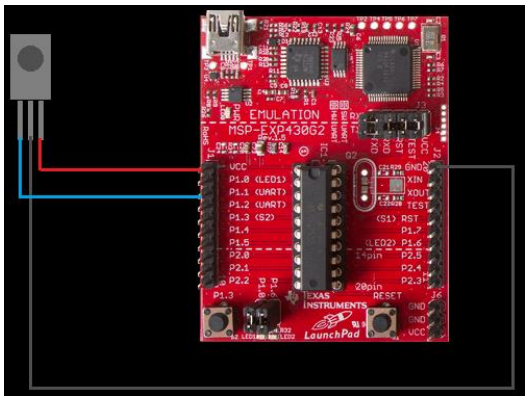


O programa implementado no MSP que controla o LED infravermelho utiliza a biblioteca msp430g2553.h de registradores e intrinsics.h para interrupções.

Primeiramente é desligado o Watchdog timer, após isso definimos os registradores que ativam os clock's de 1MHz, o pino desejado é definido como saída (P1.5). Registradores de comparação são inicializados para definir uma frequência específica para a saída P1.5.

É utilizada uma interrupção que inverte a saída toda vez que o ciclo detectado pelo comparador se completa.

Para o circuito receptor, foi feito o seguinte circuito:



A única entrada estabelecida foi a porta P1.1 para ler o dado captado pelo receptor. A saída do circuito foi a porta P1.0 que é acoplada com o LED vermelho da placa, dessa forma, quando o sensor detectava algum sinal IR, o LED vermelho era aceso.

II. RESULTADOS

Os circuitos propostos foram testados, começando pelo LED infravermelho, a funcionalidade foi conferida com uma câmera de celular, o LED pisca perfeitamente com a frequência definida. Após isso o funcionamento do receptor foi conferido com um controle remoto infravermelho de um equipamento qualquer.

O LED infravermelho foi apontado para o receptor e seus pulsos foram detectados pelo mesmo, porém com uma intensidade muito baixa (LED que indicava o recebimento do sinal acende com baixa intensidade). Após algumas conferências no circuito e alguns testes para averiguar a causa do problema, foi constatado que o LED infravermelho recebia pouca potência, o que resultava em uma emissão fraca de luz, refletindo em uma percepção também fraca pelo receptor. Este problema pode ser resolvido com um circuito amplificador de potência para o LED infravermelho.

ANEXO II

Partes editadas do PC1.

REQUISITOS

O sistema deve conseguir primeiramente receber um sinal infravermelho através do receptor codificado e armazená-lo em uma memória do microcontrolador MSP430 utilizando seus registradores e comparadores para identificar a sequência recebida o melhor possível. Após isso o código armazenado deverá ser disponibilizado em um botão ou comando e emitir esse sinal por um LED infravermelho a fim de controlar dispositivos sem mais precisar de seu controle remoto. O sistema controlará equipamentos com esse tipo de comando se o controle original do equipamento estiver disponível (ou qualquer outro controle que seja capaz de comandar o mesmo) para ser possível realizar a cópia do código corretamente. O sistema também só funcionará dentro do alcance de emissão e recepção dos equipamentos (LED infravermelho e receptor infravermelho) que ainda serão definidos.