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

# 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.

# 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.

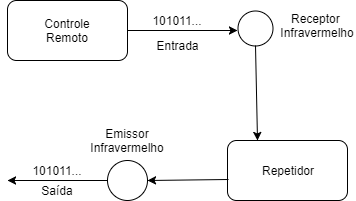


Figura 1 – esquema do projeto

Um controle remoto infravermelho envia dados por modulação em frequência, 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.

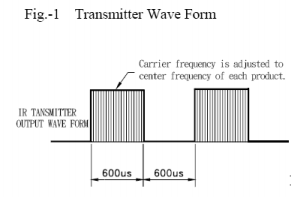


Figura2 – exemplo de modulação em frequência

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 em frequência 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 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 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.

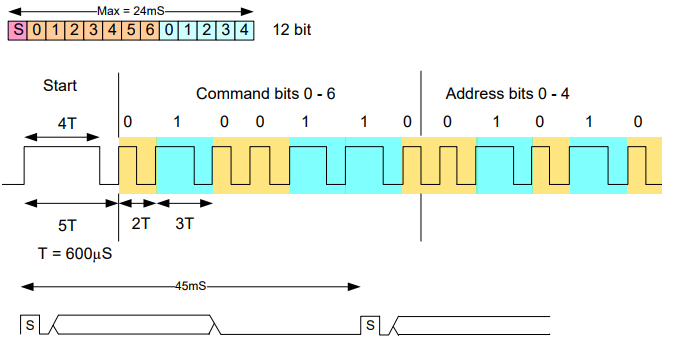


Figura3 – emissão de dados para um controle comercial sony.

# DESENVOLVIMENTO

Após a definição do modo de operação para a recepção e armazenamento do sinal, o foco foi voltado para a parte de emissão do sinal infravermelho, utilizando os valores armazenados na variável que copiou o sinal.

Como há a necessidade da utilização de uma frequência portadora, utilizamos outro chip msp430 para gerar essa frequência enquanto uma porta AND e o outro microcontrolador na placa controlam se a frequência passa ou não para a saída no LED.

Após testes utilizando a frequência portadora, foi observado que o LED perdia brilho quando a portadora era ativada, logo percebeu-se a incapacidade dos transistores utilizados de trabalhar em tal frequência, os transistores foram trocados por unidades com faixa de operação mais alta. A porta AND utilizada também foi feita com esses transistores, pois o circuito integrado disponível também não apresentou bom comportamento na frequência portadora.

O novo esquemático para a recepção e emissão do sinal infravermelho ficou da seguinte maneira:

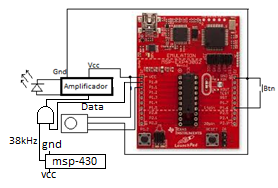


Figura4 – esquemático funcional do projeto

O código para envio dos dados para o LED infravermelho funciona de maneira semelhante a parte de recepção, desta vez utilizando o timer A em modo de comparação, que utiliza os valores salvos no vetor de dados (com os tempos entre as transições do código recebido) e inverte o nível da saída cada vez que o modo de comparação detecta que o timer atingiu o valor da posição do vetor. Esse bit em nível alto ou baixo define se a frequência portadora irá passar ou não pela AND e consequentemente ativar ou não os pulsos no LED (conferir código em Anexo I).

O erro de armazenamento de bits indesejados no vetor de dados que ocorria no envio de dados para o LED foi resolvido substituindo as funções externas utilizadas no código por funções mais simples, dentro da própria função *main*.

# RESULTADOS

Após copiar um código de um controle qualquer e tentar replicá-lo para que o aparelho pudesse reconhece-lo, os resultados foram negativos, porém esperados, pois os valores salvos no vetor de dados são bem próximos aos valores reais que o controle envia (valores conferidos no datasheet do fabricante sony), porém não são exatos, há uma pequena imprecisão quando o timer A detecta bordas de subida ou descida do controle original, definindo o tempo de detecção alguns ciclos de clock após o evento real. Este erro pode ser corrigido com um simples filtro de valores dentro do próprio código, pois os controles infravermelhos em geral seguem um padrão de tempos inteiros para definir seus dados, como o erro é pequeno, podemos aproximar o tempo detectado para esses tempos inteiros e assim conseguir reproduzir exatamente o código do controle original.

Outro fator observado foi o brilho do LED infravermelho utilizado, que não foi satisfatório quando comparado com o brilho do LED do controle remoto, mesmo com a amplificação. Estão sendo conferidas a exata tensão e corrente que o LED necessita para seu funcionamento em potência máxima.

# 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

**#define** ARRAY 80

**volatile** **unsigned** **int** i = 0;

**volatile** **unsigned** **int** j = 0;

**unsigned** **int** count[ARRAY];

**int** **main**(**void**)

{

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

BCSCTL1 = CALBC1\_1MHZ; // sets de MCLK e SMCLK XMHz

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

**while**(1){

P1OUT &= ~LEDIR;

**if**((P1IN&SELCOPY)==0){ //função copiar

TA0CTL = TACLR;

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

TA0CTL = TACLR;

TA0CTL = TASSEL\_2 + ID\_0 + MC\_2; // TA recebe SMCLK e /1, modo up até 0xFFFF

count[i] = TA0CCR0; //salva contagens num array

TA0CCTL0 &= ~CCIFG; //zera flag

TA0CCR0 = 0x0000; //também zera TA0CCR0 p/ nova contagem (desnecessário também)

i++;

**if**(i>ARRAY){ //limitando array

**break**;

}

}

}

}

**else** **if**((P1IN&SELSEND)==0){ //função enviar

TA0R = 0x0000;

P1OUT &= ~LEDIR;

TA0CCR0 = 0x100;

TA0CTL = TASSEL\_2 + ID\_0 + MC\_1;

TA0CTL &= ~TAIFG;

i=0;

j=1;

**while**(1){

**if**(TA0CTL&TAIFG){

**if**(i>(ARRAY-2)){

**break**;

}

**if**(j==1){

P1OUT |= LEDIR;

}**else**{

P1OUT &= ~LEDIR;

j=1;

}

TA0CCR0 = count[i+1]; //usa tempo salvo na função copy

TA0CTL &= ~TAIFG;

i++;

**if**(j==1){

j=0;

}

}

}

}

}

**return** 0;

}

ANEXO II

Informações do PC2

# 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:

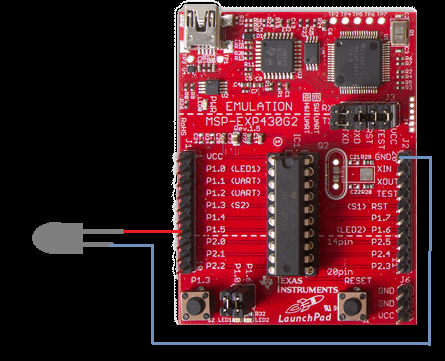
- Duas placas de desenvolvimento com microcontrolador MSP-430;

- 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

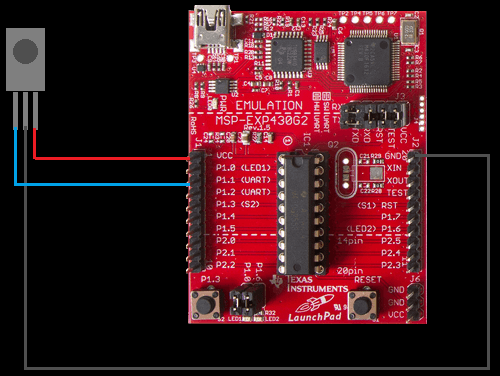


O programa implementado no MSP que controla o LED infravermelho utiliza a biblioteca msp430g2553.h de registradores e intrinsics.h para interrupções.

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.

# 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 III

Informações do PC3

# 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.



Figura4 – 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.



Figura5 – LED infravermelho

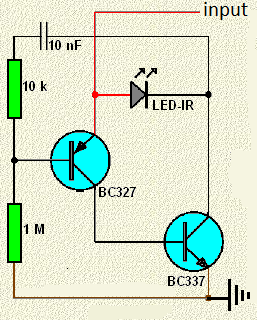


Figura6 – amplificador para LED com transistores BJT.

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

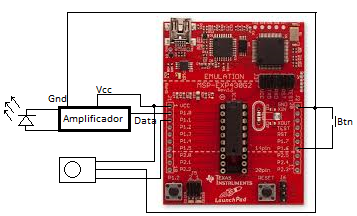


Figura7 – protótipo do projeto

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]).

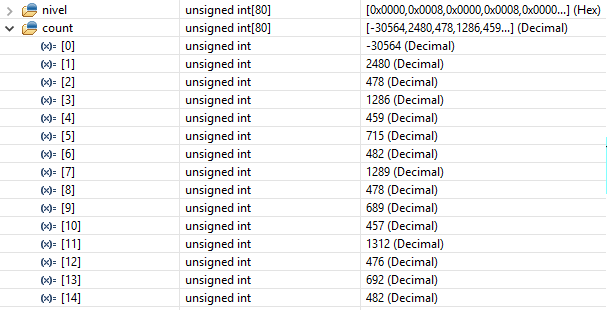
Para um melhor entendimento, observe o código em anexo (Anexo I).

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.

# 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]):



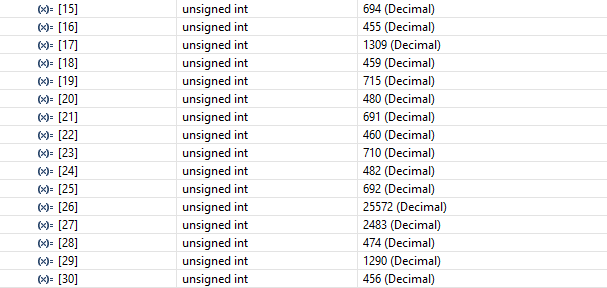


Figura8 – valores armazenados nos vetores.

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.

ANEXO IV

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.