Luiz Carlos Lanznaster

Temporizador Digital para Controle de Reuniões

Florianópolis, 2014

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE PÓS-GRADUAÇÃO (LATO SENSU) EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

LUIZ CARLOS LANZNASTER

TEMPORIZADOR DIGITAL PARA CONTROLE DE REUNIÕES

Monografia submetida ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Produtos Eletrônicos.

Orientador: Prof. Charles Borges de Lima, Dr., EE

Florianópolis, 2014

CDD 621.317
L297t

Lanznaster, Luiz Carlos
Temporizador digital para controle de reuniões [MP] / Luiz Carlos Lanznaster; orientação de Charles Borges de Lima. – Florianópolis, 2014.
1 v.: il.
Monografia de Pós Graduação (Desenvolvimento de Produtos Eletrônicos) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.
1. Controle remoto. 2. Microcontrolador. 3. Sensoriamento. 4. Cronômetro. 5. Conselhos Superiores da UDESC. I. Lima, Charles Borges de. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis

TEMPORIZADOR DIGITAL ESCALONÁVEL PARA CRONOMETRAGEM DE REUNIÕES

LUIZ CARLOS LANZNASTER

Este trabalho foi julgado adequado para obtenção do Título de Especialista e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação (*Lato Sensu*) em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 18 de setembro de 2014.

Banca Examinadora:

_____

Charles Borges de Lima, Dr., EE., Orientador

_____

Prof. Hugo Marcondes, M.Sc., CC

_____

Prof. Reginaldo Steinbach, M.Sc. Mec.

RESUMO

A ideia central deste trabalho é a construção e aplicação de um dispositivo para cronometragem e controle das rodadas de discussão em reuniões ordinárias e extraordinárias dos três Conselhos Superiores da Universidade do Estado de Santa Catarina – UDESC.
Este dispositivo ficará localizado em local adequado no plenário de reuniões, visível a todos os presentes e acionado por um controle remoto (CR) manual com função de avanço, pausa, reinício e alteração de tempo para discussão. A portabilidade do dispositivo é contemplada para permitir a mudança de locais de reunião.
Um microcontrolador comumente encontrado no mercado (ATmega328P da Atmel) é o controlador do dispositivo que, além da função principal como cronômetro, apresenta a temperatura do recinto, acrescido de um sensor de movimento para despertá-lo do *standby* e mais um alarme sonoro para indicar o término do intervalo de tempo de uma discussão.

ABSTRACT

The main idea of this work is the construction and implementation of a device for timing and discussion round control at regular and special meetings in three Supreme Councils of the Universidade do Estado de Santa Catarina - UDESC.
This device will be located in a suitable place in the plenary meeting, visible to all present and triggered by a manual remote control (RC) with function forward, pause, restart and change of time for discussion. The portability of the device is contemplated to allow the change of venues.
A microcontroller widely available on the market (ATmega328P Atmel) is the device driver that besides the main function as a stopwatch, it shows the room temperature, plus a motion sensor to wake it from standby and more an audible alarm to indicate the end of the time interval of a thread.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AC	-	*Alternating Current*
CC	-	Corrente Contínua
CI	-	Circuito Integrado
DC	-	*Direct Current*
EEPROM	-	*Electrically-Erasable Programmable Read-Only Memory*
ICSP	-	*In-Circuit Serial Programming*
IDE	-	*Integrated Development Environment*
I/O	-	*Input / Output*
IR	-	*InfraRed*
OEM	-	*Original Equipment Manufacturer*
PCM	-	*Pulse Code Modulation*
PCI	-	Placa de Circuito Impresso
PIR	-	*Passive InfraRed*
PWM	-	*Pulse Width Modulation*
RAM	-	*Random Access Memory*
RISC	-	*Reduced Instruction Set Computing*
SPI	-	*Serial Peripheral Interface*
SRAM	-	*Static Random Access Memory*
USB	-	*Universal Serial Bus*
VNC	-	*Virtual Network Computing*

SUMÁRIO

1 INTRODUÇÃO

Propõe-se, com este trabalho, a construção de um temporizador digital para controle de rodadas de discussão em reuniões dos Conselhos Superiores da Universidade do Estado de Santa Catarina – UDESC.

Para isto, há a necessidade de um temporizador que registre visualmente o intervalo de tempo restante de um orador (conselheiro) inscrito para uma rodada de manifestação nas discussões típicas em um determinado Conselho Superior.

A Universidade dispõe de um anfiteatro para as reuniões, em sua maioria, onde há uma pequena tribuna para manifestações. As discussões podem se estender por até três rodadas, sendo a primeira de 3 minutos, a segunda de 2 minutos e a final de 1 minuto.

Atualmente, este controle de tempo é realizado por um aplicativo shareware (Egg da ACAPsoft), cujo cronômetro é visualizado através de dois retroprojetores instalados às costas dos oradores e da mesa diretora. Este aplicativo está disponível em um computador e é manipulado por um servidor que auxilia o secretário do conselho.

O problema reside no fato que tanto os oradores, quanto os participantes da mesa diretora não visualizam o tempo decorrido mostrado às telas dos retroprojetores que se encontram atrás dos mesmos (ver detalhe da Fig. 1). Ademais, há necessidade de se manter um servidor em plenário com a função quase exclusiva para acionamento do controle de tempo por intermédio do aplicativo supracitado. Para amenizar o problema da visualização, utiliza-se um segundo computador instalado na mesa diretora com a função exclusiva de reproduzir a tela do primeiro (via VNC) para que, ao menos, os oradores, posicionados no púlpito, possam visualizar frontalmente o tempo decorrido. A Fig. 1 mostra como se dá esse processo.

Diante do problema exposto, a intenção deste trabalho é construir um cronômetro digital com três displays de 7 segmentos, com tamanho adequado e que, posicionado em local adequado, possa ser visível a todos os presentes à reunião. Um dígito será para os minutos e dois para os segundos. Um pequeno sinalizador acústico dará um sinal sonoro aos 30 segundos finais e ao se esgotar o tempo, sendo estas funções estas inexistentes atualmente.

Fig. 1 - Plenarinho da Reitoria da UDESC (antes)

Outra questão, será a eliminação do computador exclusivo para reproduzir da tela do retroprojetor.

O servidor auxiliar do secretário da mesa poderá se ausentar da plenária para cumprir outras funções, se necessário, pois o próprio secretário poderá fazer o acionamento do cronômetro, via controle remoto.

Portanto, esse cronômetro será acionado por um controle remoto simples que será programado com alguns procedimentos: inicialização, pausa, reinicialização e configuração dos tempos previstos para controle da oratória dos membros participantes das reuniões.

Incluiu-se neste trabalho, junto ao temporizador, um medidor da temperatura interna do anfiteatro, que será visualizada sempre que houver reuniões sem controle de tempo ou, havendo controle (como as dos Conselhos Superiores), será visualizado durante os intervalos mais longos entre as oratórias controladas.

Para automatizar o funcionamento do dispositivo, um sensor de movimento vai acioná-lo logo que uma pessoa adentrar ao plenário de reuniões. Caso contrário, o dispositivo permanecerá em estado de espera *(standby*).

Finalizando, esse dispositivo terá a incumbência de amenizar ou eliminar todas as questões inadequadas aqui levantadas em relação ao procedimento atualmente aplicado de cronometragem das reuniões supracitadas.

2 OBJETIVOS

2.1 Objetivo Geral

A partir de uma necessidade pontual, observada em um dos recintos da universidade, houve a pretensão de construir um aparelho (dispositivo) que registre visualmente o intervalo de tempo de manifestação de um orador, incluindo diversos controles temporais para pausa, retomada, reinicialização e alterações de tempo de acordo com a rodada específica em andamento.

Este dispositivo substituiria o procedimento atual, conforme supracitado na introdução deste trabalho, e seria montado com os componentes mais facilmente encontrados no mercado e que resultasse em um software de controle (*firmware*) de baixa complexidade.

Por fim, há a intenção de se construir de apenas um dispositivo para suprir a necessidade referida no início deste tópico.

2.1 Objetivos Específicos

Este tópico pode ser listado como segue:

- Desenvolver um circuito do temporizador (hardware e firmware), baseado em uma placa de circuito impresso, que programe todos os controles temporais referidos no objetivo geral.
- Implementar, junto ao temporizador, um medidor de temperatura.
- Acrescentar módulos para controle remoto (receptor de infravermelho) e para o sensor de movimento ao circuito acima descrito.
- Codificar em C o funcionamento do cronômetro e a integração entre o circuito principal e os módulos adicionados (firmware).
- Montar todo o sistema em uma caixa plástica de tamanho adequado (tipo painel) que comporte três displays de 7 segmentos, sensor de recepção de sinais de controle remoto, sensor de temperatura, sensor de movimento, sinalizador acústico e módulos acima referidos.

3 REVISÃO BIBLIOGRÁFICA

Breve descrição teórica sobre os temas necessários para a composição e entendimento deste trabalho e um resumo das características de funcionamento dos principais componentes constituintes deste projeto.

Todo o projeto pode ser visualizado através de um diagrama de blocos (Fig. 2) que dará uma ideia geral da constituição do dispositivo.

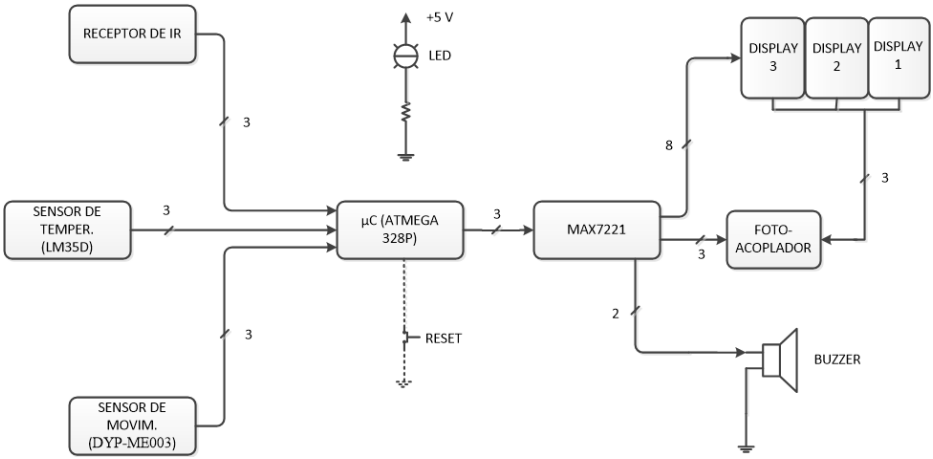

Fig. 2 - Diagrama de blocos deste projeto

3.2 Principais sistemas de transmissão/recepção para controles remotos

Em muitos sistemas de transmissão usados em controles remotos, apenas baixas taxas de dados são necessárias para transmitir funções de controle para equipamentos domésticos. A confiabilidade da transmissão é essencial, pois uma interpretação errada de um código transmitido não pode ser tolerada. Sinais corrompidos devem ser ignorados. Em muitos esquemas de codificação, comandos são repetidos até o dispositivo controlado reagir como desejado. O usuário pode observar sonora ou visualmente o resultado de uma tecla pressionada (VISHAY, 2013).

Como os sinais de IR estão restritos a uma sala ou cômodo e a transmissão dos dados tem curto período para cada tecla pressionada, não há problemas legais no uso da banda de frequência da portadora desses dispositivos, cuja faixa situa-se entre 30 e 56 kHz (VISHAY, 2013).

Um método de transmissão é o PCM (Modulação por Código de Pulsos), que transmite dados através de uma sucessão de bits seriais na forma de pulsos modulados por uma frequência portadora.

Há três representações mais comuns de transmissão de um bit em sistemas de controle remoto, que são:

a) ***Bi Phase Encoding***, que possui uma borda de subida ou descida no centro de cada intervalo de tempo (Fig. 3).

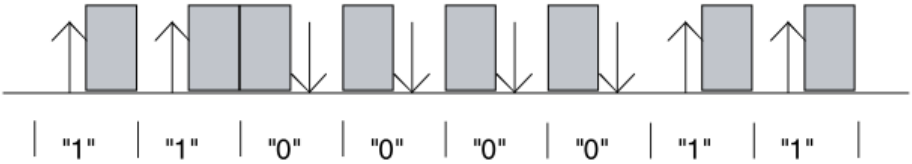

Fig. 3 - Representação do Bi Phase Encoding
Fonte: (VISHAY, 2013)

b) ***Pulse Distance Coding***, que possui todos os pulsos com o mesmo tamanho, mas com diferentes intervalos de tempo entre eles, dependendo do valor do bit (Fig. 4).

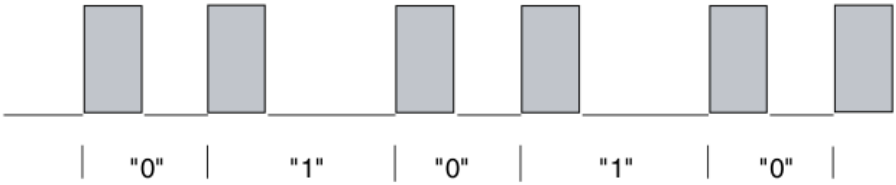

Fig. 4 - Representação do Pulse Distance Coding
Fonte: (VISHAY, 2013)

c) ***Pulse Length Code***, que possui dois pulsos com tamanhos diferentes, dependendo do valor do bit (Fig. 5).

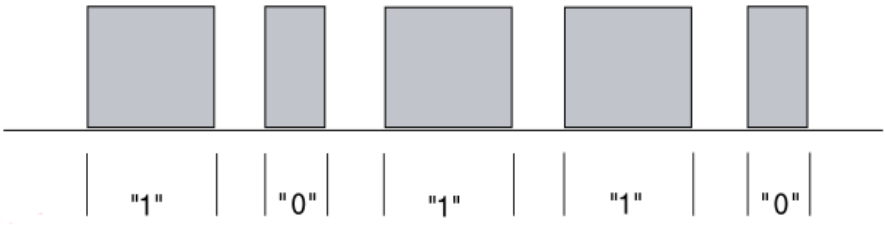

Fig. 5 - Representação do Pulse Length Code
Fonte: (VISHAY, 2013)

A frequência portadora de sistemas típicos de transmissão para controle remoto está padronizada para 30, 33, 36, 38, 40 e 56 kHz.

Além de diferentes tipos de codificação e diferentes frequências portadoras, há outras variações no formato dos dados: com ou sem um pulso identificador (*leading pulse*), com quantidade diferente de bits em um comando e com diferentes tamanhos de bits (VISHAY, 2013).

A maioria desses códigos tem bits de endereços e de dados. Para garantir uma transmissão segura, alguns códigos enviam dados duas vezes, sendo uma normal e outra invertida. Em geral, comandos de dados são repetidamente enviados, se uma tecla for mantida pressionada, sem soltá-la. Há diferentes modos para distinguir quando múltiplas teclas devem ser pressionadas para uma dada função. Outros códigos usam bit de troca (*toggle*) que muda de valor para cada tecla acionada. Há também códigos que enviam um pulso identificador no início ou no fim de cada tecla acionada. E, ainda, alguns códigos transmitem dados apenas uma vez por acionamento de uma tecla (VISHAY, 2013).

Um formato de dados comumente usado é o protocolo NEC.

3.3 Protocolo NEC

O código NEC usa *burst* de 38 kHz e tem uma propriedade paticular: tamanho (32 bits) e largura constantes dos dados (54 ms), aliado à representação *Pulse Distance Coding* (VASILIY, 2007).

Esse código inicia uma transmissão usando um pulso largo (pulso principal) com intervalo de 9 ms, seguido por uma pausa de 4,5 ms e logo em seguida os dados a serem transmitidos.

Os dados estão divididos em byte de endereço e de comando seguidos, respectivamente, por seus bytes invertidos. O byte de

endereço normalmente tem o propósito de identificar o dispositivo a ser controlado (VASILIY, 2007). Neste trabalho, este byte é irrelevante, pois apenas um dispositivo receptor é utilizado.

Cada bit lógico é representado por um pulso seguido de uma pausa (Fig. 6), sendo:

- 0 lógico: pulso de 562,5 µs, mais pausa de 562,5 µs, totalizando 1,125 ms.

- 1 lógico: pulso de 562,5 µs, mais pausa de 1,6875 ms, totalizando 2,25 ms.

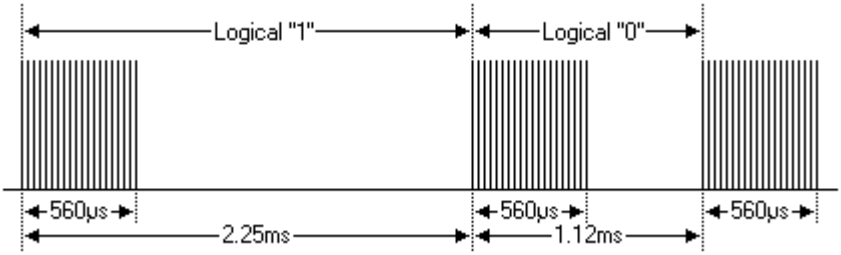

Fig. 6 - Bit 1 e bit 0 do protocolo NEC
Fonte: VASILIY

Cada pulso é formado por um *burst* de 38 kHz com *duty-cycle* de 1/3 ou 1/4. O bit menos significativo é transmitido primeiro.

Para encerrar a transmissão da mensagem correspondente a uma tecla acionada, um pulso final de 562,5 µs será gerado.

Para ilustrar, a Fig. 7 mostra um quadro completo de transmissão onde o endereço é 0b0000 0000 e o comando é 0b1010 1101 (SUNROM, 2011).

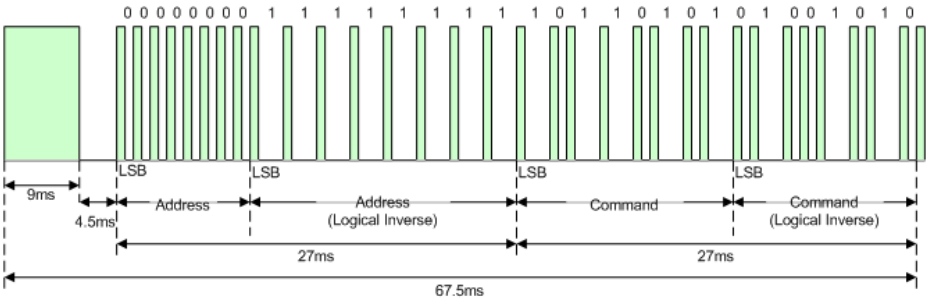

Fig. 7 – Pacote completo do protocolo NEC
Fonte: SUNROM, 2011

No caso de uma tecla ser mantida pressionada, um pulso principal de 9 ms, seguido de uma pausa 2,25 ms e finalizado por um pulso de 562,5 µs será repetido continuamente a cada 108 ms até a referida tecla ser liberada (Fig. 8).

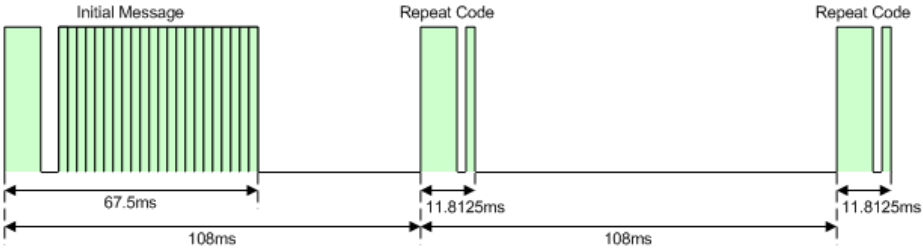

Fig. 8 - Pulso de repetição (tecla mantida pressionada)
Fonte: SUNROM, 2011

3.4 Fundamentos do Grafcet

O Grafcet é um método que permite representar, através de gráficos de estados, o comportamento (modelagem) de sistemas sequenciais.

De uma forma simples, o Grafcet (Fig. 9) consiste em:

- Elementos gráficos: **etapas** e t**ransições**.
- Elementos de controle: **ações** para as etapas e **receptividades** para as transições. Uma etapa é ligada sempre a uma transição e uma transição é sempre ligada a uma etapa, através de ligações orientadas (LEITÃO, 2004).

Etapas

O princípio fundamental do Grafcet é a redução de um processo de controle de sistemas sequenciais a etapas simples. Uma etapa corresponde a um estado do sistema, ou de parte dele, em que o comportamento permanece invariante. É representada por um quadrado com um número no interior.

Ações

A cada etapa estão associadas ações. As ações são executadas enquanto a etapa está ativa. Ações podem ser qualquer tipo de operação a ser executada por um sistema. Mesmo a etapa estando ativa, as ações podem estar condicionadas a algum outro evento.

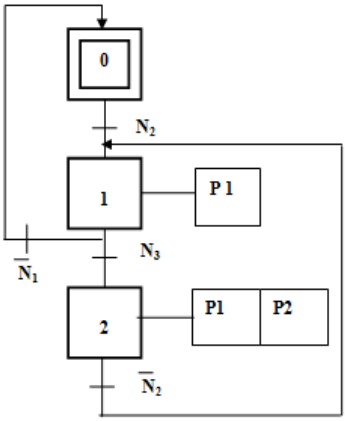

Fig. 9 - Exemplo de um Grafcet

Transições

As etapas do gráfico são separadas por transições. As transições representam a possibilidade de evolução entre etapas. A cada transição está associada uma condição lógica denominada **receptividade**.

A evolução no Grafcet para uma etapa seguinte requer que a etapa atual esteja ativa e a receptividade associada à transição seja verdadeira.

Princípio de funcionamento

Na definição do Grafcet existem várias regras que governam o seu comportamento:

- Inicialização: as etapas ativas, logo após a inicialização, são assinaladas duplicando o contorno dos quadrados (etapa 0 na Fig. 9).
- Validação: uma transição está validada se a etapa ou as etapas anteriores a ela estiverem ativas.
- Disparo de uma transição: uma transição é disparada se estiver validada (a etapa anterior está ativada) e a receptividade tornar-se verdadeira. O disparo de uma transição provoca a ativação das etapas de saída e a desativação das etapas de entrada (LEITÃO, 2004).

Razões para uso dessa metodologia

Esse método é útil para representar graficamente um algoritmo de controle sequencial de um processo. Com a intenção de substituir outros métodos como fluxograma ou máquina de estados, esta metodologia mostrou-se bastante intuitiva na definição da sequência

das operações necessárias para atingir ou alcançar o resultado esperado. A integração de seus elementos (etapas, ações e transições) para o desenvolvimento de um algoritmo facilita o arranjo das operações sequenciais em uma ordem conveniente para alcançar-se a solução de um problema.

A tradução de um Grafcet para alguma linguagem de programação é semelhante a outros métodos, porém, é facilitada pelos seus elementos gráficos:

- Etapas: em geral, conduz a um bloco de programação onde uma ou várias ações podem ser executadas.
- Ações: em geral, corresponde a um comando, simples ou complexo, com uma função específica.
- Transições: em geral, corresponde a estruturas condicionais.

3.5 Sensor de temperatura

O LM35D é um chip sensor calibrado que gera uma tensão analógica diretamente proporcional à temperatura em Celsius, com uma escala de saída de 10 mV por ºC.

Características principais:

a) Precisão aproximada de 0,5 ºC.
b) Sua faixa de temperatura situa-se entre 0 e 100 ºC.
c) Tensão de trabalho: 4 a 30 V.
d) Corrente drenada: abaixo de 60 µA.
e) Baixa impedância de saída: 0,1 Ω para 1mA de carga.
f) Encapsulamento metálico - TO-46, usado neste projeto.

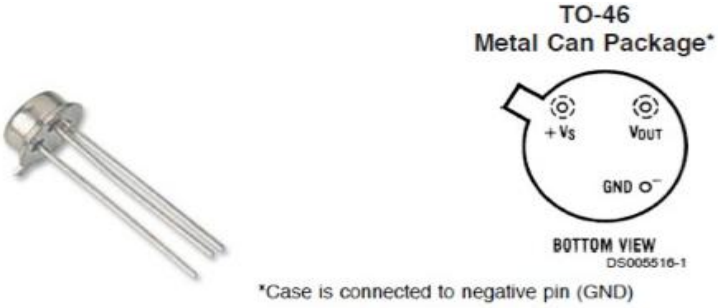

Fig. 10 - LM35D
Fonte: NATIONAL

3.6 Sensor de movimento

O módulo PIR (***P**assive **I**nfra**R**ed*) tem um sensor que detecta movimento nas proximidades por meio de uma lente de Fresnel – concentradora de radiação – acoplada a um elemento sensível a alterações nos padrões de radiação infravermelha. Este módulo é comercializado através do regime de comercialização denominado OEM.

Características principais:

a) Tensão de trabalho: 4,5 a 20 V.
b) Corrente drenada: < 50 μA.
c) Sensibilidade: 3até 7 m, ajustável, e com ângulo cônico até 100º.
d) Tensão de saída: 3,3 V (nível alto); 0 V (nível baixo).
e) Modo de disparo na saída: único ou contínuo. Neste trabalho será contínuo.
f) Tempo de atraso no disparo: de 0 a 200 segundos, ajustáveis.

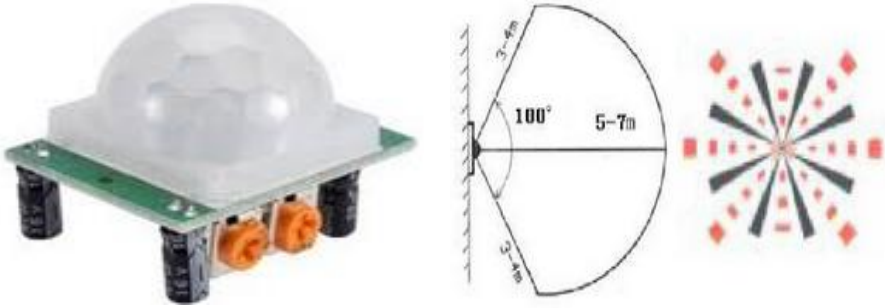

Fig. 11 - Módulo PIR com seu diagrama de alcance e sensibilidade
Fonte: ELECFREAKS

Neste trabalho, a sensibilidade foi ajustada para o máximo e o tempo de atraso no disparo para o mínimo.

3.7 Sensor de infravermelho e o controle remoto

É um receptor de sinais infravermelhos (fotodetector) montado em uma placa de circuito impresso (PCI) juntamente com capacitor de 47 μF e resistor de 100 Ω, estes com a função de filtrar pequenas variações na tensão da fonte de alimentação.

O controle remoto (CR) faz parte do conjunto de transmissão e recepção de sinais, adquiridos para este trabalho (Fig. 12).

Fig. 12 - Controle Remoto e seu módulo receptor

O minirreceptor, mostrado na Fig. 12, à direita, é um circuito integrado (HS0038B) com um diodo PIN na entrada seguido de um preamplificador mais um demodulador de sinais, inseridos em um invólucro de epóxi, este com a função de filtrar a radiação infravermelha. A Fig. 13 mostra o diagrama de blocos deste circuito (VISHAY – *datasheet*).

Características principais (receptor):

a) Filtro interno para frequência PCM.
b) Saída ativa baixa.
c) Alta imunidade contra luz ambiente.
d) Recepção contínua de dados em 800 bits por segundo.
e) Tensão de alimentação: 4,5 a 5,5 V.

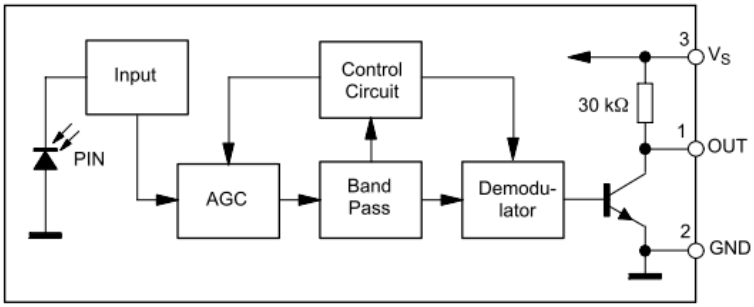

Fig. 13 - Diagrama de blocos do receptor de IR
Fonte: VISHAY

Características principais (CR):
a) Quantidade de teclas: 20.
b) Frequência de transmissão: 38 kHz.
c) Alcance: até 8 metros;
d) Ângulo efetivo de irradiação: até 60°.

4 DESENVOLVIMENTO DO DISPOSITIVO

4.1 Projeto e metodologia

A partir da menção das especificações básicas para o desenvolvimento do projeto, descritas na introdução e objetivos do presente trabalho, é possível resumi-las, como segue:

(a) definição de escala em minutos e segundos,

(b) controle remoto para acionar algumas funções de temporização,

(c) medidor básico de temperatura do recinto em ºC,

(d) sensor de presença para ligar o dispositivo e mostrar a temperatura,

(e) sinal sonoro para indicar tempo esgotado e tempo de 30 segundos para o término.

(f) função de *standby* para o dispositivo, quando o recinto não estiver ocupado, permite-se o levantamento dos principais componentes eletrônicos deste dispositivo:

- Microcontrolador Atmel AVR de 8 bits, por fazer parte da placa *open source* Arduino, que facilitará a simulação e gravação do *firmware*.
- MAX7221, *driver* para display de leds de 7 segmentos, pela economia de componentes obtida através de multiplexação interna, somada à facilidade de programação através da interface serial SPI.
- LM35, sensor de temperatura, amplamente conhecido para esta função.
- DYP-ME003, módulo sensor infravermelho para detecção de movimento de pessoas, com bom ângulo de sensibilidade e alcance de alguns metros.
- Módulo com circuito integrado HS0038B, sensor infravermelho para sinais de controle remoto, largamente comercializado, em conjunto com controle remoto manual.

Duas suítes de programas são a base para demonstração do *firmware* que controla todo o sistema projetado:

- Atmel Studio (Atmel Corporation) que, essencialmente, edita, compila e grava no microcontrolador o código fonte criado.
- Proteus (Labcenter Electronics) que promove a simulação de funcionamento dos circuitos projetados, incluindo chips programáveis, tais como os microcontroladores.

Como fase inicial, montou-se o circuito com os componentes acima relacionados com o uso de um software de simulação (Proteus/Isis). Nessa fase, elaborou-se um circuito básico para simulação e teste do funcionamento da função principal do dispositivo (cronometragem) e, numa fase posterior, fez-se o mesmo com o circuito completo no qual todos os módulos e componentes estão representados.

Após cada fase, foi elaborado e estruturado um fluxograma, através do Grafcet, de todas as operações necessárias e a sequência de encadeamento dos processos envolvidos na programação do firmware do cronômetro integrado aos módulos adicionados. Como cada um dos elementos gráficos do Grafcet possui identificações próprias, estas foram acrescentadas nos comentários do firmware (em C), nos blocos que indicam onde cada daqueles elementos estão implementados e codificados.

Na próxima fase, baseando-se no Grafcet gerado, foi programado o *firmware* por meio da suíte Atmel Studio que, através de sua IDE, proporcionou a depuração e a compilação dos arquivos fontes. Com uma pequena configuração desta suíte, foi possível gravar o microcontrolador utilizando-se de uma placa Arduino (UNO) de modo direto.

Dispondo-se do circuito e do *firmware* compilado, entrou-se na fase de visualização, testes e correções necessárias do projeto até a obtenção dos objetivos desejados.

Após, o protótipo foi montado agregando os módulos ao circuito do cronômetro e usando o Arduino Uno para a gravação do *firmware* no microcontrolador.

Tendo obtido um funcionamento adequado, foi realizada a confecção da placa de circuito impresso principal a qual, posteriormente, foi acondicionada em uma caixa plástica (painel) juntamente com os componentes externos.

Com essas etapas completas, foram realizados testes, ajustes e correções todos os problemas de hardware ou software que eventualmente apareceram.

4.2 Características básicas dos principais componentes do projeto

4.2.1 ATmega328P

Microcontrolador com arquitetura RISC de 8 bits baseado no núcleo AVR do fabricante Atmel Corporation. É a unidade de programação central deste projeto.

Características principais:

a) Memória Flash: 32 kB.
b) Memória EEPROM: 1 kB.
c) Memória SRAM: 2 kB.
d) Frequência: até 20 MHz.
e) Registros de propósito geral: 32.

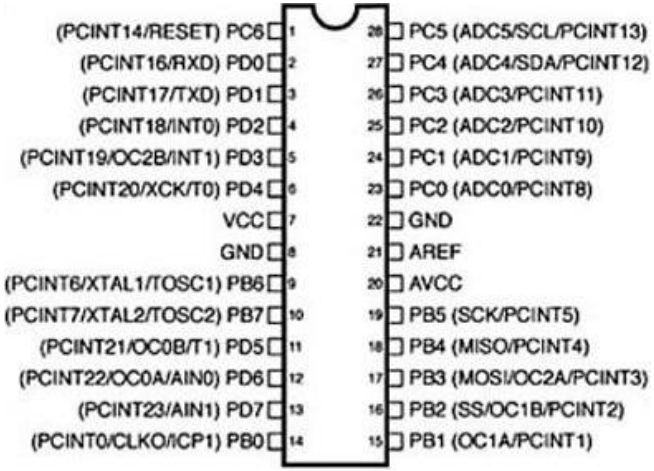

Fig. 14 - ATmega328P
Fonte: ATMEL

Este circuito integrado foi escolhido por ser facilmente encontrado no comércio local.

4.2.2 MAX7221

Trata-se de um circuito integrado (CI) com a função de *driver* para displays de 7 segmentos de catodo comum ou matriz de leds: ideal para

o emprego da multiplexação temporal baseada na persistência da retina e como interface entre microcontroladores e aqueles dispositivos.

Assim, pode-se conectar este CI ao ATmega usando o protocolo SPI com possibilidade de controlar até 8 displays ou 64 leds individuais. Contém um decodificador interno que pode converter números binários para a lógica de configuração dos displays de 7 segmentos. Portanto, simplesmente envia-se o código binário ao CI, que o decodifica, e apresenta-o nos displays (Fig. 15). O pacote de dados tem 2 bytes, sendo um byte de comando e um de dados propriamente dito (MAXIM – *datasheet*).

Características principais:

a) Controle analógico e digital do brilho.

b) Memória RAM estática 8x8 que armazena cada dígito.

c) Interface serial SPI com 10 MHz.

d) Requer apenas um componente externo: resistor para controle analógico de brilho.

e) Permite endereçar e atualizar dígitos individuais.

f) Registrador de escaneamento para o uso de 1 até 8 displays.

g) Corrente de *standby* limitado a 150 µA.

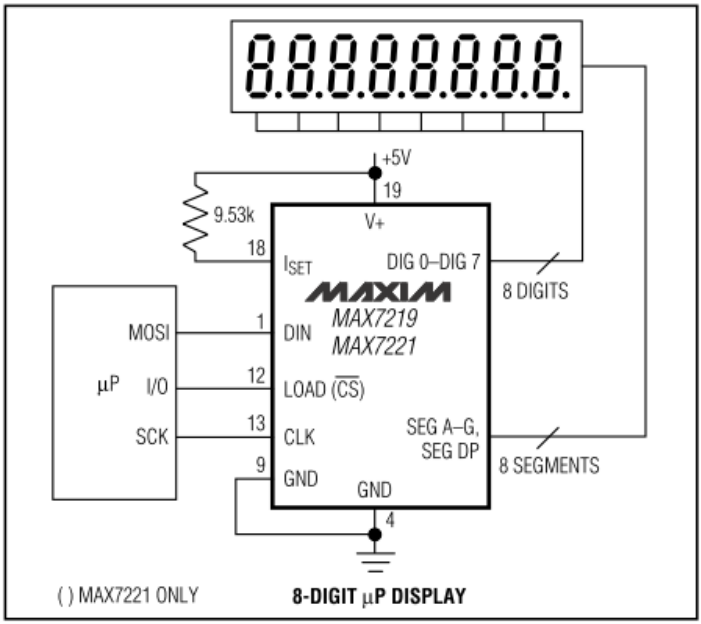

Fig. 15 - Uso típico do MAX7221
Fonte: MAXIM

4.2.3 TC962

Trata-se de um circuito integrado com a função de conversor de tensão CC-CC, tipo bomba de carga, comumente dito “conversor CC com capacitor chaveado”. Tem a vantagem de empregar apenas dois capacitores externos, nenhum diodo ou indutor e, deste modo, seu circuito torna-se muito simples (MICROCHIP – *datasheet*).

Neste trabalho, tem função a função de gerar uma fonte negativa simétrica à fonte principal (5 V), pois cada segmento do display de 7 segmentos, composto por 4 leds em série, exige pelo menos 8 V de tensão direta para seu acionamento.

Características principais:

a) Tensão de operação: 3 a 18V.
b) Corrente de saída: 80 mA.
c) Impedância de saída: 28 Ω, típico.
d) Frequência do oscilador interno: 12 kHz, podendo ir a 24 kHz.
e) Eficiência de conversão de energia: 97% típico, carga de 2 kΩ.

4.2.4 Display 7 segmentos de grandes dimensões

São displays com dimensões maiores em que cada segmento é formado por dois ou mais leds. Neste trabalho, o tamanho do display é 2,3 polegadas, os segmentos possui quatro leds e o ponto decimal, dois leds, interligados no formato catodo comum.

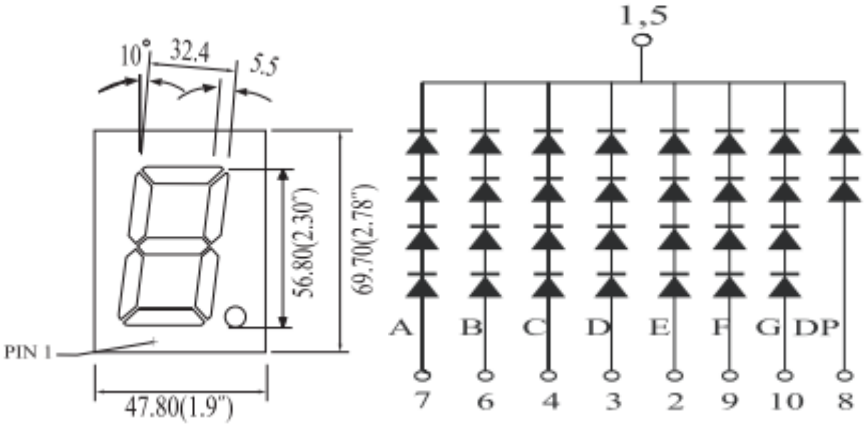

Fig. 16 - Displays de 2,3 polegadas
Fonte: XINDALI

4.2.5 Fonte chaveada de alimentação

Adaptador padrão de energia, facilmente encontrado no comércio, para uso nos mais diversos equipamentos portáteis.

Fig. 17 - Fonte de alimentação

Características principais:
a) Tensão de entrada AC: 100-240 V, 50 / 60 Hz.
b) Tensão de saída DC: 5 V.
c) Corrente máxima de saída:1 A.
d) Dimensão do plugue: P4 – 2,5 mm.

4.2.6 Arduino UNO

O Arduino Uno é uma placa microcontrolada baseada no ATmega328P e que faz parte de uma ampla plataforma de prototipagem eletrônica de hardware livre.

Ele tem 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de 16MHz, uma conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de reset.

Tem por objetivo permitir o desenvolvimento de controle de sistemas interativos, de baixo custo e acessível a todos os interessados em desenvolver protótipos eletrônicos, sem exigir apurado conhecimento de eletrônica ou programação.

Esta placa teve a importante função de gravação do microcontrolador utilizado neste projeto (ATmega328P) na fase de depuração e teste do código de programação.

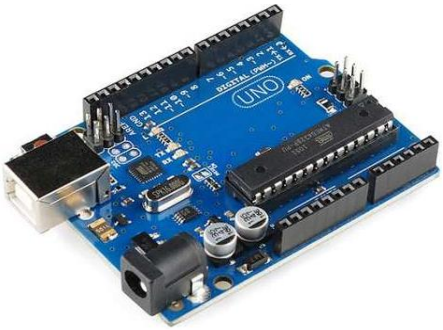

Fig. 18 - Placa do Arduino UNO
Fonte: ARDUINO

Características principais:
a) Tensão de alimentação: 5 V.
b) Tensão de entrada recomendado: 7-12 V.
c) Tensão de entrada min. / máx.: 6-20 V.
d) Corrente DC por pino: 40 mA.
e) Corrente DC para pino de 3,3 V: 50 mA.

4.2.7 Sinalizador acústico

Dispositivo piezoelétrico de sinalização por áudio com usos típicos em campainhas e apitos, atuando em dispositivos de alarme, temporizadores e confirmação da entrada do usuário.

Fig. 19 - Sinalizador acústico

4.3 Desenvolvimento do *firmware*

O Atmel Studio é uma plataforma integrada para desenvolver e depurar aplicações baseadas em microcontroladores Atmel AVR e ARM Cortex-M.

Neste trabalho, após configurá-la de modo apropriado, esta suíte de programas foi essencialmente usada para escrever o código fonte, compilá-lo e gravá-lo no microcontrolador, conforme já mencionado.

Para a gravação do microcontrolador, é empregado um utilitário independente denominado AVRdude que tem a função de carregar, deletar e manipular o conteúdo da memória dos AVR usando a técnica de programação *in-system* (ISP). Com o auxílio do AVRdude e através do programa Atmel Studio, que permite o uso de ferramentas externas para diversos propósitos, foi possível a configuração apropriada do referido programa para gravar o microcontrolador através de uma placa do Arduino UNO, introduzindo-se os seguintes argumentos:

-P com3 -b 19200 -c avrisp -p m328p -U flash:w:arquivo.hex -U lfuse:w:0xe2:m

A Fig. 20 mostra como isto foi feito na prática.

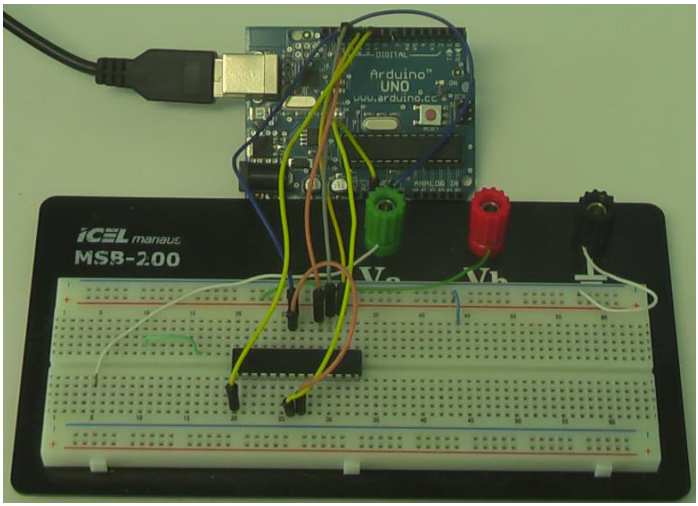

Fig. 20- Procedimento de gravação do microcontrolador

4.4 Circuito básico

O circuito base para as demonstrações iniciais das temporizações necessárias ao projeto (cronometragem) compõe-se do microcontrolador, MAX7221, display e uma chave para simular uma tecla do controle remoto (Fig. 21). Desta forma, através desse circuito foi possível desenvolver os códigos das operações do cronômetro, da conversão dos dados no MAX7221 e do modo de interrupção no microcontrolador ao simular a entrada do sinal do receptor de infravermelho por intermédio de uma chave táctil, ou seja, testar e garantir o funcionamento das operações principais do projeto.

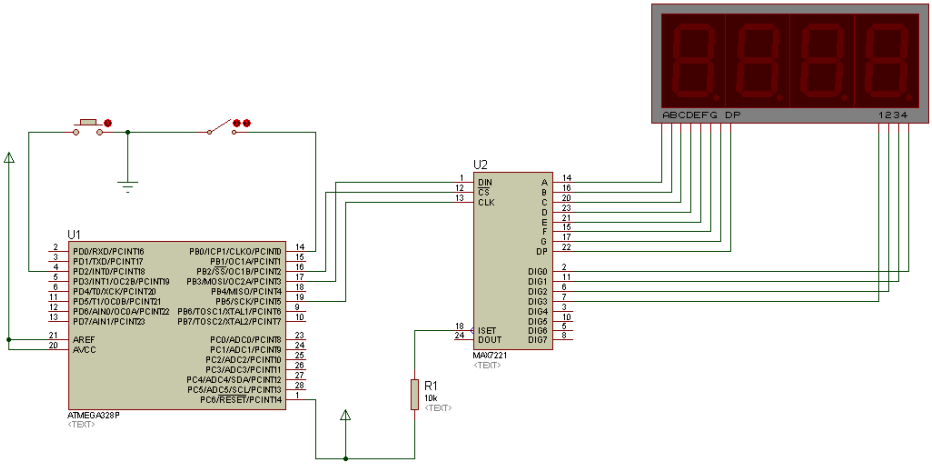

Fig. 21 - Circuito inicial do projeto

O protocolo para transmissão de dados entre o microcontrolador e o MAX7221 é o SPI, que é um padrão de comunicação de dados serial e é baseado em um protocolo síncrono muito simples (LIMA, 2012). Consiste basicamente na interligação de dois registradores de deslocamento, um no lado mestre e outro no lado escravo. Um gerador de *clock* no lado mestre sincroniza o deslocamento dos bits. Como se pode ver na Fig. 22, o pino de saída serial do registrador do mestre é conectado ao pino de entrada do registrador escravo pelo pino MOSI e, vice-versa, pelo pino MISO (MAZIDI, 2011).

No SPI, os registradores de deslocamento são de 8 bits. Isto significa que após 8 pulsos de *clock*, o conteúdo dos dois registradores

é trocado. Como os dados são enviados e recebidos ao mesmo tempo, o SPI é *full duplex*.

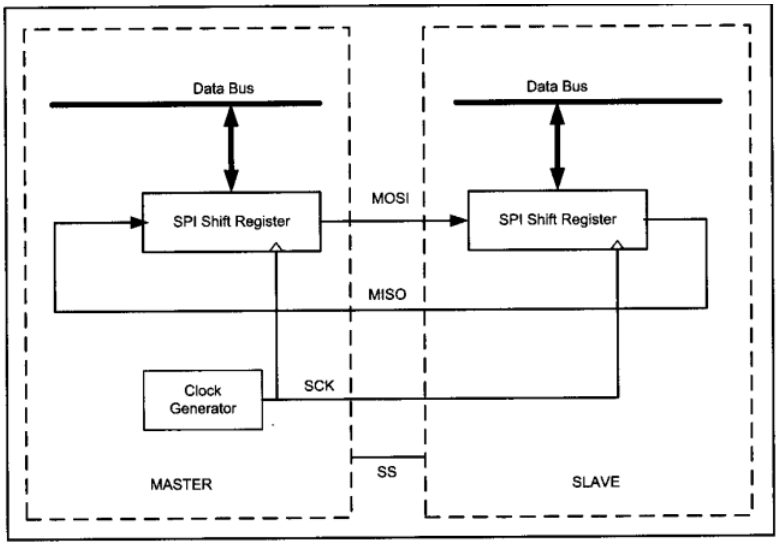

Fig. 22 - Esquemático do protocolo SPI
Fonte: (MAZIDI, 2011)

O microcontrolador envia dados ao MAX7221 por intermédio de três linhas de sinais:

- **CLK**, entrada serial do *clock* conectada ao pino SCK do microcontrolador.
- **DIN**, entrada serial de dados conectada ao pino MOSI do microcontrolador. Na descida do sinal de *clock* (CLK), dados destes pinos são carregados a um registrador de deslocamento interno. O MAX7221 usa o SPI em modo 0, ou seja, lê na borda de subida e altera na borda de descida.
- $\overline{\mathbf{CS}}$, entrada de seleção do chip conectada ao pino $\overline{SS}$ do microcontrolador (MAZIDI, 2011).

No MAX7221, os dados são recebidos em pacotes de 16 bits apenas quando o pino $\overline{CS}$ estiver em nível baixo. O primeiro byte deste pacote contém os bits de comando e o segundo, os dados a serem apresentados (Fig. 23).

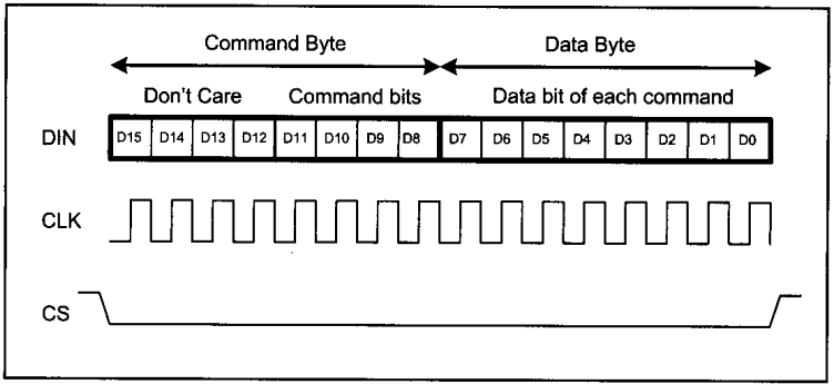

Fig. 23 - Formato do pacote de dados do MAX7221
Fonte: (MAZIDI, 2011)

Os bits D15-D12 não são usados e os bits D11-D8 identificam os diversos tipos de operações ou de comandos que serão realizados sobre o byte de dados (D7-D0), como configurações do chip e o envio dos dados a serem mostrados nos displays. Esses comandos ou operações estão resumidos na Tab. 1.

Tab. 1 – Relação de comandos do MAX7221
Fonte: (MAZIDI, 2011)

Registrador	**Endereço**					**Código Hexad.**
	D15-D12	**D11**	**D10**	**D9**	**D8**	
Desligado	x	0	0	0	0	0xX0
Dígito 0	x	0	0	0	1	0xX1
Dígito 1	x	0	0	1	0	0xX2
Dígito 2	x	0	0	1	1	0xX3
Dígito 3	x	0	1	0	0	0xX4
Dígito 4	x	0	1	0	1	0xX5
Dígito 5	x	0	1	1	0	0xX6
Dígito 6	x	0	1	1	1	0xX7
Dígito 7	x	1	0	0	0	0xX8
Modo decod.	x	1	0	0	1	0xX9
Intensidade	x	1	0	1	0	0xXA
Limite *scan*	x	1	0	1	1	0xXB
Standby	x	1	1	0	0	0xXC
Teste display	x	1	1	1	1	0xXF

A sequência das ações ou procedimentos necessários para se atingir o funcionamento satisfatório do circuito acima pode ser visualizado através do Grafcet (Fig. 24):

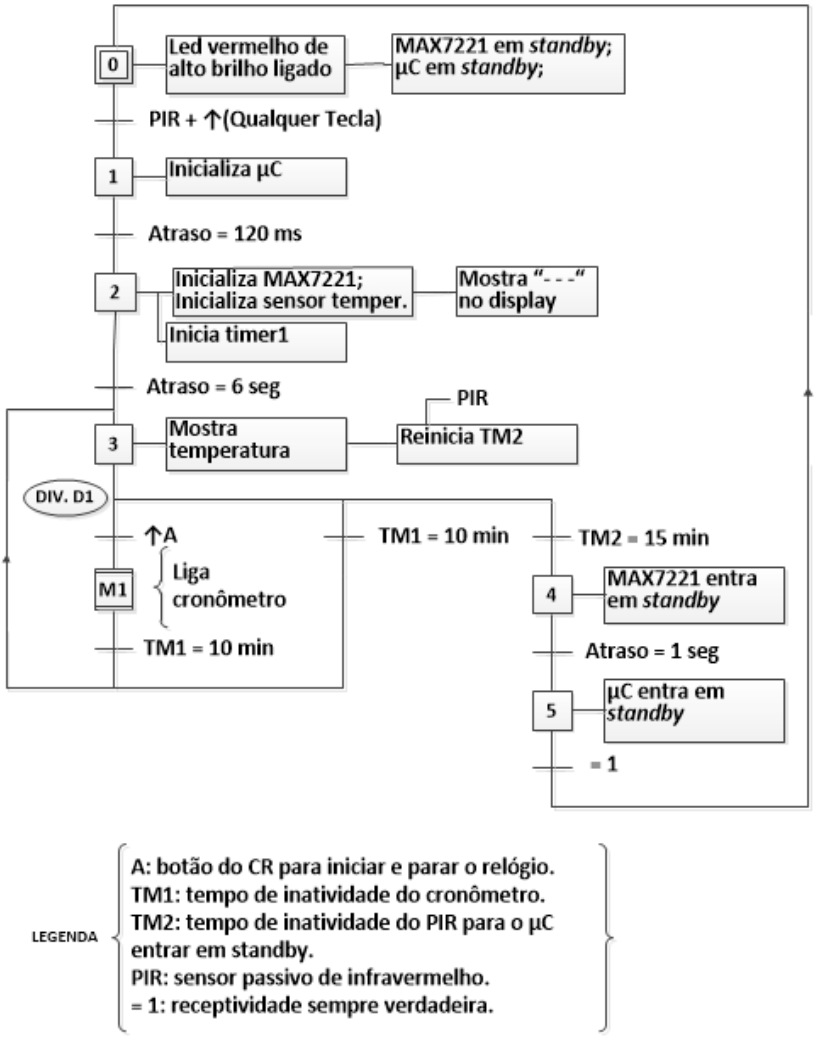

Fig. 24 - Grafcet das operações principais do projeto

Em uma fase posterior, para simular os códigos transmitidos por um controle remoto real, baseado no protocolo NEC descrito no item

3.3, um microcontrolador extra foi acrescentado e programado para esse propósito – emula um CR – e mais chaves que correspondam às teclas A, B e C (Fig. 25).

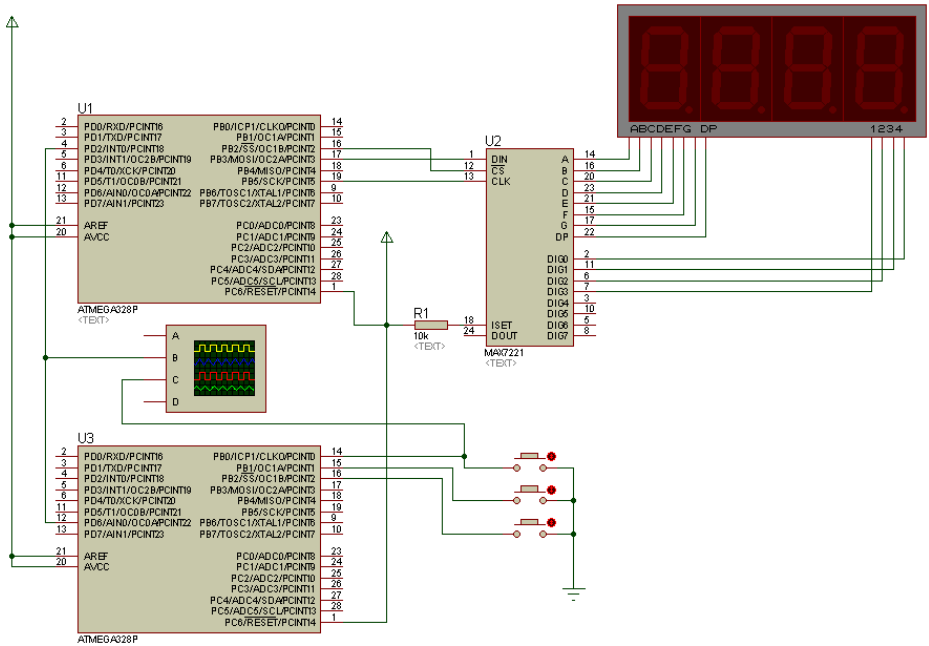

Fig. 25 - Circuito inicial com simulação do controle remoto

A sequência das ações ou procedimentos necessários para o funcionamento satisfatório do circuito da Fig. 25 pode ser visualizado através do Grafcet (Fig. 26).

Para entender as operações mostradas, é necessário o significado das teclas do CR nominadas com A, B e C. O controle remoto (Fig. 12) possui 20 teclas dispostas em três colunas. Isto posto, a tecla A corresponde à primeira coluna, a B, à segunda e a C, à terceira. Para este trabalho, apenas as duas primeiras linhas de teclas foram programadas para acionarem suas respectivas funções, em pares.

Então, resumindo as operações do Grafcet da Fig. 26:

- Display mostra, inicialmente, o tempo de 3 minutos, visualizado como “3.00”.

- Pressionando a tecla A do CR, inicia-se a cronometragem decrescente do tempo. Por outro lado, pressionando a tecla C do CR, altera-se o intervalo de tempo das rodadas de discussões.

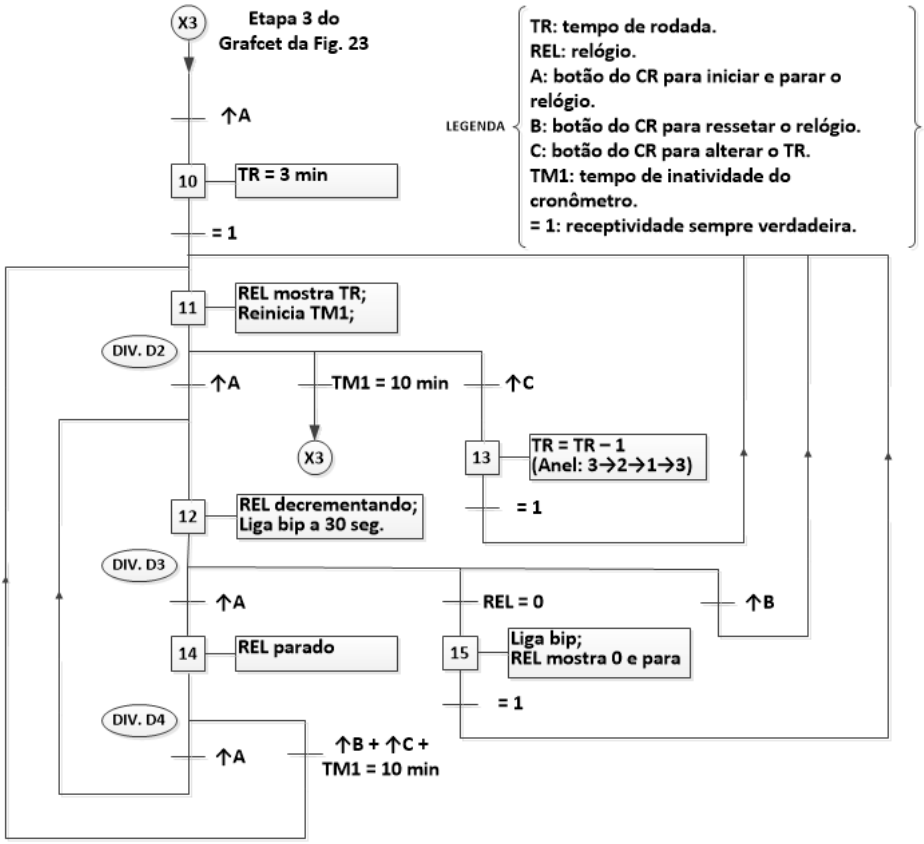

Fig. 26 - Grafcet para operações do cronômetro

- Pressionando a tecla A durante o decremento do tempo, ocorre uma pausa na contagem do tempo que permanece até se pressionar novamente a tecla A. Caso se utilize a tecla B ou C ou se atinge o tempo limite de 10 minutos, retorna-se ao estágio inicial.
- Quando o cronômetro finaliza o tempo (atinge “0.00”), permanece neste estado por alguns instantes e liga um alarme sonoro. Após isto, retorna-se ao estágio inicial.

Ademais, a sequência das ações ou procedimentos necessários para se atingir o funcionamento satisfatório da recepção e decodificação dos sinais transmitidos por um controle remoto, baseado no protocolo NEC, podem ser visualizadas através do Grafcet (Fig. 27).

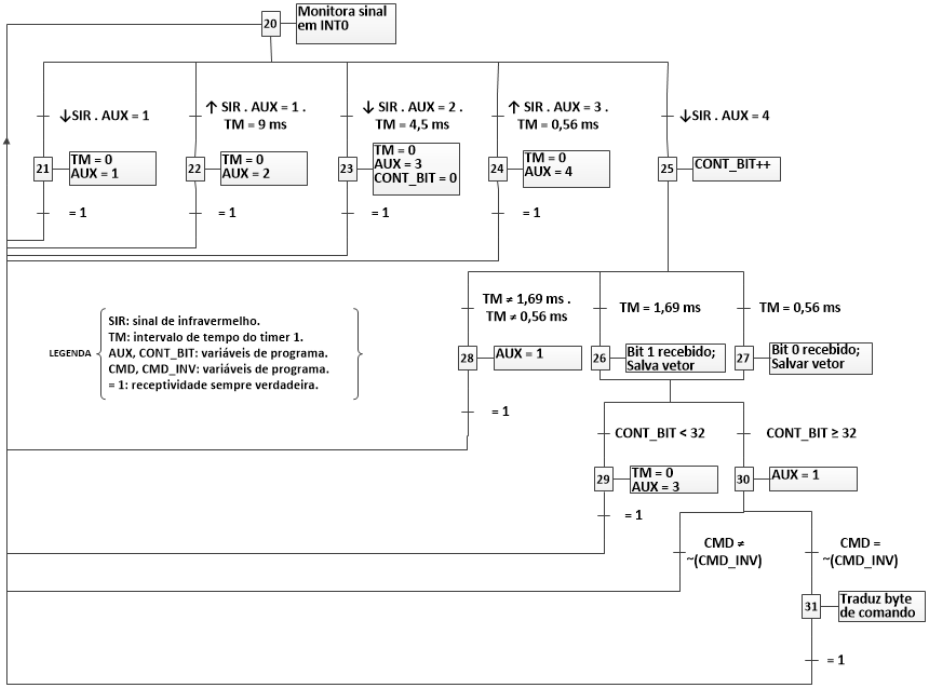

Fig. 27 - Grafcet para interpretação do protocolo NEC

Resumindo estas ações descritas na Fig. 27:

- Sinais provenientes do módulo receptor de IR são captados pelo pino 4 do ATmega 328, que está configurado como entrada da interrupção externa INT0 do microcontrolador.
- Caso o sinal, normalmente com lógica 1, for a 0, liga-se o timer1 e inicia-se o processo de reconhecimento da validade do mesmo.
- Caso o sinal, estando em 0 no passo anterior, vá a 1 e o timer1 atingir aproximadamente 9 ms, indica-se que o sinal continua válido.
- Caso o sinal, estando em 1 no passo anterior, vá a 0 e o timer1 atingir aproximadamente 4,5 ms, indica-se que o sinal continua válido.

- Caso o sinal, estando em 0 no passo anterior, vá a 1 e o timer1 atingir aproximadamente 0,56 ms, indica-se que o sinal continua válido.
- Caso o sinal, estando em 1 no passo anterior, vá a 0, inicia-se a contagem dos bits transmitidos (endereço e comando). Se o timer1 atingir aproximadamente 1,69 ms um bit 1 é recebido e salvo ou, se o timer1 alcançar aproximadamente 0,56 ms, um bit 0 é recebido e salvo. Caso nenhum destes intervalos de tempo seja medido, o sinal é inválido.
- Se a contagem dos bits chegar a 32, compara-se o byte de comando com o byte de comando inverso e, havendo correspondência, finalmente estabelece-se o referido byte com a apropriada tecla do CR apertada.

O código em C para implementar as operações acima descritas está disponível no Anexo A.

4.5 Circuito completo

O circuito final com todos os componentes eletrônicos e módulos (ou sua representação) agregados, necessários para cumprir os requisitos do projeto, está representado na Fig. 28.

Pelo diagrama de blocos (Fig. 2) verifica-se que o microcontrolador trabalha com os sinais provenientes de três módulos sensores:

- Receptor de IR, que decodifica os sinais provenientes do controle remoto,
- Sensor de temperatura, cuja tensão de saída é proporcional à temperatura ambiente,
- Sensor de movimentos, cuja saída digital indica se há movimentação de corpos quentes por meio de sua sensibilidade à variação da radiação infravermelha.

A saída do MAX7221 está dividida em dois grupos:

- Oito pinos que selecionam e estão diretamente ligados a cada segmento de um display de 7 segmentos (A, B, C, D, E, F, G e DP).
- Oito pinos que selecionam e podem estar diretamente ligados a cada display de leds, configurado como catodo comum (DIG0-DIG7). Neste trabalho foram usados apenas três pinos.

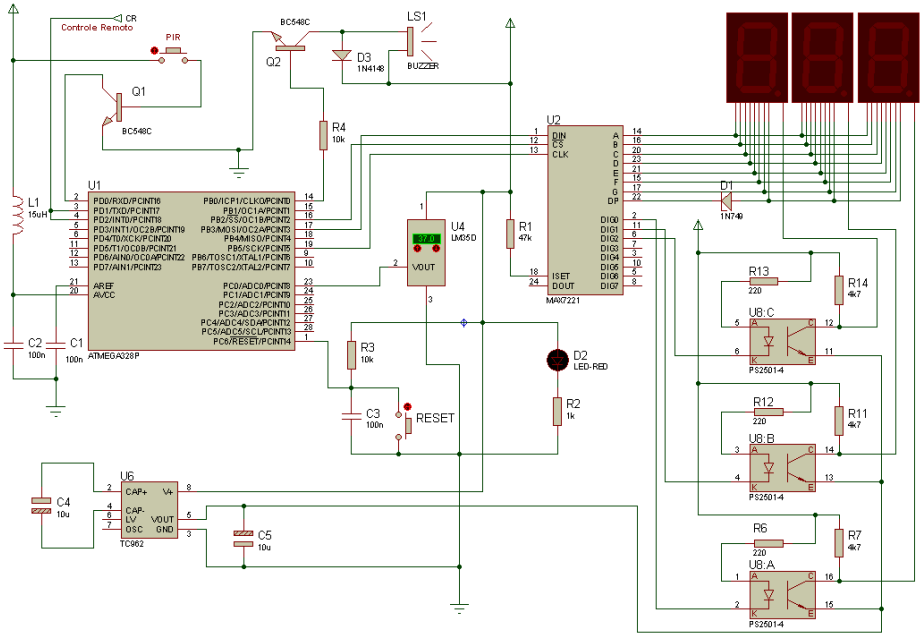

Fig. 28 - Circuito completo deste projeto

Neste dispositivo foram empregados displays de grande dimensão, vistos no item 4.2.4, onde cada segmento é formado por 4 leds em série, cuja tensão direta de trabalho aproxima-se de 8 V. Porém, como o MAX7221 pode alimentar e controlar leds ou displays diretamente, sem componentes intermediários conforme visto na Fig. 15, desde que limitados a, no máximo, dois leds em série, buscou-se uma alternativa para resolver essa dificuldade.

Partindo do princípio que a tensão da fonte de alimentação padrão desse projeto foi estabelecida como sendo 5 Vcc (Fig. 17), houve a obrigação de se adicionar componentes externos para alcançar a tensão apropriada (8 V) de funcionamento dos segmentos dos displays. Uma solução veio através do circuito integrado TC962, descrito no item 4.2.3, que age como uma fonte de tensão negativa de 5 V, que, em conjunto com a própria fonte de alimentação principal (+5 V), garantiu quase 10 V. Aliando-se isto ao fato deste CI fornecer até 80 mA de corrente contínua, permitiu aos displays serem ligados convenientemente.

Os fotoacopladores permitem que cada display seja ligado ao TC962 de forma a drenarem corrente independentemente do MAX7221

que, nesse caso, emprega seus pinos DIG0-DIG2 como chaves lógicas para acionamento daqueles displays.

O diodo zener (4,3 V) foi adicionado em série com o segmento do ponto decimal (DP) em virtude de este ser constituídos por apenas dois leds, cuja tensão direta fica em torno de 4 V, gerando, assim, a necessidade de equalização da tensão no referido segmento.

O sinalizador acústico está ligado diretamente a um pino de entrada / saída (I/O) do microcontrolador que atua com uma chave digital, ligando e desligando conforme programado para os alarmes sonoros previstos no projeto.

Finalmente, um led de alto brilho é acrescentado ao circuito que indica a ligação do dispositivo à fonte de alimentação.

O código fonte (em C) para o controle e implementação de todas as operações necessárias ao circuito da Fig. 28, está disponível no Anexo B.

4.6 Implementação do dispositivo

4.6.1 Prototipação

Inicialmente todos os ensaios e testes reais foram efetuados em matriz de contatos, onde os componentes e suas interligações podem ser vistos na Fig. 29.

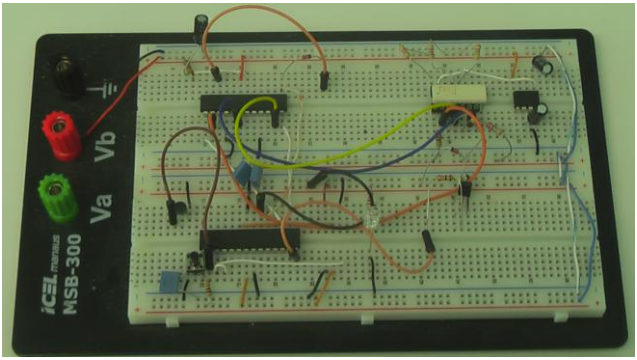

Fig. 29 - Teste do circuito em matriz de contatos

Já os displays foram agrupados em uma placa de fenolite trilhada padrão, encontrada no comércio, e na qual foram adicionados os devidos conectores de PCI (Fig. 30).

Fig. 30 - Displays interligados para teste em placa provisória

4.6.2 Integração dos módulos em painel plástico

Com o foco na portabilidade do dispositivo, optou-se por uma caixa de montagem de 19 x 11 x 5,2 cm de dimensão (Fig. 31), disponível no comércio, e na qual foi possível adaptá-la para acondicionar as duas placas de circuito impresso e os módulos extras por meio de parafusos e espaçadores e, então, interligá-los por cabos padronizados (com terminais adaptados) de 20 cm.

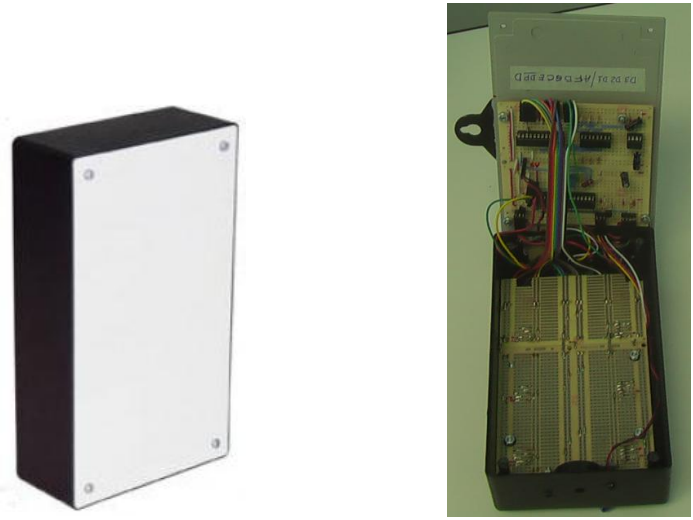

Fig. 31 - Caixa plástica para o painel

Para transformar a caixa plástica em um painel de apresentação para o dispositivo, foram feitas as perfurações adequadas para os displays, os sensores, o sinalizador acústico e o conector P4, que é a entrada da fonte de alimentação. Assim, o dispositivo, em sua versão final, ficou com a aparência indicada na Fig. 32.

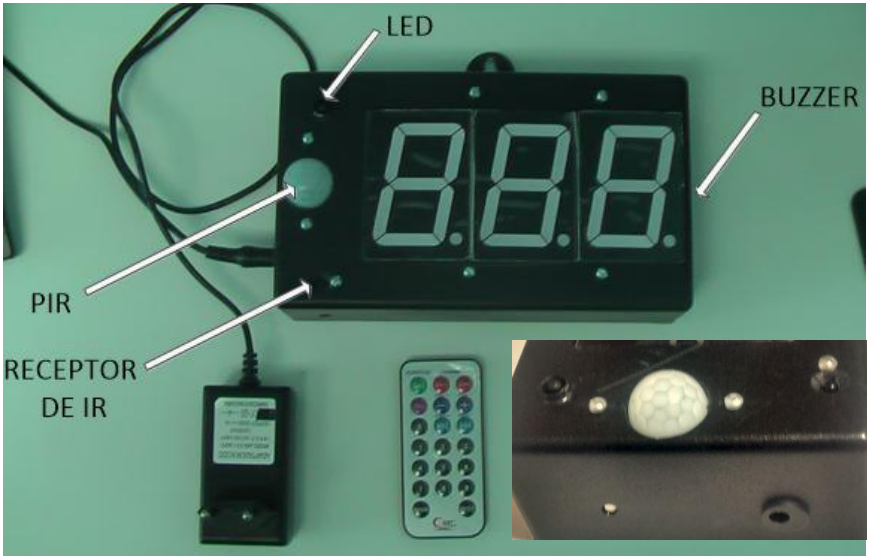

Fig. 32 - Dispositivo montado final (detalhe: sensores)

4.6.3 Teste de campo

O ponto mais crítico deste projeto é a sensibilidade do receptor de sinais infravermelhos emitidos pelo controle remoto. Esta característica é bastante dependente do material e pintura das paredes do recinto, assim como dos móveis e cadeiras, ou seja, o ambiente tem relativa influência neste quesito. Nos testes realizados em diversos locais diferentes, a sensibilidade também variava com a dimensão do recinto.

Para mensurar esta característica, é necessário avaliar duas grandezas que indicarão a confiabilidade do funcionamento do dispositivo: o alcance de transmissão (L) e o ângulo de acionamento (θ), ilustrados na Fig. 33. Considerou-se que o controle remoto e o

dispositivo estejam no mesmo plano e que este é paralelo ao plano do chão.

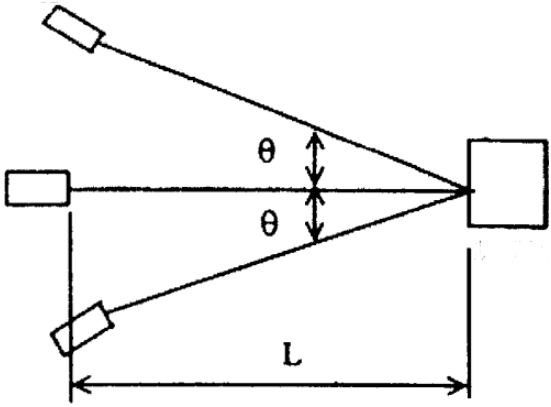

Fig. 33 - Esquemático para teste de sensibilidade do CR

Para realizar as medições do alcance (m) e do ângulo (°), escolheu-se o auditório do Centro de Ciências da Administração e Socioeconômicas – ESAG por ser amplo e plano e também o plenarinho da reitoria, local onde haverá a maioria das reuniões, mas possui menor tamanho. Com o dispositivo instalado e centrado no interior desse recinto, variou-se a posição do controle remoto, fixado em um pedestal, em sua parte posterior, permitindo que as medidas fossem tomadas com o auxílio de uma trena e um transferidor, em visada direta.

Dois conjuntos de medidas foram realizadas, diferenciadas entre si por um desnível vertical de 1,5 metros na altura do controle remoto. Para cada ângulo de acionamento anotou-se a distância aproximada no momento em que o controle remoto não controlava mais o cronômetro. O ângulo de maior sensibilidade na recepção é 0º, mas, na prática, isso nem sempre é possível. Desse modo, optou-se por variá-lo até 50º, um pouco abaixo do valor especificado para o controle remoto, que é 60º (ver item 3.6). A Fig. 34 mostra como procedimento foi feito e a Tab. 2 resume os valores obtidos:

Assim, conclui-se que o controle remoto pode acionar o dispositivo em distâncias que variam entre 3,90 e 7,10 metros e ângulos de 50° e 0°, respectivamente. Os resultados mostram valores que são compatíveis com as características apresentadas no item 3.7.

Tab. 2 – Sensibilidade do receptor de IR aos sinais do CR

Ângulo (θ) [°]	Desnível = 0,0 m	Desnível = 1,5 m
	Distância (L) [m]	Distância (L) [m]
0	7,10	6,50
10	6,80	5,90
20	6,10	5,50
30	6,00	4,80
40	5,70	4,20
50	5,00	3,90

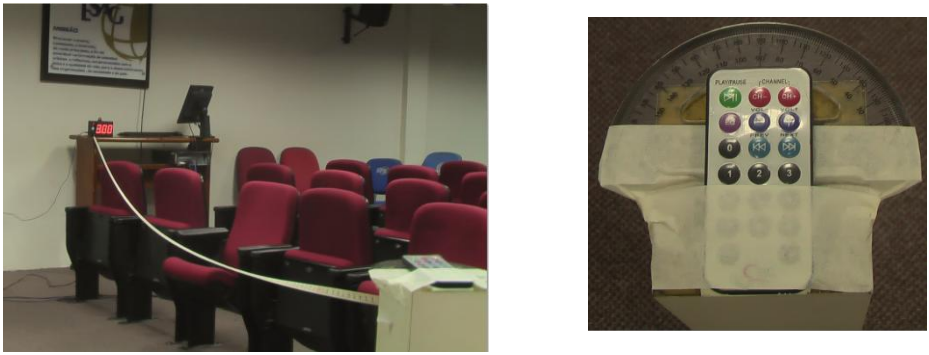

Fig. 34 - Procedimento para medir a sensibilidade do receptor de IR

4.7 Custo do Projeto

Com o objetivo de obter o custo aproximado do dispositivo, considerando a somatória de preços de todos os componentes e materiais empregados, mas excluindo-se o custo do tempo de programação, do tempo montagem e o custo do transporte, relacionaram-se todos os itens de acordo com os locais aonde foram adquiridos (Tab. 3). A exclusão dos custos supracitados foi devido à dificuldade de avaliação, como o tempo de programação e de montagem, que não foram medidos, já que esse trabalho não objetiva a produção sem série do dispositivo e, portanto, estas medições perdem o sentido. Ademais, o custo do transporte não foi avaliado pois a maioria dos componentes foram adquiridos diretamente ao comércio pelo próprio autor, conforme apresentado na Tab. 3.

Tab. 2 – Sensibilidade do receptor de IR aos sinais do CR

Item	Quant.	Local	Transporte	Preço Total R$
ATmega328P	1	Farnell - SP	E-Sedex	11,16
Sinalizador acústico	1	NCR - Fpolis	Próprio	9,50
Cabo com term. fêmeas	24	Proesi - Blumenau	Próprio	30,00
Caixa plástica	1	Proesi - Blumenau		19,50
Cap. eletrol. 25 V	3	Proesi - Blumenau	Próprio	0,15
Cap. poliéster 63 V	3	Proesi - Blumenau	Próprio	0,72
Chave táctil	1	Proesi - Blumenau	Próprio	0,15
Conector barra pino	26	Proesi - Blumenau	Próprio	0,60
Diodo 1N4148	1	Proesi - Blumenau	Próprio	0,04
Diodo Zener	1	Eletro Parts - Fpolis	Próprio	0,25
Display 2,3”	3	Proesi - Blumenau	Próprio	24,60
Fonte chaveada	1	Proesi - Blumenau	Próprio	18,30
Indutor 15 uH	1	Proesi - Blumenau	Próprio	0,25
Jack J4 DC	1	Proesi - Blumenau	Próprio	0,45
Led de alto brilho	1	Proesi - Blumenau	Próprio	0,49
LM35D	1	Farnell - SP (importado)	E-Sedex	34,43
MAX7221	1	Farnell - SP (importado)	E-Sedex	39,42
Módulo IR + CR	1	Proesi - Blumenau	Próprio	35,00
Parafuso + porca milim.	15	Ferramental - Fpolis	Próprio	2,10

Item	Quant.	Local	Transporte	Preço Total R$
Parafuso autoatarrach.	4	Ferramental - Fpolis	Próprio	0,20
Placa fenolite trilhada	2	Eletro Parts - Fpolis	Próprio	30,00
PS2501-4 (fotoacopl.)	1	Proesi - Blumenau	Próprio	7,90
Resistor 0,25 W	10	Proesi - Blumenau	Próprio	2,00
Sensor PIR	1	Proesi - Blumenau	Próprio	14,50
Soquete 16 pinos	1	Proesi - Blumenau	Próprio	1,45
Soquete 24 pinos	1	Proesi - Blumenau	Próprio	1,10
Soquete 28 pinos	1	Proesi - Blumenau	Próprio	1,80
Soquete 8 pinos	1	Proesi - Blumenau	Próprio	0,45
Suporte metálico	1	Ferramental - Fpolis	Próprio	1,20
TC962	1	Farnell - SP (importado)	E-Sedex	7,67
Transistor BC548	2	Proesi - Blumenau	Próprio	0,26
TOTAL				**295,64**

Por outro lado, o custo total real do projeto e montagem deste dispositivo foi bem superior ao demonstrado acima (R$ 295,64), devido às seguintes razões:

- Quase todos os componentes foram adquiridos duplicados, por garantia.
- Compra de uma matriz de contatos de 1.680 furos.
- Compra de vários componentes e conectores previstos, mas não utilizados, além de solda e placas de fenolite para testes.
- Custo de transporte.

5 CONSIDERAÇÕES FINAIS

Este projeto representa uma melhoria no procedimento de cronometragem das discussões nas diversas reuniões dos Conselhos Superiores da UDESC. Além disso, a portabilidade do dispositivo foi apropriada, pois atualmente algumas reuniões são itinerantes em diversas unidades da Universidade, presentes em algumas cidades de Santa Catarina.

Os objetivos previstos foram alcançados, baseando-se fortemente nos programas (*software*) de simulação de circuitos e de edição, compilação e gravação de chip: Proteus e Atmel Studio, respectivamente.

Foi interessante o desenvolvimento do código fonte a partir do grafismo do GRAFCET, em virtude de sua fácil representação da lógica para definir as diversas operações necessárias para o funcionamento adequado e confiável do dispositivo final.

A maior dificuldade encontrada foi no posicionamento do dispositivo em relação ao ponto onde ocorre o acionamento do controle remoto, pois este deve estar direcionado com certa precisão ao sensor do receptor dos sinais transmitidos. Dependendo do ambiente, fica impraticável o uso deste dispositivo quando há um desnível entre este e controle superior a 1,5 metros com um ângulo frontal acima de 50°, como demonstrado na Tab. 2.

Sugestões para melhorias

Alguns aspectos podem ser explorados no futuro para uma simplificação ou um aprimoramento técnico do projeto apresentado neste trabalho. Dentre as possibilidades de continuidade para alterações nesse projeto, destacam-se:

- Para aumentar a sensibilidade do receptor de sinais do controle remoto, é possível adicionar uma pequena lente concentradora da radiação infravermelha no painel plástico do dispositivo, de forma a minimizar a influência da distância e do ângulo relativos (ver item 4.6.3);
- É possível simplificar o sub-circuito formado pelo TC962 e os fotoacopladores, além de reduzir custo, trocando-os por transistores bipolares e FET's, conforme sugerido no *application note* 1196 da MAXIM (*Using the MAX7219/7221 to drive higher voltage or current*). Neste caso, os displays devem ser do tipo anodo comum, diferentes dos utilizados nesse projeto.

- Também é possível substituir o MAX7221 por diversos transistores e resistores com o emprego da multiplexação temporal para os displays (LIMA, 2012). Apesar de reduzir custo, esta alternativa é limitada em relação às possibilidades de programação oferecidas pelo MAX7221.
- O microcontrolador empregado (ATmega328P) pode ser substituído por outro com estrutura interna mais simples, da mesma família (AVR), e com menor custo.
- Acrescentar outra funcionalidade para o dispositivo: um relógio digital com horas e minutos. Este relógio poderia ser apresentado em alternância com a temperatura sempre que o cronômetro não estiver em uso.

A princípio, conforme visto, não há uma meta quantitativa a ser mensurada, pois este trabalho teve o objetivo de alterar a metodologia atualmente empregada de visualização do controle de tempo de discussões em plenária e, portanto, não há processos inerentes ao projeto em que se possa registrar variações em medidas de tempo, redução de custo, agilização dos trabalhos, só para dar alguns exemplos.

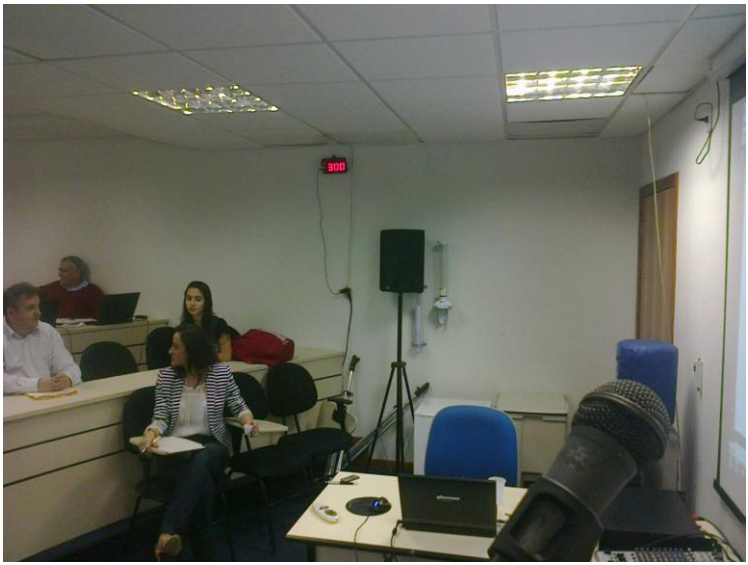

Fig. 35 - Dispositivo em uso no plenarinho da UDESC

Por fim, o dispositivo desenvolvido está sendo empregado (Fig. 35), substituindo a metodologia anterior com os benefícios e/ou resultados assumidos no item 2 deste trabalho, resultando em melhoria de um dos ambientes de trabalho da Universidade, ou seja, os recintos onde se realizam as reuniões dos seus Conselhos Superiores.

6 REFERÊNCIAS BIBLIOGRÁFICAS

LIMA, Charles Borges de; Villaça, Marco V. M. **AVR e Arduino: Técnicas de Projeto**. 2ª edição. Joinville/SC: Clube de Autores Publicações S/A, 2012. 632 p.

MAZIDI, Muhammad Ali *et alii*. **The AVR Microcontroller and Embedded System**. 1ª edição. EUA: Pearson Education, Inc., 2011. 776 p.

GADRE, Dhananjay V. **Programming and Customizing the AVR Microcontroler**. 1ª edição. EUA: McGraw-Hill Companies, Inc., 2001. 339 p.

ATMEL Corporation. ***Datasheet*: 8-bit Microcontroller with 4/8/16/32K**Byte**s In-System Programmable Flash**. EUA: Atmel, 2012. 650p.

MAXIM Integrated Products. ***Datasheet*: Serially Interfaced, 8-Digit LED Display Drivers**. EUA: MAXIM, 2003. 16p.

MICROCHIP Technology Inc. ***Datasheet*: TC962**. EUA: Microchip, 2012. 14 p.

VISHAY Semiconductors. **Data Formats for IR Remote Control**. EUA: Vishay, 2013. 4 p.

ATMEL Corporation. **AVR1200: Using External Interrupts for megaAVR Devices**. EUA: Atmel, 2011. 12 p.

NATIONAL Semiconductor Corp. ***Datasheet*: LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors**. EUA: National, 1994. 12 p.

VISHAY Semiconductors. ***Datasheet*: HS0038B - Photo Modules for PCM Remote Control Systems**. EUA: Vishay, 2000. 7 p.

LEITÃO, Paulo Jorge Pinto. **Notas de Apoio à Disciplina de Automação e Robótica**. Portugal: Instituto Politécnico de Bragança, 2004. 98 p.

MESQUITA, Renato Cardoso. **Curso de Programação em C**. Belo horizonte: UFMG, 2001. 111 p.

PHUNG, Son Lam. **Getting Started with C Programming for the ATMEL AVR Microcontrollers**. 2ª edição. Austrália: University of Wollogong, 2013. 11 p.

ALTIUM Designer. **NEC Infrared Transmission Protocol**. Disponível em: http://techdocs.altium.com/display/FPGA/NEC+Infrared+ Transmission+Protocol. Acessado em maio de 2014.

ROHINI, Laxmi. **Microcontroller Based Home Security System Using Wireless Alerts**. Índia: Gokaraju Rangaraju Institute of Engineering and Technology, 2011. 56 p.

SUNROM Technologies. **ST3679 - Infrared remote control decoder NEC**. India: Sunrom, 2011. 10 p.

VASILIY, Rubashka. **Implementation of IR NEC protocol**. Ucrânia: 2007. Disponível em: <http://www.mcselec.com/index.php?option= com_content& task=view&id= 223&Itemid=57>. Acessado em maio de 2014.

LEITÃO, Paulo Jorge Pinto. **Notas de Apoio à Disciplina de Automação e Robótica**, setembro de 2004. 46 f. Notas de aula. Documento em pdf.

XINDALI Electronics Co. ***Datasheet*: HS-23101B**. Hong Kong: Xindali, 2006. 1 p.

ARDUINO. **Arduino UNO**. Disponível em: < http://arduino.cc/en/Main/ArduinoBoardUno>. Acessado em novembro de 2013.

ELECFREAKS. ***Datasheet*: DYP-ME003**. Disponível em: <http://elecfreaks.com/store/ download/datasheet/sensor/DYP-ME003/ Specification.pdf. Acessado em fevereiro de 2014.

ANEXO A – Código em C para simulação do CR

```
/*
 * Crono_tcc_ir_simul.c
 *
 * Created: 22/03/2014 16:11:07
 *  Author: Lanza
 */

#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
	DDRD |= (1<<6);
	DDRB = 0;
	PORTB = 0xFF;
	TCNT1 = 0xCEC4;   // 110 ms.
	TCCR1B |= ((1<<CS11) | (1<<CS10));
	unsigned char i=0, ender = 0x47, comando, botao;

   while(1)
   {
		botao = 0;
		do
		{
			if (!(PINB & (1<<PINB0)))
			{
				botao = 1;
				comando = 0x00;  // botão
				_delay_ms(200);
			}
			if (!(PINB & (1<<PINB1)))
			{
				botao = 2;
				comando = 0xD3;  // botão B
				_delay_ms(200);
			}
			if (!(PINB & (1<<PINB2)))
			{
				botao = 3;
				comando = 0x40;  // botão C
				_delay_ms(200);
			}
		} while (botao == 0);
		TCNT1 = 0xCEC4;  // 110 ms.
		TCCR1B |= ((1<<CS11) | (1<<CS10));   // liga timer1.

		//*************************************************************
		OCR0A = 200;
		TIFR0 = (1<<OCF0A);
		TCCR0A = 0x42; // toogle, CTC.
```

```
TCCR0B = 0x05;  // PS=1024.

while ((TIFR0 & (1<<OCF0A)) == 0);
TIFR0 = (1<<OCF0A);

OCR0A = 70;
while ((TIFR0 & (1<<OCF0A)) == 0);
TIFR0 = (1<<OCF0A);
OCR0A = 35;
while ((TIFR0 & (1<<OCF0A)) == 0);
TIFR0 = (1<<OCF0A);

for (i = 0; i < 8; i++)
{
	if (ender & (1<<i))
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 15;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
	else
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
}
for (i = 0; i < 8; i++)
{
	if (~ender & (1<<i))
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 15;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
	else
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
}
for (i = 0; i < 8; i++)
```

```
{
	if (comando & (1<<i))
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 15;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
	else
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
}
for (i = 0; i < 8; i++)
{
	if (~comando & (1<<i))
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 15;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
	else
	{
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
		OCR0A = 5;
		while ((TIFR0 & (1<<OCF0A)) == 0);
		TIFR0 = (1<<OCF0A);
	}
}
OCR0A = 30;
while ((TIFR0 & (1<<OCF0A)) == 0);
TIFR0 = (1<<OCF0A);

TCCR0B = 0;

//************************************************************

while ((TIFR1 & (1<<TOV1)) == 0);
TIFR1 = 0x01;
TCNT1 = 0xCEC4;   // 110 ms.

while (PINB != 0xFF)
{
```

```
            TCCR0B = 0x05;

            OCR0A = 70;
            while ((TIFR0 & (1<<OCF0A)) == 0);
            TIFR0 = (1<<OCF0A);

            OCR0A = 17;
            while ((TIFR0 & (1<<OCF0A)) == 0);
            TIFR0 = (1<<OCF0A);

            OCR0A = 5;
            while ((TIFR0 & (1<<OCF0A)) == 0);
            TIFR0 = (1<<OCF0A);

            OCR0A = 5;
            while ((TIFR0 & (1<<OCF0A)) == 0);
            TIFR0 = (1<<OCF0A);

            TCCR0B = 0x00;

            while ((TIFR1 & (1<<TOV1)) == 0);
            TIFR1 = 0x01;
            TCNT1 = 0xCEC4;   // 110 ms.
        }
        TCCR1B = 0;
    }
}
```

ANEXO B – Código fonte completo (*firmware*) em C do dispositivo

```
/*
 * TCC_Cronometro_firm.c
 *
 * Created: 31/08/2013 15:35:21
 * Author: Luiz Carlos
 */

#define F_CPU 8000000UL        // define a frequência do microcontrolador - 8 MHz.

#include <avr/io.h>            // definições de entrada / saída do AVR.
#include <util/delay.h>        // biblioteca para o uso das rotinas de _delay_ms() e _delay_us().
#include <avr/pgmspace.h>      // para o uso do PROGMEM, gravação de dados na memória flash.
#include <avr/interrupt.h>     // define macros para as interrupções do AVR.
#include <avr/sleep.h>         // define macros para o modo sleep do AVR.
#include <avr/wdt.h>           // define macros para o watchdog do AVR.

// definições de macros para o trabalho com bits.
#define   set_bit(Y,bit)   (Y|=(1<<bit))          // coloca em 1 o bit x da variável Y.
#define   clr_bit(Y,bit)       (Y&=~(1<<bit))     // coloca em 0 o bit x da variável Y.
#define   cpl_bit(Y,bit)   (Y^=(1<<bit))          // troca o estado lógico do bit x da variável Y.
#define   tst_bit(Y,bit)   (Y&(1<<bit))           // retorna 0 ou 1 conforme leitura do bit.

// definições da correspondência entre CI's e uC.
#define MOSI PB3
#define SCK PB5
#define SS PB2

// declaração de variáveis globais.
volatile uint16_t conta_10min = 0, conta_15min = 0;
volatile uint8_t aux_flag = 0, aux_ir = 1, ir_cmd, ir_cmd_inv;
volatile uint8_t luiz;

// funções e tratadores de interrupção.
ISR(INT0_vect);
ISR(TIMER1_COMPA_vect);
ISR(PCINT2_vect);
void exec_SPI (unsigned char cmd, unsigned char dados);
void WDT_desliga(void);

int main(void)
{
    unsigned char min, seg1, seg2, minpt;
    unsigned char t_rodada = 1, t_rodada_aux = 3;
    uint16_t temper, temp_dez, temp_un;

    WDT_desliga();                                  // desliga o watchdog.

    // configura o microcontrolador.
    DDRD = 0x00;                                    // PORTD como entrada.
    PORTD = 0xFF;                                   // PORTD com pull-ups habilitados.
```

```
    DDRB = (1<<MOSI) | (1<<SCK) | (1<<SS);          // ajusta MOSI, SCK e SS como saída.
    DDRB |= (1<<DDB0);                              // pino PB0 com saída.
    SPCR = (1<<SPE) | (1<<MSTR);                    // habilita SPI, no modo Mestre.

    // inicializa timer0 e interrupção0.
    set_bit(EICRA, ISC00);                          // interrupção disparada na mudança lógica da INT0 (receptor do sinal IR).
    set_bit(EIMSK, INT0);                           // habilita a interrupção INT0.
    TCCR0B = (1<<CS00) | (1<<CS02);                 // timer0 em modo Normal com prescaler de 1024 (tic = 0,128 ms).

    // (X2) inicializa MAX7221.
    exec_SPI(0x09, 0x0F);                           // habilita decodificação BCD para todos os dígitos.
    exec_SPI(0x0B, 0x02);                           // seleciona 3 dígitos (varredura).
    exec_SPI(0x0A, 0x09);                           // configura a intensidade do brilho do display.
    exec_SPI(0x0C, 0x01);                           // liga o MAX7221.

    // (X2) mostra segmentos centrais (g) dos displays.
    exec_SPI(0x03, 0x0A);                           // mostra "-" em todos os dígitos.
    exec_SPI(0x02, 0x0A);
    exec_SPI(0x01, 0x0A);

    // (X2) inicia timer1.
    OCR1A = 31249;                                  // configura o timer1 para temporizar 1 segundo, em modo CTC.
    set_bit(TIMSK1, OCIE1A);                        // habilita a interrupção do comparador do timer1.
    TCCR1B = 0x0C;                                  // (X2) liga timer1 em modo CTC com prescaler de 256.

    // (X2) configura e inicializa o módulo ADC.
    ADCSRA = 0b10000111;                            // habilita o  ADC e seleciona o prescaler (clk/128).
    ADMUX = 0b11000000;                             // seleciona Vref = 1,1 V e ADC0 com canal de entrada; resultado alinhado
à direita.
    set_bit(DIDR0, ADC0D);                          // habilita pino PC0 como entrada para o ADC.

    _delay_ms(6000);                        // (X2>>X3) atraso de 6 s para atualização inicial do sensor de temperatura.
    sei();                                          // habilita a interrupção global (bit I do registrador SREG).

    while(1)
    {
        // (X3) mostra temperatura.
        set_bit(ADCSRA, ADSC);                      // inicia a conversão.
        while (tst_bit(ADCSRA, ADIF) == 0);         // espera o fim da conversão.
        set_bit(ADCSRA, ADIF);                      // limpa sinalizador de fim de conversão.
        temper = ADC * 55 / 512;                    // converte valor para graus celsius.
        temp_dez = temper / 10;                     // isola o dígito das dezenas.
        temp_un =  temper % 10;                     // isola o dígito das unidades.
        exec_SPI(0x03, temp_dez);                   // mostra a dezena da temperatura no display.
        exec_SPI(0x02, temp_un);                    // mostra a unidade da temperatura no display.
        exec_SPI(0x09, 0x0E);                       // desabilita decodificação BCD para o dígito 1 do display.
        exec_SPI(0x01, 0x63);                       // mostra o grau (°) no dígito 1 do display.

          // ********** bloco correspondente à divergência D1 do grafcet.
        if (aux_flag != 4)
            conta_15min = 0;
        aux_flag = 0;
        do
        {
            if (tst_bit(PIND, PIND0) == 0)
```

```
                conta_15min = 0;                    // (X3) reinicia TM2, se há sinal no PIR.
        }
        while (aux_flag == 0);          // (X3>>M1 ou X3>>4) aguarda sinal do controle remoto ou do timer1 (15 min).
        exec_SPI(0x09, 0x0F);                       // habilita decodificação BCD para todos os 4 dígitos.
        if (aux_flag == 5)
        {
            exec_SPI(0x0C, 0x00);                   // (X4) coloca o MAX7221 em standby.
            _delay_ms(1000);                        // (X4>>X5) atraso de 1 seg.

            // (X5) microcontrolador entra em standby.
            set_sleep_mode(SLEEP_MODE_PWR_DOWN);
            set_bit(PCICR, PCIE2);                  // habilita interrupção por mudança de estado na porta D.
            set_bit(PCMSK2, PCINT17);               // interrupção por mudança de estado no PD1 (CR).
            set_bit(PCMSK2, PCINT16);               // interrupção por mudança de estado no PD0 (PIR).
            cli();
            sleep_enable();
            sei();
            sleep_cpu();                            // (X0) µC em standby.

            sleep_disable();                // (X0>>X1) µC sai do standby através do PIR ou qualquer tecla do CR.
            wdt_enable(WDTO_120MS);         // (X1>>X2) reinicia µC através do watchdog.
            while(1);

        }
        else if (aux_flag == 1)
        {
            t_rodada = 3;                           // (X10) tempo de rodada = 3 min.

X11:    if (t_rodada == 0)
                t_rodada = t_rodada_aux;            // (X11) relógio mostra tempo de rodada.
            minpt = t_rodada | 0x80;                // acrescenta o ponto digital no dígito dos minutos.
            exec_SPI(0x01, 0);                      // mostra 0 no segundo display dos segundos.
            exec_SPI(0x02, 0);                      // mostra 0 no primeiro display dos segundos.
            exec_SPI(0x03, minpt);      // mostra o tempo de rodada com ponto digital no display dos minutos.
            min = t_rodada;
            seg1 = 6;
            seg2 = 10;

            conta_10min = 0;            // (X11) reinicia TM1.

                                        // ********** bloco correspondente à divergência D2 do grafcet.
            aux_flag = 0;
            while (aux_flag == 0);      // (X11>>X12 ou X11>>X13 ou X11>>X3) aguarda sinal do CR  ou do
                                          timer1 (10 min).
            switch (aux_flag)
            {
                case 1:
                    aux_flag = 0;
                    do                  // (X12) início do loop para contagem decrescente do tempo.
                    {
                        min--;
                        minpt = min | 0x80;         // acrescenta o ponto digital no dígito dos minutos.
                        exec_SPI(0x03, minpt);
                        do
                        {
```

```
                seg1--;
                exec_SPI(0x02, seg1);
                do
                {
                    seg2--;
                    exec_SPI(0x01, seg2);
                    if (min == 0 && seg1 == 3 && seg2 == 0)    // bip de alarme para 30 s finais.
                    {
                        set_bit(PORTB, PORTB0);
                        _delay_ms(200);
                        clr_bit(PORTB, PORTB0);
                    }
                    _delay_ms(1000);

// ********** bloco correspondente à divergência D3 do grafcet.
                    if (aux_flag != 0)     // (X12>>X14 ou X12>>X15) bloco executado
                                           se botão do CR foi pressionado.
                        switch (aux_flag)
                        {
                            case 1:    // ********** bloco da divergência D4 do grafcet.
                                aux_flag = 0;
                                while (aux_flag == 0);   // (X14) aguarda sinal do CR.
                                switch (aux_flag)        // (X14>>X12 ou X14>>X11)
                                {
                                    case 1:
                                        break;
                                    default:
                                        goto X11;
                                }        // ********** fim do bloco da divergência D4.
                                aux_flag = 0;
                                break;
                            case 2:
                                goto X11;
                        }        // ********** fim do bloco da divergência D3.
                }
                while (seg2 > 0);
                seg2 = 10;
            }
            while (seg1 > 0);
            seg1 = 6;
        }
        while (min > 0);                // fim do loop para contagem decrescente do tempo.

        // bloco do alarme sonoro.
        set_bit(PORTB, PORTB0);   // (X15) liga bip.
        _delay_ms(200);
        clr_bit(PORTB, PORTB0);
        _delay_ms(100);
        set_bit(PORTB, PORTB0);
        _delay_ms(200);
        clr_bit(PORTB, PORTB0);
        _delay_ms(1000);

        t_rodada = 0;
        aux_flag = 0;
```

```
                    goto X11;
                case 3:
                    if (t_rodada == 1)              // (X13) altera o tempo de rodada.
                        t_rodada = 4;
                    t_rodada = t_rodada - 1;
                    t_rodada_aux = t_rodada;
                    goto X11;
                case 4:
                    break;
                default:
                    goto X11;
            }
        }
        else if (aux_flag == 4)
        {
            conta_10min = 0;
            continue;
        }                                   // ********** fim do bloco da divergência D2.
    }
}
    // função que executa comandos do MAX7221 via interface SPI.
void exec_SPI (unsigned char cmd, unsigned char dados)
{
    clr_bit(PORTB, SS);                     // inicializando o pacote ao levar o pino CS a zero (0).
    SPDR = cmd;                             // inicia a transmissão do byte de comando do MAX7221.
    while (!(tst_bit(SPSR, SPIF)));         // aguarda o término da transferência.
    SPDR = dados;                           // inicia a transmissão do byte de dados do MAX7221.
    while (!(tst_bit(SPSR, SPIF)));
    set_bit(PORTB, SS);                     // finalizando o pacote ao levar o pino CS a um (1).
}

ISR(INT0_vect)                              // tratador da interrupção externa INT0.
{
    uint8_t ir_sinal = tst_bit(PIND, PIND2);    // X20
    static unsigned char cont_bit;

    if ((!ir_sinal) && (aux_ir == 1))       // (X20>>X21): primeiro sinal detectado.
    {                                   // X21
        TCNT0 = 0;
        aux_ir = 1;
        return;
    }
    switch (aux_ir)
    {
        case 1:
            if (ir_sinal && TCNT0 > 65 && TCNT0 < 75)   // (X20>>X22): largura do sinal = 9 ms.
            {                           // X22
                TCNT0 = 0;
                aux_ir = 2;
                break;
            }
            else
                aux_ir = 1;
            break;
        case 2:
```

```
        if (!ir_sinal && TCNT0 > 30 && TCNT0 < 40)      // (X20>>X23): largura do sinal = 4,5 ms.
        {                               // X23
            TCNT0 = 0;
            aux_ir = 3;
            cont_bit = 0;
            break;
        }
        else
            aux_ir = 1;
        break;
    case 3:
        if (ir_sinal && TCNT0 > 2 && TCNT0 < 7)         // (X20>>X24): largura do sinal = 0,56 ms.
        {                               // X24
            TCNT0 = 0;
            aux_ir = 4;
            break;
        }
        else
            aux_ir = 1;
        break;
    case 4:
        cont_bit++;                     // X25
        if (!ir_sinal && TCNT0 > 9 && TCNT0 < 19)       // (X25>>X26): largura do sinal = 1,69 ms.
        {                               // X26
            if (cont_bit >= 17 && cont_bit <= 24)
                set_bit(ir_cmd, (cont_bit - 17));       // salva bit 1 do comando.
            if (cont_bit >= 25 && cont_bit <= 32)
                set_bit(ir_cmd_inv, (cont_bit - 25));   // salva bit 1 do comando invertido.
        }
        else if (!ir_sinal && TCNT0 > 2 && TCNT0 < 7)   // (X25>>X27): largura do sinal = 0,56 ms.
        {                               // X27
            if (cont_bit >= 17 && cont_bit <= 24)
                clr_bit(ir_cmd, (cont_bit - 17));       // salva bit 0 do comando.
            if (cont_bit >= 25 && cont_bit <= 32)
                clr_bit(ir_cmd_inv, (cont_bit - 25));   // salva bit 0 do comando invertido.
        }
        Else                            // (X25>>X28): largura do sinal != 0,56 ms e != 1,69 ms.
        {                               // X28
            aux_ir = 1;
            break;
        }

        if (cont_bit < 32)              // (X26>>X29 ou X27>>X29).
        {                               // X29
            TCNT0 = 0;
            aux_ir = 3;
            break;
        }

        aux_ir = 1;                     // X30
        luiz = ~ir_cmd;
        if (ir_cmd_inv != luiz)
        {
            break;                      // (X30>>X20).
        }
```

```
                switch (ir_cmd)             // (X31) traduz a equivalência entre comando e tecla pressionada.
                {
                    case 0x00:              // código da tecla A
                        aux_flag = 1;
                        break;
                    case 0x01:              // código da tecla B
                        aux_flag = 2;
                        break;
                    case 0x02:              // código da tecla C
                        aux_flag = 3;
                        break;
                    case 0x04:              // código da tecla A
                        aux_flag = 1;
                        break;
                    case 0x05:              // código da tecla B
                        aux_flag = 2;
                        break;
                    case 0x06:              // código da tecla C
                        aux_flag = 3;
                        break;
                }
        }
}

ISR(TIMER1_COMPA_vect)              // tratador da interrupção do TIMER1.
{
    conta_10min++;
    conta_15min++;
    if (conta_10min == 600)
        aux_flag = 5;
}

ISR(PCINT2_vect)                    // tratador da interrupção externa por mudança de estado.
{
    return;
}

void WDT_desliga(void)              // função para desligamento do watchdog do AVR.
{
    cli();
    wdt_reset();
    MCUSR &= ~(1<<WDRF);
    WDTCSR |= (1<<WDCE) | (1<<WDE);
    WDTCSR = 0x00;
}

//exec_SPI(0x04, 0x0D);//L-teste
//exec_SPI(0x04, 0x0C);//H-teste
//exec_SPI(0x04, 0x0E);//P-teste
```