



ECOP04 - PROGRAMAÇÃO EMBARCADA

RELÓGIO E DESPERTADOR INTELIGENTE

Bruno Stanley de Jesus

28 de julho, 2021

SUMÁRIO

1	INTRODUÇÃO	3
2	OBJETIVOS	3
3	DESENVOLVIMENTO	3
3.1	Funcionalidades	3
3.2	Fluxograma	7
3.3	Dificuldades	8
3.4	Solução	10

1 INTRODUÇÃO

Sistemas embarcados são sistemas computacionais completos e independentes, mais simples que um computador de propósito geral (desktops), encarregados de executar apenas tarefas pré-determinadas, com requisitos específicos e que geralmente são executadas repetidas vezes.

Esses dispositivos são compostos fundamentalmente pelos mesmos componentes de um computador pessoal, só que com tamanho e capacidade limitadas para alguma finalidade específica.. São muito utilizados no cotidiano, e seus usuários geralmente não os consideram como um computador. São exemplos: aparelho de som, televisão, câmera digital, brinquedos e diversos outros dispositivos.

Sabendo da importância dos sistemas embarcados, será utilizado o PICSimLab para simular o funcionamento de um microcontrolador PIC18F4520, que será adequado para o correto funcionamento do projeto.

2 OBJETIVOS

Utilizar conceitos vistos durante a disciplina de ECOP04 para simular um relógio e despertador inteligente, o qual será simulado por meio do PICSimLab - PICGenios, utilizando o microcontrolador PIC18F4520. Todo código fonte será desenvolvido por meio do MPLAB X, importando algumas bibliotecas disponibilizadas durante as aulas.

3 DESENVOLVIMENTO

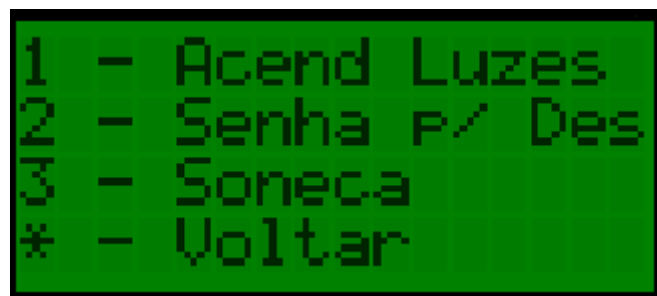
3.1 Funcionalidades



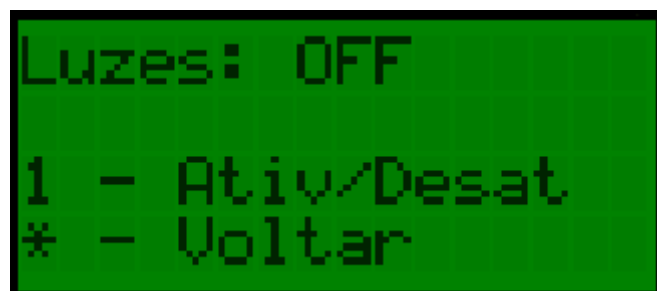
Após iniciar, o usuário pode visualizar o horário atual e selecionar a opção para ajustar o horário do alarme ou navegar pelas opções de acionamento do alarme.




Selecionando a opção de ajuste do alarme, o usuário poderá utilizar as teclas de 1 a 6 do teclado do PICSImLab para ajustar os dígitos das horas, minutos e segundos. Para criar de fato o alarme, deve ser selecionada a opção “Salvar” ou “Cancelar” para descartar qualquer alteração.



Escolhendo Opções de Alarme, o usuário poderá escolher alguma das ferramentas utilizadas no momento do acionamento, como: acender as luzes, digitar uma senha para desativar o alarme, ou então a opção de soneca para estender em alguns minutos o alarme.

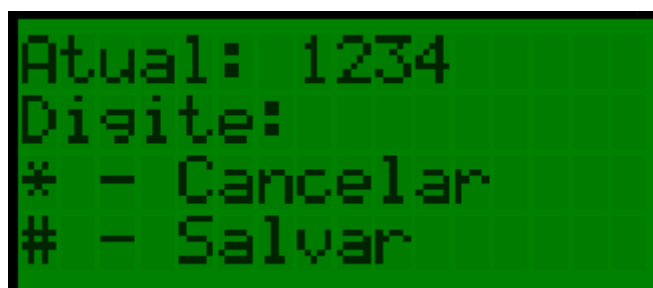


Selecionando a opção de acender Luzes, com a tecla 1 poderá estar ativando ou desativando essa opção.

A green LCD screen with black text. The text reads: "Senha: OFF", "1 - Ativ/Desat", "2 - Alter Senha", and "* - Voltar".

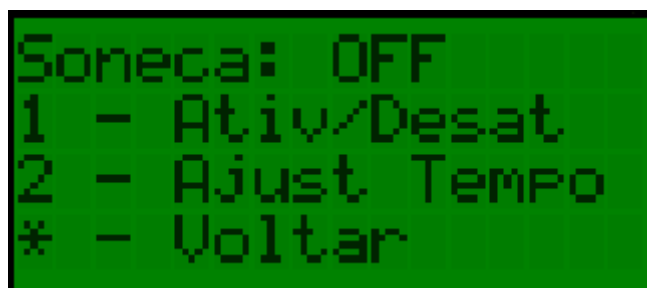
```
Senha: OFF
1 - Ativ/Desat
2 - Alter Senha
* - Voltar
```

Na opção senha para desativar, com a tecla 1 novamente, onde o usuário pode estar ativando ou desativando, e com a tecla 2 alterando ou definindo uma senha.

A green LCD screen with black text. The text reads: "Atual: 1234", "Digite:", "* - Cancelar", and "# - Salvar".

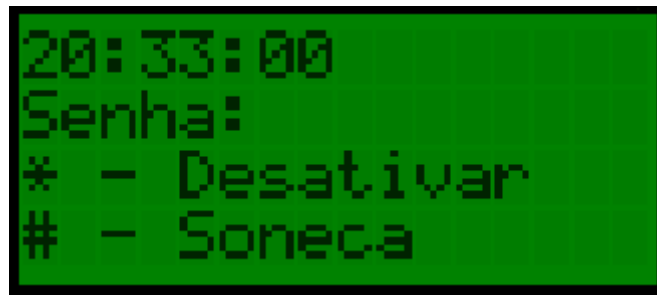
```
Atual: 1234
Digite:
* - Cancelar
# - Salvar
```

Ao escolher Alterar Senha, o usuário deve inserir uma senha de 4 dígitos, utilizando os botões do teclado, e em seguida selecionar a opção para salvar ou descartar as alterações.

A green LCD screen with black text. The text reads: "Soneca: OFF", "1 - Ativ/Desat", "2 - Ajust Tempo", and "* - Voltar".

```
Soneca: OFF
1 - Ativ/Desat
2 - Ajust Tempo
* - Voltar
```

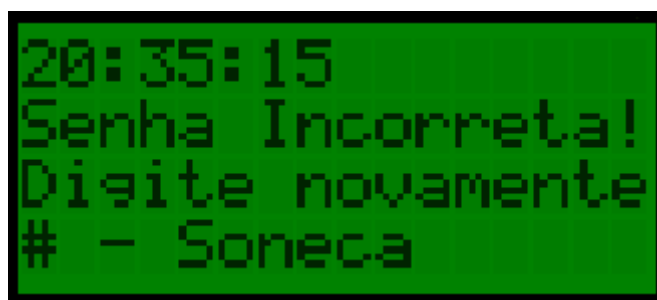
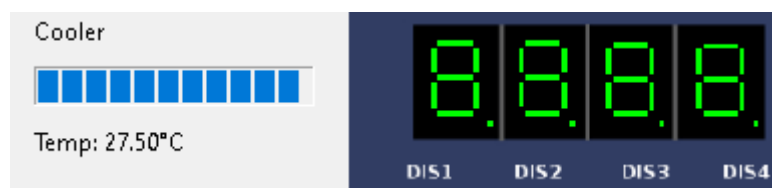
De última opção temos a Soneca, onde o usuário pode estar ativando ou desativando com a tecla 1, e com a tecla 2 ajustando o tempo de soneca, que por padrão é 5 minutos.



Assim que o alarme for acionado, uma tela aparecerá informando o horário do alarme e também pedindo uma senha para desativar, caso essa opção tenha sido ativada.

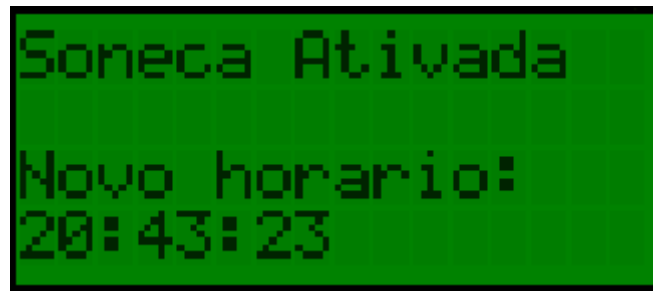
Para desativar, basta o usuário inserir uma senha utilizando os botões do teclado e em seguida pressionar “*” para desativar ou “#” para ativar a soneca, caso também tenha sido ativada.

No alarme, os displays de 7 seg. e cooler são ativados como mostra na figura abaixo.



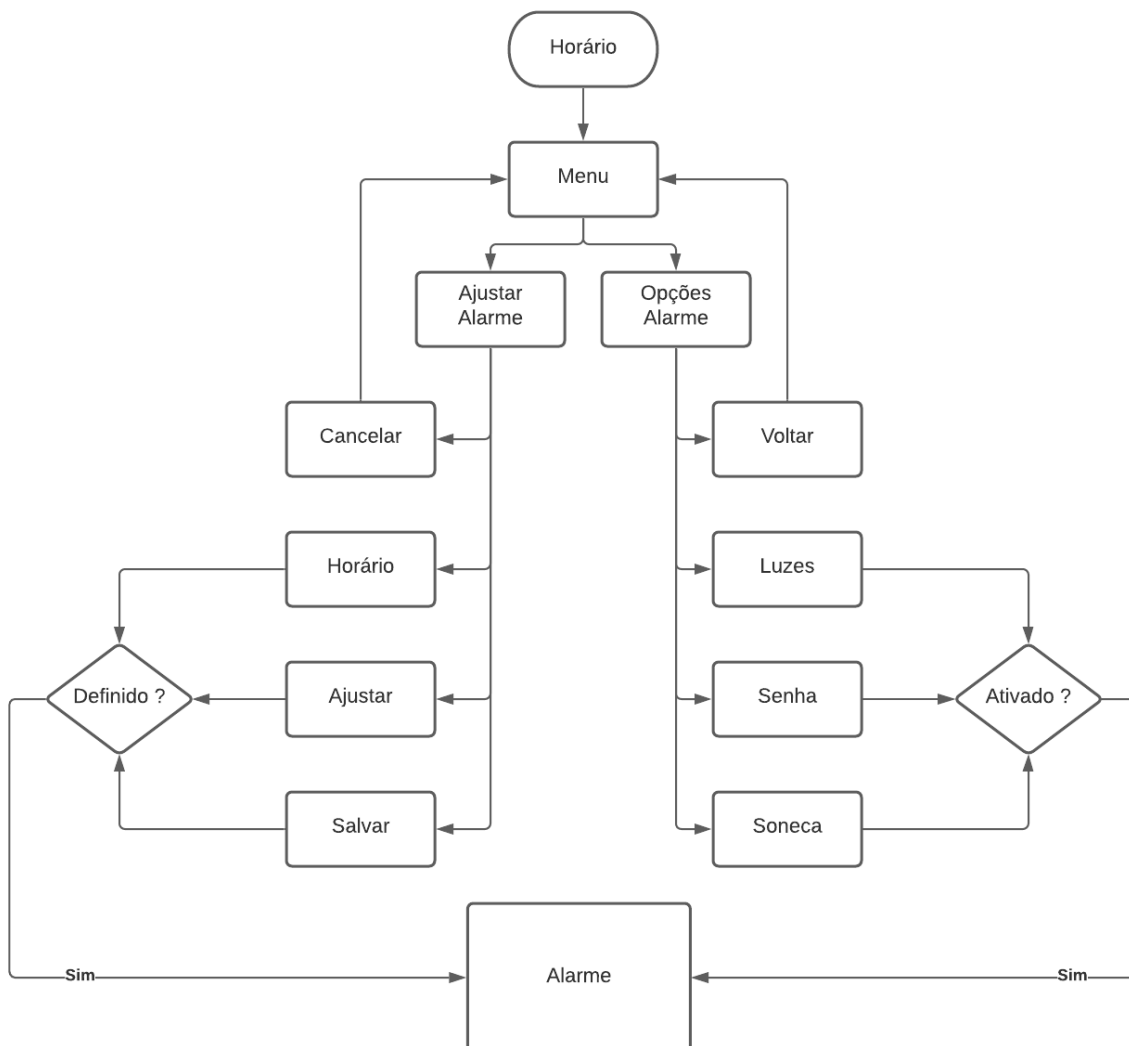
Caso o usuário informe uma senha diferente do que foi definida nas opções do alarme, uma mensagem será mostrada pedindo para o mesmo digitar novamente a senha.

Se a senha digitada estiver correta, o alarme é desativado e os displays de 7 segmentos e cooler são desligados, voltando para a primeira tela do programa.



Escolhendo a opção soneca, será mostrada uma mensagem de soneca ativada, e também um novo horário do alarme. E então, os displays de 7 segmentos e cooler são desligados, voltando para a primeira tela do programa.

3.2 Fluxograma



3.3 Dificuldades

```
//TEXTO MENU - 1-HORARIO ALARME
lcdCommand(L2);
for (int i = 0; i < 16; i++) {
    lcdData(msg[0][i]);
}

//TEXTO MENU - 2-OPCOES ALARME
lcdCommand(L3);
for (int i = 0; i < 16; i++) {
    lcdData(msg[1][i]);
}

for (;;) {
    //Atualização constante do Relógio
    lcdCommand(L0);
    rtc_r();
    for (int j = 0; j < 8; j++) {
        lcdData(time[j]);
    }

    //Verificar se algum botão foi pressionado
    kpDebounce();
    if (kpRead() != tecla) {
        tecla = kpRead();
        break;
    }
}
```

A princípio estava utilizando a função do RTC para pegar o horário ao iniciar o programa, e também para ficar atualizando ele constantemente. Porém o primeiro erro surgiu, ao tentar pressionar o botão para entrar nas opções de ajustes do Alarme e não conseguir ter a tecla lida pela função `kpRead()`.

Após tal fato, surgiu a ideia de utilizar a função do RTC somente ao iniciar o programa para pegar o horário, e a partir de então ir atualizando o relógio utilizando um contador e fazendo algumas operações básicas.


```

    }

    horario[0] = (horas / 10) + 48;
    horario[1] = (horas % 10) + 48;
    horario[3] = (minutos / 10) + 48;
    horario[4] = (minutos % 10) + 48;
    horario[6] = (segundos / 10) + 48;
    horario[7] = (segundos % 10) + 48;

    lcdCommand(L0);

    for (int j = 0; j < 8; j++) {
        lcdData(horario[j]);
    }

    atraso_ms(1000);

    kpDebounce();
    if (kpRead() != tecla) {
        tecla = kpRead();
        break;
    }
}

```

Posteriormente, o botão para entrar nos ajustes já funcionava enquanto o horário ia se atualizando, entretanto, ainda não estava 100% perfeito, pois como já era de se esperar, o tempo do relógio desajustado e atualizando muito rápido. Então, a primeira coisa a se fazer foi adicionar alguns loops "for" para atrasar alguns milissegundos, e consequentemente o mesmo erro de quando se utilizava a função do RTC passou a se apresentar novamente. O `kpRead()` não estava lendo o botão pressionado no momento que o programa entrava no loop de atraso.

3.4 Solução

```
for (;;) {  
    //Atualização constante do Relógio  
    cont++;  
  
    if((cont%200) == 0){  
        segundos++;  
    }  
  
    if(segundos >= 60){  
        segundos = 0;  
        minutos++;  
        if(minutos >= 60){  
            minutos = 0;  
            horas++;  
            if(horas >= 24){  
                horas = 0;  
            }  
        }  
    }  
  
    horario[0] = (horas/10) + 48;  
    horario[1] = (horas%10) + 48;  
    horario[3] = (minutos/10) + 48;  
    horario[4] = (minutos%10) + 48;  
    horario[6] = (segundos/10) + 48;  
    horario[7] = (segundos%10) + 48;
```

Depois de vários ajustes, surgiu uma simples solução que foi ao invés de utilizar o for para atrasar a execução do programa e ajustar o relógio, se utilizar uma variável "cont" e ir incrementando os segundos a cada múltiplo de um valor pré-ajustado através da operação de resto (%). Assim, o programa não era desviado para nenhuma função de atraso, permitindo então que a função `kpRead()` pudesse verificar a todo momento se algum botão estava sendo pressionado.