UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA – EEL
CENTRO TECNOLÓGICO – CTC
CAMPUS UNIVERSITÁRIO - TRINDADE - CEP 88040-900
FLORIANÓPOLIS - SANTA CATARINA

EEL7323: Programação C++ para Sistemas Embarcados

Relatório do Projeto Final:

Sistema de registro de vigilante

Rafael Bidese Puhl
Romano Sabetzki Weirich

Florianópolis, Dezembro de 2014.

1. INTRODUÇÃO

O projeto final da disciplina consiste no desenvolvimento de um software para monitoramento de rondas de guardas. Em cada posto, o guarda deve entrar com sua matricula e confirmar que esteve naquele ponto no momento em que devia. O projeto foi desenvolvido em uma plataforma FPGA Atlys da Digilent, onde foi carregado um processador Leon3, além disso foi utilizado um módulo de OLED chamado PmodOLED.

O sistema foi desenvolvido de forma modular, em que várias classes foram desenvolvidas para facilitar o desenvolvimento do projeto e também a leitura do mesmo por um terceiro que deseje utilizar o código. O sistema inteiro é baseado em uma lista que acumula nós e os demonstra em um display de OLED. O sistema fica em um loop, esperando que um botão seja apertado para que alguma decisão seja tomada, e enquanto isso, o relógio que é mostrado no display é atualizado. Uma vez que um dos botões é apertado, ele invoca uma rotina que fará o que aquele botão deveria causar.

2. CÓDIGOS DESENVOLVIDOS

a. Node

Foi criado uma classe Node com os métodos necessários para a implementação do projeto, já que não era permitido utilizar a biblioteca STL.

Class Node:
private
ClockCalendar checkpoint;
Int ID; //matricula do vigia
Node next;*
Node prev;*
public:
Node(int _ID, ClockCalendar data, Node nxt, Node*prev);// Construtor de nó*

ClockCalendar getCheckpoint(); //método para pegar a data/hora*

Void setCheckpoint(); //metodo para adicionar o checkpoint ao nó

Node getNext();//retorna um ponteiro para o próximo*

Node getPrev();//retorna um ponteiro para o anterior*

Void setNext(Node nxt);//altera o próximo do nó*

Void setPrev(Node prv);//altera o anterior do nó*

Void readCheckpoint();//le o valor da matricula e escreve no nó

b. List

Foi também elaborada uma classe List, já que não se podia utilizar a biblioteca STL.

Class List:

Private:

Node head;*

Node tail;*

Public:

List();

~List();

Void insertBeforeFirst(ClockCalendar data, int _ID);

Node readNode(int pos);*

c. Clock

Uma classe relógio também foi criada para lidar com os horários que devem ser adicionadas aos nós. Além disso também para mostrar o relógio em tempo real.

Class Clock:

Protected:

Int hr, min, sec, pm;

Public:

Clock();

Clock(int _hr,int _min, int _sec, int _pm);

Void setClock(int _hr,int _min, int _sec, int _pm);//para iniciar o relógio no valor desejado

Int advance();//para avançar a cada segundo

d. Calendar

Para atualizar e manter as datas, foi criada a classe Calendar.

Class Calendar:

Protected:

Int yr, mo, day;

Public:

Calendar();

Calendar(int _yr, int _mo, int _day);

Void setCalendar(int _yr, int _mo, int _day);//para iniciar o calendário na data desejada

Void advance(); //para avançar os dias

e. ClockCalendar

A classe ClockCalendar foi criada para que possa serem utilizadas as classes Clock e Calendar como uma coisa só, por exemplo, quando o relógio mudar de um dia para o outro, já seja atualizado no calendário.

Class ClockCalendar: public Clock, publick Calendar

ClockCalendar();

ClockCalendar(int _hr,int _min, int _sec, int _pm, int _yr, int _mo, int _day);

Void showClockCalendar(); //funcao para mostrar o valor do ClockCalendar no oled

Void advance(); //avanca o clock e quando é necessário atualiza o calendar.

f. Oled

Nesta classe, é definido os pinos que são utilizados para fazer a comunicação SPI com o OLED. Assim, define-se primeiro os pinos, posteriormente são criadas rotinas para inicialização, enviar dados, enviar comandos, desligar e etc.

Class Oled:

#define CS 24 //pin 1

#define SDIN 25 //pin 2

#define SCLK 27 //pin 4

#define DC 28 //pin 7

#define RES 29 //pin 8

#define VBATC 30 //pin 9

#define VDDC 31 //pin 10

#define command 1

#define data 0

Void delay(clock_t time)

Void setPin(int pin, int value);// seta o pino pin para o value desejado

Void sendHex(unsigned hex, int command_or_data);//envia commando ou data hex para o OLED utilizando a SPI em software

Void displayInit();//Inicializa o display de acordo com os comandos presentes no datasheet

Void displayOff();//rotina de power-down do display

Void setLine(unsigned line);

void printChar(char aux);//envia caractere

void printString(char string);//envia cadeia de caracteres*

g. Main

O main, parte mais importante do projeto foi programada como segue:

Main:

Primeiro é setado o relógio inicial, data e hora;

Posteriormente, é criada a lista para armazenamento dos checkpoints;
O display de OLED é inicializado;
É selecionada a primeira linha do OLED e então, escreve-se o nome dos desenvolvedores;
Posteriormente entra-se em um loop, que fica ativo enquanto o botão “direcional” para cima não é apertado;
Neste loop, primeiramente há uma cadeia de “ifs”, primeiro if confere se um segundo se passou desde a ultima atualização do relógio, se tiver se passado, então o relógio é atualizado e novamente printado no OLED;
A próximo ”if” confere se o botão de desligar foi apertado, se sim, termina o programa;
Os próximos “ifs” serão acionados se um dos botões (direita, esquerda ou centro) forem apertados;
Se o botão para a esquerda for apertado, um ponteiro anda para a esquerda dentro da lista dos checkpoints, desta forma atualizará o OLED com o nó que estiver a esquerda;
Se o botão para a direita for apertado, da mesma forma, o ponteiro apontará para o próximo item da lista;
Se o botão central for apertado, então um novo nó deve ser criado, utilizando as informações de data e hora deste momento e também o estado atual dos switches que são a entrada da matrícula do vigilante. Depois de este nó ser criado, o mesmo aparece como a primeira posição na lista e pode ser visto no OLED;
Finalmente, caso o botão de desligar tenha sido pressionado, o OLED então inicia sua rotina de desligar e o programa será finalizado com sucesso.

3. CONCLUSÃO

Foi desenvolvido um protótipo de controle para ronda de vigilantes em C/C++ utilizando uma plataforma de desenvolvimento Atlys da Digilent. O projeto foi desenvolvido utilizando conceitos de programação orientada a objetos aprendidos ao decorrer da disciplina. Além disso, parte dos códigos utilizados no projeto final também foram desenvolvidos no decorrer do programa da disciplina, o que mostra que a programação orientada a objetos tem grande desenvoltura quando necessário desenvolver o projeto em blocos separados. Por exemplo, as bibliotecas de Clock, Calendar, ClockCalendar, List, e Node, já haviam sido desenvolvidas na disciplina e só precisaram pequenas mudanças para serem utilizadas no projeto final.

Além disso, foi implementado o protocolo de comunicação SPI utilizado para comunicação com o OLED em software, o que é algo que proporcionou grande aprendizado para os alunos. Mais projetos deste tipo poderiam ser incluídos na disciplina, visto que no decorrer da graduação o aluno precisaria recorrer a recursos próprios e aprender por conta este tipo de prática já que nenhuma outra disciplina cobre tal tema.

Um problema que foi presente neste projeto foi a dificuldade de quantificar o tempo, devido a não possuir um relógio de tempo real (RTC), o projeto teria sido facilitado, caso se tivesse um periférico como este para estipular os delays do dentro do projeto. Além disso, rotinas de interrupção também teriam sido bastante úteis principalmente para lidar com o acionamento dos botões, portanto, este tema poderia ser também incluído nas próximas versões desta disciplina.