Gean Francesco Deroso Chu
Guilherme Sadao Hayashi
Johnny William Flauzino de Souza

Projeto Linux Embedded

Projeto realizado como requisito parcial para avaliação do Programa de Aprendizagem em Microprocessadores II e Eletrônica II do Curso de Engenharia de Computação da Pontifícia Universidade Católica do Paraná, sob a orientação dos professores: Afonso Ferreira Miguel e Ivan Jorge Chueiri.

Curitiba
2009

Indice

1 RESUMO

O projeto consiste em uma parcela de um outro, que está sendo desenvolvido para partcipar do programa Premio Inova da Whirlpool. Essa parcela propõe o desenvolvimento de um programa capaz de fazer aquisição de temperatura e o mesmo Rodar em um processador ARM9.

Também liberamos todas as portas necessárias para o projeto principal e adquirimos o conhecimento necessário para compilação de aplicativos para o mesmo.

2 Projeto Inova

O projeto consiste em um novo sistema de controle para eletrodomésticos, aplicado em um aparelho microondas. O sistema possuirá uma interface touch-screen em conjunto com um display LCD, de forma que o usuário possa estender as funcionalidades do aparelho em questão através de uma série de outros recursos implementados.

Utilizando uma conexão à internet, o usuário terá informações sobre, por exemplo, previsão climática, notícias, serviços, entre outros. Por meio de sensores infravermelho de temperatura, será possível controlar o período de acionamento do forno microondas não apenas pelo tempo, mas também pela temperatura média do alimento. Será possível buscar receitas em servidores na internet, assim como desenvolver um sistema inteligente que automaticamente gerencie o processo de cozimento de acordo com a temperatura do alimento.

3 Objetivo

3.1 Geral

Com base no programa do projeto para Whirlpool, desenvolver as partes de sensoriamento e atuação e assim conhecer a aqruitetura ARM principalmente o microprocessador ARM9.

3.2 Espefico

I. Estudar um SO embarcado.
II. Criar drivers Necessários.
III. Realizar a leitura de temperaturas, por meio de um sensor do mesmo.
IV. Enviar instruções e atuar em buzzers e leds, para testar atuadores.

4 MATERIAIS UTILIZADOS

- Kit de desenvolvimento Mini2440.
- LCD Touchscreen 7 in.
- Cabos Variados.
- LM35.

5 MINI2440

5.1 Especificação

- Dimension: 100 x 100 mm
- CPU: 400 MHz Samsung S3C2440A ARM920T (Max freq. 533 MHz)
- RAM: 64 MB SDRAM, 32 bit 100 MHz Bus
- Flash: 64 MB NAND Flash and 2 MB NOR Flash with BIOS
- EEPROM: 1024 Byte 24C08 (I2C)
- Ext. Memory: SD-Card socket
- Serial Ports: 1x DB9 connector (RS232), total: 3x serial ports on the PCB
- USB: 1x USB Host, 1x USB Device
- Audio Output: 3.5 mm stereo jack
- Audio Input: on PCB + condenser microphone
- Ethernet: RJ-45 10/100M (DM9000)
- RTC: Built in Real Time Clock with battery
- Beeper: On board PWM buzzer
- Camera: 20 pin Camera interface
- LCD Interface:
  - STN Displays:
    - 4 bit dual scan, 4 bit single scan or 8 bit single scan display type
    - monochrome, 4 gray levels, 16 gray levels, 256 colors or 4096 colors
    - Max: 1024x768, 4096 colors
  - TFT Displays:

- 1, 2, 4 or 8 bpp palletized color displays
- 16 or 24 bpp non-palletized true-color displays
- Max: 1024x768, 64k colors

- Touch Panel: 4 wire resistive
- User Inputs: 6x push buttons and A/D pot
- User Outputs: 4x LEDs
- Expansion: 40 pin System Bus, 34 pin GPIO (2.0mm)
- Debug: 10 pin JTAG (2.0mm)
- Power Connector: 5V with switch and LED

5.2 Sistemas operacionais Suportados

- Android
- Linux 2.6

6 DESCRIÇÃO DO PROJETO

Para desenvolver o projeto proposto, utilizaremos um display LCD colorido com uma película sensível ao toque baseada na tecnologia 4-wire resistive, podendo desta maneira ser operado mesmo com os dedos.

Para realizar o controle da unidade de processamento utilizaremos um microcontrolador ARM, trabalhando com um sistema operacional linux embarcado. Desta maneira, pretende-se facilitar o gerenciamento das múltiplas tarefas realizadas pelo sistema, interface gráfica, bem como permitir fácil manutenção de código, implementação de novos recursos, etc.

Como mostrado na figura 2, o sistema operacional será responsável pela interface entre os vários periféricos utilizados no sistema e os aplicativos, além de gerenciar programas trabalhando em diferentes threads.

Para o desenvolvimento foi escolhido o Kit de desenvolvimento Mini2440 da friendlyarm. Onde o mesmo já vem com alguns periféricos para inteface.

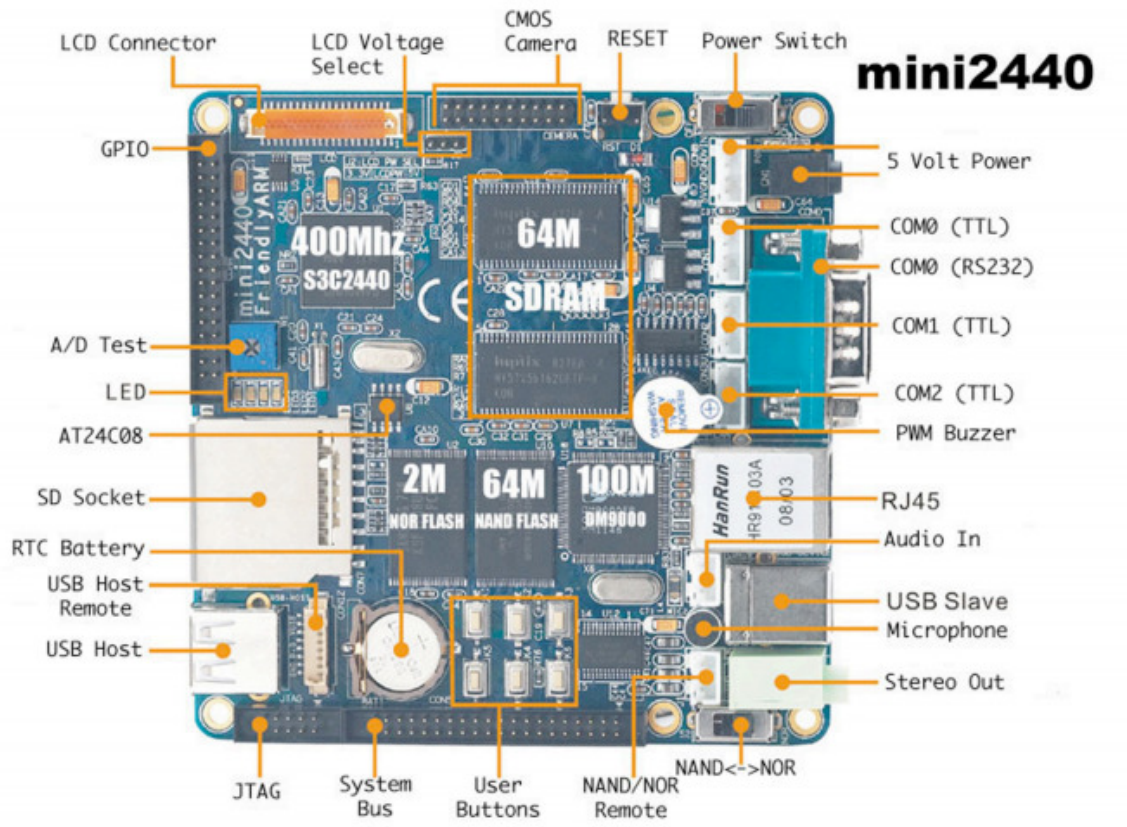

Fig 1 MINI2440

O sistema operacional foi o Linux 2.6.29. O mesmo já possuindo acesso a algumas interfaces, como serial, USB, áudio e outras. A entrada USB foi diversas vezes usada como entrada para um teclado para facilitar o desenvolvimento. A tela touchscreen usada é da mesma empresa que produz o kit, FriendlyArm, e possui .

Fig 2 Tela

7 Projeto

Antes de iniciar o projeto aconteceu etapa de estudos do Kit escolhido, um Mini2440 da empresa FriendlyArm. Nesta etapa foram abordadas tanto as funções do kit quanto o SO embarcado.

Logo no inicio aconteceu um imprevisto, as portas GPIO que usaríamos para os sensores e atuadores não estavam liberadas pelo SO embarcado. Chegamos à conclusão que deveríamos recompilar o Kernel e assim liberar as portas GPIO. Após essa etapa iniciamos os testes com as portas GPIO na CON4.

Iniciando a implementação verificamos que para a aplicação especifica o sensor não seria o ideal. Assim o recomendado foi um sensor com uma qualidade um pouco melhor ou receber os dados analógicos. Decidimos por receber dados analógicos, pois para a apresentação poderíamos usar um LM35 e também aproveitar os pinos ADC do Kit.

Com a nova etapa de estudo ADC, depois de muitos testes, verificou que dos quatro pinos AD somente o pino AIN0 estava liberado, e o mesmo já era usado para teste no próprio kit estando ligado em um trimpot. Com isso tivemos que alterar o código driver para liberar as outras portas AD.

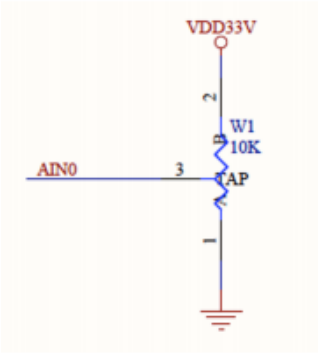

Fig 3 AIN0

Para a apresentação do projeto decidimos usar o buzzer e os leds do próprio kit. O buzzer do kit esta em curto com o pino 31 da CON4, então um buzzer externo seria acessado da mesma maneira. Os leds não fazem parte da CON4 e possuem um driver próprio para acesso, porem podem ser acessados da mesma forma que os pinos GPIO da CON4, pois também são GPIOs, sendo assim qualquer led externo seria acessado de forma idêntica.

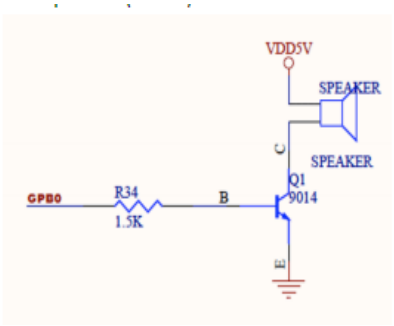

Fig 4 Buzzer

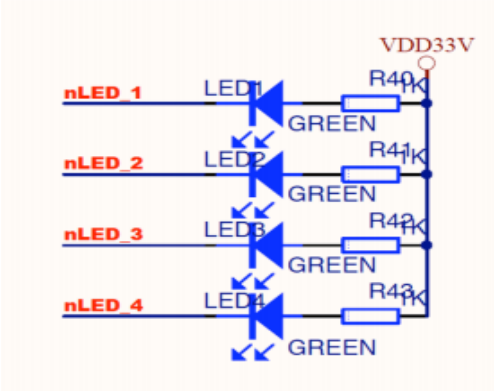

Fig 5 Leds

7.1 GPIO

O Kit conta com portas GPIO e dessas algumas são pinos na CON4 para realizar inteface, usando os SO que vem junto deve-se realizar alterações para liberar o acesso a esses pinos. Basicamente muitas das interfaces do kit usam estes pinos também, sendo assim o acesso pode ser feito da mesma forma.

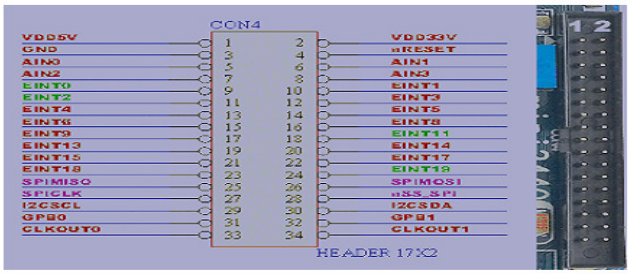

Fig 6 CON4

7.2 AD

O kit possui oito pinos AD sendo quatro usados na interface com o LCD, e as outras disponíveis. Dessas uma esta ligada a um trimpot, e as outras

três não são abertas pelo Driver. Para se ter acesso deve-se modificar o código do driver mini2440_ad.c e assim abrir as ADs desejadas.

7.3 Compilação

Para compilar os programas foi compilado em maquinas com o SO Ubuntu, para facilitar o desenvolvimento. Foi usado o ARM-Linux GCC 4.3.2. Esse é um arquivo tgs e recomentadamos extrai-lo na pasta USR. Após feito isso basta adicionar o caminho a pasta Bash, ou executar o comando: "export PATH=/usr/local/arm/4.3.2/bin:$PATH" . Em seguida execute "export CROSS=arm-linux-". Após isso basta somente ir até a pasta onde esta o código e compila-lo.

7.4 Envio

Para envio de dados existe alguma opções como USB, Serial e FTP. A usada no projeto foi por não precisar de outros softwares. Para isso bastava configurar a rede do Pc semelhante a do Kit. Também usávamos o comando “ifconfig eth0 192.168.1.237 netmask 255.255.255.0”, a rede assim ficava configura bansando apenas dar ftp 192.168.1.230.

8 CUSTOS

Material	Preço
KIT MINI2440	R$ 350,00
LM35	R$ 2,00
Materiais Diversos	R$ 10,00

Alguns componentes usados serão importados, por isso está incluso a importação da quantidade necessária, como também as taxas. Custos relacionados à energia, mão de obra e instrumentos não foram contabilizados, pois a universidade estará fornecendo

9 Códigos

9.1 Biblioteca GPIO

```
#ifndef GPIOLIB_H
#define GPIOLIB_H

#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

//Biblioteca para acesso de GPIO's
//20/11/2009
//Gean F. D. Chu
//Johnny W. Flauzino

void setDirection(char* porta, int direcao); //1 = output, e 0 =  input
void writeBit(int value ,char* porta);
int readBit(int porta);
int* readBits(int* bits, int size);
void openPort(int porta);
void openPorts(int* bits, int size);

FILE* fp;

void setDirection(char* porta, int direcao){

        char set_value[4];
        char comando[80];
        strcpy(comando, "/sys/class/gpio/gpio");
        strcat(comando, porta);
        strcat(comando, "/direction");
        if ((fp = fopen(comando, "rb+")) == NULL)
        {
                printf("Cannot open direction file.\n");
                exit(1);
        }
        rewind(fp);

        if(direcao == 1){

                strcpy(set_value,"out");
                fwrite(&set_value, sizeof(char), 3, fp);
                printf("...direction set to output\n");
        }
        else{
```

```
                strcpy(set_value,"in");
                fwrite(&set_value, sizeof(char), 2, fp);
                printf("...direction set to input\n");
        }
        fclose(fp);

}

void writeBit(int value, char* porta){



                        char set_value[4];
                        char comando[80];
                        strcpy(comando, "/sys/class/gpio/gpio");
                        strcat(comando, porta);
                        strcat(comando, "/value");

                        if(value == 1)
                                strcpy(set_value,"1");
                        else
                                strcpy(set_value,"0");

                        if ((fp = fopen(comando, "rb+")) == NULL)
                        {
                                printf("Cannot open value file.\n");
                                exit(1);
                        }

                        rewind(fp);
                        fwrite(&set_value, sizeof(char), 1, fp);
                        fclose(fp);
}

int readBit(int porta){

        char P[4];
        char get[1];
        char comando[80];
        int result;
        sprintf(P,"%d",porta);
        strcpy(comando, "/sys/class/gpio/gpio");
        strcat(comando, P);
        strcat(comando, "/value");

        if ((fp = fopen(comando, "rb+")) == NULL)
                {
                        printf("Cannot open value file.\n");
                        exit(1);
                }
```

```
	rewind(fp);
	fread(&get, sizeof(char), 1, fp);
	fclose(fp);
	result = (*get)-0x30;
	return result;
}

int* readBits(int* bits, int size){

	int i=0;
	int* vetor = malloc (sizeof(int)*size);
	for(i=0; i<size;i++){

		vetor[i]=readBit(bits[i]);
	}
	return vetor;

}


void openPort(int porta){

	char set_value[4];
	int size;
	char P[4];
	if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)
			{
				printf("Cannot open export file.\n");
				exit(1);
			}



	sprintf(P,"%d", porta);
	for(size=0;size<4;size++){

		if(P[size]==0)
			break;
	}
			rewind(fp);
			strcpy(set_value,P);
			fwrite(&set_value, sizeof(char), size+1, fp);
			fclose(fp);
}

void openPorts(int* bits, int size){

	int i=0;
	for(i=0;i<size;i++){
```

```
                openPort(bits[i]);
        }
}


#endif
```

9.2 Programa da Apresentação

```
#include <fcntl.h>
#include <linux/fs.h>
#include <errno.h>
#include <string.h>
#include "GPIOlib.h"

int main(void)
{
        int fd = open("/dev/adc", 0);
        float temp;
        float aux = 1.78;
        int toggle=1;
        int i=0;
        int value = -1;
        int portas[4]={38,32,33, 37};
        openPorts(portas, 4);
        setDirection("33", 1);
        setDirection("32", 0);
        setDirection("37", 1);
        setDirection("38", 1);
        if (fd < 0) {
                perror("Erro ao abrir ADC:");
                return 1;
        }

        for(;;) {

                char buffer[30];
                int len = read(fd, buffer, sizeof buffer -1);

                if (len > 0) {
                        buffer[len] = '\0';
                        sscanf(buffer, "%d", &value);
                        if((value)<(46)*294/100+5){
                                writeBit(0, "37");
                                writeBit(1, "38");
                                writeBit(0, "33");
                                printf("menor que 40\n");
```

```
                                        }
                                        else {
                                                if(value >(46)*294/100+5 && value
<(56)*294/100+5){
                                                        printf("entre 40 e 50\n");
                                                        if(toggle){
                                                                writeBit(1, "33");
                                                                writeBit(0, "38");
                                                                writeBit(1, "37");
                                                                toggle = 0;
                                                        }
                                                        else{
                                                                writeBit(0, "33");
                                                                writeBit(1, "38");
                                                                writeBit(1, "37");
                                                                toggle = 1;
                                                        }
                                                }
                                                else{
                                                        if(value>(56)*294/100+5){
                                                                printf("maior que
50\n");
                                                                writeBit(1, "33");
                                                                writeBit(0, "38");
                                                                writeBit(1, "37");

                                                        }
                                                }


                                                }
                                }
                                else {
                                perror("erro ai ler adc:");
                                return 1;
                                }
                                usleep(500000);
                        }

                        close(fd);
                        return 0;
                }
```

10 Conclusão

A implementação do projeto apresentou diversos problemas, uma vez que alguns drivers tiveram de ser refeitos, e o OS recompilado algumas vezes, o que gerou atrasos. Mas o objetivo foi concluído.

Assim o projeto apresentou sucesso em suas metas, agora partimos para a segunda etapa onde iremos usar o conhecimento adquirido e o kit já modificado para a implementação do programa do projeto Inova.

11 Referências

- http://findarticles.com/p/articles/mi_m3092/is_5_39/ai_60122380/

- http://www.friendlyarm.net/home

- www.arm.linux.org.uk

- http://www.avrfreaks.net/wiki/index.php/Documentation:Linux/GPIO