PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

Projeto Da Vinci

CURITIBA
2009

GEOVANE VINICIUS FERREIRA
JHONY KAESEMODEL PONTES
MARCELO EDUARDO MENEZES GLAUSER

Projeto Da Vinci

Proposta de trabalho apresentado ao curso de Engenharia de Computação (Turma U – Noturno) do Centro de Ciências Exatas e de Tecnologia da Pontifícia Universidade Católica do Paraná, como critério de avaliação do PA Microprocessadores II.

Orientador: Prof. Afonso Ferreira Miguel.

CURITIBA
2009

ÍNDICE

1- INTRODUÇÃO **4**

2- SISTEMAS EMBARCADOS **5**

3- LINUX EMBEDDED **6**

4- KIT FRIENDLYARM MINI 2440 ..... **7**

5- SENSOR LDR **10**

6- PASSOS DO PROJETO **11**

6.1. Familiarização com o kit e processador Samsung S3C2440 ..... **11**

6.2. Configuração de uma IDE **11**

6.2.1 Configuração de uma IDE (Ambiente Integrado de Desenvolvimento) – ECLIPSE (Linux) **11**

6.2.2 Configuração de uma IDE (Ambiente Integrado de Desenvolvimento) – Visual Studio (Windows CE) **12**

6.3 Configuração de um ambiente QT integrado com eclipse (Para forms) **12**

6.4. Dificuldades encontradas em configurar o ambiete (Sem Sucesso) **13**

6.5. Compilaçao por lina de comando usando Toolchain arm-linux-gcc **13**

6.6. Códigos de acesso interno do kit(para acender led, botoes e A/D) **13**

6.7. Compilação de Kernel 2.6.29 ..... **13**

6.8. Como Acessar as GPIO no linux **14**

6.9. Uso do Converso AD **14**

6.10. Trechos Do Codigo **16**

7- CONCLUSÃO **17**

8- ANEXOS **18**

ÍNDICE DAS FIGURAS

FIGURA 01: Kit FriendlyARM **8**

FIGURA 02: Kit FriendlyARM **9**

FIGURA 03: Entradas analógicas **14**

FIGURA 04: Esquematico Buzzer **.....15**

FIGURA 05: AD Interno**.....15**

1- INTRODUÇÃO

Este projeto teve como objetivo aprender a manipular um sistema embarcado, o qual é muito útil no dias atuais, o nosso sistema tem como proposta verificar a intensidade luminosa de um ambiente qualquer. Muito útil para o controle de estufas no crescimento de plantas, pois havendo uma grande intensidade de luz elas podem morrer ou tendo um nível muito baixo elas podem não crescer perfeitamente.

Para tal adquirimos o Kit FriendlyARM mini2440, o qual possui um display touch screen controlado por um processador ARM9 com Linux embarcado.

A obtenção da intensidade luminosa é dada através de um foto sensor LDR, o qual capta a intensidade de luz e manda o sinal para a entrada AD do Kit e este exibe na tela uma mensagem dizendo como esta a intensidade de luz no ambiente e também aciona os seguntes LEDs:

- Muita Luz – Acende o LED vermelho e aciona o buzzer
- Ambiente Normal – Acende o LED Verde
- Pouca Luz – Acende o LED amarelo
- Nenhuma Luz – Acende o LED vermelho e aciona o buzzer

2- SISTEMAS EMBARCADOS

Um sistema embarcado (ou sistema embutido) é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

Sistemas como PDAs são geralmente considerados sistemas embarcados pela natureza de seu hardware, apesar de serem muito mais flexíveis em termos de software. Fisicamente, os sistemas embarcados passam desde MP3 players à semáforos.

3- LINUX EMBEDDED

Embedded Linux é um sistema operacional Linux embarcado, que ao contrário do desktop e versões de servidor do Linux, este é projetado para dispositivos com recursos relativamente limitados, como telefones celulares e set-top boxes, e para o nosso caso no uso do Kit FriendlyARM mini2440 . Devido a preocupações como custo e tamanho, geralmente possuem dispositivos incorporados RAM muito menos e armazenamento secundário de computadores desktop, e são susceptíveis de utilização de memória flash em vez de um disco rígido. Desde dispositivos embarcados específicos em vez de servir a propósitos gerais, útil para termos controle sobre os port de entrada e saída do microprocessador.

4- KIT FRIENDLYARM MINI2440

O Kit Friendlyarm adquirido para o desenvolvimento deste projeto possui as seguintes configurações:

- **Processador** - Samsung S3C2440A, 400MHz, Max. 533Mhz;
- **SDRAM** - 64M SDRAM - 32bit DataBus - SDRAM Clock 100MHz;
- **Flash -** 64M Nand Flash, - 2M Nor Flash,BIOS installed;
- **LCD** - 4096 colores, 1024x768 pixels;
- **Interface –** Ethernet, Porta Serial, USB Host, SD Card Interface, Saida de Audio, Entrada de Audio, Seis Botoes, JTAG, 4 LEDs, Buzzer, Entrada A/D;
- **Oscillator Frequency** - 12MHz.

Figure 1 - Kit FriendlyARM

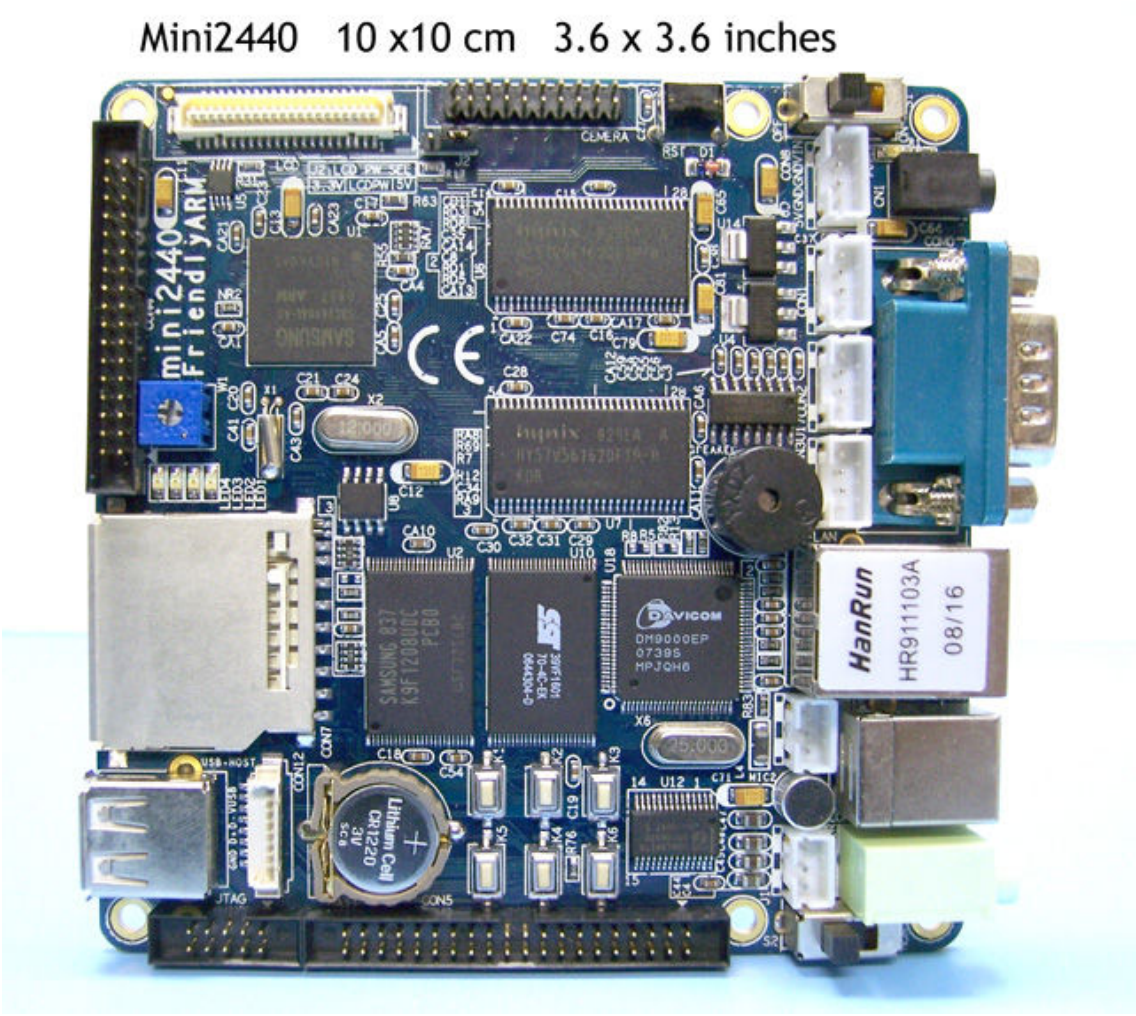

Figure 2 - Kit FriendlyARM

5- SENSOR LDR

LDR (do inglês Light Dependent Resistor ou em português Resistor Dependente de Luz) é um tipo de resistor cuja resistência varia conforme a intensidade de radiação eletromagnética do espectro visível que incide sobre ele.

Um LDR é um transdutor de entrada (sensor) que converte a (luz) em valores de resistência. É feito de sulfeto de cádmio (CdS) ou seleneto de cádmio (CdSe). Sua resistência diminui quando a luz é muito alta, e quando a luz é baixa, a resistência no LDR aumenta. Um multímetro pode ser usado para encontrar a resistência na escuridão ou na presença de luz intensa. Estes são os resultados típicos para um LDR padrão:

• Escuridão : resistência máxima, geralmente acima de 1M ohms;

• Luz muito brilhante : resistência mínima, aproximadamente 100 ohms.

6- PASSOS DO PROJETO

6.1. Familiarização com o kit e processador Samsung S3C2440

Ao adquirmos o kit Friendlyarm, primeiramente estudamos o seu processador Samsung S3C2440, para podermos ter noção sobre os ports de entrada, saída, timers, conversor AD e diversos outros recursos deste processador o qual possui um núcleo ARM920T, é microprocessador RISC de 16/32, para uso de diversos dispositivos moveis e aplicações em geral como no nosso kit, este tem uma boa relação custo eficaz, de baixo consumo e alto desempenho. Este processador inclui os seguintes componentes, a instrução separada 16 KB e 16 KB de cache de dados, a MMU lidar com o gerenciamento de memória virtual, TFT e STN controlador de LCD, flash NAND gerenciador de inicialização, gestor do sistema (lógica chip selecionar e controlador de SDRAM), 3 -- UART ch, 4-CH DMA, 4-temporizadores com ch PWM, portas I / O, RTC, 8-ch 10 bit ADC e interface de touch screen, interface da câmera, interface AC97 AudioCodec, CII-bus, IIS-bus, USB, dispositivo USB, SD Host e multimedia interface de cartão.

6.2 Uso de uma IDE .

Nosso Projeto foi Baseado em dois ambientes Linux e Windows para uso de uma IDE no Linux foi usado o Eclipse e no Windows foi usado o Visual Studio 2003.

6.2.1 Configuração de uma IDE (Ambiente Integrado de Desenvolvimento) – ECLIPSE (Linux)

Por Termos Usado um Processador ARM , em um ambiente linux o ideal seria usar uma interface IDE para programação, principalmente para ajudar no desenvolvimento do software, nesta hora encontramos algumas dificuldades, pois existem vários ambiente, então acabamos escolhemos usar o ambiente Eclipse o qual faz compilação Cruzada com o ARM-GCC, pois o código gerado tinha que ser compatível com nosso processador, somente o GCC ou o Eclipse não geram o código para funcionamento do software , mesmo tendo um sistema operacional gerenciando, o código binário tem que ser compatível com a arquitetura do processador.

6.2.1 Configuração de uma IDE (Ambiente Integrado de Desenvolvimento) – Visual Studio (Windows CE)

Como parte dos testes também tentamos usar uma IDE para programação do KIT usando o Sistema Operacional Windows CE 5.0 o qual tem uma ótima IDE o Visual Studio. Esta Feramente já é conhecida, pois usamos muito ela para plataforma x86, e ela apresenta também compatibilidade com processadores arm usando alguns módulos para tal, estes módulos podem ser encontrados no site da Microsoft, infelizmente não podemos usar a ultima versão do Visual Studio 2008 pois era necessário ter compatibilidade com a Framework 2 ou 3.5 e nossa imagem que foi disponibilizada pelo fabricante do Kit não oferecia compatibilidade , então tivemos que usar a versão 2005 a qual usa Framework 1.0 para geração do código, o Programa de teste e os Debugs funcionaram muito bem , mais não foi possível usar o dar continuidade nos testes pois o Windows Apresentava um pequeno problema o qual o Linux não tinha. Ele não era compatível com nenhum tipo de acesso a Hardware no caso as Portas mais especificamente, isso inviabilizou a utilização do Ambiente. Tendo por este o fim dos testes com esta IDE e também acabamos desistindo de usar o sistema Operacional Windows CE.

6.3. Configuração de um ambiente QT integrado com eclipse (Para forms)

A Escolha da Interface Gráfica foi baseada em uma ampla pesquisa, no qual decidimos usar a interface QT, ela é uma interface relativamente nova, desenvolvida pela Nokia, a qual usa uma framework poderosa que roda em quase todas as plataformas existentes ate em Linux com processadores ARM, por estas características acabamos optando por usar estas interface, existem varias documentações o qual deixa mais fácil para programação em Janelas , infelizmente não conseguimos usar pois tivemos problemas para configurar a IDE a qual descreveremos mais abaixo e sem isso ficou impossível usar o QT mesmo o QT tendo o Designer uma fermenta de modelagem de form muito boa. A nossa pesquisa foi baseada no site da plataforma, http://qt.nokia.com/.

6.4. Dificuldades encontradas em configurar o ambiete (Sem Sucesso)

A Primeira Dificuldade encontrada na configuração do Ambiente foi relacionada ao Windows CE, o qual não suportava Framework 2.0 ou superior, o que nos forçou a usar um ambiente mais antigo e depois mais tarde a desistência do ambiente.

A partir desta desistência nos focamos no sistema operacional Linux, e nossas dificuldades só começaram, infelizmente não existe uma boa documentação para configuração usando ide no Linux , pois grande parte dos programadores da plataforma usam editores simples de texto para fazer seus programas e nos não estávamos acostumados a usar tal recurso, a Principal dificuldade estava principalmente na configuração do eclipse com a complicação Cruzada com o GCC, pois ele não encontrava as variáveis ou as bibliotecas do GCC-ARM, e por este motivo ele não chegava nem a copilar, tivemos varias tentativas com varias versões do GCC - ARM, todas frustradas, chegamos a usar ate versões Trial de sistemas que funcionam Muito bem e muito usados por programadores embededd, mais não obtivemos sucesso.

Conforme o tempo ficava mais apertado tentávamos usar outras formas para complilar os programas que gerávamos, por exemplo um Hello Word. Mais não conseguíamos, mais uma hora conseguimos usar o gcc arm Linux usando linha de comando, uma luz no fim do túnel, mais em contra partida acabamos desistindo de usar o QT , pois a mesma necessitava usar uma Interface IDE para geração dos forms. A partir deste ponto acabamos usando o console para gerenciamento da nossa aplicação. O código era gerado em um Desktop com linux e era enviado por FTP para o KIT .

6.5. Compilaçao por lina de comando usando Toolchain arm-linux-gcc

Como a não Obtivemos sucedo com o uso de uma Ide acabamos usando compilação direto pelo console, usando o ToolChain´s Arm na verdade é o GCC que suporta processadores arm, no qual conseguimos gerar Objetos e executáveis os quais foram usados no projeto .

6.6. Códigos de acesso interno do kit(para acender led, botoes e A/D)

Como usamos uma distribuição Linux padrão o qual vem com o KIT, muitos dos recursos do kit estão desabilitado por padrão, e devido a isso tivemos que recompilar o Kernel do Sistema para que pudéssemos usar o GPIO, e outros acesso a hardware.

6.7. Compilação de Kernel 2.6.29

Como usamos uma distribuição Linux padrão o qual vem com o KIT, muitos dos recursos do kit estão desabilitado, e devido a isso tivemos que recompilar o Kernel do Sistema Operacional para que possamos usar o GPIO, e outros acesso a hardware. Nesta compilação acabamos adicionando novos drives de acesso os quais nos dão acesso ao hardware de entrada e saída e ainda os AD que tem no GPIO. Ao fazermos isso conseguimos ligar e desligar LEDs e ainda obter tenções do AD.

6.8. Como Acessar as GPIO no linux

O Acesso aos GPIO são feitos como uma leitura de um arquivo, assim o uso fica muito mais pratico e fácil de implementar.

6.9 Uso do Converso AD

Para o uso do AD tivemos que usar o GPIO, e conforme esquemático abaixo ele tem três entradas analógicas, e mais algumas saídas e entradas.

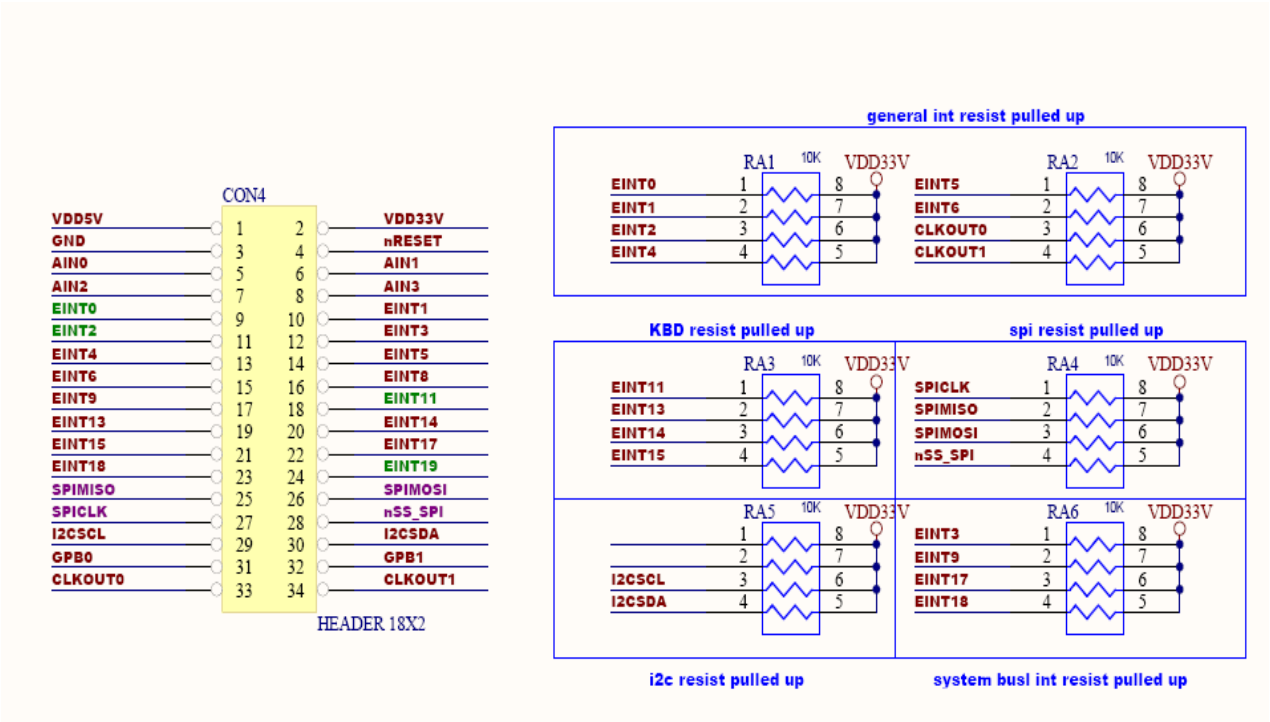

Figura 3 - Entradas analógicas

No nosso Projeto Ultilizamos a AIN1, para leitura do sensor de luminosidade . E para saida utlilizamos as portas EINT0 ao 1 para uso dos led´s .

O esquematico do Buzer esta abaixo. O qual tambem foi ultilizado.

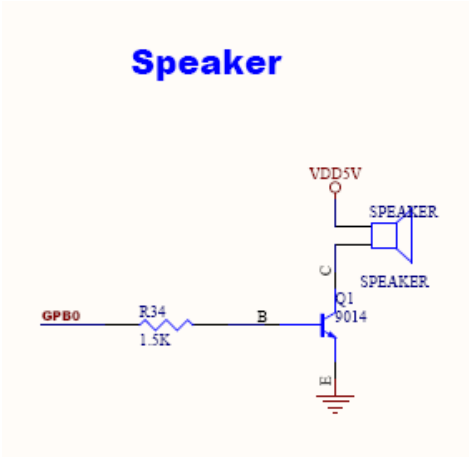

Figura 4 - Esquematico Buzzer

Aqui tambem tem um exemplo do AD Interno com um Potenciomentro.

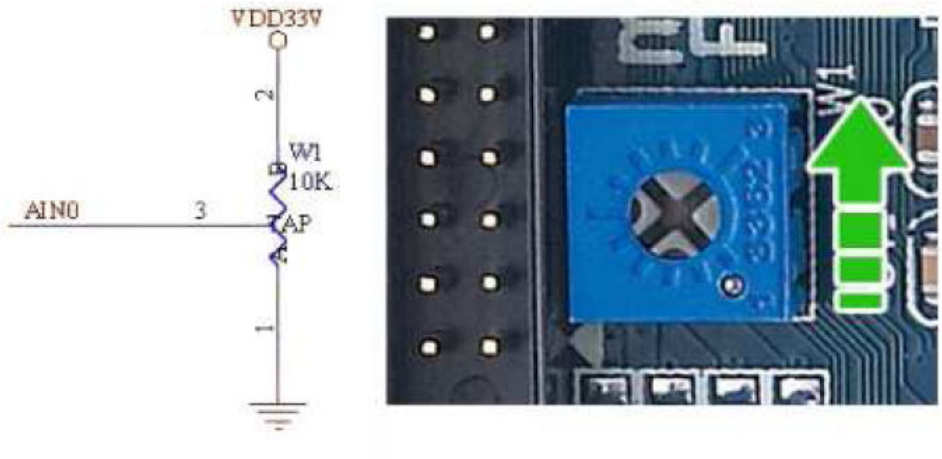

Figura 5 - AD Interno

6.10 - TRECHOS DO CODIGO.

Neste Trecho de código vimos como é lido o ad e ele tomando a decisão de muita luminosidade:

```
char buffer[30];
int len = read(fd, buffer, sizeof buffer -1);
if (len > 0)
{
        buffer[len] = '\0';
        int value = -1;
        sscanf(buffer, "%d", &value);
        //printf("Valor do AD: %d\n", value); //Mostra valor do AD
        if (value >= 0 && value <= 500 )
        {
                system("clear");
                Menu();
                printf("\n");
                printf("    ****************************************\n");
                printf("    *                                      *\n");
                printf("    *      AMBIENTE MUITO ILUMINADO        *\n");
                printf("    *                                      *\n");
                printf("    ****************************************\n");
                GPIO_F0();
                GPIO_F1_off();
                GPIO_F2_off();
                Buzzer_off()
```

Neste Trecho vimos como são setados os valores para acesso ao GPIO:

```
        //SETANDO O VALOR
        //Abre o arquivo em binário para leitura e escrita, armazena o
ponteiro em fp
        if ((fp = fopen("/sys/class/gpio/gpio162/value", "rb+")) ==
NULL)
        {
                printf("Nao eh possivel abrir o arquivo de valor.\n");
                exit(1);
        }

                }
```

7- CONCLUSÃO

O processo desenvolvimento do projeto ocorreu com conforme o esperado, porem com um pequeno atraso em relação ao cronograma esperado.

Um dos primeiros problemas foi conseguir compilar uma API em Linux para o kit de ARM, nesse procedimento foi perdido cerca de um mês, comprometendo assim o desenvolvimento de outras funções.

Devido não obtermos sucesso em gerar o executável para o kit, partimos para o desenvolvimento do programa por linhas de comando.

Outra dificuldade foi re-compilar o kernel do Linux para habilitar GPIO para podermos ter acesso aos ports de entrada e saída.

Apos conseguirmos re-compilar o kernel, obtivemos sucesso para finalizar o processo, com o kernel re-compilado tivemos acesso a todos os recursos do kit, como o conversor AD, o qual passamos para a parte de programação que não tivemos problemas, pois foi feito em linguagem C.

8- ANEXOS