Universidade Estadual de Campinas
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

EE641 (Laboratório de Eletrônica II)

Prof Hudson Zanin (hudson@dsif.fee.unicamp.br)
Prof João Carlos Martins de Almeida (joaocarlosma@gmail.com)

Roteiro para Experimento I

Conversor Analógico Digital tipo Modulação por Largura de Pulso

1. Objetivo:

Projetar e montar um Conversor Analógico Digital (CAD) com saída por Modulação de Largura de Pulso (PWM). Familiarizar com o Raspberry PI (RasPi). Conectar o RasPI ao conversor e realizar a leitura da grandeza analógica mostrando-a na tela.

2. Componentes:

- Kit RasPI
- Amplificador Operacional: LM324, encapsulamento DIP
- Resistores: 1 × 180 Ω, 1 × 560 Ω, 1 × 220 Ω, e diversos conforme projeto
- 1 Soquete de 14 pinos, terminal curto
- 1 Diodo zener 3,3 V – 1N4728
- 1 LED vermelho
- 1 push-button para conectar na placa
- 1 placa (PCB) para desenvolvimento
- 1 conector barra de pinos curtos
- Fios e estanho

3. Introdução:

Nosso curso utilizará o RasPI para o desenvolvimento dos experimentos de Eletrônica. Para começar, vamos projetar e montar um CAD que permita transferir um sinal analógico em tensão à memória digital do RasPI. Este último é um computador de baixo custo (USD 35,00) do tamanho de um cartão de crédito. Ele utiliza um processador ARM (operando a 700 MHz), roda Linux, e possui várias interfaces (HDMI, USB, Audio, GPIO [General Purpose Input/Output], RCA etc) que possibilitam aplicações flexíveis e ao mesmo tempo criativas envolvendo eletrônica, computação, automação, instrumentação etc.

4. CAD tipo PWM:

4.1 Desenhe abaixo um comparador, realizado através de um amplificador operacional, onde a entrada inversora é conectada a um gerador de sinal de onda triangular com amplitude pico a pico definida por V_{pp} e a entrada não-inversora conectada a um potencial v_A qualquer que varie dentro da amplitude do sinal triangular. Mostre e comente seu desenho no vídeo.

Figura do gerador PWM

4.2 Desenhe a forma de onda do sinal de tensão na saída (v_o) em função do tempo e calcule o *Duty Cycle* (DC = t_H / T), para os seguintes casos:

a) v_A = 10% de V_{pp}

b) v_A = 50% de V_{pp}

c) v_A = 80% de V_{pp}

Explique no seu vídeo o efeito do DC no valor de v_o

5. Projeto do CAD com saída PWM

Projete e monte um conversor que atenda as seguintes especificações: O CAD deve converter um sinal em tensão analógica com amplitude variando de 0 V até 5 V para um sinal digital tipo PWM com *Duty Cycle* variando de 0 % até 100 %, respectivamente. A amplitude do PWM deve variar de 0 V até no máximo 3,3 V.

OBS1: Consulte a folha de dados do RasPI e também do amplificador operacional antes de iniciar o projeto. Observe principalmente as restrições com relação à fonte de alimentação. Atente para o fato de ser necessário incluir um circuito limitador de tensão na saída do conversor PWM. Este circuito deve proteger a entrada digital do RasPi evitando danificá-la.

6. Parte Experimental e Apresentação

6.1 Monte o CAD e verifique seu funcionamento. Gere a tensão analógica (v_A) com uma fonte c.c. e a onda triangular, a 250 Hz, com o gerador de sinais. Monitore a saída (v_o) do CAD com o osciloscópio. Compare com os valores teóricos apresentados no item 4.2. Mostre o passo a passo no vídeo.

6.2 Programe o RasPI para realizar a leitura do sinal PWM através de um pino de entrada digital (GPIO). Mostre o programa comentado sucintamente no vídeo. Dica: Você pode reaproveitar qualquer programa exemplo que existe na biblioteca *WiringPI.*

6.3 Verifique experimentalmente o funcionamento do CAD interligado com o RasPI. Monitore ambos: a saída v_0 do CDA usando o osciloscópio e a leitura do sinal *Duty Cycle* realizada pelo RasPI. A tensão v_A deve ser gerada com a fonte d.c.. Faça a leitura do sinal analógico, com amplitude variando de 0 V até 5 V, pelo RasPI e mostre o valor na tela. Elabore um gráfico (x × y) da tensão de entrada (v_A) (medida por um voltímetro de referência) pela medida usando o RasPi. Para isto, varie a entrada de 0 até 5 V em intervalos de 0,5 V. Elabore um gráfico em separado indicando o erro obtido. Faça as observações e comparações entre o resultado experimental com relação ao previsto na teoria. Conclua e apresente os passos futuros.

6.4 Faça simulação do circuito CAD tal como item 6.1 no PSpice, LTSpice, Proteus 7.1 ou qualquer outro software da preferência do grupo e sua apresentação no vídeo permitirão ao grupo alcançar mais 2 (dois) pontos na Nota Final do vídeo (Ex. Vídeo ordinário vale 0-10, vídeo que apresentou e discutiu simulação 0-12). Dica: Aproveite a aula para realizar o

experimental e faça a simulação depois. A simulação pode ser apresentada em um vídeo separado. Só não se esqueça de subir tudo que fizerem no Blog da Turma.

6.5 Além dos 2 pontos extras da simulação do experimental, o grupo pode obter mais 1 ponto extra ao fazer o LED piscar (Vide detalhes no anexo 1).

6.6 DICA: Monte uma *playlist* no YOUTUBE ao divulgar seu vídeo.

Apresente o projeto completo (com esquemático, lista de material e software [comentado]) incluindo gerador PWM, circuito de proteção entre gerador PWM e entrada digital do RasPI no seu Vídeo para conclusão do vídeo.

7. Bibliografia

7.1 http://www.pcmag.com.br/us/article2/0,2817,2407058,00.asp

7.2 http://www.raspberrypi.org/

7.3 http://www.seucurso.com.br/index.php?option=com_content&view=article&id=232:como-acessarremotamente-a-interface-grafica-do-raspberry-pi&catid=914:raspberry-pi&Itemid=74

7.4 http://wiringpi.com

7.5 F. Fruett, Notas de aula, EE530, http.www.dsif.fee.unicamp.br/~fabiano/EE530/EE530.htm

Ganhe 1 ponto extra executando a tarefa do LED piscar até o final implementando o *push-button*

Configure o Rapberry PI, seguindo os passos indicados no documento: INSTRUÇÕES PARA CONFIGURAÇÃO DO RASPBERRY PI fornecido na página do curso de EE641.

Introdução à programação no RasPI:

Executada a instalação da biblioteca WiringPi do item *"Instalação da biblioteca WiringPi:"* do manual de configuração, vamos explorar os códigos de exemplo que a biblioteca fornece. No terminal, navegamos até a pasta onde foi instalado o wiringPi através do comando "*cd*": digite "*cd wiringPi*", note que o caminho do terminal foi alterado para algo do tipo: `pi@raspberrypi1 ~/wiringPi $`, isto mostra a "pasta" em que estamos. O comando "*ls*" exibe o conteúdo desta pasta, executando-o, teremos algo como:

```
pi@raspberrypi1 ~/wiringPi $ ls
build           devLib    gpio     People  projetos    wiringPi
COPYING.LESSER  examples  INSTALL  pins    README.TXT
pi@raspberrypi1 ~/wiringPi $
```

Entraremos agora na pasta *examples* através do comando cd, basta digitar "*cd examples*". Executamos novamente o comando *ls* e teremos uma relação com os arquivos presentes dentro da pasta *examples*:

```
pi@raspberrypi1 /wiringPi/examples $ ls
blink12.c       COPYING.LESSER  isr-osc         PiFace        serialTest.c
blink12drcs.c   delayTest.c     isr-osc.c       PiGlow        servo.c
blink8.c        ds1302.c        lcd-adafruit.c  pwm.c         softPwm.c
blink.c         Gertboard       lcd.c           q2w           softTone.c
blink.rtb       header.h        Makefile        README.TXT    speed.c
blink.sh        isr             nes.c           rht03.c       wfi.c
clock.c         isr.c           okLed.c         serialRead.c
```

Neste momento, iremos trabalhar com os programas (em linguagem c) "*blink.c*" (que atua na saída digital fazendo piscar um LED) e, posteriormente, iremos implementar um programa para controlar o LED através do acionamento de um botão. Para visualizar o arquivo "*blink.c*" usaremos o editor de texto "*nano*", um

simples editor executável através da linha de comando. Digite "*sudo nano blink.c*" e note que o programa abriu na própria linha de comando:

```
  GNU nano 2.2.6                File: blink.c

 *    You should have received a copy of the GNU Lesser General Public License
 *    along with wiringPi.  If not, see <http://www.gnu.org/licenses/>.
 ***********************************************************************
 */

#include <stdio.h>
#include <wiringPi.h>

// LED Pin - wiringPi pin 0 is BCM_GPIO 17.

#define LED     0

int main (void)
{
  printf ("Raspberry Pi blink\n") ;

  wiringPiSetup () ;
  pinMode (LED, OUTPUT) ;

  for (;;)
  {
    digitalWrite (LED, HIGH) ;  // On
    delay (500) ;               // mS
    digitalWrite (LED, LOW) ;   // Off
    delay (500) ;
  }
  return 0 ;
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Note que neste programa está sendo usada a biblioteca "*wiringPi.h*". As funções disponíveis nesta biblioteca estão listadas em http://wiringpi.com/reference/. Observe o funcionamento do programa, ele configura a porta 0 (chamada de LED) como saída através da função "*pinMode*" e, então, executa indefinidamente a rotina de escrever um 1 lógico (3.3V) na porta 0, aguarda um "*delay*" de 500 ms, escreve 0 na mesma porta e repete o processo. A relação dos pinos do conector (de nossa placa) com as portas usadas pela biblioteca é mostrada a seguir:

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Observe que a porta 0 do *WiringPi* (chamada de LED, em nosso programa) corresponde ao pino 11 do conector do RasPi.

Uma vez editado o programa, tecle CTRL-X para sair do editor e voltar à linha de comando. Se alguma alteração tiver sido feita, o programa perguntará se deseja salvá-la. Novamente para a linha de comando, vamos compilar o programa que acabamos de visualizar/alterar. Para tal, usaremos o compilador GCC do sistema operacional. A sintaxe a ser escrita na linha de comando será a seguinte:

"sudo gcc –o blink blink.c –lwiringPi -lpthread", onde "*blink*" é o nome do programa executável a ser criado e blink.c é o arquivo de texto com o código em C. Se a compilação ocorrer sem problemas, teremos nosso arquivo executável. Mas antes devemos instalar nosso LED na placa. Ligue um LED em série com um resistor limitador de corrente (como calcular o valor? Lembre-se que o LED é um diodo e apresenta uma queda de tensão, quando em condução, de aproximadamente 1V a 1.7V e consome aproximadamente 5mA). Montamos então o resistor em série com o LED (atenção para a polaridade do LED) no pino do RasPi (porta 0 do WiringPi corresponde ao pino 11). Conferida a instalação, podemos executar nosso programa através do comando "sudo ./blink". Observe o LED piscando. Para parar a execução do programa, tecle CTRL+C.

Agora, você deve escrever um programa para a leitura de um dado digital (lembre-se que o RasPi trabalha a 3.3V e não é tolerante a tensões acima de 3.3V). Implemente um botão (push-button) que, quando pressionado, acenda o LED. Dica: use a função "*digitalRead*" do wiringPi. Não se esqueça que quando trabalhamos com entradas digitais, é necessário o uso de resistores de pull-up/down, dependendo da lógica de acionamento. Os resistores de pull-up/down são responsáveis por manter a tensão em determinado nível lógico quando a chave/botão estiver desconectado. No entanto, o RasPi possui resistores internos configuráveis, atente para a função "*PullUpDnControl*".

Bom divertimento!