Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Sistemas Elétricos de Automação e Energia
ENG10032 Microcontroladores

Roteiro de Laboratório 3
***General Purpose I/O* - GPIO**

Prof. Walter Fetter Lages

20 de agosto de 2018

1 Objetivo

O objetivo deste laboratório é explorar as portas de I/O da Galileo Gen2 e compreender os multiplexadores e resistores de *pull-up* ou *pull-down* que precisam ser configurados para acessar os pinos de I/O disponíveis no conector de *shield* da Galileo.

2 Fundamentação Teórica

Cada pino do conector de *shield* da Galileo possui várias funções. Para usar cada pino é necessário primeiro configura-lo para função em que será utilizado. Para tanto, existem vários multiplexadores que devem ser configurados para rotear o sinal de forma adequada à função com a qual se deseja utilizar o pino. Adicionalmente, podem ser configurados resistores de *pull-up* ou *pull-down* nos pinos.

Os pinos `IO0-IO6` e `IO9-IO13` quando configurados como GPIO são comandados diretamente pelo Quark X1000. Os demais são gerados através de expansores de GPIO PCAL9535A e portanto possuem uma latência maior. Os pinos `IO14` a `IO19` são compartilhados com o conversor A/D e portanto tem capacitores de 22 nF acoplados, o que afeta o desempenho destes pinos no modo GPIO.

Os multiplexadores que controlam as funções dos pinos do conector de *shield* também são comandados por portas GPIO. Ou seja, algumas portas GPIO estão disponíveis no conector de *shield* e outras são usadas internamente na Galileo para controlar os multiplexadores e *buffers*. Para tanto, a Galileo Gen2 possui 3 expansores de GPIO PCAL9535A[1] conectados ao Quark X1000 através do barramento

[1] Na versão original da Galileo a configuração dos multiplexadores é diferente.

I2C nos endereços `0x25` `0x26` e `0x27`.

O acesso às portas de GPIO, tanto as que são conectadas aos pinos de I/O da Galileo quanto as que são usadas para configurar os multiplexadores, pode ser feito no espaço do usuário através de pseudo-arquivos no diretório `/sys/class/gpio`.

Para usar um pino de I/O no conector de *shield* da Galileo Gen2, é necessário configurar:

1. O(s) multiplexador(es) que conectam o pino no conector ao Quark ou ao *chip* na Galileo que provê a funcionalidade desejada
2. A direção do *buffer* associado ao sinal
3. Se um resistor de *pull-up* ou *pull-down* de $22\text{k}\Omega$ deve ser conectado ao pino

Além disso, se o sinal associado ao pino que se deseja utilizar é digital, é necessário configurar também a sua direção. Note que deve-se configurar independentemente a direção do *buffer* e a direção do pino de GPIO.

O mapeamento de qual porta GPIO controla qual sinal disponível no *shield* da Galileo Gen2 e quais portas GPIO controlam o roteamento dos sinais pode ser obtido através da análise do diagrama esquemático da placa [1]. É importante perceber que o arquivo com mapeamento de I/O da Galileo disponível no site da Intel é válido apenas para a Galileo e não para a Galileo Gen2. Portanto, não é útil para as placas existentes no laboratório.

Uma tabela com este mapeamento para a Galileo Gen2 está disponível no apêndice B de [2]. Outra versão da tabela está disponível em `<http://moodle.ece.ufrgs.br>`.

2.1 Configuração dos Pinos do *Shield* para Uso Como GPIO

Para configurar pino `IO13`, por exemplo, como saída digital, pode-se verificar, através da tabela de mapeamento de pinos de I/O, que deve-se configurar os `gpio46` e `gpio30` em nível lógico baixo, enquanto o `gpio31` controla se será usado ou não o resistor de *pull-up* ou *pull-down* e, finalmente o `gpio7` controla o sinal que aparecerá no pino `IO13`.

Para configurar um determinado `gpio`, é necessário primeiro exportar a porta. Isto é feito escrevendo-se o número (em ASCII) da porta GPIO em `/sys/class/gpio/export`. Por exemplo, usando comandos do *shell*:

```
echo -n "46" > /sys/class/gpio/export
```

Com isso, surgirá um diretório correspondente à porta. No caso, `/sys/class/gpio/gpio46`.

Quando a porta de I/O não for mais utilizada, ela deve ser "desexportada" escrevendo-se o número da porta no arquivo `/sys/class/gpio/unexport`.

A direção da porta é controlada escrevendo-se `"in"` ou `"out"` em `/sys/class/gpio/gpioXX/direction`, onde `XX` é o número da porta:

```
echo -n "out" > /sys/class/gpio/gpio46/direction
```

As `gpio64`, `gpio66`, `gpio68`, `gpio70`, `gpio72`, `gpio74`, `gpio76`, `gpio77`, `gpio78` e `gpio79` (todas as de número igual ou maior do que 64) não possuem o pseudo-arquivo `direction`, pois são sempre saída já que são usadas apenas para controlar multiplexadores e *buffers*.

Pode-se escrever ou ler a porta GPIO, escrevendo-se ou lendo-se em `/sys/class/gpio/gpioXX/value`:

```
echo -n "0" > /sys/class/gpio/gpio46/value
```

ou

```
cat /sys/class/gpio/gpio46/value
```

Para as portas que controlam multiplexadores ou direção dos *buffers* também é possível escrever `"low"` ou `"high"` no pseudo-arquivo `direction`. Isso é equivalente a configurar simultaneamente `direction` para `"out"` e `value` para `"0"` ou `"1"`, respectivamente.

2.2 Ajuste das Permissões

Por *default*, os arquivos em `/sys/class/gpio` só podem ser escritos pelo superusuário, embora alguns possam ser lidos por usuários comuns. Isso iria requerer que todos os programas que usassem GPIO tivessem que executar com privilégios de superusuário, o que não é uma boa idéia por questões de segurança.

Para permitir o acesso de usuários comuns à portas de GPIO selecionadas, a configuração destas e o ajuste das permissões será feito através dos *scripts* de inicialização do Linux, que executam com privilégios de superusuários.

Será criado o grupo `gpio` e as permissões serão ajustadas para que os usuários membros deste grupo possam acessar os arquivos adequados em `/sys/class/gpio`.

O *script* mostrado na listagem 1 configura o pino IO13 como saída e ajusta as permissões para que possa ser controlado por um programa de um usuário normal que pertença ao grupo `gpio`. Note que os comentários no início do *script* **não são** meros comentários, mas são interpretados pelos sistema de instalação do *script* e portanto são necessários.

Listagem 1: *Script* de inicialização.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          eng10032lab03_1
# Required-Start:
# Should-Start:
# Required-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Configures GPIO for Lab 03.
# Description:       Configures GPIO for Lab 03.
### END INIT INFO

case "$1" in
    start|restart|force-reload)
        if [ ! -d /sys/class/gpio/gpio46 ] ; then
            echo -n "46" > /sys/class/gpio/export
        fi
        echo -n "out" > /sys/class/gpio/gpio46/direction
        echo -n "0" > /sys/class/gpio/gpio46/value

        if [ ! -d /sys/class/gpio/gpio30 ] ; then
            echo -n "30" > /sys/class/gpio/export
        fi
        echo -n "out" > /sys/class/gpio/gpio30/direction
        echo -n "0" > /sys/class/gpio/gpio30/value

        if [ ! -d /sys/class/gpio/gpio7 ] ; then
            echo -n "7" > /sys/class/gpio/export
        fi
        echo -n "out" > /sys/class/gpio/gpio7/direction
        chgrp gpio /sys/class/gpio/gpio7/value
        chmod g+rw /sys/class/gpio/gpio7/value
        ;;
    stop)
        echo -n "in" > /sys/class/gpio/gpio7/direction
        echo -n "7" > /sys/class/gpio/unexport
        echo -n "1" > /sys/class/gpio/gpio30/value
        echo -n "30" > /sys/class/gpio/unexport
        echo -n "46" > /sys/class/gpio/unexport
        ;;
    status)
        ls -d /sys/class/gpio/gpio*
        ;;
```

```
44      *)
45          echo -n "Usage: $0 "
46          echo "{start|stop|restart|force-reload|status}"
47          exit 1
48          ;;
49  esac
50
51  exit 0
```

3 Experimentos

3.1 Configuração da Galileo para Usar GPIO

1. Digite o *script* mostrado na listagem 1 em um arquivo chamado, por exemplo, `eng10032lab03_1` e transfira-o para a Galileo como o comando:

   ```
   scp eng10032lab03_1 <galileo>:
   ```

   onde `<galileo>` é o nome da sua Galileo.

2. Logue-se na Galileo como superusuário e crie o grupo `gpio` com o comando:

   ```
   groupadd -r gpio
   ```

3. Inclua o seu usuário no grupo `gpio` com o comando:

   ```
   groupmems -g gpio -a <login>
   ```

   onde `<login>` é o seu `login` na Galileo.

4. Copie o arquivo `eng10032lab03_1` para o diretório `/etc/init.d` na Galileo com o comando:

   ```
   cp ~/eng10032lab03_1 /etc/init.d
   ```

5. Torne o *script* executável com o comando:

   ```
   chmod +x /etc/init.d/eng10032lab03_1
   ```

6. Configure o Linux para chamar o *script* durante a inicialização com o comando:

```
update-rc.d eng10032lab03_1 defaults
```

7. Reinicialize a Galileo com o comando

```
reboot
```

3.2 Uso dos Pinos de GPIO

8. Conforme pode ser visto na folha 23 do diagrama esquemático da Galileo Gen2 [1], há um LED conectado ao pino `IO13` do *shield*. Fisicamente, este LED está ao lado do conector USB *host*.

   Como já descrito na seção 2.1, o pino `IO13` (onde está conectado o LED) é controlado pela porta `gpio7`, que é roteada para este pino colocando-se a `gpio46` em nível lógico baixo. A `gpio30` controla a direção do *buffer* associado (nível lógico baixo para saída e alto para entrada). Neste caso, é irrelavante o estado do resistor de *pull-up* ou *pull-down*, controlado por `gpio31`.

9. Verifique que o *script* da listagem 1 configura o pino `IO13` como GPIO e como saída e portanto capaz de acionar o LED.

10. Digite (em um diretório chamado `blink`, por exemplo) e compile o programa mostrado na Listagem 2. O `Makefile` é mostrado na Listagem 3.

11. Transfira o programa para a Galileo e execute-o para verificar se o LED realmente pisca.

12. Modifique o programa da Listagem 2 e o *script* de inicialização para piscar um LED conectado no pino `IO3`.

13. Utilizando o LED do *Grove Starter Kit*, mostrado na Figura 1 e o *shield* base mostrado na Figura 2, teste o programa desenvolvido em 12.

14. Faça um programa para ler uma entrada digital no pino `IO2` e acender e apagar o LED conforme o valor lido.

15. Modifique o *script* de inicialização para permitir o acesso ao pino `IO2`, instale-o e reinicialize a Galileo.

16. Utilizando o *push-button* do *Grove Starter Kit*, mostrado na Figura 3, teste o programa.

Listagem 2: Programa para piscar o LED na Galileo Gen2

```
1  #include <fcntl.h>
2  #include <unistd.h>
3
4  int main(int argc,char *argv[])
5  {
6          char state='0';
7          int fd;
8
9          fd=open("/sys/class/gpio/gpio7/value",O_WRONLY);
10
11         for(;;)
12         {
13                 lseek(fd,0,SEEK_SET);
14                 write(fd,&state,sizeof state);
15                 sleep(1);
16                 state^='0'^'1';
17         }
18
19         close(fd);
20
21         return 0;
22 }
```

Figura 1: *Grove-LED Socket Kit*.

Listagem 3: Arquivo `Makefile` para o programa `blink`.

```
1  TARGET=blink
2  SRCS=$(TARGET).c
3
4  FLAGS=-O2 -Wall -g -MMD
5  INCLUDE=
6  LIBDIR=
7  LIBS=
8
9  CC=$(CROSS_COMPILE)gcc
10 CFLAGS=$(FLAGS) $(INCLUDE)
11 LDFLAGS=$(LIBDIR) $(LIBS)
12
13 all: $(TARGET)
14
15 $(TARGET): $(SRCS:.c=.o)
16         $(CC) -o $@ $^ $(LDFLAGS)
17
18 %.o: %.c
19         $(CC) $(CFLAGS) -c -o $@ $<
20
21 -include $(SRCS:.c=.d)
22
23 clean:
24         rm -f *~ *.bak *.o *.d
25
26 distclean: clean
27         rm -f $(TARGET)
```

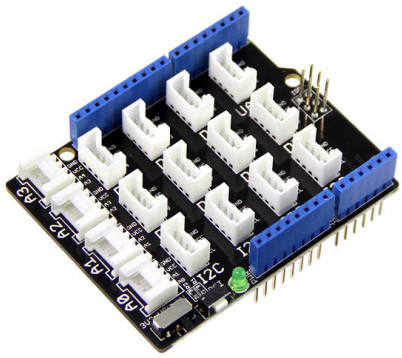

Figura 2: *Shield* base.

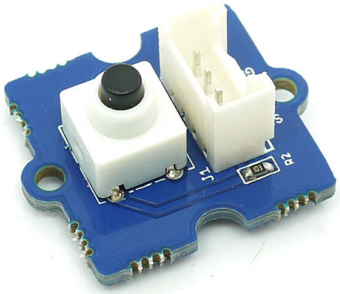

Figura 3: *Push-button*.

17. Na Galileo, como super usuário, remova o *script* de configuração do GPIO, para que não interfira nos próximos laboratórios, com o comando (remova também outros *scripts* que foram criados neste laboratório):

```
update-rc.d -f eng10032lab03_1 remove
```

Referências

[1] Intel Corporation. *Intel Galileo Gen2 Board Schematic*, 2014.

[2] M. C. Ramon. *Intel Galileo and Intel Galileo Gen 2: API Features and Arduino Projects for Linux Programmers*. Apress Media, New York, 2014.