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

Roteiro de Laboratório 7
Conversor Analógico/Digital (ADC)

Prof. Walter Fetter Lages

1 de outubro de 2019

1 Objetivo

O objetivo deste laboratório é explorar o conversor A/D da Galileo.

2 Fundamentação Teórica

A Galileo Gen 2 possui um conversor A/D de 10 bits e 8 canais, dos quais 6 estão disponíveis no conector de *shield*. Estes canais são implementados pelo *chip* ADC108S102 da Texas Instruments, que está conectado diretamente ao Quark X1000 através do barramento SPI 0, *chip select* 0.

As entradas do conversor A/D estão mapeadas nos pinos `IO14-IO19` do conector de *shield*.

Como nos outros casos, é necessário configurar os pinos do conector de *shield* para ler as medidas corretas do conversor A/D. Note, em especial, que o resistor de *pull-up*/*pull-down* associado ao pino deve ser configurado em *off* para não influenciar a medida.

A resolução do A/D é de 10 bits, mas os dados são escalonados para 12 bits para serem apresentados na interface do espaço do usuário. Isto é feito para manter a compatibilidade do *driver* com o ADC128S102 que é uma variante de 12 bits do mesmo *chip* e com a primeira geração da Galileo, que utiliza o *chip* AD7298, também de 12 bits.

O conversor A/D pode ser usado em dois modos: modo de disparo único e modo contínuo. No modo de disparo único é feita apenas uma conversão por requisição, enquanto no modo contínuo, são feitas conversões em sequência em todos os canais habilitados.

2.1 Modo de Disparo Único

O acesso aos canais do conversor A/D através do espaço do usuário é feito através de arquivos no diretório `/sys/bus/iio/devices/iio:device0`:

`in_voltageN_raw`: Valor bruto da medida em ASCII.

`in_voltageN_scale`: Multiplicador para converter o valor bruto da medida para mV.

onde `N` é o número do canal do conversor A/D.

2.2 Modo Contínuo

Os dados brutos são lidos do arquivo `/dev/iio:device0` em binário. A configuração do *buffer* para receber os dados e dos canais que serão amostrados também é feita através de arquivos no diretório `/sys/bus/iio/devices/iio:device0`:

`buffer/length`: Número de conjuntos de amostras do *buffer*.

`buffer/enable`: Habilita/desabilita as conversões no modo contínuo.

`scan_elements/in_timestamp_en`: Habilita o *timestamp* dos dados.

`scan_elements/in_timestamp_index`: Índice da posição do *timestamp* no *buffer*.

`scan_elements/in_timestamp_type`: Formato com o qual o *timestamp* é armazenado no *buffer*.

`scan_elements/in_voltageN_en`: Habilita a amostragem do canal `N`.

`scan_elements/in_voltageN_index`: Índice da posição do canal `N` no *buffer*.

`scan_elements/in_voltageN_type`: Formato com o qual os dados do canal `N` são armazenados no *buffer*.

`trigger/current_trigger`: Configura o *trigger* a ser usado para as disparar as conversões.

onde `N` é o número do canal do conversor A/D.

O formato dos dados é descrito por uma *string* no formato:

`<endianness>:<sinal><precision>/<size>>><alignment>`

onde:

<endianness>: representa a *endianness* do dado, `be` para *big endian* e `le` para *low endian*

<sinal>: indica se o valor é com sinal (`s`) ou sem sinal (`u`)

<precision>: indica o número de bits com informação relevante

<size>: indica o número de bits usados para armazenar o dado

<alignment>: indica o alinhamento dos bits de informação

Assim, `be:u12/16>>0` significa que os dados estão em um formato *big endian*, sem sinal, com 12 bits de informação, armazenados em 16 bits e alinhados no bit 0.

Para realizar conversões no modo contínuo, deve-se habilitar os canais desejados, configurar o tamanho do *buffer* (número de amostras) e habilitar o *buffer*. Os pseudo-arquivos com os índices são de leitura apenas e somente os canais habilitados são realmente inseridos no *buffer*.

As conversões são disparadas por um *trigger*. Os *triggers* disponíveis aparecem em `/sys/bus/iio/devices` quando o módulo do *kernel* correspondente é carregado. Estes módulos não são carregados no *boot* da Galileo. É necessário carrega-los explicitamente com os comandos `modprobe` ou `insmod`. Os *triggers* disponíveis na Galileo Gen2 são:

iio-trig-sysfs: totalmente baseado em *software*. Permite disparar por *software* cada amostragem, como no modo de disparo único, mas com a API do modo contínuo.

iio-trig-hrtimer: baseado no temporizador HPET. permite configurar a frequência com que será feita a amostragem.

Quando os módulos dos *triggers* são corregados, surgem os seguintes arquivos no diretório `/sys/bus/iio/devices`:

iio_sysfs_trigger/add_trigger : Usado para criar um *trigger* do tipo `iio_sysfs_trigger`, ao se escrever um número inteiro no arquivo.

iio_sysfs_trigger/remove_trigger : Usado para remover um *trigger* do tipo `iio_sysfs_trigger`, ao se escrever um número inteiro no arquivo.

iio_hrtimer_trigger/add_trigger : Usado para criar um *trigger* do tipo `iio_hrtimer_trigger`, ao se escrever um número inteiro no arquivo.

iio_hrtimer_trigger/remove_trigger : Usado para remover um *trigger* do tipo `iio_hrtimer_trigger`, ao se escrever um número inteiro no arquivo.

Para cada *trigger* é criado um diretório `/sys/bus/iio/devices/trigger<n>`, onde `<n>` é um número crescente a partir de 0. Note que `<n>` não é o número que foi escrito no arquivo `add_trigger` para criação do *trigger*, mas um número que é incrementado a cada *trigger* criado. O conteúdo deste diretório depende do tipo do *trigger*:

name: nome do *trigger*, usado para configurar o *trigger* a ser usado.

trigger_now: qualquer escrita neste arquivo dispara o *trigger*. Só existe para *triggers* do tipo `iio_sysfs_trigger`.

frequency: configura a frequência do *trigger*. Só existe para *triggers* do tipo `iio_hrtimer_trigger`.

Quando é usado um *trigger* do tipo `iio_sysfs_trigger`, após a habilitação do *buffer*, cada escrita no arquivo `trigger_now` realiza uma amostragem. Quando é usado um *trigger* do tipo `iio_hrtimer_trigger`, a habilitação do *buffer* inicia a amostragem com a frequência programada no arquivo `frequency`.

As conversões são paradas desabilitando-se o *buffer* e removendo-se o *trigger*.

3 Experimentos

1. Logue-se na Galileo como superusuário, crie o grupo `adc` e inclua o seu usuário neste grupo.

2. Verifique no mapa de configuração dos pinos da Galileo Gen2 como configura-la para usar o `ADC_A0`. Note que, para não alterar o valor medido pelo conversor A/D, não deve haver resistor de *pull-up* nem de *pull-down* no pino. Vide tabela no Moodle em `<http://moodle.ece.ufrgs.br>`.

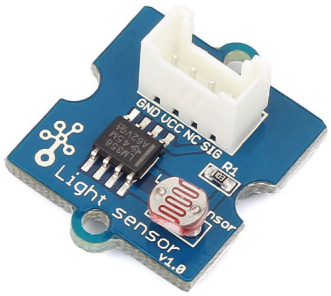
3. Faça um *script* de inicialização para configurar o uso do `ADC_A0` em modo de disparo único. Configure permissões de leitura para o grupo `adc` nos arquivos `in_voltage0_raw` e `in_voltage0_scale`.

4. Configure o *script* para executar na incialização e reinicialize a Galileo.

5. Faça um programa que fique em *loop* lendo o `ADC_A0` no modo de disparo único e mostrando os valores em Volts.

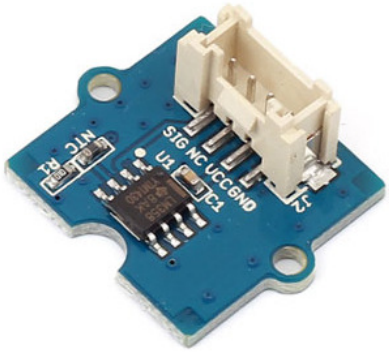
6. Teste o programa com o *Grove Rotary Angle Sensor*, mostrado na Figura 1(a). Apesar do nome pomposo, trata-se apenas de um potenciômetro de 10 kΩ conectado a Vcc, GND e ao pino de sinal, como mostra o esquemático da Figura 2.

(a) Rotary Sensor.



(b) Light Sensor.

(c) Sound Sensor.



(d) Temperature Sensor.

Figura 1: *Grove Sensors*.

7. Varie a posição do eixo do potenciômetro e verifique se a tensão medida corresponde à posição do eixo. Anote os valores máximo e mínimo de tensão, obtidos com o potenciômetro em cada um dos extremos, respectivamente.

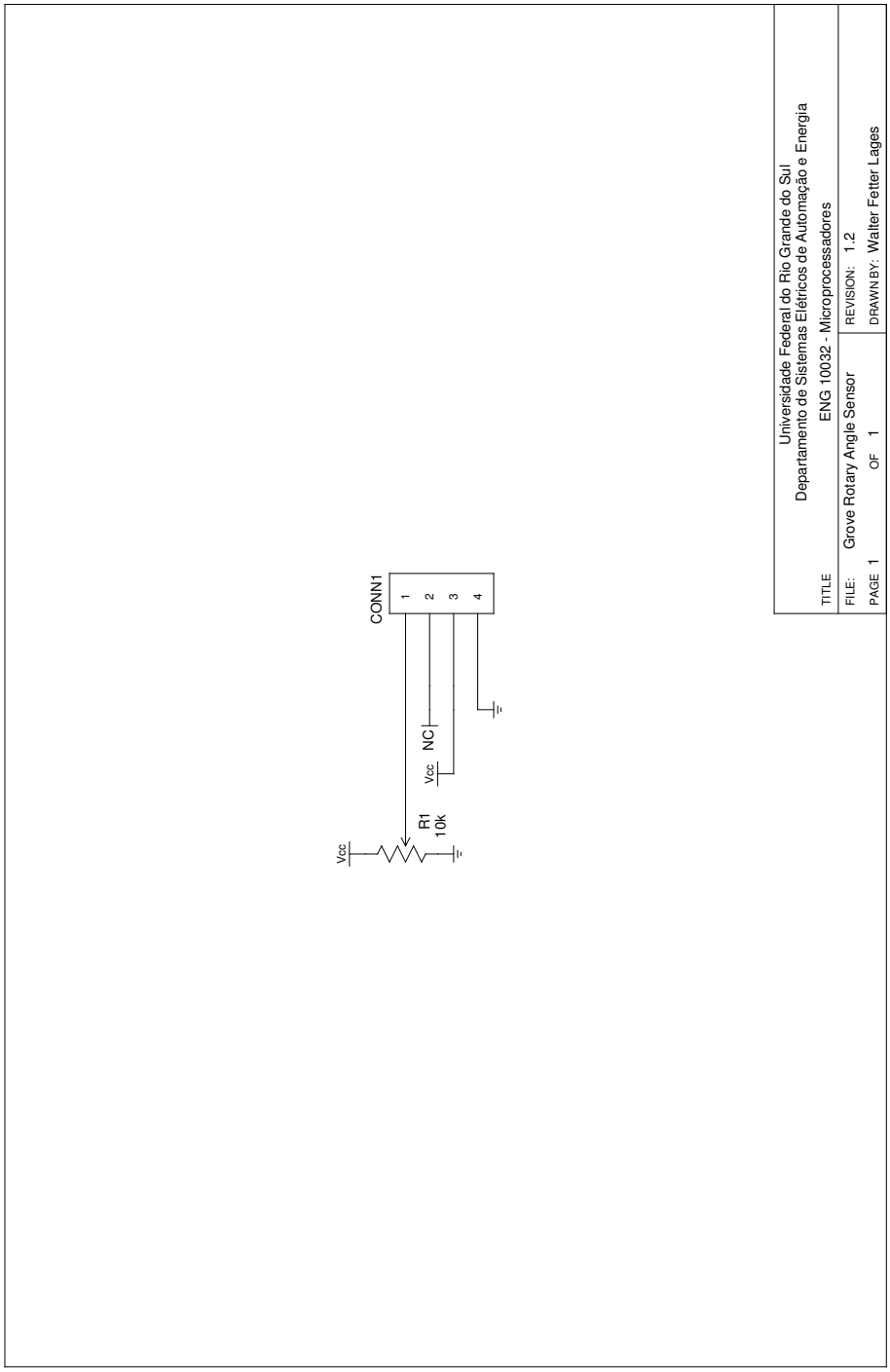


Figura 2: Esquemático do *Grove Rotary Angle Sensor*.

8. Faça um *script* de inicialização para configurar o uso dos `ADC_A0`, `ADC_A1`, `ADC_A2` e `ADC_A3` em modo contínuo. Configure permissões de leitura para o grupo `adc` nos arquivos `in_voltageN_scale` e `/dev/iio:device0` e para leitura e escrita nos arquivos `scan_elements/in_voltageN_en`, `scan_elements/in_timestamp_en`, `buffer/length`, `buffer/enable`, `/dev/iio:device0`, `trigger/current_trigger` e `trigger0/frequency`.

9. Configure o *script* para executar na incialização e reinicialize a Galileo.

10. Além do *Grove Rotary Angle Sensor* já conectado no `ADC_A0`, conecte o *Grove Light Sensor*, mostrado na Figura 1(b), no `ADC_A1`; o *Grove Sound Sensor*, mostrado na Figura 1(c), no `ADC_A2`; e o *Grove Temperature Sensor*, mostrado na Figura 1(d), no `ADC_A3`

11. Faça um programa para obter 1000 amostras de cada sensor usando o modo contínuo com um período de amostradem de 1 ms. Inclua o *timestamp* nas amostras. Salve os dados em um arquivo ASCII onde cada linha deve ter o *timestamp* e os dados dos sensores em Volts separados pelo caractere `TAB` (`\t` em C). Subtraia o *timestamp* inicial, para que todos os tempos sejam relativos ao início da amostragem.

12. Transfira o arquivo com os dados dos sensores para o *host*.

13. Use o Gnuplot, ou outro *software*, para fazer um gráfico dos valores do sensores em função do tempo, a partir do arquivo salvo. Obviamente, isso deve ser feito no *host* e não na Galileo.