IIO intro

FLUSP: FLOSS at USP

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

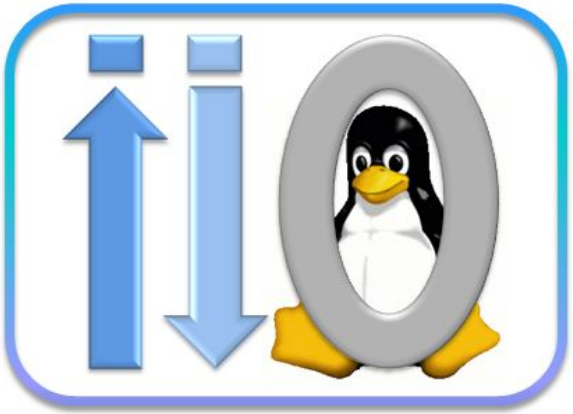
COLLABORA

DigitalOcean

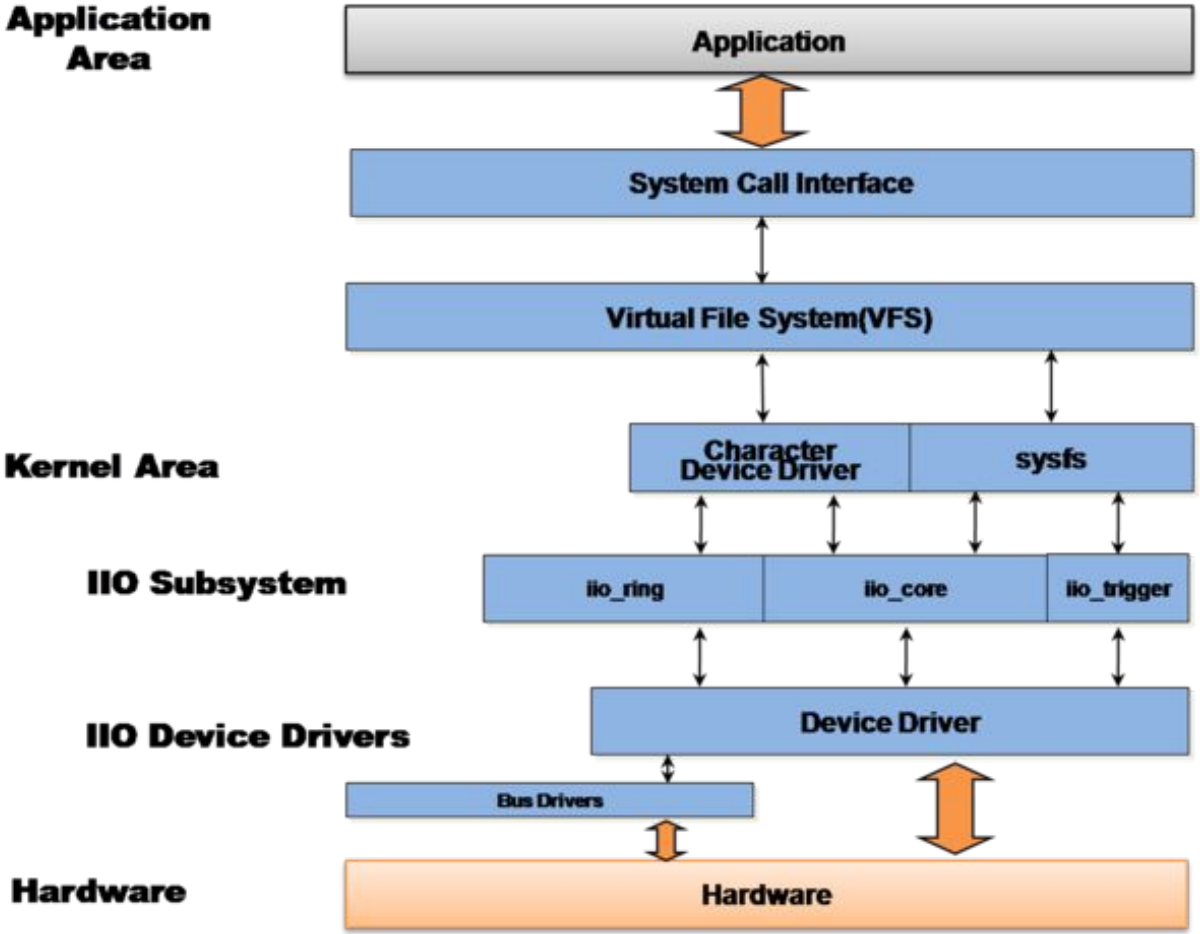
ProFUSION
embedded systems

Industrial Input Output (IIO)

- Diversas categorias de dispositivos
  - ADCs
  - DACs
  - acelerômetros
  - magnetômetros
  - temperatura
  - luminosidade
  - etc.



- Dispositivos podem prover mais de 100 milhões medições por segundo
- API própria para facilitar o desenvolvimento de drivers
- Comunicação direta, ou quase direta, com o hardware
- Cobre lacunas dos subsistemas hwmon e input



Fonte: https://wiki.analog.com/software/linux/docs/iio/iio

Anatomia de um driver do IIO

Conceitos gerais

- Canais
  - São representações de canais de dados
  - Um termômetro tem 1 canal para se obter as medidas de temperatura
  - Um sensor de luz visível e infravermelho pode ter 2 canais
  - Um acelerômetro pode ter 3 canais, representando os eixos X, Y e Z
  - Contém metadados sobre a representação do dado
    - Escala de conversão
    - Organização dentro dos registradores
    - Se o número tem ou não sinal

Anatomia de um driver do IIO

Principais `structs`

- <driver_name>_state
  - Contém as informações do estado do dispositivo
    - structs relativas ao protocolo de comunicação (i2c, spi, etc.)
    - Possivelmente um mutex
    - Possivelmente um buffer
- <protocol_name>_driver
  - spi_driver, i2c_driver
- iio_chan_spec
  - Array de structs que descrevem um conjunto de canais
  - Utilizado na *probe* para definir os canais do driver
- iio_info
  - Utilizado no probe, para descrever o dispositivo
  - Contém ponteiro para read_raw
  - Contém ponteiro para write_raw

Anatomia de um driver do IIO

Funções principais

- read_raw
  - Lê do dispositivo
  - Geralmente composta de dois switches: para canais
  - Geralmente usa um mutex
  - Utiliza o estado do driver para leitura (mutex, buffer e protocolo de comunicação)
- write_raw
  - O mesmo que a read_raw, mas para escrita
- probe
  - Alocar memória para o dispositivo
  - Inicializar dados/variáveis
  - Registrar dispositivo

Driver de brinquedo

Driver de exemplo do IIO:

- IIO dummy (drivers/iio/dummy/iio_simple_dummy.c e iio_simple_dummy.h)
- https://flusp.ime.usp.br/iio/2019/03/16/iio-dummy-anatomy/
- https://flusp.ime.usp.br/iio/2019/04/20/experiment-one-iio-dummy/

Exemplo simples:

- AD2S90 (drivers/iio/resolver/ad2s90.c)

Fluxo de trabalho

1. Rode `git log <path-do-driver>` e confira se ninguém contribuiu recentemente com o driver
2. Encontre algo que possa ser melhorado
   a. Leia os últimos patches sugeridos em https://patchwork.kernel.org/project/linux-iio/list/
   b. Procure por discussões na lista de e-mail ou em outras fontes
3. Tente entender o que precisa ser feito, lendo:
   a. O datasheet do dispositivo
   b. O código do driver
   c. O código de outros drivers do mesmo tipo
   d. A documentação do IIO https://01.org/linuxgraphics/gfx-docs/drm/driver-api/iio/index.html
4. Codifique e envie um patch
   a. Depois de estudar o driver, proponha as modificações enviando um patch.

Como saber o que deve ser feito? - I

- Leia o código de outros drivers do mesmo tipo (resolver, adc, cdc, etc.) que já estejam na mainline e compare com o do driver escolhido.
- Leia o datasheet e veja se a implementação bate com as especificações.
- Cheque se o driver está atualizado com a atual API do IIO: https://01.org/linuxgraphics/gfx-docs/drm/driver-api/iio/index.html

Como saber o que deve ser feito? - II

- Cheque se o registro do driver é feito apenas no final do *probe*.
- Confira se o driver tem documentação de devicetree binding em YAML.
- Cheque se os atributos do dispositivo pertencem à ABI padrão.
- Veja se é possível adicionar mais algum channel ainda não implementado.

* Para drivers da analog devices, esse link pode ser útil:
https://wiki.analog.com/resources/tools-software/linux-drivers-all

Bora codar!

Patrocínio