Realizzazione di un driver Linux per Raspberry Pi

Pietro Lorefice
Linux & Firmware Engineer, Develer

Pietro Lorefice pietro@develer.com

OBIETTIVI

- Prendere familiarità con il workflow di sviluppo su SoC ARM Linux

- Acquisire le basi per lo sviluppo su piattaforme Linux

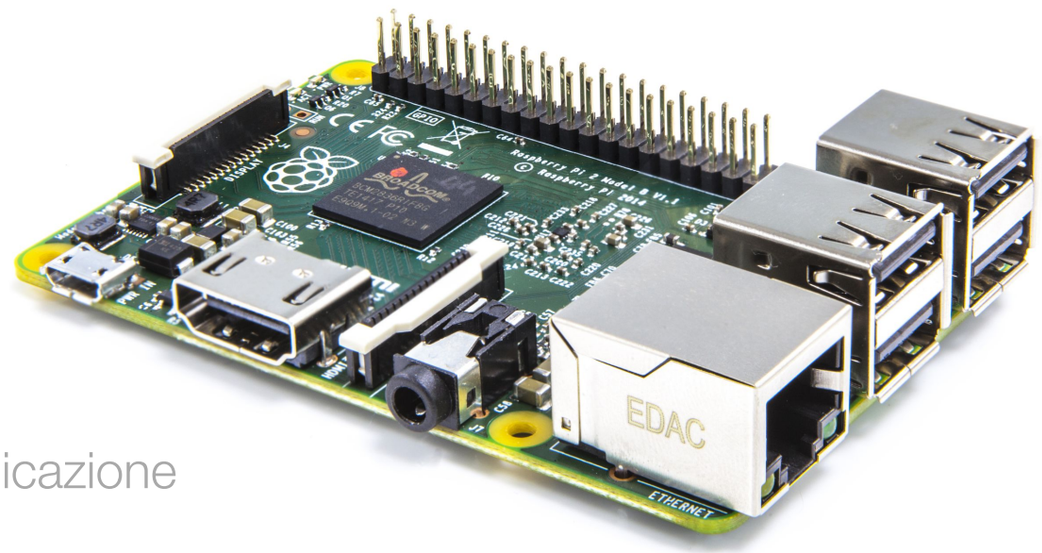
- Realizzare un driver completo e funzionante per il kernel Linux!

Pietro Lorefice pietro@develer.com

HARDWARE

Raspberry Pi 3 - Model B+

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB SDRAM
- WiFi, BT 4.2, BLE & Ethernet
- Header da 40 pin per GPIO e bus di comunicazione



Pietro Lorefice pietro@develer.com

HARDWARE

Raspberry SenseHat

- 8×8 RGB LED matrix
- Five-button joystick
- 9-DoF IMU (**LSM9DS1**)
- Temperature & Pressure sensor (**LPS25H**)
- Humidity sensor (**HTS221**)

Pietro Lorefice pietro@develer.com

SOFTWARE

Raspbian

- Distribuzione Linux Debian-based
- Kernel Linux v4.14.98
- Pacchetti APT: **sense-hat** & **raspberrypi-kernel-headers**

Pietro Lorefice pietro@develer.com

LAB SETUP

- Assegnare l'IP statico **10.10.0.1/16** all'interfaccia Ethernet del proprio PC
- Verificare che la uSD sia correttamente inserita nella RPi
- Collegare la RPi tramite cavo Ethernet al PC e alimentarla tramite cavo micro-USB
  - Da terminale **Linux** e **macOS**: `ssh pi@raspberrypi`
  - Da **Windows**: PuTTY → SSH → `pi@raspberrypi`

IP della scheda: `10.10.10.254` **Username:** `pi` **Password:** `raspberry`

develer

Pietro Lorefice pietro@develer.com

RISORSE UTILI

- Documentazione dei sorgenti kernel con link clickabili:

  https://elixir.bootlin.com/linux/latest/source

- Documentazione ufficiale kernel:

  https://www.kernel.org/doc/Documentation/

Pietro Lorefice pietro@develer.com

REQUISITI PER SVILUPPO KERNEL

Workflow classico

- Ambiente installato sulla macchina di sviluppo
  - Toolchain, sysroot, etc.

- Sorgenti kernel o header sulla macchina di sviluppo
  - Che matchino la versione del kernel installata sul target!

- Strumenti di deploy sul target
  - `scp`, server TFTP, boot media, etc.

Pietro Lorefice pietro@develer.com

REQUISITI PER SVILUPPO KERNEL

Workflow di oggi

- Tutto lo sviluppo verrà fatto sul target!
  - Più comodo e veloce, non richiede deploy incrementale
- Tutti i pacchetti necessari sono già disponibili:
  - Toolchain: **build-essential**
  - Header kernel: **raspberrypi-kernel-headers**

Pietro Lorefice pietro@develer.com

CODE TIME!

Pietro Lorefice pietro@develer.com

LPS25H

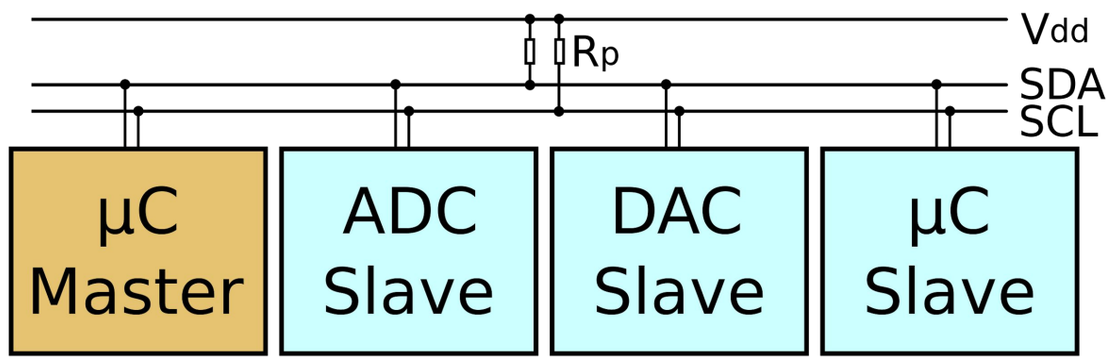
- Sensore barometrico MEMS
- Range di pressione: 260 - 1260 hPa
- Interfaccia SPI e I^2C
- Integra un sensore di temperatura
- Datasheet @ http://dvlr.it/lps25h

develer

Pietro Lorefice pietro@develer.com

I2C

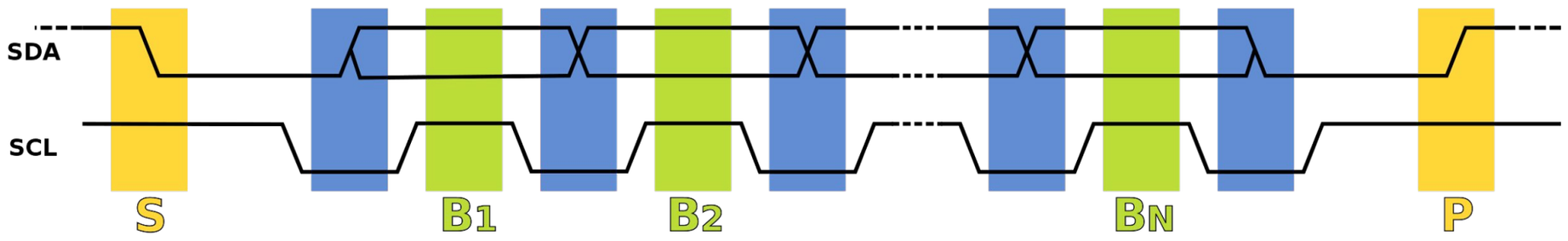
- Bus di comunicazione seriale sincrono
- Modello di comunicazione master-slave
- Utilizzato per collegamento di periferiche a bassa velocità verso uC



Pietro Lorefice pietro@develer.com

COMUNICAZIONE

- Una transazione sul bus è costituita da uno o (raramente) più *messaggi*
- **Solo** il master può iniziare una nuova transazione!
- Ogni messaggio inizia con un simbolo di **START** (**S**) e termina con uno **STOP** (**P**)
- Ogni messaggio può contenere un numero variabile di **bit** (**Bn**)



Pietro Lorefice pietro@develer.com

INDIRIZZAMENTO

- Ogni slave sul bus è identificato da un **indirizzo a 7 bit**
- Su ciascun bus gli indirizzi degli slave devono essere *univoci*
- Il primo byte di ogni trasferimento contiene *sempre* l'indirizzo dello slave!
  - Il bit immediatamente successivo all'indirizzo indica la direzione del trasferimento
    - Se vale '**1**' (**read**), il trasferimento avviene dallo slave al master
    - Se vale '**0**' (**write**), il trasferimento avvene dal master allo slave

develer

Pietro Lorefice pietro@develer.com

CODE TIME!

Pietro Lorefice pietro@develer.com

REGISTER-MAPPED IC

- Molti IC con interfaccia I^2C sono organizzati in *registri*

- Ciascun registro è raggiungibile ad uno specifico *indirizzo*
  - In genere l'indirizzo è lungo un byte per IC semplici

- Il contenuto di ciascun registro è tipicamente cosituito da *un singolo byte*
  - Dati più lunghi sono in genere suddivisi tra più indirizzi

- Esistono eccezioni, in particolare per IC che implementano *memorie*

Pietro Lorefice pietro@develer.com

MAPPA REGISTRI LPS25H

Table 15. Registers address map

Name	Type	Register address	Default	Function and comment
		Hex	Binary	
Reserved (do not modify)		00-07 0D - 0E		Reserved
REF_P_XL	R/W	08	00000000	
REF_P_L	R/W	09	00000000	
REF_P_H	R/W	0A	00000000	
WHO_AM_I	R	0F	10111101	ID register
RES_CONF	R/W	10	00000101	
Reserved (Do not modify)		11-1F		Reserved
CTRL_REG1	R/W	20	00000000	
CTRL_REG2	R/W	21	00000000	
CTRL_REG3	R/W	22	00000000	
CTRL_REG4	R/W	23	00000000	
INT_CFG	R/W	24	00000000	
INT_SOURCE	R	25	00000000	
Reserved (Do not modify)		26		Reserved
STATUS_REG	R	27	00000000	
PRESS_POUT_XL	R	28	output	
PRESS_OUT_L	R	29	output	
PRESS_OUT_H	R	2A	output	
TEMP_OUT_L	R	2B	output	
TEMP_OUT_H	R	2C	output	
Reserved (do not modify)		2D		Reserved
FIFO_CTRL	R/W	2E	00000000	
FIFO_STATUS	R	2F	00000000	
THS_P_L	R/W	30	00000000	
THS_P_H	R/W	31	00000000	
Reserved		32-38		
RPDS_L	R/W	39	00111000	
RPDS_H	R/W	3A	00000000	

develer

Pietro Lorefice pietro@develer.com

MAPPA REGISTRI LPS25H

- Consideriamo il registro **`WHO_AM_I`**

- Molto diffuso in IC di questo tipo
  - Può assumere altre nomenclature, ma il significato è spesso simile

- Molto utile per testare il funzionamento del bus di comunicazione verso l'IC

Table 15. Registers address map

Name	Type	Register address	Default	Function and comment
		Hex	Binary	
Reserved (do not modify)		00-07 0D - 0E		Reserved
REF_P_XL	R/W	08	00000000	
REF_P_L	R/W	09	00000000	
REF_P_H	R/W	0A	00000000	
WHO_AM_I	R	0F	10111101	ID register
RES_CONF	R/W	10	00000101	
Reserved (Do not modify)		11-1F		Reserved
CTRL_REG1	R/W	20	00000000	
CTRL_REG2	R/W	21	00000000	
CTRL_REG3	R/W	22	00000000	
CTRL_REG4	R/W	23	00000000	
INT_CFG	R/W	24	00000000	
INT_SOURCE	R	25	00000000	
Reserved (Do not modify)		26		Reserved
STATUS_REG	R	27	00000000	
PRESS_POUT_XL	R	28	output	
PRESS_OUT_L	R	29	output	
PRESS_OUT_H	R	2A	output	
TEMP_OUT_L	R	2B	output	
TEMP_OUT_H	R	2C	output	
Reserved (do not modify)		2D		Reserved
FIFO_CTRL	R/W	2E	00000000	
FIFO_STATUS	R	2F	00000000	
THS_P_L	R/W	30	00000000	
THS_P_H	R/W	31	00000000	
Reserved		32-38		
RPDS_L	R/W	39	00111000	
RPDS_H	R/W	3A	00000000	

Pietro Lorefice pietro@develer.com

MAPPA REGISTRI LPS25H

- Consideriamo il valore di temperatura
- E' un *intero con segno* da **16 bit**
  - Suddiviso in due registri da **8 bit**
- Il valore grezzo va elaborato

  secondo le indicazioni descritte nel datasheet:

```
T[°C] = 42.5 + (TEMP_OUT/480)
```

Table 15. Registers address map

Name	Type	Register address	Default	Function and comment
		Hex	Binary	
Reserved (do not modify)		00-07 0D - 0E		Reserved
REF_P_XL	R/W	08	00000000	
REF_P_L	R/W	09	00000000	
REF_P_H	R/W	0A	00000000	
WHO_AM_I	R	0F	10111101	ID register
RES_CONF	R/W	10	00000101	
Reserved (Do not modify)		11-1F		Reserved
CTRL_REG1	R/W	20	00000000	
CTRL_REG2	R/W	21	00000000	
CTRL_REG3	R/W	22	00000000	
CTRL_REG4	R/W	23	00000000	
INT_CFG	R/W	24	00000000	
INT_SOURCE	R	25	00000000	
Reserved (Do not modify)		26		Reserved
STATUS_REG	R	27	00000000	
PRESS_POUT_XL	R	28	output	
PRESS_OUT_L	R	29	output	
PRESS_OUT_H	R	2A	output	
TEMP_OUT_L	R	2B	output	
TEMP_OUT_H	R	2C	output	
Reserved (do not modify)		2D		Reserved
FIFO_CTRL	R/W	2E	00000000	
FIFO_STATUS	R	2F	00000000	
THS_P_L	R/W	30	00000000	
THS_P_H	R/W	31	00000000	
Reserved		32-38		
RPDS_L	R/W	39	00111000	
RPDS_H	R/W	3A	00000000	

develer

Pietro Lorefice pietro@develer.com

ACCESSO AI REGISTRI

- **Lettura** di un registro di un IC tramite I²C:
  1. Inviare un comando I²C di scrittura con l'*indirizzo* del registro da leggere
  2. Inviare un comando I²C di lettura, in cui l'IC trasmetterà il contenuto del registro

- **Scrittura** di un registro di un IC tramite I²C:
  1. Inviare un comando I²C di scrittura con l'*indirizzo* del registro da leggere
  2. Inviare un comando I²C di scrittura con il *valore* da scrivere nel registro
     - Equivalente a trasmettere un singolo comando di scrittura da 2 byte

- Queste sono regole generali, non necessariamente valide per *ogni* IC!

Pietro Lorefice pietro@develer.com

CODE TIME!

Pietro Lorefice pietro@develer.com

PSEUDO FILE SYSTEMS

- Nella filosofia Unix, tutto è rappresentabile come un file
- Ciascun file deve necessariamente risiedere in un *file system*
- I file system classici risiedono su supporti non-volatili per la memorizzazione dati
  - Eg. dischi, memorie flash, nastri, etc.
- Gli pseuso file system espongono un'interfaccia standard a strutture interne al kernel
  - Eg. descrittori di processi, statistiche, configruazioni dispositivi, etc.

Pietro Lorefice pietro@develer.com

SYSFS

- Tipicamente *montato* su `/sys`
- Esporta informazioni su diversi sottosistemi del kernel
- Molto utile per esportare anche informazioni sull'hardware e relative configurazioni
- Ciascun driver del kernel può esportare e popolare uno o più in sysfs

Pietro Lorefice pietro@develer.com

CODE TIME!

Pietro Lorefice pietro@develer.com

INDUSTRIAL IO

- Noto più diffusamente come *IIO*

- Sottosistema contenente supporto per apparati che operano nel dominio analogico
  - Eg. DAC, ADC, sensori di temperatura, etc.

- Fornisce un’interfaccia standard sia a chi scrive il driver che a chi lo usa

- Sottosistema molto vasto e articolato, ma sono disponibili molti esempi

Pietro Lorefice pietro@develer.com

CODE TIME!

Pietro Lorefice pietro@develer.com

DOMANDE?

Pietro Lorefice pietro@develer.com

CONTATTI

github

github.com/plorefice

e-mail

pietro@develer.com

web

develer.com/pietro-lorefice

Pietro Lorefice pietro@develer.com