Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Studio e realizzazione di un emulatore grafico didattico per circuiti elettronici collegati a Raspberry PI

Tesi di Laurea in Progetto di Sistemi Virtuali

Relatore:
Chiar.mo Prof.
Renzo Davoli

Presentata da:
Octavian Bujor

Sessione II
Anno Accademico 2012-2013

Introduzione

L'idea della creazione di un computer economico per bambini venne nel 2006 ai quattro ricercatori inglesi Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft presso l'Università di Cambridge. Iniziarono a preoccuparsi del cambiamento nell'interazione che avevano i bambini con i computer negli anni 2000, e quello che hanno progettato e realizzato in diversi anni di lavoro e prototipizzazione è il Raspberry Pi[1], che nel 2011 era in versione alpha ed all'inizio del 2012 è stato messo in commercio.

Si tratta di un mini computer che ha le dimensioni di una carta di credito e la potenza dei PC di qualche anno fa, ma con capacità multimediali più avanzate. Viene messo in commercio nel 2012 in due modelli, A e B, con caratteristiche e prezzo leggermente diversi e, nonostante sia stato ideato per bambini, ha avuto un enorme successo tra gli appassionati di tutte le età di tutto il mondo. Il prezzo, 25 USD per il modello A e 35 USD per modello B, ha avuto sicuramente una notevole influenza viste le sue caratteristiche.

Durante l'attesa tra la presentazione e la commercializzazione, visti i tanti appassionati impazienti di provarlo, nascono le prime idee sulla sua virtualizzazione, ampliandone gli impieghi possibili, che vanno dalla didattica alla sperimentazione.

L'obiettivo della tesi è quello di studiare e realizzare un emulatore grafico per la progettazione di piccoli esperimenti di interfacciamento tra le emulazioni di un Raspberry Pi e un circuito elettronico. Lo scopo principale di questo emulatore è la didattica, in questo modo sarà possibile imparare le basi senza paura di danneggiare i componenti e sopratutto senza la necessità di avere il dispositivo fisico.

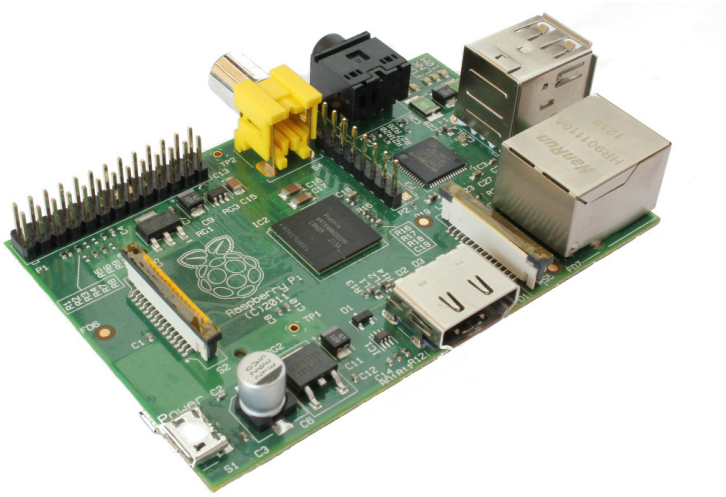
[1] http://www.raspberrypi.org

Indice

Capitolo 1

Stato dell'arte

1.1 Raspberry Pi



Il Raspberry Pi è un calcolatore implementato su una scheda elettronica di dimensioni di una carta di credito, sviluppato dalla Raspberry Pi Foundation nel Regno Unito. Lo scopo principale è l'educazione, con l'idea di creare un dispositivo per stimolare l'insegnamento dell'informatica e della programmazione nelle scuole.

La sua commercializzazione è iniziata nei primi mesi del 2012 con le versioni A e B al prezzo di 25 USD e 35 USD. La vendita è stata affidata ai due distributori mondiali

Farnell e RS Components, che sono specializzati nel campo elettronico con diverse filiali nel mondo, e già nei primi 8 mesi hanno venduto circa 500.000 unità.

Questo calcolatore è stato progettato per funzionare sui sistemi Linux e RiscOS. Ci sono diverse distribuzioni linux pronte all'utilizzo scaricabili dalla pagina dedicata[1], una di queste è Raspbian[2], distribuzione consigliata dalla stessa fondazione, basata su Debian 7 Wheezy che include un l'ambiente desktop LXDE e tanti altri software pronti all'uso.

1.1.1 Hardware

La scheda è guidata dal Broadcom BCM2835, un System-on-a-chip (SoC) che incorpora la CPU, GPU e la memoria. Il processore è un ARM1176JZF-S della famiglia ARM11 con una frequenza di 700 MHz, che è possibile aumentare fino ad 1 GHz senza invalidarne la garanzia. La GPU invece è una Broadcom VideoCore IV con supporto alle librerie OpenGL ES 2.0 ed una risoluzione massima di 1080p. La memoria, che inizialmente era di 256 MB e successivamente aggiornata a 512 MB (solo per il modello B)[4], viene condivisa dai due processori ed è possibile deciderne il frazionamento.

Nel dispositivo non vi è presente nessun tipo di memoria secondaria, ma troviamo invece il lettore di schede di memoria SD, MMC e SDIO. Il boot del sistema viene effettuato dalla scheda, più precisamente da una partizione di tipo FAT32 [2] contenente il firmware, il kernel ed alcuni file di configurazione.

Per quanto riguarda la multimedialità, la scheda è dotata di un'uscita HDMI e di due connettori, RCA per il video composito e jack da 3.5 mm per l'audio.

Grazie ai 26 contatti GPIO disposti sulla scheda, in due linee da 13 pin, si ha la possibilità di collegare al dispositivo diverse periferiche di basso livello.

Le caratteristiche appena elencate sono comuni ai due modelli A e B, ma essi presentano anche delle differenze. Il modello A presenta una sola porta USB 2.0, mentre il modello B ne possiede due utilizzando un hub integrato, ed è dotato di un'interfaccia di rete di tipo Ethernet 10/100 che causa un assorbimento maggiore di corrente, che è poco più del doppio rispetto al modello A. La tabella 1.1 mostra le caratteristiche a confronto dei due modelli.

[1] http://www.raspberrypi.org/downloads
[2] http://www.raspbian.org

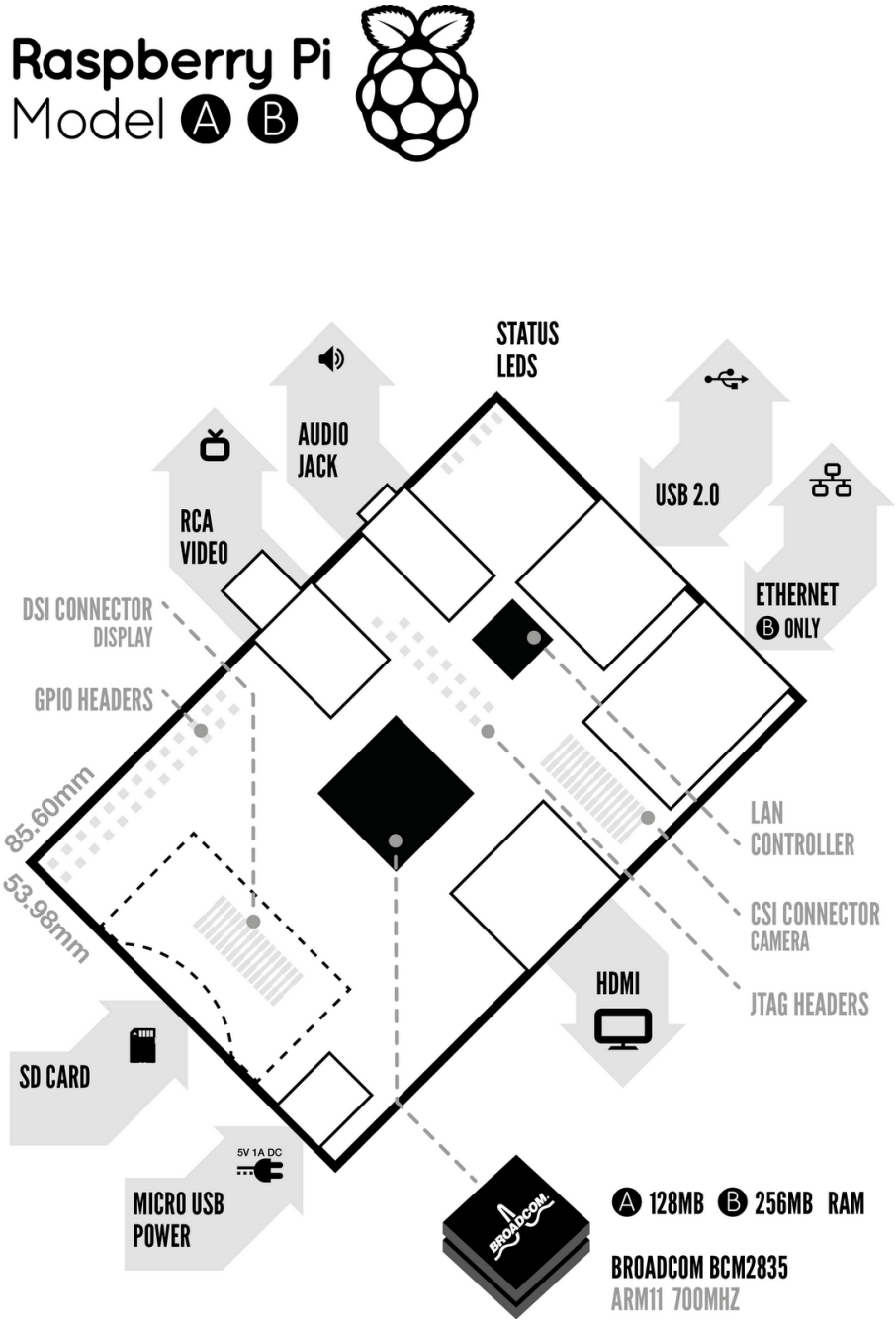


Figura 1.1: Diagramma Raspberry Pi Rev1

Tabella 1.1: Modelli a confronto [1]

	Modello A	**Modello A**
Prezzo:	US$ 25	US$ 35
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)	
CPU:	700 MHz ARM1176JZF-S core (famiglia ARM11)	
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 H.264	
Memoria (SDRAM):	256 MB	256 o 512 MB
Porte USB 2.0:	1	2 (hub integrato)
Output video:	Connettore RCA per il video composito, HDMI	
Output audio:	3,5 mm jack, HDMI	
Memoria (persistente):	SD / MMC / SDIO	
Interfaccia di rete:	non presente	Ethernet 10/100 (RJ-45)
Periferiche di basso livello:	2x13 contatti GPIO, SPI, I2C, UART, +3,3 Volt, +5 Volt	
Corrente (potenza) assorbita:	300 mA, (1,5 W)	700 mA, (3,5 W)
Alimentazione:	5 Volt via MicroUSB o GPIO	
Dimensioni:	85.60mm x 56mm x 21mm	
Peso:	45g	

Led di stato

Vicino al connettore audio ci sono dei led, che hanno la funzione di mostrare lo stato dell'alimentazione, della scheda di memoria e per il modello B anche dell'interfaccia di rete. La tabella 1.2 elenca tali led e le loro caratteristiche. Quello che si occupa dello stato della scheda di memoria è particolarmente interessante per il fatto che è collegato alla linea GPIO 16 e successivamente verrà descritto più nel dettaglio.

Tabella 1.2: Led di stato

Nome		Colore	Funzione
Revisione 1	**Revisione 2**		
OK	ACT	Verde	Attività scheda di memoria
PWR	PWR	Rosso	Presenza alimentazione
Modello B			
FDX	FDX	Verde	Connettività Full Duplex (LAN)
LNK	LNK	Verde	Attività (LAN)
10M	100	Arancione	Connettività (LAN)

1.1.2 Software

Quasi tutti i sistemi operativi per Raspberry Pi sono basati sul kernel Linux, che è il più grande progetto open source mai esistito. Anche se la maggior parte del software in uso è libero, purtroppo esistono ancora delle parti a codice chiuso che si trovano nei driver e nelle librerie della scheda video. Nella figura 1.2 sono illustrati i componenti che usano le API della GPU con il relativo tipo di sorgente.

Alcune distribuzioni includono diversi programmi necessari per permettere all'utente meno esperto di poter configurare ed usare il Raspberry Pi. Raspi-config è uno di questi e, come si intuisce dal nome, il suo compito è quello di aiutare l'utente nella configurazione. Offre un'interfaccia grafica semplice e permette di configurare sia l'hardware che i diversi servizi del sistema operativo. Spesso viene incluso anche LXDE che è un ambiente desktop molto leggero ma completo.

New Out Of Box System (NOOBS) è un programma abbastanza recente che è stato introdotto dalla fondazione a giungo del 2013 per l'installazione facilitata del sistema operativo effettuando anche la partizione della scheda SD, offrendo la possibilità all'utente di scegliere il sistema operativo che più preferisce fra quelli disponibili. Sempre per rendere più facile l'esperienza utenti è stato sviluppato Pi Store, pensato per facilitare sia la pubblicazione delle applicazioni da parte degli sviluppatori che la loro installazione da parte degli utenti.

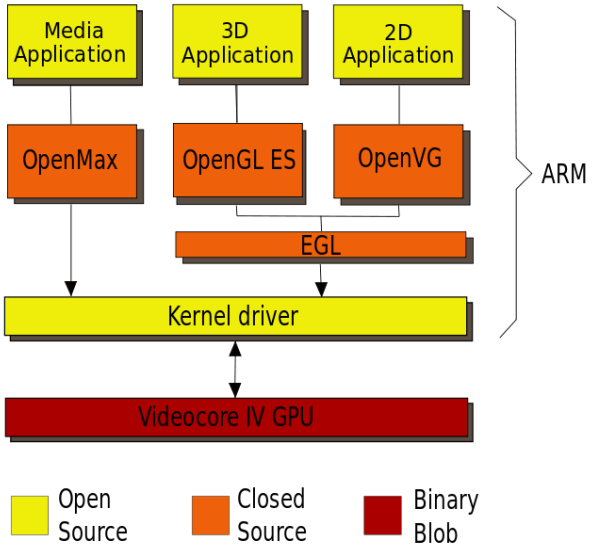


Figura 1.2: Diagramma API della GPU

1.1.3 GPIO

Sono presenti 54 linee di I/O a scopo generico (GPIO) che hanno almeno due funzioni alternative e sono gestite in due serie diverse, la prima incorpora le linee da 0 a 31 e la seconda da 32 a 53. Le linee sono suddivise in diversi gruppi in base al loro uso e non tutte permettono di cambiare la loro funzione perchè alcune sono usate solo per scopi specifici. Tante linee non sono veramente generiche in quanto sono riservate alla gestione dei diversi componenti presenti sulla scheda come il lettore di schede SD, l'uscita audio e i vari LED di stato. È anche presente il connettore CSI (camera serial interface) per la fotocamera, che è già possibile trovare in commercio, e l'interfaccia DSI (display serial interface) per lo schermo che è invece presente ma non è stata ancora abilitata dal software.

Il gruppo che interessa di più è quello del pettine da 26 pin, due linee da 13 ben visibili sulla scheda che offrono il supporto a SPI, I2C, UART seriale e forniscono 3V3 e 5V ma con logica dei pin a solo 3V3, infatti non è presente la tolleranza per i 5V, e questo è importante in quanto non è presente nessuna protezione sulla scheda e si possono causare diversi danni. Questi pin vengono numerati dall'alto in basso e da sinistra a destra e

il primo, indicato con "P1", viene di solito scritto sulla scheda. L'elenco dei pin con la loro descrizione si trova nella tabella 1.3. La maggior parte dei pin ha delle funzioni alternative ma si possono configurare per essere usati come GPIO, per arrivare fino ad un massimo di 17 pin GPIO. Inizialmente tutti i pin sono impostati a GPIO a parte i GPIO 14 e GPIO 15 che sono inizializzati per l'utilizzo di UART, ma via software anche questi possono essere riconfigurati a GPIO.

Ogni GPIO supporta gli interrupt che possono essere delle seguenti tipologie: high, low, rise, fall e change. Per la loro gestione sono presenti tre linee di interrupt, una linea per ogni serie e la terza viene condivisa. Il kernel ufficiale non offre il supporto agli interrupt, esistono però delle patch da applicare alle sorgenti per ottenere il supporto. Per poter sfruttare gli interrupt senza applicare le patch al kernel è possibile usare la distribuzione Raspbian che offre già il kernel con il supporto agli interrupt.

Nel settembre del 2012 è stata annunciata la revisione della scheda, chiamata "Rev2". Una delle parti modificate è l'assegnamento di certi pin, nella tabella 1.3 sono indicati quali pin differiscono dalla prima revisione.

Tabella 1.3: Descrizione pin pettine

Numero	**Nome**	**Funzioni alternative**
P1-01	3V3	-
P1-02	5V0	-
P1-03	GPIO 0 (Rev1) / GPIO 2 (Rev2)	I2C0_SDA / I2C1_SDA
P1-04	5V0	-
P1-05	GPIO 1 (Rev1) / GPIO 3 (Rev2)	I2C0_SCL / I2C1_SCL
P1-06	GND	-
P1-07	GPIO 4	GPCLK0 / ARM_TDI
P1-08	GPIO 14	UART0_TXD / UART1_TXD
P1-09	GND	-

Continua...

Tabella 1.3 – continuazione

Numero	Nome	Funzioni alternative
P1-10	GPIO 15	UART0_RXD / UART1_RXD
P1-11	GPIO 17	UART0_RTS / SPI1_CE1_N / ART1_RTS
P1-12	GPIO 18	PCM_CLK / SPI1_CE0_N / PWM0
P1-13	GPIO 21 (Rev1) / GPIO 27 (Rev2)	PCM_DOUT / SPI1_SCLK / GPCLK1 (Rev1) SD1_DAT3 ARM_TMS (Rev2)
P1-14	GND	-
P1-15	GPIO 22	SD1_CLK / ARM_TRST
P1-16	GPIO 23	SD1_CMD / ARM_RTCK
P1-17	3V3	-
P1-18	GPIO 24	SD1_DAT0 / ARM_TDO
P1-19	GPIO 10	SPI0_MOSI
P1-20	GND	-
P1-21	GPIO 9	SPI0_MISO
P1-22	GPIO 25	SD1_DAT1 / ARM_TCK
P1-23	GPIO 11	SPI0_SCLK
P1-24	GPIO 08	SPI0_CE0_N
P1-25	GND	-
P1-26	GPIO 07	SPI0_CE1_N

Registri

Ci sono 41 registri a 32 bit per la gestione delle linee che vengono usati per configurarne le funzioni alternative, impostarne il valore e tutto quello che riguarda la modifica di stato delle stesse. Possono essere di diverso tipo: alcuni sono solo in scrittura o solo in lettura, ed altri sia in lettura che in scrittura. Nella tabella 1.4 sono elencati e descritti i registri disponibili. Diversi registri sono riservati e non si hanno informazioni di che tipo sono e che scopo hanno. La Broadcom ha deciso di separare il compito dei registri per

renderli più semplici possibile e, per ognuna delle operazioni di base, come l'impostazione dell'output, il suo azzeramento e la lettura del valore dei pin, esistono due registri che le gestiscono.

Tabella 1.4: Registri GPIO (quelli riservati e di test sono stati omessi) [3]

Indirizzo	Nome	Descrizione	Dimensione	Tipo
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 00010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 00014	GPFSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 00020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 00028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 0002C	GPCLR1	GPIO Pin Output Clear 1	32	W
0x 7E20 00034	GPLEV0	GPIO Pin Level 0	32	R
0x 7E20 00038	GPLEV1	GPIO Pin Level 1	32	R
0x 7E20 00040	GPEDS0	GPIO Pin Event Detect Status 0	32	R/W
0x 7E20 00044	GPEDS1	GPIO Pin Event Detect Status 1	32	R/W
0x 7E20 0004C	GPREN0	GPIO Pin Rising Edge Detected Enable 0	32	R/W
0x 7E20 00050	GPREN1	GPIO Pin Rising Edge Detected Enable 1	32	R/W
0x 7E20 00058	GPFEN0	GPIO Pin Falling Edge Detected Enable 0	32	R/W

Continua...

Tabella 1.4 – continuazione

Indirizzo	Nome	Descrizione	Dimensione	Tipo
0x 7E20 0005C	GPFEN1	GPIO Pin Falling Edge Detected Enable 1	32	R/W
0x 7E20 00064	GPHEN0	GPIO Pin High Detected Enable 0	32	R/W
0x 7E20 00068	GPHEN1	GPIO Pin High Detected Enable 1	32	R/W
0x 7E20 00070	GPHEN0	GPIO Pin Low Detected Enable 0	32	R/W
0x 7E20 00074	GPHEN1	GPIO Pin Low Detected Enable 1	32	R/W
0x 7E20 0007C	GPAREN0	GPIO Pin Async. Rising Edge Detect 0	32	R/W
0x 7E20 00080	GPAREN1	GPIO Pin Async. Rising Edge Detect 1	32	R/W
0x 7E20 00088	GPAREN0	GPIO Pin Async. Falling Edge Detect 0	32	R/W
0x 7E20 0008C	GPAREN1	GPIO Pin Async. Falling Edge Detect 1	32	R/W
0x 7E20 00094	GPPUD	GPIO Pin Pull-up/down Enable	32	R/W
0x 7E20 00098	GPPUDCLK0	GPIO Pin Pull-up/down Enable Clock 0	32	R/W
0x 7E20 0009C	GPPUDCLK1	GPIO Pin Pull-up/down Enable Clock 1	32	R/W

Registri di selezione GPFSELn

I sei registri di selezione sono usati per definire le operazioni dei pin GPIO. Le operazioni possibili sono: input, output e altre sei funzioni alternative, e inizialmente sono tutti impostati all'operazione di input. Nella tabella 1.5 sono elencate le operazioni con i relativi indici di lunghezza tre bit. Ognuno dei 5 registri da GPFSEL0 a GPFSEL4 definisce le operazioni di 10 linee, l'ultimo (GPFSEL5) di solamente 4 linee per un totale di 54 linee. Come mostrato nella tabella 1.4 tali registri sono sia di lettura che di scrittura, per ottenere le operazioni questi vengono letti e per impostarle vengono scritti. La struttura interna dei registri è descritta nella tabella 1.6 dove per ogni registro sono illustrate le linee GPIO gestite.

Tabella 1.5: Operazioni GPIO

Indice	**Descrizione**
000	input
001	output
100	funzione alternativa 0
101	funzione alternativa 1
110	funzione alternativa 2
111	funzione alternativa 3
011	funzione alternativa 4
010	funzione alternativa 5

Registri per impostare l'output GPSETn

I due registri GPSETn vengono usati per impostare l'output dei pin GPIO, sono solo in scrittura e vengono interpretati come mappe di bit, dove il campo SETn definisce il pin rispettivo da impostare. Non si hanno cambiamenti di stato se viene scritto il valore "0" oppure se il pin risulta di input, e per impostare il valore a "0" si devono usare i registri GPCLRn. La tabella 1.7 riassume le caratteristiche di tali registri.

Tabella 1.6: Struttura registri GPFSELn e le linee GPIO

Bit	31-30	29-27	26-24	23-21	20-18	17-15	14-12	11-9	8-6	5-3	2-0
GPFSEL0	-	9	8	7	6	5	4	3	2	1	0
GPFSEL1	-	19	18	17	16	15	14	13	12	11	10
GPFSEL2	-	29	28	27	26	25	24	23	22	21	10
GPFSEL3	-	39	38	37	36	35	34	33	32	31	30
GPFSEL4	-	49	48	47	46	45	44	43	42	41	40
GPFSEL5	-	-	-	-	-	-	-	53	52	51	50

Tabella 1.7: Registri GPSETn

Bit	**Nome del campo**	**Descrizione**	**Valore iniziale**
GPSET0			
31-22	SETn (n=0..31)	0 - Nessun effetto, 1 - Imposta il pin n	0
GPSET1			
31-22	-	Riservato	0
21-0	SETn (n=32..53)	0 - Nessun effetto, 1 - Imposta il pin n	0

Registri per azzerare l'output GPCLRn

I registri GPCLRn vengono usati per azzerare l'output dei pin GPIO, sono solo in scrittura e vengono interpretati come mappe di bit, dove il campo CLRn definisce il pin rispettivo da azzerare. Non si hanno cambiamenti di stato se viene scritto il valore "0" oppure se il pin risulta di input, e per impostare il valore a "1" si devono usare i registri GPSETn. La tabella 1.8 ne riassume il funzionamento.

Tabella 1.8: Registri GPCLRn

Bit	Nome del campo	Descrizione	Valore iniziale
GPCLR0			
31-22	CLRn (n=0..31)	0 - Nessun effetto, 1 - Azzera il pin n	0
GPCLR1			
31-22	-	Riservato	0
21-0	CLRn (n=32..53)	0 - Nessun effetto, 1 - Azzera il pin n	0

Registri di livello GPLEVn

GPLEV1 e GPLEV2 sono registri abilitati alla sola lettura, restituiscono il valore attuale dell'ennesimo pin. Le loro specifiche sono mostrate nella tabella 1.9.

Tabella 1.9: Registri GPLEVn

Bit	Nome del campo	Descrizione	Valore iniziale
GPLEV0			
31-22	LEVn (n=0..31)	0 - Pin n basso, 1 - Pin n alto	0
GPLEV1			
31-22	-	Riservato	0
21-0	LEVn (n=32..53)	0 - Pin n basso, 1 - Pin n alto	0

Esempio concreto

Il led chiamato "ACT" oppure "OK" ha la funzione di mostrare lo stato della scheda di memoria lampeggiando quando questa è attiva, come accennato prima il led è collegato alla linea GPIO 16, visto la sua funzionalità è la prima linea ad essere configurata dal sistema. Siccome all'inizio tutte le linee sono di input ed il led è di output allora la linea 16 viene configurata come output.

La famiglia di registri GPFSELn gestisce le operazioni delle linee, come possiamo vedere nella tabella 1.6, l'operazione di questa specifica linea viene memorizzata nei bit 20-18 del registro GPFSEL1. La scrittura di un registro comporta la sovrascrittura di tutti i 32 bit, quindi per cambiare l'operazione di un'unica linea e per non alterare le operazioni delle altre, prima viene letto il valore del registro e successivamente si cambiano i tre bit corrispondenti, infine viene scritto il nuovo valore. Nel caso specifico viene letto l'indirizzo 0x4[3] che corrisponde al registro GPFSEL1 (tabella 1.4), dato che "000" è l'indice dell'operazione di input (tabella 1.5) e tutte le linee di default sono di input allora il valore letto sarà "0" cioè 32 bit tutti a "0". A questo punto nel registro viene scritto il nuovo valore che sarà dato da quello precedente ma con i bit 20-18 impostati a "001" (l'indice dell'operazione di output, tabella 1.5) e risulta il valore 0x40000 cioè tutti i bit a "0" tranne il diciottesimo.

Ora che la linea 16 è di output, il led viene acceso impostando il valore di output della linea a "1" e questo viene fatto assegnando a "1" il bit in posizione 16 del registro GPSET0, che ha la funzione di impostare i valori di output delle linee 0-31 (tabella 1.7). Per ottenere l'effetto lampeggiante, il led deve poter essere spento cioè l'output della linea 16 deve avere il valore "0", e questo non si ottiene scrivendo tale valore nella giusta posizione del registro GPSET0 perché non produrrebbe nessun effetto, bensì viene scritto il valore "1" nella posizione 16 del registro GPCLR0. Sia per impostare che per azzerare il valore di output viene scritto lo stesso valore (0x10000) ma in due indirizzi diversi, 0x1C e 0x28 che corrispondo ai registri GPSET0 e GPCLR0. Il valore di 0x10000 è dato da una mappa di 32 bit con tutti a "0" tranne il sedicesimo.

Riassumendo, per avere il led sulla linea GPIO 16 lampeggiante il sistema effettua le operazioni elencate sotto, mentre le prime due sono di impostazione e vengono effettuate solo all'accensione le altre servono ad accendere e spegnere il led e quindi possono essere ripetute anche più volte al secondo in base al livello di attività della scheda di memoria.

```
read 0x4; // Legge le operazioni delle linee 10-19
write 0x4 = 0x40000; // Imposta GPIO 16 ad output
write 0x1C = 0x10000; // Imposta a 1 il valore di output di GPIO 16
write 0x28 = 0x10000; // Azzera il valore di output di GPIO 16
```

[3]Tutti gli indirizzi partono dalla base dei registri che è 0x7E200000

1.2 QEMU

QEMU[4] è un software open source in grado di emulare diversi microprocessori creato da Fabrice Bellard [5], un programmatore francese che ha sviluppato diversi progetti open source di grande importanza. QEMU ha la caratteristica di supportare tantissimi dispositivi come il lettore CD, lettore di schede SD, dischi rigidi, interfacce di rete e il supporto all'USB. Per ogni lettore è possibile impostare un file di immagine che verrà messo a disposizione del sistema emulato.

Questo software è molto importante per la fattibilità del progetto di questa tesi perché supporta la famiglia di processori ARM1176 di cui fa parte quello del Raspberry Pi.

Esecuzione QEMU

Per l'esecuzione di QEMU la prima cosa da decidere è l'eseguibile da usare, ce ne sono diversi in base alle architetture disponibili. Una volta scelto l'eseguibile si può configurare la macchina emulata a piacimento scegliendo tra diverse possibili configurazioni, come il processore da usare, la memoria, l'immagine del disco fisso e tante altre. L'esempio 1.2.1 mostra alcune configurazioni possibili per il lancio del programma.

Esempio 1.2.1. Lancio di QEMU, semplice configurazione impostando la quantità di memoria e l'immagine del disco da usare.

```
./qemu-system-i386 -m 1024 debian.img
```

Cambiando l'eseguibile viene emulata un'architettura diversa.

```
./qemu-system-arm -m 512 debian_arm.img
```

1.2.1 QEMU rpi

Come detto in precedenza il Raspberry Pi viene pilotato da un SoC Broadcom BCM2835, e purtroppo QEMU non supporta l'intero SoC ma solo la CPU. QEMU rpi, creato da Gregory Estrade[6], è il branch della copia del repository QEMU ufficiale, che ha

[4] http://qemu.org
[5] http://bellard.org
[6] https://github.com/Torlus

lo scopo di portare tutto il BCM2835 su QEMU. Il branch implementa diversi dispositivi che permettono di fare boot del sistema operativo senza fare modifiche al kernel, ed è possibile lanciare la stessa immagine che viene usata sul Raspberry Pi reale.

Per la scelta dei dispositivi da usare a basso livello, QEMU prevede la possibilità di fare delle macchine, e QEMU rpi implementa la macchina "raspi" che contiene tutti i dispositivi dedicati al SoC BCM2835 e si occupa di inizializzare tutto il sistema, la CPU, la memoria e gli stessi dispositivi. Alla fine carica il kernel tenendo conto degli argomenti passati al lancio del programma.

Dispositivi BCM2835

In questo branch sono stati implementati dei dispositivi QEMU per avere il supporto del SoC BCM2835 grazie ai quali si riesce a caricare l'immagine originale, creata per il Raspberry Pi. Tuttavia finora non tutte le funzionalità del SoC sono state implementate, per questo motivo è stato creato un dispositivo speciale "bcm2835_todo" il cui compito è semplicemente di stampare gli indirizzi dei registri da mappare.

Dispositivi implementati:

* **bcm2835_ic** - controllore degli interrupt
* **bcm2835_st** - timer di sistema
* **bcm2835_timer** - timer ARM
* **bcm2835_usb** - controllore USB
* **bcm2835_mphi** - gestione dell'interfaccia MPHI (Message-based Parallel Host Interface)
* **bcm2835_sbm** - gestione dei semafori, "doorbells", "mailboxes"
* **bcm2835_power** - gestione dell'alimentazione
* **bcm2835_fb** - framebuffer
* **bcm2835_property** - gestione del canale property

* **bcm2835_vchiq** - VCHIQ
* **bcm2835_emmc** - controllore EMMC (Extended Mass Media Controller)
* **bcm2835_dma** - gestione dei canali DMA
* **bcm2835_todo** - debug, stampa degli indirizzi non mappati

Immagine del kernel

Un elemento fondamentale nella partenza della macchina emulata è l'immagine del kernel che può essere estratta dalla partizione FAT32 di una della immagini SD disponibili per il Raspberry Pi, questo procedimento è illustrato nell'esempio 1.2.2 usando l'immagine SD di Raspbian Wheezy.

Esempio 1.2.2. Estrazione dell'immagine del kernel dall'immagine SD "2013-02-09-wheezy-raspbian.img". Il kernel si trova nella partizione FAT32, per copiare il file "kernel.img" essa deve essere montata in una cartella nel sistema. Per effettuare il montaggio abbiamo bisogno dell'immagine SD e di due dati riguardanti la partizione: il settore di inizio e la dimensione dei settori, entrambi i valori possono essere trovati usando l'utilità "fdisk". Partendo da questi dati possiamo calcolare l'offset necessario per il montaggio della partizione, basta moltiplicare il settore iniziale per la sua dimensione.

```
mount -t vfat -o loop,offset=4194304 2013-02-09-wheezy-raspbian.img /mnt/
cp /mnt/kernel.img .
umount /mnt
```

Esecuzione QEMU rpi

Alla partenza di QEMU per usare la macchina "raspi" si deve assegnare al parametro "M" il nome della stessa, l'immagine del kernel invece viene specificata con il parametro "kernel". Per un'impostazione corretta, si devono specificare alcune caratteristiche della macchina che vogliamo emulare, la memoria con il parametro "m" che può essere 256 o 512 dipendente dalla versione del Raspberry Pi desiderata, il parametro "cpu" che appunto specifica la CPU, deve essere impostato a "arm1176". Inoltre si devono inserire

diversi parametri al kernel per la configurazione dei dispositivi bcm2835, questi parametri nel caso reale vengono passati dal bootloader. Il parametro "sd" rappresenta il lettore di schede SD e deve essere valorizzato con un l'indirizzo dell'immagine della scheda. L'esempio 1.2.3 mostra una configurazione possibile per il lancio del programma.

Esempio 1.2.3. Lancio di QEMU specificando la macchina "raspi" e la sua configurazione:

```
./qemu-system-arm -kernel kernel.img -m 512 -cpu arm1176 -M raspi
-append "rw earlyprintk loglevel=8 panic=120 keep_bootcon rootwait
dma.dmachans=0x7f35 bcm2708_fb.fbwidth=640 bcm2708_fb.fbheight=480
bcm2708.boardrev=0xf bcm2708.serial=0xcad0eedf dwc_otg.lpm_enable=0
smsc95xx.macaddr=B8:27:EB:D0:EE:DF sdhci-bcm2708.emmc_clock_freq=100000000
vc_mem.mem_base=0x1c000000 vc_mem.mem_size=0x20000000 kgdboc=ttyAMA0,115200
console=ttyS0 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait"
-sd 2013-02-09-wheezy-raspbian.img -device usb-kbd -device usb-mouse
```

I parametri "vc_mem.mem_base=0x1c000000 vc_mem.mem_size=0x20000000" passati al kernel definiscono la divisione della memoria tra il processore ARM e il VideoCore. Questi valori possono essere cambiati, per evitare problemi di corruzione della memoria è importante modificare anche il valore della costante "VCRAM_SIZE" che si trova nel file "bcm2835_common.h" [6]. I parametri "bcm2708_fb.fbwidth=640 bcm2708_fb.fbheight=480" invece, impostano la dimensione della memoria framebuffer, quindi modifica la risoluzione dello schermo, che può essere scelta a piacere tra quelle supportate[7].

1.3 noVNC

noVNC[8] è un client per VNC scritto in JavaScript che usa le tecnologie "WebSocket" e "Canvas" di HTML5. Funziona in tutti i browser moderni senza necessità di software aggiuntivo. Diverse aziende lo integrano nei loro prodotti, è stato progettato per essere facilmente integrato[5].

[7] http://elinux.org/RPiconfig

[8] https://github.com/kanaka/noVNC

1.3.1 Requisiti

Client

Per la visualizzazione delle immagini viene usato l'elemento Canvas di HTML5, per la loro trasmissione vengono usati i WebSockets, queste due tecnologie devono essere supportate dal browser ma per il supporto dei WebSockets sui browser più datati esiste una libreria[9] in Adobe Flash che li implementa. L'autore ha messo a disposizione una lista dei browser supportati[10].

Server

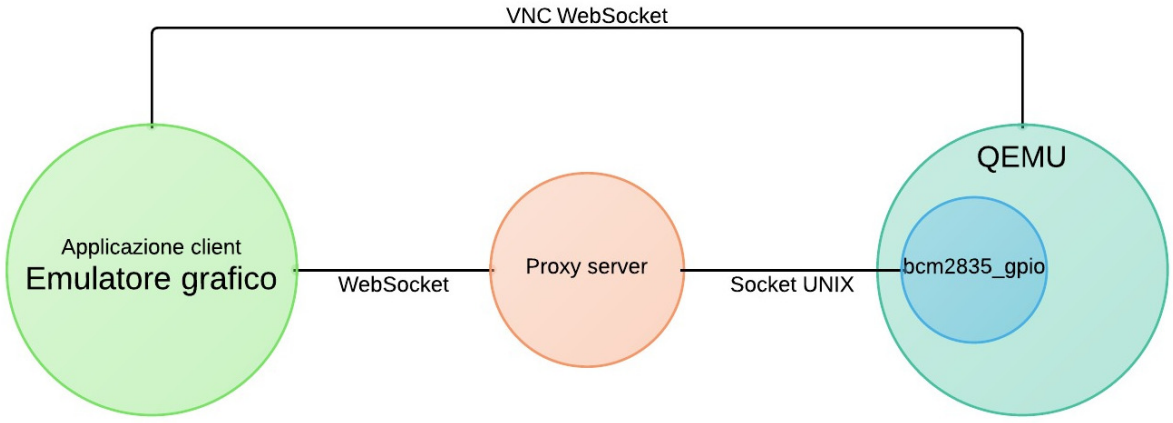
Il server deve supportare il protocollo WebSocket oppure è necessario usare un proxy da WebSocket a TCP, ed uno di questi viene in incluso in noVNC, è scritto in Python, il suo nome è "websockify". "x11vnc/libvncserver"[11] è un server VNC che supporta nativamente questo protocollo, anche il server VNC integrato in QEMU supporta il protocollo[7].

[9] https://github.com/gimite/web-socket-js
[10] https://github.com/kanaka/noVNC/wiki/Browser-support
[11] http://libvncserver.sourceforge.net/

Capitolo 2

Implementazione

La realizzazione dell'emulatore grafico didattico consiste nell'implementazione di più componenti software: l'emulatore, un dispositivo QEMU ed un proxy server, che sono illustrati con i relativi collegamenti nella figura sottostante.



2.1 Dispositivo QEMU

Nella sezione 1.2.1 dedicata a QEMU rpi sono descritti i dispositivi implementati da Gregory Estrade per il supporto del SoC BCM2835, tra questi non ne troviamo nessuno riguardante il GPIO e i suoi indirizzi vengono mappati nel dispositivo di debug "bcm2835_todo" per indicare che è un parte ancora da implementare.

2.1.1 bcm2835_gpio

Il dispositivo implementato è "bcm2835_gpio" mappato sugli indirizzi GPIO partendo dalla base "GPIO_BASE" definita nel file "bcm2835_platform.h". Questo dispositivo non era stato implementato come gli altri, perché non è essenziale per il funzionamento del Raspberry Pi emulato e senza l'emulazione dei contatti GPIO non avrebbe nessun senso implementarlo.

È un dispositivo molto semplice, all'inizializzazione viene effettuata una connessione tramite un socket di tipo UNIX con il proxy server a cui vengono inoltrate tutte le richieste. Mentre alla scrittura la richiesta viene solo inoltrata alla lettura attende la risposta che viene restituita al chiamante.

Protocollo

Per l'inoltro delle richieste viene usato un semplice protocollo in cui si scambiano semplici messaggi che hanno la seguente struttura: "tipo, indirizzo (, valore)?;" dove "tipo" può essere "r" o "w", che stanno per lettura e scrittura, e nel caso di struttura viene indicato anche il valore da scrivere. L'esempio 2.1.1 mostra alcuni possibili messaggi.

Esempio 2.1.1. Alcuni messaggi di esempio:

```
r,4; //lettura dell'indirizzo 4
w,4,262144; //scrittura di 262144 nell'indirizzo 4
```

2.2 Proxy server

Il proxy server è uno script scritto in Python che implementa due server, uno che usa socket UNIX per lo scambio di dati con il dispositivo QEMU e l'altro di tipo WebSocket per lo scambio di dati con l'emulatore grafico. Il suo compito principale è di trasmettere i messaggi da una parte all'altra, ma si occupa anche della gestione della macchina QEMU e, su richiesta dell'emulatore grafico, può farla partire o spegnere.

Per l'implementazione dei due server viene usato il framework Twisted[1] che facilita molto la programmazione di rete in Python. Per l'implementazione del server WebSocket viene usato anche AutobahnPython[2] che implementa il protocollo WebSocket per la creazione di server e client.

2.3 Emulatore grafico

L'emulatore grafico è un'applicazione web scritta in JavaScript che utilizza la libreria jQuery[3] per la manipolazione del DOM e la tecnologia SVG (Scalable Vector Graphics) per il disegno dei componenti. Integra il client VNC "noVNC" per la visualizzazione della macchina QEMU che implementa un server VNC con il supporto del protocollo WebSocket. Diversi aspetti dell'applicazione prendono spunto da Fritzing[4] che è un software libero per la progettazione elettronica (EDA).

Per la gestione delle immagini SVG non viene usato nessuna libreria perché nessuna di quelle prese in considerazione offre il supporto completo alle immagini già esistenti, che è un requisito importante visto che l'emulatore deve gestire anche le immagini provenienti da altri progetti liberi.

Graficamente l'applicazione è composta da: un circuito basato da una basetta sperimentale (breadboard) e diversi componenti tra cui anche il Raspberry Pi, e due finestre, una che contiene una tabella aggiornata delle linee GPIO e l'altra contenente il client VNC che viene visualizzata all'accensione del Raspberry Pi.

[1] http://twistedmatrix.com
[2] http://autobahn.ws/python/
[3] http://jquery.com
[4] http://fritzing.org

2.3.1 Caratteristiche

L'emulatore è sempre connesso al proxy server tramite i WebSocket per lo scambio di messaggi usando un semplice protocollo JSON. Per accendere o spegnere la macchina QEMU l'emulatore invia un messaggio al proxy server che si occupa di effettuare l'operazione richiesta.

Il circuito e anche i singoli componenti possono essere spostati usando la tecnica del "drag & drop" senza perdere le connessioni esistenti con i fili e altri componenti. Inoltre sono supportati anche l'ingrandimento e la diminuzione dell'intero circuito che si attivano utilizzando la rotella di scorrimento del mouse, con uno spostamento verso l'alto si effettua l'ingrandimento e con lo spostamento opposto la diminuzione.

È supportata la creazione dei fili di collegamento, la procedura è molto semplice e ha inizio nel momento in cui viene fatto click sopra ad un pin, e allo spostamento del mouse il filo appena creato si allunga opportunamente fino al secondo click sopra ad un pin che completa la sua creazione.

2.3.2 Circuito

Il circuito è rappresentato da un elemento SVG e contiene minimo due componenti, il Raspberry Pi e una basetta sperimentale (breadboard), e viene costruito partendo da un file JSON che lo descrive. L'esempio 2.3.1 mostra un possibile file di configurazione di un semplice circuito che viene illustrato nella figura 2.1.

Struttura del file JSON

Il file JSON contiene i componenti e i fili presenti nel circuito, ha due liste "components" e "wires", quest'ultima può essere anche vuota se nel circuito non ci sono fili.

La lista dei componenti contiene oggetti aventi le seguenti proprietà:

* **name** - stringa, nome del componente
* **file** - stringa, path del file che descrive il componente
* **transform** (opzionale) - stringa, trasformazioni al componente.

* **connections** (opzionale) - oggetto che contiene le eventuali connessioni tra componenti

Il valore impostato a "transform" verrà assegnato all'attributo "transform"[5] dell'elemento SVG che rappresenta il componente e quindi deve essere una stringa di trasformazione valida.

L'oggetto "connections" contiene una o più coppie chiave - valore dove la chiave rappresenta il nome del connettore che viene connesso al connettore del componente descritto nell'oggetto valore.

```
"connections": {
    "nome_connettore": {
       "component": "nome_componente",
       "connector": "nome_connettore"
    }
}
```

La stessa struttura hanno gli oggetti che descrivono i fili ma contengono esattamente due connettori con nomi "start" e "end", opzionalmente è possibile specificare il colore del filo assegnando un valore di colore esadecimale nella proprietà "color".

Esempio 2.3.1. Configurazione di un semplice circuito contenente un Raspberry Pi ed una breadboard:

```
{   "components": [{
            "name": "raspberry",
            "file": "components/raspberry/raspberry.json"
        }, {
            "name": "Half_breadboard",
            "file": "components/Half_breadboard/Half_breadboard.json",
            "transform": "translate(300, 30)"
        }
    ],
```

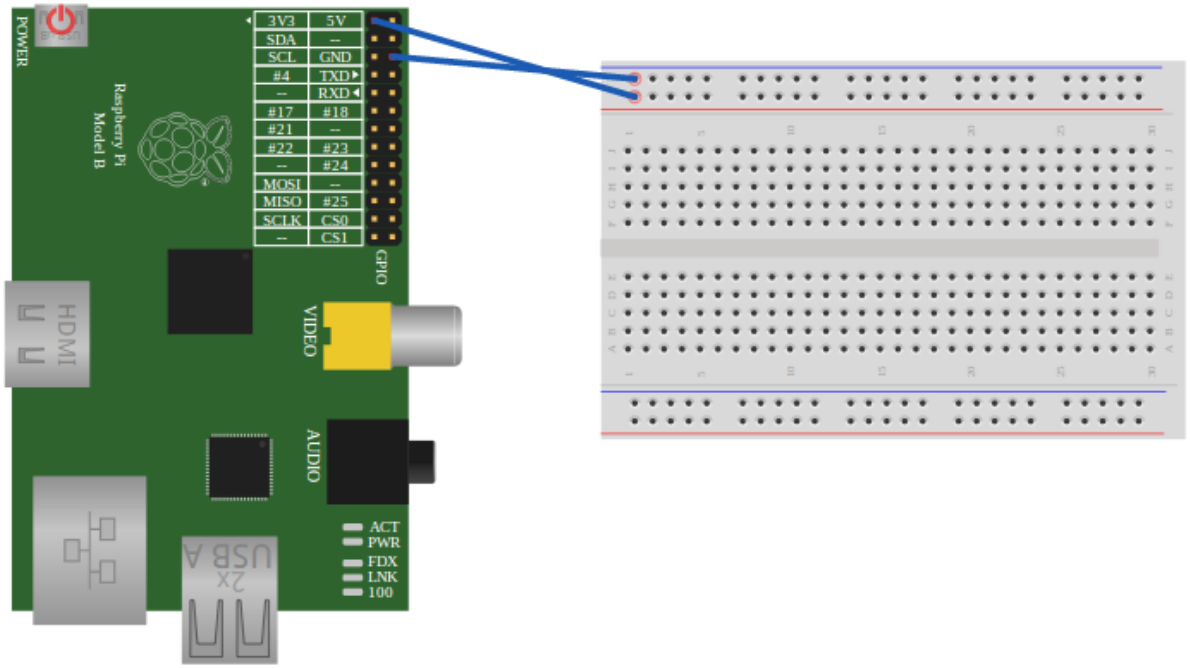
[5] http://www.w3.org/TR/SVG/coords.html#TransformAttribute

```
    "wires": [{
        "start": {
            "connector": "connector1",
            "component":"raspberry"
        },
        "end": {
            "connector": "pin3Y",
            "component":"Half_breadboard"
        }
    },{
        "start": {
            "connector": "connector6",
            "component":"raspberry"
        },
        "end": {
            "connector": "pin3Z",
            "component":"Half_breadboard"
        }
    }]
}
```

2.3.3 Componenti

Anche i componenti vengono creati partendo da un file JSON di configurazione in cui viene specificato un file SVG, che rappresenta l'aspetto del componente, e se sono necessari comportamenti personalizzati anche un file JavaScript. Siccome creare i file SVG è abbastanza complesso sopratutto se anche i componenti sono complessi, vengono inclusi file già esistenti provenienti da altri progetti liberi, come per esempio "Fritzing" da cui vengono presi alcuni file SVG per i componenti implementati. Allo stato di sviluppo attuale sono supportati soltanto i componenti di output.

Figura 2.1: Circuito generato dal file dell'esempio 2.3.1



Struttura del file JSON

Il file di configurazione, l'esempio 2.3.2 ne mostra una possibile versione, consiste in un oggetto JSON contenente le seguenti proprietà:

* **viewFile** - stringa, path del file SVG
* **connectors** - oggetto contenente i connettori del componente
* **rootElementId** (opzionale) - stringa, id dell'elemento da prendere in considerazione, se non si vuole l'intera immagine
* **controllerFile** (opzionale) - stringa, path del file JavaScript
* **buses** (opzionale) - lista che contiene eventuali gruppi di connettori

L'oggetto "connectors" contiene delle coppie chiave - valore dove la chiave è il nome del connettore e il valore è un oggetto che lo descrive, avente "svgId" e "type" come proprietà,

dove alla prima viene assegnato l'id dell'elemento SVG che corrisponde al connettore e la seconda definisce il tipo di connettore e che può essere "female" o "male".

La proprietà "rootElementId" viene specificata quando l'immagine SVG contiene più elementi di quelli che si vogliono visualizzare e si vuole limitare la vista ad un solo elemento.

La lista "buses" contiene oggetti che descrivono dei bus, cioè dei connettori che sono connessi tra di loro. Ogni oggetto ha le proprietà "id" e "members" dove la prima è l'identificatore del bus e la seconda è una lista contenente i nomi del connettori che fanno parte del gruppo.

Esempio 2.3.2. Configurazione di un semplice componente:

```
{
    "viewFile": "part.svg",
    "rootElementId": "breadboard",
    "controllerFile" : "part.js",
    "connectors": {
        "connector0": {
            "svgId": "connector0pin",
            "type": "male"
        },
        "connector1": {
            "svgId": "connector1pin",
            "type": "male"
        }
    }
}
```

controllerFile

Il file JavaScript (controller) che viene specificato come "controllerFile" ha il pieno controllo sul componente sia sull'elemento SVG che sulle sue proprietà questo permette di creare componenti che hanno comportamenti diversi e di rendere le viste dinamiche. Lo

script deve invocare la funzione “applyConfig” e passare come parametri il nome del componente ed un oggetto che rappresenta la configurazione da applicare che può contenere informazioni utili per la gestione del componente e può implementare anche delle funzioni. Dopo il caricamento del componente verrà chiamata la funzione “componentLoaded” che lo script può implementare se si devono fare delle modifiche.

Breadboard

La breadboard è un componente composto da tanti pin di tipo femmina raggruppati in diversi bus, il file SVG, che può essere visto nella figura 2.1, e la maggior parte delle configurazioni provengono dal corrispettivo componente di Fritzing. Il controller ha lo scopo di rendere la vista interattiva e di evidenziare i pin e il bus di cui fanno parte al passare del mouse.

Raspberry Pi

Il componente che rappresenta il Raspberry Pi è il più complesso in quanto deve gestire le linee GPIO e led di stato. Il file SVG, che può essere visto nella figura 2.1, deriva dalla libreria “AdaFruit Fritzing Library”[6] è stato migliorato aggiungendo i led di stato e il pulsante di accensione.

Le linee GPIO vengono memorizzate e gestite dal oggetto “gpio” del controller che si occupa di gestire anche le operazioni di lettura e di scrittura delle linee chiamate dall'emulatore con le funzioni “write” e “read”. Ogni messaggio proveniente dalla macchina QEMU viene decodificato e in base all'operazione richiesta viene chiamata la funzione opportuna, nel caso della lettura il valore restituito della funzione viene inviata alla macchina tramite il proxy server.

Nel file di configurazione sono specificati i pin GPIO come connettori del componente, dopo il caricamento per ogni pin viene copiato e modificato il corrispondente elemento SVG per evidenziare i pin al passare del mouse. Tutti i pin hanno una funzione associata che verrà invocata alla loro scrittura, alcuni di loro che in realtà sono collegati fisicamente a dei componenti come il pin GPIO 16 che è collegato al led “OK/ACT” a cui viene

[6] https://github.com/adafruit/Fritzing-Library

assegnata una funzione apposita che gestisce il led accendendolo alla scrittura del valore "1" e spegnendolo altrimenti.

Led

Il led è un componente molto semplice che può avere la configurazione dell'esempio 2.3.2, il file SVG è stato importato da Fritzing a cui sono state apportate delle leggere modifiche per essere compatibile con la breadboard. Questo è l'unico componente che ha implementato le proprie funzioni "activate" e "deactivate" che sono chiamate alla scrittura della linea GPIO a cui è collegato, in base al valore scritto viene chiamata una di queste funzioni che rendono il led dinamico con un effetto grafico simulando l'accensione e lo spegnimento.

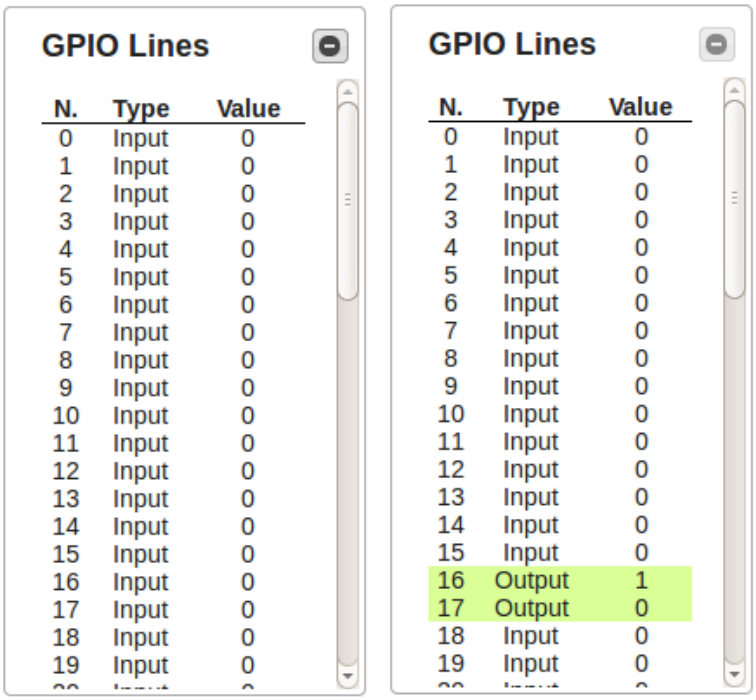
2.3.4 Fili

I fili hanno lo scopo di collegare due pin e vengono rappresentati da delle semplici linee che non necessitano di un file SVG ma possono essere creati al momento dell'esecuzione. Le coordinate necessarie per la creazione di un filo vengono calcolate partendo dai due pin che sono specificati nel file di configurazione del circuito oppure vengono determinati al momento del disegno.

2.3.5 Tabella GPIO

Come è stato detto precedentemente le linee GPIO vengono gestite internamente dal componente Raspberry Pi e per visualizzare il loro stato è stata creata una tabella che viene aggiornata ad ogni cambiamento. La figura 2.2 mostra la tabella in due momenti diversi.

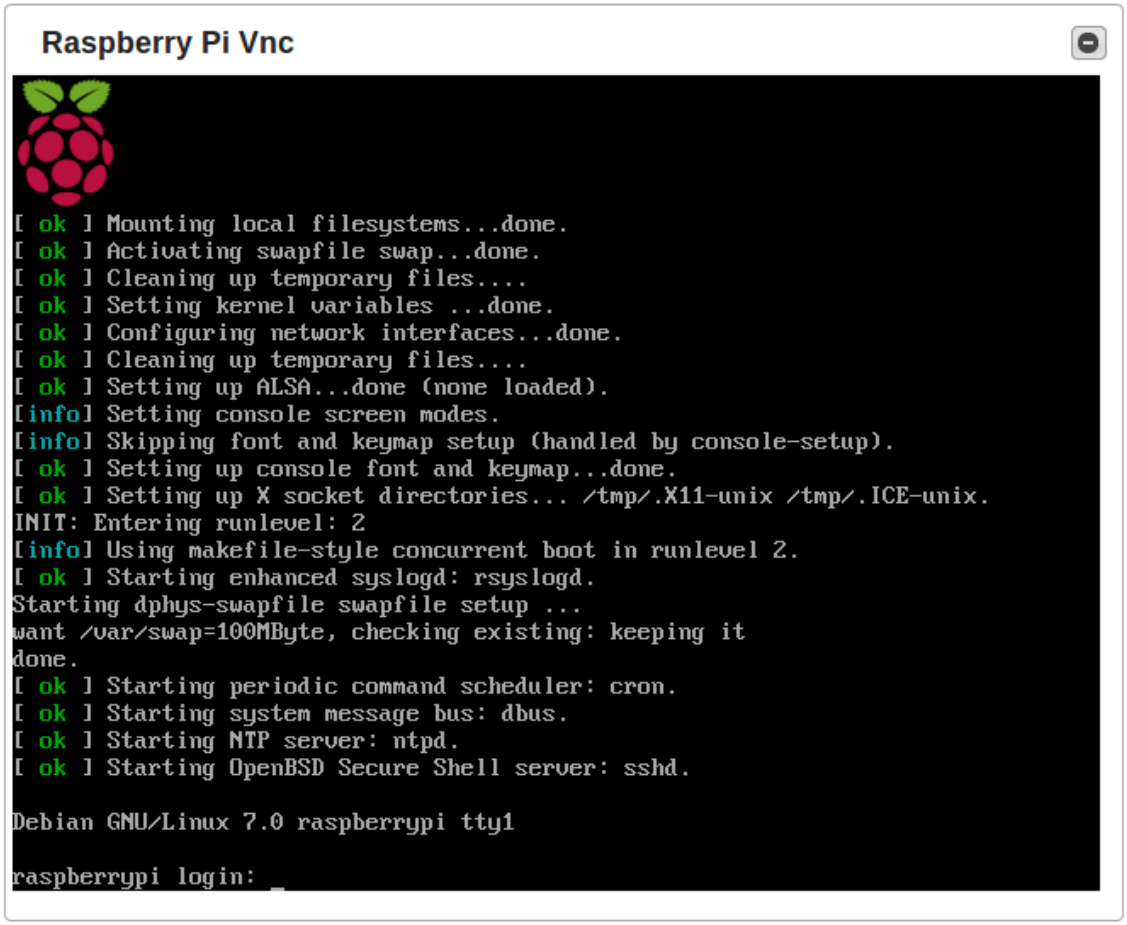
Figura 2.2: Tabella GPIO in momenti diversi



2.3.6 noVNC

Il client VNC “noVNC” viene integrato in una finestra (visibile nella figura 2.3) che può essere spostata a piacere oppure ridotta a icona, inizialmente è nascosta e viene visualizzata all’accensione della macchina QEMU. La connessione di tipo WebSocket viene fatta direttamente con il server VNC di QEMU senza necessità di passare per il proxy, queste è possibile solo se QEMU è stato compilato abilitando il supporto al protocollo WebSocket per VNC.

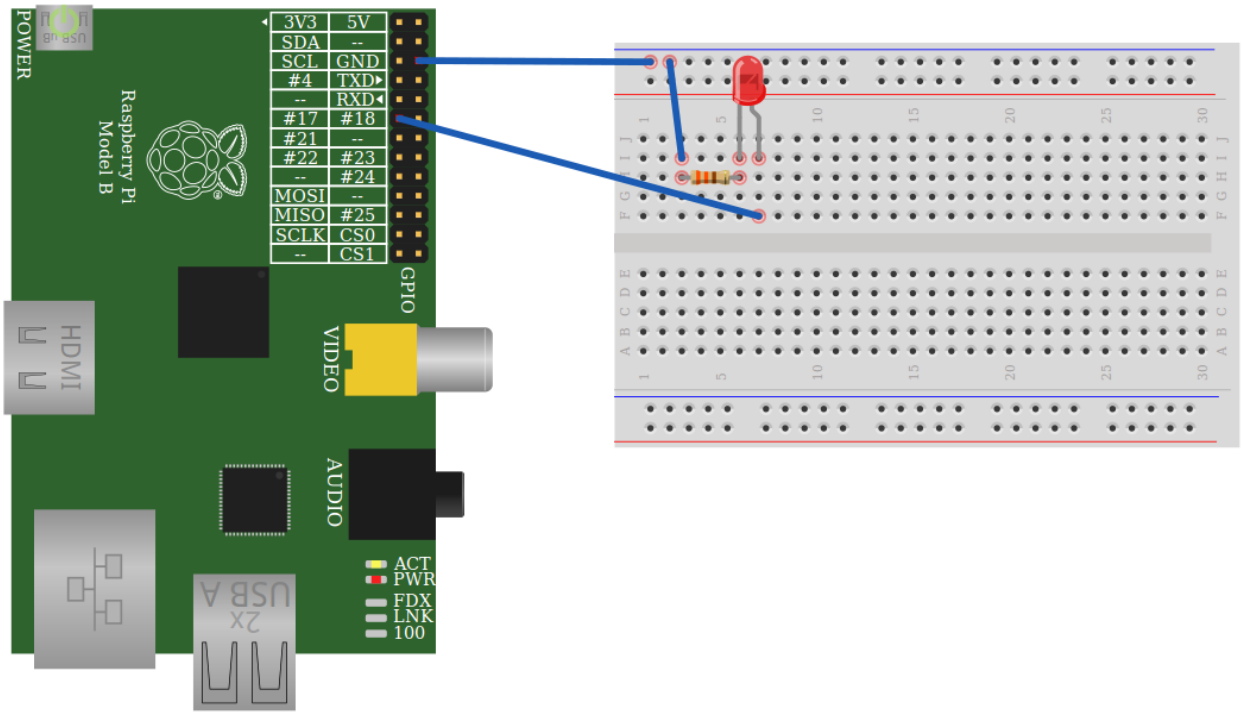
Figura 2.3: Finestra VNC



2.3.7 Esempio di un circuito completo

Per la dimostrazione del funzionamento del software viene usato un circuito (illustrato nella figura 2.4) completo ma minimale, contiene tutti i componenti disponibili. Il circuito è composta da un led e da un resistore collegati alla breadboard e al Raspberry Pi, il led può essere controllato tramite il pin GPIO 17 che deve essere di output per il corretto funzionamento.

Figura 2.4: Circuito completo



Capitolo 3

Evoluzioni future

Questo studio ha portato ad una realizzazione di un emulatore grafico didattico in cui ci sono le basi che possono essere estese con ulteriori studi e implementazioni per arrivare ad un ampio progetto.

3.1 Implementazione di componenti

I componenti realizzati sono pochi e servono a dimostrare il funzionamento dell'emulatore perciò una possibile estensione è l'implementazione di nuovi componenti per poter fare più esperimenti. Grazia alla struttura flessibile e modulare la creazione di nuovi componenti è un compito semplice che è possibile suddividere in modo che diverse persone possano sviluppare parti diverse, la vista SVG può essere progettata e disegnata da un grafico mentre la logica, il controller JavaScript implementato da un programmatore.

3.1.1 Componenti di input

Il supporto ai componenti di input permette di fare circuiti con cui è possibile interagire dando la possibilità di fare esperimenti più complessi e interessanti. Per arrivare a questo scopo è necessario uno studio più approfondito della gestione degli interrupt nel SoC BCM2835 e si dovranno effettuare diverse estensioni al software, dall'emulatore al dispositivo QEMU.

Sensore di temperatura

Uno dei componenti di input molto interessante è il sensore di temperatura dando la possibilità di impostare la temperatura desiderata permetterebbe di testare il comportamento dei software che leggono la temperatura per il controllo di dispositivi. Per essere compatibile con il Raspberry Pi deve essere un sensore digitale e potrebbe essere l'emulazione del sensore DS18S20[1] che usa il protocollo "1-Wire"[2] per la trasmissione dei dati.

3.2 Collegamento seriale

Un'altra possibile estensione è l'implementazione del collegamento seriale tra due Raspberry Pi che permetterebbe di testare la comunicazione seriale senza usare dispositivi fisici. Questa estensione può essere complicata in quanto si devono avere due Raspberry Pi nel circuito e quindi si devono fare le opportune modifiche per la gestione di più macchine QEMU che potrebbero necessitare di tante risorse.

3.3 Simulatore di circuiti elettronici

Uno studio futuro potrebbe essere fatto sulla possibilità di integrare un simulatore di circuiti elettronici questo permetterebbe di poter fare esperimenti più realistici e dare la possibilità di vedere quando certi componenti sono sovraccaricati. Ci sono due alternative, il simulatore può essere implementato appositamente per essere integrato nel progetto oppure cercare tra i progetti liberi un simulatore integrabile.

3.4 Interfaccia di rete

Allo stato attuale la macchina QEMU non ha nessuna interfaccia di rete funzionante e siccome per l'installazione o aggiornamenti dei programmi la rete e indispensabile una buona estensione potrebbe essere un'interfaccia di rete funzionante. Anche in questo

[1] http://www.maximintegrated.com/datasheet/index.mvp/id/2815
[2] http://www.maximintegrated.com/products/1-wire/

caso ci sono diversi approcci, può essere implementata l'emulazione dell'interfaccia di rete presente sul Raspberry Pi, cercare di collegare una scheda di rete USB oppure integrare la rete virtuale VDE[3]. Tutte le alternative sono valide e siccome l'implementazione può essere molto complessa è preferibile cercare di integrare qualcosa che è stato già implementato.

3.5 Scelta della distribuzione

Dare la possibilità all'utente di scegliere la distribuzione da caricare nel Raspberry Pi è un'altra estensione possibile, si vuole ampliare l'utilizzo dell'emulatore anche per testare le diverse distribuzioni senza l'uso delle schede SD. Viste le tante distribuzioni dedicate al Raspberry Pi mettere a disposizione tutte le immagini sarebbe un compito complesso e necessiterebbe tantissimo spazio forse è meglio dare la possibilità all'utente di specificare l'immagine desiderata.

[3]http://vde.sourceforge.net

Conclusioni

Questo lavoro di tesi ha dimostrato la possibilità di implementare un emulatore grafico per circuiti elettronici collegati a Raspberry Pi, creando così un ambiente integrato per progettare piccoli esperimenti. Lo scopo principale di questo progetto è la didattica, infatti permette agli utenti di divertirsi imparando ad usare il Raspberry Pi, la breadboard e altri componenti senza il rischio di danneggiare qualcosa. La possibilità di avere la stessa immagine di Raspbian che si usa nel Raspberry Pi fisico permette di testare il software senza nessuna preoccupazione.

Per svolgere questo progetto è stato necessario studiare in modo approfondito la gestione delle linee GPIO e la struttura dei loro registri. Nella realizzazione, che va dal dispositivo QEMU all'applicazione web passando dal proxy server, sono stati usati diversi linguaggi di programmazione e tecnologie, senza le quali sarebbe stato tutto più complicato. La grafica SVG che permette di creare immagini anche dinamicamente in JavaScript è stata scelta per diversi motivi ma in particolare per la possibilità di poter integrare elementi grafici che sono stati creati per lo sviluppo di altri progetti liberi. L'emulatore grafico è stato progettato per essere modulare dando così la possibilità ad altre persone di contribuire alla creazione di nuovi componenti e di creare nuovi esperimenti.

Bibliografia

[1] Raspberry Pi FAQs, http://www.raspberrypi.org/faqs.

[2] Embedded Linux Wiki - RPi Easy SD Card Setup, http://elinux.org/RPi_Easy_SD_Card_Setup.

[3] BCM2835 ARM Peripherals, http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf.

[4] Raspberry Pi blog, http://www.raspberrypi.org/archives/2180.

[5] noVNC Wiki - Integration, https://github.com/kanaka/noVNC/wiki/Integration.

[6] qemu-rpi - Readme, https://github.com/Torlus/qemu-rpi.

[7] Qemu-devel - Mailing list, http://lists.gnu.org/archive/html/qemu-devel/2012-11/msg01570.html.