Università degli studi di Palermo 2019 - 2020

Raspberry Semaphore

Controllo di un semaforo attraverso un raspberry Pi 3b+ utilizzando forth

Giuseppe Terrasi | Sistemi Embedded | Prof. Daniele Peri

Sommario

[**1.** **Introduzione** 2](#_Toc32274367)

[**2.** **Componenti hardware utilizzati** 3](#_Toc32274368)

[**3.** **Circuito** 4](#_Toc32274369)

[**4.** **Descrizione dell’ambiente di lavoro** 5](#_Toc32274370)

[**5.** **Descrizione drivers** 5](#_Toc32274371)

[5.1. GPIO.F 5](#_Toc32274372)

[5.1.1. Costanti 5](#_Toc32274373)

[5.1.2. GPIO ( n - - n ) 5](#_Toc32274374)

[5.1.3. MODE ( n -- a b c) 5](#_Toc32274375)

[5.1.4. OUTPUT (a b c -- ) 6](#_Toc32274376)

[5.1.5. INPUT (a b c -- ) 6](#_Toc32274377)

[5.1.6. ON ( n -- ) 6](#_Toc32274378)

[5.1.7. OFF ( n -- ) 6](#_Toc32274379)

[5.1.8. LEVEL ( n -- b ) 7](#_Toc32274380)

[5.1.9. GPFSELOUT! ( n -- ) 7](#_Toc32274381)

[5.1.10. GPFSELIN! ( n -- ) 7](#_Toc32274382)

[5.1.11. GPON! ( n -- ) 7](#_Toc32274383)

[5.1.12. GPOFF! ( n -- ) 7](#_Toc32274384)

[5.1.13. GPLEV@ ( n – n1 ) 7](#_Toc32274385)

[5.2. TIME.F 8](#_Toc32274386)

[5.2.1. Costanti 8](#_Toc32274387)

[5.2.2. NOW ( -- n ) 8](#_Toc32274388)

# **Introduzione**

Lo scopo di questo progetto è quello di costruire un semaforo a chiamata di un attraversamento pedonale e di controllarlo attraverso un Raspberry PI 3B+. Oltre al semaforo è presente anche uno display LCD 16x02 su cui viene visualizzato lo stato del semaforo ed il tempo rimanente prima che esso cambi.

Lo stato iniziale del semaforo prevede il verde per le auto ed il rosso per i pedoni e con la funzione di chiamata del rosso per le auto abilitata. Successivamente alla pressione del tasto di chiamata il semaforo inizia la routine di cambio stati di seguito elencati (per ragioni didattiche sono state usate delle tempistiche ridotte):

1. Spegnimento del verde per le auto ed accensione del giallo per esse. Attesa di cinque secondi.
2. Spegnimento del giallo per le auto ed accensione del rosso per esse con contestuale accensione del verde per i pedoni. Attesa di dieci secondi.
3. Spegnimento del verde per i pedoni ed accensione del giallo per essi. Attesa di cinque secondi.
4. Spegnimento del giallo per i pedoni ed accensione del rosso per essi con contestuale accensione del verde per le auto. Inibizione di un’ulteriore chiamata pedonale per dieci secondi.
5. Abilitazione della chiamata pedonale ed attesa indefinita.

Durante tutta la routine il display mostra sulla prima riga lo stato corrente del semaforo e sulla seconda riga il tempo rimanente per esso. Al punto 5 mostra solo sulla prima riga l’abilitazione della chiamata.

L’intero progetto è stato scritto nel linguaggio FORTH ed è eseguito in un ambiente installato sulla memoria SD del Raspberry PI 3B+.

# **Componenti hardware utilizzati**

* N. 1 Raspberry PI 3B+
* N. 1 Breadboard
* N. 1 LCD 16x02
* N. 2 LED Rossi
* N. 2 LED Gialli
* N. 2 LED Verdi
* N. 6 Resistenze da 1KΩ
* N. 1 Pulsante
* N. 1 Potenziometro da 10KΩ
* N. 1 GPIO Extension Board
* N. 1 40 pin GPIO Cable
* N. 1 FT232RL FTDI USB (Uart Serial)
* Jumper wires per i collegamenti

# **Circuito**

# **Descrizione dell’ambiente di lavoro**

Repository GitHub: <https://github.com/giuseppe-terrasi/raspberrypi-semaphore-forth>

1. bootcode.bin 🡪 Raspberry 3B+ bootloader
2. fixup.dat 🡪 Linker file per il Raspberry
3. kernel.img 🡪 Immagine dell’interprete FORTH (pijforth modificato)
4. start.elf 🡪Basic Raspberry Firmware
5. jonesforth.f 🡪 File del repository pjforth (<https://github.com/organix/pijFORTHos/blob/master/jonesforth.f>) per avere le word .” “ – ABORT – s” “ .
6. gpio.f 🡪 driver per l’utilizzo dei GPIO
7. time.f 🡪 driver per l’utilizzo del System Timer
8. lcd.f 🡪 driver per l’utilizzo del display LCD 16x02
9. semaphore.f 🡪 driver per controllare il semaforo
10. LICENSE 🡪 Licenza MIT del progetto
11. README.md 🡪 File di copertina per il repository GitHub

# **Descrizione drivers**

## GPIO.F

### Costanti

**GPFSEL0** = **0x3F200000** 🡪 Registro di controllo dei GPIO da 0 a 9 per la selezione della funzione del GPIO.

**GPSET0 = 0x3F20001C** 🡪 Registro per settare un GPIO da 0 a 30 con il valore “high”

**GPCLR0 = 0x3F200028** 🡪 Registro per settare un GPIO da 0 a 30 con il valore “low”

**GPLEV0 = 0x3F200034** 🡪 Registro contenenti i livelli dei GPIO da 0 a 30

### GPIO ( n - - n )

: **GPIO** DUP 30 > IF ABORT THEN ;

Questa word garantisce che il numero passato in input sia un valore valido per i GPIO. In questo progetto vengono usati i GPIO con numero inferiore a 30.

### MODE ( n -- a b c)

: **MODE** 10 /MOD 4 \* GPFSEL0 + SWAP 3 \* DUP 7 SWAP LSHIFT ROT DUP @ ROT INVERT AND ROT ;

Prende in input il numero del GPIO di cui impostare la modalità e lascia sullo stack il numero di left shift necessari per settare il corrispondente GPIO control bit del registro GPFSELN, dove N è il numero di registro, insieme all’indirizzo del registro GPFSELN ed il valore corrente settato in esso “pulito” da una maschera di bit. Il valore di N ed (a) sono calcolati dividendo il numero del GPIO per 10; N sarà il quoziente moltiplicato per 4 mentre (a) sarà il resto. Il registro GPFSELN è calcolato a partire da GPFSEL0 aggiungendo il valore di N.

( es. GPIO 21 è controllato da GPFSEL2 quindi 21 / 10 --> N = 2 \* 4, a = 1 --> GPFSEL0 + 8 = GPFSEL2 )

I tre bit di controllo del registro GPFSELN che controllano il GPIO vengono impostati a zero tramite una machera di bit ottenuta shiftando verso sinistra il valore 0x7 di un numero di posizioni pari a 3 per il resto della divisione per 10 effettuata in precedenza.

( es. 21 / 10 -> 3 \* 1 -> 7 shiftato a sinistra di 3 )

### OUTPUT (a b c -- )

: **OUTPUT** 1 SWAP LSHIFT OR SWAP ! ;

Prende in input I tre valori lascati sullo stack dalla word MODE e setta la modalità output per il GPIO nel registro GPFSELN. Il bit del registro GPFSELN che controlla la modalità output del GPIO è settato tramite l’OR tra il valore memorizzato nel registro GPFSELN, “pulito” dalla maschera di bit, e un 1 shiftato a sinistra di un numero di posizioni pari al resto della divisione per 10 effettuata dalla word MODE moltiplicato per 3.

( es. Con GPIO 21 e @GPFSEL2: 011010--> 111000 011010 INVERT AND --> 000010 001000 OR --> 001010 )

### INPUT (a b c -- )

: **INPUT** 1 SWAP LSHIFT INVERT AND SWAP ! ;

Prende in input I tre valori lascati sullo stack dalla word MODE e setta la modalità input per il GPIO nel registro GPFSELN.

Stesso meccanismo della word OUTPUT ma utilizza l’operazione INVERT AND invece dell’OR

### ON ( n -- )

: **ON** 1 SWAP LSHIFT GPSET0 ! ;

Prende il numero del GPIO dallo stack, effettua un left shift del valore 1 di un numero di posizioni pari al numero del GPIO e setta il bit corrispondente del registro GPSET0.

### OFF ( n -- )

: **OFF** 1 SWAP LSHIFT GPCLR0 ! ;

Prende il numero del GPIO dallo stack, effettua un left shift del valore 1 di un numero di posizioni pari al numero del GPIO e setta il bit corrispondente del registro GPCLR0.

### LEVEL ( n -- b )

: **LEVEL** 1 SWAP LSHIFT GPLEV0 @ SWAP AND ;

Prende il numero del GPIO dallo stack, effettua un left shift del valore 1 di un numero di posizioni pari al numero del GPIO, legge il valore del bit corrispondente del registro GPLEV0 e lo lascia sullo stack.

### GPFSELOUT! ( n -- )

: **GPFSELOUT!** GPIO MODE OUTPUT ;

Scorciatoia per settate la modalità output di un GPIO

### GPFSELIN! ( n -- )

: **GPFSELIN!** GPIO MODE INPUT ;

Scorciatoia per settate la modalità input di un GPIO

### GPON! ( n -- )

: **GPON!** GPIO ON ;

Scorciatoia per settare un GPIO con il valore “high”.

### GPOFF! ( n -- )

: **GPOFF!** GPIO OFF ;

Scorciatoia per settare un GPIO con il valore “low”.

### GPLEV@ ( n – n1 )

: **GPLEV@** GPIO LEVEL ;

Scorciatoia per leggere il valore del GPIO passato in input.

## TIME.F

### Costanti

**SYSCS** = **0x3F003000** 🡪 Registro di controllo / stato del System Timer

**SYSCLO** = **0x3F003004** 🡪 Registro contenente i 32 bit inferiori del System Timer Counter

**SYSCHI** = **0x3F003008** 🡪 Registro contenente i 32 bit superiori del System Timer Counter

**SYSC1** = **0x3F003010** 🡪 Registro del System Timer Compare 1

**SYSC3** = **0x3F003018** 🡪 Registro del System Timer Compare 3

### NOW ( -- n )

: **NOW** SYSCLO @ ;

Legge il valore attuale dei 32 bit inferiori del System Timer Counter corrispondenti al valore in microsecondi dall’avvio del sistema (il valore esatto è l’intero valore a 64 bit contenuto in SYSCLO e SYSCHI. In questo progetto è sufficiente leggere il solo valore di SYSCLO)