Matrix GSM Alarm

Luca Giuriato

26 aprile 2010

Indice

1 Introduzione **7**

I Progetto hardware **9**

2 Sensore PIR **11**
2.1 Effetto piroelettrico 11
2.2 Lenti di Fresnel 12
2.3 Modalità d'utilizzo 12

3 Cicuito di condizionamento del segnale **15**
3.1 Filtro attivo 16
3.1.1 Primo stadio 16
3.1.2 Secondo stadio 18
3.2 Comparatore a finestra 21

4 Microcontrollore **25**
4.1 Caratteristiche elettriche 25
4.2 Configurazione circuitale di base 26
4.3 Interfaccia di programmazione ISP 27
4.3.1 SPI 27
4.3.2 Modalità di programmazione 28

5 Modem GSM **29**
5.1 Caratteristiche elettriche 29
5.2 Procedura d'accensione 30

II Progetto software **31**

6 Operazioni principali **33**
6.1 Ciclo principale 33

6.2 Ciclo d'allarme 35

7 Inizializzazione **37**
7.1 Configurazione 37
7.2 EEPROM 38
7.3 Watchdog 39

8 Lettura degli ingressi **41**
8.1 Struttura del filtro 41
8.2 Implemetazione software 42

9 Lettura della batteria **43**
9.1 ADC 43
9.2 Soluzione circuitale 43
9.3 Ciclo di lettura 45

10 Comunicazione con il modem **49**
10.1 UART 49
10.2 Configurazione 50
10.3 Sintassi comandi 51
10.4 Ciclo di comando 51
10.4.1 Invio comando 51
10.4.2 Ricezione ed analisi comando 53

11 Gestione SMS **57**
11.1 Invio SMS 57
11.2 Ricezione SMS 58

12 Gestione chiamate **61**
12.1 Esecuzione chiamate 61
12.2 Ricezione chiamate 64

III Prodotto finale **67**

13 Caratteristiche del prodotto **69**
13.1 Funzionalità 70
13.2 Guida per l'utente 70

14 Considerazioni finali **73**
14.1 Problemi riscontrati 73
14.2 Evoluzioni future 74

14.2.1 Interfaccia generica 74
14.2.2 Interfaccia seriale 75

Bibliografia **77**

Capitolo 1

Introduzione

Il mondo dei sistemi d'allarme è ricco e variegato, si possono trovare in commercio una grande quantità di soluzioni adatte a quasi ogni situazione. Molto in voga negli ultimi anni sono i sistemi d'allarme composti da reti di sensori collegati in wireless, in grado di coprire un'area anche molto ampia utilizzando diverse tipologie di sensori, per far fronte a diverse necessità di controllo. Possiamo trovare sensori magnetici o inerziali dedicati a verificare un'eventuale apertura di una porta o una finestra, sensori sismici in grado di rilevare le vibrazioni dovute al passaggio di una persona, barriere ad ultrasuoni o all'infrarosso dedicate alla protezione di zone anche molto ampie.

Per il rilievo della presenza umana in una zona circoscritta invece vengono utilizzati soprattutto sensori passivi all'infrarosso, che negli anni hanno dimostrato la loro grande affidabilità nonostante non necessitino di sistemi complessi per funzionare, tanto che vengono comunemente utilizzati per automatizzare l'accensione di luci o l'apertura di porte. Se associati ad un buon circuito amplificatore sono in grado di raggiungere una precisione molto

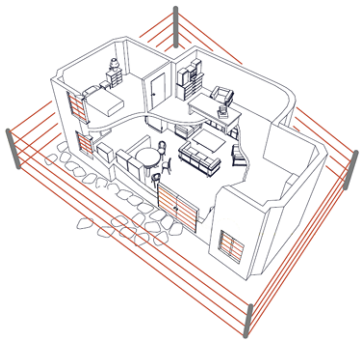


Figura 1.1: Esempio di protezione tramite barriera all'infrarosso.

Figura 1.2: Comune sensore infrarosso commerciale.

elevata. L'idea alla base di questo progetto è il desiderio di creare un sistema d'allarme plug&play semplice e diretto, facilmente utilizzabile dall'utente senza la necessità di complicate e costose installazioni, pur garantendo un'alta affidabilità del servizio. Nasce così un apparecchio dotato di batteria che può essere posizionato ovunque si voglia, che grazie al suo sensore ad infrarosso può captare la presenza umana fino a diversi metri di distanza, e contenente un modem GSM in modo che possa essere controllato dall'utente in qualsiasi situazione, utilizzando comodamente il proprio cellulare. Otteniamo così un sistema d'allarme molto semplice ma flessibile e dalle grandi potenzialità, utilizzabile in sicurezza nella maggior parte dei casi più comuni.

Le pagine successive contengono un tour riassuntivo delle modalità in cui questo progetto è stato realizzato, sia da parte hardware che software, per poi mostrare brevemente l'interfaccia di utilizzo e configurazione dedicata all'utente.

Parte I

Progetto hardware

Capitolo 2

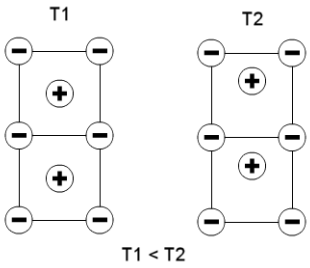
Sensore PIR

Il sistema utilizza un sensore PIR (*Passive InfraRed*), in grado di rilevare la radiazione infrarossa dovuta ad una presenza umana nella zona sotto controllo. Questo tipo di sensori è ritenuto molto affidabile e viene largamente utilizzato in molti settori ove è necessario rilevare la presenza umana, come ad esempio i sistemi di avvio automatico di porte e luci.

2.1 Effetto piroelettrico

Il cuore del sensore è il materiale cristallino al suo interno, sensibile alle radiazioni infrarosse grazie alle sue proprietà piroelettriche. Questà proprietà rende un materiale capace di generare una differenza di potenziale ai suoi capi proporzionale al calore assorbito, dovuto in questo caso all'emissione di onde infrarosse da parte di un corpo caldo quale un animale o un essere umano.

La piroelettricità è un effetto secondario che si può riscontrare in alcuni materiali piezoelettrici. Questa particolare famiglia di materiali è caratterizzata da una struttura cristallina principale molto stabile, dove al suo interno però è posto un ulteriore struttura atomica libera di muoversi all'interno della propria cella, causando così una polarizzazione spontanea del materiale a seguito di un suo spostamento dovuto all'agitazione termica.



L'energia termica assorbita causa la variazione della polarizzazione spontanea del materiale, generando così una piccola carica elettrica superficiale. Tramite l'analisi di questa differenza di potenziale generata è possibile così determinare la presenza di un corpo in grado di emettere radiazione infrarossa nel raggio d'azione del sensore.

2.2 Lenti di Fresnel

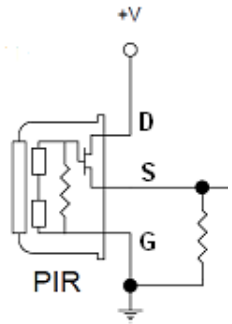
Per aumentare la capacità del sensore di rilevare la presenza umana viene aggiunta una particolare lente detta *lente di Fresnel*. Questa famiglia di lenti è in grado di aumentare notevolmente la portata del sensore mantenendo contenute le proprie dimensioni, a differenza dalle comuni lenti convesse.

Una lente di Fresnel è caratterizzata da una buona capacità di concentrare i raggi provenienti da diverse angolature pur mantenendo uno spessore molto contenuto. Anche se a prima vista potrebbe sembrare una comune lente a superficie liscia ad un'analisi più attenta si nota che è composta da un insieme di sezioni circolari, ognuna con un'inclinazione tale da poter riflettere i raggi provenienti da un determinato angolo.

La lente è composta da un materiale in grado di trasmettere radiazione infrarossa di lunghezza d'onda compresa tra gli 8 e i 14 μm, rendendola maggiormente sensibile alle emissioni umane e ponendo quindi un primo livello di filtro.

2.3 Modalità d'utilizzo

Internamente il sensore presenta un FET di amplificazione e disaccoppiamento del segnale proveniente dall'elemento cristallino piroelettrico, permettendo così una migliore misura e stabilità del segnale.



Simbolo	Descrizione	Valore tipico
D	alimentazione	[2V,15V]
S	segnale	5mVpp
G	massa	0V

Il package si presenta con 3 pin, drain e ground dedicati all'alimentazione e source per la lettura del segnale elettrico generato. Mentre ground rappresenta la massa, al drain va collegata la tensione di alimentazione, che può essere scelta nel range 2.2V e 15V e va opportunamente filtrata tramite un filtro RC. Sul pin source ritroviamo un segnale in tensione rappresentativo della radiazione infrarossa rilevata istantaneamente, di ampiezza tipica 5mV e offset compreso tra 0.6 e 0.8V.

Normalmente per essere correttamente utilizzato questo segnale va opportunamente amplificato e filtrato da un circuito di condizionamento, ed inoltre va posta una resistenza di carico tra source e massa del valore tipico di 100kΩ.

Capitolo 3

Cicuito di condizionamento del segnale

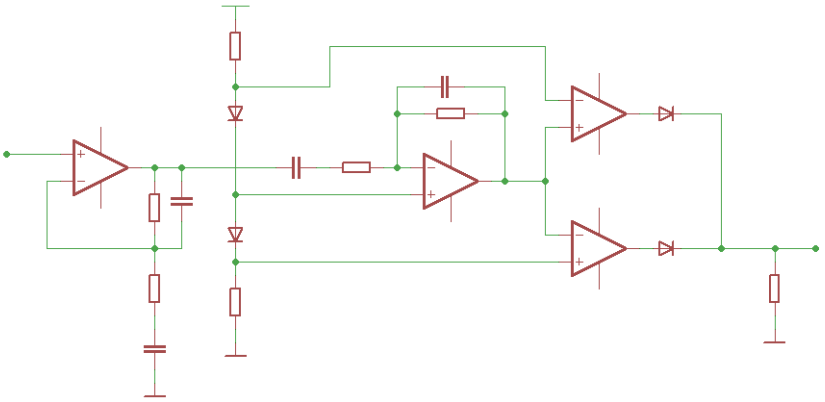
Il segnale prodotto dal sensore è molto debole e questo lo rende sia sensibile ad eventuali disturbi che difficile da utilizzare. Per questo motivo l'utilizzo di un circuito di amplificazione e filtro del segnale risulta indispensabile.

Il circuito di amplificazione del segnale è composto da 2 blocchi principali:

- filtro attivo
- comparatore a finestra

Al primo viene affidato il compito di amplificare sufficientemente il segnale attenuando a dovere il rumore, in modo da poterlo analizzare in sicurezza. Il comparatore invece analizza il segnale amplificato e provvede a fornire un valore digitale al microcontrollore, di livello adeguato in modo che ne possa venire facilmente riconosciuto il valore logico.

Schema elettrico:



3.1 Filtro attivo

Il segnale V_{PIR} in uscita dal sensore PIR si può scomporre in 2 componenti principali

- una componente alternata
- un valore di offset costante

L'offset è dovuto alla polarizzazione parziale dei componenti attivi del sensore dovuta a fattori secondari, quali la temperatura ambiente, eventuali vibrazioni o l'esposizione alla luce solare. Quando invece il sensore rileva il passaggio di un corpo caldo genera un segnale molto debole a bassa frequenza (legata alla velocità di movimento del corpo), che va a sovrapporsi al valore di offset. Questo è il segnale che ci interessa, in grado di discriminare la presenza o l'assenza di un essere umano di fronte al sensore.

Riassumento, in uscita dal sensore PIR ritroviamo un segnale V_{PIR} caratterizzato da:

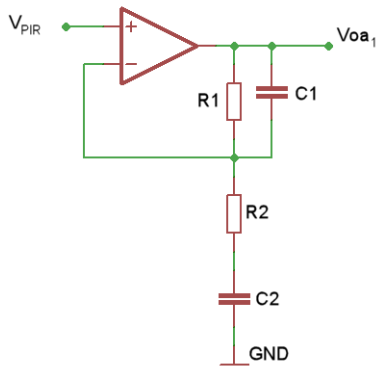
- offset costante $V_{OFF} = 1V$ circa
- segnale di ampiezza $V_S = \pm 5mV$
- frequenza del segnale compresa tra 0.1Hz e 10Hz

Di conseguenza il filtro attivo dovrà

- attenuare i segnali a bassa frequenza, in modo da eliminare l'offset
- attenuare i segnali ad alta frequenza, in modo da aumentare la resistenza al rumore
- amplificare i segnali di frequenza compresa tra 0.1Hz e 10Hz

3.1.1 Primo stadio

Questo stadio è composto da un comune amplificatore operazionale in configurazione non invertente.



$$R_1 = 1M\Omega \quad R_2 = 10k\Omega$$
$$C_1 = 1\mu F \quad C_2 = 10\mu F$$

Considerando le impedenze $Z_1 = R_1 \parallel C_1$ (il parallelo tra R_1 e C_1) e $Z_2 = R_2 + C_2$ (la serie tra R_2 e C_2) il calcolo della funzione di trasferimento $G_1(s)$ risulta più agevole

$$G_1(s) = 1 + \frac{Z_1}{Z_2} = \frac{Z_1 + Z_2}{Z_2}$$

che numericamente diventa

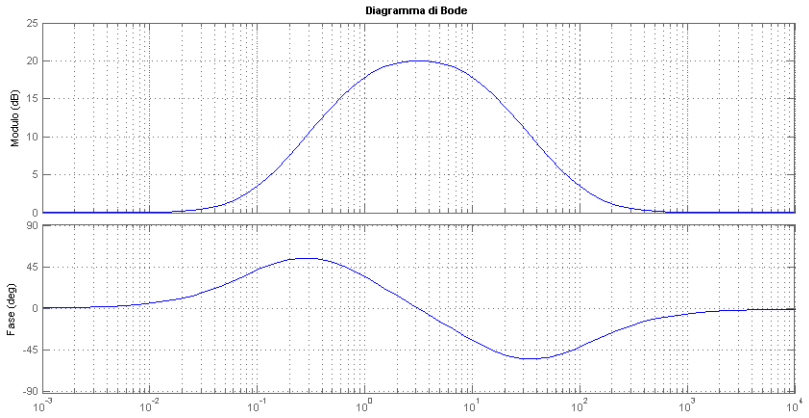
$$G_1(s) = \frac{0.1s^2 + 11.1s + 1}{0.1s^2 + 1.1s + 1}$$

Questa funzione di trasferimento presenta 2 zeri e 2 poli, che determinano la banda di frequenze entro cui il segnale viene amplificato.

$$z_1 = -110.9 \quad p_1 = -1$$
$$z_2 = -0.09 \quad p_2 = -10$$

In particolare il segnale otterrà un'amplificazione massima nella banda compresa tra 0.1Hz e 10Hz pari a 20dB, mentre per valori di frequenza esterni a questo range il segnale subirà un'amplificazione nulla o trascurabile.

Tutto questo si nota anche grazie al diagramma di Bode



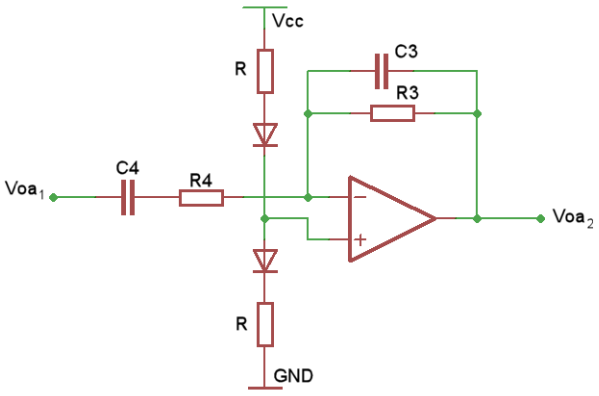
Riassumendo, questo primo stadio provvede ad un'amplificazione massima pari a circa *20dB* dei segnali con frequenza compresa tra *1Hz e 10Hz*, lasciando quasi inalterati i segnali con frequenza esterna alla banda [0.1Hz , 100Hz].

3.1.2 Secondo stadio

Anche questo stadio di amplificazione è composto da un amplificatore operazionale collegato tramite una rete RC, ma risulta leggermente più complesso in quanto ai suoi ingressi l'operazionale vede 2 segnali distinti:

Ingresso invertente: segnale pre-amplificato dal primo stadio

Ingresso non invertente: valore fisso pari a $V_{cc}/2$



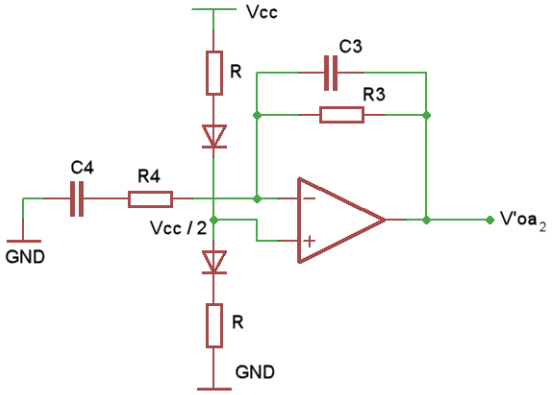
Per analizzare correttamente la situazione è conveniente suddividere il comportamento di tale circuiteria secondo il principio di sovrapposizione degli effetti. Secondo questo principio, la tensione in uscita V_{oa2} risulta essere la somma di 2 contributi, V'_{oa2} derivante dall'amplificazione del segnale posto

sull'ingresso non invertente V_+ e V''_{oa2}, causato dall'amplificazione del segnale posto sull'ingresso invertente V_{oa1}. In questo modo possiamo analizzare separatamente i 2 circuiti che si ottengono e successivamente sommare il loro risultato.

Prima parte

Per prima cosa analizziamo come viene trattato il segnale presente sul pin non invertente.

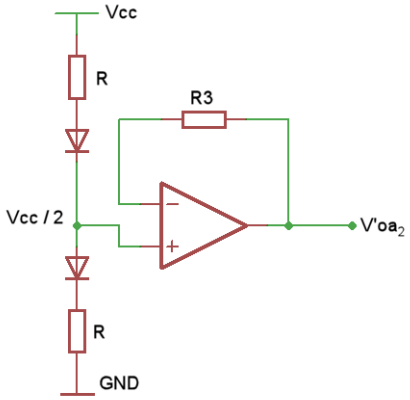
Ponendo a massa il segnale V_{oa1} otteniamo la classica configurazione non invertente



ma prima di procedere nell'analisi possiamo introdurre un'ulteriore semplificazione, viste le caratteristiche del segnale V_+.

$$V_+ = \frac{V_{cc}}{2}$$

Essendo un segnale costante, si può considerare che a regime i condensatori vengono assimilati ad un circuito aperto, semplificando enormemente il circuito risultante

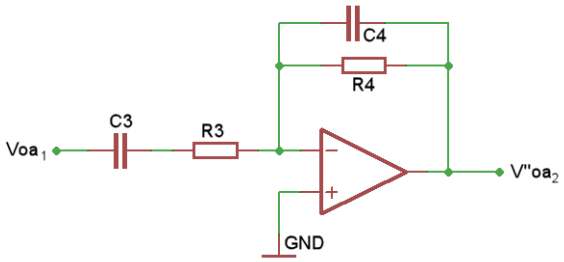


che diventa un semplice inseguitore di tensione, un buffer non invertente. Di conseguenza

$$V'_{oa2} = V_+ = \frac{V_{cc}}{2}$$

Seconda parte

Viceversa, ponendo a massa il pin non invertente otteniamo uno stadio amplificatore in configurazione invertente.



$$R_3 = 1M\Omega \quad R_4 = 10k\Omega$$
$$C_3 = 1\mu F \quad C_4 = 10\mu F$$

Anche in questo caso per semplificare il calcolo della funzione di trasferimento $G_2(s)$ consideriamo le impedenze $Z_3 = R_3 \parallel C_3$ ottenuta dal parallelo tra R_3 e C_3, e $Z_4 = R_4 + C_4$ rappresentante la serie tra R_4 e C_4. Di conseguenza

$$G_2(s) = -\frac{Z_3}{Z_4}$$

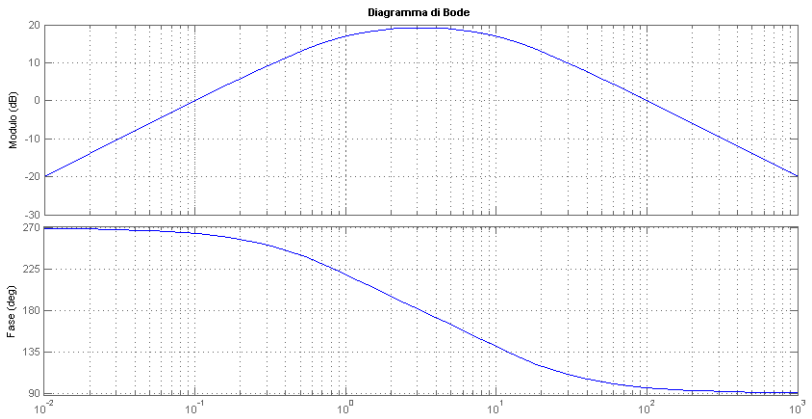
che numericamente diventa

$$G_2(s) = \frac{-10s}{0.1s^2 + 1.1s + 1}$$

Questa funzione di trasferimento presenta uno zero nell'origine e 2 poli.

$$z = 0 \quad p_1 = -1 \quad p_2 = -10$$

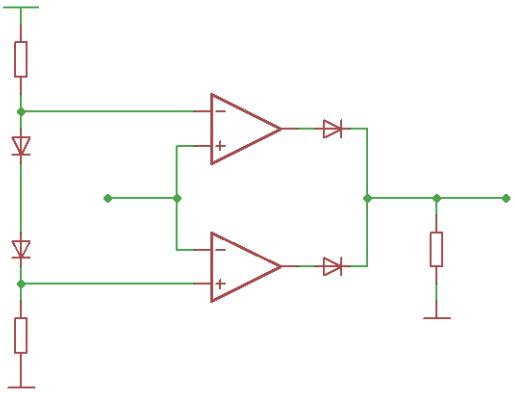
In modo analogo allo stadio precedente, anche in questo caso assistiamo all'amplificazione dei soli segnali appartenenti ad una determinata banda di frequenza, con la differenza che i segnali esterni risultano tanto più attenuati quanto più ci si allontana in frequenza. Il comportamento del circuito risulta chiaro grazie relativo al diagramma di Bode



Qui si nota un'attenuazione dei segnali con frequenza esterna alla banda [0.1Hz , 100Hz] mentre per i segnali compresi tra *1Hz e 10Hz* otteniamo un'amplificazione massima pari a *20dB*.

3.2 Comparatore a finestra

Questo è lo stadio finale del processo di condizionamento del segnale, indispensabile per interfacciarsi correttamente al microcontrollore.



Questo circuito è composto da 2 amplificatori operazionali funzionanti come comparatori, che cioè provvedono ad amplificare fino alla saturazione la differenza tra i segnali V_+ e V_-, presenti sui rispettivi pin dell'integrato. In questo modo se V_+ è maggiore di V_- otterremo in uscita il valore massimo, pari quindi a V_{cc}, mentre in caso contrario otterremmo in uscita il valore minimo, pari quindi a $0V$.

L'uscita dei 2 operazionali viene infine collegata insieme, cosicchè il suo valore

risulti dalla somma dei loro 2 contributi

$$V_o = V_{oc1} + V_{oc2}$$

anche se in realtà il funzionamento di questo circuito prevede che possano influire sull'uscita al massimo uno per volta, evitando ogni eventuale conflitto dovuto alla condivisione della linea d'uscita.

Il primo comparatore riporta sul pin V_+ il segnale V_{oa2} proveniente da precedenti stadi di amplificazione, mentre sul pin V_- ritrova un valore costante pari a

$$V_- = \frac{V_{cc}}{2} + 200mV$$

dove i 200mV sono dovuti alla tensione di conduzione del diodo D_1. In questo modo:

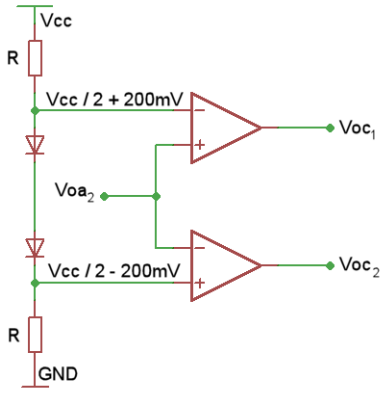
$$V_{oc1} = \begin{cases} 0V, & \text{se } V_{oa2} \text{ è minore di } \frac{V_{cc}}{2} + 200mV, \\ V_{cc}, & \text{se } V_{oa2} \text{ è maggiore di } \frac{V_{cc}}{2} + 200mV \end{cases}$$

Analogamente il secondo comparatore riporta stavolta sul pin V_- il segnale V_{oa2} proveniente da precedenti stadi di amplificazione, mentre sul pin V_+ ritrova un valore costante pari a

$$V_+ = \frac{V_{cc}}{2} - 200mV$$

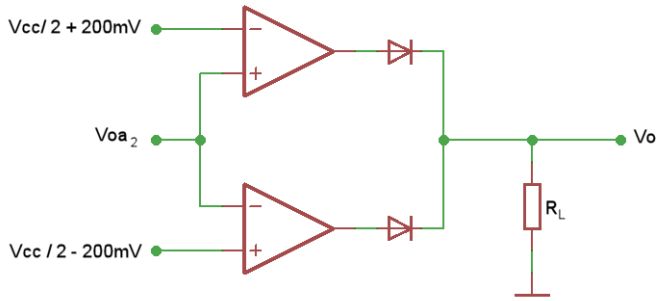
dove i 200mV sono sempre dovuti alla tensione di conduzione del diodo D_2. In questo caso:

$$V_{oc2} = \begin{cases} 0V, & \text{se } V_{oa2} \text{ è maggiore di } \frac{V_{cc}}{2} - 200mV, \\ V_{cc}, & \text{se } V_{oa2} \text{ è minore di } \frac{V_{cc}}{2} - 200mV \end{cases}$$



Per ovviare agli inevitabili conflitti che si avrebbero collegando direttamente insieme le uscite dei 2 operazionali, sono stati inseriti in serie a questi dei diodi, isolando così l'operazionale dalla linea d'uscita quando la sua uscita è pari a $0V$. In questo modo possiamo suddividere il risultato finale in 3 diversi casi:

- $V_{oa2} > V_{cc} + 200mV \Rightarrow V_o = V_{cc}$ impostata dal primo comparatore
- $V_{cc} - 200mV < V_{oa2} < V_{cc} + 200mV \Rightarrow V_o = 0V$ grazie alla resistenza di carico collegata a massa
- $V_{oa2} < V_{cc} - 200mV \Rightarrow V_o = V_{cc}$ impostata dal secondo comparatore



Sommando i contributi dei 2 comparatori otteniamo un circuito che assicura di mantenere l'uscita a livello basso se il segnale amplificato V_{oa2} rimane all'interno di una finestra di $\pm 200mV$ nei dintorni di $\frac{V_{cc}}{2}$, mentre la porta velocemente a V_{cc} nel caso si esca da tale range, quindi in presenza di un segnale utile dovuto a monte al passaggio di un corpo caldo di fronte al sensore.

Grazie a tutto questo, al microcontrollore è sufficiente verificare lo stato logico di questa linea per identificare l'avverarsi o meno di uno stato d'allarme e reagire di conseguenza.

Capitolo 4

Microcontrollore

Come logica di controllo del sistema stato scelto un microcontrollore Atmel ATMega32.

Questo integrato general purpouse dispone di 32 linee di I/O programmabili e svariate periferiche tra cui

- 2 timer a 8 bit
- 1 timer a 16 bit
- convertitore ADC a 10 bit e 8 canali
- interfaccia seriale UART
- watchdog interno
- memoria EEPROM interna da 1024 bit

che lo rendono una soluzione valida per la gestione di svariate tipologie di sistemi, da complesse automazioni industriali a più semplici applicazioni embedded.

4.1 Caratteristiche elettriche

Il circuito integrato è operativo a tensioni di alimentazione comprese tra 2.7V e 6V, dove la tensione minima è determinata dal Brown-out detector, un circuito di sicurezza che provvede a resettare automaticamente il microcontrollore nel caso in cui la tensione di alimentazione scenda al di sotto del minimo stabilito.

La corrente assorbita invece raggiunge un massimo di 15mA, a cui vanno

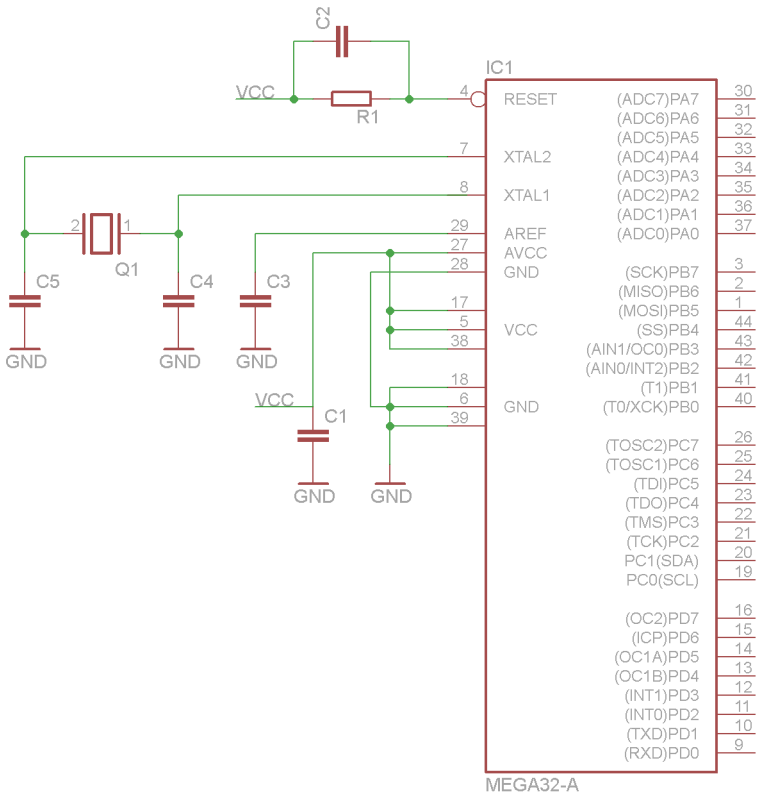
sommate le eventuali correnti dovute al pilotaggio dei pin in output o l'utilizzo di periferiche quali UART o l'ADC, fino ad un massimo globale di 200mA.

$V_{MAX}(V)$	$V_{nom}(V)$	$V_{min}(V)$	$I_{avgMAX}(mA)$
6	3.3	2.7	200

4.2 Configurazione circuitale di base

Per funzionare al meglio il microcontrollore ha bisogno che tutte le sue linee di alimentazione siano collegate, preferibilmente con l'aggiunta di un condensatore di stabilizzazione.

Nel caso non si utilizzi l'oscillatore interno è necessario collegare un quarzo tra gli ingressi XTAL1 e XTAL2, riportandolo a massa tramite altri 2 condensatori come mostrato in figura.



Particolare attenzione è necessaria per la linea di reset, che va alimentata tramite un filtro RC per evitarne il più possibile eventuali sbalzi dovuti a disturbi esterni, che potrebbero causare malfunzionamenti del sistema.

Infine visto l'utilizzo nel programma del circuito di conversione analogico digitale è consigliato collegare a massa tramite un condensatore di stabilizzazione il pin A_{REF}. Internamente questo pin è collegato alla sorgente della tensione di riferimento utilizzata per la conversione, che per l'appunto necessita di essere mantenuta stabile.

Ora tutto quello che manca sono i collegamenti con la circuiteria esterna, quale eventuali led di segnalazione o il modem GSM, che va collegata utilizzando le 32 linee di I/O libere.

4.3 Interfaccia di programmazione ISP

Questo microcontrollore provvede un sistema di programmazione *in-circuit* tramite l'interfaccia di comunicazione SPI interna, permettendo così l'aggiornamento del firmware salvato nel chip senza la necessità di togliere lo stesso dalla scheda.

4.3.1 SPI

L'interfaccia SPI (*Serial Peripheral Interface*) classica viene normalmente utilizzata per la trasmissione di dati tra più dispositivi collegati in una configurazione *Master Slave*. Essa presenta 4 pin fondamentali:

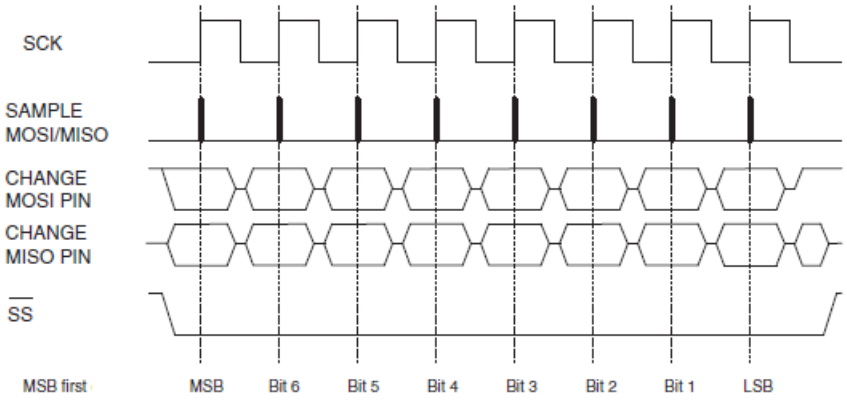
SS Slave Select, sincronia e inizializzazione trasmissioni

SCK Serial ClocK, segnale di temporizzazione dei dati

MISO Master In Slave Out, ricezione dati

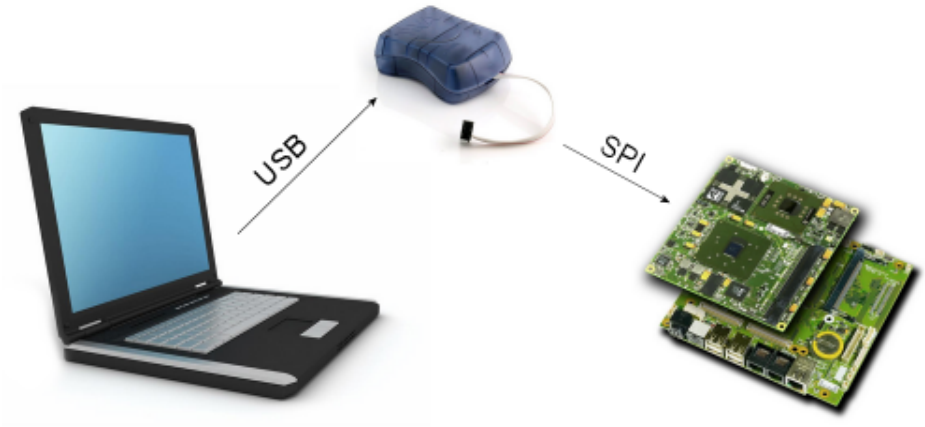
MOSI Master Out Slave In, trasmissione dati

La trasmissione inizia quando il master pone $\overline{SS}$ a livello logico basso. Successivamente viene generata un'onda quadra di frequenza costante sul pin SCK ed il dato da trasmettere viene posto un bit alla volta sul pin MOSI, in modo che il suo valore logico sia costante durante il fronte scelto di SCK permettendone così una corretta lettura da parte degli slave. La risposta da parte dei dispositivi slave avviene in modo similare con la particolarità che il dato trasmesso viene posto sul pin MISO.



4.3.2 Modalità di programmazione

Per programmare il microcontrollore viene utilizzato un programmatore, cioè uno strumento elettronico apposito che viene collegato tra il computer e la scheda in questione.



In questo modo è possibile scrivere nella flash del microcontrollore il programma precedentemente compilato sul computer, in modo che questo possa venire eseguito.

Capitolo 5

Modem GSM

Per la comunicazione tra il sistema e l'utente è stato scelto l'utilizzo di un modem GSM. In questo modo è possibile controllare la centralina d'allarme in qualsiasi momento tramite un comune telefoni cellulare.

Per questo progetto è stato scelto di utilizzare un modem GSM Telit GM862, comandato tramite semplici comandi testuali inviati attraverso la porta seriale UART direttamente dal microcontrollore. L'enorme quantità di comandi riconosciuti fanno parte dello standard AT e permettono l'esecuzione di numerose operazioni quali

- la gestione della rubrica telefonica
- la gestione della connessione alla rete mobile
- l'invio e la ricezione di sms
- l'esecuzione e la ricezione di chiamate vocali

oltre ad altre opzioni avanzate quali la trasmissione dati GPRS e l'invio di fax.

Grazie a questo complesso dispositivo è possibile eseguire operazioni avanzate quali l'utilizzo della rete GSM senza la necessità di eccessive risorse hardware e software, rendendo così più semplice lo sviluppo del progetto.

5.1 Caratteristiche elettriche

Il modem è operativo a tensioni di alimentazioni comprese tra 3.4V e un massimo di 4.2V. La quantità di corrente assorbita varia enormemente in base alla fase di funzionamento, partendo da un minimo di 4mA in standby fino

ad un picco di corrente di quasi 2A nelle fasi di trasmissione su rete GSM.

Risulta così essenziale un progetto accurato del sistema di alimentazione in grado di fornire la corrente adeguata in ogni momento senza causare eccessive oscillazioni che potrebbero portare ad instabilità del sistema.

$V_{MAX}(V)$	$V_{nom}(V)$	$V_{min}(V)$	$I_{avgMAX}(mA)$	$I_{ppmax}(A)$
4.2	3.8	3.4	350	1.9

5.2 Procedura d'accensione

Lo stato d'accensione del modem si controlla tramite i pin ON#, RESET# e PWRCTL.

Sul pin PWRCTL è possibile monitorare lo stato del modem, mentre l'accensione e lo spegnimento si comandano attraverso ON#.

Valore PWRCTL	**Stato**
LOW	modem spento
HIGH	modem acceso

Questa linea viene mantenuta normalmente a riposo a livello alto,e se abbassata per un periodo di circa 1s impone al modem il cambio di stato. In questo modo in seguito ad un impulso di questo tipo se il modem è spento si accende e viceversa.

Viene fornita anche un'ulteriore linea d'emergenza per gestire eventuali blocchi del modulo. Nel caso in cui il modem non risponda correttamente è possibile ripristinare il suo funzionamento tramite un reset hardware attraverso il pin RESET#. Anche RESET# viene normalmente mantenuto a riposo a livello alto, mentre portandolo a livello basso per almeno 200ms si causa un reset forzato del modulo a prescindere dalle operazioni in corso.

Parte II

Progetto software

Capitolo 6

Operazioni principali

Per il corretto funzionamento del sistema il microcontrollore deve eseguire ciclicamente delle istruzioni di base che permettano

- di conoscere lo stato dell'allarme
- di conoscere lo stato della batteria
- di leggere e processare eventuali sms ricevuti
- di gestire la ricezione delle chiamate

in modo da essere sempre in grado di rispondere a questi eventi.

Ognuna di queste operazioni di base provvede autonomamente all'avvio di una sottoprocedura dedicata quando dovuto, quindi ad esempio in caso di rilievo di un segnale d'allarme partirà automaticamente la procedura di emergenza.

6.1 Ciclo principale

In seguito ad ogni reset l'esecuzione del programma inizia dalla funzione $main()$. Questa funzione è suddivisa in 2 parti principali:

- inizializzazione
- ciclo di programma

la prima eseguita una sola volta ad ogni reset, la seconda invece ripetuta ciclicamente. Seguendo questo concetto la struttura base di ogni programma risulta sempre la stessa:

```
void main(void)
{
    // inizializzazione sistema
    init();

    // ciclo principale
    while(1)
    {
        ...
    }
}
```

L'utilizzo di un ciclo infinito (qui definito da *while*(1)) è strettamente necessario, altrimenti una volta giunto alla fine della funzione *main*() il microcontrollore rimarrebbe fermo per un tempo indeterminato, senza poter eseguire alcuna operazione.

All'interno del ciclo vengono generalmente inserite le funzioni di *polling*, cioè funzioni che analizzano incessantemente lo stato di un dato ingresso o di una data variabile, predisposte ad avviare automaticamente una determinata procedura una volta identificato l'evento atteso.

Ad esempio nel ciclo principale è stata posta la funzione *read_pin*(), con il compito di analizzare ad ogni ciclo lo stato dell'ingresso d'allarme, che provvede ad attivare automaticamente la procedura d'emergenza una volta rilevato un segnale di pericolo dal sensore. In modo simile sono state poste le funzioni *sms_check*(), *call_check*() e *read_batt*(), in modo da poter gestire automaticamente la ricezione di sms e chiamate, oltre che la lettura della batteria.

```
...
while( 1 )
{
    read_pin();
    read_batt();
    ...
    sms_check();
    call_check();
    ring_check();
    ...
}
...
```

6.2 Ciclo d'allarme

Il ciclo d'allarme è forse la parte più importante del programma, in quanto implementa la funzione principale del sistema.

Innanzitutto si determina lo stato d'allarme tramite la lettura del segnale amplificato e filtrato del sensore PIR, del quale si mantiene in memoria anche il valore precedente per poter eseguire un confronto.

```
// verifica un fronte di salita
if( pir.value && !pir.old ) alarm(); // avvia la segnalazione
```

In questo modo

- si può determinare il fronte di salita del segnale d'allarme
- si può avviare un singolo ciclo di segnalazione per ogni allarme rilevato
- si evitano inutili segnalazioni multiple
- si evitano loop di allarmi in grado di esaurire velocemente il credito della SIM

La procedura d'allarme provvede a segnalare l'emergenza all'utente tramite

- l'invio di un sms
- l'esecuzione di 3 chiamate senza risposta
- l'accensione del LED montato sulla scheda

in modo da assicurare entro i limiti del possibile che l'utente venga avvisato velocemente in ogni situazione. Infine, alla conclusione della segnalazione viene avviato un timer di durata di alcuni minuti atto ad inibire la lettura di ulteriori eventuali allarmi, in quanto è ritenuto privo di importanza la segnalazione multipla della stessa fonte d'allarme.

Capitolo 7

Inizializzazione

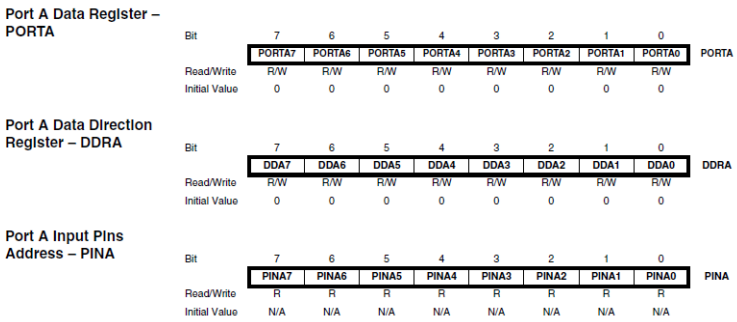
All'accensione è necessario prima di tutto configurare ed avviare le periferiche utilizzate, in modo che operino come voluto nel corso del programma. In questo caso bisogna:

- inizializzare le variabili del sistema
- selezionare ingressi e uscite
- configurare ed avviare il timer
- configurare l'UART
- inizializzare il watchdog
- abilitare gli interrupt
- avviare e configurare opportunamente il modem GSM

7.1 Configurazione

La configurazione di ogni parametro o periferica del microcontrollore avviene settando ad un valore opportuno dei registri dedicati, dove ogni bit ha un significato preciso.

Ad esempio la configurazione di PORT A, cioè dell'insieme dei primi 8 pin general purpouse, avviene tramite i registri a 8 bit DDRA e PORT A. In DDRA (Data Direction Register A) e PORT A ogni bit in base alla sua posizione corrisponde ad un determinato pin, quindi ad esempio il bit0 (LSB) di entrambi i registri va a modificare lo stato del pin PA0, mentre il bit7 (MSB) andrà a settare il pin PA7, comportamento facilmente intuibile anche osservando la descrizione visiva del registro.



Ora, tramite DDRA è possibile determinare se utilizzare un dato pin come ingresso o come uscita, mentre il comportamento di PORT A varia in base allo stato del pin. In particolare ponendo ad esempio il bit2 di DDRA a 1 scegliamo di utilizzare il pin PA2 come uscita e il suo valore verrà direttamente determinato dal valore del bit2 di PORT A. Viceversa, ponendo al contrario il bit2 di DDRA a 0 settiamo PA2 come ingresso, e il valore del bit2 di PORTA andrà solo ad influenzare la presenza o l'assenza di un pullup interno.

```
void port_init(void)
{
    // esempio di configurazione

    PORTA = 0b00001111;
    DDRA  = 0b00110011;

    // PA0 e PA1 diventano uscite a livello alto
    // PA2 e PA3 diventano ingressi con pullup
    // PA4 e PA5 diventano uscite a livello basso
    // PA6 e PA7 diventano ingressi senza pullup
}
```

L'intero funzionamento del microcontrollore viene determinato in modo simile, grazie alla giusta configurazione dei registri al suo interno secondo le informazioni facilmente ricavabili dal suo *datasheet*.

7.2 EEPROM

Normalmente le variabili utilizzate nel programma risiedono in RAM. In questo modo si ottiene una maggiore velocità d'utilizzo ma ogni valore viene perso in seguito ad un reset. Quando è necessario mantenere il valore di una certa variabile nel tempo, si può ricorrere al suo salvataggio nella memoria EEPROM interna. Questo microcontrollore contiene un banco di memoria EEPROM da 1024 bytes, e supporta fino a 100000 cicli di scrittura e cancellamento.

L'inizializzazione ed il caricamento dei dati salvati in EEPROM viene eseguito

automaticamente nella fase di inizializzazione del microcontrollore, nella funzione init_vars(): questa funzione inizialmente verifica la presenza di dati validi precedentemente salvati in EEPROM, ed in caso di insuccesso prima assegna a queste il loro valore di default, e poi ne salva il valore in memoria in modo che sia subito disponibile al successivo riavvio.

```
void init_vars(void)
{
    ...
    // lettura dati in eeprom
    if( !eeprom_read() )
    {
        // nel caso in cui la lettura fallisca
        // si inizializzano le variabili con i
        // loro valori di default e si salvano
        // in eeprom per il successivo riavvio

        id = DEFAULT_ID;

        eeprom_write();
    }
    ...
}
```

La verifica dell'attendibilità dei dati presenti in EEPROM viene determinata verificando il valore di un determinato byte di controllo. In questo sistema ad esempio è stato scelto di scrivere il valore esadecimale 0x77 nella posizione 1 dell'EEPROM a seguito di una scrittura dei dati in EEPROM. In questo modo al successivo riavvio è sufficiente verificare il valore letto in quella posizione per determinare se la memoria contiene dati validi da caricare.

Grazie a questo semplice controllo è possibile dotare ogni singola scheda con lo stesso firmware di una configurazione propria, semplicemente programmando tramite un computer il banco di memoria EEPROM (che altrimenti risulta interamente settato a 0xFF in seguito ad ogni comune programmazione).

7.3 Watchdog

Un watchdog è un timer autonomo interno, autodecrementante e predisposto per resettare il microcontrollore una volta giunto a zero. Questa periferica viene utilizzata per far fronte ad eventuali malfunzionamenti del microcontrollore, in modo da pervenire entro il limite del possibile eventuali blocchi a causa di forti disturbi esterni.

Durante il normale ciclo di funzionamento del microcontrollore il watchdog

viene regolarmente ricaricato in modo da non causare mai il reset forzato del sistema. In questo modo può giungere a zero solamente nel caso in cui il normale flusso del programma subisca un blocco inatteso. In questo caso un reset hardware è l'unica possibilità per poter riprendere il corretto funzionamento del sistema.

La configurazione del watchdog avviene tramite il registro WDTCR, tramite il quale è possibile anche determinare l'utilizzo di un prescaler, cioè un divisore di frequenza in grado di modulare il tempo di azzeramento del watchdog.

A questo punto, in assenza di prescaler è necessario che almeno una volta ogni 17ms il watchdog venga ricaricato tramite l'istruzione assembly *wdr* altrimenti si incorrerà in un reset del microcontrollore.

Capitolo 8

Lettura degli ingressi

L'ingresso dedicato alla lettura del segnale amplificato del sensore viene letto e filtrato dal microcontrollore, in modo che eventuali variazioni di durata inferiore a 20ms vengano completamente ignorate dal software, permettendo così una maggiore resistenza alle interferenze e riducendo le possibilità di un eventuale falso allarme.

8.1 Struttura del filtro

Il filtro utilizza un bitfield di un byte dove vengono salvati il dati relativi all'ingresso analizzato:

```
// Struttura dati dell'ingresso

struct pin_value {

    unsigned now   : 1; // valore attuale
    unsigned old   : 1; // valore precedente
    unsigned value : 1; // valore filtrato

    unsigned timer : 5; // timer filtro [ 0 - 31 ]

};
```

I primi 2 campi indicano lo stato dell'ingresso, mentre il terzo contiene il valore filtrato, che verrà utilizzato in tutto il resto del codice a riferimento di questo ingresso.

Per ottenere una normalizzazione a 8 bit della struttura rimangono 5 bits, sufficienti per essere utilizzati come buffer per il valore del timer (valore massimo 20).

8.2 Implemetazione software

Ad ogni ciclo si analizza lo stato del pin:

```
// ingresso PIR
volatile struct pin_value pir = { 0 , 0 , 0 , 0 };
...
// lettura stato corrente del pin
if( PINA & 0b00000010 )
      pir.now=1;
else  pir.now=0;
```

Quando viene rilevato un fronte positivo viene caricato il timer. Se nel tempo in cui il timer giunge a zero l'ingresso permane alto, il valore filtrato viene aggiornato ad 1, altrimenti rimane a 0.

Il contatore del timer viene decrementato dalla routine di interrupt del *Timer0* del sistema, chiamata una volta ogni millisecondo (per questo è stato preferibile definire *volatile* la struttura dati pir).

```
// avvio il timer in caso di fronte positivo
if( pir.now && !pir.old ) pir.timer=20;

// verifica stato del timer
if( pir.now && !pir.timer )
      pir.value=1;
else  pir.value=0;

pir.old=pir.now;
```

Grazie a questo filtro nel resto del programma è possibile utilizzare la variabile *pir.value* per poter analizzare il valore dell'ingresso d'allarme già filtrato.

Capitolo 9

Lettura della batteria

La natura stessa del sistema d'allarme impone la necessità di avere un funzionamento continuo a prescindere dalle condizioni della linea elettrica esterna, per far fronte ad eventuali blackout e tentativi di manomissione. Ovviamente però una batteria permette un'autonomia limitata, e qui nasce la necessità di essere costantemente a conoscenza del suo stato di carica, in modo da poter avvertire l'utente in caso di necessità.

9.1 ADC

Per poter leggere il valore analogico della batteria e poterlo correttamente interpretare è necessario l'utilizzo di un convertitore analogico digitale.

Internamente questo microcontrollore contiene un ADC a 10 bit ad approssimazioni successive completamente configurabile via software. Dotato di circuito *sample&hold*, provvede 8 canali utilizzabili, e diverse tensioni di riferimento selezionabili.

Proprio quest'ultima caratteristica risulta in questo caso indispensabile, in quanto pur essendo l'ADC stesso alimentato dalla batteria è necessario che fornisca un valore digitale univoco a prescindere dal valore corrente dell'alimentazione. Questo si ottiene semplicemente utilizzando il reference interno dedicato, posto al valore costante di 2.56V.

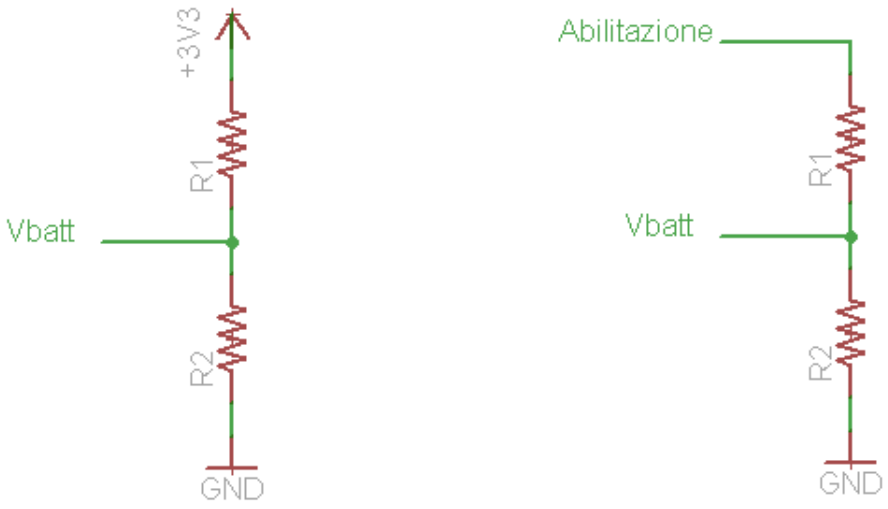
9.2 Soluzione circuitale

L'utilizzo nell'ADC della tensione di riferimento a 2.56V pone in campo un'ulteriore problema. Mentre la batteria utilizzata raggiunge un valore di

4.2V da carica il convertitore è in grado di leggere solamente valori minori o uguali alla sua tensione di riferimento, e quindi minori o uguali di 2.56V, impossibilitando così la corretta lettura del valore di batteria. Portando in questo modo il valore d'alimentazione all'ingresso dell'ADC otterremmo in uscita un valore costante pari a 0xFF per qualunque valore maggiore a 2.56V, rendendo la lettura totalmente inutile.

Per risolvere questo inconveniente è sufficiente utilizzare un partitore di tensione in grado di adattare il valore secondo le nostre esigenze. Il partitore qui utilizzato è semplicemente composto da 2 resistenze di valore uguale e molto elevato, collegate in serie tra alimentazione e massa. In questo modo è possibile leggere ai capi della seconda resistenza un valore pari alla metà di quello d'alimentazione con perdite relativamente contenute dovute alla corrente che percorre costantemente il partitore.

Queste perdite si possono ridurre enormemente ricorrendo ad un piccolo stratagemma. Invece di collegare un capo del partitore all'alimentazione questo è stato collegato ad un'altra porta del microcontrollore. In questo modo è possibile determinare via software quando e per quanto tempo porre quest'uscita a valore logico 1 oppure 0, è cioè se portare in uscita il valore di alimentazione o la massa.



Così è possibile mantenere a zero la tensione sul partitore per la maggior parte del tempo, annullando così i consumi, ed attivarlo solamente ad intervalli brevi e regolari, in concomitanza con le letture dell'ADC, a patto di assicurare il tempo necessario al partitore di arrivare a regime prima di eseguire la conversione.

9.3 Ciclo di lettura

La lettura del livello della batteria viene eseguita una volta ogni 20 secondi. Quest'operazione per essere eseguita correttamente deve seguire fasi precise:

Inizializzazione	prima di tutto è necessario abilitare la linea di alimentazione del partitore di tensione, controllata da PA2, ed abilitare l'ADC (senza avviare la conversione)
Attesa transitorio	prima di procedere alla vera lettura bisogna attendere almeno 3ms per permettere al partitore di giungere a regime, ed all'ADC di accendersi correttamente
Lettura iniziale	a questo punto è finalmente possibile avviare una prima conversione, il cui valore verrà però scartato in quanto essendo la prima ad avvenire dal momento dell'accensione dell'ADC non ne è assicurata nessuna attendibilità
Lettura effettiva	la successiva lettura invece viene considerata valida ed attendibile. In questo momento però entra in campo un ulteriore livello di filtraggio, in quanto nel programma non si utilizza mai il singolo valore derivato da una conversione, bensì la media dei risultati delle ultime 20 letture, ottenendo così l'immunità al rumore richiesta.
Conclusione	a questo punto vengono disabilitati sia il potenziometro che l'ADC, in modo da limitare il più possibile lo spreco di energia, e viene ricaricato il timer che avvierà il successivo ciclo di lettura.

ognuna di queste operazioni va a verificare e modificare il valore di alcune flag di controllo senza mai porre il programma in attesa passiva di un evento (quale la conversione o i timeout). In questo modo durante l'esecuzione di questa procedura il microcontrollore è in grado di eseguire anche gli altri compiti, di verificare gli allarmi e rispondere alle richieste dell'utente in tempo reale.

Di seguito riportiamo il listato completo riguardante la routine di lettura della batteria, per permetterne una maggiore comprensione e per mostrare lo stile di base con cui è stato scritto il codice.

```
// flag di accensione dell'adc
unsigned char adc_on=0;
// numero di letture preventive
unsigned char adc_del=1;
// contatore decrementato ogni ms dal Timer0
volatile unsigned int batt_timer=20000;
...
if( !batt_timer ) // timeout
{
    // fase 1
    if( !adc_timer && !adc_on )
    {
        ADCSRA |= 0b10000000; // abilitazione ADC
        PORTA  |= 0b00000100; // abilitazione potenziometro

        adc_on = 1;
        adc_timer = 3; // timeout transitorio
    }

    if( adc_on && !adc_timer )
    {
        // fase 2
        // conversione NON in corso
        if( !(ADCSRA & 0b01000000) )
        {
            // start conversion
            ADCSRA |= 0b01000000;
        }

        // conversione completata
        if( ADCSRA & 0b00010000 )
        {
            // lettura valore
            value = ADCH;
            // clear flag
            ADCSRA |= 0b00010000;

            // fase 3
            if( adc_del ) adc_del --;
            else{

                // fase 4
                // aggiunta al filtro
                filter( value );
```

```
                        // fase 5
                        // turn off ADC
                        ADCSRA &= 0b01111111;
                        // turn off potentiometer
                        PORT1  &= 0b11111011;

                        adc_on=0;
                        adc_del=1;

                        batt_timer = 20000;
                    }
                }
            }

        }
```

Capitolo 10

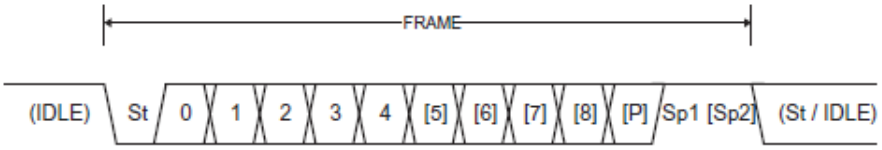
Comunicazione con il modem

Il microcontrollore conversa con il modem GSM tramite la linea UART dedicata, che permette una comunicazione seriale a media velocità con l'utilizzo di solamente 2 pin.

10.1 UART

L'Universal Asynchronous serial Receiver and Transmitter (UART) è un'interfaccia di trasmissione seriale di dati ampiamente utilizzata, soprattutto nella comunicazione tra computer e dispositivi elettronici integrati (a seguito di opportuni sistemi di conversione elettrica). Quest'interfaccia necessita di un solo filo per la trasmissione e uno per la ricezione, permettendo così la gestione separata e contemporanea delle 2 operazioni. L'UART supporta la selezione di un ampio range di baudrate e sono disponibili diversi formati di pacchetti (o frame).

Nel nostro caso la scelta ricade nella struttura più classica, composta da 1 bit di start, 8 bit di dati e 1 bit di stop, per un frame totale di 10bit.



Per poter utilizzare l'UART presente nel microcontrollore è necessario attivarla, settando correttamente i suoi registri di configurazione

```
// inizializzazione UART
void uart0_init(void)
{
```

```
        UCSRB = 0b00000000;
        UCSRA = 0b00000000;

        UBRR = 0x0019; // 19200 baud
        UCSRC = 0b10000110; // 8 bit transfer size

        UCSRB = 0b10011000; // RXCIE | RXEN | TXEN
    }
```

Questo codice imposta la configurazione di base, abilita sia la ricezione che la trasmissione, e attiva l'interrupt rx complete. Quest'interrupt è particolarmente utile perchè permette di attivare automaticamente la routine di ricezione ogniqualvolta si riceve un byte da parte del modem.

10.2 Configurazione

Per poter essere correttamente utilizzabile il modem deve prima essere correttamente configurato, tramite opportuni comandi inviati tramite la linea seriale. A questo fine in seguito all'accensione del modem vengono inviati dei comandi in serie, ad intervalli di 110ms l'uno dall'altro, secondo un determinato ordine:

AT per verificare che la comunicazione sia in piedi

AT&F1 per resettare ogni registro di configurazione

AT&K0 per disabilitare il controllo hardware delle trasmissioni

AT+IPR=19200 per impostare il baud rate utilizzato

ATE1 per disabilitare l'echo dei comandi

ATQ0 per abilitare le risposte del modem

ATV1 per abilitare le risposte estese

ATX1 per determinare il formato delle risposte

AT#DIALMODE=0 per abilitare i messaggi di stato delle chiamate

AT+CMEE=1 per abilitare i messaggi di errore

Queste operazioni risultano fondamentali per la corretta comunicazione con il modem in quanto determinano il formato delle stringhe che invierà in risposta ai nostri comandi, permettendo così una intercomunicazione veloce precisa e sicura. Senza queste operazioni preliminari sarebbe quasi impossibile prevedere una risposta software automatica ai svariati formati di risposta che

il modem è in grado di inviare per ogni singolo messaggio.

Tutti questi comandi vengono eseguiti senza la verifica della risposta inviata dal modem, in quanto:

- all'accensione non conosciamo il formato di risposta del modem
- in assenza di problemi hardware il modem dovrebbe rispondere sempre in modo affermativo a questi comandi
- in caso invece di problemi hardware non c'è possibilità di soluzione da parte del software, quindi risulta irrilevante analizzare queste risposte

Solo successivamente alla corretta configurazione del modem viene abilitato il processo di analisi automatica dei dati ricevuti dal modem, che d'ora in poi verificherà la corretta esecuzione di ogni comando.

10.3 Sintassi comandi

Come già detto il modem va comandato tramite stringhe di testo trasmesse tramite il collegamento seriale, ed è strettamente necessario che queste direttive seguano una sintassi e una tempistica ben determinata.

I comandi utilizzati seguono lo standard AT, e sono composti dall'identificativo standard AT+ seguito dal nome del comando, e da eventuali altri caratteri indicanti la modalità d'esecuzione del comando. Poniamo ad esempio di voler conoscere lo stato della connessione alla rete del modulo. Il comando predisposto a questa operazione è AT+CREG? dove

AT+ è l'identificativo standard

CREG è il comando vero e proprio

? indica l'azione da eseguire, in questo caso una lettura

Questa stringa va inviata al modem seguita dal valore esadecimale 0x0D, indicante la fine del comando.

10.4 Ciclo di comando

10.4.1 Invio comando

Ogni comando per poter essere correttamente processato deve seguire alcune regole principali:

- la stringa di comando deve terminare con il valore esadecimale 0x0D
- devono passare almeno 20ms l'ultima ricezione ed una nuova trasmissione
- è necessario determinare il parser da utilizzare

La determinazione del parser è forse l'operazione più importante, in quanto va a determinare la porzione di codice che verrà utilizzata per analizzare la risposta del modem.

La corretta esecuzione di ogni comando secondo i canoni previsti viene affidata alla funzione send_cmd.

```
unsigned char send_cmd( char *cmd ,
                        unsigned int timeout ,
                        unsigned char parser )
{
    unsigned char ret=0;
    unsigned char retry=1; // massimo numero di errori possibili

    do{
        tx_sync( parser ); // init tx
        tx_cmd( cmd ); // send command

        ret=wait_rx( timeout );
        if( ret != tx_timeout ) return ret;

    }while( retry-- );

    fatalerr();

    return tx_err; //tx_timeout;
}
```

Per la sua chiamata necessita di 3 argomenti:

cmd puntatore alla stringa di comando

timeout attesa massima per la ricezione (in ms)

parser routine di analisi della risposta

mentre osservando il codice si nota di come questa funzione sia composta da ulteriori 3 sottofunzioni principali:

tx_sync() assicura il giusto ritardo tra comandi successivi

tx_cmd() invia fisicamente il comando al modem

wait_rx() attende che sia completata l'analisi della risposta

10.4.2 Ricezione ed analisi comando

Nella funzione precedente quello che non si nota è l'intero processo di ricezione ed analisi della rispostadel modem, che avviene interamente sotto interrupt.

La ricezione di un intero byte causa l'interrupt dedicato (RX Complete Interrupt, RXC) che avvia automaticamente la funzione uart0_isr()

```
#pragma interrupt_handler uart0_isr: 14
void uart0_isr(void)
{
    unsigned char rxbyte;

    // lettura byte ricevuto
    rxbyte=UDR;

    // verifica errori
    if( UCSRA & 0b00011100 ) return;

    // analisi byte ricevuto
    rx( rxbyte );
}
```

questa funzione esegue solamente le operazioni più basilari, quali la lettura del dato ricevuto e la verifica di eventuali errori in ricezione. La vera analisi dei dati avviene nelle fasi successive

Fase 1: uart0_isr() lettura del byte ricevuto e verifica degli errori

Fase 2: rx() ricostruzione del messaggio, composto da stringhe ASCII terminate dai valori esadecimali 0x0D 0x0A (CR LF)

Fase 3: parser() analisi della risposta, verifica del significato della stringa ricevuta nel contesto attuale e pianificazione delle operazioni conseguenti

in particolare la funzione parser() provvede ad identificare la validità della risposta ed il suo significato, confrontando i dati ricevuti con delle stringhe che rappresentano la risposta standard che ci si aspetterebbe dall'esecuzione del comando corrente.

Analizziamo ad esempio il comando AT+CPIN, utilizzato per verificare lo stato della SIM. Esso prevede 7 possibili risposte diverse:

+CPIN: SIM PIN	sim presente in attesa di codice pin
+CPIN: SIM PUK	sim presente in attesa di codice puk
+CPIN: READY	sim pronta, non necessita nessun codice pin
+CME ERROR: 10	sim assente
+CME ERROR: 13	sim inserita male
+CME ERROR: 14	sim occupata, ritentare
+CME ERROR: 15	sim non valida

Quindi notiamo subito che la presenza della parola *READY* nella risposta indica che la sim è pronta all'utilizzo, mentre in tutti gli altri casi ci troviamo di fronte ad una situazione d'errore (in quanto è stato scelto di non gestire ogni eventuale codice di identificazione). In questo modo la routine di parsing risulta molto semplice ma allo stesso tempo molto precisa nell'indicare l'informazione voluta

```
void parser(void)
{
    // indicizzazione stringa
    string_analyzer( rxbuf );
    ...
    switch( current_parser )
    {
        ...
        case cpin_parser :

                // verifica del formato della risposta
                if( !strcmp( field[0] , "CPIN" ) ) break;

                if( strcmp( field[1] , "READY" ) )
                     txresult = tx_ok;
                else txresult = tx_err;

                break;
        ...
    }

    // Verifica eventuali errori

    if( !txresult )
    if( current_parser )
    {
        if( strcmp( field[0] , "OK" ) )
        {
            txresult = tx_ok;
        }

        if( strcmp( field[0] , "ERROR" ) ||
            strcmp( field[0] , "CME" )   ||
```

```
                strcmp( field[0] , "CMS" ) )
            {
                txresult = tx_err;
            }
        }
    }
```

Tramite queste poche linee di codice otteniamo un'analisi esauriente della risposta, andando a settare opportunamente la variabile *tx_result* come atteso nella funzione send_cmd() analizzata precedentemente.

Ricordiamo infine che questa funzione viene chiamata a seguito di un interrupt causato dalla ricezione di un byte, ed avvenendo nel bel mezzo della trasmissione è necessaria la massima velocità per non rischiare di perdere la ricezione del byte successivo. Qui l'operazione più onerosa in termini di tempo è sicuramente il confronto tra la stringa ottenuta in ricezione e i valori costanti che ci aspettiamo di trovare. Per questo motivo prima dell'analisi vera e propria la risposta viene analizzata dalla funzione *string_analyzer()*, che suddivide la stringa in un insieme di parole terminate da 0x00 indicizzate dall'array ordinato di puntatori *field[]*. Grazie a questa operazione preventiva il lavoro del parser viene enormemente semplificato e velocizzato, in quanto non è più necessario ricercare un pattern lungo l'intera lunghezza della risposta ma è sufficiente confrontarlo con un solo campo, relativo alla posizione dove dovrebbe trovarsi.

Capitolo 11

Gestione SMS

Matrix GSM Alarm è in grado di dialogare con l'utente attraverso l'invio e la ricezione di semplici messaggi testuali. Grazie a questa funzionalità è possibile sia venire avvisati dal sistema d'allarme quando avviene un particolare evento, sia tenerlo sotto controllo da remoto, diventando così uno strumento importantissimo per gestire al meglio le potenzialità del sistema.

11.1 Invio SMS

L'invio di un sms viene eseguito tramite il comando *AT+CMGS*, seguito da una stringa indicante il numero telefonico del destinatario. Successivamente, si inserisce il testo voluto, si lascia una riga vuota e si conclude il tutto tramite l'invio di un byte di valore 0x1A.

```
// Esempio di comunicazione tra microcontrollore (U)
// e modem (M) per l'invio di un sms

U: AT+CMGS=3401122334
M: >
U: Messaggio di prova
U:
U: 0x1A
M: OK (213)
```

Quest'operazione viene eseguita automaticamente tramite la comoda funzione *send_sms*(), che necessita come argomenti i puntatori alle stringhe contenenti il testo del messaggio e il numero del destinatario.

```
unsigned char send_sms(char *dest,unsigned char *mess)
{
    // inizializzazione trasmissioni
    if( !start_sms(dest) ) return 0;
```

```
        // invio testo
        tx_string( mess , 0 );

        // verifica risposta
        return stop_sms();
    }
```

Genericamente l'invio di un sms può avvenire secondo 2 differenti modalità:

- modalità testuale
- modalità PDU

Al giorno d'oggi la seconda risulta essere più diffusa, in quanto è l'unica che necessariamente deve essere implementata in ogni modem GSM. Pur essendo lo standard de facto risulta però essere enormemente più complessa da implementare pur avendo alla fine dei conti le stesse identiche potenzialità della modalità testuale, e proprio per questo è stato scelto di utilizzare la modalità testuale all'interno di questo progetto.

La modalità testuale semplifica sia il processo di invio che di ricezione dei messaggi, e risulta molto utile in fase di debug in quanto i dati comunicati tra modem e microcontrollore risultano in chiaro, senza aver subito alcun processo crittografico o di compressione.

11.2 Ricezione SMS

In seguito alla ricezione di un sms, il modem invia automaticamente al microcontrollore la stringa *+CMTI: SM,1*, dove

+CMTI indica la ricezione di un messaggio

SM indica in quale memoria è stato salvato il messaggio

1 indica la posizione che ha assunto il messaggio

In questo caso quindi il messaggio ricevuto risulta essere stato salvato nella prima posizione della memoria della SIM.

A questo punto il microcontrollore può recuperare questo messaggio richiedendone la lettura tramite il comando *AT+CMGR*, come da esempio:

```
// Esempio di comunicazione tra microcontrollore (U)
// e modem (M) per la ricezione di un sms

// ricezione del messaggio
```

```
M: +CMTI: "SM",1

// richiesta di lettura
U: AT+CMGR=1
M: +CMGR: "REC UNREAD","+393401234567",,"09/11/23,10:31:20+04"
M: Testo di prova
M: OK

// cancellazione del messaggio
U: AT+CMGD=1
M: OK
```

Come si vede nell'esempio, prima di inviare il testo del messaggio il modem invia anche una stringa contenente altre informazioni utili, quali il numero del mittente, lo stato del messaggio (REC UNREAD, cioè messaggio non letto) ed il relativo timestamp. Grazie a queste informazioni è possibile verificare l'utilità del messaggio.

Matrix GSM Alarm è stato configurato per accettare solamente i messaggi provenienti da mittenti il cui numero è contenuto nelle prime posizioni della rubrica della SIM, quindi se il numero letto in questo momento non combacia con nessuno di quelli validi il messaggio viene direttamente scartato.

Quando invece il numero di telefono del mittente risulta presente nella rubrica, il corpo del del messaggio segue un secondo processo di analisi, dove si procede al suo confronto con i comandi accettati predefiniti, e nel caso ci sia una corrispondenza viene eseguito il codice relativo al comando. Tutto questo avviene nella funzione *text_parser*():

```
void text_parser(char *str)
{
    // Commands are case insensitive

    uppercase( str );

    ...
    if( strcmp( str , "VER" ) )
    {
        send_sms( mitt , "Version 1.0" );
        return;
    }
    ...
}
```

Nell'esempio si nota subito che se il messaggio è composto dalla stringa *VER* il mittente riceverà automaticamente un sms indicante la versione software corrente. Anche gli altri comandi vengono verificati in questo modo,

aggiungendo semplicemente ulteriori blocchi *IF* in successione.

Prima dell'analisi del comando però avviene un'altra importante operazione: grazie alla funzione *uppercase*() ogni lettera del messaggio diventa maiuscola. In questo modo è possibile processare correttamente il comando a prescindere dal modo in cui è stato scritto. Ad esempio, i comandi VER, Ver, ver e vEr vengono processati tutti allo stesso modo.

Capitolo 12

Gestione chiamate

Per quanto la gestione del sistema tramite sms risulta comoda, pratica e veloce, causa necessariamente dei piccoli costi aggiuntivi, in quanto il servizio di invio di sms viene pagato al gestore della rete mobile esattamente come accade con i comuni telefoni cellulari.

Un'alternativa interessante e totalmente priva di costi è la gestione del sistema tramite comuni chiamate senza risposta (*squilli*). Ovviamente la varietà di operazioni eseguibili è molto più limitata, in parte per la povertà di questo metodo di comunicazione, in parte per mantenere alta la semplicità d'utilizzo.

In particolare si è scelto di implementare 2 funzionalità principali:

- l'utente può abilitare e disabilitare il sistema d'allarme tramite un numero predefinito di squilli
- in caso d'allarme il sistema può essere configurato per eseguire 3 squilli di durata di 1 minuto distanziati da un ulteriore minuto per avvertire l'utente dell'emergenza

In questo modo sono assicurate le funzionalità basilari a prescindere dalla quantità di credito residua nella scheda SIM.

12.1 Esecuzione chiamate

Per avviare una chiamata si utilizza il comando *ATD* seguito direttamente dal numero del destinatario e da un punto e virgola finale, che assicura l'esecuzione di una chiamata in modalità vocale.

```
// Esempio di comunicazione tra microcontrollore (U)
// e modem (M) per l'esecuzione di una chiamata vocale
```

```
// esecuzione chiamata
U: ATD3401122334;
M: DIALING
M: RINGING

// conclusione chiamata
U: ATH
M: OK
```

Durante la telefonata il modem GSM provvede ad informarci del suo stato corrente secondo stringhe di testo predefinite:

DIALING inizializzazione chiamata in corso

RINGING il telefono del destinatario sta squillando

CONNECTED il destinatario ha risposto alla telefonata

RELEASED telefonata terminata da noi

DISCONNECTED telefonata terminata dal destinatario

in modo da poter gestire correttamente ogni situazione.

Il processo di chiamata deve avvenire in realtime senza bloccare però l'esecuzione del resto del ciclo del programma, che deve continuare ad analizzare l'ingresso di allarme e gli eventuali comandi inviati dall'utente. Per avviare la telefonata si utilizza la funzione *auto_call*(), che provvede semplicemente a settare determinate *flags* che determinano il numero di squilli da eseguire e la loro durata.

```
void auto_call(unsigned char nr,unsigned char ns)
{
    if( call_counter )
    {
        // chiamata in corso?
        if( calling )
        {
            // hung-up
            send_cmd( "ATH" , 30000 , default_parser );
            call_timer=5000;

        }else call_timer=0;
    }

    calling=0;
    num_ptr=0;
```

```
    call_counter=nr; // numero chiamate
    call_timeout=ns; // durata chiamate
}
```

La vera gestione delle telefonate avviene nella funzione *call_check*(), eseguita automaticamente ad ogni ciclo del programma. Per la comprensione di questa parte proponiamo di seguito il codice della funzione, semplificato ed ampiamente commentato, in modo da risultare di facile comprensione.

```
void call_check(void)
{
    // ogni operazione qui svolta richiede un determinato
    // tempo prima di poter risultare completata, ad esempio
    // una volta avviata una chiamata è necessario attendere
    // almeno 60 secondi prima di concluderla. Per far fronte
    // a questi "tempi morti" una volta eseguita l'operazione
    // voluta viene caricata la variabile call_timer con il
    // valore voluto, in modo da inibire la chiamata a questa
    // funzione per il tempo di durata dell'operazione corrente

    if( call_timer ) return;

    // se il programma raggiunge questo punto significa che
    // è passato il tempo voluto ed è arrivato il momento di
    // eseguire l'operazione successiva

    // ad esempio, se è attiva una chiamata (calling=1), ora
    // è giunto il momento di terminarla

    if( calling )
    {
        // hungup
        send_cmd( "ATH" , 30000 , default_parser );

        ...

        if( call_counter )
            call_counter--;

        ...

        // chiamata conclusa
        calling=0;
        // attendiamo un minuto prima della
        // prossima chiamata
        call_timer=60000;

    }else{
```

```
            // procedura avvio automatico
            // di una telefonata

            // verifica il numero di squilli già eseguiti
            if( !call_counter ) return;
            // verifica se è impostato un tempo minimo di chiamata
            if( !call_timeout ) return;

            // esecuzione chiamata
            start_call( &rubrica[num_ptr][0] );
            calling=1;

            // settiamo la durata della chiamata
            if( call_timeout < 60)  call_timer=call_timeout*1000;
            else                    call_timer=60000;
        }

    }
```

Tutto questo dà all'utente l'illusione di un ambiente multitasking. Tuttavia il micocontrollore è in grado di eseguire una sola operazione per volta, l'abilità del programmatore sta solamente nella capacità di far convivere più funzionalità in successione senza creare blocchi o conflitti.

12.2 Ricezione chiamate

L'operazione di ricezione e conteggio delle chiamate risulta molto più semplice della precedente. Durante la ricezione della chiamata, il modem invia ad intervalli regolari di 2 secondi una stringa identificativa al microcontrollore:

```
// Esempio di comunicazione tra microcontrollore (U)
// e modem (M) durante la ricezione di una chiamata vocale

// ogni riga corrisponde ad un tono di chiamata
// come sentito da parte del mittente
M: +CLIP: "+393401234567",145,"",128,"Nome",0
M: +CLIP: "+393401234567",145,"",128,"Nome",0
M: +CLIP: "+393401234567",145,"",128,"Nome",0

// conclusione automatica della chiamata
// dopo 3 squilli consecutivi
U: ATH
M: OK
```

Come facilmente si nota nell'esempio, il modem fornisce una stringa che non solo indica che è in corso una chiamata (+CLIP) ma anche il mittente di tale telefonata. In questo modo è possibile determinare se il chiamante è presente

nella rubrica telefonica e determinare l'operazione da eseguire.

Nel caso il numero del mittente non sia presente in rubrica la chiamata viene immediatamente terminata.

Se invece il numero viene riconosciuto tra quelli autorizzati, il microcontrollore provvede al conteggio delle stringhe ricevute, cioè del numero di squilli ricevuti. Una volta rilevato il giusto numero di squilli (da 1 a 3) la chiamata viene terminata automaticamente e si esegue l'operazione associata.

La funzione dedicata all'analisi delle chiamate ricevute è *ring_check*(), chiamata anch'essa automaticamente all'interno del ciclo principale nella funzione *main*().

```
void ring_check(void)
{
    if( new_ring ) // nuovo squillo da analizzare
    {
        new_ring=0;
        ring_counter ++;

        switch( ring_counter )
        {
            case 1:
                    // abilitazione allarme
                    if( !alarm_enable )
                    {
                        enable_alarm();

                        ring_counter=0;
                        send_cmd( "ATH" , 30000 , default_parser );
                    }

                    break;
            case 3:
                    // disabilitazione allarme
                    if( alarm_enable )
                    {
                        disable_alarm();

                        ring_counter=0;
                        send_cmd( "ATH" , 30000 , default_parser );
                    }

                    break;

            default: break;
        }
```

```
            }
        }
```

In questo modo questa piccola porzione permette all'utente di eseguire le operazioni principali quali la gestione dello stato di abilitazione dell'allarme grazie all'esecuzione di semplici squilli.

Parte III

Prodotto finale

Capitolo 13

Caratteristiche del prodotto

In conclusione abbiamo ottenuto un prodotto finale in grado di portare in ogni casa una valida protezione antifurto ad un prezzo contenuto grazie alla semplicità del sistema che però si rivela efficace, flessibile ed affidabile.

I punti di forza di *Matrix GSM Alarm* sono:

semplicità d'installazione: si può posizionare l'allarme in qualunque luogo senza la necessità di particolari accortezze

flessibilità: grazie all'utilizzo del sensore PIR è possibile monitorare una zona di medie dimensioni quali una finestra, un ingresso, o un'intera stanza semplicemente posizionando correttamente l'apparecchio

efficacia: il sensore PIR è in grado di rilevare una presenza umana con la massima precisione e rapidità, rendendolo il sistema più efficace per proteggersi da furti o effrazioni

comodità: l'allarme può essere controllato completamente tramite un comunissimo cellulare in grado di inviare e ricevere SMS oppure comuni chiamate vocali, in modo da minimizzare così i costi dovuti all'utilizzo della linea GSM

autonomia: grazie alla batteria di serie il sistema rimane in funzione per parecchie ore anche in seguito ad eventuali problemi alla linea elettrica, assicurando così la massima copertura

Otteniamo così un prodotto semplice ma dalle enormi potenzialità, che non necessita di particolari conoscenze ma al contrario è utilizzabile facilmente da chiunque.

13.1 Funzionalità

Matrix GSM Alarm è in grado di monitorare e rilevare la presenza umana nella zona prescritta fino ad una distanza di *10m* grazie al sensore PIR ad alta precisione.

Dotato di batteria, è in grado di resistere per svariate ore anche in caso di malfunzionamenti della linea elettrica pubblica.

Grazie all'utilizzo di un modem GSM è possibile tenere completamente sotto controllo l'apparecchio grazie ad un comunissimo telefono cellulare, ovunque ci si trovi. In particolare in caso d'allarme l'utente verrà avvisato in modo tempestivo nel giro di *1 o 2 minuti* (i tempi variano in base allo stato della rete GSM), ovunque si trovi.

La modalità di segnalazione d'allarme è configurabile tramite un semplice sms, ed è possibile abilitare o disabilitare indipendentemente gli avvisi tramite sms o squillo. In questo modo è possibile eliminare del tutto le segnalazioni tramite sms, in modo da cancellare i costi dovuti all'utilizzo della rete GSM.

Una caratteristica interessante è la capacità di inoltrare all'utente ogni messaggio estraneo ricevuto, come ad esempio i classici messaggi indicanti il credito residuo, in modo da permettere all'utente di tenere sotto controllo le comunicazioni verso la centralina e rilevare eventuali tentativi d'attacco.

Infine nel caso in cui il modulo non riesca ad avvisare l'utente nè attraverso sms nè tramite chiamate telefoniche a causa del credito residuo insufficiente, *Matrix GSM Alarm* provvede automaticamente ad eseguire delle chiamate con addebito. In questo modo si assicura di riuscire in ogni caso a trasmettere l'informazione voluta all'utente.

Tutto questo fa di *Matrix GSM Alarm* un prodotto altamente affidabile e tuttavia flessibile, adatto a svariate circostanze e dotato di un'interfaccia d'utilizzo semplice ed intuitiva, che permette all'utente di sfruttare appieno le sue potenzialità senza eccessive difficoltà.

13.2 Guida per l'utente

Il dispositivo si accende appena viene alimentato, come si può notare dal LED posto frontalmente, per poi rimanere per un minuto in attesa di eventuali comandi. Nel caso non arrivi alcun comando tramite sms o chiamata telefonica il sistema di allarme si attiva automaticamente, e da questo momento in poi

provvederà a monitorare lo stato del segnale del sensore.

Come già detto precedentemente questo sistema si può controllare tramite l'invio di comuni sms testuali. In particolare:

START mette in funzione il dispositivo, che da questo momento in poi segnalerà ogni eventuale segnale d'allarme
STOP disattiva l'allarme, inibisce ogni segnalazione
SMS segnalazioni d'allarme solo tramite sms
CALL segnalazioni d'allarme solo tramite chiamate
ALL segnalazioni d'allarme tramite sms e chiamate
HELP mostra una sintesi dei comandi disponibili
VER riferisce la versione software in funzione

Alternativamente è possibile attivare e disattivare il sistema d'allarme tramite delle chiamate. Se il sistema è disattivo, in seguito al primo squillo viene automaticamente attivato. Viceversa, se è gia attivo verrà a disattivarsi a seguito del terzo squillo. Non è necessario contare il numero di squilli della chiamata in quanto il dispositivo provvede autonomamente al conteggio e alla conclusione della chiamata, senza mai rispondere e quindi senza mai causare costi dovuti al traffico telefonico.

D'altra parte, *Matrix GSM Alarm* provvede a segnalare ogni informazione importante all'utente. L'utente così potrà ricevere diversi sms da parte del dispositivo, ognuno con un preciso significato

MATRIX ON: sistema attivo

MATRIX OFF: sistema disattivo, in attesa di ulteriori comandi

ALLARME: segnalazione di uno stato di allarme

RING ON: chiamate di segnalazione abilitate

CALL ON: sms di segnalazione abilitati

RING&CALL ON: chiamate ed sms di segnalazione abilitati

Verifica credito: rilevato un livello di credito residuo basso

Batteria in esaurimento: rilevato un livello basso di batteria

Capitolo 14

Considerazioni finali

14.1 Problemi riscontrati

Durante i primi test hardware sono stati notati dei problemi non previsti durante la fase di progettazione, dovuti alla convivenza del sensore PIR e del modulo GSM.

La presenza del modulo GSM va ad influenzare il segnale in uscita dal sensore in 2 modi:

1. disturbi ad alta frequenza
2. oscillazioni dell'alimentazione

La ricetrasmissione su rete GSM comporta la creazione di campi elettromagnetici molto forti, in grado di accoppiarsi con il package metallico del sensore PIR che va a fungere da antenna provocando impulsi di ampiezza nell'ordine dei volt in grado di durare per interi milisecondi. In questo scenario il filtro attivo provvede ad un'ulteriore amplificazione del disturbo e la lettura del segnale d'allarme risulta falsata.

D'altro canto durante le operazioni di trasmissione il modulo GSM comporta consumi elettrici notevoli, assorbendo dall'alimentazione picchi di corrente dell'ordine dei 2A in grado di generare oscillazioni dell'alimentazione che per quanto ridotte vanno ad alterare quanto basta il funzionamento del sensore PIR, generando anche in questo caso dei falsi allarmi.

Entrambi questi problemi hanno però già trovato soluzione in ambito sperimentale, e la versione finale del progetto sarà immune da questi disturbi. Innanzitutto è stato necessario rivedere la struttura del circuito di alimentazione in modo da renderlo più stabile ed isolare e rafforzare per quanto

possibile il circuito di alimentazione del sensore da quello del modem. Infine è stato rivisto e riprogettato il circuito di filtro ed amplificazione del segnale del sensore, con l'accortezza di porlo il più vicino possibile al sensore stesso in modo da diminuire l'area in grado di accoppiarsi con eventuali segnali ad alta frequenza.

Nel frattempo è stato implementata anche una soluzione software temporanea per permettere il funzionamento anche ale prime schede provvisorie prive di queste correzioni. I problemi di incompatibilità tra sensore e modulo GSM sono stati ovviati suddividendo il ciclo di lavoro del modulo in 2 fasi principali:

Fase di attesa: sensore abilitato e modulo GSM spento

Fase di allarme: sensore disabilitato e modulo GSM acceso

Durante la prima fase il sistema analizza il segnale del sensore pir, in attesa. Una volta rilevato un allarme il pir viene disabilitato, si accende il modulo GSM e si procede con la segnalazione d'emergenza. In seguito il modem viene nuovamente spento e dopo un minuto viene riattivato il sensore, tornando così allo stato iniziale.

Grazie a questo espediente provvisorio il sistema funziona senza problemi, risolvendo momentaneamente il problema di incompatibilità hardware.

14.2 Evoluzioni future

Per questo prodotto è già stato previsto lo sviluppo di ulteriori versioni, analoghe dal punto di vista della comunicazione via GSM ma differenti nel tipo di ingresso d'allarme utilizzato.

14.2.1 Interfaccia generica

In questa versione al posto del sensore PIR sono presenti 2 fili d'allarme

- il primo con pullup, che se rilasciato si mantiene a livello alto
- il secondo con pulldown, che se rilasciato si mantiene a livello basso

il microcontrollore verifica ciclicamente lo stato di questi 2 ingressi e nel caso in cui uno dei 2 si ritrovi invertito viene avviata la procedura di allarme.

In questo modo otteniamo un sistema d'allarme generico che può reagire a segnali diversi, in base a quale filo viene utilizzato come sorgente d'allarme ed in base al tipo di segnale utilizzato. Generalmente è possibile

- collegare un sensore apposito ad uno dei fili d'allarme
- collegare il sistema in parallelo ad un'altro allarme, in modo da estendere il controllo di quest'ultimo tramite le funzionalità da noi implementate

Ad esempio è possibile collegarlo parallelamente alla linea d'allarme preesistente di un comune antifurto per auto per dotare quest'ultimo delle caratteristiche di controllo remoto da noi implementate, oppure collegare un ingresso d'allarme ad un sensore magnetico in modo da poter tener sotto controllo l'ingresso di un'abitazione.

Le molteplici possibilità di questo prodotto lo rendono flessibile e facilmente personalizzabile in base alle necessità del cliente senza alcuno sforzo progettuale aggiuntivo.

14.2.2 Interfaccia seriale

Questa versione viene dotata di una linea seriale aggiuntiva a bassa velocità, che permette di interfacciare il nostro sistema dotato di modulo GSM ad altri sistemi dotati di microcontrollore o diversa logica di controllo, purchè abbiano a bordo almeno una linea di comunicazione UART.

In quest'ottica è possibile sia espandere il nostro sistema con periferiche aggiuntive quali ad esempio

- moduli di comunicazione IR
- moduli di comunicazione wireless
- moduli interfacciamento a reti CAN

sia permettere a sistemi preesistenti di utilizzare le nostre funzionalità di comunicazione tramite rete GSM.

Con questo prodotto è possibile realizzare un'infinità di combinazioni e sistemi diversi, dai più semplici ai più complessi, mantenendo allo stesso tempo la semplicità d'utilizzo insita nell'interfaccia di controllo da noi progettata.

Bibliografia

[1] Atmel Corporation, *ATMega32 Datasheet*, `http://www.atmel.com/`

[2] Telit Wireless Solutions, *Telit GM862 Hardware User Guide*, `http://www.telit.co.it/`

[3] Telit Wireless Solutions, *Telit GM862 Software User Guide*, `http://www.telit.co.it/`

[4] Telit Wireless Solutions, *Telit AT Commands Reference Guide*, `http://www.telit.co.it/`

[5] Glolab Corporation, *Infrared Parts Manual*, `http://www.glolab.com/`