I.T.I.S. BENEDETTO CASTELLI --- BRESCIA
ANNO SCOLASTICO 2005/2006 5^E

PILOTAGGIO DI UN CANCELLO ELETTRICO

DI

Renato Priuli & Gabriele Sammartino

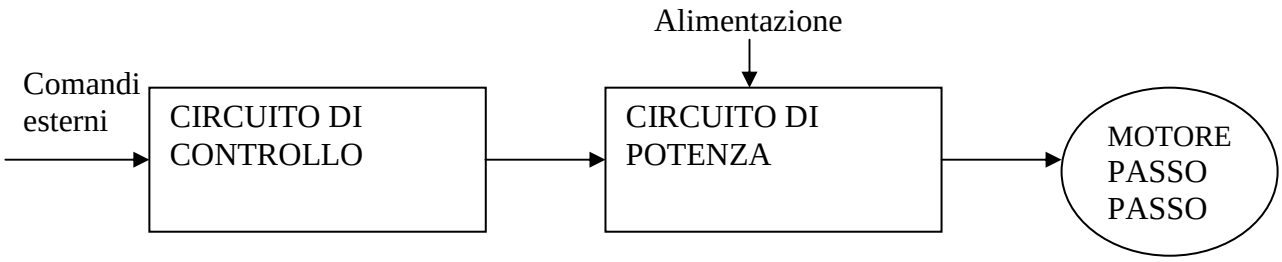
INTRODUZIONE

Si è ipotizzato di gestire i vari servizi necessari in un'abitazione come l'illuminazione, il riscaldamento, l'allarme, l'irrigazione esterna, l'apertura automatizzata dei cancelli tramite un PC che comunica,mediante una rete (BUS),tipo quelle utilizzate in ambito industriale (RS485),con vari microcontrollori (PIC) che acquisiscono lo stato delle variabili da controllare (temperatura,apertura e chiusura cancelli,grado di illuminazione,..).
Il PC in base ai valori ricevuti invia, tramite il bus, opportuni segnali di controllo ai corrispondenti attuatori (motore passo-passo,motore in corrente continua,pompe,..) in base ai valori reimpostati dall'utente ed accessibili e modificabili tramite opportune finestre grafiche (FORM).

In questo lavoro in particolare ci si è occupati della motorizzazione del cancello automatico.
Il progetto consiste nell'apertura e chiusura di un cancello elettrico mediante un motore passo-passo e una guida meccanica. Il motore , a sua volta sarà pilotato da un microcontrollore (PIC16F628A).

Inizialmente il progetto prevedeva che il comando di apertura fosse fornito da un telecomando a raggi infrarossi (I.R.) ma, per motivi di tempo, ci si è limitati ad un comando fornito da una chiave,senza,per questo,perdere generalità nel progetto (aspetto che peraltro potrà essere approfondito da altri studenti in futuro).

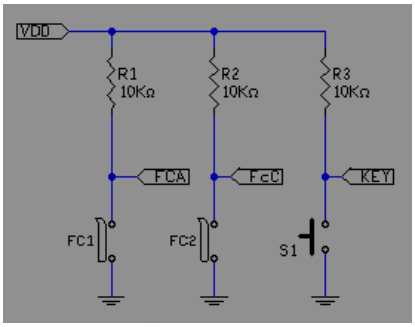
SCHEMA A BLOCCHI



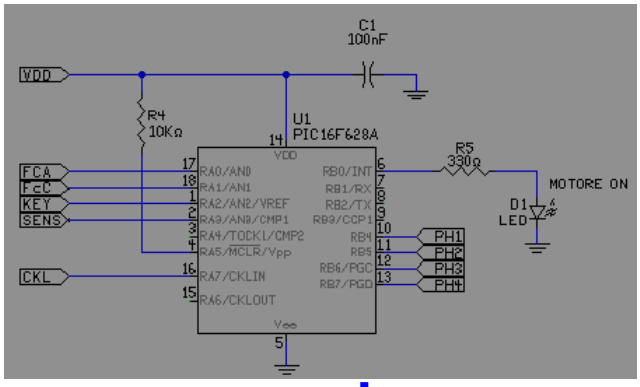
Spiegazione schema a blocchi:

Il circuito logico di controllo ha il compito di impostare l'azionamento(es: sequenza di movimenti) in base ai comandi esterni (finecorsa di apertura e di chiusura,fotocellula,chiave d'azionamento) e agisce sul commutatore di potenza che, oltre ad alimentare il motore in quanto collegato alla rete elettrica esterna, ha il compito di commutare l'alimentazione delle varie fasi secondo la sequenza prevista.

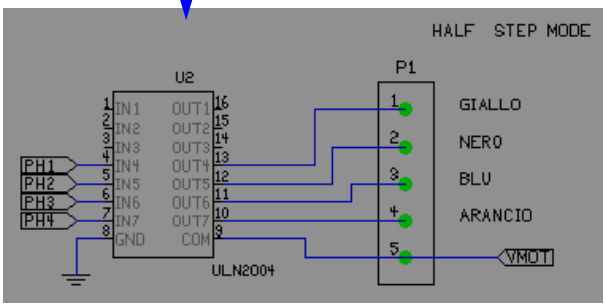
Schema circuitale dei vari blocchi del sistema



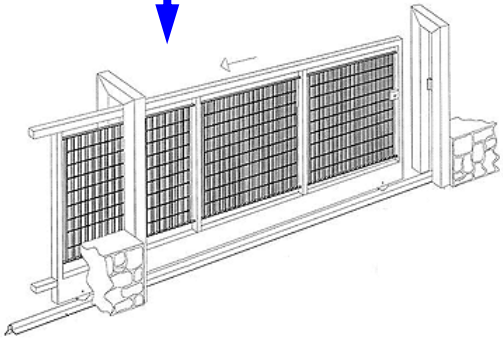
COMANDI ESTERNI



CIRCUITO DI CONTROLLO



CIRCUITO DI POTENZA



CANCELLO

HARDWARE

L'intero meccanismo del cancello elettrico è stato realizzato attraverso l'utilizzo del microcontrollore ***PIC16F628A*** *, di un motore passo-passo montato su guida meccanica e comandato dal PIC stesso , di un integrato* ***LM567*** *che coadiuvato da un apposito circuito realizza una fotocellula e di un integrato* ***ULN2003*** *che rende possibile il movimento del motore.*
Il circuito elettrico che fa capo all'azionamento e controllo del motore è stato realizzato su basetta e alimentato tramite l'ausilio di un trasformatore e di un integrato ***LM7805*** *che fornisce in uscita i 5V necessari per il funzionamento del circuito; il motore , invece, viene alimentato a 12V.*

Il funzionamento del circuito è molto semplice.

*Siccome tutto il processo è governato dal PIC , quest'ultimo attende che la rotazione della chiave avvenga per generare una sequenza di bit che, ciclando, permette al motore di ruotare versodestra, muovendo il cancello (****FASE DI APERTURA****) fino a raggiungere il primo fine-corsa.*

*Con il fine-corsa chiuso il programma entra in una fase di stop nella quale il cancello resta fermo per circa 5 secondi (****FASE DI ATTESA****).*

*Dopo di che il PIC controlla lo stato del bit riguardante la fotocellula;se questo si trova a livello logico basso il PIC stesso fornisce una sequenza di bit inversa a quella precedente in modo tale da far ruotare il motore in senso opposto e , giungendo al secondo fine-corsa,chiudere automaticamente il cancello. Se , invece , il bit della fotocellula si trova a livello logico alto il PIC non esegue nulla e resta fermo per altri 5 secondi ripetendo quest'operazione fino a che il bit si trova a livello logico basso e il cancello si chiude (****FASE DI CHIUSURA****).*
Questo accade perché , quando la fotocellula è attivata , significa che vi è un ostacolo che si contrappone al raggio d'azione della fotocellula stessa.
Durante la fase di apertura e quella di chiusura un led lampeggiante segnala che il motore è in movimento.

COMPONENTI UTILIZZATI

MOTORE PASSO-PASSO

I motori passo-passo sono diversi dagli altri motori; lavorano tramite impulsi elettrici invece che con una tensione che faccia girare il motore. Gli impulsi elettrici agli avvolgimenti del motore fanno girare il rotore di un grado preciso definito appunto "passo".

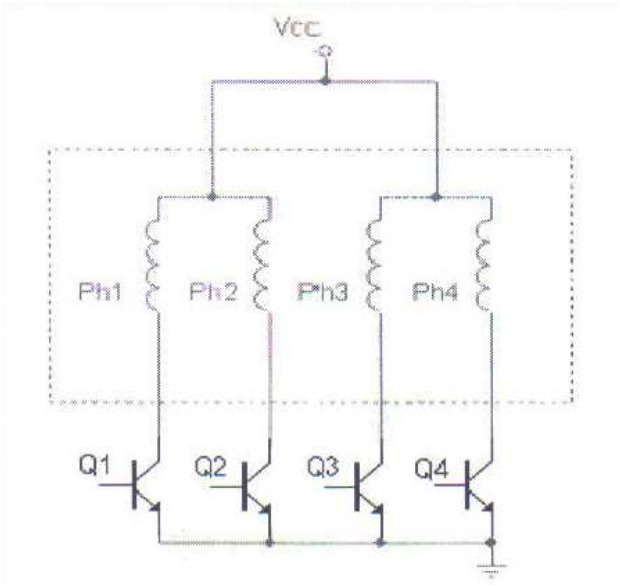
I motori passo passo sono dei motori elettrici che permettono all'utente di definire l'angolo di rotazione con una buona precisione ed invertire con estrema semplicità il senso o la velocità di rotazione senza apportare modifiche circuitali, ma solo attraverso degli impulsi digitali. Esistono due tipi di motori passo passo, quelli unipolari e quelli bipolari

Bipolare*: Lo statore presenta due avvolgimenti o fasi, facenti capo rispettivamente ai terminali essi vengono percorsi dalla corrente nei due sensi. Il rotore è costituito da un magnete permanente cilindrico, solitamente di materiale ceramico, sulla cui superficie laterale si affacciano i poli magnetici.*
Unipolare: *La differenza di questi motori al tipoi precedente consiste nel fatto che le fasi vengono percorse dalla corrente in un sol verso. Ciascun avvolgimento vien sdoppiato in due fasi aventi un' estremità in comune. Di solito possono trovarsi dai quattro fili in su.*

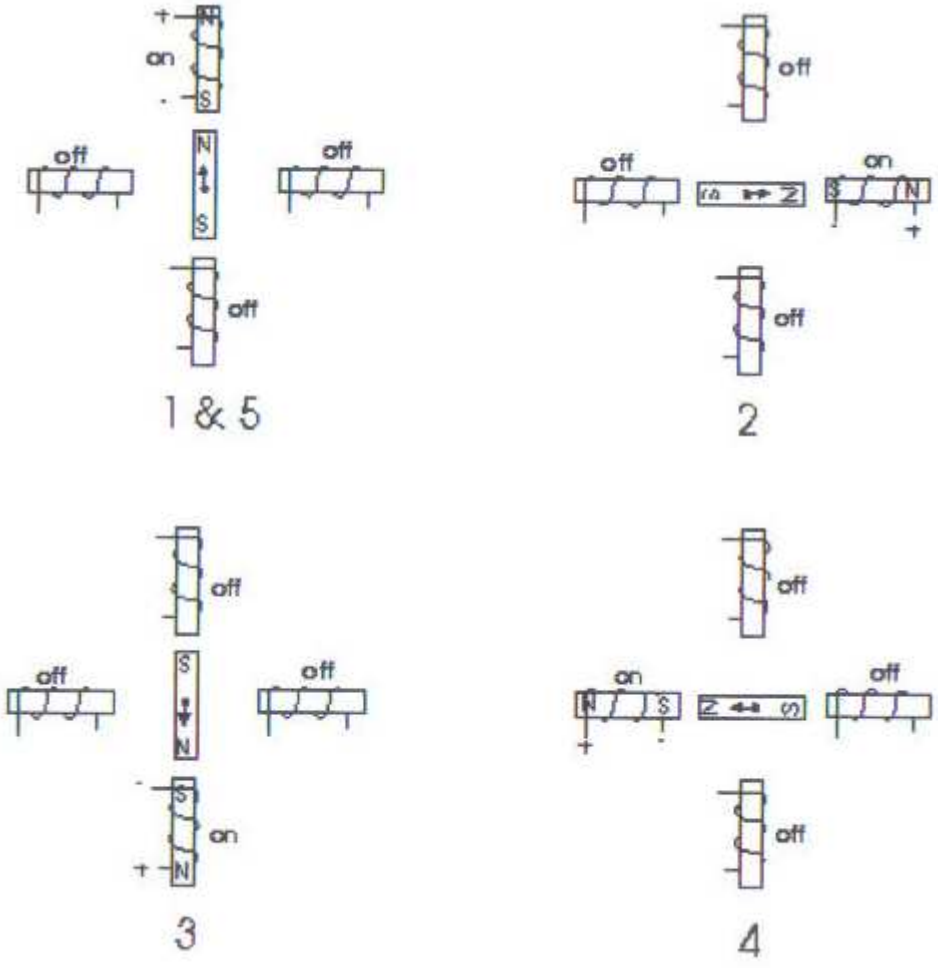
I motori passo passo (stepper) a prima vista sembrano dei normali motori elettrici forse un po' grandi e pesanti; l'unica cosa che li contraddistingue è il numero di cavi che fuoriescono dal loro interno, infatti i motori elettrici normali hanno sempre due cavi, uno per il polo positivo, e l'altro per quello negativo.

I motori passo passo, invece, hanno 5 o 6 cavi. Questa differenza è dovuta al loro particolare modo di funzionamento. Ognuno dei cavi viene utilizzato per pilotare il motore. Come tutti i motori elettrici, i motori passo passo sono formati da un perno centrale chiamato albero che è la parte che effettivamente ruota. Nei motori passo passo non ci sono altre parti mobili. Il perno è retto da dei cuscinetti a sfera che gli permettono di ruotare con grande precisione e poco attrito.

Tutto attorno all'albero ci sono varie bobine (o avvolgimenti) di filo conduttore che al passaggio della corrente, per un noto principio fisico, diventano degli elettromagneti. Quando una di queste bobine viene caricata (si dice anche energizzata) attira a se qualsiasi materiale ferroso, in particolare l'albero del motore.
L'albero tende a spostarsi di fronte alla bobina caricata così come un chiodo di ferro viene attratto da una calamita.
L'albero, che è vincolato dai cuscinetti a sfera, può solo ruotare nella direzione della bobina caricata fino a trovarsi di fronte alla bobina. Per ottenere una completa rotazione dell'albero bisogna energizzare in sequenza tutte le bobine.
Nei motori passo passo la sequenza non è scandita automaticamente, ma viene indicata dall'utente nei modi più diversi. In questo modo la sequenza può essere invertita, può essere scandita solo in parte o solo un passo della sequenza alla volta, in pratica a piacere dell'utente.
Questo è uno dei vantaggi di questo tipo di motori, è possibile invertire la rotazione in qualsiasi momento, eseguire solo una certa parte della rotazione e quindi una ben determinata rotazione angolare ecc.
A questo punto entra in gioco il numero di cavi del motore passo passo. Tutti gli avvolgimenti sono suddivisi in quattro gruppi chiamati fasi, gli avvolgimenti di ogni fase non sono posti in sequenza, ma sono alternati con gli avvolgimenti delle altre fasi in un preciso ordine. Ogni cavo del motore passo passo è collegato ad una ben precisa fase, mentre il cavo restante (o i due cavi restanti) rappresenta l'alimentazione.
Per energizzare una determinata fase basta applicare una determinata differenza di potenziale (tensione) tra il cavo associato a quella fase e il cavo di Vcc, in pratica bisogna chiudere il circuito relativo a quella particolare fase!
Per questo motivo esistono più di due cavi.
Dal punto di vista delle fasi il circuito che rappresenta un generico motore passo passo è il seguente:



Il numero delle fasi è per convenzione 4, mentre il numero di cavi può essere 5 o 6. Le induttanze del circuito rappresentano gli avvolgimenti, aprendo o chiudendo gli interruttori è possibile determinare quale fase energizzare alla volta.



Il disegno presenta ben quattro fasi è vuole chiarire il funzionamento del motore passo-passo , il rotore polarizzato sta iniziando all'elettromagnete superiore ,che è al momento attivo. In questo caso per far ruotare il rotore in senso orario, l'elettromagnete superiore viene disattivato e viene attivato quello destro (fase 2) , ciò causa lo spostamento di 90° del rotore che si allinea con il magnete attivo. Questo processo è ripetuto fino a che l'elettromagnete ritorna alla fase 1.
Nel nostro progetto il motore funziona in modalità half step.
Ci sono vari tipi di pilotaggio come:
E' necessario "scorrere" molte volte la tabella per ottenere la rotazione dell'albero di un giro.
Per esempio in un motore con quattro fasi e 200 passi/giro è necessario applicare per 200 volte (400 volte per l'half-step) gli impulsi di corrente per ottenere la rotazione di un giro dell'albero: in pratica occorre
scorrere 50 volte la tabella.
Ci sono altri tipi di pilotaggio come:

Wavemode*: è il sistema base di funzionamento; con esso la corrente è applicata ad una sola delle fasi alla volta, secondo la tabella seguente :*

Passo	Ph1	Ph3	Ph2	Ph4
1	I	0	0	0
2	0	I	0	0
3	0	0	I	0
4	0	0	0	I

Per ottenere la rotazione del motore è necessario scorrerre le righe della tabella, cambiando lsa fase in cui la corrente scorre. E' necessario tener presente che la tabella deve essere vista come circolare : dopo l'ultima riga , ritroviamo infatti la prima.

Two phase-on*: la corrente è applicata contemporaneamente due fasi. La coppia disponibile è circa1.4 volte maggiore di quella ottenuta con una sola fase attiva alla volta. Il consumo di corrente e quindi il riscaldamento raddoppiano. Questo fatto potrebbe creare problemi in alcuni motori non adatti ad alcuni tipi di pilotaggio.*

Passo	Ph1	Ph3	Ph2	Ph4
1	I	I	0	0
2	0	I	I	0
3	0	0	I	I
4	I	0	0	I

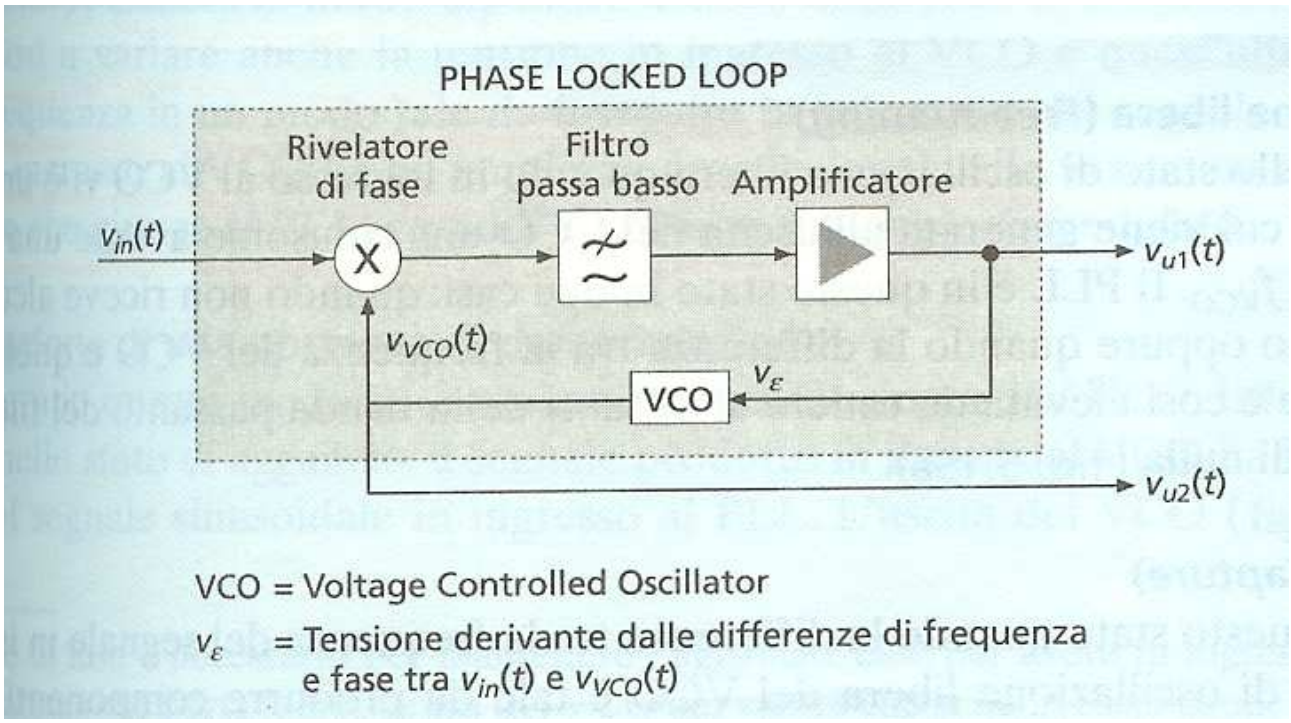
Half-step*: è in pratica l'alternarsi delle configurazioni dei due metodi appena visti. Il vantaggio è che raddoppia il numero di passi disponibile per un certo motore. Applicando continuamente la sequenze 1-2-3-4-1-2… si ottiene la rotazione dell'albero in un verso; per invertire il senso di rotazione basta invertire l'ordine con il quale sono lette le righe delle tabelle: 4-3-2-1-4… (non va quindi cambiato il verso delle correnti, che rimane invariato).*

Passo	Ph1	Ph3	Ph2	Ph4
1	I	0	0	0
2	I	I	0	0
3	0	I	0	0
4	0	I	I	0
5	0	0	I	0
6	0	0	I	I
7	0	0	0	I
8	I	0	0	I

E' necessario "scorrere" molte volte la tabella per ottenere la rotazione dell'albero di un giro.
Per esempio in un motore con quattro fasi e 200 passi/giro è necessario applicare per 200 volte (400 volte per l'half-step) gli impulsi di corrente per ottenere la rotazione di un giro dell'albero: in pratica occorre
scorrere 50 volte la tabella.

LM567

*Il **PLL** (Phase Locked Loop) é un circuito retroazionato composto da un moltiplicatore (mixer),un filtro passa-basso e un **VCO** (Voltage Controlled Oscillator).*
*Il **MIXER** realizza un comparatore di fase che fornisce un'uscita proporzionale alla differenza di frequenza e di fase tra il segnale che giunge dall'esterno e quello generato dal **VCO.***
*Il **FILTRO PASSA-BASSO** elimina i termini spuri,in alta frequenza,che vengono generati dal moltiplicatore prelevandone il valor medio (Vm);determina inoltre la caratteristiche dinamiche del **PLL**;poiché a regime la differenza di fase è molto piccola risulta utile introdurre un amplificatore dopo il filtro,per aumentare la sensibilità del **VCO.***
*Il **VCO** è un oscillatore la cui frequenza varia in modo proporzionale alla tensione applicata al suo ingresso.*



*Il funzionamento del **PLL** evolve secondo i tre stati che seguono.*

1. ***OSCILLAZIONE LIBERA (Free running)***

*Un **PLL** è nello stato di oscillazione libera quando in ingresso al **VCO** vi è una tensione nulla,per cui viene generata all'uscita del **VCO** una sinusoide avente una frequenza prefissata: **fVCO:***
*Il **PLL** è in questo stato in due casi:quando non riceve alcun segnale dall'esterno oppure quando la differenza tra la frequenza del **VCO** e quella del segnale*

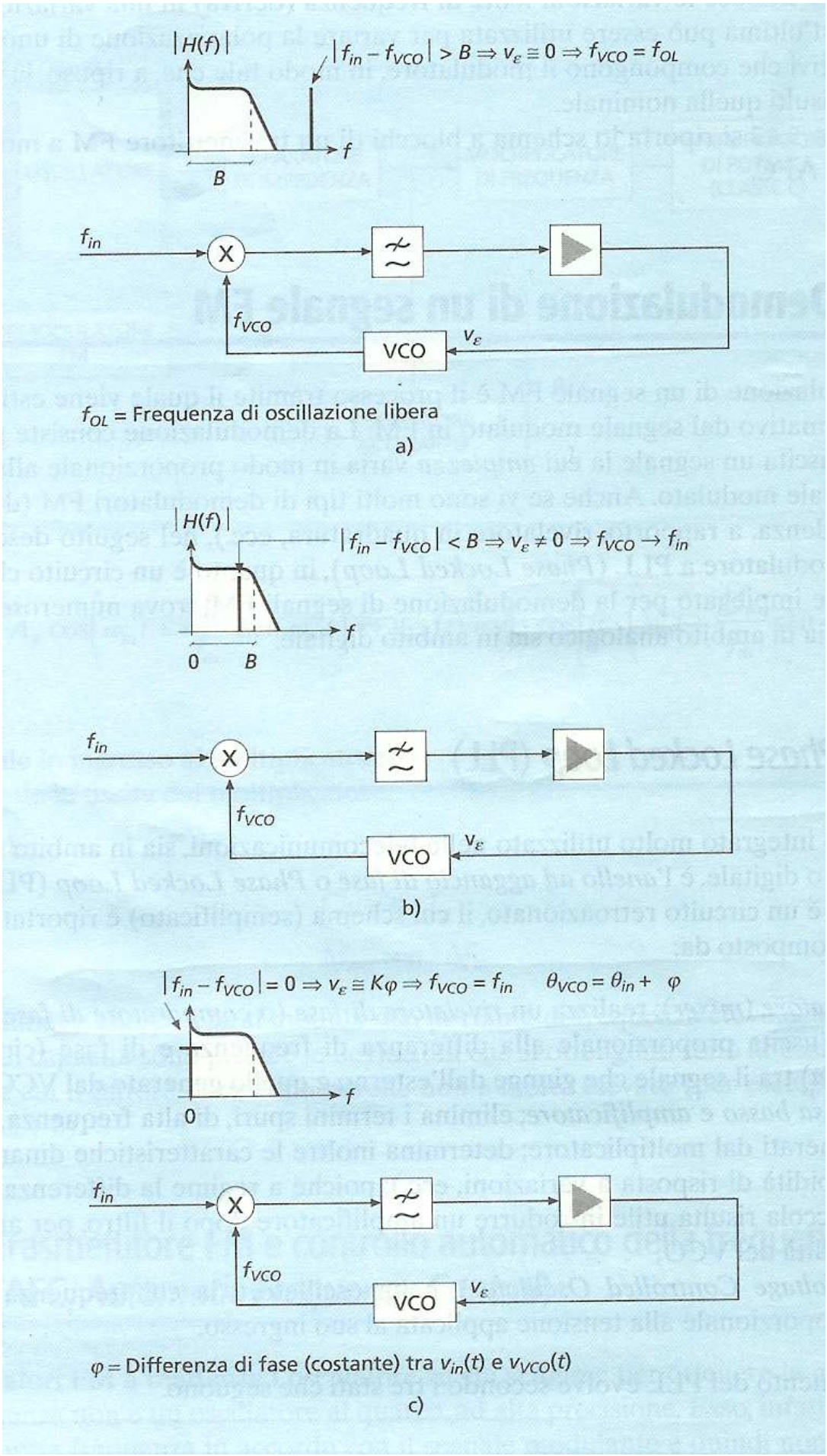
esterno è così elevata da cadere al di fuori della banda passante del filtro,la cui uscita è quindi nulla.

2. CATTURA (Capture)

*Il **PLL** è in questo stato quando la differenza tra la frequenza del segnale in ingresso la frequenza di oscillazione libera del **VCO** è tale da produrre componenti spettrali che cadono nella banda del filtro passa basso,per cui quest'ultimo produce un'uscita diversa da zero.*
*In ingresso al **VCO** si ha così una tensione che ne fa variare la frequenza e la fa avvicinare a quella del segnale in ingresso.*
*Lo stato di cattura è in sostanza un transitorio durante il quale la frequenza del **VCO** continua a variare finchè non risulta esattamente uguale a quella del segnale in ingresso.*

3. AGGANCIO DI FASE (Phase –lock)

*Il **PLL** è in questo stato quando l'uscita del **VCO** è un segnale sinusoidale avente la stessa frequenza del segnale in ingresso e un piccolo errore di fase;il segnale uscente dal **VCO** è così un segnale agganciato in frequenza e fase al segnale ricevuto,in quanto differisce da esso solo per un piccolo errore di fase.*
*Nel caso in cui il segnale in ingresso non sia puramente sinusoidale,il **PLL** si aggancia alla componente spettrale che rientra nel suo campo di cattura e che lo porta nello stato di aggancio.*
*Se il **PLL** riceve un segnale utile affetto da rumore,esso è comunque in grado di agganciarsi al segnale in quanto il rumore ha un andamento casuale,per cui non è in grado di portare il **PLL** in uno stato diverso da quello di aggancio sul segnale utile.*
*Se però la differenza tra la frequenza del segnale in ingresso e quella del **VCO, fIN-fVCO**,diventa eccessiva e cade al di fuori della banda del filtro passa basso,allora il segnale uscente dal mixer viene totalmente eliminato dal filtro stesso,per cui l'ingresso del **VCO** va a zero e il **PLL** viene riportato nello stato di oscillazione libera:il **PLL** ha perso l'aggancio.*


|H(f)|
$|f_{in} - f_{VCO}| > B \Rightarrow v_\varepsilon \equiv 0 \Rightarrow f_{VCO} = f_{OL}$
f
B
f_{in}
X
f_{VCO}
VCO
v_ε
f_{OL} = Frequenza di oscillazione libera
a)
|H(f)|
$|f_{in} - f_{VCO}| < B \Rightarrow v_\varepsilon \neq 0 \Rightarrow f_{VCO} \to f_{in}$
f
0
B
f_{in}
X
f_{VCO}
VCO
v_ε
b)
$|f_{in} - f_{VCO}| = 0 \Rightarrow v_\varepsilon \equiv K\varphi \Rightarrow f_{VCO} = f_{in}$
$\theta_{VCO} = \theta_{in} + \varphi$
f
0
f_{in}
X
f_{VCO}
VCO
v_ε
φ = Differenza di fase (costante) tra $v_{in}(t)$ e $v_{VCO}(t)$
c)

Nella figura sono rappresentati i tre stati che caratterizzano il PLL:
a)oscillazione libera
b)stato di cattura
c)stato di aggancio

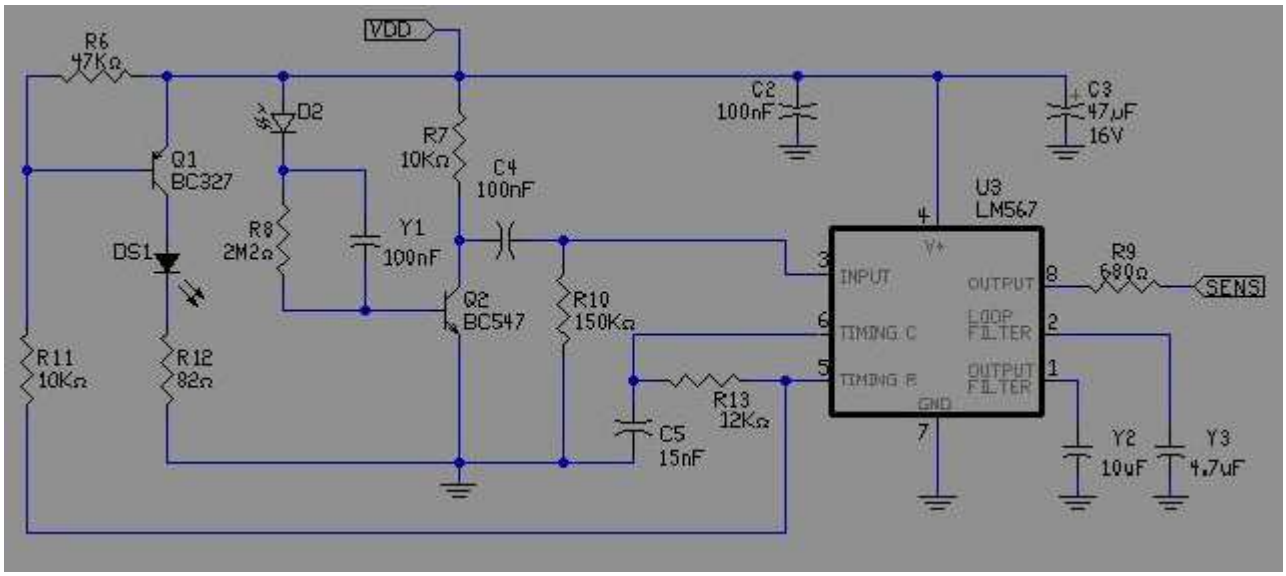
*Il **PLL** viene utilizzato per la demodulazione di segnali FM o PM,per la ricostruzione di una portante di demodulazione,per la ricostruzione del clock di ricezione nei sistemi digitali e come sintetizzatore di frequenza.*

*Nel nostro caso,il **PLL**,è utilizzato per determinare lo stato della fotocellula ed inviare il segnale in ingresso al microcontrollore che,attraverso opportune sequenze,determina lo stato del cancello (attesa o chiusura).*
Questo avviene in quanto il PLL genera un segnale a frequenza interna al VCO che, finchè resta agganciata(il raggio emesso dai due diodi led non interrotto) l'uscita sarà nulla e il cancello si chiuderà liberamente;se il segnale si sgancia dal PLL (raggio interrotto da un ostacolo),l'uscita del PLL sarà alta e il PIC provvederà a tenere aperto il cancello ed attendere che il raggio torni uniforme per poter chiudere il cancello.

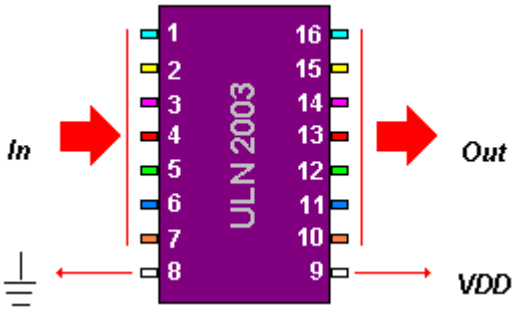
*Fo=1/(1,1*R1*C1)=5 kHz frequenza interna PLL*

R13=12 kΩ
C5=15 nF

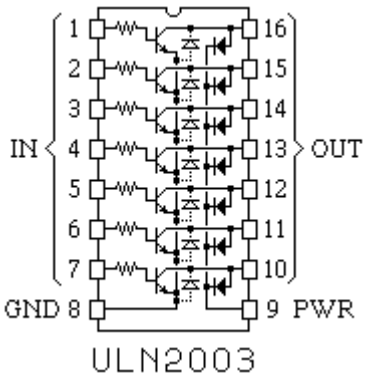
La frequenza interna al PLL è uguale alla frequenza di oscillazione libera del VCO.



ULN2003



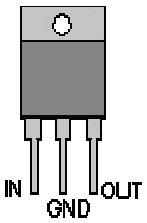
Il problema di pilotare dei motori passo passo è che, generalmente, si vuole che ciò sia automatico,ovvero si vuole costruire un circuito che mandi automaticamente al motore passo passo i segnali giusti.
Un tale circuito può essere realizzato tramite degli integrati il cui compito è aprire o chiudere opportunamente i circuiti delle fasi. Quindi il circuito di pilotaggio deve controllare degli interruttori.
Il problema è che i motori passo passo funzionano con una differenza di potenziale di 12 volt, mentre i circuiti integrati (con i quali abbiamo ipoteticamente realizzato il circuito di pilotaggio) sono realizzati in tecnologia TTL o CMOS e quindi funzionano a 5 volt. Bisogna, quindi, disporre di alcuni interruttori pilotabili da una tensione di 5 volt, ma che siano in gradi di aprire o chiudere circuiti da 12V.
Si può aggirare il problema utilizzando un'interfaccia tra il circuito a 5v e quello a 12v. In pratica si utilizzano interruttori da 5v e la loro uscita viene convertita a 12v.
L'integrato utilizzato è un ULN2003 (array di Darlington)



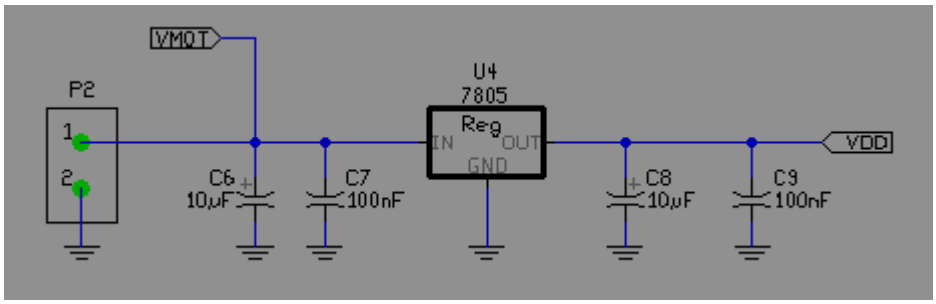
Nell'ULN2003 i primi 7 pin ricevono in input una tensione di 5v proveniente da un circuito di tecnologia CMOS o TTL, i pin dal 10 al 16 portano in uscita una tensione di 12 v adatta per i motori passo passo. I piedini 8 e 9 sono rispettivamente la massa e l'alimentazione a 12v.

Questo integrato è in pratica un'interfaccia tra il motore passo passo che lavora con 12 v ed una logica di pilotaggio che invece lavora con 5v.
Anche utilizzando questo integrato bisogna comunque inserire il diodo di ricircolo, in questo caso può essere utilizzato un diodo zener da 12v.

LM7805



Stabilizzatore a tre piedini che con in ingresso una tensione avente un certo ripple, garantendogli una certa tensione di drop,tra in e out, di 2-3 v, per il mantenimento,in uscita fornisce una tensione stabilizzata.
Nel caso del componente LM7805 la tensione fornita in uscita è di 5v.



In questo progetto è stato utilizzato per realizzare,con l'ausilio di 4 condensatori,l'alimentazione del circuito che controlla il movimento del cancello e tutto ciò ad esso connesso.
La tensione in ingresso al componente è fornita da un trasformatore che in uscita offre 12v alternati, utilizzati per alimentare il motore,che verranno raddrizzati dai condensatori e mantenuti dal LM7805.

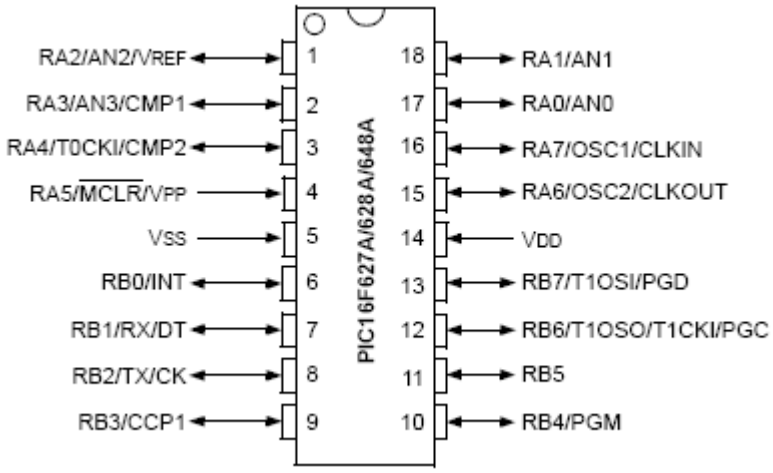
PIC16F628A

Il dispositivo PIC 16F628A è un microcontrollore particolarmente indicato per applicazioni didattiche e quindi può essere tramite apposito programmatore, cancellato elettricamente e nuovamente riprogrammato.

Le caratteristiche principali del PIC 16f628A:

- *Memoria di programma EEPROM con un estensione di 2k- word per 14 bit (2048 parole); memoria RAM dati 224 bytes e memoria dati EEPROM di 128 bytes.*
- *Massima frequenza di clock pari a 20MHz.*
- *Alimentazione compresa tra 3-5.5 Volt.*
- *Tre timer denominati rispettivamente TMR0 , TMR1, TMR2.*
- *Il dispositivo può essere interrotto durante il nor,male funzionamento del programma per eseguire routine d'interrupt in zone distinte dal programma principale.*
- *Variabili si inizializzano a partire dall'indirizzo 20h.*
- *Possiede 16 linee dedicate ad operazioni di input/output distinte in due gruppi: PORTA PORTB 8 linee ciascuno*
- *Contiene un insieme pin 36 codici tutti con un estensione di 14 bit. I codici sono rappresentati con mnemonici in linguaggio assembly utilizzato per qualunque microcontrollore della microchip.*

PIEDINATURA INTERNA PIC 16F628A



Sono presenti su chip:

- *Due linee per l'alimentazione Vdd (da 3 a 5.5 Volt) e Vss (massa).*
- *Otto linee del PORTA da RA0 a RA7. Ogni linea può essere programmata indipendentemente dalle altre come input o output. La linea RA4, può essere utilizzata anche come clock esterno per l'utilizzo per il timer TMR0 o come comparatore. La linea RA3 può essere abilitato anch'essa come comparatore.*
- *RA6 e RA7 possono essere utilizzate come oscillatori rispettivamente OSC2/CLKOUT e OSC1/CLKIN per il collegamento con l'oscillatore esterno o per l'uscita di un segnale di clock e per l'uscita del clock.*
- *Otto linee del PORTB da RB0 a RB7. La linea RB0 è anche utilizzata come linea d'ingresso per segnali d'interrupt.*
- *L' ingresso RA5 che può essere utilizzato come MCRL (linea di reset) attiva a livello basso.*

FOTOEMETTITORI

I componenti optoelettronici sono dispositivi che possono interagire con onde elettromagnetiche di lunghezza d'onda compresa fra 'infrarosso e l'ultravioletto.
Un dispositivo optoelettronico è un emettitore quando genera un'onda elettromagnetica,cioè è un dispositivo in grado di convertire energia elettrica in una radiazione luminosa.
I diodi emettitori di luce hanno la proprietà di emettere una radiazione luminosa quando,per effetto di una polarizzazione diretta del diodo,si ha il fenomeno della ricombinazione delle cariche elettriche.
*I diodi led emettono radiazioni che si collocano nello spettro visibile e radiazioni comprese nella banda dell'infrarosso (**IRED**).*
*I diodi **IRED** sono largamente utilizzati nel campo della comunicazione dati,nei sistemi di controllo e nei fotoaccoppiatori.*

La generazione di una coppia elettrone-lacuna in un materiale semiconduttore può avvenire per effetto dell'energia fornita da una radiazione luminosa di opportuna lunghezza d'onda che permette all'elettrone presente nella banda di valenza di saltare nella banda di conduzione.
Nel processo di ricombinazione di un elettrone e di una lacuna,l'elettrone passa dalla banda di conduzione alla banda di valenza emettendo energia sotto forma di calore ,che viene assorbita e dispersa dal cristallo ,oppure sotto forma di una radiazione luminosa.

La radiazione è prodotta dalla ricombinazione diretta fra le bande di conduzione e di valenza o da transizioni dei portatori di carica fra i livelli di energia intermedi,quali quelli degli atomi donatori e accettori,presenti nella banda interdetta.
La possibilità che un processo di ricombinazione emetta una radiazione luminosa dipende dal tipo di materiale utilizzato,che può essere del tipo diretto o indiretto.

La lunghezza d'onda λ della radiazione emessa dipende dal salto di energia necessario per il passaggio degli elettroni dalla banda di conduzione a quella di valenza.

$$\lambda = (h*c)/\Delta W$$

dove:
*h è la costante di Planck (=4,135 *10^-15 eV*s)*
*c è la velocità della luce (=2,99*10^8 m/s)*
ΔW è la differenza di energia tra le due bande

*I diodi **IRED** sono creati per funzionare con lunghezza d'onda λ inferiore a 600 nm.*

Nel nostro progetto questi componenti sono utilizzati,con opportuna rete di controllo,per la realizzazione della fotocellula,in quanto,quando montati in modo da trovarsi uno di fronte all'altro,emettono un fascio di luce del campo degli infrarossi. Il tutto è pilotato dal componente LM567 che svolge la funzione di PLL.
Quando il fascio di luce resta uniforme il PLL resta agganciato e in ingresso al PIC ci sarà uno 0 logico che permetterà al motore passo-passo di muoversi tranquillamente per la chiusura del cancello;mentre se il fascio viene interrotto da un ostacolo esterno il PLL si sgancia inviando un 1 logico al PIC che,tramite opportuna procedura,blocca il motore e attende che il PLL si riagganci per poter eseguire la fase di chiusura del cancello

Simbolo grafico di un diodo IRED

TRASMISSIONE

TRASMISSIONE SERIALE

In collegamenti a lunga distanza non è possibile adottare la trasmissione parallela, giacchè i costi derivanti dalla realizzazione e dalla posa dei cavi risulterebbero eccessivi. È stata così introdotta una modalità di trasmissione adatta ad essere utilizzata in collegamenti di lunghezza qualsiasi e denominata trasmissione seriale. La trasmissione seriale consiste nell' inviare sequenzialmente , su una singola linea i dati da trasferire, un bit dopo l'altro. Poiché un elaboratore opera al proprio interno in modo parallelo (è presente un 2bus2 che trasporta un certo numero di bit alla volta), è necessario impiegare un' apposita circuiteria per passare da tx parallela a seriale e viceversa, i cui elementi base sono dei registri a scorrimento di tipo(PISO)e SIPO. I circuiti integrati che, tra l'altro, svolgono tale funzione sono denominati UART , utilizzabile per la sola trasmissione seriale asincrona, e USART, utilizzabile per trasmissioni seriali sia asincrone che sincrone.
Nella trasmissione seriale è di fondamentale importanza la temporizzazione dei segnali da scambiare, ottenuta tramite appositi clock.
Infatti :

a) *lato trasmissione è indispensabile utilizzare un clock per serializzare ,nonché per trasmettere ogni singolo bit come un impulso avente una durata ben precisa;*
b) *lato ricezione è indispensabile utilizzare un clock, in qualche modo agganciato a quello di trasmissione, per poter leggere i bit ricevuti nell' istante migliore(a metà del tempo di bi), minimizzando così la probabilità di errore. Inoltre è necessario che vi sia un' ulteriore sincronizzazione che permetta di riconoscere i diversi caratteri(o più in generale i campi informativi)presenti all' interno del flusso di bit ricevuto. Normalmente nella trasmissione seriale i segnali elettrici vengono associati ai valori logici (0,1) dei singoli con la seguente convenzione:*

0 → + V_0 (livello alto), questo stato viene anche denominato space.
1 → -Vo (livello basso), questo stato viene anche denominato mark.
A seconda della modalità con al quale viene realizzata la sincronizzazione sul bit, vi sono due tipi di trasmissione seriale: la trasmissione seriale asincrona e la trasmissione seriale sincrona. Il DCE è caratterizzato dal bit rate che è in grado di accettare in ingresso dalla propria velocità di modulazione o baud rate, cioè il numero di simboli/s che esso emette.

TRASMISSIONE SERIALE ASINCRONA

La trasmissione seriale asincrona è caratterizzata dal fatto che i clock impiegati lato trasmissione e lato ricezione sono indipendenti; essi devono avere una precisione tale da consentire la corretta lettura dei bit che costituiscono un singolo carattere. Finchè è attivo il collegamento, la trasmissione di un carattere può avvenire in un momento qualsiasi ed è indipendente dalla trasmissione di altri caratteri. In altri termini tra due caratteri trasmessi può intercorrere un intervallo di tempo qualsiasi. Il ricevitore deve quindi essere attivato, assieme al suo clock, solamente nel momento in cui giunge un carattere e una volta effettuata la ricezione esso può tornare in una condizione di riposo. Nella trasmissione asincrona è perciò indispensabile inserire dei bit di controllo che definiscano l'inizio e la fine di un carattere e che vengono così denominati:
-bit di start è un impulso che precede i bit (dati) relativi ad un carattere; alla durata del tempo di un bit e polarità opposta rispetto a quella della line a riposo.
Con la convenzione sopra citata il bit di start è uno 0 logico.

-bit di stop: al termine della trasmissione dei bit di un carattere , la linea deve tornare a riposo per un certo tempo minimo; tale condizione di riposo obbligatoria viene denominata bit di stop e può avere durata pari ad 1, 1.5, 2 tempi di bit. Il bit di stop a quindi polarità opposta rispetto a quella del bit di start(è un 1 logico)e serve semplicemente per indicare la fine di un carattere, in modo di consentirne la determinazione sicura anche nel caso vengano trasmessi più caratteri uno di seguito all'altro.

- *bit di dati: sono bit informativi, il loro numero può variare. Per inviare un carattere di 8 bit è quindi necessario trasmettere un numero di bit compreso tra i seguenti valori :*
- *1 bit di start + 8 bit dati + 1 un bit di stop= 10 bit;*
- *1 bit di start + 8 bit dati + 2 bit di stop =11 bit.*

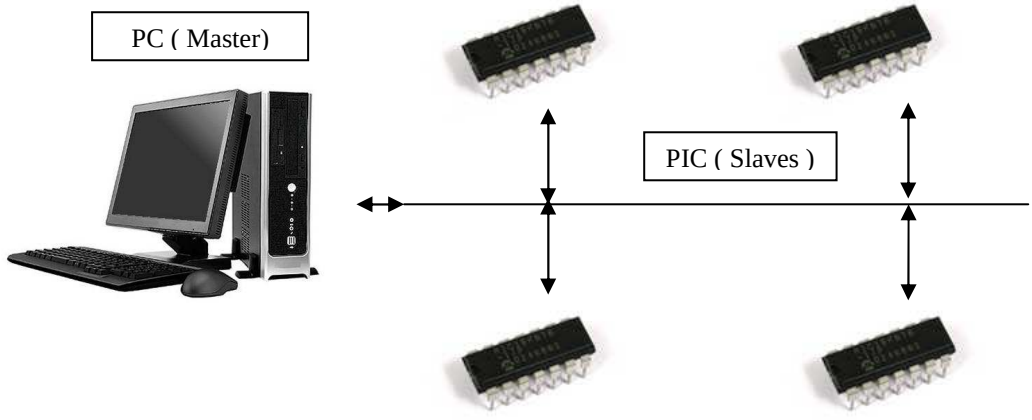
Va sottolineato che i bit di start e di stop servono esclusivamente alla sincronizzazione sul bit della trasmissione e quindi in un DTE asincrono vengono trattati a livello d'interfaccia seriale e non giungono al bus del pc. Inoltre poiché la linea torna a riposo al termine della trasmissione di un carattere è necessario che il clock di ricezione abbia una precisione sufficiente a garantire la corretta lettura di un singolo carattere. Il clock viene infatti riallineato (resettato) da ogni bit di start

PROBLEMA:COMUNICAZIONE TRA PC E VARI PIC

Lo scopo del nostro progetto è quello di realizzare un sistema di comunicazione gestito da un opportuno protocollo, che regoli la trasmissione dati tra un PC ed i sensori (di temperatura, di luce, citofoni… ecc) dislocati in una ipotetica casa elettronica. Si chiama demotica la scienza che si occupa delle applicazioni dell' informatica e dell' elettronica all' abitazione. Infatti il nostro progetto ha richiesto nozioni relative ai sistemi di telecomunicazione e sistemi informatici. I primi consentono lo scambio di informazioni digitali, o dati tra elaboratori, mentre i secondi scambiando dati e operando sotto il controllo di software opportuno, sono in grado di fornire agli utenti del sistema servizi di vario tipo. Nel nostro caso si trattava di comandare vari sensori facenti capo ad altrettanto micro controllori(PIC) grazie ad un PC con una interfaccia GUI (Interfaccia grafica , graphical user interface).

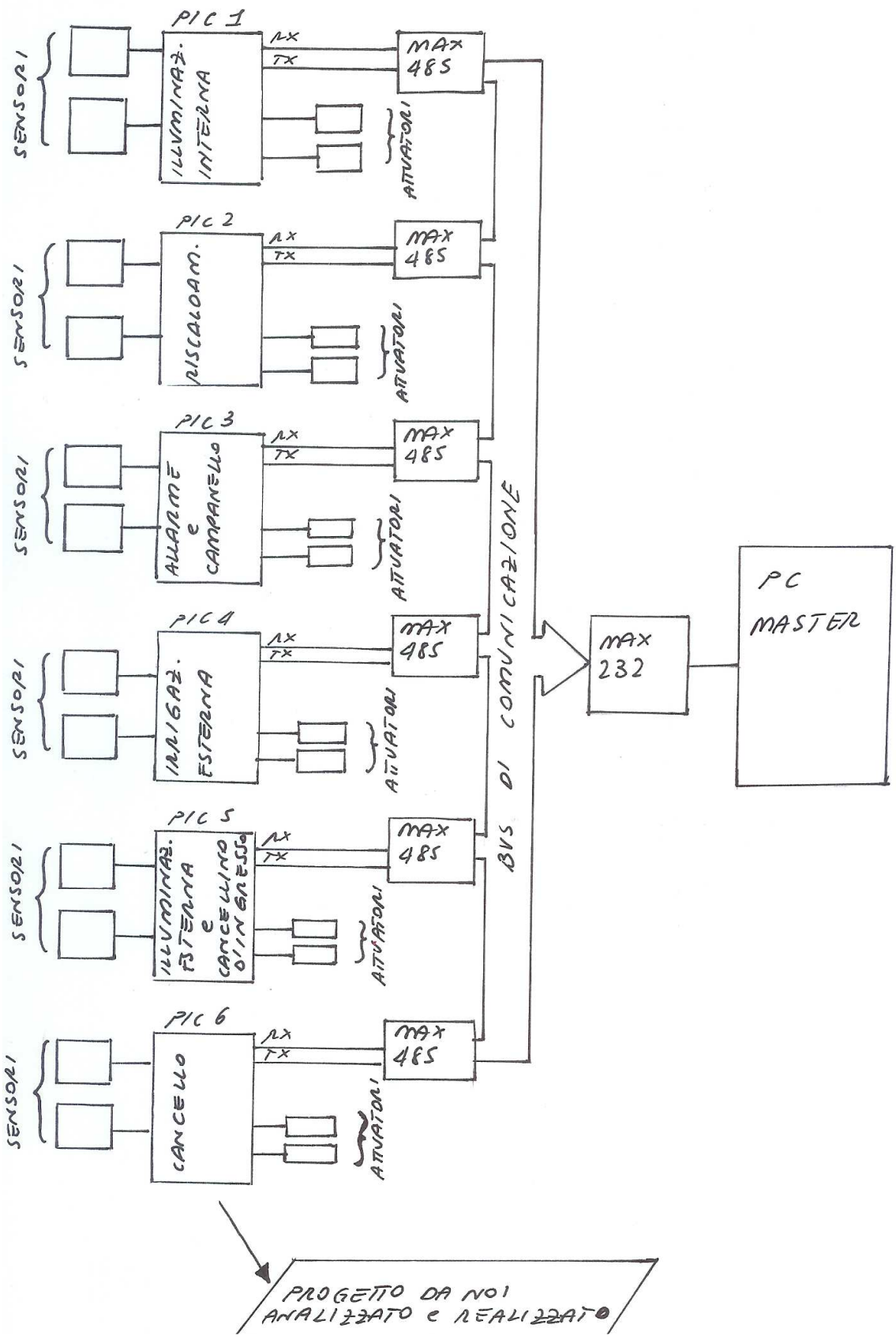
LA RETE DI COMUNICAZIONE:LO STANDARD RS485

Per la trasmissione abbiamo è stato realizzato un collegamento multipunto secondo lo standard RS485 essendoci la necessità di interconnettere in modalità Half-Duplex. In questo modo il mezzo trasmissivo che è condiviso permette di trasmettere dati tra queste, ma in modo alternato. Quindi prima il canale può essere usato per trasmettere dal primo al secondo punto, successivamente il secondo nodo può rispondere al primo e così via. Nel nostro caso essendoci più terminali, ovvero più PIC, che condividono un mezzo trasmissivo comune a tutti, è necessario impiegare un metodo di accesso multiplo, implementato da un opportuno protocollo, per regolamentare l'accesso al mezzo stesso, in modo che non si verifichino trasmissioni contemporanee, indicate con il termine di collisioni.

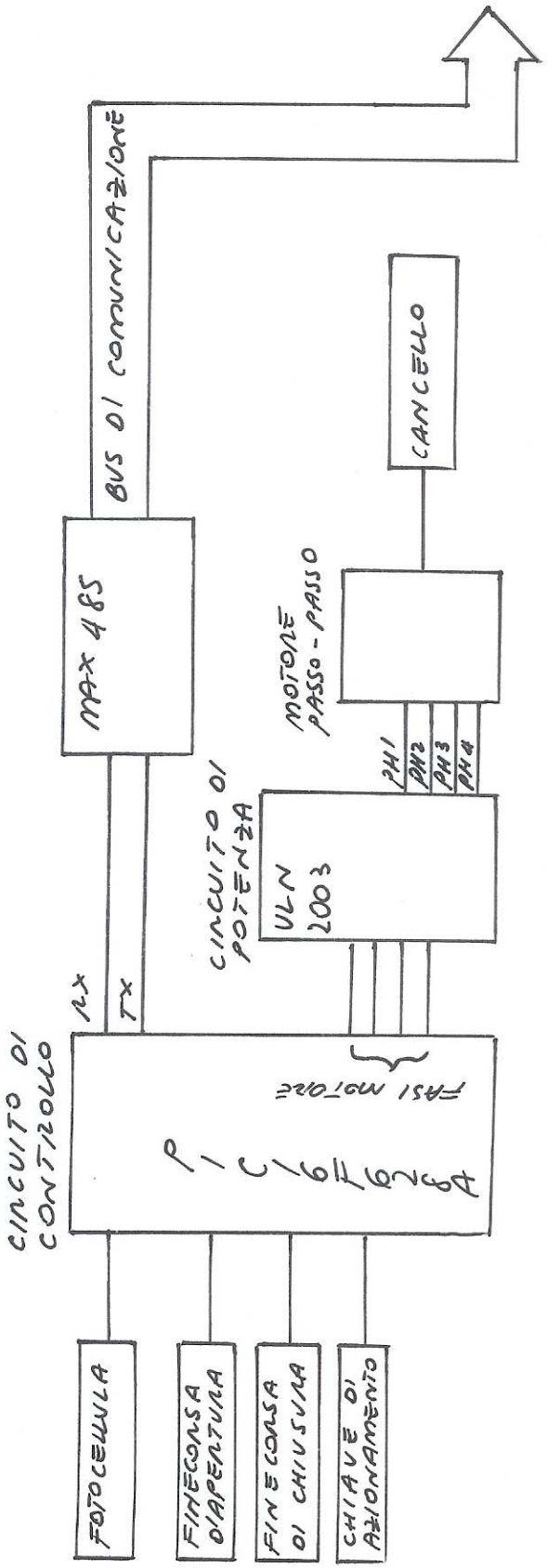


Quindi chiameremo chi è il "master", il pc che gestisce e regola la comunicazione, e "slaves", i sensori collegati ai corrispondenti PIC.
Di conseguenza il PC (che può essere anche denominato "server") controlla l'intero processo di comunicazione attraverso un programma principale basato su linguaggio Delphi. Grazie ad esso l'utente potrà amministrare la sua abitazione in modo semplice ed intuitivo.

SCHEMA REALIZZATIVO GENERALE DELLA COMUNICAZIONE TRA MASTER E SLAVES



SCHEMA DI RIFERIMENTO AL PROGETTO IN QUESTIONE

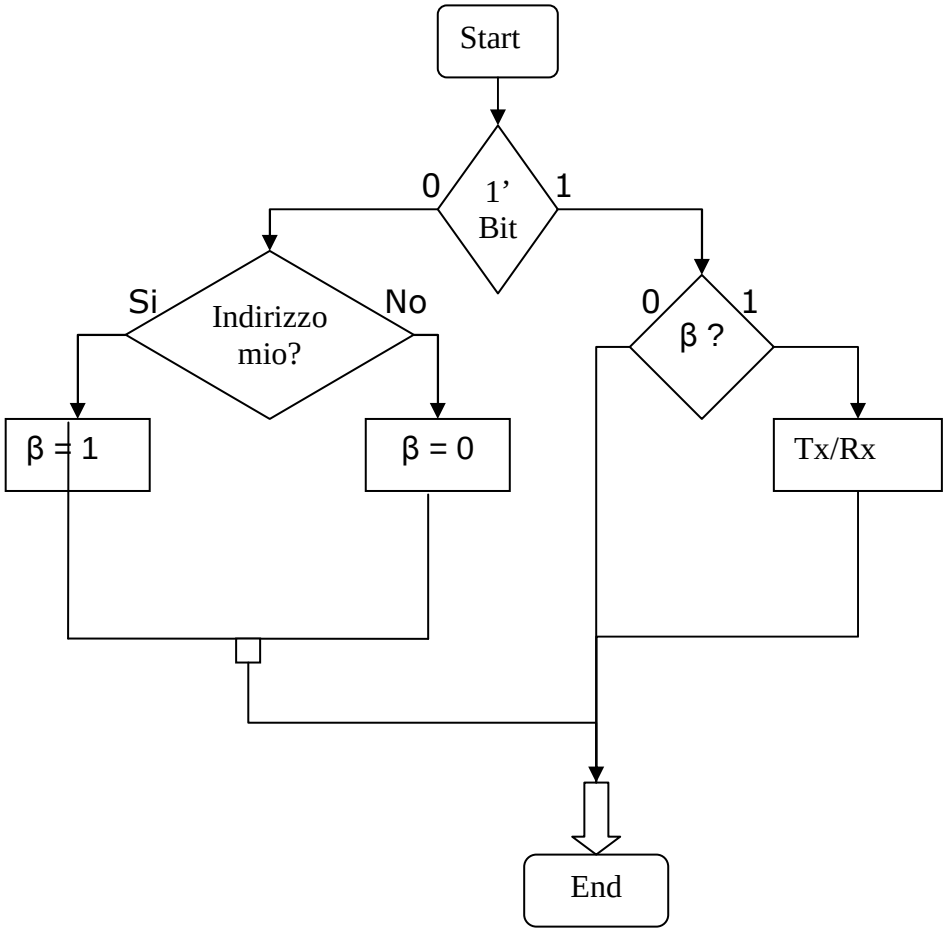


PROCEDURA CONTROLLO BIT:

La stringa come detto precedentemente è composta da 1 bit di I/D, i 3 successivi che indicano l'istruzione e gli ultimi 4 che possono essere indirizzamento o Dati.
I passaggi per analizzare la sequenza di Bit è la suddetta:

1. *Controllo del Primo Bit. Questo se è alto (1 logico) significa che la stringa è di Dati.Altrimenti (0 logico) indica che è un indirizzamento*
    a. *Caso 0 logico : La stringa è un indirizzamento e di conseguenza si salterà a vedere direttamente i 4 bit di indirizzo per vedere se si è i diretti interessati. In caso affermativo, si dovrà settare una variabile (β) a 1 in modo che successivamente si è abilitati a ricevere dei dati. In caso negativo invece si dovrà impostare la variabile (β) a 0, così successivamente nella comunicazione le stringhe di Dati verranno ignorate.*

    b. *Caso 1 Logico : Essendo una stringa Dati, prima di tutto si dovrà vedere se si era abilitati a parlare guardando una variabile (β). In caso affermativo si analizzano i 3 bit di Istruzione, e verifica tramite una comparazione che cos'è il dato che segue.Esempio : se l'istruzione è 010, il programma individuirà che il dato che è stato trasmesso è il valore impostato della temperatura; se fosse 001, invece, rappresenta il comando di fine tx ecc...*



STRUTTURA DELLA FRAME DI COMUNICAZIONE

Per effettuare la comunicazione PIC/PC si utilizza una Frame (trama) come illustrato in fig.1 :

BIT DI START - CAMPO DI INFORMAZIONE (8 BIT) - BIT DI STOP

FIG 1

Ogni singola trama è lunga di conseguenza 10 Bit :

1- *Bit di start che indica l'inizio del campo di dati.*
2- *8 Bit di istruzione suddivisi nel seguente modo :*
   a. *Il primo bit indica se la Frame è un'indirizzo (I) o un'istruzione (D)*
   b. *Gli altri 3 bit, se è una Frame di istruzione indicano l'istruzione stessa*

c. *I 4 bit successivi invece possono essere o il dato o l'indirizzo della porta a cui è destinata la trama.*

3- *Bit di stop che indica la fine della trasmissione*

I bit del campo di informazioni sono suddivisi nel seguente modo:

1 BIT I/D - 3 BIT Istruzione - 4 BIT di Indirizzo (I) o Dati (D)

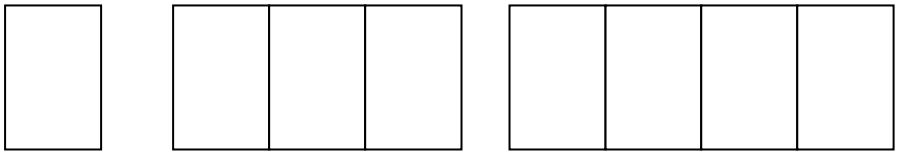


FIG 2.A

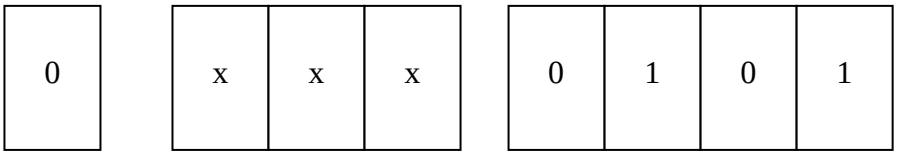


FIG 2.B

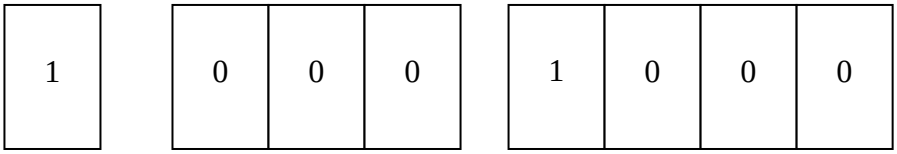


FIG 2.C

I 4 bit di indirizzo, indicano l'indirizzo del pic con il quale si vuole comunicare. Ad ogni gruppo della classe corrisponde uno o più PIC individuato da un indirizzo il quale sarà personale e non condivisibile con gli altri PIC. In questo modo non posso chiamare più periferiche contemporaneamente ma solo quella prescelta.

Ad ogni PIC, come spiegato in precedenza, di conseguenza è stato affidato il suo indirizzamento: '1000'

▪ *Durante l'invio e la ricezione dei dati, il PIC è in uso, ovvero non gli è possibile lavorare sul programma principale e di conseguenza è come "bloccato" in una routine di polling.*
Questo tempo varia dai 5ms ai 50ms.

▪ *Il massimo valore che può assumere un dato è di 2^8 Bit*

▪ *Il Quarzo dei PIC deve essere impostato a 20Mhz.*

▪ *La Baudrate deve essere impostata a 57600 ed il BRGH (High Baud Rate Select Bit), deve essere settato (BRGH = 1), ovvero lavorare in modalità high speed.*

SOFTWARE

Microcontrollori Pic

Prima di effettuare la progettazione di un sistema elettronico è opportuno definire tutti i parametri e le operazioni da gestire con una certa precisione.
Può succedere talvolta che in certe occasioni si debba andare a modificare tali parametri, per vari motivi, durante o dopo la fase di progettazione.
In questo caso è necessario avere a disposizione un sistema flessibile che consenta la modifica di tali parametri senza dover necessariamente rieseguire daccapo la fase di progettazione.
I microcontrollori rispondono a questa esigenza.
Questi,infatti,sono dei dispositivi integrati, programmabili e cancellabili più volte dall'utente,dotati al loro interno di una serie di dispositivi (linee di I/O,memorie, timer, convertitori A/D,USART,generatori di PWM,ecc…)già dimensionati e cablati tra loro così da mettere a disposizione dell'utente tutti i moduli di cui un sistema può avere bisogno.
Un microcontrollore può dunque ritenersi un dispositivo programmabile in grado di svolgere autonomamente diverse funzioni in relazione al programma in esso implementato.
In base alle istruzioni ricevute durante l'esecuzione di un programma presente in una memoria al suo interno il "micro" può svolgere varie operazioni,che in un progetto unicamente hardware sarebbero svolte da molti più componenti elettronici.
Un progetto che sfrutta la logica software è però più lento di uno meramente hardware,infatti il "micro" per svolgere le istruzioni del programma al suo interno implementato richiede un certo lasso di tempo,maggiore di quello richiesto per esempio ad una porta logica per mandare segnali in uscita.

STRUTTURA INTERNA DEI MICROCONTROLLORI

L'architettura interna delle varie famiglie di microcontrollori della MICROCHIP può essere ricondotta ad una struttura base alla quale vanno di volta in volta aggiunte le parti relative alle varie funzioni implementate nei diversi integrati. Quindi all'interno di un microcontrollore si è soliti evidenziare i seguenti blocchi funzionali:

- *l'**UNITA' ALGORITMICA** o CPU è la parte centrale del dispositivo preposta allo svolgimento delle operazioni aritmetiche e logiche;*
- *la **MEMORIA PER IL PROGRAMMA** che può essere, a seconda del dispositivo,di tipo EPROM o ROM o EEPROM;*
- *la **MEMORIA PER I DATI** di tipo RAM o a volte RAM ed EPROM;*
- *Uno o più **TIMER***
- *Le **PERIFERICHE DI I/O** che sono suddivise nei seguenti tipi:*
  *-**Porte di I/O** (input/output)*
  *-A**DC** (analog digital converter),convertitori analogico digitali per l'acquisizione di segnali analogici.*
  *-**Comparatori**,che effettuano la comparazione tra due grandezze analogiche.*
  *-**I/O seriale**,che permette il collegamento seriale del microcontrollore.*

*All'interno dei microcontrollori PIC è inoltre presente un complesso sistema di **bus** rappresentante l'insieme di linee che può trasportare informazioni.*

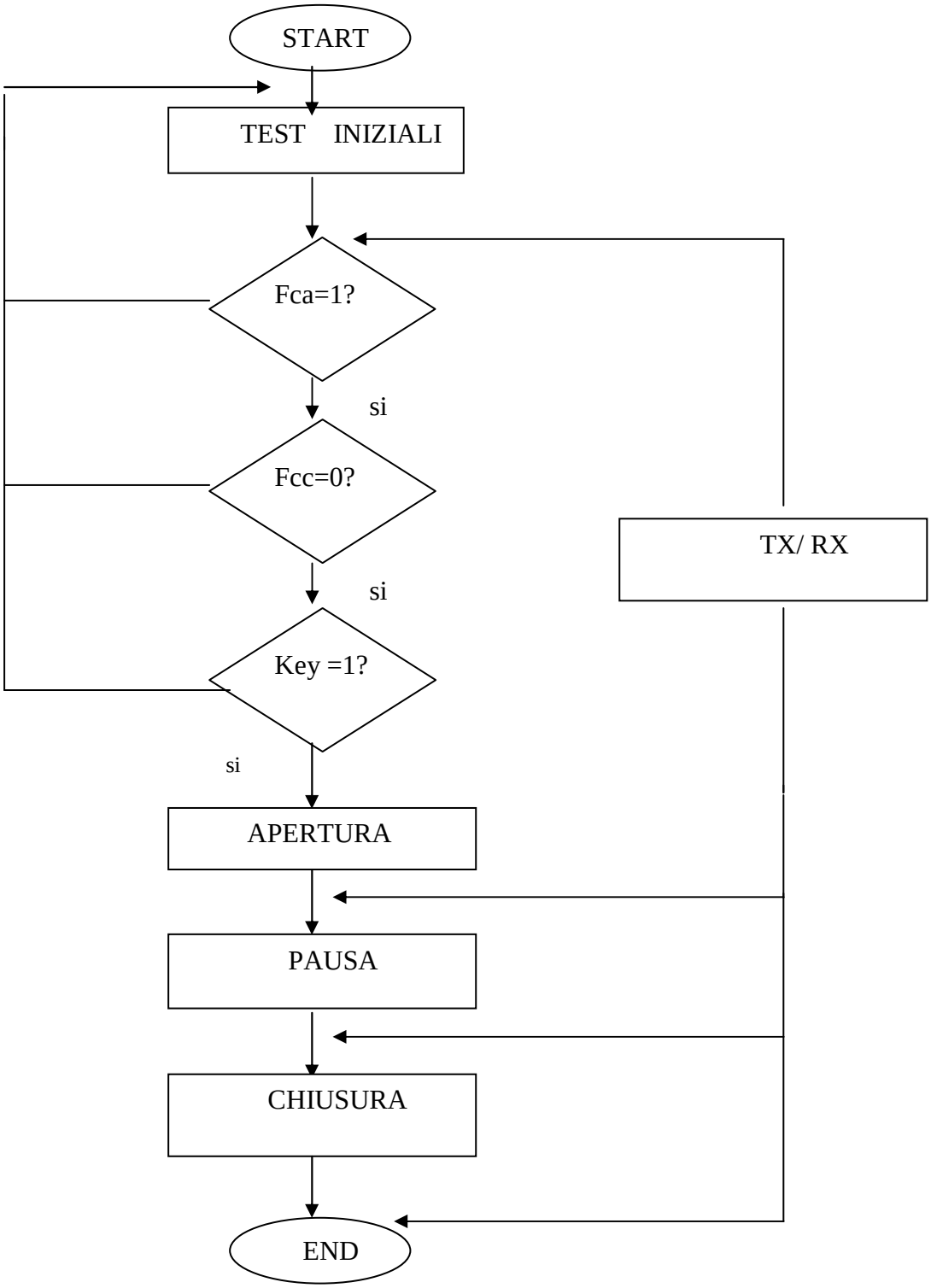
Esistono tre tipi di bus:

*-il **bus indirizzi**,che seleziona tramite una rete logica combinatoria la locazione di memoria o il dispositivo I/O desiderato.*

*-il **bus dati**,che è utilizzato per il trasferimento dei dati in tutti i blocchi del PIC.*

*-il **bus di controllo**,costituito da un insieme di segnali che coordinano e controllano il flusso dell'informazione trasportata dagli altri due bus.*

Il programma di pilotaggio del motore è stato effettuato con il programma IDE MPLAB e il PIC 16F628A. Il programma prevede quattro fasi evidenziate dal flowchart di primo livello


START
TEST INIZIALI
Fca=1?
si
Fcc=0?
si
Key =1?
si
APERTURA
PAUSA
CHIUSURA
END
TX/ RX

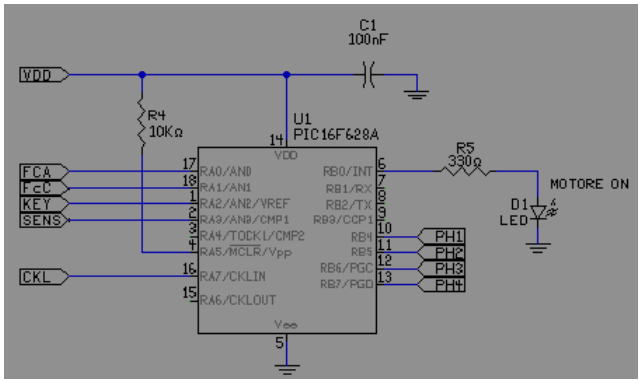
Prima di procedere con l'apertura e la chiusura il Software deve dichiarare le variabili, impostare delle configurazioni iniziali per abilitare la trasmissione e ricezione, la velocità di 57600 baud rate, impostare le porte d'ingresso ed uscita, come si può vedere nella tabella seguente.

PORTA

RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
QUARZO 20MHz	N.C	VDD	N.C	FOTO CELLU.	KEY	FC C	FCA

PORTB

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
PH 4	PH 3	PH 2	PH 1	RS485	TX	RX	LED



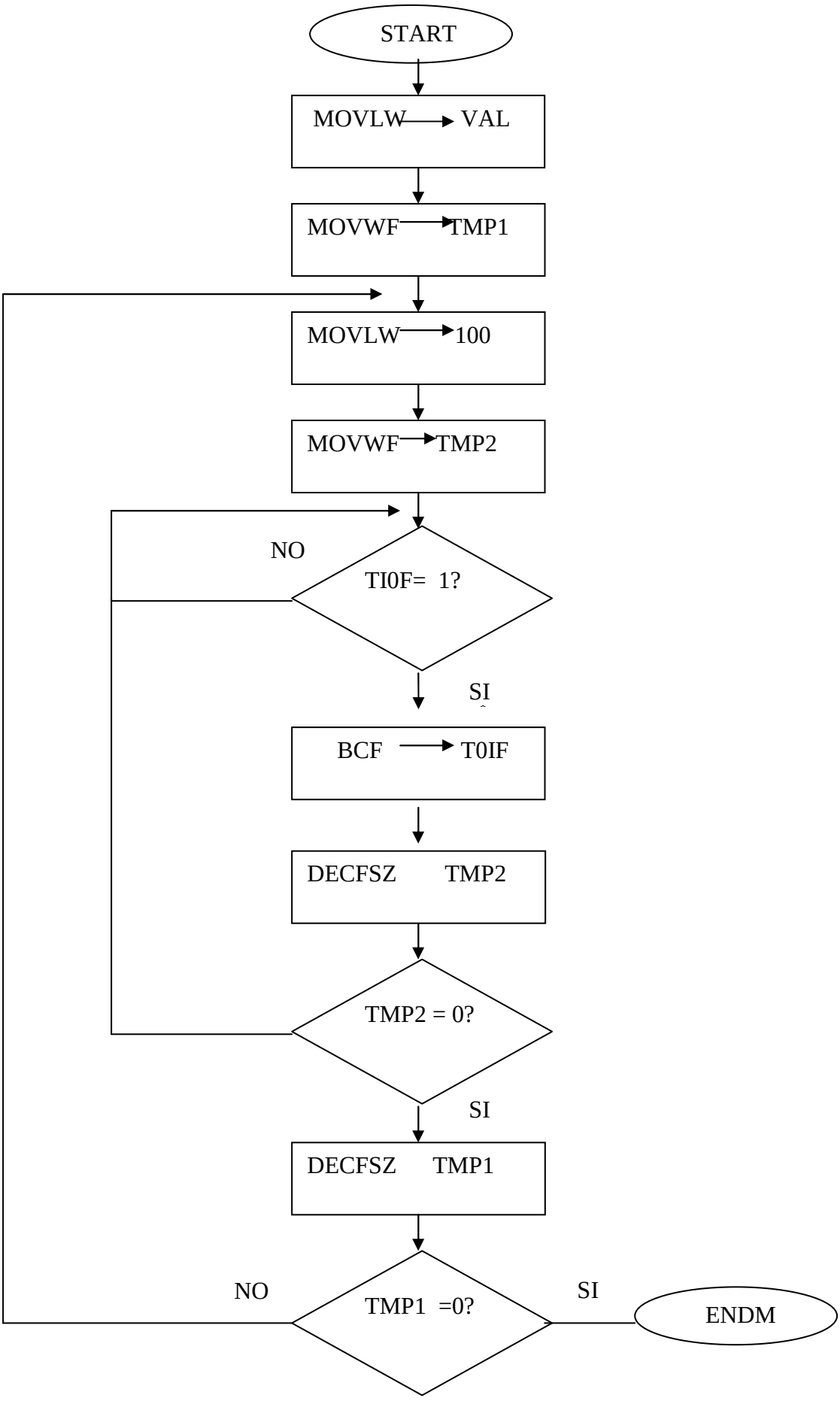
Nella pagina seguente sarà riportato il flowchart

SPIEGAZIONE GENERALE :

*Il cancello è stato realizzato per permettere l'entrata nel perimetro che circonda la casa. Come già asserito nell'introduzione il sistema è pilotato dal pic16F628A un motore passo-passo e una guida meccanica. Il motore riceve gli impulsi dal PIC, il programma testa ciclicamente il pulsante di start (chiave); il fine corsa di chiusura che deve dare uno zero logico al suo rispettivo ingresso; il fine corsa di apertura che deve fornire un uno logico. Testate queste condizioni iniziali, il PIC provvederà a fornire gli impulsi in un ordine tale da fare ruotare in maniera opportuna il motore ed aprire il cancello (**fase di apertura**). Ricordiamo che l'ordine della sequenza d'impulsi viene fornita da una procedura (tab) che riporta la tabella half step del motore passo-passo . Premuto il fine corsa di apertura esso fornirà al rispettivo ingresso uno zero logico, il PIC darà un ritardo e farà entrare, quindi, il cancello in una **fase di stop** dove rimarrà aperto per circa 3 s (tempo ipotizzato senza particolari esigenze) . Trascorsi i 3 s il PIC automaticamente fornirà una sequenza di impulsi opposta a quella utilizzata per l'apertura fornendo così la **fase di chiusura**. Durante quest'ultima fase il software provvede anche al controllo di una fotocellula, quando un ostacolo s'interpone il raggio della fotocellula s'interrompe e ciò sarà avvertito dal Pic con un impulso e immediatamente il cancello si aprirà. Conclusasi l'apertura e la chiusura del cancello, il Pic effettuerà i test iniziali fino a quando non si presenteranno nuovamente le condizioni idonee a reiniziare un nuovo ciclo di apertura e chiusura. La trasmissione e ricezione può avvenire in tempi prestabiliti e in qualsiasi parte del programma , ma questo parte verrà approfondita in un altro paragrafo.*

Bisogna aggiunger che il ritardo introdotto per realizzare la fase di stop o pausa si è utilizzata una macro. Essa esegue un ritardo di un millisecondo e lo ripete tante volte quante indicate dal parametro che gli è stato passato dal programma principale. Il parametro che viene passato alla macro è messo nella variabile locale "VAL".

Nella pagina seguente è riportato il flow chart della macro


START
MOVLW VAL
MOVWF TMP1
MOVLW 100
MOVWF TMP2
NO
TI0F= 1?
SI
BCF T0IF
DECFSZ TMP2
TMP2 = 0?
SI
DECFSZ TMP1
NO
TMP1 =0?
SI
ENDM

PROGRAMMA IN DELPHI

Delphi è un ambiente di programmazione ad oggetti per lo sviluppo rapido di applicazioni(RAD / Rapid Application Development) a carattere generale e di applicazioni client server/server per Windows 95 e 98 e Window NT. Con Delphi è possibile creare applicazioni Windows altamente efficienti riducendo al minimo i tempi di programmazione.
Delphi comprende una libreria di componenti riutilizzabili VCL e un insieme di strumenti di progettazione RAD, tra cui i modelli di applicazioni standard e di schede export di programmazione. Con questi strumenti e con il compilatore Delphi a 32 bit è possibile creare rapidamente e testare prototipi, trasformandoli in robuste applicazioni perfettamente in linea con le moderne esigenze.
Delphi può essere utilizzato per sviluppare qualsiasi tipo di applicazioni, dalle utilità di analisi e test dei PC, fino ai più sofisticati strumenti di acceso ai database. Gli strumenti di gestione dei database
Gli strumenti di gestione dei database e i componenti di gestione dei dat previsti in Delphi permettono di sviluppare strumenti di gestione dati e applicazioni client/ server in tempi notevolmente ridotti. Con i controlli di gestione dei dati , i dati vengono visualizzati direttamente durante la creazione dell' applicazione, consentendo una immediata verifica del risultato delle interrogazioni al database e delle modifiche all'interfaccia dell' applicazione.
Nell'ambiente di sviluppo integrato di Delphi IDE (Integrated, Development, Enviroment) mantiene le funzioni di sviluppo, verifica e gestione delle applicazioni in unico ambiente. E' possibile creare o modificare una applicazione compreso schede di inserimento dati, report, menù, finestre di dialogo, database e definizioni di file, moduli dati, componenti, senza uscire dal Delphi.
Delphi aumenta la produttività automatizzando le operazioni di programmazione ripetitive. E' possibile creare applicazioni semplicemente trascinando pochi componenti dalla Component Palette in una scheda chiamata Form creando l'architettura dell' applicazione velocemente e facilmente, con il minimo di programmazione.
Procedendo con la selezione e la modifica delle proprietà dei componenti e delle schede, i risultati vengono aggiornati automaticamente nel codice sorgente, e viceversa. Modificando il codice sorgente direttamente con un editor di testo, come il Code Editor incorporato, le modifiche vengono immediatamente riflesse negli strumenti visuali.

Per questioni di tempo si è voluto realizzare un programma in delphi che rileva soltanto lo stato e permette l'apertura del cancello : APERTO, CHIUSO, MOVIMENTO.
Questo stato viene fornitogli dal Pic mediante trasmissione seriale, a seconda del valore che arriva, il programma riconosce uno dei tre stati.

LISTATO DEL PROGRAMMA IN DELPHI:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, VaClasses, VaComm;

type
  TForm1 = class(TForm)
    BtApri: TButton;
    VaComm1: TVaComm;
    Timer1: TTimer;
    Edit1: TEdit;
    BtExit: TButton;
    Timer2: TTimer;
    Button1: TButton;
    procedure BtApriClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure BtExitClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  ind:integer;
```

```
implementation

{$R *.DFM}

procedure TForm1.BtApriClick(Sender: TObject);
begin
VaComm1.Open;
Timer1.Enabled:=true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
Var stato:integer;
begin
VaComm1.SetRTSState(true);
VaComm1.ReadBuf(stato,1);
case stato of
256: Edit1.Text:='Aperto';
0: Edit1.Text:='Chiuso';
1: Edit1.Text:='In movimento';
else Edit1.Text:='Error';
timer2.Enabled:=true;
end;
end;

procedure TForm1.BtExitClick(Sender: TObject);
begin
VaComm1.Close;
Close;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
VaComm1.Open;
timer2.Enabled:=true;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
ind:=8;
if ind=8 then timer1.Enabled:=true
else timer2.Enabled:=true;
end;

procedure TForm1.Button1Click(Sender: TObject);
```

```
Var apri:byte;
begin
apri:=1;
VaComm1.WriteBuf(apri,1);
VaComm1.PurgeReadWrite;
end;
end.
```

*Questo è il listato completo del programma in delphi il programma prevede tre pulsanti uno di **rileva stato**, uno di **exit** e l'altro di **apertura**. Sostanzialmente il programma è strutturato da tre procedure principali , vediamo quali:*

```
procedure BtApriClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure BtExitClick(Sender: TObject
```

" procedure TForm1.BtApriClick(Sender: TObject);
begin
VaComm1.Open;
Timer1.Enabled:=true;
end; "

con questa procedura, una volta azionato il pulsante "rileva stato" , il programma apre la seriale ed abilita il timer con l'istruzione
Timer1.Enabled:=true;

```
" procedure TForm1.Timer1Timer(Sender: TObject);
 Var stato:integer;
 begin
 VaComm1.ReadBuf(stato,1);
 case stato of
 256: Edit1.Text:='Aperto';
 0: Edit1.Text:='Chiuso';
 1: Edit1.Text:='In movimento';
 else Edit1.Text:='Error';
 end;
 end; "
```

questa seconda procedura dichiara la variabile stato come variabile integer , comanda alla seriale di leggere il buffer, dal quale arriverà una la cifra che rappresenterà lo stato del cancello. Successivamente si ha una struttura case of ovvero una struttura che a seconda del caso che si presenta darà una riposta ben precisa. Nel caso in cui dal buffer arriverà 256 nell' edit ossia la casella di testo sarà visualizaato aperto; nel caso arrivi zero sarà visualizzato chiuso; nel caso arrivi 1 si visualizzerà in movimento e in qualunque altro caso vedremo la scritta error. I dati sono rappresentati in decimale in quanto la variabile stato è stata dichiarata integer, il Pic quando il suo programma riconoscerà lo stato di aperto provvederà ad inserire il numero 0xff, appunto 256 in decimale, nella variabile chiamata anche in quel caso stato; nel caso riconosca lo sato di chiuso in base hai suoi finecorsa caricherà il dato 0x00 (ossia 0 in decimale) nella stessa varibile stato e infine caricherà 0x01 ,ovvero 1, nel caso vede che si sta provvedendo al pilotaggio del motore passo- passo. Con una particolare procedura di trasmissione s'invierà il valore presente nella variabile di stato a delphi.

```
procedure TForm1.Button1Click(Sender: TObject);
Var apri:byte;
begin
apri:=1;
VaComm1.WriteBuf(apri,1);
VaComm1.PurgeReadWrite;
end;
```

La terza procedura si occupa di mandare l'impulso nel pin della chiave è attuare così l'apertura del cancello, tutto sarà possibile naturalmente quando si rileverà lo stato "chiuso"

```
" procedure TForm1.BtExitClick(Sender: TObject);
  begin
 VaComm1.Close;
 Close;
 end; "
```

La quarta ed ultima procedura prevede il caso in cui sia attivato il tasto exit, se ciò avviene il programma chiude la seriale con l'istruzione VaComm1.Close e chiude l'intero programma.

```
type
  TForm1 = class(TForm)
    BtApri: TButton;
    VaComm1: TVaComm;
    Timer1: TTimer;
    Edit1: TEdit;
    BtExit: TButton;
```

questa parte del programma indica invece la struttura e le caratteristiche del programma.

ENGLISH VERSION

THE MICROPROCESSOR

The microprocessor is a chip with a memory which can store lists of coded instructions.
By following these instruction the same chip can do a variety of jobs.
It uses logic to decode instruction and manipulate data. The result can then be stored in a memory or sent to other devicese such as a display screens.
A microprocessor, also, consist of ALU (arithmetic logic unit) which performs calculations on numbers(addition, subtraction, multiplication)and makes logical decisions (Boolean algebra operations) and the control unit which provides timing instruction and synchronization signals for all the other units.

Elements of microprocessor

In digital processing the microprocessor is copmosed by three basic elements:

- *memory circuit;*
- *logic circuits*
- *control circuit.*

The first element store data and bits permanently or temporarily
Memory contains thousand of capacitor arranged in rows.
The capacitor hold bits either in the form of an eletric charge or the abscence of a charge.
Each capacitor is connected to the system by a metal conductor, with transistor or diodes acting as switches.

The second element change data.
In the CPU manipulate data according to instruction. The bits go through a sequence of switches that changes them in some way, for example, by adding two or them together.
During the process, bits are store temporarily in areas called registers, while they wait for the next instruction.
The third elements direct and coordinate the operations of whole system.
Organize the movement of bits throught the system. This is done by means of an oscillator, called the clock, which generated continuos pulses. Other important elements are bus connections, which carry digital information.

A bus is a set of lines, each line used for a single- bit signal, that connects several units.
When a bus used to pas signal the sending and the receiving units are enabled and other units are disabled, so that the same buse can be used for different signals in either deirection.
Modern microprocessor operate with bus widths of 64 bits, meaning that 64 bits of data can be transferred at the same time.

Memory in microprocessor

In the processor there aree two basic kinds of memory circuits:
-ROM & RAM
the first(read-only- memory) cen be neither erased nor added to. It store permanently. The second (random- acess memory)cen be added to or erased. It uses to store information for short periods. Data is stored only when the power is on. It strore temporarily.

Other Information

Microprocessor are fabricated using techniques similar to those used for other integrated circuits. They generally have a more complex structure than do other chips, and their manufacture requires extremely precise techniques. The technology of microprocessors and integrated circuits is changing rapidly.
Currently, the most sophisticated microprocessor contain about 10 million transistors , and about 800 million by 2010.
The limiting factor in microprocessor performance will almost certainly be the behavior of the electrons as they are set in action throught the transistors.
New devices and circuit designs may be necessary as microprocessors approach atomic dimensions.

CONCLUSIONI

La parte di progetto relativa al pilotaggio del cancello con l'effetto dello stato dei fine corsa di chiusura e di apertura funziona. Purtroppo la ricezione e trasmissione con il master non si è potuta verificare per scarsità di tempo a disposizione e da sopravvenute difficoltà. La trasmissione e ricezione e l'interfaccia in delphi sono state effettuate però non si è potuta verificarne l' effettiva funzionalità.
Inizialmente ci eravamo posti l'obiettivo di effettuare l' apertura del cancello anche tramite un telecomando ma per i problemi prima citati non è stato possibile.
Si sono riscontrati problemi nel software inerente alla trasmissione, per incomprensione tra i vari gruppi e anche per problemi di tipo tecnico.
Era previsto anche un sensore di posizione (fotocellula) con tecnologia ad infrarossi. Inizialmente sembrava funzionare correttamente ma dopo l'ultimo collaudo in uscita il pll(LM567) non forniva un impulso di 5 v necessario al pic.

RINGRANZAMENTI

Vorremmo ringraziare tutti coloro che hanno reso possibile la realizzazione di questo progetto, in modo particolare a colui che ha dato un grande contributo a ciò:il prof. Marco Belloni .
Un altro ringraziamento ai prof. Meroni Stefano, Barbara Costantini.
Ringranziamo infine la scuola per la fornitura dei materiali, e per il tempo extra-scolastico concessoci per il completamento anche se parziale del nostro lavoro.

ALLEGATI

LISTATO PROGRAMMA PILOTAGGIO CANCELLO:

```
;*********************************************************
;       Comando stepper motor PIC16F628A
;       controllo direzione
;       apertura con chiave (KEY)
;       sensore controllo in chiusura (SENS)
;
;       RA0--> FcA (fine corsa apertura)
;       RA1--> FcC (fine corsa chiusura)
;       RA2--> KEY (apertura manuale)
;       RB4-RB5-RB6-RB7 -->  fasi stepper motor
;
;       CLOCK esterno 20 MHz  - Frequenza step 5KHz
;
;       FEBBRAIO 2006
;
;       SAMMARTINO GABRIELE & RENATO PRIULI
;*********************************************************

        TITLE "Stepper controller"
        SUBTITLE " velocità e direzione"
        PROCESSOR 16F628A
        RADIX DEC
        INCLUDE "P16F628A.INC"
        ERRORLEVEL -302, -305
        __CONFIG 3F22H

;****   DEFINIZIONE COSTANTI  ****
DRA EQU B'11111111'
DRB EQU B'00000000'

#DEFINE LED  PORTB,1
#DEFINE FCA  PORTA,0
#DEFINE FCC  PORTA,1
#DEFINE KEY  PORTA,2
#DEFINE SENS PORTA,3
```

```
;****  DEFINIZIONE BANCHI   ****
BANK0 MACRO
        BCF STATUS,RP0
        BCF STATUS,RP1
        ENDM

BANK1 MACRO
        BSF STATUS,RP0
        BCF STATUS,RP1
        ENDM

;****  MACRO DI SERVIZIO      *******
DELAYMS MACRO VAL
       LOCAL RIP,RIP1
       MOVLW VAL
       MOVWF TMP1
RIP
       MOVLW 100
       MOVWF TMP2
RIP1
       BTFSS INTCON,T0IF
       GOTO $-1
       BCF INTCON,T0IF
       DECFSZ TMP2
       GOTO RIP1
       DECFSZ TMP1,1
       GOTO RIP
       ENDM

;****  DEFINIZIONE VARIABILI   ****
       CBLOCK  0X20
       TIME,TEMP,PTR,CONT,FASE
       TMP1,TMP2              ;variabili per macro
       ENDC

;***  VETTORE DI RESET       ****
       ORG 0X00
       GOTO MAIN

;**** VETTORE DI INTERRUPT    ****
       ORG 0X04
       NOP
       NOP
       RETFIE
```

```
;*****   MAIN PROGRAM   *****
MAIN
        CALL INIT
MAINLOOP
        BTFSC KEY
        GOTO $-1
        CALL TASK1
        CALL TASK2
        GOTO MAINLOOP

;****  ROUTINE DI SERVIZIO STEPPER    *****
TASK1
        MOVLW 0X00
        MOVWF PTR
LOOP1
        MOVFW PTR
        CALL TAB
        ANDLW B'11110001'
        MOVWF PORTB
        CALL DELAY
        INCF PTR,1
        BTFSS PTR,3
        GOTO LOOP1
        BTFSC FCC
        GOTO TASK1
        RETURN

TASK2
        DELAYMS 5
TASK22
        MOVLW 0X08
        MOVWF PTR

LOOP2
        ;BTFSS SENS
        ;RETURN
        MOVFW PTR
        CALL TAB
        ANDLW B'11110001'
        MOVWF PORTB
        CALL DELAY
        DECFSZ PTR,1
```

```
        GOTO LOOP2
        BTFSC FCA
        GOTO TASK22
        RETURN


;*****   CONFIGURAZIONE INIZIALE    **********
INIT
        BANK1
    MOVLW DRA
    MOVWF TRISA
    MOVLW DRB
    MOVWF TRISB
    MOVLW B'10000111'
    MOVWF OPTION_REG   ; PSA TMR0 1:256
    MOVLW B'00000000'
    MOVWF INTCON
    BANK0
    MOVLW B'00000111'
    MOVWF CMCON      ;DISABLE COMPARATOR
    CLRF PORTA
        CLRF PORTB
    CLRC
    CLRF TMR0

    RETURN
;*****************************************************************
;       TABELLE PER CONTROLLO STEPPER HALF-STEP
;       PTR <-- 0   per avanti
;       PTR <-- 7  per indietro
;*****************************************************************

TAB
        ADDWF PCL,F
        RETLW B'10000001'
        RETLW B'11000001'
        RETLW B'01000001'
        RETLW B'01100001'
        RETLW B'00100000'
        RETLW B'00110000'
        RETLW B'00010000'
        RETLW B'10010000'
        RETLW B'10000000'
```

```
DELAY
        MOVLW 10
        MOVWF TEMP
LOOP                        ;circa 10 ms
        MOVLW 200
        MOVWF TIME
        DECFSZ TIME,F
        GOTO $-1
        DECFSZ TEMP,F
        GOTO LOOP
        RETURN

        END
```

LISTATO PROGRAMMA PILOTAGGIO CANCELLO E TRASMISSIONE E RICEZIONE *:*

```
;*********************************************************
;       Comando stepper motor PIC16F628A
;       controllo direzione
;       apertura con chiave (KEY)
;       sensore controllo in chiusura (SENS)
;
;       RA0--> FcA (fine corsa apertura)
;       RA1--> FcC (fine corsa chiusura)
;       RA2--> KEY (apertura manuale)
;       RB4-RB5-RB6-RB7 -->fasi stepper motor
;       RB1--> Ricezione rx
;       RB2-->Trasmissione tx
;       RB3-->Abilitazione RS485
;      - Frequenza step 5KHz
;
;       FEBBRAIO 2006
;
;       Prof. SAMMARTINO PRIULI
;*********************************************************

        TITLE "Stepper controller"
        SUBTITLE " velocità e direzione"
        PROCESSOR 16F628A
        RADIX DEC
        INCLUDE "P16F628A.INC"
        ERRORLEVEL -302
        __CONFIG 3F70H

;****   DEFINIZIONE COSTANTI  ****
DRA EQU B'11111111'
DRB EQU B'00000110'

#DEFINE RS485 PORTB,3
#DEFINE LED  PORTB,1
#DEFINE FCA  PORTA,0
#DEFINE FCC  PORTA,1
#DEFINE KEY  PORTA,2
#DEFINE SENS PORTA,3
```

```
;****  DEFINIZIONE BANCHI   ****
BANK0 MACRO
        BCF STATUS,RP0
        BCF STATUS,RP1
        ENDM

BANK1 MACRO
        BSF STATUS,RP0
        BCF STATUS,RP1
        ENDM

;****  DEFINIZIONE VARIABILI   ****
       CBLOCK  0X20
       TIME,TEMP,PTR,CONT,FASE
   TIME1,TIME2,TIME3,STEP,VAL
       STATO
       RX_DATA , TX_DATA,INDIRIZZOMIO, BETA
       NANO
       ENDC

;****  DEFINIZIONE MARCRO     ****
DELAYMS MACRO VAL
       LOCAL RIP, RIP_O
       MOVLW VAL
       MOVWF TIME3
RIP
       MOVLW 0X06
       MOVWF TIME2
RIP_0
       MOVLW 0XFF
       MOVWF TIME1
       DECFSZ TIME1,1
       GOTO $-1
       DECFSZ TIME2,1
       GOTO RIP_0
       DECFSZ TIME3,1
       GOTO RIP
       ENDM

;***  VETTORE DI RESET       ****
       ORG 0X00
       GOTO MAIN
```

```
;**** VETTORE DI INTERRUPT    ****
        ORG 0X04
        NOP
        NOP
        RETFIE

;*****   MAIN PROGRAM   *****
MAIN
        BSF BETA,0
        CALL INIT
MAINLOOP

        MOVLW 0X00
        MOVWF STATO
        MOVF TX_DATA        ; STATO CHIUSO
        BTFSS PIR1,RCIF     ;VEDERE SE MI KIEDE DI COMUNICARE
        GOTO KEYT
        CALL RX_00
KEYT
        BTFSS KEY
        GOTO $-1
TEST_A
        BTFSS FCA               ;TERMINARE
        CALL PAUSA
        BTFSS FCC
        CALL TASK2          ;APERTURA
        BTFSS PIR1,RCIF
        GOTO TEST_A
        CALL RX_00
TASK1
        MOVLW 0XFF
        MOVWF STATO                 ;STATO "APERTO"
        MOVF TX_DATA
        BTFSC PIR1,RCIF
        CALL RX_00          ; CHIEDE TX?
        MOVLW 0X00
        MOVWF PTR
LOOP1
        MOVLW 0X01
        MOVWF STATO                     ; STATO" MOVIMENTO"
        MOVF TX_DATA
        MOVFW PTR
        CALL TAB
        ANDLW B'11110001'
```

```
        MOVWF PORTB
        CALL DELAY
        BTFSC PORTA,3          ; CONTROLLA FOTOCELLULA
        CALL TASK2
        INCF PTR,1
        BTFSS PTR,3
        GOTO LOOP1
        GOTO MAINLOOP

PAUSA
        MOVLW 57
        MOVWF STEP
        BSF LED
RIP
        DELAYMS 55                    ; RICHIAMA MACRO 60 ms
        DECFSZ STEP,1
        GOTO RIP
        CALL TASK1                   ; CHIUSURA



TASK2
        MOVLW 0XFF
        MOVWF STATO                   ; STATO "APERTO"
        MOVF TX_DATA
        MOVLW 0X08                    ; MUOVE IL VALORE 0X08
NELL'ACCUMULATORE
        MOVWF PTR
LOOP2
        MOVLW 0XFF
        MOVWF STATO                   ; STATO "APERTO"
        MOVF TX_DATA
        MOVFW PTR
        CALL TAB                    ; RICHIAMA LA PROCEDURA TAB
        ANDLW B'11110001'
        MOVWF PORTB                   ; TRASMETTE AL MOTORE GLI
IMPULSI
        CALL DELAY
        DECFSZ PTR,1
        GOTO LOOP2
        RETURN
```

```
;*****  CONFIGURAZIONE INIZIALE   **********
INIT
        BANK1
   MOVLW DRA
   MOVWF TRISA                  ; IMPOSTA I PIEDINI DEL PORTA IN I/O
   MOVLW DRB
   MOVWF TRISB                  ;IMPOSTA I  PIEDINI DEL PORTB IN I/O
   MOVLW B'00000000'
   MOVWF OPTION_REG             ; PSA TMR0 1:2
   MOVLW B'00000000'
   MOVWF INTCON
       MOVLW 21               ; 57600 BAUD
       MOVWF SPBRG
       BSF TXSTA,BRGH           ;SPEED E TABELLA BAUD RATE
       BSF TXSTA,TXEN           ;ABILTIA TX
   BANK0
   MOVLW B'00000111'
   MOVWF CMCON                  ;DISABLE COMPARATOR
   CLRF PORTA
   CLRF PORTB
   CLRC
   CLRF TMR0
       BSF RCSTA,SPEN           ;ABILITA SERIALE
       BSF RCSTA,CREN           ;ABILITA RICEZIONE
   RETURN


;*****************************************************************************
***
; CHIAMATA SUBROURTINE DI TX_DATA
;*****************************************************************************
***

TX_00
       BSF PIR1, TXIF
       BSF PORTB,0             ;DISABILITO 485
       MOVWF TX_DATA
       MOVFW TXREG
       BCF PORTB,0             ;ABILITO 485
       RETURN
;*****************************************************************************
***
; SUBROUTINE  RX DATI
```

```
;*****************************************************************
***
RX_00
        MOVF RCREG,W
        MOVWF RX_DATA
        MOVLW B'00001000'
        MOVWF INDIRIZZOMIO
        BTFSC RX_DATA, 7    ; CONTROLLA BIT 7 = 1? DIR RXDATA
        GOTO EMME
        MOVF RX_DATA, W
        ANDLW B'00001111'
        SUBWF INDIRIZZOMIO
        BTFSS STATUS,Z
        GOTO CLEAR
        BSF BETA,0
        GOTO EMME
CLEAR
        BCF BETA ,0
        GOTO EMME
EMME
        BTFSS BETA,0
        RETURN
        MOVLW B'01110000'
        MOVWF NANO
        MOVF RX_DATA, W
        ANDLW B'01110000'
        SUBWF NANO              ;CONTROLLA SE DEVO RX 111 O TX 000
        BTFSC STATUS ,Z
        CALL RX
        CALL TX_00

RX
        BTFSS RX_DATA,7         ;
        RETURN
        BSF PORTA ,2            ;SETTO LA CHIAVE
        GOTO KEYT
```

```
;*****************************************************************
;       TABELLE PER CONTROLLO STEPPER HALF-STEP
;       PTR <-- 0   per avanti
;       PTR <-- 7  per indietro
;*****************************************************************

TAB
        ADDWF PCL,F
        RETLW B'10000001'
        RETLW B'11000001'
        RETLW B'01000001'
        RETLW B'01100001'
        RETLW B'00100000'
        RETLW B'00110000'
        RETLW B'00010000'
        RETLW B'10010000'
        RETLW B'10000000'



DELAY
        MOVLW 3
        MOVWF TEMP
LOOP                                    ;circa 10 ms
        MOVLW 200
        MOVWF TIME
        DECFSZ TIME,F
        GOTO $-1
        DECFSZ TEMP,F
        GOTO LOOP
        RETURN

        END
```