Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

ANNEXOS

Desenvolupament d'un sistema de transferència de software per una placa entrenadora

Guillem Soler i Franquesa

Enginyeria Tècnica Industrial especialitat Electrònica Industrial
Director: Jordi Serra i Serra
Vic, juny de 2008

ÍNDEX

1. ANNEX 1

1.1 Esquema de la placa entrenadora

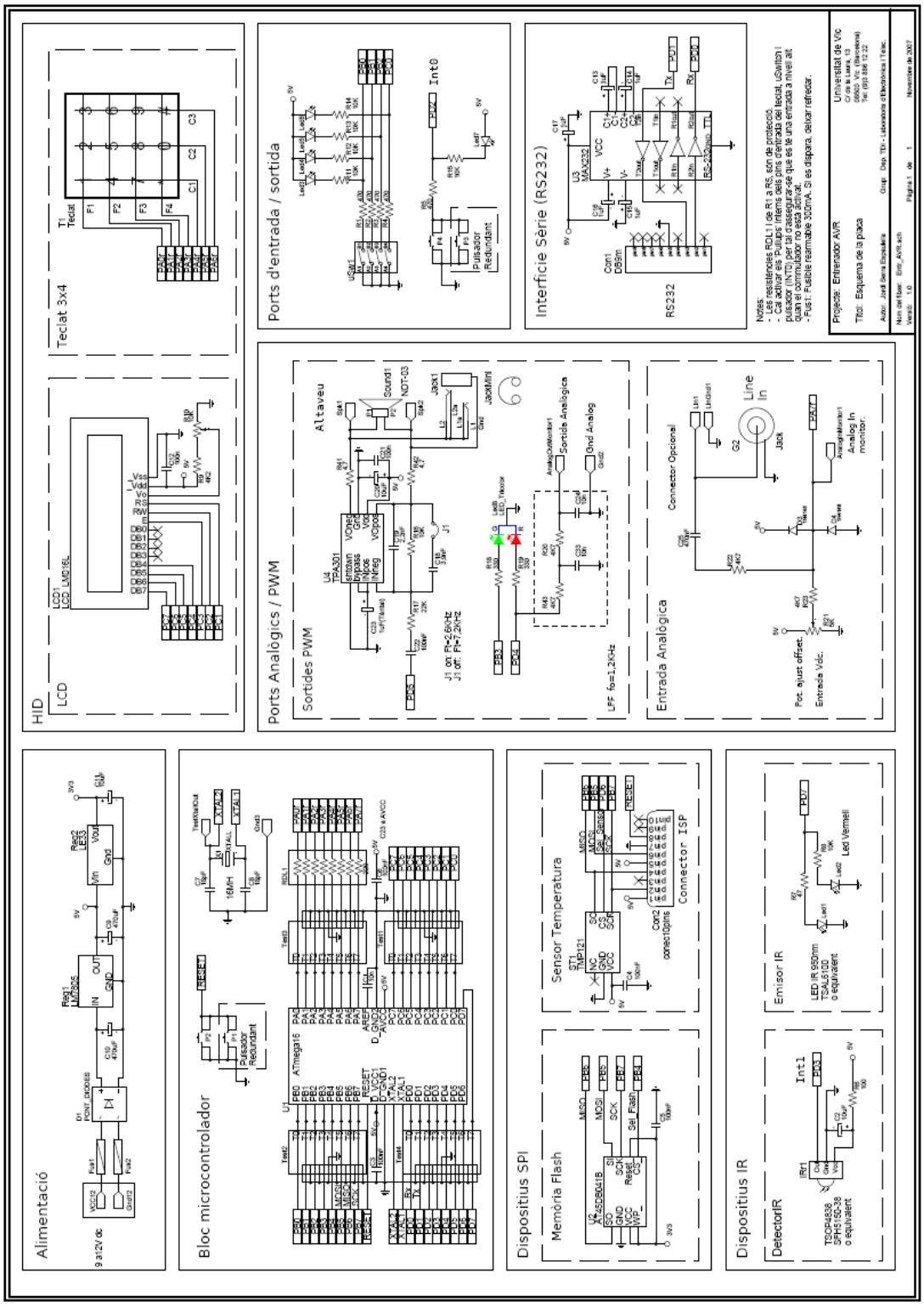


Fig. A.1: Esquema de la placa entrenadora

1.2 Llistat de programes d'aprenentatge

En aquest apartat de l'annex descriurem la llista de programes que s'han dissenyat per aprendre a manipular el llenguatge i les funcions que s'utilitzen tant en el programador com en el bootloader. Dividirem aquest apartat en dos per diferenciar els programes fets pel programador i els programes fets pel bootloader.

1.2.1 Programes pel desenvolupament del programador

Llistat de programes

- Escriure una frase per pantalla
- Obrir un arxiu de text, treure el contingut per pantalla, tancar l'arxiu
- Obrir un arxiu de text, treure el contingut per pantalla,tancar l'arxiu (funcions)
- Enviar pel port sèrie de la computadora una cadena de caràcters
- Rebre pel port sèrie de la computadora una cadena de caràcters
- Enviar pel port sèrie de la computadora el contingut d'un fitxer
- Rebre pel port sèrie de la computadora el contingut d'un fitxer
- Enviar pel port sèrie de la computadora el contingut d'un fitxer (funcions)
- Rebre pel port sèrie de la computadora el contingut d'un fitxer (funcions)
- Enviar un caràcter pel port sèrie i rebran un altre.
- Creació del programador per parts:
    - Obrir el fitxer hex i obtenir-ne el contingut de cada línea.
    - Anàlisi del contingut de l'arxiu hex per crear trames
    - Disseny i creació d'una trama.
    - Implementació del CRC
    - Preparar la trama per ser enviada pel port sèrie
    - Comprovació de rebre el caràcter ACK del micro.
    - Disseny d'un sistema de comunicació amb el micro
    - Incorporà la interfície gràfica.

1.2.2 Programes pel desenvolupament del bootloader

Alguns dels programes d'aprenentatge per desenvolupar el bootloader s'han fet amb els dos llenguatges per aprendre millor l'arquitectura i el funcionament del microcontrolador.

Llistat de programes

Fets amb llenguatge assemblador i en C:

- Activar X pins d'un port (Encendre un led per exemple)
- Activar X pins d'un port si rebem senyal d'un pin d'entrada(utilitzant un polsador)
- Interrupció externa 1
- Enviar un caràcter per sèrie per flag

- Enviar un caràcter per sèrie per interrupció
- Rebre un caràcter per sèrie per flag
- Rebre un caràcter per sèrie per interrupció
- Escriure i llegir a la EEPROM
- Escriure a la flaix

Fets només amb C

- Rebre un caràcter per sèrie, sumar-li un i retornar-lo
- Rebre una direcció pel sèrie des del PC, consulta el valor d'aquella direcció a la flash i enviar-lo pel sèrie
- Rebre una trama de valors, tractar-la i retornar-la
- Rebre una trama de valors, aplicar el CRC a la trama, comprovar el CRC i retornar la trama.
- Creació del bootloader simple
- Escriure el codi a partir d'una secció en concret de la flash
- Disseny d'un sistema d'accés al bootloader o a l'aplicació situada a l'adreça 0 depenent d'un valor enviat per el PC
- Escriure un valor a una direcció de la flash, consultar-lo i enviar-lo pel port sèrie

1.3 Esquema interfície RS-232

La interfície RS-232 és la que s'utilitza al port sèrie. El connector d'aquesta interfície pot ser de tipus DB-25 (25 pins) o bé DB-9 (9 pins) [1]. El DB-9 és una versió reduïda del DB-25 que conté només els pins més utilitzats d'aquest. Cada pin pot ser d'entrada o de sortida i té una funció determinada:

DB-25

PIN	Nom	E/S	Descripció
1	CG	-	Terra de la carcassa
2	TxD	S	Pin per on es transmet la informació
3	RxD	E	Pin per on es rep la informació
4	RTS	S	Es rep senyal quan es demana una nova dada
5	CTS	E	El receptor està acceptant les dades enviades
6	DSR	E	S'ha establer connexió
7	GND	-	Connexió a terra
8	DCD	E	Detector de transmissió
9		E	Test de voltatge positiu
10		E	Test de voltatge negatiu
11		-	Sense ús
12	SCDC	E	DCD secundari
13	SCTS	E	CTS secundari
14	SBA 118	S	TxD secundari
15	TC	E	Rellotge de transmissió
16	SRD	E	RxD secundari
17	RC	E	Rellotge de recepció
18		-	Sense ús
19	SRTS	S	RTS secundari
20	DTR	S	Receptor preparat
21	SQ	E	Qualitat del senyal
22	RI	E	Indicador de senyal
23	DSR	S	Selector de DataRate
24		S	Rellotge de transmissió extern
25		S	Entra senyal quan el receptor està ocupat

Taula A.3.1: Llistat de pins d'un connector DB-25

[1] La figura A.3.2 i A.3.4 han set tretes de

EMILIO TOBOSO
http://perso.wanadoo.es/pictob
/comserie.htm [Maig de 2008]

Hardware:

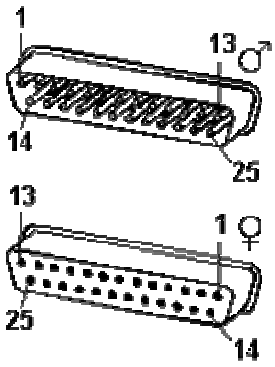


Fig. A.3.2: Numeració de pins d'un connector DB-25

DB-9

PIN	Nom	E/S	Descripció
1	DCD	E	Detector de transmissió
2	RxD	E	Pin per on es rep la informació
3	TxD	S	Pin per on es transmet la informació
4	DTR	S	Receptor preparat
5	GND	-	Connectat a terra
6	DSR	E	S'ha establer connexió
7	RTS	S	Es rep senyal quan es demana una nova dada
8	CTS	E	El receptor està acceptant les dades enviades
9	RI	E	Indicador de senyal

Taula A.3.3: Llistat de pins d'un connector DB-9

Hardware:

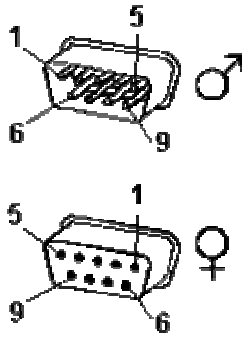


Fig. A.3.4: Numeració de pins d'un connector DB-25

1.4 Algorismes dissenyats

En aquest apartat mostrarem el conjunt d'algorismes que es van dissenyar pel programador i pel bootloader. Es van fer més algorismes pel programador que no pas pel bootloader perquè el bootloader depenia del programador.

Primer algorisme

- Programador

```
Programa principal{
        Obrir Port();
        Configurar Port();
        Obrir_arxiu();
        Comprovar_primera_fila();
        Tractar();
        Tancar fitxer();
        Tancar port();
}


Tractar{
Mentre no fi de fitxer{
        Obtenir_fila();
        Si final = 0 {
                Enviar_direccio(direccio);
                Num_pag_antic = div(direccio,0x80);
                Per i = 0 fins i=Num_bytes{
                        Num_pag_actual =div((direccio+i),0x80);
                        Si Num_pag_actual != Num_pag_antic{
                                Comanda_escriure_pagina;
                                Enviar direccio(direccio+i);
                                Num_pag_actual = Num_pag_antic;
                        }
                        enviar_byte(Bytei(i));
                Fiper
        }
        Si final = 1{
                Enviar_final_de_programa();
        }
Fimentre
}
```

La funció “Comprovar_primera_ultima_fila” mira si la primera línea de l’arxiu “hex” és vàlida i conté codi de la memòria de programa. La funció “Tractar” analitza l’arxiu hex i n’envia el contingut:

- Bootlaoder

```
Programa principal{
        Inicialitzacio i configuració del port();
        Tractar();
}

Tractar{
        i=0;
        Mentre(Dada != 0x04){
                Dada = rebre_sèrie();
                Si dada = 0x02 {
                        Rebut();
                        Byte = rebre_sèrie();
                        Buffer_flash(Byte,Direccio+i);
                        i = i+1
                        Rebut();
                Si dada = 0x05{
                        i=0;
                        Rebut();
                        DireccioL = rebre_sèrie();
                        Rebut();
                        DireccioH = rebre_sèrie();
                        Direccio=DireccioH:DireccioL;
                        Rebut();
                        }
                Si dada = 0x03{
                        Escriure_buffer_flash(Direccio);
                        Rebut();
                        }
        }
}
```

Segon algorisme

- Programador

```
Programa Principal{
        Obrir Port();
        Configurar Port();
        Obrir_arxiu();
```

```
        Comprovar_primera_fila();
        Tractar();
        Tancar fitxer();
        Tancar port();
}

Tractar{
        Obtenir_fila();
        Num_pag_antic = div(direccio,0x80);
        Trama[0]=0x02;
        Crc update(Trama[0]);
        Trama[2]=direccio;
        Crc update(Trama[2]);
        Mentre no fi de fitxer{
                Per i = 0 fins i=Num_bytes{
                        Num_pag_actual =div((direccio+i),0x80);
                        Si Num_pag_actual != Num_pag_antic{
                                Enviar_trama();
                                Crc=0;
                                t=0;
                        }
                                Trama[4+t]=Byte[i];
                                Crc update(Byte[i]);
                                t=t+1;
                                i=i+1;
                fi per
                i=0;
                Obtenir_fila();
                Num_pag_actual =div((direccio+i),0x80);
                Si Num_pag_actual != Num_pag_antic{
                        Enviar_trama();
                        Num_pag_antic=num_pag_actual;
                        Inicialitzar_Trama;
                        Crc=0;
                        t=0;
                }
                Enviar_trama{
                Trama[1]=[t+5];
                Crc update(Trama[1]);
                Enviar_trama_escriure_pagina;
        }
```

Tercer algorisme

- Programador

Programa Principal{

```
    Obrir Port();
    Configurar Port();
    Obrir_arxiu();
    Comprobar_primera_fila();
    Tractar();
    Tancar fitxer();
    Tancar port();
}
    Tractar{
        c = 0;
        Mentre  Llegir_fila(fila) =! NULL{
            Obtenir_parts(fila);
            i=0;
            Per i = 0 fins Num_bytes{
                Si Canvi_pagina(direccio_abans,direccio);{
                    Preparar_trama(Trama,k);
                    Gravar_Trama();
                    Sinó{
                        Si !(seguit(direccio_abans,direccio+i)){
                        Preparar_trama(Trama,k);
                    }
                    Trama[k+4]=Byte[i]
                    Direccio_abans = direccio+i;
            }
        }

    }

    Preparar_trama ( Trama, k ){
        Arreglar_trama(Trama,k);
        Mentre (c<3){
            Enviar_trama(Trama);
            Si Control_errors = fals{
                c=c+1;
            }Sinó{
                c =4;
        }
        Si c = 3{
            Escriure ("Error de CRC en el tercer intent")
            sortir();
        }
    }
```

La funció "Arreglar trama" afegeix a la trama l'operador, la longitud dels bytes de dades, la direcció i fa el càlcul del CRC per afegir-lo també al final de la trama.
La funció "Enviar trama" envia la trama i n'espera el caràcter de comprovació. Si el microcontrolador ha rebut la trama i operat correctament retorna el caràcter ACK.

En aquest cas la variable booleana “Control_errors” prendrà el valor cert, i en cas contrari el valor fals. Si rep el valor fals s’enviarà la trama fins a 3 cops i si en el tercer intent no hi ha èxit s’informarà de l’error i es finalitzarà l’aplicació.

1.5 Sistemes de control d’errors

Aquests són alguns sistemes de control d’errors que es van consultar a part del de CRC:

Paritat simple:

Com s’ha explicat en l’apartat anterior, la paritat es fa servir normalment en les comunicacions asíncrones i es pot automatitzar en la majoria de sistemes. Consisteix en situar, entre l’últim bit de dades i el primer bit de parada, un bit que pren un valor depenent de la quantitat de bits de dades amb valor 1 lògic.

La paritat es pot configurar de quatre maneres diferents:

- Parell: El bit de paritat pren el valor 1 si els bits de dades contenen un número parell de 1 o 0 si contenen un número imparell de 1.
- Imparell: El bit de paritat pren el valor 1 si els bits de dades contenen un número imparell de 1 o 0 si contenen un número parell de 1.
- Mark: El bit de paritat sempre pren el valor 1.
- Space: El bit de paritat sempre pren el valor 0.

Per concretar el valor del bit de paritat el sistema utilitza l’equació:

$$bp = bd1 + bd2 + ... bdn$$

on bp és el bit de paritat, bdx són els bits de dades i + és l’operació OR booleana que calcula el valor en funció de:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0$$

Un cop s’obté el bit de paritat, només cal que el sistema el negui per obtenir la paritat imparell . Per tant, la diferència entre el mecanisme de la paritat parell i la imparell sols és la negació final del bit de paritat.

Tot i que el microcontrolador permet incorporar, configurar i revisar el bit de paritat no l’utilitzaré perquè es farà servir un mètode de control d’errors més efectiu. El bit de paritat fa un control per cada dada enviada però té l’inconvenient de que hi hagi una alta probabilitat de que s’escapin casos, com per exemple, que canviïn dos bits de la mateixa transmissió.

Paritat Creuada:

La paritat creuada és una millora de la paritat simple que es basa amb el mateix sistema però incorporant finalment una última dada a enviar que correspon al bit de paritat dels bits de les mateixes posicions de cada dada.

Dada	Bit de paritat
1010	0
1110	1
0010	1
1111	0

I al final s'envia una última dada, cada bit és el bit de paritat de la corresponent columna

1001 0

Tot i aquesta millor que permet filtrar els casos de dos errors en una mateixa dada, no deixa de ser un mètode amb poca eficiència respecta d'altres.

Suma de comprovació

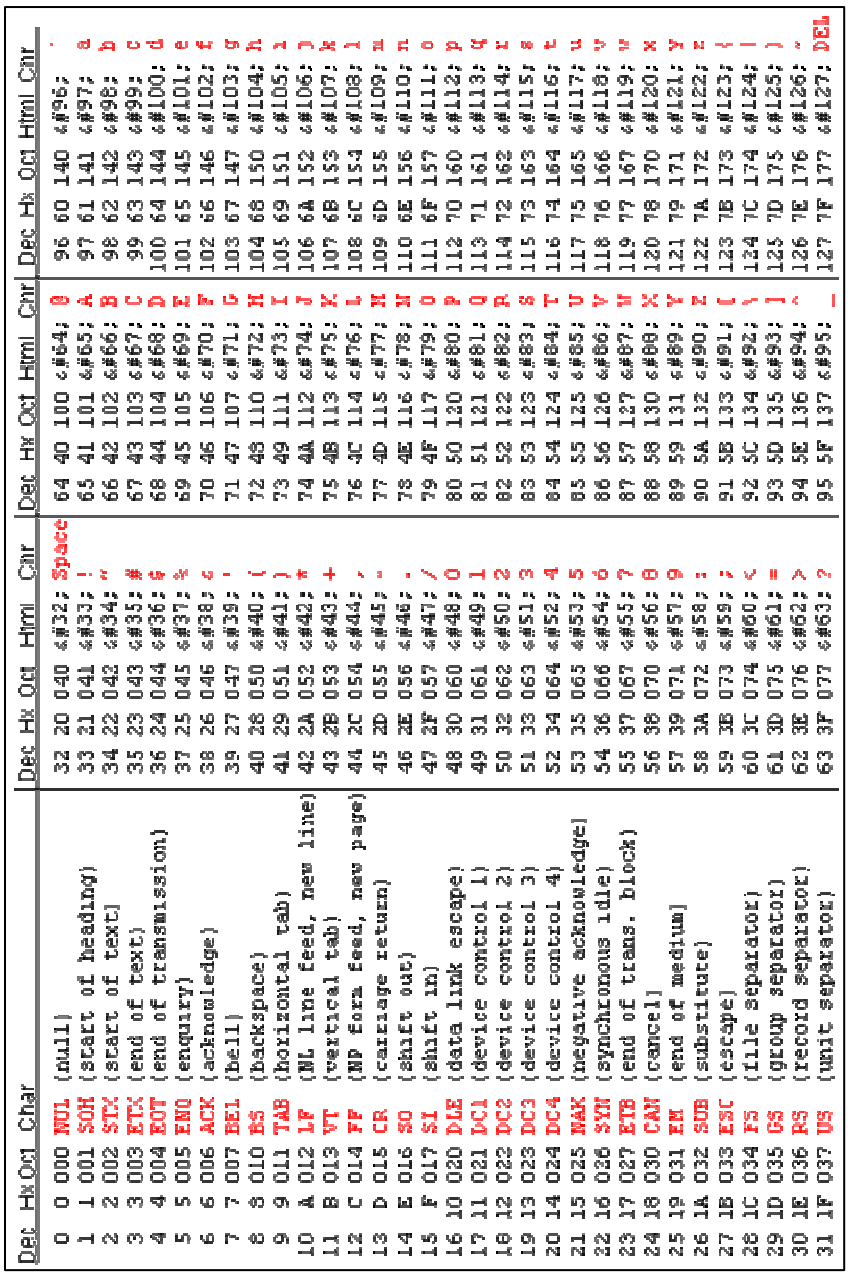
Aquest mètode tracta d'afegir al final d'una trama de dades el valor que sumat amb la resta de dades doni com a resultat un 0. Per tant, aquest valor haurà de ser negatiu.
Exemple:

Dades a enviar : 5 6 7 8 9
Valor de comprovació d'errors: -(5+6+7+8+9) = -35
Trama a enviar: 5 6 7 8 9 -35

Aquest mètode no podria funcionar en aquesta aplicació perquè es poden arribar a enviar trames de 130 bytes i el número de comprovació hauria de prendre un valor tan gran que ens ocuparia molts bytes. És un mètode que únicament serveix per trames de bits curtes, de 16 bits per exemple

1.6 Taula de caràcters ASCII

Aquesta taula ASCII és la que s'ha consultat durant el projecte[2]



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Fig. A.5: Taula ASCII consultada

[2] Taula obtinguda de

OSCAR MARTINEZ
http://personal.telefonica.terra.es/web/oscarmartinez
/_imatges/ascii.gif [Febrer de 2008]

1.7 Estudi del temps de transferència i gravació d'una pàgina

Aquest estudi calcula el temps aproximat de transferència i gravació d'una pàgina de la memòria de programa del microcontrolador. Està fet per tal de tenir una noció del temps que tardarà el sistema de gravació a finalitzar.

Aquest temps serà igual a:

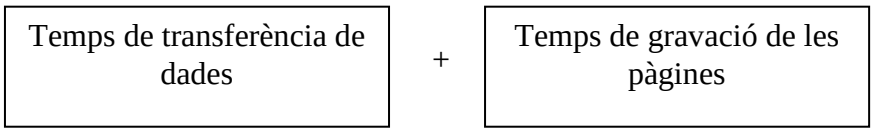


Fig. A.6.1: Temps del sistema de programació

La memòria del microcontrolador per emmagatzemar-hi programa serà de 15360 bytes, que equival a 153600 bits, agrupats en 120 pàgines (blocs de 128 bytes).

Calculem els bits que s'han de transferir per guardar una pàgina suposant que:

- No hi haurà un error de comprovació d'error
- La pàgines està completa i s'envia una sola trama de dades

Conjunt de bits	**Descripció**
1340 bits d'una trama que equival a la pàgina	Bytes de dades que es guardaran a la memòria de programa i altres bytes (operador,direcció incial, número de bytes i CRC).
10 bits per l'ordre d'escriure pàgina	Byte que s'utilitza per ordenar que s'escrigui una pàgina.
20 bits de ACK o NACK	Bytes de comprovació d'errors. Són la resposta que el microcontrolador envia a l'ordinador després de rebre una ordre (Per cada pàgina són dos ordres: la de rebre les dades i la de gravar la pàgina).
Total: 1370 bits a transferir per una pàgina	

Taula A.6.2: Bits transferits en la descàrrega d'una pàgina

Per saber els segons que trigarem a fer la transferència només cal dividir aquesta quantitat per el Baud Rate que es vol utilitzar. Per exemple:

Baud Rate (bps)	**Temps de transferència estimat (ms)**
9600 (baix)	142,708
57600 (mig)	23,784
115200 (alt)	11,892

Taula A.6.3: Temps de trasnferència

A Aquest temps se li ha de sumar el temps que tarda el microcontrolador a esborrar i gravar la pàgina a la seva memòria de programa. Consultant el manual del microcontrolador trobem que el temps d'una d'aquestes operacions és:

Temps mínim de gravació	**Temps màxim de gravació**
3.7 ms	4.5 ms

Taula A.6.4: Temps de gravació

Com que abans d'escriure una pàgina l'hem de esborrar, fem dues operacions que consumiran dues vegades aquest temps

Temps mínim esborrar i gravar	**Temps màxim esborrar i gravar**
7.4 ms	9.0 ms

Taula A.6.5: Temps de total de gravació

Per tant obtindrem que en funció del Baud Rate la fòrmula per calcular el temps total és:

	Temps mínim (seg)	Temps màxim (seg)
1 pàgina	1370/BR(bps)+7.4·10-3	1370/BR(bps)+4.5·10-3
120 pagines (tota la memòria)	164400/BR(bps)+0,888	164400/BR(bps)+1,08

Taula A.6.6: Temps de total de transferència i gravació en funció del Baud Rate

Substituint la variable BR pel valor 57600, que és el Baud Rate que farem servir:

	Temps mínim	Temps màxim
1 pàgina	27.48+7,4·10-3 ms	28.28+9·10-3 ms
120 pagines (tota la memòria)	2.85+0,88 segons	2.85+1.08 segons

Temps total estimat màxim per la transferència i gravació d'una pàgina.	**35,68 +- 1,4 milisegons**
Temps total estimat màxim per la transfarència i gravació de totes les 120 pàgines (ms)	**3,83 +- 1 segons**

Taula A.6.7: Temps de total de transferència i gravació a 57600 bps

Podem observar que:

Temps gravació = 30.8% - 37.9% Temps de transferència.

Aquest temps serà aproximat ja que també s'hi hauria d'incloure el temps de la resta de codi (inicialització, carregar dades de l'arxiu hexadecimal,...) del programador i del bootloader.

1.8 Codi de programa del bootloader1

```
#include <util/crc16.h>
#include <avr/boot.h>

// INICIALITZACIÓ
void Iniciali(void)
{

	UBRRL = 15; //Baud Rate a 57600
	UCSRB = (1<<TXEN | 1<<RXEN); //Habilitar recepció i enviament sèrie
	UCSRC = (1 << URSEL | 1 << UCSZ0 | 1 << UCSZ1); // 8 bits de dades (011)
}

// FUNCIO UNIFICAR 2 BYTES EN 1 INT
uint16_t Unio(uint8_t baix,uint8_t alt){ //Unir dos bytes

		int unit = 0;
		unit = alt;
		unit = unit<<8;
		unit = unit | baix;
		return unit;
}

//FUNCIÓ REBRE DADA
char Rebre(){

	while(!(UCSRA & (1 << RXC)));  //Esperar que arribi dada nova
			return UDR; // retornem contingut del buffer serie
}

//FUNCIÓ ENVIAR DADA
void Enviar(char dada){

		while(!(UCSRA & (1 << UDRE))); //Esperar que estigui buit el buffer
			UDR=dada;    //Enviem dada
}

//FUNCIÓ PER COMPROBAR SI BOOTLOADER O APLICACIÓ
unsigned char Comprobar(){

	DDRD = 0xFB; //Configuració pin 2 com a sortida
	if (bit_is_set(PIND, 2)) { //si el polsador està activat
		DDRD = 0x00; //Reconfigurem port 2
		return 0;
	}
	DDRD = 0x00; //Reconfigurem port 2
```

```
	return 1;
}

//PROGRAMA PRINCIPAL
int main(void) //PROGRAMA PRINCIPAL
{

	//Inicialització de variables
	 uint16_t crc = 0;
	uint8_t i = 0;
	uint8_t operador = 0;
	uint8_t longitud = 0;
	uint8_t dirh = 0;
	uint8_t dirl = 0;
	uint16_t direccio = 0;
	uint8_t crch = 0;
	uint8_t crcl = 0;
	uint16_t crcr = 0;
	uint16_t Paraula = 0;

	Iniciali();

if (!Comprobar()){ //SI NO ES TROBA EL PROGRAMADOR...
		boot_rww_enable();
		goto *0x0000; //Saltar adreça aplicació
}

DDRB = ( 1<< DDB3); //Configuració led tricolor
DDRD = (1 << DDD4);
PORTB = (1 << 3); //Color verd


//ENVIEM CARÀCTER ACK.

while(1){

PORTD = (0 << 4); //Apaguem vermell
	operador = Rebre(); // Rebre comanda

PORTD = (1 << 4);  //Encenem vermell
//Operació rebre trama i emmagatzemar les dades al buffer

		if (operador == 0x02){

			crc = _crc_ccitt_update(crc,operador);
			//longitud
			longitud = Rebre();
			crc = _crc_ccitt_update(crc,longitud);
```

```
                        //direccio
                        dirh = Rebre();
                        crc = _crc_ccitt_update(crc,dirh);

                        dirl = Rebre();
                        crc = _crc_ccitt_update(crc,dirl);

                        i = 0;
                        //Rebem trama de bytes
                        uint8_t Trama[longitud];
                        while (i<longitud){
                                Trama[i] = Rebre();
                                crc = _crc_ccitt_update (crc,Trama[i]);
                                i=i+1;
                        }
                        //Rebem el crc i l'unim
                        crch = Rebre();
                        crcl = Rebre();
                        crcr = Unio(crcl,crch);

                        //Comprovem el CRC, si està OK, unim la direcció i posem als
                        //valors al buffer
                        if (crcr == crc){
                                direccio = Unio(dirl,dirh);
                                i=0;
                                while (i<(longitud)){ //i sempre serà un nombre parell
                                        Paraula = Unio(Trama[i],Trama[i+1]);
                                        boot_page_fill_safe(direccio+i,Paraula);
                                        i=i+2;
                                }
                                Enviar(0x06); // Enviar ACK
                        }else{
                                Enviar(0x15); //Enviar NAK

                        }
                        crc = 0;

                }


//Operació escriure pàgina
        if (operador == 0x03){
                        boot_page_erase(direccio);
                        boot_spm_busy_wait();
                        boot_page_write(direccio);
                        boot_spm_busy_wait();
                        Enviar(0x06);
        }
```

```
//Operació finalitzar
        if (operador == 0x04){
                        Enviar(0x06);
                         PORTB = (0 << 3); //Apaguem Verd
                        while (1); //Bucle infinit

        }
} // claudàtor while(1)

} // claudàtor main
```

1.9 Codi de programa del programador

```
//---------------------------------------------------------------------------
//
// Name:        Programador.cpp
// Author:      Guillem Soler - Universitat de Vic
// Created:     18/04/2008 15:29:50
// Description: Programador de la placa entrenadora de l'ATMEGA16
// Funcionament: Cridar aquest l'executable passant-li com a paràmetre la direcció del //
//  fitxer hexadecimal a descarregar
//
//---------------------------------------------------------------------------

#include <stdio.h>
#include <stdint.h>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>

//VARIABLES GLOBALS

HANDLE Port; //Handle del port sèrie
DCB configuracio; //Configuració port sèrie
COMMTIMEOUTS temps; //time-outs
FILE *arxiu, *informe; //fitxer hexadecimal i informe extens
char Com[50]; //Nom del port sèrie
char linea[50]; //Línea de l'arxiu hexadecimal
unsigned char Trama[133]; //Trama que enviarem
unsigned int direccio_ant; //direcció del byte anterior
unsigned char num_bytes; //num de bytes de cada fila de l'hex
unsigned int direccio; //direccio inicial de cada fila de l'hex (dirl + dirh)
unsigned char dirh; //direcció alta de l'arxiu hexadecimal
unsigned char dirl; //direcció baixa de l'arxiu hexadecimal
unsigned char final; //valor que indica tipus de gravació
unsigned char bytes[50]; //Conjunt de bytes de cada fila de l'hex
unsigned int crc; //crc de cada trama que enviarem al micro
int reset = 0; //Variable que s'incrementa amb NAK


//ERRORS DE COMUNICACIÓ SERIE

void Informar_error(){
   DWORD error;
   COMSTAT cs;
   ClearCommError(Port,&error,&cs);

        if(error & CE_BREAK){ printf("#Error CE_BREAK (API's Windows)\n");}
        if(error & CE_FRAME){ printf("#Error CE_FRAME (API's Windows)\n");}
```

```
        if(error & CE_IOE){       printf("#Error CE_IOE (API's Windows)\n");}
        if(error & CE_MODE){   printf("#Error CE_MODE (API's Windows)\n");}
        if(error & CE_OVERRUN){printf("#Error CE_OVERRU(API's Windows)");}
        if(error & CE_TXFULL){ printf("#Error CE_TXFULL (API's Windows)\n");}
        if(error & CE_RXOVER){ printf("#Error CE_RXOVER (API's Windows)\n");}
}

//OBERTURA I INICIALITZACIÓ DEL PORT

bool Inicialitzar_port(){

  // Obrim el port per llegir, escriure.
 Port=CreateFile(Com, GENERIC_WRITE | GENERIC_READ,
FILE_SHARE_READ, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,NULL);

 // Mirem si s'ha produït un error en obrir el port
  if(Port==(INVALID_HANDLE_VALUE)) {
     printf("#Error al obrir el port serie %s\n",Com);
     Informar_error();
     return false;
  }

  //Llegir la configuració del port
  configuracio.DCBlength = sizeof(DCB);

  //Guardem a l'estructura dcb configuració
  if (!GetCommState(Port, &configuracio)){
     printf("#Error al llegir la configuracio del port:\n");
     Informar_error();
     return false;
  }

  //Configurem el paràmetres del port sèrie
  configuracio.ByteSize = 8; //num. bits
  configuracio.Parity = NOPARITY; // NO PARITAT JA UTILITZEM UN CRC
  configuracio.fParity = FALSE; // habilitem control de paritat
  configuracio.StopBits = ONESTOPBIT; // 1 Bit de stop
  //Actualitzem la configuració amb els nous valors amb la funció SetCommState
  if (!SetCommState(Port, &configuracio)){
     printf("#Error al salvar la configuracio del port:\n");
     Informar_error();
     return false;
  }

 //Obtenim configuració del Time-outs
  if (!GetCommTimeouts(Port, &temps)){
     printf("#Error al llegir la configuracio dels 'time-outs' per la comunicació serie\n");
```

```
        Informar_error();
        return false;
    }
    //Configurem els Time-outs
    temps.WriteTotalTimeoutMultiplier = 1; //Temps entre bytes max. Cada byte (8 bits dades + start i stop) tardaran 0,17 ms a enviar-se.
    temps.WriteTotalTimeoutConstant = 0; //Temps de marge després d'enviar tots els bytes
    temps.ReadTotalTimeoutMultiplier = 1000; //Temps d'espera entre bytes
    temps.ReadTotalTimeoutConstant = 0; //Temps de marge després de rebre
    //Guardem la configuració del time-outs
    if (!SetCommTimeouts(Port, &temps)){
        printf("#Error al salvar la configuració dels 'time-outs' per la comunicació serie\n");
    }

    return true;
}

//TANCAR EL PORT

void Tancar_port(void){
    SetCommMask(Port, 0);
    if (!CloseHandle(Port)){
        printf("#Error al tancar el port\n");
    }
}

//TANCAR ARXIU HEXADECIMAL

void Tancar_arxiu(){

     //Si dona 1 és que el fitxer no s'ha tancat
     if (fclose (arxiu) != 0)
        {
            printf("#Error al tancar el fitxer hexadecimal\n");
        }

}

//FUNCIÓ FINALITZAR APLICACIÓ

void Finalitzar(bool error){
    printf ("\n");

    //Comprovem si es finalitza amb error o  no
    if (error == true){
        printf("L'aplicacio ha finalitzat a causa d'un error\n");
```

```
   }else{
      printf("L'aplicacio ha finalitzat correctament\n");
   }
   Tancar_port();
   Tancar_arxiu();

   printf("\n\n---Fi del programa---\n");
   exit(error);  //Retornem 0 o 1 (0 correcte, 1 error)
}

//FUNCIÓ ENVIAR UNA TRAMA

void Enviar(uint8_t *Trama,unsigned int t){
    DWORD n = 0;

    //Enviem "t" caràcters de la cadena "Trama" pel port sèrie
   if (!WriteFile(Port,Trama,t,&n, NULL)) {
   printf("#Error en enviar les dades\n");
         Informar_error();
         Finalitzar(1);
   }
   ///Comprovar si s'han enviat totes les dades amb el time-out
   if (n<t){
      printf("#El 'time-out' de transmissió a expirat i s'han enviat nomes %d dades de
%d\n",t,n);
      Finalitzar(1);
   }

}

//OBRIR ARXIU HEXADECIMAL

void obrir(char* direccio_arxiu)
{
   fprintf(informe,"    - Arxiu hexadecimal seleccionat: %s\n\n",direccio_arxiu);
   arxiu = fopen(direccio_arxiu,"r");
   if (arxiu == NULL)
   {
      printf("#Error: No es pot obrir el fitxer hexadecimal\n");
      exit(1);
   }
}
```

```
//FUNCIÓ ACTUALITZAR CRC
///CRC16 CCITT (usat en PPP i IrDA, standard Europe), extret de "AVRlib-c"

uint16_t crc16_update( uint16_t crc, uint8_t data)
   {

      uint8_t lo = crc;
      uint8_t hi = crc>>8;

      data ^= lo;
      data ^= data << 4;

      return ((((uint16_t)data << 8) | hi ) ^ (uint8_t)(data >> 4)
            ^ ((uint16_t)data << 3));
   }

//FUNCIÓ RETALLAR (ANALISI DE L'ARXIU HEX)

void retallar(unsigned char* valor,int i,int num){

   char nou[num+1]; //Creem una cadena amb la longitud que voldrem
   strncpy(nou,linea+i,num); //copiem la part de la cadena que volem a nou
   nou[num]='\0'; //inserim al final el caràcter fi de cadena
   sscanf(nou,"%X",valor); //guardem a valor la conversió de la cadena nou amb
hexadecimal
  }

//FUNCIÓ OBTENIR PARTS DE L'ARXIU HEX (ANALISI DE L'ARXIU HEX)

void Obtenir_parts(){

    //Num_bytes
    retallar(&num_bytes,1,2);
    //direccio
   retallar(&dirh,3,2);
   retallar(&dirl,5,2);
   direccio = 0;
   direccio = dirh;
   direccio = direccio<< 8;
   direccio = direccio | dirl;

   //final (tipus de gravació)
   retallar(&final,7,2); //copiem el tros de la fila corresponent al num. de bytes
   //Cadena de bytes
   int i=0;
   while (i<num_bytes){
   retallar(&bytes[i],9+(i*2),2);
   i=i+1;
```

```
    }
}

//INICIALITZACIÓ DE LA TRAMA

void Inicialitzar_trama(int i){
    Trama[0]=0x02; //STX (start of text)
    Trama[2]=(direccio+i) >>8; //Part alta de la direccio
    Trama[3]=direccio+i; //Part baixa de la direccio;

}

//CALCUL DEL CRC I INCORPORACIO A LA TRAMA
//Calcula el CRC de la trama que li passem i el situa a les 2 ultimes posicions

void Calcular_crc(int k){
    crc = 0;
    int i=0;
    while(i<k+4){

        crc = crc16_update(crc,Trama[i]);
        i=i+1;
    }

    Trama[k+4]=crc>>8;
    Trama[k+5]=crc;

}

//FUNCIO REPOSTA DEL MICROCONTROLADOR

bool Comprobacio_ACK(){

    DWORD n,dwEvtMask;
    char t;
    fprintf(informe,"Comprovant que les dades hagin set rebudes correctament...\n");
    // Llegim el caràcter i l'emmagatzemem a t
    if(!ReadFile(Port,&t,1,&n, NULL)){
                printf("#Error en rebre les dades:\n");
                Finalitzar(1);
    }
    //Si no es rep ACK
    if (t != 0x06){
        reset = reset + 1; //Variable intents de rebre ACK
        if (reset == 3){
            printf("#Error: tercer intent d'enviar trama/operacio fallit, la carrega no ha set
            possible\n");
                Finalitzar(true);
```

```
        }
        //Si es rep NAK
        if (t==0x15 ){
                fprintf(informe,"Trama no rebuda correctament,es torna a enviar la
trama/operacio (intent %d)\n\n",reset);
        //Si es rep un altre valor
        }else{
                fprintf(informe,"El temps d'espera de la resposta ha expirat,es torna a enviar
la trama/operacio (intent %d)\n\n",reset);
                return false;
        }
        //Si es rep ACK
    }else{
       printf("La trama/operacio ha set rebuda amb exit\n\n");
       reset = 0;
       return true;
    }
}

//FUNCIÓ DE PREPARAR TRAMA

void Preparar_trama(int k){
    int p = 0;
    div_t divisio;
    fprintf(informe,"Preparem la trama seguent:");
    divisio=div(k,2);
    //Si hi ha un nombre imparell de bits se n'haurà d'afegir un
    if (divisio.rem!=0){
        Trama[k+4]=0xFF;
        k=k+1;
    }
    //La posició 1 hi ha el número de bytes(Només bytes de dades)
    Trama[1]=k;
    Calcular_crc(k);
    //Escrivim tota la trama que enviarem per pantalla
    while(p<k+6){
        fprintf(informe,"%X-",Trama[p]);
        p=p+1;
    }
    fprintf(informe,"\n\n");
    printf("Enviant trama...\n\n");
    do{
    //Enviem la trama i la longitud d'aquesta mentre no es rebi ACK
    Enviar(Trama,Trama[1]+6);
    }while(Comprovacio_ACK()== false);

}
```

```
//FUNCIÓ PER ENVIAR UNA COMANDA(GRABAR PAG. O FIN.) AL MICROCONTROLADOR

void Comanda_micro(unsigned char t){
   //si t = 0x03 -> Gravar pàgina, t = 0x04 -> Aturar micro
   if (t == 0x03){
      printf("Procedim a gravar pagina...\n");
   }else{
      printf("Procedim a finalitzar el programa\n");
   }
   do{
      //Enviem la comanda (només volem una posició (un caràcter))
        Enviar(&t,1);
   }while(Comprovacio_ACK() == false);
}

//DETECTAR EL CAMBI DE PAGINA, RETORNA 0 SI NO HI HA CANVI, 1 SI N'HI HA

bool Canvi_pagina(int direccio_ant,int direccio){
   div_t num_pag_act,num_pag_ant;
   num_pag_act=div(direccio,0x80);
   num_pag_ant=div(direccio_ant,0x80);
   if (num_pag_ant.quot != num_pag_act.quot){
      return true;
   }else{
      return false;
   }
}
//COMPROBAR SI LES DIRECCIONS VAN SEGUIDES

bool Seguit(int direccio_ant,int direccio){
   if(direccio<=direccio_ant+1){
      return true;
   }else{
      return false;
   }
}
//COMPROBA LONGITUD HEX, OBTÉ LA PRIMERA LÍNEA

bool Comprobar_hex(){

  //Comprobar longitud /
  unsigned int longitud = 0;
   while (fgets(linea,50,arxiu)!= NULL){
      Obtenir_parts();
      if ((direccio+num_bytes)>=0x3C00){
```

```
            printf("#Error: El codi de programa sobrepasa la seccio aplicació en direccio %X\n",direccio+num_bytes);
                return 0;
        }
    }
    //Rebobinar l'arxiu
    rewind(arxiu);
    //Obtenim la primera direccio vàlida
     do{
        fgets(linea,50,arxiu);
        Obtenir_parts();
     //Si l'hex es tret de codi assemblador té una primera fila que no és de codi on el Record type és 0x02
     }while(final!=0x00);
     direccio_ant = direccio;
     rewind(arxiu);
     return 1;
}

//FUNCIÓ PER OBTENIR EL PORT DE L'ARXIU DE LA CONFIGURACIO

void obtenir_conf(){

     FILE *conf;
     //Obrim el fitxer conf i comprovem que existeixi
     conf = fopen("programador.conf","r");
     if (conf == NULL) {
            printf("#Error: No s'ha pogut obrir el fitxer de la configuracio\n");
            exit(1);
     }else{

          int c;
          //llegim caràcters fins a trobar el caràcter "="
          do{
            c = fgetc(conf);
          }while(c !='=');
          //Guardem el Port, atenció que també es guarda el caràcter salt de línea
          char Comb[26];
          fgets(Comb,26,conf);
          //Guardem el nom del Port sense el caràcter salt de línea
          strncpy(Com,Comb,strlen(Comb)-1);

          if (fclose(conf)!= 0) {
                printf("#Error en tancar fitxer de configuracio\n");
                exit(1);
          }
        }
}
```

```
//PROGRAMA PRINCIPAL-----------------------------------------------------------

int main(int num_arg,char **direccio_arxiu,char **ext){

   printf("\n---Programador executat---\n\n");
   //Comprobar que ens passen l'arxiu hexadecimal com a paràmetre
   if (num_arg == 1){
        printf("#Error: Falta afegir el nom del fitxer hexadecimal com a parametre");
        exit(1);
   }
   //Mirem si es vol informe llarg (arguments >=3) o informe curt (contrari)
   if (num_arg < 3){
      //L'informe serà un arxiu temporal que serà esborrat al final del programa
      informe = tmpfile();
   }else{
      //L'informe serà igual a stdout (per tant s'escriurà a stdout)
      informe = stdout;
   }

   //informar a l'usuari d'informe extens
   fprintf(informe,"    - S'ha seleccionat el tipus d'informe extens:\n");

   unsigned int i = 0;
   unsigned int k = 0;

   //Obrim arxiu hexadecimal
   obrir(*(direccio_arxiu+1));

   //Comprovació de validesa de l'arxiu hexadecimal
   if (!Comprobar_hex()){
     fprintf(informe,"Arxiu 'hex' no valid\n");
     exit(1);
    }

  //Obrir arxiu de configuració i obtenir-ne nom del port
   obtenir_conf();

   //Inicialització del port
   if (!Inicialitzar_port()){
     fprintf(informe,"Error en la inicialitzacio del port\n");
     exit(1);
   }

 Inicialitzar_trama(0); //Inicialització de la trama
//---------------------------------------
   while (fgets(linea,50,arxiu) != NULL){
```

```
        Obtenir_parts();
        //Si no és linea de codi no la tractem.
                if (final==0x00){
        i=0;
           while(i< num_bytes){
           //Comprovem si hi ha una cambi de pàgina
            if (Canvi_pagina(direccio_ant,direccio+i)){
                //Preparar i enviar trama
                Preparar_trama(k);
                //Inicialitzar la trama
                Inicialitzar_trama(i);
                //Enviem comanda de gravar pàgina
                Comanda_micro(0x03);
                 k=0;//Apuntador de la trama a 0
              }else{
                //Si no hi ha canvi de pàgina comprovem si el que ve va seguit per enviar
                ja una trama o afegir-ho a la mateixa
                if (!Seguit(direccio_ant,direccio+i)){
                        Preparar_trama(k);
                        Inicialitzar_trama(i);
                        k=0;
                }
              }
              //Afegim byte a trama
              Trama[k+4]=bytes[i];
              //Substituïm la direcció anterior per la que hem tractat
              direccio_ant=direccio+i;
              i=i+1;
              k=k+1;
           }
      }
}
Preparar_trama(k);
Comanda_micro(0x03);
Comanda_micro(0x04);
Finalitzar(0);

}

//-------------------------------------------------------------------------------
```

2. ANNEX 2 - Manual del programador

2.1 Requeriments tècnics

Per utilitzar aquest sistema de gravació es necessari disposar del hardware següent:

- Ordinador amb el sistema operatiu Microsoft Windows (95 o superior) amb una interfície RS-232 (Port sèrie) de 9 pins. Si no es disposa d'aquesta interfície, es pot adquirir un adaptador USB-sèrie (DB-9)
- Placa entrenadora compatible amb el sistema i alimentador.
- Cable Null – Modem, amb connexió femella en els dos extrems.

El software que es requereix és el següent:

- Programador (Interfície gràfica és optativa)
- Fitxer de configuració

També es necessari tenir el microcontrolador de la placa entrenadora configurat:

- Configurar l'adreça d'arrencada a la direcció inicial de la secció *bootloader*
- Configurar la direcció inicial de la secció bootloader a l'adreça 0x3C00
- El microcontrolador ha de contenir el programa del bootloader.

2.2 Muntatge i preparació

Les instruccions que s'han de seguir per fer el muntatge del sistema de gravació són les següents:

1. Connectar un extrem del cable Null-Modem a d'interfície RS-232 de la placa entrenadora.

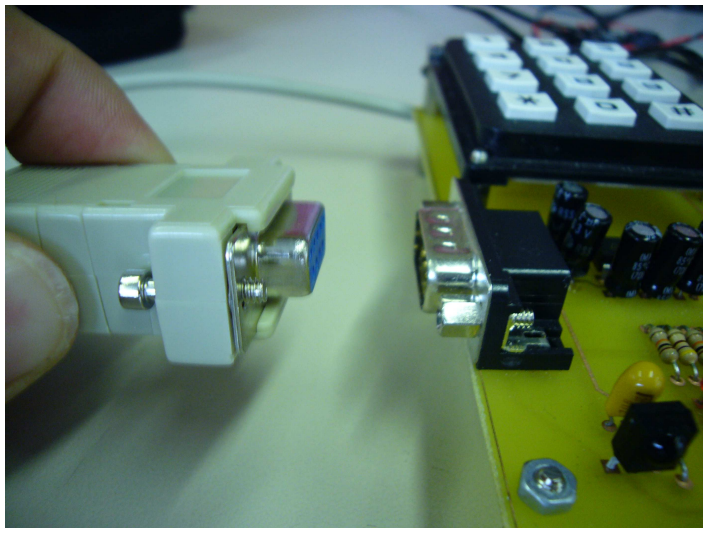


Figura A.2.1.1: Connexió Null-Modem – Placa

2. Connectar l'altre extrem del cable Null-Modem a d'interfície RS-232 de l'ordinador.

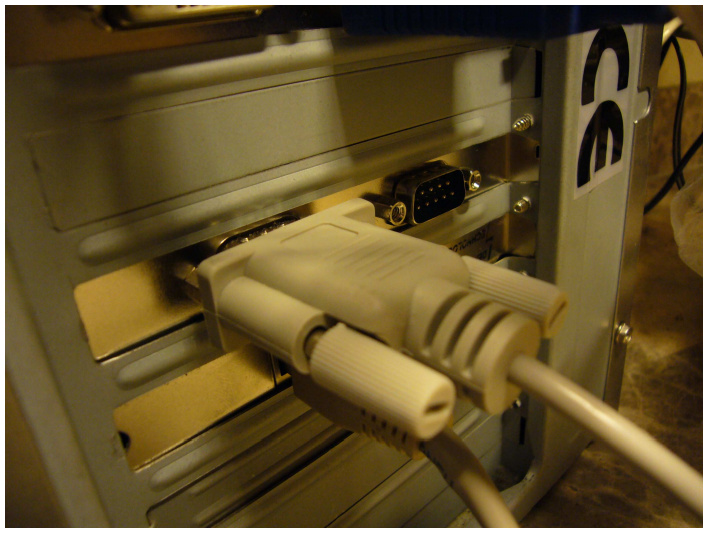


Figura A.2.1.2: Connexió Null-Modem – PC

3. Engeguem l'ordinador i el S.O. Microsoft Windows

4. Polsem el polsador PD2 de la placa entrenadora i el mantenim polsat mentre alimentem (o polsem el polsador de reset si ja està alimentada) la placa (**Figura A.2.5.1**). D'aquesta manera accedirem al *bootloader*. Si el led tricolor de la placa s'encén de color verd, s'haurà accedit correctament al *bootloader*. Sinó, s'ha de repetir la instrucció número 4.

2.3. Utilització del programador

El programador pot ser utilitzat des de d'interfície gràfica o bé des de la consola. És necessari disposar del fitxer de configuració del programador. En cas de no tenir aquest fitxer, la interfície gràfica el pot crear entrant en l'apartat d'opcions. Els passos que s'han de seguir per executar correctament el programador des de la consola són:

1. Obrir el fitxer de configuració i especificar el nom del port a utilitzar.
2. Executar la consola (Menú inicio -> Ejectuar... -> cmd)
3. Cridar el programador passant-li com a paràmetre la direcció de l'arxiu hexadecimal. Si la direcció de l'arxiu hexadecimal o del programador, conté espais en blanc, situar-la entre cometes. Exemple:

```
C:>Programador.exe "C:\Documents and Settings\usuari\FitxerHex.hex"
```

- L'informe que apareixerà per pantalla un cop s'executi el programador serà breu. Si es vol visualitzar un informe extens cal afegir un segon paràmetre (No importa quin valor tingui). Per exemple:

```
C:>Programador.exe "C:\Documents and Settings\usuari\FitxerHex.hex" llarg
```

4. Es pot redireccionar l'informe que surt per pantalla en un fitxer. Per això, cal afegir l'operador ">" (sobreescriure fitxer) o ">>" (afegir a final de fitxer) seguit del nom del fitxer. Exemple:

```
C:>Programador.exe "C:\Documents and Settings\usuari\FitxerHex.hex" llarg > info.txt
```

2.4. Utilització de la interfície gràfica

2.4.1 Finestra principal

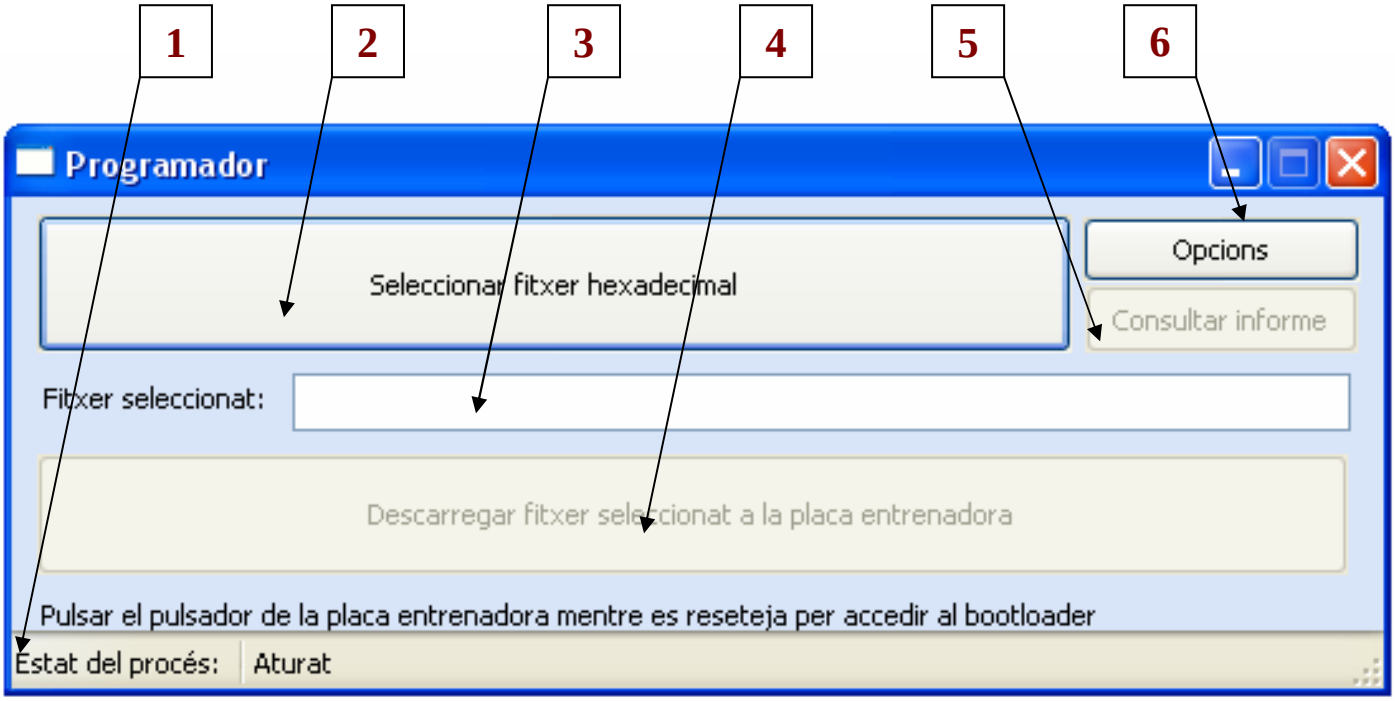


Figura A.2.4.1: Finestra principal

1	Indica l'estat del procés
2	Clicar aquest botó per seleccionar el fitxer hexadecimal que es vulgui descàrregar a la placa.
3	Informe de la direcció del fitxer hexadecimal seleccionat
4	Clicar aquest botó per descarregar el fitxer hexadecimal seleccionat a la placa entrenadora
5	Clicar aquest botó per consultar l'informe de la descàrrega, si s'ha generat.
6	Clicar aquest botó per accedir a la finestra de configuració

2.4.2 Finestra de configuració

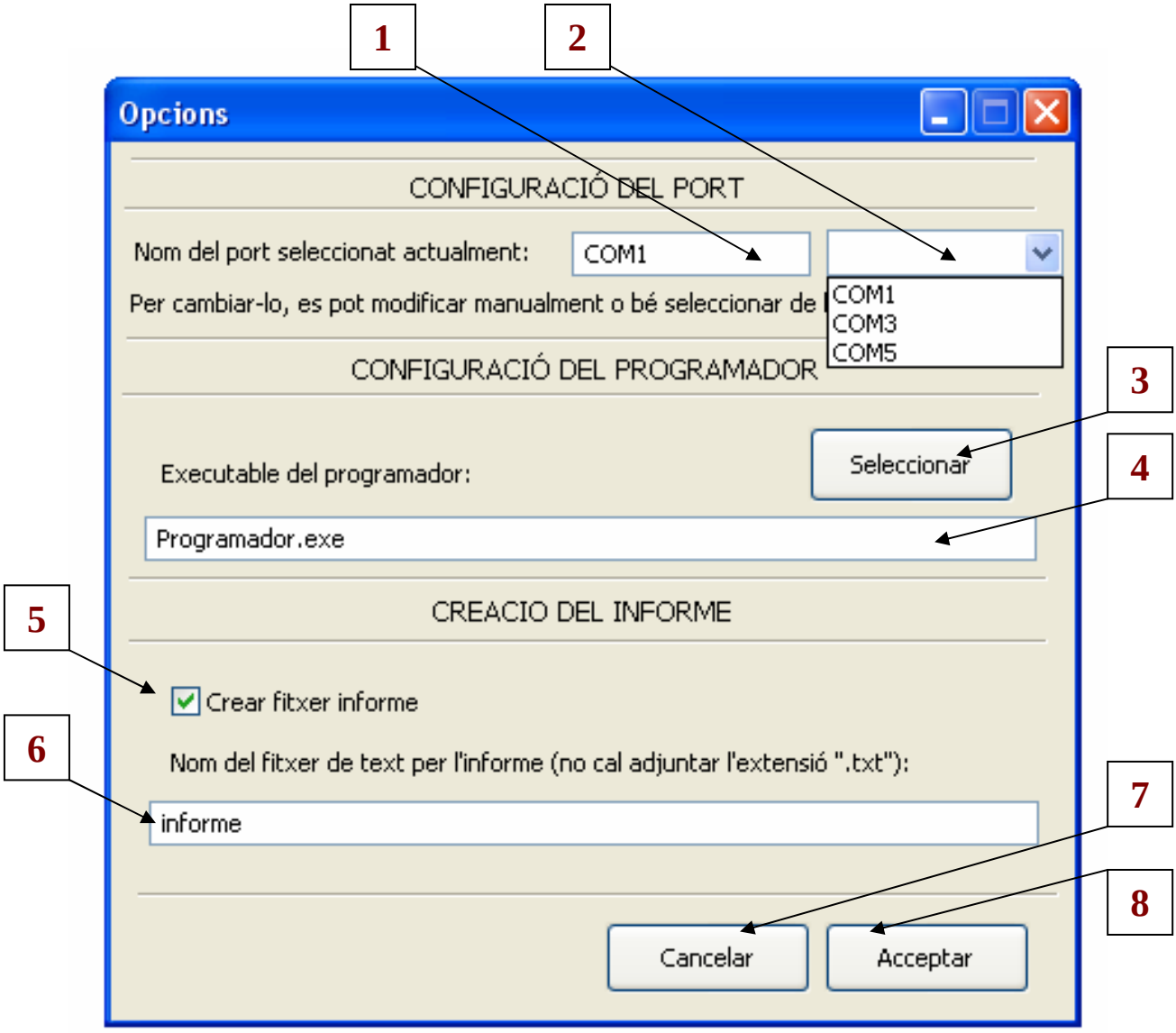


Figura A.2.4.2: Finestra de configuració

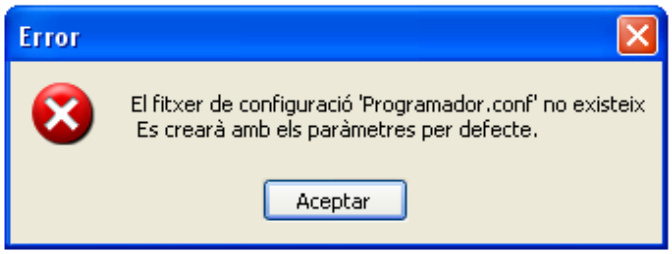
1	Permet modificar el nom del port a utilitzar (També es pot escollir de la llista)
2	Llista de ports disponibles. Es pot seleccionar el port a utilitzar de la llista
3	Clicar aquest botó per seleccionar l'executable del programador. Aquest, s'ha de trobar dins la mateixa carpeta que la interficie gràfica.
4	Ens mostra la direcció del programador seleccionat

5	Si està activada aquesta opció, se'ns generarà un fitxer amb un informe del programador
7	Sortir i acceptar els paràmetres establerts en la finestra de configuració
6	Nom del fitxer informe que es crearà. No és necessari adjuntar l'extensió ".txt"
8	Sortir i deixar la configuració tal com estava abans d'entrar a la finestra de configuració

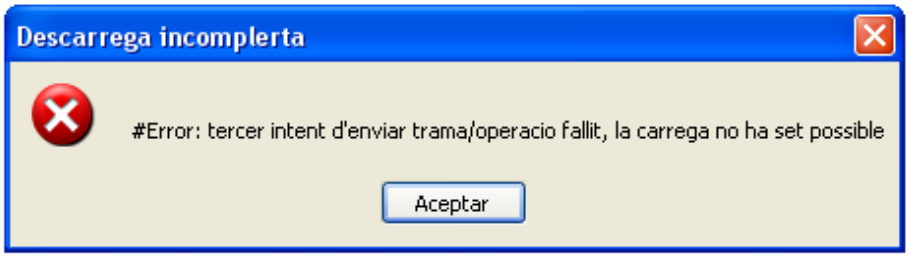
2.4.3 Notes

- Els paràmetres de configuració també poden ser modificats dins el fitxer de configuració "programador.conf"

- En cas de no disposar d'aquest fitxer l'aplicació no deixarà efectuar la programació de la placa entrenadora. Per crear aquest fitxer, polsar el botó "Opcions" i acceptar el següent missatge:



- Un cop el programador hagi finalitzat s'informarà a l'usuari de l'estat final. Si ha sorgit algun error, també es mostrarà per pantalla. Exemple:



2.5. Utilització del bootloader

Per efectuar una gravació a la placa entrenadora, el microcontrolador ha d'entrar dins el programa de *Bootloader*. Per fer això, abans d'executar el programador. Alimenteu o feu un "reset" (Polsador de reset) a la placa entrenadora mantenint polsat el polsador PD2 (**Figura** . El led tricolor indicarà que s'ha accedit al bootloader.

Aquest led, prendrà un color diferent depenent de l'estat en que es trobi el bootloader.

	El bootloader està aturat, esperant dades del programador
	El bootloader està efectuant operacions o rebent dades del programador
	El bootloader ha finalitzat la gravació del programa correctament

Per accedir a l'aplicació carregada feu un "reset" a la placa entrenadora sense polsar el polsador.

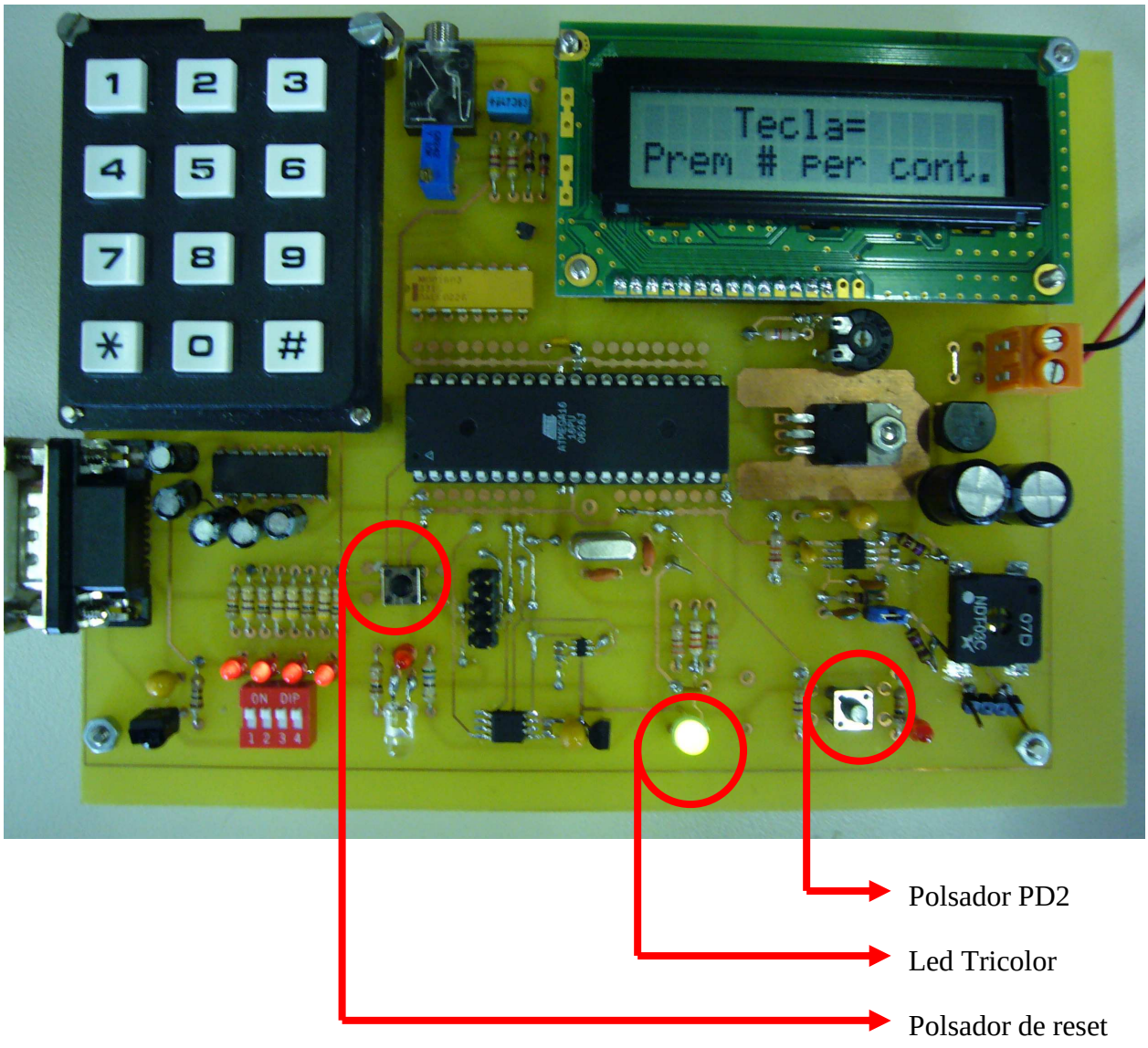


Figura A.2.5.1: Polsador i led tricolor de la placa entrenadora