# Contingut

# Resum de l'enunciat

A la pràctica 2, fase B de l'assignatura de SDM, es demana realitzar un circuit amb una PIC18F4321 per poder jugar al clàssic joc de *l'SNAKE*. Per aconseguir-ho, s'utilitza el microcontrolador mencionat amb uns sensors i components auxiliars com un joystick pels moviments de la serp, un altaveu amb un mòdul amplificador per la música, un teclat pel menú del joc i també per utilitzar-lo com a fletxes i una pantalla LCD de 2 files i 16 columnes per mostrar el menú i diverses dades dels usuaris i configuració a més a més de la PICKIT per programar la mateixa PIC.

A trets generals, el menú s'utilitza per inicialitzar una nova partida a l'ordinador connectat per sèrie, per visualitzar les millors puntuacions, visualitzar els usuaris guardats i modificar-los, veure l'hora del sistema i finalment modificar-la.

Per la pràctica s'utilitzaran conceptes com la Analògic-Digital-Converter, L'EUSART, el sistema d'un teclat matriu, el sistema d'una pantalla LCD en format cooperatiu i altres conceptes sempre d'es d'un punt de vista cooperatiu que permeti simular que funcionin diverses tasques al mateix instant.


*Il·lustració 1 - Diagrama de connexions extret de l'enunciat.*

## Plantejament del software

El codi de la PIC s'executa en dues fases: una primer al començar el cicle d'engegada quan es connecta per primer cop on s'inicialitzen alguns valors i processos i el més important que és una fase on s'executa el codi de forma cíclica i cooperativa que permet simular que s'executen i es duen a terme tasques a la vegada: l'anomenat cooperatiu. Amb aquest sistema, es poden fer diverses tasques a la vegada com el control d'un *joystick* i actualitzar la pantalla LCD a la mateixa vegada.

A la primera part s'inicialitza el *TIMER0* per interrompre cada cert temps necessari tenint en compte *l'EUSART*, el *PWM* de l'altaveu i el mínim permès aproximat per sistema cooperatiu en llenguatge C, l'EUSART per la transmissió en sèrie, la memòria EEPROM i el conversor ADC a més a més dels ports d'entrada i sortida.

Per començar a dissenyar el software, s'ha començat amb un prototip de diagrama de TADs i s'ha fet una segona versió a mesura que s'escrivia el codi, el qual es pot veure a l'apartat del diagrama de TADs. Un cop feta la primera versió, s'ha plantejat que amb el sistema cooperatiu a diferència del C, es pot plantejar el codi d'una manera independent per cada funció, procediment o més ben dit: TAD, de manera que s'ha separat el projecte en blocs o TADs on cada un (o dos en algun cas) realitza una o unes accions específiques sense trencar un *"low coupling"* necessari per poder avançar correctament.

Un dels punts important a decidir és el *timer*, que s'ha escollit que interrompi cada 0.5ms que permet realitzar el *PWM* de l'altaveu al 50% del cicle de treball i és possible carregar el registre del valor inicial amb un valor de no masses bits amb l'oscil·lador utilitzat format per un cristall de 10MHz de quarts i un HSPLL, funcionant a 40MHz d'oscil·lació.

Finalment s'ha de tenir en compte que cal optimitzar molt bé la memòria ja que amb llenguatge C s'ocupen múltiples línies d'*assembler* i s'emplena la memòria RAM i FLASH extremadament ràpid.

Per tots els punts dits anteriorment, la resta del plantejament sols serà necessari quan s'escriu codi i les parts més importants ja estan establertes en aquest mateix apartat.

## Configuracions del microcontrolador

És necessari configurar alguns paràmetres del microcontrolador per obtenir el funcionament volgut. Alguns registres venen per defecte amb el valor "0" ja inserit i s'han estalviat de configurar en el cas que el valor desitjat ja sigui 0.

Els diferents registres s'han configurat de la següent manera:

- L'oscil·lador

L'oscil·lador del projecte funciona a 40MHz per poder executar tasques prou ràpid. A més permet que el Timer0 arribi al temps volgut. Es configura a High Speed amb multiplicador PLL amb la comanda "#pragma config osc = OFF".

- Els Ports

Per decidir on va cada element, s'ha tingut en compte que el PortC ja té uns pins ocupats pel PickKit i per la *EUSART*, que el PortE té el pin de RESET, al PortB es poden fer interrupcions (però no s'utilitzaran) amb l'ajuda de les resistències de *pull-up* i finalment que el PortA té l'ADC i se'n necessiten dos de canals. Finalment, la distribució de ports és la taula inferior.

| PORT | PIN | Dispositiu | PIC TRISx |
|---|---|---|---|
| Port A | A7,A6 | OSC | 0,0 |
| | A0 | X JoyStick | 1 |
| | A1 | Y JoyStick | 1 |
| Port B | B7,B6 | PicKit | 1,1 |
| | B3 | LED Debug | 0 |
| | B1 | Speaker | 0 |
| Port C | C7, C6 | RX,TX | 1,1 |
| | C5,C4 | LCD RS, LCD E | 0,0 |
| | C3,C2,C1,C0 | LCD Data[7..4] | 0,0,0,0 |
| Port D | D6,D5,D4 | Teclat Col[3..1] | 0,0,0 |
| | D3,D2,D1,D0 | Teclat Fila[4..1] | 1,1,1,1 |

*Taula 1 - assignació de ports.*

- L'EUSART i L'LVT

La EUSART s'utilitza per la comunicació entre la placa i el joc i és necessari que sigui bidireccional. Per tant, funcionant a 9600 de baudrate, un valor mig prou alt i no extremadament ràpid.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 16-bit/Asynchronous | |
| 1 | 0 | x | 8-bit/Synchronous | Fosc/[4 (n + 1)] |
| 1 | 1 | x | 16-bit/Synchronous | |

Primer es configura el registre BAUDCON:

| R/W-0 | R-1 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

| manual | actiu | lògica + | lògica + | càlcul | - | No monitoritzat | Des-activat |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

I es carrega el valor "n", movent el valor "high" a *SPBRGH* i el valor "low" a *SPBRG*. En aquest moment el *baud rate* està configurat i falta configurar el registre d'emissió i recepció, començant per l'TXSTA:

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-1 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

| - | 8bits tx | actiu | càlcul | ASYNC | càlcul | - | - |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

I l'RCSTA

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

| actiu | 8bits | - | actiu | desactivat | No framing | No overrun | - |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Per acabar també es configura l'LVT amb "#pragma config LVP = OFF".

- Les interrupcions

No s'utilitzen interrupcions més enllà del Timer0 i els flags de l'EUSART, però interrupcions com a tal no s'utilitzaran. En aquest cas es deixen els registres sense modificar amb els valors predeterminats.

- EL Timer0

S'utilitza per les interrupcions que necessiten controlar-se cada cert temps. En concret, necessitem comptar cada 1 minut, pel control d'uns 20ms del teclat pels rebots i pel PWM de l'altaveu. Són diferents temps però el més ràpid o menor és l'altaveu cada 0.5ms pel mig període. Per tant, saltarà la interrupció cada 0.5ms i per configurar-lo a aquesta velocitat, es fa el següent:

$$Temps = \left(2^{bits\ Timer} - valor\ càrrega\right) \times prescaller\ \times \frac{4}{Fosc};$$

$$0.5ms = \left(2^{bits\ Timer} - valor\ càrrega\right) \times prescaller\ \times \frac{4}{40MHz}$$

On es necessita un prescaller de 256 i 8 bits, amb un valor de càrrega de 236.

El T0CON es configura amb "11000111" i l'INTCON amb "1110100" i també es deshabilita l'IPEN del registre RCON. Cada cop que salta la interrupció es desactiva el flag del Timer0 amb INTCONbits.TMR0IF.

Cal dir que el fitxer .c i .h del Timer0 ja ve donat per la universitat però s'ha hagut de modificar a les necessitats.

- L'ADC

S'utilitza per convertir el valor de l'eix del JoyStick d'entre 0V i 5V pels canals AN0 i AN1. Per configurar-lo utilitzem els següents registres:

**ADCON2***:*

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |
| bit 7 | | | | | | | bit 0 |

| Alineat esquerra | - | 2*Tad | = | = | Conversió cada F/64 | = | = |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

**ADCON1:**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0(1) | R/W(1) | R/W(1) | R/W(1) |
|---|---|---|---|---|---|---|---|
| — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

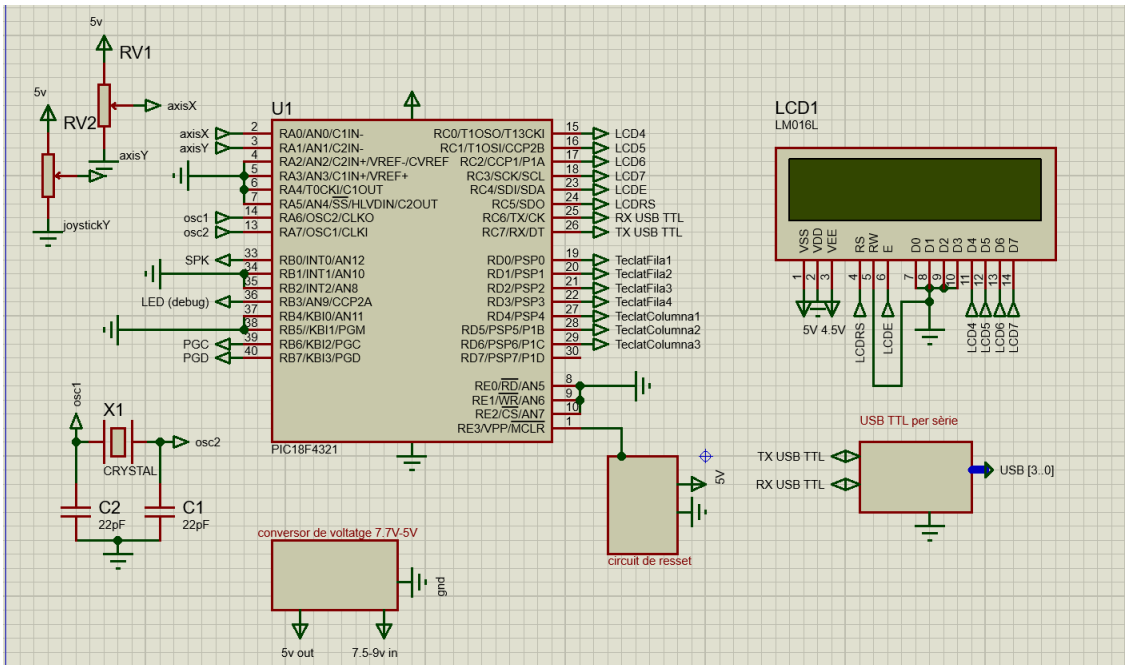
| - | - | no Vref- | No Vref+ | AN0 analog | = | = | = |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

**ADCON0:**

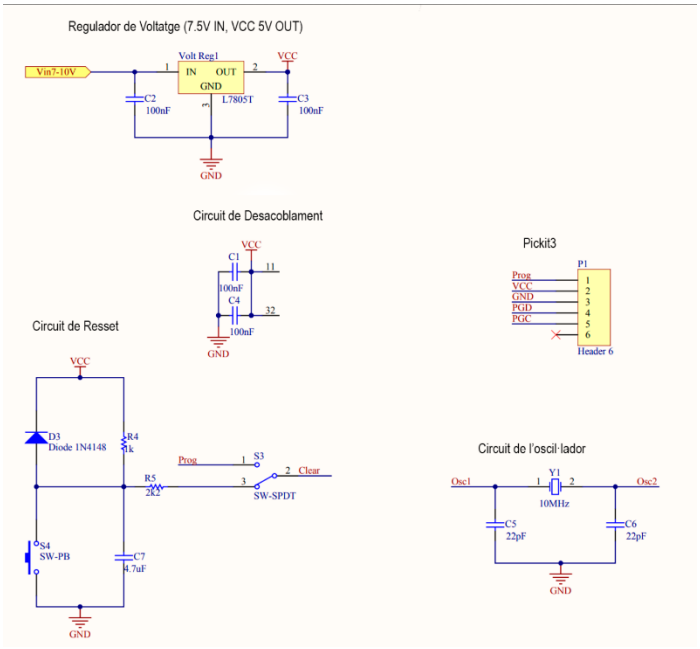
| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON |
| bit 7 | | | | | | | bit 0 |

| - | - | AN0 anlog | = | = | = | pausa | ADC ON |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## Esquema elèctric




*Il·lustració 3 - Imatge amb l'esquema elèctric utilitzat.*




*Il·lustració 2 - Diagrama obtingut del datasheet pic18f4321 rev 26/3/2009*

A l'esquema es pot veure el microcontrolador PIC18F4321 amb els diferents mòduls i ports utilitzats. Les sortides no utilitzades es troben connectades a terra. El port de la

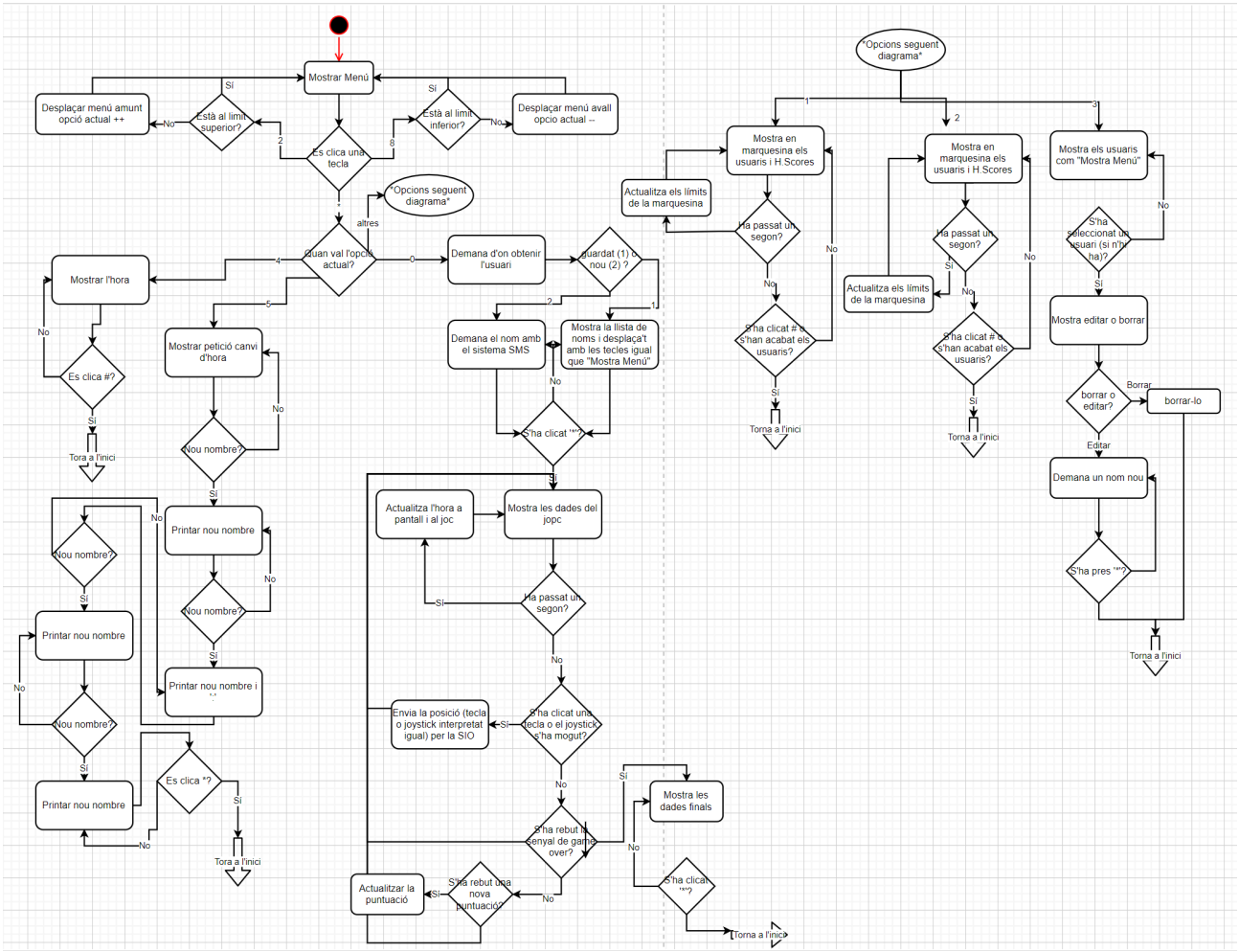
pantalla LCD RW sempre està connectat a terra per no utilitzar les funcions respectives a introduir-hi 5V.

El circuit de reset consisteix en un circuit on el botó indica l'acció de reset a la vegada que si es selecciona amb un interruptor un segon circuit on la PicKit indica la seqüència de resets per programar la placa, no hi hagi un canvi sobtat de voltatge i es carregui/descarregui un condensador per evitar-ho.

El conversor de voltatge assegura que mentre s'utilitzi un voltatge al voltant de 7V i no molt superior, proporcioni 5V fixes i constants sense pics per protegir la placa de les fonts de voltatge i pujades de tensió.

## Diagrama d'activitat del software

A continuació es mostra el diagrama d'interacció de l'usuari amb el programa: una versió reduïda del procés progressiu que el codi executa cíclicament. Està format per un menú principal i diferents branques (6) que cauen per cada opció:




*Il·lustració 4 - Diagrama d'activitats de la interacció amb l'usuari del programa.*

## Diagrama i funcionament dels TADs

Els TADS utilitzats són:

**TAD Timer:** pel temporitzador i comptador del temps.

**TAD Menú:** Amb el control de les frases de text del menú i les opcions seleccionades, unifica el teclat amb la LCD.

**TAD Gestor LCD:** S'encarrega d'introduir les cadenes de caràcters al TAD LCD i gestiona la visualització per marquesina.

**TAD Usuari:** incorpora les funcions per modificar els usuaris i crear-ne. També els guarda.

**TAD LCD:** Format per la combinació del TAD proporcionat per la universitat i editat per tenir una funció "put String" cooperativa creada nova i una més per comprovar si la pantalla està disponible.

**TAD Hora:** S'encarrega de controlar l'hora del sistema. L'emmagatzema en les seves variables.

**TAD SMS:** converteix els nombres del teclat en lletres seguint el format SMS.

**TAD Joc:** s'utilitza per les funcions de quan s'està jugant, controla el joc i la puntuació.

**TAD JoystickADC:** Converteix els potenciòmetres del joystick amb l'ADC.

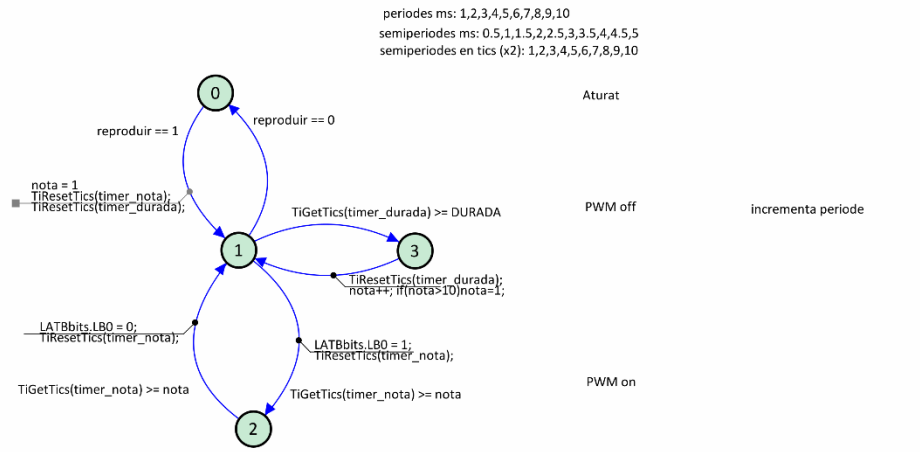
**TAD SIO** l'EUSART, encapsula les dades i en rep de l'ordinador.

**TAD Speaker:** S'ocupa de reproduir música per l'altaveu en 10 tons diferents quan se li indica.

**TAD Teclat:** llegeix i escombra el teclat per descobrir si es prem una tecla i comunica quina tecla s'ha pres.

## Diagrames i funcionament dels motors

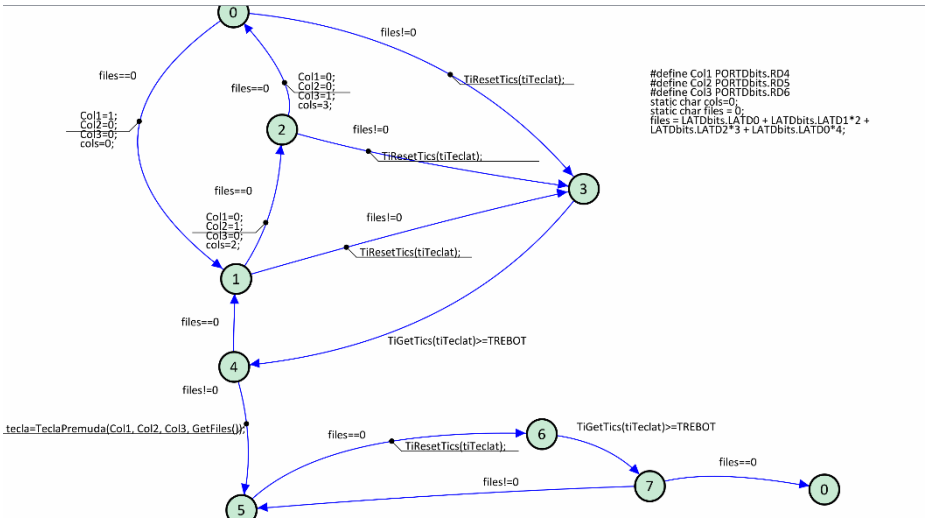
- **TAD Altaveu**
  S'encarrega del PWM de l'altaveu per realitzar 10 notes diferents d'un període concret i DC del 50% durant un temps també especificat:




*Il·lustració 5 - Diagrama del motor altaveu.*

En aquets TAD un timer controla el PWM de la nota actual i un segon timer controla cada quan es canvia el període. Totes les notes tenen la mateixa duració ja que és optativa però si es controla el temps amb flanc a u i a zero. Aquest sistema s'activa quan la variable "reproduir" s'activa a "1" i es desactiva quan canvia a "0" des d'una funció.

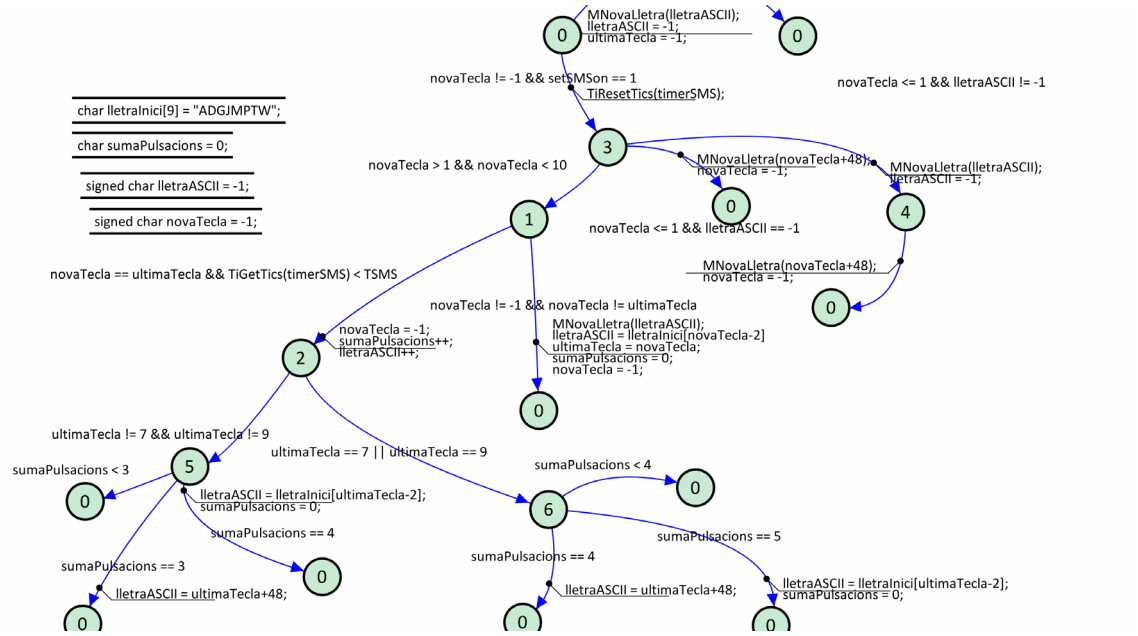
- **TAD Teclat**




*Il·lustració 6 - Diagrama del motor Teclat.*

Controla l'escombrat continuo del teclat i emet al menú la tecla premuda. Cada vegada que es prem una tecla, la comunica al TAD Menú sense dependre d'aquest de manera que sap a l'instant quan es prem i no necessita ser activat. Per escombrar, s'activa columna a columna i comprova si almenys una de les files està també activada. En cas d'haver una fila activada verifica quina tecla ha de ser la premuda i si en 20ms pel control de rebots continua la mateixa tecla activada, decideix que s'ha premut i l'envia.

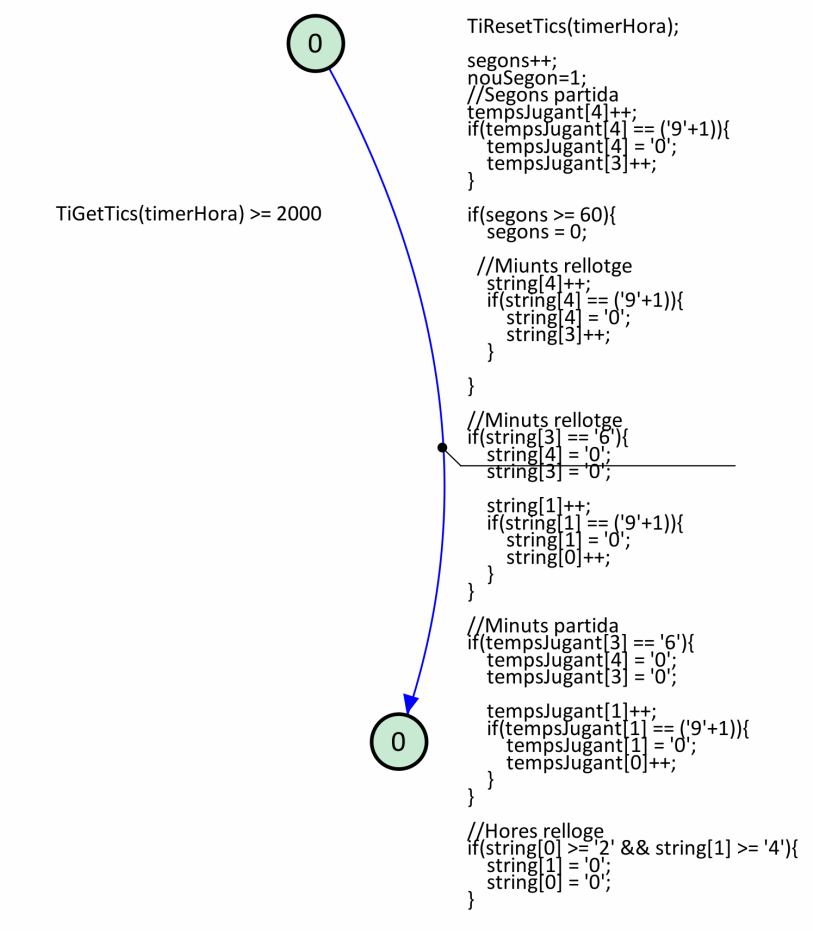
- **TAD SMS**




*Il·lustració 7 - Diagrama del motor TAD SMS*

S'encarrega de tractar les tecles que es premen i provenen del TAD Teclat per convertir-les a caràcters SMS sempre i quan s'activi la seva comprovació. Per fer-ho, verifica després de cada tecla premuda si la nova tecla és la mateixa i si ho és, incrementa de manera cíclica el caràcter a obtenir fins que es premi una nova tecla o passi el temps d'espera per acceptar la tecla premuda amb un *timer*. Després dels tres o quatre caràcters de cada tecla apareix el nombre real de la tecla per si es vol utilitzar. Utilitza un array de caràcters sempre igual i no modificable per comptar a partir de quin caràcter sumar clics. A la imatge es pot veure una cascada amb diferents desviacions per cada tecla diferent: 1 i 0 sense lletrs, * i # i finalment les altres tecles amb lletres. Cada branca també es desvia per si és un dels 3 o 4 clics amb lletra o el 4rt o 5è amb el nombre.

- **TAD Hora**



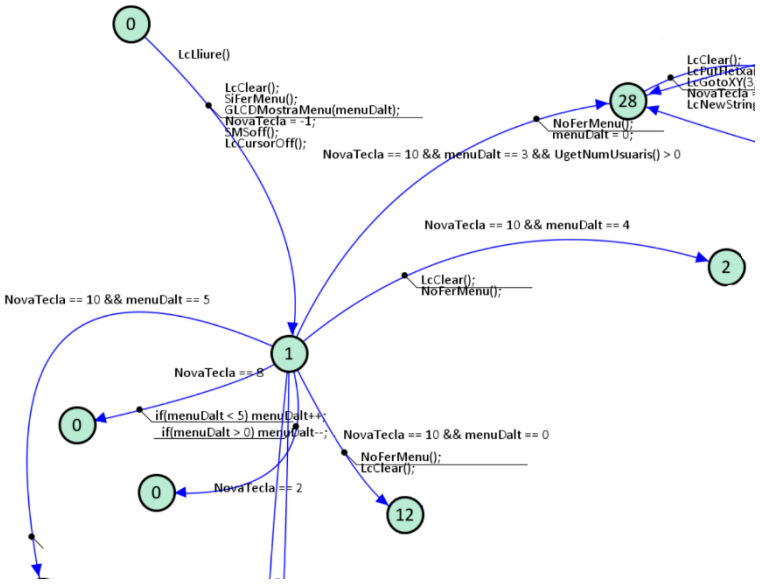

*Il·lustració 8 - Diagrama del motor TAD Hora.*

El TAD Hora, per cada minut incrementa una array amb el format “HH:mm” de manera que cíclicament passa de 0 a 9 a les unitats dels minuts i de 0 a 5 a les dècimes dels minuts. El caràcter “:” no es modifica i el mateix opera amb les hores: de 0 unitats a 9 i de 0 a 2 a les dècimes. L’array del temps actual es pot consular amb una funció independentment del motor i també es pot substituir en qualsevol moment per una nova array quan es vulgui modificar. Addicionalment s’ha utilitzat un array tempsJugant per saber quanta estona ha passat des del inici de la partida.

- **Motor Menú**

El motor del menú és el motor més complex. Conté el codi del funcionament que uneix les diferents parts i opcions del joc i deriva o sol·licita la feina als altres TADs. Té accés a la pantalla per mostrar text, al Joc per iniciar-lo, al TAD Hora per obtenir el temps, al TAD Gestor LCD per generar la visualització en marquesina, a Usuaris per gestionar-los i al teclat i al control SMS de forma inversa per obtenir els inputs del teclat directament (nombres) o convertits al format sms (nombres i lletres). El motor té forma de palmera on es bifurca per cada una de les cinc opcions diferents: *new game* per iniciar el joc amb el TAD Joc un cop creat o escollit un nou usuari utilitzant el TAD Usuaris, *modify Users* per modificar-los directament amb el TAD Usuaris i l'entrada del teclat amb sms, *show users* amb l'informació de Usuaris i també per *shop top 5 users* on es converteix la puntuació de char a ascii de tres caràcters i *show time* amb *modify time* per veure i editar l'hora (amb l'entrada del teclat també) amb el TAD Hora que compta el temps en tot moment.

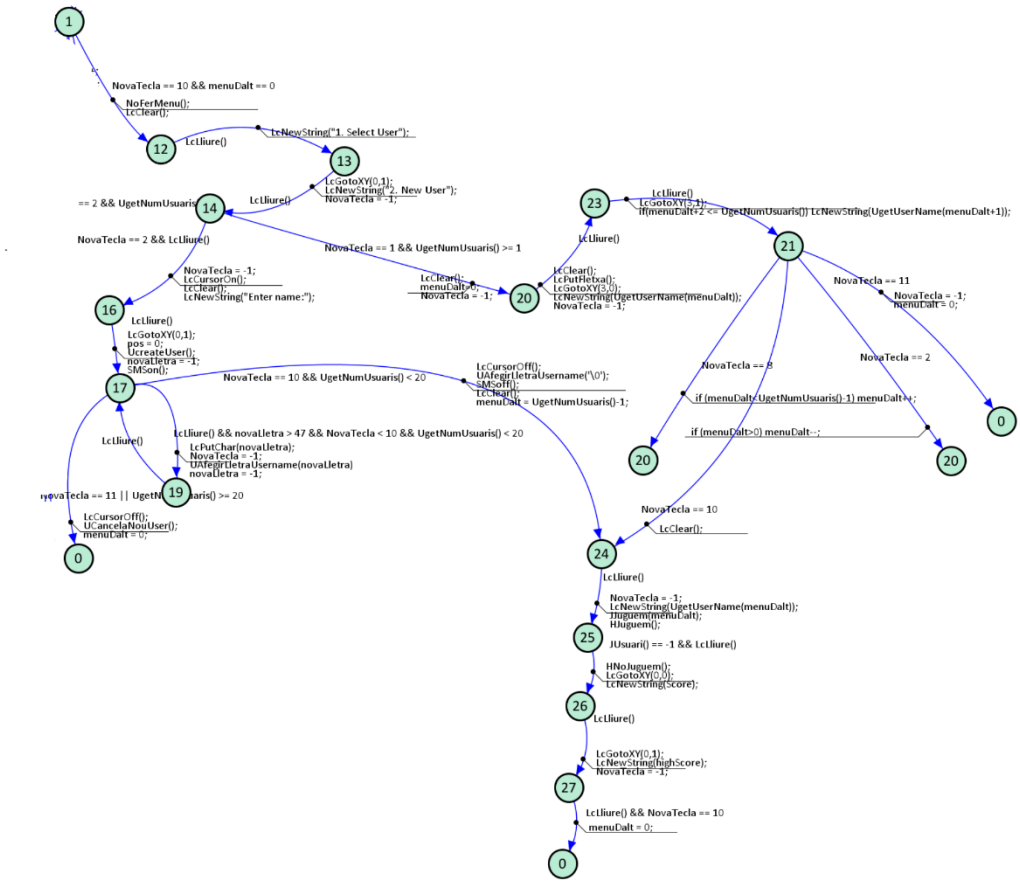
**Estructura principal: selector d'opcions:**




*Il·lustració 9 - Diagrama del motor Menú amb el selector principal d'opcions.*

En aquets primer conjunt es desplaça l'usuari per les diferents opcions de dalt a baix. De l'estat 1 es mou al 0 per desplaçar amb la tecla de pujar o amb la tecla de baixar l'opció llegint "NovaTelca". Amb "menuDalt" se sap l'última opció senyalada per la fletxa de la pantalla i poder sortir cap a una operació o funció.

### Estructura New Game:

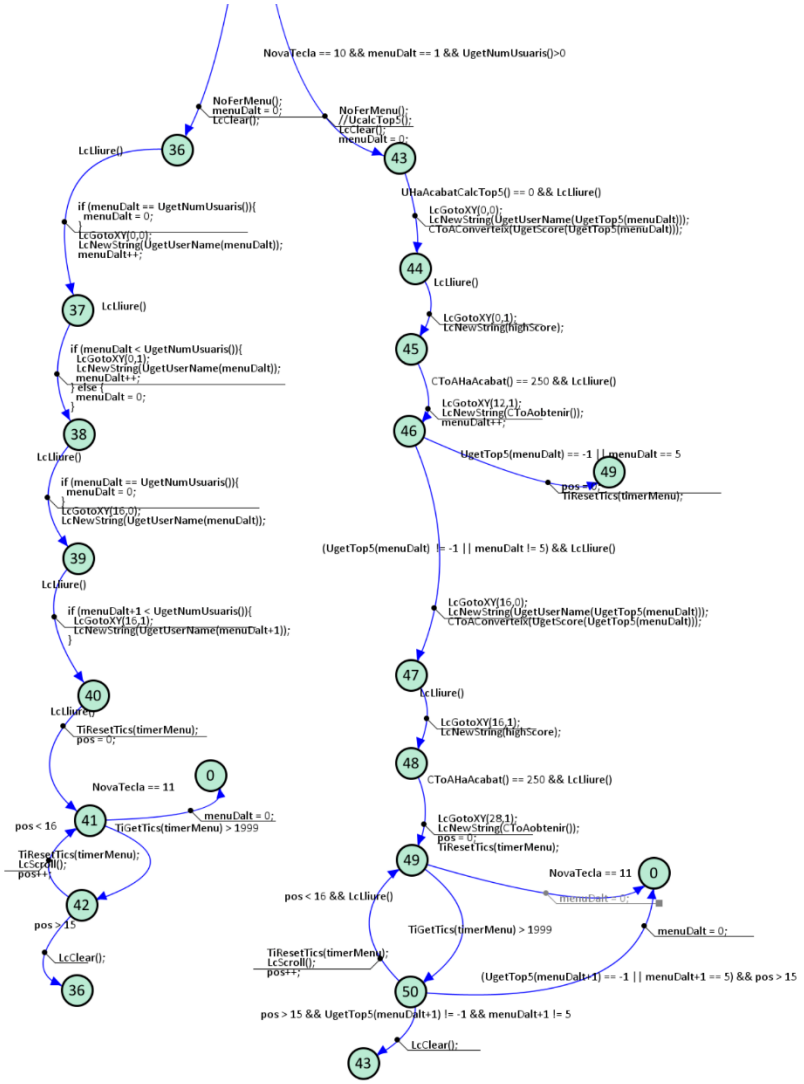



Aquí, l'usuari selecciona crear un nou usuari o escollir-ne un de ja creat, i de la mateixa manera que es poda desplaçar per les opcions del menú, es pot seleccionar un usuari ja creat en el cas que ja n'hi hagin. En cas contrari pot escollir crear un nou usuari i sortir-ne durant la seva creació en el bucle 17-19 cap a l'estat 0 un altre cop o començar a jugar a l'estat 24. Durant el joc, s'espera a que el TAD Joc indiqui que s'ha acabat la partida amb la funció "JUsuari()" quan retorna un -1 en comptes de l´índex de l'usuari.

### Estructura de "Show Top 5 Users" i "Show Users":

Si es selecciona el top 5 primer es demana que es torni a calcular la nova llista de jugadors amb 5 millors puntuacions al TAD Usuaris i llavors es procedeix a escriure'ls a la pantalla dos per línia als píxels visibles i dos als píxels amagats a continuació dels 16 visibles, de manera que quan es sol·liciti un "LcScroll" o "left shift" es faran visibles. Tant si s'estan mostrant la llista d'usuaris com la dels cinc millors amb "Top 5", es carreguen

fins a quatre usuaris i es *shiften* fins obtenir els segons de posicions de píxels superiors a 16 (els dos o un amagats) i llavors es repeteix el carregat dels dos amagats fins acabar de mostrar-los tots o es premi el caràcter "#" per sortir. En el cas de "Show Users" no es surt mai al acabar de mostrar-los sinó que és cíclic. Es pot veure el funcionament al diagrama següent:

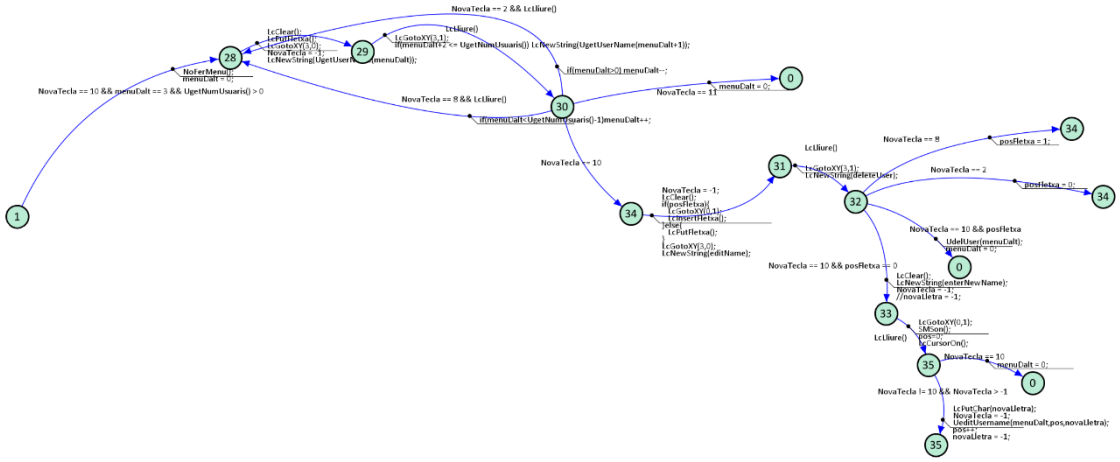



*Il·lustració 10 - Diagrames del motor Menú de Show Users (esquerra) i Show Top 5 Users (dreta). Les dels fletxes superiors provenen de l'estat 1 del "Selector d'opcions" vist en aquest apartat.*

### Modify Users

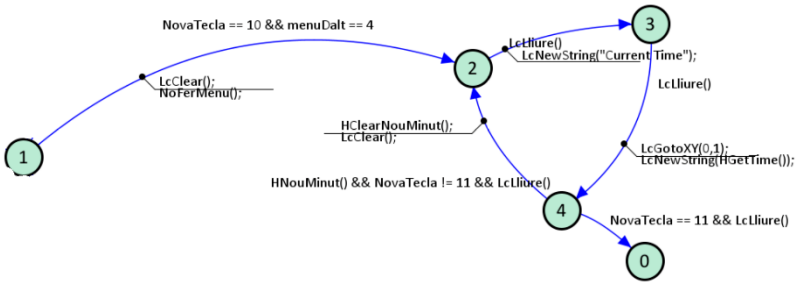
A "Modify Users", si hi ha creat més d'un usuari, es preguntarà quin usuari es vol modificar amb un menú com la llista del menú principal dels estats 0 i 1. A continuació el codi es desvia a l'estat 0 i mostrar el menú si es borra l'usuari havent-lo eliminat a la

l'acció de la transició o es procedeix a escriure un nou nom per l'usuari als estats 33-35 i finalment 0 al acceptar el nou nom. Es pot veure al següent diagrama:



*Il·lustració 11 - Diagrama del motor Menú amb la part de Modify Users.*

### Show Time:



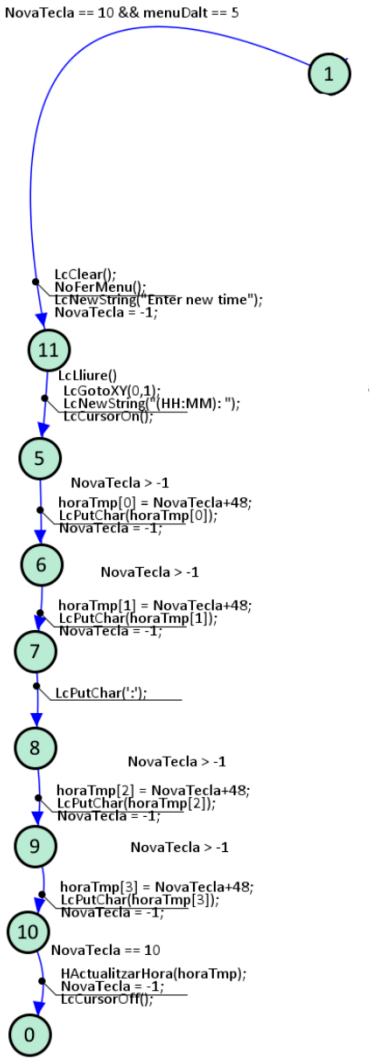

*Il·lustració 12 - Retall del diagrama del motor Menú amb l'estructura de Show Time.*

Quan es seleccioni la opció de veure l'hora, per cada unitat de minut que canvia s'actualitza la pantalla amb l'hora obtinguda del TAD Hora en format *string* i es mostra per pantalla. El TAD Hora funciona de forma asíncrona al motor Menú i per tant amb el cooperativisme del sistema pot avançar l'hora tot i no estar-la mostrant o fent altres activitats al capdavant.

### Modify Time

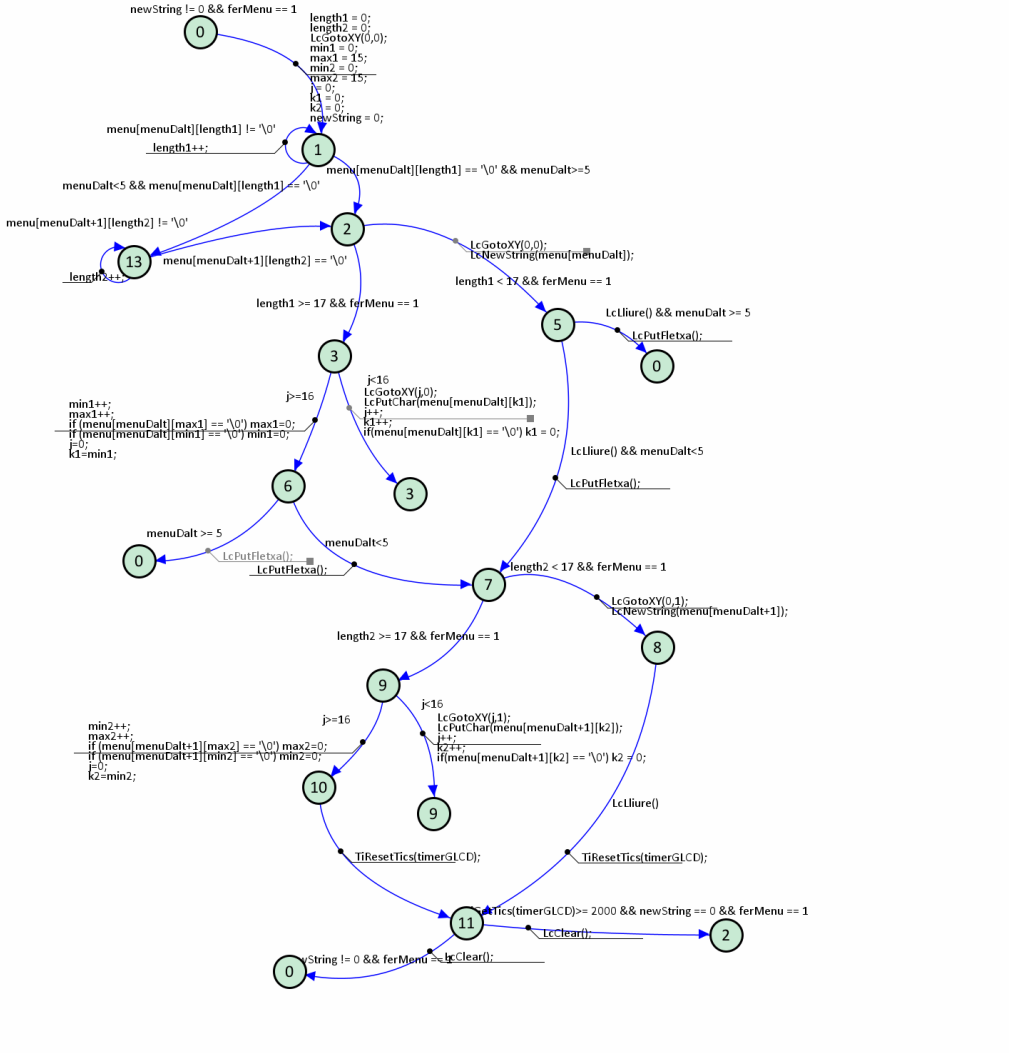
Quan s'executi l'opció de modificar el temps, es demanarà que s'introdueixin les hores i els minuts de l'esquerra fins a la dreta. No es comprovaran si els nombres creen una hora existent (de 00 a 24 i de 00 a 59 o 60 tenint en compte que si les dècimes de l'hora és un 2 només es pot escriure de 0 a 4 0 a 3 depenent de l'estàndard del format i altres

condicions). Això s'ha fet per ordre dels becaris de l'assignatura per estalviar memòria ja que les condicions dels estats ocupen bastants bytes però inicialment s'havia fet el codi per comprovar-ho. L'hora es substitueix al TAD Hora i es continua comptant a partir de l'enviada. A continuació es veu l'ordre de l'execució en el diagrama d'estats:




*Il·lustració 13 - Diagrama d'estats del motor Menú amb el fragment de modificar l'hora.*

- **Motor Gestor LCD**

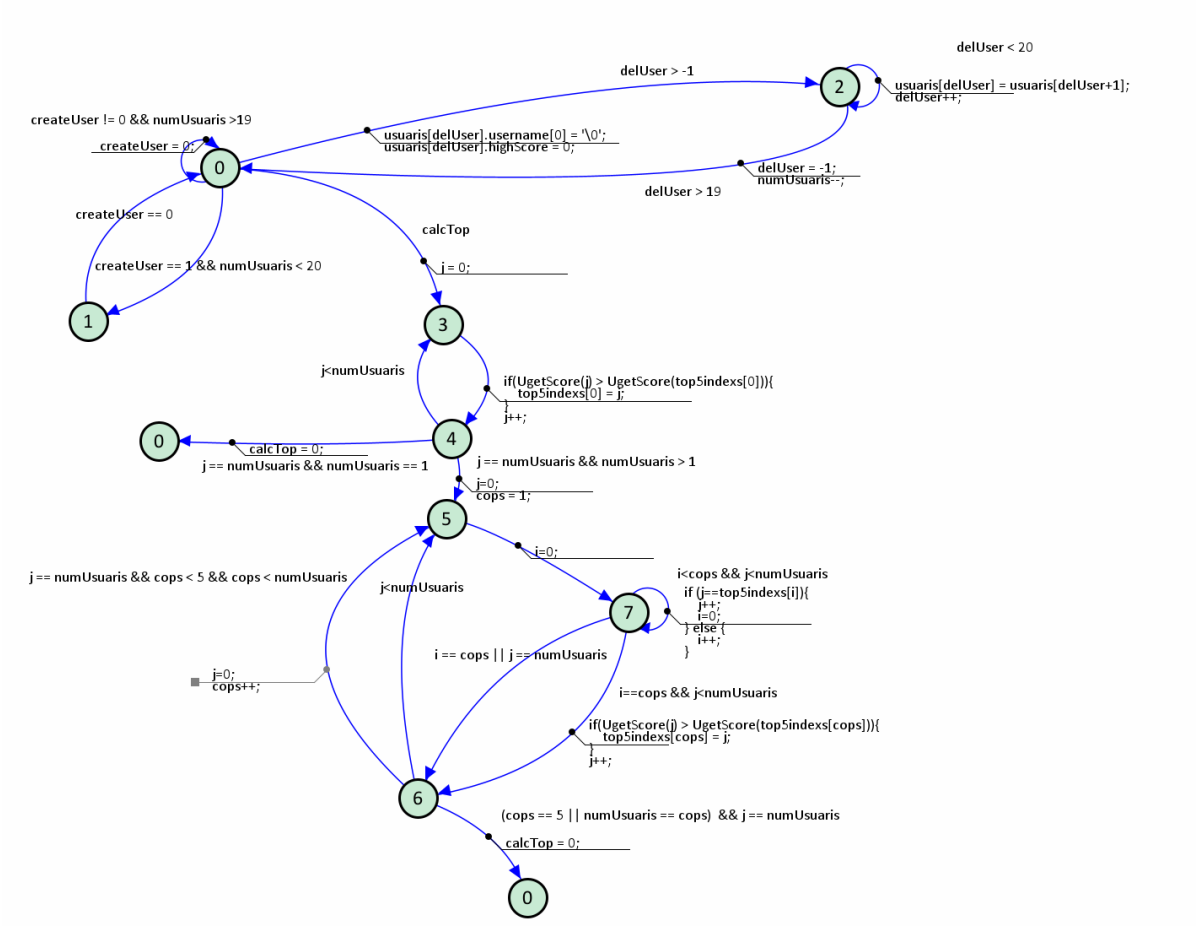



*Il·lustració 14 - Diagrama del motor del TAD GestorLCD.*

Aquest motor s'encarrega de controlar les marquesines del menú principal (i les opcions sense marquesina). Primer de tot, mesura la longitud del text de l'opció per comprovar si necessita realitzar marquesina. En cas que no sigui necessari escriurà l'opció tal com és. Si es necessita, mostrarà caràcter a caràcter i avançarà els límits del text de forma cíclica. Aquest algorisme es repeteix igual per la primera i la segona línia de la pantalla amb l'excepció que si es tracta de la segona i no hi ha una opció més a mostrar, no la mostrarà deixant en blanc tota la segona fila. Tot el motor es controla amb una variable enable o *flag* que s'activa o es desactiva quan s'ha de mostrar el menú principal.

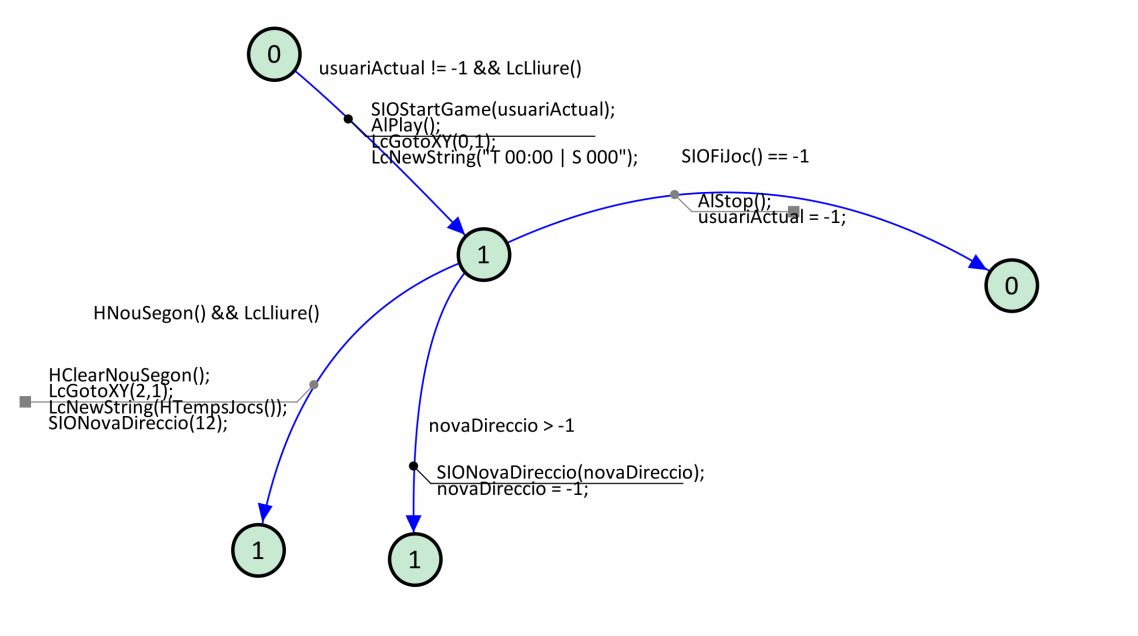
- **Motor Usuari**

El motor del contorl d'usuaris s'utilitza per eliminar o crear nous usuaris. En cas d'eliminar-los, controla que no quedin espais a l'estructura on es guarden en blanc i transporta els ja creats fins a la primera posició lliure de memòria de l'estructura fent que sempre estiguin tots els usuaris posats consecutivament. En cas contrari quan es vulgui crear un nou usuari, comprovarà que hi hagi menys de 20 usuaris i permetrà que una funció construeixi el nou usuari caràcter a caràcter fora del motor.




*Il·lustració 15 - Diagrama del motor del TAD Usuari.*

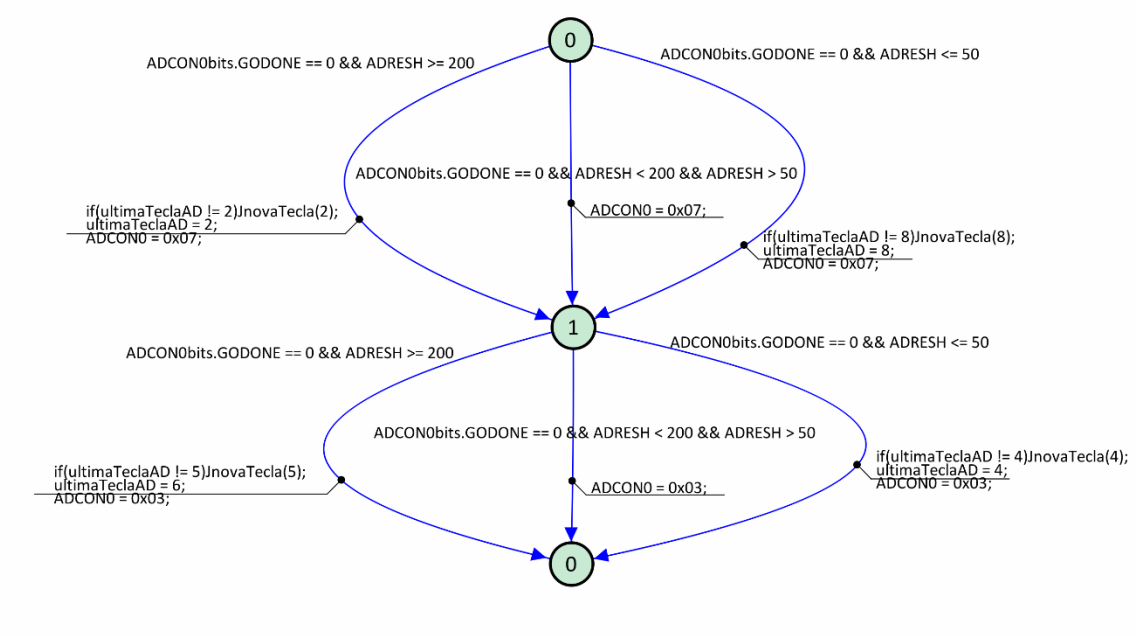
- **Motor Joc**




*Il·lustració 16 - Diagrama del motor del TAD Joc.*

El motor del TAD Joc s'encarrega de controlar l'altaveu i actua com a interfície entra la SIO i els controls de l'usuari: el teclat i el joystick. Quan s'activa el joc insertant un numero d'usuari a "usuari Actual", utilitza aquest usuari per enviar-lo per la SIO i controla la pantalla per mostrar les dades del joc. Cada segon avisa a la SIO que s'ha d'enviar el nou temps i quan es prem una nova tecla o es mou el joystick, una sola variable s'actualitza independentment de quin dels dos components s'ha utilitzat i ho envia per la SIO també. Quan descobreix que la SIO ha rebut una senyal de *game over*, atura l'altaveu que s'havia iniciat al començament del joc i estableix l'usuari actual a un valor fora de límits per permetre el motor i TAD del menú identificar que s'ha acabat la partida.

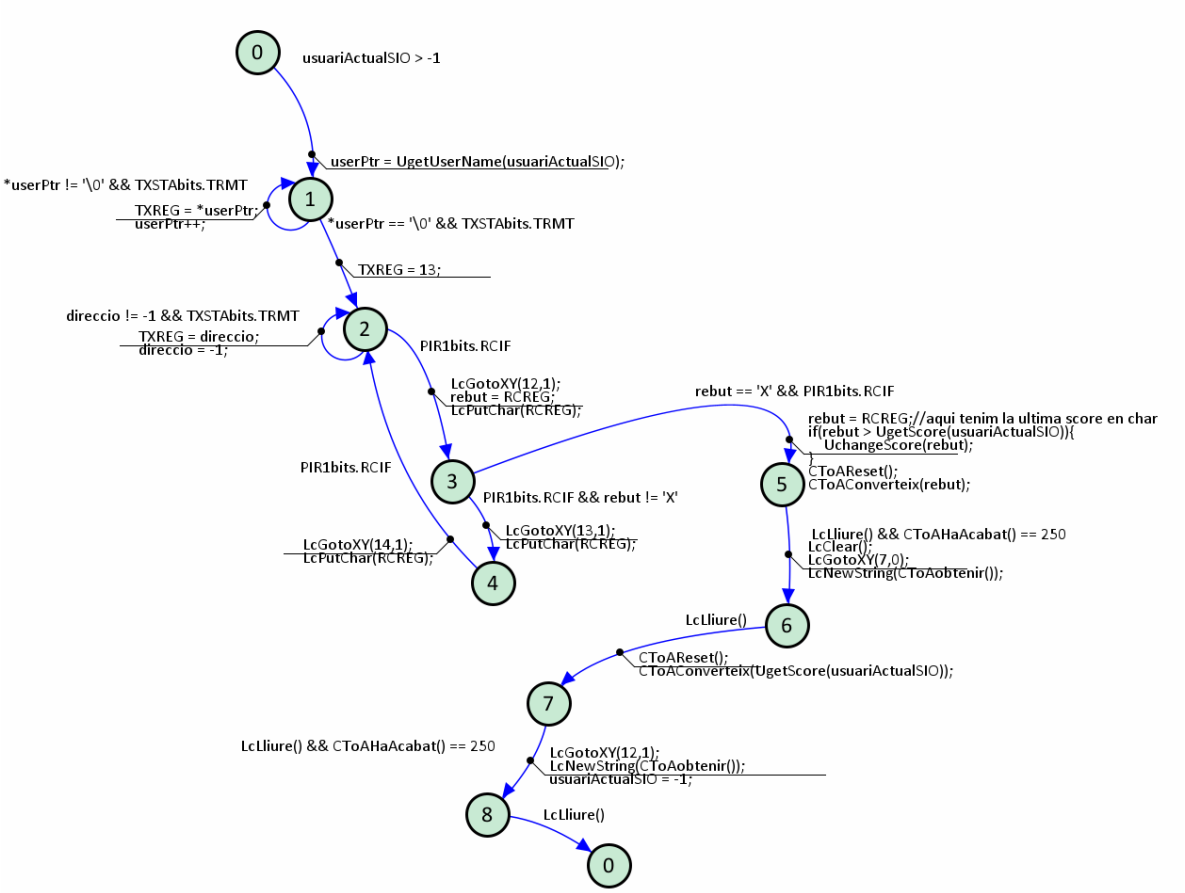
- **Motor JoystickADC**




*Il·lustració 17 - Diagrama del motor del Joystick i ADC.*

El motor del TAD Joystick/ADC està constantment llegint els valors analògics de la X i la Y del joystick, en el cas de que aquest sigui un dels 4 extrems i sigui diferent a l'últim extrem al qual s'ha anat s'avisarà al TAD joc que hi ha una nova direcció. El TAD joc és l'encarregat de saber si s'està jugant i utilitzar la informació o descartar-la. El propi motor del joystick controla si es manté una direcció establerta a varies lectures, evitant que s'enviï varies vegades el mateix valor o direcció.

- **Motor SIO**




*Il·lustració 18 - Diagrama del motor de la SIO.*

Al principi del TAD SIO s'espera a que ens indiquin quin és el usuari amb el qual és jugarà i acte seguit s'envia el seu nom per la SIO i un '\0' que s'utilitza com a stop bit. Mentre s'està jugant és van comprovant dues coses, la primera que no tinguem una nova direcció (en el cas d'haver passat un segon rebrem la direcció 12) en cas de que si aquesta és enviada per la SIO; d'altra banda comprova que no s'hagi rebut res per la SIO, en cas afirmatiu si és una 'X' (stop bit) ens indica que l'usuari a perdut, si és una altre caràcter rebrem tres nombres seguits que serà la puntuació en ASCII i aquests és mostren pel LCD. Quan l'usuari ha perdut, després de la 'X' rebem un byte amb la puntuació final, si aquesta és més alta que la millor del usuari s'actualitza. Per acabar és converteix la puntuació de la partida i la millor puntuació del usuari de binari a ASCII i és mostren pel LCD.

## Conclusions i problemes observats

Amb aquest projecte, s'ha vist molt clar que cal optimitzar molt el codi per estalviar i organitzar la memòria de la PIC per no quedar-se sense a mig projecte. Pot semblar en primer moment que sobrarà o ja serà suficient però amb la indicació de "han de cabre fins a 20 usuaris", s'ha hagut de modificar molt la pràctica i ajornar l'entrega per poder complir aquest requisit.

A més a més, amb la pràctica hem aprés d'una manera més real i pràctica què significa un codi cooperatiu, que utilitzant "motors", per cada execució d'un motor o TAD es va canviat d'estat del motor i no s'executa el codi de forma lineal tasca darrera tasca bloquejant altres funcions simultànies. Es pot dir que és una bona manera de crear un codi "multitasking" que no bloqueja la cooperativitat. Un bon detall per observar si es trenca la cooperativitat és dissenyar i programar l'altaveu amb la música i observar si deixa de funcionar. En el moment que no sona, es pot saber que no s'executa el PWM i no està fent el codi de manera cooperativa.

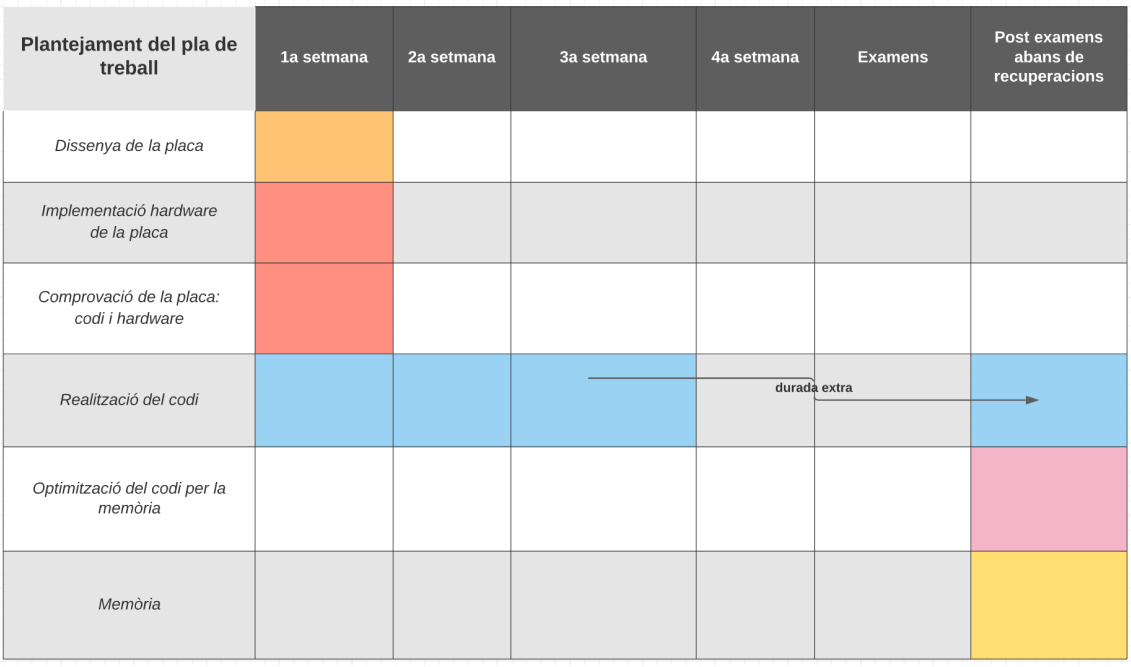
Finalment, s'han recordat conceptes de programació en C i sobretot detalls d'utilització de la PIC que ja s'havien vist a la practica anterior.

Durant el procés de la pràctica, s'han trobar diversos problemes que han atrassat l'entrega: la necessitat de crear 20 usuaris i haver d'optimitzar molt la memòria de la PIC ha sigut el problema principal, però també un primer problema on el Timer0 no activava la interrupció i era deguda a que no es coneixia el funcionament de la plantilla donada del fitxer "main.c" i es va necessitar l'ajuda dels professors de l'assignatura per poder solucionar el problema. Aquest segon problema va fer que no es pogués progressar i finalment al veure que no s'acabava a temps abans d'exàmens es va decidir continuar per la següent entrega i poder estudiar els exàmens i evitar fer pràctica a costa de no estudiar tant les assignatures com al primer semestre, tot i anar atalentats respecte a molts companys amb la primera fase i la implementació física (soldar) d'aquesta segona fase.

Com a possibles millores, es podria optimitzar millor les condicions del Menú (TAD Menú i Motor Menú) per no tornar a comprovar condicions que ja s'hagin trobat en un condicional "else-if" i estalviar memòria flash. També es podrien substituir-les comandes "state = x" per "state++" o "State--" sempre que sigui possible per estalviar una comanda equivalent a assemblar de "movlw x / movwf xxxxx" i utilitzar un "decrement" únicament. Però, com que s'ha tingut en ment que cal optimitzar memòria reaprofitant variables i simplificant el codi al màxim, no ha calgut al finalitzar la pràctica.

## Planificació

Primerament es volia acabar la pràctica abans de l'entrega i es veia possible. Però pel problema relacionat amb el Timer0, no es va poder avançar durant una setmana i per no solapar l'estudi de les assignatures es va decidir continuar durant el període de després d'exàmens i abans de les recuperacions.




*Taula 2 - Diagrama del temps emparat.*

Es pot dir que hi ha hagut un desplaçament del projecte, on durant la segona setmana no es va poder treballar prou i la feina que s'hi hauria d'haver fet es va desplaçar al període posterior. Tot i això s'ha pogut entregar la pràctica finalment amb el següent temps emparat:

Disseny: 1h

Implementació de hardware: 5h

Codi: 140h

Optimització: 1h

Memòria: 4h

Temps total: 152h.