Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2024-2025

Pràctica 1

LSMusical

	Login	Nom
Alumnes	jesus.gascon	Jesús Gascón Cesari
	alberto.marquillas	Alberto Marquillas Marsà

	Placa	Memòria	Nota
Entrega			

Data	09/01/2025

Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2024-2025

Pràctica 1

LSMusical

	Login	Nom
Alumnes	jesus.gascon	Jesús Gascón Cesari
	alberto.marquillas	Alberto Marquillas Marsà

	Placa	Memòria	Nota
Entrega			

Data	09/01/2025

Índex

1. Síntesi de l'enunciat

La segona fase de la pràctica consisteix en implementar la dinàmica del joc utilitzant el microcontrolador PIC18F4321 programat en llenguatge assembler. Aquesta fase gestiona les entrades i sortides del sistema per assegurar que el joc funcioni correctament.

Quan el jugador prem el botó de *StartGame*, el sistema comença a mostrar la seqüència de notes prèviament emmagatzemades en la fase 1. Per cada nota, es mostra el seu valor en un display de 7 segments (*CurrentNote[7..0]*) i la seva durada mitjançant LEDs (*Length[1..0]*). A més, es mesura la distància entre la mà del jugador i el sensor ultrasònic, reproduint la nota corresponent mitjançant l'altaveu (*Speaker*). Cada distància mesurada està associada a una nota musical específica (per exemple, de 0 a 10 cm, la nota Do).

El joc comprova si la nota tocada pel jugador coincideix amb la que es mostra en el display. Si coincideixen durant 500 ms, la nota es considera correcta, s'activa un LED verd (*AnswerCorrect*) i el joc passa a la següent nota. Si no coincideix, el sistema dona la nota per incorrecta i activa el LED vermell (*AnswerIncorrect*). La puntuació es calcula en temps real mitjançant un servomotor (*GameScore*), que mostra el percentatge d'encerts respecte a les notes totals de la melodia.

Una vegada totes les notes de la melodia s'han reproduït, el joc es bloqueja i es manté la puntuació final fins que es reinicií el joc mitjançant el polsador de reset (*PCI*). Durant tot el joc, és important que els components, com el sensor ultrasònic HC-SR04 i el servomotor, estiguin correctament configurats per evitar interferències. També es farà servir una font d'alimentació externa per als servomotors per evitar que afectin al microcontrolador.

Per lliurar aquesta fase, els estudiants han de proporcionar un fitxer .zip que inclogui la memòria del projecte, fotos de la placa, un vídeo demostratiu del funcionament del sistema i els arxius de codi en MPLAB X. Aquesta fase permet als alumnes desenvolupar habilitats en la programació de microcontroladors i la gestió de perifèrics per a crear una experiència de joc interactiva.

2. Disseny del software

2.1. Configuració dels ports

PORT A

El port A consta de 6 pins analògics(RA0, RA1, RA2, RA3, RA4 i RA5) i dos pins d'oscil·lador (RA6 i RA7).

Pins analògics:

- RA0:

Pin de sortida de Length_0 (LED). Inicialitzat a 0.

- RA1:

Pin de sortida de Length_1 (LED). Inicialitzat a 0.

- RA2:

Pin de sortida del Trigger (Ultrasons). Inicialitzat a 0.

- RA3:

Pin de sortida de Answer Correct (LED RA3). Inicialitzat a 0.

- RA4:

Pin de sortida de Answer Incorrect (LED RA4). Inicialitzat a 0.

- RA5:

Pin de sortida de GameScore (Servo). Inicialitzat a 0.

Pins d'oscilador:

Els pins RA6 i RA7 s'utilitzen per a ala configuració de l'oscil·lador HSPLL.

PORT B

El port A consta de 5 pins analògics(RB0, RB1, RB2, RB3 i RB4).

Pins d'interrupció:

- RB0:

Pin d'entrada de NewNote (senyal provinent de la Fase 1).

- RB1:

Pin d'entrada de StartGame (senyal provinent de la Fase 1).

- RB2:

Pin d'entrada del Echo (Ultrasons).

PORT C

El port C consta de 8 pins analògics(RC0, RC1, RC2, RC3, RC4, RC5, RC6 i RC7).

Pins analògics:

- RC0:

Pin d'entrada de Note_0 (senyal provinent de la Fase 1).

- RC1:

Pin d'entrada de Note_1 (senyal provinent de la Fase 1).

- RC2:

Pin d'entrada de Note_2 (senyal provinent de la Fase 1).

- RC3:

Pin d'entrada de Note_3 (senyal provinent de la Fase 1).

- RC4:

Pin d'entrada de Duration_0 (senyal provinent de la Fase 1).

- RC5:

Pin d'entrada de Duration_1 (senyal provinent de la Fase 1).

- RC6:

Pin de sortida de la senyal ACK (amb la Fase 1 com a destí). Inicialitzat a 0.

- RC7:

Pin de sortida del Speaker. Inicialitzat a 0.

PORT D

EL port D es un port de sortida que s'encarrega de la senyal CurrentNote[7..0] (7-seg). Inicialitzat a 0.

2.2. Variables

- FLAGS:

Posició 0x0001. En aquesta variable guardarem tots els flags per avisar a les interrupcions. El bit 0 l'utilitzarem per indicar quan el joc ha començat (StartGame és actiu). El bit 1 indicarà que s'han de comptar tres segons. El bit 2 d'aquesta variable l'utilitzarem per activar el procés del servo. El bit 3 l'utilitzarem per activar el procés de l'altaveu. Finalment el bit 4 d'aquesta variable l'utilitzarem per indicar quan s'han de comptar 500ms. Tots aquests bits estaran inicialitzats a 0.

- NUM_NOTES:

Posició 0x0002. Variable inicialitzada a 0.

- LECTURA:

Posició 0x0003. Variable inicialitzada a 0.

- COMT_ESPERA_ACK:

Posició 0x0004. Aquesta variable la utilitzarem com a comptador per a generar la senyal ACK. Variable inicialitzada a 0.

- NOTA LLEGIDA:

Posició 0x0005. Variable inicialitzada a 0.

- DURATION_LLEGIDA:

Posició 0x0006. Variable inicialitzada a 0.

- COMPTADOR_3S_L:

Posició 0x0007. Variable inicialitzada a 0.

- COMPTADOR_3S_H:

Posició 0x0008. Variable inicialitzada a 0.

- FLAGS_INT:

Posició 0x0009. En aquesta variable guardarem tots els flags que vinguin de les interrupcions. El bit 0 d'aquesta variable indicarà quan s'han comptat tres segons. El bit 1 indica quan s'ha comptat la duració. El bit 3 indica quan s'han comptat 500ms. Tots aquest bits estaran inicialment inicialitzats a 0.

- COMPTADOR_500MS_L:

Posició 0x000A. Variable inicialitzada a 0.

- COMPTADOR_500MS_H:

Posició 0x000B. Variable inicialitzada a 0.

- COMPT_ESPERA_TRIGGER:

Posició 0x000C. Variable inicialitzada a 0.

- COMPTADOR_DURATION_H:

Posició 0x000D. Variable inicialitzada a 0.

- COMTADOR_DURATION_L:

Posició 0x000E. Variable inicialitzada a 0.

- COMPT_ACK:

Posició 0x000F. Variable inicialitzada a 0.

- COMPT_CM:

Posició 0x0010. Variable inicialitzada a 0.

- COMPT_ESPERA_60US:

Posició 0x0011. Variable inicialitzada a 0.

- NOTA_TROBADA:

Posició 0x0012. Variable inicialitzada a 0.

- COMPTADOR_2S_L:

Posició 0x0013. Variable inicialitzada a 0.

- COMPTADOR_2S_H:

Posició 0x0014. Variable inicialitzada a 0.

- COMPTADOR_TRIGGER:

Posició 0x0015. Variable inicialitzada a 0.

- TICKS_ALTAVEU:

Posició 0x0016. Variable inicialitzada a 0.

- PWM_ALTAVEU:

Posició 0x0017. Variable inicialitzada a 0.

- DC_ALTAVEU:

Posició 0x0018. Variable inicialitzada a 0.

- NOTES_CORRECTES_AUX:

Posició 0x0019. Variable inicialitzada a 0.

- COMPT_ESPERA_11US:

Posició 0x0020. Variable inicialitzada a 0.

- GARUS_AUX:

Posició 0x0021. Variable inicialitzada a 0.

- GRAUS:

Posició 0x0022. Variable inicialitzada a 0.

- INTERRUPCIONS_FLASH:

Posició 0x0023. Variable inicialitzada a 0.

- NOTES_CORRECTE:

Posició 0x0024. Variable inicialitzada a 0.

- NOTES_CORRECTES:

Posició 0x0025. Variable inicialitzada a 0.

- QUANTITAT_GRAUS:

Posició 0x0026. Variable inicialitzada a 0.

- NUM_NOTES_AUXILIAR:

Posició 0x0027. Variable inicialitzada a 0.

- FI_20MS:

Posició 0x0028. Variable inicialitzada a 0.

2.3. Funcions

2.3.1. Funcions inicialització de comptadors

Inicialitzar comptador de 500ms

INIT_COMPT_500MS

- Interrupcions de 500u per arribar a 500ms:

$$\frac{500ms}{500us} = 1000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_500MS_H i COMPTADOR_500MS_L)
- 1000d = 0011 1110 1000 b
- Quedaran inicialitzats a 1000

Inicialitzar comptador de 1s per la duració

INIT_COMPT_1S_DURATION

- Interrupcions de 500u per arribar a 1s:

$$\frac{1s}{500us} = 2000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_DURATION_H i COMPTADOR_DURATION_L)
- Quedaran inicialitzats a 2000

Inicialitzar comptador de 2s per la duració

INIT_COMPT_2S_DURATION

- Interrupcions de 500u per arribar a 2s:

$$\frac{2s}{500us} = 4000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_DURATION_H i COMPTADOR_DURATION_L)
- Quedaran inicialitzats a 4000

Inicialitzar comptador de 3s per la duració

INIT_COMPT_3S_DURATION

- Interrupcions de 500u per arribar a 3s:

$$\frac{3s}{500us} = 6000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_DURATION_H i COMPTADOR_DURATION_L)
- Quedaran inicialitzats a 6000

Inicialitzar comptador de 2s

INIT_COMPT_2S

- Interrupcions de 500u per arribar a 2s:

$$\frac{2s}{500us} = 4000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_2S_H i COMPTADOR_2S_L)
- Quedaran inicialitzats a 4000

Inicialitzar comptador de 3s

INIT_COMPT_3S

- Interrupcions de 500u per arribar a 3s:

$$\frac{3s}{500us} = 6000 \text{ interrupcions}$$

- Utilitzarem dos registres 65535 (COMPTADOR_3S_H i COMPTADOR_3S_L)
- Quedaran inicialitzats a 6000

2.3.2. Funcions comptadors

Funció que compta 2ms per a la senyal ACK

COMPTA_ACK

- Espera 2ms

$$\text{Tosc} = \frac{1}{40M} = 25\text{ns}$$

$$Cicle\ màquina\ 4 \times Tosc\ = 4 \times 25ns = 100ns$$

$$Cicles = \frac{2ms}{100ns} = 20000\ (20\ repeticions\ de\ 1000\ cicles)$$

- Utilitzarem un bucle per a fer tots els cicles necessaris per a esperar la senyal ACK.

Funció que compta 3 segons

COMPTAR_3S

- Va decreixent les variables COMPTADOR_3S_L i COMPTADOR_3S_H fins que estan a zero.
- Seguidament activa el LED incorrecte, desactiva 500ms i apaga el LED correcte.
- Finalment posa a 1 el bit 0 de la variable Flags_Int, indicant que ja ha comptat 3s. També posa a 0 el flag de comptar 3 segons.

Funció que compta 500ms

COMPTAR_500MS

- Va decreixent les variables COMPTADOR_500MS_L i COMPTADOR_500MS_H fins que estan a zero.
- Un cop passats 500ms hi ha tres opcions:
    - 500ms, comencem duració
    - 3s & !500ms, comptador de 3s (fet)
    - Nota incorrecte durant 500ms, ultrasons para el comptatge
- Seguidament posa a 0 el flag de 3 segons i activa TM0 duració.
- Crida a les funcions d'inicialitzar comptadors d'1s, 2s i 3s
- Finalment posa a 1 el bit 1 de la variable Flags_Int, indicant que ja ha comptat 500us i posa a 0 el flag que indica que s'han de comptar 500ms.

Funció que compta 500ms

COMPTAR_DURATION

- Va decreixent les variables COMPTADOR_DURATION_L i COMPTADOR_DURATION_H fins que estan a zero.
- Un cop ha passat duration activa el LED correcte, incrementa la variable NOTES_CORRECTES.
- Finalment posa a 1 el bit 1 de la variable Flags_Int, indicant que ja ha comptat duration.

2.3.3. Funcions d'esperes

Funció per esperar 60 us

ESPERA_60US

- S'utilitza per comptar 1 centímetre, es compten 60 us perquè cada cm son uns 60us.

$$Velocitat\ del\ so = \frac{343m}{s}$$

$$Dist = velocitat\ \times temps = \frac{343m}{s} * temps(s)$$

- L'ultrasons retorna la distància calculada d'anada i de tornada, per tant la distancia haurà de ser dos cops dist.

$$dist = \frac{velocitat * temps}{2} = \frac{\frac{343m}{s} * t(s)}{2}$$

- Si dist = 1 cm = 0.01m:

$$t = \frac{0.01 \times 2}{343} = 58.3us \sim 60us$$

- Ara hem de calcular quants cicles son necessaris per fer 60us:

$$Tosc = \frac{1}{40M} = 25ns$$

$$Cicle\ màquina = 4 \times Tosc = 4 \times 25ns = 100ns$$

$$Cicles = \frac{60us}{100ns} = 600$$

- Utilitzant un bucle esperem tots el cicles calculats prèviament i aconseguim generar una espera de 60us.

Funció d'espera per la generació del trigger

ESPERA_GENERAR_TRIGGER

$$Tosc = \frac{1}{40M} = 25ns$$

$$Cicle\ màquina = 4 \times Tosc = 4 \times 25ns = 100ns$$

$$Cicles = \frac{10us}{100ns} = 100$$

- Utilitzant un bucle esperem rots els cicles calculats prèviament i aconseguim generar una espera de 10us (espera per la generació del trigger).

Funció d'espera per a que es mostri bé el LED incorrecte

- S'utilitza la INIT_COMPT_2S per inicialitzar les variables (COMPTADOR_DURATION_L i COMPTADOR_DURATION_H) per poder comptar 2 segons.
- S'utilitza un bucle per decrementar les variables per arribar a esperar 2 segons

2.3.4. Funcions per el final de l'execució

Funció amb un bucle infinit amb el final

MOSTRA_FINAL

- Aquesta funció desactiva l'altaveu i l'apaga.
- Seguidament entra en un bucle que s'encarrega de mostrar el final

2.3.5. Funcions per nota OK

Funció en cas de que la nota sigui la correcta

NOTA_IGUALS

- Si no està comptant 500ms i no esta comptant duració, comença a comptar 500ms.
- Activa el flag de comptar duració i activa el flag de comptar 500ms en un bucle que també inicialitza el comptador de 500ms.

2.3.6. Funcions per nota KO

Funció que s'executa en el cas de fallar de nota

NOTA_KO

- Si està comptant 500ms, para el comptador de 500ms.
- Si està comptant la duració, para el comptador de la duració i activa el LED incorrecte i va a la següent nota (activa flag_int).

2.3.7. Funcions ultrasons

Funció per generar trigger per l'ultrasons

GENERAR_TRIGGER

- Activa el bit 2 del port A, on està connectat el trigger.
- Crida a la funció ESPERAR_TRIGGER per esperar-se 10us.

2.3.8. Funcions per START GAME

Funcions de reproducció de notes

NOTA_X_ULTRASONS

- X amb valors del 0 al 6 inclosos.
- S'encarrega de la reproducció per l'altaveu de la corresponent nota trobada.

NOTA_ERRONIA

- EN cas que la nota trobada sigui errònia, apaga la interrupció de l'altaveu i seguidament apaga l'altaveu.

Funció per activar la interrupció de l'altaveu

ACTIVA_ALTAVEU

- Posa a 1 el bit 3 de la variable FLAGS (flag d'interrupció de l'altaveu).

Funció per trobar la nota retornada per l'echo

TROBAR_NOTA

- Mesura la distància amb l'ultrasons
- Compara la distància amb les diferents notes
- Verifica que la nota detectada es correcte
- Depenen si es correcte o no, crida a la funció NOTA_IGUALS o NOTA_KO, respectivament.

Funció per esperar el temps per detectar la nota

ESPERAR_TEMPS

- Gestiona el temps d'espera per detectar una senyal de la nota detectada.
- Comprova si han passat 3 segons sense detectar-se res.
- En el cas de que s'hagi detectat, comprova la duració en la que s'ha detectat.
- Comprova si durant la duració que s'ha detectat la mà, la mà s'ha mogut (duració incorrecte).
- Genera la senyal del trigger amb la funció GENERAR_TRIGGER, espera el echo, el processa i al acabar el processament posa a 0 el flag corresponent per poder tornar a comptar.

Funció per mostrar la nota pel 7-segments

MOSTRAR_NOTA

- Compara la nota llegida amb un valor del 0 al 6 i en funció de la nota, crida a la seva funció corresponent per l'encès dels corresponents LEDs del 7-segments.

Funció per mostrar la duració pels LEDs

MOSTRAR_LEDS

- Aplica una mascara per quedar-se amb la duració.
- Comprova els valors del LED, depenen del valor cridarem a la funció corresponent per a l'encès del LEDs respectius.

Funció per encendre el 7-segments amb la combinació corresponent a la nota

NOTA_X

- X pren valors del 0 al 7 inclosos.
- Cada funció carrega la combinació per mostrar el numero desitjat per el 7-segments al port de sortida.
- Finalment torna a l'execució de StartGame per mirar els LEDs.

Funció per a controlar la duració dels LEDs

LED_X

- X pren valors del 1 al 3 inclosos.
- Cada funció activa length_0 i desactiva length_1 i torna a l'execució de StartGame.

Funció per comprovar si hi ha notes a la RAM

COMPROVAR_NOTES

- Verifica si NUM_NOTES es igual a zero, si ho es, significa que no hi ha notes emmagatzemades a la RAM i el programa no realitzarà mes operacions, saltant al final del procés. En canvi, si no es zero, el programa seguirà executant-se amb normalitat.

Funció StartGame

START_GAME

- Comprova si hi ha notes.
- Llegeix la posició, mostra la nota al 7-segments i mostra la duració als LEDs.
- Compta tres segons amb el TMR0, indicat amb un flag.
- Comprova si hi ha notes.

2.3.9. Funcions NEW NOTE

Funció d'espera per l'ACK

ESPERA_ACK

- Espera 2 cicles per a que la fase 1 detecti la senyal ACK.

Funció per activar ACK

ACTIVAR_ACK

- Activa l'ACK (posa a 1 la sortida RC6).
- Crida a la funció d'espera de l'ACK.
- Desactiva l'ACK (posa a 0 la sortida RC6).

Funció per guardar la nota a la RAM

GUARDAR_NOTA_RAM

- Mou el punter d'escriptura de la RAM per escriure una nota nova.
- Màscara per guardar Nota[3..0] : Duration[1..0]
- Multiplica la mascara pel PORT C i guarda el resultat al work.
- Guarda la nota i escriu a la RAM.

2.3.10. Funcions d'interrupcions

Funció del servomotor

SERVO

- Controla el funcionament del servo
- Compta les interrupcions.
- Cada 20ms fa una actualització i desactiva el flag.

ESPERA_SERVO

- Espera 0,5ms
- Verifica si hi ha notes correctes, en cas de que no hi hagi, termina la subrutina.

ESPERA_05MS

- Espera 0,5ms:

$$Tosc = \frac{1}{40M} = 25ns$$

$$Cicles\ màquina = 4 \times Tosc = 4 \times 25ns = 100ns$$

$$Cicles = \frac{0.5ms}{100ns} = 5000$$

- Com que 5000 és molt,

$$\frac{0.1ms}{100ns} = 1000$$

- Fa 5 vegades 1000 cicles amb la funció COMPTA_1000_CICLES per fer 5000 cicles en total.

STEP

- Cada 11us augmenta un grau.
- Crida a la funció ESPERA_11us en un loop que va decrementant els graus depenent del temps que passi.

ESPERA_11us

- Espera 11us:

$$Tosc = \frac{1}{40M} = 25ns$$

$$Cicles\ màquina = 4 \times Tosc = 4 \times 25ns = 100ns$$

$$Cicles = \frac{11us}{100ns} = 110$$

Funció per processar NEWNOTE

NEW_NOTE

- Neteja el flag de la interrupció.
- Guarda la nota a la RAM cridant a la funció GUARDAR_NOTA_RAM.
- Incrementa el numero de notes.
- Activa ACK cridant a la funció ACTIVAR_ACK.

Funció per controlar la interrupció de StartGame

ACTIVAR_START_GAME

- Neteja el flanc d'interrupció.
- Activa el flag per començar el joc.

Funció d'acció del TMR0

ACTION_TMR0

- Neteja el flag i carrega el valor de TMR0 de nou.
- Mira el flag de comptar 3 segons i en cas de que sigui necessari, crida a la funció COMPTAR_3S.
- Mira el flag d'activar el servo i en cas de que sigui necessari, crida a la funció SERVO.
- Mira el flag d'activar l'altaveu i en cas de que sigui necessari, crida a la funció ALTAVEU.
- Mira el flag de comptar 500ms i en cas de que sigui necessari, crida a la funció COMPTAR_500MS.
- Mira el flag de comptar duració i en cas de que sigui necessari, crida a la funció COMPTAR_DURATION.
- Mira el flag del trigger i en cas de que sigui necessari, crida a la funció ESPERA_TRIGGER.

2.3.11. Funcions Principals

Funcions d'interrupcions d'alta prioritat

HIGH_RSI

- Mira el flag de TMR0 i en cas de que sigui necessari, crida a la funció ACTION_TMR0.
- Mira el flag de NewNote i en cas de que sigui necessari, crida a la funció NEW_NOTE.
- Mira el flag de StartGame i en cas de que sigui necessari, crida a la funció ACTIVAR_START_GAME.

INIT_FLASH

- Inicialitza la flash.

VES_FLASH

- Configura el punter per accedir als continguts de la flash.

Funció MAIN

MAIN

- Configura els VO ports(CONFIG_PORTS).
- Inicialitza O ports i variables(INIT_PORTS, INIT_VARS).
- Configura Interrupcions(CONFIG_INTERRUPTS).
- Inicialitza els punters de la RAM(PUNTER_RAM).

- Inicialitza TMR0 per interrompre(INIT_TMR0).
- Carrega TMR0 per generar la interrupció(CARREGA_TMR0).
- Posiciona el punter de lectura de la RAM a la primera posició.
- Entra en un bucle amb la funció START_GAME a dins.

3. Configuracions del microcontrolador

CONFIG OSC = HSPLL ; 40MHz

CONFIG PBADEN = DIG ; PORTB

CONFIG WDT = OFF

Configuració de les interrupcions

Per a la configuració de les interrupcions utilitzarem la funció CONFIG_INTERRUPTS, aquesta funció estarà a dins del main. Dins de la funció ens trobarem el següent:

Per activar les interrupcions posem a 1 el bit 7 del registre INTCON (unmasked) i el bit 6 d'aquest mateix registre (high).

Per activar la interrupció a RB0 i indicar que es per flanc de baixada (la senyal NewNote), posarem a 1 el bit INT0IE del registre INTCON (activo interrupció RB0) i posarem a 0 el bit INTEDG0 del registre INTCON2 (indico que és per flanc de baixada).

Per activar la interrupció a RB1, posarem a 1 el bit INT1IE del registre INTCON3.

Finalment per activar la interrupció de TMR0, posarem a 1 el bit TMR0IE del registre INTCON.

Inicialització de la RAM

Per inicialitzar la RAM utilitzarem la funció PUNTER_RAM, aquesta funció estarà a dins del main. Dins d'aquesta funció ens trobarem el següent:

Carregarem un 1d al registre WORK per posteriorment carregar-lo al registre FSR0H, d'aquesta manera queda inicialitzat al banc 1 el punter d'escriptura de la RAM. Posarem a 0 el registre FSR0L per inicialitzar el punter a l'adreça 0.

Carregarem el 1d del WORK al registre FSR1H per a que quedi inicialitzat al banc 1 el punter de lectura de la RAM. Posarem a 0 el registre FSR1L per inicialitzar el punter a l'adreça 0.

Inicialització i càrrega de TMR0

Per inicialitzar TMR0 utilitzarem la funció INIT_TMR0, aquesta funció estarà a dins del main. Dins d'aquesta funció ens trobarem el seguent:

Carregarem la combinació de bits '10001000' al WORK per posteriorment inicialitzar el registre T0CON amb aquesta combinació de bits. El bit 7 (1) correspon als enables de TMR0, el bit 6 (0) correspon al timer 8/16 bit, el bit 5 (0) correspon al CLKO, el bit 4 (0) correspon al High-To-Low, el bit 3 (1) correspon a "Not prescale output", i finalment els bits [2..0] (tots 0) corresponen al "prescale value".

Per carregar el valor de TMR0 utilitzarem la funció CARREG_TMR0, aquesta funció estarà dins del main. Dins d'aquesta funció ens trobarem el següent:

Carregarem 60536d a TMR0H (high) i a TMR0L(low). Aquest valor el treiem de:

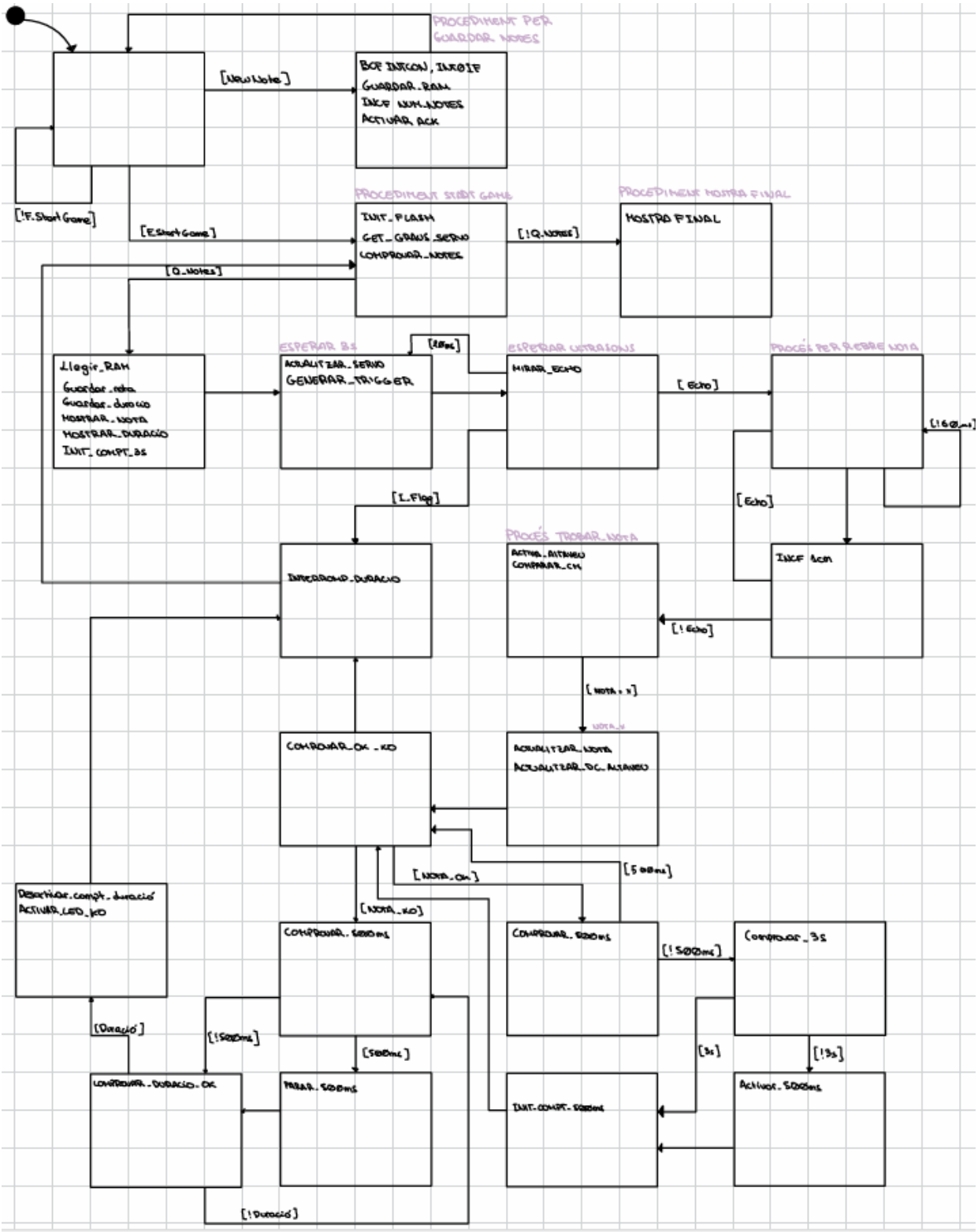
$$Tint = \left(\frac{4}{Fosc}\right) \times Preescarler \times \left(2^{bits} - L\right)$$

$$Preescaler = 246, \quad Bits = 16$$

$$500us = \left(\frac{4}{40M}\right) \times 256 \times (2^{16} - L)$$

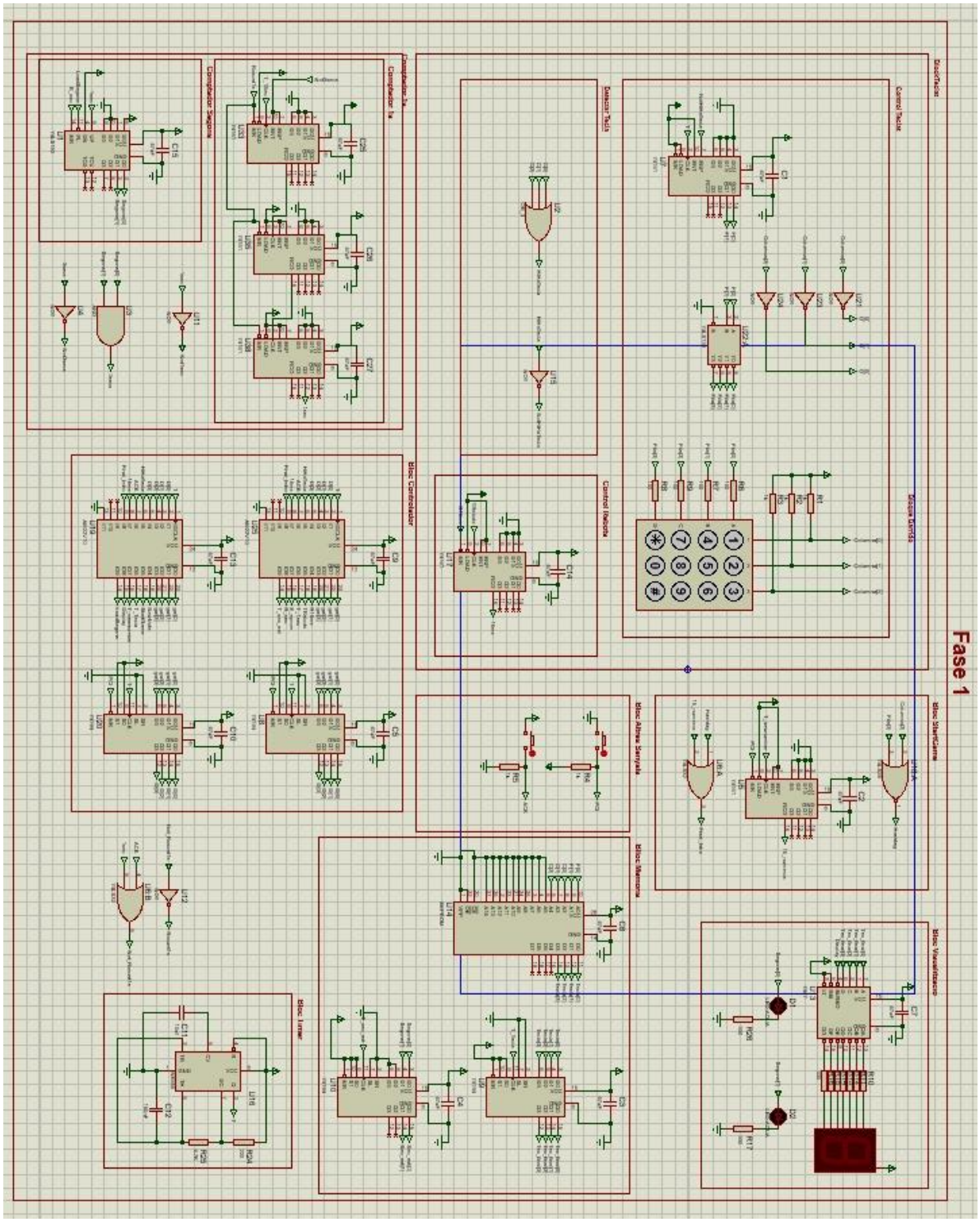
$$L = 60536$$

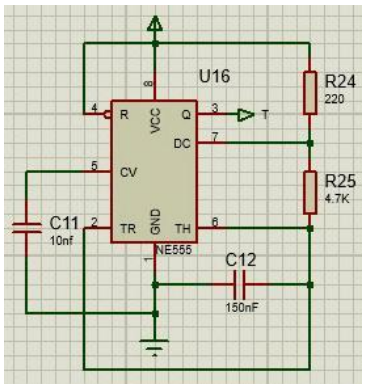
4. Diagrama d'activitats del software



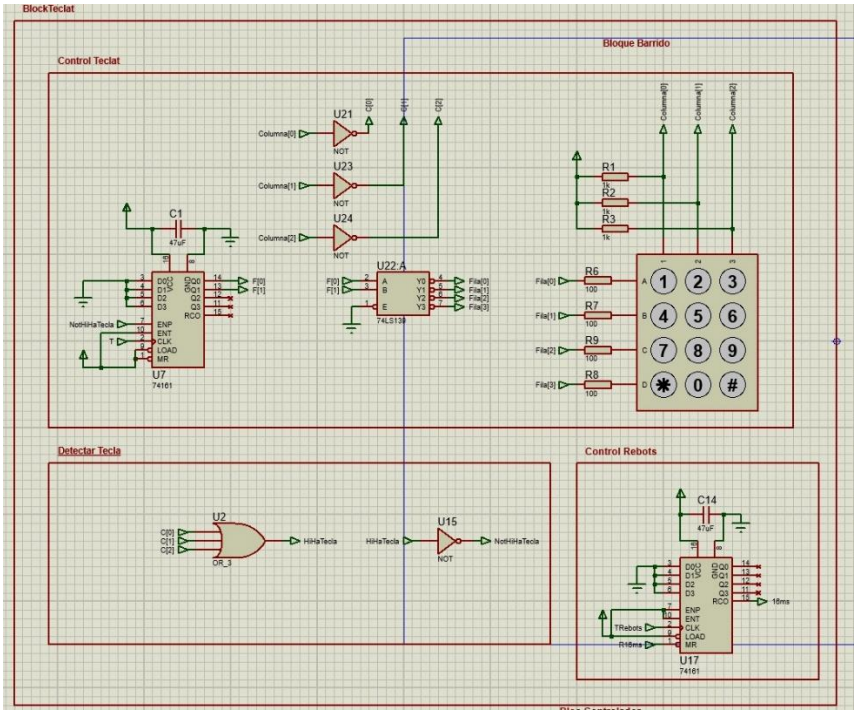
5. Esquema elèctric

Fase 1





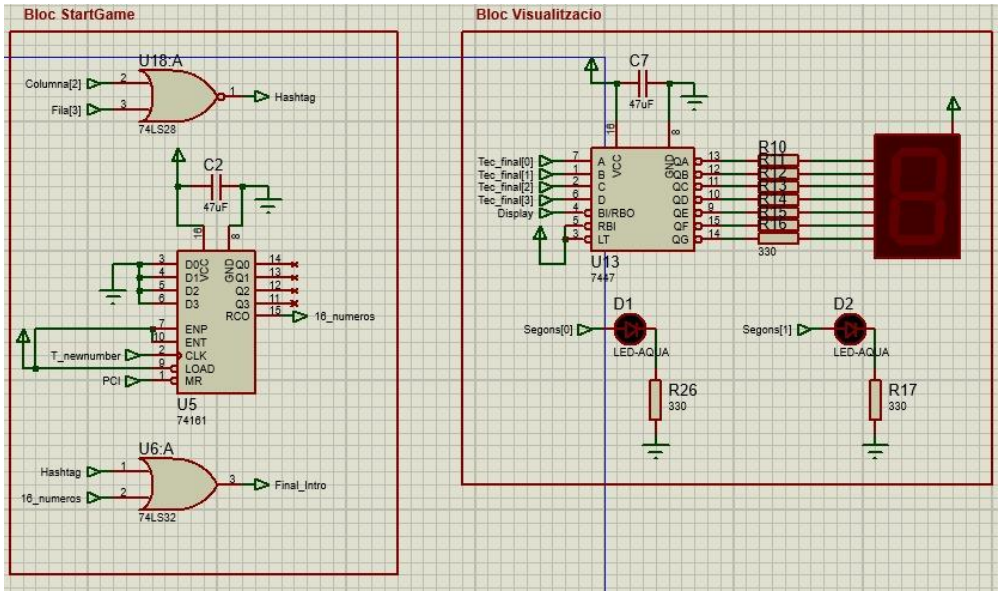
El Bloc Timer genera la senyal de sincronisme de tot el sistema d'1ms. El xip utilitzat es el NE555. En aquest bloc el sistema s'encarrega de generar una senyal quadrada amb un període d'1ms amb un duty cycle d'un 50% aproximadament.



El Bloc Teclat esta format per un comptador de 4 bits (74LS161) que envia les seves sortides a un decodificador de 2 bits a 4 bits (74LS139). Les sortides del decodificador fan referencia a les files del teclat matriu. Degut a que el teclat matriu funciona amb lògica negativa, es fa us de portes NOT (74LS04) per fer el correcte escombrat del teclat.

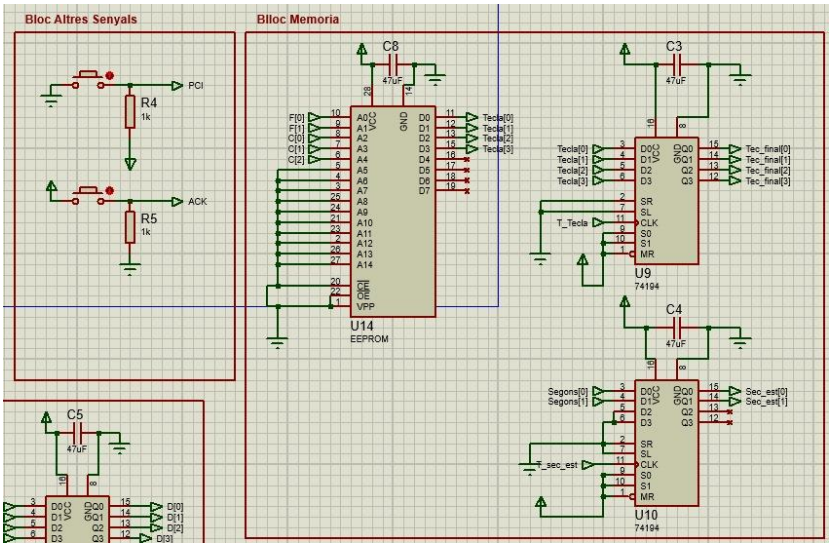
Per un altre banda, mitjançant una porta OR i una NOT generem les senyals HiHaTecla i NoHiHaTecla.

Per controlar els rebots utilitzem un comptador de 4 bits (74LS161) que compta 16ms i ens permet controlar els rebots. Amb la sortida RCO d'aquest chip generem la sortida 16ms.



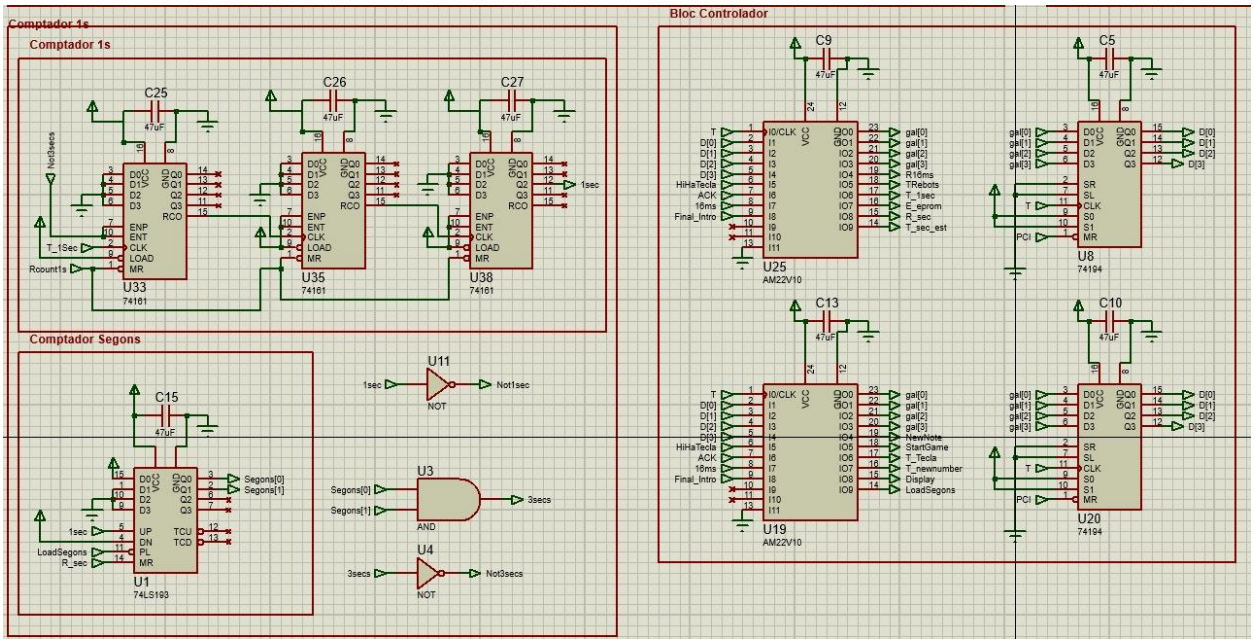
El bloc StartGame esta format per un comptador de 4 bits (74LS161) i un parell de portes OR. Amb la Columna[2] i la Fila[3] es pot saber quan l'usuari a polsat la tecla hashtag. El comptador compta totes les tecles polsades en un mateix joc. Si arriben a 16, s'activa la sortida RCO del comptador i es genera la senyal Final_Intro (o bé si l'usuari prem la tecla hashtag).

El Bloc Visualització esta format per un decodificador de 4 bits a 7seg (74LS47), un display 7seg i dos LEDs. Mitjançant aquest bloc es pot mostrar de manera visual la informació que s'ha introduït pel teclat com la tecla o la duració d'aquesta.



El Bloc Memòria esta format per la EEPROM (PT M27C256) i dos registres(74LS194). La EEPROM s'encarrega de processar l'escombrat del teclat matriu amb els busos de files (F[1..0]) i columnes (C[2..0]). Un cop tenim les dades de la tecla polsada les passem per un registre per guardar les dades estables. El mateix fem amb el bus dels segons que la tecla ha estat polsada (Segons[1..0]), guardem les dades de forma estable amb un registre. La senyal de sincronisme dels registres es controla amb la ME.

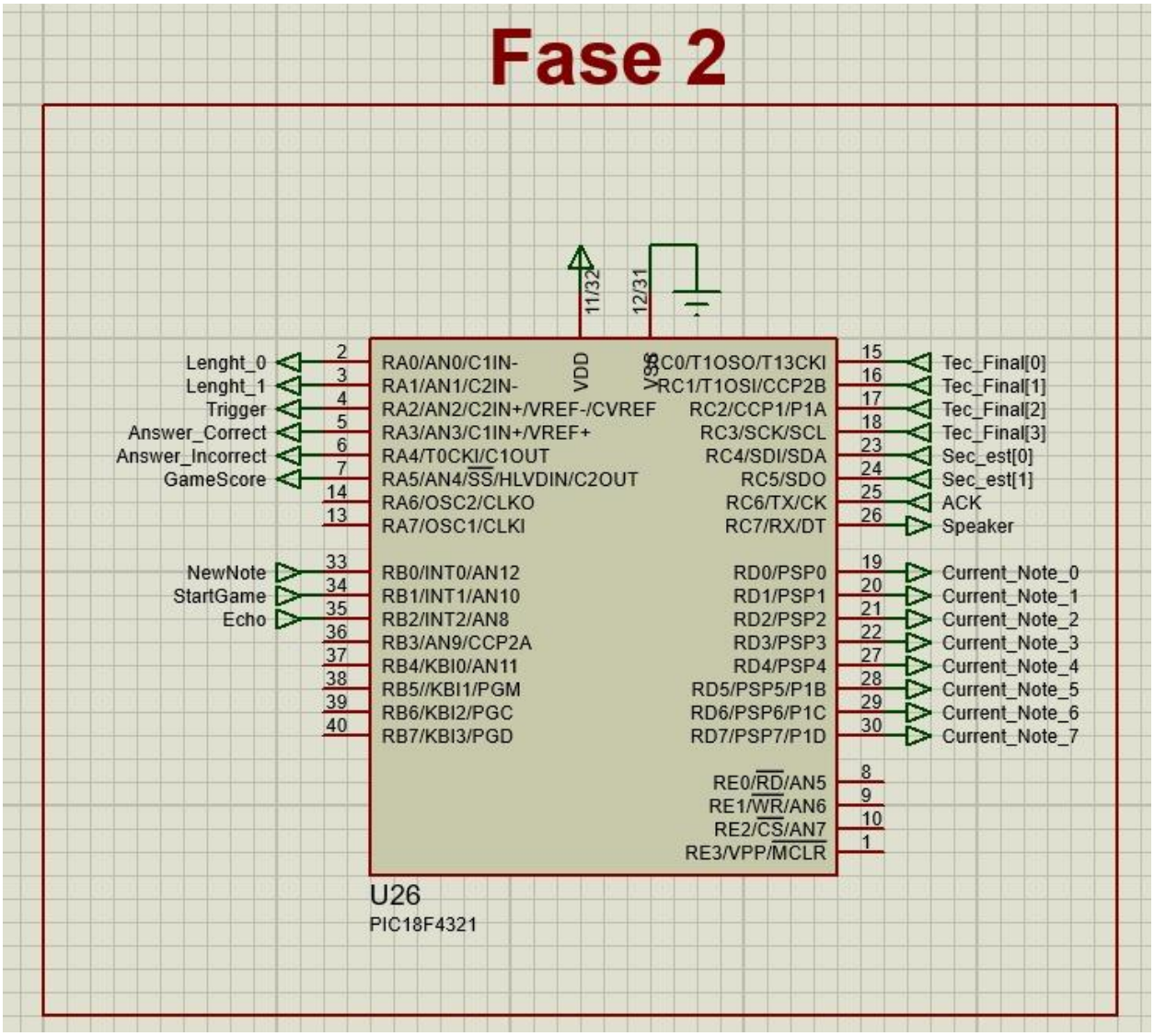
El Bloc Altres Senyals esta format pels polsadors que generen les senyals PCI i ACK.



El Bloc Comptador 1s esta format per tres comptadors de 4 bits (74LS161), un quart comptador de 4 bits (74LS193) i un parell de portes lògiques. Amb els primers 3 comptadors (connectats entre si), quan la sortida Q2 del tercer comptador val un 1 lògic, sabem que ha passat un segon. Aquesta sortida del tercer comptador fa que el quart comptador compti un cop, d'aquest manera els tres primers comptadors compten un segon tota la estona, i el quart guarda els segons i te la possibilitat de fer una carrega en cas de que fos necessari.

El Bloc Controlador esta format per dos GALs (AM22V10) i dos registres(74LS194). Les GALs son les encarregades de controlar la màquina d'estats, seguint el codi programat en aquestes. La informació provinent de la sortida de les GALs (estats) es guarden de forma estable mitjançant l'ús de dos registres.

Fase 2


Fase 2
11/32
12/31
VDD
VSS
Lenght_0
Lenght_1
Trigger
Answer_Correct
Answer_Incorrect
GameScore
NewNote
StartGame
Echo
RA0/AN0/C1IN-
RA1/AN1/C2IN-
RA2/AN2/C2IN+/VREF-/CVREF
RA3/AN3/C1IN+/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/HLVDIN/C2OUT
RA6/OSC2/CLKO
RA7/OSC1/CLKI
RB0/INT0/AN12
RB1/INT1/AN10
RB2/INT2/AN8
RB3/AN9/CCP2A
RB4/KBI0/AN11
RB5//KBI1/PGM
RB6/KBI2/PGC
RB7/KBI3/PGD
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2B
RC2/CCP1/P1A
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5/P1B
RD6/PSP6/P1C
RD7/PSP7/P1D
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
RE3/VPP/MCLR
Tec_Final[0]
Tec_Final[1]
Tec_Final[2]
Tec_Final[3]
Sec_est[0]
Sec_est[1]
ACK
Speaker
Current_Note_0
Current_Note_1
Current_Note_2
Current_Note_3
Current_Note_4
Current_Note_5
Current_Note_6
Current_Note_7
U26
PIC18F4321

6. Problemes observats

Durant la implementació de la Fase 2, hem trobat diversos problemes que han dificultat el correcte funcionament del sistema. Un dels principals obstacles va ser la sincronització de temps entre els diferents perifèrics, especialment amb el temporitzador TMR0. A l'utilitzar aquest temporitzador per controlar els intervals de temps, vam observar que la sincronització no era consistent, cosa que provocava errors en la temporització d'algunes accions, com la durada de les notes o el temps que es mostrava la informació als display i LEDs. Això es va traduir en un comportament inestable del sistema, amb errors en la transició entre notes o en la durada d'aquestes, que no coincidia amb les expectatives establertes en la melodia.

A més, vam tenir problemes a l'hora de generar les senyals PWM per al servomotor i l'altaveu. Les interrupcions del microcontrolador estaven afectant negativament els càlculs dels temps de la senyal PWM, ja que les interrupcions tallaven els temporitzadors de manera inesperada, cosa que provocava desajustos en la senyal generada. Això va provocar que la durada de les senyals PWM variés sense motiu aparent, afectant la precisió del moviment del servomotor i la reproducció de sons a l'altaveu. Aquest error es va identificar com un mal càlcul dels temps d'interrupció, que ocasionava un desfasament en la generació de la senyal.

Ambdós problemes van ser fonts importants de desconnexió entre els diferents components del sistema i van requerir un procés de depuració acurat per garantir que la sincronització fos precisa i que els temporitzadors i les senyals PWM funcionessin de manera correcta.

7. Conclusions

En aquesta pràctica, hem aconseguit desenvolupar un sistema interactiu per a un joc musical digital mitjançant la combinació de disseny de circuits digitals i la programació en llenguatge assembler per al microcontrolador PIC18F4321. La Fase 1 va establir les bases amb la introducció de notes i la seva durada, mentre que la Fase 2 ha permès implementar la lògica del joc, integrant diversos perifèrics com el sensor ultrasònic, el display de 7 segments, LEDs indicadors i el servomotor per a la puntuació.

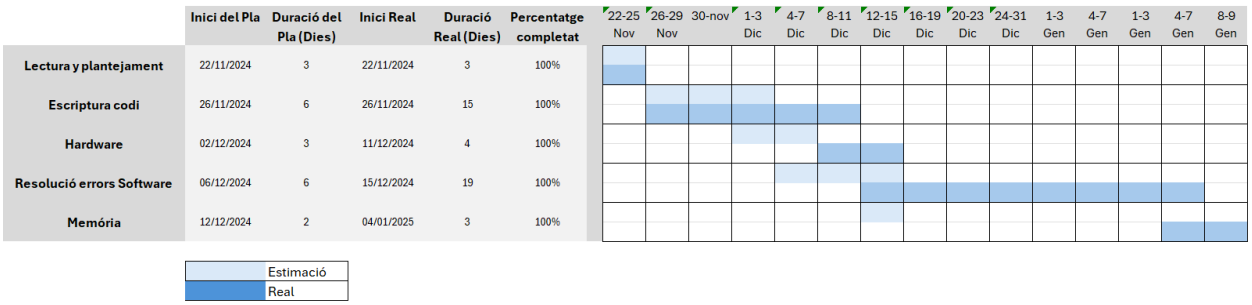
Aquesta pràctica ha estat essencial per comprendre la gestió de senyals digitals en temps real, la interacció entre components electrònics i el control precís de dispositius com el servomotor mitjançant senyals PWM. La integració del sensor ultrasònic ha representat un repte addicional, ja que s'ha hagut de programar per mesurar la distància entre la mà i el sensor, associant aquesta distància amb notes musicals específiques.

Una de les conclusions més importants és la importància de la gestió de la sincronització i la comunicació entre els diferents components del sistema per assegurar una experiència de joc fluida. La implementació de la puntuació a través del servomotor ha proporcionat una forma visual i immediata de retroalimentar el jugador, afegint un component motivacional al joc.

Finalment, aquesta pràctica no només ha permès adquirir coneixements tècnics en l'àmbit de la programació de microcontroladors i circuits digitals, sinó que també ha fomentat la creativitat en el disseny d'una experiència interactiva que uneix música i tecnologia.

8. Planificació

	Inici del Pla	Duració del Pla (Dies)	Inici Real	Duració Real (Dies)	Percentatge completat
Lectura y plantejament	22/11/2024	3	22/11/2024	3	100%
Escriptura codi	26/11/2024	6	26/11/2024	15	100%
Hardware	02/12/2024	3	11/12/2024	4	100%
Resolució errors Software	06/12/2024	6	15/12/2024	19	100%
Memória	12/12/2024	2	04/01/2025	3	100%



	Estimació
	Real

En aquesta pràctica no s'ha pogut completar la planificació feta a l'inici d'aquesta. Al plantejament de la pràctica es va decidir de començar quan abans i planificar les tasques per entregar abans de Nadal, tal i com es veu al diagrama (color blau cel). Això no ha estat complert, ja que amb la demora que vam tenir en la escriptura del codi, la pràctica s'ha hagut que dur a terme durant el Nadal. Treballar al Nadal ha provocat encara mes demora en la resolució d'errors del software degut a les festivitats i feina d'altres assignatures. Tot i això, la pràctica ha estat 100% completada.