ECE 414 – Embedded Systems

Final Project – PING PONG GAME

IMPLEMENTATION PLAN

Chikomborero Dhire

Tafita Rakotozandry

1 Introduction

During the 1970s the pong game was quite popular among young people. It was among the earliest games ever made. For our final project, we decided to recreate this iconic game using a PIC32 and an LCD display. It can be used as a way to entertain people, who throughout the years were always looking for creative ways to entertain themselves. For this project, our goal was to fit the overall system in a single breadboard and using the least budget possible.

2 Project Description

The first version of the pong game came out in 1969 (Sellers, 2001). It was developed by William Rusch. The patent number is RE28507. The game was licenced to Magavox, the first commercial home video game console. The game is known for being one of the earliest video games ever created. Throughout the years, different companies such as Target, have developed different versions of it, one of the most notable versions is the foot pong, which is a pong game using a foot controlled paddle (Sellers,2001). A sensor senses the position of the foot and moves the paddle accordingly.

In our version of the game, two players compete with each other. Each one of them will have a potentiometer to allow them to control the movement of the paddle of the game. The ball will bounce every time it hits the paddle. The first user who misses touching the ball with the paddle loses. The system will keep track of the score. The game will be set in a way that can play for as long as none of them has exceeded the maximum score. The layout of the game is similar to the image below.

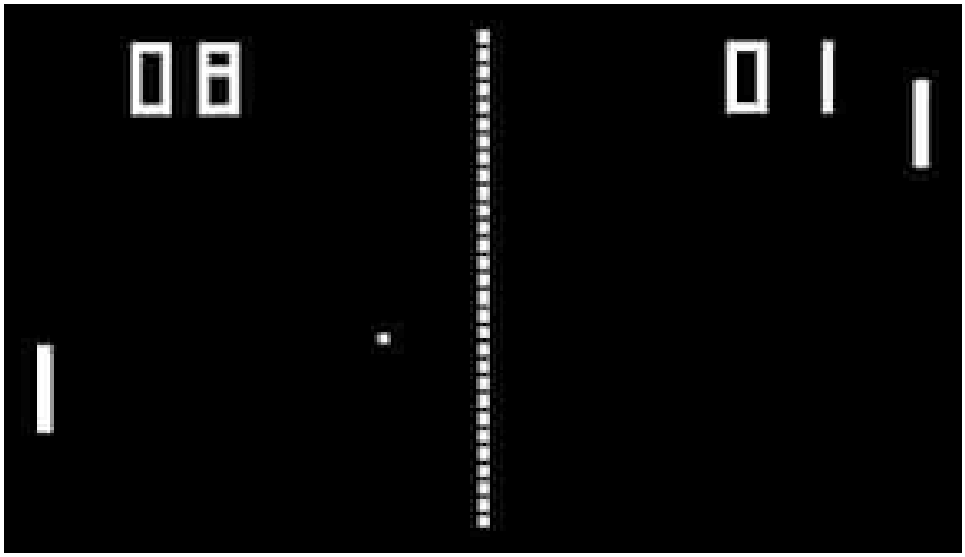


Figure 1: User Interface From the Original Game

3 Requirements

In this game, the two participants of the game will be able to control the paddles using potentiometers. The ball will bounce every time it hits the paddle. The first user who misses touching the ball with the paddle loses. The system will keep track of the score. The game will be set in a way that it continues as long as the maximum score is exceeded. The following lists detail the requirements of the game:

1. The hardware design shall include the PIC32 microcontroller / Microstick II, 2.4” TFT LCD display and potentiometers.

2. The LCD display shall display the paddle (represented as a rectangular bar), the ball (represented as a circle) and the current score.

3. The paddles shall only be able to move in one axis i.e either up or down both at the same based on the user’s input via the potentiometer.

4. The ball shall bounce symmetrically when it hits the two horizontal edges and it will bounce relative to the position at which it hit the paddle, that is, the direction of the ball after hitting a paddle will be determined by the position at which it hit the paddle e.g if the ball is coming straight but hits the paddle’s lower end, it’s direction will be downwards.

5. The game shall be played by the following rules and conditions:

a) Initially, the ball is assigned to one of the players and this is indicated by the ball (filled circular object) in front of the paddle.

b) When that player presses the push button to the side of the ball , the serve starts.The ball starts moving towards the second player in the opposite direction.

c) When the ball reaches the other end, that player must “hit" the ball by making sure it comes into contact with the paddle before it goes beyond the screen.

d) Whenever the ball comes in contact with the paddle, it should travel in the opposite direction.

e) The ball shall travel back and forth in this way until one of the players “misses". A miss is when the ball goes beyond the line of movement of the paddle.

f) When a player misses, a sound will be produced, a unit point will be added to the opposite player. (optional)

g) The player who missed will get to make the initial serve and it goes back to b).

h) When a player gets a score equal to the maximum score in that game, they would have won. The game will be over and it starts again at a).

6.The speed of the ball shall also be altered depending on the position on which it hit the paddle

- if the ball contacts the top third of the paddle, its y speed should decrease to a set minimum (i.e so that the ball does not either stop or go back towards the other player at any moment).
- if the ball contacts the middle third of the paddle, its y speed should stay the same.
- If the ball contacts the bottom third of the paddle, its y speed should increment until it reaches the set maximum.

7. When a player wins a unit point, a tune will be produced and when a player wins the game, a different tune will be played.
8. When a player wins the game, the final score will be displayed in a larger font.

4 Architectural Description

4.1 Inventory:

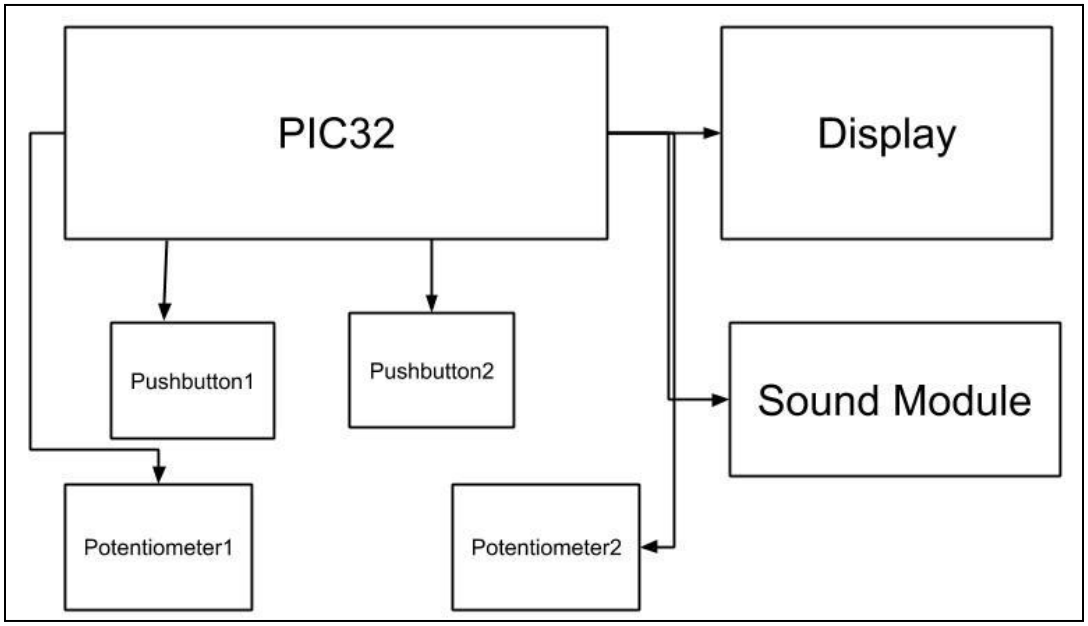
As mentioned earlier, our goal was to use components that are the least expensive possible yet quite accessible. Below is the list of the different tools we used:

- 1x PIC 32
- 1x Adafruit LCD
- 2X Potentiometers
- 2x Pushbuttons
- 1 x 1microFarad Capacitor
- 1x 560 Ohm Resistor
- 1x MCP4B22 DA

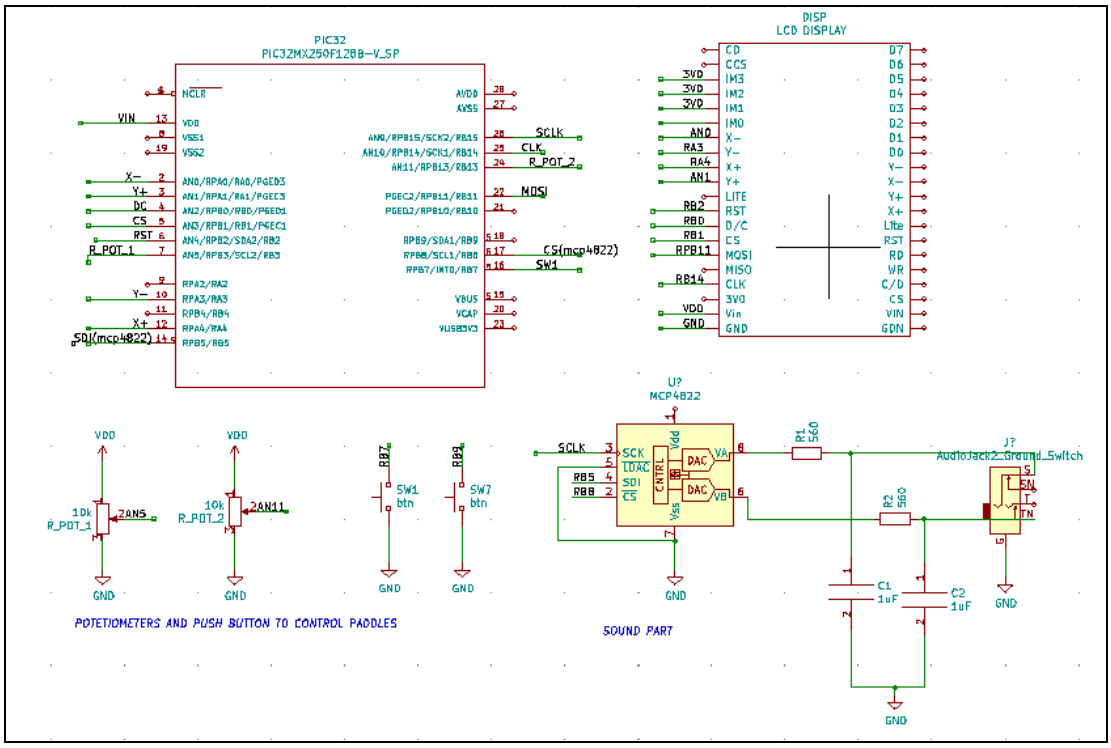
We used MPLAB for most of the programming. We used RIBS to generate our FSM code. It was very practical for debugging the codes.

4.2 Hardware Connection

The different components of the system are connected altogether through the PIC32. As illustrated in the block diagram below :



The following figure represents the wiring diagram between the connections.



The LCD connection is similar to the previous lab. We kept the same wiring as presented in the table. The two potentiometers are connected to the Vcc from one side and to the GND to the other side. They are connected to the PIC 32 through the PIN 7 and PIN 24. The PIC32 will read the voltage drop across these components and convert it into the corresponding LCS coordinate.

The MCP4B22 DAC is connected to pin 26 of the PIC 32 and the other end is connected to a low pass filter and the audio jack.

TFT LCD Pin	Microstick II Pin
GND	GND (jumper)
Vin	VDD (jumper)
CLK	SCK1 / RB14 (25)
MOSI	SDO1 / RPB11 (22)
CS	RB1 (5)
D/C	RB0 (4)
RST	RB2 (6)
IM1	LCD 3Vo*
IM2	LCD 3Vo*
IM3	LCD 3Vo*
X+	RA4 (12)
X-	AN0 (2)
Y+	AN1 (3)
Y-	RA3 (10)
*connects to pin *3Vo* on the LCD	

4.3 Software Modules

We have 12 different software modules in this project but some of them were reused from the old lab. The block diagram below summarizes the over all of them

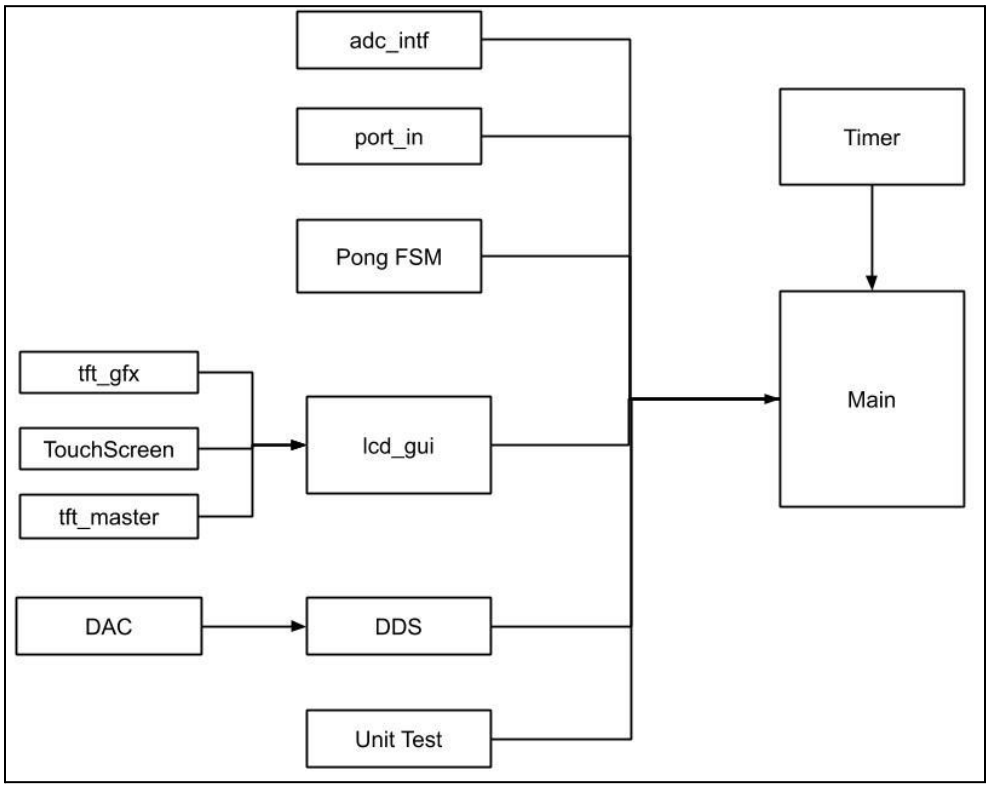


Figure 2: Modules Diagram

4.3.1 adc_intf :

This module allows us to initialize the analog input reading. It allows reading the potentiometer value directly from 0 to 3.3V to 0 to 1023. This reading output is divided to 5.7 which is a constant that we found from dividing the height of the screen into a unit scale.

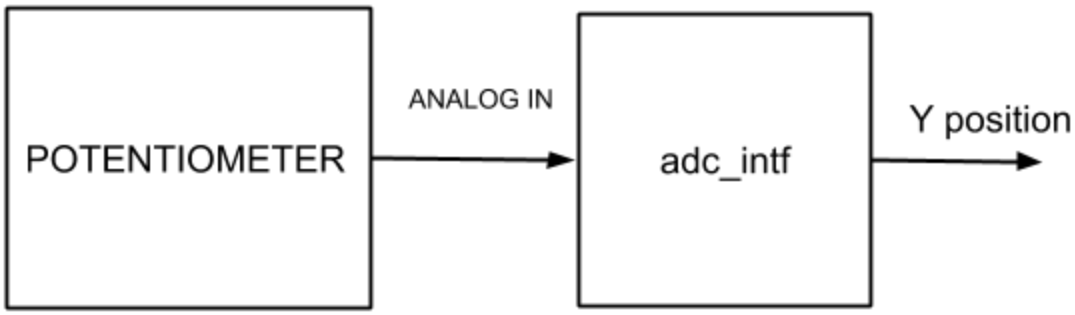


Figure 3: A depiction of adc_intf in relation to the potentiometer

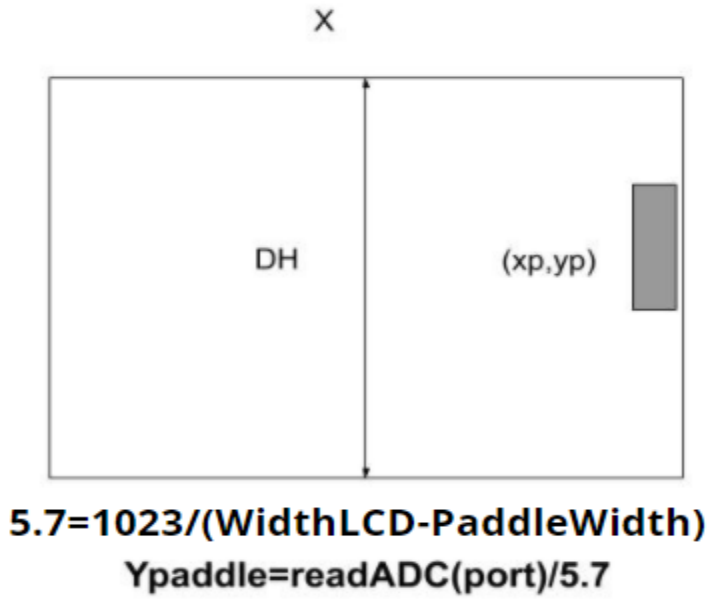


Figure 4: The calculations done to come up with the y value

4.3.2 Port_in :

This module initialises the push button pin as an input. It also includes two functions ***btnL_read()*** and ***btnR_read()*** which return the boolean reading of the pushbuttons.

4.3.3 lcd _gui

The drawing methods are implemented in this module: *drawBall(), drawPaddle(),drawMiddleLine()*. This module imports tft-gfx and tft-master modules. They were already provided during the class.

4.3.4 dds

This module allows it to generate a frequency which can be processed by the DAC. The signal produced will be filtered by the low pass filter and goes to the speaker. We took this program from the lecture.

4.3.5 pong_fsm

This module implements the game state machine.There are 9 different states: *Initial,Cust_Score, Start_L,Start_R, Move, left_pos,right_pos,restart, gameover*.

In the *Initial* state, the score of both players is initialized. After it comes in the *Cust_Score* which allows to customize the maximum score. When the maximum score is chosen, we move to the *start_R* state which starts the game to the right of the screen. The ball will only move when the right button is pressed. The *Move* state directs the movement of the ball. The ball moves until it reaches the left paddle. The *left_pos* will check if the ball hit the ball or not. If yes, it will move back to the Move state. It goes in the opposite direction. If it does not touch the paddle. The right player is the winner. The game will go to the restart state. Based on the actual score, the restart state will restart the game to *start_L* or *start_R*. If not, the game is over.

The representation below summarizes that finite state machines:

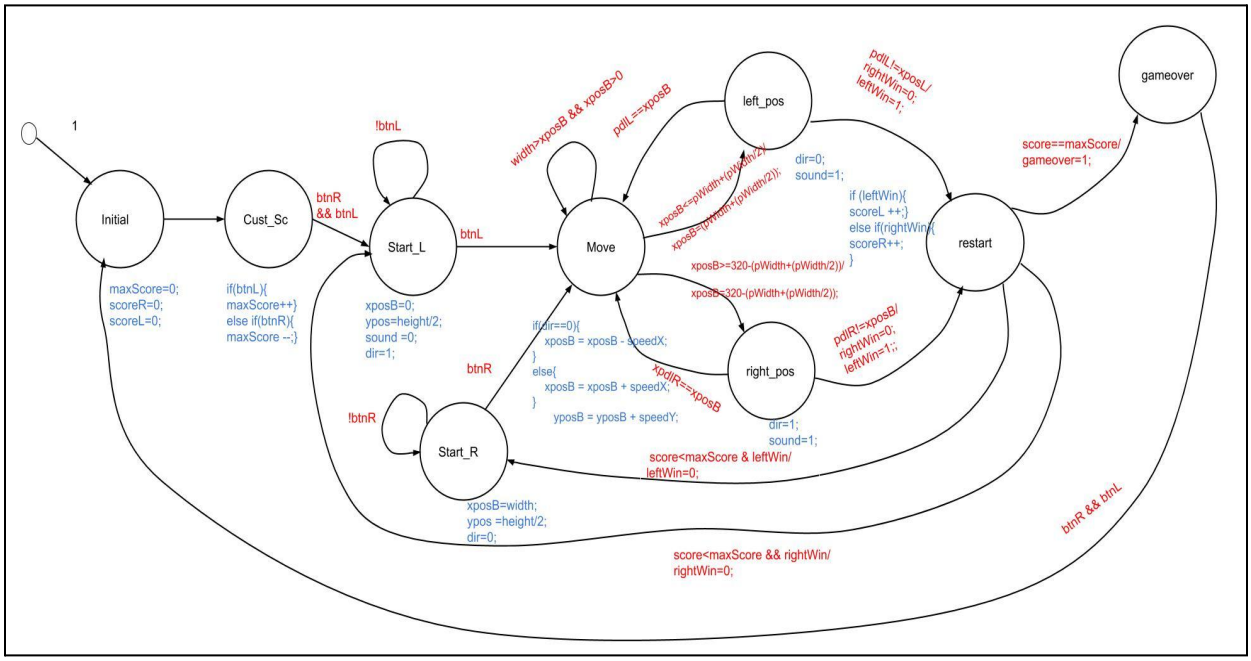


Figure 3: Main game Finite State Machine.

The first state “Initial” sets all the scores to zero. The next state which it will automatically enter has the user customise the maximum score and enters the “startL” state when both players press their buttons.

The move state is responsible for the movement and bouncing of the ball on the edges of the LCD. The “left_pos” and “right_pos” check to see if the ball has either been hit or missed by the paddle. If it has been hit, it goes back to the “move” state, bounces and goes to the opposite direction.

A variable named sound was created in the pong_fsm in order to activate the tick sound on the main everytime the ball hit the paddle. In addition to that, at the beginning of the game and at the end of the game, we play a melody. Therefore, we make use of another variable called state to activate that melody.

NB: The melody is an array of frequencies with different values that when combined produce a musical melody. We found these arrays from a website that is listed in the appendix.

One of the problems that we encountered is that the mechanical button exhibits noise. It makes the use of the pushbuttons quite unstable. In order to avoid the debouncer FSM was implemented. We implemented this code inside the module pong_fsm.

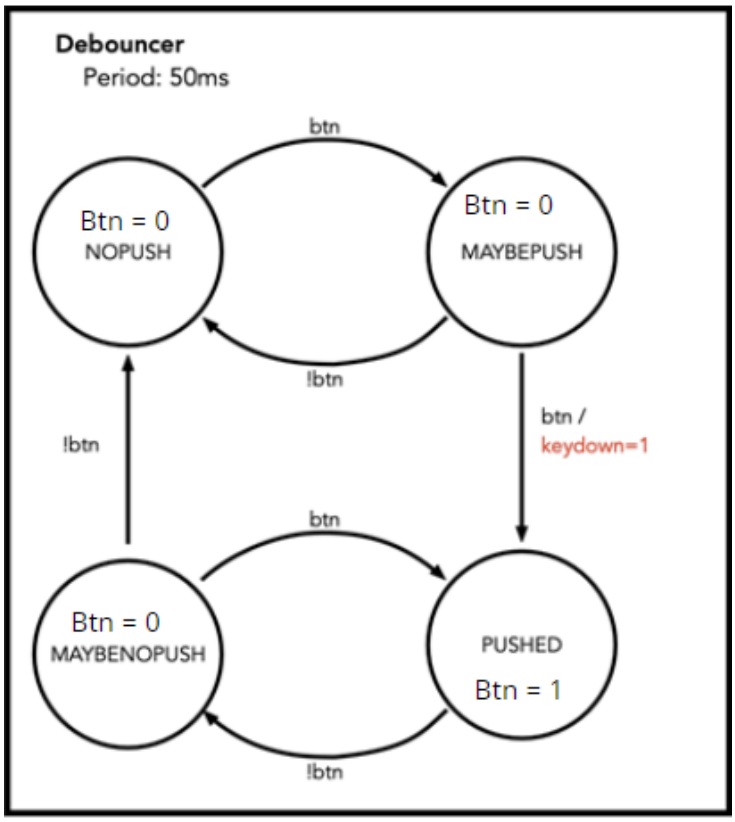


Figure 4: Debouncer State machine.

4.3.6 unit _tests

This module has programs to unit test each module individually. For most of the modules we tested their functionality by printing and their values on the LCD screen.

4.3.7 Main

The main combines puts all the different modules all together. It uses the timer1 from class to control the flow of the code. Each state machine has a period of 100 ms.

5 Test Plan

As part of any project, testing is an important part because it allows us to validate a project. Apart from testing the game in general, we also defined a unit test in order to guarantee that each module works appropriately.

1. **Unit Testing**

The unit test was made possible using another program which allows us to verify each module individually. Here are the different test plan that we designed:

T1.1The display:

- Run the unit test code
- Observe if the LCD displays a ball a the paddle
- Verify that the two paddles are in the rightmost and leftmost of the screen

T1.2

Potentiometer

- Run the unit test code
- Verify in the UART that potentiometer reading is changing
- Verify that the paddle moves according to the potentiometer orientation in the y axis
    - Turn potentiometer left, moves the paddle to the up
    - Turn potentiometer right, moves the paddle to the down

T1.2

Push Button

- Run the unit test code
- Verify that the UART prints “ON” when the paddle is pressed.

T1.3

Sound module (if only if the the sound is implemented)

- Generate a signal of 1KHz frequency.
- Verify that the speaker generates the sound relative to that frequency.

2. **Game Test**

Test 2

- Start the game by serving (moving the potentiometer corresponding to paddle with ball).
- Wait for the ball to arrive on the other side and “hit” it with the paddle.
- Repeat the same procedure at least 2 times and make sure at some instances it hits the horizontal edges: observe how it bounces.
- Observe if the ball goes back and forth from one paddle to the other.
- Observe the movement of the paddle.

Test 3

- Start playing the game and make one player lose a point by missing to hit the ball. Observe if a sound is produced and if the score increments.
- Continue playing until one player wins. Observe if the final scores are displayed in larger font. Also observe a different sound is produced.

TEST	REQUIREMENT													
	1	2	3	4	5a	5b	5c	5d	5e	5f	5g	5h	6	7
T1.1		X												
T1.2			X											
T1.3													X	
2	X	X	X	X	X	X	X	X	X	X	X	X		
3	X												X	X

3. Test Result

We conducted our test one by one multiple times. The table below summarizes the overall ends of the project. In summary, our project met the requirements that we proposed

TEST	REQUIREMENT														
	1	2	3	4	5a	5b	5c	5d	5e	5f	5g	5h	6	7	TESTER
T1.1		P													Chiko
T1.2			P												Tafita
T1.3													P		Chiko
2	P	P	P	P	P	IP	P	P	P	P	P	P			Tafita
3	P												P	P	Tafita

P-Pass

F-Fail

IP- Improved and Passed

4. Acceptance test

The game met all the requirements and passed the tests that we put it under. The larger part of our acceptance test was about the usability and competitiveness of the game.

The game was tested in its entirety by playing it and it was competitive enough for one person to win within a reasonable amount of time.

The only notable issue that we had in this project is the screen flickering. At the beginning we attempted to clear the entire screen every 100 ms to refresh the screen. However , it was not quite pleasant. We ended up erasing only the specific part of the screen that was moving.

Conclusions

In summary, we replicated the famous pong game during the 1970's using PIC 32 , LCD Display and a breadboard. We developed the hardware architecture as well as the software architecture. We tested our implementation and concluded that our project met with the requirements that we suggested at the beginning of the project. Although the success of our project however there are several features that can be improved in this project to make it as perfect as possible. An example for that is the counter. Having a counter that limits the duration can be practical if we wanted the player to continue playing forever. The ability to change the level of the game can also be another feature that can be added in this game.

The project took us 4 weeks to complete. The most challenging part of the project was to debug the FSM. We came up with our own FSM, therefore, it was hard to guarantee if our initial versions would work.

References

Microchip, PIC32MX Family Data Sheet”, 2008. [Online serial]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

NorthWestern Center For Robotics and Biosystems,” Driving a piezo speaker with a PIC”, 2012. [Online serial]. Available: http://hades.mech.northwestern.edu/index.php/Driving_a_piezo_speaker_with_a_PIC

Cornell University,” ECE4760 PIC32MX250F128B experiments”, 2008. [Online serial]. Available: https://people.ece.cornell.edu/land/courses/ece4760/PIC32/index.html

J3,” Pong Game Explained Finally”,Mar 7, 2017. [Online serial]. Available: https://medium.com/jungletronics/pong-game-explained-finally-ac964140a6e5

Sellers, John,” "Pong". Arcade Fever: The Fan's Guide to The Golden Age of Video Games”, (August 2001) Running Press. pp. 16–17. ISBN 0-7624-0937-1.

Appendix A: Schematics

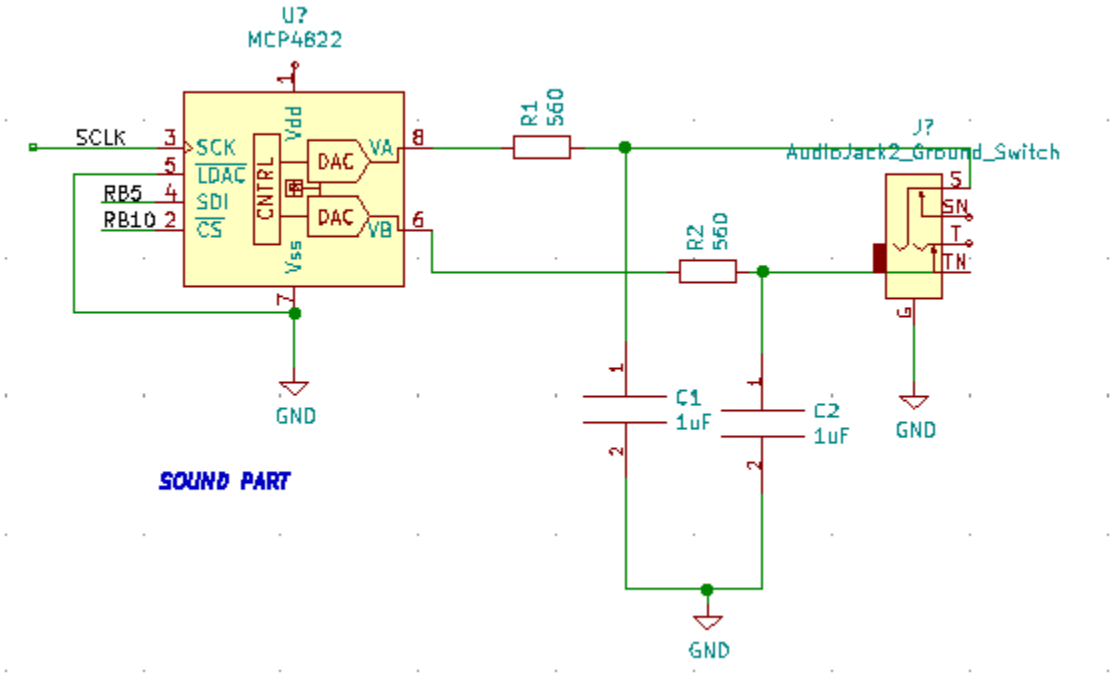


Figure 2:Sound Module Schematic

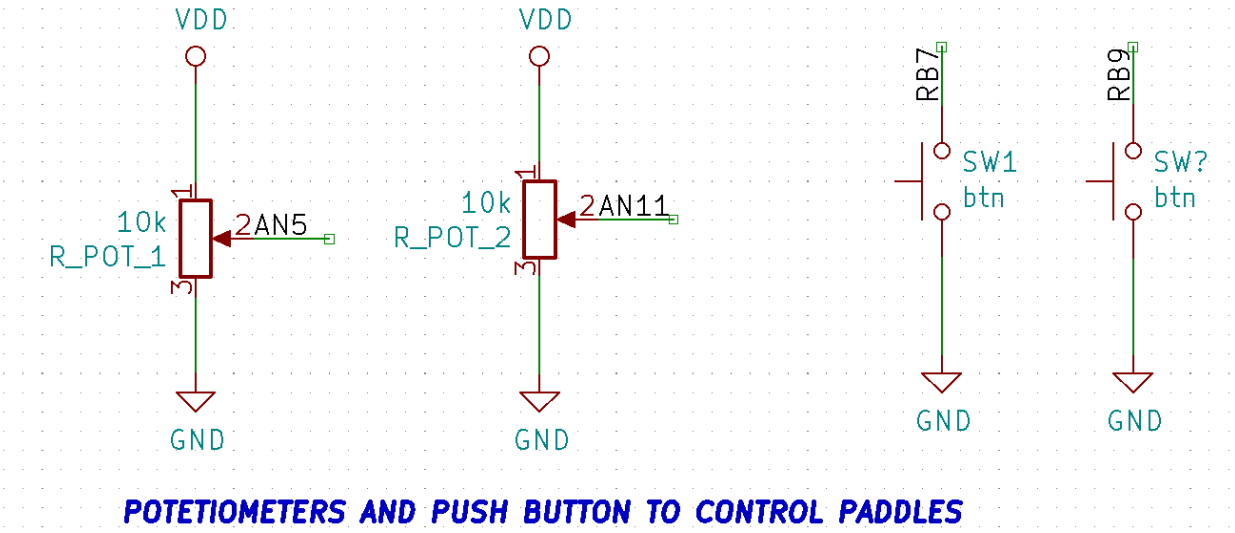


Figure3: Push Buttons and Potentiometers Wiring

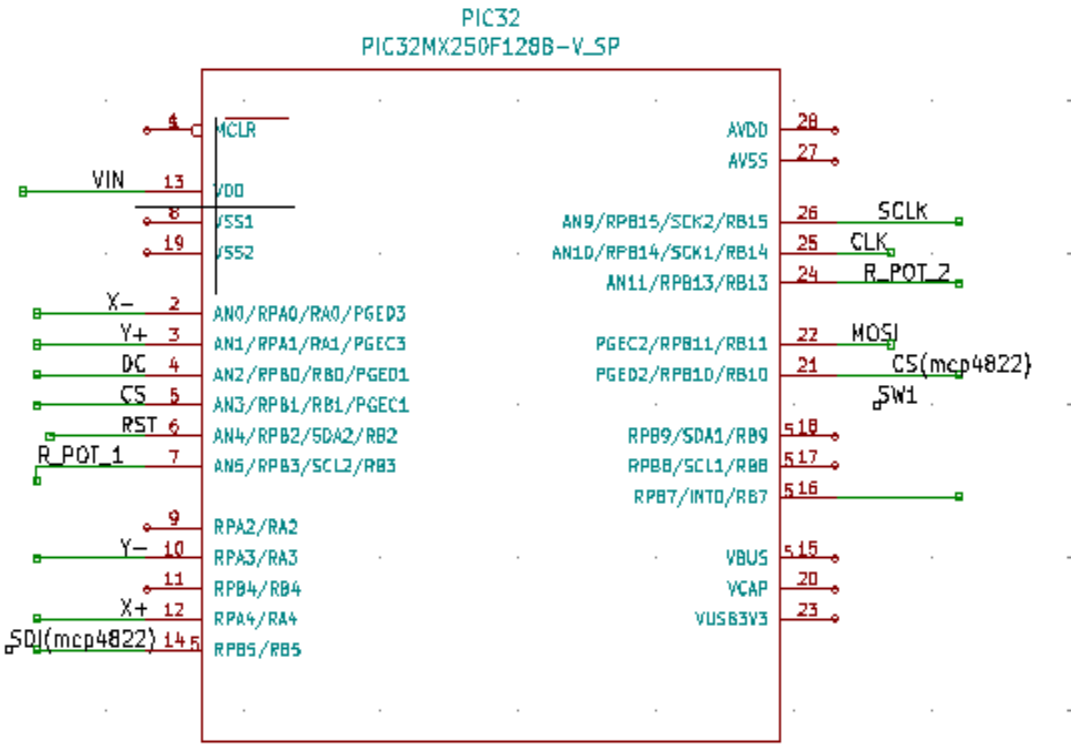


Figure 4: PIC 32 Connection Diagram

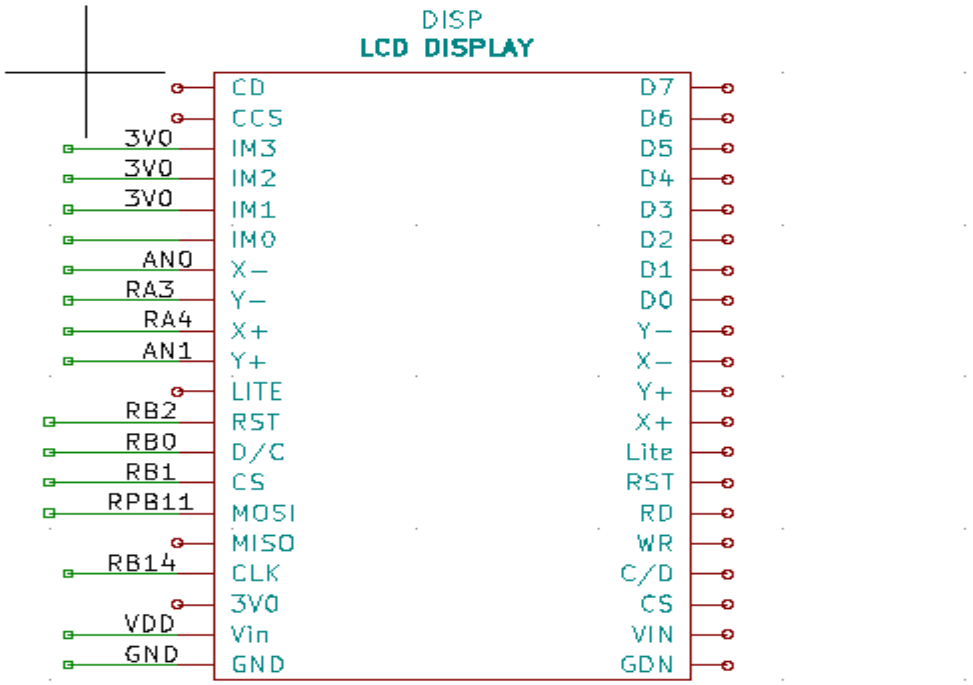


Figure 5: LCD Connection Diagram

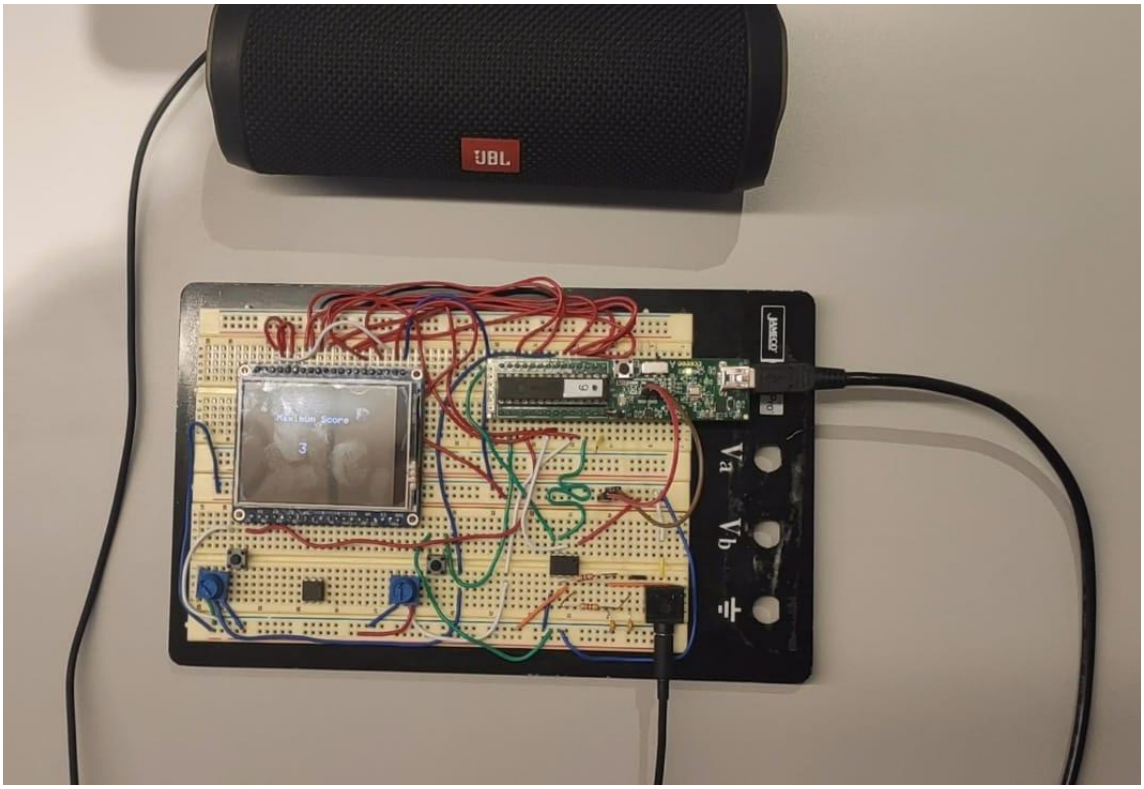


Figure 6: Hardware implementation of the schematics

Appendix B: Cost Approximation

Component	Cost
Microstick II with PIC32	$34.95
Adafruit LCD	$27.50
Potentiometer x 2	$1
MCP4822	$2.50
1microFarad Capacitor	$0.47
560 Ohm Resistor	$1.01
Total	**$67.43**