U-BYTE

2020

MICRO-CONTROLLER WORKSHOP

PROJECT_NAME: PINGPONG68

BY

ADMIN368

INTRODUCTION

This Project was to replicate the famous game of Ping Pong on an 8x8 led Matrix using an 8051 Microcontroller. It slowly evolved to include key controls implementation, 2 player functionality, basic score record keeping and intermediate Ai player option to allow one player to play with the game itself.

HARDWARE AND WIRING

This Ping Pong Recreation was done on YF-K1 board, running a 89C52RC chip and coded in C programming Language.

Apart from the Board and Chip the other components used were:

- 8x8 Led Matrix
- 4 push buttons
- And 8 individual Leds

The Wiring Diagram and Schematics are present for reference attached with the other files in the project files.

DEVELOPMENT

The first thing we developed was the drawing mechanism, We decided it would be very efficient to develop each function to be modular in a way that one part does a single job and so that there is no repetition of code doing the same job multiple time unnecessarily.

This helps to easily truck bugs or problems as well as each job is only executed from one place and If there is an error with a specific job you know where to look first.

The drawing mechanism involved creating an 8x8 matrix that is mapped to the exact dots on the 8x8 led matrix.

It was designed to have 1's in areas where we want the Leds to light up and 0's in places where the Leds should be off.

When drawing the ball and rackets the drawing functions only add 1's to the Array and then the Leds are lit where the ball and rackets are.

THE COORDINATE SYSTEM

To avoid writing array coordinates all the time which becomes hard to follow in the code,

Another function was designed to translate the coordinates of the array into a linear map to make the drawing of objects in the code much easier, faster and understandable.

Normal Array:

Array[0][0]	Array[0][1]	Array[0][2]	Array[0][3]	Array[0][4]	Array[0][5]	Array[0][6]	Array[1][7]
Array[1][0]	Array[1][1]	Array[1][2]	Array[1][3]	Array[1][4]	Array[1][5]	Array[1][6]	Array[2][7]
Array[2][0]	Array[2][1]	Array[2][2]	Array[2][3]	Array[2][4]	Array[2][5]	Array[2][6]	Array[3][7]
Array[3][0]	Array[3][1]	Array[3][2]	Array[3][3]	Array[3][4]	Array[3][5]	Array[3][6]	Array[4][7]
Array[4][0]	Array[4][1]	Array[4][2]	Array[4][3]	Array[4][4]	Array[4][5]	Array[4][6]	Array[5][7]
Array[5][0]	Array[5][1]	Array[5][2]	Array[5][3]	Array[5][4]	Array[5][5]	Array[5][6]	Array[6][7]
Array[6][0]	Array[6][1]	Array[6][2]	Array[6][3]	Array[6][4]	Array[6][5]	Array[6][6]	Array[7][7]
Array[7][0]	Array[7][1]	Array[7][2]	Array[7][3]	Array[7][4]	Array[7][5]	Array[7][6]	Array[8][7]

Coordinate System:

B1	B9	B17	B25	B33	B41	B49	B57
B2	B10	B18	B26	B34	B42	B50	B58
B3	B11	B19	B27	B35	B43	B51	B59
B4	B12	B20	B28	B36	B44	B52	B60
B5	B13	B21	B29	B37	B45	B53	B61
B6	B14	B22	B30	B38	B46	B54	B62
B7	B15	B23	B31	B39	B47	B55	B63
B8	B16	B24	B32	B40	B48	B56	B64

So an 8x8 Array results in a linear coordinate of 1 to 64 boxes, going column by column.
The function "draw(uchar box)" is used to print the 1's in the specific box

you wish which will then be translated to the right location in the 8x8 array by the math:

> **row=box-(8*(col-1));**
> **map[row-1][col-1]=1;**

Which brings coordinates for to give me the right values to place 1 in the 8x8 array. Eg:

> **array[row][col] = 1;** would become by writing :
> **Box(1);** as a simplified example.

The array structure and functions are in the “paulo8x8_v1.h” file for further details.

BALL MECHANICS

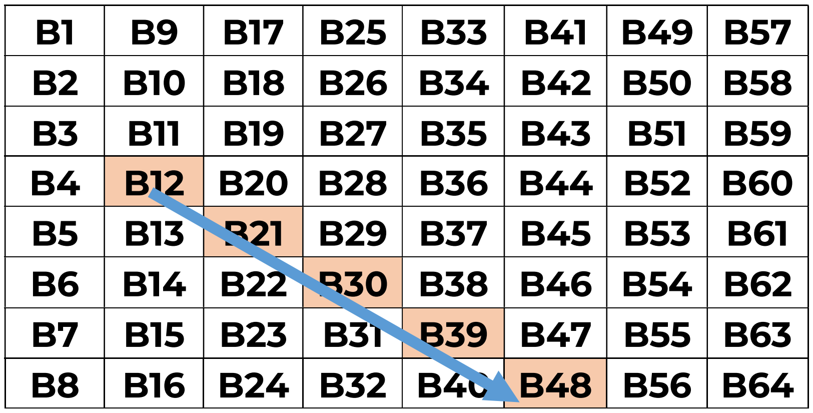
The ball is practically a value of 1 being drawn moving in the coordinate system.
After each interrupt the position is moved with a value stored in variable called “increment”.
This increment value also determines the direction of the ball.
Example:

- if the value of increment = 8 and the initial ball location = 12 ,Then :
    - After interrupt the ball’s location will be B12+8 resulting in B20, which will look like the ball is moving in a straight direction.

B1	**B9**	**B17**	**B25**	**B33**	**B41**	**B49**	**B57**
B2	**B10**	**B18**	**B26**	**B34**	**B42**	**B50**	**B58**
B3	**B11**	**B19**	**B27**	**B35**	**B43**	**B51**	**B59**
B4	**B12**	**B20**	**B28**	**B36**	**B44**	**B52**	**B60**
B5	**B13**	**B21**	**B29**	**B37**	**B45**	**B53**	**B61**
B6	**B14**	**B22**	**B30**	**B38**	**B46**	**B54**	**B62**
B7	**B15**	**B23**	**B31**	**B39**	**B47**	**B55**	**B63**
B8	**B16**	**B24**	**B32**	**B40**	**B48**	**B56**	**B64**

- But if the value of increment=9 and the initial ball location= B12,Then:

- after interrupt the ball's location will be B12+9 resulting in B21 , which will look like the ball is moving in a diagonal direction going downwards



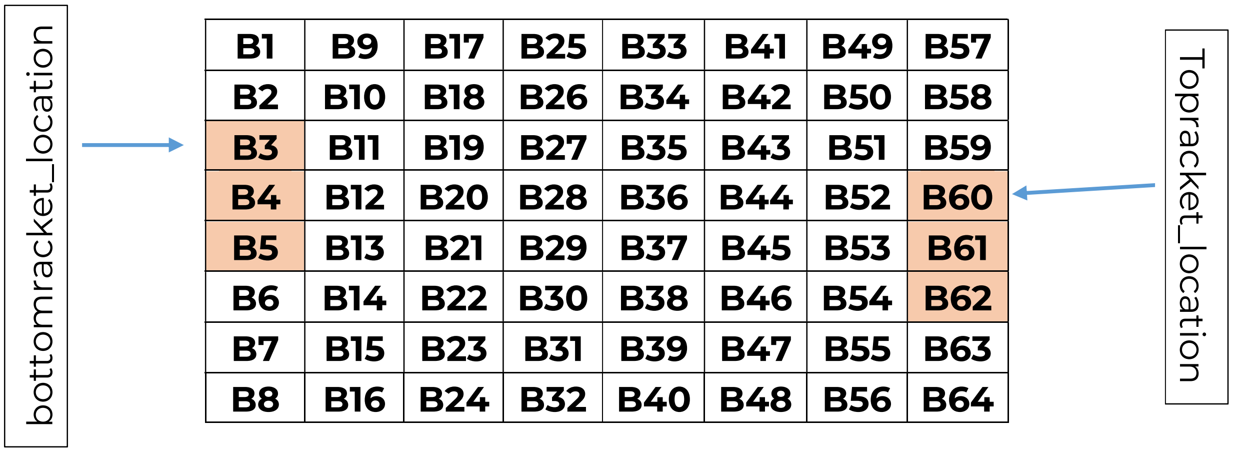
B1	B9	B17	B25	B33	B41	B49	B57
B2	B10	B18	B26	B34	B42	B50	B58
B3	B11	B19	B27	B35	B43	B51	B59
B4	B12	B20	B28	B36	B44	B52	B60
B5	B13	B21	B29	B37	B45	B53	B61
B6	B14	B22	B30	B38	B46	B54	B62
B7	B15	B23	B31	B39	B47	B55	B63
B8	B16	B24	B32	B40	B48	B56	B64

- The ball speed also gets faster and faster the longer you play.

RACKET MECHANICS

The rackets stay in the first and last column that is [B1-B8] and [B57-B64] for player 1 and 2 respectively.

The variables "bottomracket_location" and "topracket_location" keep the location of the rackets then 2 more 1's are placed in the boxes after to finish drawing the rackets.



B1	B9	B17	B25	B33	B41	B49	B57
B2	B10	B18	B26	B34	B42	B50	B58
B3	B11	B19	B27	B35	B43	B51	B59
B4	B12	B20	B28	B36	B44	B52	B60
B5	B13	B21	B29	B37	B45	B53	B61
B6	B14	B22	B30	B38	B46	B54	B62
B7	B15	B23	B31	B39	B47	B55	B63
B8	B16	B24	B32	B40	B48	B56	B64

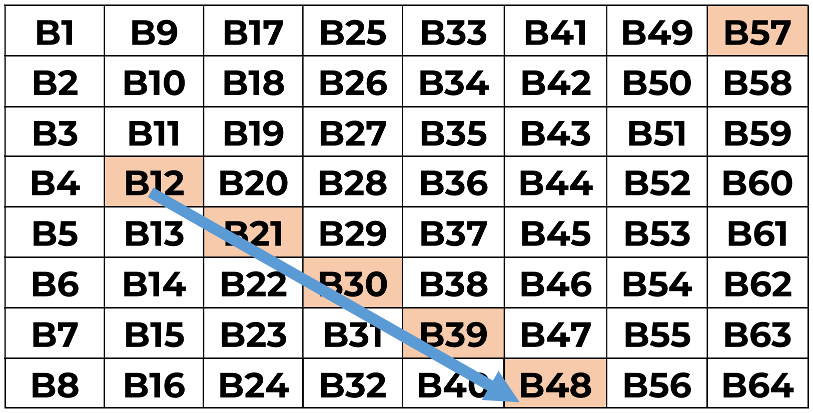
The Racket's 3 dots allow a sense of variation on direction,
if you hit the:

- Top dot: the ball has increment of 9 ,going diagonal downwards
- Middle dot: increment of 8, going straight
- Bottom dot: increment of 7, going diagonal upwards

This is also influenced by the direction the ball is coming at from the opponent allowing you to cancel a ball's direction by hitting with the opposite intended direction to result in the ball going straight.

If the bottom racket hits the ball the increment is positive allowing the ball to move Right,
and If the top racket hits it, the increment becomes negative allowing the ball to move Left.

BOUNCING
Another challenge that was faced, was that the walls [B8,B16,B24,B32,B40,B48,B56 and B64] including their opposite corresponding boxes at the end of the column do not function well with the increment math when the ball location hits these areas.



B1	B9	B17	B25	B33	B41	B49	B57
B2	B10	B18	B26	B34	B42	B50	B58
B3	B11	B19	B27	B35	B43	B51	B59
B4	B12	B20	B28	B36	B44	B52	B60
B5	B13	B21	B29	B37	B45	B53	B61
B6	B14	B22	B30	B38	B46	B54	B62
B7	B15	B23	B31	B39	B47	B55	B63
B8	B16	B24	B32	B40	B48	B56	B64

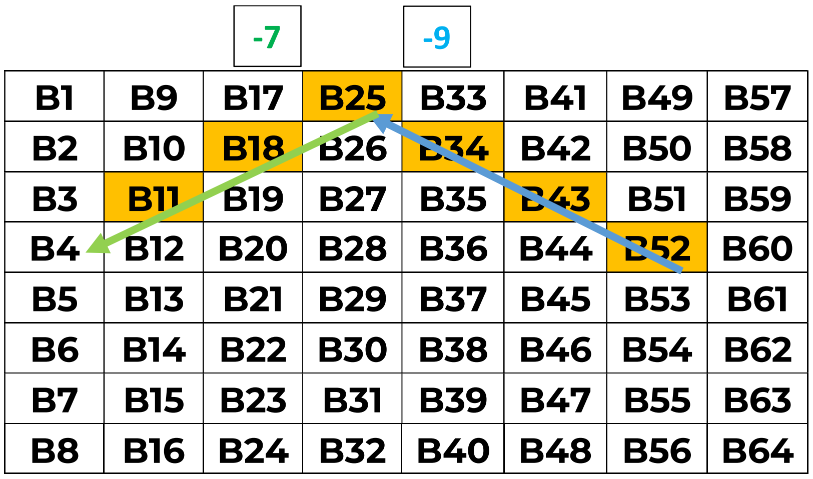
For Example is the ball was at B39 with increment=9 as on the above diagram:

- The next box will be B39+9=B48 [Correct]
- The next box will be B48+9=B57 [abnormal]

This will be a problem as the ball will suddenly be on the top part of the display on B57, making it unpredictable and hard to play with such abnormality.

The solution was to implement additional math for the wall bouncing, which first needed the knowledge of the direction of the ball which was determined by the “forward” variable which changes every racket bounce.

An increment change to either 7 or 9 or their equivalent negative values based on direction of the ball archived desired bouncing on the walls, example:


-7
-9
B1 B9 B17 B25 B33 B41 B49 B57
B2 B10 B18 B26 B34 B42 B50 B58
B3 B11 B19 B27 B35 B43 B51 B59
B4 B12 B20 B28 B36 B44 B52 B60
B5 B13 B21 B29 B37 B45 B53 B61
B6 B14 B22 B30 B38 B46 B54 B62
B7 B15 B23 B31 B39 B47 B55 B63
B8 B16 B24 B32 B40 B48 B56 B64

If the ball is coming with increment “-9” before it bounces of the wall at B25, the increment of “-7” at B25 will result in the ball bouncing to B18 which is more desirable.

To bounce the ball off the Rackets, it is done by using the coordinate system by the bouncing of the ball off the rackets based on presence of a racket on the location of the ball and then the current increment is turned negative to change the direction of the ball to the opposite location.

KEY INPUT

The Keys (K1,K2,K3,K4) are connected to pins P3.4,P3.5,P3.6,P3.7 respectively and also to Ground.
The function “getinput()” checks if these keys are placed by checking if the value is 0 showing the circuit is complete. If any of the keys are are pressed the racket position values are updated.

K1 & K2 for up and down for bottom racket (player1) and K3 & K4 for up and down for top racket (player 2) only if AI_Mode = 0.

DISPLAY

The display mechanism is very simplified by the coordinate system.
every game cycle the array called “map” gets cleared and the the “draw(uchar box)” functions is used to prints 1’s in all locations of the rackets and ball by their respective functions which also stores the current location of these objects in variables.

SCORE

With this Coordinate System, knowing is a score has happened is simply by checking if the ball location is trying to print a 1 either below B1 or over B64.

The lights on the Pin P1 show you the score of the game:

- Lights D1 to D4 show player1 score
- Lights D8 to D8 SHOW player2 score

The lighting used function xLed(), by giving P1 the required hex to switch on the Leds. Comments on that are included in the file "paulobetaX5.h".

GAME AI OPPONENT

In the Code if the variable "AI_MODE"= 1 , the game uses a primitive game Ai to play with you as 2nd player. It is not very advanced and dynamic but it can estimate where the ball will go and bounces it back to allow one player to play the Ping-Pong game with it.

In this single player mode the game can be played with keys K1 and K2 for left and right respectively of the bottom racket the top racket moves automatically.

The Game Ai works by calculating the location of the ball and decided whether to go up or down based on which row the ball is in.

This first method to archive this created a very basic repetitive movement of the racket controlled by the Ai, but the addition of a random seed number (0-2) created variation for the movement of the Racket controlled by the Ai opponent, hence also creating a sense of error to allow a human player to be able to score by chance.

If you have a friend and would like to play 2 player mode, the Variable "AI_MODE" can be set to 0 and then the:

- keys K1,K2 can be used for player 1 controlling the bottom racket
- keys K3,K4 can be used for player 2 controlling the top racket

GAME CLOCK

The game clock uses the system interrupt to constantly keep the ball moving by pushing it one column at a time based on the variable of game speed that is set.

ERROR SYSTEM

The error system was implemented to detect un accounted for occurrences that were not designed to happen to avoid damage to the game and also weird, faulty or undesired gameplay or logic, in simple terms if the logic is now working as intended the game stops implying that somewhere some logic was over looked or miscalculated.

It also states the error number by drawing dots to show which specific error occurred as referenced in the code.

A good example would be if by any chance the functions that draw the rackets and ball are requested to draw outside of the “B1-B64”, there a mistake has occurred as the game is intended to only happen inside this area. This would result in an error 4, and you would be able to know that the problem is about the drawing functions

Likewise the increments only range 7,8 and 9 and their negatives only and if by any chance it goes out of this range the logic of the game will be undefined and undesirable hence the game stops and reports error 2, by drawing 2 dots above the “er” sign.

This became very useful when developing and debugging this game as each time I would know where to start looking is there was a problem.