İZMİR KÂTİP ÇELEBİ UNIVERSITY

2010

TETRIS GAME

EEE316 MICROPROCESSORS

Halime Özge KABAK

180403001

Asst. Prof. Dr. Volkan KILIÇ

CONTENTS

ABSTRACT

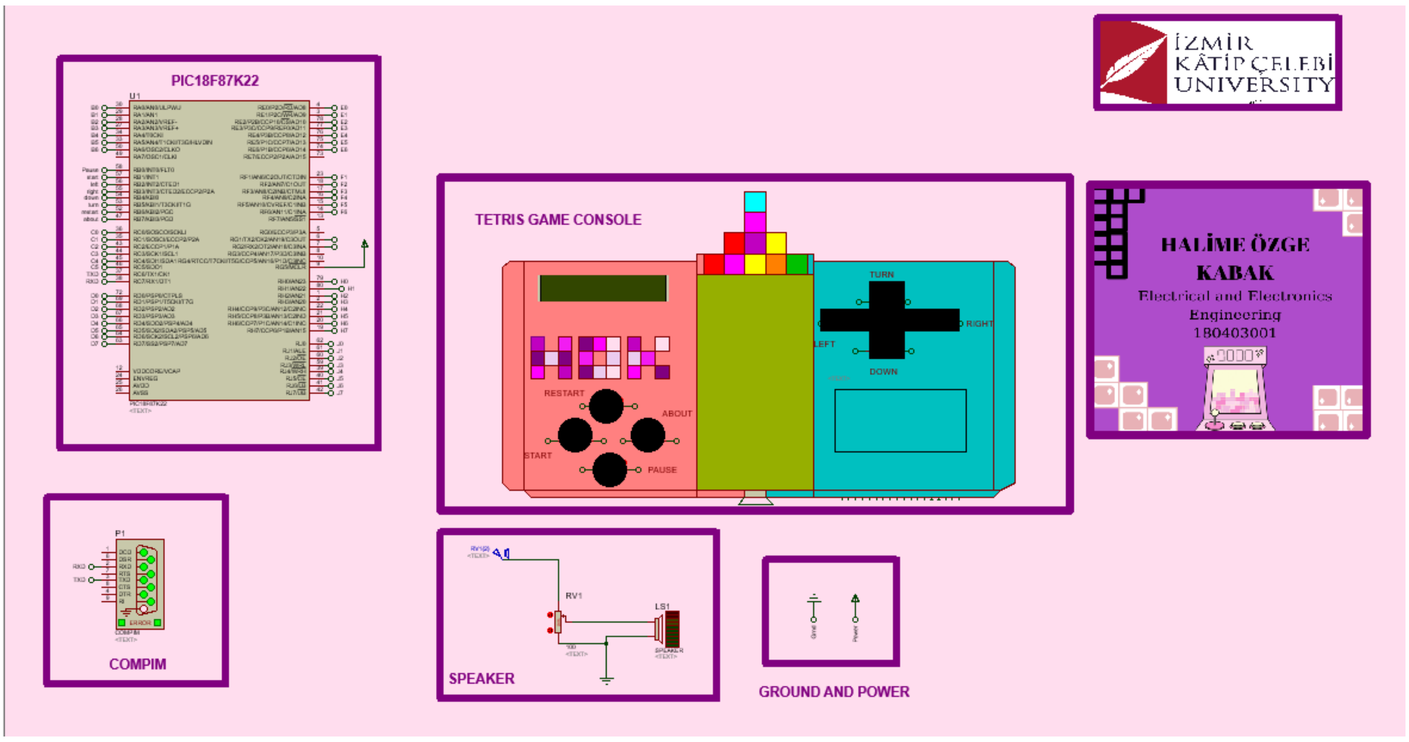
The aim of this project is to create the first version of the Tetris, one of the most famous games in the world, using PIC18 microcontroller. The game was realized in Proteus simulation environment. Unlike the classic Tetris game, thanks to the android application I added to the game, the game can be played with the help of voice commands or with the control keys over the phone.

In the first week, I did the necessary research to realize the project and drew the pictures to be used using Paint. For the second week, I struggled to operate the ILI9341 LCD. I wanted to use a large and colourful LCD because the most important component to realize the game is the LCD, but the LCD screen did not work. So, I used a black and white lcd screen which is smaller. In the third week, I created the menu buttons and made the drawings for each block. In the fourth week, I completed the motion buttons of the game and made the algorithm of the game and the necessary calculations. I did also carry out the design of the game on Proteus. Last week, I completed the game's deficiencies and fixed the errors that needed to be fixed. Finally, I prepared the Android application.

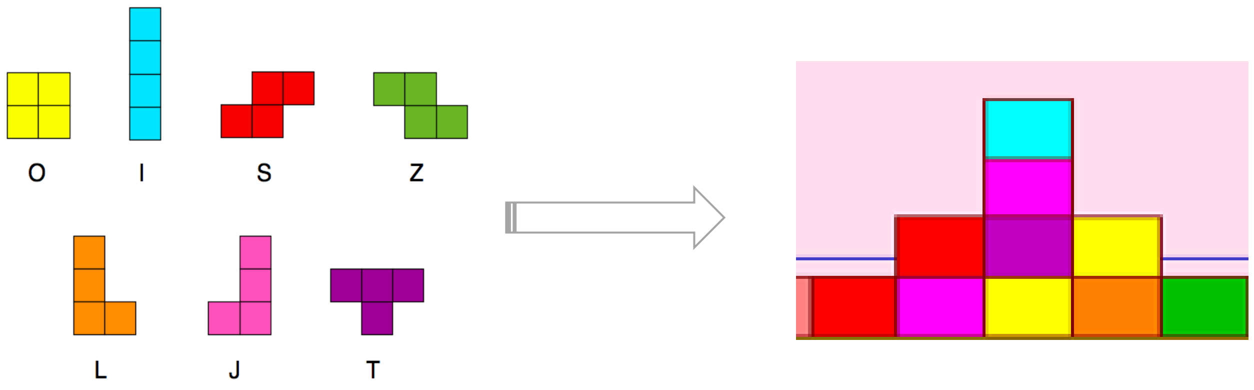
COMPONENTS

Component Type	Number Of Components	Component Name
Main GLCD	1	LM3229
GLCD	1	AMPIRE128X64
LCD	1	LM01F6L
Microcontroller	1	PIC18F87K22
Serial Port	1	COMPIM
Potantiometer	1	Pot-HG
Audio Generator	1	-
Speaker	1	-
Button	8	-

PROTEUS DESIGN



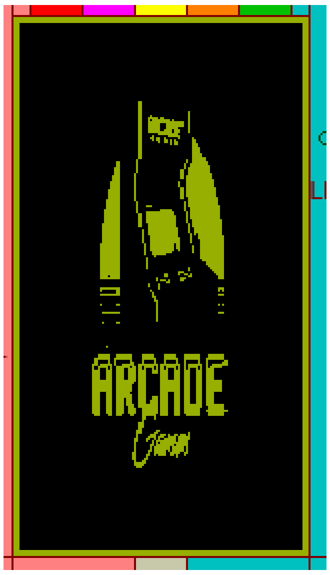
I made a game console shape for a Tetris on Proteus and placed my buttons and LCD screens on this game console. In the block stack on the top of the game console, all 7 basic blocks in the Tetris game are in the shape and the colours of the blocks are used in the design. I also added a logo with the initials of my name on the console using Tetris blocks. I highlighted the other components by enclosing them in rectangles. I also designed my own logo and added both the school logo and my own logo to Proteus as png files.



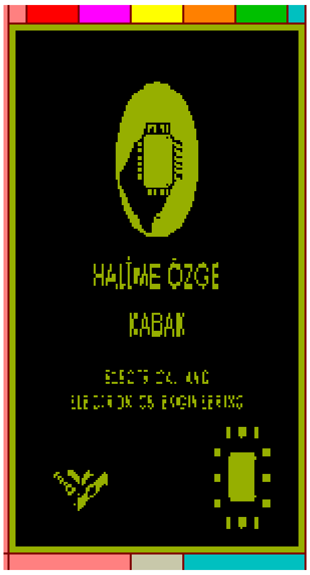
PROJECT EXPLANATION

1.INTRO PART

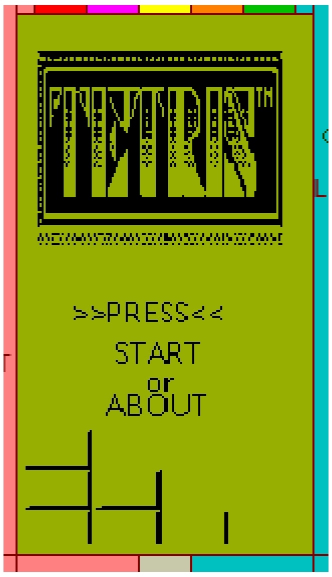
I used the screen in an upright position so that the LCD I used in the project was suitable for the game. I determined the x and y axes of the LCD according to its original state. In this case, the part with 240 pixels became the x-axis, and the part with 128 pixels became the y-axis. When the game is first opened, the arcade games logo appears, and right after that, the picture I prepared for the game and my information is displayed. Then the opening screen of the Tetris game opens. On this screen, if desired, the game can be started by pressing the Start button or the information about me can be accessed by pressing the About button.



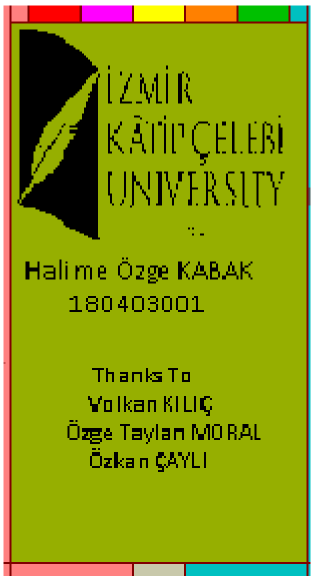
1



2



3

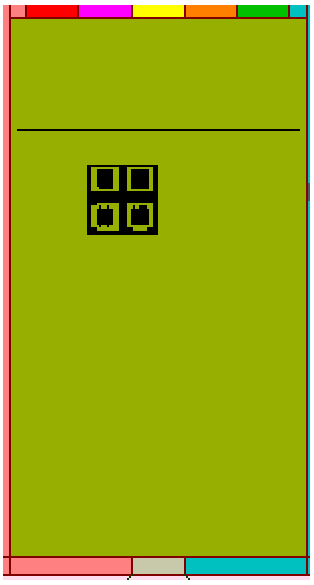


İZMİR
KÂTİP ÇELEBİ
UNIVERSITY

Halime Özge KABAK
180403001

Thanks To
Volkan KILIÇ
Özge Taylan MORAL
Özkan ÇAYLI

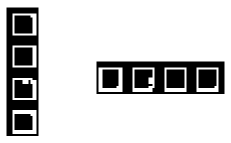
ABOUT



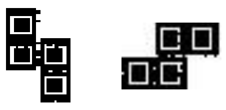
START

2.BLOCKS

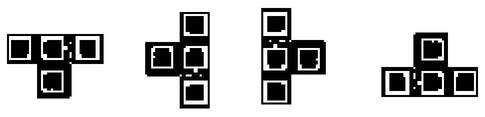
There are all the blocks found in the my Tetris game. There are also versions of all the blocks as they are rotated.



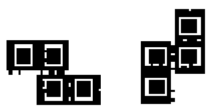
I BLOCKS



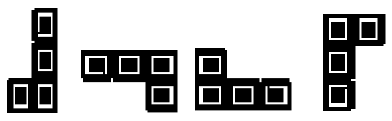
S BLOCKS



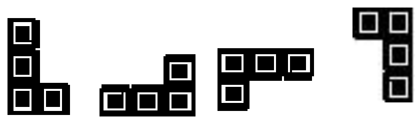
T BLOCKS



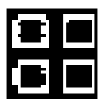
Z BLOCKS



J BLOCKS



L BLOCKS

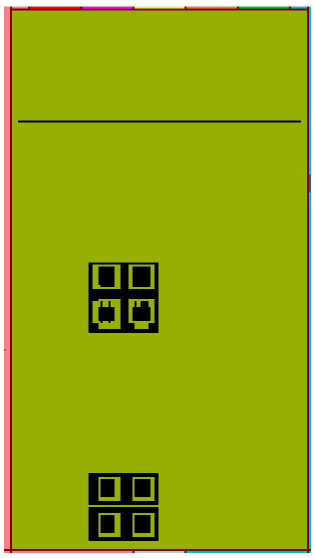


O BLOCK

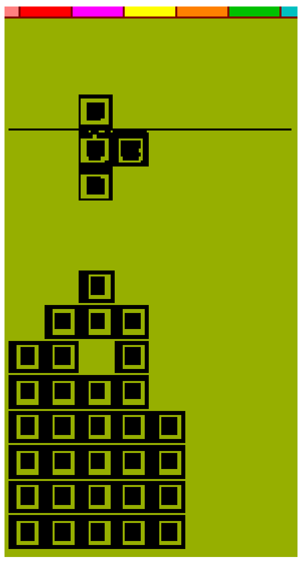
I used the rand() function to get the blocks mixed up, so that one of 7 different basic shapes can come randomly. I saved these numbers in an array so that I could show the next incoming block in GLCD at the next stage. The blocks can be moved left and right as long as the LCD screen allows. Two different buttons are used to move the blocks. When the buttons are active, the positions of the blocks on the y-axis increase and decrease, allowing the blocks to be replaced. I also prepared the fast bring down button that was in the original of the game. When this button is activated, the positions of the blocks on the x-axis decrease more and they descend rapidly. Another move button I added is the rotate button, when this button is pressed, the blocks are set to rotate 90 degrees with each press. In this case, the I block can only rotate in one different I shape, the O block does not change, and the S and Z blocks can rotate in one different shape. Finally, L, J and T blocks can rotate in three different directions except their original shapes. In order to ensure the rotational movements of the blocks, separate drawings are made for each block, and it is ensured that one picture goes and the other is loaded at each button press. Since the dimensions of each block are different, the length and width values for each block are assigned by changing. While this process takes place, the blocks go down by decreasing continuously on the x-axis in the while(1) loop, while after each descent, the block is deleted from its old place and loaded into the new place. I wrote clean() and drawblock() functions for this.

3.GAMEPLAY

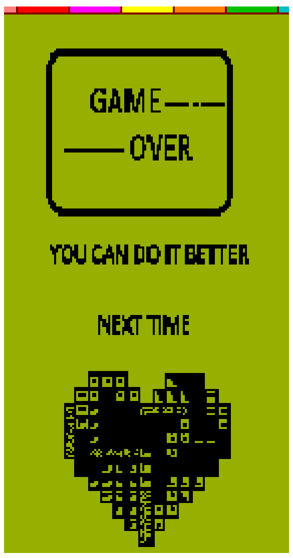
I used an interrupt in the game, thanks to this interrupt function, the game can be stopped and restarted. While the game is stopped, a very long delay comes into play, then when the pause button is pressed again, the delay time is shortened and the game starts again. In addition, thanks to the restart button I created, the game returns to the initial login screen when the button is pressed while the game is running. I developed a function to assign the location of each as the blocks descend. Thanks to this function, the position of the LCD on the x and y axis is determined according to the incoming block and saved in two different LCD arrays. In this way, the location of the incoming blocks can be found. Since the descending blocks were pictures, they could overwrite each other. To avoid this situation, after I reach every block down, I placed a block for each block occupied by the blocks. In this way, overwriting is prevented. The locations of the blocks are checked after each descent. If the area under it is full, the block stays there. In addition, each line is controlled by the lineclear() functions. If the line is full, the part containing the line is deleted by resetting it. If the rows above are occupied, their position is also reduced to a lower row. There is a score board on the LCD screen. In the score board, 80 points are earned for each deletion of a row, and the number of deleted rows is shown in the line section when each row is deleted. In addition, the best score is kept on the LCD screen. If the location size of the blocks on the x-axis reaches 192, the game becomes game over and there is a line in that part. When the line is crossed, the game ends and the game over screen appears. Then the game returns to the initial menu. Meanwhile, on the other GLCD screen, the picture of the next block is shown there, according to the order of the block that I have stored in the array.



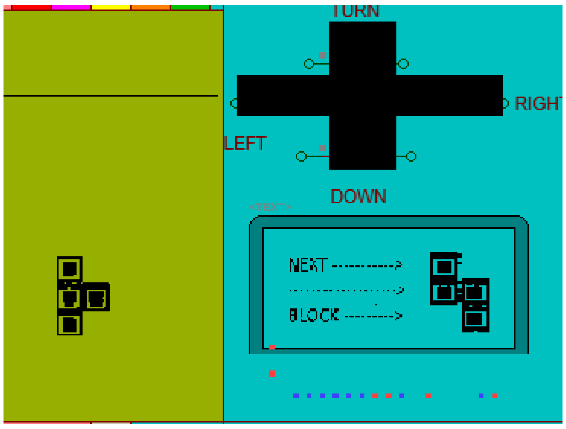
Blocks change after stopping because each block is printed one by one according to their location.



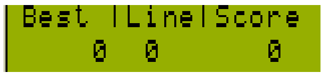
Blocks stack on top of each other.



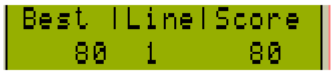
Game over screen after the blocks reach the line above.



Displaying the block that comes during the game and the next block on the GLCD screen.



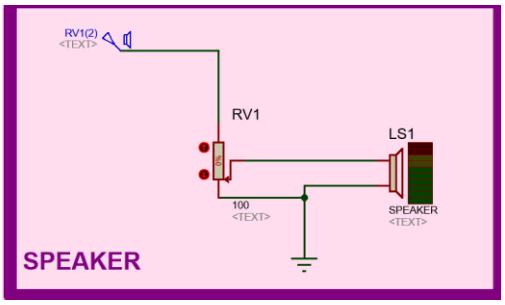
```
Best |Line|Score
  0   0      0
```



```
Best |Line|Score
 80   1     80
```

Score Boards

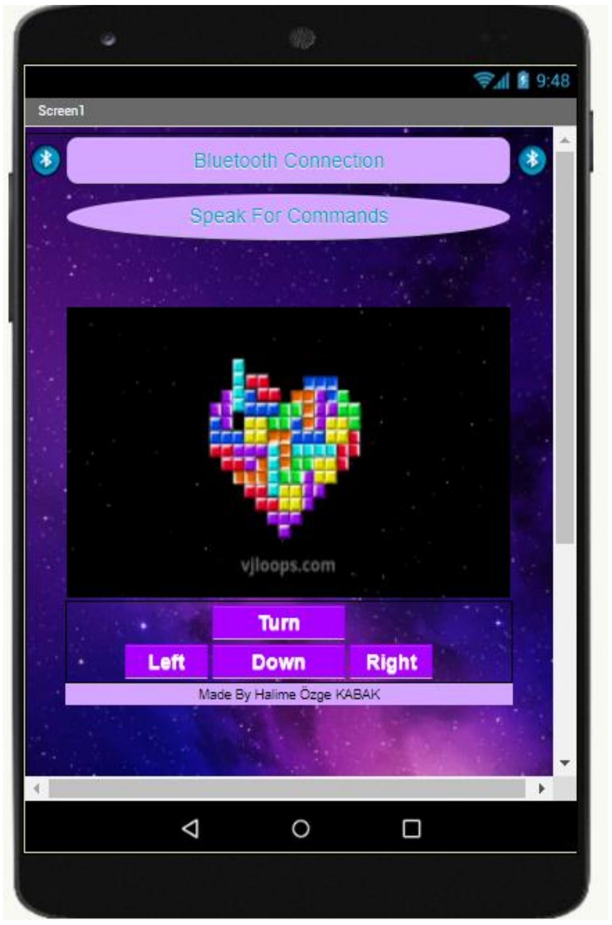
4.SPEAKER



Then I added the intro music of Tetris to the game by using an audio generator for the music effect in the game and played the music with the help of the speaker. With the help of the potentiometer I added, the volume of the music can be turned down and increased.

5.ANDROID APPLICATION

Finally, I created a connection between the serial communication port of my computer and COMPIM, and I can execute the commands I gave from the application on Proteus with the help of Bluetooth communication using UART in the code. I wrote the commands I want this to happen inside the Bluetooth function. Then I used MIT App Inventor to build the app. First of all, I added the necessary buttons and components to the application and created its design. Then I prepared the code blocks for the operations that need to be done according to the commands coming in the background. I downloaded the MIT App Inventor application on the phone and from there I connected my phone and my computer via Bluetooth while Proteus was running. Then I could play the game over the phone.



The design of the application that appears on the screen:

Bluetooth assistance is provided with the help of the button at the top.

When the button under it is pressed, the voice recognition feature is activated and commands such as "Turn", "Right"," Left", "Down" can be given from here.

The desired commands can be performed with the help of the arrow keys at the bottom.

ENCOUNTERED PROBLEMS AND SOLUTIONS

✗ The first problem I encountered was the ILI9341 LCD screen not working. To solve this problem, I tried to run the lcd by making connections in different ways, but I could not succeed.

✓ To solve this problem, I used the smaller and colourless LM3229 LCD screen.

✗ Another problem I have encountered is that the ultrasonic distance sensors I want to use do not work.

✓ To solve this problem, I gave up the motion sensors and decided to add a GLCD instead to show the next block.

✗ The blocks did not overlap. For this reason, they did not accumulate. I created an algorithm to solve this problem, but it didn't work.

✓ After researching the code for a long time for the solution, I realized that my algorithm was not working because the if else structures in Mikroc did not work properly. I tried other commands to solve this but it didn't work. That's why I wrote functions that manually save and check the locations for each block one by one, even if it takes very long lines.

✗ I wanted to create a multidimensional array of 240 by 128, which is the size of the lcd, to record the location of the arrays, but I got an error in Mikroc.

✓ To solve this problem, I created 2 different arrays and defined separate arrays for x, y axes.

✗ Since each of the blocks was a separate picture, they were overwriting each other.

✓ To solve this problem, I used an image that was a single block size and fixed it by placing a block for each location. In this way, it was easy for the lines to slide down again after they were deleted.

✗ I made a speaker connection with the audio for the intro music, but Proteus gave a CPU error and could not play the song properly.

✓ To solve this problem, I ran the speaker in another Proteus project.

✗ Finally, when I finished the codes, the remaining ROM of the PIC dropped to 2% and I even got an error that there was no room left. Also, the blocks slowed down a lot in Proteus.

✓ To solve this problem, I deleted or commented some lines of code, as I cannot change the PIC. I was able to write functions for basic blocks but not for rotating blocks.

FUTURE WORKS

❖ Another arcade games can be added such as Brick Breaker, Quiz Show, Galaga, Asteroids.
❖ Found errors can be fixed and a more accurate game can be revealed.
❖ Different difficulty levels can be added to the game.
❖ Other added games can be controlled via the application with the help of voice commands and keys.
❖ Sound effects can be added as blocks descend and move.
❖ By changing the LCD screen, the blocks can be coloured and a larger area can be created for the game.
❖ Analysing and shortening the code algorithm and thus adding more features.

- By adding motion sensors, the game can be played with the help of motion control.
- The game can be designed to be played as two players.

CONCLUSION

When I started the project, I had some difficulties even while operating the LCD screen. As I progressed in the project, I had the opportunity to develop myself more in coding microcontrollers, so I could use much more components, I also improved myself in c programming and was able to write my codes much faster. Thanks to this project, I both developed the theoretical knowledge I learned in the course and gained experience in creating algorithms myself. I also gained experience in subjects such as designing pictures and designing games.