

**An-Najah National University**

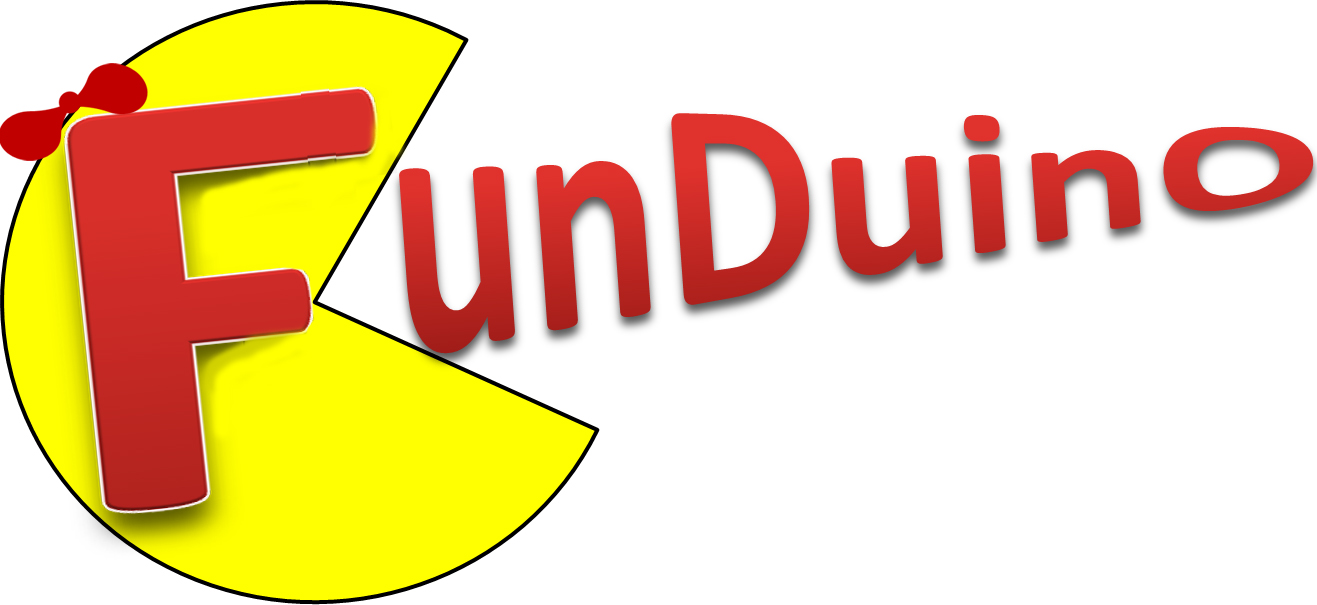
**Faculty of Engineering**

**Computer Engineering Department**

|  |  |
| --- | --- |
| **Hardware graduation project documentation** | **FunDuino** |

**Supervised by:  
Dr. Luai Malhis.**

**Prepared by:  
Suad Seirafy/ Fatima Zubaidi**

****

**2012**

**الإهداء**

**الحمد لله كما ينبغي لجلال وجهه وعظيم سلطانه ... الحمد لله حمداً كثيراً مباركاً ملء السموات و ملء الأرض و ملء ما شئت يا ربنا من شيء بعد. و الصلاة والسلام على عبده الذي أسري به إلى المسجد الأقصى, وعرج به إلى السموات العلى..**

**وإلى فلسطين الأبية نقدم ثمرة جهدنا المتواضع, و إلى جامعتنا حماها الله التي وضعنا اسمها " النجاح" نصب أعيننا, إلى من رعانا و مهد لنا سبل النجاح أساتذتنا المخلصين حفظهم الله و رعاهم.. و نخص بالذكر الدكتور لؤي ملحيس,الدكتور رائد القاضي ,الدكتور سامر العرندي والدكتور هنال أبو زنط وكما نشكر أساتذتنا في قسم هندسة الحاسوب الذين دائما ما حرصوا على ان نصل الى هذه المرحله من العلم والمعرفة , نشكرهم جميعا على جهودهم .**

**وإلى الوالدين العزيزين نبض القلب وشفاء الروح....**

**الى كل هؤلاء نهدي مشروعنا.. اللهم ما كان من توفيق ونجاح فمن عندك.. و ما كان من تقصير و أخطاء فمن أنفسنا.. نسأل الله التوفيق و الإخلاص في العمل..**

# 

# Acknowledgements

We wish to express our deep Gratitude to Allah (SWT), who owns the first and last thank, and our parents in the second place as they are the ones who brought us to this world, raised us up so we can now work on our studies.

We are indebted to our Project supervisors Dr. Luai Malhis, Dr. Ra’ed Al-Qadi, Dr. Samer Al-Arandi and Dr. Hanal Abu-Zant (An Najah National University) for their critical reading of an earlier version of this paper and for making many constructive comments and suggestions that have helped us shape it into its present form.

A continuous thank for professors and friends who were giving us the promotion and the courage every time we felt down.

We thank them all for their contribution in a way or another, May Allah blesses them all.

**Abstract:**

FunDuino is a video games system based in tiny microcontroller at which the user can control the game wirelessly. The project presents an approach to display VGA signals from a tiny microcontroller to a monitor in order to allow the users to play video games using the VGA output. There are limits to how much useful information you can display on a basic character LCD, or even a dot matrix LCD, since most have low resolution, no color, and a poor refresh rate. We came up with this problem by finding an effective way to display the data and especially the video games on a VGA monitor or connecting it with a projector.

In this project we achieved a big trend which is referring to the current trend “eliminate the hardware as much as possible, make the hardware small as much as possible, build systems that depend on tiny microcontrollers without using the real computer”

The user can play with the controller that is connected wirelessly and control it interactively by moving the arm of the game in a specific way.  
The final product could be used to play demonstrative video games with different levels and can be played interactively with two players.

**1.1 TABLE OF CONTENTS:**

**1 TABLE OF CONTENTS AND FIGURES...............................................................1**

**1.1 Table of Contents.………………………………………………………….1**

**1.2 Table of Figures…………………………………………………………….2**

**1.3 Tables ………………………………………………………………………3**

**2 INTRODUCTION...................................................................................................... 4**

**2.1 Project Overview......................................................................................... 4**

**2.2 Project Applications.................................................................................... 5**

**2.3 Report Overview…….................................................................................. 6**

**3 SYSTEM DESCRIPTIONS….……...........................................................................7**

**3.1 Hardware Components……………………..……………………….……..7**

**3.2 Software Information……………………….……………………….……..7**

**3.3 System Constraints…..…………..………………………………….……...8**

**3.4 System Flow Chart..…………..………………………………….…….......8**

**4 PROJECT PROGRESS............................................................................................ 35**

**5 PROBLEMS AND SOLUTIONS............................................................................. 35**

**7 CONCLUSION......................................................................................................... 35**

**7.1 Project Goals.................................................................................................35**

**7.2 Future Work.…….........................................................................................35**

**8 REFERENCES........................................................................................................... 36**

**9 APPENDICES............................................................................................................. 37**

**9.1 Data Sheets ……………………................................................................... 37**

**1.2 Table of Figures**

Figure 1: Asteroids Game…………………….…………………............….…...8  
Figure 2: Pacman Game……………………….………………….......................9  
Figure 3: Frogger Game.…………..………….....…………………….………....9

Figure 4: Basic Circuit Connections…………….……….…………………..…11

Figure 5: Accelerometer MMA7260Q sensor.….……………………...…..…..12

Figure 6: Flex Sensor.…………………………….………………………….....12

Figure 7: XBEE Module...……….……………….……………………….....…13

Figure 8: System Flow Chart……………………….…………………….….…14

Figure 9: User Controller Basic Circuit..…….……….…………………...........15

Figure 10: PIC C Read Accelerometer Code..……….…………………….…..16

Figure 11: Accelerometer connections with the PIC . ……………………........16

Figure 12: PIC C Read Flex Sensor Code ………………………………….….17

Figure 13: Flex Sensor Connections with the PIC…..…………………….…...17

Figure 14: XBEE connection with the PIC……...……………………………..18

Figure 15: Gameduino Adapter...……………………………………………....19

Figure 16: Our own XBEE Receiver Shield...…………………………….....…19

Figure 17: Our First Arduino Code……………………………………….……20

Figure 18: Frogger Game Background….…………………………….…..........21

Figure 19: Frogger Game Sprit.………….……………………………………..22

Figure 20: XBEE Communications...……....……………………………….......23

**1.3 Tables:**

Table 1: Gameduino Connection……………………...……………..….19

**2. Introduction:**

The goal of this project is to create a way that is capable of outputting VGA signals from the microcontroller to a monitor in order to allow the users to play video games without using the real computer and controlling the game wirelessly with the help of some sensors that can detect the control signals.

We’ve designed a game by programming it in the microcontroller with different levels and allow two players to interactively play.

This project was chosen because it effectively combined different types of components in an effective way. In addition, we all enjoy video games, and were curious if it was possible to construct an entire video game system using a microcontroller, instead of simply the software side.

For quite some time we’ve wanted to design a way to output VGA signals from the scratch. Our main goal was of course to be able to display whatever we wanted on the monitor with as much colors as possible besides that the refresh rate should be enough to display video games. After extensively research we became very frustrated to find little information on the topic. So we decided to go ahead and expand on the information we found and do this project with as best output quality as we can.

**2.1 PROJECT OVERVIEW:**

In our project, we used a main microcontroller that is responsible to produce the VGA signals with the help of other components, and we used another microcontroller for the user controller part. The two are connected with each other wirelessly, so that the user controller reads the data from the sensors that we used to control the game – We used the accelerometer and the flex- and sends them by using the XBEE to the main microcontroller that is connected with the monitor to display the game. In the case of the two players, we used another copy of the user controller that also sends the data to the main one.

We chose the main controller to be ATmega328 with 16 MHz frequency to be able to process the VGA signals as fast as possible. The hard part was to find the suitable way to produce the VGA signals themselves.

At the first days, we’ve wanted to design a way to output VGA signals from the scratch as our main goal was of course to be able to display video games on the monitor with as much colors as possible besides that the refresh rate should be enough to display video games. After extensively research we found that the best colors and refresh rate are found in the gameduino, which is a game adapter with an SPI interface built as a single shield that works with the controller and has plugs for a VGA monitor and speakers. The adapter which uses the FPGA technology is controlled via SPI read/write operations, and looks to the CPU like a 32Kbyte RAM. With the help of the gameduino we were able to produce 400\* 300 pixels to the VGA monitor. In this part also, we have the XBEE receiver that is connected using Tx and Rx with the ATmega328.  
 We’ve designed the game in a way that can be controlled via the user friendly.  
 For the user controller part, we used PIC18F4620 as the microcontroller, which will read the sensors values from the ADC and send them to the XBEE sender serially, and the XBEE sender will send the data to the XBEE receiver on the main controller part. The sensors that are used are the accelerometer which can bring the x, y and z axes of the user, and the flex sensor which can detect whether the user’s finger is bent or not and these values are enough to control the game.

**2.2 PROJECT APPLICATIONS**

The games that we implemented using our hardware are:

**1. Asteroids:**

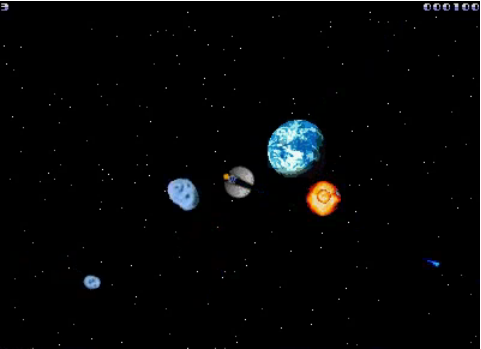
This is our first game, which is considered as our first step after the integration between the user controller side and the main side, we used an existing demo from Gameduino and developed it. We controlled this game wirelessly by the controller’s accelerometer and flex sensor. Here the user can rotate the space shuttle by rotate the accelerometer (x-direction) and shoots towards the incoming asteroids by folding the flex sensor.

Figure 1: Asteroids Game.

**2. Pacman:**

This our first complete accomplishment in the gameduino world, we designed the background then put the ghosts and the pacman itself then we implemented the game logic (moving the ghosts and the pacman, the collision, and the score) then we allow to control it wirelessly via the user controller’s accelerometer (x-direction, y-direction), after the success of this, we developed it to be multi-level game, such that the user can choose the level he wants via moving the accelerometer up and down and pick the level by folding the flex sensor.

Figure 2: Pacman Game.

**3. Frogger:**

We designed and implemented the Frogger game which the user can play it wirelessly via the accelerometer (x-direction, y-direction), first we designed it to be a one-player game that means one frog in the game, then we developed it to be an interactive game (2-players game) which means two frogs in the game each one is controlled via a user’s controller wirelessly, one user will be the winner at the end.

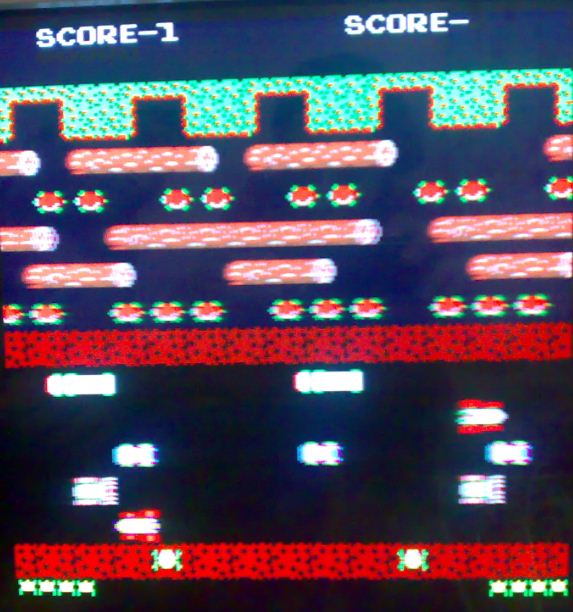
****

Figure 3: Frogger Game.

**2.3 Report Overview:**

* Description of the hardware components, software used in our project will be covered in chapter 3.
* The Project Progress will be covered in chapter 4.
* The challenging problems that we’ve faced and how we covered them will be in chapter 5.
* The Conclusion and Future Work in chapter 6.
* The References that helped us will be mentioned in chapter 7.
* Data Sheets for the hardware that are used will be explained in the Appendices in chapter 8.

**3. SYSTEM DESCRIPTIONS:**

**3.1 Hardware Components**

**User Controller Side (2 controllers):**

**-Basic circuit:**

* 2 x PIC18F462
* 4 x 100nf capacitor
* 2 x 6.8 mf capacitor
* 2 x 33k resistor
* 4 x 12pf capacitor
* 2 x 1k resistor
* 2 x MAX232
* 8 x 1mf capacitor
* 2 x 100nf capacitor
* 2 x connector DB9

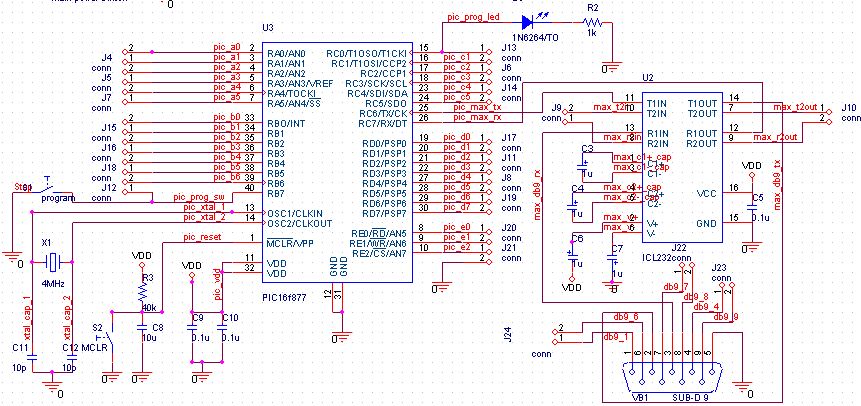
****

Figure 4: Basic Circuit Connections.

**-2\* Accelerometer MMA7260Q (Tri-Ax breakout)**

“±1.5g - 6g Three Axis Low-g Micromachined Accelerometer”

The MMA7260Qis a low cost capacitive micromachined accelerometer features signal conditioning, a 1-pole low pass filter, temperature compensation and g-Select which allows for the selection among 4 sensitivities. Zero-g offset full scale span and filter cut-off are factory set and require no external devices.

Features

. Selectable Sensitivity (1.5g/2g/4g/6g)

. Low Current Consumption: 500 μA

. Low Voltage Operation: 2.2 V . 3.6 V

. High Sensitivity (800 mV/g @1.5 g)

. Fast Turn On Time

. Integral Signal Conditioning with Low Pass Filter

. Environmentally Preferred Package

. Low Cost



Figure 5: Accelerometer MMA7260Q sensor.

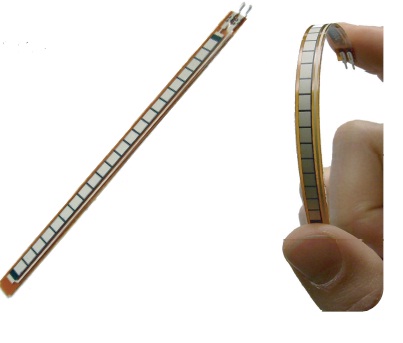
**- Flex sensor:**

Figure 6: Flex sensor.

Flex Sensor is a component that changes its resistance when it’s bent.



**-2\* XBEE 4214A S1 transceiver**

We used the XBEE module in order to send the data from the user controller to the main part.

Figure 7: XBEE Module.

**Main Microcontroller:**

* ATmega328 Microcontroller with 16 MHz clock speed.
* Gameduino Adapter.
* XBEE 4214A S1 transceiver: We used the XBEE module in order to receive the data from the controller.

**3.2 Software Information**

**PIC C Compiler:**

In the user controller part to write the C code for PIC18F4620 microcontroller and produce the hex file.

**Arduino0023 Compiler:** In the main part to program the ATmega328 microcontroller.

**SPI and GD Libraries:**

helped us dealing with the gameduino.

**Gameduino Online Convertor Tool:** To convert the backgrounds and the sprites used in the games to corresponding header files.

**3.3 System Constraints:**

The user should be within the 100 meters around the main controller because it’s the maximum distance that the XBEE support which can be considered as a good distance, and no need to be more than this.

**3.4 System Flow Chart:**

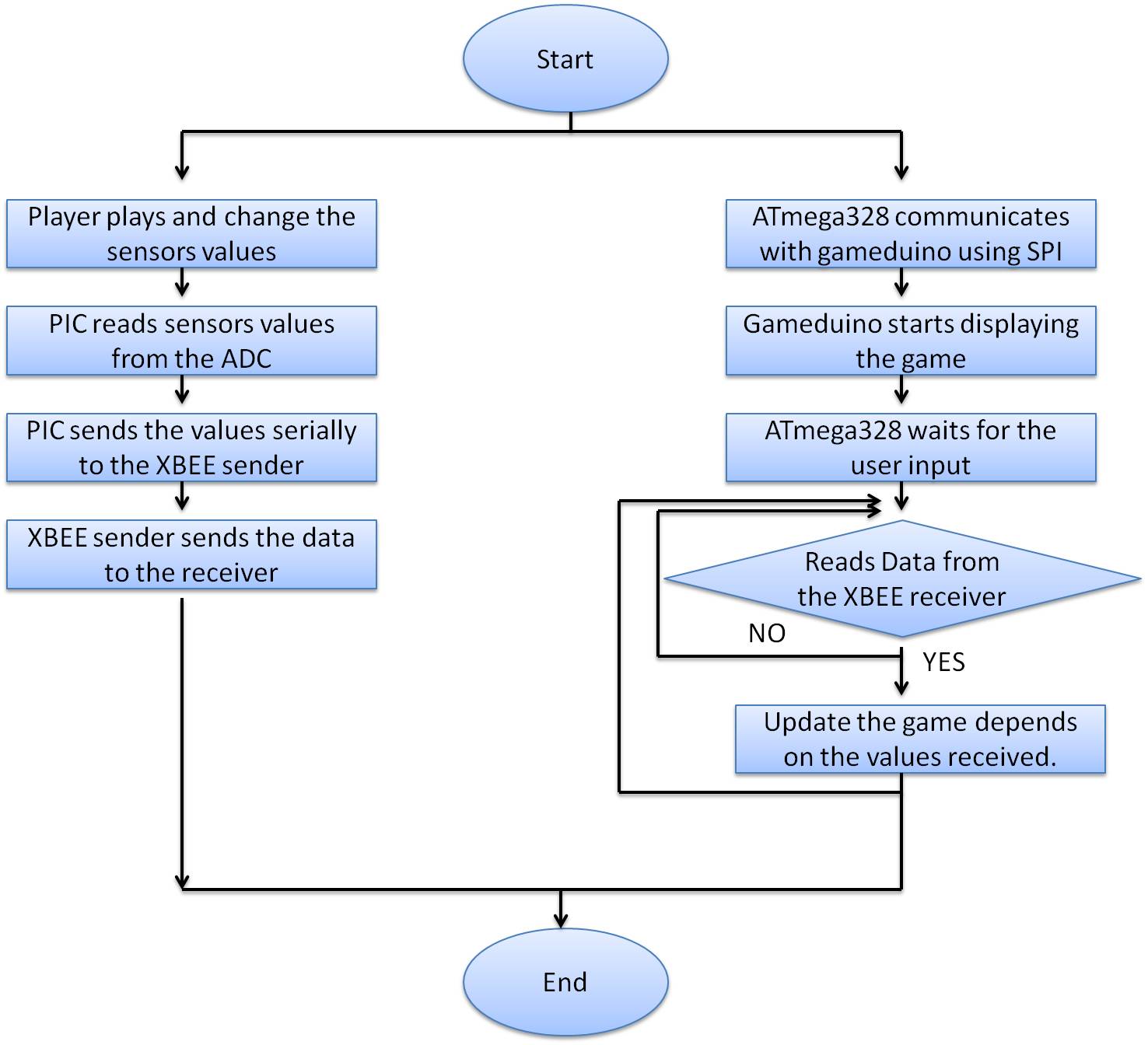
****

Figure 8: System Flow Chart.

**4. PROJECT PROGRESS:**

* **User Controller**
* Design the two user controllers basic circuits for the PIC18F4620, The basic circuit and all of its connections can be found in the Appendix. Then we installed the boot loader on it and check if it is working properly.

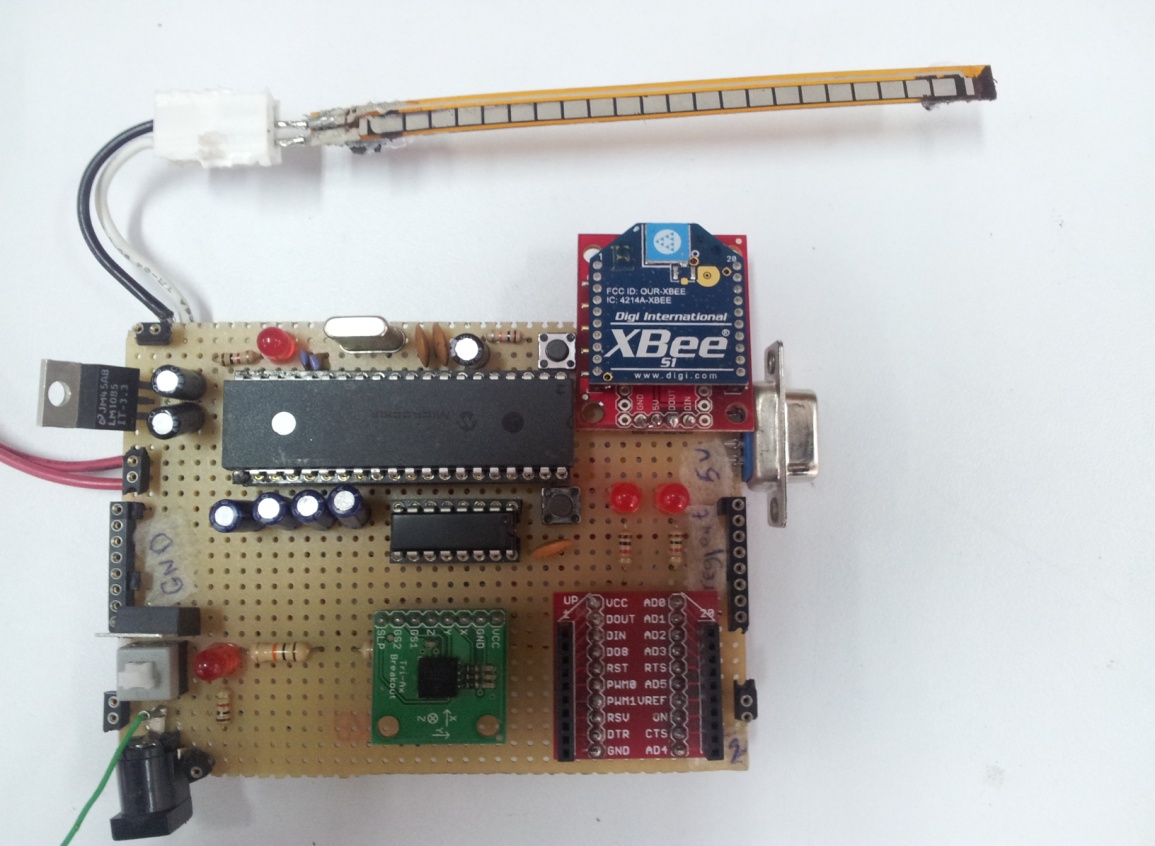


Figure 9: User Controller Basic Circuit.

* Connect the accelerometer to the PIC18F4620 through the ADC. We’ve connected each axis output on an ADC channel then we tested it by reading its values – the x, y and z axes- via the hyper terminal while changing them by rotating the accelerometer each time at an axis and check the values that it gives.  
  We wrote a function that reads the values from the ADC several times and take the average one at each time:

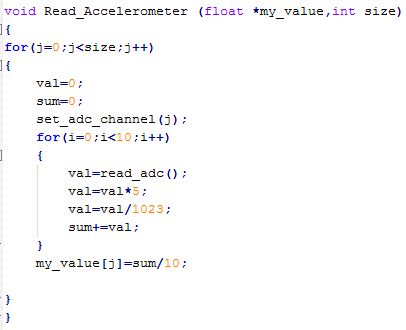


Figure 10: PIC C Read Accelerometer Code.

We noticed that if we rotate the accelerometer towards +X axis, it gives a value that is above x1 and if we rotate it towards –X axis, it gives a value below x2 and the same for Y and Z. So we were able to detect the direction of the accelerometer movement by the user.  
Note: x1, x2, y1, y2, z1 and z2 values were taken through the experiments and their values are shown in the appendix.

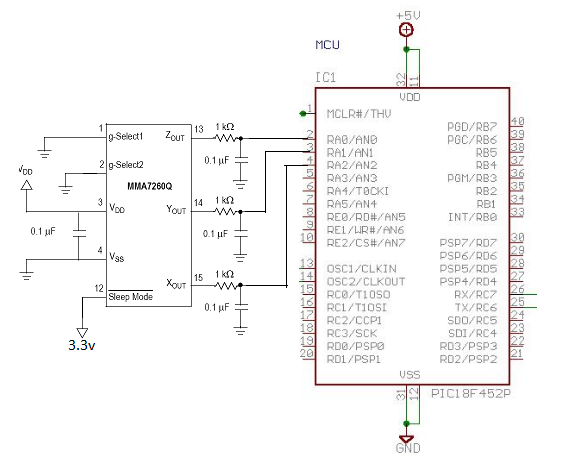
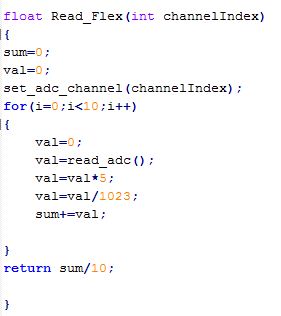


Figure 11: Accelerometer connections with the PIC .

* Connect the flex sensor to the PIC18F4620 through an ADC channel. We tested it by reading the value and find a suitable threshold at which we can determine whether the flex is folded by the user or not.

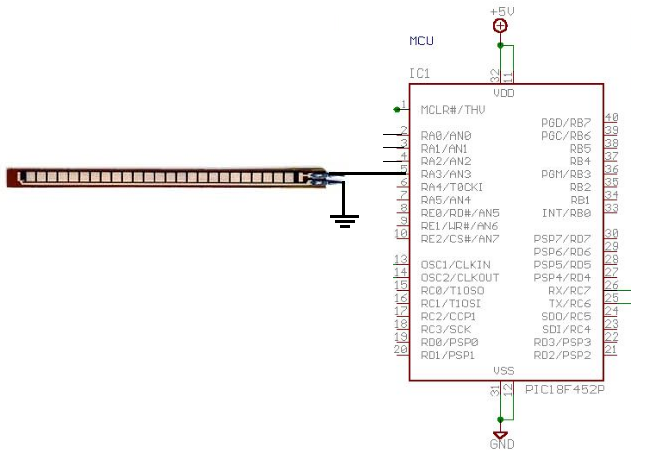


Figure 12: PIC C Read Flex sensor Code.

Figure 13: Flex Sensor Connections with the PIC.

* Connect the XBEE (sender) to the PIC18F4620, We used the XBEE to send the accelerometer rotating direction and the flex status to the main side in order to control the games.

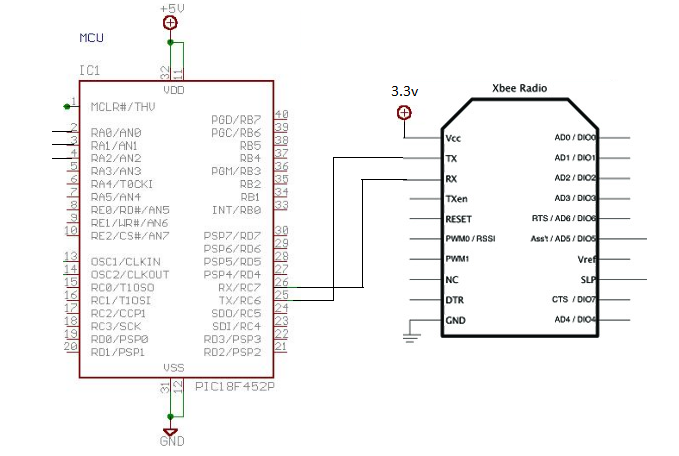


Figure 14: XBEE connections with the PIC.

* **Main Microcontroller:**

We brought the Arduino Uno from Sparkfun, installed the arduino-0023 software in order to program the ATmega16U2 via the cable and getting started with it.

* Gameduino

Brought the Gameduino from Sparkfun, installed its libraries in order to use them during the programming.

Since Gameduino is a game adapter with an SPI, it has plugs for a VGA monitor and stereo speakers. We stacked it on the top of our microcontroller and plugged it to monitor in order to display the games later, also connect the audio output to a speakers in order to hear the sounds that will be generated in games later.

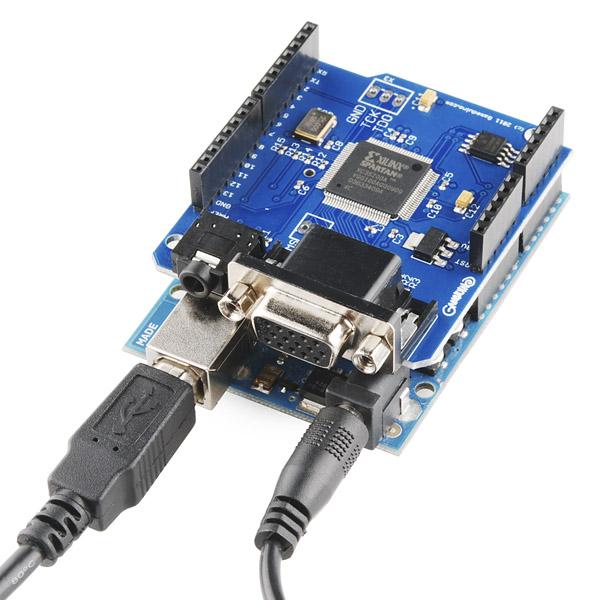


Figure 15: Gameduino Adapter.

Then we started learning how to write sample of codes and sent it to the Arduino in order to test the graphics, sounds, colors, and the animations.

The Gameduino required connections are:

|  |  |
| --- | --- |
| GND | Signal ground |
| 3.3V | VCC |
| 5V | Main supply: 3-6V |
| 11 | SPI MOSI |
| 12 | SPI MISO |
| 13 | SPI SCK |
| 9 | SPI SEL |

Table 1: Gameduino connections.

* Designing our own XBEE receiver shield and connecting it via Tx and Rx to the microcontroller.

We wrote a code in order to read the received data from the XBEE sender.

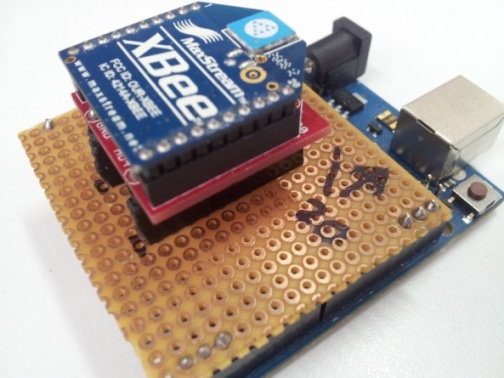


Figure 16: Our own XBEE receiver shield.

Figure 1: XBEE connections with the PIC.

* Control a single sprite on Gameduino via the accelerometer. We allowed the control of a sprite via the controller accelerometer.

This Sprite was designed by us with the help of the online tool convertor.

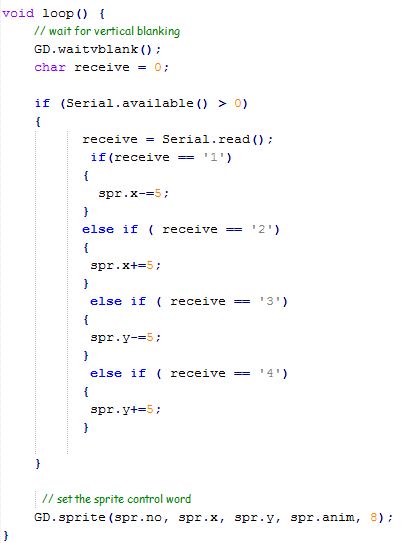
This step is considered as the start of designing games and playing them via the controllers.

Figure 17: Our First Arduino Code.

* **Asteroids:**

This is our first game, which is considered as our first step after the integration between the user controller side and the main side, we used an existing demo from Gameduino and developed it. We control this game wirelessly by the controller’s accelerometer and flex sensor. Here the user can rotate the space shuttle by rotate the accelerometer (x-direction) and shoots towards the incoming asteroids by folding the flex sensor.

* **Pacman**:
* Design the background.
* Draw the ghosts and pacman sprites.
* Implement collisions, scores and the full played game.
* Convert it to multi-level game by increasing the number of ghosts in each level and change the maze.
* **Frogger:**
* Designed the background with the help of the online convertor.



Figure 18: Frogger Background.

* Wrote the code that updates the score.
* Drawing and animation of sprites (cars, road, river, frog … etc).



Figure 19: Frogger Sprites.

* Generate sounds depending on cases.
* Integrate all things together and implement the full game.

**5. PROBLEMS AND SOLUTIONS:**

* The big problem that we faced was the communication between the two microcontrollers and especially in the ATmega328 side. It was not receiving any data using the regular XBEE shield. The problem was solved just when we built our own shield for the receiver in a way that reduces the noise as much as we can as shown in the figure below:
* Another problem was dealing with the gameduino and getting started with it as it’s a new technology and the online resources are limited.
* When we developed our frogger game to be 2-players (interactive) this required 2 microcontroller for the players side that means two XBEE senders and one XBEE receiver, receiving from 2 senders made conflict in playing and need synchronization, this was solved by synchronization the 2 XBEEs.

Figure 20: XBEE communications.

**6. CONCLUSION:**

In today’s digitized world, small sized solutions in every field are needed.  
We were able to produce wirelessly controlled video games by using tiny microcontrollers.

**7.1 Project Goals**

The goal of this project was to create a way that is capable of outputting VGA signals from the microcontroller to a monitor in order to allow the users to play video games and controlling the game wirelessly with the help of some sensors that can detect the control signals.

We’ve designed a game by programming it in the microcontroller with different levels and allow two players to interactively play.

**7.2 Future Work**

As a future work for our project, we are thinking to develop it to be content driven game, also there is a plan to add SDRAM to the main controller in order to include huge gaming systems (a lot number of games with multi levels).

**7. REFERENCES:**

1. Arduino Board Uno

<http://arduino.cc/en/Main/ArduinoBoardUno>

1. Gameduino

<http://www.sparkfun.com/products/10593>

<http://excamera.com/sphinx/gameduino/>

1. Gameduino Online Tools:

<http://gameduino.com/tools/>

1. XBEE Communication:

<http://www.digi.com/xbee/>

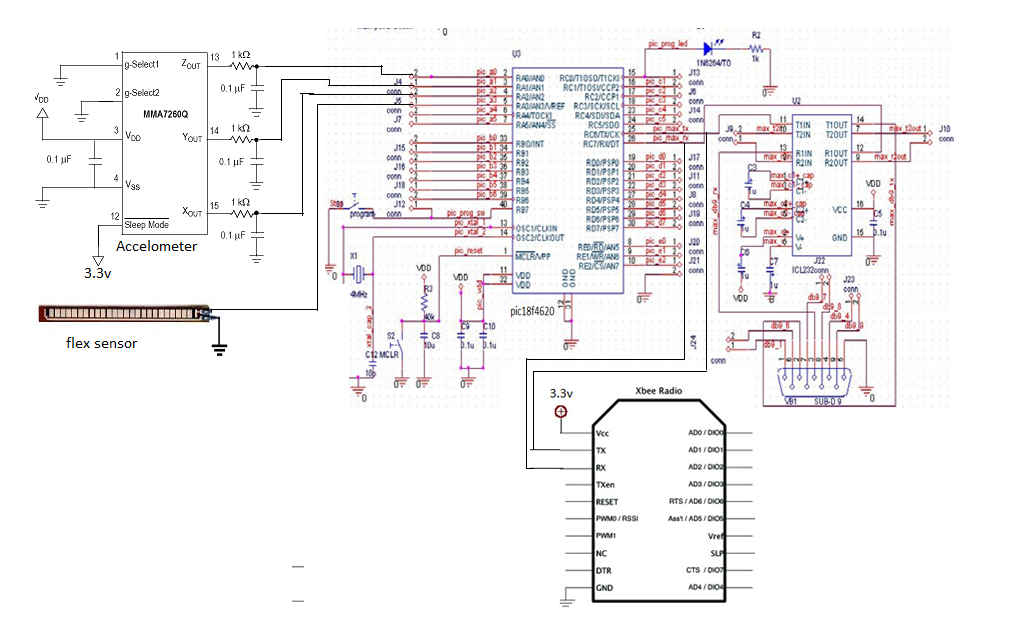
1. Accelerometer:

http://www.freescale.com/files/sensors/doc/data\_sheet/MMA7260QT/

1. Flex Sensor

<http://www.sparkfun.com/datasheets/Sensors/Flex/flex22.pdf>

**8. APPENDICES**

** User Controller Side:**

**Data Sheets:**