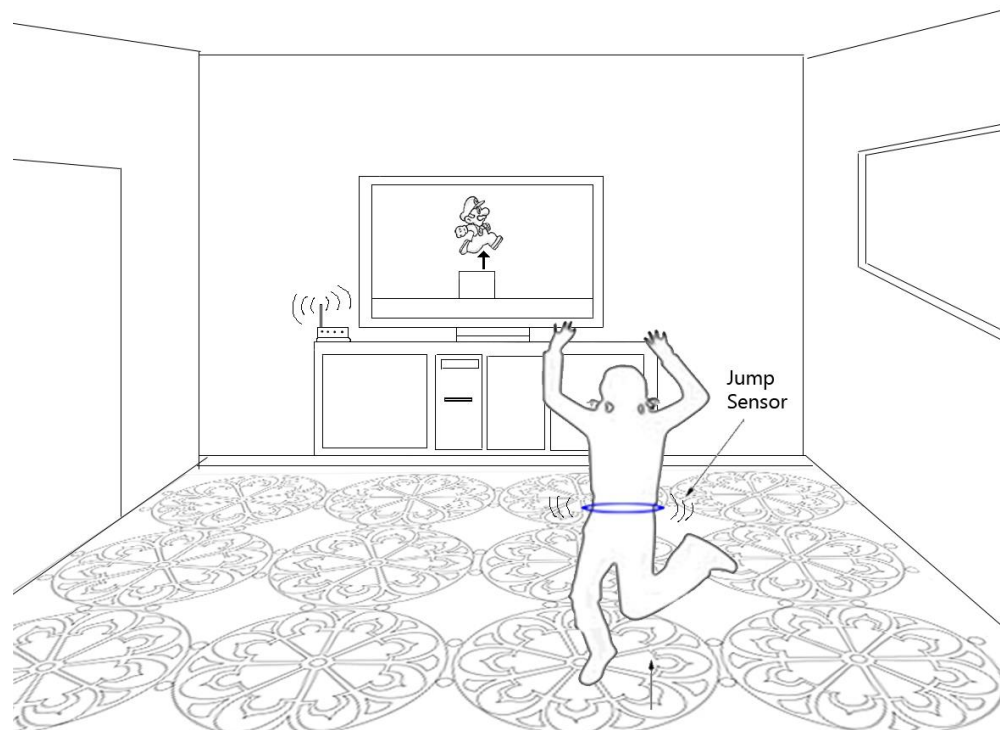


## 1. Conceptual Diagram

Jump controller is a belt-like controller which can be worn by users at their waists so as to control the game by jumping.



**Figure 1.1- Project Conceptual Diagram**

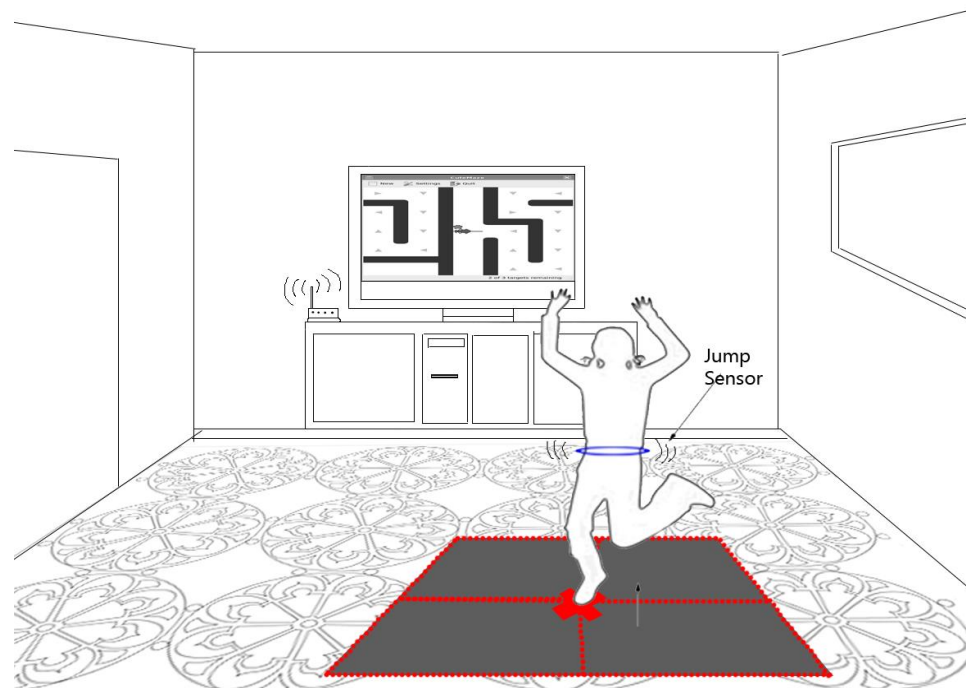
The core of this project is accelerometer's ability in sensing an object's acceleration in three axes: X, Y, Z and Arduino's capability in connecting sensors, transmitters and other circuit elements.

## 2. Scope and Bounds

This project will use the technology of the open source electronics prototyping platform, Arduino, to create a wearable motion-sensitive game controller. The Arduino will be set up to sense environmental movements using an accelerometer, and send wireless data to the receiver on the computer side to update the game (using the XBEE and ProtoShield).

However there are several possible movements in a game which can be mapped into a game controller action buttons, etc. For this project we have chosen to focus mainly on the “Jump” action (a.k.a Vertical Movement) and we will cover neither other kinds of vertical, diagonal or a combination of these movements nor we will do any gesture recognition, face recognition, etc.

A possible extension for this project has been shown in the picture below which will add the four directional movements to the Game controller capabilities. To read more about this refer to section 3.d: Function Descriptions.



**Figure 2.1 – Project possible extension Conceptual Diagram**

### 3. Design Description

#### a. Software Architecture:

This project has two main parts. The jump algorithm which will run on the Arduino and we will develop during the semester after doing research on existing algorithms. The second part is the game itself which runs on the PC and we will develop it concurrently with the Jump algorithm during the project implementation phase.

The software architecture of this project has been shown in the two diagrams below:

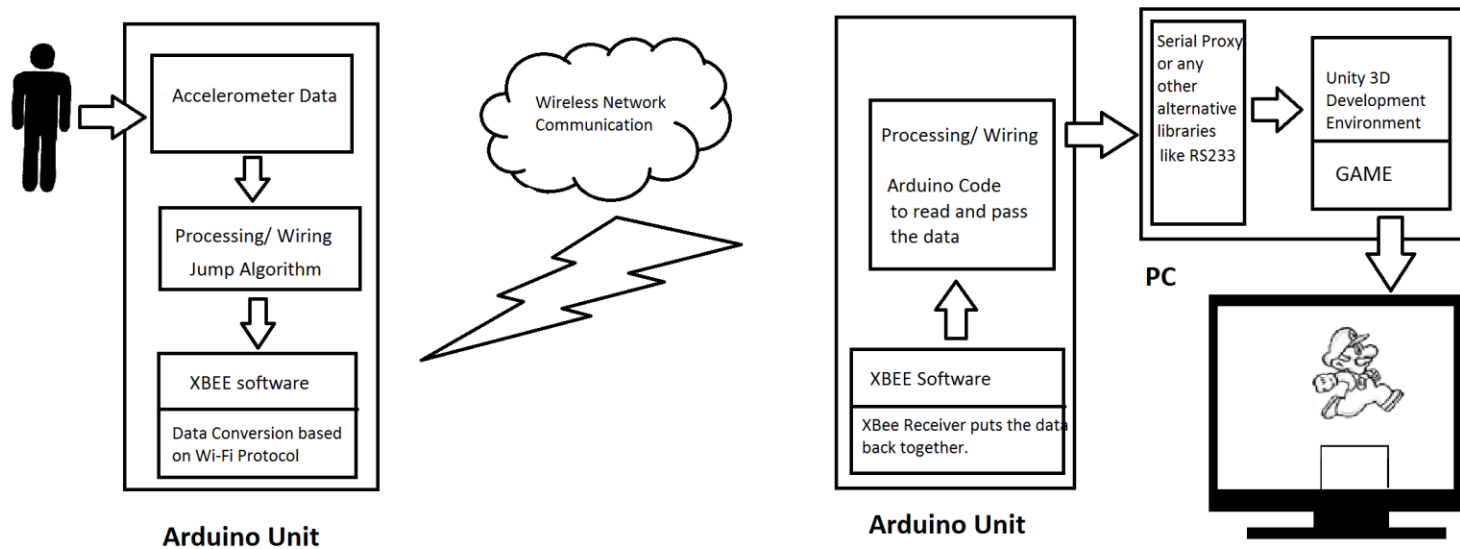


Figure 3.1 - Software Architecture Diagram

# System Block Diagram with Layers

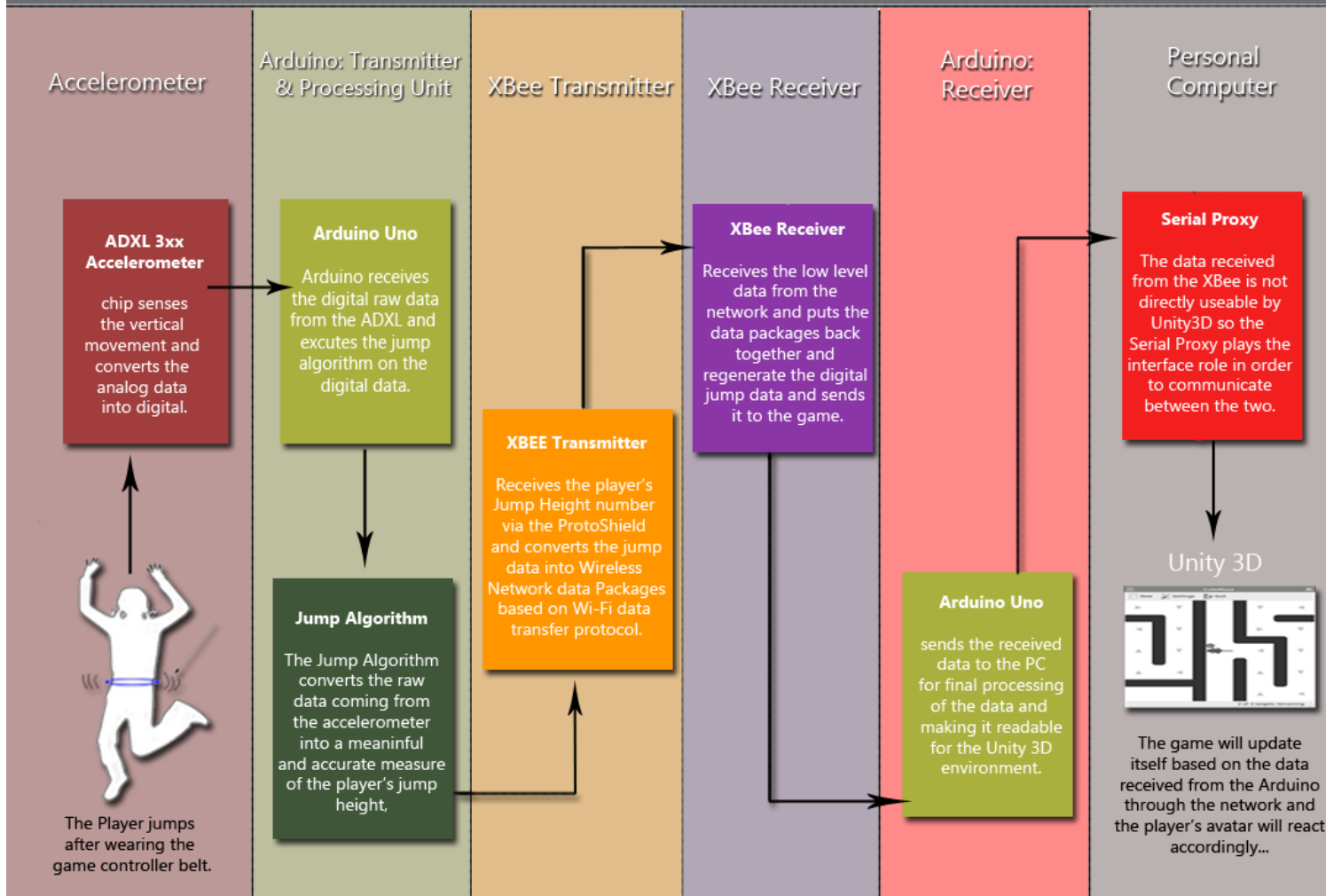


Figure 3.2 – System Block Diagram with layers explains the flow of the algorithms running through the project.

## **b. Hardware Architecture**

The Jump controller hardware architecture consists of:

- Accelerometer or motion/ distance sensors
- 1 Arduino Uno's on each sides (Transmitter & Receiver)
- 1 XBee or SD Shields on each sides (Transmitter & Receiver)
- The Main computer CPU, graphics card and monitor for display

The process of updating the game according to the player's jump height is described below:

1. The jump sensor core will be an ADXL3xx accelerometer. The ADXL3xx is a triple axis accelerometer with extremely low noise and power consumption (only 320uA) and the sensor has a full sensing range of +/-3g. The accelerometer measures in three axes: X, Y, and Z. The x and y-axis movement will sense tilt movement across the accelerometer and the z-axis will sense vertical movement, like a jump.
2. The Arduino will detect these movements, especially the z-axis changes, and once a pre-determined value is passed it will run the jump algorithm (raw data conversion algorithm) on the data and will send the result as a signal through the XBEE wireless shield (transmitter) to the computer.
3. Jump controller will communicate wirelessly using an XBee transmitter and a receiver. The module can communicate up to 100 feet indoors or 300 feet outdoors (with line-of-sight). The signal will be received through the computer connected XBEE to tell a game character to move accordingly.
4. For our project, the Arduino system will be connected to a wearable belt the participant can wear. The participant will jump which in turn will cause the character in the game to jump. This system will make a more interactive and physically demanding game.

You can see the Hardware architecture and their connections in the diagram below:

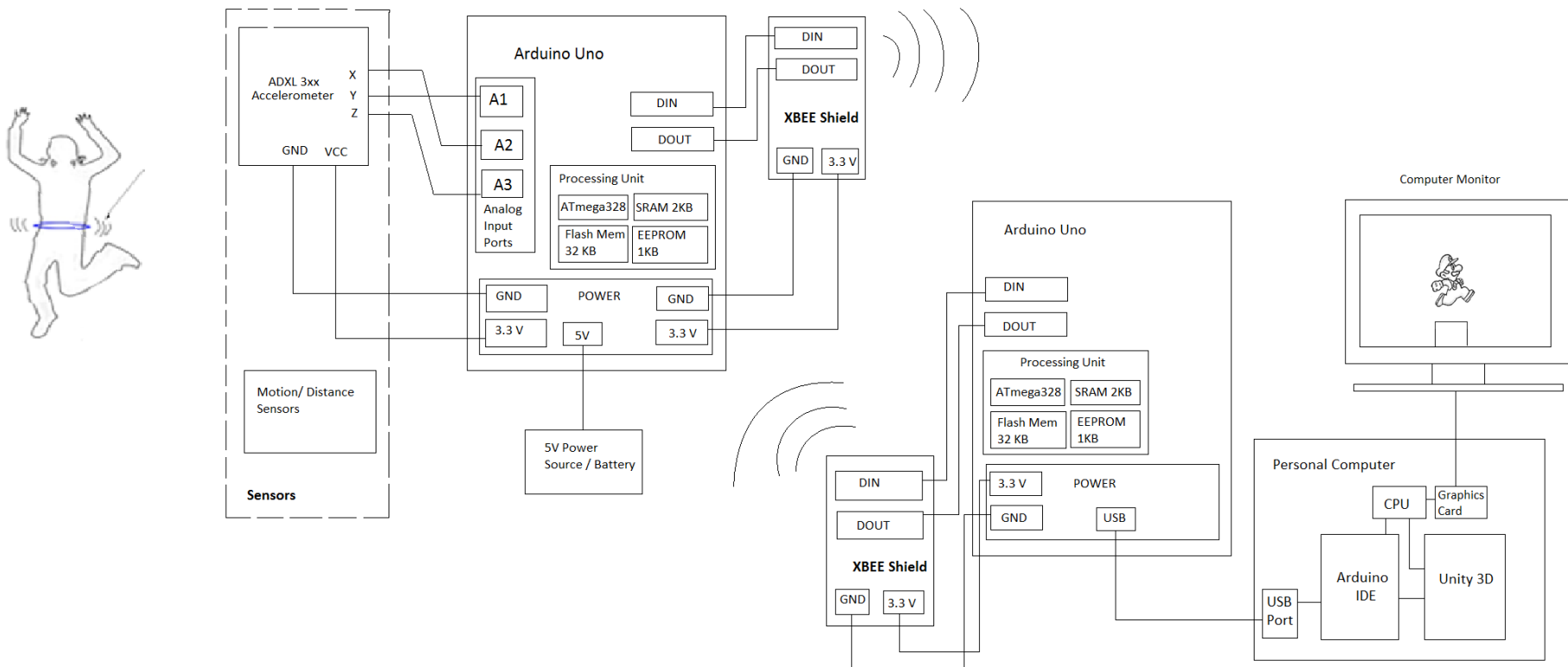



Figure 3.3 – Hardware Architecture Diagram

### c. Development Tools

#### Arduino IDE: Processing/ Wiring/ C & C++

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface.

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier.

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0". The main text area contains the following code:

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);          // wait for a second
}
```

The IDE has a teal header bar with icons for compile, upload, and help. At the bottom, a status bar shows "1" and "Arduino Uno on /dev/tty.usbmodemfd131".

Figure 3.4 – Arduino Integrated Development Environment

### External Libraries:

Since we are making our game outside of the Arduino IDE environment we need to establish a connection between the Arduino code and the game code in order to be able to update the game according to the player's movement. For that reason we will use a third party library (.dll file) based on the game engine we will use to make the game in order to communicate with the Arduino from our game.

### Unity 3D:

Unity is a cross-platform game engine and IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices. The game engine was developed in C/C++, and is able to support code written in C# or JavaScript. The game engine grew from an OS X supported game development tool in 2005 to the multi-platform game engine that it is today.

Scripting with Unity brings you fast iteration and execution and the strength and flexibility of a programming environment. In Unity, you write simple behavior scripts in JavaScript, C# or Boo. All three languages run on the Open Source .NET platform, Mono, with rapid compilation times.

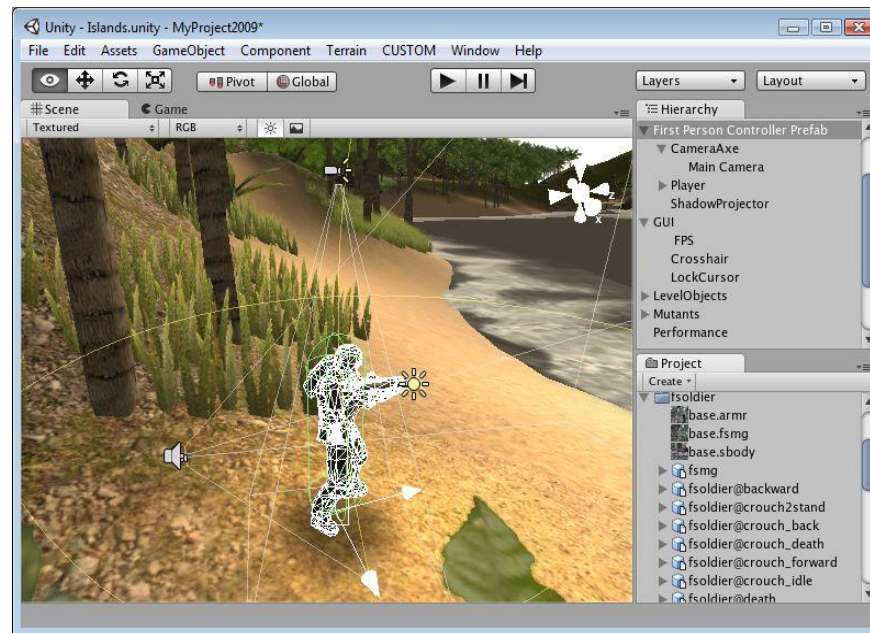


Figure 3.5 – Unity 3D Development Environment



**d. Function Descriptions**

Feature	Implementation (Spring 2013)	Description
Using accelerometer data	√	We will use accelerometer data instead of other sensors such as pressure sensor used in the Wii fit or similar devices.
Serial Communication (cable)	√	We will test the communication of the Arduino and accelerometer with the computer using a serial communication [using cables] first to make sure the hardware and code works correctly.
XBee/XBee Wireless Communication	√	We will use two XBee chips and two Arduinos one on each side (Transmitter & Receiver) to establish a wireless connection.
XBee/Modem Wireless Communication (Using a C# server)	--	Since the XBee chip works based on the Wi-Fi protocol any other Wi-Fi device such as your computer's modem should be able to communicate with it however it will need a server in between to receive the data from the network and pass the data to another program such as Unity or Flash.
Converting accelerometer data to jump height	√	The raw data coming from the accelerometer does not represent the jump height of the player and needs to be processed through a custom algorithm to be able to generate the proper data for the game. We will search for different algorithms of this kind during this project or we will come up with our own algorithm.
Connecting Arduino to Unity3D	√	For this project we have chosen to work with Unity3D which is a 3D game engine since we are using a 3 axis accelerometer and we have a vertical jump

		movement along the Z axis.
Connecting Arduino to Flash	--	We would like to try the wireless communication using a C# server if we had enough time and that would require a program which can communicate well with the C# server. Flash is an ideal choice for this purpose and also gives us the ability to make and test 2D games for this project.
Connecting Arduino to GameMaker	--	We would also like to make it easy for hobbyists to use our jump sensors in their own games and GameMaker provides an easy game engine to create games. There is an external library called RS232.dll which allows you to connect your Arduino to GameMaker and run your own game in GameMaker.
Four directional Movement with distance sensors (Left, Right, Up, Down)	TBD	The four directional movements are the base of every game and an important part of any game controller. We would like to implement this type of movement integrated in our design to be able to claim that we have a fully functional game controller. We have a preliminary design for this idea but we will not start working on it or implement it until we get the jump sensor done and test it completely.
Joystick movement (Analog data)	--	Joystick data is analog like the accelerometer data so it would be interesting to try to implement the joystick action as well as four directional movements.
Action sensors (Shooting, Kicking, etc)	--	Besides the Joystick and four directional movements we have couple other action buttons on a regular game controller which usually controls the players' weapons, special moves, etc. By implementing these actions we can have a fully featured motion-based game controller.

#### 4. Experimental Design

The experiment for jump controller will be mainly investigating the usability of the control however it's not limited to this purpose only. A within subject design will be used in the study. A certain number of participants (at least 10) will try the jump sensor after signing the consent form. During the study, the participants will firstly play the game via ordinary controllers such as keyboard and mouse and then wear the belt-like controller at their waists and play the game by jumping. Their scores will then be compared. Preliminary and post surveys will be given to the participants and the gaming process will be recorded.

##### Research Questions:

In this study we are mostly focusing on investigating and finding answers to the questions below:

- 1) How can an accelerometer raw data received from the real world be converted properly to a meaningful data in a digital environment?
- 2) How can a jump sensor be used as a game controller? How is this controller different from a regular game controller? How effective it would be?
- 3) How does changing the game controller affect the player's performance in the same game?
- 4) How can the jump controller be used in other areas such as sports, training, etc?
- 5) How prior gaming and sports experience does affects the player's performance in the game?

Our hypotheses is that an accelerometer can effectively be used as a vertical jump sensor, the players will enjoy using the jump controller more because of its uniqueness, they will do better in the game using the jump controller since it's more tangible in the context of the game and makes them feel like they are really in the game. Also the level of accuracy that we will achieve during the implementation of the jump sensor will determine how the jump sensor can be used in other areas such as sports and training. We believe that having prior gaming and sports experience is not essential to use the controller but they can have a positive effect.

**Independent variables: game controller**

The participants will experience two types of controllers in the study: mouse/keyboards and jump controller. Using mouse/keyboards, they will play the game by sitting in front of the computer and hitting the keyboard as they usually do; using the jump controller, they will wear a belt at their waist, stand some distances away from the computer and jump within an area to control the movements of the object in the game.

**Dependent variables: Jump controller's usability**

The controller's usability will be indicated by several factors including:

- 1) **Efficiency & Simplicity:** Network delay, how much time is required to set up the game controller and start playing, etc.
- 2) **Accuracy:** How accurately players' jumping will be mapped to the in-game object's movement and how many mistakes will occur;
- 3) **Emotional responses:** means how they will enjoy the process of gaming by jumping and what is their emotional arousal to the game control.
- 4) **Physical responses:** While wearing the jump controller are the participants able to react as quickly as when they use regular game controllers? How does the game controller affect their performance?

**Measurements**

A preliminary survey will be given to participants before they start the game to get their demographic information. All participants will need to complete a post survey to describe how they enjoy the different controllers, how easy/difficult each controller is, etc. Players' performance in the game and the videotapes will be investigated afterwards to provide information about the controllers' accuracy and efficiency. Their skin conductance will be measured before and after they play the game via different controllers to indicate their emotional arousal.

5. Schedule (Spring 2013 semester)

Item	Jan				Feb				Mar				Apr				May				Note
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
Project Proposal				√																	Due January 21st
Project & Research Design					√	√															Due February 10th
Circuit boards Assembly						√	√	√	√	√	√	√	√	√	√						5 weeks development cycle until the first pilot test week. The rest will be making changes as we go through the testing.
Game Design						√	√	√	√	√	√	√	√	√	√						Game Design process will happen concurrently with the boards' assembly process.
Game Development							√	√	√	√	√	√	√	√	√						As soon as the game design is finalized the development process will start. [flexible on time]
First Prototype										√											First prototype due March 13th
Pilot Testing										√	√										First round of pilot testing – We will bring in at least 10 participants to test our first prototype and give us feedback.

Revised Prototype											√	√							As soon as we collect the data we will start making changes and revising the project design based on the feedback received.
Formal Testing												√	√						Pilot Testing due: April 3 <sup>rd</sup> – All the data should have been collected by this date.
Final Presentation													√						Final presentation will happen at SVAD Annual Digital Media Showcase on April 22 <sup>nd</sup> at Center for Emerging Media.
Final Report														√					Final report will be written in a conference style paper and will be submitted by May 6 <sup>th</sup> .