Media Engineering and Technology Faculty
German University in Cairo

Controlling home devices with Microsoft Kinect

Bachelor Thesis

Author: Khaled Osman
Supervisors: Prof. Georg Jung

Submission Date: 04 June, 2013

Media Engineering and Technology Faculty
German University in Cairo

Controlling home devices with Microsoft Kinect

Bachelor Thesis

Author: Khaled Osman
Supervisors: Prof. Georg Jung

Submission Date: 04 June, 2013

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Khaled Osman
04 June, 2013

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Georg Jung for the continuous support of my bachelor study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study. My sincere thanks also goes to Dr. Fatma Meawad for teaching me the software engineering course where I was first introduced to Kinect developing, which later gave me the idea to implement this project. Last but not the least, I would like to thank my family for giving birth to me at the first place and supporting me spiritually throughout my life.

Abstract

Todays technology is getting smarter producing amazing new opportunities and making our lives easier. One of the most amazing technologies emerging nowadays are touchless interface devices as they offer an easy way to allow users to interact with applications with much less effort. These devices could be used to help handicapped people as they have difficulties using the home devices we use daily. We could make their lives easier by offering different ways to interact with their home devices through a touchless interface sensing device like the Microsoft Kinect sensor to provide a home automation system using a microcontroller. This project provides an interface to control electronic devices by using Microsoft Kinect sensor and Arduino microcontroller saving power, time and effort and removing obstacles.This is a thesis presented in the development of home automation technology by using Microsoft Kinect sensor and Arduino board offering depth and skeletal tracking, voice recognition, gesture recognition and a graphical interface.

Contents

Chapter 1

Introduction

Lately, touchless interface devices have appeared in the technology world making our lives easier. One of these newest and most selling technologies currently being used worldwide is the Microsoft Kinect sensor. Microsoft Kinect sensor is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PC. The Kinect's mechanical components make it superior to other motion sensing devices as it offers many capabilities such as face tracking, voice recognition, skeletal tracking and more. The idea is to extend the Kinect's potential uses from gaming to be used in different aspects and make more and better use of its capabilities. Since Kinect is a touchless interface device, it can be used as an input to interact with handicapped friendly applications and even electronic home devices aswell by using a microcontroller. This would probably make our lives easier and more elegant as it would save time, effort and even power consumption. Currently the average household pays over 100 dollars a month for electricity according to the U.S. Energy Information Administration. Half of those costs are to keep the motors and compressors operating in air conditioners and heating systems. If consumers had the ability to control their home's climate control systems remotely, they could save money without sacrificing convenience and comfort. In this project we aim to provide an easy way to control home devices using a Microsoft Kinect sensor and an Arduino board, (a simple microcontroller that can be used for taking inputs from a variety of switches and sensors, and controlling a variety of lights, motors, and other physical outputs), by saying voice commands, doing hand and body gestures and interacting with a touchless graphical user interface.

Chapter 2

Background

2.1 Kinect sensor

In this section we will introduce the kinect sensor and its Software Development Kit (SDK). The following information was collected from the book "Beginning Kinect Programming with the Microsoft Kinect SDK" [8] and Microsoft.com's online tutorials for Kinect for Windows retrieved from (http://www.microsoft.com/en-us/kinectforwindows/develop/tutori). [2] Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. It enables users to control and interact with the Xbox 360 without the need to touch a game controller through a natural user interface using gestures and spoken commands to provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

2.1.1 Kinect history

Kinect itself was first announced on June 1, 2009 at E3 under the code name "Project Natal". It was officially launched on November 4th 2010.On June 17, 2011, Microsoft finally released the Kinect for Windows SDK beta to the public under a non-commercial license after demonstrating it for several weeks at events like MIX. As promised, it included the skeleton recognition algorithms that make an initial pose unnecessary as well as the AEC technology and acoustic models required to make Kinect speech recognition system work in a large room. Every developer now had access to the same tools Microsoft used internally for developing Kinect applications for the computer. Since then Microsoft has been developing the SDK providing new features and easier usage of Kinect's capabilities. The latest Kinect for Windows SDK 1.7 was released on March 17, 2013.

2.1.2 Capabilities and constraints

The kinect sensor consists of an infrared projector, a Integrated Development Environment (IDE) light indicator, a color camera, an infrared camera, a microphone array and a tilt motor with a range of -27 to 27 degrees. See *fig.* 2.1

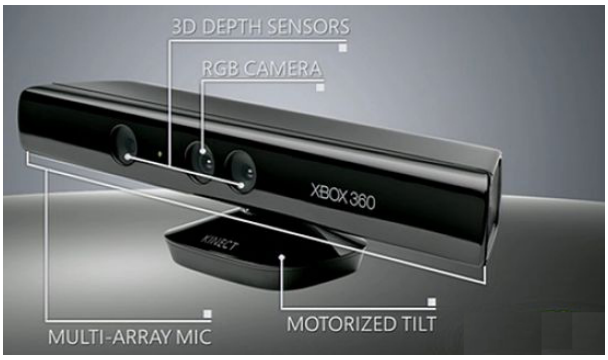

Figure 2.1: Kinect sensor

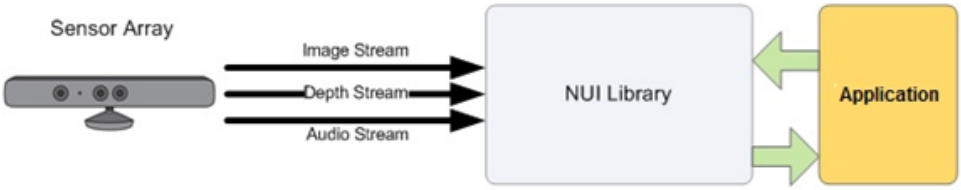

Figure 2.2: Data stream

The Kinect drivers support the use of multiple Kinect sensors on the same computer. The Natural User Interface (NUI) Application Programming Interface (API) allows you to determine how many Kinect sensors are connected to the computer, get the name of a particular sensor, initialize its streaming characteristics for color, depth, player index, and skeleton data streams (See *fig.* 2.2)

The depth data stream provides frames which represent the distance in millimeters from the camera plane to the nearest object in the depth sensor's Field Of View (FOV). The Skeleton API provides information about the location of up to six people who are standing in front of the Kinect sensor, with detailed information about position and orientation and full tracking of maximumly two people. The detailed data is provided to the application code as a set of points, called skeleton positions, that compose a skeleton, providing the users' current position and pose (See *fig.* 2.3) The area required to play Kinect is roughly 6 meters squared, although the sensor can maintain tracking through an extended range of approximately (2.320 feets (ft)). The sensor has an angular field of view of 57 degrees horizontally and 43 degrees vertically, while the motorized pivot is capable of tilting the sensor up to 27 degrees either up or down. The horizontal field of the Kinect sensor at the minimum viewing distance of 0.8 meters (m) is therefore 87 centimeters (cm), and the vertical field is 63 cm, resulting in a resolution of just over 1.3 millimeters (mm) per pixel.The microphone array features four microphone capsules and operates with each channel processing 16-bit audio at a sampling rate of 16 Kilo hertz (kHz).

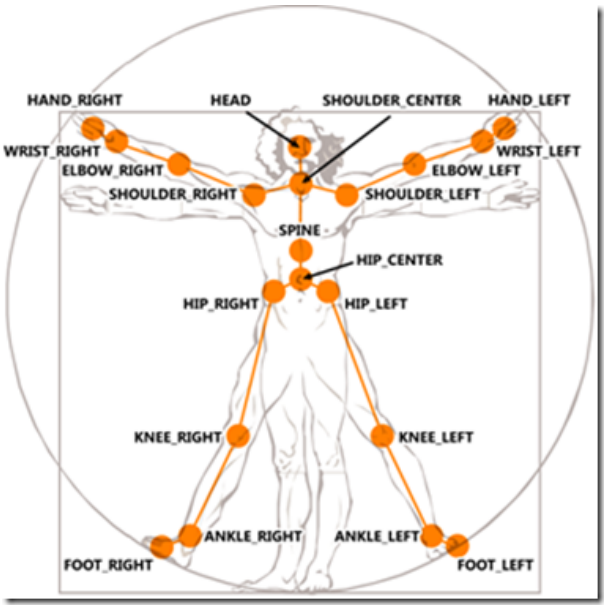

Figure 2.3: Skeleton Joints

2.1.3 What Kinect can not do

Despite Kinect's many capabilities, it has some defects. Kinect's voice recognition sometimes does not work as expected especially when you are surrounded by ambient noise from other guests and it is accent-dependent. Adding that The sensor can not detect gestures that does not require position changing of joints, diffrentiate between similar gestures or tell whether you are facing the kinect or not. In order to let the Kinect see you do something, you have to move in very pronounced motions because it seems only large movements can be seen by the stereoscopic cameras on the sensor. This works great for exercise games where you have to run in place or otherwise be full body active, but when all you are doing is making a pushing motion, the Kinect can not see this and does not do anything. Additionally, the Kinect does not like when there is a bunch of movement taking place all at once. it is just too much input and it may simply not detect the motion.

2.2 Microsoft Kinect for Windows SDK

The Kinect for Windows SDK is a set of libraries that allows us to program applications on a variety of Microsoft development platforms using the Kinect sensor as input. With it, we can program Windows Presentation Foundation (WPF) applications, WinForms applications, XNA applications and even browser-based applications running on the Windows operating system, however we cannot create Xbox games with the Kinect for Windows SDK.

2.2.1 Kinect for Windows SDK 1.7

The Kinect for windows 1.5 SDK offered 'Kinect Studio', a new application that allows developers to record, playback, and debug clips of users interacting with applications, support for new "seated" or "10-joint" skeletal system that will let apps track the head, neck, and arms of a Kinect user whether they are sitting down or standing and support for four new languages for speech recognition French, Spanish, Italian, and Japanese. Additionally it would add support for regional dialects of these languages along with English. The new Kinect for Windows SDK 1.7 adds two main features; interactions and kinect fusion. Interactions is a combination of new controls that makes it easier for users to interact through natural computing technology. These include "push" to select virtual objects, "grip" to pan and scroll, recognition of up to four hands simultaneously, and updated human interface guidelines. Fusion is a tool by which the Kinect sensor can accurately map and render three dimensional objects in real time. Kinect Fusion fuses together multiple snapshots from the Kinect for Windows sensor to create accurate, full, 3-D models. Developers can move a Kinect for Windows sensor around a person, object, or environment and paint a 3-D image of the person or thing in real time. These 3-D images can then be used to enhance countless real-world scenarios, including augmented reality, 3-D printing, interior and industrial design, and body scanning for things such as improved clothes shopping experiences and better-fitting orthotics.

2.2.2 Requirements

Unlike other Kinect libraries, the Kinect for Windows SDK, as its name suggests, only runs on Windows operating systems. Specifically, it runs on x86 and x64 versions of Windows 7. It has been shown to also work on early versions of Windows 8. Because Kinect was designed for Xbox hardware, however it requires roughly similar hardware on a PC to run effectively. Hardware Requirements:

- Computer with a dual-core, 2.66-GHz or faster processor
- Windows 7compatible graphics card that supports Microsoft DirectX 9.0c
- 2 GB of RAM (4 GB or RAM recommended)
- Kinect for Xbox 360 sensor
- Kinect USB power adapter Software Requirements:
- Microsoft Visual Studio 2010 Express or other Visual Studio 2010 edition:
- Microsoft .NET Framework 4
- The Kinect for Windows SDK (x86 or x64):
- For C++ SkeletalViewer samples:

- DirectX Software Development Kit, June 2010 or later version:
- DirectX End-User Runtime Web Installer:

To take full advantage of the audio capabilities of Kinect, you will also need additional Microsoft speech recognition software: the Speech Platform API, the Speech Platform SDK, and the Kinect for Windows Runtime Language Pack. Fortunately, the install for the SDK automatically installs these additional components for you.

2.2.3 OpenKinect history

Almost as soon as the Kinect was released, people began trying to figure out how Kinect worked. Then came the hacks, then the alternate applications. On November 6, a hacker known as AlexP was able to control Kinect's motors and read its accelerometer data. On Monday, November 8, AlexP posted video showing that he could pull both RGB and depth data streams from the Kinect sensor and display them. Hector Martin, a computer science major in Bilbao, Spain, had just purchased Kinect and managed to write the driver and application to display RGB and depth video. Martin became a contributor to the OpenKinect group and a new library, "libfreenect", became the basis of the community's hacking efforts. Throughout November, people started to post videos on the Internet showing what they could do with Kinect. Kinect-based artistic displays, augmented reality experiences, and robotics experiments started showing up on YouTube. Sites like KinectHacks.net sprang up to track all the things people were building with Kinect. By November 20, someone had posted a video of a light saber simulator using Kinectanother movie aspiration checked off. Microsoft, meanwhile, was not idle. The company watched with excitement as hundreds of Kinect hacks made their way to the web. On December 10, PrimeSense announced the release of its own open source drivers for Kinect along with libraries for working with the data. This provided improvements to the skeleton tracking algorithms over what was then possible with libfreenect and projects that required integration of RGB and depth data began migrating over to the OpenNI technology stack that PrimeSense had made available. Without the key Microsoft Research technologies, however, skeleton tracking with OpenNI still required the awkward T-pose to initialize skeleton recognition.

2.2.4 Microsoft SDK vs OpenNI

The Official SDK is developed by Microsoft which also develops the hardware and therefore should know internal information about the device that the open source society must reverse engineer. Also having economy behind it gives many advantages. On the other hand, The OpenKinect open source community consists of over 2000 members contributing their time and code to the Project. OpenKinect was released long before the official SDK as the kinect device was hacked on the first or second day of its release.

- Programming languages supported: Official SDK: C++, C#, or Visual Basic by using Microsoft Visual Studio 2010. OpenKinect: Python, C, C++, C# , Java, Lisp and more. Obviously not requiring Visual Studio.
- Operating systems support: Official SDK: only installs on Windows 7. OpenKinect: runs on Linux, OS X and Windows
- Documentation and support: Official SDK: well documented and provides a support forum. OpenKinect: appears to have a mailing list, twitter and irc. but no official forum. Documentation on website is not as rich as Microsoft SDK. The official SDK provides drivers that installed and came with examples and code for easy getting started unlike the OpenKinect.

2.3 Microsoft XNA

In this section we will explain the Microsoft XNA framework used to create the graphical user interface.

2.3.1 Definition and Overview

Microsoft XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates video game development and management. XNA is based on the .NET Framework for Xbox 360 development and .NET Framework on Windows. It includes an extensive set of class libraries, specific to game development, to promote maximum code reuse across target platforms.

2.3.2 Requirements

Since XNA games are written for the runtime, they can run on any platform that supports the XNA Framework with minimal or no modification. Games that run on the framework can technically be written in any .NET-compliant language, but only C# in XNA Game Studio Express IDE and all versions of Visual Studio 2008 and 2010 (as of XNA 4.0) are officially supported.

2.3.3 Libraries Documentation

Below is the documentation of the XNA framework as published by Microsoft.XNA Framework retrieved from http://msdn.microsoft.com/en-us/library/bb203940.aspx [7]

- Microsoft.Xna.Framework: Provides commonly needed game classes such as timers and game loops.

- Microsoft.Xna.Framework.Audio: Contains low-level application programming interface (API) methods that can load and manipulate XACT-created project and content files to play audio.
- Microsoft.Xna.Framework.Content: Contains the run-time components of the Content Pipeline.
- Microsoft.Xna.Framework.Design: Provides a unified way of converting types of values to other types.
- Microsoft.Xna.Framework.GamerServices: Contains classes that implement various services related to gamers. These services communicate directly with the gamer, the gamer's data, or otherwise reflect choices the gamer makes. Gamer services include input device and profile data APIs.
- Microsoft.Xna.Framework.Graphics: Contains low-level application programming interface (API) methods that take advantage of hardware acceleration capabilities to display 3D objects.
- Microsoft.Xna.Framework.Graphics.PackedVector: Represents data types with components that are not multiples of 8 bits.
- Microsoft.Xna.Framework.Input: Contains classes to receive input from keyboard, mouse, and Xbox 360 Controller devices.
- Microsoft.Xna.Framework.Input.Touch: Contains classes that enable access to touch-based input on devices that support it.
- Microsoft.Xna.Framework.Media: Contains classes to enumerate, play, and view songs, albums, playlists, and pictures.
- Microsoft.Xna.Framework.Net: Contains classes that implement support for Xbox LIVE, multiplayer, and networking for XNA Framework games.
- Microsoft.Xna.Framework.Storage: Contains classes that allow reading and writing of files.

2.3.4 Basics

Using Microsoft XNA your game class must override the 4 main methods stated below and execute them in order.

- Initialize(): Which allows the game to perform any initialization it needs to before starting to run. This is where it can query for any required services and load any non-graphic related content.

- LoadContent(): Which will be called once per game and is the place to load all of your content such as: textures, animations, fonts or any external files.
- UnLoadContent(): Which works as a destructor for your content.
- Update(GameTime): Allows the game to run logic such as updating the world,checking for collisions, gathering input, and playing audio. it works as a while(true) loop according to certain boolean values. gameTime Provides a snapshot of timing values.
- Draw(GameTime): This is called when the game should draw itself and it is the place to draw your textures and text.

2.4 Arduino

In this section we will explain the Arduino microcontroller and its IDE. The following information were collected from Arduino's official website (http://arduino.cc/en/Guide/Introduction) [1] and Michael Margolis's arduino cookbook. [9] Arduino is a simple microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible.It can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel Advanced RISC Machines (ARM). The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller. See *fig.* 2.4

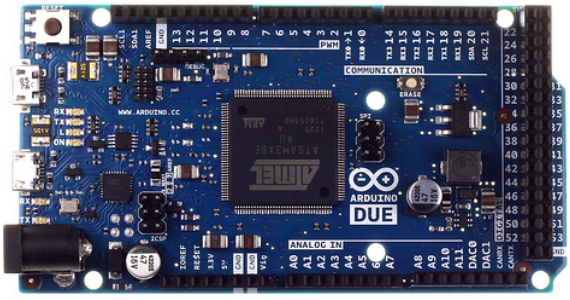

Figure 2.4: Arduino Board

2.4.1 Requirements

The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

2.4.2 IDE

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment. 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 projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. 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. A program or code written for Arduino is called a sketch. Sketches are saved with the file extension .ino. It has features for cutting, pasting and for replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. 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. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings
- loop(): a function called repeatedly until the board powers off

to execute your code you only need two buttons: Verify which Checks your code for errors. Upload which Compiles your code and uploads it to the Arduino I/O board.

2.4.3 Arduino Mega

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 15 can be used as Pulse Width Modulation (PWM) outputs), 16 analog inputs, 4 Universal Asynchronous Receiver-Transmitter (UART) (hardware serial ports), a 16 MHz crystal oscillator, a Universal Serial Bus (USB) connection, a power jack, an International Committee on Systematics of Prokaryotes (ICSP) header, and a reset button. It contains everything needed to support the microcontroller, simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila. The Mega2560 differs from all preceding boards in that it does not use the Future Technology Devices International (FTDI) USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 boards) programmed as a USB-to-serial converter. Revision 2

of the Mega2560 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into Device Firmware Upgrade (DFU) mode. Revision 3 of the board has the following new features:

- added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin
- the IOREF that allow the shields to adapt to the voltage provided from the board.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

Power: The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7Volts (V), however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The power pins are as follows:

- VIN. The input voltage to the Arduino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We do not advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 milli Amperes (mA).
- GND. Ground pins.
- IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory: The ATmega2560 has 256 Kilo Bytes (KB) of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output: Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- RX and TX: Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- External Interrupts: These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- IDE: 13. There is a built-in IDE connected to digital pin 13. When the pin is HIGH value, the IDE is on, when the pin is LOW, it is off.
- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication: The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines

will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX IDEs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

Chapter 3

Implementation

3.1 Project Design

In this section we will introduce our project design and the software architecture Due to lack of room to provide a real home automation system, we instead made a simulation of the project on a white board using Light Emitting Diode (LED)s. Each LED is connected to a pin from the arduino board from one side and its other wire connected to the ground pin through a resistor.However if we remove the LEDs and connect a lamp to the same spots using a suitable resistance and relay it will work. This chapter explains an overview of the project design and features.

3.1.1 User Stories

The user stories and the program features are as follows:

- As a system, I should be able to recognize certain gestures and control a device accordingly.
- As a system, I should be able to recognize certain voice commands.
- As a user, I can interact with a touchless interface by moving my hand controlling a cursor.
- As a user, I can see an avatar that shows if I am standing in the correct position .
- As a system, I should have a circuit connection between the arduino board and the device I want to control including the needed sensors, resistors, LEDs and relays.
- As a system, I should have an arduino application that can control a device and switch it on/off when needed.

- As a system, I should implement a way to establish the communication between the kinect and the arduino.
- As a user, I should see convinient feedback from the application.
- As a system, I should have a screen manager to control the flow of the graphical user interface.
- As a user, I should see an introductory fading screen to the application.
- As a system, I should have a music player.
- As a user, I should see an introduction screen that explains the application.

3.1.2 Software Architecture

The software architecture of the project as shown in the figure (see *fig.* 3.1) is divided into six main structures; the communication manager, the arduino application, the screen manager, the music player, the kinect class and the gesture controller which will be explained each individualling in the following sections.

3.2 Project Features

In this section we will explain the project components stated in the software architecture.

3.2.1 kinect class

The kinect class is where we define our kinect sensor, initialize it and enable the skeleton and color streams. We store the recognized skeletons in an array and on each new frame we update the array with the new skeletons.

3.2.2 Arduino program

The arduino application initializes the input and output pins then sets the baudrate to its default value 9600.In the loop it keeps checking for sent data and according to each value sent, lights a certain LED or turn a device on or off.

3.2.3 Communication between kinect and arduino

The communication between the kinect and the arduino is done through serial ports. Each of the kinect and the arduino are connected to the computer's USB ports which are used to send and recieve data as shown in the figure below (See *fig.* 3.2) Serial ports, also called communication (COM) ports, are bi-directional. Bi-directional communication allows each device to receive data as well as transmit it. Serial devices use different pins to receive and transmit data using the same pins would limit communication to half-duplex, meaning that information could only travel in one direction at a time. Using different pins allows for full-duplex communication, in which information can travel in both directions at once. The communication manager class manages the connection details such as the baudrate, portname, start and stop bits and have the methods for openning and closing a port and sending and recieving data through it. when an expected gesture is detected, the application opens the port, sends a certain character to the arduino application which performs an action accordingly, for example a "1" turns the internal LED attached to pin 13 on while a "0" turns it off.

3.2.4 Screen Manager

The screen manager is the core of the graphical interface, it handles how to add a screen on top of another, how to show, hide, load, draw, freeze and unfreeze a screen.It basically manages the flow of the screens and keeps store of them in an organised data structure. The class GameScreen is the super class which implements the methods initialize, load, draw and update and it is where you can add common features you want to add in all screenns such as the user avatar or the music player.

3.2.5 User Avatar

The avatar on top right of the screen represents the user's distance from the kinect device and tells if the user is standing in correct position. The avatar was implemented according to how accurate the Kinect SDK can keep track of the skeleton in this range and according to its **fov!** (**fov!**). The grey avatar means that the user is out of range.The red one means that the user is partially detected and is standing too close to the kinect sensor. The green one means the user is standing in the correct position and is being tracked successfully. The white avatar means that the user is standing too far away from the kinect and it is starting to lose track of him.

3.2.6 Music Player

This project also provides a music player where you can add and play your own music. To add your own music, simply copy and paste them to the folder named "Audio" in the project content folder. The project automatically loads the files in this folder, adds them to a list and plays a random song.

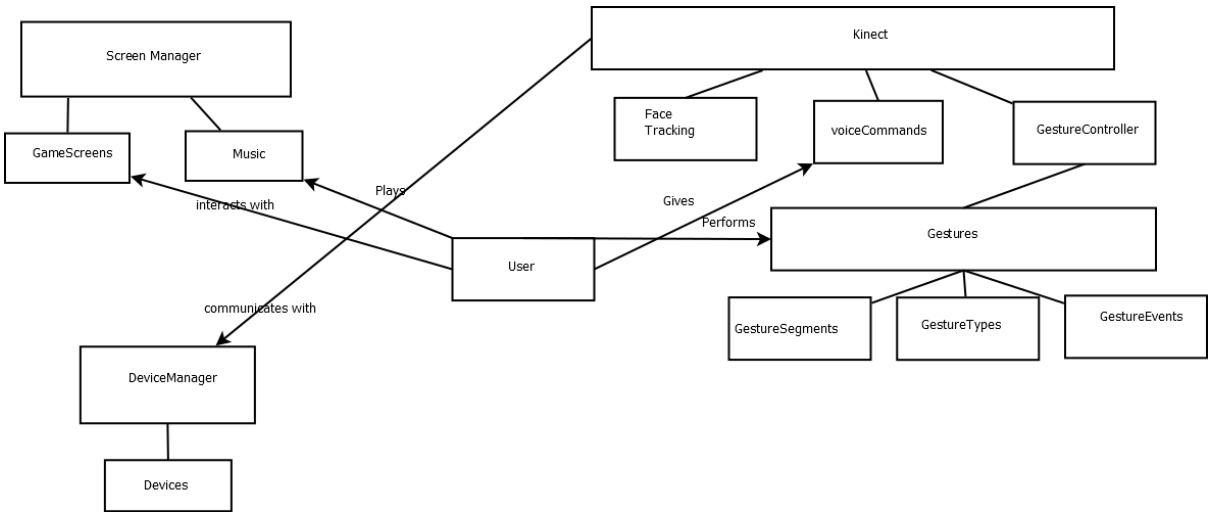

Figure 3.1: Software Architecture

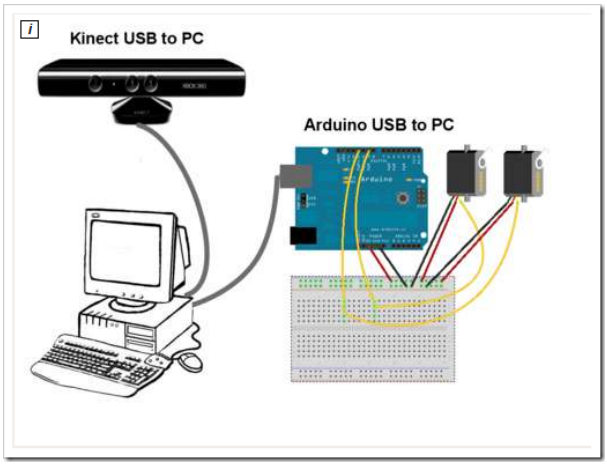

Figure 3.2: Kinect Arduino Connection

3.2.7 Adding a new Device

to add a new device, connect it to the arduino board instead of the LEDs, then add the correct values of resistors and relays needed.

3.3 Gesture Recognition

The Microsoft SDK provides the player's skeleton as an array of joints of a vector4 value each, which contains each joint's x,y,z and w values. The x,y,z position values in 3d space and the quality level (Range between 0-1) of the position that indicates the center of mass for that skeleton. This w value is only 1 for passive fully tracked players. As stated before kinect can only detect gestures that require change in joints positions so to build a gesture recognition system; you can either record your gestures as frames and set them as a standard to compare the newly performed gestures to, or compare body joints relatively to each others.However using the first approach, image proccessing would be expensive and may make the real time application slower depending on the application itself and its real time features. Furthermore it would be very difficult to set the standards and the error ratio that defines each gesture as the two frames will never be matching completely in terms of joints positions and the gesture performing speed. Adding that there are no gesture recording applications compatible with the Microsoft SDK yet. The gesture recognition system in this project works by comparing joints' positions relatively to each others and measuring the angle between the moving body joints required for each gesture. The information below about the gesture recognition system and its architecture was shared on Microsoft's msdn website. [4] Each gesture is divided into smaller gesture parts/segments if needed which are being checked for acceptance in order through each frame. When all gesture segments are detected in order within a suitable range of frames, the gesture itself is recognized.

3.3.1 How it works

A gesture controller contains all gestures defined and updates them with the newly updated skeleton data and constantly checks if any is detected through a certain amount of frames. A gesture is defined by an array of gesture segments and the gesture type.A segment is a part of a gesture such that a gesture can be one segment or more. Basically a gesture is a sequence of segments performed in certain order. Each segment gesture check can return three values; fail for when it is not detected, pause for when only part of the segment is done which means it is actually being performed thus wait for the next few frames and succeed for when the segment is detected. Each segment check is done by comparing the joints' x,y and z positions relatively to each others and by measuring the angle between the joints if needed.(See *fig.* 3.3 and *fig.* 3.4)

For example a wave gesture consists of 3 parts; the first segment is when the hand is on the left of the elbow, the second one is when the hand is above the elbow, the third one when the hand is on the right of the elbow.

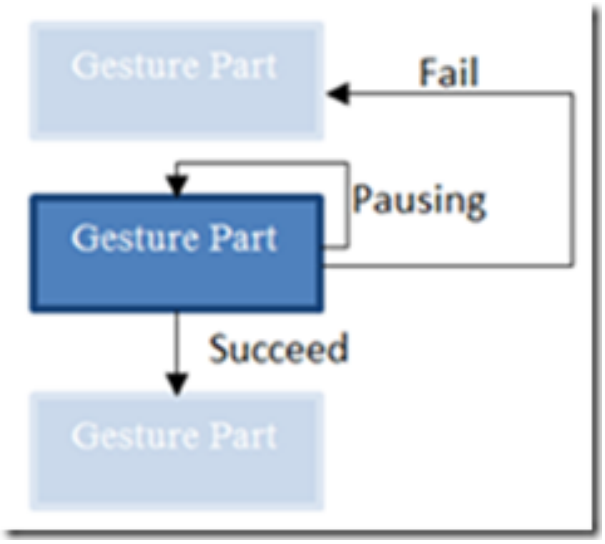

Figure 3.3: Gesture Architecture1

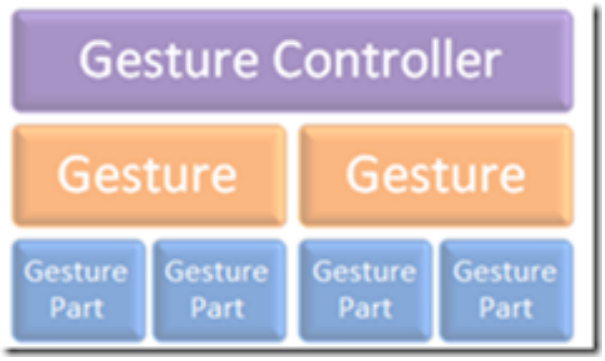

Figure 3.4: Gesture Architecture2

3.3.2 How to add a new gesture

To add a new gesture, create a new class for each gesture segment, implement the IRelativeGestureSegment interface, override the checkgesture method and write the correct contraints for each segment. Then add these segments to your gesture array and add it to the gesture controller.

3.3.3 Defects

However this system may not work as expected if the gestures are somehow similar or overlapping. If a gesture's sequence of segments is a prefix or a suffix of another then both gestures will be detected. Also as stated before the Kinect can only detect gestures that require change in joints positions, so for an example the system will not be able to differentiate between a fist bump gesture and a clap gesture. However it is really rare a developer may want to implement two similar gestures so in this project the gesture recognition system works fine.

Chapter 4

Conclusion

In this project we have implemented the core needed for any Kinect or XNA application; a graphical user interface, a gesture recognition, voice recognition and skeletal tracking systems. We also managed to process data from the Kinect and send it to the Arduino giving the application the ability to control any electronic device.With a few additions this project can be expanded to control robots, RC (remote control) toys, ACs, television, and more. Using this project for home automation would save power since the Kinect uses approximately less than 5W, the controller and other peripherals (such as USB hubs, input devices, lighting system interface, etc.) would use about 14W, and the lights can save nearly 300W in a room with three or four lamps. However why use a Kinect instead of an occupation or motion sensor? The Kinect detects presence, not motion, so the lights do not turn off when you hold still.It also generates a 3D picture of the room, so it can replace multiple motion or IR trip sensors.A single Kinect could be used for both gesture and zone-based occupancy control.

Chapter 5

Future Work

5.0.4 Project future

This project has showed that there is several new features that can be added. We want to be able to control new devices such as an electric garage, a television, electric curtains, door security, an AC and more. We want to provide more control of electronic devices rather than just switching them on and off. Since this is a home application we can add some security by implementing a screen where the user can enter a password by pressing alphabet buttons.The project will also have some home applications and games.Adding that the application should show more feedback to the user providing information like date, time, weather and the status of each device. Adding that it would be more convinient to make the communication between the application and the arduino wirelessly so the user does not need to have his laptop near the arduino board which acts as a home automation device. Furthermore with the new Kinect sensor and new Kinect for windows SDK coming out soon we can provide many new features that will make our appliction more effecient and improve our gesture and voice recognition systems.

5.0.5 Kinect and Kinect for Windows future

A new blog post on Microsoft's msdn announced that Microsoft debuted a new, far more sensitive version of its Kinect sensor. The new Kinect recognizes both the user and the controller and features a 1080P RGB camera, 30Frames Per Second (FPS) sensing and "time of flight" measurement that takes into account how long it takes for light to get from the Kinect to the user and back. The new sensor can also measure things like heartbeat and recognize individual joints in more detailed fashion, leading to more accuracy and polish. The new camera also has an increased field of view, which should mean a less fidgety experience when standing closer, further away or close to the edges of the room. Both the new Kinect sensor and the new Kinect for Windows sensor are being built on a shared set of technologies. Just as the new Kinect sensor will bring opportunities for revolutionizing gaming and entertainment, the new Kinect for Windows sensor will

revolutionize computing experiences. The precision and intuitive responsiveness that the new platform provides will accelerate the development of voice and gesture experiences on computers. Some of the key capabilities of the new Kinect sensor include:

- Higher fidelity The new sensor includes a high-definition (HD) color camera as well as a new noise-isolating multi-microphone array that filters ambient sounds to recognize natural speaking voices even in crowded rooms. Also included is Microsoft's proprietary Time-of-Flight technology, which measures the time it takes individual photons to rebound off an object or person to create unprecedented accuracy and precision. All of this means that the new sensor recognizes precise motions and details, such as slight wrist rotation, body position, and even the wrinkles in your clothes. The Kinect for Windows community will benefit from the sensor's enhanced fidelity, which will allow developers to create highly accurate solutions that see a person's form better than ever, track objects and environments with greater detail, and understand voice commands in noisier settings than before. The enhanced fidelity and depth perception of the new Kinect sensor will allow developers to create apps that see a person's form better, track objects with greater detail, and understand voice commands in noisier settings.

- Expanded field of view The expanded field of view accommodates a multitude of differently sized rooms, minimizing the need to modify existing room configurations and opening up new solution-development opportunities. The combination of the new sensor's higher fidelity plus expanded field of view will give businesses the tools they need to create truly untethered, natural computing experiences such as clicker-free presentation scenarios, more dynamic simulation and training solutions, up-close interactions, more fluid gesture recognition for quick interactions on the go, and much more.

- Improved skeletal tracking The new sensor tracks more points on the human body than previously, including the tip of the hand and thumb, and tracks six skeletons at once. This not only yields more accurate skeletal tracking, it opens up a range of new scenarios, including improved avateering, the ability to develop enhanced rehabilitation and physical fitness solutions, and the possibility to create new experiences in public spacessuch as retailwhere multiple users can participate simultaneously.

- New active infrared (IR) The all-new active-IR capabilities allow the new sensor to work in nearly any lighting condition and, in essence, give businesses access to a new fourth sensor: audio, depth, color and now active IR. This will offer developers better built-in recognition capabilities in different real-world settingsindependent of the lighting conditionsincluding the sensor's ability to recognize facial features, hand position, and more.

Bob Heddle, Director of Kinect for Windows said that the combination of the new sensor's higher fidelity plus expanded field of view will give businesses the tools they need to

create truly untethered, natural computing experiences such as clicker-free presentation scenarios, more dynamic simulation and training solutions, up-close interactions, more fluid gesture recognition for quick interactions on the go, and much more. He added that the next generation model will not arrive until 2014.

5.0.6 Microsoft XNA future

Microsoft has confirmed that it does not plan to release future versions of the XNA development toolset. A blog post from developer Promit Roy earlier detailed Microsoft's plans to fully retire the XNA Game Studio tools on April 1, 2014, while also suggesting that the future of API collection DirectX is uncertain. The company has now further explained the situation to Polygon, assuring developers that DirectX development will continue, but stating that XNA has received its last update. "XNA Game Studio remains a supported toolset for developing games for Xbox 360, Windows and Windows Phone" said the representative. Many developers have found financial success creating Xbox LIVE Indie Games using XNA. However, there are no plans for future versions of the XNA product.

Appendix

Appendix A

Lists

SDK	Software Development Kit
API	Application Programming Interface
IDE	Integrated Development Environment
LED	Light Emitting Diode
PWM	Pulse Width Modulation
FOV	Field Of View
ARM	Advanced RISC Machines
NUI	Natural User Interface
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
ICSP	International Committee on Systematics of Prokaryotes
WPF	Windows Presentation Foundation
DFU	Device Firmware Upgrade
FTDI	Future Technology Devices International
FPS	Frames Per Second
ft	feets
mm	millimeters
m	meters

cm	centimeters
kHz	Kilo hertz
V	Volts
mA	milli Amperes
KB	Kilo Bytes

List of Figures

List of Tables

Bibliography

[1] Arduino - introduction.

[2] Kinect tutorials — kinect for windows.

[3] New features — microsoft kinect for windows.

[4] Writing a gesture service with the kinect for windows sdk - mcs uk solution development team - site home - msdn blogs, 2010.

[5] It is official xna is dead, 2012.

[6] The new generation kinect for windows sensor is coming next year - kinect for windows product blog - site home - msdn blogs, 2013.

[7] Xna framework class library, 2013.

[8] James Ashley Jarrett Webb. *Beginning Kinect programming with the Microsoft Kinect SDK*. Apress, 2012.

[9] Michael Margolis. *Arduino cookbook*. O'Reilly Media, 2011.