STUDENTARBEID

HO2-300 BACHELOR PROJECT 2013

Pål Rambjørg (5), Tormod Eikelid (8),
John André Grimseth (14) & Fredrik Myrvang (18)

Report delivered: Thursday, 23. May 2013

HØGSKULEN i SOGN OG FJORDANE

HØGSKULEN i SOGN OG FJORDANE

STUDENT REPORT

Boks 523, 6803 FØRDE. Tlf: 57722500, Faks: 57722501 www.HiSF.no

TITLE	REPORTNR.	DATE
GestuRemote	1.0	23.05.13
PROJECT TITLE	**ACCESS**	**PAGES**
HO2-300 Bachelor Report	Open	63
AUTHORS	**SUPERVISOR**	
Pål Rambjørg (Project leader) [5] John André Grimseth [14] Tormod Eikelid [8] Fredrik Myrvang [18]	Joar Sande	
EMPLOYER		
Høgskulen i Sogn og Fjordane		

SUMMARY

Gesture Development Team (GDT) consists of the members Pål Rambjørg (Project leader), Tormod Eikelid, John Andre Grimseth and Fredrik Myrvang.

By using Java (Recognition), Arduino (Universal IR Remote) and Raspberry Pi (Media Center++), a simple wave becomes an action.
GestuRemote is meant to replace the traditional "OLD" TV-remote.
Plus adding some extra features.

SUBJECT

Hand Gestures, Home Theater PC, Recognition software, OpenCV/JavaCV, Kinect, Webcam, Raspberry Pi, Arduino, XBMC, Universal IR remote, RF remote.

Preface

Our last semester at Høgskulen i Sogn og Fjordane, department for Engineering and Science, the students was to conduct a bachelor project as close to reality as possible. We are encouraged to define a project ourselves or get an external client, where the latter is preferable. We decided to look into gesture based software on our own initiative.

This is a technical project where we look at the technical and theoretical solutions. We chose this project to get an insight of tomorrow's technology and hopefully be a part of it.

Førde, Thursday 23rd May, 2013

Pål Rambjørg	Tormod Eikelid
John A. Grimseth	Fredrik Myrvang

Abstract

GestuRemote Development Team (GDT) consists of Pål Rambjørg, Tormod Eikelid, John Andre Grimseth and Fredrik Myrvang.

By using Java (Recognition), Arduino (Universal IR Remote) and Raspberry Pi (Media Center++), a simple wave becomes an action. GestuRemote is meant to replace the traditional/outdated TV-remote, plus adding some extra features.

GestuRemote Project takes use of multiple platforms in its creation; Webcam (Java), Kinect (C++), Raspberry Pi (XBMC) and Arduino (Wiring) GDT have been working parallel with two solutions from the start; Webcam and Kinect. The Kinect part was put on ice approximately one month before project closure (3.2.4).

www.GestuRemote.tk

Sammendrag

GDT består av Pål Rambjørg, Tormod Eikelid, John Andre Grimseth og Fredrik Myrvang.

Ved å bruke Java (Gjenkjenning), Arduino (Universal IR kontroll) og Raspberry Pi (Media Senter++), blir eit enkelt vink til ein kommando. GestuRemote sitt mål er å erstatte den tradisjonelle/utdaterte fjernkontrollen, pluss litt ekstra funksjoner.

Prosjektet GestuRemote tar i bruk fleire plattformar i sin oppbygging; Webkamera (Java), Kinect (C++), Raspberry Pi (XBMC) og Arduino (Wiring) GDT har i frå starten jobba parallelt med to løysingar; Webkamera og Kinect. Kinect delen blei satt på is ca. ein månad før prosjekt fullføring.

www.GestuRemote.tk

Abbreviations

HiSF	Høgskulen i Sogn og Fjordane
GDT	GestuRemote Development Team
PC	Personal Computer
OS	Operating System
SDK	Software Development Kit
NUI	Natural User Interface
GUI	Graphical User Interface
HTPC	Home Theater PC
USB	Universal Serial Bus
JavaCV	Java Open Source Computer Vision Library
OpenCV	Open Source Computer Vision Library
OpenNI	Open Natural Interaction
NiTE	Natural Interaction Middleware
CMS	Content Management System
PHP	Hypertext Preprocessor
SSH	Secure Shell
TCP	Transmission Control Protocol
FFMPEG	Fast Forward Moving Picture Experts Group
BLOB	Binary Linked Object
RGB	Red Green Blue(Color Model)
PGR FlyCapture	Point Grey FlyCapture
HSV	Hue Saturation Value
AR	Augmented Reality
ARToolKitPlus	Augmented Reality Tracking Library
XBMC	Xbox Media Center
HDMI	High-Definition Multimedia Interface
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MaxMSP	Max Max Signal Processing
GPIO	General Purpose Input/Output
DC	Direct Current
CSI	Camera Serial Interface
DSI	Display Serial Interface
RCA	Radio Corporation of America

LED	Light-Emitting Diode
SD-Card	Secure Digital Card
ICSP	In Circuit Serial Programming
UART	Universal Asynchronous Receiver/Transmitter
ARM	Acorn Reduced Instruction Set Computing Machine (Microcontroller Single
AVR's	Microcontroller Single chip
SOC	System On a Chip
SRAM	Static Random-Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
PWM	Pulse-Width Modulation
IR	Infrared Light
RF	Radio Frequency
PiHat	Rasberry Pi Home Automation Transmitter
PiFM	Raspberry Pi Frequency modulation Transmitter
AMS	American Mathematical Society
JSON-RPC	JavaScript Object Notation Remote Procedure Call
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
CAD	Computer-Aided Design
TV	TeleVision
DVD	Digital Versatile Disc
CD	Compact Disc

LaTeX Quote from www.latex-project.org

GestuRemote Bachelor Project Report is written/compiled with LaTeX[1]

> LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.
>
> LaTeX is not a word processor! Instead, LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content.

LaTeX contains features for:

- Typesetting journal articles, technical reports, books, and slide presentations.
- Control over large documents containing sectioning, cross-references, tables and figures.
- Typesetting of complex mathematical formulas.
- Advanced typesetting of mathematics with AMS-LaTeX.
- Automatic generation of bibliographies and indexes.
- Multi-lingual typesetting.
- Inclusion of artwork, and process or spot color.
- Using PostScript or Metafont fonts.

Contents

List of Figures

Chapter 1
Introduction GestuRemote

We present **GestuRemote**, a research into computer vision and communication.

1.1 Motivation

The original idea was to replace the old Television (TV) remote with a camera and hand gestures. But why stop there? If we got our gestures recognized, why not use that to control everything in the house? Like the stereo, navigation within game consoles, Digital Versatile Disc (DVD) players, and the coffee maker in the kitchen. So it's like an easy implementation of a smart house.

To walk in a dark room and illuminate it by the wave of a hand, dragging the curtains without touching them and turning on the media center in a manner that seems natural and self explanatory is the way we want to control our houses. This is not something we were aiming towards though, but we wanted to research the possibilities and see how close our prototype could get.

1.2 Literature

In the first stages of the project we were searching the internet looking for literature on similar projects just to get a point of view.

Our greatest sources of information:

- Questions concerning Java and C#, there was an amazing wealth of information available from:
  - **http://stackoverflow.com/**
  - **http://opencv.org/**
- Questions concerning Arduino, there were many project-sketches and other information available from:
  - **http://arduino.cc/en/**
- Questions concerning Raspberry Pi, there was much information available from:
  - **http://www.raspbmc.com/**
  - **http://xbian.org/**

1.3 Previous work

When performing literary searches on the web one is likely to find similar works performed by other people. We found a lot of projects and information concerning gesture recognition online.

Some of the projects we took a closer look at were:

- JavaCV and OpenCV
- NiTe 2, PrimeSense
- CandeScent
- TipTep Skeletonizer
- SigmaNIL Framework

We chose to put our main focus on **CandeScent NUI and JavaCV**

1.4 Outline of the bachelor report

The contents of each chapter in the bachelor report are as follows.

Chapter 1: Introduction

We present **GestuRemote**, a research into computer vision and communication.

Chapter 2: Challenges

Before we started our research, we had to set the challenges we had to overcome.

Chapter 3: Kinect

Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs.

Chapter 4: Java

An effort in detecting a hand with webcam.

Chapter 5: Communication

We needed our devices to communicate, to forward our wave gestures.

Chapter 6: Raspberry Pi

Raspberry Pi is a credit card-sized single-board computer.

Chapter 7: Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.

Chapter 8: Website

As project documentation and presentation evolves, a website is a good way to present results to the public.

Chapter 9: GestuRemote Administration

GestuRemote Administration consists of; Pål Rambjørg, Tormod Eikelid,
John A. Grimseth and Fredrik Myrvang. GestuRemote is contracted by HiSF.

Chapter 10: Discussion

Through this project we have had several unknown challenges we had to overcome.

Chapter 11: Results

The resulting software does recognize a gesture, under the right circumstances.

Chapter 12: Further Work

Our open source code will be published on our webpage so anyone who want to develop it, borrow a method or just get some inspiration are welcome to try it out.

Chapter 13: Conclusion

This project was meant to teach us more about "gesture based" software, and what it actually takes to control objects with gesture tracking/recognition.

Appendices

The bachelor report is concluded with 7 appendices:

- Logbook
- HiSF Contract
- Meeting Minutes
- GestuRemote Budget
- Quality and Risk Assessment Scheme
- Gantt-Chart
- GestuRemote IR Remote Sketch

Chapter 2
Challenges

Before we started our research, we had to set the challenges we had to overcome.

2.1 GestuRemote

Gesture

- First of all we need to recognize our hand, face or something we could make several different gestures with.
- We want to use a Webcam or Microsoft Kinect. Kinect already got different solutions that recognize gestures, but the programs are complex. It is also not as universal as the webcam, which you find everywhere.

Recognition

- For recognition we have several programs to choose from. The logical choice would be Java, since we in earlier semesters have had good experience with the language.
- In Java there are many steps to find the specific hand or face;
  - Take a picture with a webcam
  - Live stream using a webcam
  - Color filters to find skin color
  - Edge detection
  - Hand algorithm
  - Recognize gestures made by a hand
  - Create commands based on gestures

Communication

- The gestures we make or catch with the webcam needs to communicate with other devices, we think Raspberry Pi is suitable for our project considered it's potential power versus size and prize.

Report

- We are self-employed, and have never done a project of this magnitude or relevance. This makes planning our progress and use of time mainly guesswork, but with previous projects we have had some experience, and use this to produce our Gantt chart and preliminary report. For the rest of our progress we have to rely on our growing experience to make sure the project is completed within its timeframe.

Chapter 3
Kinect

Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows Personal Computer (PC).

3.1 Kinect for Xbox 360

Kinect for Xbox 360 (3.1) is revolutionizing gaming with motion detection technology. It changes how we play the game, how we watch TV, listen to music, and watches movies. With Kinect there are no physical controllers or remotes, there is just you!

Kinect Makes You The Remote

- Kinect registers how you move and mimics every movement onto screen.
- Kinect inspires people to get off the couch and take control of the game.

Figure 3.1: Kinect for Xbox 360

http://0.tqn.com/d/familyinternet/1/0/L/2/-/-/Kinect_Lifestyle1.jpg

3.1.1 How the Kinect works?

Kinect was originally launched as a gesture-based game controller for Microsoft's Xbox 360 console. But Kinect offers a lot more to its consumers than just a game controller for Xbox 360!

The Kinect Sensor **Quote from HowStuffWorks.com**

> The innovative technology behind Kinect is a combination of hardware and software contained within the Kinect sensor (3.2) accessory that can be added to any existing Xbox 360. The Kinect sensor is a flat black box that sits on a small platform, placed on a table or shelf near the television you're using with your Xbox 360.
>
> Newer Xbox 360s have a Kinect port from which the device can draw power, but the Kinect sensor comes with a power supply at no additional charge for users of older Xbox 360 models.
>
> For a video game to use the features of the hardware, it must also use the proprietary layer of Kinect software that enables body and voice recognition from the Kinect sensor[2].

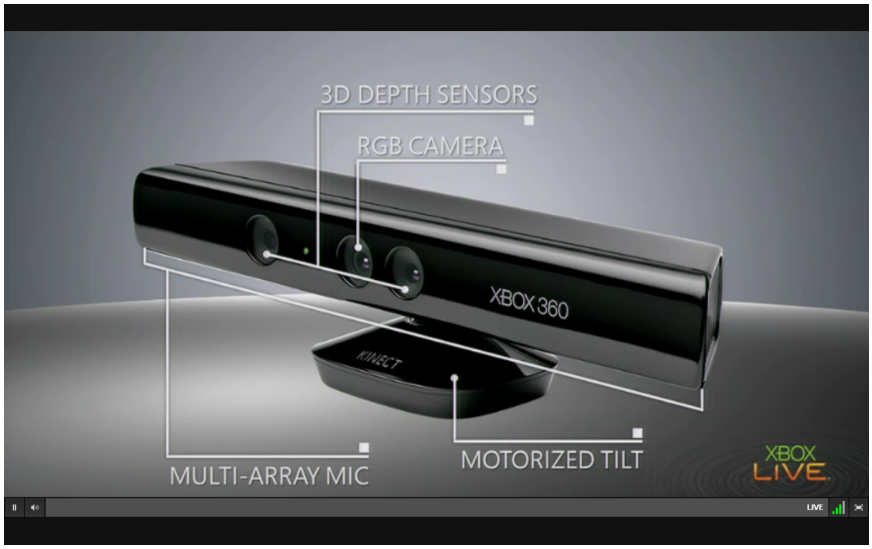

Figure 3.2: Kinect Sensor

http://mygggo.com/wp-content/uploads/2010/06/kinect12.png

There's a trio of hardware innovations working together within the Kinect sensor:

- **Color VGA video camera**; This video camera aids in facial recognition and other detection features by detecting three color components: red, green and blue. Microsoft calls this an "RGB camera" referring to the color components it detects.
- **Depth sensor**; An infrared projector and a monochrome CMOS (complimentary metal-oxide semiconductor) sensor work together to "see" the room in 3-D regardless of the lighting conditions.
- **Multi-array microphone**; This is an array of four microphones that can isolate the voices of the players from the noise in the room. This allows the player to be a few feet away from the microphone and still use voice controls.

A further look at the technical specifications for Kinect reveal that both the video and depth sensor cameras have a 640 x 480-Pixel resolution and run at 30 FPS (frames per second).

The specifications also suggest that you should allow about 6 feet (1.8 meters) of play space between you and the Kinect sensor, though this could vary depending on where you put the sensor [source: Microsoft Store].

The Kinect hardware (3.3), though, would be nothing without the breakthrough software that makes use of the data it gathers. Leap forward to the next page to read about the "brain" behind the camera lens.

Figure 3.3: Kinect Hardware

http://images.pcworld.com/news/graphics/209834-ifixitkinectteardown_original.jpg

3.2 How we use Kinect

Using the Kinect sensor (3.4) when playing Xbox games is a great experience but Kinect really shine when you connect it to a PC!

Kinecthacks.net[3] is a site where one can find many cool "hacks" which allow Kinect to be used by a computer rather than a Xbox. By using these hacks, we gain access to Kinect's cameras (color and depth), Light-Emitting Diode (LED), accelerometer and motor. With both the raw and the depth image obtained from the Kinect, we can easier build Machine Vision applications.

But the Kinect device can't work it's magic without body-tracking algorithms. Fortunately, PrimeSense, the company behind Kinect, released Open Natural Interaction (OpenNI) framework and Natural Interaction Middleware (NiTE). With these powerful tools available we now can access Kinect features such as real-time skeleton tracking, gesture recognition, wave detection and much more!

Figure 3.4: How we use Kinect

3.2.1 Implementations in Source Code

By taking use of Kinect and its powerful hardware we encountered a bigger problem; the source code. Libraries for Kinect are huge and are dependant upon middleware software to execute the different functions within the libraries.

Middleware Software's are everywhere and serve all kinds of purposes, we needed something more centered around our task and goals which were handtracking, handshape recognition and hand gestures.

To be able to design our software we wanted a library that was not only open source, but also clean and stable. Candescent[4] is such a software but it still presented us with instability and crashes.

3.2.2 Candescent NUI

Candescent Natural User Interface (NUI) is a hand and finger tracking software that uses Kinect depth data. Candescent is developed with OpenNI and Microsoft Kinect Software Development Kit (SDK), Language: C#

Its creator is Stefan Stegmueller and it is open source software which we could modify under license demands[5].

3.2.3 Sigmanil NUI Quote from Kinecthacks.com

> SigmaNIL[6] is, according to the developer, "the most powerful vision framework for natural user interfaces." The tool also comes with elaborate features such as finger level precision hand shape recognition, hand gesture recognition, and hand skeleton tracking.
>
> For those looking for more, SigmaNIL framework also provides customization tools for each of these advanced features. SigmaNIL is designed to support all depth sensor devices and base libraries such as OpenNI and KinectSDK. It is also designed so that developers can enhance it by adding new powerful modules.
>
> The tool is composed of the following parts; SigmaNIL Core (source code included), SigmaNIL Modules (HandSegmentation, HandSkeleton, HandShape and HandGesture) SigmaNIL Tools (Mainly training tools to customize the modules by creating relevant data files. Current distribution includes HandShapeTrainingTool only)

Sigmanil is a made by SigmaRD and is an "open source" software which we thought we could modify under license demands (3.2.4).

3.2.4 Sigmanil versus Candescent

Sigmanil NUI had a pre-stored gesture library in its source which includes:

- Circle Counter Clockwise
- Circle Clockwise
- Sigma
- Triangle
- Lightning
- Cross
- X
- Z
- N

Out of all the pre-made gestures we found that; Circle-CCW, Circle-CW, Sigma, Z and N were the most stable and easiest to reproduce on command.

SigmaNil was probably the most promising software we found, that was close enough to our vision. But sadly SigmaNil license states that their software isn't completely open source, which meant we couldn't implement our commands to it's source code!

SigmaNil HandGesture Software(3.5) showed us some good responses to following examples:

(a) Circle Clockwise (b) Circle Counter-CW (c) Sigma

Figure 3.5: Some Sigmanil Gestures

Candescent NUI also had hand tracking and gesture capabilities but the software were not optimalized for "Kinect for Xbox 360" rather "Kinect for Windows" (3.6).

(a) Kinect for Xbox 360 (b) Kinect for Windows

Figure 3.6: The Different Kinect versions

Conclusion: Based on experiences gathered while trying the different softwares:

- Sigmanil was the best candidate considering looks and stability.
- Sigmanil is an acknowledged product who has won several competitions related to depth sensor technology.
- Sigmanil was not as open source as we first thought and we had to put it on ice.
- Candescent was not optimized for our Kinect hardware.
- Candescent fortunately was sufficiently open source.

Despite Candescents slight issues with our Kinect it still worked on some levels and was still our most logical candidate, considering.

We decided to put the Kinect part of the project on ice in favor of our Java software.

Chapter 4
Java

An effort in detecting a hand with webcam.

4.1 Java

Java is a programming language we were all familiar with, so we wanted to research the hand detection possibilities within java using a webcam. After doing some research around Java image processing and shape recognition, some names kept repeating themselves. Open Source Computer Vision Library (OpenCV) with the Java Open Source Computer Vision Library (JavaCV) libraries were used in almost every example we found.

JavaCV provides wrappers to commonly used libraries in the field of computer vision like: OpenCV, Fast Forward Moving Picture Experts Group (FFmpeg), libdc1394[1], Point Grey FlyCapture (PGR FlyCapture), OpenKinect, videoInput, and Augmented Reality Tracking Library (ARToolKitPlus). We can call on these libraries using utility classes in java. This way java can reach down in to the computer to get a faster graphical calculation. Detection becomes easier using pre built image processing classes.

We were hoping to find some open source projects that at least had found the hand for us, so that we only had to implement the functions and bring them out to the "transmitting part" of the project. We had no such luck. The ones we found in a language we could understand were just bits and pieces of what we had in mind. So we started to think for ourselves. But how can we detect a hand in a picture. Skin color maybe?

[1] Library that provides a complete high level application programming interface

The colors in an image (4.1) can be made by mixing three specific colors, Red Green Blue(Color Model) (RGB). By measuring each pixel RGB values, we can find the color value of our skin.

Figure 4.1: Webcam image

When we have the color of our skin we can try to filter it out from the rest of the image by adjusting the image threshold. We made a new image (4.2), where if the RGB was "skin color" we put the pixel white and if not, black.

Figure 4.2: Webcam image after color filtering

We then encountered a problem with our filtering. Because of the quality of our, and most webcams, we did not only get skin color. The webcam has a light adjusting function which changes the RGB value of the whole Picture. Since it was winter and some of us had a skin color that was almost gray, we could not filter out the background in a good way.

To try to filter out illumination we studied normalization. In normalization we use the three values in RGB to remove light difference. Red, Green and Blue values stretch from 0, which is black, to 255 which is white. When we move between pixels within the same color, but with different illumination, the three colors increase and decrease in value with about the same amount.

$$NormalizedRed = \frac{Red}{\sqrt{Red^2 + Green^2 + Blue^2}}$$

$$NormalizedGreen = \frac{Green}{\sqrt{Red^2 + Green^2 + Blue^2}}$$

$$NormalizedBlue = \frac{Blue}{\sqrt{Red^2 + Green^2 + Blue^2}}$$

From this we get a number scaling from 0-1, so we need to multiply it with 255 so the scale matches the RGB scale. So we get the value [Red, Green, Blue] from each Pixel and replace it with [Normalized Red, Normalized Green, Normalized Blue].

When we have the normalized (4.3) picture we can use the same color filter as before to set the color ranges we want.

(a) Original Photo (b) Normalized Photo (c) Filtered Photo

Figure 4.3: Normalization Process

Trying to improve our hand detection we wanted to find an edge detection method. One of the better ones was Canny Edge detection (4.4) which is developed by John F. Canny. His intension with the Canny Edge was to improve/enhance the already existing edge detectors.

Canny Edge is based on these three criteria's; First criteria and also the most obvious one, is to get a low error rate. This means that every spot there is an edge occurrence will be marked, and that non edges won't be marked. Second criteria is that the edge point will be localized, that means that the drawn edge is a minimum from the actual edge. The third criterion is to have only one response to a single edge.

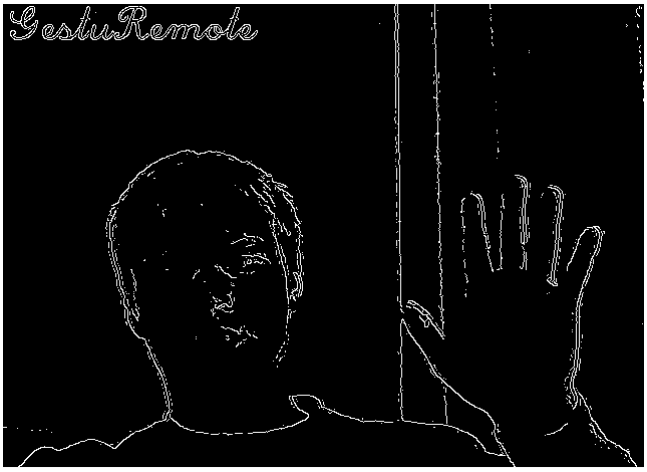

Figure 4.4: Canny edge detection

Based on these criteria's, the "steps" in the canny edge detection is as follows; First it smoothes the image to eliminate the noise. It then uses high spatial derivatives to highlight regions where it find the image gradient. Then an algorithm tracks along these regions and suppresses any pixel that is not at the maximum. Gradient array reduced by hysteresis, which is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds. If the magnitude is below the first threshold, it is then set to zero and then made as a non-edge. If the magnitude is above the high threshold, it becomes an edge. If it is between it depends on if there is a path from this pixel to another pixel with a gradient above the second threshold.

We designed a simple edge detection algorithm (4.5), using some if sentences. In short: if major difference in color, there's an edge. This actually worked very well but we got some noise in the edge image.

To remove some of the noise we used an algorithm where we summed all pixels in an 5x5 grid around each pixel. If there were more than "X" black pixels in the grid, the middle pixel stayed black, else it was painted white.

Figure 4.5: Result from our edge detection

The last skin color filtering method we tried was to convert the camera image to the Hue Saturation Value (HSV) plane. In the HSV plane (4.6) there are no red, green and blue as the three variables that makes every color. Instead there is Hue, Saturation and Value, hence the name. Hue says what color the image is, while saturation tells us how much color, where no saturation is white. And lastly value tells us the intensity of the color, where no intensity is black.

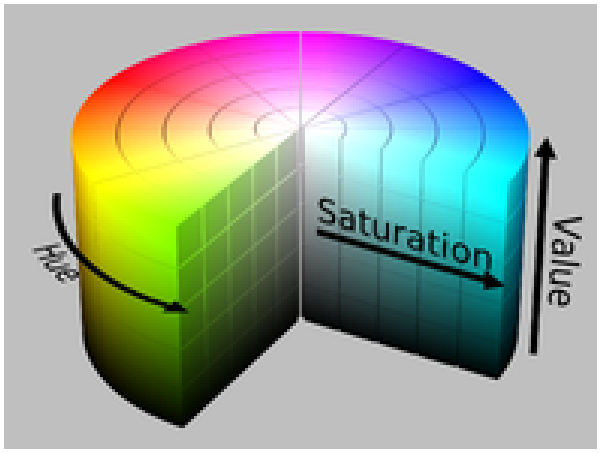

Figure 4.6: Cylindrical description of HSV

http://upload.wikimedia.org/wikipedia/commons/0/0d/HSV_color_solid_cylinder_alpha_lowgamma.png

We got the skin color values from another skin detection project at [7]. This worked very well with the right lighting on Tormods PC, but was not so functional on Fredriks PC. This is probably caused by difference in camera quality and light adjusting functions. So here our program split in two, where we had one based on normalization-filtering, and one based on HSV.

When we have our filtered, black and white representation of our webcam image (4.7), we have to make the computer "find" the white objects. Luckily JavaCV got a class called cvFindContours which finds all the binary linked objects (Binary Linked Object (BLOB)) and puts them in an hierarchy.

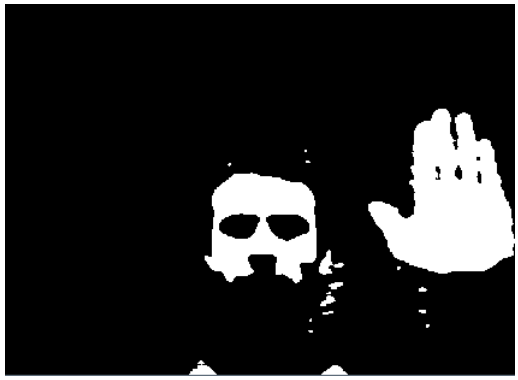

Figure 4.7: The result after HSV-filtering

By using the predefined hierarchy we can now try to remove the objects that is not hand. We set a max and min to the object area so the noise and large objects is ignored. We then try to separate a hand from a face by defining a hand as a object with less than two holes and a face as an object with two or more holes (4.8).

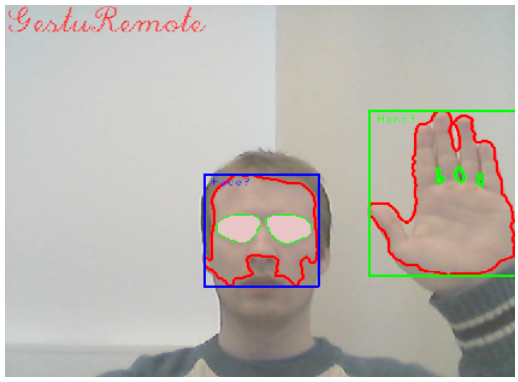

Figure 4.8: Hand and face, separated.

To get some understandable data to save as a gesture, we cropped the image around the object that is most likely to be a hand. We then summed the number of Pixels, that were white, for every x and y value and put them in to two arrays. To make it easier for us we made two diagram (4.9) representations of the arrays.

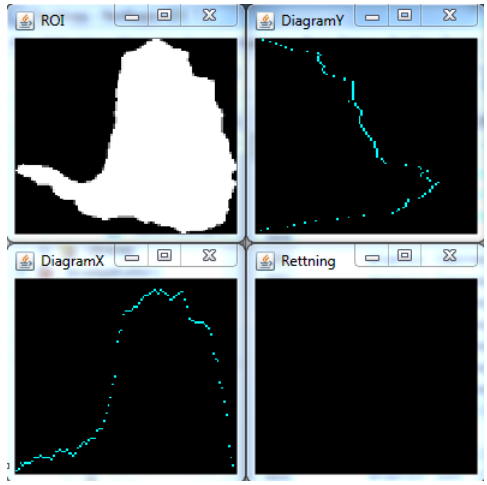

Figure 4.9: Object diagram

The diagrams show that when we hold up our hand with the thumb out, the first x values are low and ending up in a large spike in the end, while the y values are around middle values and ending in a low spike. These are values we can use to recognize gestures (4.10). To get a more reliable recognition we also have to get enough hits per second, where hits is when the object is within the right values.

To have the ability to navigate in menus in a good way we wanted some movement commands. From the cvFindContours we can get the width, height and the position of the upper left corner of the square surrounding the object. The outer edges of the object is flickering allot, so we added half the width and height to the position components so we had the middle point of the object. This point is much more stabile. By making arrays, where we put the x and y value of the middle point, we can see what direction the object is moving. If the values is increasing or decreasing enough over a one sec period we acknowledge this as a up, down or sideway command. With the width and height we also made an array with the area of the square. This area was used to see if the object was moving towards or away from the camera.

At this point we did not have time to continue improving the recognition part and had to concentrate on documenting what we had done. Here is what the recognition part looked like in the end (4.10).

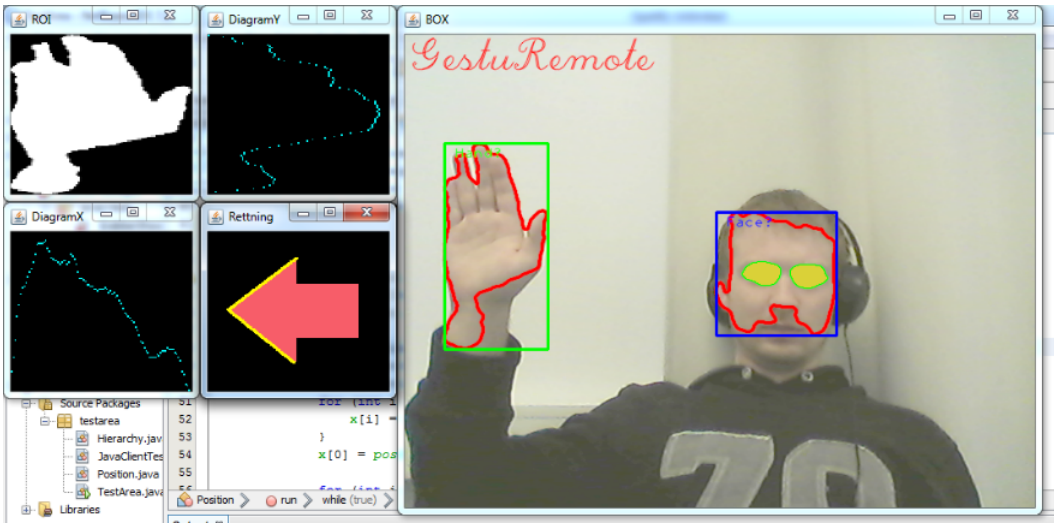

Figure 4.10: Our hand detection

4.2 OpenCV Quote from OpenCV.org

OpenCV[8] is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to:

- Detect and recognize faces
- Identify objects
- Classify human actions in videos
- Track camera movements
- Track moving objects
- Extract 3D models of objects
- Produce 3D point clouds from stereo cameras
- Stitch images together to produce Hi-Res image
- Find similar images from database
- Remove red yes from images
- Follow eye movements
- Recognize scenery and establish markers for AR
- Etc...

OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 5 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV.

OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, C, Python and Java interfaces and supports Windows, Linux, Android and Mac Operating System (OS). OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available.

A full-featured CUDA interface is being actively developed right now.
There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms.

OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

4.3 Our hierarchy

That any command can come at any time is not a good way to control more than one device. So we made a hierarchy (4.11) of confirmation commands before the computer sends the execute command out to the signal-mimicking devices (Arduino and Raspberry-Pi)

Figure 4.11: Description of the Hierarchy class

As you can see the hierarchy waits for the "Hi" command, which is a left hand palm. Before this command, no other gesture will do anything. After getting the "Hi" command it's possible to choose which device to enter. We have illustrated three options, but the real program has one more (three devices). The options give the opportunity to choose a device or go back to the waiting stage.

We have also inserted a timer which is renewed every time the program gets a gesture usable in that part of the hierarchy. If this timer runs out, the program goes back to the waiting stage. If a device is chosen we can execute a "send" command, go back, or directly change device. We choose the execute command with the vector algorithms. Then, depending on what device you are in and what vector command, the program choose a communication method and sends the given command.

Chapter 5
Communication

We needed our devices to communicate, to forward our wave gestures.

5.1 What we needed

We thought of different possible solutions to communicate between our recognition software in Java and our Raspberry Pi. We needed a simple but effective communication (5.1), that could send specific commands our Raspberry Pi would be able to execute and perform our given tasks.

Some of our thoughts where Bluetooth and Infrared communication, since we knew that Xbox Media Center (XBMC) on the Raspberry Pi supported this. After some thought we figured that this would only complicate the matter, and new issues like Line of sight would be needed. Eventually we checked raw Transmission Control Protocol (TCP) communication, and found that XBMC on Raspberry Pi had compatibility with TCP commands over JavaScript Object Notation Remote Procedure Call (JSON-RPC). This gave us a simpler solution, since we then just had to create a means of sending JSON parsed strings to Raspberry Pi and it would know what we wanted it to do.

Figure 5.1: Communication illustrated

http://pdsgn.files.wordpress.com/2009/11/mpj0439347000011.jpg

JSON-RPC **Quote from Wiki.Xbmc.org**

> JSON-RPC[9] is a stateless, light-weight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.
>
> It is designed to be simple!

By using this already implemented software in XBMC we would also not have to redo our "receiving methods" when XBMC eventually is updated, only if they change JSON-RPC for something else.

5.2 How we solved it

By using JSON-RPC socket based communication, we could write a simple java method that used basic socket programming to connect to Raspberry Pi and send preset commands already implemented in the library, and achieve control. We would then link these commands to our recognition software already written in Java, and then be able to send commands from the same software, with no additional tweaking.

This solved how to control XBMC, but it did not work controlling our light sockets. To do this, we had a program inside the XBian distro on our Raspberry Pi. This software needed some inputs to be able to work as we intended it to, so we made a new Java program. This time, we made an Secure Shell (SSH) method. SSH is a secure method to send text commands to systems, mostly used by Linux and other secure devices. It requires a user and a password to be able to connect to the server.

SSH **Quote from Techterms.com**

> SSH[10] is a method of securely communicating with another computer. The "secure" part of the name means that all data sent via an SSH connection is encrypted. This means if a third party tries to intercept the information being transferred, it would appear scrambled and unreadable. The "shell" part of the name means SSH is based on a Unix shell, which is a program that interprets commands entered by a user.

By using SSHTools[11] we wrote a program in Java that opened a new SSH session at the start of our program, and when it had sent our command string, closed the session, but did not terminate the connection, so it did not have to go through the login process again.

By sending strings to Rasberry Pi Home Automation Transmitter (PiHat)[12] inside XBian we would then tell the software that we now wanted the light to toggle with a gesture, then send the command to Raspberry Pi, and then send it over radio frequency to our power sockets and toggle the lights.

Now the only thing left was to communicate with our Arduino that replicated remote Infrared Light (IR) signals. Our program had a webserver running on it with a web interface for us to toggle sending commands. What we did, was to use the Hypertext Transfer Protocol (HTTP) command "GET" to get a connection with the server, be accepted by the server, and then sending a string to it, so that our program could find this string to know what to send.

Chapter 6
Raspberry Pi

Raspberry Pi is a credit card-sized single-board computer.

6.1 How Raspberry Pi works Quote from HowStuffWorks.com

The Raspberry Pi[13] is developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

The Raspberry Pi device looks like a motherboard, with the mounted chips and ports exposed (something you'd expect to see only if you opened up your computer and looked at its internal boards), but it has all the components you need to connect input, output, and storage devices and start computing.

You'll encounter two models of the device: Model A and Model B. The only real differences are the addition of Ethernet and an extra Universal Serial Bus (USB) port on the more expensive Model B (6.1).

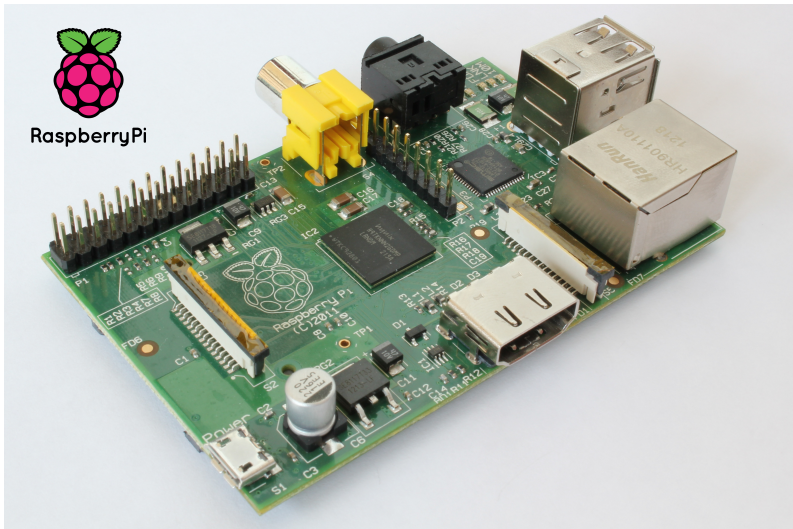

Figure 6.1: Raspberry Pi Model B

http://www.hoektronics.com/wp-content/uploads/2013/04/RaspberryPi.jpg

6.1.1 Raspberry Pi Model B Card

We chose the Raspberry Pi Model B (6.2) card because of it's very potent power to size ratio.

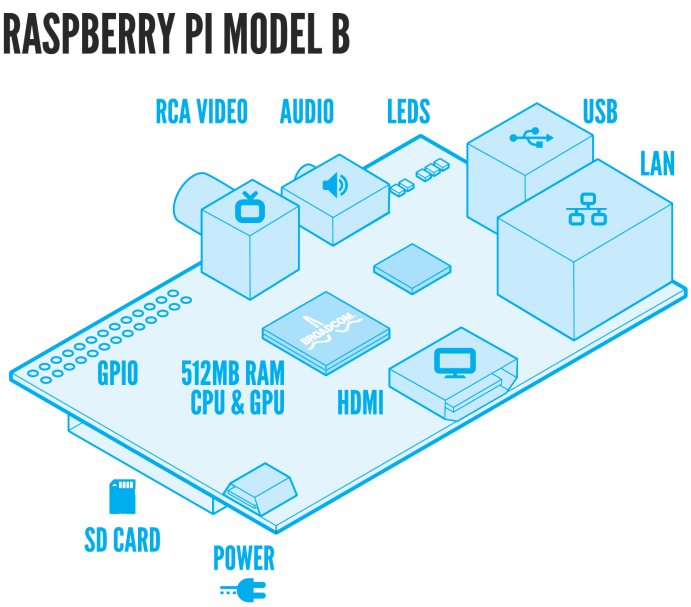

Figure 6.2: Raspberry Pi Details

http://www.RaspberryPi.org/wp-content/uploads/2011/07/RasPiModelB.png

Raspberry Pi Connectors

- **Ethernet**; This connector allows for wired network access (Model B).
- **USB**; 2.0 Ports(2x)
- **Audio**; 3.5 mm Stereo Output
- **RCA/"Phono"**; Video Output Radio Corporation of America (RCA)
- **GPIO**; General Purpose Input/Outputs
- **SD-card slot**; OS installed on Secure Digital Card (SD-card) is required for booting the device.
- **MicroUSB**; This is a 5V Micro USB power connector.
- **HDMI**; This connector allows you to hook up a high-definition television.
- **LED's**; Light Emitted Diodes.

Processor ARM CPU/GPU; This is a Broadcom BCM2835 System On a Chip (SOC) that's made up of an Acorn Reduced Instruction Set Computing Machine (ARM) central processing unit (Central Processing Unit (CPU)) and a Videocore 4 graphics processing unit (Graphics Processing Unit (GPU)). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.

6.2 How we use Raspberry Pi

The Raspberry Pi (6.3) sits in the center of our project, since the little mini-computer first of all is our media center, but also a hub that connects our software with the rest of our devices. The Pi is loaded with XBian, a simplified version of the Linux distro Debian, designed to run on ARM Architecture devices. This gives us a stable operative system with the added goods of XBMC which was developed for the original Xbox but grew in popularity and got ported to other devices.

With Linux running underneath, it gives us the complexity to add other programs, execute scripts and commands for other tasks. What we have added, is a library to interpret commands over SSH and control our NEXA power sockets. The NEXA power sockets are activated and controlled by our Raspberry Pi via our Java program that recognizes gestures. When a gesture is recognized and "light control" is selected, JAVA sends commands to our Pi over SSH, and the Pi uses the PiHat library to send the command to our power sockets.

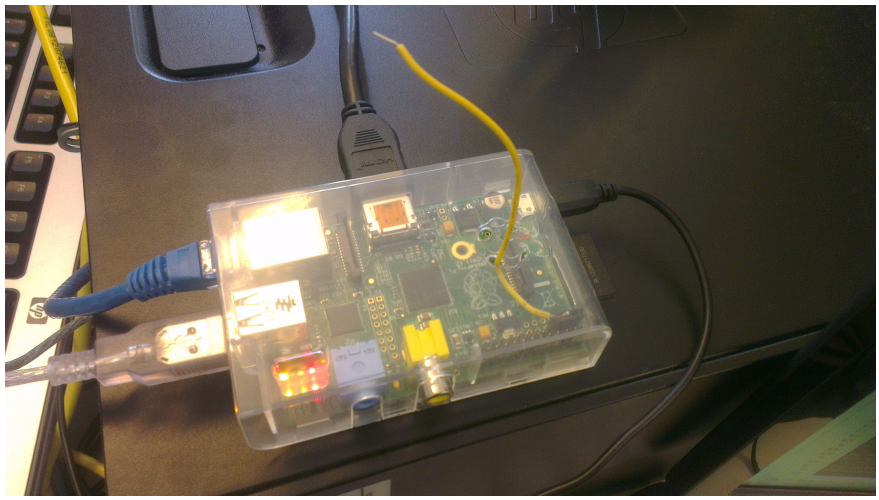

Figure 6.3: Raspberry Pi, Connected

6.3 Raspberry Pi Tools

To load our OS onto the Raspberry Pi, we need to format our SD-card, and write a pre-created image. We use XBian as it is the most stable of the different XBMC (6.4) versions.

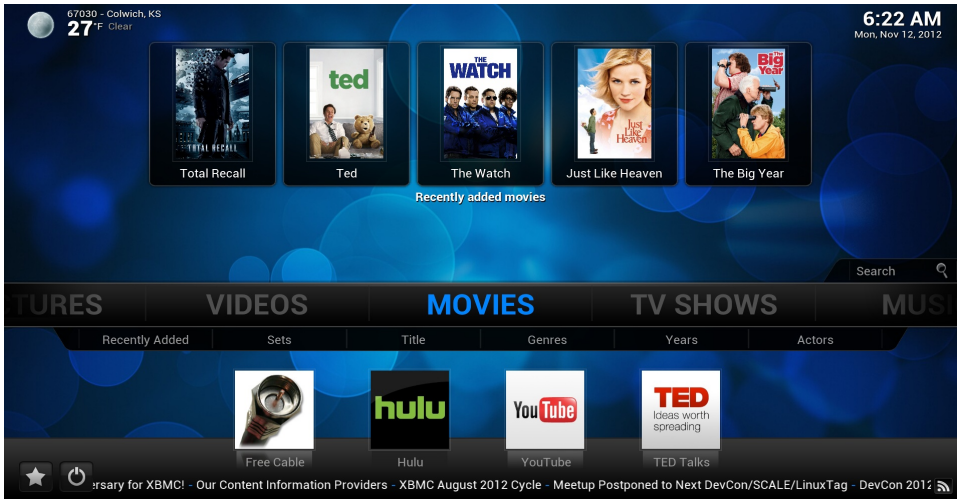

Figure 6.4: Xbox Media Center (XBMC)

http://xbmc.org/wp-content/uploads/2012/11/xbmc_beta_front.jpg

When the image has been written to the SD card, we just plug the micro USB into the Pi and with Ethernet connected, the distro automatically starts comPiling and checking online if there are any available updates. After a few minutes we get the XBMC home screen, and it is fully functional.

We have added add-ons for Youtube and some other “nice to have” add-ons, and PiHat. The library designed by Jon Petter Skabmo, evolved from the project Raspberry Pi Frequency modulation Transmitter (PiFM). This library makes the Pi transmit radio frequencies that we can use to control our power sockets.

The possibilities with the Raspberry Pi are immense. The little credit size computer is being used for more than the developers could dream of. It has quickly become a favorite of both people that want to explore the technology and people that just want a cheap computer.

Chapter 7
Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.

Arduino is intended for;

- Artists
- Designers
- Hobbyists

Arduino is basically for anyone interested in creating interactive objects or environments.

7.1 How Arduino works Quote from Arduino.cc

Arduino[14] 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 (e.g. Flash, Processing, Max Max Signal Processing (MaxMSP)). The boards can be built by hand or purchased preassembled; the software can be downloaded for free.

The hardware reference designs/Computer-Aided Design (CAD) are available under an open-source license, you are free to adapt them to your needs.

Arduino received an Honorary Mention in the Digital Communities section of the 2006 Ars Electronica Prix. The Arduino team is: Mssimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis.

7.1.1 Arduino GUI

The Arduino Graphical User Interface (GUI) (7.1) offers a simple editor interface, that reflects the simplicity and flexibility of the Arduino cards.

- **Button Bar**

  Compile, Upload, New, Load and Save Buttons
- **Input/Edit Area**

  Simple Editor with highlighting function.
- **Status Bar**
- **Notification Area**

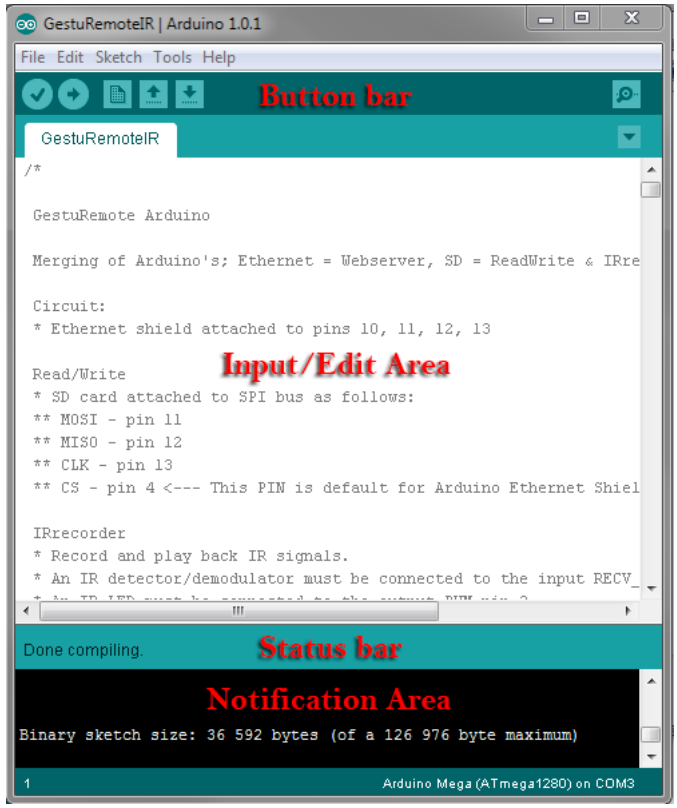

Figure 7.1: Arduino GUI

7.1.2 Arduino Mega Card

GestuRemote chose to use the Arduino Mega 2560 card (7.2) because of it's many connectivity's.

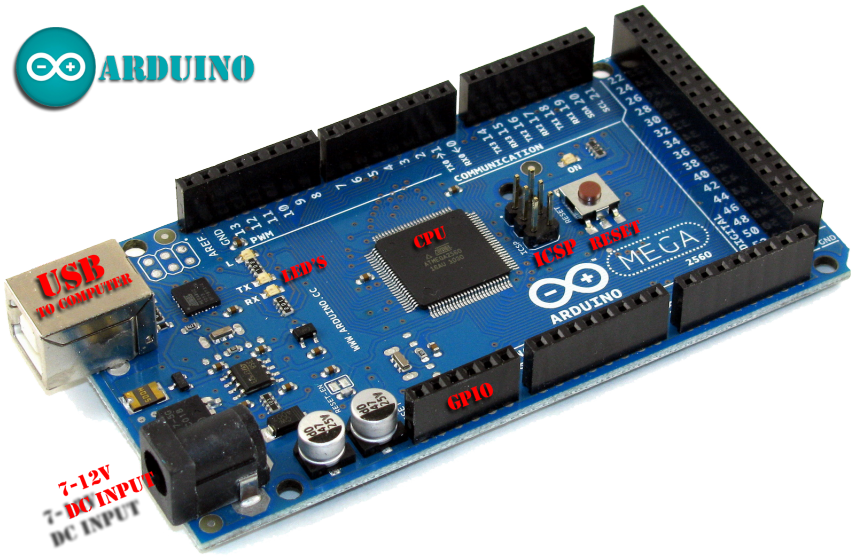

Figure 7.2: Arduino Mega 2560

http://www.robotshop.com/blog/en/files/arduino-mega2560.jpg

Arduino Board Connectors

- **USB**; 2.0 Port; 5V Power supply
- **DC Input**; Direct Current Input (7 to 12V)
- **GPIO**; 54 Digital Input/Output Pins
  - 15 PWM Outputs
  - 16 Analog Inputs
  - 4 UARTs (Hardware Serial Ports)
  - 1 16 MHz Crystal Oscillator
- **ICSP**; Method for directly programming AVR's
- **Reset**; Soft-Resets the Arduino Card
- **LED's**; Light Emitted Diodes.
- **Memory**; 256KB Flash Memory
  - 8KB Boot loader
  - 8KB Static Random-Access Memory (SRAM)
  - 4KB Electrically Erasable Programmable Read-Only Memory (EEPROM)
- **Clock Speed**; 16MHz

7.2 How we use Arduino

We wanted to build an IR/Radio Frequency (RF) recorder (7.3) and transmit recorded signals via Html-commands coming from JavaCV software.

Constructing the Universal IR Remote:

- We mounted the Arduino Mega 2560 card on a breadboard with an Ethernet shield w/SD-card connected on top.
- For optimal performance we put our components; IR LED (transmitter) and IR receiver, in strategic places.
- Wiring and status LED's was added to complete the build.
- A sketch with information on the build is attached in appendix (7).

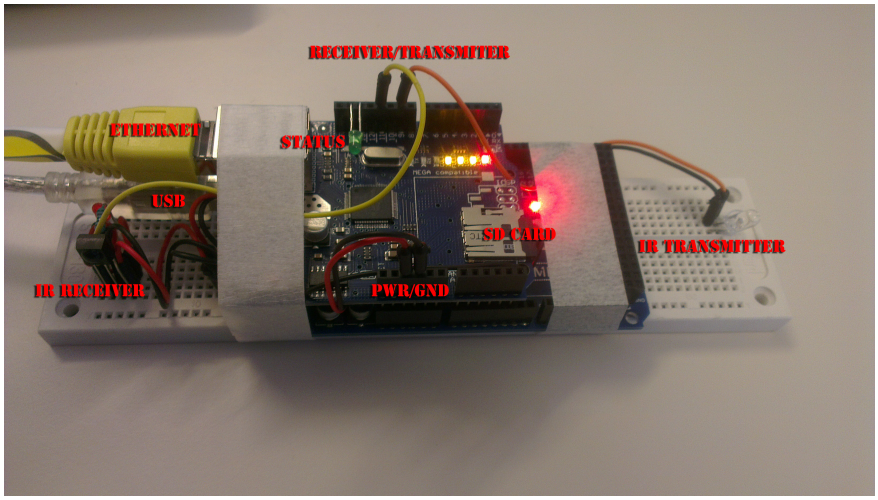

Figure 7.3: GestuRemote IRremote

Arduino community provided us with premade sketches/softwares that gave us basic codes that we tweaked to perform as we wanted. We merged three different software codes into one, which ultimatly became an universal remote that can record most known IR-codes and retransmit them whenever we wished.

7.2.1 About the Merge

GestuRemote IR Remote code build:

- **IR Remote**; IRremote code was made by Steve Spence [15]
- **WebServer**; WebServer code was made by David A. Mellis [16]
- **SD Card**; SD Card code was gathered from ladyada.net [17]
- **GestuRemote IR Remote**; Source Code is attached in appendix (B).

The **IR Remote** is the core within the arduino software. An IRremote library registers every IR signal within its reach and catagorizes and classifies it accordingly. If the IR code matches any found within the IRremote library it will classify it as coheirent to that class, if not it will be classified as raw code. The registered IR code then gets stored in an active buffer on the card until webserver gets the command to send.

The registered IR code also gets stored on an onboard Arduino **SD-card**, in an array which we want to able to obtain again upon restart of the device. Sadly we were not able completely finish this function.

Around the core functions (IR and SD-card) there is the **WebServer** "shell" code that builds up a webinterface with html-commands that will be initiated and sent when a hand gesture is registered in the JavaCV software.

The **RF Remote** code was supposed to be integrated alongside the other codes, but for practical reasons we decided to control RF commands from the Raspberry Pi module[18].

7.2.2 GestuRemote IR Remote

Our code flow (7.4) starts by initializing our webserver shell and SD card module, it then waits for any signal to be registered via IR or Hypertext Markup Language (HTML).

If any codes are registered the program matches it to its coheirent library and acts accordingly; If an unknown IR code is found the program will send its RAW data, but if an unknown HTML code is found the program will not conduct any actions. Any registered IR codes gets stored on SD card.

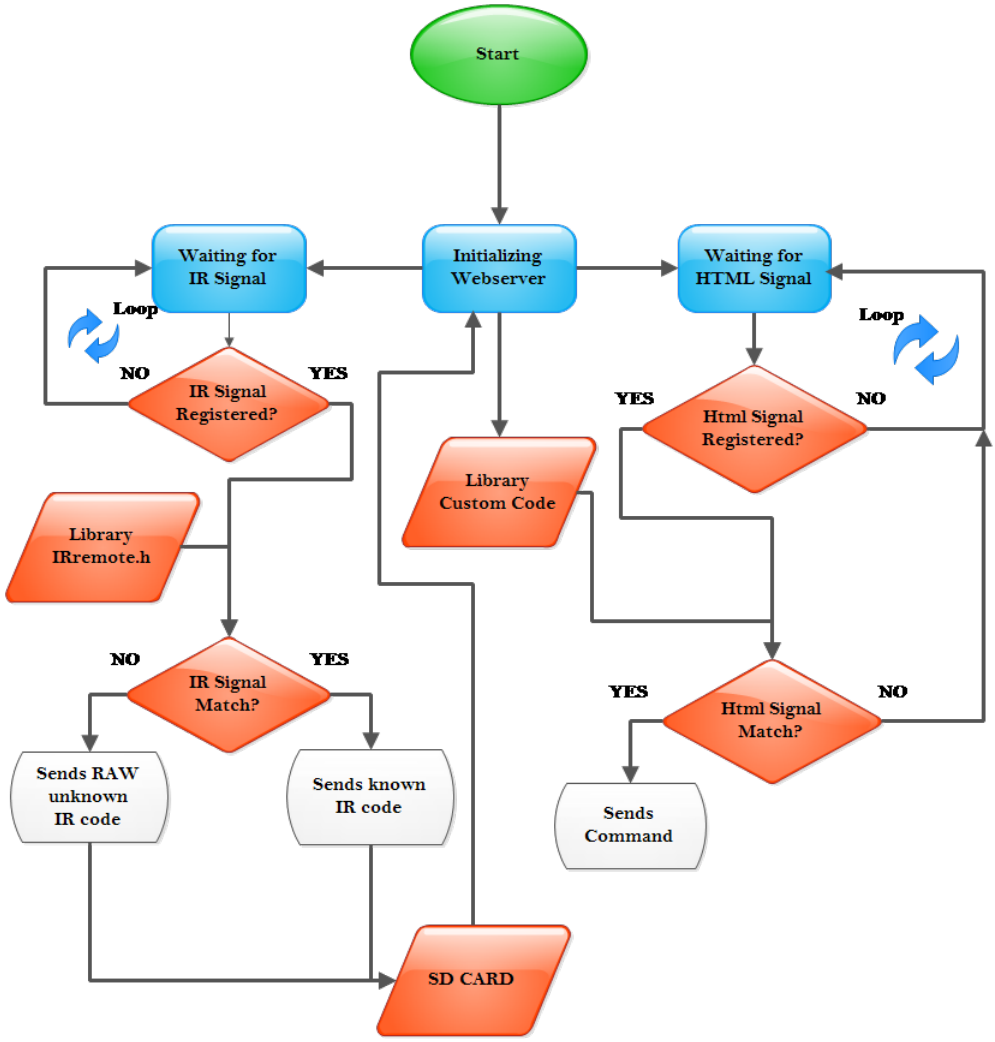

Figure 7.4: GestuRemote IR Remote Flowchart

7.2.3 Fritzing, an Arduino Sketcher

Fritzing is free-to-use software where you can construct a sketch of your Arduino projects, included with all the most common components used with the Arduino. Fritzing Software Example illustrated on Fig.(7.5)

http://fritzing.org/

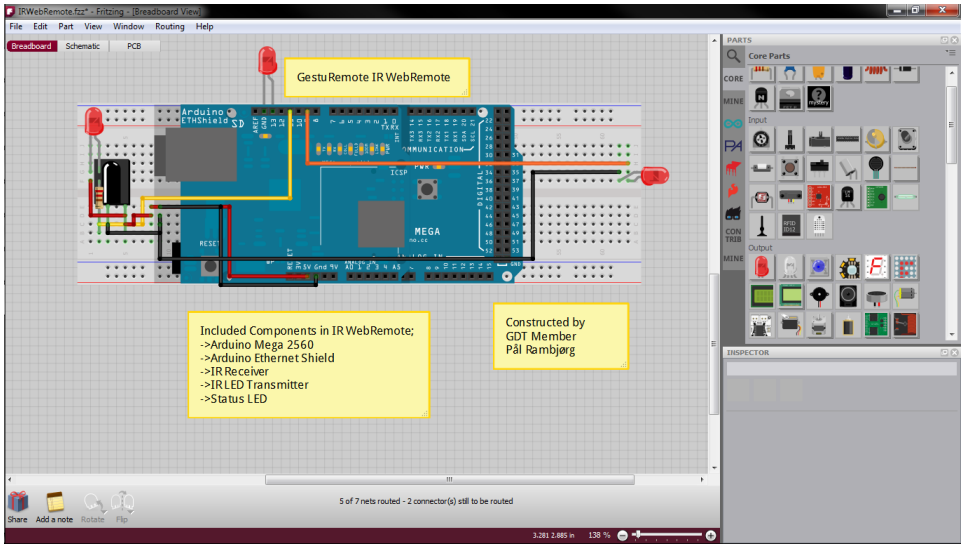

Figure 7.5: Fritzing Programming Sketch

7.2.4 Why we use Fritzing

By using Fritzing we could accurately document how we constructed our GestuRemote IR Remote model, where every component is identical to the actual components used on our model.

Frizting offers a powerful library and premade sketches linked to Arduinos own sketch-libraries.

Chapter 8
Website

As project documentation and presentation evolves, a website is a good way to present results to the public.

8.1 GestuRemote website

Giving updates as we work is a solid method of documentation and by using a website we can easily provide this information for easy access.

Website Features

- Virtual work log for the public
- Short introduction to our project and what we do
- Publish our plans and work documents for everyone to see
- Gallery for things we want to share
- "Kontakt Oss" if there is something someone wants to ask us/want to use
- Website is shown in Fig.(8.1)

8.1.1 Domain

We were given storage space for our website by Høgskulen i Sogn og Fjordane (HiSF), but this address where embedded inside the HiSF folder structure. We didn't want to buy a dedicated domain for our project, so we instead used dot.tk to redirect our "hard to remember" address to an easier www.gesturemote.tk[19]. This address redirection service is free and works with all domains.

8.1.2 Wordpress

Wordpress is a compilation of Hypertext Preprocessor (PHP) files into what is called a Content Management System (CMS), which means that Wordpress is a layer that takes care of design, functionality and a control panel for managing the site.

We chose Wordpress because of the simplicity for multiple users to edit and update the site without the need of extensive PHP/HTML knowledge. It is easy to mold a nice working site, without having to focus on all the coding.

We try to update whenever we do something worth mentioning in our project. We have also published our Gantt chart and quality/risk assessment scheme. This so that anyone can see our progress according to our plans. A short introduction to our project is also present, with some info about us as a group and why we are doing what we are doing.

As every real “project site” we have a gallery for pictures; some work related, others are “bloopers”. All in all, we are not that strict about the content only that it has to be project related. Why only show the hardcore focused work, when a project is so much more?

8.2 Website Design

In the design of our site, one thing was to be our focus: Simplicity. We wanted to present the content without much "extra fuzz", plain graphics, but still with some style. Wordpress gave us that possibility and with small tweaks we ended up with something we were comfortable with.

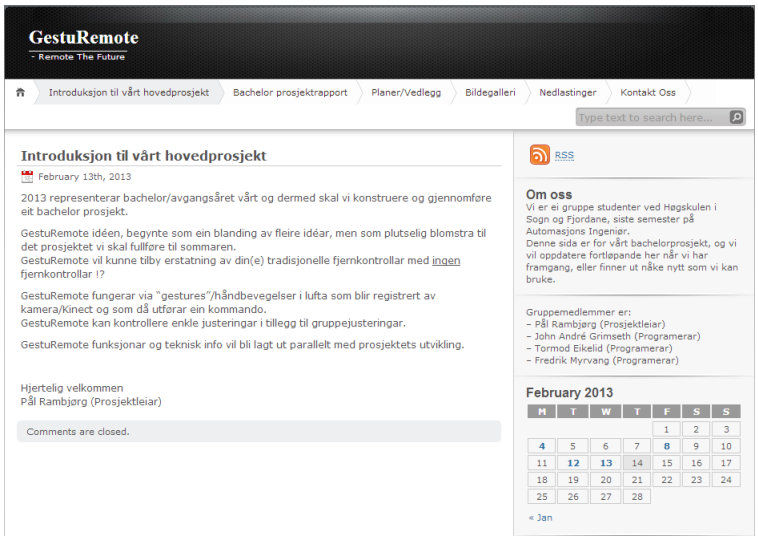

Figure 8.1: GestuRemote.tk

Chapter 9
GestuRemote Administration

Administration

Project Administration Hierarchy (9.1):

- Pål Rambjørg is the leader of GDT and has main focus on Project Report, Kinect and Arduino.
- Tormod Eikelid has main focus on the Java part of the project
- John A. Grimseth has main focus on Raspberry Pi, Webpage and Kinect
- Fredrik Myrvang has main focus on documentation (Gantt Chart, etc) and Java

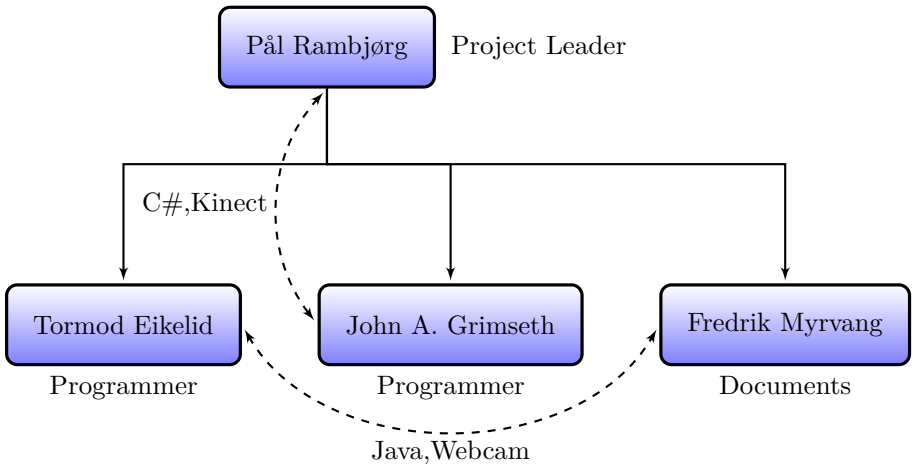

Figure 9.1: Administration for GestuRemote

GestuRemote is contracted by Høgskulen i Sogn og Fjordane (HiSF)

Chapter 10
Discussion

Through this project we have had several unknown challenges we had to overcome.

The hand detection was far more difficult than what we previously thought. Since the entire open source examples and projects were far too basic. It is understandable that any complex, working hand detection is commercial and protected.

In Java, the web camera gave us some difficulties with its poor coloring and frequent light adjustment. This problem was the hardest to overcome since we had to filter out the hand to continue the hand recognition part. We could have used a glove in a specific color that is not common in a household, but if we had to have a glove we might as well have a remote. One of the core ideas was that we should be able to control the devices without anything but a hand. Since the color filter we ended up with was not optimal we had to be even fuzzier in our gesture recognition. This gave us yet another problem. If we are not precise enough, the number of gestures we are able to recognize shrinks. This is because the properties of the gestures start to merge and then one object can be within the properties of more than one gesture. This is why we used vectors. With them, the shape of the object doesn't matter and we get enough functions for the hand to control some basic functions on our chosen devices. We do need to wear a sweater that is not red for this to work though, since we have not found a way to successfully split the hand from the arm.

The Kinect part of our project was not trouble free either, as we used many different programs, both official and unofficial. The results where that most of the time some parts of the programs would work, and then other parts made it crash. After a while we decided to split up the installers, and we ended up reinstalling a new computer just to start fresh. We the figured the correct installation procedure to make our current samples and programs work like they should.

Kinect software is mostly written in C# or C++. This also forced us to research more of the layouts and the language, rather than just the code that we might need. This made modifying and molding the samples to something that we could use that much harder. We lost many hours researching everything from installation procedures to trial and error with samples that would not compile or that where flat out broken cause of updated tools or depreciated libraries.

When we finally conquered most of the errors and made samples behave like we wanted to, time was against us and we looked more towards Java and JavaCV. This was sad, as the Kinect is made for what we were to use it as; to capture gestures and recognize shapes/movements. At least we got an introduction on what the Kinect is, how it works and last but not least; how powerful it can be given enough time to write software for it.

Compatibility issues has slowed us down allot. With all the different libraries and programs that had to work together with the right bit version and the right software version. And even if all the new software is compatible with each other, it still needs to fit the computer and its pre-installed software. We actually ended up with the installing JavaCV with all the underlying programs in 64 bit on one computer and 32 bit on another, even though they both were 64 bit machines.

We got another compatibility issue when we tried to document our program for the first time. One of our computers crashed because of a graphic driver that did not work well with capturing the screen while processing an image. This driver somehow became corrupt and we could not update it, even after we tried to reinstall the computer. We then installed Linux which managed to force an open source graphic driver to run the graphics. This worked fine, but working within Linux gave us grater compatibility issues when it came to everything else. So we installed windows again and decided to document in another way. When we reinstalled windows everything worked.

We can't explain what was different this time, but we could now document the running program. That a computer may crash is something we accounted for in our risk assessment, so the whole project was always stored online so we only lost the last lines that were written that day when the crash occurred.

To execute our commands where Raspberry Pi and Arduino boards chosen cause they fit right in with our needs. Raspberry Pi made it easy to implement a media center so we could control it with gestures. We can now move around inside the menus, turn up and down the volume and change file with gestures. The media center commands where extracted from already existing implemented methods of communication. We feel that this is a good way of taking advantage of already implemented solutions.

Adding light socket control to the Raspberry Pi did not make any problems other than the need for an antenna out of the Raspberry Pi box, but that was an easy fix. Again, implementing an known working solution to our project, with minor tweaks.

Arduino with its programing language similar to Java, which we already knew how to program, and its large collection of software samples made it easier to use. Our infrared remote signal duplicator circuit was made with some trial and error from smaller pieces of code, and then weaved together to create one code that did what we wanted it to do. This was indeed a lot of trial and error, but we also feel that we learned a lot from this kind of work.

The one thing with Arduino that gave us the most hassle was storing the codes decoded from the remotes to our SD card. This took some time, but in the end we made it work as we wanted it to. What we didn't got to work however, was to put the codes from the SD card back in the buffer to be sent.

One of the criteria's was that we were to design a website. When we got our designated area on the HiSF servers for our website, we got a link that where deep inside the server folder structure, and we thought that this was going to be hard for people to remember. We could have spent money on a dedicated domain, but then again that would have to be a subscription. We would rather have it on HISF servers, so we used a free redirection service and ended up with the domain www.gesturemote.tk.

Towards the end of our project (6), we made some changes in our gantt chart so that we were able to complete the project. We feel that this is justified because of different obstacles we could not have foreseen. In the beginning we did not know what we were to expect out of this project, and everything was uncharted territory.

Chapter 11
Results

The resulting software does recognize a gesture, under the right circumstances.

We have five gestures we can recognize: Left hand, right hand, one, two and three. By using these gestures we can enter the desired device. While in the desired device, we can exit, change device or interact with the device. Interaction is done by vector movement based algorithms.

When we have found the right device and given the right command, the software sends commands to the selected device. If media center is selected, the software sends JSON-RPC commands over TCP network, if it is to turn the TV on or off, we send HTML commands and if it's light control, we send SSH commands. All this is connected to the device gestures.

11.1 Graphical Project Description

This is a graphical description of the process through GestuRemote Software (11.1)

- **HandGesture**
  - A persons hand presented in right angle and posture.
- **JavaCV**
  - Recognizes a hand if present and registers any movement.
  - Send commands to Arduino or Raspberry Pi.
- **Arduino/R-Pi**
  - Arduino: Registers/Stores and Sends IR signals to coheirent component.
  - Raspberry Pi: Receives Html commands, Controls XBMC.
- **Storage**
  - Stores every registered IR signal on a SD card (erasable)
- **Visual Component**
  - Tv, Stereo and Lights: Controlled by Arduino or R-Pi.

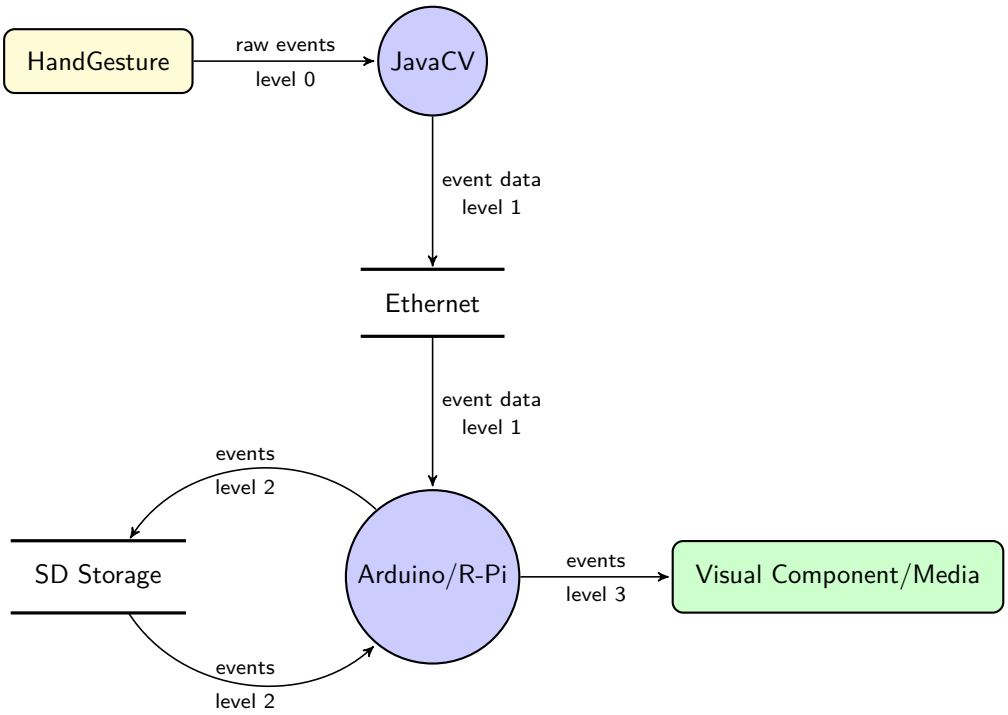

Figure 11.1: Process grid

Chapter 12
Further Work

Our open source code will be published on our webpage [19] so anyone who want to develop it, borrow a method or just get some inspiration are welcome to try it out.

To make this a better functioning program, there is a need for;

- A better color filter that changes its values depending on the light.
- Fitting it to a camera with better coloring.
- Adjusting the parameters of the gestures.
- Including Kinect to get a depth perception.

Chapter 13
Conclusion

This project was meant to teach us more about “gesture based” software, and what it actually takes to control objects with gesture tracking/recognition.

Even though the recognition needs the right illumination to work, we feel we made our main goal. After researching, testing and failing, we did wave on the TV. The knowledge we acquired through our research has shown us that our idea for controlling everything within a household with the wave of a hand, is possible. The communication methods have given us a greater diversity within computer communication. Switching between programming languages has taught us that most programming languages are the same, with the same logic, just with different ways of writing the commands. We now feel qualified to jump in to any language and "learn on the go".

Having this bachelor project has given us training in running a technological project ourselves. Where we have had to document all the hours used and what we used them for, so we can keep improving the approach to the next project. Finding information, self-learning and discipline was crucial for this project, and we feel we succeeded in all three points. We had to take the information we found and interpret it ourselves, and even think of new methods without any information. Setting work hours, pushing each other, and distributing jobs so everyone had something to work on at all times helped in completing this project.

References

[1] LaTeX3, "Latex a document preparation system." http://www.latex-project.org/.

[2] S. Crawford, "How Microsoft Kinect works." http://www.howstuffworks.com/microsoft-kinect2.htm.

[3] Kinecthacks, "Kinect hacks." http://www.kinecthacks.net/.

[4] S. Stegmueller, "Candescent nui." http://candescentnui.codeplex.com/.

[5] S. Stegmueller, "Candescent license." http://candescentnui.codeplex.com/license.

[6] Kinecthacks, "Sigmanil is the most powerful vision framework.." http://www.kinecthacks.com/sigmanil-is-the-most-powerful-vision-framework-for-natural-user-interfaces/.

[7] Nash, "Skin detection." http://bsd-noobz.com/opencv-guide/60-4-skin-detection.

[8] T. O. Development, "About open cv." http://opencv.org/about.html.

[9] J.-R. W. Group, "Json-rpc api." http://www.jsonrpc.org/specification.

[10] Techterms.com, "Secure shell." http://www.techterms.com/definition/ssh.

[11] S. Hunold, "Welcome to sshtools." http://sshtools.sourceforge.net/.

[12] J. P. Skagmo, "Pihat." http://www.skagmo.com/page.php?p=projects/22_pihat.

[13] B. Johnson, "How the raspberry pi works." http://computer.howstuffworks.com/raspberry-pi2.htm.

[14] Arduino, "Arduino." http://arduino.cc/en/Guide/Introduction.

[15] S. Spence, "Arduino ir receiver." http://arduinotronics.blogspot.it/2012/11/arduino-ir-receiver-part-2.html, November 2012.

[16] Arduino, "Webserver." http://arduino.cc/en/Tutorial/WebServer.

[17] Ladyada, "Micro sd card tutorial." http://www.ladyada.net/products/microsd/.

[18] S. Nilsson, "433 mhz rf nexa." http://sebastiannilsson.com/projekt/arduino/433-mhz-rf-nexa-saendare-och-mottagare-med-arduino/, Februar 2012.

[19] GestuRemoteTeam, "Gesturemote." www.GestuRemote.tk, 2013.

Appendix A

System documentation

Appendices

1. **Logbook**
2. **HiSF Contract**
3. **Meeting Minutes**
4. **GestuRemote Budget**
5. **Quality and Risk Assessment Scheme**
6. **Gantt-Chart**
7. **GestuRemote IR Remote Sketch**

– **Appendices is provided on an external Compact Disc (CD).**

Appendix B
Source Code

GestuRemote Codes

GestuRemote **Java**; Code generated with Java software.
GestuRemote **IR Remote**; Code generated with Arduino software.

Source Codes is provided on an external CD.