The Magic Wand

Idar Douglas Hillgaar

A dissertation submitted to the University of Dublin, Trinity College

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

September 2006

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Idar Douglas Hillgaar

Dated: September 15, 2006

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Idar Douglas Hillgaar

Dated: September 15, 2006

To My loving parents and family for their unquestionable love and support

Abstract

The paradigm of computational accessability where the world of devices available to us are of an ubiquitous nature. Some questions arise naturally when we consider a world where the environment around us can talk back

> *With the introduction of The Magic Wand metaphor in the ubiquitous environment, this project is hoping to raise questions on how interaction might be performed between the user and this environment through gestures that would be considered intuitive*

Acknowledgements

With these words I would like to give my thanks to some very important people who has been a great help to me, in completing this dissertation. First I would like to thank my supervisor Dr. Mads Haahr for his help in guiding me to understand Ubiquitous Computing, and gave support and ideas for the thesis. A special thanks goes out to Dr. Peter Barron who has helped me with Linux and C++ questions, Dr. P. Barron also put together the Gumstix and set it up for me for the evaluation. I would like to thank George Maistros, a research postgraduate student from the University of Edinburgh who aided me to implement the C++ wrappers for the GHMM library. I would like to thank Alison Ryan and Dr. Peter Barron for proof reading and giving me valuable feedback. Finally many thanks to friends, family and Alison Ryan for their support and patience in completing this dissertation.

Idar Douglas Hillgaar

University of Dublin, Trinity College

September 2006

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

We go to shops, banks, childcare centers, photo booths and also interact with an array of devices in the home too, like washing machines, TV's, microwave ovens and lights, etc. The defacto method of interacting with all these devices and contexts are by a payment card, switches, remote controls, knobs and handles. This report asks if another form of interaction by the Magic Wand metaphor is more suitable for this interaction?

1.2 Metaphor for The Magic Wand

The wand is a pre-Norman unit of length used in the British Isles equal to approximately the modern metre, apparently dating from an early use as a yardstick [3]. The wand is used in ceremonial or in official contexts where a mace, sceptre or staff is used to symbolise authority or representation of power. This representation of power

together with the magic elements from Harry potter where he makes a gesture and points at an object of interest to perform that magic, is the association the author has in mind at the development of The Magic Wand (called TMW in the rest of the paper).

1.3 Technical Challenges

The author had to research gesture recognition, gesture storing, Linux, C++, implementation for portable computers using Xscale, sensors and Tiny Linux, and communication technology and methods for interfacing with 3rd party libraries.

1.4 Tangible User Interface (TUI)

'Tangible interface give physical form to digital information, employing physical artifacts both as representations and controls for computational media. TUI's couple physical representations with digital representations, yielding user interfaces that are computationally mediated but but not necessarily identified as the common perception of a computer' [8]

1.5 Dissertation Objective

This project sets out to prove that a computational tangible user interface, Magic Wand, could be implemented as an extension of our will to manipulate objects of interests.

1.6 Outlining the Dissertation

Chapter 2 covers the state of arts of current and past good projects in gesture recognition by a device. Chapter 3 covers design alternatives and techniques with the final UML project details. Chapter 4 is the final implementation and discussion of why some solutions were selected. Chapter 5 Evaluation of user perception af TMW and TMW met its criteria set in chapter 3. Chapter 6, concludes on the success of the project. Chapter 7 covers where author recommendations for future work.

Chapter 2

State of The Art

2.1 XWAND

2.1.1 Implementation

This project went out to test several gesture recognition techniques to utilise a wand for intuitive interaction. Several sensors were embedded on the wand to return orientation and acceleration data, this data where sent to a main computer, that analysed the data values to gestures. XWand [11] as shown in in figure 2.1 consists of a number of embedded sensors that detect the orientation and acceleration of the wand. The data from these sensors is transmitted wirelessely, when a button is pushed on the wand to a remote computer. On analysing and interrupting the sensor data the computer can, if a gesture is recognised, perform an action associated with gesture.

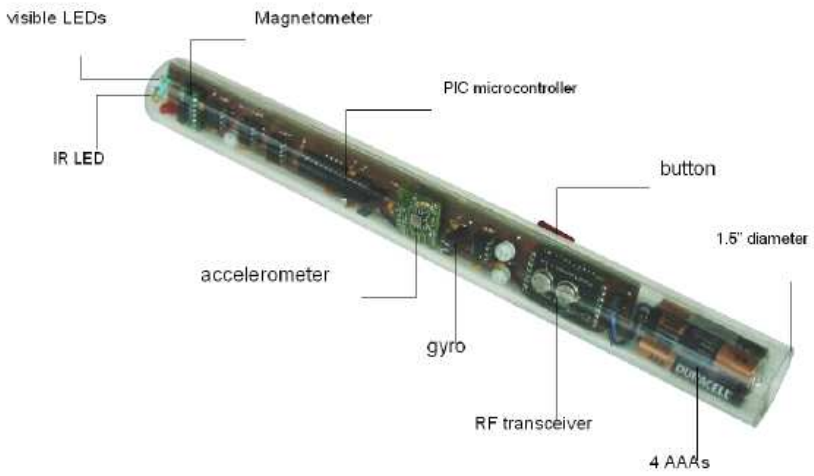


Fig. 2.1: Hardware Overview of the XWAND

2.1.2 Software Architecture

XWand have implemented three different gesture recognition techniques, namely Linear Time Warp, Dynamic Time Warp and Hidden Markov Model.

2.1.2.1 Hardware Architecture

Linear Time Warp (LTW) LTW algorithm matches a sequence of sensor values $S = S_1, ...S_T$, with a stored prototype, $p = p_1, ..., p_T$. Where the prototype is the trained gesture, it can be matched with a real world gesture by squared Euclidean distance for every time t. The Matched is found through computing for the whole sequence given time and scale, where maximum score over scale and warps is considered

the match.

Dynamic Time Warping (DTW) DTW [28] got prototype gesture $p = p_i, ..., p_t$ and gesture for recognition is $S = S_1, ...S_T$, and we have a k-dimensional feature vector with values collected from k sensor. Each element of grid contains a Euclidean distance measure D_{ij} representing the distance between s_i and p_j. The best time warp will be the minimised optimal distance.

Hidden Markov Model (HMM) The implementation of HMM [21] was compiled by the HTK [26] library that retrieves gestures.

2.1.3 Feedback

Feedback is provided to the user in the room at a success or failure of the gesture being recognised. This is displayed on a nearby screen. There a certain amount of lag in the feedback due to the user inconsistencies by button press.

2.1.4 Analysis and Critique

As the paper went on to find gesture recognition techniques and generic gestures, they where successful at finding the better technique and some generic gestures. However they met some problems at the push button as users held the button in at various length of time and distorted the gesture, as it is time and amplitude dependent. Other issues at using the library and algorithms that were not addressed, were if they supported multiple readings at time t. Since the architecture depends on a master computer to analyse the gestures prior to routing a command to the correct device. The architecture implies interconnected devices with infrared interfaces and a dedicated computational

machine. Robustness can be said to be low, but the paper did not go in depth with in to the pervasive elements other than in the introduction but is explored in depth in later XWand research papers [4] Since the wand is only sending out data at button press, battery life time can be similar to a common remote control if the sensors and radio were turned on at button press.

2.2 An Inertial Measurement Unit (IMU) for User Interfaces

2.2.1 Implementation

Gesture recognition based on atomic gestures [9]. A. Y. Benbasat's M.Sc dissertation describes how all atomic gestures are hard coded, and when performing a gesture the application looks for all these atomic gestures that makes an advanced gesture. The IMU for user interfaces project uses raw data from the sensor, that it filters by kalman filtering, scales the sensor data and recognise by Matlab linear feature detection. This feature data is then run through a windowing algorithm that checks how many features are in this window. By combining these windows, by scripting, the results are the identification of a trained gesture. The final application was 3D character walking and performing some actions like kick etc from real world gestures performed on a physical doll.

2.2.1.1 Hardware Architecture

Sensor unit for acceleration data and gyroscopic data, sends its raw data to an embedded device for transmission to a master computer, figure 2.2. The master computer

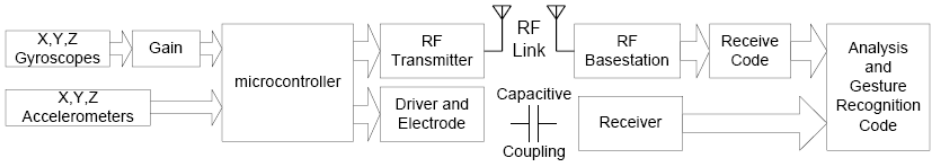


Fig. 2.2: Architecture of information flow of the atomic gestures, IMU project

identifies the gestures from stored training sets.

2.2.1.2 Software Architecture

The sensor data is collected by embedded code to a buffer of 24 bytes, then the data is radio transmitted. Feature algorithm, with the generic windowing algorithm detects sensor activity where a gesture recognition is applied for atomic gestures.

2.2.2 Feedback

Synthetic Character Group created by MIT Media Laboratory, where user controls movement of a character based on gestures.

2.2.3 Analysis and Critique

The final application and testing showed a 87% [9] correctness where user got 2 trials per gesture to get the right results. They proved atomic gesturing can be used to infer more complicated gestures, however the atomic gestures are dependent on the sensor raw data from some very specific sensors and thus the sensor parameters were

implemented by method of qualified estimation. Since the implementation suggests incremental atomic gestures must be recognised before a complete gesture model can be reached, then there are issues when some gestures are equal to eachother except in time. This system will recognise the shortest gesture first, so if the user continue the gesture to a longer gesture, the recognition algorithm will think it has found the gesture and will move on to the next gesture.

2.3 Glove is in the air

2.3.1 Implementation

Glove is in the air (Glisa) [17] is a project to integrate an analysis tool *SciCraft* with a human computer interface enabled by gloves to control this virtual reality application.

2.3.1.1 Hardware Architecture

Hardware of Glisa, depicted in figure 2.3 consists of *flock of birds* orientation sensors on gloves to utilise orientation information on a 3D virtual environment displayed on a projected screen.

2.3.1.2 Software Architecture

The application is described to be a middleware consisting of orientation data, mouse emulator and gesture recognition. Gesture recognition is done by the hidden Markov model, visualised in figure 3.4.3, by using the Open Source, Generalised Hidden Markov Model [5] developed by University of Berlin. Gesture information and orientation data is fed directly into the virtual reality world and enables user to orientate and interact with objects in this world. As the GHMM library will only accept one sensor reading

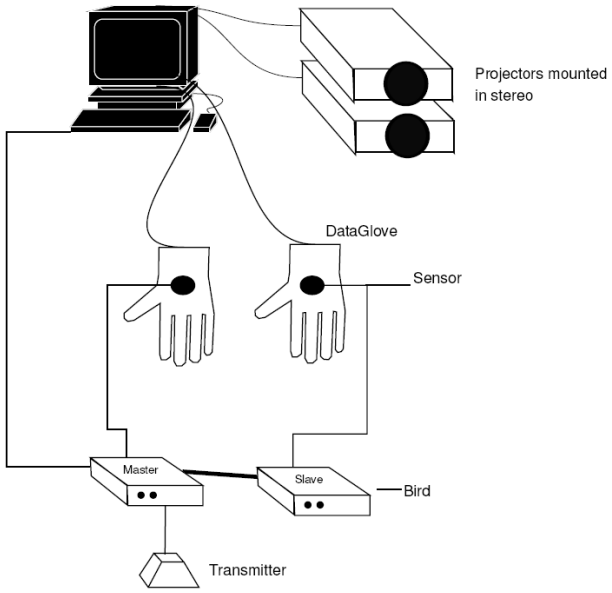


Fig. 2.3: Hardware Architecture of Glisa

at time t. The average of several readings for time t were implemented to overcome the limitations of the library. A threshold determines if the match is good or not so the GHMM library will not output false positive gestures. A wrapper was created to control the windowing system's mouse pointer and then a 3D orientation to 2D orientation coordinate was implemented. Data from the orientation of the fingerprints was triggered by a threshold scheme to identify no more than 2 such postures. These postures are used as mouse button 1 and 2.

2.3.2 Feedback

The Glisa project was divided into three parts. Part one handles the sensor drivers for 2D and 3D space and device communication, part two is integration the glove on a normal windows program and part three to enable gestures to a virtual environment.

The two first parts of the project was to enable the glove and simply use it to point and click on a 2D environment, where the mouse icon is the general feedback. The last part where the glove interact in a 3D environment is actually done by 2D movement together with 3D gestures. the visual feedback works similar as to the mouse icon, but through gestures interactions with objects in a 3D environment are enabled.

2.3.3 Analysis and Critique

The customer driven project at NTNU developed a glove interface with 2D and 3D computer environment, which is not in fact a new Human Computer Interface, the integration of 3D interactions with a scientific tool and a virtual reality library and gestures interaction is new.

The middleware that enabled the glove were implemented on Linux's X window system, the interaction with the VR lab [2] where a 3D visualisation was used, the glove's interaction was pretty rudementary by using a gesture linked to some action. The HMM library did not utilise gesture recognition where several values must be read at time t, so the way around that was to use a median value per time t, which, in practical terms this means that it's how the gestures are made determines recognition rather than the gesture itself. See GHMM figure 4.1.2. False positives in gestures are solely determined by a threshold of log(p) per gesture. A probability threshold of this manner will indicate that few gestures will be used that does not look like each other from the statistical data and is pre-set by the trial and error method.

Movement on the screen is 2D and movement with the glove is 3D. The direct correlation from 3D to 2D was done, so to utilise the 2D screen the glove moves in a 2D coordinate system, not using the gloves 3D capacity. Gestures were quite limited and was used to utilise a 3D environment where movement is still 2D.

2.4 the magic wand

2.4.1 Implementation

The Virtual landscape with a flying carpet was implemented to display how the 'Magic Wand' and speech can enable user to travel through this landscape.

The sensor data gives orientation information for the Wand, while voice commands will direct the simulator to act on wherever the wand is pointing. Voice recognition is possible by speech recognition library.

Application consists of a virtual landscape that take orders from speech commands and direction of orientation from the wand.

The Magic Wand [14] demonstrated how the concept of a 'Magic Wand' can be used to interface with a virtual landscape on a large screen. Where voice and wand to aid interaction within environment and objects that occupy it.

2.4.1.1 Hardware Architecture

The hardware elements consists of headphones, a microphone, a wooden wand, a directional sensor and a computer connected to a large screen. A wooden wand is used to eliminate data contamination a metal rod will cause the magnetic sensor. The microphone on a headset was the choice over a stand alone microphone, placed in the centre of the room cause interference and echo.

2.4.1.2 Software Architecture

The Sensor fusion techniques analyses speech input and determines the correct wand sensor orientation for the 3D landscape. These two different pieces of data is used to manipulate the 3D objects and to maneuver through the 3D landscape. The Magic

Wand's architecture is outlined in figure 2.4

2.4.2 Feedback to user

When the interaction is dependent on two different actions, like speech and gesture to perform an expected action. Some kind of feedback is vital when something goes wrong or the expected actions is triggered. The feedback put in place where to warn about error in speech, not however in gestures as they will work or not work.

The feedback should either be visual, audio or both, in an example the user says a command word where the system do not recognise the word and returns an error message and the user tries again until he gets the speech command right.

2.4.3 Analysis and Critique

This project utilises euler data for directions. The directions are divided into 5 sectors where the middle one is the point of origin and the other sectors are divided into up, down, left and right. In order to fly through the landscape a direction is chosen from the point of origin, then a command is given ie "fly". The virtual landscape will then let the user fly to the destination until a new command is given. The approach of direction together with speech commands give some benefits where end users can use the wand intuitively and the speech commands will not appear to be hard to learn as they are relative few.

The projects used an early sphinx package that gave 30 Percent failure rate, this degree of failure is quite substantial when regarding only 6 keywords where used.

Issues with the magnetic sensor, when calibrated for the room where the sensor data are universal compared to the earths magnetic pole. When changing locations the sensor will be off course, unless it was reset for its world.

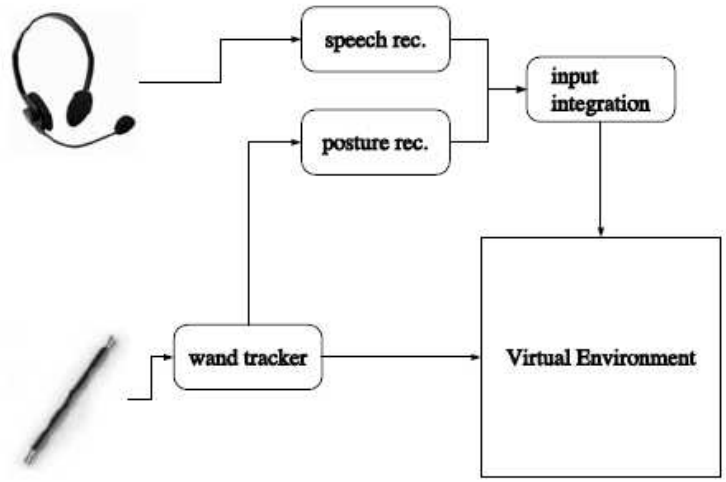


Fig. 2.4: Overview of 'The Magic Wand' Architecture

This robustness question where the sensor is pointed to the centre of the image in other locations than the prototype locations are solved easily by re calibrating when pointing at the point of origin.

2.5 Evaluation

Where the XWand and IMU deals with finding the optimal gesture recognition techniques and by comparing the different technologies, its safe to assume that they can be used in similar applications, depending on architecture and hardware. XWand, Glisa and The Magic Wand deals with calibrated sensor data, where focus is on sensor fusion capability for application integration.

The Techniques in Gesture recognition The Linear Time Warping received only 40% correct gestures [11], Dynamic Time Warping received 71% [11] and Hidden Markov Model has received everything from 70 to 90% [11] [17] on its gestures. The atomic gesture approach received in its study around 87% [9] although with questionable quantitative evaluation data.

2.6 Concluding State of the Art

In concluding the state of the art the author will deal with the papers in three categories to get each papers contribution to the field.

Sensor Data The goal for any implementation is sensor data that are returned in a manner that we can understand generically, that can be used for any sensor fusion techniques. As such, the XWand, the Glisa and the magic wand all returned discrete values, while the IMU project returned raw data that will need fusion techniques to return discrete values.

Sensor Fusion For gesture recognition the Hidden Markov Model was the most successful technique. HMM success depend on using the most suitable libraries that supports advanced and unique gestures through sensor data accuracy. The IMU project is parameter, threshold dependent on specific sensors and will prove more to more labour intensive for a generic implementation by anyone who will attempt on using other sensors and hardware but using the same technique.

Application The state of the art projects all implemented for an existing a 3D application to prove gestures in the real world could manupulate object in the virtual world,

the XWand started off as a proof of concept and is now considered to be implemented with Microsofts XBox 360.

Chapter 3

Design

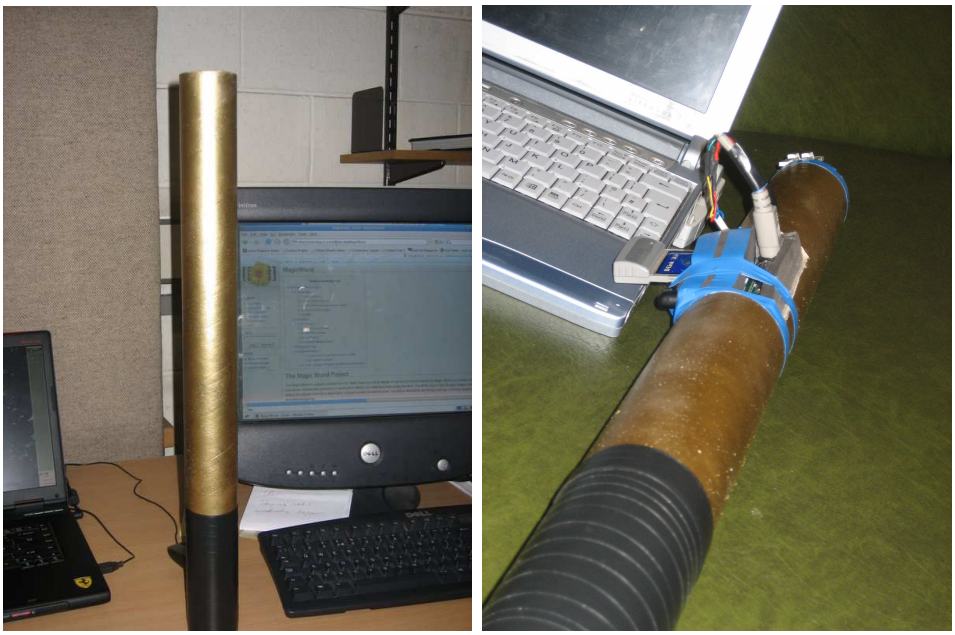
3.1 Overview of TMW

In this dissertation we propose to use the metaphor of a magic wand in figure 3.1 to provide an intuitive tangible interface to an ubiquitous environment. In the following chapter we describe and analyse some techniques available, hardware and finally design the application.

3.1.1 User requirements

When the users need to interact with ubiquitous devices. This requires some minimum requirements for the user to use the wand efficiently. The wand needs to interconnect with devices in commercial and business contexts and be linked with the user making the commands. The requirements for a Magic Wand interface to meet end user demands, are, intuitive use, autonomy, performance and visual/audio feedback to confirm some action.

To make end users more productive, TMW would need to be as common as the



(a) TMW, wired to computer (b) TMW, wireless

Fig. 3.1: TMW

mobile phone or available wherever it might be needed, like shops, museums, presentations, visual or audio systems, payments, security, theme parks etc.

TMW in brief

- Tangible user interface
- Intuitive use
- Association of a gesture with a reaction

3.1.2 System requirements

Users must be able to take advantage of its autonomous quality and the mobility TMW offers, the wand needs to be small, be computational, build in security and have wireless connectivity.

Hardware requirements

- Inertial Measurement Unit
- Processing platform
- Wireless communication

Software requirements

- Discrete Sensor Data
- Sensor Fusion technique
- Gesture to command association

- Communication application

3.1.3 Environment Devices

Ubiquitous enabled devices are any device that an end user would come across in almost any interactive context. These devices can be activated by the user's TMW and receive commands for them to respond to.

3.2 TMW System

TMW interaction, occurs when a user choose a device to interact with and then what action to take. TMW should keep details on gesture models, recognise such models and return a command to the device. The problem that should be resolved before implementation is how to best identify the correct device for interaction, as there could be hundreds to choose from at any location.

3.2.1 TMW Application

TMW approach for interaction would be for the user to point at the device, then to perform a gesture where the device would perform some kind of action. When transmitting a command there are two alternatives on how the commands could be, the first as raw data, where TMW transmits a perceived gesture, and the device receives all raw data to post-process the information into useful commands. The second method is when TMW itself identifies gestures, returns the command and transmit it. The first method leaves all post processing at the device side and it is given that a ubiquitous device should carry intensive processing capability and vice versa for the second method.

3.2.2 The Pervasive Environment

The ubiquitous environment must be enabled to meet a common standard for it to be interfaced with by TMW. How can the user find, identify and communicate with these devices that could be numerous and densely populated indoors and outdoors? These are some of the issues.

How to distinguish a device from other devices:

- Listen on a specific IP address
- Activate on infrared channel
- Bluetooth activation

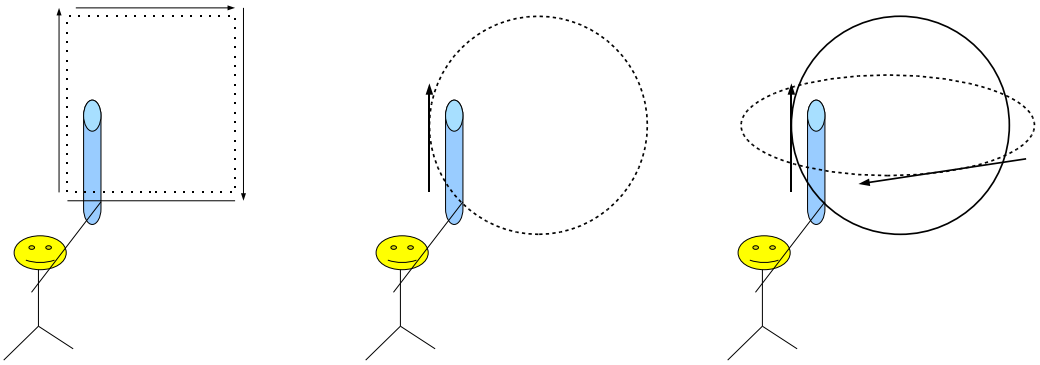
Examples of TMW communication techniques:

- UDP protocol
- TCP protocol
- Bluetooth
- ZigBee

3.3 TMW Gestures

The assumptions are, that users would use the wand on a regular basis where interactions are the gestures performed by TMW as in figure 3.2. These gestures can be familiar or unfamiliar, based on the users previous knowledge of gestures. Example

of issues with gestures are a circle interpreted in many ways, as a fast small circle, a large slow circle, an oval circle seen in figure 3.2(c). A square could be oval or it could be a strict square with small pauses at each corner. To find out what type of gesture pays off for the general public the author has considered two options, the first is where the wand ships to the user ready with generic gestures, the second design could allow end-users to create their own unique gestures from an available training mode.



(a) Squared gesture

(b) Circle gesture

(c) Circle interpret in two ways

Fig. 3.2: Example of gestures

3.3.1 User Specific Gestures

This scenario is one where, the user picks his own gestures, and a few assumptions must be made.

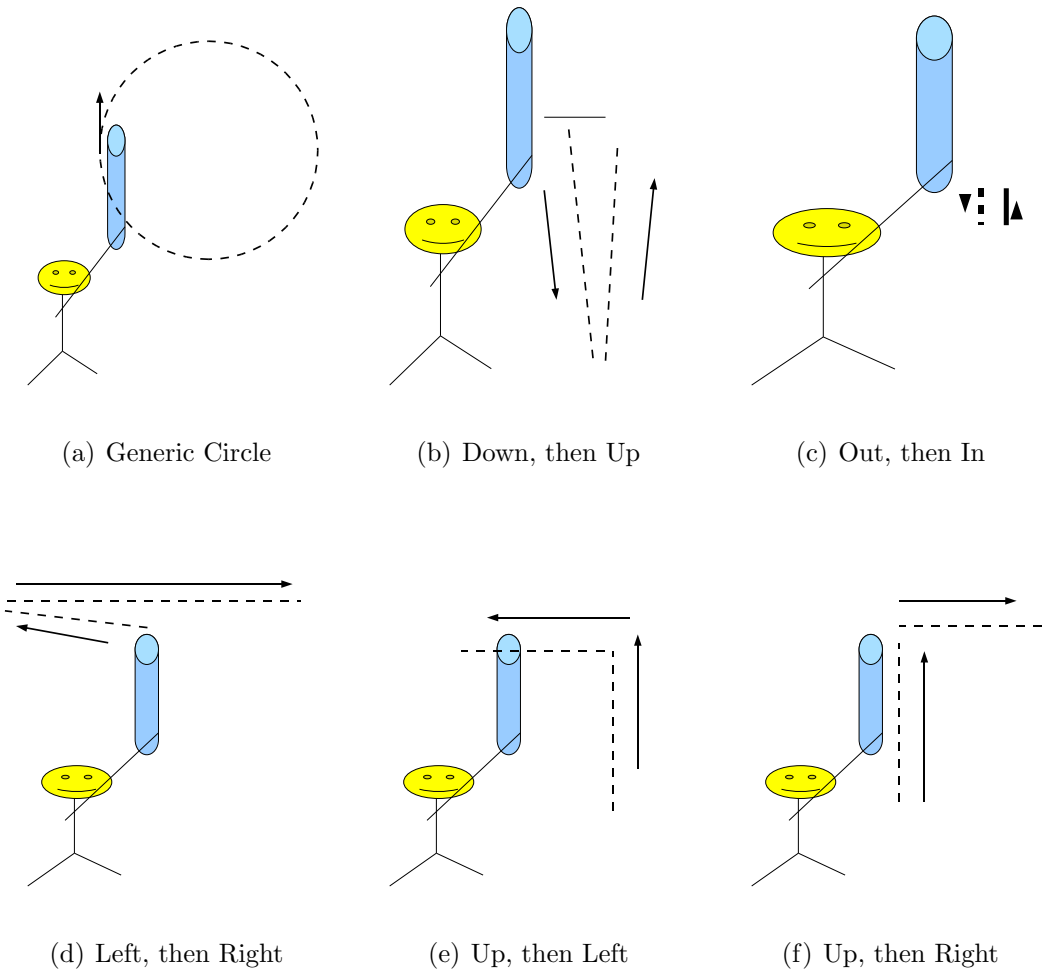
- The wand has knowledge of the commands
- The wand must be taught the gestures

- The trained gestures need to be associated with commands

Some general system issues would need to be addressed as the receiving device would have knowledge of the state of the gesture, the wand would need to process the gesture and then send out the appropriate command. The training issue occurs where user inconsistencies develop and trained gestures could end up as illegal gestures. Users might also get things wrong regarding when to start and stop a gesture. The user might select gestures that are too ambiguous and recognition system could fail to understand the unique gestures from each other. Measures need to be put in place to force the user to perform a valid and unique gesture. However, if the gesture requirements are that they should be unique, easy to perform, and easy to remember, and the system in place for training makes sure all these criteria are met, there will be a good argument to implement generic gestures.

3.3.2 Generic Gestures

Generic [7] gesturing, in the context of TMW are intuitive, simple, unique and easy to remember for any end user with just basic training. Generic gestures should be trained by a sample of people, representing the many ways a gesture could be interpreted by a user. Obviously, users should be guided to understand the scale and timeliness of the gesture. The gestures of TMW should be neither too complicated, nor too numerous. People remember 7, plus or minus 2 [22], so a number of generic gestures between 5 and 9 are reasonable to assume. The assumption is that fewer and more distinct gestures are easier to remember, however this will force an operational constraint for operations on ubiquitous devices. Some tested generic gestures are: Down, then Up, figure 3.3(a); Circle, figure 3.3(b); Out, then In, figure 3.3(c); Left, then Right, figure 3.3(d); Up, then Left, figure 3.3(e); Up, then Right, in figure 3.3(f);



(a) Generic Circle (b) Down, then Up (c) Out, then In

(d) Left, then Right (e) Up, then Left (f) Up, then Right

Fig. 3.3: Generic Gestures

3.3.3 Gestures and Interaction

With the assumption that 5 or 6 gestures are the maximum an end user would be comfortable with, the ubiquitous environment would also be limited to performing 5 or 6 actions. These devices would need to be context aware, aiding TMW in enabling users with a total intuitive performance. The ubiquitous device has knowledge of commands and expects those commands from TMW, but depending on the context the commands could mean different things. If we look at a device, in this instance an MP3 player, the players functions are, load playlist, select playlist, select songs, add to playlist, create playlist, play, stop, pause, fast forward, fast rewind, skip to next song, skip to the last song, power on and power off. A washing machine's functions are, open door, select Celsius degrees (30, 40...90), select coloured, white or wool, select spinning cycle (50,100...300), and power on. To operate these machines with 6 gestures could possibly be quite cumbersome if applied directly without altering how the devices operate. As it stands now, to operate the washing machine, it would be gesture 1 to open the door, 2 to select category Celsius degrees, gesture 3 to select clothing type, gesture 4 to select spinning cycle category and gesture 5 to start the washing machine. Then gesture 1 to 6 selects different settings within the 3 categories. To operate a washing machine this way would take at least 8 gestures, 9 if we need to abort the operation. Clearly, operating anything with gestures needs a complete rethink on how gesture commands perform on these devices. The author will not try to answer how ubiquitous devices should perform, but will outline how to enable an ubiquitous element to everyday devices.

3.3.4 Enabling Devices for an Ubiquitous Environment

We need to simplify the user interactions to enable devices for TMW. The methods of simplification are menu's [24], context menus, [27] and context awareness [18]. When interacting with a device the device should provide feedback on how well the user is doing, this could be through GUI, voice or sounds. The context awareness with some form of learning should take care of when to turn the music off, standby, genre of music, favorites etc. Ubiquitous devices are examplified by an MP3 player, device operations could narrow down behaviour of the mp3 player, to play, pause, play next and play previous track. The washing machine example could be narrowed down to 1 gesture, whereby sensors know what type of clothes have been loaded, finds the right temperature etc and starts when the door closes. The only reason to use a gesture in this example could be to abort the cycle. Artificial Context Intelligence could detect user profile and anticipate what actions to make. On a simpler level, the play music command by gesture could mean different things depending on what the previous action was or previous interaction has been over time, letting the device learn preferences.

3.4 Gesture Recognition Techniques

For this dissertation the authors aim is not to come up with a new middleware for gesture recognition but to describe the different techniques considered and feature availability of a good open source library for implementation in this project.

3.4.1 Bayesian Networks (BN)

No relevant libraries for gesture recognition exists for Bayesian Networks [15], but the methods are well known. A BN covers variables that are called nodes, these nodes

are linked in a network with parent and children relationships. Each of these variables has a conditional probability distribution $P(Xi|Parents(Xi))$ for $i = 1$ to n If X_i The Bayesian network can be said to be representing the conditional independence relationships in the system being modeled.

3.4.2 Support Vector Machines (SVM)

The author found no relevant open source library in gesture recognition or equivalent for Support Vector Machines [10]. The method requires to scale the data to apply values within a uniform scale. The method for data classification embodies training prototype with features and one target value, and classification is performed by a model predicting the target value for the training sets. To classify data, the user must choose between the kernels or models available, linear, polynomial, radial basis function, and sigmoid.

3.4.3 Hidden Markov Model (HMM)

The author found the HTK [26] and GHMM open source library [5] for general use. HMM is another statistical pattern recognition, that returns a probabilistic value for a given sequence of observations given a specific HMM. To answer an HMM [21], there are three problems of interest.

- Problem 1: Given the observation sequence $O = O_1, O_2..O_T$ and model $\lambda = (A, B, \pi)$, how would we compute $P(O|\lambda)$?
- Problem 2: Given observation sequence $O = O_1, O_2..O_T$ and model λ, how do we choose corresponding state sequence Q that best explains Observations O?

- Problem 3: How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$ the probability of the observation sequence?

Describing the Elements in HMM

- The model is $\lambda = (A, B, \pi)$
- $a_{i,j} = P[q_t + 1 = S_j | q_t = S_i], 1 \leq i, j \leq N$ which is the state transition probability distribution, where $A = a_{i,j}$
- $b_j(k) = P[v_k att | q_t = S_j], 1 \leq j \leq N, i \leq k \leq M$ where the observation symbol probability distribution in state j, $B = (b_j(k))$
- Initial state distribution $\pi = \pi_j = P[q_1 = S_i], 1 \leq i \leq N$

Solution to problem 1, 2 and 3

- Problem 1: Forward-Backward Algorithm, calculates the probability of the partial observation sequence until observation O_t finishes at O_T. This gives calculation of $\alpha_T(i)$, which is $P(O|\lambda)$. Problem 1, solves recognition probability of a sequence given a model.
- Problem 2: Optimal states are found by some criteria the developer or user decides on. Viterbi Algorithm is considered the best practice to find the best state sequence given an observation sequence. The state sequence is used to train new models or to return a probability with a given model.
- Problem 3: Baum-Welch algorithm is utilized to train prototypes in statistical patterns. It works by maximizing the probability of state sequence until a critical point is reached. $\frac{\max}{\lambda}[Q(\lambda, \overline{\lambda})] \Rightarrow P(O|\overline{\lambda} \geq P(O|\lambda)$

HMM overview To utilize the HMM we need Observations over time T, calculated to a state sequence S_T that can be either trained with the Baum-Welch algorithm or the state sequence can be used to return a probability for the observations given a model.

3.4.4 Concluding Gesture Classification and Recognition

The gestures made by TMW will be limited to 6 to cover most situations, with assumptions that the ubiquitous devices are enabled for TMW communication. Classification and recognition of these gestures would benefit most from a library using the Hidden Markov Model, due to the freely available libraries and continuing success rate.

3.5 TMW Communication

Device wireless transmission is necessary for TMW's criteria of autonomous usage by its independent form factor.

3.5.1 Traditional IP Transfers

With IP transfers, TMW could connect by ad-hoc network or connect in an expected IP range, and by multi-casting finding the appropriate available devices. When these devices are found they will need to reply their IP, then identify where they are according to their IP address, some kind of activation flag will be necessary to implement when TMW requests interaction mode.

UDP As UDP [19] is quite well known and documented the Author will not delve into the technical specifications. The pro's for using this protocol with TMW are the

send and forget attribute and eliminating the latency of acknowledgement messages. However, we need to answer how an acknowledgement for a successful transmitted command could be resolved, like response commands, light or sound.

TCP The well known TCP protocol [20] has the drawback in TMW, style ad-hoc connectivity that the client on the wand side needs to wait for acknowledgements before it can terminate the opened port. This protocol is not recommended.

3.5.2 Bluetooth

Bluetooth [1] has been successfully implemented in an array of different devices and in 2006 is still considered de-facto standard for lightweight mobile wireless communications. The biggest problems for the Bluetooth protocol is the time lag it suffers from when searching for new devices to actual device to device communication, from 13 seconds and upwards. The second issue is limitations on how many slaves a Magic Wand master can have. With capability to support 7 devices where connection time lag is 13 seconds, is considerably limiting the ubiquitous element of interaction with devices. On the capability side the protocol can be trimmed to handle specific wand and device communication faster and lighter.

3.5.3 Infra red

A similar concept to a remote control, with transmission capability of 16 mbps. The IrDa protocol [13] is directional and therefore contradicts TMW non-directional gesture application.

3.5.4 ZigBee

ZigBee [12] is an ad-hoc network protocol with attributes like low power consumption, great ad-hoc capabilities and low cost. The main reason to investigate ZigBee is its ad-hoc routing capability for networking if needed, it also features low latency due to direct sequence spread spectrum [23]. ZigBee is simpler than Bluetooth, has lower cost, and is more energy efficient, it allows for only 256 Kbps compared to Bluetooths 1Mbps, but features support for 254 nodes.

3.5.5 Concluding on TMW Communication

The design for TMW should allow for it to activate a device it points to or is in proximity with, then a gesture command should be sent out to the correct ubiquitous device. To meet these criteria the user should select the device of interest, this device could be activated by infrared, RFID [6], hand touch or Bluetooth. Device activation should ideally include identification of the wand or user or both allowing authorised commands only. The commands could be transmitted by ZigBee, Bluetooth, or UDP, in ad-hoc mode.

3.6 Magic Wand Architecture

TMW design is divided into layers, figure 3.4, where sensor consists of calibrated acceleration data from the MTx, Inertial Measurement Unit. In the sensor fusion layer, the sensor data is scaled, then applied to the Hidden Markov Model for training or recognition. For the application layer the gesture models are utilised for TMW communication. The last layer is communication where the gesture command is sent to the right device.

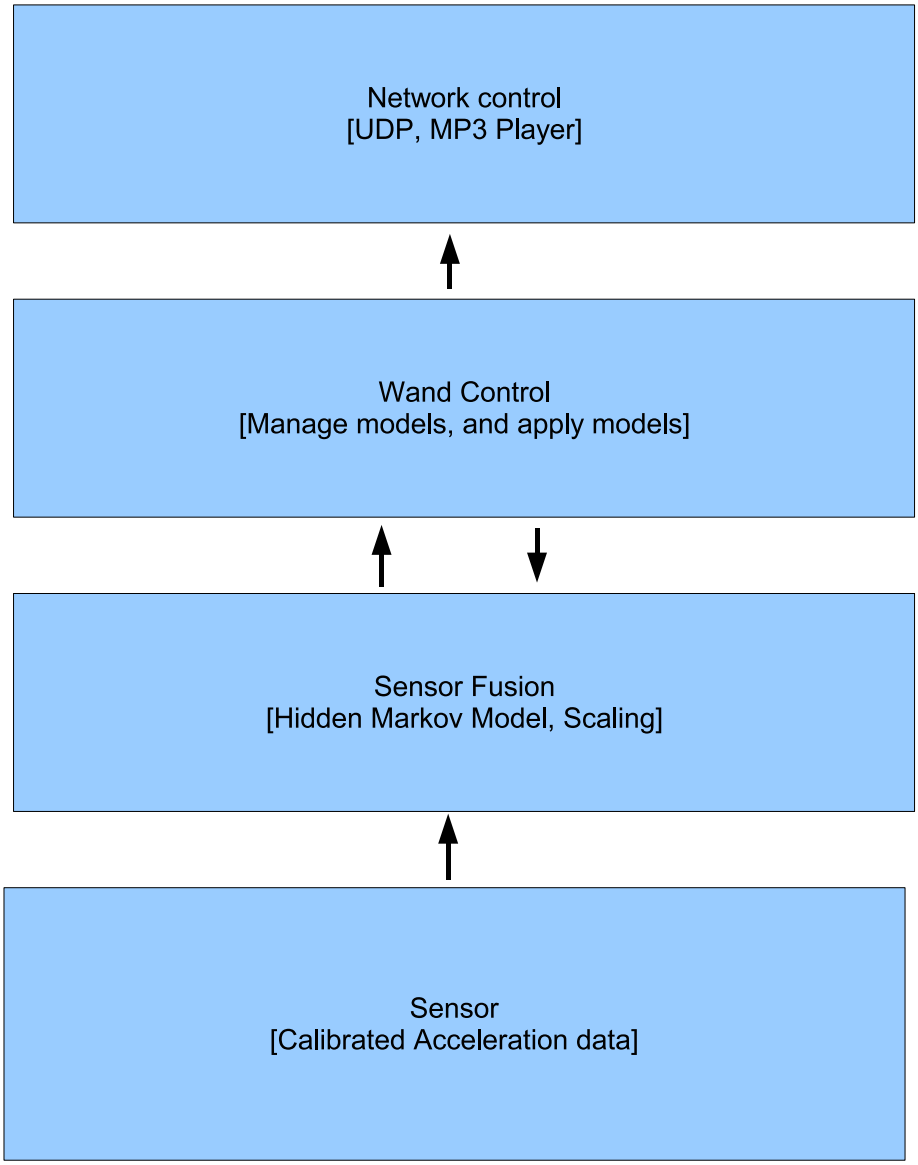


Fig. 3.4: Layers of TMW

3.6.1 Use Cases

When the user makes a choice from the initial menu in the system, depicted in figure 3.5, making a new gesture to re-training a current gesture, they must have knowledge of gestures, their names and how to perform them in order to not get out of the gesture scale. The first gesture sets the standard for gestures and will not get scale errors. The recognition mode takes in continuous gestures where each gesture is unique within a frame of time. It then returns a model with the highest probability, where it transmits the probable command UDP protocol to an expecting device server.

3.6.2 Sequence Diagrams

These sequences in figures 3.10, 3.11, 3.12, 3.13, 3.14 describe the flow of each of the use cases in figure 3.5 and 3.7. The first sequence diagram 3.10, where the MP3 player listens for incoming commands, and TMW waits to send commands until a valid gesture has been found. The second sequence 3.11, where user creates a new gesture, by first performing the gesture, then select the appropriate choice in the menu, and entering in the gesture name. The third sequence diagram 3.12, depicts where users re-estimate an already stored gesture, called Markov model. The user can perform a gesture, then choose the appropriate option in the menu. Here a model must be loaded and be re-estimated with the new gesture data. The fourth sequence diagram 3.13, depicts how the sensor fusion life cycle works. Firstly after sensor data has been read, it's being scaled to three symbols, then the observation set is stored to memory, then assigned to a Markov sequence and finally memory is reset. The fifth sequence diagram, depicts semi continuous gesture recognition. The life cycle starts when a gesture is performed and a flag is turned on by the user to enable semi continuous recognition, where it

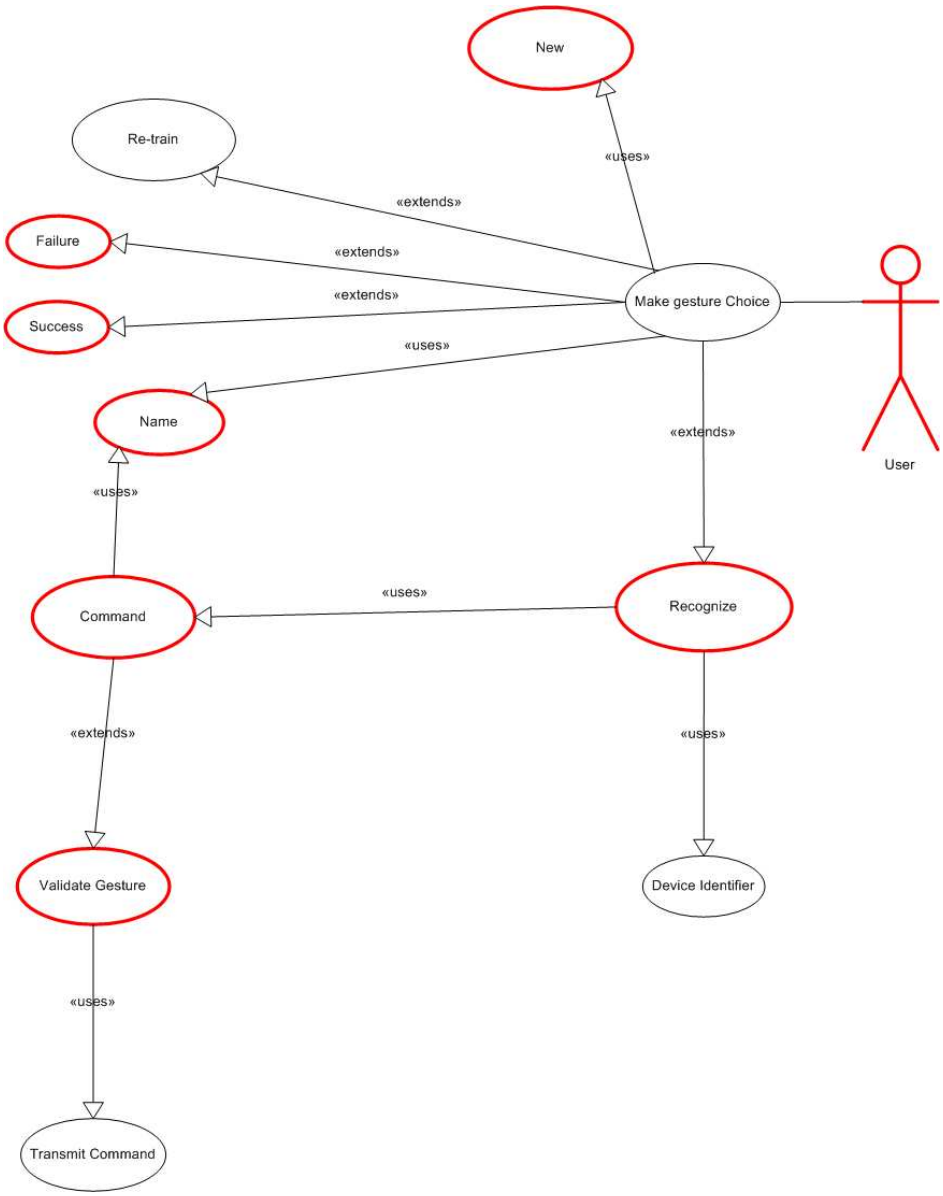


Fig. 3.5: TMW, use cases for application

Use Case Name	Device Connection	
Actors	Magic wand & MP3 player	
Description	MP3 player listen for incoming commands	
Typical Course of Events	MP3 player	The Magic Wand
	#1: Listens for commands on UDP port 5000	#2: Knows the ip address of the mp3 player
Alternative courses of events		
Pre-Conditions	There exists model's	
Post-conditions		
Assumptions	The Magic Wand knows the IP to the MP3 player	

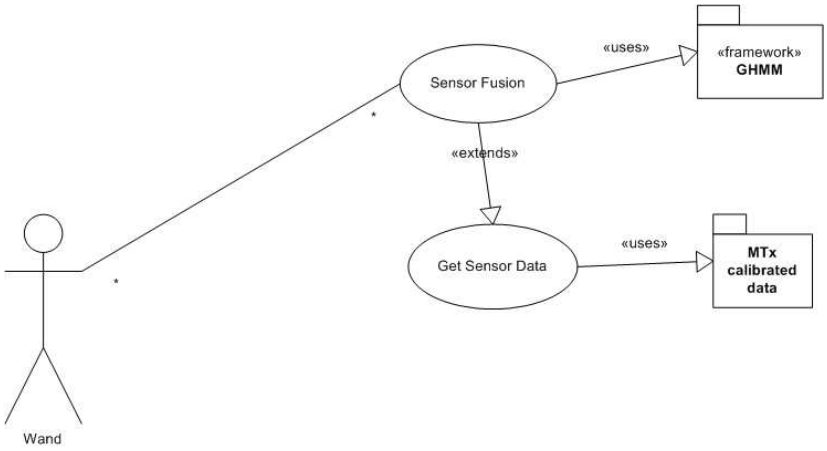
Fig. 3.6: Use Case narrative, Device connection

Use Case Name	New Gesture &, or Train Gesture	
Actors	The Magic Wand & User	
Description	User creates a new gesture or train a current gesture	
Typical Course of Events	User	The Magic Wand
	#1: User performs a gesture #2: Selects to load a model or new model	#3: Stores the model with .hmm extension
Alternative courses of events	Loading a model, and use the data to reestimate it	
Pre-Conditions	NA	
Post-conditions	NA	
Assumptions	User should know the model names	

Fig. 3.7: Use Case narrative, Training or New model

Use Case Name	Magic Wand communicates with a device (Ubiquitous environment)	
Actors	Magic Wand, MP3 player	
Description	The Magic Wand Send a command to the mp3 player	
Typical Course of Events	Wand	MP3 player
	#1: Retrieves all models, and finds the most likely gesture based on the gesture taken by the user #2: Wand return the filename which dual purpose is the command #3: File name is transmitted by UDP on a ip address on port 5000	#3: Device find out what the file name (command) means on its device and performs the action matching the command.
Alternative courses of events	Gesture not found, wrong gesture found	
Pre-Conditions	User knows the gesture and makes that gesture	
Post-conditions	MP3 player plays a song	
Assumptions	User tries to perform the gesture correctly	

Fig. 3.8: Use Case narrative, interact with the ubiquitous environment



(a) TMW, use cases for sensor fusion

Use Case Name	Sensor Fusion	
Actors	Magic wand	
Description	The Magic Wand gets data from the sensor and applies it to sensor fusion	
Typical Course of Events	The Magic Wand	NA
	#1: Scale sensing data #2: Store sensing data to memory #3: Adding sensing data to a Markov statistical sequence #4: Reset values	NA
Alternative courses of events		
Pre-Conditions	Sensor reading only if a threshold for acceleration data is broken	
Post-conditions	We have a markov sequence that can be used to train or recognize models	
Assumptions	NA	

(b) TMW, narratives for sensor fusion

Fig. 3.9: Use case narratives

starts out by loading all of the gesture models and return the model with the highest probability. This Markov model name is transmitted to an IP address.

3.6.3 Class Diagram

The classes outlined here are the classes on TMW side, excluding the GHMM [5] library and the MTComm [29] library.

3.7 Design Summary

This chapter is answering capabilities and limitations in techniques on each implementation layer of TMW. In the lowest layer, the sensor data is calibrated. The sensor fusion layer implement sensor data scaling, then a technique will classify and recognize statistical patterns, preferably in near real time. The technique most popular by other gesture recognition has been HMM. For the application the author will need to make use of the gestures taken, recognise and train for an actual new tangible user interface. The communication outlined will depend on available hardware for the wireless platform.

Fig. 3.10: Sequence diagram, alignment of wand and device

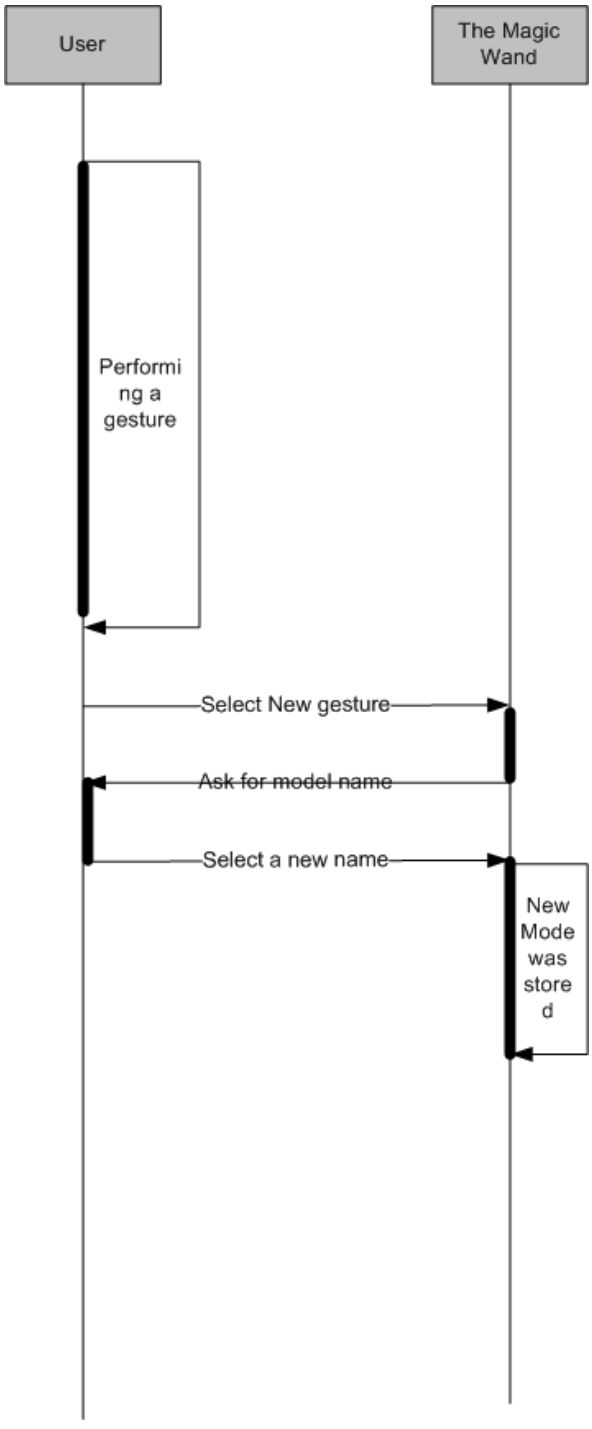


Fig. 3.11: Sequence diagram, user create new gesture

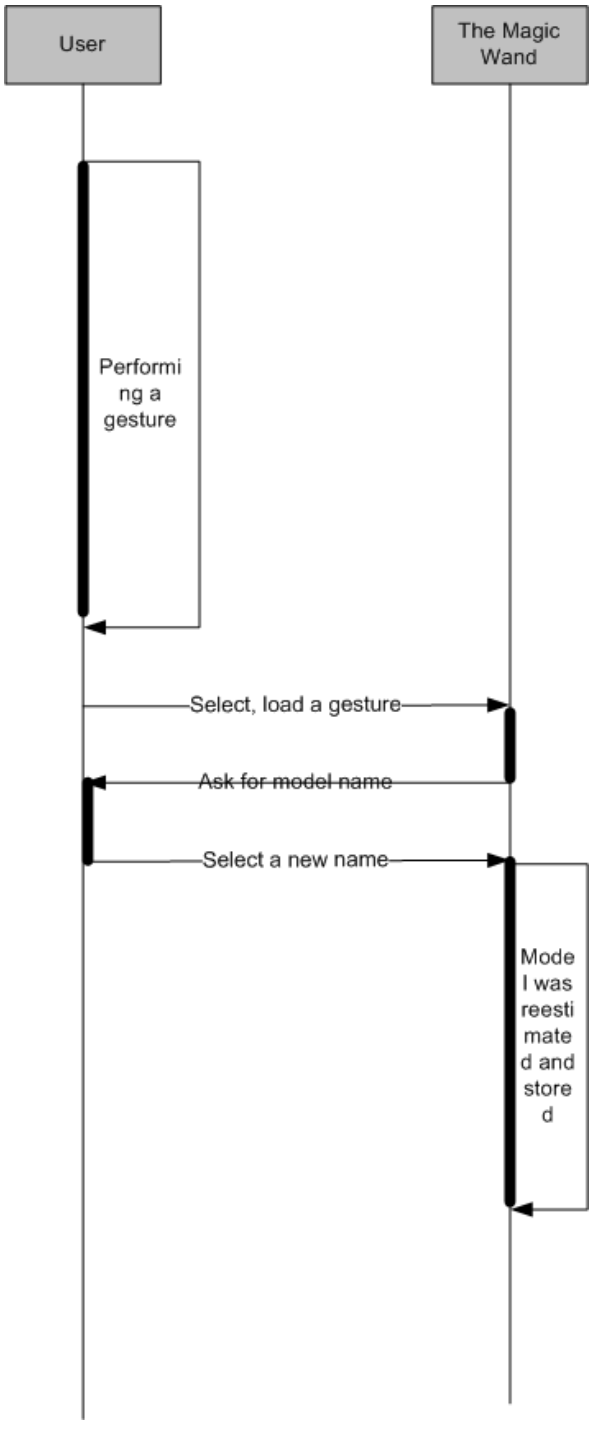


Fig. 3.12: Sequence diagram, user reestimate an existing model

Fig. 3.13: Sequence diagram, sensor Fusion sequence

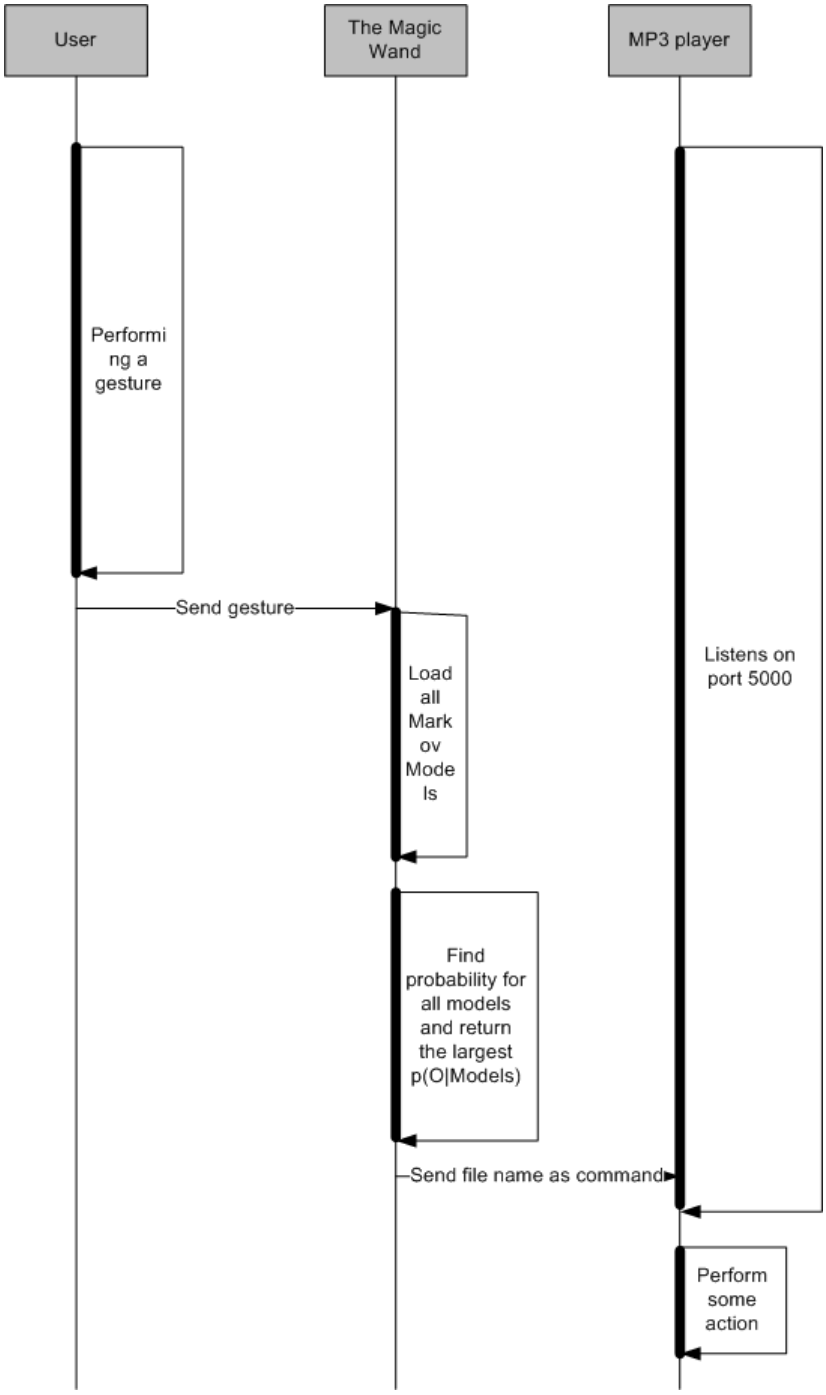


Fig. 3.14: Sequence diagram, communication, wand to device

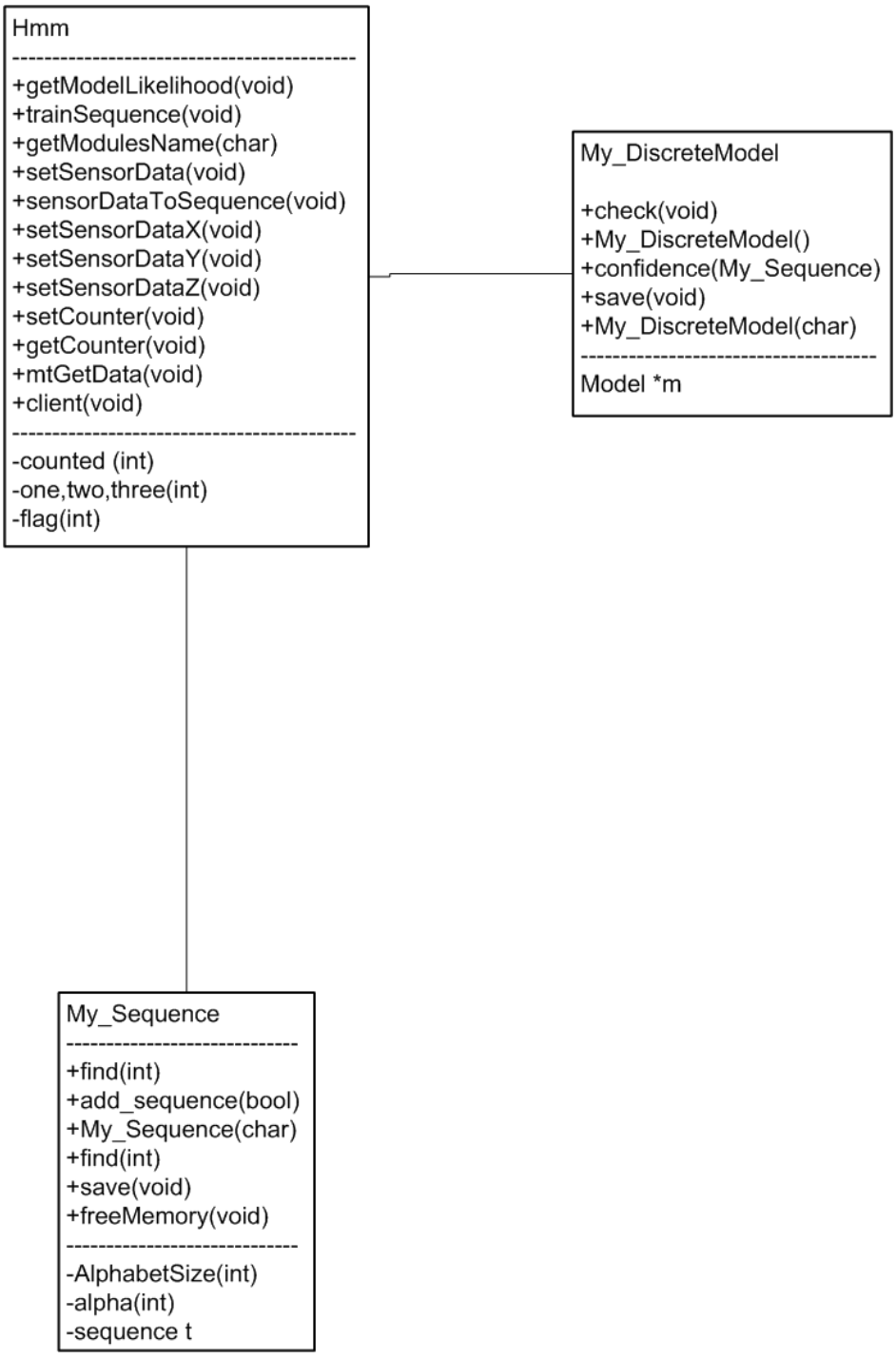


Fig. 3.15: TMW, class diagram

Chapter 4

Implementation

Implementation was carried out by dividing the project into four layers, Sensor Data, Sensor Fusion, Application, and Communication.

4.1 Overview of TMW

TMW is drawn up to be an autonomous device, seen in figure 4.1, with wireless only access to and from the ubiquitous-enabled devices. The battery, a computer, sensor, and a wireless card, depicted in 4.2 need t be on the same wand. The only suitable wand like shape which will fit these components is a tubular shape, components can be fitted and protected within the tube. A cardboard tube, made for posters, was finally chosen as seen in figure and painted gold, figure 3.1(b).

4.1.1 Inertial Measurement Unit (IMU)

An IMU from Xsens, called MT, seen in figure 4.3, was ordered by the college and chosen for this implementation for its quality of calibrated output data. The MT

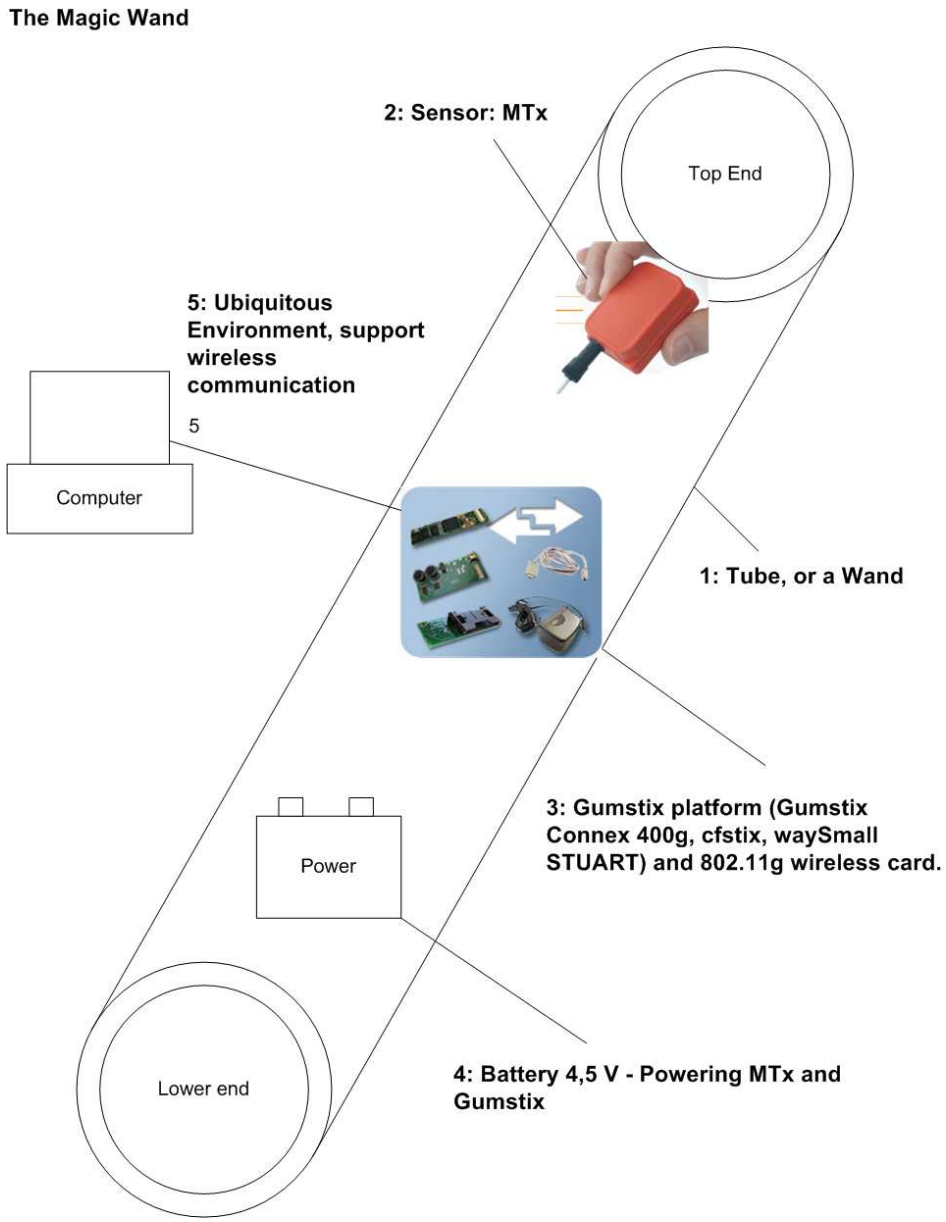


Fig. 4.1: TMW's components

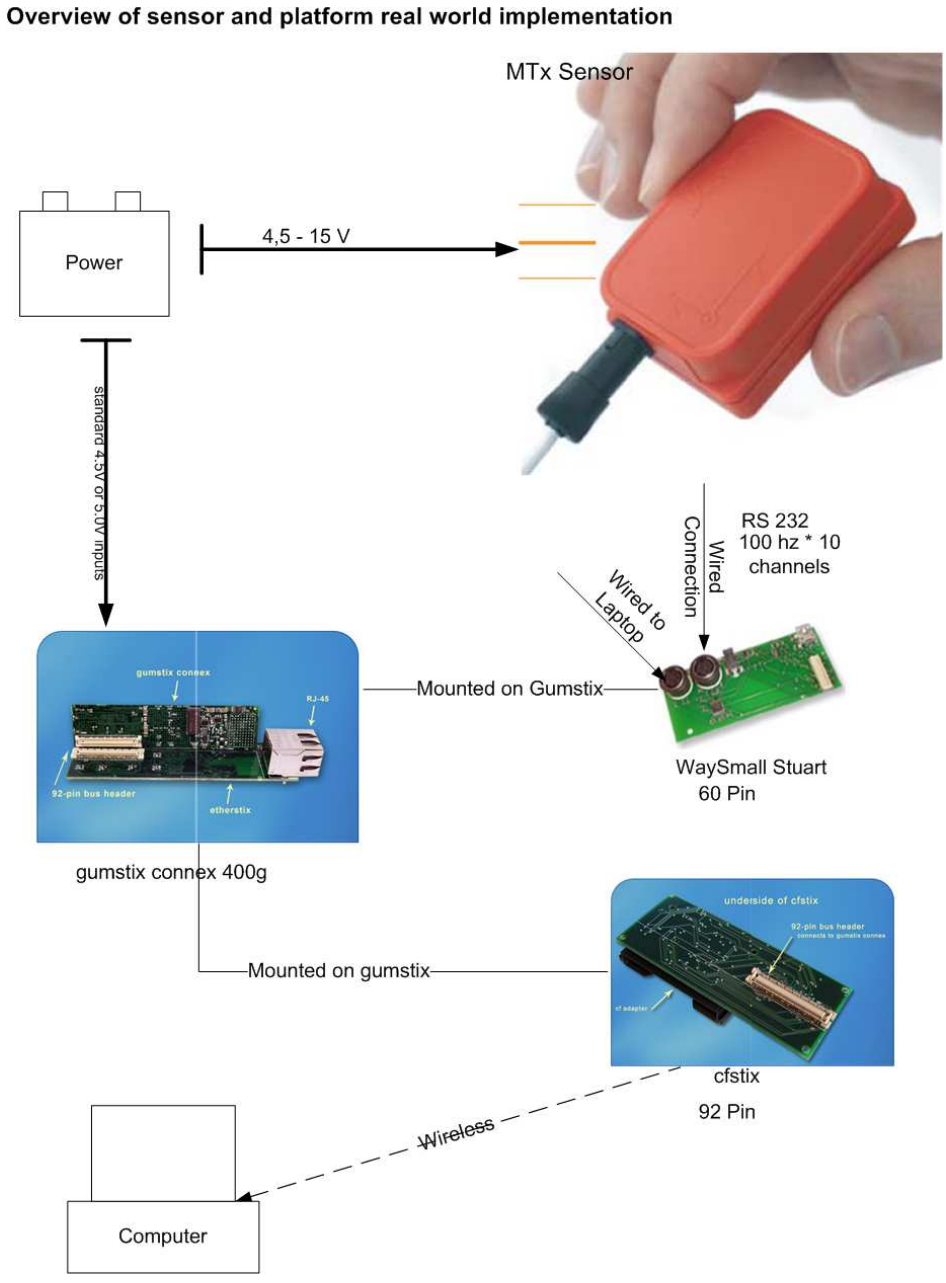


Fig. 4.2: TMW's connected components

sensor returns accelerated, gyroscopic, magnetic and gravity data.

MTx Sensor This is Xsens newest human motion tracking IMU, it's shipped as a standalone sensor module with the sise of a matchbox. The MTx comes with its own library to interface with the sensing data, as it is IP of Xsens and not open source. The C++ compiled library was the only means to return sensor data.

MT9 Sensor This obsolete Xsens sensor has a simpler library, with less sophisticated configuration alternatives available.

4.1.2 Platform Hardware and Software

To meet requirements of autonomy and independence from wires, a small computer was essential. The choice was between a PDA and the Gumstix, figured in 4.4. The Gumstix was the obvious choice, with low cost, less than (200$), small composite and features Tiny Linux [16] with support for wireless, serial ports, sound, Bluetooth with much more. The capability for Gumstix was the CPU card featuring a 400 Mhz ARM CPU and a USB port, 2 additional cards could be placed on its pin connectors. The authors choice for the additional cards was the serial card and the flash adapter, which is compatible with a wireless card (due to the fact that Bluetooth card was out of stock) and the serial was necessary for the serial-only interfaced MTx sensor, and wireless communication to meet the autonomous TMW requirement.

x86 Preliminary testing and implementation were done on a Pentium 2, 800 Mhz, 256 MB ram with serial interface for the sensor.

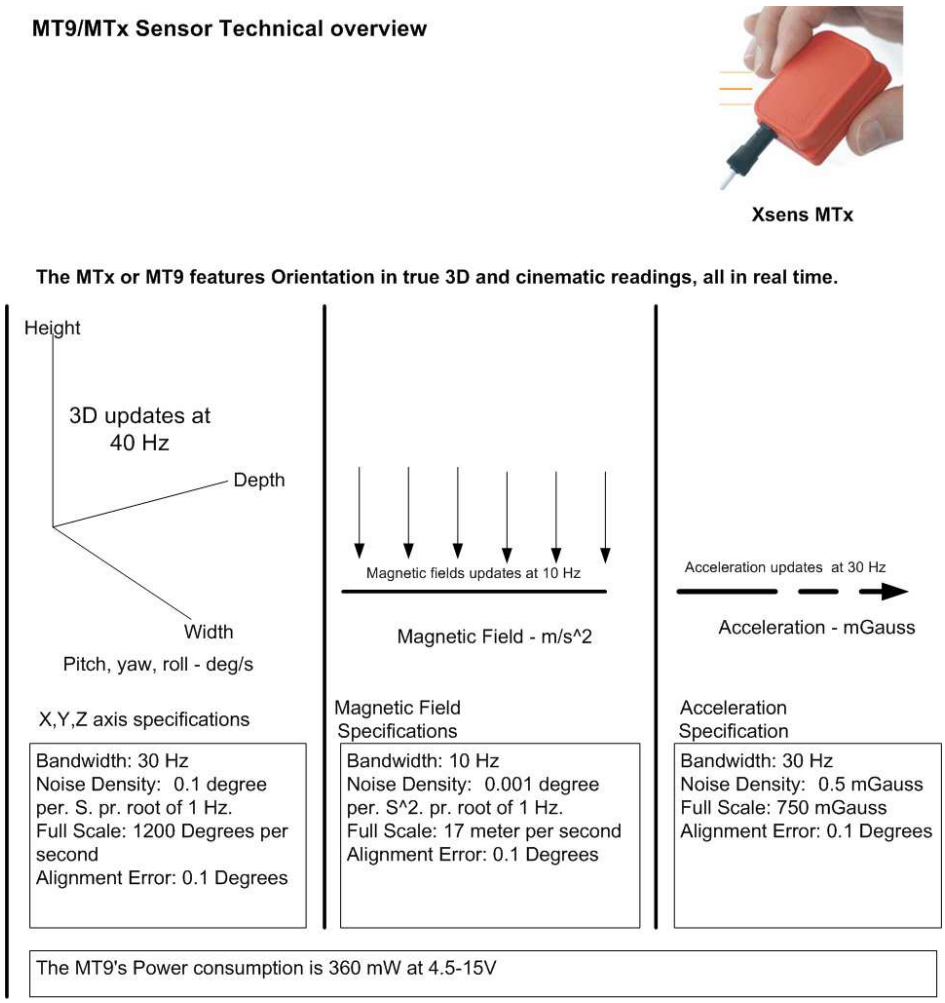
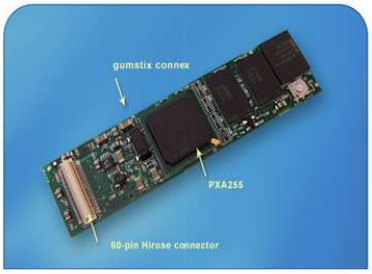


Fig. 4.3: MTx and MT9 Sensor Specifications

Gumstix Platform

The Gumstix Connex platform



Technical Specifications:

CPU:	Intel XScale® PXA255 400MHz
Memory:	64MB SDRAM 4 MB Strataflash
Storage:	via Type II compact flash on cfstix
O/S:	Linux kernel is 2.6.11
Dev. Tools:	gcc 3.4
Connectors:	60-pin Hirose I/O header and 92-pin bus header
Wireless:	Yes, through the cfstix board
Power:	standard 4.5V
Dimensions:	80mm x 20mm x 6.3mm

The CFSTIX device

Technical Specifications:

Interface:	Type II Compact Flash adapter
Power:	3.5-5V
Connectors:	via 92-pin bus header

WaySmall STUART

Technical Specifications:

Interface:	20 gpio and other signals
Power:	3.5V - 5V
RS232:	FF-UART and ST-UART available on two RS232 serial ports
Connector:	60-pin Hirose connector

Fig. 4.4: TMW's Computational Platform

ARM Final implementation was compiled for ARM, and installed to operate totally autonomous.

GHMM and the HTK Library The rudimentary GHMM library was finally selected by default after the HTK library proved dependent on the X window to install on Linux. Our Tiny Linux was incompatible with the X window and the GHMM library was then chosen for implementation.

4.1.3 Communication Device and Protocol

The wireless device the author purchased was the D-Link 802.11b Wireless Compact Flash Adapter, which at the time was the only wlan card that would be available within a short time, at a low cost. For a future implementation TMW would connect to the devices within close proximity where ideally a Bluetooth device could be most beneficial. The author chose UDP for the 802.11b device. A future transfer to Bluetooth would be a simple process, because the application sends a command to whichever device is nearby. It is expected the device in question should be enabled for the generic commands of TMW.

4.1.4 The Device, Representing the Environment

The ubiquitous enabled device was implemented as a MP3 player called XMMS, which features a C++ compiled control header. A UDP server was also developed to greet commands by TMW and execute upon them by interfacing with the control header.

4.2 Sensor Layer

The data layer covers sensor data. The MTx delivers fully calibrated gyroscopic data, acceleration data, magnetised data, and temperature. For TMW project we would need acceleration data. Acceleration data is delivered in X, Y and Z axis and only delivers data when a motion in space is performed, for this reason, acceleration data was the reasonable choice for reading gestures.

4.2.1 MTComm Library

The MTComm library for the MTx Inertial Measurement Measurement that allows for configuring the data output.

Capabilities The configuration is set to 1 sensor, 40Hz, acceleration data, where at startup the alignment is reset.

Limitations When attempting a sensor reset with MTx firmware 1.09 which failed and an upgrade to 1.11 was necessary for the reset code to work.

4.2.2 Challenges at Implementation

The limited accessability for the sensor created a situation where the author had to make a sensor simulation of the data output. When the author had access to the MT9 sensor for a few days, the calibrated data log was stored for 3 different gestures. These log files where then used to simulate the sensor data for the sensor fusion techniques. When the MTx came back from repairs after it short-circuited, the author could then set the Hz of the readings and implement a semi continuous sensor reading.

4.3 Sensor Fusion

The sensor data varies between ±10 G's [25], displaying the data as X, Y and Z with 8 decimals. As there are some limitation set on the Generalised Hidden Markov model library [5] as discrete data, rounding up of data to nearest integer was implemented. To solve how users might perform the gesture fast, slow etc. scaling was implemented.

4.3.1 Generalised Hidden Markov Model Library (GHMM)

The library is coded in C, with obsolete C++ wrappers, but features support for PHP on all new versions. As the hardware platform was a limited device, computational efficiency was a higher priority than functionality. The library core is made in C, featuring classes and methods for the various hidden Markov functions, like re-estimating, forward-backward algorithm, models and sequences. A C++ wrapper was developed to append observations to sequences, training of models, re-estimating thos models and recognising models.

Additonal development Functions that were developed allowed saving and loading of models to be used at any time. The sensor data was scaled to three symbols with the same amount of states, which actually eliminates the hidden states, to simplify processing. An issue that remained were that sensor data returned X, Y and Z value per time t, and the GHMM library only accepts one value at time t. Instead of calculating a mean average, the choice was to insert all the sensor values, in effect multiplying the data per time t, by three. Since the HMM is using distribution, the values that actually change are the ones that are being weighted, and hence better results were obtained than averaging techniques that compromised gesture accuracy.

Capabilities The core C library allows for the developer to utilise the HMM algorithms. The PHP feature functions to read, store, write to screen or files. It simplifies a lot of the practical processes in treating data and files.

Limitations For the core C library, the developer has to create all support facilities. The GHMM library is limited to read in discrete numbers, one value at the time.

4.3.2 Concluding implementation

In testing, the memory and CPU footprint for the sensor fusion was almost non-detectable, and would most likely allow for a higher level programming/scripting language like PHP to make use of the continuous models function, for an improved application layer. When scaling was implemented, a lot of the gesture accuracy got lost, but it kept the observation data simple and manageable. The author investigated the X, Y and Z values per time t, by averaging the values, but it proved difficult to return gestures that would actually be distinctive. This can be thought about, as data for a circle looks and has the same distribution as a square or and oval, and this can happen with more or les any gesture. There are plans to make the library support multiple values at time t, this is an ultimate requirement for high quality gesture recognition.

4.4 Application Layer

The observations are made into Markov Sequences, and then added to models or recognition functions, at application layer this data must be made into a viable program that meets the requirements in autonomous user interface. The observations are read into time segments of 2.4 seconds which is enough time for most gestures to finish,

then the data is made into a Markov sequence ready to be used. An initial menu sets TMW in either recognition mode or training mode, where a trained person can make new gestures or train pre-defined gestures to make its models better. The recognition is semi continuous, it loads the observation data into a sequence that is being measured against every model, this returns a log probability value where the highest value is the most likely gesture. This model also works as a command that will be transmitted to the MP3 player when successfully identified.

4.4.1 Autonomous Usage in a Tangible User Interface

By definition, autonomy in the context of TMW means independence from a graphical user interface and wires. The user's will however need some verification that the gesture was performed and performed correctly. The gesture verification was implemented on a remote screen where the MP3 player was set up.

4.4.2 Concluding Implementation

Observations are read into a time slot, then recognised by comparing Markov models and finally a command is returned. The nature of the implementation means it has a time slot where the gestures have to conform to rigid start and end times. The MP3 player would need to aid the user with those start and end time slot for performing gestures.

4.5 Communication Layer

A UDP server on port 2000 was set up to interact with the XMMS developer control head. Three gestures were recognised with over 80% positive recognised gestures. Down

and up gesture plays the track, left and right pauses, and then out and in skips the track.

4.6 Programming Methods and Tools Used

The author had no experience in development for Linux, but it was necessary to develop in Linux to accommodate the Gumstix implementation. C++ was the chosen language as the author wanted to learn and improve skills of this language. The MT-Comm, XMMS was C++ and The GHMM library was a C library, a C++ wrapper was developed where needed, to implement all layers of the implementation in a common language. Another learning curve was integrating user made classes with already existing libraries in a correct manner. For the physical implementation the application for all levels had to be compiled for ARM, transferred to the Gumstix platform, and run on the Gumstix. The success of the implementation was obvious when gestures were made by TMW and the gestured commands were sent over the wireless link to start the mp3 player, pause it, skip to next song and do nothing when a bad gesture was performed.

4.6.1 Hardware Implementation

The design was created to be fitted in a limited device, Gumstix. The allocated memory was 64 MB, and hard drive was a meager 4 MB, however we extended this space by NFS mounted drive. The wireless card was set up to start at boot up, connect to NFS, and then by SSH remote login, the author could run the software on TMW. The MTx sensor worked very well, but some issues were discovered with the reset alignment code that were approved by Xsens, it did not work before a factory reset were made for every

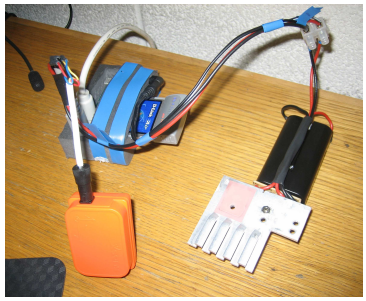


Fig. 4.5: The components of TMW

time it was used in a different project. The Gumstix gave the author some challenges, to build the kernel on a host, then uploading the image to Gumstix, configure the Gumstix and the wireless card. The Gumstix in itself has a small physical form, but with the battery, the serial card, flash adapter card, the wireless interface and the battery, the Gumstix became quite heavy with a substantial form factor, in figure 4.5.

4.7 Problems and Issues During Implementation

The MTx sensor arrived in the middle of July, so a few weeks were spent testing different ways of getting the data at a speed comfortable for the Gumstix to process in conjunction with the HMM library. However, the MTx sensor was short circuited in a different project team after only a few days and a backup sensor, the Xsens MT9, was borrowed from another project, for short periods as this sensor performs in similar ways. The library for both the MTx and the MT9 are very different and both were developed so they could perform equally with the same tasks.

Chapter 5

Evaluation

In post implementation, TMW project featured wireless, autonomous user interface, and communication with a remote MP3 application. This evaluation will answer if the project met requirements set for the project, and if the project answered the expectations of TMW as the extension for our will on the ubiquitous environment.

Requirements were:

- Intuitive use
- Autonomy
- Performance
- Visual or audio feedback

5.1 Architecture Evaluation

The architectural points for this project that weighted down, was that it ran in one process in cycles, the opening and closing of the MTx port every 3 seconds caused the port to fail within minutes. A better solution would have been to return the observation data continuously in a thread. The opening and closing of the port also opted for a sleep function to wait 0.5 seconds for the port to initialise properly, otherwise the port would fail within the first cycles.

5.2 Quantitative & Qualitative Evaluation

For user evaluation we brought in 10 people from various backgrounds. All test subjects were given a short introduction and demonstration of the unique gestures, then they were allowed to practice each gesture twice. For the test, the author asked test subjects to perform each one of the gestures when a visual aid would tell the user to start.

User Interaction - Qualitative Report In this part of the evaluation the author was evaluating, seen in table 5.1, the users as they performed the gestures, circle, down-up, left-right, out-in.

Table 5.1: Evaluation of the end users performance

Question	*Answers*
Does the subject know how to use gestures	7/Yes
Has the subject performed gesture before?	10/Yes
Getting the gestures one by one from memory?	10/Yes
Returning correct gesture from memory?	6/Yes

User Interaction - Quantitative Report The test subjects all saw past the prototype form sise and had an idea of the concept, they all thought integration to their mobile phone was sensible as they brought it with them everywhere figured in table 5.2.

Table 5.2: End user Questionnaire

Question	*Answers*
Would you like to use this concept regularly [yes/no]?	10/Yes
I found the system unnecessarily complex [yes/no]?	9/Yes
The system was easy to use [yes/no]?	10/Yes
I would need the support of a technical person to use this system [yes/no]?	10/No
I think most people would learn this system quickly [yes/no]?	10/Yes
I thought the concept was cumbersome to use [yes/no]?	10/No
What if TMW was embedded into your mobile phone [yes/no]?	10/Yes

5.3 Results from Evaluation

The users felt at once comfortable using TMW to operate devices once they received training in what gesture did which action. It became apperant that gestures could be more intuitive than they were, when some users got confused for down-up, and out-in gestures. As such, some changes would have to be made. The new gestures after the evaluation would be Up-Down instead of Down-Up and Out-In would have to be disregarded for a better gesture that would require a better sensor fusion library. The user interaction by TMW with an MP3 player worked well and it proved a Magic Wand interface could be used for device interaction. The end users also were more comfortable to activate a device if it became embedded into their mobile phone. To illustrate this for the end users, were shown a motorola SLVR phone and asked to imagine that it had embedded acceleration sensor built in to it.

5.4 Did TMW Meet Requirements?

The first requirement was intuitive usage of TMW, and with training, the end users were comfortable with the gestures right away. The second requirement was autonomy of TMW; TMW was implemented to function with computational and wireless capability and it therefore became autonomous up to the point that it needed a network infrastructure in place, and it couldn't yet select the device of interest. The performance could be considered low as the user had to wait for a time slot to perform a gesture instead of intuitively swinging the wand in a gesture pattern. The gesture recognition was good for three gestures, but above that the rates of wrong gesture were more than randomly chosen gestures would be. This gesture recognition problem can be traced back to the limitations of the GHMM library that reads in one mean value per time t and the authors scaling down to three symbols. Physical feedback was evident as the device of interaction was an MP3 player, and onscreen display told the user when a gesture time slot was open and closed. The result of a successful gesture was also put on the display window.

Chapter 6

Conclusion

This final chapter draws a conclusion from chapter 2 (State of the Art) and TMW's design, implementation and testing. The conclusion is divided into 4 layers, Sensor Data, Sensor Fusion, Application, and Communication.

Sensor Data The sensor data provided by the MTx and MT9 was very accurate, and had no drift problems. The issue which arose was the complexity to perform relatively simple tasks like reset or set Hz, this has to be done in configuration mode where measurement data must come to a halt.

Sensor-Fusion Scaling the data down to three symbols did not perform too well, a better solution could be to keep the number of states to three, but increase the number of values to return more accurate data. Better data accuracy also means it is harder to repeat a gesture and a lot more training examples would be needed. The GHMM library limited the accuracy by only allowing one gesture value per time t, where ideally a gesture could have three values per time t or more.

Application The software trains gesture models and returns most probable models from a given gesture. The approach taken is one where the process does all the computations in the cycle, the gesture is read, it is returned to Markov sequence and recognised, and the recognised command is transmitted wirelessly. This approach gave a latency for the next gesture time slot to begin of almost 1 second. With 2.4 second recognition time slot, the users would depend on a prompt from visual or audio references.

Communication Transferring the commands was done with pre configured addresses, where the UDP server would wait on a remote device, to recognise the commands it could use. The MP3 player working from gestures was received quite well, and UDP was the correct choice of IP protocols as TMW would not wait for connection termination to go back to gesture recogntion.

Hardware Platform Gumstix worked well, it received gesture data without any memory overflow, and its component cards worked after specifications. The add on cards for the Gumstix were quite chunky, but during the summer of 2006 new smaller cards became available and specially made batteries are available too for future work.

Chapter 7

Future Work

Hardware The XWand button from chapter 2 works well to start a gesture reading, and by forcing each gesture reading to last a maximum of 2.5 seconds the length of time the user holds down the button, will not impact on the gesture reading. 2.5 seconds is long enough to make even complicated gestures like up-left-up or circles. The ubiquitous device could be activated by infrared signal and data could be sent through ZigBee, saving power and limiting the number of computational devices on the wand.

Software For the gestures we should consider upgrading to a library that is compatible with multi valued gestures per time t. Future research should be carried out on continuous gestures and model recognition. The program could benefit from using threads for sensor data and sensor fusion to eliminate latency on sensor port connections.

Bibliography

[1] Bluetooth specifications.

[2] *Virtual Reality Lab for research.*

[3] Wand. Wikipedia.

[4] *XWand: UI for Intelligent Spaces, Andrew Wilson and Steven Shafer*, 2003.

[5] Wasinee Rungsarityotin Ivan G. Costa Janne Grunau Matthias Heinig Alexander Schliep, Benjamin Georgi. *GHMM Library.* Max PLanck Institute for Molecular Genetics, Department Computational Molecular Biology Ihnestrasse 73 D-14195 Berlin, Germany, 0.7 edition, 09 2005.

[6] Allied Business Intelligence. *RFID White Paper.*

[7] Answers.com.

[8] H. Ishii B. Ullmer. *Emerging frameworks for tangible user interfaces.* IBM, April 2000.

[9] Ari Yosef Benasat. An inertial measurement unit for user interfaces. Technical report, Massachusetts Institute of Technology, September 2000.

[10] Chih-Chung Chang and Chih-Jen Lin. *A library for support vector machines*, 2001. Software available.

[11] Andy Wilson Daniel Wilson. Gesture recognition using the xwand. Technical report, Microsoft Research, 2001.

[12] Sinem Coleri Ergen. *ZigBee/IEEE 802.15.4 Summary*. ZigBee Alliance, 2004.

[13] Infrared Data Association. *IrDa The secure Wireless Link*.

[14] Frederic Vexo Daniel Thalmann Jan Ciger, Mario Gutierrez. The magic wand. Technical report, VRlab+EPFL, 2003.

[15] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[16] Matt Mackall. *Tiny Linux Project*.

[17] Trond Valen Frode Ingebrigtsen Lars-Erik Bjoerk Stein Jakob Nordboe Oeyvind Boe Syrstad, Erik Rogstad. Glove is in the air. Technical report, Norges Tekniske Naturvitenskapelige Universitet, 2004.

[18] Special Issue of Human-Computer Interaction, editor. *Introduction to this special issue on Context-Aware Computing by Thomas P. Moran and Paul Dourish*, volume 16. IBM Almaden REsearch Center, University of California, Irvine, 2001.

[19] J. Postel. User datagram protocol, 1980.

[20] Jon Postel. Transmission control protocol darpa internet program protocol specification, 1981.

[21] Lawrence L Rabiner, editor. *A tutorial on Hidden Markov Models and Selected Applications in Speech recognition*, volume 77. IEEE, 1989.

[22] The Physological Review, editor. *The Magical Number Seven, plus or minus two, by George A. Miller*, volume 63, 1956.

[23] SearchNetworking.com. *Definition: - Direct sequence spread spectrum.*

[24] Stanford Research Institute, Menlo Park, California 94025 USA. *Augmenting Human Intellect: A conceptual Framework by Douglas Engelbert*, 1962.

[25] Xsens technologies. Mtx 3dof orientation tracker.

[26] University of Cambridge. *Hidden Markov Model Toolkit.*

[27] University of CValifornia, Berkeley. *User Directed Screen reading for Context Menus on Freeform Text by Ka-Ping Yee*, 2003.

[28] Stuart N Wrigley. *Dynamic Time Warping*, 1998.

[29] XSens. Low-level communication documentation. Manual, 2005.