Wearable Gesture Recognition Device

Submitted By,
Vignesh Jathavara - 91327265
Arisha Kumar - 94161446

INSTRUCTOR: Professor KJ Lin
DATE: 12/15/2016
COURSE: Individual Research EECS

Table of Contents

1. INTRODUCTION

Gesture recognition is becoming an essential feature in modern day devices. Gesture recognition enables humans to communicate with machines in the most natural way wirelessly without even touching them.

A typical mouse has only three gestures namely left click, right click, and scroll wheel. A trackpad adds few more gestures like zoom in and zoom out and swipe left and right. Apart from these existing well accustomed gestures, gestures which can be utilized for presentations, controlling a media player (volume control, play next/previous etc.), gaming etc. can be incorporated.

Existing gesture detection techniques include mainly vision based gesture detection using devices like Microsoft Kinect and Intel RealSense. Apart from Vision based techniques there are other gesture detection devices like Leap Motion which uses camera as well as Infrared LEDs and Hover by HoverLabs which detects variations in the electromagnetic field around it.

Applicability of Vision based techniques is seriously affected by their limited range and wearability, their high computational and processing requirements and their sensitivity to lighting conditions, camera facing angles and calibration, which renders them unsuitable for use in a truly mobile context. Leap Motion and Hover are limited by the gesture detection range i.e. they can only detect gestures made close to the device(less than 3 feet). Motion sensors on the other hand are low cost and low power and can be used in much less constrained conditions, offering users more freedom in their interaction with computers. In this project, we intend to develop a prototype that uses motion sensing along with flex sensors to replace a mouse/track pad and also providing an array of added gestures that enable a user to perform various tasks be it for presentations, controlling a media player, gaming etc.

2. HARDWARE, TOOLS AND ALGORITHMS

Accelerometer:

An accelerometer is an electromechanical device that measures acceleration forces in three axis - x, y, z. This includes both static as well as dynamic forces. Static force is the earth’s gravitational pull and the dynamic forces are the ones caused by movements of the accelerometer.

These acceleration readings on three axis are unique to every movement made and the orientation of the accelerometer when the movement is made. This fact can be utilized to detect the kind of movement made. Once we have the data of sampled acceleration values of a particular movement, detecting this movement is done by pattern recognition algorithms.

Pattern Recognition Algorithms:

Pattern recognition is a branch of machine learning that focuses on the recognition of patterns and regularities in data. These algorithms can be either Supervised Learning (labeled data are available) or Unsupervised Learning(labeled data are unavailable). Since we already have labeled patterns of gestures, we will consider only supervised learning algorithms for recognition. We have utilized two such algorithms, namely Dynamic Time Warping and Neural Network.

Dynamic Time Warping:

The Dynamic Time Warping (DTW) distance measure is a technique that has long been known in speech recognition community. It allows a non-linear mapping of one signal to another by minimizing the distance between the two.

Suppose we have two time series, a sequence Q of length n, and a sequence C of length m, where Q = q1,q2,…,qi,…,qn and C = c1,c2,…,cj,…cm

To align these two sequences using DTW, we first construct an n-by-m matrix where the (i th , jth) element of the matrix corresponds to the squared distance, d(qi , cj) = (qi – cj) 2 , which is the alignment between points qi and cj. To find the best match between these two sequences, we retrieve a path through the matrix that minimizes the total cumulative distance between them as illustrated in Figure 1.

In particular, the optimal path is the path that minimizes the warping cost

$$DTW(Q, C) = \min\left(\sqrt{\sum_{k=1}^{K} w_k}\right)$$

where wk is the matrix element $(i,j)_k$ that also belongs to k^{th} element of a warping path W, a contiguous set of matrix elements that represent a mapping between Q and C.

This warping path can be found using dynamic programming to evaluate the following recurrence.

γ(i,j) = d(qi,cj) + min{ γ(i-1,j-1) , γ(i-1,j) , γ(i,j-1) } *where d(i,j) is the distance found in the current cell, and γ(i,j) is the cumulative distance of d(i,j) and the minimum cumulative distances from the three adjacent cells.*

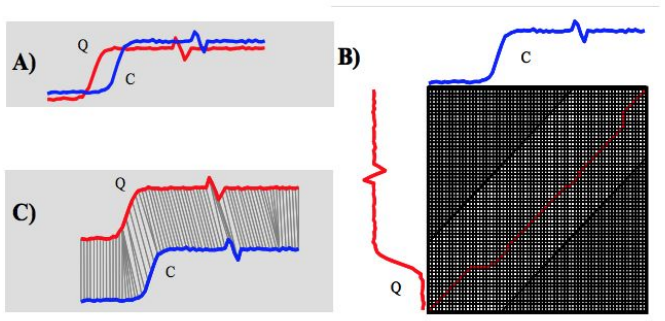


Figure 1. A) Two similar sequences Q and C, but out of phase. B) To align the sequences, we construct a warping matrix and search for the optimal warping path, shown with solid squares. Note that the 'corners' of the matrix (shown

in dark gray) are excluded from the search path as part of an Adjustment Window condition. C) The resulting alignment.

To reduce the number of paths to consider during the computation, several well-known constraints (Boundary Conditions, Continuity condition, Monotonic condition, and Adjustment Window Condition) have been applied to the problem to restrict the moves that can be made from any point in the path and so restrict the number of paths that need to be considered.

Neural Network:
An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward (figure 2.) neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

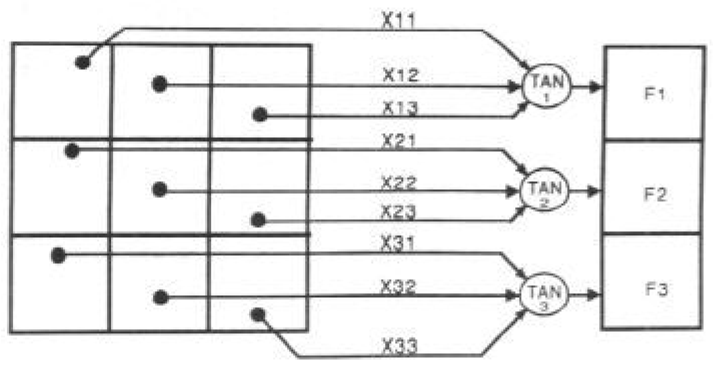


Figure 2.

Gesture Recognition Toolkit:
The Gesture Recognition Toolkit (GRT) is a cross-platform, open-source, C++ machine learning library that has been specifically designed for real-time gesture recognition. GRT contains

machine-learning algorithms such as: Adaptive Naive Bayes Classifier, Dynamic Time Warping, Hidden Markov Model, K-Nearest Neighbor Classifier, Support Vector Machine, Artificial Neural Network (Multi-Layer Perceptron), Linear Regression, Logistic Regression, etc.

Apart from these, the GRT also contains a large number of pre-processing, post-processing, and feature-extraction algorithms like filters, FFT etc.

OpenFrameworks:
openFrameworks is an open source C++ toolkit designed to assist the creative process by providing a simple and intuitive framework for experimentation. OpenFrameworks is designed to work as a general purpose glue, and wraps together several commonly used libraries. The code is written to be massively cross-compatible.

Flex Sensor:
A flexible resistor which increases resistance with increased angle. This can be used to detect bends. When the sensor is straight, the resistance is about 30k Ohms and when the sensor is bent, the resistance increases(to about 50k-70K Ohms when the sensor is bent to 90° as shown in figure 3). When the sensor straightens, the resistance returns to the original value. These sensors are fixed along fingers to detect when a finger is bent.

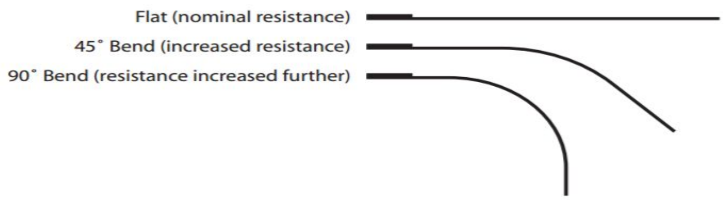

Flat (nominal resistance)
45° Bend (increased resistance)
90° Bend (resistance increased further)

Figure 3.

Intel Curie Module:
The Curie module is a Low-Power hardware module for wearable and other consumer and industrial edge devices. Powered by the Intel Quark SE SoC, it is extremely power efficient and meant for always on applications. The Curie Module includes:

- Low-Power 32-bit Intel Quark Microcontroller
- 384kb flash memory and 80kb SRAM
- Low Power integrated sensor hub and pattern matching technology
- Bluetooth Low Energy
- 6-axis combo sensor with accelerometer and gyroscope
- Battery Charging Circuitry

The pattern matching engine is a 128 node neural network, each of which is an Arithmetic unit. The neural network supports two Learning algorithms- K nearest neighbor and Radial Basis Function.

Arduino 101:
The Arduino 101 keeps the same robust form factor and peripheral list of the Arduino UNO along with an Intel Curie Module. The module contains two tiny cores, an x86 (Quark) and a 32-bit ARC architecture core, both clocked at 32MHz. It has 14 digital input/output pins (of which 4 can be used as PWM outputs), 6 analog inputs, a USB connector for serial communication and sketch upload, a power jack, an ICSP header with SPI signals and I2C dedicated pins. The board operating voltage and I/O is 3.3V but all pins are protected against 5V overvoltage.

Communication:

RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a data terminal equipment (DTE) such as computer terminal, and a data circuit-terminal equipment (DCE), such as a modem.

Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs).

BLE or Bluetooth low energy transmits less data over shorter distances using much less power than Bluetooth. BLE is designed for periodic transfers of very small amounts of data. BLE finds its use in number of applications such as healthcare, sports and fitness.

HID:
Human Interface Device or HID is type of computer device by which a human interacts with electronic information system either by inputting data or by providing an output. The most common HID devices are keyboard, mice, webcams, speakers and headsets. These devices provide an interface between the user and computer machines.
In the HID protocol, there are two entities: the "host" and the "device". The device (HID) is the entity that directly interacts with a human, such as a keyboard or mouse. The host communicates with the device and receives input data from the device on actions performed by the human. Output data flows from the host to the device and then to the human. The HID protocol makes implementation of devices very simple. Devices define their data packets and then present a "HID descriptor" to the host. The HID descriptor is a hard coded array of bytes that describe the device's data packets. This includes number of packets the device supports, the size of the packets, and the purpose of each byte and bit in the packet.

BLUETOOTH HID
The Bluetooth HID profile enables users to develop wireless products such as computer keyboards and keypads, trackballs, mice, and other pointing devices, and game controllers (gamepads, joysticks, steering wheels, etc.).

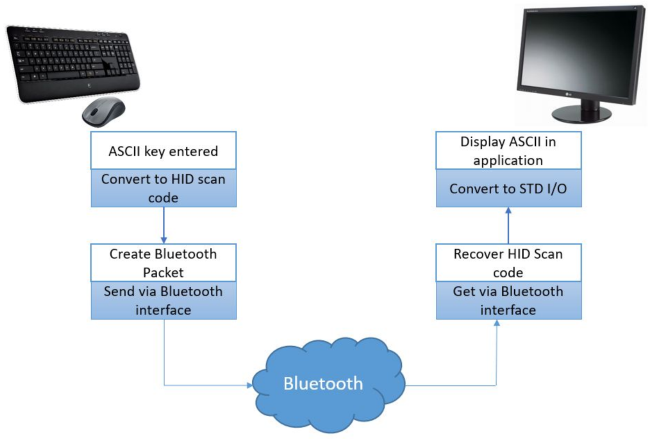


Figure 4

In a typical usage scenario such as a keyboard, a device using the Bluetooth HID profile replaces the USB cable. In this case, the ASCII value of a key press is converted to a scan code in a raw HID report that the Bluetooth module sends over the Bluetooth link to the host. The host driver software decodes the raw HID report and passes the key values to the application running on the PC. Figure 4 shows some typical HID environments.

Virtual HID
A virtual HID is a software implemented HID device which behaves like a HID device over USB or bluetooth. All commands are sent to the virtual HID in a similar fashion as done on USB HID using HID descriptors and reports.

Moving Average Filter
A moving average (rolling average or running average) is a calculation to analyze data points by creating series of averages of different subsets of the full data set. It can do the work of a simple low pass filter and removes unwanted low noise.

3. HARDWARE CONFIGURATION

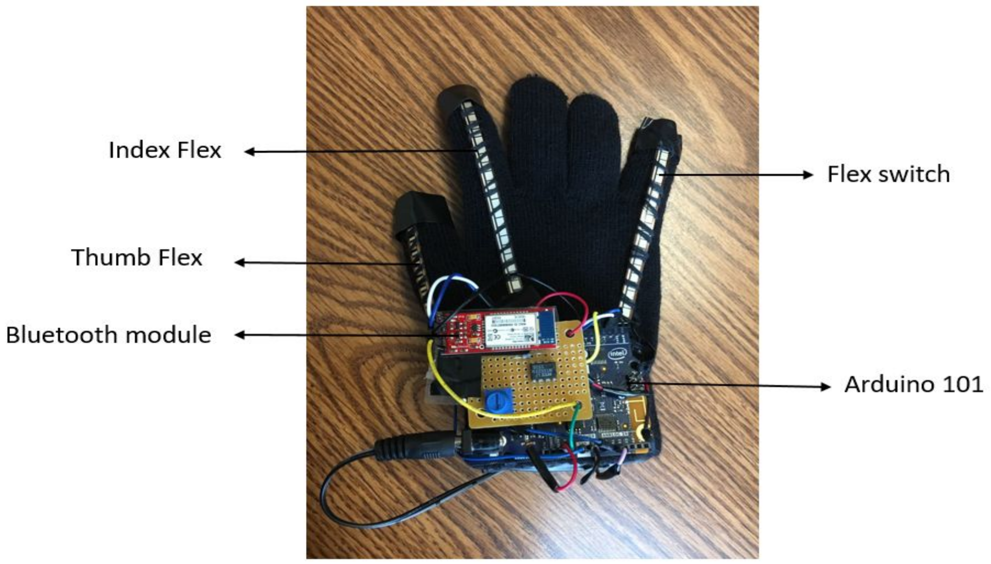


Figure 5.

The final prototype of our design is shown in Figure 5. We have used three flex sensors for detecting the finger movements and Arduino 101 for detecting wrist and hand movements using in built IMU and the pattern recognition engine of the Intel Curie Module.
Index flex sensor is used to incorporate the clicks in our design. The Thumb along with index finger is used for implementing the zoom in and zoom out gestures. The flex sensor on the little finger is used as a switch to trigger hand gesture detection such as swipe left/right etc. The circuit diagrams of the various components as well as the complete device is shown in Figures 6,7 and 8.

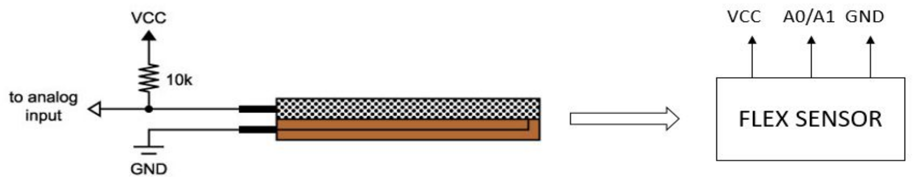


Figure 6. Circuit Diagram of Index/Thumb flex Sensors

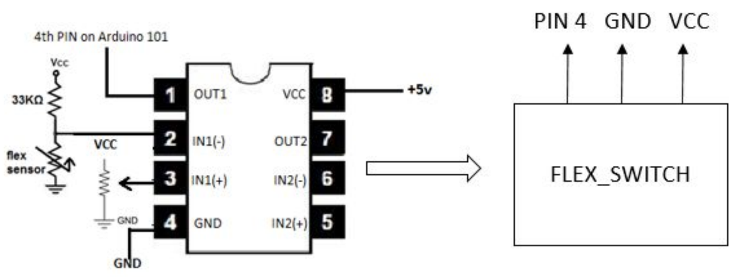


Figure 7. Circuit Diagram of the Flex Switch(little finger)

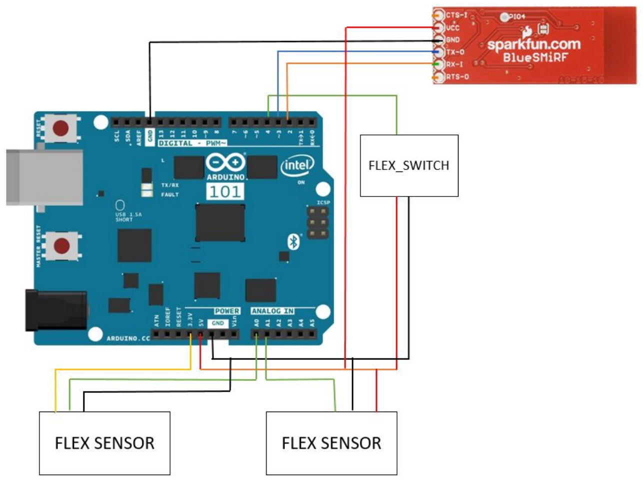


Figure 8. Complete circuit Diagram of the Device

4. GESTURES

Hand gestures can be performed in different ways. In this project we target gestures made by finger movements, wrist movements and also complete hand movements. We have zeroed in on a set of 19 such gestures which can be effectively detected by the device. Table 1 gives a list of all the gestures.

No.	OPERATION	SENSOR	GESTURES
HAND MOVEMENTS			

1	Scroll UP	IMU	Move Up with palm facing left.
2	Scroll DOWN	IMU	Move Down with palm facing left.
3	Scroll LEFT	IMU	Move Left with palm facing left.
4	Scroll RIGHT	IMU	Move Right with palm facing left.
5	Copy	IMU	Draw a ‘C’
6	Paste	IMU	Draw a ‘P’
7	Volume UP	IMU	Draw a Left Circle
8	Volume DOWN	IMU	Draw a Right Circle
9	Swipe LEFT	IMU	Move Left with palm facing up.
10	Swipe RIGHT	IMU	Move Right with palm facing up.
FINGER MOVEMENTS			
11	Single Click	Flex - Index	Bend Index finger once.
12	Double Click	Flex - Index	Bend Index finger twice.
13	Click & Hold	Flex - Index	Bend and hold Index finger.
14	Zoom IN	Flex - Thumb & Index	Pinch In
15	Zoom OUT	Flex - Thumb & Index	Pinch Out
WRIST MOVEMENTS			
16	Cursor Left	IMU	Tilt Left
17	Cursor Right	IMU	Tilt Right
18	Cursor Up	IMU	Tilt Up
19	Cursor Down	IMU	Tilt Down

TABLE 1.

4.1 GESTURES USING FLEX SENSORS

Flex sensors are used to detect finger movements. The flex sensors can detect the extent to which fingers are bent. This can be used to model gestures for click and zoom(pinch action). Flex sensors are used to detect 5 gestures - Single Click, Double Click, Click and hold, Zoom In and Zoom Out.

Single Click, Double Click, Click and hold are done using the flex sensor on index finger, by bending it once, twice or keeping it bent for more than 0.5 seconds.

Zoom In and Zoom Out are done using flex sensors on thumb and index fingers. The flex sensor on the thumb acts as a trigger for the pinch action. The range of movement of the index finger is divided into two segments, the lower half for Zoom In and the upper half for Zoom out. A pinch in action in the lower half segment will invoke a zoom in and similarly a pinch out action on the upper half will invoke zoom out.

4.2 MOTION DETECTED GESTURES

Motion gestures can be divided into two classes. One which models cursor movements by detecting wrist movements and the other for general gestures made by complete hand movements.

To model cursor movements, the wrist needs to be tilted in the required direction. This action transfers the gravitational forces onto the X and Y axis of the accelerometer from the default Z axis(when the accelerometer is placed upright). The more the tilt, more is the gravitational force on the particular axis and this can be used to move the cursor in the particular direction.

To model general gestures, complete hand movements are considered. Table 1 lists the various gestures that are detected. Each gesture takes about 1 to 2 seconds to perform. During this duration, the accelerometer readings of all three axis are sampled and then used to match a particular gesture. The flex sensor on the little finger is used as a trigger to start the detection process. This flex sensor acts as a switch and as long as the little finger is bent, the accelerometer data is collected, once the finger is released, the data collected is used to recognise the pattern drawn.

4.2.1 CONTINUOUS vs DISCRETE DETECTION

Continuous detection involves constantly monitoring the accelerometer for valid movements. This would not only increase the computation and power consumption, but also the freedom of hand movement is reduced as lot of unintentional movements can be misinterpreted as valid gestures.To avoid this, we use discrete detection by using the flex sensor on the little finger as a switch to trigger gesture detection.

5. METHODOLOGY

The three types of gestures(finger, wrist and complete hand) are handled in different ways.
The finger movements are detected based on the resistance of flex sensors. Effective value thresholds and timing constraints are used to detect gestures. Figure 9. Shows a state transition diagram for gestures single click, double click and click and hold.Figure 10. Shows a state transition diagram for Zoom In and Zoom Out gestures.

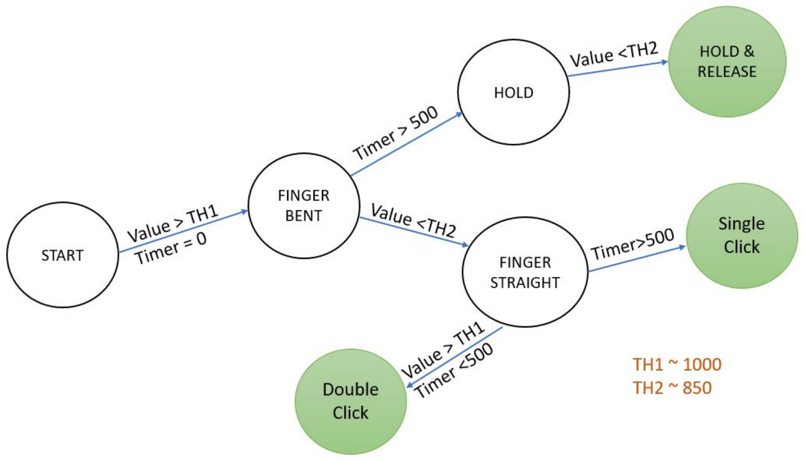


Figure 9.

To model cursor movements, wrist movements are used. The accelerometer readings are constantly measured for any increase. Any increase is multiplied by a conversion factor and this value is used to move the cursor relative to its current position. Thresholds are also set to ignore slight movements.

Motion Gesture Recognition can be subdivided into the following three steps. Sample the accelerometer to get data, Preprocess the Data, Either train the Machine Learning module or predict the pattern. The accelerometer readings are three signed 16 bit values and is sampled at 200Hz. A moving average filter is used which acts as a simple low pass filter, removing low noise and other small anomalies, thereby smoothing out the data stream. The next step can be done in two ways, either on the device or on the pc.

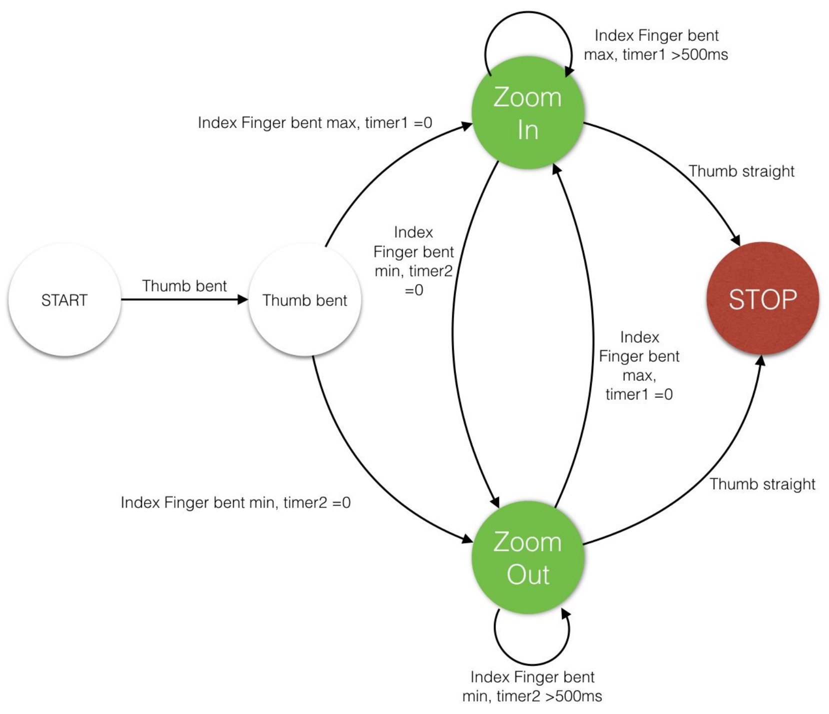

Index Finger bent max, timer1 >500ms
Zoom In
Index Finger bent max, timer1 =0
Thumb straight
Index Finger bent min, timer2 =0
START
Thumb bent
Thumb bent
STOP
Index Finger bent max, timer1 =0
Thumb straight
Index Finger bent min, timer2 =0
Zoom Out
Index Finger bent min, timer2 >500ms

Figure 10.

5.1 PATTERN RECOGNITION ON PC

Pattern recognition requires lot of processing power and generally cannot be handled by microcontrollers. Hence the accelerometer readings are sent to the PC and all processing takes place on the PC. This can also support continuous gesture detection as a pc has no energy and processing constraints. But this method will also require a mechanism to signal the Operating System about detected gestures so that necessary actions can be performed. This is generally done by using a virtual HID. The process is shown in Figure 11. Apart from that, the values from the device will have to be read constantly and this will require one thread to be committed to this task at all times. In the project we use Dynamic Time Warping from Gesture recognition toolkit for detection.

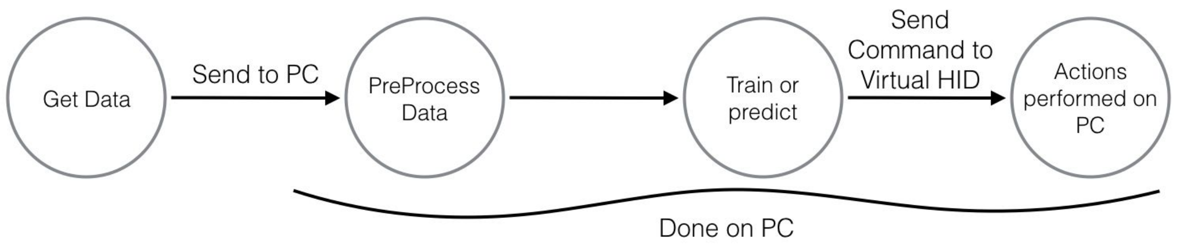


Figure 11.

5.2 PATTERN RECOGNITION ON BOARD

Another possibility is to perform all the processing on the device itself and send only necessary commands to the PC. This can be done using the Intel curie module which has an onboard Quark SE processor with a pattern matching engine. The raw accelerometer values are read, preprocessed and fed to the neural network which uses radial basis function as the activation function. Each neuron has a memory of 128 Bytes. Each gesture takes about 1 to 2 seconds to perform and the accelerometer is sampled at 200 Hz. So this will give about 240 samples every 1.2 seconds and each sample is 3 signed 16 bit values. Each neuron can only fit 21 of these samples. This will have to be condensed to fit into each neuron and this is achieved by first mapping the accelerometer values onto a number from 0 - 255 so each axis requires only 1 byte instead of two. Only every nth value is considered to fit into the memory size constraint. This would cover most of the sample window and have some semblance of the original pattern.
The network is trained once and the trained state of the neural network (including weights of committed neurons, type of classifier etc) is stored on the device in a file on the onboard flash memory. This can be restored each time the device is powered on.
The HID protocol can be used to send appropriate commands to the PC. This completely eliminates any processing on the PC. This also helps by not requiring to implement custom drivers on all platforms as the HID protocol is standard to all. Figure 12. Shows the process of pattern recognition.

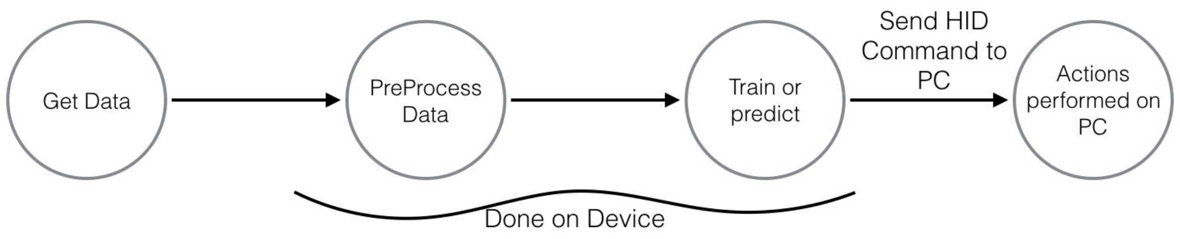
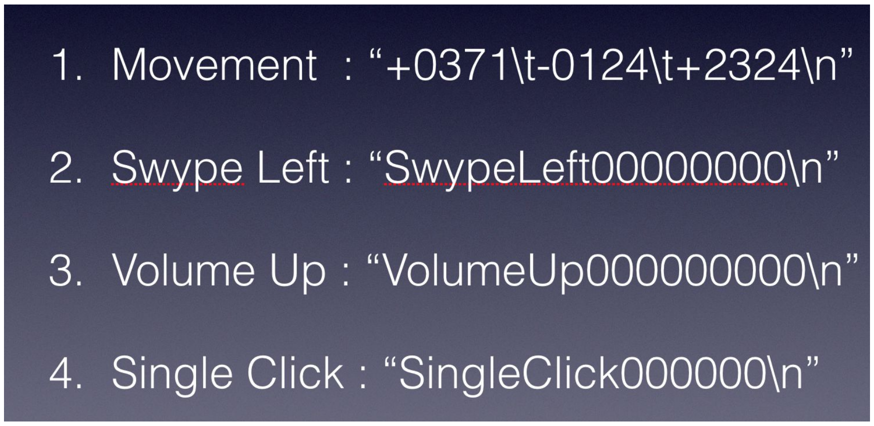


Figure 12.

6. COMMUNICATION

Initially, serial communication was used for all communication between the device and the PC. All data was sent in packets of size 18 Bytes. The limit of 20 bytes was chosen because the limit for data packets sent using BLE is 20 Bytes. The three readings from accelerometer were restricted to 4 digits each along with a sign. This coupled with two tabs and a newline character makes up 18 Bytes. All other commands for gestures modelled using flex sensors, padding is added so that the packet size is 18 Bytes. Figure 13. Shows examples of these. The PC also reads data 18 Bytes at a time.

Since the BLE onboard Curie module has no good support yet, a Bluetooth modem called Bluesmirf from sparkfun was used. This too did the same thing as serial communication with the only change being wireless.
The Bluesmirf is also capable of acting as a HID (Bluetooth HID which is very similar to USB HID but wireless). This was then configured to directly send the commands to PC instead of sending command information and then using a virtual HID to perform tasks. A serial to HID converter can also be used for wired communication.



1. Movement : “+0371\t-0124\t+2324\n”
2. Swype Left : “SwypeLeft00000000\n”
3. Volume Up : “VolumeUp000000000\n”
4. Single Click : “SingleClick000000\n”

Figure 13.

7. PERFORMANCE

The detection performance of the device depends on certain factors like orientation and stability of the device as well as the speed of gestures. Factors such as stability and orientation can be addressed by fixing the board onto a base(We have used a glove) so that it remains in the same orientation and position during each use. For speed of the gestures performed, a general advisory would be to perform the gesture in such a way that it lasts at least for one second, this might not be a problem because in our tests, that was the average time taken to perform the actions by all the testers.

The neural network pattern matching engine, provides the result of prediction to be the closest pattern matched unless the movement is entirely different when a no match is predicted. This is not a problem when a gesture is performed, as the closest one is almost always the one intended. This has been ensured by using different axis to take the bulk of acceleration for most of the commands.For example all the scroll gestures are performed with the palm facing left to ensure gravitational forces are acting on the X axis of the device. But this poses a problem when the user triggers a detection and performs no gesture. To avoid such false positives, the mean of difference between values of two successive samples is calculated. This mean if less than one for each axis, then prediction is not performed and a no match is returned.

7.1 DETECTION PERFORMANCE

The detection mechanism using onboard pattern matching performs better than the DTW algorithm from GRT. The implementation using GRT sometimes gives multiple detection on single gesture movement for certain gestures like circles. The pattern matching engine on the

other hand is very accurate in detecting gestures but does detect certain false positives for movements which are similar to valid gestures.
The detection performance of the device is described in the tables 2(for detection using DTW and processing on PC) and 3(on board detection using neural network)

Case	True Positives	True Negatives
Trained and tested by same person	75%	99%
Trained and tested by different people	60%	99%

Table 2. Performance of detection on PC using DTW

Case	True Positives	True Negatives
Trained and tested by same person	95%	80%
Trained and tested by different people	95%	80%

Table 3. Performance of On board detection using neural network

8. APPLICATIONS

The idea behind the device was to replace the conventional mouse, to better aid during presentations, playing games other general operations. The precision of cursor movements is not so well suited to the minute UI buttons of existing Operating Systems but with the UI design moving more towards touch inputs, the new UI designs like Metro UI of windows can be easily controlled using the device.
The device can be connected to phones and tablets and the gestures can be mapped to control tasks like receiving/rejecting calls, media player controls etc.
Another use of the device will be in factories to map gestures to control heavy machinery, control drones etc.

9. REFERENCES

https://www.dimensionengineering.com/info/accelerometers

https://en.wikipedia.org/wiki/Pattern_recognition http://wearables.cc.gatech.edu/paper_of_week/DTW_myths.pdf

https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

http://www.spectrasymbol.com/flex-sensor

http://www.intel.com/content/www/us/en/wearables/intel-curie-fact-sheet.html

https://en.wikipedia.org/wiki/Human_interface_device

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/Bluetooth/RN-HID-User%20Guide-1.1r.pdf

https://en.wikipedia.org/wiki/Bluetooth

https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide?_ga=1.257220099.909569890.1470808021

Chotirat Ann Ratanamahatana and Eamonn Keogh, "Everything you know about Dynamic Time Warping is Wrong", Department of Computer Science and Engineering University of California, Riverside Riverside, CA 92521

David Mace, Wei Gao and Ayse K. Coskun, "Accelerometer-Based Hand Gesture Recognition using Feature Weighted Naïve Bayesian Classifiers and Dynamic Time Warping", IUI'13 Companion, March 19–22, 2013, Santa Monica, CA, USA

Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya and Venu Vasudevan, "uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications",

Katarzyna Barczewska and Aleksandra Drozd, "Comparison of methods for hand gesture recognition based on Dynamic Time Warping algorithm", Proceedings of the 2013 Federated Conference on Computer Science and Information Systems pp. 207–210

Xian Wang, Paula Tarrío, Ana M. Bernardos, Eduardo Metola, José R. Casar, "User-independent accelerometer-based gesture recognition for mobile devices"

Michael Xie and David Pan, "Accelerometer Gesture Recognition"

Jiahui Wu , Gang Pan , Daqing Zhang , Guande Qi , and Shijian Li, "Gesture Recognition with a 3-D Accelerometer "