Abstract

Computers have come a long way to become a part and parcel of people's everyday life. There is a rising need to provide innovative solutions to meet the ever-increasing need and provide new features which will make work easier for people. This report gives a detailed insight into one such new feature which would possibly soon see the light of the day. The idea is to create air type keyboard, which would take the typing technology one step further by doing away the need for physical keyboard. The gloves worn by the user would allow it to type directly on the surface. This report first introduces to the background and explains the idea followed by providing a detailed explanation of the algorithm used and its working. The description is first categorized in two broad category of hardware side and the software side. The Hardware part focuses mainly on the sensors and the motherboard used followed by the glove construction details and its implementation. The software category gives a detailed explanation of the algorithm used and the following code implemented.

1. Introduction

1.1. Motivation

As the product becomes more popular, more and more innovations are required to ease the current challenges and provide more features. It is important that product should keep its pace and keep on upgrading. Since the advent of computers, a keyboard has undergone a vast change in its design and features. The current generation of keyboard take it to a new level with idea being to do away with the physical keyboard. A different variant of this scheme in the form of finger sensor keyboard is being proposed in this report. Now every computer user have their own unique typing pattern, with some using their Index finger to type most of characters while others use all of their fingers to type. This correlation between fingers and the keys is known as the Keystroke Dynamics. Every user will have his own unique Keystroke Dynamic. Finger Motion Keyboard uses this correlation to become their personal keyboard by learning the keystroke dynamics of all the individual users. The device makes use of the hand gloves to monitor finger motion and learn the user's keystroke dynamics thus eliminating the requirement of physical keyboard.

1.2. Problem Statement

The main objective of this report is to design gloves with motion sensor, pressure sensor along with an algorithm to associate keys with finger motion and to design an algorithm which makes use of the training data to predict keys.

1.3. Keyboard Technology

In early era when computer technology was an infant, typewriters were used as text based entry device. Modern computer keyboard evolved from early teleprinters and keypunch devices. As continues use of QWERTY layout, invented by Christopher Sholes in 1870s, in typewriter the

same layout was adopted by electromechanical keyboard because of its widely use. He could not have known that his design persists for 150 years even in digital world also. In this rapid change of high technology, computer keyboard is undoubted king of computer input device.

Some recent technology started taking place of keyboard and requirement of physical keyboard has been seriously threatened. Innovation in image processing, speech recognition and gesture control are raising its power to replace physical existence of computer keyboard. There are some existing keyboard technologies or concepts which eliminates the need of physical key describes below.

- **Orbital Keyboard Technology**

Orbitouch keyboard is one of the keyboard technology that doesn't need fingers or wrist motion to type. It uses two domes and it can slide through 8 direction in its orbit.

Figure 1 : Orbital Keyboard Technology [1]

To type a letter, user needs to slide right dome to that letter and left dome to its corresponding color irrespective to which dome moves first. These keyboard is well used for person with low

vision, Cerebral palsy and other disabilities. [2]

- **No-key keyboard**

No- Keyboard is concept keyboard designed by Kong Fanwen. This keyboard is design is minimalistic and uses projecting laser touch. This keyboard uses a very thin light beam just above the well etched glass sheet and camera inline with light beam. Camera senses user's contact with glass and send appropriate information to PC. This ultimate motion capture technology is design is waterproof also.[4]

Figure 2 : No-key keyboard [3]

- **Projection Keyboard**

Engineers at IBM patented optical virtual keyboard in 1992. This keyboard takes human finger movement input by camera mounted on device and it tracks key pressing actions and interprets it as operations which make on physically non-existent input device like a plain surface or keyboard layout printed surface. This keyboard uses laser or beamer to project virtual keyboard on a surface. One camera or sensor has been mounted on this device to track down movement of keyboard. They use second infrared based layer just above the surface to track down keystrokes action. When this

sensor breaks or get inference in layer, it track down finger coordinates to predict key.[5]

Figure 3 : Projection Keyboard [6]

- **The Ring**

The ring is wearable input device that allows user to control. This device allows user to use gesture control, text based typing and alerts. It uses letter recognizing software to recognize letter by action of writing that letter. They use motion sensors and senses finer motion for every letter and predict letter. This is similar to handwriting recognition with its motion. [7]

Figure 4 : The Ring [8]

- **Tap**

Tap is wearable input device that allows user to type and control mobile as well as computer

devices. It is one hand device that user needs to wear on fingers which track finger movements to predict letters. It doesn't work with traditional QWERTY layout keyboard, instead it uses pre-defined finger taping for character like tapping one finger will give best fit vowel and combination of other fingers throw defined character. Device uses Bluetooth to send all commands to connected devices. This is also used for people who uses VR environment. They also provide some practice games and sources to get used to with this keyboard interface. [9]

Figure 5 : Tap [10]

- **Gest**

Gest is wearable keyboard that allows user to control computer or mobile with hand gestures. [11] It uses IMU(inertial measurement unit) to track down finger movement. It also used for designing, gesture based any operation and typing. This is also used for some virtual reality devices to control actions. This wearable is designed for pro typist who can actually type as predefined keystroke dynamics. [12]

Figure 6 : Gest [13]

2. Gloves design

This chapter of thesis is divided in two subsections. In the first subsection we discuss about the designing of sensor gloves (Hardware work). While the second subsection describes the proposed algorithm to train the system and utilize it to predict words. (Software work)

2.1. Hardware Work

The key challenge in creating the AIRtouch keyboard is the need to measure the finger motion with apt precision. An important requirement for this is to design wearable sensors gloves with specific focus on bending and tapping sensors. Bending sensor will provide the angular position of finger while tapping sensor senses key pressing activity.

Figure 7 : Keyboard Layout

In our approach it is of prime importance to find out which finger is being used by the user to press the key, accordingly Tapping sensor will help in finding out which finger is utilized. Since a user can use multiple finger for pressing the same key as well as press different keys with the same finger, tapping sensor alone stands insufficient in deciding which key is being pressed. This is where bending sensor comes into the play. Based on the curvature of the bend or the angle along with the data from the tapping sensor the decision is made for which key is pressed. This is a

pivotal role in the entire functioning of the keyboard and at the same time it is equally tedious. To simplify it further and to get more accurate results the keyboard is further divided into three layers. Fig.1 depicts the three different layers. Red rectangle indicates Upper Layer of the keyboard, blue rectangle indicates Middle Layer, and green rectangle indicates Lower Layer. After setting up the sensors on the gloves it is important to note that besides its ability to measure bending motion, gloves should be easy donning, easy removals, cost effective, durable, and comfortable.

Another vital component of this design is mother board. A rapid evolution in controller technology makes selection of controller harder. Merging functionality and faster processing times are fundamentals of controller. Teensy USB development board, MSP430 emulation board, beaglebone A6 and Arduino are great fit for our requirement. Considering cost, project complexity, availability in the market, support for various sensors and the vision of future development in this design, open source development kit Arduino board was chosen. Among all open source platforms, Arduino is one of the best open source electronics pattern development platform with flexibility in use of hardware and software. Arduino is good at sensing environment through interfacing various sensors and processing in real time data using ATmega328P.

There are two parameters to measure from user, one is finger bending measurement and another is finger tapping activity. These sensors should work with low power with higher efficiency. As we are designing wearable gloves, sensors needs to be flexible and compact enough to fit over finger. After several research, we selected pressure sensor to sense the pressing activity and flex sensor for determining motion activity.

2.2. Bending Sensors

- **Microbending sensors**

Microbend sensors are based on coupling and leakage of modes that are propagating in a deformed fibre. Usually one achieves this deformation by employing corrugated plates that deforms the fibre into a series of sharp bend with small bending radii. In our laboratories we have developed a highly sensitive chemical sensor by inducing permanent microbends on a bare plastic optical fibre. The output intensity is found to be linearly dependent on the logarithm of concentration of the absorbing species surrounding the the bent portion of the fibre. This sensor can even detect very low concentrations , of the order of nanomoles per liter. and the dynamic range of the sensor is found to 6 order of magnitude. By carefully choosing the reagents this microbend sensor can be used to detect different chemical species.[14]

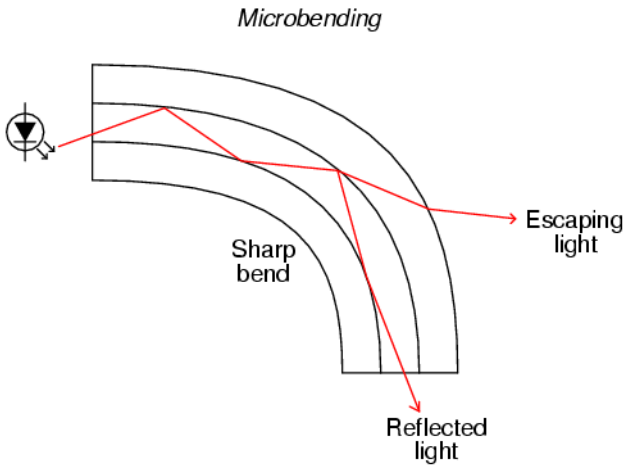


Figure 8 : Microbanding Sensors [15]

- **Flex Sensor**

Flexion Sensor (from Latin flectere, 'to bend') also called bend sensor, measures amount of deflection caused by bending of the sensor. This is nothing but flexible conductive ink printed on

flexible layer of polymer. As sensor bends, it stretches flexible ink inside, extending itself, which results in reduction of cross section area of ink layer. As the sensor flex, resistance across its two point change accordingly.

As micro bending sensors are very expensive and it has very rare support as well as it needs some laboratory experimental devices to use it. Whereas flex sensors are easily available, much cheaper option and it has support which describes the use sensors with various microcontroller and processor based system. [16]

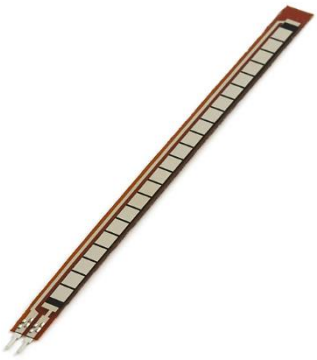


Figure 9 : Flex Sensors [16]

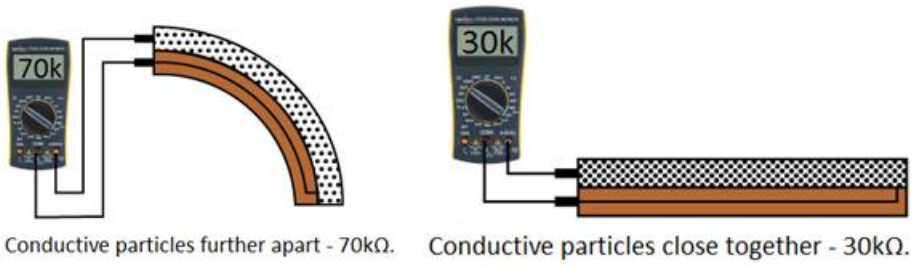
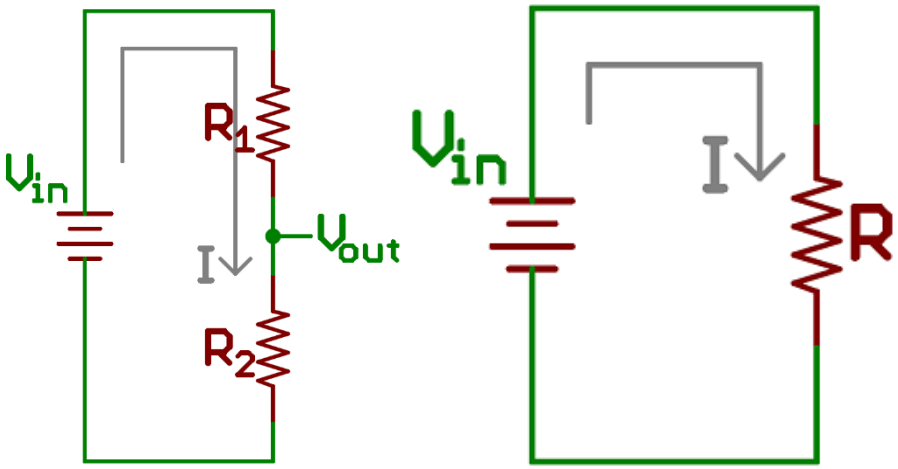


Figure 10 : Flex Sensor Measurement [16]

The simplest way to implement this circuit in my project is using voltage divider circuit, where we pass current from one pin and measure voltage from another end. Based on input current and output

voltage with input voltage value we can calculate resistance value using below equation.



Here R1 is flex sensor, R2 is voltage divider resistor, V_{in} is input provided by Arduino board, V_{out} is output voltage given to analog pin of Arduino board. [16]

Taking effective total resistance R = R1 + R2

Current $I = V_{in} / R$

$V_{out} = R2 * V_{in} / (R1 + R2)$

Using above equation, we know value of R2, input voltage V_{in} and measured output voltage V_{out}, we can calculate value flex sensor resistance R1.

Below are some reading from flex sensors which we recorded while testing the device.

```
CLEARDATA
0 245
0 245
0 256
0 245
0 230
END
1 139
1 129
1 133
END
2 354
2 365
2 350
2 350
END
3 85
3 81
3 68
END
4 517
4 501
4 501
4 563
END
5 474
5 430
5 440
END
6 64
6 64
6 64
6 64
END
7 351
7 354
7 351
7 351
7 351
7 351
7 351
END
```

Figure 11 : Flex Sensor Readings

2.3. Force Sensitive Sensor

The purpose of Force Sensitive Resistor also known as FSR is to measure amount of force or pressure. It contains conductive polymer which are used in Bend Sensors. The conductive polymer changes resistance when force is applied on the sensor. Here, this sensor is used to detect finger tapping considering it same as key pressing activity. Various force sensitive sensors are available

in market, as per my thesis requirement, I want sensor that fits on fingertip and also it should be printed over flexible PCB so that it can fit beneath the fingertip without hassle. To cover finger tapping activity, choice of round shaped sensor is the good option among rectangular, square and strip shape sensor.

Figure 12 : Force Sensitive Resistor [16]

2.4. Arduino Board

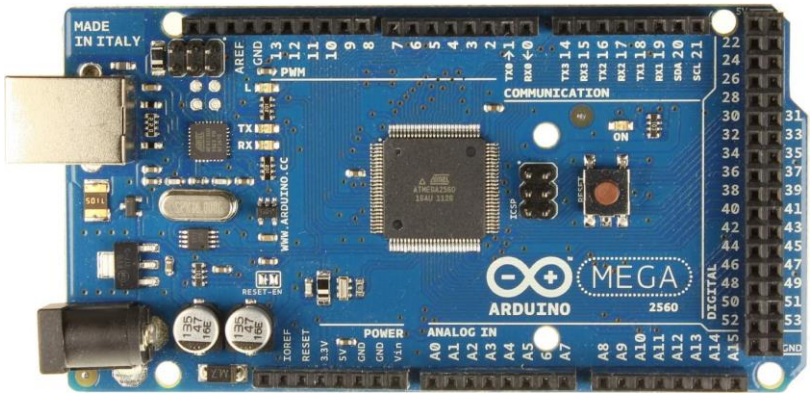


Figure 13 : Arduino Mega [17]

Arduino is a company which design and develop open source computer hardware and software.

They develop hardware based on microcontroller and user friendly coding language with general purpose interface support. They have designed several kits based on different requirements. Sensor gloves require eight digital input pins to detect finger tapping action, eight analog input pins to detect bending of fingers and UART interface to transmit data to computer. Arduino mega seems a good fit as it fulfills all the requirements for the glove development. The Arduino Board runs over a 5V power adapter input or USB power. In early project process, I used Arduino UNO board by Arduino that cost me around $25. That board has 6 analog input pins and 2 PWM pins which I used as analog pins by making resistor capacitor circuit and adding small programming module. Arduino UNO has 8 digital input pins. Due to wire mishandling board got burnt and it started showing random behavior. At the same time, I got deal on Arduino Mega board from craigslist. Due to budget constraints, we must go with cheaper option. [17]

3. Gloves construction:

To make the gloves easy to wear and more comfortable for the user and knowing that ease of use would be of prime importance, few models were made to experiment with various size and distinctive material of gloves. Different permutation and combinations were done. Several materials like cotton, rubber and leather were tried, but it was important that the material exhibit stretch so that fingers' motion would not be restricted because of gloves's less adaptability and sensors must slide on material instead of stick on it. Once the gloves to be used were finalized the pressure sensors were mounted correctly on finger tip with accurate precision with the sole intention that the sensors gave key pressing notification in all three layers. Flex sensors were placed on top of finger with sliding mechanism on proximal phalanx of finger and fixed from tip of finger.

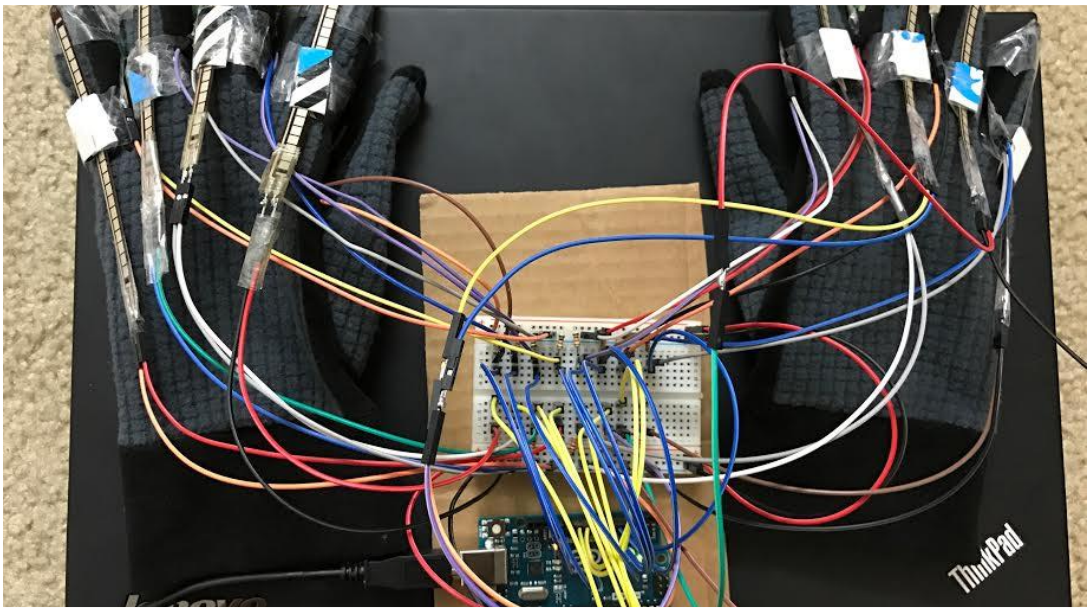


Figure 14 : Sensor gloves and Arduino board

Figure 14 presents the gloves along with their connection to Arduino board. Wires from sensors plugged in to bread board to give all connection line pull ups with registers 22K and 10K, to flex sensor and to pressure sensor respectively. To provide pull ups to all connection line, We used

breadboard where all wires from sensors connected with respective resistor and then further extended to Arduino board pins.

3.1. Hardware Schematic Diagram

Figure # shows schematic of gloves design, Rectangular box represent flex sensor component and square symbol represent Force sensitive resistors. Flex sensors denoted Fx : x € {0,1, … 7} represents respective finger, connected to analog input pins of Arduino board with 22K pull up resistors.

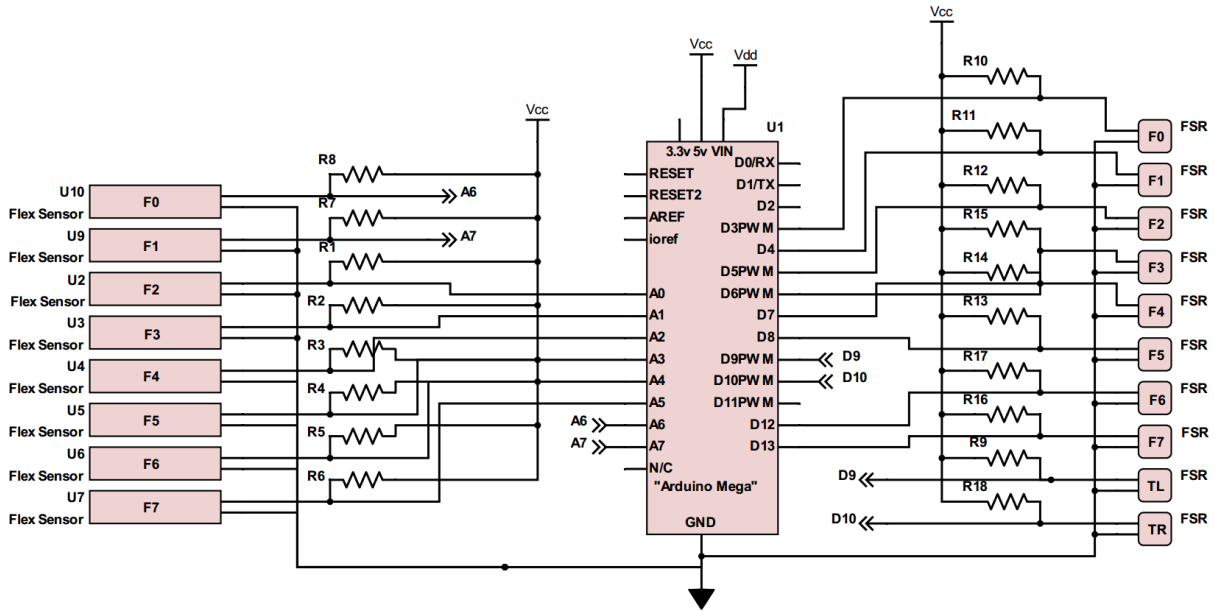


Figure 15 : Schematic Diagram

Whereas Force sensitive resistors denoted Fx : x € {0,1, … 7} represents respective finger & TL and TH represents left hand thumb and right hand thumb respectively, connected to digital input pins of Arduino board with 10K pull up resistors. Resistance value of R1-R8 is 22K Ohm and R9-R18 is 10K Ohm. As a result of several trial and errors, I concluded these values of resistors to get

desired result. U1 symbol represents for Arduino Mega board, which created with only useful pins only. Unnecessary pins are not shown in symbol. There are two power supplies are given in schematic, one is Vdd which represents Arduino input power supply given through USB interface and Vcc represents 5V output power provided by Arduino board to sensors. Power consumption of this circuit is very nominal(not more than 20 mA at 5V supply).

3.2. Top Level Block Diagram

Figure # represent top level block diagram of our design. Two hand gloves are separately connected breadboard where sensors input lines are pulled up and further connected to Arduino board's respective input pins. Arduino board has two communication port, one is UART and one is USB interface. UART interface requires lot of component and hardware work where as USB interface only require USB compatible cable and it also provide power supply to Arduino board. So it does not require to provide power to Arduino board with external power supply.

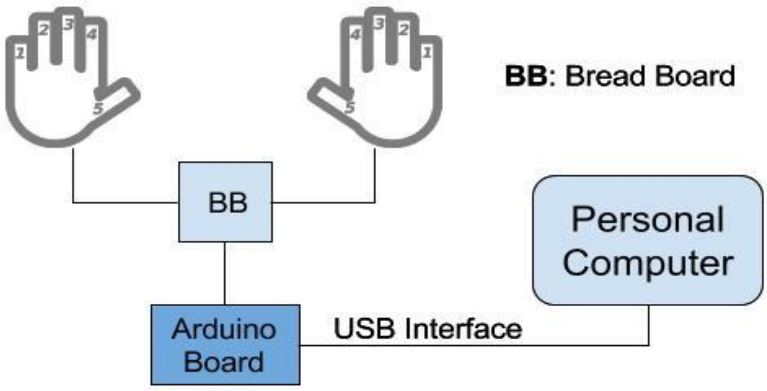


Figure 16 : Top Level Block Diagram

Arduino is connected with personal computer by USB cable which has one end USB type-B plug

and another end with USB type-A plug because Arduino has USB type-B jack connector and computer has USB type-A jack connector.

3.3. Software:

My thesis used following software and codes to execute.

- Arduino code
- 'PLX-DAQ' - Parallax Data Acquisition Tool
- Python code for data acquisition

Arduino Code

Arduino board can be programmed using Arduino language. Arduino language is merely a set of C/C++ functions that can be called from your code. Your sketch undergoes minor changes (e.g. automatic generation of function prototypes) and then is passed directly to a C/C++ compiler (avr-g++). All standard C and C++ constructs supported by avr-g++ should work in Arduino.[18] The code is written in Arduino language to collect data from sensors and to send this data to personal computer. The coding structure of Arduino always consists two parts: Setup function and Loop function.

- **Setup Function**: This function initializes all the pins as per the requirements, variables which are used in Loop function and also start using libraries. This function will execute once after Arduino board powered up or restart. We initialize sensor inputs pins and imported serial library for serial communication in setup function.
- **Loop Function**: After creating setup function, which initialize values and libraries, the loop function runs in loop and continuously allow program to change and respond

accordingly. This function is used to actively control Arduino board. In this loop function, we take input from sensors, segregate it accordingly and this send data to computer in comma separated format using USB interface.

PLX_DAQ

PLX-DAQ, Parallel data acquisition tool, used to get data from microcontroller with sensor system to Microsoft excel. PLX_DAQ is an add-on for Microsoft excel, acquires up to 26 channels from any parallel microcontroller interface. This tool gives real time data input facility on any COM port with baud rate up to 128K. Figure # displays reading of flex sensors coming from Arduino board. The data is exported to an excel sheet for determining word phase once training phase has been done. *Now, I have created python module which takes input from COM port and store it in CSV file. I don't need this tool anymore.*

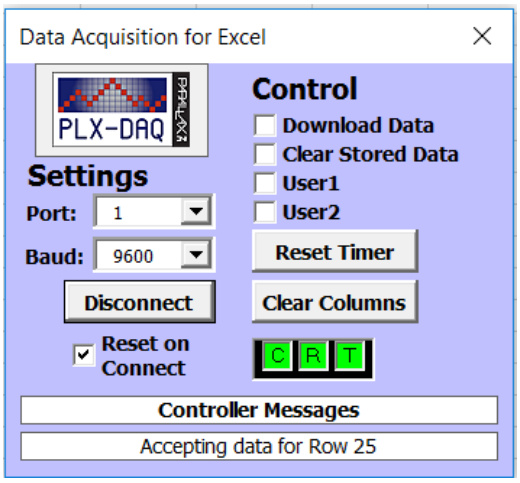


Figure 17 : PLX-DAQ tool

Python code for data acquisition

PLX- DAQ is good tool for sensor based microcontroller system. Our system required to process these data in real time within the excel spreadsheet where PLX-DAQ stores data. PLX-DAQ spreadsheet doesn't allow any other operation to work on its spreadsheet. So every time fetching this data to another excel spreadsheet and to process it, makes this process bit slower and more complex. We found better option of using python as these programming language has pool of libraries that interact with hardware also. Python library pySerial encapsulate the access of the serial port. (https://pythonhosted.org/pyserial/pyserial.html#overview) This library allows user to take input in different bit size, stop bits, and parity flow control with RTC(Real Time Clock). It also allows to set input delay. It also compatible with all I/O libraries of python.

In our project, Arduino board is connected via USB port to the computer on communication port 1 (COM1). From pySerial library, we used serial.Serial() function which initialize port value, baudrate, bytesize, parity and delay.

```
ser =serial.Serial(port='COM1', baudrate=9600, bytesize=serial.EIGHTBITS ,
parity=serial.PARITY_NONE, timeout=100)
```

Generally, we set baud rate to 9600 bits per second where speed isn't critical criteria. (https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial) We are reading ascii value as input and ASCII defined in 7-bit character set. While reading sensors input, it gives bunch of readings at every one tapping activity. So timeout feature gives timeout error if it does not see any activity in 100s.

Below is output data coming from arduino board,

```
CLEARDATA
0 245
0 245
0 256
0 245
0 230
END
1 139
1 129
1 133
END
2 354
2 365
2 350
2 350
END
3 85
3 81
3 68
END
4 517
4 501
4 501
4 563
END
5 474
5 430
5 440
END
6 64
6 64
6 64
6 64
END
7 351
7 354
7 351
7 351
7 351
7 351
7 351
END
```

Figure 18 : Input Data Readings

4. Model Flow Chart

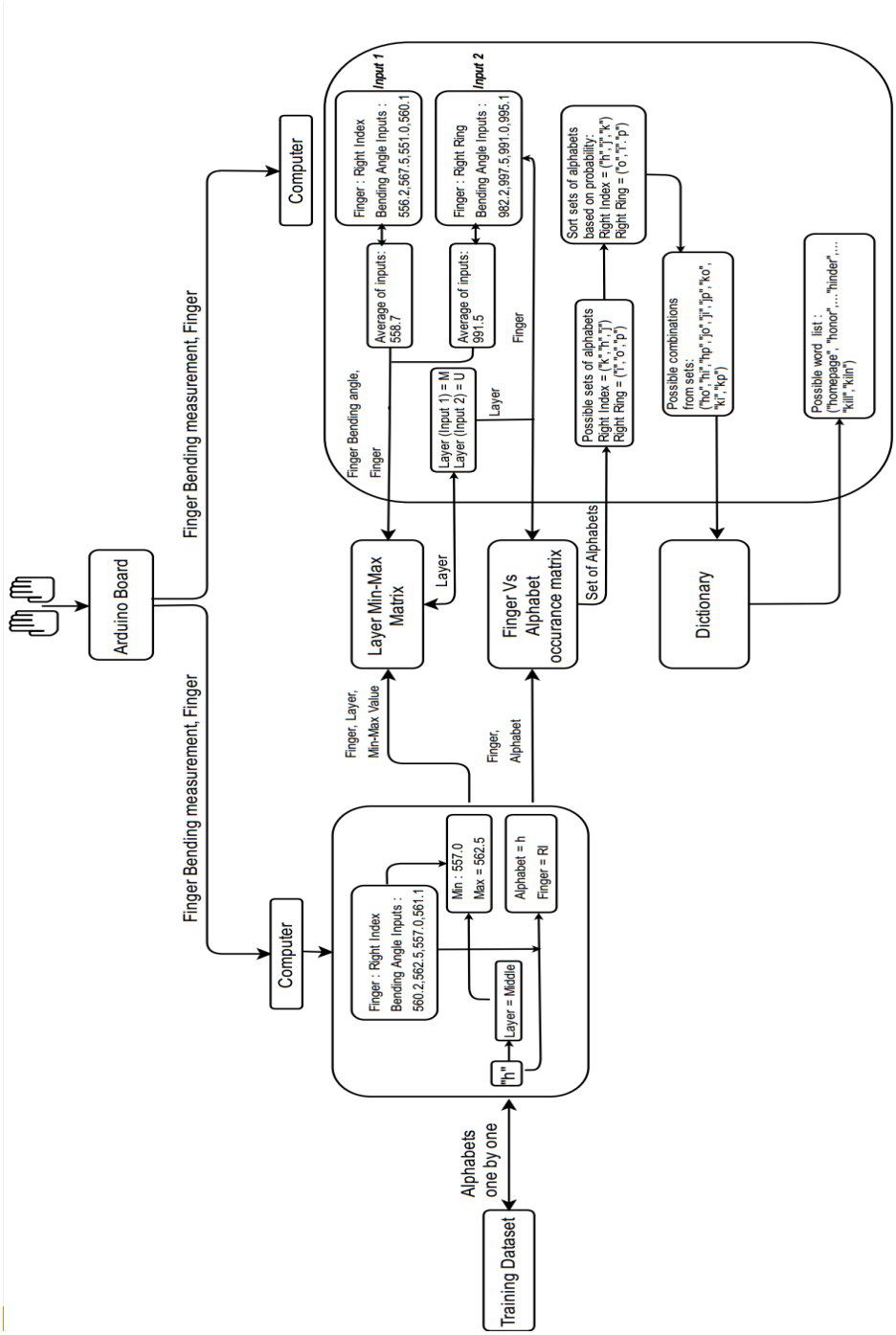


Figure 19 : Model Flow Chart

5. Model for prediction of keystroke

Assuming that the model is trained already. Training has been done in required training environment by following training instructions and limitations were strictly followed. This chapter illustrates keystroke prediction algorithm. Considering this device as concept device, it runs with certain limitations and assumptions. This concept device predict keystroke activity of user and gives predicted word based on input sequence under given limitations and assumptions.

Assumption: User has predictable typing pattern; random typing pattern may lead this device to an unpredictable state. In training part, the algorithm capture finger bending measurement range for each layer. To predict correct layer user is not supposed to lift wrist once starts typing.

Limitations: As of now system uses a dictionary, created from Shakespeare's work, to predict word. So, system is unpredictable for nouns. Nouns which are used in Shakespeare's work can be predict using this device. Right now, system is designed to predict word which contain only alphabets(a-z).

Input parameters while device in use : Finger, Finger bending measurement

Pre-req : Layer Min-Max matrix, Finger VS. Alphabets occurrence matrix, Transition matrix – must be updated based on dictionary words, pre-made word dictionary

Taking in consideration that we already have all pre-requested matrices and dictionary. When system is in "Device in Use mode", make sure that gloves are connected to Arduino board and Arduino board is connected with Computer on COM1. Arduino transmit information in serial data form, serial.py module receive data from serial communication port -COM1 and separate information about finger, thumb and finger bending measurement. System derive layer information

from Layer Min-Max matrix using below process.

- **Layer Min_Max Matrix :**

In this process, our inputs are finger value from where we observed tapping activity and bending measurements from that finger. As we know, each tapping activity gives us more than one measurement data, we average out those reading to remove noise in data. Algorithm takes finger input checks in that finger's min & max range and find out layer information in which range this value lies.

Now system has three parameters: Finger, Finger bending measurement and layer Information. Among those parameters finger and layer information used to predict possible alphabets sets from Finger Vs. Alphabet occurrence matrix. This matrix is filled with information about occurrences of alphabets with it's associated finger.

Input Observations:	Q1,Q2,Q3,Q4,Q5…Qn where Q1,Q2,…Qn are finger bending measurement inputs from Arduino board. F where F denotes finger input from Arduino board having values from 0 to 7. Values 0 to 7 denotes LL, LR, LM, LI, RI, RM, RR and RL respectively.

Qavg = avg(Q1, Q2, Q3 … Qn) where avg() gives arithmetic average.

Output:	{I: $[Layer_i][Finger\_min_y]$ <= Qavg <=$[Layer_i][Finger\_max_y]$ | i € {0,1,2}, y = F}

Table 1 : Layer Min_Max Matrix Operation

- **Finger Vs. Alphabet occurrence matrix**:

Algorithm extract out one table from given matrix based on layer value. Further process extract out set of characters, pressed by input finger; for which table gives some integer value other than zero. '0' represents that user has never used that finger to press given letter. This list of possible letters are extracted with the probability of the letter being pressed. The probability is calculated as shown below:

$$p(l, f) = \frac{Number\ of\ times\ letter\ l\ typed\ by\ finger\ f}{Total\ number\ of\ times\ letter\ l\ is\ typed}$$

For every input system fetches sets of alphabets and rearrange them based on occurrence of that alphabet by given finger so that most probable pressed alphabet with input finger comes first in set. At every fetches system create combination of alphabets except first fetch.

Input Observations:	F0,F1,…F7 , where F0,F1,…F7 are finger inputs from Arduino board L0, L1 and L3 where L0, L1 and L3 are Identified layer from Layer Min_Max Matrix
Output:	Set {C0,C1…Cn} = sorted(Lx) where Lx would be sets of possible pressed letter with their probability of being typed. The set is sorted in the descending order of the probability.

Table 2 : Finger Vs. Alphabet occurrence matrix operation

Input_1 : M RI //Middle layer, Right Index finger

Character set : ['h', 'j', 'k']

Input_2 : U LM // Upper layer, Left Middle finger

Character set : ['h', 'j', 'k', 'he', 'hr', 'hw', 'je', 'jr', 'jw', 'ke', 'kr', 'kw']

To reduce time complexity of character combinations, this possible combinations further rearrange based on current alphabets to following alphabets probability which extracted from transition matrix. This module of algorithm takes input as current possible pressed character and previous predicted character and gives count of how many times this combination has come in dictionary. This count is divided by total occurrences of that character to calculate probability for that sequence.

- **Transition Matrix:**

All the combinations with zero possibilities are eliminated and hence removed from the list of possible combinations. Remaining list is sorted in the descending order of the probability.

Character set : ['h', 'j', 'k', 'he', 'hr', 'hw', 'je', 'jr', 'jw', 'ke', 'kr', 'kw']

New Character set : ['he', 'ke', 'hr', 'je', 'hw', 'kr']

Input Observations:	C_{CUR} , C_{PRE} where C_{CUR} is current character and C_{PRE} is previous character typed by user.
Output:	N = Matrix[C_{CUR}][C_{PRE}] where N denotes occurrences of this combination of characters.

Table 3 : Transition Matrix Operation

New set of combinations again checks possible words from dictionary; which is already sorted according to words' frequency, and suggest to user. Current system only suggest top 10 most probable words from dictionary. If system doesn't find a word from dictionary, it auto corrects up

to two alphabets using two distance word correction algorithm. This algorithm will be executed only if user doesn't find word from dictionary. When user has pressed wrong finger to type word and get wrong combination of letters, this code can correct up to two letters. This algorithm can do delete, replace and insert operation to correct word. Moreover, it will fetch most probable word for given wrong spelled word.

6. Model Training

AIRtouch concept gadget runs machine learning technique to predict keystroke. Every machine learning process needs some set of training to follow. Our prediction algorithm predicts keystrokes and updates its elements to increase prediction accuracy. Considering this device as concept gadget, it follows certain assumptions in training phase to run device algorithm fluently. The training of the model becomes an essential part of the system, as the trained model is used later to predict characters when user is typing. In training phase device learn user's typing pattern based on following parameters: Finger which is used to type letter, Finger bending measurement, layer in which key resides and letter which is pressed. Training phase mainly rely on Finger, training letter which is pressed and bending measurement of finger. As user is typing on actual keyboard while getting trained layer parameter does not consider because keyboard layout is prefixed and location of each key (relation between key and layer) is known.

Assumptions: Assuming that the user is typing the correct letter while training the gloves. User is not lifting wrist after started typing. User has placed his hand within mentioned criteria. User typing by moving and bending finger not just changing palm position.

Limitations: Every keystrokes by the user will associate with each letter of training data. It won't change. User can not leave system before completing the full training.

Training process:

To train gloves user needs to teach his/her typing pattern to gloves. Before start training check below checkpoints:

- Hand is placed in marked area

- ❖ Arduino board is connected with computer
- ❖ Gloves has been tighten to user's hand.
- ❖ Make sure that hand movement without tapping activity is not giving any output.
- ❖ Hand and keyboard both are placed at same level or very negligible difference would be accepted.

- **Training data**

As we know machine learning deals with program learning from data sets. Training data is the data on which the machine learning program learn to perform correlation tasks. Training data sets and input data together we believe that construct predictive relationship between input data and output action. Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify apparent relationships in the training data that do not hold in general.(https://en.wikipedia.org/wiki/Test_set) Training data should be as closer to the actual output as possible. A well pre-labeled and impartial data will help trained classifier to perform more accurate prediction. Amount of training data set depends on complexity of the concept of predictive model. Data preparation is large subject that involves a lot of iterations, exploration and analysis. (http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/)

Our concept device made to predict character alphabets from human finger movement. As per training data set guideline our data should as close to our actual output. Initial training process would take this alphabets input and learn user typing keystroke dynamics by storing finger banding value, finger and alphabets in respective correlated table. As of now, concept device designed to predict all alphabets only, so our training dataset should cover all alphabets with enough frequency

to run this algorithm smoothly and more accurately. To cover all the alphabets, we decided to designed training text dataset from pangram sentences which covers all alphabets in meaningful sentences. A simple set of training data also gathered from typing practice websites which helps user to improve typing speed.[19] Some of are mentioned below, *"The five boxing wizards jump quickly"*

"Sixty zippers were quickly picked from the woven jute bag"

"Sphinx of black quartz: judge my vow."

We are using this data to learn typist's typing pattern. Module shows one by one character to user to type. At every tapping activity, system records typist's finger and finger bending measurement for each letter in training dataset. We already assumed that user is typing correct letter only while training gloves, we can determine layer information in which particular key resides. System only takes input when pressure sensor changes its input from 0 to 1, that process determine key pressing/ finger tapping activity of user.

We are taking input from finger bending at every 100 milliseconds for smoothing the data. Usually user presses one key for around 500 milliseconds to 900 milliseconds, so we are getting 5 to 9 inputs of bending angle for every key. For future purpose, we determine its maximum and minimum value among all sample data which will be used to set layer's minimum and maximum limit for particular finger. This information is stored in Layer Min_Max matrix.

- **Layer Min_Max Matrix**

Layer Min_Max Matrix is used to identify the layer information based on the flex sensor input. We have divided computer keyboard alphabets in three layers: (1) Upper layer, (2) Middle layer

and (3) Lower layer. A finger while used in different layers will give different flex sensor values. For example, when a finger is used to press a key in a middle layer will give higher flex sensor input than when the same finger is used to press a key in a upper layer. The same relation is observed between middle layer and lower layer. So, by saving the flex sensor values for all fingers for each layer, it is easy to identify layer using flex sensor input.

Each layer covers following alphabets:

- Upper Layer covers keys : ‘q’, ’w’, ’e’, ’r’, ’t’, ’y’, ’u’, ’I’, ’o’, ’p’,
- Middle Layer covers keys: ‘a’, ’s’, ’d’, ’f’, ’g’, ’h’, ’j’, ’k’, ’l’
- Lower Layer covers keys : ‘z’, ’x’, ’c’, ’v’, ’b’, ’n’, ’m’.

Layer Min_Max matrix is 3 x 16 matrix which contains data of possible minimum and maximum flex sensor input for each layer. By default we set all maximum values of layer Min_Max matrix to 2000 while all minimum values of layer Min_Max matrix to 0. One naive approach to organize this matrix is to replace current min/max value with new min/max value at each key stroke. But, this may produce overlapping layer min-max range. Therefore, we will not be able to know exactly in which layer the user had pressed the key. This problem would force our algorithm to consider two or more layers in consideration instead of one which weakens the system’s prediction model. To avoid this problem, layer min_max value is replaced by average of current layer min_max values and flex sensor input values. Very first entry of minimum and maximum value would compare 2000 and 0 respectively and following entries were taken from average of present value and current input values. This running average approach reduced overlapping problem.

Now, as our flex sensor output only ranges from 100 to 1200, if we find any maximum flex sensor

values to 2000 and minimum flex sensor values to 0 for any finger in any layer, we can safely assume that that particular finger wasn't used in that particular layer.

Table # depicts the layer Min_Max matrix for left hand fingers which contains minimum and maximum value of flex sensors while typing in particular layer by left hand fingers. While Table # shows minimum and maximum value of flex sensors while typing in particular layer using right hand fingers.

	F_min_0	F_max_0	F_min_1	F_max_1	F_min_2	F_max_2	F_min_3	F_max_3
L_0	571	629	474	810	463	752	474	857
L_1	401	658	618	846	506	879	521	973
L_2	582	662	774	857	615	890	589	1078

Table 4 : Layer Min_Max Matrix for left hand finger

	F_min_4	F_max_4	F_min_5	F_max_5	F_min_6	F_max_6	F_min_7	F_max_7
L_0	557	853	474	1295	155	984	387	600
L_1	701	987	557	1364	209	973	376	759
L_2	698	1157	1074	1327	2000	0	387	832

Table 5 : Layer Min_Max matrix for right hand fingers

Training Phase:

Input Observations: Q1,Q2,Q3,Q4,Q5…Qn where Q1,Q2,…Qn are flex sensor inputs from Arduino board.

Minimum = Qmin = min(Q1,Q2,Q3,….Qn)

Maximum = Qmax = max(Q1,Q2,Q3,….Qn)

Output:	$[Layer_x][Finger\_min_y] = min([Layer_x][Finger\_min_y] , Qmin)$ $[Layer_x][Finger\_max_y] = max([Layer_x][Finger\_max_y] , Qmax)$ where x € {0,1,2} that denotes the layer and y € {0,1,2,3,4,5,6,7} that denotes the fingers.

Table 6 : Layer Min_Max Matrix training operation

System also update one more matrix database which track how many times particular character has been pressed by particular finger. This finger vs alphabet occurrence matrix helps algorithm to understand user's typing behavior.

- **Finger Vs Alphabets occurrence matrix**

Finger vs Alphabets occurrence matrix is mapping matrix between letters and fingers. It is 27 x 10 sized matrix with labeled finger name in first row and alphabets in first column. Second column filled with layer information in form of '0','1' or '2'. '0' represent Upper Layer(UL) , '1' represent Middle Layer(ML) and '2' represent Lower Layer(LL).

The matrix contains data of which letter is how many pressed by which fingers. In the training phase, simply the matrix is updated at each key stroke. While the system is in use, this matrix is used to get possible list of letters pressed by the users based on layer data and finger data.

Here, l denotes the letter, which can have any value from 'a' to 'z'. f denotes the finger, that can be LL, LR, LM, LI, RI, RM, RR or RL. '0' represents that user has never used that finger to press given letter.

This matrix is updated in both phase, training phase and when device is in actual use.

Letters	Layer	LL	LR	LM	LI	RI	RM	RR	RL	Total Occurrences
q	0	0	7	0	0	0	1	0	0	8
w	0	0	23	0	1	1	1	0	1	27
e	0	1	0	61	0	0	2	2	3	69
r	0	0	0	2	28	1	1	0	0	32
t	0	0	0	0	46	1	0	0	1	48
y	0	0	0	0	2	12	6	0	1	21
u	0	0	1	0	0	11	17	1	0	30
i	0	0	0	0	0	1	34	5	3	43
o	0	0	0	0	2	2	22	43	0	69
p	0	0	0	0	1	0	2	11	0	14
a	1	36	10	0	0	1	0	0	3	50
s	1	1	31	3	0	1	0	0	2	38
d	1	0	0	8	10	0	0	0	1	19
f	1	0	0	0	14	0	0	0	1	15
g	1	0	0	1	10	2	0	0	1	14
h	1	1	0	1	0	33	1	0	0	36
j	1	0	0	0	0	9	0	0	0	9
k	1	0	0	0	0	0	14	4	0	18
l	1	0	0	1	2	0	2	27	0	32
z	2	5	3	0	0	0	0	0	0	8

x	2	0	2	6	0	0	0	0	0	8
c	2	0	0	8	10	0	0	0	1	19
v	2	0	0	0	14	0	0	0	0	14
b	2	0	0	0	0	15	1	0	0	16
n	2	0	0	0	2	34	4	0	2	42
m	2	0	0	0	0	21	2	0	1	24

Table 7 : Finger Vs. Alphabets Occurrence Matrix

Training Phase:

Input Observations: F0,F1,…F7 , where F0,F1,…F7 are inputs from Arduino board for fingers LL, LR, LM, LI, RI, RM, RR and RL respectively.

L0, L1 and L2 where L0, L1 and L2 are identified layer from Layer Min_Max Matrix.

C0,C1,….Cn , where C0,C1,….Cn are single character input from training data.

Output: FAM[Fx][Cy] = FAM[Fx][Cy] + 1 where x € {0,1,2,3,4,5,6,7} and y €{0,1,2,…n} and FAM is Finger Vs. Alphabet Matrix

Table 8 : Finger Vs. Alphabets Occurrence Matrix training operation

This describes relation between which finger user uses to presses particular key. In prediction model, algorithm uses combination of alphabets to predict word. To reduce complexity of combinations and reduce process time system maintain one more matrix which called transition matrix. This matrix doesn't get updated while training. It only uses in prediction process.

- **Transition Matrix**

To make prediction more powerful, we use probability of transition of alphabets in words. It is 26 x 26 sized matrix with considering labels alphabets from a-z in row and column both. This matrix shows how many time particular alphabets given the previous alphabet. The purpose of building such matrix is to predict next most probable alphabets based on current alphabet input and also to remove unwanted alphabets with zero probability. This matrix is already constructed based on words from a dictionary, which used to predict the words. This matrix also gets updated when device is in actual use to get more accurate prediction.

In training phase, the matrix is just updated at every key stroke as shown below:

Training Phase:

Input Observations:	C_{CUR} , C_{PRE} where C_{CUR} is current character and C_{PRE} is previous character typed by user.
Output:	$Matrix[C_{CUR}][C_{PRE}] = Matrix[C_{CUR}][C_{PRE}] + 1$

Table 9 : Transition Matrix training operation

Now we have all information about finger bending measurement to determine layer and finger-letter associativity to determine which key is pressed with probabilistic data, which are enough to determine user's typing behavior. To avoid errors and increase accuracy of prediction we built a word dictionary which used to predict English word user is typing. Once user complete training, system will be ready to use.

- **Dictionary**

The dictionary is used to predict final word based on possible combinations of letters. The

dictionary, which is used, is a pool of words collected from open source Shakespeare's work. The list of possible combinations is already sorted. So, one by one possible words for the particular combinations are extracted from the dictionary and shown to the user to make the final choice. And the whole process from layer Min_Max matrix is continued until the user finalize the word. Once, the user finalize the word all new list of possible combinations is emptied and the same process is repeated again.

7. Learning model

This concept device works with machine learning techniques. After getting trained machine has to learn user's typing behavior continuously to improve its prediction algorithm. This improvement depends upon finger - alphabet association. User will get most possible word list at every keystroke activity. The keystrokes entered in by the user is directly proportional to number of characters in the word. Where generically, the user will confirm the length of the word by tapping left thumb after keying in the desired number of character. He will only see words with exact length. Long pressing of left thumb will further allow user to switch between different choices of possible words provided to confirm the word.

Once user confirm the word, system updates following data:

1. It updates particular word frequency in dictionary so that next time that word comes first with given combinations of alphabets set.

2. It updates Viterbi matrix, helps to predict best alphabets combinations.

3. When concept device is in use, it records user's finger data until user confirm the word so that system can update alphabets occurrence for associate finger in finger Vs. Alphabet occurrence matrix.

Example :

If keystroke activity is ['LI', 'RM', 'RI', 'LM'] and confirm word is ['find'] then system will update following parameters:

1. Dictionary : frequency('find') = frequency('find') + 1

2. Viterbi matrix :

Matrix[‘f’][‘i’] = Matrix[‘f’][‘i’] + 1

Matrix[‘i’][‘n’] = Matrix[‘i’][‘n’] + 1

Matrix[‘n’][‘d’] = Matrix[‘n’][‘d’] + 1

3. Finger Vs. Alphabet occurrence :

FAM[‘LI’][‘f’] = FAM[‘LI’][‘f’] + 1

FAM[‘RM’][‘i’] = FAM[‘RM’][‘i’] + 1

FAM[‘RI’][‘n’] = FAM[‘RI’][‘n’] + 1

FAM[‘LM’][‘d’] = FAM[‘LM’][‘d’] + 1

8. Model Evaluation

Training is in process … Will update data once training has been completed.

9. References (Unformated references – will update it once reference all text in thesis)

[1] http://dev2.blueorb.me/wp-ontent/uploads/2014/12/orbitouch_white_background_1024x1024.jpg

[2] http://orbitouch.com/

[3] http://www.yankodesign.com/2008/06/12/lights-camera-glassaction/

[4] http://gizmodo.com/5015723/no-key-glass-touch-keyboard-is-antithesis-of-steampunk/

[5] https://en.wikipedia.org/wiki/Projection_keyboard

[6] http://www.uncommongoods.com/images/items/24900/24904_1_1200px.jpg

[7] https://www.kickstarter.com/projects/1761670738/ring-shortcut-everything

[8]https://ksr-ugc.imgix.net/assets/001/684/308/5a26f009a1e10b8058810ada74289f40_original.png?w=680&fit=max&v=1393497222&auto=format&lossless=true&s=aacd298f5ce26ccd9cdff0d4acc109cc

[9] http://www.tapwithus.com/

[10] http://cdn.hiconsumption.com/wp-content/uploads/2016/05/Tap-wearable-Keyboard-01.jpg

[11] https://gest.co/

[12] https://www.kickstarter.com/projects/asdffilms/gest-work-with-your-hands

[13]https://cdn0.vox-cdn.com/thumbor/0_0gAXUOI5KpDxp3ksaCfqhgI8g=/800x0/filters:no_upscale()/cdn0.vox-cdn.com/uploads/chorus_asset/file/4208015/DSC_4261.0.JPG

[14] http://photonics.cusat.edu/Research_Fiber%20Sensors_work%20at%20ISP.html

[15] http://obrazki.elektroda.pl/2733383600_1383139811.png

[16] https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide

[17]https://arduino.stackexchange.com/questions/10041/can-i-connect-a-pwm-pin-on-one-arduino-to-an-analog-input-on-another

[18] (https://www.arduino.cc/en/Main/FAQ

[19] https://10fastfingers.com/text/119-A-simple-Paragraph-to-practice-simple-typing)

[20] http://www.rinkworks.com/words/pangrams.shtml