***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.*

# Introduction

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.

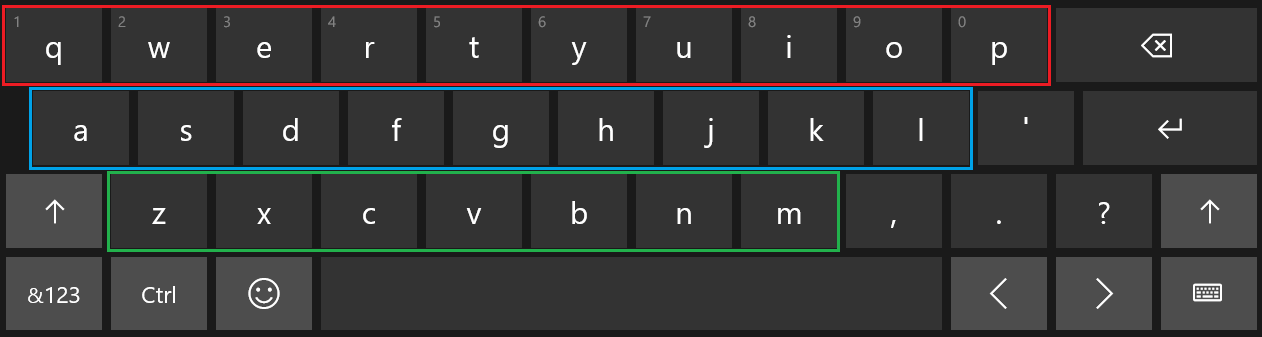
The main objective of this report is to design gloves with motion sensor, pressure sensor along with an algorithm to associate keys with finger motion and to design an algorithm which makes use of the training data to omit resultant keys.

# **Description**

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

## Hardware Work

The key challenge in creating the AIRtype 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.

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 

**Figure 1: Three different layers of keyboard**

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. Considering the vision of future development in this design, open source development kit 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.

Two sensors are needed to fulfill the goal, one to sense the pressing activity and other for determining motion activity. Therefore, Sensor gloves are made with two sensors: Flex sensor and Force Sensitive Sensor.

### 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.

By passing current from one end and measuring it from the other end, resistance can be calculated.



**Figure 2 : Flex Sensor \*\***

### Force Sensitive Sensor

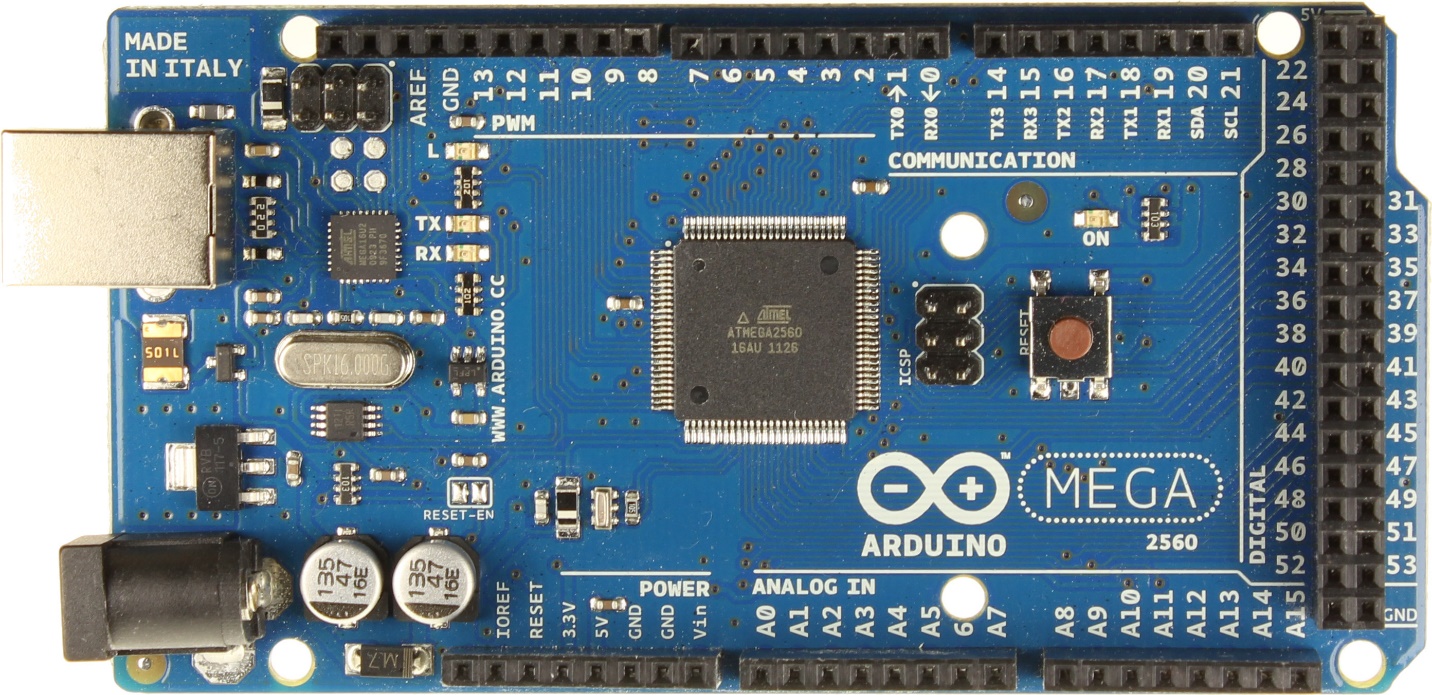
The purpose of Force Sensitive Resistor also known as FSR is to measure amount of force or pressure. It contains conductive polymer which are used in Bend Sensors. The conductive polymer changes resistance when force is applied on the sensor. Here, this sensor is used to detect finger tapping considering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



**Figure 3 :Force Sensitive Resistor \*\***

### Arduino Board

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.



**Figure 4 : Arduino Uno \*\***

### Gloves construction:

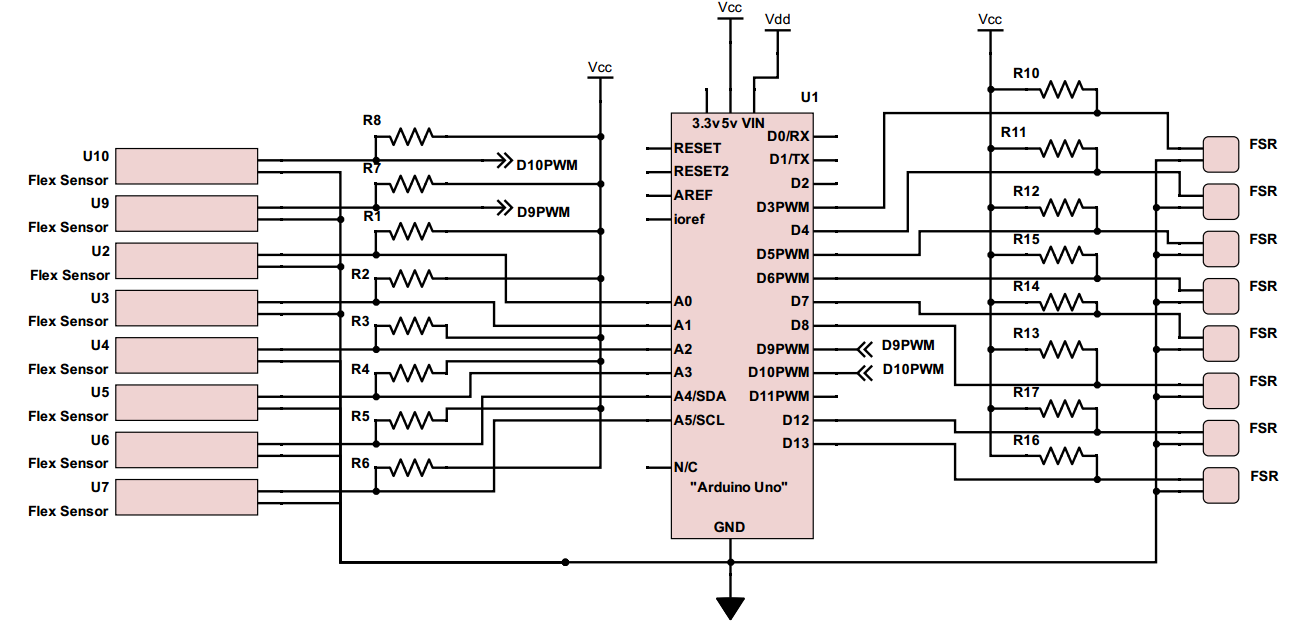
To make the gloves easy to wear and more comfortable for the user and knowing that ease of use would be of prime importance, few models were made to experiment with various size and distinctive material of gloves. Different permutation and combinations were done. Several materials like cotton, rubber 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 # presents the gloves along with their connection to arduino board.



**Figure 5 : Sensor gloves and arduino board \*\***

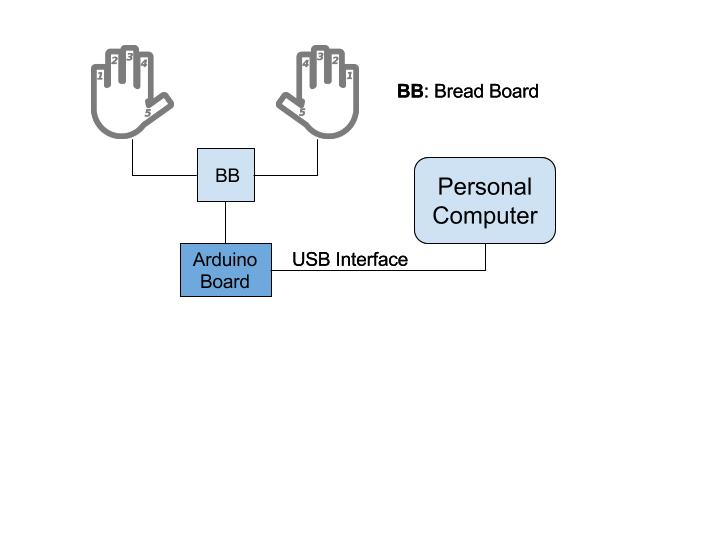
### Hardware Schematic Diagram

Figure # shows schematic of gloves design, Resistance value of R1-R8 is 22K Ohm and R10-R17 is 10K Ohm. As a result of several trial and errors, I concluded these value of resisters to get desired result.



**Figure 6 : Schematic diagram**

## Top Level Block Diagram



**Figure 7 : Top level block diagram**

# Software:

I developed two code 1) Arduino code and 2)Python code (Currently working on) and used software named ‘PLX-DAQ’ - Parallax Data Acquisition Tool.

## Arduino Code

The code is written to collect data from sensors in Arduino language. The coding structure of Arduino always consists two parts: Setup function and Loop function.

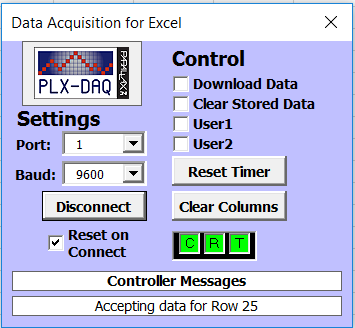
**Setup Function**: This function initializes all the pins as per the requirements. It also initializes variables which are used in Loop function.

**Loop Function**: This function runs in loop and continuously send data to computer on UART interface.

## PLX\_DAQ

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 8 : PLX-DAQ tool**

## Algorithms

### Decision tree

Decision tree is decision support algorithm which uses branch and tree structure to portray possible outcomes. It constructs classification model in form of tree structure. Every node of a decision tree is divided into smaller subsets and design an associated decision tree with incremental development. Final tree carries tree nodes and decision nodes. One of the approaches to implement decision tree is ID3 algorithm.

Construction of decision tree follows top down approach from root node and imply classification of data into its subset that contain different values. ID3 algorithm uses entropy of attributes and collect information based on calculated entropy and target values.

### Markov chain Model

Markov chain introduced by Andrew Markov, is an algorithm that calculates probabilistic model of system which changes system state over a time. Markov chain follows the principle of ‘Memoryless property’, a property of certain probability distributions. Using this model one can predict the probability of future states solely based on the current state instead of sequence of states. This nature of algorithm makes it easy to calculate conditional probability that is to be applied in numerous applications. Markov chain records previous state changes and calculate conditional probability transition matrix to predict future state transition probability. Markov chain provides run time training features. Algorithm can change transition probability at runtime based on current data input. This feature will help AIRtype to get train from users while users are using it.

### Selection of Algorithm

As decision tree is constructed based on entropy of attributes of a system, it is harder to update decision tree when the probability of any attribute changes in real time. This forces the system to reconstruct a decision tree every time the probability of any attribute changes. While Markov chain helps to overcome this redundant reconstruction activity and makes it faster by chaining particular probability only. That is why, I decided to use Markov chain model over the decision tree.

## Implementation of Markov chain for design

This design depends on the correlation between key, in which layer key resides and which finger is pressing that key. Process of determining key includes below three steps:

# First determine which finger is pressing the key

# Examine the layer in which finger is being pressed

# Collect set of keys pressed by that finger with their probability

To predict which key is being pressed, we need correlation between keys and fingers. Markov chain helps to construct transition matrix of finger vs keys. Hence, we decide to make three transition matrix (One per layer) that contains probability of key pressed by certain fingers. Figure # depicts transition matrix populated during the training mode.

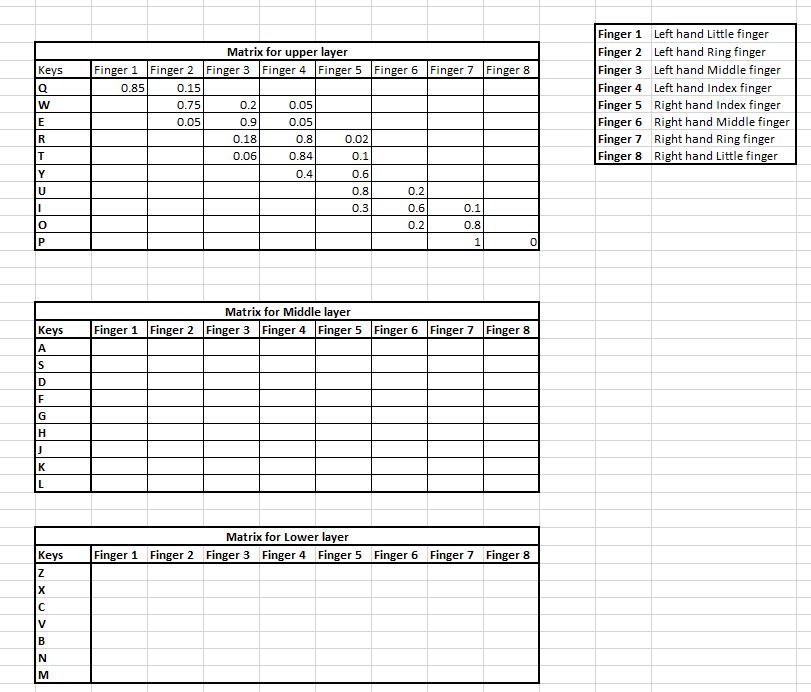


Figure : Transition Matrix

Viterbi Algorithm:

The Viterbi algorithm is a [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the most [likely](https://en.wikipedia.org/wiki/Likelihood_function) sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of [Markov information sources](https://en.wikipedia.org/wiki/Markov_information_source) and [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_model).(wiki). Markov process which is finite-state discrete-time process comes under compound decision theory, which decides probability of upcoming event based on observations.

In this product, our observation sequence is alphabets\*\* in word, and our desired output is to predict what would be the next alphabet based on previous alphabets typed in a word.

\*\* remove Viterbi and come up with mathematical computation of your own algorithm.

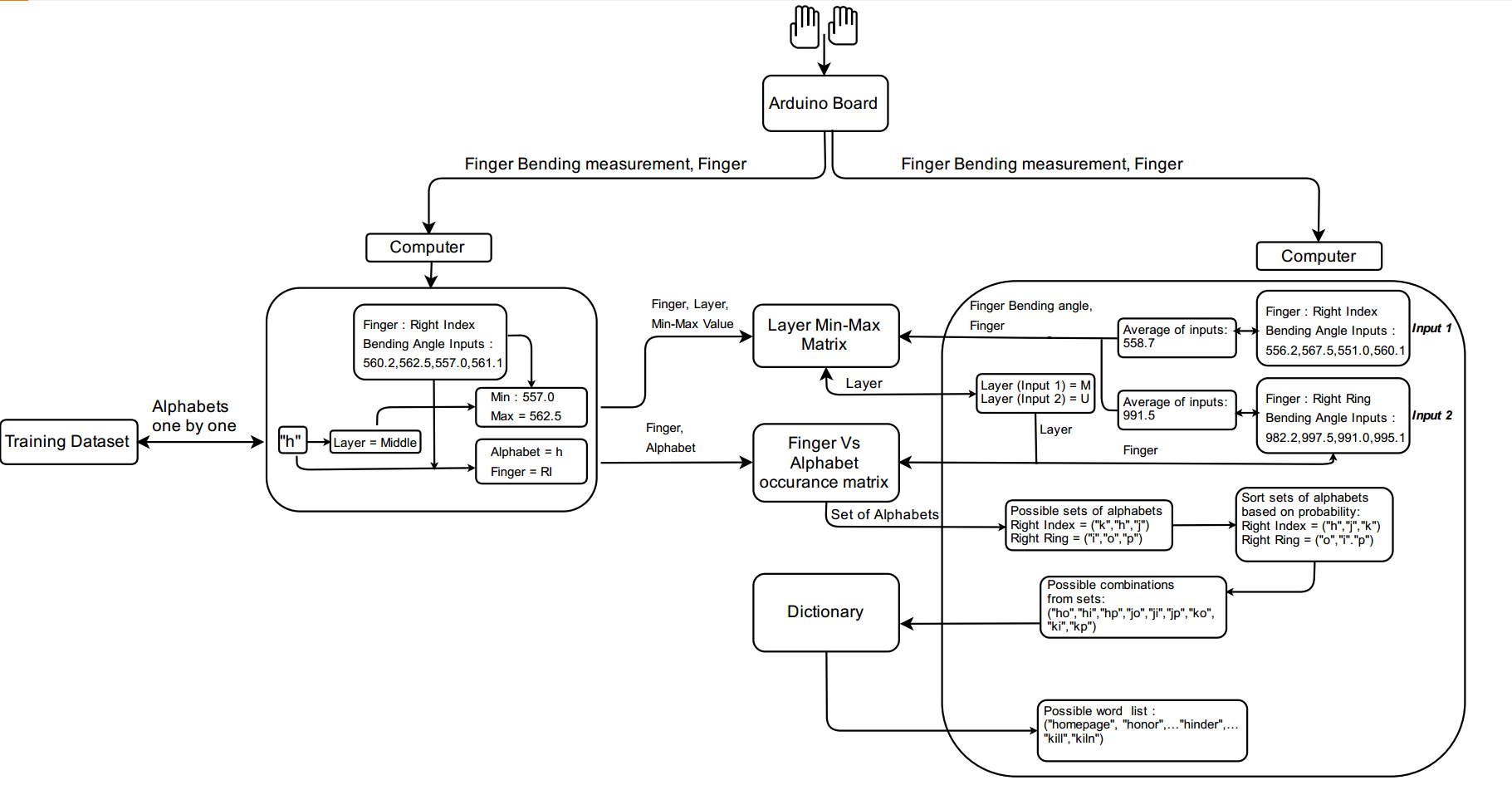
Training Device:

Certain assumptions were made while designing training phase to run device algorithm fluently. As this device is using user’s typing pattern to predict alphabets while user use this device, so training becomes an essential part of this process. In training phase device learn user’s typing pattern based on following parameters :Finger which is used to type letter, Finger bending angle, layer in which key resides and letter which is pressed. Training phase mainly rely on Finger, training letter which is pressed and bending angle 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.

Training process:

To train gloves user needs to teach his/her typing pattern to gloves. We designed training dataset which contains all alphabets in various combination of words. This data was made from pangram sentences (pangram words – sentence which contain all 26 letters of alphabets) which used to test out new typewriters. We are using this data to learn typist’s typing pattern. We are recording typist’s finger and finger bending angle for each letter in training dataset. As we already assumed user is typing correct letter only while training gloves, we can determine layer in which particular key resides. System only takes input when pressure sensor changes it’s input from 0 to 1, that process determine key pressing activity of user. At key pressing activity, algorithm takes finger bending input from particular finger only. Which helps device to determine/set layer determine parameter while system is in use. We are taking input from finger bending at every 100 milliseconds for smooth data. Usually user presses one key for around 500miliseconds to 900 miliseconds, so we are getting 5 to 9 inputs of bending angle for every key. To smooth it we determine it’s maximum and minimum value among all sample data which will be used to set layer’s minimum and maximum limit for particular finger. One 3\*16 matrix is maintained which stores minimum and maximum layer limit for each finger. System also update one more matrix database which contain data of finger vs alphabet, which helps system to understand user’s typing behavior. It describes relation between which finger user uses to presses particular key. It will associate sets of letters to particular finger with probability to press that keys with the finger. Now we have all information about finger bending angle to determine layer and finger-letter associativity to determine which key is pressed with probability data, which are enough to determine user’s typing behavior. Once user complete training, system will be ready in use.

\*Update block Diagram once you complete writing



Device In Use:

After device got trained, we can put this device in use. This device runs over certain limitations and assumptions. This device learns user’s typing pattern to predict key value based on user’s finger movements. User must have predictable typing pattern; random typing pattern may lead this device to an unpredictable state. In training part algorithm capture finger bending angle range for each keyboard layer. To predict correct layer user can not 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.

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

Pre-req : Layer Min-Max matrix, Finger VS. Alphabets occurrence matrix, Viterbi transition matrix – already updated letter transition based on dictionary words, premade word dictionary

When system is in “Device in Use mode”, connect gloves with Arduino board and connect Arduino board with Computer on COM1. Arduino transmit information in serial data form, serial.py module receive data from serial port COM1 and separate information about finger and finger bending angle. System derive layer information from Layer Min-Max matrix. Now system has three parameters: Finger name, Finger bending angle and Layer Information. Among those parameters finger and layer information used to predict possible alphabets set from Finger Vs. Alphabet occurrence matrix. From every input system fetch sets of alphabets and rearrange based on occurrence of that alphabet by input 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. Now possible combinations again rearrange based on current alphabets to following alphabets probability from Viterbi matrix. New set of combinations again checks possible words from dictionary and suggest to user. If system doesn’t find a word from dictionary, it auto corrects up to two alphabets using two distance word correction algorithm. 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. System records finger inputs until user confirm the word so that system can update alphabets occurrence for associate finger in finger Vs. Alphabet occurrence matrix.

**Training data**

Every machine learning algorithm needs sets of data, which is called training dataset. Training data sets and input data together we believe that construct predictive relationship between input data and output action. 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.

Our algorithm required training data sets of alphabets as we are predicting alphabets. Concept device designed to predict all alphabets only, so our training dataset would cover all alphabets with enough frequency. We designed training text dataset from pangram sentences which covers all alphabets in meaningful sentences.

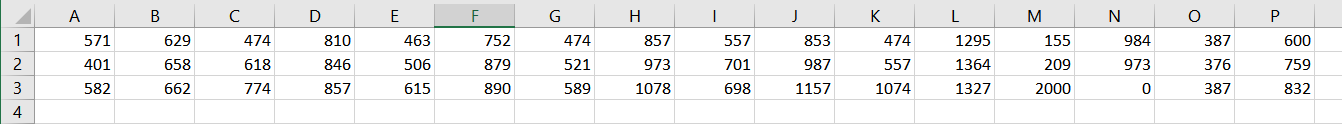
*“The five boxing wizards jump quickly”*

*“Sixty zippers were quickly picked from the woven jute bag”*

*“Sphinx of black quartz: judge my vow.”*

**Layer Min\_Max Matrix**

As discussed in setup procedure, we divided computer keyboard alphabets in three layers : Upper layer (keys:‘q’,’w’,’e’,’r’,’t’,’y’,’u’,’I’,’o’,’p’), middle layer (keys: ‘a’,’s’,’d’,’f’,’g’,’h’,’j’,’k’,’l’) and lower layer (keys: ‘z’,’x’,’c’,’v’,’b’,’n’,’m’). This 3 X 16 matrix contain data of flex sensor input for each layer row wise. Left part of half matrix (3X8) contains minimum value of flex sensors while typing in particular layer by fingers. Remaining half part contains maximum value of flex sensors for the same process. By default we set all values to 2000 because our flex sensor only goes from 100 – 1200, that means after training if we find any two values 2000, we can conclude non used finger in particular layer.



**Finger Vs Alphabets occurance metrix**

Our algorithm selects most probable alphabet for that finger in that layer. To achieve this goal, matrix is constructed by monitoring occurrence of alphabets by finger and updating it in format like alphabets in row and finger in columns. This matrix updates in both phase training phase and when device is in actual use. It is 27X10 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)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LETTER | LAYER | LL | LR | LM | LI | RI | RM | RR | RL | Total\_Occurances | |
| 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 |  |

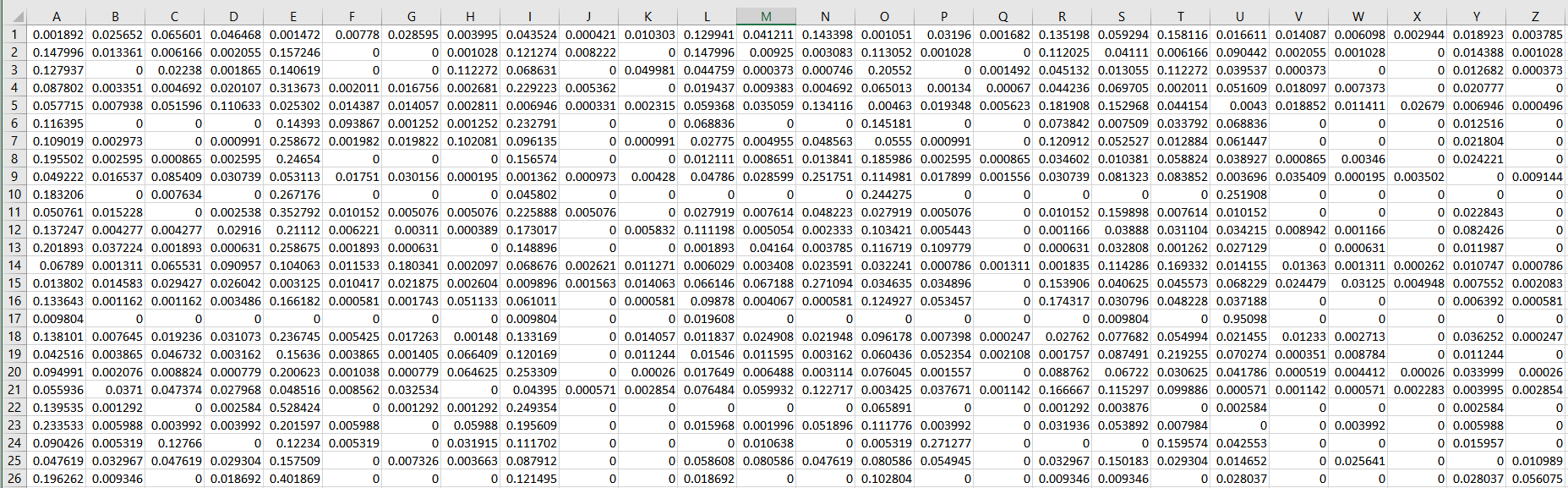
**Transition Matrix**

To make prediction more powerful, we use probability of transition of alphabets in words. It is 26X26 sized matrix with considering labels alphabets from a-z in row and column both. This matrix contains occurrences of alphabet to the following alphabet. Purpose of building this matrix is to predict next probable alphabets based on current alphabet input and also remove unwanted zero probability alphabets.

This matrix is already constructed based on words from a dictionary, which will be used to predict the words. This matrix also gets updated when device is in actual use to get more acuurate prediction. (Refer Image on Page#18)

**Dictionary**

One dictionary is maintained to predict final word based on input alphabets. This is pool of words collected from open source Shakespeare’s work. Concept device only predict words from dictionary only.



(<http://www.rinkworks.com/words/pangrams.shtml> (pagramword))