ABSTRACT

AIRTOUCH KEYBOARD – INTELLIGENT REMOTE COMPUTER KEYBOARD

By

Kuldeep D. Luvani

August, 2017

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 AIRtouch keyboard, which would take the typing technology one step further by doing away the need for the 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 categories 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.

ACKNOWLES

CHAPTER 1

INTRODUCTION

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 takes it to a new level with the 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 has their own unique typing pattern, with some using their Index finger to type most of the characters while others use all of their fingers to type. This correlation between fingers and the keys is known as the Keystroke Dynamics. (University 2016) Every user will have his own unique Keystroke Dynamic. (Anna Maria Feit n.d.) 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 a physical keyboard.

Problem Statement

The main objective of this report is to design gloves with a 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.

Keyboard Technology

In an 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 used. He could not have known that his design persists for 150 years even in digital world also. In this rapid change of high technology, the computer keyboard is undoubted king of computer input device.

Some recent technology started taking place of keyboard and requirement of the 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 eliminate the need of physical key describes below.

Orbital Keyboard Technology

The 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 directions in its orbit.

FIGURE 1.1. Orbital keyboard technology. (Orbitouch n.d.)

To type a letter, the user needs to slide right dome to that letter and left dome to its corresponding color irrespective to which dome moves first. This keyboard is well used for a person with low vision, Cerebral palsy, and other disabilities. (Orbitouch n.d.)

No-key Keyboard

No-key Keyboard is concept keyboard designed by Kong Fanwen. This keyboard's 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 in line with the light beam. The camera senses user's contact with the glass and sends appropriate information to PC. This ultimate motion capture technology design is waterproof also. (JAMES 2008)

FIGURE 1.2. No-key keyboard. (JAMES 2008)

Projection Keyboard

Engineers at IBM patented optical virtual keyboard in 1992. This keyboard takes human finger movement input by a camera mounted on the device and 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 a virtual keyboard layout on a surface. One camera or sensor has been mounted on this device to track down movement of the fingers. They use second infrared based layer just above the surface to track down keystrokes action. When this sensor breaks or gets inference in the layer, it tracks down finger coordinates to predict key. (WikiPedia, Projection Keyboard 2017)

FIGURE 1.3. Projection keyboard. (Uncommon Goods n.d.)

The Ring

The ring is a wearable input device that allows the user to control. This device allows the user to use gesture control, text-based typing, and alerts. It uses letter recognizing software to recognize letter by the 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. (Logbar n.d.)

FIGURE 1.4. The Ring (Logbar n.d.)

Tap

Tap is a wearable input device that allows the 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 characters. Like tapping one finger will give best-fit vowel and combination of other fingers throw defined character.

FIGURE 1.5. Tap keyboard technology (TAP n.d.)

The 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. (TAP n.d.)

Gest

Gest is a wearable keyboard that allows the user to control computer or mobile with hand gestures. (Gest, Gest Homepage n.d.) It uses IMU(inertial measurement unit) to track down finger movement. It is also used for designing, any gesture based 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. (Gest, Gest - Work with your hands n.d.)

FIGURE 1.6. Gest keyboard (Gest, Gest - Work with your hands n.d.)

CHAPTER 2

Gloves design

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

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 a specific focus on bending and tapping sensors. Bending sensor will provide the angular position of a 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 fingers 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.

FIGURE 2.1. Keyboard layout is divided into three layer.

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 to 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. The red rectangle indicates Upper Layer(UL) of the keyboard, a blue rectangle indicates Middle Layer(ML), and green rectangle indicates Lower Layer(LL). 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 the motherboard. A rapid evolution in controller technology makes a selection of controller harder. Merging functionality and faster processing times are fundamentals of the controller. Teensy USB development board, MSP430 emulation board, beagle bone A6 and Arduino are a 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 the 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 we are getting as inputs, one is finger bending measurement

and another is finger tapping activity. These sensors should work with low power and higher efficiency. As we are designing wearable gloves, sensors need to be flexible and compact enough to fit over a finger. After rigorous research, we selected pressure sensor to sense the pressing activity and flex sensor for determining motion activity.

Bending Sensors

Microbending sensors. Microbend sensors are based on coupling and leakage of modes that are propagating in a deformed fiber. Usually one achieves this deformation by employing corrugated plates that deform the fiber into a series of the sharp bend with small bending radii. Scientists have developed a highly sensitive chemical sensor by inducing permanent microbeads on a bare plastic optical fiber. The output intensity is linearly dependent on the logarithm of the concentration of the absorbing species surrounding the bent portion of the fiber.

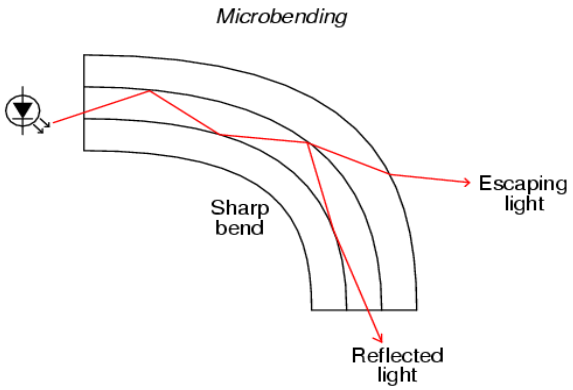

FIGURE 2.2 Micro binding Sensors (Obrazki.electroda n.d.)

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 micro bending sensor can be used to detect different chemical species. (Photonics n.d.)

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 a flexible layer of polymer. As sensor bends, it stretches flexible ink inside, extending itself, which results in a reduction of cross section area of the ink layer. As the sensor flex, the 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. (Jimbo n.d.)

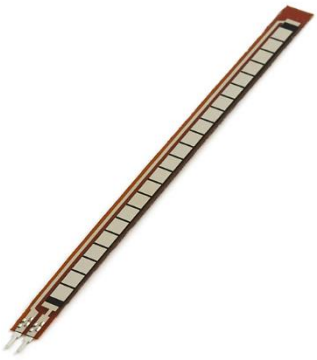

FIGURE 2.3. Flex sensor (Jimbo n.d.)

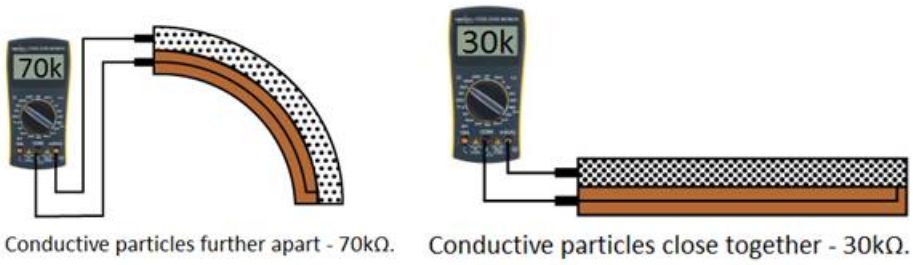

FIGURE 2.4. Flex sensor measurement (Jimbo n.d.)

The simplest way to implement this circuit in my project is using voltage divider circuit, where we pass current from one pin and measure the voltage from another end. Based on input current and output voltage with input voltage value we can calculate resistance value using below equation.

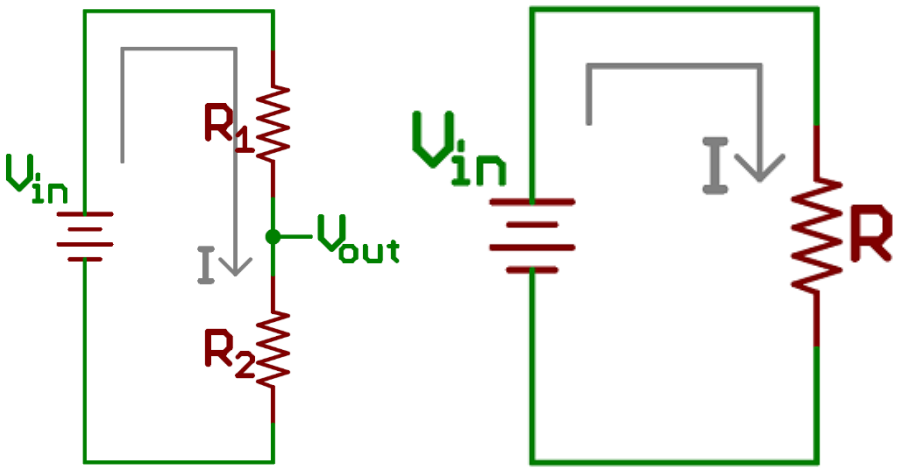

FIGURE 2.4. Voltage divider circuit.

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. (Jimbo n.d.)

Taking effective total resistance R = R1 + R2

```
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 2.5. Flex sensor readings.

Current $I = V_{in} / R$

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

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

An example of reading from flex sensors recorded while testing the device is shown in Figure 2.5.

Force Sensitive Sensor

The purpose of Force Sensitive Resistor, which is also known as FSR, is to measure the amount of force or pressure. It contains a conductive polymer which is used in Bend Sensors. The conductive polymer changes resistance when force is applied to 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 the market, as per my thesis requirement, I want a sensor that fits on a fingertip and also it should be printed on 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. (Sparkfun n.d.)

FIGURE 2.6. Force sensitive resistor. (Sparkfun n.d.)

Arduino Board

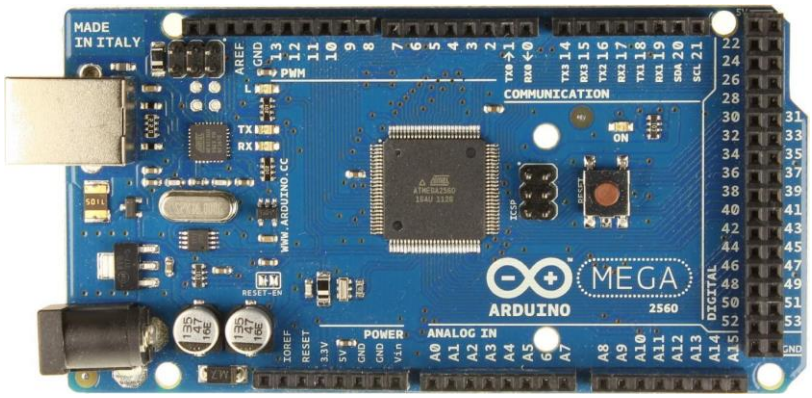

FIGURE 2.7. Arduino Mega. (MARIAN n.d.)

Arduino is a company that designs and develops 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 the 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 the 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 a deal on Arduino Mega board from craigslist. Due to budget constraints, we must go

with the cheaper option. (McMahon 2015)

Gloves construction:

To make the gloves easy to wear and more comfortable for the user and know 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' less adaptability and sensors must slide on the material instead of stick on it. Flex sensors were placed on top of fingers with a sliding mechanism on proximal phalanx of fingers and fixed from the tip of fingers.

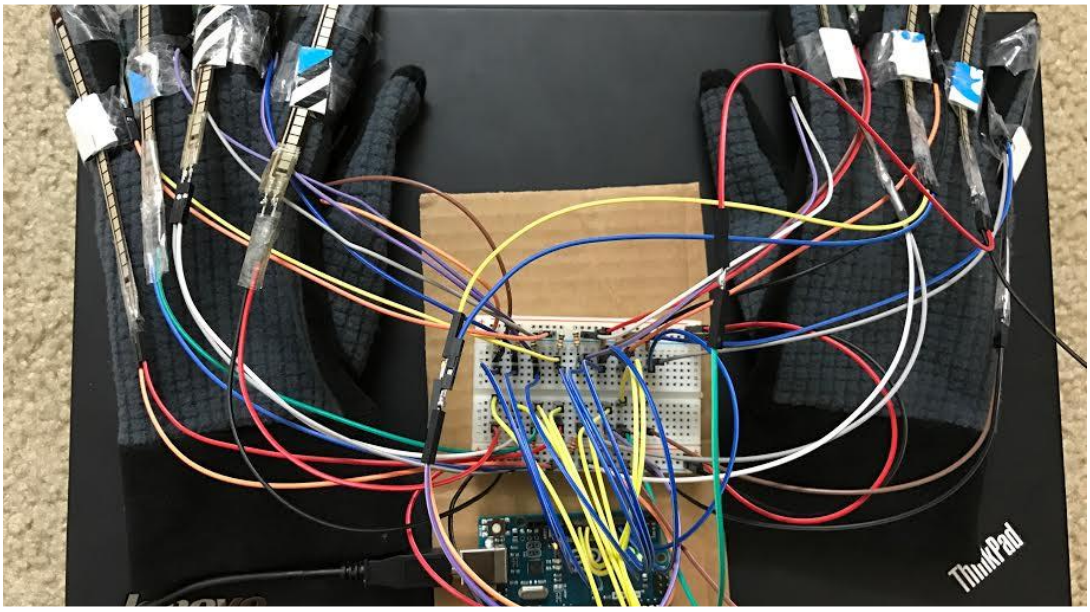

FIGURE 2.8. Sensor gloves and Arduino board.

Figure 2.8 presents the gloves along with their connection to Arduino board. Wires from sensors plugged into the breadboard 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 a respective resistor and then further extended to Arduino board pins.

Hardware Schematic Diagram

Figure 2.9 shows schematic of gloves design, Rectangular box represent flex sensor component and the square symbol represents 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.

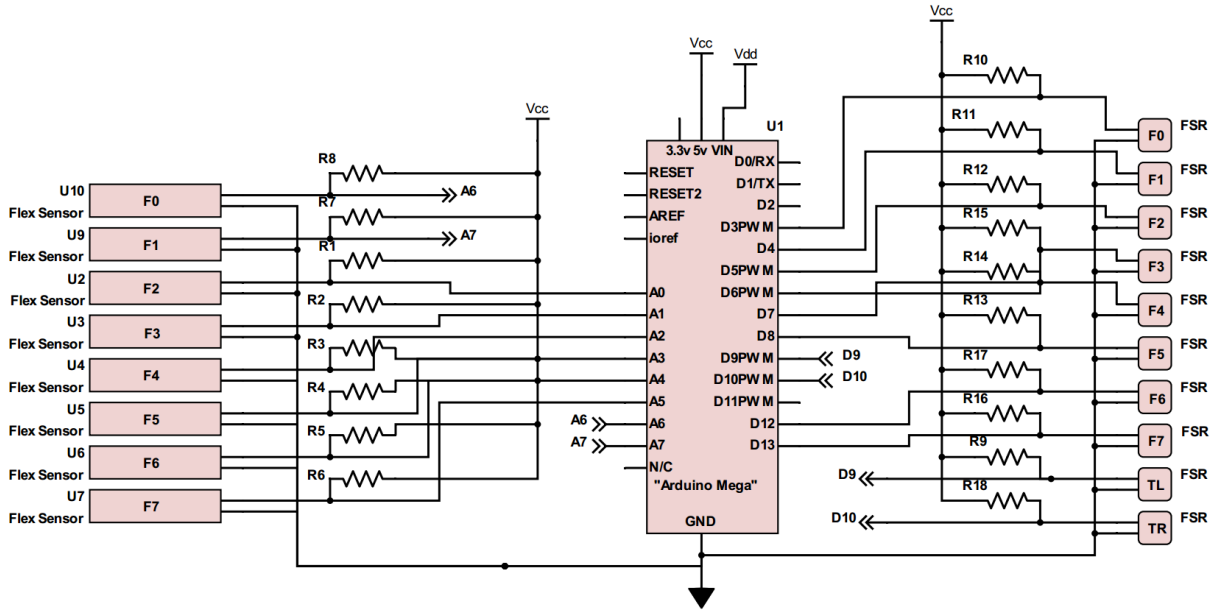

FIGURE 2.9. Schematic diagram.

Whereas Force sensitive resistors denoted Fx : x € {0,1, … 7} represents respective finger & TL and TR represents left-hand thumb and right-hand thumb respectively, connected to digital input pins of Arduino board with 10K pull-up resistors. The 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 the desired result. U1 symbol represents for Arduino Mega board, which created with only useful pins only. Unnecessary pins are not shown in the 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. The power consumption of this circuit is very nominal(not more than 20 mA at 5V supply).

Top Level Block Diagram

Figure 2.10 represents a top level block diagram of our design.

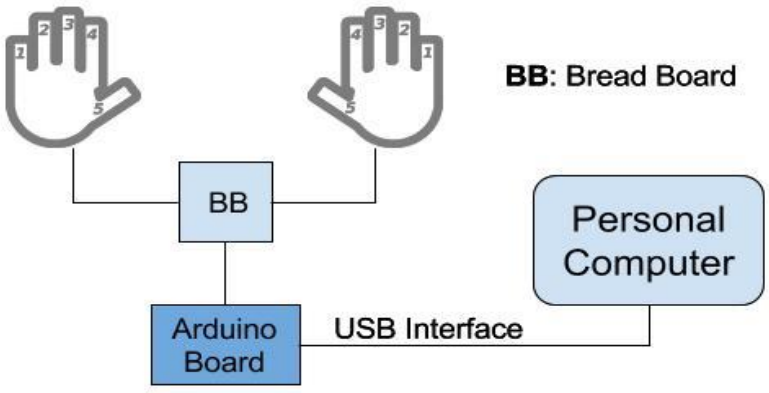

FIGURE 2.10. Top level block diagram.

Two hand gloves are separately connected to a 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 a lot of component and hardware work where as USB interface only require USB compatible cable and it also provides power supply to Arduino board. So it does not require to provide power to Arduino board with external power supply.

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 the computer has USB type-A jack connector.

Software:

The following softwares are used to execute the program.

- 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. (Arduino n.d.)The code is written in Arduino language to collect data from sensors and

to send this data to the 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 the setup function.
- ➢ **Loop Function**: After creating setup function, which initializes values and libraries, the loop function runs in a loop and continuously allow the 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 sends data to a computer in comma separated format using USB interface.

PLX_DAQ

PLX-DAQ, Parallel data acquisition tool, used to get data from sensor based microcontroller 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 2.5 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.

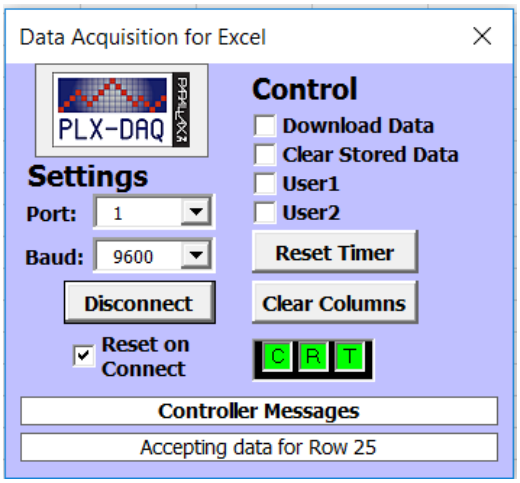

FIGURE 2.11. PLX-DAQ tool.

Python code for data acquisition

PLX- DAQ is a good tool for sensor based microcontroller system. Our system is 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 the better option of using Python as the programming language has a pool of libraries that interact with hardware also. Python library pySerial encapsulate the access to the serial port. (Pythonhosted n.d.)This library allows the 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 the pySerial library, we used serial.Serial() function

which initializes the port value, baud rate, bytesize, parity, and delay.

ser =serial.Serial(port='COM1', baudrate=9600, bytesize=serial.EIGHTBITS ,

parity=serial.PARITY_NONE, timeout=100)

```
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 2.12. Input data readings.

Generally, we set the baud rate to 9600 bits per second where speed isn't critical criteria. (JIMB0 n.d.) We are reading ASCII value as input and ASCII defined in the 7-bit character set. While reading sensors input, it gives a bunch of readings at every tapping activity. So timeout feature gives timeout error if it does not see any activity in 100s.

Figure 2.12 depicts output data from Arduino board, first integer number represents finger value and second represents bending measurement value. Entire device's learning process and prediction process flow chart shown in below image.

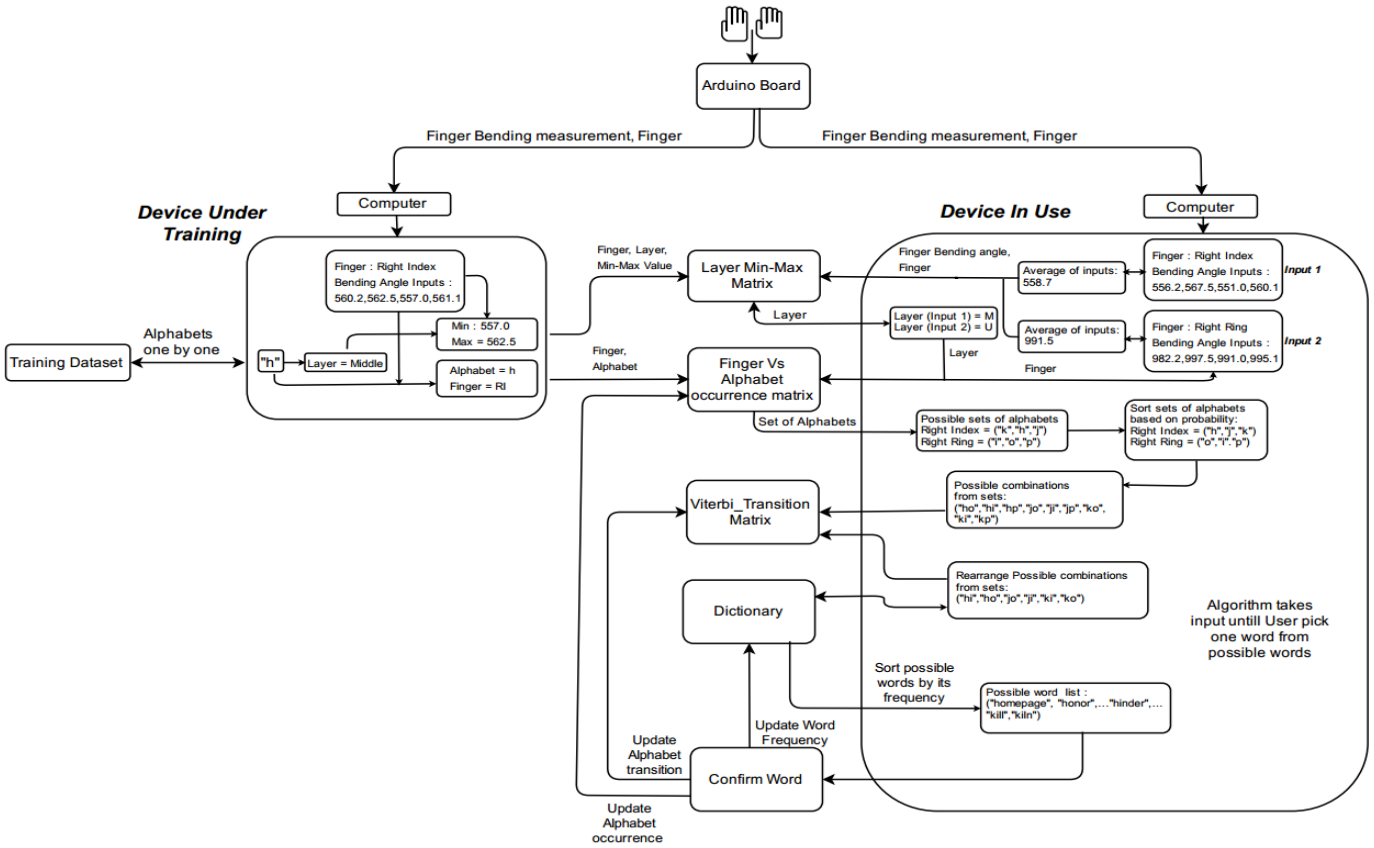

FIGURE 2.13 Model flow chart

CHAPTER 3

Model Training

AIRtouch concept gadget runs machine learning techniques 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. Assuming that the user is typing the correct letter while training the gloves. The user should not lift his/her wrist after started typing. The user has to place his hand within mentioned criteria. The user should type letters just by moving and bending fingers. The training of the model becomes an essential part of the system, as the trained model is used later to predict characters when the user is typing. In training phase, device learns user's typing pattern based on following parameters: Finger which is used to type a letter, Finger bending measurement, layer in which key resides and letter which is pressed. Training phase mainly relies on Finger, training letter which is pressed and bending measurement of a finger. As the user is typing on the actual keyboard while getting trained, layer parameter is not considered because keyboard layout is pre-fixed and location of each key (relation between key and layer) is known. The limitation of the system is that the user can not leave the system before completing the full training.

Training process:

To train the model, the user needs to teach his/her typing pattern. Before starting the training, check the following points: User's hands must be placed in pre-marked area, Arduino board is connected with the computer using USB port, make sure that the gloves are comfortably

fit to user's hand, make sure that any activity other than tapping activity is not giving any kind of input to the system, hand and keyboard both are placed on the same level or very negligible difference would be accepted. Once the system is set up in the condition mentioned above, the user can start the training.

Training data

As we know, machine learning deals with learning techniques from training data sets. Training data is the data on which the machine learning program learns to perform correlation tasks. We believe that training data sets and input data together construct the predictive relationship between input data and output action. Most approaches that search through training data for empirical relationships tend to overfit the data, that is, they can identify apparent relationships in the training data that do not hold in general. (WikiPedia, Test Set n.d.)Training data should be as close to the actual output as possible. A well pre-labeled and impartial data will help trained classifier to perform more accurate predictions. The amount of training data set depends on the complexity of the concept of predictive model. Data preparation is a large subject that involves a lot of iterations, exploration, and analysis. (Brownlee 2013)

Our concept device is made to predict character alphabets from human finger movements. The initial training process will take these alphabets as an input and learn user's typing keystroke dynamics by storing finger bending measurements, fingers, and alphabets in the respective correlated table. As of now, concept device is designed to predict all the alphabets only, so our training dataset should cover all the alphabets with enough frequency to run this

algorithm smoothly and more accurately. To cover all the alphabets, we decided to design training text dataset from pangram sentences which covers all the alphabets in meaningful sentences. (rinkworks n.d.)A simple text paragraph is also included in training dataset collected from typing practice website. (10fastfingers n.d.) Some of the pangram sentences 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. The module shows one by one character for the user to type. At each tapping activity, the system records the typist's finger and finger bending measurement for each letter from the training dataset. We already assumed that the user is typing correct letter only while training the gloves, we can determine layer information in which particular key resides, as we know which key user has pressed. The system only takes the input when pressure sensor changes its state from 0 to 1. By default, pressure sensors' state is 0, which changes to 1 when any tapping activity is detected.

The default interval of input from flex sensors is 1 millisecond. However, we are taking input from flex sensors at the interval of 100 milliseconds for smoothing the data. Usually, the user presses one key for around 400 milliseconds to 600 milliseconds, so we are getting 4 to 6 inputs of bending measurement for each keystroke. For the future purpose, we determine its maximum

and a minimum value among all the sample data which will be used to set layer's minimum and maximum limit for a particular finger. This information is stored in matrix L.

Matrix L

Matrix L is used to identify the layer information based on the flex sensor input. We have divided computer keyboard alphabets into 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 the middle layer, it will give higher flex sensor input than when the same finger is used to press a key in the upper layer. The same relation is observed between the middle layer and the lower layer. So, by saving the flex sensor values for all fingers for each layer, it is easy to identify the layer using flex sensor input.

Layer classification is given below:

- 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'.

Matrix L is 3 x 16 size matrix which contains data of possible minimum and maximum flex sensor input for each layer. By default, maximum and minimum bending measurement values for each finger is set to 2000 and 0 accordingly for each layer. One naive approach to organizing this matrix is to replace current min/max value with the new min/max value at each keystroke. But, this approach may produce overlapping values between two layers' range. For example, if the value of the Upper layer is in the range of 0 to 600 and the value of the Middle layer is in the

range of 400 to 1000, it will produce overlapping values between 400 to 600. Therefore, we will not be able to know exactly in which layer the user had pressed the key. This problem is further explained in the model evaluation chapter. This problem would force our algorithm to take two or more layers in consideration instead of one, which weakens the system's prediction model. To avoid this problem, current minimum and maximum values are replaced by taking an average of current minimum-maximum values and flex sensor input's minimum-maximum values. Very first entry of minimum and maximum value would compare 2000 and 0 respectively and following entries were taken from an average of the present value and current input values. This running average approach reduced the overlapping problem.

TABLE 1. Matrix L for Left-hand Bending Measurement Values

	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	0	2000	544	508	664	505	648	563
L_1	405	351	838	817	875	739	788	596
L_2	0	2000	932	890	877	781	810	724

TABLE 2. Matrix L for Right-hand Bending Measurement Values

	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	730	588	1080	954	532	498	0	2000
L_1	878	783	1273	1154	645	596	0	2000
L_2	907	811	1357	1264	0	2000	0	2000

Now, as our flex sensor output only ranges from 100 to 1900, 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 particular finger wasn't used in that particular layer.

Table 1 depicts the layer matrix L for left-hand fingers which contains minimum and the maximum value of flex sensors while typing in particular layer by left-hand fingers. While Table 2 shows the minimum and the maximum value of flex sensors while typing in particular layer using right-hand fingers.

In these tables, F_min_X and F_max_X represents minimum and maximum bending measurement for finger X. X € {0,1,2,3,4,5,6,7} and 0,1,...,7 represents Left-hand Little Finger, Left-hand Ring Finger, Left Hand Middle Finger, Left-hand Index finger, Right-hand Index finger, Right Hand Middle Finger, Right-hand Ring finger, Right-hand Little finger. L0, L1 and L2 represents Upper layer(UL), Middle Layer(ML) and Lower Layer(LL) respectively.

Above table shows that Left-hand Little finger's reading in L0 and L2 are 0 in minimum value column and 2000 in the maximum value column. That shows that Left-hand Little finger was not used in the Upper layer and a Lower layer as its values are default set values. Below table demonstrates the mathematical expression of how data is stored in the matrix.

TABLE 3. Construction Process of Matrix L

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: $[Layer_x][Finger\_min_y] = avg([Layer_x][Finger\_min_y] , Qmin)$

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

The system also updates one more matrix database which tracks how many times a particular character has been pressed by a particular finger. Matrix F helps the algorithm to understand user's typing behavior.

Matrix F

Matrix F is a mapping matrix between letters and fingers. It is 27 x 10 sized matrix with labeled finger name in the first row and alphabets in the first column. A second column filled with layer information in the form of '0', '1' or '2' which represents Upper Layer (UL), Middle Layer (ML) and Lower Layer (LL) respectively.

This matrix contains occurrences of pressed alphabets by different fingers. In the training phase, simply the matrix is updated at each keystroke. While the system is in use, this matrix is

used to get a possible list of letters pressed by the users based on the layer data and finger data.

Here, ‘CH’ denotes the letter/character, 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 the device is in actual use.

TABLE 4. Matrix F

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

TABLE 4. Continue

Letters	Layer	LL	LR	LM	LI	RI	RM	RR	RL	Total Occurrences
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

Below table demonstrates mathematical representation of the construction of Matrix F and how it gets updated while training the gloves/device. To update this matrix we need two parameters one is finger value and another is Layer value.

TABLE 5. Construction Process of Matrix F

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 Matrix L. CH0, CH1,...CHn, where CH0, CH1,...CHn are single character input from training data.
Output:	F[Fx][CHy] = F[Fx][CHy] + 1 where x € {0,1,2,3,4,5,6,7} and y €{0,1,2,…n} and F denotes Matrix F

This describes the relation between a finger and a particular key pressed by that finger. To reduce the complexity of the combinations and process time, the system maintains one more matrix, Matrix T. This matrix does not get updated in training phase. It is only used in the prediction process.

Matrix T

To make prediction more accurate, we use the 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 times a particular alphabet occurs in the given 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. Let's say, if user types 'q' as the first letter and second probable alphabet set is ('e','r'), combination 'qe' has no possibility to come in any English word. So only considerable combination would be 'qr'. This matrix is already constructed based on words from a dictionary, which is used to predict the words. This matrix also gets updated when the device is in actual use, to get more accurate prediction. In the training phase, the matrix is just updated at each keystroke as shown below:

TABLE 6. Construction Process of Matrix T

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

Above table depicts the mathematical representation of how this matrix is constructed and how it gives the output when we feed previous and current character.

Now we have all information about finger bending measurement to determine layer and finger-letter associativity. Finger-letter associativity is used to determine which key is pressed with probabilistic data and is enough to determine user's typing behavior. To avoid errors and increase the accuracy of predictions, we built a word dictionary which used to predict the English

word typed by the user. Once the user completes training, the system will be ready to use.

Dictionary

The dictionary is used to predict the final word based on possible combinations of letters. The dictionary is a pool of words collected from open source Shakespeare's work. The reason behind choosing Shakespeare's work is, it has a pool of unique words and it is an open source free license book to use. The list of possible combinations is already sorted. So, based on the combinations, a list of words is generated which is shown to the user for selection. And the whole process of predicting keystrokes is continued until the user finalizes the word. Once, the user finalizes the word, all new list of possible combinations are eliminated and the same process is repeated again.

CHAPTER 4

Model for prediction of keystroke

This chapter illustrates keystroke prediction algorithm. After training the module successfully, the user can use this model to predict characters. Assuming that the training has been done in required training environment by following training instructions. Considering this device as a prototype, it runs with certain limitations and assumptions. Assuming that, while training gloves, the user is not lifting his wrist once he started typing. The algorithm captures finger bending measurement range for each layer. Lifting the wrist in the middle of the process and placing it again may change bending measurement values for layers. As of now, the system uses a dictionary, which is created from Shakespeare's work to predict words. So, the system can not predict any nouns other than nouns which are used in Shakespeare's work. Currently, the system is designed to predict word which contains only alphabets (a-z).

Before using this device, make sure that Matrix L, Matrix F, and Matrix T is created while training the device. When the system is in "Device in Use mode", make sure that the gloves are connected to Arduino board and Arduino board is connected with Computer on COM1. Arduino transmits information in the form of serial data. pySerial, a python library receives the data from serial communication port -COM1. A python script segregates information about finger, thumb and finger bending measurements from the input. The system derives layer information from Matrix L using below process.

Fetching layer information from matrix L:

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 the data. The algorithm takes finger input and bending measurement to maps it to that finger's min-max range to find out layer information. If this value lies in two different ranges which are overlapped, the algorithm considers both layer values as predicted layer. For example, bending measurement value '450' comes from the right-hand index finger. Suppose, right-hand index finger's upper layer value range is 100-500 and middle layer value range is 400-800. Bending measurement lies in both layer's minimum and maximum value range. So, the system will consider both layers in the further prediction process.

Below table depicts mathematical expression of layer fetching process from matrix L. System will fetch out two columns according to input finger value. One of them is filled with that finger's minimum values and other is filled with maximum values in each layer. Now algorithm will compare bending measurement input with minimum and maximum value in extracted table for each layer and fetch out layer value.

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

TABLE 7. Fetching Layer Information from Matrix L

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

Fetching a set of alphabets from matrix F:

As discussed in training phase, the second column of Matrix F is filled with layer values. After layer fetching process, we have layer values to use it in the further process. Algorithm extract out one table from given matrix based on layer value. Eventually, the system extracts out the set of characters pressed by input finger; for which table gives some integer value other than zero. '0' in the table represents that user has never used that finger to press given a letter. For an example, refer (table 4), for getting alphabet set. Suppose we have Upper Layer as an input which is also denoted as layer '0' and input finger is right-hand index finger. Ignoring zero values in a column, we get two alphabets 'y' and 'u'. This list of possible letters is extracted with the occurrences of the letter being pressed by that finger.

List of possible letters : [(“u” : 14 , “y”: 11)]

For every input, the system fetches sets of alphabets and rearrange them based on the occurrence of that alphabet by given finger. So that, most probable pressed alphabet with input finger comes first in the set. In our example letter “u” pressed for 14 times by right-hand index finger while letter “y” pressed for 11 times with the same finger. So, “u” stands ahead of “y” in the list of possible letters. At every fetch, the system will get one set of alphabets according to input finger and layer information. The system will start making every possible combination out of these sets of alphabets except the first fetch. Because in the first fetch we would have only one set that is not enough to make combinations. In our example, if our second input gives us [(“n”: 16, “b”: 13)] then the system will make combination as shown below:

Possible combinations: [“un”, “ub”, “yn”, “yb”]

TABLE 8. Fetching Set of Alphabets from Matrix F

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 the possible pressed letter with their probability of being typed. The set is sorted in the descending order of the probability.

To reduce the time complexity of prediction process, character combinations are further filtered from Matrix T. Filtering process takes a previous and current character from the

combination and removes invalid combinations from the list.

Filtering invalid combinations using matrix T:

From possible combinations list system will pick one by one combination and check whether there is any dictionary word having that combination in it. '0' value in the matrix represents that there is zero possibility of coming that pair in a dictionary. For an example, in our system, there is no word having combination "yb" and "yn" at all. All the combinations with zero possibilities are eliminated and hence removed from the list of possible combinations.

Combination set: ["un", "ub", "yn", "yb"]

New Combination set: ["un", "ub"]

Below table elaborate filtering process from possible combinations list.

TABLE 9. Filtering Invalid Combination

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

A new set of combinations again checks possible words from the dictionary using a regular expression. Dictionary is already sorted according to words' frequency, so system suggests possible words in order to its frequency. The current system only suggests top 10 most probable words from the dictionary.

In typing process, at each keystroke activity, the system shows predicted set of alphabets. If the user feels to backward his last finger tapping activity, he can do so up to 4 times. As of now, he can use his right-hand little finger to do backspace activity. In future, this action will be done by the right-hand thumb.

Example:

This example illustrates prediction process of our model. Inputs are given to the system with the desire to type word ‘universe’. When we connect Arduino board with computer and run prediction model, it will load all pre-requisites and print a log on consol. Below image shows that all required data has been loaded.

```
{'a': 0, 'c': 2, 'b': 1, 'e': 4, 'd': 3, 'g': 6, 'f': 5, 'i': 8, 'h': 7, 'k': 10, 'j': 9, 'm': 12, 'l': 11, 'o': 14, 'n': 13,
[[9.0, 122.0, 312.0, 221.0, 7.0, 37.0, 136.0, 19.0, 207.0, 2.0, 49.0, 618.0, 196.0, 682.0, 5.0, 152.0, 8.0, 643.0, 282.0, 752.
, 0.0, 122.0, 53.0, 13.0, 62.0, 0.0, 0.0, 0.0, 22.0, 0.0], [226.0, 3.0, 1.0, 3.0, 285.0, 0.0, 0.0, 0.0, 181.0, 0.0, 0.0, 14.0,
3.0, 13.0, 90.0, 123.0, 3.0, 5.0, 7.0, 436.0, 646.0, 54.0, 52.0, 5.0, 1.0, 41.0, 3.0], [53.0, 56.0, 113.0, 100.0, 12.0, 40.0,
65.0, 83.0, 49.0, 85.0, 15.0, 57.0, 0.0, 77.0, 1.0, 5.0, 134.0, 105.0, 215.0, 6.0, 66.0, 2.0, 292.0, 202.0, 175.0, 1.0, 2.0, 1
Serial port is open
CLEARDATA

LABEL, TIME ,Finger 1,Finger 2,Finger 3,Finger 4,Finger 5,Finger 6,Finger 7,Finger 8
```

FIGURE 4.1. Console print when we connect device with computer and run the module.

Below image shows reading of first input and model’s prediction outputs.

```
['4', '627\r\n']
4 681
Finger Record : ['RI']
(730, 588)
(878, 783)
(907, 811)
input layer  1 is :  U

Probability set :  [('u', 14.0), ('y', 11.0), ('i', 1.0)]
All possible combinations :  ['u', 'y', 'i']
error!

Validated combinations :  ['u', 'y', 'i']

Top 10 possible words : ['us', 'upon', 'up', 'use', 'used', 'under', 'unto', 'uncle', 'unless', 'until', 'understand']
```

FIGURE 3.2. Console print of input data and prediction algorithm.

This input predicts characters set {‘u’,’y’,’i’}. Now we try to type letter ‘n’ and we get this print on the console.

```
['4', '1094\r\n']
4 1111
Finger Record : ['RI', 'RI']
(730, 588)
(878, 783)
(907, 811)
(730, 588)
(878, 783)
(907, 811)
input layer  1 is :  L

Probability set :  [('n', 16.0), ('b', 13.0), ('m', 1.0)]
Viterbi  update called
All possible combinations :  ['un', 'ub', 'um', 'yn', 'yb', 'ym', 'in', 'ib', 'im']

Validated combinations :  ['un', 'ub', 'um', 'yn', 'yb', 'ym', 'in', 'ib', 'im']

Top 10 possible words : ['under', 'unto', 'uncle', 'unless', 'until', 'understand', 'united', 'unknown', 'unhappy', 'undone',
```

FIGURE 4.3. Console print for second key input.

```
['5', '1050\r\n']
5 1046
Finger Record : ['RI', 'RI', 'RM']
(1080, 954)
(1273, 1154)
(1357, 1264)
input layer  1 is :  U

Probability set :  [('o', 25.0), ('i', 17.0), ('p', 10.0), ('e', 1.0)]
Viterbi  update called
All possible combinations :  ['uno', 'uni', 'unp', 'une', 'ubo', 'ubi', 'ubp', 'ube', 'umo', 'umi', 'ump', 'ume', 'yno', 'yni'

Validated combinations :  ['uno', 'uni', 'unp', 'une', 'ubo', 'ubi', 'ubp', 'ube', 'umo', 'umi', 'ump', 'ume', 'yno', 'yni', '

Top 10 possible words : ['unopened', 'unobserved', 'unowed', 'united', 'union', 'universal', 'unite', 'unity', 'universes', 'u
```

FIGURE 4.4. Console print for third input.

```
['3', '905\r\n']
3 905
Finger Record : ['RI', 'RI', 'RM', 'LI']
(648, 563)
(788, 596)
(810, 724)
(648, 563)
(788, 596)
(810, 724)
input layer  1 is :  L

Probability set :  [('v', 14.0), ('c', 7.0)]
Viterbi  update called
All possible combinations :  ['unov', 'unoc', 'univ', 'unic', 'unpv', 'unpc', 'unev', 'unec', 'ubov', 'uboc', 'ubiv', 'ubic',

Validated combinations :  ['unov', 'unoc', 'univ', 'unic', 'unpc', 'unev', 'unec', 'ubov', 'uboc', 'ubiv', 'ubic', 'ubpc', 'ub

Top 10 possible words : ['universal', 'universes', 'universe', 'university', 'universities', 'universally', 'univ', 'unicorns'
```

FIGURE 4.5. Console print for forth input.

Now, we have word ‘universe’ as third word. If user wants to confirm the third word whithout typing further. He needs to press his left-hand thumb for longer time and one by one word will iterate and asks user to confirm the word. Once user lift his left-hand thumb, last iterated word will become confirmed word.

```
LTL
Current selected word :  universal
Keep long pressing left thumb to iterate over the list of possible words :  ['universal', 'universes', 'universe', 'university

LTL
Current selected word :  universes
Keep long pressing left thumb to iterate over the list of possible words :  ['universal', 'universes', 'universe', 'university

LTL //Desired word is picked up
Current selected word :  universe
Keep long pressing left thumb to iterate over the list of possible words :  ['universal', 'universes', 'universe', 'university

LTL END //Release left thumb long pressing action.

selected word :  universe // Selected word
Start typing ahead
```

FIGURE 4.6. Word confirmation process

But, if user wants to continue typing till the end of word. He can start typing and this word will further come ahead in list.

```
['2', '534\r\n']
2 539
Finger Record : ['RI', 'RI', 'RM', 'LI', 'LM']
(664, 505)
(875, 739)
(877, 781)
input layer  1 is :  U

Probability set :  [('e', 26.0), ('r', 1.0)]
Viterbi  update called
All possible combinations :  ['unove', 'unovr', 'unoce', 'unocr', 'unive', 'univr', 'unice', 'unicr', 'unpce', 'unpcr', 'uneve
ce', 'ibecr', 'imove', 'imovr', 'imoce', 'imocr', 'imive', 'imivr', 'imice', 'imicr', 'impce', 'impcr', 'imeve', 'imevr', 'ime

Validated combinations :  ['unove', 'unovr', 'unoce', 'unocr', 'unive', 'univr', 'unice', 'unicr', 'unpce', 'unpcr', 'uneve',
, 'ibecr', 'imove', 'imovr', 'imoce', 'imocr', 'imive', 'imivr', 'imice', 'imicr', 'impce', 'impcr', 'imeve', 'imevr', 'imece'

Top 10 possible words : ['universal', 'universes', 'universe', 'university', 'universities', 'universally', 'uneven']
```

FIGURE 4.7. Console print for fifth input.

```
['3', '589\r\n']
3 615
Finger Record : ['RI', 'RI', 'RM', 'LI', 'LM', 'LI']
(648, 563)
(788, 596)
(810, 724)
Found multi layer :  2
input layer  2 is :  U
input layer  2 is :  M

Probability set :  [('r', 22.0), ('t', 21.0), ('f', 15.0), ('g', 11.0), ('o', 1.0), ('s', 1.0)]
Viterbi  update called
All possible combinations :  ['unover', 'unovet', 'unovef', 'unoveg', 'unoveo', 'unoves', 'unovrr', 'unovrt', 'unovrf', 'unovr
ceg', 'uboceo', 'uboces', 'ubocrr', 'ubocrt', 'ubocrf', 'ubocrg', 'ubocro', 'ubocrs', 'ubiver', 'ubivet', 'ubivef', 'ubiveg',
'imicrr', 'imicrt', 'imicrf', 'imicrg', 'imicro', 'imicrs', 'impcer', 'impcet', 'impcef', 'impceg', 'impceo', 'impces', 'impcr

Validated combinations :  ['unover', 'unovet', 'unovef', 'unoveg', 'unoveo', 'unoves', 'unovrr', 'unovrt', 'unovrf', 'unovrg',
ibiver', 'ibivet', 'ibivef', 'ibiveg', 'ibiveo', 'ibives', 'ibivrr', 'ibivrt', 'ibivrf', 'ibivrg', 'ibivro', 'ibivrs', 'ibicer
rr', 'imicrt', 'imicrf', 'imicrg', 'imicro', 'imicrs', 'impcer', 'impcet', 'impcef', 'impceg', 'impceo', 'impces', 'impcrr', '

Top 10 possible words : ['universal', 'universes', 'universe', 'university', 'universities', 'universally']
```

FIGURE 4.8. Console print for sixth input.

```
['1', '801\r\n']
1 813
Finger Record : ['RI', 'RI', 'RM', 'LI', 'LM', 'LI', 'LR']
(544, 508)
(838, 817)
(932, 890)
(544, 508)
(838, 817)
(932, 890)
input layer  1 is :  M

Probability set :  [('s', 11.0)]
Viterbi  update called
All possible combinations :  ['unovers', 'unovets', 'unovefs', 'unovegs', 'unoveos', 'unovess', 'unovrrs', 'unovrts', 'unovrfs
vrrs', 'ubovrts', 'ubovrfs', 'ubovrgs', 'ubovros', 'ubovrss', 'ubocers', 'ubocets', 'ubocefs', 'ubocegs', 'uboceos', 'ubocess'
imevros', 'imevrss', 'imecers', 'imecets', 'imecefs', 'imecegs', 'imeceos', 'imecess', 'imecrrs', 'imecrts', 'imecrfs', 'imecr

Validated combinations :  ['unovers', 'unovets', 'unovefs', 'unovegs', 'unoveos', 'unovess', 'unovrrs', 'unovrts', 'unovrfs',
s', 'ubovrts', 'ubovrfs', 'ubovrgs', 'ubovros', 'ubovrss', 'ubocers', 'ubocets', 'ubocefs', 'ubocegs', 'uboceos', 'ubocess', '
vros', 'imevrss', 'imecers', 'imecets', 'imecefs', 'imecegs', 'imeceos', 'imecess', 'imecrrs', 'imecrts', 'imecrfs', 'imecrgs'

Top 10 possible words : ['universal', 'universes', 'universe', 'university', 'universities', 'universally']
```

FIGURE 4.9. Console print for seventh input.

```
2 454
Finger Record : ['RI', 'RI', 'RM', 'LI', 'LM', 'LI', 'LR', 'LM']
(664, 505)
(875, 739)
(877, 781)
(664, 505)
(875, 739)
(877, 781)
input layer  1 is :  U

Probability set :  [('e', 26.0), ('r', 1.0)]
Viterbi  update called
All possible combinations :  ['unoverse', 'unoversr', 'unovetse', 'unovetsr', 'unovefse', 'unovefsr', 'unovegse', 'unovegsr',
, 'unicessr', 'unicrrse', 'unicrrsr', 'unicrtse', 'unicrtsr', 'unicrfse', 'unicrfsr', 'unicrgse', 'unicrgsr', 'unicrose', 'un
pcegse', 'ibpcegsr', 'ibpceose', 'ibpceosr', 'ibpcesse', 'ibpcessr', 'ibpcrrse', 'ibpcrrsr', 'ibpcrtse', 'ibpcrtsr', 'ibpcrfs
sr', 'imovrose', 'imovrosr', 'imovrsse', 'imovrssr', 'imocerse', 'imocersr', 'imocetse', 'imocetsr', 'imocefse', 'imocefsr',
'impceosr', 'impcesse', 'impcessr', 'impcrrse', 'impcrrsr', 'impcrtse', 'impcrtsr', 'impcrfse', 'impcrfsr', 'impcrgse', 'impc

Validated combinations :  ['unoverse', 'unoversr', 'unovetse', 'unovetsr', 'unovefse', 'unovefsr', 'unovegse', 'unovegsr', 'u
unicessr', 'unicrrse', 'unicrrsr', 'unicrtse', 'unicrtsr', 'unicrfse', 'unicrfsr', 'unicrgse', 'unicrgsr', 'unicrose', 'unicr
ybicrsse', 'ybicrssr', 'ybpcerse', 'ybpcersr', 'ybpcetse', 'ybpcetsr', 'ybpcefse', 'ybpcefsr', 'ybpcegse', 'ybpcegsr', 'ybpce
impceosr', 'impcesse', 'impcessr', 'impcrrse', 'impcrrsr', 'impcrtse', 'impcrtsr', 'impcrfse', 'impcrfsr', 'impcrgse', 'impcr

Top 10 possible words : ['universes', 'universe']
```

FIGURE 4.10. Console print for eighth input.

Now, user has typed all 8 letters that build word universe. Here word ‘universe’ is still at second place. Once user confirm the word length, that is 8 in our example, words having length other than confirmed length will be removed from the list. Below image shows length confirming process.

```
['LTT\r\n']
['LTT']
Length of the words are :  8
['universe']
Long tap left thumb to select the word
```

FIGURE 4.11. Console print for word length confirmation process.

Below image shows word confirmation process.

```
['LTL\r\n']
['LTL']
Current selected word :  universe
Keep long pressing left thumb to iterate over the list of possible words :  ['universe']
Leave the left thumb to select the word :  universe

['LTL', 'END\r\n']

 selected word :  universe
 Dictionary has been updated
Start typing ahead
```

FIGURE 4.12. Console print for word prediction process.

CHAPTER 5

Learning model

This prototype works with machine learning techniques. After getting trained, the machine has to learn user's typing behavior continuously to improve its prediction algorithm. In our prediction model, the user will get a list of most possible words at each keystroke activity. The keystrokes activity by the user is directly proportional to a number of characters in the word. The user needs to confirm the length of the word by tapping left thumb, after typing the desired number of characters. The system only displays a list of words which have a similar length that is confirmed by the user. The long pressing activity of left thumb will further allow user to switch between different choices of possible words provided to confirm the word.

Once the user confirms the word, system updates following data:

- It updates particular word frequency in the dictionary. So, next time particular word comes ahead with given combinations of alphabets set.
- It updates Viterbi matrix, which helps to predict best alphabet combinations.
- When the prototype is in use, it records user's finger data until the user confirms the word so that system can update alphabets occurrence for associated finger in matrix F.

Example :

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

1. Dictionary : frequency(‘find’) = frequency(‘find’) + 1
2. Matrix T :

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

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

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

3. Matrix F:

   F[‘LI’][‘f’] = F[‘LI’][‘f’] + 1

   F[‘RM’][‘i’] = F[‘RM’][‘i’] + 1

   F[‘RI’][‘n’] = F[‘RI’][‘n’] + 1

   F[‘LM’][‘d’] = F[‘LM’][‘d’] + 1

CHAPTER 6

MODEL EVALUATION

As of now, this prototype is trained as per my keystroke dynamics. We do not have any reference to compare results with any other devices so that we compare results with its previous experiments. Predictive model's one of the evaluation parameter is the amount of data required to train the gloves. Early experiments were done by pressing each alphabet for 15-20 times, which was not enough to predict letters and it also gives a narrow range for layer limits. After some initial testing, I could conclude that using every alphabet for at least 45-50 times provides better predictive analysis of characters as well as the layer. It will also reduce the overlapping of layer limits. The readings mentioned below are taken by providing gloves each alphabet for 15-20 times. Blue marked readings shows that layer range values are overlapping with each other.

TABLE 10. Matrix L for Left-hand Fingers in Test 1

	F_max_0	F_min_0	F_max_1	F_min_1	F_max_2	F_min_2	F_max_3	F_min_3
L_0	212	169	820	545	670	501	702	554
L_1	401	357	900	817	864	644	804	562
L_2	0	2000	932	890	873	790	819	720

TABLE 11. Matrix L for Right-hand Fingers in Test 1

	F_max_4	F_min_4	F_max_5	F_min_5	F_max_6	F_min_6	F_max_7	F_min_7
L_0	752	574	1006	931	530	464	0	2000
L_1	896	750	1270	997	679	588	0	2000
L_2	921	851	1304	1257	0	2000	0	2000

TABLE 12. Matrix F in Test 1

LETTER	LAYER	LL	LR	LM	LI	RI	RM	RR	RL	Total_Occurrences
q	0	3	14	0	0	0	0	0	0	17
w	0	0	15	2	0	0	0	0	0	17
e	0	0	3	16	0	0	0	0	0	19
r	0	0	0	2	16	0	0	0	0	18
t	0	0	0	0	15	0	0	0	0	15
y	0	0	0	0	3	13	0	0	0	16
u	0	0	0	0	0	19	0	0	0	19
i	0	0	0	0	0	2	17	0	0	19
o	0	0	0	0	0	0	20	0	0	20
p	0	0	0	0	0	0	14	4	0	14
a	1	20	0	0	0	0	0	0	0	20
s	1	0	17	0	0	0	0	0	0	17
d	1	0	0	16	0	0	0	0	0	16

TABLE 13. Continued

Letters	Layer	LL	LR	LM	LI	RI	RM	RR	RL	Total Occurrences
f	1	0	0	0	20	0	0	0	0	20
g	1	0	0	0	19	0	0	0	0	19
h	1	0	0	0	0	19	0	0	0	19
j	1	0	0	0	0	19	0	0	0	19
k	1	0	0	0	0	0	16	0	0	16
l	1	0	0	0	0	0	0	15	0	15
z	2	0	14	0	0	0	0	0	0	14
x	2	0	12	2	0	0	0	0	0	14
c	2	0	0	9	6	0	0	0	0	15
v	2	0	0	0	16	0	0	0	0	16
b	2	0	0	0	6	12	0	0	0	18
n	2	0	0	0	0	17	0	0	0	17
m	2	0	0	0	0	1	18	0	0	19

After training gloves by giving each character for 45-50 times. We got improvement in overlapping as well as in predicting alphabet sets by a particular key.

TABLE 14. Matrix L for Left-hand Fingers in Test 2

	F_max_0	F_min_0	F_max_1	F_min_1	F_max_2	F_min_2	F_max_3	F_min_3
L_0	2000	0	544	508	664	505	648	563
L_1	405	351	838	817	875	739	788	596
L_2	2000	0	932	890	877	781	810	724

TABLE 15. Matrix L for Right-hand Fingers in Test 2

	F_max_4	F_min_4	F_max_5	F_min_5	F_max_6	F_min_6	F_max_7	F_min_7
L_0	730	588	1080	954	532	498	2000	0
L_1	878	783	1273	1154	645	596	2000	0
L_2	907	811	1357	1264	2000	0	2000	0

TABLE 16. Matrix F in Test 2

LETTER	LAYER	LL	LR	LM	LI	RI	RM	RR	RL	Total_Occurrences
q	0	11	34	0	0	0	0	0	0	45
w	0	0	37	7	0	0	0	0	0	34
e	0	0	6	40	0	0	0	0	0	46

TABLE 17. Continued

Letters	Layer	LL	LR	LM	LI	RI	RM	RR	RL	Total Occurrences
r	0	0	0	5	40	0	0	0	0	45
t	0	0	0	0	43	0	0	0	0	43
y	0	0	0	0	9	35	0	0	0	44
u	0	0	0	0	0	50	0	0	0	50
i	0	0	0	0	0	3	47	0	0	50
o	0	0	0	0	0	0	50	0	0	50
p	0	0	0	0	0	0	40	4	0	44
a	1	50	0	0	0	0	0	0	0	50
s	1	0	48	0	0	0	0	0	0	48
d	1	0	0	47	0	0	0	0	0	47
f	1	0	0	0	48	0	0	0	0	48
g	1	0	0	0	48	0	0	0	0	48
h	1	0	0	0	0	50	0	0	0	50

TABLE 18. Continued

Letters	Layer	LL	LR	LM	LI	RI	RM	RR	RL	Total Occurrences
j	1	0	0	0	0	50	0	0	0	50
k	1	0	0	0	0	0	47	0	0	47
l	1	0	0	0	0	0	0	49	0	49
z	2	0	41	0	0	0	0	0	0	41
x	2	0	40	5	0	0	0	0	0	45
c	2	0	0	30	16	0	0	0	0	46
v	2	0	0	0	50	0	0	0	0	50
b	2	0	0	0	2	48	0	0	0	50
n	2	0	0	0	0	47	0	0	0	47
m	2	0	0	0	0	1	49	0	0	50

The results show that less amount of training is not giving the desired results, while more amount of training will help in predicting layers more precisely.

The increment in word frequency will not get improvement immediately, because word frequency difference between extracted most used word and second most used word based on

combinations, may be accountably big. Once the user uses this device for a longer period of time, it will improve word frequency and desired word can be predicted earlier than it is used to be.

At an early stage, the system has stored word's frequency as it appears in Shakespeare's work. With word combination of "ph", we could get many words as shown below:

Wordlist with it's frequency : ['philip', 'phrase', 'phebe', 'physician', 'physic', 'physical', 'philario', 'philosophy', 'phonograph', 'philosopher', ...]

'philip' - 74

'phrase' - 56

'phebe' - 47

'physician' - 33

'physic' - 33

'physical' - 23

'philario' - 23

'philosophy' - 21

'phonograph' - 21

'philosopher' - 18

For a Physician, when he types a word physician for 41 more times then onwards, combination 'ph' will give him a word 'physician at the first place as its frequency will cross the most frequent word 'philip'. So to Philosopher, when he types 'philosophy' for 53 times more, then in future this gadget will give philosopher a word 'philosophy' at first place with 'ph'

combination.

Furthermore, this device has been tested for 50 random words. Amongst them, the system doesn't recognize 6 words which are not stored in the dictionary and requires to press backspace for 17 times as this device is not well trained yet. However, when the system was tested with the same 50 words, it required pressing backspace for only 5 times. This experiment shows that if we don't consider words that were not in the dictionary at all, the early experiment has an accuracy of 61.36%, but next experiment shows that it increased to 88.63%. This increment in accuracy shows that if user trains this gloves well enough and use this gloves for several days, it will increase device's accuracy.

As this is a prototype, right now sensors are mounted on regular hand gloves which are made from fabric. All the wires are connected between sensors and Arduino board. Right now user needs one helping hand who can help the user to wear both the gloves. The device is taking one by one keystroke input to predict word. We have measured the response time between the point we are giving input to the system and we are getting predicted a list of words based on the current input which is less than a second. So, the system is pretty fast enough to use in real time.

CHAPTER 7

FUTURE WORK

Currently, this device is unable to predict proper nouns. In future, we can see a feature to learn nouns and remember them for further use by the system. We can also update dictionary with a large pool of words from the internet or from multiple documents to increase word database. This gloves are made with sensors mounted on fingers and connected physical wires to transmit data to the controller. Physical wires and connections reduce the mobility of the gloves. Miss handling of the device may disconnect these wires and this issue further produce malfunction in the prediction process. We can customize sensors and gloves as per its application and usability to avoid hardware malfunction. It will also reduce noise in sensors' data and increase prediction accuracy. In future, we can also add the Bluetooth module to make wireless communication between controller and sensors. This feature also increases device's application. After mounting wireless module we can use this device with mobile devices, too. It can be also used with modern VRs.

Our device is also capable of doing some gesture-based operations. In future, by adding small accelerometer sensor module, we can add the functionality of a touchpad. The combination of these two modules can allow the user to operate computer or VR device fully remotely.

If we add essential modules to this device, we can create multiple application of this gloves in VR technology. For example, only hardware can be used to create a robotic hand.

Bibliography

10fastfingers. *A simple Paragraph to practice simple typing.* n.d. https://10fastfingers.com/text/119-A-simple-Paragraph-to-practice-simple-typing.

Anna Maria Feit, Daryl Weir, Antti Oulasvirta. "How We Type: Movement Strategies and Performance in Everyday Typing." *In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM*, n.d.

Arduino. *Frequently Asked Questions.* n.d. https://www.arduino.cc/en/Main/FAQ.

Brownlee, Jason. *How to Prepare Data For Machine Learning.* 12 25, 2013. http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/.

Gest. *Gest - Work with your hands.* n.d. https://www.kickstarter.com/.

—. *Gest Homepage.* n.d. https://gest.co/.

Henry, Alan. *How Predictive Keyboards Work (and How You Can Train Yours Better).* 08 11, 2014. http://lifehacker.com/how-predictive-keyboards-work-and-how-you-can-train-yo-1643795640.

JAMES, ANTHONY. *Light, Camera, Glass ... Action.* 06 12, 2008. http://www.yankodesign.com.

JIMB0. *Serial Communication.* n.d. https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial.

Jimbo. *Flex Sensor Hookup Guide.* n.d. https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide.

Logbar. *Ring by logbar- shortcut to everything.* n.d. https://www.kickstarter.com/.

MARIAN, P. *Arduino Mega 2560 Pinout.* n.d. http://www.electroschematics.com/7963/arduino-mega-2560-pinout/.

Maxion, Kevin Killourhy and Roy. *Keystroke Dynamics - Benchmark Data Set.* n.d. http://www.cs.cmu.edu/~keystroke/.

McMahon, Russell. *Can I connect a PWM pin on one Arduino to an analog input on another?* 04 15, 2015. https://arduino.stackexchange.com/questions/10041/can-i-connect-a-pwm-pin-on-one-arduino-to-an-analog-input-on-another.

Obrazki.electroda. n.d. http://obrazki.elektroda.pl/2733383600_1383139811.png.

Orbitouch. *orbiTouch.* n.d. http://www.orbitouch.com/.

Photonics. *Finbre Optic Sensor.* n.d. http://photonics.cusat.edu/Research_Fiber%20Sensors_work%20at%20ISP.html.

Pythonhosted. *pySerial.* n.d. https://pythonhosted.org/pyserial/pyserial.html#overview.

rinkworks. *Fun with words.* n.d. http://www.rinkworks.com/words/pangrams.shtml.

Sparkfun. *Force Sensitive Resistor 0.5".* n.d. https://www.sparkfun.com/products/9375.

TAP. *TAP Homepage.* n.d. http://www.tapwithus.com/.

TypingDNA. *Homepage.* n.d. https://typingdna.com/?gclid=CjwKEAjwjunJBRDzl6iCpoKS4G0SJACJAx-VOvzitC4YdNFZYxjoZMB7pmCEHu6YBsUyFdOj2WZkCxoCofbw_wcB.

Uncommon Goods. n.d. http://www.uncommongoods.com/.

University, Aalto. *Ten fingers not needed for fast typing, study shows.* 02 10, 2016. https://phys.org/news/2016-02-ten-fingers-fast.html.

WikiPedia. *Projection Keyboard.* 06 01, 2017. https://en.wikipedia.org.

—. *Test Set.* n.d. https://en.wikipedia.org/wiki/Test_set.