CS6650 (Smart sensing for IoT)

IOT PROJECT REPORT

Alphabet identification using capacitive sensing

Problem statement

Using Capacitive sensing to identify the alphabets written on a 2d surface.
We want to build a surface through which we can identify touch gestures and identify the Gestures using some ML algorithm.

Motivation

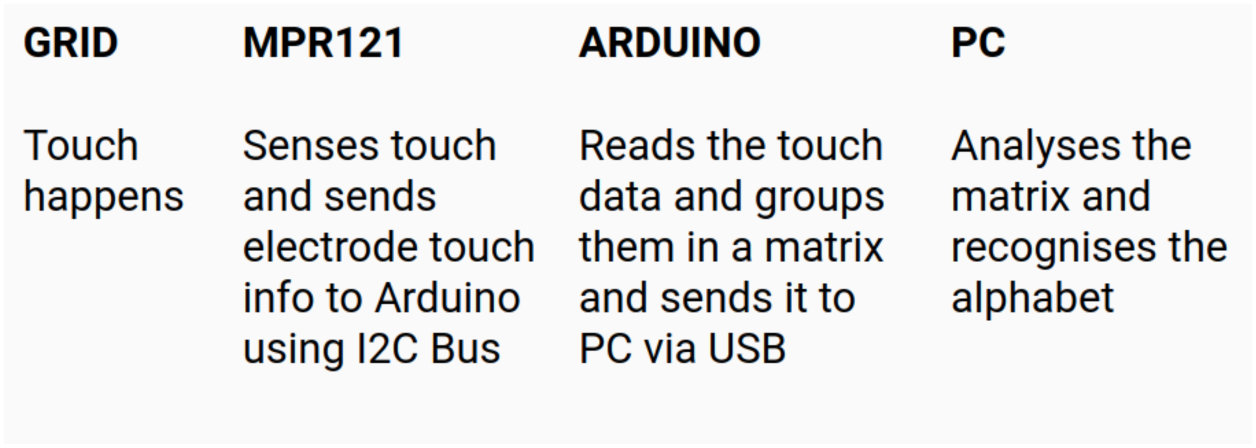
We liked the concept of capacitive touch sensing and wanted to explore more.
We went through multitouch kit by which we were amazed and so thought of doing something similar to that. We thought of exploring what we can do with a simple grid which is not as high resolution as touch pads and with discussion with prof we finally landed up with alphabet recognition.

Importance:

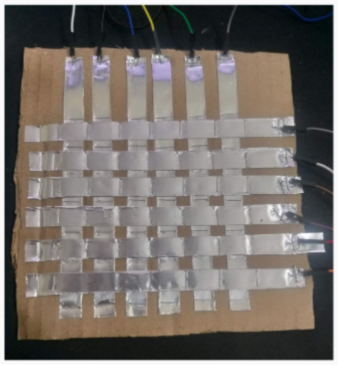
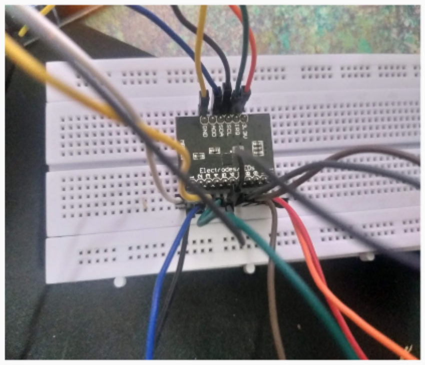
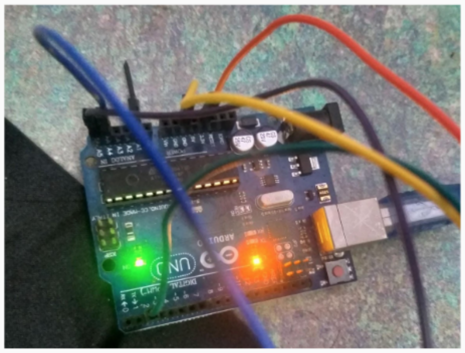
We can replace many high cost high resolution devices with these simple cheap alternatives for every day gesture recognition .

System Details

We used an aluminium grid , MPR121 sensor and arduino for this project.



GRID	MPR121	ARDUINO	PC
Touch happens	Senses touch and sends electrode touch info to Arduino using I2C Bus	Reads the touch data and groups them in a matrix and sends it to PC via USB	Analyses the matrix and recognises the alphabet

Lets see details of each component

Grid

- We used Aluminium tape to form a 6x6 grid (we have done it because MPR121 have 12 electrode pins)
- We arranged grids such that at least one stripe is touched when the finger is placed on the grid area.
- Ensured that the horizontal grid and vertical grid do not touch so that capacitance of the system doesn't go high, so that voltage drop becomes low .
  Initially we arrange them without any thing between vertical and horizontal stripes. Which made capacitance higher. And when we are seeing mpr121 readings there is almost negligible change when there is a touch and not touched.
- Each row and column is connected to electrode pin of MPR121
- The width of tape that we pot is 2.4cm we made it into half so that it correctly suffices for finger

Initially we thought of using an 8x8 grid and using 2 MPR121 sensors one for rows and other for columns but due to synchronization issues we decided to stick with 6x6.

Without touch voltage readings for insulated grid

```
Filt: 59        52      54      51      64      60      54      67      54      57      67      47
Base: 56        52      52      48      60      56      52      64      52      56      64      44
```

With touch voltage readings for insulated grid

```
Filt: 51        53      54      52      64      60      29      67      54      57      67      47
Base: 48        52      52      48      60      56      24      64      52      56      64      44
```

We can see that direct touch at 7th electrode and indirect touch on 1st electrode are clearly observable in insulated grid

Without touch voltage readings for non insulated grid

```
Filt: 15        13      14      16      14      17      12      13      16      16      15      17
Base: 12        12      12      12      12      16      8       12      12      12      12      16
```

With touch voltage readings for non insulated grid

```
Filt: 12        13      14      16      14      17      12      13      16      16      16      17
Base: 12        12      12      12      12      16      8       12      12      12      12      16
```

We can see that voltage values decreased in non insulated grid and there are no significant changes in readings with or without touch

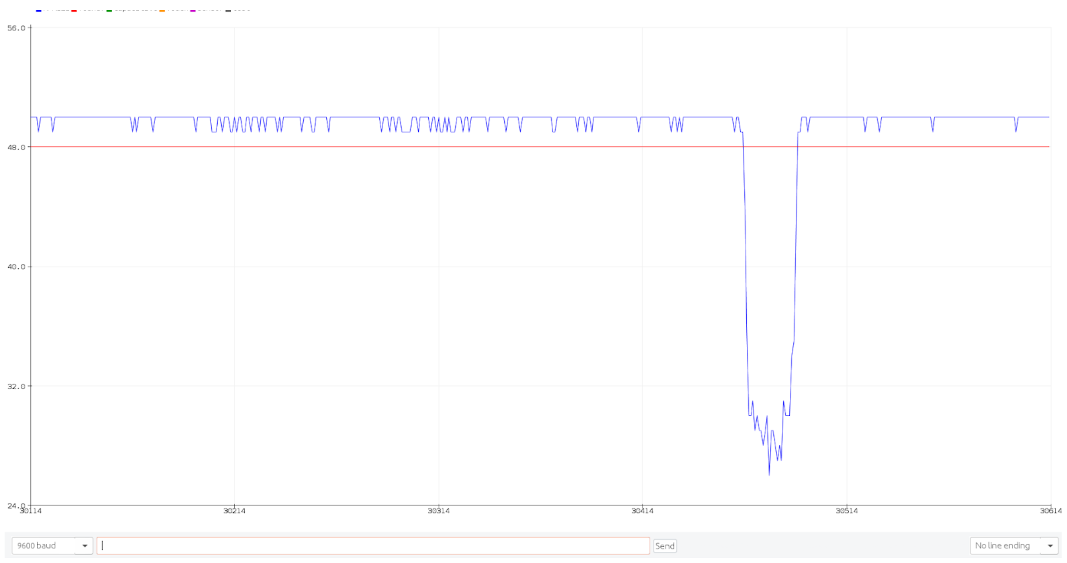
MPR121

- It is a touch sensor which has 12 electrode pin and uses I2C bus for communication
- There are many libraries available to interface it with auduno (we used adafruit_MPR121 library)

How does it work ?

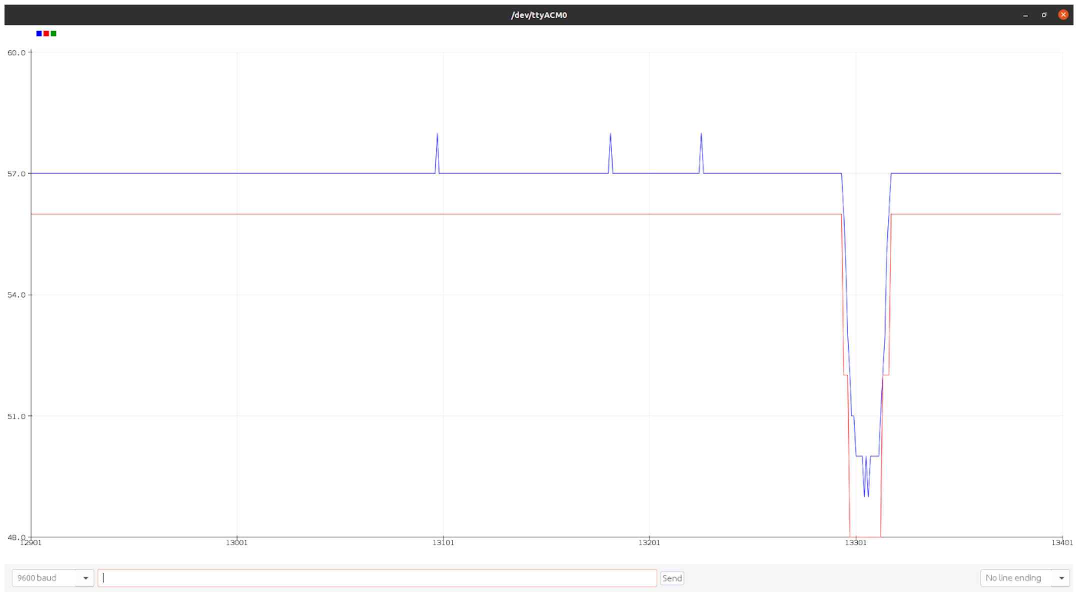
- It sends constant current to each electrode for some time and measures voltage values
- Whenever there is touch capacitance increases and voltage decreases
- It senses that change in voltage by maintaining bace and filtered values
- Bace is obtained after one more filter on filtered data so that only low frequency components remain and sudden jerks are neglected.

Plot of bace and filtered data for electrode corresponding to touched row in grid (direct touch)



Blue line represents filtered data and red line represents bace whenever filtered data becomes lower than base by some threshold

Plot of bace and filtered data for electrode corresponding to touched column in grid (indirect touch)



Now in our grid arrangement rows are present over columns so whenever touch happens on intersections touch will be direct on rows and indirect on columns.

In graph 1 we can see that touch is clearly identified by sensor but in second graph touch will not be recognised by sensor as change is reading is low and base also tried to adjust.

So from the sensor we can directly get the row touched but we need to analyse separately to find which column is touched.

Audrino

- Here we try to identify the exact point where touch happens.
- Since MPR121 can only detect which row is touched directly we need to get data voltage data from MPR121 and try to get columns as well.
- We need to group all the touched vertices and send it in the form of a matrix to the PC so we can detect what alphabet it is on the PC using a ML algorithm.
- We are noting how many iterations each corner point is touched in a matrix so that we can also have weight to vertex based on the amount of time it is pressed.

Arduino Code Details :

Since we need to detect touch we need voltage information from MPR121.
We are using Adafruit_MPR121 Library to get voltage data.

How are we detecting the touch :
We are splitting execution in 2 phases
1) Initialization phase: Here we are trying to find the average voltage value of each electrode to use it as base.
2) Touch Detection phase: Here we are trying to sense touch and update matrix and send matrix to PC whenever total alphabet input is finished. We see the average amount of change for each electrode from the base that was computed in the initialization phase for 10 iterations and we declare the column with max change as touched column.
We were able to detect the exact touch point when the intersection point is touched. For all tests we done it gave accurate result

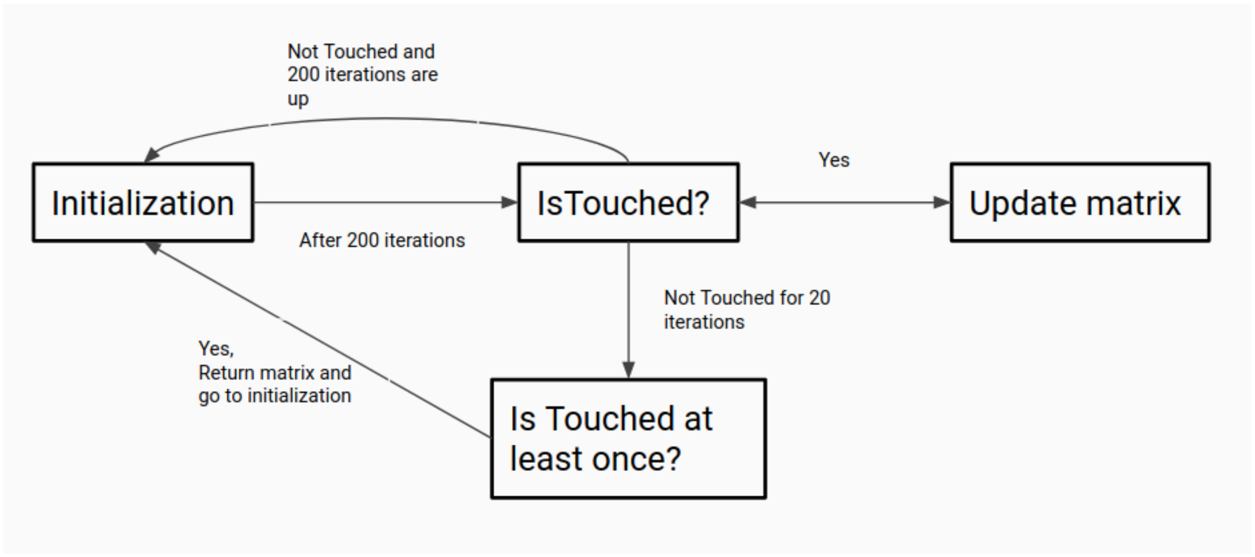

Not Touched and 200 iterations are up
Initialization
IsTouched?
Yes
Update matrix
After 200 iterations
Not Touched for 20 iterations
Yes, Return matrix and go to initialization
Is Touched at least once?

Image representing what we are implementing

We will be going from the initialization phase and touch detection(Is touched?) phase as shown in the diagram.

We will be in the initialization phase for 200 iterations and go to touch detection phase where we get touch points and populate in matrix and when we think input is done we will go back to initialization phase as writing something may change the state of the grid. We will detect completion of input if there is no input for 20 iterations . We absorbed that 17 iterations takes to jump from one point to another point while writing an alphabet so 20 iterations is enough to detect that it is completion of input and not some jump while writing alphabet.

We neglect the touches that happen in the initialization phase and and restart the initialization if it happens as touch may increase average voltage for electrodes.

Functions used from Adafruit Library :

Adafruit_MPR121() : used to initialise Adafruit_MPR121 object

begin(address) : used to start connection between Arduino and MPR121

touched() : returns all the electrodes where touch

filteredData(i) : return voltage information of ith electrode

Functions Implemented:

setup(): Starting connections with MPR121 and PC is done here.
loop(): this is the main loop of the program, all jumping from phases is implemented here.
mark_touched(): this is the function that detects touch and marks the touch point in the matrix.
_init(): this is a function that executes in the initialization phase and notes the average electrode voltages.
send_output(): sends the matrix to PC.

PC

- Here we need to analyse matrix sent by arduino and detect the alphabet
- We are using the Serial module in python to communicate with Audrino.
- We used the KNN algorithm to detect what alphabet the matrix is.

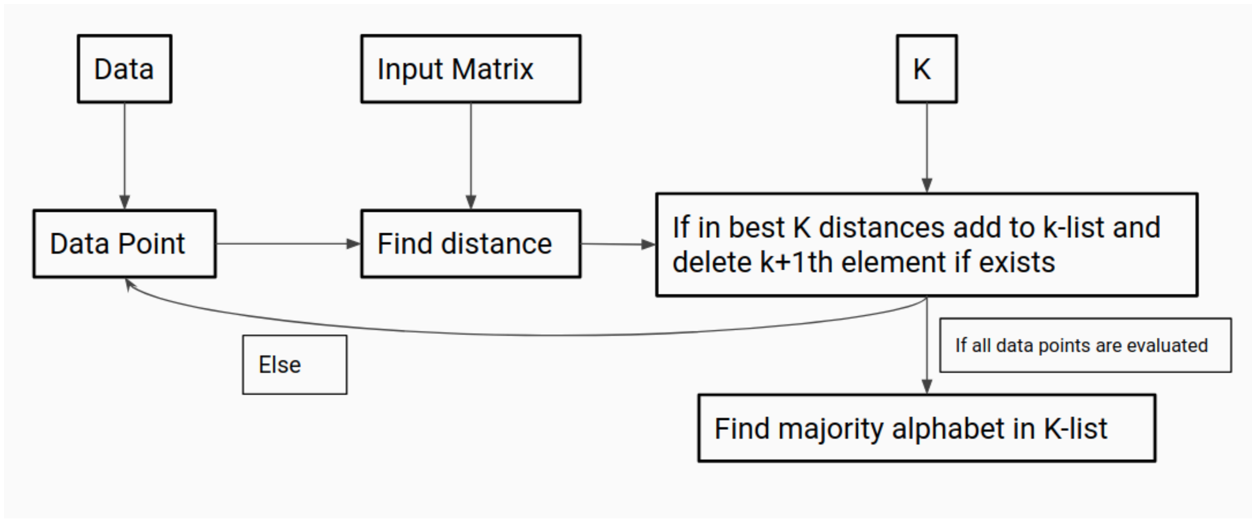
Initially We collected 20 data points for each alphabet and used them as data for the KNN algorithm.

We tried to collect data such that alphabet of all sizes are included in data.

We then collected 5 data points for each alphabet for testing.

We wrote 5 python files: (more details in readme file in src folder)
driver.py: module for communication with arduino
KNN.py: module for Kmm algorithm
Alpha_rec.py: module for alphabet recognition (combines driver.py and KNN.py)
Identificationtest.py: code that prints alphabet scribbled on grid (uses Alpha_rec.py)
Accuracytest.py: runs KNN algorithm on test data and prints the results
dataCollection.py: used to collect data

Algorithm Details


Data
Input Matrix
K
Data Point
Find distance
If in best K distances add to k-list and delete k+1th element if exists
If all data points are evaluated
Else
Find majority alphabet in K-list

Outline of KNN algorithm

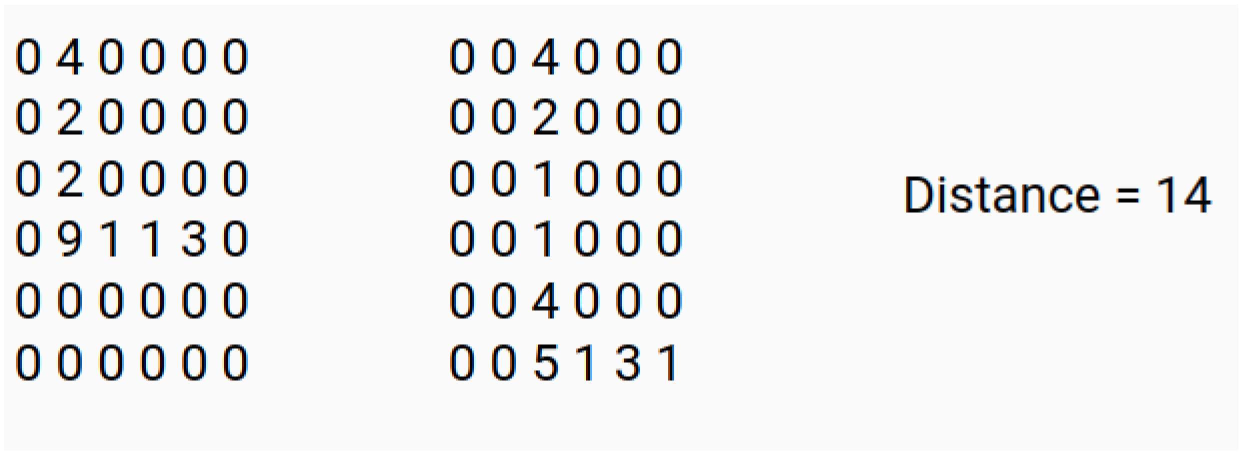
The above picture represents the Knn algorithm that we implemented.

We used the KNN algorithm with K as 3.

We tried 2 distance metrics

Metric1:

number of vertices where touch is detected in one matrix and not in another matrix

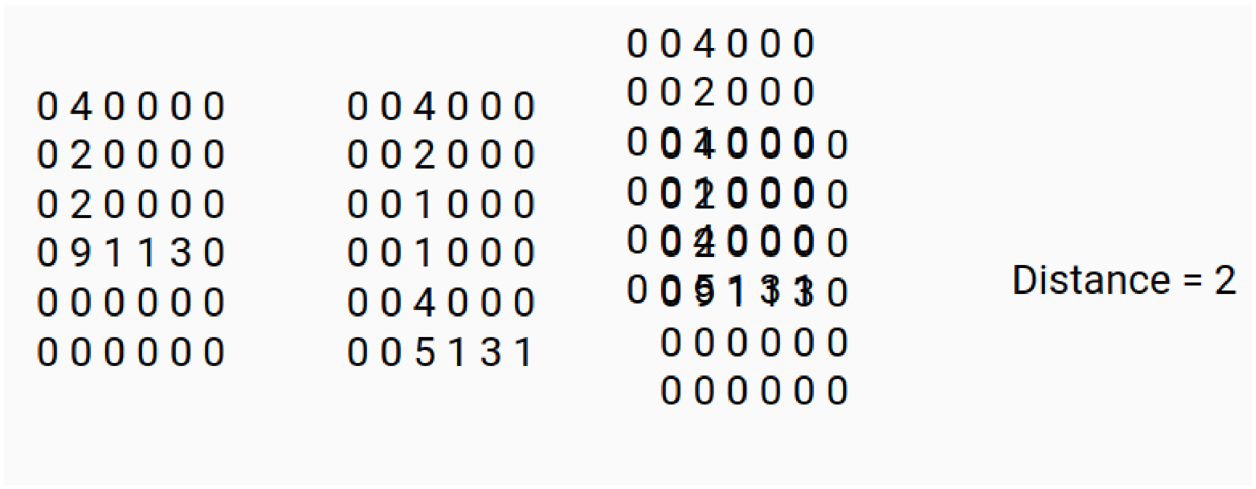


$$\begin{matrix} 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 9 & 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 1 & 3 & 1 \end{matrix} \qquad \text{Distance} = 14$$

Clearly this noth the best metric as we are having 14 distance for the same letter L.

Metric2:

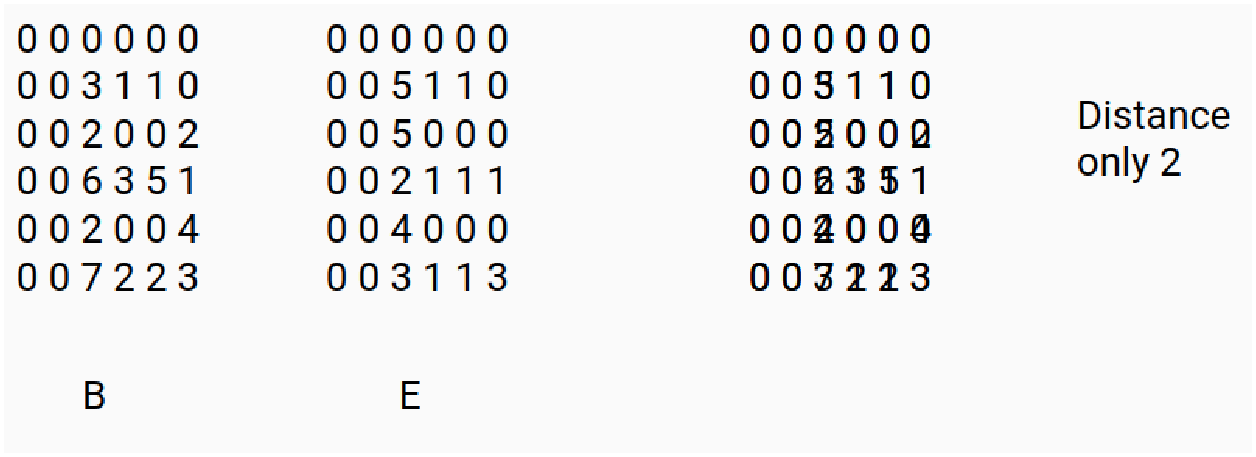
Shifting origin of one matrix and finding difference using Metric 1 and finding the least of it



```
0 4 0 0 0 0      0 0 4 0 0 0
0 2 0 0 0 0      0 0 2 0 0 0
0 2 0 0 0 0      0 0 1 0 0 0
0 9 1 1 3 0      0 0 1 0 0 0
0 0 0 0 0 0      0 0 4 0 0 0
0 0 0 0 0 0      0 0 5 1 3 1
```

Clearly this is better metric than first one

Drawback:



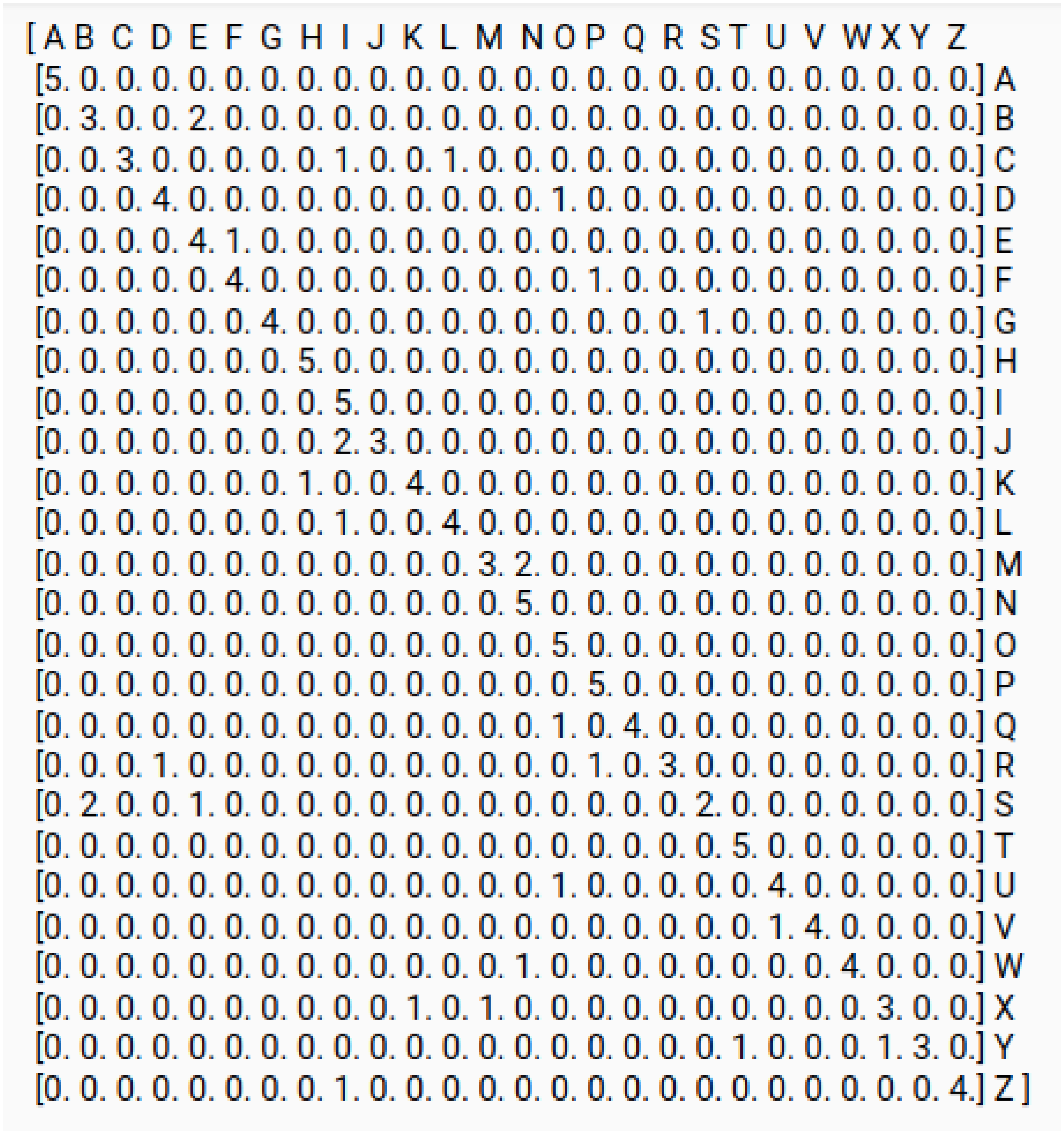
```
0 0 0 0 0 0      0 0 0 0 0 0
0 0 3 1 1 0      0 0 5 1 1 0
0 0 2 0 0 2      0 0 5 0 0 0
0 0 6 3 5 1      0 0 2 1 1 1
0 0 2 0 0 4      0 0 4 0 0 0
0 0 7 2 2 3      0 0 3 1 1 3

     B                E
```

Even for different alphabets we are getting small distance

Results:

We ran Accuracytest.py with K=3 and we got 77.7% accuracy which is low. We expected something near 95%.



```
[A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 [5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] A
 [0. 3. 0. 0. 2. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] B
 [0. 0. 3. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] C
 [0. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] D
 [0. 0. 0. 0. 4. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] E
 [0. 0. 0. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] F
 [0. 0. 0. 0. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.] G
 [0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] H
 [0. 0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] I
 [0. 0. 0. 0. 0. 0. 0. 0. 2. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] J
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] K
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] L
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 2. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] M
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] N
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] O
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] P
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0.] Q
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0.] R
 [0. 2. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. 0. 0.] S
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 0. 0. 0. 0. 0. 0.] T
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 4. 0. 0. 0. 0. 0.] U
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 4. 0. 0. 0. 0.] V
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 4. 0. 0. 0.] W
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0.] X
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 3. 0.] Y
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 4.] Z ]
```

Dependency matrix

In the above image (i,j) of the matrix represents the number of characters of i that were recognised as j.

What we could have done better:

- We didn't used weight of matrix entry may be by using that we could have go better results
- We could have used direction of movement of finger as data that may give good distingustion between alphabets
- Maybe we could have collected more data

References Used:

https://www.hackster.io/gatoninja236/capacitive-touch-sensing-grid-f98144

https://www.sparkfun.com/datasheets/Components/MPR121.pdf