Contents:

1. Introduction

1.1. Objectives & Goals

To design a keyboard interface device, that facilitates non-visual input guidance, enables faster input feeding and helps redistribute focus on other vital device peripherals.

1.2. Applications

- Non-visual Keyboard interface
  Useful in retaining focus on peripherals which demand more visual attention

- Re-configurable tap gestures, can be customised for interactive applications like games and design software

- Fast-typing, post gesture familiarization

1.3. Features

- USB compatibility

  Our system uses the Arduino Leonardo which has its own USB interface and can be recognized as a PS2 input device. Plug and play as well as portability are inferred facts as a result.

- Gesture reconfigurability

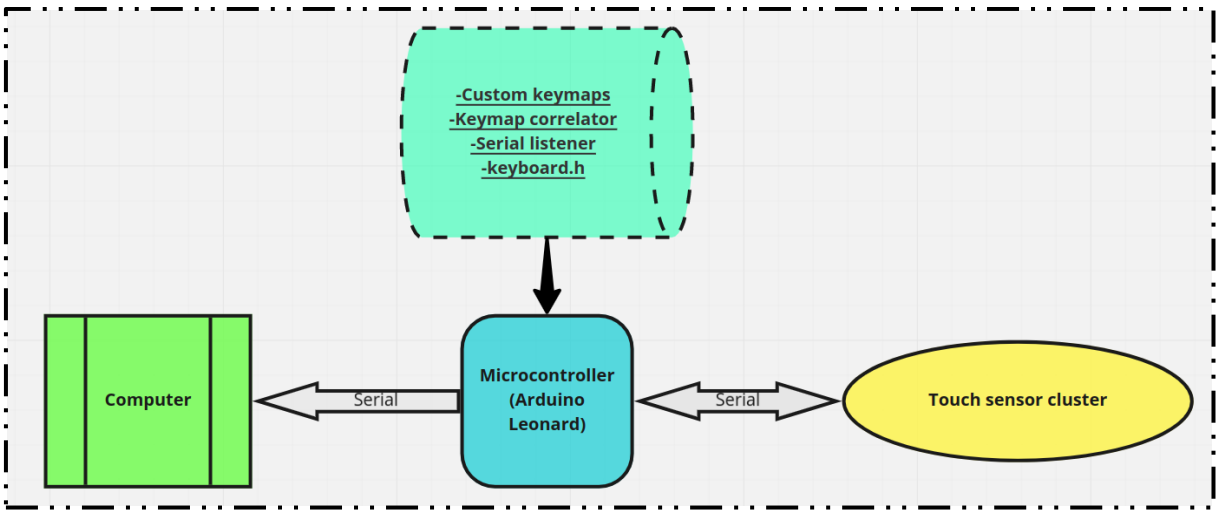
  The embedded software is simple and APIs can be made to reconfigure the input gestures and the corresponding characters they render.

2. System design & analysis

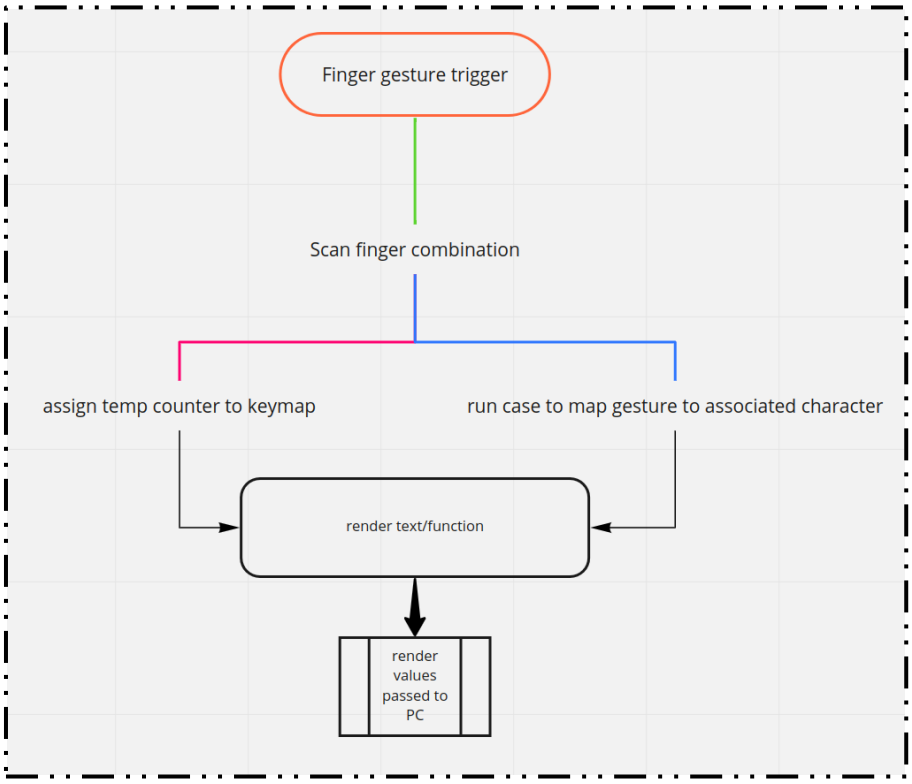
2.1. Block Diagrams

Shown below are the block diagrams for the hardware and software architectures of the system to be designed.

Hardware Architecture



Software Architecture



2.2. Hardware Analysis

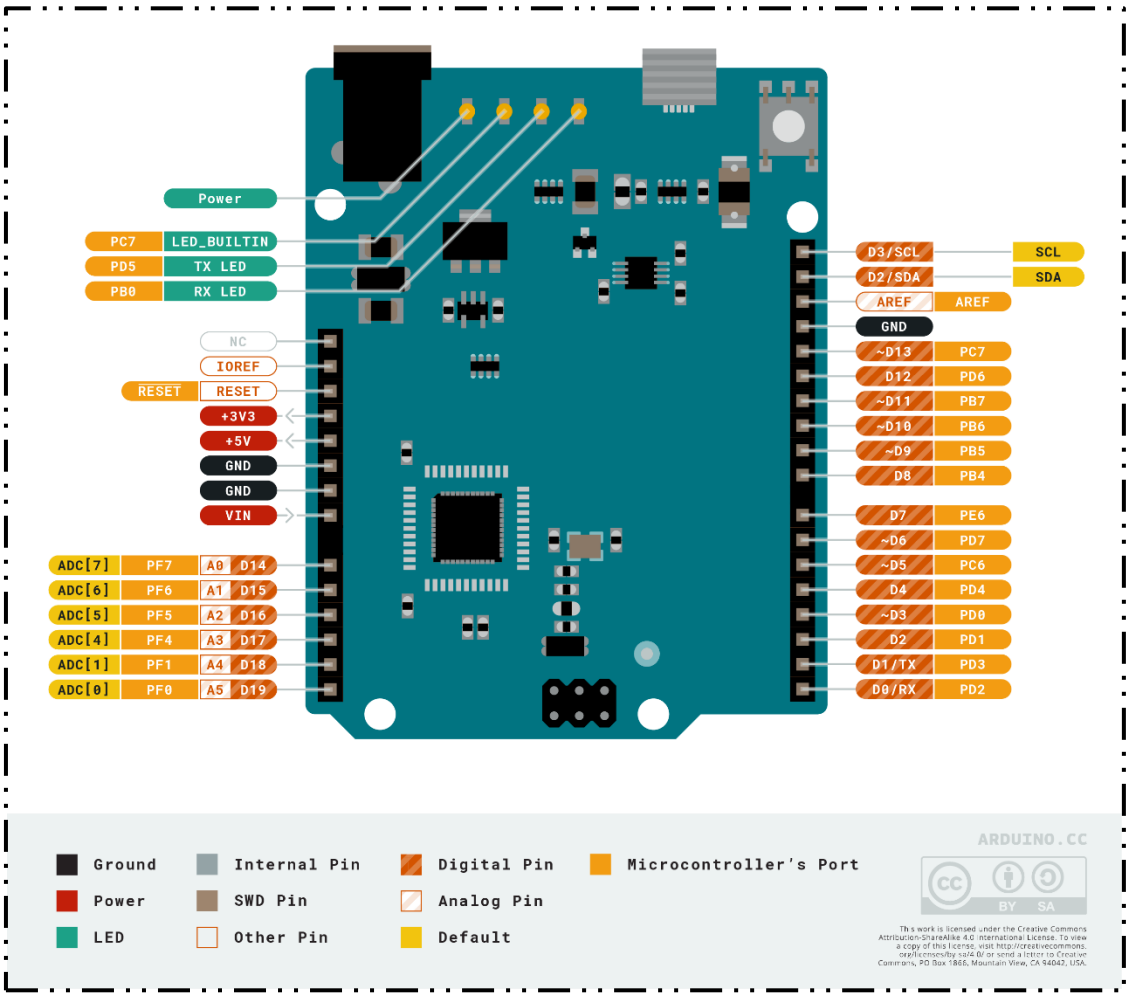
2.2.1. Core Unit

The core unit used is an Arduino Leonardo.

Features

- The Arduino Leonardo is a microcontroller board based on the ATmega32u4.
- It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs),
- a 16 MHz crystal oscillator,
- a micro USB connection,
- a power jack,
- an ICSP header,
- and a reset button.

A detailed board fritzing illustration



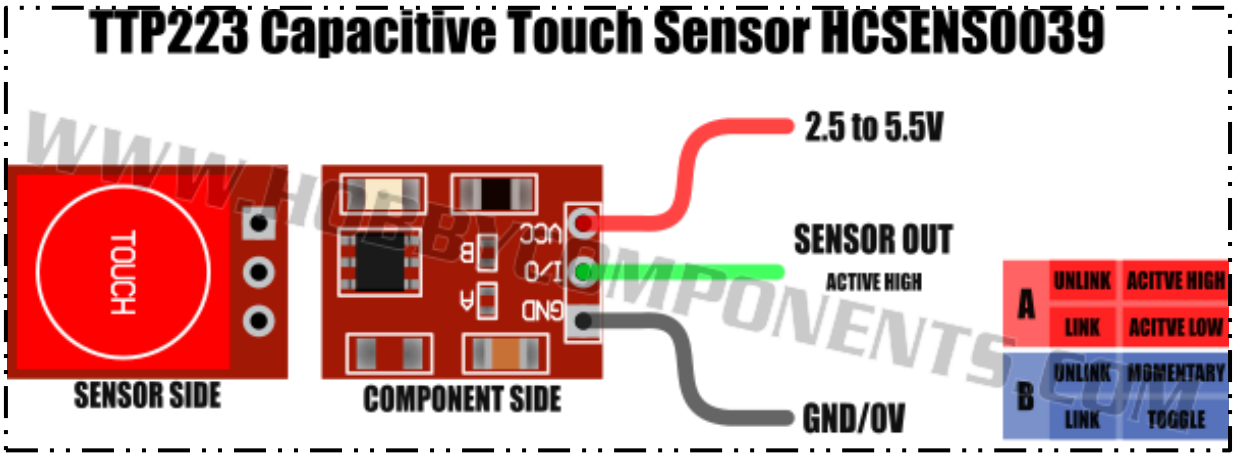
2.2.2. Peripherals

The TTP223 touch sensor is used to take in the input feed.

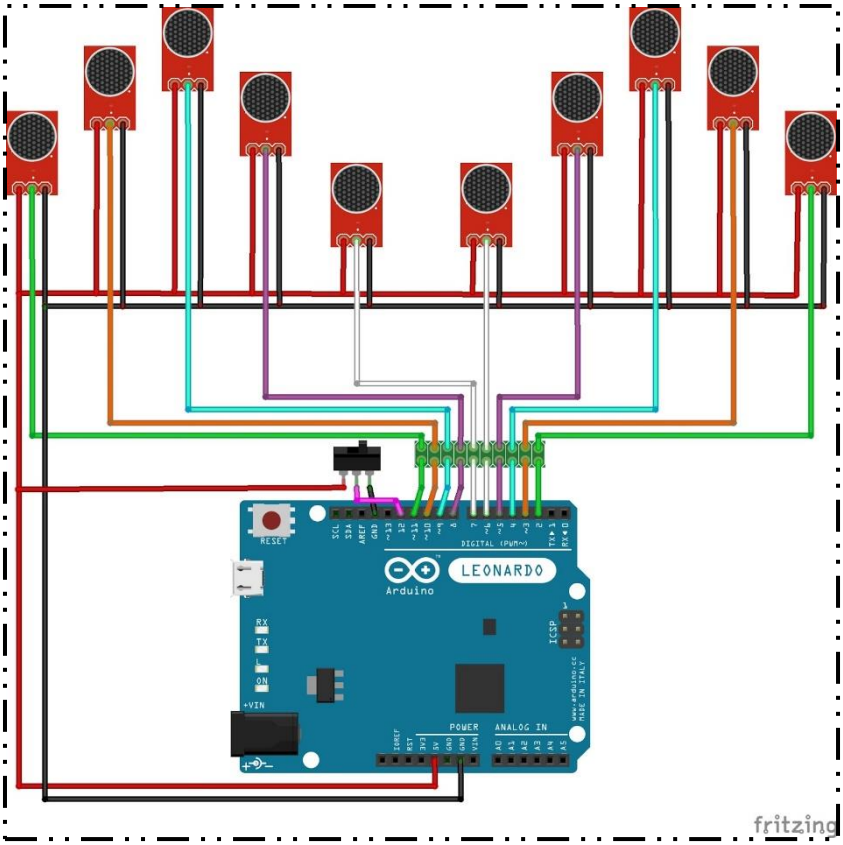
Features

- Single channel touch key sensing
- 2.0V - 5.0V Operating Voltage
- Operating current 1.5uA typically. Maximum operating current 3.0uA at Low power mode.
- Response time - 220ms at low power mode.
- Adjustable sensitivity using external capacitor (0-50 pF)
- Stable touch detection of human body
- All output modes can be configurable for high or active low by pad options.
- Direct mode and Toggle mode selection pin available

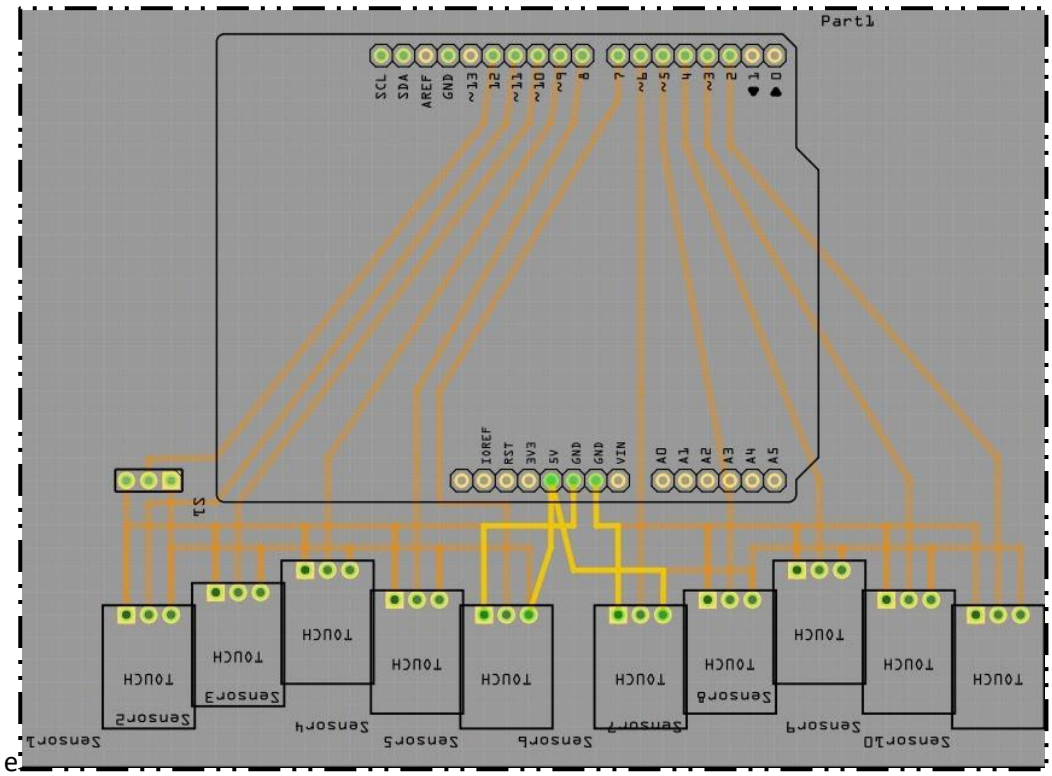
A detailed sensor illustration



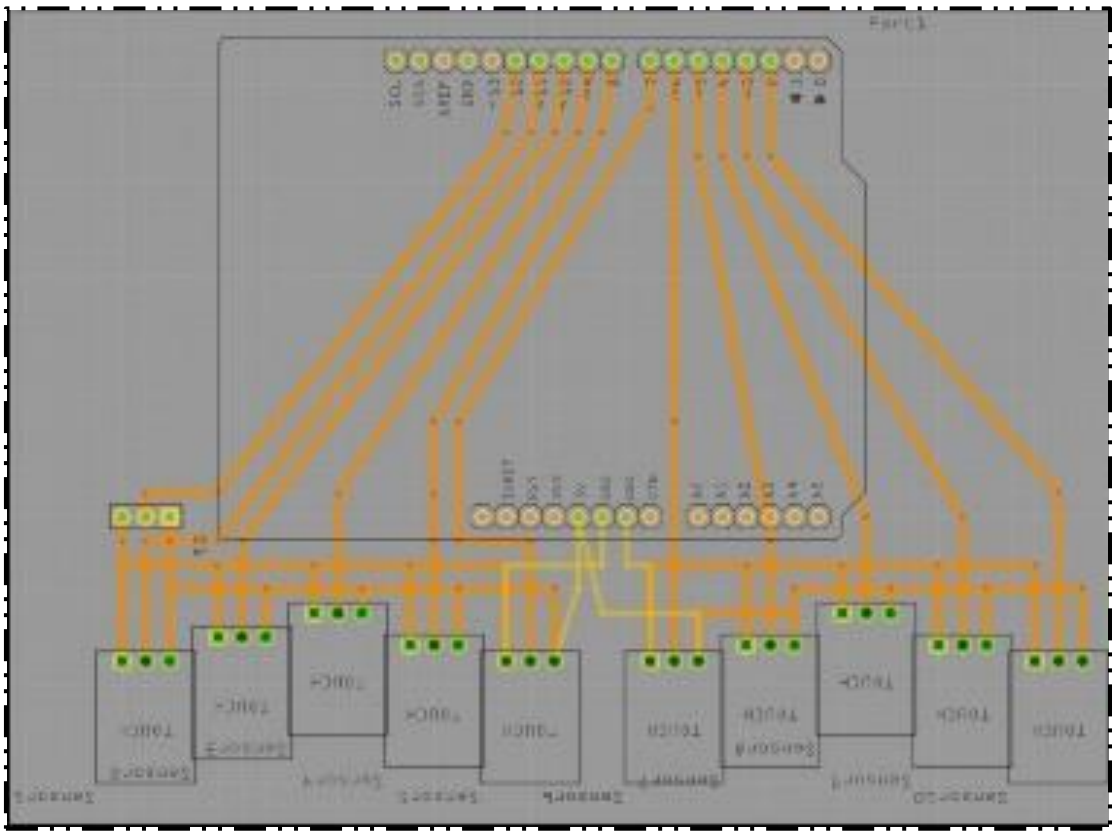
➔ Design Schematic:



➔ PCB Schematics:



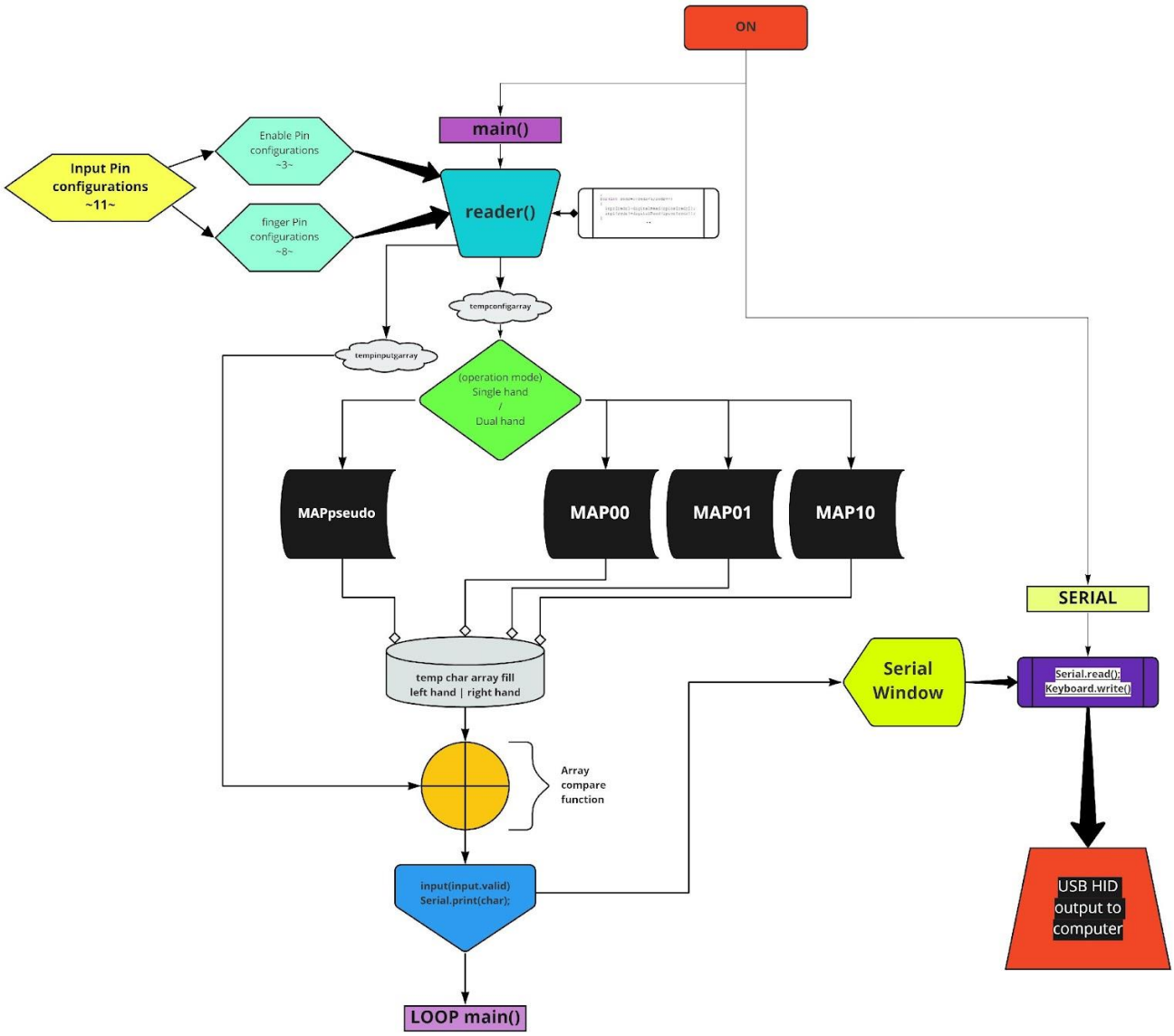
TOP Layer



Bottom Layer

3.Software/coding Analysis

3.1. Algorithm



The flowchart shown above involves certain complex mapping behaviours as well as input output management. Its understandable once reviewed keenly.

3.2. Simulation/code analysis

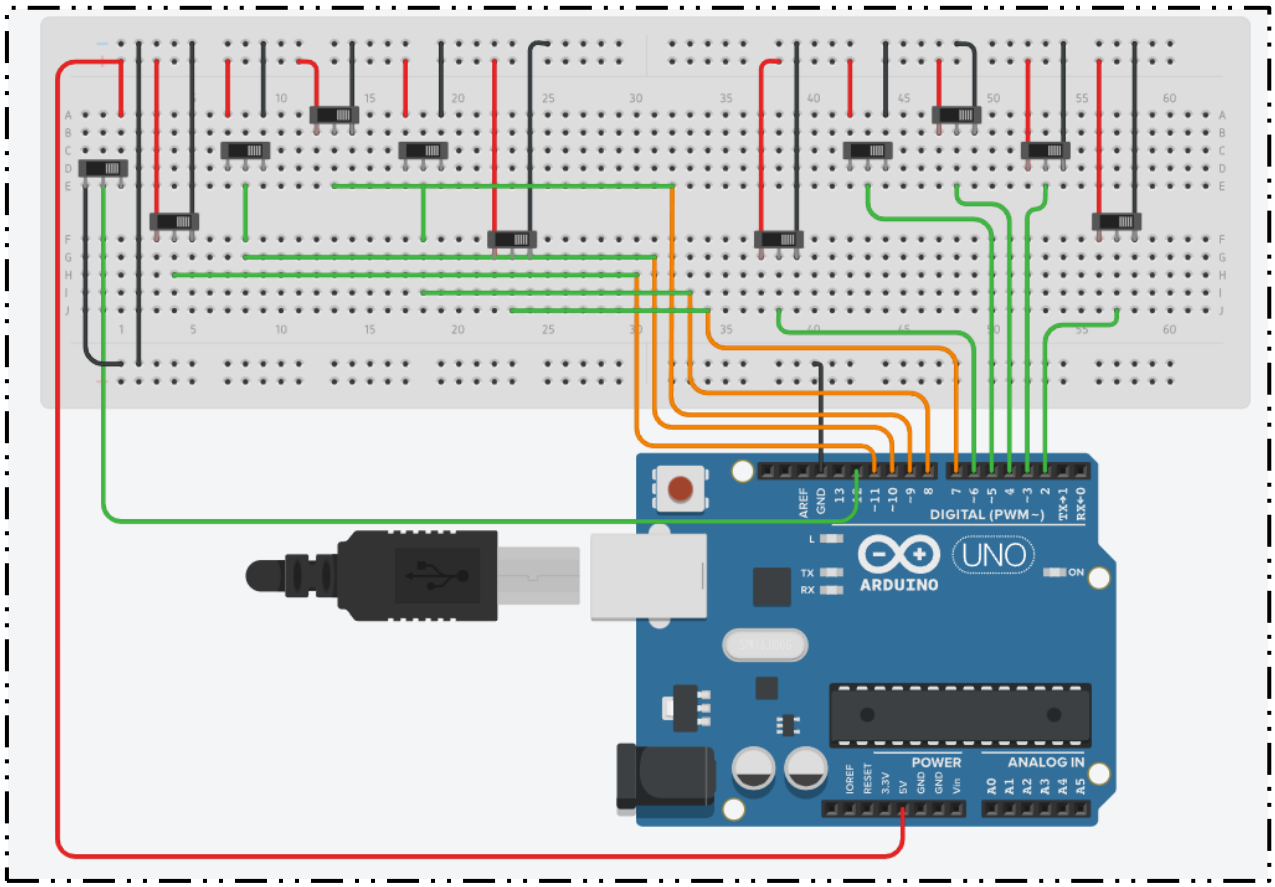
The simulation of the design will be done on Tinkercad. The design is easy and can be implemented with the non-USB counterpart of the Leonardo and certain features' excluded microcontroller, the Arduino UNO, as it is the only choice offered in the web application.

The function of the capacitive touch sensors can be probed (function emulation) using the 3 terminal slide switches.

Tinkercad Share link

<<https://www.tinkercad.com/things/bh4b9EvrlHo-esdjpro/editel?sharecode=MlDAtb3MiKJ_xc2fJjAStAs4xgwHUak0OgWpz9L-tSY>>

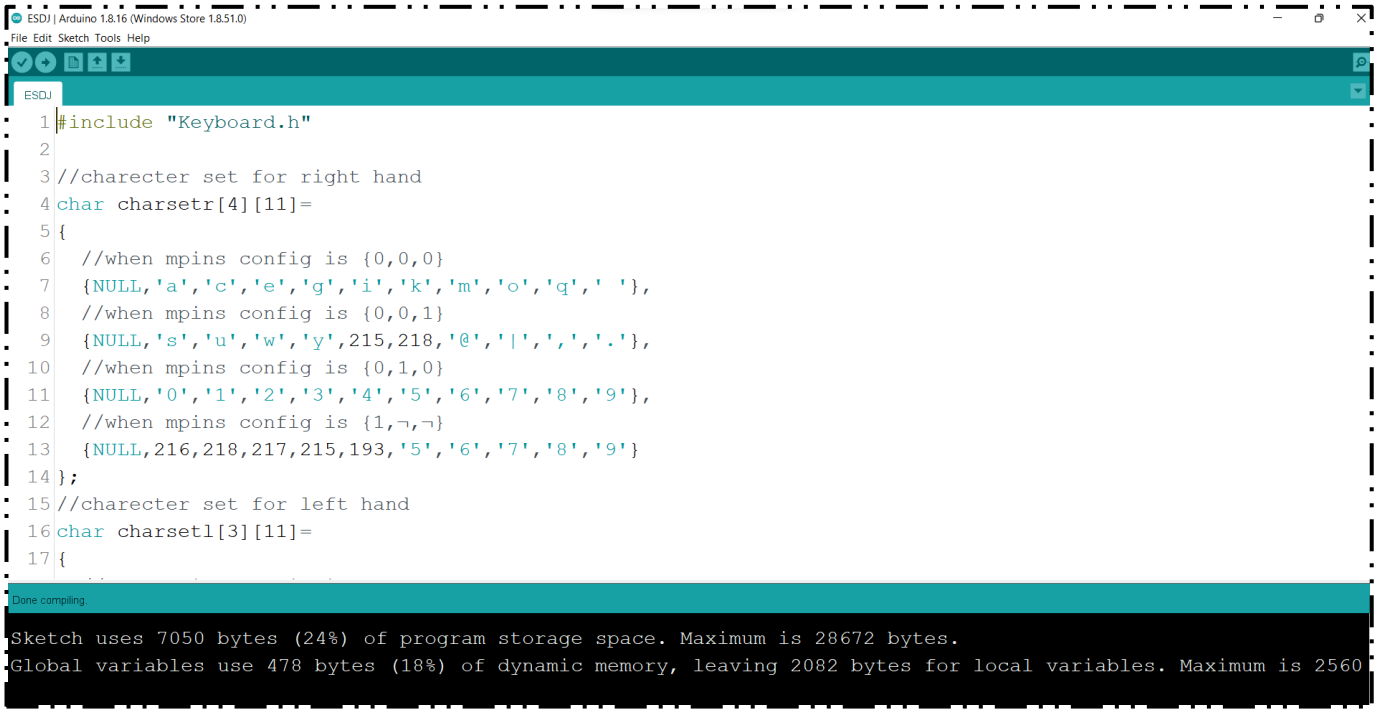
Tinkercad Circuit



➔Software design deployment

The code will be written to the Arduino using the Arduino IDE.

Code compiles successfully



➔Code Blocks analysis *244 lines* (read //comments)

```
#include "Keyboard.h"
//charecter set for right hand
char charsetr[4][11] =
{
  //when mpins config is {0,0,0}
  {NULL, 'a', 'c', 'e', 'g', 'i', 'k', 'm', 'o', 'q', ' '},
  //when mpins config is {0,0,1}
  {NULL, 's', 'u', 'w', 'y', 215, 218, '@', '|', ',', '.'},
  //when mpins config is {0,1,0}
  {NULL, '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
  //when mpins config is {1,¬,¬}
  {NULL, 216, 218, 217, 215, 193, '5', '6', '7', '8', '9'}
};
//charecter set for left hand
```

```
char charset1[3][11] =
{
  //when mpins config is {0,0,0}
  {NULL, 'b', 'd', 'f', 'h', 'j', 'l', 'n', 'p', 'r', 178},
  //when mpins config is {0,0,1}
  {NULL, 't', 'v', 'x', 'z', 216, 217, '!', '#', '"', '?'},
  //when mpins config is {0,1,0}
  {NULL, '+', '-', '*', '/', '^', '%', '(', ')', '~', '='},
};

//right hand config pins
int rpins[] = {6, 5, 4, 3};
//left hand config pins
int lpins[] = {10, 9, 8, 7};
//map enable config pins
int mpins[] = {11, 2, 12};
//right hand enable matrix
int rconf[11][4] =
{
  {0, 0, 0, 0},
  {1, 0, 0, 0}, //1
  {0, 1, 0, 0}, //2
  {0, 0, 1, 0}, //3
  {0, 0, 0, 1}, //4
  {1, 1, 0, 0}, //5
  {0, 1, 1, 0}, //6
  {0, 0, 1, 1}, //7
  {1, 1, 1, 0}, //8
  {0, 1, 1, 1}, //9
  {1, 1, 1, 1}, //10
};
//left hand enable matrix
int lconf[11][4] =
{
  {0, 0, 0, 0},
  {0, 0, 0, 1}, //1
  {0, 0, 1, 0}, //2
  {0, 1, 0, 0}, //3
```

```
  {1, 0, 0, 0}, //4
  {0, 0, 1, 1}, //5
  {0, 1, 1, 0}, //6
  {1, 1, 0, 0}, //7
  {0, 1, 1, 1}, //8
  {1, 1, 1, 0}, //9
  {1, 1, 1, 1}, //10
};
//temporary storage variables and arrays
int tempr;
int templ;
int temparr[4];
char tempchar[11];
char tempchal[11];
//


//input configuration arryas

//right hand
int inpr[4];
//left hand
int inpl[4];
//enable pins (little fingers, folding)
int inpm[2];


//pinMode initialization
void setup()
{
  for (int i = 0; i < 4; i++)
  {
    pinMode(rpins[i], INPUT);
    pinMode(lpins[i], INPUT);
    pinMode(mpins[i], INPUT);
  }
  Serial.begin(9600);
  Keyboard.begin();
}
```

```
void loop()
{
  moin();

  if (Serial.available() > 0)
  {
    // read incoming serial data:
    char inChar = Serial.read();
    // Type the next ASCII value from what you received:
    Keyboard.write(inChar);
  }
}//main method starts here
void moin()
{
  rest();//reset method
  reader();
  //char ack
  templ = arrary(inpl, lconf);
  tempr = arrary(inpr, rconf);
  if (templ < 11 && tempr < 11)
  {
    printer();
    delay(1000);
  }
  rest();
}//reset method starts here
void rest()
{
  for (int i = 0; i < 4; i++)
  {
    digitalWrite(rpins[i], 0);
    digitalWrite(lpins[i], 0);
  }

  for (int i = 0; i < 3; i++)
  {
    digitalWrite(mpins[i], 0);
  }
```

```
}//print method starts here
void printer()
{
  switch (templ)
  {
    case 0:
      break;
    default:
      Serial.println(tempchal[templ]);
      break;
  }
  switch (tempr)
  {
    case 0:
      break;
    default:
      Serial.println(tempchar[tempr]);
      break;
  }
}//array comparison reference
int arrary(int mp[4], int kol[11][4])
{
  int booga, flag, k;
  for (k = 0; k < 11; k++)
  {
    for (int j = 0; j < 4; j++)
      temparr[j] = kol[k][j];
    booga = compareArray(mp, temparr, 4);
    if (booga == 0)
    {
      flag = k;
      break;
    }
    else
      continue;
  }
  return flag;
}
```

```
//character mapping functions
void map00()
{
  for (int t = 0; t < 11; t++)
    tempchar[t] = charsetr[0][t];
  for (int t = 0; t < 11; t++)
    tempchal[t] = charsetl[0][t];
}
void map01()
{
  for (int t = 0; t < 11; t++)
    tempchar[t] = charsetr[1][t];
  for (int t = 0; t < 11; t++)
    tempchal[t] = charsetl[1][t];
}
void map10()
{
  for (int t = 0; t < 11; t++)
    tempchar[t] = charsetr[2][t];
  for (int t = 0; t < 11; t++)
    tempchal[t] = charsetl[2][t];
}
//input reader method starts here
void reader()
{
  for (int i = 0; i < 4; i++)
  {
    inpr[i] = 0;
    inpl[i] = 0;
  }
  for (int redr = 0; redr < 4; redr++)
  {
    inpr[redr] = digitalRead(rpins[redr]);
    inpl[redr] = digitalRead(lpins[redr]);
  }
  for (int redm = 0; redm < 3; redm++)
    inpm[redm] = digitalRead(mpins[redm]);
```

```
  if (inpm[2] == 0)
  {
    for (int t = 0; t < 11; t++)
      tempchar[t] = charsetr[3][t];
    for (int t = 0; t < 11; t++)
      tempchal[t] = NULL;
  }
  else
  {
    switch (inpm[0])
    {
      case 0:
        switch (inpm[1])
        {
          case 0:
            map00();//Serial.println("map00");
            break;
          case 1:
            map01();
            //Serial.println("map01");
            break;
        }
        break;
      case 1:
        map10();
        //Serial.println("map10");
        break;
    }
  }
}
int compareArray(int a[], int b[], int size)  {
  int i;
  for (i = 0; i < size; i++) {
    if (a[i] != b[i])
      return 1;
  }
  return 0;
}
```

4. Conclusion & Future work

4.1. Conclusion

The project was modelled and integrated successfully, minor connection issues were faced in component interconnections.

The final project works as intended, matches simulation outputs and satisfies the goal of functioning as a Tap Keyboard emulator.

4.2. Cost Analysis

Considering base pay standard for a design engineer as 100 INR for a duration of 1hr.

➔ Design Cost

Design phase	Time spent (hrs.)	Est. Cost (INR)
Requirements analysis	2	200
Specifications	1	100
Hardware & Software partitioning	2	200
Architecture	2	200
System Integration	6	600
NET	13	1300

➔ Components Cost

Design phase	Qty.	Cost (INR)
Arduino Leonardo ETH	2	2500
TTP223 touch sensors	10	220
NET		2720

Total Estimated cost = 4020 INR

➔ Future work

The existing design can be upgraded in the future to take analog values by using accelerometers in the place of the touch sensors to increase input precision and enable further gesture customization with 3D gestures when used along with an integrated gyroscope.

REFERENCES:

Books:

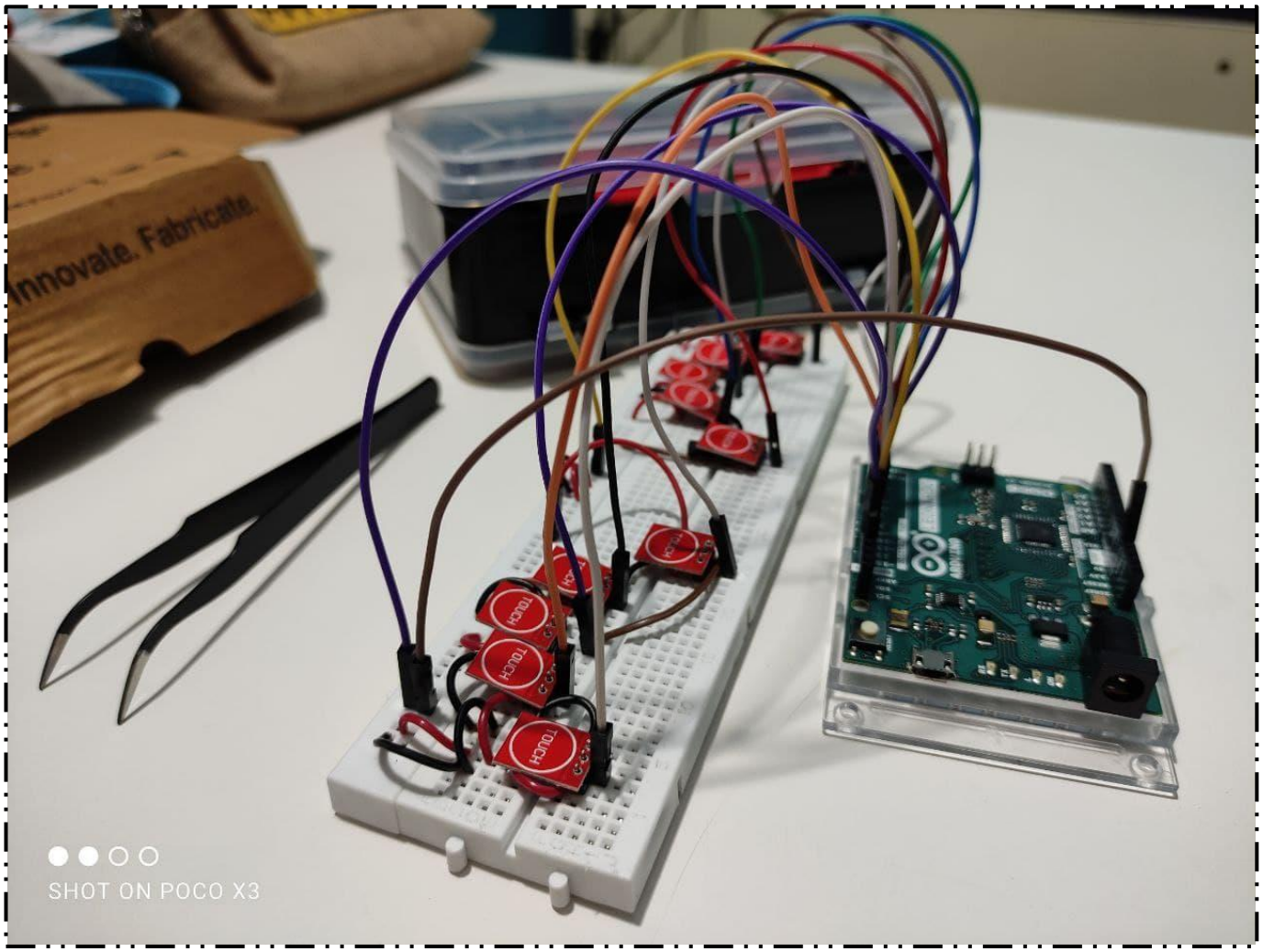
- **Computers as Components: Principles of Embedded Computing System Design (The Morgan Kaufmann Series in Computer Architecture and Design)**

Web sources

- Keyboard Serial | Arduino
- Programming the Arduino Leonardo (msu.edu)
- Arduino - KeyboardModifiers
- Keyboard - Arduino Reference

Team Final Results:

Candid Shot



Top view