

Hand-Sign recognition Software

1 Structure

MST_Z software requires the below files for proper functioning

- MST_Z_PREDICT.exe
- main.exe
- Images

MST_Z_PREDICT.exe file contains user friendly GUI support which displays useful information like gesture representation, dynamic representation, sign number, calibration, and connection status etc.

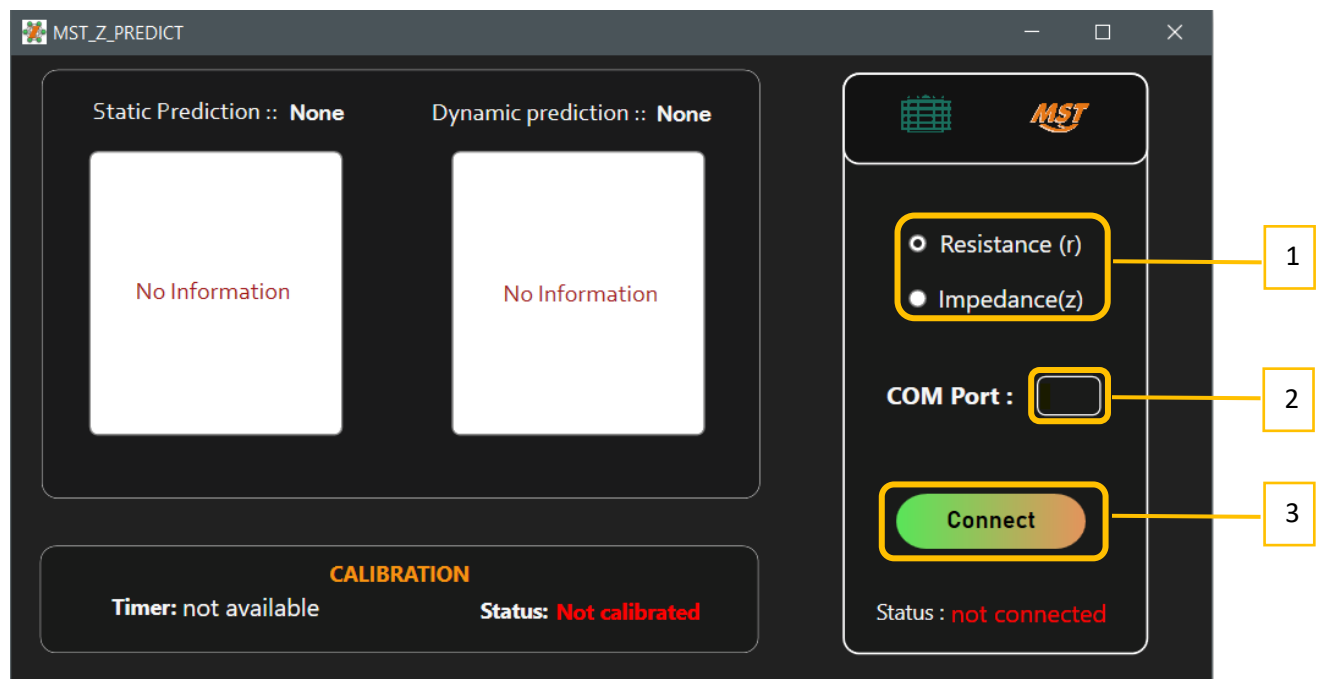
main.exe file contains low level C++ code to interact with microcontroller over serial communication at 115200 bps speed

images folder contains all the pictural representations of the hand signs currently used. Main GUI software takes input from images folder as a reference to show images during operation

2 How to use

please make sure proper firmware version flashed in microcontroller. Verify the COM port being used by microcontroller. In case of no COM port allotted, please install the driver from https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip

once everything setup, please open MST_Z_MESH.exe file. It gives user interface to communicate with the microcontroller. Select resistance/Impedance method, Enter COM port and connect to the board






3 Required software for modifications

3.1 Visual studio

Modifications of GUI application requires visual studio install in PC. If not, please install from <https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&passive=false&cid=2030>

After successful installation, please clone or download source code from git repository https://github.com/nagka98/cpp_app/tree/4ch

Once downloaded, open software by double clicking app1.sln file

 .vs	09-06-2022 19:59	File folder	
 app1	12-06-2022 13:53	File folder	
 app1.sln	12-04-2022 16:53	Visual Studio Solut...	2 KB

3.2 Make, GCC and G++ compiler

Gcc and g++ compilers are natively available in Linux environment, but we need to install it in windows. Please download Msys2 software from https://github.com/msys2/msys2-installer/releases/download/2022-06-03/msys2-x86_64-20220603.exe

- After successful installation, open msys2 and type following command
- `pacman -S mingw-w64-x86_64-gcc`
- and install gdb
- `pacman -S mingw-w64-x86_64-gdb`
- and install 'make'
- `pacman -S make`
- after execution of above commands, close all msys2 windows
- please clone low level source code from git repo https://github.com/nagka98/CLapp_SF
- after download, open msys2 and change directory "`cd {your path}/Clapp_SF`"
- enter command "`make`"
- after successful completion, you will find main.exe file in your folder

3.3 Firmware

Download source code from repo https://github.com/nagka98/ESP_IDF/tree/5ch

Open README file and follow the instructions

4 Procedure to add new sign

4.1 Pictural representation

Please find appropriate sign picture and place it under images folder. Give it appropriate name

4.2 Low level changes

Current low level C++ code provides R_Elim array of size 10. Each element represents individual sensor bit after calibration. This can be found on 25th line Clapp_SF/main.cpp

```
24 //Resistor proterties
25 int R_elim[10] = {0,0,1,0,0,1,0,1,1,0};
```

Based on the complexity of hand-sign, please enable appropriate filaments

The software gives series of 1's and 0's corresponding to total enabled filaments in R_Elim array. Please formulate appropriate code based on custom hand sign

After successful changes, execute make command in msys2 terminal (check 3.2 section)

Once main.exe file generated, please enter “./main.exe COM R_En I_En” in msys2

Example:

```
$ ./main.exe 3 0 1
```

3 is COM PORT,

0 is resistance disable,

1 is impedance enable

(don't enable both R&Z)

This gives you timer on screen

- First timer is for board configuration time
- Second timer is for calibration 1
- Third timer is for calibration 2
- After timer, series of 1's and 0's start appearing on screen

```
0
0
0
0
0
0
0
0
0
0
0
0
0
calib_done
1101
1101
1001
1001
1001
1001
1001
1001
1001
1001
1001
1001
1001
1001
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
0001
1111
```

- Please calibrate carefully and verify the appropriate code for you hand sign
- Generally, 1 indicates finger-joint open and 0 indicates close
- Once testing done, please note down corresponding codes
- Copy main.exe and replace with old main.exe

4.3 GUI changes

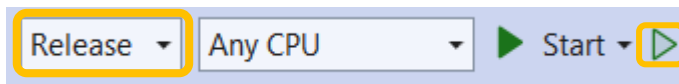
- Please open app1.sln file in cpp_app/app1 location (refer 3.1)
- Please open "Mainwindow.xaml.cs" file and find the sw2itch logic on line 104
- Add new case with custom code and modify image source and image label

```

case "1011":
prediction_label.Content = "8";
//imagebox.Source = new BitmapImage(new Uri("C:/Users/Mani/Desktop/thesis/ASL-Numbers/ASL-8.png"));
imagebox.Source = new BitmapImage(new Uri("pack://siteoforigin:,,,/images/ASL-8.png"));
break;
case "0111":
prediction_label.Content = "9";
//imagebox.Source = new BitmapImage(new Uri("C:/Users/Mani/Desktop/thesis/ASL-Numbers/ASL-9.png"));
imagebox.Source = new BitmapImage(new Uri("pack://siteoforigin:,,,/images/ASL-9.png"));
break;
case "0000":
prediction_label.Content = "10";
//imagebox.Source = new BitmapImage(new Uri("C:/Users/Mani/Desktop/thesis/ASL-Numbers/ASL-10.png"));
imagebox.Source = new BitmapImage(new Uri("pack://siteoforigin:,,,/images/ASL-10.png"));
break;
case "xxxxx":
prediction_label.Content = "xx";
//imagebox.Source = new BitmapImage(new Uri("C:/Users/Mani/Desktop/thesis/ASL-Numbers/ASL-10.png"));
imagebox.Source = new BitmapImage(new Uri("pack://siteoforigin:,,,/images/ASL-xx.png"));
break;

```

- Save it, build it on Release profile available on Menu bar



- After successful build, find MST_Z_PREDICT.exe file on "cpp_app/app1/app1/bin/release" location
- Copy it and replace old executable file

Once all changes complete, follow normal procedure to initiate application and test your sign