**FINAL YEAR PROJECT**

****

**Abdul Ahad Mushir(**[**a.ahad.mushir@gmail.com**](mailto:a.ahad.mushir@gmail.com)**)**

**Sandhya.(**[**s.gopchandani@gmail.com**](mailto:s.gopchandani@gmail.com)**)**

**Rashida Moiz(rashida.moiz@hotmail.com)**

**Mahwash Merchant(**[**merchant.mahwash@gmail.com**](mailto:merchant.mahwash@gmail.com)**)**

Table of Contents

[Introduction: 3](#_Toc451435435)

[High Level Design: 4](#_Toc451435436)

[Rationale and sources of your project idea 4](#_Toc451435437)

[Logical structure 4](#_Toc451435438)

[Background 4](#_Toc451435439)

[Hardware / Software tradeoffs 5](#_Toc451435440)

[Relationship with available past projects or standards 5](#_Toc451435441)

[Patents, copyright and trademarks 5](#_Toc451435442)

[Software / Hardware Design: 6](#_Toc451435443)

[Overview 6](#_Toc451435444)

[Program Details: 7](#_Toc451435445)

[Overview 7](#_Toc451435446)

[Errors 7](#_Toc451435447)

[Trails and tests 7](#_Toc451435448)

[Hardware Details: 8](#_Toc451435449)

[Overview 8](#_Toc451435450)

[User interface hardware 8](#_Toc451435451)

[Things that did not work 8](#_Toc451435452)

[Trails and tests 8](#_Toc451435453)

[Results & Conclusion: 9](#_Toc451435454)

[Acknowledgements: 11](#_Toc451435455)

[References: 12](#_Toc451435456)

[Appendix 1: Code: 13](#_Toc451435457)

[Appendix 2: Schematic of your hardware: 35](#_Toc451435458)

[Appendix 3: Software/parts list: 36](#_Toc451435459)

[SOFTWARE: 36](#_Toc451435460)

[HARDWARE: 36](#_Toc451435461)

[Appendix 4: Work distribution: 37](#_Toc451435462)

[Appendix 6: Project timeline: 38](#_Toc451435463)

# Introduction:

Our project aims to develop an android based application that can assist paralyzed person to control their room environment to some extent. The idea is to make him independent enough so that he becomes able to perform simple tasks without constant assistance. This app will help patients to be independent and not rely on an assistant to be around them 24/7. This application will also reduce the frustration level that is easily known phenomenon among these people.

The application will allow users to operate switches using simply their facial gestures. For example: wink of an eye, will signal the application to switch the fan ON. Controllo is designed specifically for users with very limited physical movement. Because no other muscle, be it hands or legs, is involved, Controllo becomes the ideal solution.

How it works:

This application will run on most versions of Android operating system. The smartphone will be placed in a way where the camera is directly pointed at the user’s face. The app will utilize the camera of the smartphone to capture user gestures.

* User makes a facial gesture
* Camera captures the gesture
* Application processes the gesture to do what it is meant to do

Example:

* Wink = switch the fan ON
* Wink with smile = switch the lights OFF
* Application sends a signal to the arduino through bluetooth
* Arduino receives the signal
* Arduino controls the specific relay (which acts as a switch) to operate the circuit of the relevant appliance connected to it.

Controllo falls under the domain of assistive technology. Although there have been numerous efforts under this domain, there is still a vast area unexplored. This application aims to improve the quality of life of paralyzed people by granting them the power to independently control their room appliances.

# High Level Design:

## Rationale and sources of your project idea

We aim to promote the use of technology to solve a problem in our surroundings. Stats.

Technology for gesture recognition and using Arduino to control the room appliances is available but nothing has been done in the domain of using facial gestures to control the room appliances. We wish to bridge that gap.

Our motivation is to help those who are completely dependent on others and give them a sense of relief of not requiring assistance every time they want to do something. The idea is to make him independent enough so that he becomes able to perform simple tasks without constant assistance by using facial gesture recognition

## Logical structure

## Background

One of the things that frustrate paralyzed people is their inability to perform everyday task such as getting up opening/closing room’s light or fan. It makes them annoyed to call someone every time they need to do something. There are assistive tools available to call the assistant by a click but then again they are dependent on someone. Their own free will is lost and they feel worthless and useless.

This defines the core idea behind our project. Our project is based on assistive technology that will provide assistance to paralyzed patients. Paralysis is a loss of muscle function in any or whole part of the body. We are concerned specifically about those who are paralyzed shoulder down. These are the people who can’t move either their hands or legs and have become bed ridden. Not much work has been done to help these kind of patients. There are tools available that can control the environment with a button click but that

## Hardware / Software tradeoffs

* The application is designed for mobile phones with Android Operating System.
* The application will require to access to camera and Bluetooth, so the Bluetooth needs to be turned on at the beginning for our app to work.
* The application can be run on android froyo or more advanced versions

## Relationship with available past projects or standards

## Patents, copyright and trademarks

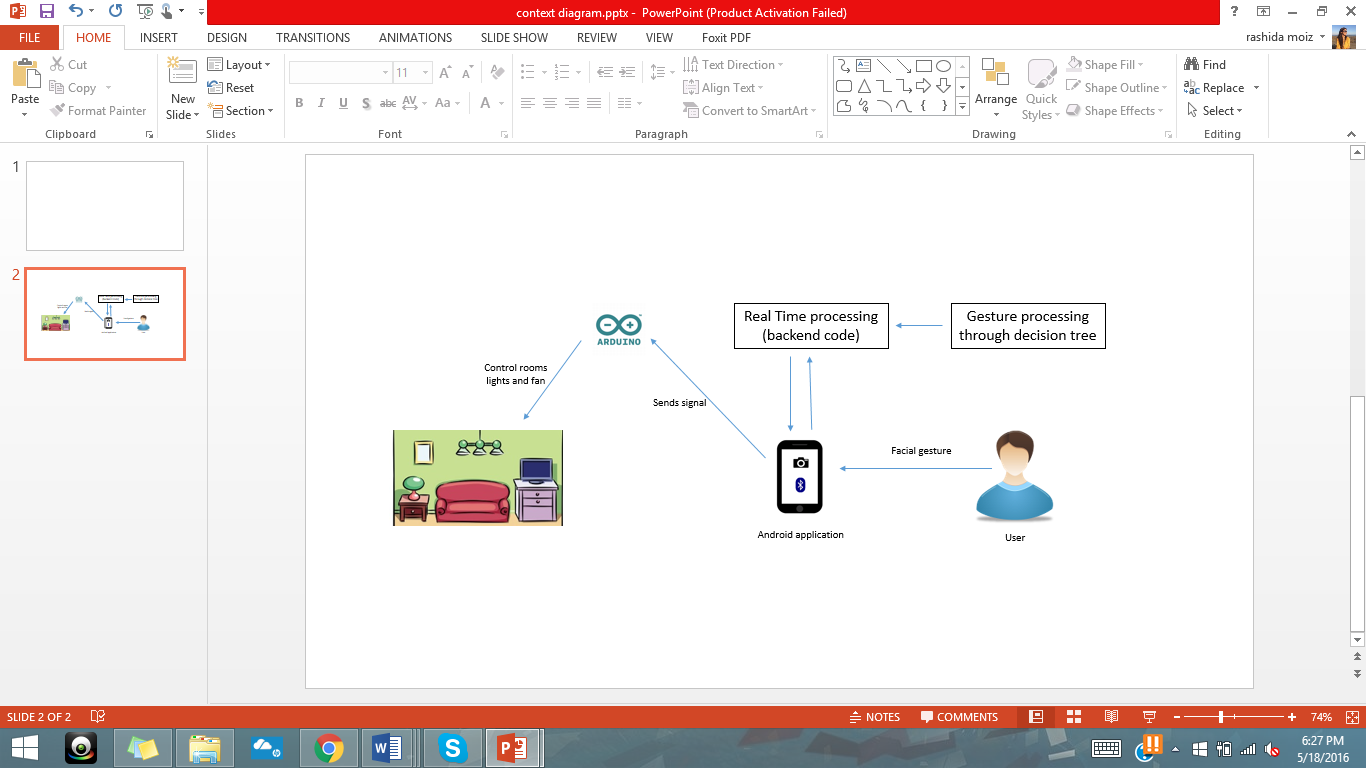
We do not have any patents, copyright or trademark for or related to our project.

# Software / Hardware Design:

## Overview

The functionality of our project depends upon two entities

* Mobile Application with camera and Bluetooth(software)
* Arduino with Bluetooth component and relays(hardware)

This portion will be covering the technicalities and design involved in our mobile application

**Figure Description:**

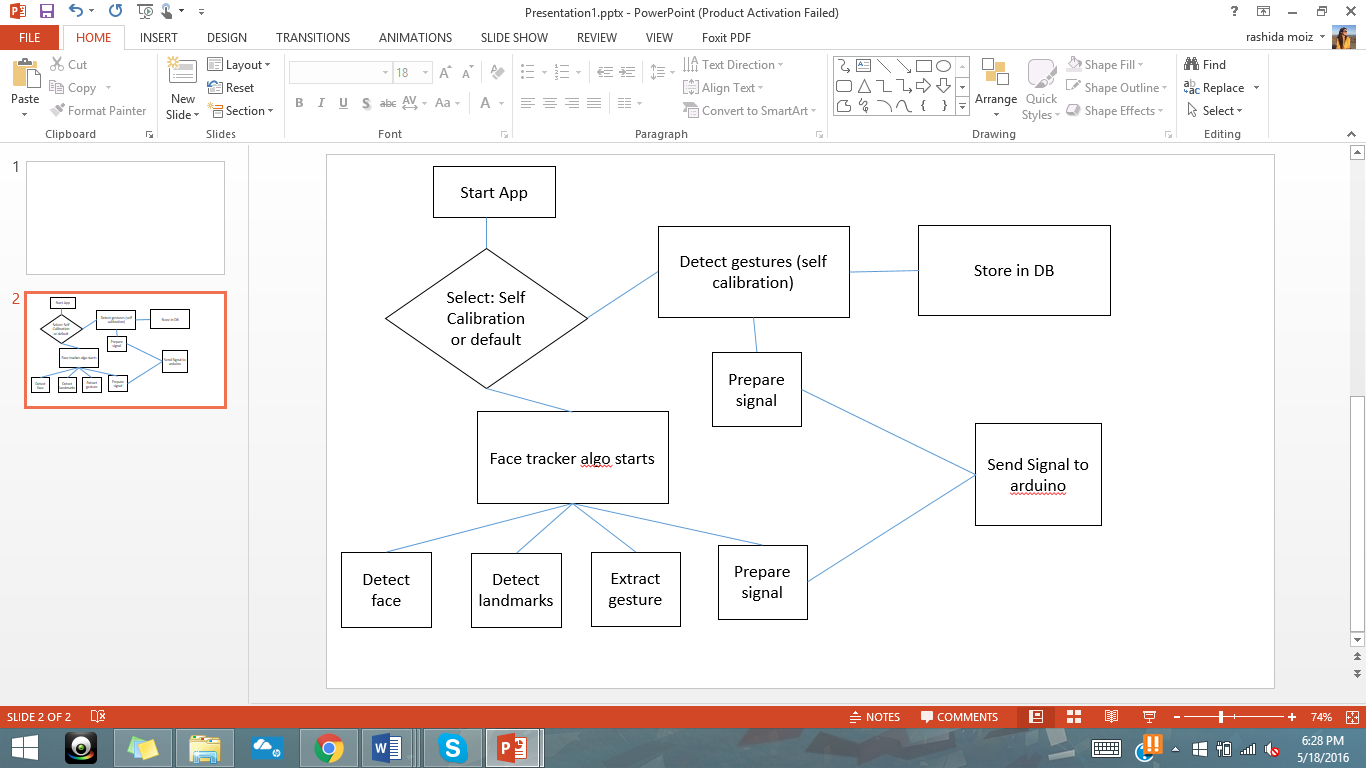
The data is first collected by noting down different calibrations and probability info by making people perform the desired gestures. The data collected is then cleansed manually and is pre-processed in knime using decision tree. Once the acceptable threshold is received its incorporated in the recognition algorithm.

When the app is opened the first thing it does is it to check if bluetooth is on or not. If not, it gives a user a option to turn it on or exit the app. When a gesture is made in front of the camera the app recognizes it and according to the gesture made sends a signal to Arduino through bluetooth.

The arduino is connected through relays in such a way that it can control the rooms switchboard to control the appliances.

## Program Details:

### Overview



### Errors

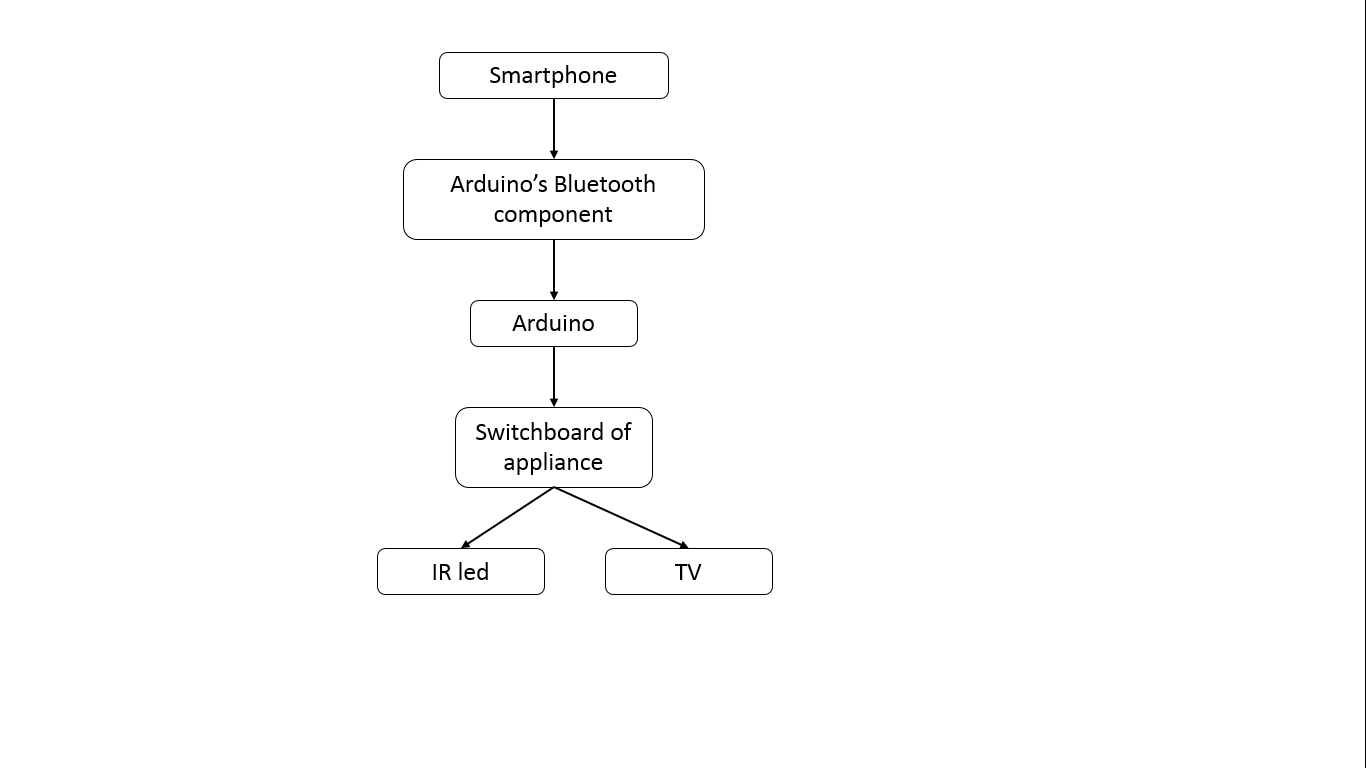
The additional feature introduced i.e. self-calibration to enable user to select gestures themselves have some range selection issues.

### Trails and tests

Testing was not done on all phases and it was followed by every step taken in the progress of development. The final testing was done after the completion of application. Bluetooth component on Arduino can at a time be connected with only one smartphone

## Hardware Details:

### Overview



### User interface hardware

App running on the smart phone.

### Things that did not work

We were not able to implement the functionality of IR to control all IR devices hence restricted it to just controlling the television.

### Trails and tests

# Results& Conclusion:

The motivation behind Controllo was to assist people with motor disabilities; relieving them of constant dependency on others for the smallest tasks. In order to come up with a solution, we had to first understand the problem.

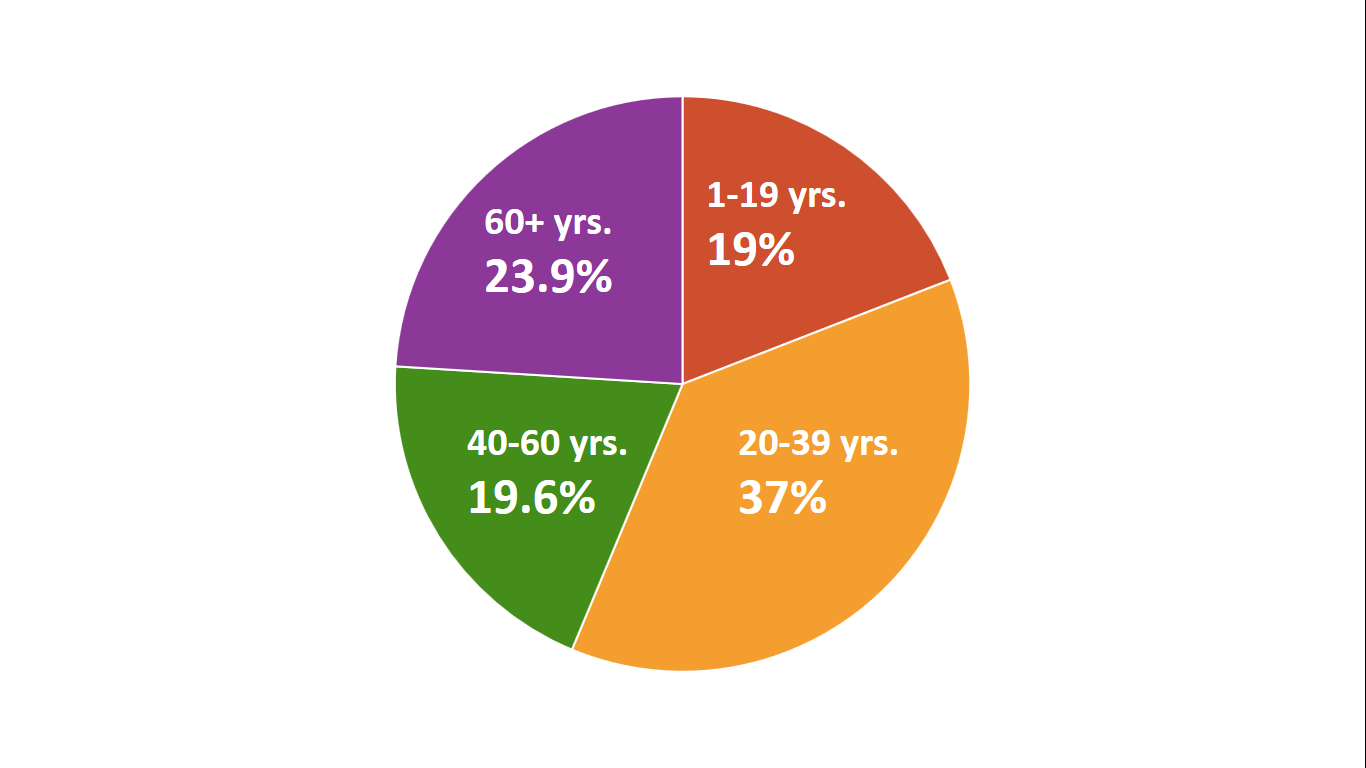
**Questions:** What is motor disability and impairment?

**Answer:** The International Neuromodulation Society, USA, defines Motor Impairment as: “… the partial or total loss of function of a body part, usually a limb or limbs.  This may result in muscle weakness, poor stamina, lack of muscle control, or total paralysis. Motor impairment is often evident in neurological conditions such a cerebral palsy, Parkinson’s disease, stroke and multiple sclerosis.” (Motor Impairment, 2016 International Neuromodulation Society, San Francisco, CA, USA)

After a lot of research into the medical world, visiting doctors and medical specialists, as well a general survey, we can to conclude our user space. Our average user would be a person who is unable to conduct movement of at least one of his limbs, willingly, due to any reason. There may be number of causes of limb dysfunction as a doctor we met at Civil Hospital righty said “the number of causes of dysfunction can go as far as your imagination takes you”.

During our research we realized that there was a lot of US data available but we were determined to find out the statistics about the people around us. So we threw in an online survey, distributed it around the community, to get feedback and here is what we discovered.

**AGE GROUP OF PEOPLE WITH MOTOR DISABILITY**



Surprisingly, roughly 37% of the motor impaired people around us lie between the ages of 20 and 39 years. These people are young adults, to whom the idea of constant dependency would be highly frustrating. The need for Controllo was very clear. All that was left was to figure out the details.

Defining our user space helped narrow the scope down to investigate into facial gesture recognition tools and techniques. We stumbled upon an android based API called the VISION API. Vision API is designed to mark the landmarks upon recognising a human face. We used those landmarks to identify the user’s gestures and attached a microcontroller to do the hardware work.

Thinking about how many people Controllo would end up helping, was the real motivation that got us through the difficult challenges in our project. As far as our final year project goals were and as far as our resources could take us, we believe we have met them fairly. We have experimented with a diverse range of technologies, from hardware microcontrollers to software IDEs, which gave us a lot of knowledge and exposure. We realize that what we have now is only a functional prototype. There are a ton of ways of making it bigger, better and more useful. It would be an honour to watch Controllo as a product that every person with motor impairment could make use of.

# Acknowledgements:

We would like to express our gratitude and appreciation to all those faculty members who gave us the possibility to complete the first and second phase of our Final year Project with great zeal and helped us in following a direction on a specific and accurate idea. They also helped us in improving our mistakes and guided us throughout.

A special thanks to our supervisor, Dr. Muhammad Sarim, whose help, stimulating suggestions and encouragement, helped us in making timely and efficient submissions and enhance our understanding about the correct implementation of the project.

We would also like to thank all our friends and people in iba who cooperated with us and allowed us to take their pictures for app calibration purpose.

Sincerely,

Abdul Ahad Mushir

Sandhya Gopchandani

Rashida Moiz

Mahwash Merchant

# References:

<http://www.neuromodulation.com/motor-impairment>

<http://www.disabled-world.com/disability/types/mobility/>

<http://webaim.org/articles/motor/motordisabilities>

<https://en.wikipedia.org/wiki/Physical_disability>

<https://www.lds.org/topics/disability/list/physical-disability?lang=eng>

<https://www.dosomething.org/facts/11-facts-about-physical-disability>

<http://www.disabilitycompendium.org/docs/default-source/2014-compendium/annual-report.pdf>

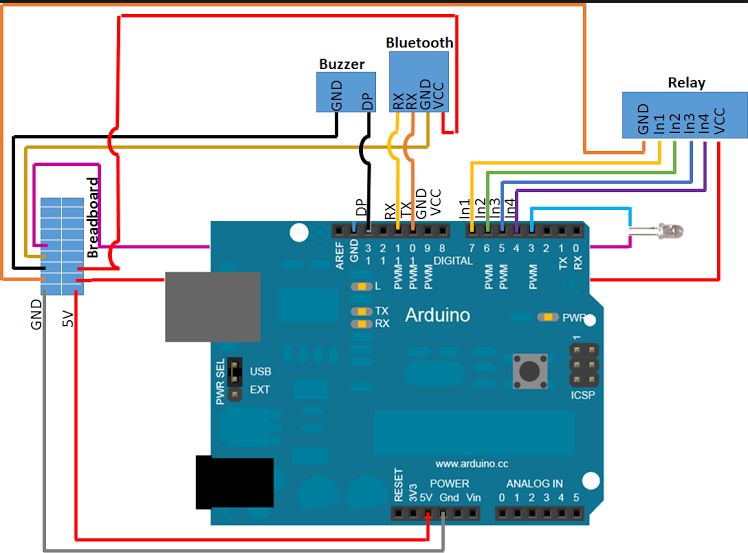
# Appendix 1: Code:

package com.google.android.gms.samples.vision.face.facetracker;  
  
import android.Manifest;  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.app.Dialog;  
import android.bluetooth.BluetoothAdapter;  
import android.bluetooth.BluetoothDevice;  
import android.bluetooth.BluetoothSocket;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.database.sqlite.SQLiteDatabase;  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.media.AudioManager;  
import android.net.Uri;  
import android.os.Bundle;  
import android.os.Environment;  
import android.support.design.widget.Snackbar;  
import android.support.v4.app.ActivityCompat;  
import android.support.v7.app.AppCompatActivity;  
import android.telephony.SmsManager;  
import android.util.Log;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.LinearLayout;  
import android.widget.Toast;  
  
import com.google.android.gms.common.ConnectionResult;  
import com.google.android.gms.common.GoogleApiAvailability;  
import com.google.android.gms.vision.CameraSource;  
import com.google.android.gms.vision.MultiProcessor;  
import com.google.android.gms.vision.Tracker;  
import com.google.android.gms.vision.face.Face;  
import com.google.android.gms.vision.face.FaceDetector;  
import com.google.android.gms.samples.vision.face.facetracker.ui.camera.CameraSourcePreview;  
importcom.google.android.gms.samples.vision.face.facetracker.ui.camera.GraphicOverlay;  
  
importjava.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Set;  
import java.util.Timer;  
import java.util.UUID;  
  
  
*/\*\*  
 \* Activity for FaceTracking and Facial landmarks detections. This app detects faces with the rear or front facing camera, and  
 \* uses recognised facial gestures to send a BT signal to arduino.  
 \*  
 \*  
 \*/*public final class FaceTrackerActivityextends AppCompatActivity {  
private static final String *TAG* = "FaceTracker";  
*/\*\*  
 \*The following instances are used to initialise the camera preview.  
 \*  
 \*/*private CameraSourcemCameraSource= null;  
 private CameraSourcePreviewmPreview;  
 private GraphicOverlaymGraphicOverlay;  
 private static final int*RC\_HANDLE\_GMS* = 9001;  
// permission request codes need to be < 256  
private static final int*RC\_HANDLE\_CAMERA\_PERM* = 2;  
  
*/\*\*  
 \*The following instances are used for the core part of our application  
 \* algorithm for obtaining a gesture and sending a BT signal  
 \*/*private booleanmegaCheck= false;  
 private booleanmegaCheck1 = false;  
 private booleanactivate = true;  
 private booleanfirstFrame= true;  
 private intfaceCounter= 0;  
 public SQLiteDatabasemydatabase;  
  
*/\*\*  
 \*Bluetooth referencing variables  
 \*/*ArrayAdapter<BluetoothDevice>pairedDeviceAdapter;  
BluetoothAdapterbluetoothAdapter;  
ArrayList<BluetoothDevice>pairedDeviceArrayList;  
 private UUID myUUID;  
 private final String UUID\_STRING\_WELL\_KNOWN\_SPP =  
"00001101-0000-1000-8000-00805F9B34FB";  
ThreadConnectBTdevicemyThreadConnectBTdevice;  
ThreadConnectedmyThreadConnected;  
 private static final int*REQUEST\_ENABLE\_BT* = 1;  
  
*/\*\*  
 \* Initializes the UI and initiates the creation of a face detector.  
 \*/*@Override  
public void onCreate(Bundle icicle) {  
super.onCreate(icicle);  
setContentView(R.layout.*main*);  
  
mPreview= (CameraSourcePreview) findViewById(R.id.*preview*);  
mGraphicOverlay= (GraphicOverlay) findViewById(R.id.*faceOverlay*);  
  
  
  
*/\*\*  
 \* Required for BT intent  
 \*/*if (!getPackageManager().hasSystemFeature(PackageManager.*FEATURE\_BLUETOOTH*)){  
Toast.*makeText*(this,  
"FEATURE\_BLUETOOTH NOT support",  
Toast.*LENGTH\_LONG*).show();  
finish();  
 return;  
}  
  
myUUID= UUID.*fromString*(UUID\_STRING\_WELL\_KNOWN\_SPP);  
bluetoothAdapter= BluetoothAdapter.*getDefaultAdapter*();  
 if (bluetoothAdapter== null) {  
Toast.*makeText*(this,  
"Bluetooth is not supported on this hardware platform",  
Toast.*LENGTH\_LONG*).show();  
finish();  
 return;  
}  
  
*/\*\*  
 \* Check for the camera permissions, before accessing the camera. If the  
 \* permissions are not granted yet, request permission.  
 \*/*intrc = ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*);  
 if (rc == PackageManager.*PERMISSION\_GRANTED*) {  
createCameraSource();  
} else {  
requestCameraPermission();  
}  
  
  
// setup(); //required when the app re-opens after resume!  
 //issue on my qmobile.  
  
}  
  
public void gotoBack(View view) {  
// onBackPressed();  
if (DbAct.*storedGestures*== 8) {  
DbAct.*isCalibrated*= true; //as on faceGraphic it will cause logical error!  
}  
else {  
DbAct.*isCalibrated*= false;  
}  
 Intent i = new Intent(getApplicationContext(),DbAct.class);  
startActivity(i);  
setContentView(R.layout.*activity\_db*);  
}  
  
  
// public Bitmap screenShot(View view) {  
// Bitmap bitmap = Bitmap.createBitmap(view.getWidth(),  
// view.getHeight(), Bitmap.Config.ARGB\_8888);  
// Canvas canvas = new Canvas(bitmap);  
// view.draw(canvas);  
// return bitmap;  
// }  
  
public void screenSh(View view) {  
 Bitmap bitmap = takeScreenshot();  
saveBitmap(bitmap);  
}  
  
// private void takeScreenshot() {  
// Date now = new Date();  
// android.text.format.DateFormat.format("yyyy-MM-dd\_hh:mm:ss", now);  
//  
// try {  
// // image naming and path to include sd card appending name you choose for file  
// String mPath = Environment.getExternalStorageDirectory().toString() + "/" + now + ".jpg";  
//  
// // create bitmap screen capture  
// View v1 = getWindow().getDecorView().getRootView();  
// v1.setDrawingCacheEnabled(true);  
// Bitmap bitmap = Bitmap.createBitmap(v1.getDrawingCache());  
// v1.setDrawingCacheEnabled(false);  
//  
// File imageFile = new File(mPath);  
//  
// FileOutputStreamoutputStream = new FileOutputStream(imageFile);  
// int quality = 100;  
// bitmap.compress(Bitmap.CompressFormat.JPEG, quality, outputStream);  
// outputStream.flush();  
// outputStream.close();  
//  
// openScreenshot(imageFile);  
// } catch (Throwable e) {  
// // Several error may come out with file handling or OOM  
// e.printStackTrace();  
// }  
// }  
//  
// private void openScreenshot(File imageFile) {  
// Intent intent = new Intent();  
// intent.setAction(Intent.ACTION\_VIEW);  
// Uri uri = Uri.fromFile(imageFile);  
// intent.setDataAndType(uri, "image/\*");  
// startActivity(intent);  
// }  
  
// Bitmap bitmap;  
// View v1 = findViewById(R.id.topLayout);// get ur root view id  
// v1.setDrawingCacheEnabled(true);  
// bitmap = Bitmap.createBitmap(v1.getDrawingCache());  
// v1.setDrawingCacheEnabled(false);  
//  
public Bitmap takeScreenshot() {  
 View rootView = findViewById(R.id.*topLayout*).getRootView(); //findViewById(android.R.id.content).getRootView();  
rootView.setDrawingCacheEnabled(true);  
 return rootView.getDrawingCache();  
}  
  
public void saveBitmap(Bitmap bitmap) {  
 Date now = new Date();  
android.text.format.DateFormat.*format*("yyyy-MM-dd\_hh:mm:ss", now);  
File imagePath = new File(Environment.*getExternalStorageDirectory*() + "/" +now+ ".png");  
FileOutputStreamfos;  
 try {  
fos = new FileOutputStream(imagePath);  
bitmap.compress(Bitmap.CompressFormat.*PNG*, 100, fos);  
fos.flush();  
fos.close();  
} catch (FileNotFoundException e) {  
Log.*e*("GREC", e.getMessage(), e);  
} catch (IOException e) {  
Log.*e*("GREC", e.getMessage(), e);  
}  
 }  
  
@Override  
protected void onStart() {  
super.onStart();  
  
//Turn ON BlueTooth if it is OFF  
if (!bluetoothAdapter.isEnabled()) {  
 Intent enableIntent = new Intent(BluetoothAdapter.*ACTION\_REQUEST\_ENABLE*);  
startActivityForResult(enableIntent, *REQUEST\_ENABLE\_BT*);  
  
  
Log.*w*("for loop", "If not Enabled");  
}  
 }  
  
@Override  
public void onBackPressed() {  
 Intent i = new Intent(getApplicationContext(),DbAct.class);  
startActivity(i);  
setContentView(R.layout.*activity\_db*);  
}  
  
  
*/\*\*  
 \* Sms component module for our app  
 \*/*private void SendSms() {  
 String phoneNo = "03311331952";  
String sms = "Hello, I need assistance!";  
  
 try {  
SmsManagersmsManager = SmsManager.*getDefault*();  
smsManager.sendTextMessage(phoneNo, null, sms, null, null);  
// Toast.makeText(getApplicationContext(), "SMS Sent!",  
// Toast.LENGTH\_LONG).show();  
inta = Tweak.*ivar1*;  
a = 0;  
} catch (Exception e) {  
// Toast.makeText(getApplicationContext(),  
// "SMS failed, please try again later!",  
// Toast.LENGTH\_LONG).show();  
e.printStackTrace();  
}  
 }  
  
  
private void test() {  
Log.*d*("test", "test: first component ");  
megaCheck= true;  
algoTime();  
}  
  
  
private void test2() {  
Log.*d*("test", "second component");  
megaCheck= true;  
algoTime();  
}  
  
private void algoTime() {  
new java.util.Timer().schedule(  
new java.util.TimerTask() {  
@Override  
public void run() {  
// your code here  
megaCheck= false;  
Log.*d*("test", "algoTime");  
}  
 },  
10 \* 1000  
);  
  
}  
  
private void facee() {  
AlertDialog.Builder alertDialog2 = new AlertDialog.Builder(  
FaceTrackerActivity.this);  
alertDialog2.setTitle("Error!");  
alertDialog2.setMessage("More than one face detected");  
  
alertDialog2.setPositiveButton("YES",  
 new DialogInterface.OnClickListener() {  
public void onClick(DialogInterface dialog, intwhich) {  
// Write your code here to execute after dialog  
  
}  
 });  
  
alertDialog2.setNegativeButton("NO",  
 new DialogInterface.OnClickListener() {  
public void onClick(DialogInterface dialog, intwhich) {  
// Write your code here to execute after dialog  
  
dialog.cancel();  
}  
 });  
  
  
alertDialog2.show();  
  
}  
  
  
*/\*\*  
 \* Calling component of our app  
 \*/*private void call() {  
 Intent in=new Intent(Intent.*ACTION\_CALL*, Uri.*parse*("tel:03311331952"));  
 try{  
startActivity(in);  
AudioManageraudioManager = (AudioManager)getSystemService(Context.*AUDIO\_SERVICE*);  
// audioManager.setMode(AudioManager.MODE\_IN\_CALL)  
if (!audioManager.isSpeakerphoneOn()){  
audioManager.setSpeakerphoneOn(true);  
}  
  
intq = Tweak.*ivar1*;  
q = 0;  
  
}  
  
catch (android.content.ActivityNotFoundException ex){  
Toast.*makeText*(getApplicationContext(), "yourActivity is not founded", Toast.*LENGTH\_SHORT*).show();  
}  
 }  
  
  
*/\*\*  
 \*SendToBluetooth function takes an Integer provided by a gesture  
 \* and sends the desired signal to arduino.  
 \*/*private void SendToBluetooth(intgest)  
 {  
 String msg1 = null;  
 if (gest == 1) {  
 msg1 = "a";  
}  
else if (gest == 2) {  
 msg1 = "1";  
}  
else if (gest == 3) {  
 msg1 = "b";  
}  
else if (gest == 4) {  
 msg1 = "2";  
}  
else if (gest == 5) {  
 msg1 = "c";  
}  
else if (gest == 6) {  
 msg1 = "3";  
}  
else {  
 msg1 = "0";  
Log.*d*("inte", "for 7 and 8");  
}  
  
  
  
  
Log.*w*("for loop", "Replacement of SEND BUTTON");  
 if(myThreadConnected!=null) {  
Log.*w*("for loop", "Duplicate myThreadConnected!=null");  
//byte[] bytesToSend = inputField.getText().toString().getBytes();  
byte[] bytesToSend = msg1.getBytes();  
myThreadConnected.write(bytesToSend);  
}  
  
// activate = false;  
  
// megaCheck = true;  
megaCheck1 = false;  
// algoTime();  
  
}  
  
*/\*\*  
 \* Setup function for BT connectivity  
 \*/*private void setup() {  
 Set<BluetoothDevice>pairedDevices = bluetoothAdapter.getBondedDevices();  
 if (pairedDevices.size() >0) {  
pairedDeviceArrayList= new ArrayList<BluetoothDevice>();  
 for (BluetoothDevice device : pairedDevices)  
 {  
pairedDeviceArrayList.add(device);  
}  
for(inti = 0; i<=pairedDeviceArrayList.size()-1; i++) {  
BluetoothDevice Device = pairedDeviceArrayList.get(i);  
 if (Device.getName().contentEquals("HC-05")) {  
myThreadConnectBTdevice= new ThreadConnectBTdevice(pairedDeviceArrayList.get(i));  
myThreadConnectBTdevice.start();  
}  
 }  
 }  
  
 }  
  
  
*/\*\*  
 \* For cancelling the BT connections  
 \*/*public void cancel() {  
  
Toast.*makeText*(getApplicationContext(),  
"close bluetoothSocket",  
Toast.*LENGTH\_LONG*).show();  
  
 try {  
myThreadConnectBTdevice.bluetoothSocket.close();  
} catch (IOException e) {  
// *TODO Auto-generated catch block*e.printStackTrace();  
}  
  
 }  
  
*/\*\*  
 \* BT code  
 \*/*@Override  
protected void onActivityResult(intrequestCode, intresultCode, Intent data) {  
if(requestCode==*REQUEST\_ENABLE\_BT*){  
if(resultCode == Activity.*RESULT\_OK*){  
 setup();  
}else{  
Toast.*makeText*(this,  
"BlueTooth NOT enabled",  
Toast.*LENGTH\_SHORT*).show();  
finish();  
}  
 }  
 }  
  
private void startThreadConnected(BluetoothSocket socket){  
myThreadConnected= new ThreadConnected(socket);  
Log.*w*("for loop", "new ThreadConnected(socket)");  
myThreadConnected.start();  
Log.*w*("for loop", "myThreadConnected.start()");  
}  
  
*/\*\*  
 \* BT thread connectivity class  
 \*/*private class ThreadConnectBTdeviceextends Thread {  
  
private BluetoothSocketbluetoothSocket= null;  
 private final BluetoothDevicebluetoothDevice;  
  
  
 private ThreadConnectBTdevice(BluetoothDevice device) {  
bluetoothDevice= device;  
  
 try {  
bluetoothSocket= device.createRfcommSocketToServiceRecord(myUUID);  
//textStatus.setText("bluetoothSocket: \n" + bluetoothSocket);  
} catch (IOException e) {  
// *TODO Auto-generated catch block*e.printStackTrace();  
}  
 }  
  
@Override  
public void run() {  
  
booleansuccess = false;  
 try {  
bluetoothSocket.connect();  
success = true;  
} catch (IOException e) {  
e.printStackTrace();  
  
 try {  
bluetoothSocket.close();  
} catch (IOException e1) {  
// *TODO Auto-generated catch block*e1.printStackTrace();  
}  
 }  
  
if(success){  
//connect successful  
startThreadConnected(bluetoothSocket);  
// SendToBluetooth();  
}else{  
//fail  
}  
 }  
  
  
  
}  
  
private class ThreadConnectedextends Thread {  
private final BluetoothSocketconnectedBluetoothSocket;  
 private final OutputStreamconnectedOutputStream;  
  
 public ThreadConnected(BluetoothSocket socket) {  
connectedBluetoothSocket= socket;  
OutputStream out = null;  
 try {  
 out = socket.getOutputStream();  
} catch (IOException e) {  
// *TODO Auto-generated catch block*e.printStackTrace();  
}  
connectedOutputStream= out;  
}  
  
public void write(byte[] buffer) {  
try {  
connectedOutputStream.write(buffer);  
} catch (IOException e) {  
// *TODO Auto-generated catch block*e.printStackTrace();  
}  
 }  
  
 }  
  
*/\*\*  
 \* Handles the requesting of the camera permission. This includes  
 \* showing a "Snackbar" message of why the permission is needed then  
 \* sending the request.  
 \*/*private void requestCameraPermission() {  
Log.*w*(*TAG*, "Camera permission is not granted. Requesting permission");  
  
 final String[] permissions = new String[]{Manifest.permission.*CAMERA*};  
  
 if (!ActivityCompat.*shouldShowRequestPermissionRationale*(this,  
Manifest.permission.*CAMERA*)) {  
ActivityCompat.*requestPermissions*(this, permissions, *RC\_HANDLE\_CAMERA\_PERM*);  
 return;  
}  
  
final Activity thisActivity = this;  
  
View.OnClickListener listener = new View.OnClickListener() {  
@Override  
public void onClick(View view) {  
ActivityCompat.*requestPermissions*(thisActivity, permissions,  
*RC\_HANDLE\_CAMERA\_PERM*);  
}  
 };  
  
Snackbar.*make*(mGraphicOverlay, R.string.*permission\_camera\_rationale*,  
Snackbar.*LENGTH\_INDEFINITE*)  
 .setAction(R.string.*ok*, listener)  
 .show();  
}  
  
*/\*\*  
 \* Creates and starts the camera.  
 \*/*private void createCameraSource() {  
  
 Context context = getApplicationContext();  
FaceDetector detector = new FaceDetector.Builder(context)  
 .setClassificationType(FaceDetector.*ACCURATE\_MODE*) //sandhyaIsGenius  
.build();  
  
  
detector.setProcessor(  
new MultiProcessor.Builder<>(new GraphicFaceTrackerFactory())  
 .build());  
  
 if (!detector.isOperational()) {  
Log.*w*(*TAG*, "Face detector dependencies are not yet available.");  
}  
if (!DbAct.*camm*) {  
mCameraSource= new CameraSource.Builder(context, detector)  
 .setRequestedPreviewSize(640, 480)  
 .setFacing(CameraSource.*CAMERA\_FACING\_FRONT*) //front and back cam view  
.setRequestedFps(30.0f)  
 .build();  
}  
else {  
mCameraSource= new CameraSource.Builder(context, detector)  
 .setRequestedPreviewSize(640, 480)  
 .setFacing(CameraSource.*CAMERA\_FACING\_BACK*) //front and back cam view  
.setRequestedFps(30.0f)  
 .build();  
}  
 }  
  
*/\*\*  
 \* Restarts the camera.  
 \*/*@Override  
protected void onResume() {  
super.onResume();  
  
startCameraSource();  
}  
  
*/\*\*  
 \* Stops the camera.  
 \*/*@Override  
protected void onPause() {  
super.onPause();  
mPreview.stop();  
}  
  
*/\*\*  
 \* Releases the resources associated with the camera source, the associated detector, and the  
 \* rest of the processing pipeline.  
 \*/*@Override  
protected void onDestroy() {  
super.onDestroy();  
 if (mCameraSource!= null) {  
mCameraSource.release();  
}  
  
if(myThreadConnectBTdevice!=null){  
 cancel();  
}  
 }  
  
*/\*\*  
 \* Callback for the result from requesting permissions. This method  
 \* is invoked for every call on {****@link*** *#requestPermissions(String[], int)}.  
 \* <p>  
\* <strong>Note:</strong>It is possible that the permissions request interaction  
 \* with the user is interrupted. In this case you will receive empty permissions  
 \* and results arrays which should be treated as a cancellation.  
 \* </p>  
\*  
 \** ***@param****requestCodeThe request code passed in {****@link*** *#requestPermissions(String[], int)}.  
 \** ***@param****permissions The requested permissions. Never null.  
 \** ***@param****grantResultsThe grant results for the corresponding permissions  
 \* which is either {****@link*** *PackageManager#PERMISSION\_GRANTED}  
 \* or {****@link*** *PackageManager#PERMISSION\_DENIED}. Never null.  
 \** ***@see*** *#requestPermissions(String[], int)  
 \*/*@Override  
public void onRequestPermissionsResult(intrequestCode, String[] permissions, int[] grantResults) {  
if (requestCode != *RC\_HANDLE\_CAMERA\_PERM*) {  
Log.*d*(*TAG*, "Got unexpected permission result: " + requestCode);  
super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 return;  
}  
  
if (grantResults.length!= 0 &&grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
Log.*d*(*TAG*, "Camera permission granted - initialize the camera source");  
// we have permission, so create the camerasource  
createCameraSource();  
 return;  
}  
  
Log.*e*(*TAG*, "Permission not granted: results len = " + grantResults.length+  
" Result code = " + (grantResults.length>0 ? grantResults[0] : "(empty)"));  
  
DialogInterface.OnClickListener listener = new DialogInterface.OnClickListener() {  
public void onClick(DialogInterface dialog, intid) {  
 finish();  
}  
 };  
  
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle("Face Tracker sample")  
 .setMessage(R.string.*no\_camera\_permission*)  
 .setPositiveButton(R.string.*ok*, listener)  
 .show();  
}  
  
*/\*\*  
 \* Starts or restarts the camera source, if it exists. If the camera source doesn't exist yet  
 \* (e.g., because onResume was called before the camera source was created), this will be called  
 \* again when the camera source is created.  
 \*/*private void startCameraSource() {  
  
// check that the device has play services available.  
intcode = GoogleApiAvailability.*getInstance*().isGooglePlayServicesAvailable(  
getApplicationContext());  
 if (code != ConnectionResult.*SUCCESS*) {  
 Dialog dlg =  
GoogleApiAvailability.*getInstance*().getErrorDialog(this, code, *RC\_HANDLE\_GMS*);  
dlg.show();  
}  
  
if (mCameraSource!= null) {  
try {  
mPreview.start(mCameraSource, mGraphicOverlay);  
} catch (IOException e) {  
Log.*e*(*TAG*, "Unable to start camera source.", e);  
mCameraSource.release();  
mCameraSource= null;  
}  
 }  
 }  
  
*/\*\*  
 \* Factory for creating a face tracker to be associated with a new face. The multiprocessor  
 \* uses this factory to create face trackers as needed -- one for each individual.  
 \*/*private class GraphicFaceTrackerFactoryimplements MultiProcessor.Factory<Face> {  
@Override  
public Tracker<Face>create(Face face) {  
return new GraphicFaceTracker(mGraphicOverlay);  
}  
 }  
  
*/\*\*  
 \* Face tracker for each detected individual. This maintains a face graphic within the app's  
 \* associated face overlay.  
 \*/*private class GraphicFaceTrackerextends Tracker<Face> {  
private GraphicOverlaymOverlay;  
 private FaceGraphicmFaceGraphic;  
  
GraphicFaceTracker(GraphicOverlay overlay) {  
mOverlay= overlay;  
mFaceGraphic= new FaceGraphic(overlay);  
}  
  
*/\*\*  
 \* Start tracking the detected face instance within the face overlay.  
 \*/*@Override  
public void onNewItem(intfaceId, Face item) {  
mFaceGraphic.setId(faceId);  
faceCounter+= 1;  
 if (faceCounter>1) {  
//facee();  
}  
 String c = String.*format*("%d",faceCounter);  
Log.*d*("test", c);  
  
}  
  
*/\*\*  
 \* Update the position/characteristics of the face within the overlay.  
 \* MAIN ALGORITHM  
 \*/*@Override  
public void onUpdate(FaceDetector.Detections<Face>detectionResults, Face face) {  
mOverlay.add(mFaceGraphic);  
mFaceGraphic.updateFace(face); //face overlay  
if (!DbAct.*selfCalib*) { //checking if user has opted for self calibration or default  
  
if (firstFrame) { //if its a first activating gesture frame  
// mFaceGraphic.updateFace(face);  
new java.util.Timer().schedule(  
new java.util.TimerTask() {  
@Override  
public void run() {  
final intma = Tweak.*ivar1*; // obtaining the gesture value from FaceGraphic  
// mFaceGraphic.updateFace(face);  
if (ma == 4 && !megaCheck) { // if the activating gesture is received so TRUE  
activate = false;  
firstFrame= false;  
megaCheck= true;  
  
} else {  
  
 }  
 }  
 },  
3000  
);  
  
}  
if (!firstFrame) { //recieving the next frame soon after the activating gesture  
firstFrame= true;  
new java.util.Timer().schedule(  
new java.util.TimerTask() {  
@Override  
public void run() {  
final intma = Tweak.*ivar1*; // obtaining a new gesture value  
if (ma != 0 && !megaCheck1) {  
megaCheck1 = true;  
SendToBluetooth(ma); //sending the gesture to BT signal  
megaCheck= false;  
  
  
} else {  
  
 }  
 }  
 },  
5000  
);  
  
  
}  
  
 }//selfCalib IF  
else { //if user has opted for self calibrations  
 if (DbAct.*nowValRet*) { // checking if the gestures are present in the Database  
  
 if (firstFrame) { //if its a first activating gesture frame  
  
// mFaceGraphic.updateFace(face);  
new java.util.Timer().schedule(  
new java.util.TimerTask() {  
@Override  
public void run() {  
final intma = Tweak.*ivar2*; //obtaining the activating gesture value  
if (ma == 2 && !megaCheck) { //if activating gesture recieved so proceed  
activate = false;  
firstFrame= false;  
megaCheck= true;  
megaCheck1 = false;  
  
} else {  
  
 }  
 }  
 },  
3000  
);  
  
}  
if (!firstFrame) { //second frame recieved  
firstFrame= true;  
// mFaceGraphic.updateFace(face);  
// megaCheck = false;  
new java.util.Timer().schedule(  
new java.util.TimerTask() {  
@Override  
public void run() {  
final intma = Tweak.*ivar2*; // obtaining the gesture values from FaceGraphic  
if (ma != 0 && !megaCheck1) {  
  
  
megaCheck1 = true;  
  
SendToBluetooth(ma);//Send Signal to BT  
megaCheck= false;  
  
} else {  
  
 }  
 }  
 },  
5000  
);  
  
  
}  
 }  
  
  
 } //selfCalib ELSE  
}  
  
*/\*\*  
 \* Hide the graphic when the corresponding face was not detected. This can happen for  
 \* intermediate frames temporarily (e.g., if the face was momentarily blocked from  
 \* view).  
 \*/*@Override  
public void onMissing(FaceDetector.Detections<Face>detectionResults) {  
mOverlay.remove(mFaceGraphic);  
}  
  
*/\*\*  
 \* Called when the face is assumed to be gone for good. Remove the graphic annotation from  
 \* the overlay.  
 \*/*@Override  
public void onDone() {  
mOverlay.remove(mFaceGraphic);  
}  
 }  
}

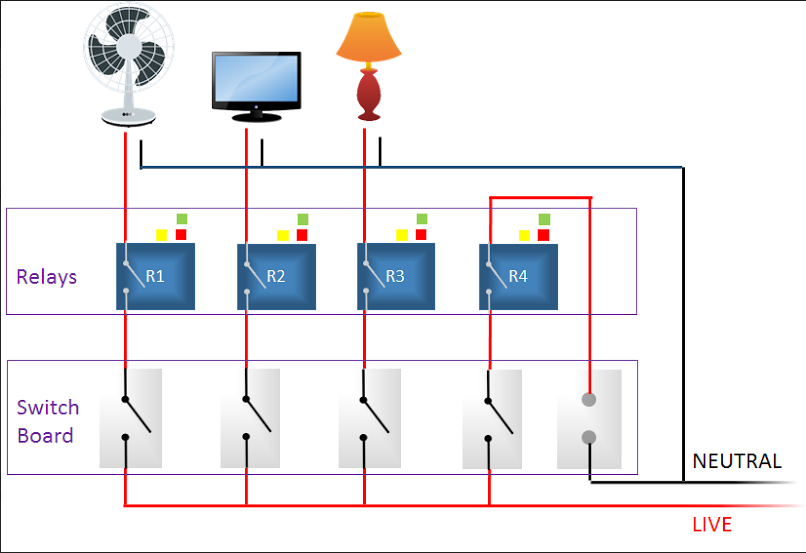
|  |
| --- |
|  |

# Appendix 2: Schematic of your hardware:

Arduino:



Switchboard:



# Appendix 3: Software/parts list:

## SOFTWARE:

Android studio

Arduino IDE

## HARDWARE:

Android smartphone

Arduino UNO

Relays (10A)

Arduino Bluetooth

IR LED

Bread board

Wires

Switch Board

Arduino Buzzer

Appliances (TV, Lights, Fan)

# Appendix 4: Work distribution:

The project’s work was divided into two major parts:

1. Software

* Application Front end (Sandhya & Ahad)
  + GUI – App user interface
  + GUI – App menu
  + GUI – App design
* Application Backend (Ahad & Mahwash)
  + Functionality of the App
  + Facial Gesture and algorithm optimizing
  + Microcontroller programming

1. Hardware

* Microcontroller wiring (Mahwash & Rashida)
* Switch board wiring (Sandhya & Rashida)

# Appendix 6: Project timeline:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Elapsed time since start of the project** | **Milestone** | **Deliverable** |
| **S.No** |  |  |  |
| 1 | 1 month (December) | Meet Neurologist, Meet Physiotherapist and Meet Patients, their attendants and family | Idea about the situation and needs of disabled patients. |
|
|
| 2 | 6 months (May) | Gesture recognition | Appropriate gestures with optimal threshold to facilitate a variety of facial configurations |
|
| 3 | 4 months (March) | Android application |          GUI |
|          Bluetooth configuration |
|          SMS Configuration |
|          Working application |
| 4 | 2 months (January) | Arduino connection | Wired Bluetooth component, Relay, Transistor, switchboard |
|
| 5 | 6 months (May) | Testing/debugging | Integrated, working application. |