



Title: Anti face touching bracelet and mobile application

Proposed by: Sami Ben Ali, ENIT's SC projects manager.

Team members : Sami Ben Ali , Achref Benmohamed

Problem and project's compliance with EFS mission:

Humans have a rather unusual habit we don't share with many other animals, and, unfortunately, it makes us particularly susceptible during certain disease outbreaks – we keep touching our faces..

Our species is one of the few in the animal kingdom known to touch their faces regularly, and we often do it without even noticing. Studies have shown that we are particularly prone to touching our chins and the areas around the mouth, nose and eyes. When it comes to a disease like the new coronavirus, Covid-19, that is a recipe for fast transmission , the average person touches their face 23 times an hour , thus rises a necessity to put an end to such behavior or try to limit it as much as possible .

Our solution:

Development of a Smart bracelet that detects hand movement using a built-in accelerometer and notifies the user when they are about to touch their face by emitting an audio signal , this bracelet could be also connected to a mobile application to Bothe calibrate the sensor and help monitoring their behavior using a dashboard , this project is mainly focused towards children to help parents monitor their behavior and keep them from repeating this dangerous habit (of face touching) whoever adults can use it as well , specially under the current circumstances .

Execution requirements:

1. Hardware requirements :

- ESP8266 ESP-12E NodeMCU
- Resistors : 3*10k , 1*2.2k,1*400k (Unit:OHM)
- 1*GY-521 Accelerometer
- 1* PIEZO BUZZER12
- 1* 2N3904S Transistor
- 1*HTT333 regulator
- 1*Push button
- 1*1N4007 Diode
- 1*1000μF capacitor
- 1*Lithium ion battey (400mah)
- 1* Uxcell a14052100ux1434 Lithium Battery Charging module

- Android studio
- Firebase real time database
- Arduino IDE

The diagram illustrates a temperature monitoring system using an ESP12F module. The system components and their connections are as follows:

- Power Supply:** A 3.3V regulator (Q1, 2N3904S-RTK/PS) is connected to the ESP12F module. A 1000µF capacitor (C1) is used for decoupling. The ESP12F module is powered by a 3.3V regulator (Q1) and a 1000µF capacitor (C1).
- Temperature Sensor:** An HT7333 (U2) is connected to the ESP12F module via I2C (SCL, SDA).
- Output Devices:** A buzzer (BUZZER1) and an LED (D1, 1N4007) are connected to the ESP12F module through GPIO pins. A push button (KEY1, 500k, Silicone silent key button) is connected to the ESP12F module.
- ESP12F Module:** The ESP12F (U1, ESP-12F(ESP8266MOD)) is the central microcontroller. It is connected to the temperature sensor (U2) via I2C (SCL, SDA). It controls the buzzer (BUZZER1) and the LED (D1) through GPIO pins. It is also connected to the GY-521 module (G1) via I2C (SCL, SDA).
- GY-521 Module:** The GY-521 (G1) is connected to the ESP12F module via I2C (SCL, SDA).

The diagram is labeled with component values and pin numbers. The ESP12F module is connected to the temperature sensor (U2) via I2C (SCL, SDA). The ESP12F module is also connected to the buzzer (BUZZER1) and the LED (D1) through GPIO pins. The ESP12F module is connected to the GY-521 module (G1) via I2C (SCL, SDA).

The user button and the transistor were added to the design in order to boot up the mcu and the accelerometer after they entered sleep mode when the accelerometer is not reading any changing data, this feature was added to limit power consumption.

Once the power button is pushed it will link the CH_PD pin to the output of the regulator therefore waking up the chip from sleep mode , the reset button and VCC are always set to high in order to boot the chip , in order to make sure that the chip stays power on even after we let go of the push button ,the GPIO0 will be set to HIGH once the chip has started functioning :

```
void setup() {  
    // put your setup code here, to run once:  
  
    pinMode(holdPin, OUTPUT);  
    digitalWrite(holdPin, HIGH);
```

And in order to limit power consumption the circuit will enter sleep mode by setting the same pin to LOW if it doesn't record any movement after approximately 5 minutes :

```
if ( xprev==x) { counter++; }  
  
if ( counter > 4000000000) // freq is 80Mhz so if counter exceeds approximetaly 5 minutes  
{  
    digitalWrite(holdPin, LOW);  
}
```

Important note :

The prototype code was tested on a different version of ESP8266 (ESP8266 12-E with a chip Silabs CP2102 for RS232 comunication on USB bus)

The reason for this is to facilitate debugging using the arduino IDE and also due to lack of componants, therefore the final pcb based on the previous schematic was not tested , however the prototype is based on the same chip and prouves to be fully functional .

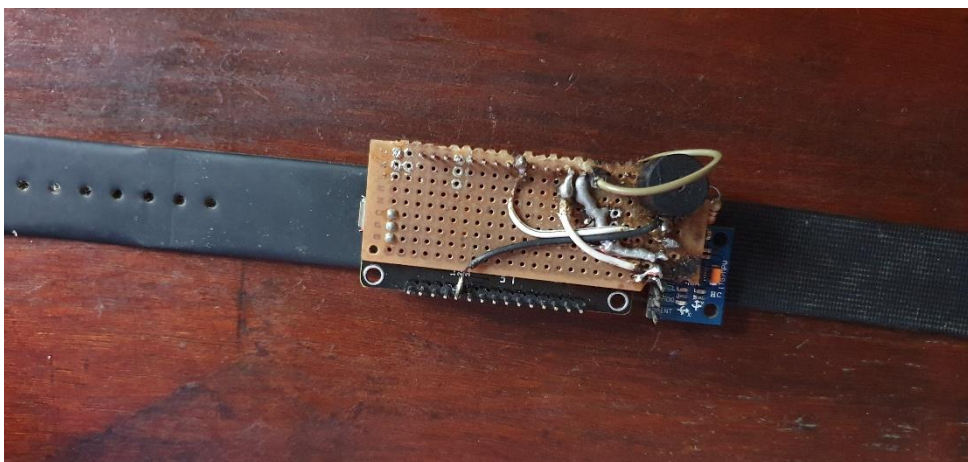
the only difference between the two is in the power option and the wake up and sleep feature , the prototype is powered using the usb connection , while the untested circuit is powered using an external regulator with an input from rechargeable lithium battery .

the lithium battery is rechargeable using the battery charging module .

This is the chip used in the prototype along with a picture of the prototype soldered on a test circuit :

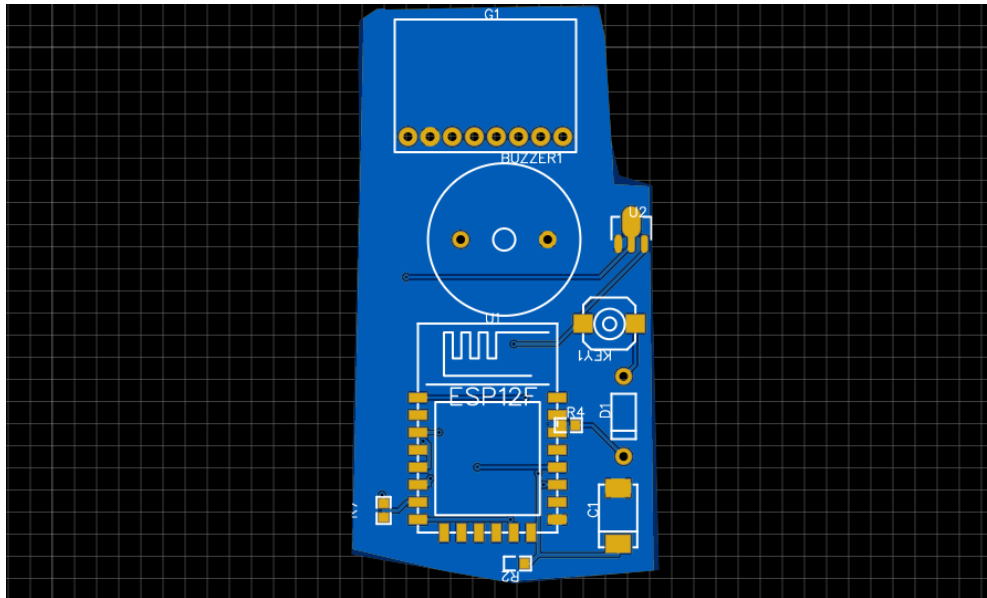


The final prototype (powered by the usb cable during the testing phase) :

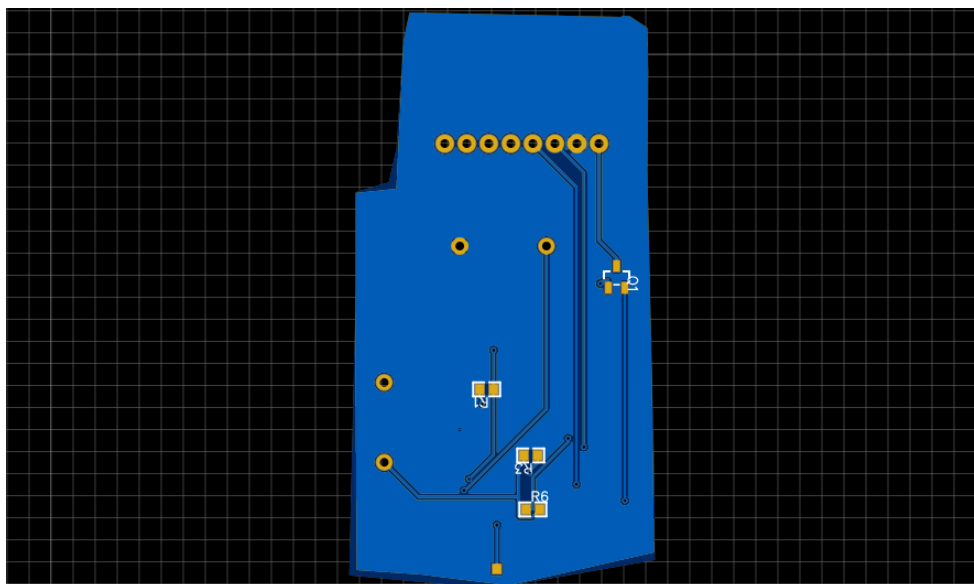


These are the untested PCB Top and Bottom views (designed using online easy eda tool) :

Top view :



Bottom view :



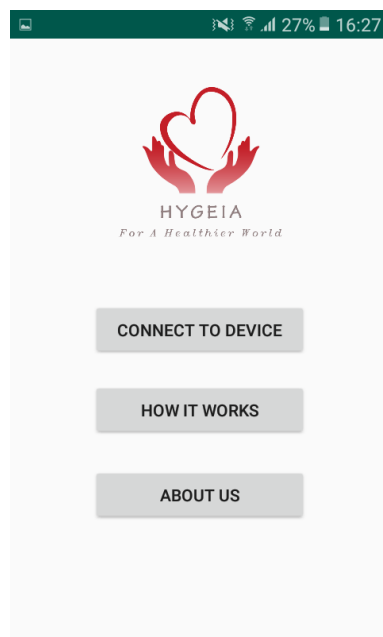
The android application and how it interacts with the mcu :

The device will be connected to an android app via google's firebase , the app has two main roles :

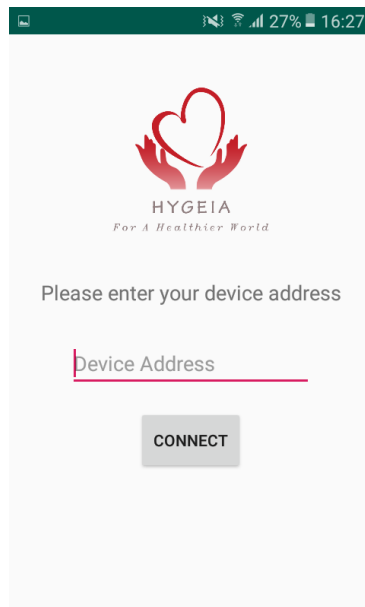
1. In order to use the bracelet it must be configured correctly to detect the following positions of the accelerometer: mouth position , nose position left eye and right eye positions .
2. Presents a dashboard to enable real time monitoring of the user's behaviour.

Connection and configuration:

The user must first establish a connection to the firebase and therefore to the device by pushing the “connect to device button” (the device also needs to be connected to internet in order to be configured , please refer to the source code for further details) :

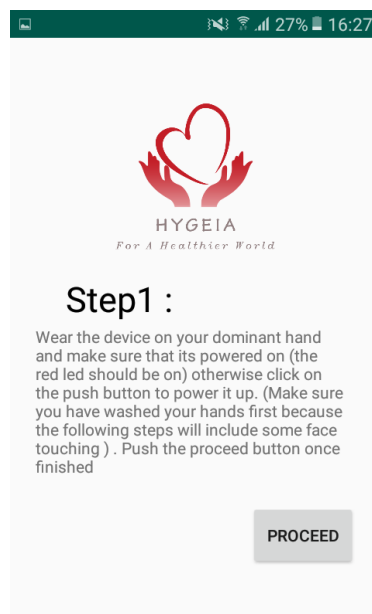


Then they must enter the device's address (at this point we didn't introduce an authentication feature to the project but all ameliorations are extremely welcome):



The screenshot shows the HYGEIA app interface. At the top is a green status bar with icons for signal, Wi-Fi, 27% battery, and the time 16:27. Below the status bar is the HYGEIA logo, which consists of two red hands holding a heart, with the text "HYGEIA" and "For A Healthier World" underneath. The main text on the screen says "Please enter your device address". Below this is a text input field with the placeholder "Device Address". At the bottom is a grey button labeled "CONNECT".

As soon as they press the connect button, they will be presented a series of configuration steps in order to configure the device, each step is responsible of updating certain reference values in the database (for the eye, mouth, nose positions), the first step is as follows :



The screenshot shows the HYGEIA app interface for Step 1. At the top is a green status bar with icons for signal, Wi-Fi, 27% battery, and the time 16:27. Below the status bar is the HYGEIA logo, which consists of two red hands holding a heart, with the text "HYGEIA" and "For A Healthier World" underneath. The main text on the screen says "Step1 :". Below this is a paragraph of text: "Wear the device on your dominant hand and make sure that its powered on (the red led should be on) otherwise click on the push button to power it up. (Make sure you have washed your hands first because the following steps will include some face touching) . Push the proceed button once finished". At the bottom is a grey button labeled "PROCEED".

During this step the app will connect to the firebase and set the “config” value to 0 (the config variable is used to indicate that the user will be resetting the reference values and the mcu will execute the code responsible of updating those values):

```
public class Configuration extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_configuration);
        FirebaseDatabase database = FirebaseDatabase.getInstance();

        DatabaseReference myRef = database.getReference("config");
        myRef.setValue(0);

    }
```

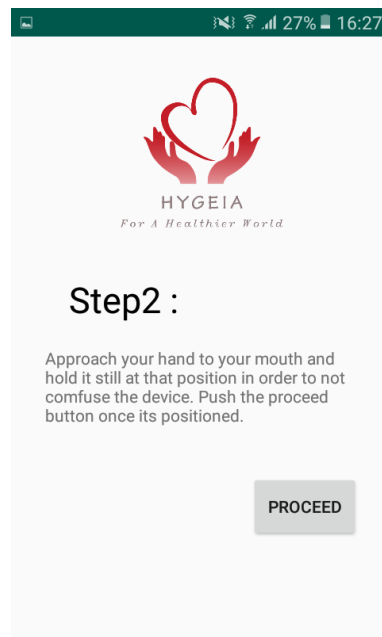
As soon as config changes to 0 the mcu will execute the code responsible of updating the reference values , each 3 axes variables of the reference variables are labeled by a “flag” .

```
else {
    //config=0 so we upload the values

    while(!Firebase.get(firebaseData, "/flag"))
    {
        //Failed?, get the error reason from firebaseData

        Serial.print("Error getting config value \n, ");
        Serial.println(firebaseData.errorReason());
        delay(200);
    }
```


The second step :



During this step the app will connect to the firebase and set the “flag” value to 1 (the flag variable is used to indicate that the user has started step 1 of the actual configuration to change the reference values of mouth in the database)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_config_step2);
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("flag");
    myRef.setValue(1);
}
```

Once that flag value changes to 1 the mcu will execute the code responsible on updating the mouth's reference values in the firebase (mouthx,mouthy,mouthz)

```
if(flag==1){

while(!(Firebase.setInt(firebaseData, "/mouthx", x)))
{

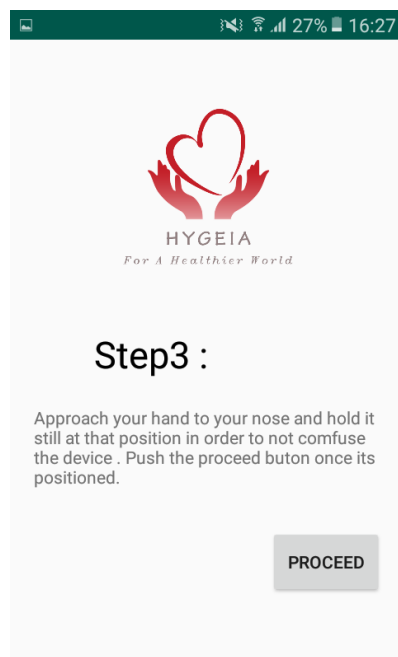
    //Failed?, get the error reason from firebaseData

    Serial.print("Error in setInt mouthx, ");
    Serial.println(firebaseData.errorReason());
    delay(200);

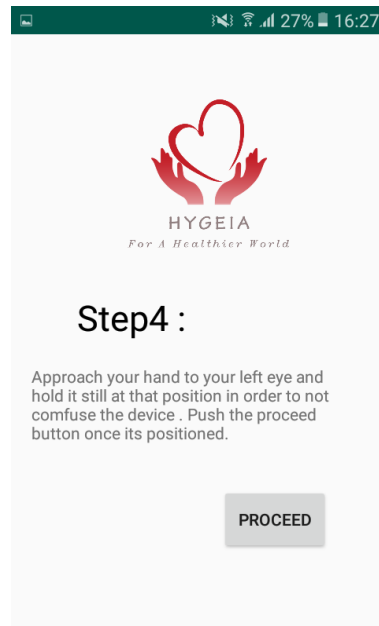
}

    //Success
    Serial.println("Set int data success mouthx");
```

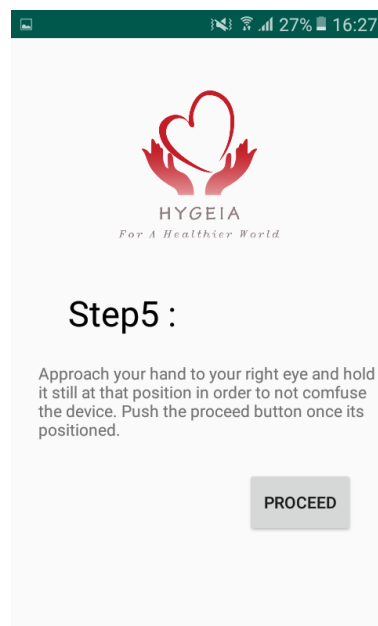
The remaining steps will do the exact same thing but with a different value to the flag variable (for example during step 3 it well be set to 2 and the mcu will execute the code that uploads the new refrence values for the nose position : noseX,noseY,noseZ)



The fourth step :



The fifth and final step :



Once the configuration steps are complete the flag variable will be set to 5 and the config variable to 1 ,

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_config_step5);
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("flag");
    myRef.setValue(4);

}
public void goToMenu(View v){
    Intent menu = new Intent(this,Menu.class);
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef1 = database.getReference("config");
    myRef1.setValue(1);
    DatabaseReference myRef = database.getReference("flag");
    myRef.setValue(5);
    startActivity(menu);
}
```

Once config is set to 1 the mcu will start comparing the new reference values located on the firebase with the data given by the accelerometer and will start the notification procedure if the values are equal :

```
else if( (((mouthx-10<x)&&(x<mouthx+10))&&((mouthz-10<z)&&(z<mouthz+10))&&((mouthy-10<y)&&(y<mouthy+10)))or

{

    digitalWrite(BUZZER, HIGH);
    delay(250);
    digitalWrite(BUZZER, LOW);
    delay(250);
    digitalWrite(BUZZER, HIGH);
```

The mcu will write to the notif variable in the firebase each time the bracelet detects that the user touched one of the referenced areas (previously configured by the app) , the variable will contain the exact time of the occurrence.

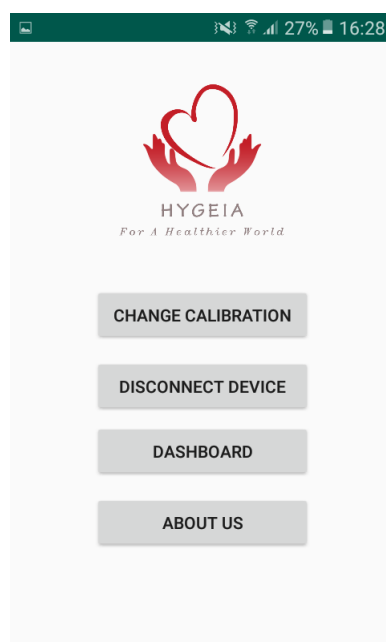
```
timeClient.begin();
delay(500);
Serial.println(timeClient.getFormattedTime());
Serial.print(daysOfTheWeek[timeClient.getDay()]);
while(!Firebase.setString( firebaseData,"/notif", daysOfTheWeek[timeClient.getDay()]+timeClient.getFormattedTime()))
{

    //Failed?, get the error reason from firebaseData

    Serial.println("Error in setInt time, ");
    Serial.println(firebaseData.errorReason());
    delay(200);

}
//Success
```

The user will also be able to access the dashboard and monitor the logged data , the app will fetch the notif information from the firebase and store it locally and display it on the dashboard , the user will also be able to change the calibrations by repeating those steps if they misconfigured the device .



Source code :

- For the mcu :

https://github.com/samiBENALI/ESP8266_HYGEAI_CODE

- For the app :

<https://github.com/AbM7797/Hygeai>