Project Report – Final

Codes for Arduino:

```
#include <SoftwareSerial.h>
long pmcf10=0;
long pmcf25=0;
long pmcf100=0;
long pmat10=0;
long pmat25=0;
long pmat100=0;
char buf[50];
int count = 0;
unsigned char c, last_c;

SoftwareSerial BT(2, 3); // 接收腳 RX, 傳送腳 TX
SoftwareSerial Blueteeth(10,9);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Blueteeth.begin(9600);
  BT.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:

  bool start = false;
  unsigned char high;
  while(BT.available())
  {
    //Serial.println("x");
    last_c = c;
    c = BT.read();

    if(last_c == 0x42 && c == 0x4d)
    {
      //Serial.print("start");
      count = 1;
    }

    if(count == 4 || count == 6 || count == 8 || count == 10 || count
== 12 || count == 14)
    {
      high = c;
    }
    else if(count==5)
    {
    }
    else if(count==7)
    {
      pmcf25 = 256*high + c;
      Serial.print("CF=1, PM2.5= : ");
      Serial.print(pmcf25);
      Serial.print(" ug/m3");
      Serial.print("\t\t");

      Blueteeth.println(pmcf25);

    }
    else if(count==9)
    {
    }
    else if(count==11)
    {
    }
    else if(count==13)
    {
      pmat25 = 256*high + c;
      Serial.print("atmosphere, PM2.5= : ");
      Serial.print(pmat25);
      Serial.println(" ug/m3");
    }
    else if(count==15)
    {
    }

    if (count < 100) count++;
    //Serial.println(c);
  }
}
```

Codes for android:

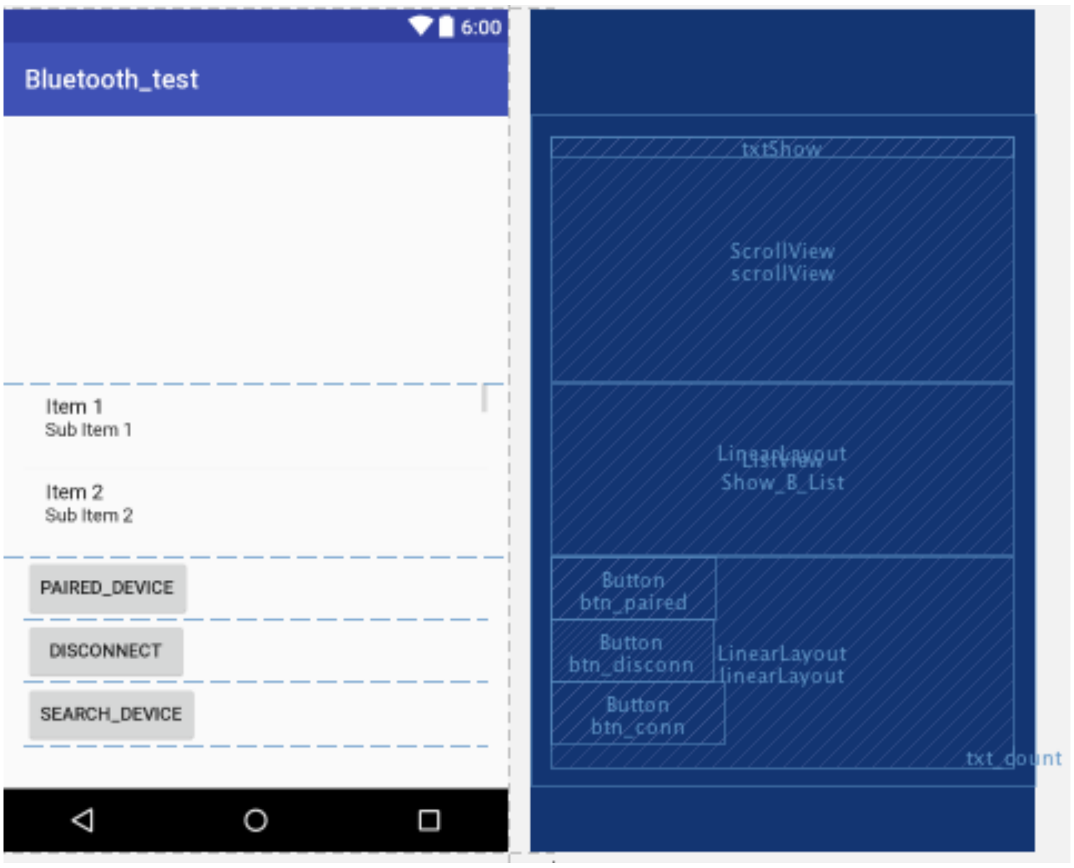
```
public class Trans_page extends AppCompatActivity {
    private Button button_paired;
    private Button button_disconnect;
    private Button button_find;
    private TextView show_data;
    private ListView event_listView;
    private TextView show_count;
    private BluetoothAdapter mBluetoothAdapter;
    private ArrayAdapter<String> deviceName;
    private ArrayAdapter<String> deviceID;
    private Set<BluetoothDevice> pairedDevices;
    private String choseID;
    private BluetoothDevice bleDevice;
    private BluetoothSocket bluesoccket;
    private InputStream mmInputStream;
    private OutputStream mmOuputStream;
    Thread workerThread;//宣告多執行續名為 wokerThread
    volatile boolean stopWorker; // 宣告布林值 stopWorker, 在主記憶體工作
(資料同步)
    private int readBufferPosition;
    private byte[] readBuffer;
    private String uid;
    private int count; //計算接收到幾筆資料了
    private LocationManager locMgr;
    private String bestProv;
    private static final int REQUEST_FINE_LOCATION_PERMISSION = 102;
// android 6.0 之後定位設備授權請求代碼
    private GoogleApiClient client;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_trans_page);
        getView();
        setListener();
        deviceName = new ArrayAdapter<String>(this,
android.R.layout.simple_expandable_list_item_1);
        deviceID = new ArrayAdapter<String>(this,
android.R.layout.simple_expandable_list_item_1);
        count = 0;
        locMgr = (LocationManager) getSystemService(LOCATION_SERVICE);
        Criteria criteria = new Criteria();
        bestProv = locMgr.getBestProvider(criteria, true);
        requestLocationPermission();
        // ATTENTION: This was auto-generated to implement the App
Indexing API.
        // See https://g.co/AppIndexing/AndroidStudio for more
information.
        client = new
GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
    }
    private void requestLocationPermission() {
        // 如果裝置版本是 6.0（包含）以上
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            // 取得授權狀態，參數是請求授權的名稱
            int hasPermission = checkSelfPermission(
                    Manifest.permission.ACCESS_FINE_LOCATION);

            // 如果未授權
```

```
          if (hasPermission != PackageManager.PERMISSION_GRANTED) {
              // 請求授權
              //     第一個參數是請求授權的名稱
              //     第二個參數是請求代碼
              requestPermissions(
                      new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                      REQUEST_FINE_LOCATION_PERMISSION);
              return;
          }
      }
  }
  private void getView() {
      button_paired = (Button) findViewById(R.id.btn_paired);
      button_disconnect = (Button) findViewById(R.id.btn_disconn);
      show_data = (TextView) findViewById(R.id.txtShow);
      event_listView = (ListView) findViewById(R.id.Show_B_List);
      button_find = (Button) findViewById(R.id.btn_conn);
      show_count = (TextView) findViewById(R.id.txt_count);
  }

  private void setListener() {
      button_paired.setOnClickListener(new Button.OnClickListener()
{
          @Override
          public void onClick(View v)
          {
                      findPBT();

          }
      });
      button_disconnect.setOnClickListener(new
Button.OnClickListener() {
          public void onClick(View v) {
              try {
                  closeBT();
              } catch (IOException e) {
                  e.printStackTrace();
              }
          }
      });
      button_find.setOnClickListener(new Button.OnClickListener() {
          @Override
          public void onClick(View v) {
              findPBT();
          }
      });
      event_listView.setAdapter(deviceName);
      event_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> partent, View view,
int position, long id) {
              choseID = deviceID.getItem(position);
              try {
                  openBT();
              } catch (IOException e) {
                  e.printStackTrace();
              }
              Toast.makeText(Trans_page.this, "選擇了:" + choseID,
Toast.LENGTH_SHORT).show();
              deviceName.clear();
          }
      });
  }
  private void findPBT() {
      mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();//儲存
已配對藍芽設備名稱
      if (mBluetoothAdapter != null) { // 取得是否有藍芽裝置
          show_data.setText("No bluetooth adapter available"); //若沒
```

```
有，則顯示找不到藍芽裝置
        }
        if (!mBluetoothAdapter.isEnabled()) { //檢查手機藍芽是否開啟，沒有
的話，則跳到 android 設定藍芽畫面
            Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, 1); //回傳找到幾個藍
芽周邊
        }
        pairedDevices = mBluetoothAdapter.getBondedDevices();
        if (pairedDevices.size() > 0) {
            for (BluetoothDevice device : pairedDevices) {
                String str = "已配對完成的裝置有 " + device.getName() + "
" + device.getAddress() + "\n";
                //String
                uid = device.getAddress();
                //show_data.setText(str);
                bleDevice = device;
                deviceName.add(str);//將以配對的裝置名稱儲存，並顯示於 LIST 清
單中
                deviceID.add(uid); //好像沒用到
            }
            event_listView.setAdapter(deviceName);
        }
    }
    private void openBT() throws IOException {
        UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB"); //藍芽模組 UUID 好像都是這樣
        if (bleDevice != null) {//DeviceID != null // 如果有找到設備
            bluesoccket =
bleDevice.createRfcommSocketToServiceRecord(uuid); //使用被選擇的設備
UUID 建立連線
            bluesoccket.connect();
            mmOuputStream = bluesoccket.getOutputStream();
            mmInputStream = bluesoccket.getInputStream();
            beginListenForData();

            show_data.setText("Bluetooth Opened: " +
bleDevice.getName() + "" + bleDevice.getAddress());
        }
    }
    private void beginListenForData() {
        final Handler handler = new Handler();
        final byte delimiter = 10; // 將十進制，轉換成 ASCII code LF(line
feed-換行)
        stopWorker = false; // 監控多執行續的運作(false 為開啟)
        readBufferPosition = 0; // readBuffer 陣列位置 ， 預設為 0
        readBuffer = new byte[1024]; // 宣告一個為 byte 資料型別的陣列
        workerThread = new Thread(new Runnable() { //建立 Thread 是否運
作，是的話進入迴圈，不是就跳出迴圈
            @Override
            public void run() { // Thread 物件會調用 Runnable 物件的 run()方
法，來控制。
                while (!Thread.currentThread().isInterrupted()
&& !stopWorker) // 監控 Thread 是否運作，是的話進入迴圈，不是就跳出迴圈
                {
                    try {
                        int bytesAvailable = mmInputStream.available();
// 宣告接收資料變數
                        if (bytesAvailable > 0) //如果有資料進來
                        {
                            byte[] packetBytes = new
byte[bytesAvailable]; // 宣告 byte 陣列，數量由 bytesAvailable 決定
                            mmInputStream.read(packetBytes);
                            for (int i = 0; i < bytesAvailable; i++) {
                                byte b = packetBytes[i];   // 將資料一個個從
packetBytes 取出直到 b 的變數
                                if (b == delimiter) // 如果 b 等於換行指令
                                {
```

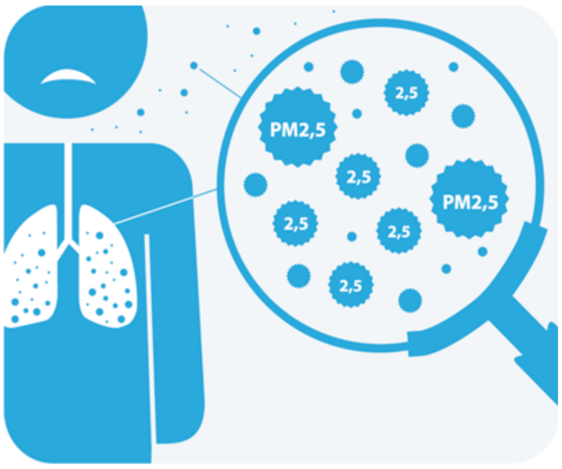
```
                              byte[] encodedBytes = new
byte[readBufferPosition];
                              System.arraycopy(readBuffer, 0,
encodedBytes, 0, encodedBytes.length); //把 readBuffer 陣列的值複製到
encodeBytes 陣列裡
                              final String data = new
String(encodedBytes, "US-ASCII"); // 轉成文字，無法被更改
                              readBufferPosition = 0; //readBuffer 陣
列位置，歸零。
                              count++;
                              handler.post(new Runnable() {
                                 @Override
                                 public void run() {
                                    String prevString =
show_data.getText().toString();
                                    String dataText =
String.format("%s\nCF=1, PM2.5=%sug/m3,收到了第%s 筆資料", prevString,
data,count);
                                    show_data.setText(dataText);

                                 }
                              });
                           } else // 若沒有換行，一直存進來
                           {
                              readBuffer[readBufferPosition++] = b;
//將接受到資料放入陣列裡面
                           }
                        }
                     }
                  } catch (IOException ex) {
                     stopWorker = true;
                  }
               }
            }
        });
        workerThread.start();
    }
    private void closeBT() throws IOException {
        stopWorker = true;
        mmOuputStream.close();
        mmInputStream.close();
        bluesoccket.close();
        deviceName.clear();
        show_count.setText("");
        show_data.setText("Bluetooth Closed");
    }
```

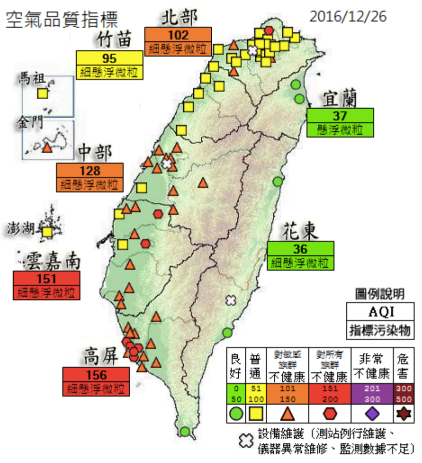
layout:



Marketing Statements

PM2.5 refers to particles with a diameter of 2.5 μm or less. The IARC and WHO have designated airborne particulates (including PM2.5) a Group 1 carcinogen(致癌物). These particles are very deadly due to their ability to penetrate into blood streams and lungs unfiltered, causing various kinds of disease.

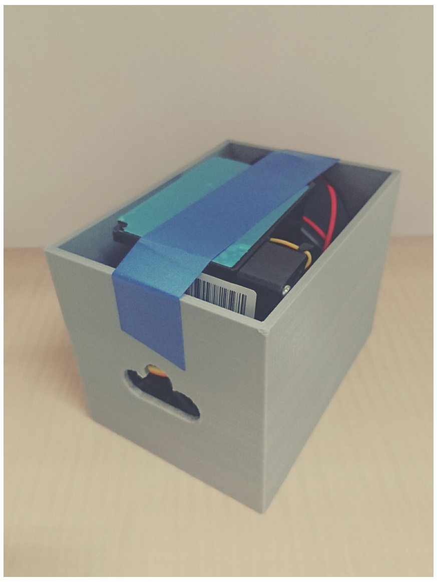




Searching for PM2.5 data to determine whether to wear a face mask or not has become a daily routine.

From now on, you don’t have to open your computer, open your web browser and check the PM2.5 readings. By the way, do you trust the readings from your government?

Now, you just have to tap on your cell phone app, put this delicate outside your window and then you can get the very accurate PM2.5 readings in just a blink of an eye.



Product Feature:

1. Very compact, you can take it anywhere anytime.
2. The cloud on one sides not only represents the function of this product but also represents “雲真美”.
3. You can easily modify this model cause it’s based on Arduino Development Kit.