Table of Contents

1  Introduction.............................................................................................................................................. 2
2  UG Wristband Hardware Components ................................................................................................. 2
3  Android Software ...................................................................................................................................... 3
   3.1  Pre-Requisites .................................................................................................................................... 3
   3.2  Steps to Develop an Android App ...................................................................................................... 4
   3.3  UG Android API ............................................................................................................................... 5
4  UG Wristband Firmware Update ............................................................................................................. 7
1 Introduction

Ultigesture UG wristband is a small sensor platform for wearable applications. It integrates a 3-axis accelerometer sensor, a 3-axis gyroscope sensor, and a 3-axis magnetometer sensor, which meets the needs of most wearable sensing study. A System on Chip is embedded in Ultigesture wristband as MCU, which includes a powerful Cortex-M4 processor and a 2.4GHz transceiver that supports BLE protocol. It provides strong computational capability and low energy consumption in BLE communication. The goal of Ultigesture wristband is to provide a compatible platform for wearable sensing study, which is with open API, comfortable to wear, and affordable in price.

2 UG Wristband Hardware Components

Ultigesture UG wristband integrates an nRF52832 System on Chip (SoC) from Nordic Semiconductor as MCU. The nRF52832 SoC incorporates a powerful Cortex-M4 processor with 512kB flash and 64kB RAM, and a 2.4GHz transceiver that supports BLE protocol. The Cortex-M4 processor provides strong computational capability. The BLE module enables UG wristband to run for a long period of time, and communicate with other devices that also support BLE. The hardware parameters for antenna are carefully tuned so that the communication range of BLE can reach as far as 40 meters.

In terms of motion sensing, a 9-axis motion sensor MPU-9250 is embedded in UG wristband. The MPU-9250 is a System in Package that combines two chips: the MPU-6500, which contains a 3-axis gyroscope, a 3-axis accelerometer; and the AK8963, a 3-axis digital compass sensor. The orientation of sensors is shown as below.

The capacity of the battery is a restriction for wearable devices. UG wristband is powered by a coin-size Li-Ion battery (3.7V, 75mAH). When turned on, UG wristband consumes 10.5mAH. When turned off, UG wristband only consumes 1.9uAH, which greatly prolongs the battery lifespan. The UG wristband can be charged through Micro-USB port. The charging time is around 1 hour.
The size of the PCB is very small with 26mm length and 25mm width. The size of the UG wristband shell is: 50mm length, 30mm width, and 11.7mm thickness, which is very easy to carry.

A UG wristband has 5 LEDs and 1 toggle button, which saves as user interfaces. A short button press (< 3 seconds) wakes up the system, while a long button press (> 3 seconds) puts the system into sleep.

<table>
<thead>
<tr>
<th>Features</th>
<th>Hardware Components</th>
<th>Capabilities</th>
</tr>
</thead>
</table>
| Processing | nRF52832 | 64MHz  
512kB flash  
64kB RAM |
| Communication | A 2.4GHz transceiver that supports BLE | Sensitivity: -96dB  
Range: 40 meters |
| Sensing | MPU9250 | The measure range:  
Accelerometer: ±4g  
Gyroscope: ±2000°/sec  
Magnetometer: ±4800μT |
| Charging | Micro-USB | Charge time: 1 hour |
| Power | Li-Ion battery | Capacity: 3.7V, 75mAH  
Work current: 10.5mAH  
Sleep current: 1.9μAH |
| UI | 1 button  
5 LEDs | Button is used to turn the system on or off  
5 LEDs are configurable by developer |

3 Android Software

This section introduces how to use UG Android API to build BLE connection with multiple UG wristbands and control them. The minimum supported Android version is Android 6.0 (API 23).

3.1 Pre-Requisites

The following are the pre-requisites needed to use the UG Android API library:

- JDK installed on your PC
• Android SDK
• Android IDE (such as Android Studio)
• UG Android API (can be downloaded from our website)
• Android device with Bluetooth
• UG wristband

3.2 Steps to Develop an Android App

The following shows the steps to use the UG Android API library to develop an Android app:

1) Create an Android project.
2) Put UG Android API library (ug_api.aar) in /app/libs folder of your Android project.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Date Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>ug_api.aar</td>
<td>Sep 16, 2017, 3:46 PM</td>
</tr>
</tbody>
</table>
```

3) Add UG Android API library name and path to Android Build Gradle.

```java
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    compile(name:'ug_api', ext:'aar')
}
```

```java
repositories {
    flatDir {
        dirs 'libs' //this way we can find the .aar file in libs folder
    }
}
```

4) Add Bluetooth permission to Android Manifest.
5) Write an Android application.

3.3 UG Android API

The UG Android API provide a series of APIs for developers to manipulate UG wristbands. It contains three classes: UGManager, UGDevice, and UGProfile. The UGManager class is used to search for UG wristbands. The UGDevice class is used to build BLE connection and communicate with UG wristbands. The UGProfile class is used to show the status of UG wristbands.

UGManager class:

Void startScan (ScanCallback cb)

- Start a scan for UG wristbands. Results of the scan are reported using the onScan (UGDevice device) callback of the ScanCallback interface.

Void stopScan ()

- Stop an ongoing UG wristbands scan.

Void close ()

- Stop BLE scanning and close all BLE connection.

Interface ScanCallback{
    void onScan (UGDevice device);
}

- Callback Interface reporting a UG wristband found during a device scan. It is initiated by the startScan (UGManager.ScanCallback) function.

UGDevice class:

Void connect (StatusChangeCallback cb)

- Attempt to connect the Android device to a UG wristband. The connection status is reported using the onStatusChange(UGDevice device, int status) callback of the StatusChangeCallback interface.
Void disconnect()

- Attempt to disconnect from a UG wristband.

Void startDataSensing(DataAvailableCallback cb, int interval)

- Start to read sensor data from a UG wristband with certain interval. The sensor data is reported using the onSensorDataAvailable(UGDevice device, float[] data) callback of the DataAvailableCallback interface. The minimum interval is 11.25ms.

Void stopDataSensing()

- Stop reading sensor data from a UG wristband.

Void calibrateSensor()

- Calibrate the accelerometer and gyroscope of a UG wristband.

Void setLED(Byte[] ledMask)

- Set LEDs for a UG wristband
- e.g. mUGDevice.setLED(new Byte[]{0x01,0x01,0x00,0x00,0x01}): set five LEDs to be {on, on, off, off, on} for mUGDevice.

Void getBatteryLevel()

- Get the percentage of the battery level of a UG device. The battery level is reported using the onBatteryLvlAvailable(UGDevice device, int data) callback of the DataAvailableCallback interface.

String getAddress()

- Get the Mac address of a UG wristband.

Interface StatusChangeCallback{
    void onStatusChange(UGDevice device, int status);
}

- Callback Interface reporting the connection status of a UG. It is initiated by the connect(StatusChangeCallback cb) function.

Interface DataAvailableCallback{
    void onSensorDataAvailable(UGDevice device, float[] data);
    void onBatteryLvlAvailable(UGDevice device, int data);
}
- Callback interface used to deliver sensor data and battery level.
- `onSensorDataAvailable (UGDevice device, float[] data)` returns the accelerometer, gyroscope and magnetometer data from a UG wristband. It is initiated by `startDataSensing (DataAvailableCallback cb, int interval)` function.
- `onBatteryLvlAvailable (UGDevice device, int data)` returns the battery level of a UG wristband. It is initiated by `getBatteryLevel()` function.

**UGProfile class:**

```c
int STATUS_DISCONNECTED
```
- The UG wristband is in disconnected state
- Constant Value: 0 (0x00000000)

```c
int STATUS_CONNECTED
```
- The UG wristband is in connected state
- Constant Value: 0 (0x00000001)

```c
int STATUS_DATA_SENSING_ON
```
- The UG wristband is in data sensing state
- Constant Value: 0 (0x00000002)

```c
int STATUS_DATA_SENSING_OFF
```
- The UG wristband is in data sensing off state
- Constant Value: 0 (0x00000003)

## 4 UG Wristband Firmware Update

The UG wristband firmware can be updated by using nRF Toolbox app developed by Nordic. This app can connect with UG wristband in bootloader mode and send update package to the UG wristband. Once the update package is received, the UG wristband replaces the old firmware with the new one and restart the whole system. The update process is shown as follows:


2) To update the firmware in UG wristband, the UG wristband needs to be in bootloader mode. When the system is turned off, long press button (>20 seconds) puts the system in bootloader mode. In this mode, the LEDs blink one by one.

3) Open nRF Toolbox app. Click “DFU” button.
4) Import update package to nRF Toolbox app. Click “Select File” → Click “Inbox” → select update package.

5) Click “SELECT DEVICE” button → Click “DfuTarg”. DfuTarg can only be discovered when the UG wristband is in bootloader mode.
6) Click “Upload”. After the UG wristband receives the update package, it automatically deletes the old firmware, installs the new one, and restarts the system.