**Documentación Mota Sensora IMUDS**

Mota sensora acelerómetro + giróscopo con comunicación BLE. Versión 0.1

**Características**

* Muestreo ~20Hz
* Palabras de 16 bits en complemento a2

|  |  |  |  |
| --- | --- | --- | --- |
| **Sensor** | **Rango** | **Factor de escalado** | **Unidad** |
| Acelerómetro | ±2 | /16384 | g |
|  | ±4 | /8192 | g |
|  | ±8 | /4096 | g |
|  | ±16 | /2048 | g |
| Giróscopo | ±250 | /131 | º/s |
|  | ±500 | /65.5 | º/s |
|  | ±1000 | /32.8 | º/s |
|  | ±2000 | /16.4 | º/s |

|  |  |  |  |
| --- | --- | --- | --- |
| **Temperatura** | **Rango** | **Factor de escalado** | **Desviación sobre ZRO\* del giróscopo** |
|  | -40ºC hasta 85ºC | /340 +36.53 | ±20 |

\*Zero-Rate output

**Configuración**

La configuración se realiza mediante la escritura de la característica BLE “Conf” perteneciente al servicio BLE “Basic Service”. La escritura de configuración consiste en un byte array de 3 elementos, cuyo contenido podemos ver en la tabla correspondiente.

Tabla . Estructura byte array de configuración.

|  |  |  |
| --- | --- | --- |
| Start/Stop | Rango Accel | Rango Gyro |
| [0] | [1] | [2] |

Tabla . Leyenda byte array de configuración.

|  |  |
| --- | --- |
| **Start/Stop** |  |
| **Iniciar Medida** | 1 |
| **Detener Medida** | 0 |
| **Rango Accel** |  |
| **±2** | 0 |
| **±4** | 1 |
| **±8** | 2 |
| **±16** | 3 |
| **Rango Gyro** |  |
| **±250** | 0 |
| **±500** | 1 |
| **±1000** | 2 |
| **±2000** | 3 |

Ejemplo byte configuración:

* Comenzar a medir con rango de aceleración ±4 y rango de giro ±1000:
* ByteConf = {1,1,2}

**Adquisición de datos**

La adquisición de datos se realiza mediante la lectura de la característica BLE “Data” perteneciente al servicio BLE “Basic Service”.

Cada lectura nos devuelve un byte array de 14 elementos correspondientes a acelerómetro, temperatura y giróscopo en complemento-a2.

Tabla . Contenido byte array de lectura

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | [13] |
| AX\_H | AX\_L | AY\_H | AY\_L | AZ\_H | AZ\_L | T\_H | T\_L | GX\_H | GX\_L | GY\_H | GY\_L | GZ\_H | GZ\_L |

Tabla . Leyenda contenido byte array lectura

|  |  |
| --- | --- |
| **AX\_H** | Byte High. Eje X. Accel |
| **AX\_L** | Byte Low. Eje X. Accel |
| **AY\_H** | Byte High. Eje Y. Accel |
| **AY\_L** | Byte Low. Eje Y. Accel |
| **AZ\_H** | Byte High. Eje Z. Accel |
| **AZ\_L** | Byte Low. Eje Z. Accel |
| **T\_H** | Byte High. Temperatura |
| **T\_L** | Byte Low. Temperatura |
| **GX\_H** | Byte High. Eje X. Gyro |
| **GX\_L** | Byte Low. Eje X. Gyro |
| **GX\_H** | Byte High. Eje Y. Gyro |
| **GX\_L** | Byte Low. Eje Y. Gyro |
| **GX\_H** | Byte High. Eje Z. Gyro |
| **GX\_L** | Byte Low. Eje Z. Gyro |

Ejemplo ilustrativo de interpretación de lectura

1. **Lectura[0]( AX\_H) =**1100 0000
2. **Lectura[1]( AX\_L)=**0000 0000
3. **AX(C-A2)=**1100 0000 0000 0000
4. **AX(bin)=**0100 0000 0000 0000
5. **AX(int)=**16384 (Rango ±2g)
6. **AX(g)=**16384/16384 = 1
7. **AX (**m/s2**)≈** 9.8

**Suscripción**

La característica de lectura posee un descriptor al que podemos suscribirnos. Al suscribirnos, los datos producidos por la lectura llegarán automáticamente a nuestro dispositivo de adquisición a través del CallBack BLE.

**Anexo I: Tabla servidor GATT.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UUID** | **Tipo** | **Nombre** | **Valor** | **Servicio** | **Descriptor** |
| **1800** | **Servicio** | Generic Acces Profile |  |  |  |
| **2a00** | Característica | Device Name | “MotaIMUDSV001”[Const] | Generic Acces Profile | No |
| **2a01** | Característica | Appearence | “0000” [Const] | Generic Acces Profile | No |
| **cc31c41a-f835-4af5-ab48-81bb6baba638** | **Servicio** | Basic Service |  |  |  |
| **ca5fd17a-c934-4118-96bc-f3e3892196f6** | Característica | Conf | Variable. Longitud = 3 bytes | Basic Service | No |
| **ca5fd17a-c934-4118-96bc-f3e3892196f8** | Característica | Data | Variable. Longitud = 14 bytes | Basic Service | 2902 |

**Anexo II: Clase Ejemplo para aplicación Android.**

import android.bluetooth.BluetoothAdapter;  
import android.bluetooth.BluetoothDevice;  
import android.bluetooth.BluetoothGatt;  
import android.bluetooth.BluetoothGattCallback;  
import android.bluetooth.BluetoothGattCharacteristic;  
import android.bluetooth.BluetoothGattDescriptor;  
import android.bluetooth.BluetoothManager;  
import android.content.Context;  
import android.util.Log;  
  
import java.util.UUID;  
  
*/\*\*  
 \* Created by SamuelPS on 18/05/2017.  
 \*/*public class MotaBle {  
  
 Context context;  
 BluetoothGatt motaGatt;  
 BluetoothManager mBluetoothManager;  
 BluetoothAdapter mBluetoothAdapter;  
 String mBluetoothDeviceAddress;  
 public boolean init;  
 public byte[] conf=new byte[3];  
  
 public static final UUID *Service* = UUID.*fromString*("cc31c41a-f835-4af5-ab48-81bb6baba638");  
 public static final UUID *Conf\_GattChar* = UUID.*fromString*("ca5fd17a-c934-4118-96bc-f3e3892196f6");  
 public static final UUID *Data\_GattChar* = UUID.*fromString*("ca5fd17a-c934-4118-96bc-f3e3892196f8");  
  
 public static final int *START*=1;  
 public static final int *STOP*=0;  
  
 public static final int *ACCEL\_RANGE\_2*=0;  
 public static final int *ACCEL\_RANGE\_4*=1;  
 public static final int *ACCEL\_RANGE\_8*=2;  
 public static final int *ACCEL\_RANGE\_16*=3;  
  
 public static final int *GYRO\_RANGE\_250*=0;  
 public static final int *GYRO\_RANGE\_500*=1;  
 public static final int *GYRO\_RANGE\_1000*=2;  
 public static final int *GYRO\_RANGE\_2000*=3;  
  
 public MotaBle(Context ctx){  
 context=ctx;  
 init=false;  
 }  
  
  
 */\*\*  
 \* Inicializamos bluetooth related  
 \** ***@return*** *\*/* public boolean init(){  
 Log.*i*("Init", "Inicializando...");  
 conf[0]=0;  
 conf[1]=0;  
 conf[2]=0;  
 mBluetoothDeviceAddress=null;  
 init=true;  
 mBluetoothManager = (BluetoothManager) context.getSystemService(Context.*BLUETOOTH\_SERVICE*);  
 mBluetoothAdapter = mBluetoothManager.getAdapter();  
  
 return true;  
  
 }  
  
 */\*\*  
 \* Conectamos a la mota  
 \** ***@param*** *mblecallback  
 \** ***@param*** *address  
 \** ***@return*** *\*/* public boolean connect(final BluetoothGattCallback mblecallback, final String address){  
  
 //Si existe conexion, reconectamos.  
 if (mBluetoothDeviceAddress != null && address.equals(mBluetoothDeviceAddress)  
 && motaGatt != null) {  
 if (motaGatt.connect()) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 //Comenzamos connexion  
 mBluetoothDeviceAddress=address;  
 BluetoothDevice device= mBluetoothAdapter.getRemoteDevice(mBluetoothDeviceAddress);  
 motaGatt = device.connectGatt(context,false,mblecallback);  
 return true;  
 }  
  
 public void disconnect(){  
 if (mBluetoothAdapter == null || motaGatt == null) {  
 return;  
 }  
 motaGatt.disconnect();  
 }  
  
 */\*\*  
 \* Escribimos caracateristica de configuración  
 \** ***@param*** *startstop  
 \** ***@param*** *accelrange  
 \** ***@param*** *gyrorange  
 \** ***@return*** *\*/* public boolean conf(int startstop, int accelrange, int gyrorange){  
 conf[0]=(byte)startstop;  
 conf[1]=(byte)accelrange;  
 conf[2]=(byte)gyrorange;  
  
 BluetoothGattCharacteristic confChar = motaGatt.getService(*Service*).getCharacteristic(*Conf\_GattChar*);  
 confChar.setValue(conf);  
 return motaGatt.writeCharacteristic(confChar);  
 }  
  
 */\*\*  
 \* Nos suscribimos a la caracteristica de lectura  
 \*/* public boolean suscribe(){  
 BluetoothGattCharacteristic mychar = motaGatt.getService(*Service*).getCharacteristic(*Data\_GattChar*);  
 motaGatt.setCharacteristicNotification(mychar,true);  
 UUID uuid = UUID.*fromString*("00002902-0000-1000-8000-00805f9b34fb");  
 BluetoothGattDescriptor descriptor = mychar.getDescriptor(uuid);  
 descriptor.setValue(BluetoothGattDescriptor.*ENABLE\_NOTIFICATION\_VALUE*);  
 return motaGatt.writeDescriptor(descriptor);  
 }  
  
 public void close(){  
 if (motaGatt == null) {  
 return;  
 }  
 motaGatt.close();  
 motaGatt = null;  
 }  
  
  
}