PCLinkLibrary

For Android API

TAIDOC TECHNOLOGY CO.

2014/12/17

Version 1.2.0

PCLinkLibrary

For Android API

INDEX

Version 1.2.0

Abstract

PCLinkLibrary tool, which is a convenient API to communicate with meter, include Blood Glucose Monitor, Blood Pressure Monitors, Blood Glucose and Blood Pressure 2-in-1 Monitors, Infrared Thermometers and Weight Scale Monitor.

PCLinkLibrary can easy to communicate with meter, such as create a connection channel, connect meter and get the information which the standard TaiDoc Bus format.

PCLinkLibrary support the Android Mobile Device now, but require Android 2.3.3 and up.

How to use PCLinkLibrary

PCLinkLibrary has several important features to pay attention to. There is a tutorial to connect a type one meter with android mobile device. What is type one meter? That is a meter and transfer data by Bluetooth. When the meter in PCL mode, it will start and wait other device to connect. If you want to use PCLinklibrary and connect a type one Bluetooth meter with android mobile device. First, in the android Bluetooth transfer process, it must be execute another thread to connect a Bluetooth device. For example

```
new Thread(new Runnable() {
    @Override
    public void run() {
        // Do work to connect a meter
        connectBluetoothMeter();
    }
}).start();
```

Note: connectBluetoothMeter() is a fictional method.

If you never pair a meter, PCLinklibray cannot work, there are two way to pair a meter.

1. Go to Settings→Wireless & networks→Bluetooth settings, and pair meter manual.
2. Using PCLinklibrary BluetoothUtil to pair meter and get the information. For example

```
// Search all device in region
List<BluetoothDevice> remoteDeviceList = new ArrayList<BluetoothDevice>();
remoteDeviceList = BluetoothUtil.searchRemoteDevices(getBaseContext());

// Filtering the remoteDeviceList by device name, and prepare the wantToPairRemoteDeviceList
List<BluetoothDevice> wantToPairRemoteDeviceList = new ArrayList<BluetoothDevice>();
for (BluetoothDevice bluetoothDevice : remoteDeviceList) {
    if(bluetoothDevice.getName().toLowerCase().contains("taidoc")) {
        wantToPairRemoteDeviceList.add(bluetoothDevice);
    } /* end of if */
} /* end of for */

// Pair all remote devices list
List<BluetoothDevice> pairedRemoteDeviceList = new ArrayList<BluetoothDevice>();
pairedRemoteDeviceList = BluetoothUtil.pairWithRemoteDevices(getBaseContext(),
wantToPairRemoteDeviceList);

// Get the MAC Address from pairedRemoteDeviceList
String macAddress;
if (pairedRemoteDeviceList.size() > 0 ) {
   //get the first meter in pairedRemoteDeviceList, and get the MAC address
   macAddress = pairedRemoteDeviceList.get(0).getAddress();
} /* end of if */
```

After pair the meter, it needs to create a Bluetooth connection object and a meter object.

For example

```
// Local Bluetooth adapter
private BluetoothAdapter localBluetoothAdapter = null;
// Member object for the android bluetooth connection
private AndroidBluetoothConnection connection = null;

// Create an android bluetooth connection and start to listen mode(type II)
// Get local Bluetooth adapter
localBluetoothAdapter = BluetoothUtil.getBluetoothAdapter();

// If BT is not on, let it be enabled.
if (!localBluetoothAdapter.isEnabled()) {
    BluetoothUtil.turnOnBlutoothModule();
} /* end of if */

// Initialize the connection to perform android bluetooth connections
if (connection == null) {
    connection = ConnectionManager.createAndroidBluetoothConnection();
} /* end of if */

// When android Bluetooth connection was be create, start to connect to a meter
if (connection.getState() != AndroidBluetoothConnection.STATE_CONNECTED) {
    BluetoothDevice device = BluetoothUtil.getPairedDevice(selectMeterAddress);
    // Attempt to connect to the device, Type I
    connection.connect(device);
    // connect time out default is 10s
    long startConnectTime = System.currentTimeMillis();
    while (connection.getState() != AndroidBluetoothConnection.STATE_CONNECTED) {
        long conntectTime = System.currentTimeMillis();
        if ((conntectTime - startConnectTime) >
        AndroidBluetoothConnection.BT_CONNECT_TIMEOUT)){
            // throw a CommunicationTimeoutException and break the loop.
            throw new CommunicationTimeoutException();
        } /* end of if */
    } /* end of while */
}/* end of if */

// Detect connected meter information.
AbstractMeter meter = MeterManager.detectConnectedMeter(connection);
```

There is a meter object that was created by PCLinkLibrary MeterMannger.detectConnectedMeter(). You can communicate with meter right now; there are several methods to get the information of meter.

- getSystemClock: read meter system clock time.
- setSystemClock: write clock time to meter
- getDeviceModel: read meter model information
- getStorageNumberAndNewestIndex: read storage date count
- getStorageDataRecord: read the storage record.
- getSerialNumberRecord : read the serial number of meter
- turnOffMeterOrBluetooth: power off the meter
- clearMeasureRecords: clear storage record of meter

There is an example about import medical record to your application.

Example

```
// Detect connected meter information.
AbstractMeter meter = MeterManager.detectConnectedMeter(connection);

// Start to import the medical record in meter
// Sync current time to meter
Date nowTime = new Date();
meter.setSystemClock(nowTime);

// Get the project code
String projectCode = meter.getDeviceModel().getProjectCode();

// Get the serial number
String serialNumber = meter.getSerialNumberRecord().getSerialNumber();

// Get the storage record count
int storageCount = meter.getStorageNumberAndNewestIndex(User.CurrentUser).getStorageNumber();

// Import all medical record in meter
List<AbstractRecord> recordList = new ArrayList<AbstractRecord>();
for (int i = 0; i < storageCount; i++) {
    AbstractRecord record = meter.getStorageDataRecord(0, User.CurrentUser);
    recordList.add(record);
} /* end of for */

// Convert the List<AbstractRecord> to Java object
for (AbstractRecord record : recordList) {
    if (record instanceof BloodGlucoseRecord) { //BG record
        SimpleDateFormat formatterDate = new SimpleDateFormat("yyyy/MM/dd hh:mm aa");
        String measurementDate =
               formatterDate.format(((BloodGlucoseRecord) record).getMeasureTime());
        int bgValue = ((BloodGlucoseRecord) record).getGlucoseValue();
    } else if (record instanceof BloodPressureRecord) { //BP record
        SimpleDateFormat formatterDate = new SimpleDateFormat("yyyy/MM/dd hh:mm aa");
        String measurementDate =
               formatterDate.format(((BloodPressureRecord) record).getMeasureTime());
        int sysValue = ((BloodPressureRecord) record).getSystolicValue();
        int diaValue = ((BloodPressureRecord) record).getDiastolicValue();
        int pulseValue = ((BloodPressureRecord) record).getPulseValue();
    } else if (record instanceof TemperatureRecord) { //Temperature record
        SimpleDateFormat formatterDate = new SimpleDateFormat("yyyy/MM/dd hh:mm aa");
        String measurementDate =
               formatterDate.format(((TemperatureRecord) record).getMeasureTime());
        double thermometerValue = ((TemperatureRecord) record).getObjectTemperatureValue();
        DecimalFormat df = new DecimalFormat("#.##");
        String thermometerStringValue = df.format(thermometerValue)
    } /* end of if */
} /* end of for */

// Import done, clear all record in meter
meter.clearMeasureRecords(User.CurrentUser);

// Clear done, power off the meter
meter.turnOffMeterOrBluetooth(0);
```

How to connect to BLE meter

1. Add the meter mac address to updatePairedList.

For example

```
private void updatePairedList() {
    Map<String, String> addrs = new HashMap<String, String>();
    String addrKey = PCLinkLibraryDemoConstant.BLE_PAIRED_METER_ADDR_ + String.valueOf(0);
    addrs.put(addrKey, mMacAddress);
    mConnection.updatePairedList(addrs, 1);
}
```

2. Call LeListen of AndroidBluetoothConnection to scan for meter on updatePairedList.

For example

```
updatePairedList();
if (mConnection.getState() == AndroidBluetoothConnection.STATE_NONE) {
    // Start the Android Bluetooth connection services to listen mode
    mConnection.LeListen();
}
```

3. If the meter on updatePairedList is advertising, then triggers MESSAGE_STATE_SCANED_DEVICE id.

For example

```
case MESSAGE_STATE_SCANED_DEVICE:
    BluetoothDevice device =
       BluetoothUtil.getPairedDevice(mConnection.getConnectedDeviceAddress());
    // Attempt to connect to the device
    mConnection.LeConnect(getApplicationContext(), device);
    // connect time out default is 10s
    long startConnectTime = System.currentTimeMillis();
    while (connection.getState() != AndroidBluetoothConnection.STATE_CONNECTED) {
        long conntectTime = System.currentTimeMillis();
        if ((conntectTime - startConnectTime) >
        AndroidBluetoothConnection.BT_CONNECT_TIMEOUT)){
            // throw a CommunicationTimeoutException and break the loop.
            throw new CommunicationTimeoutException();
        } /* end of if */
    } /* end of while */
    mConnection.LeConnected(device);
    // Detect connected meter information.
    AbstractMeter meter = MeterManager.detectConnectedMeter(connection);
```

How to connect to KNV-V125 meter

1. Add the meter mac address to updatePairedList.

For example

```
private void updatePairedList() {
    Map<String, String> addrs = new HashMap<String, String>();
    String addrKey = PCLinkLibraryDemoConstant.BLE_PAIRED_METER_ADDR_ + String.valueOf(0);
    addrs.put(addrKey, mMacAddress);
    mConnection.updatePairedList(addrs, 1);
}
```

2. Call LeListen of AndroidBluetoothConnection to scan for meter on updatePairedList.

For example

```
updatePairedList();
if (mConnection.getState() == AndroidBluetoothConnection.STATE_NONE) {
    // Start the Android Bluetooth connection services to listen mode
    mConnection.LeListen();
}
```

3. If the meter on updatePairedList is advertising, then triggers MESSAGE_STATE_SCANED_DEVICE id.

For example

```
case MESSAGE_STATE_SCANED_DEVICE:
    BluetoothDevice device =
       BluetoothUtil.getPairedDevice(mConnection.getConnectedDeviceAddress());
    // Attempt to connect to the device
    mConnection.LeConnect(getApplicationContext(), device);
    // connect time out default is 10s
    long startConnectTime = System.currentTimeMillis();
    while (connection.getState() != AndroidBluetoothConnection.STATE_CONNECTED) {
        long conntectTime = System.currentTimeMillis();
        if ((conntectTime - startConnectTime) >
        AndroidBluetoothConnection.BT_CONNECT_TIMEOUT)){
            // throw a CommunicationTimeoutException and break the loop.
            throw new CommunicationTimeoutException();
        } /* end of if */
    } /* end of while */
    mConnection.LeConnected(device);
    // Detect connected meter information.
    AbstractMeter meter = MeterManager.detectConnectedMeter(connection, handler);
    ….
    ….
    ….
    Private getStorageDataRecordInterface handler = new getStorageDataRecordInterface() {
        @Override
        Public void onFinish(WeightScaleRecord record) {
            if (record != null) {
                // got record here
            }
            else {
            }
        }
    };
```

Note: The handler must be registered in "detectConnectedMeter". AndroidBluetoothConnection will call this handler after each measurement.

Note:

1. Stop or disconnect the local android bluetooth connection, call connection.disconnect();

```
// Stop the Android Bluetooth connection.
if (connection != null) {
    connection.disconnect();
} /* end of if */

// for ble
If (connection != null) {
    Connection.LeDisconect();
} /* end of if */
```

2. sendBACommand() is the unique method in TD4283 which will beep once after the meter receive BA command. How to use the method,

```
// Get the meter
AbstractMeter meter = MeterManager.detectConnectedMeter(connection);

// Send BA command to TD4283 meter.
((TD4283) meter).sendBACommand();
```

Related reference

1. Android Developer - Bluetooth
2. Android Developer - BluetoothChat Example