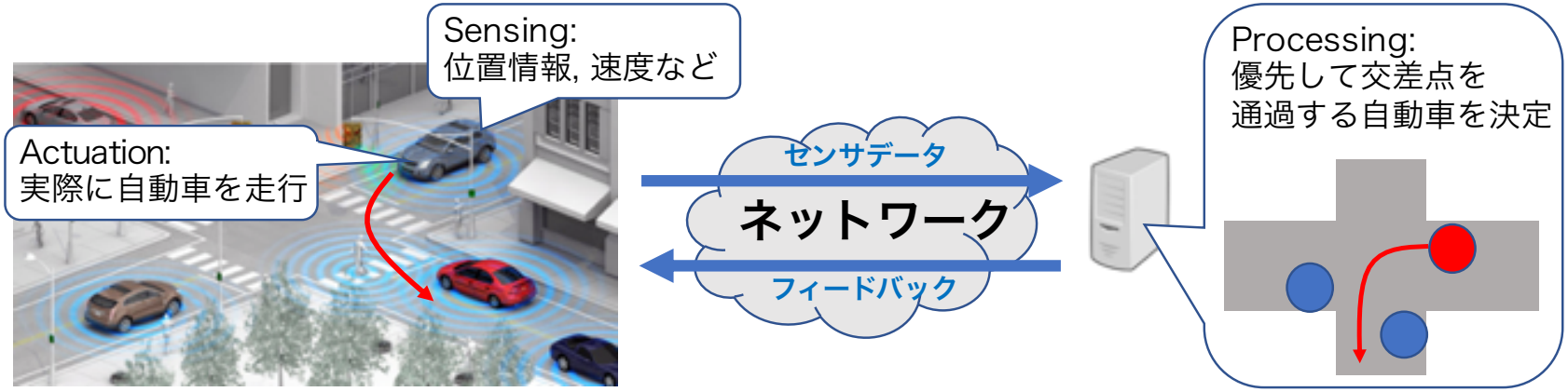
センサネットワーク実験

担当教員: 山本 寛

実験の目的

- IoT (Internet of Things)とは？
  - 様々なモノをインターネットに接続し、相互に制御することで人の様々な活動を支援する技術の総称
- IoT(Internet of Things)システムの機能構成
  - 例: コネクテッドカー



- センシング: 様々なセンサにより現実世界の状態を計測
- プロセッシング: 計測結果を分析して現実世界をどのように制御するか判断
- アクチュエーション: 制御内容を現実世界にフィードバック

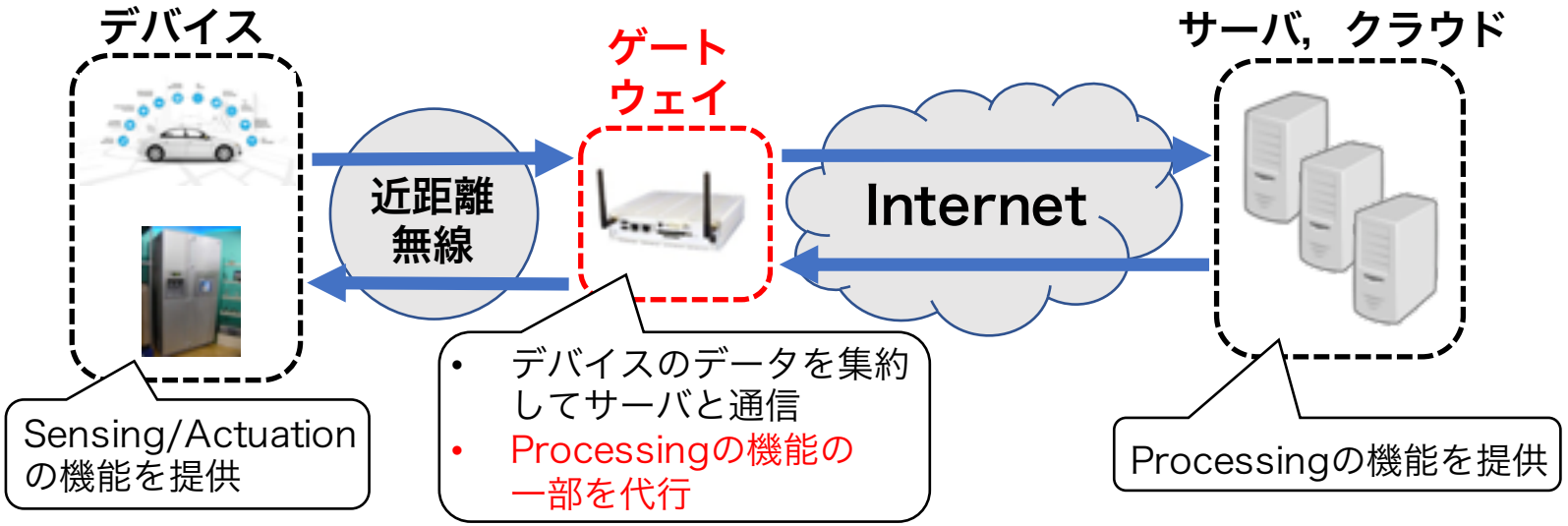
目的: 3つの機能を含む簡単なIoTシステムを開発

実験の注意点

- 必ずUSBメモリを購入して持参すること
  - 実験用に貸し出しているシンクライアントシステムのノートPCは、シャットダウンするとデータが消えます！！
  - 作業終了前に、必ずバックアップを取ること
  - OneDriveなどのオンラインストレージへのバックアップも可

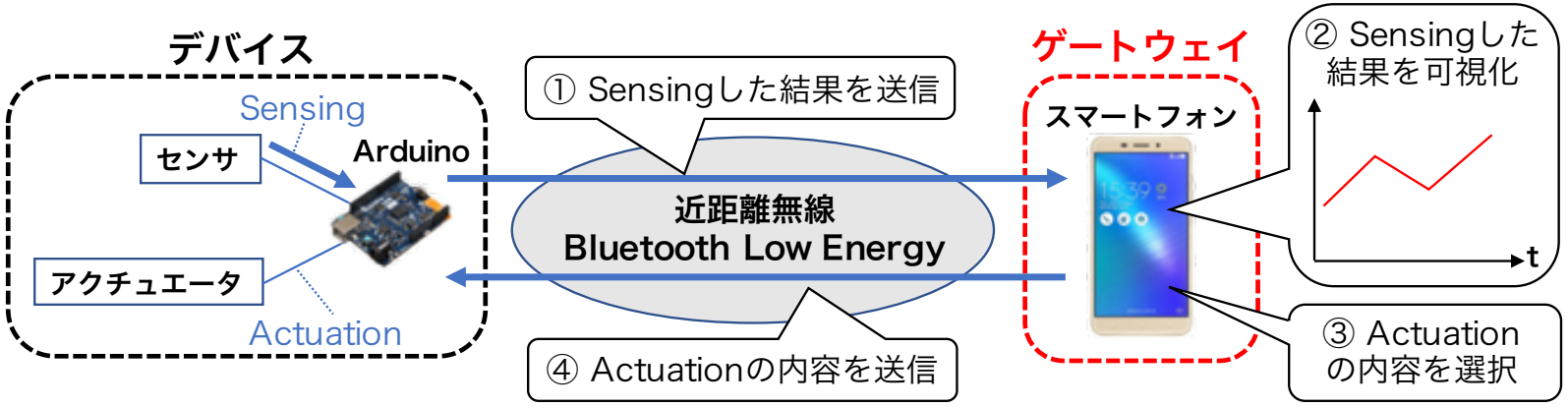
- 自分のノートPCを利用する(BYOD)のがベスト！！
  - インストールが必要なツール
  - Android Studio: https://developer.android.com/studio/
  - Arduino IDE: https://www.arduino.cc/en/Main/Software
  - インストール方法についてはテキストを参照
    - ただし、インストールには10GB程度の空き容量が必要

開発するシステムの基本的な構成



- 本実験では、簡略化したIoTシステムを開発
  - 「Androidスマートフォン」を利用したゲートウェイの開発
  - 組込システムである「Arduino」を利用したデバイスの開発
  - デバイス/ゲートウェイ間のBLE (Bluetooth Low Energy)による通信機能の開発

実験の手順



- Arduino
  - センシング/アクチュエーションの機能を備えたデバイスの開発
  - センサによるセンシングの結果をもとに、アクチュエーションの種類を選択してアクチュエータを制御
    （例: 照度センサがある閾値を下回るとLEDをONする）
- Androidスマートフォン
  - センシング結果を可視化するアプリケーションの開発
  - Androidスマートフォンに搭載されているセンサ
    （加速度、ジャイロなど）が計測した結果を折れ線グラフで表示
- Androidスマートフォン/Arduino
  - 近距離無線(BLE: Bluetooth Low Energy)による通信機能の開発
- Androidスマートフォン/Arduino : 独自のIoTシステムの開発

Arduino開発手順

1. Arduino IDEの起動
2. （PC起動初回のみ）開発用のライブラリを導入
3. サンプルプログラムをベースにプログラミング
4. プログラムをArduinoにインストール

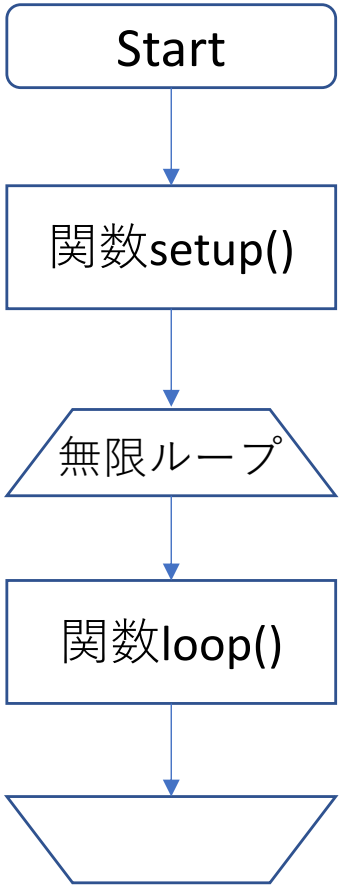
プログラムの基本構成 (1/2)

- 「情報理工基礎演習」で学習したProcessingと同じ構成

- 関数 setup
  - Arduinoが起動した後に、一度だけ実行される処理を記述
  - 変数や入出力ポートの初期化などの処理を行う

- 関数 loop
  - 関数setupの処理が実行された後に、繰り返し実行される処理を記述
  - Processingにおける関数drawと同じ役割
  - センサからの値の読み取り、判断、アクチュエータの制御、などの処理を行う

プログラムの基本構成 (2/2)



- 変数の初期化
- 入出力ポートの初期化

例: pinMode

- センサからの値の取得

例: analogRead, digitalRead

- センサデータの処理、BLE通信、など
- アクチュエータの制御

例: analogWrite, digitalWrite

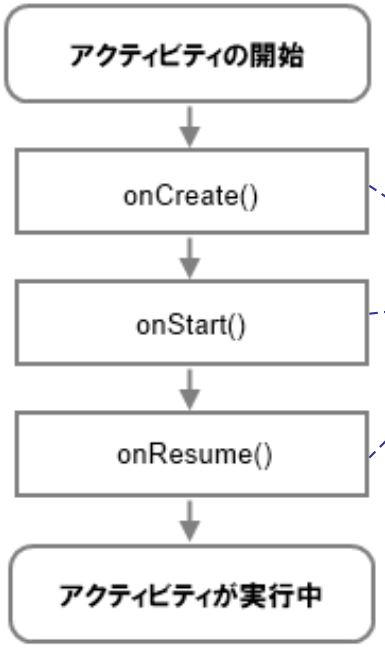
Androidアプリケーション開発手順

1. Android Studioの起動
2. 新しいプロジェクトの作成
3. 開発前の事前作業
4. 画面のレイアウトを設計
5. スマートフォンの機能を利用する権限を設定
6. ソースコードを編集
7. アプリケーションをスマートフォンにインストール
8. （参考）アプリケーションの開発を終了
9. （参考）過去に作成したプロジェクトの読み込み

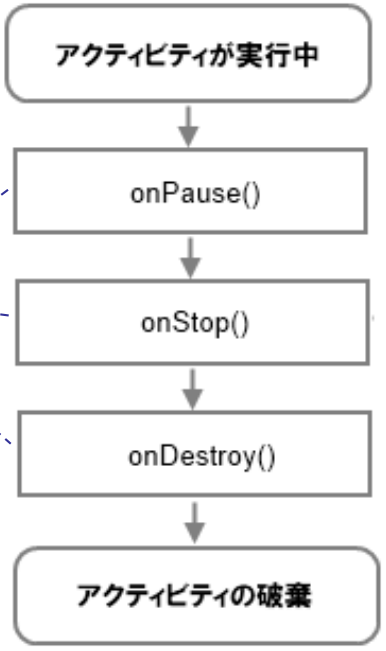
Androidプログラミングのポイント

- main関数がありません、、、
- アプリの「起動」、「停止」などのイベントに対応したコールバック関数が順番に実行

プログラムの開始時



プログラムの終了時

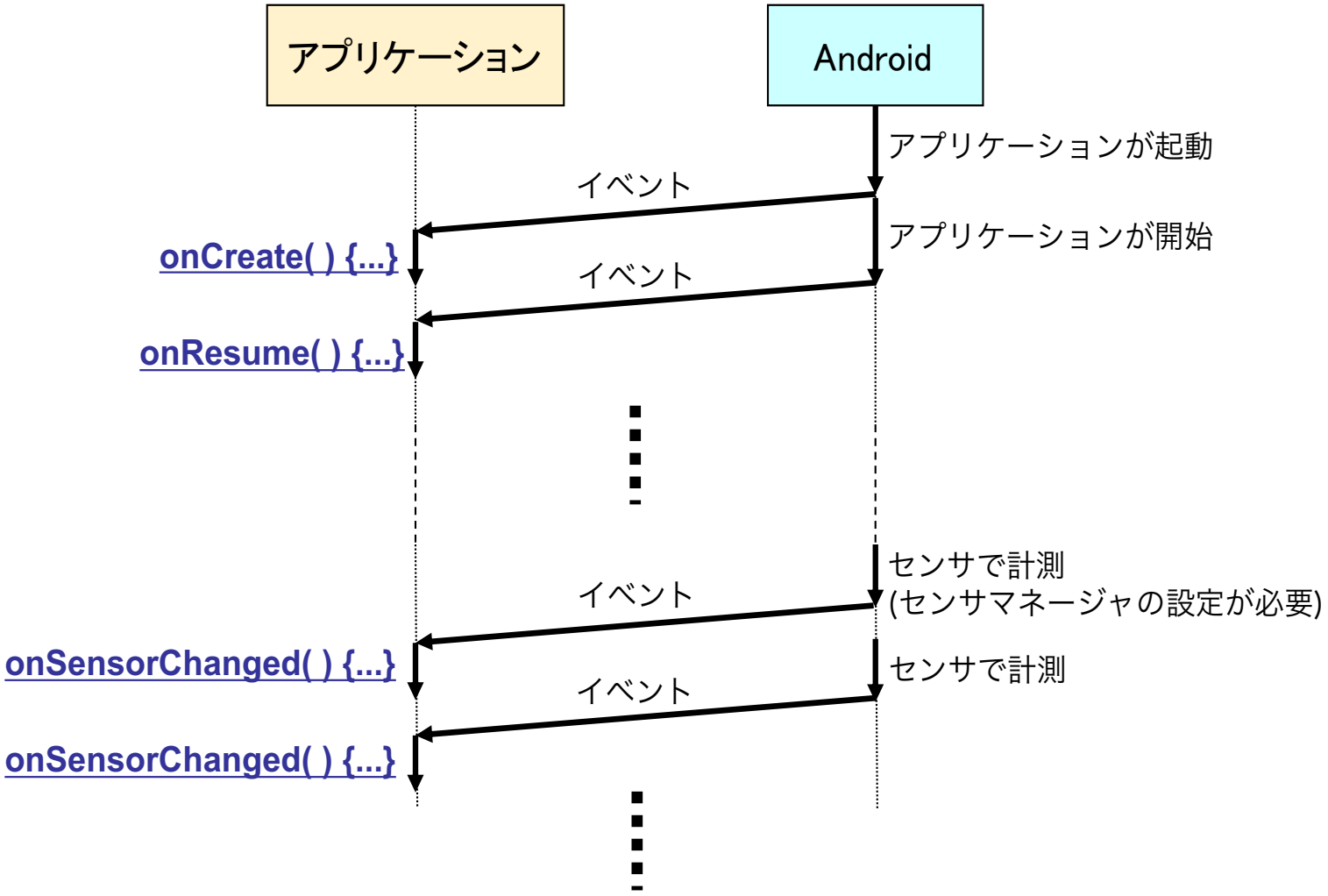


コールバック関数が順番に呼び出される

C言語やJavaのプログラムと違い，main関数が存在しない

（参考）コールバックメソッドがどのタイミングで呼ばれるのかのテスト
http://www.javadrive.jp/android/activity/index7.html

主要なコールバック関数 (1/3)



主要なコールバック関数 (2/3)

- onResume()

```
super.onResume();

// 情報を取得するセンサーの設定（今回は加速度センサを取得）
List<Sensor> sensors = manager.getSensorList(Sensor.TYPE_ACCELEROMETER);
Sensor sensor = sensors.get(0);

// センサーからの情報の取得を開始
manager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_UI);
```

定数	意味
Sensor.TYPE_ACCELEROMETER	加速度センサー
Sensor.TYPE_GYROSCOPE	ジャイロスコープ
Sensor.TYPE_MAGNETIC_FIELD	地磁気センサー（コンパス）
Sensor.TYPE_LIGHT	照度センサー
Sensor.TYPE_PROXIMITY	接近センサー

主要なコールバック関数 (3/3)

- onSensorChanged(SensorEvent event)
  - 引数eventの中に、センサが計測した値が格納

```
// 取得した情報が加速度センサーからのものか確認
if(event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){

    // 受け取った情報を格納用の配列にコピー
    values = event.values.clone();

    // 受け取った情報を表示欄に表示
    sensor1.setText("Acc X-axis: " + values[0]);
    sensor2.setText("Acc Y-axis: " + values[1]);
    sensor3.setText("Acc Z-axis: " + values[2]);
}
```

グラフ表示機能の追加

1. グラフ表示用のライブラリを追加
   - build.gradleを編集（2箇所）

ライブラリのダウンロード元を指定

```
repositories{
    maven {url "http://jitpack.io"}
}
```

ライブラリを指定

```
implementation
'com.github.PhilJay:MPAndroidChart:v3.0.3'
```

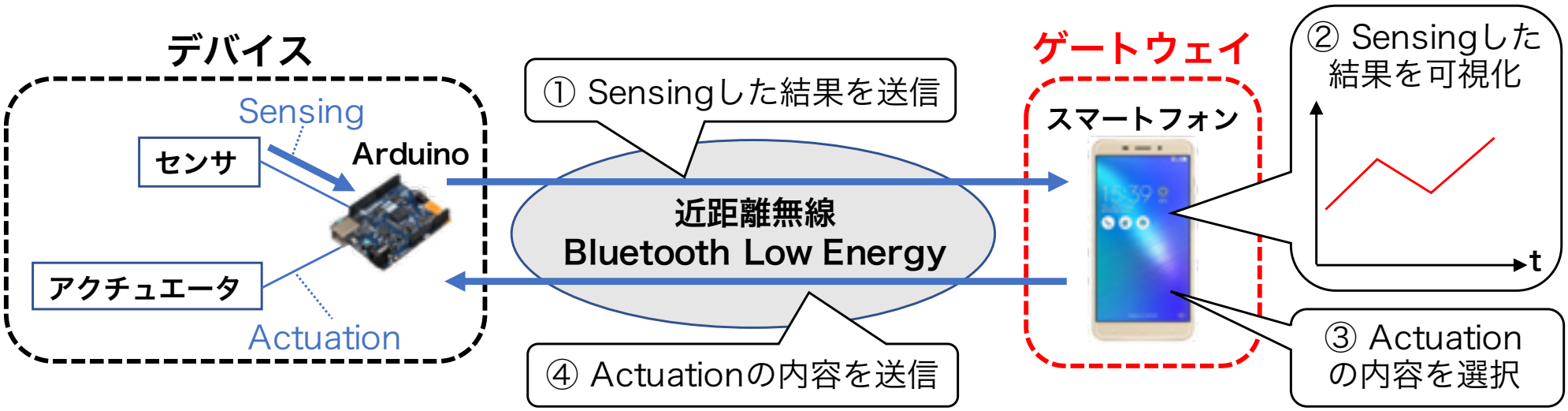
   - 「Sync Now」をクリックしてライブラリをダウンロード
2. グラフ表示を含むように画面レイアウトを再設計
   - activity_main.xmlを編集
   - TextViewの代わりに以下を使用
     「**com.github.mikephil.charting.charts.LineChart**」
3. ソースコードを編集

BLEによるデバイスとゲートウェイの連携

1. センサから取得したデータをデバイスからゲートウェイへ送信

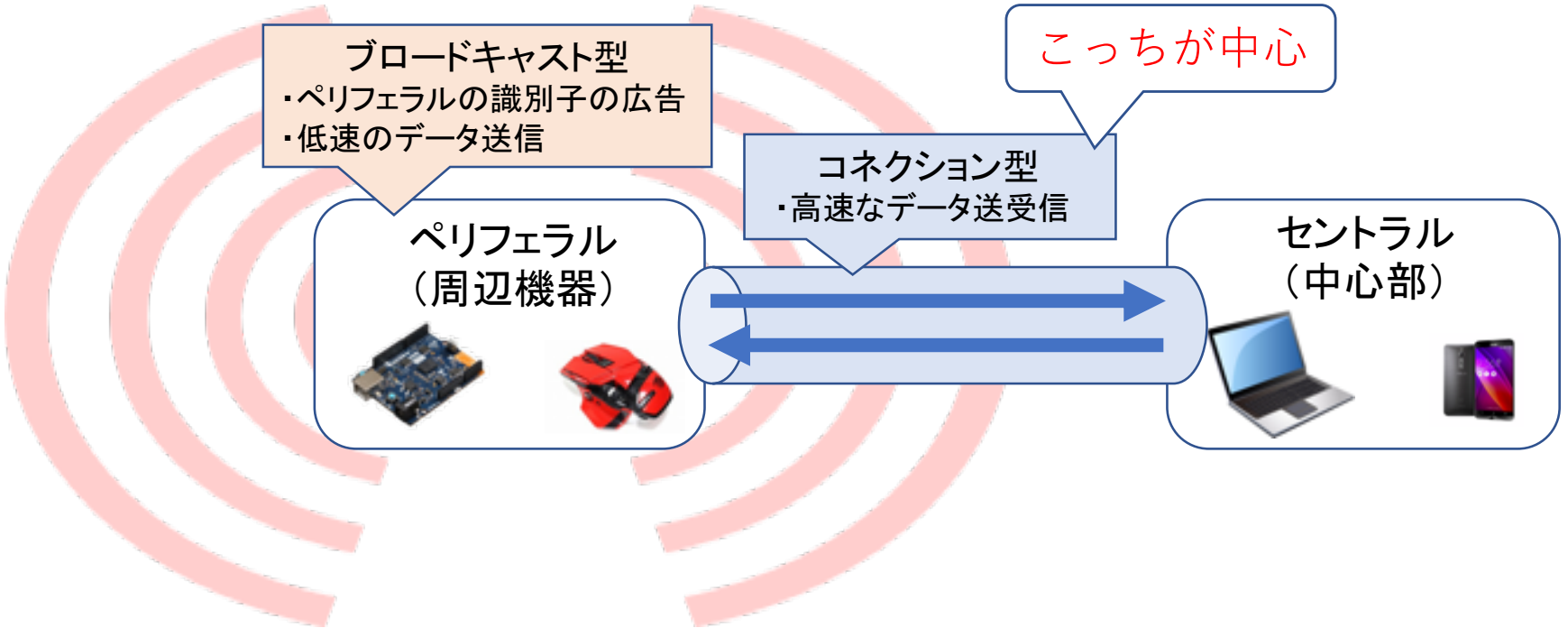
[ゲートウェイ]
センサデータを解析し、アクチュエータの制御に関するデータを生成

2. アクチュエータの制御に関するデータをゲートウェイからデバイスへ送信



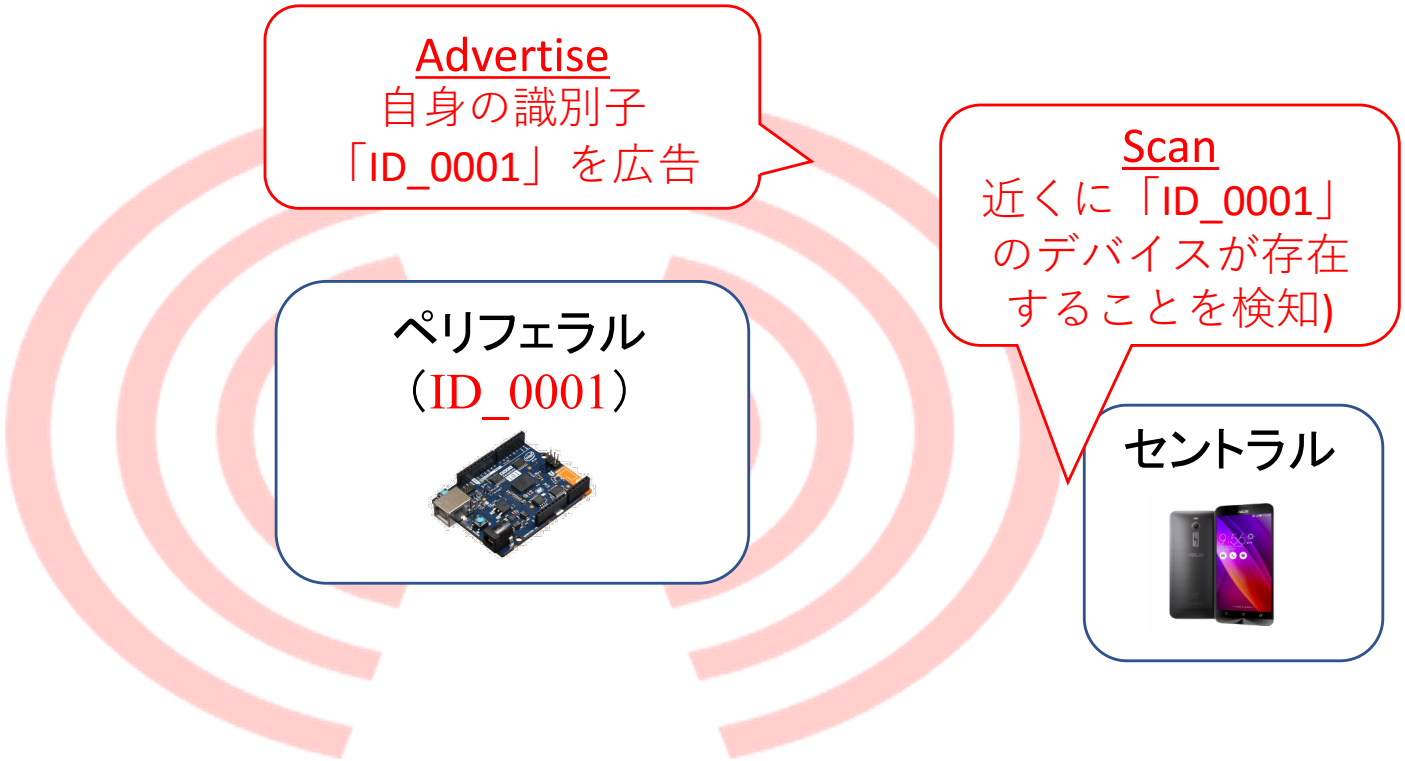
BLEによるデータ通信の種類

- セントラル
  - BLE通信の中心になる機器
  - スマートフォン/タブレットやPC
- ペリフェラル
  - セントラルから利用される周辺機器
  - キーボード/マウス、ヘッドセット、デバイス



コネクション型通信の基礎 (1/3)

- 通信の手順
  1. Advertise: ペリフェラルが周辺に広告パケットをブロードキャスト
  2. Scan: セントラルが周辺にどんなペリフェラルが存在するか検索

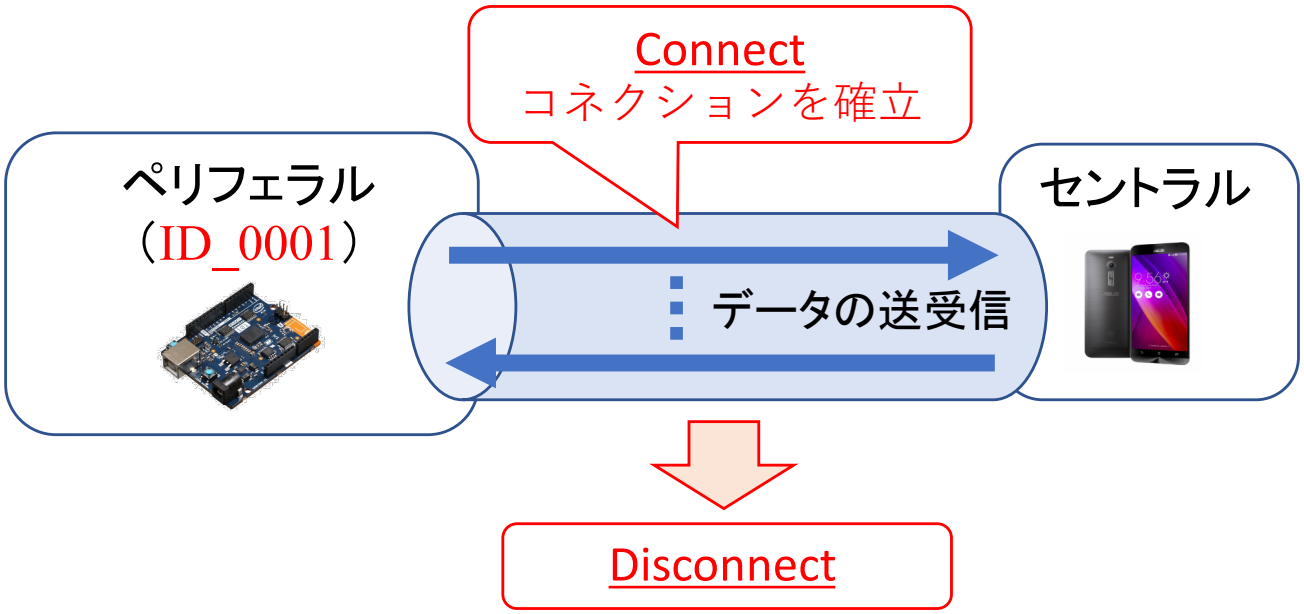


コネクション型通信の基礎 (2/3)

- 通信の手順（続き）
  3. Connect: セントラルが指定したペリフェラルとの間でコネクションを確立

セントラル・ペリフェラル間でデータ送受信

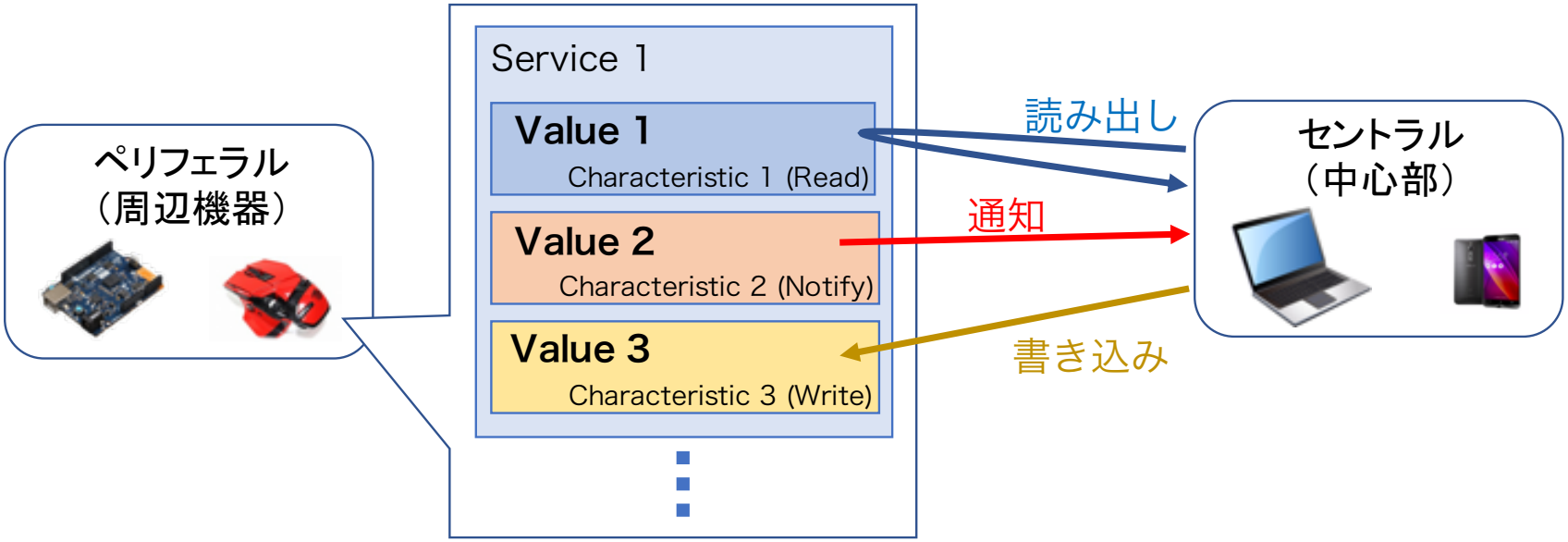
  4. Disconnect: コネクションを破棄



コネクション型通信の基礎 (3/3)

- データ送受信の種類
  - Read: セントラルがペリフェラルから値を読み出す
  - Notify: 値が更新された際に、ペリフェラルからセントラルへ通知
  - Write: セントラルがペリフェラルへ値を書き込む

- データ送受信の対象となる変数(Characteristic)ごとに、種類(Read, Notify, Write)を選択



Arduino (ペリフェラル)側のプログラミング

- データ送受信の対象となる変数(Characteristic)の定義が重要
  - writeCharacteristic: データ受信 (Write)用の変数
  - notifyCharacteristic: データ送信 (Notify)用の変数

送受信できるバイト数

```
// write(Android -> Arduinoへの通信)用Characteristicの作成
BLECharacteristic writeCharacteristic
                  ("19B10011-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite, 2);
// notify(Arduino -> Androidへの通信)用Characteristicの作成
BLECharacteristic notifyCharacteristic
                  ("19B10012-E8F2-537E-4F6C-D104768A1214", BLERead | BLENotify, 2);
```

- データ送信用の変数への値の設定 (セントラルへの送信)

```
// センサーから取得した値をAndroidに送信
notifyCharacteristic.setValue(value, 2);
```

- データ受信用の変数の値読み出し (セントラルからの受信)

```
// Androidからアクチュエータ制御用の値を受信
memcpy(value, writeCharacteristic.value(), 2);
```

Android (セントラル)側のプログラミング (1/2)

- 「スマートフォンの機能を利用する権限を設定」が必要
  - AndroidManifest.xml
  - Bluetoothを利用する設定と、位置情報を利用する設定

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="…">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

…
</manifest>
```

Android (セントラル)側のプログラミング (2/2)

- ソースコードにおける大まかな処理の流れ
  1. ペリフェラルから送信されている広告パケットの検知
     - コールバック関数: onScanChanged
     - ペリフェラルの識別子や受信電波強度を取得
  2. ペリフェラルとのコネクションの確立
     - コールバック関数: onConnectionStateChanged
     - 変数の探索を要求
  3. 変数(Characteristic)の発見
     - コールバック関数: onServiceDiscovered
     - 変数の更新通知を受け取る準備
  4. 変数の更新通知を受信
     - コールバック関数: onCharacteristicChanged
     - 値を取得してArduinoに返信する値を決定 (DecideControlParameter)

最重要

発表会資料の内容

- プログラムの概要
  - 外部仕様、内部仕様など
- プログラムで工夫した点、
  苦労した点、独自機能などについての説明
- プログラムの実行結果
  （動画が推奨）

発表時間は5分