マルチストリーミングセンサデータ向け リアルタイム空間補間可視化システム

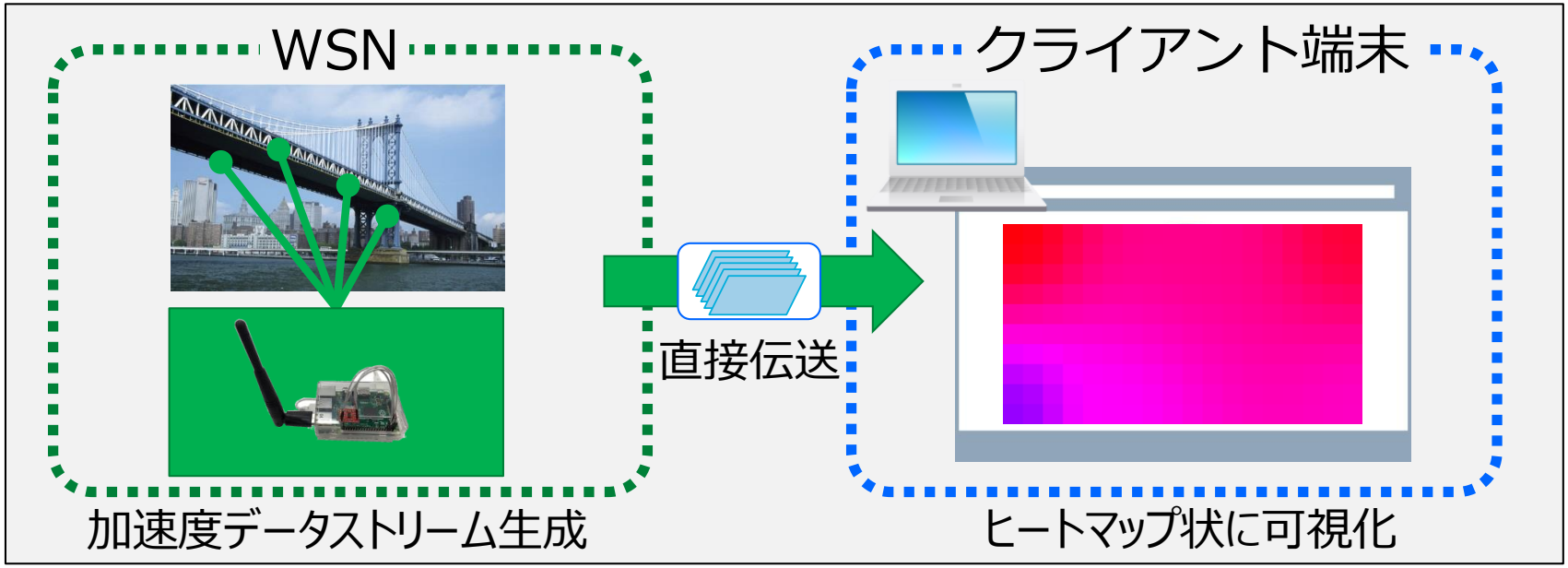
静岡大学 情報学部 峰野研究室

2017/10/05

システム概要

概要

- **マルチストリーミングセンサデータ向けリアルタイム空間補間可視化システム**
  - 無線センサネットワークから加速度データストリームを生成
    - クライアント端末へ直接伝送
  - データの空間補間を用いてヒートマップ状に可視化
    - 振動状況の状態を感覚的に評価
  - ウェブシステムとして構築
    - ウェブブラウザからシステムを利用でき専用環境が不要

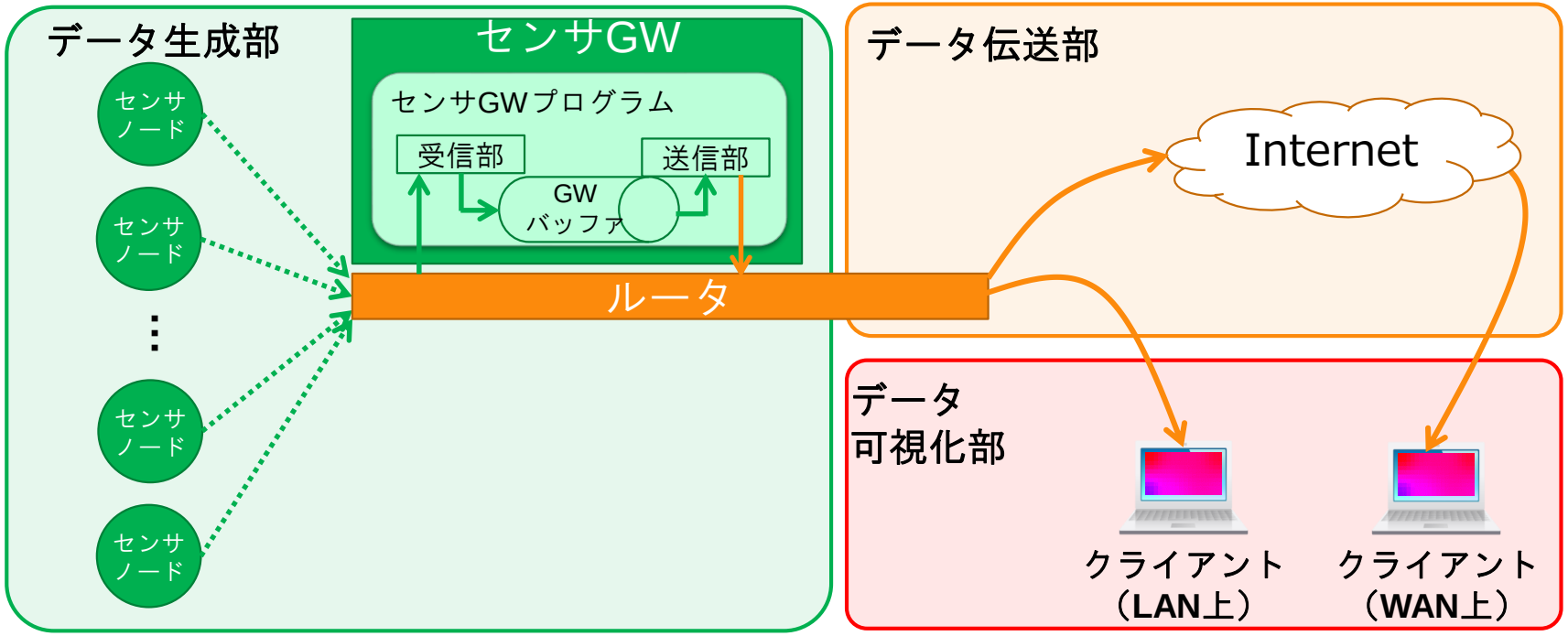


システムアーキテクチャと動作

データ生成
- 無線センサノードにてデータサンプリング
- センサGWにてセンサデータを集約

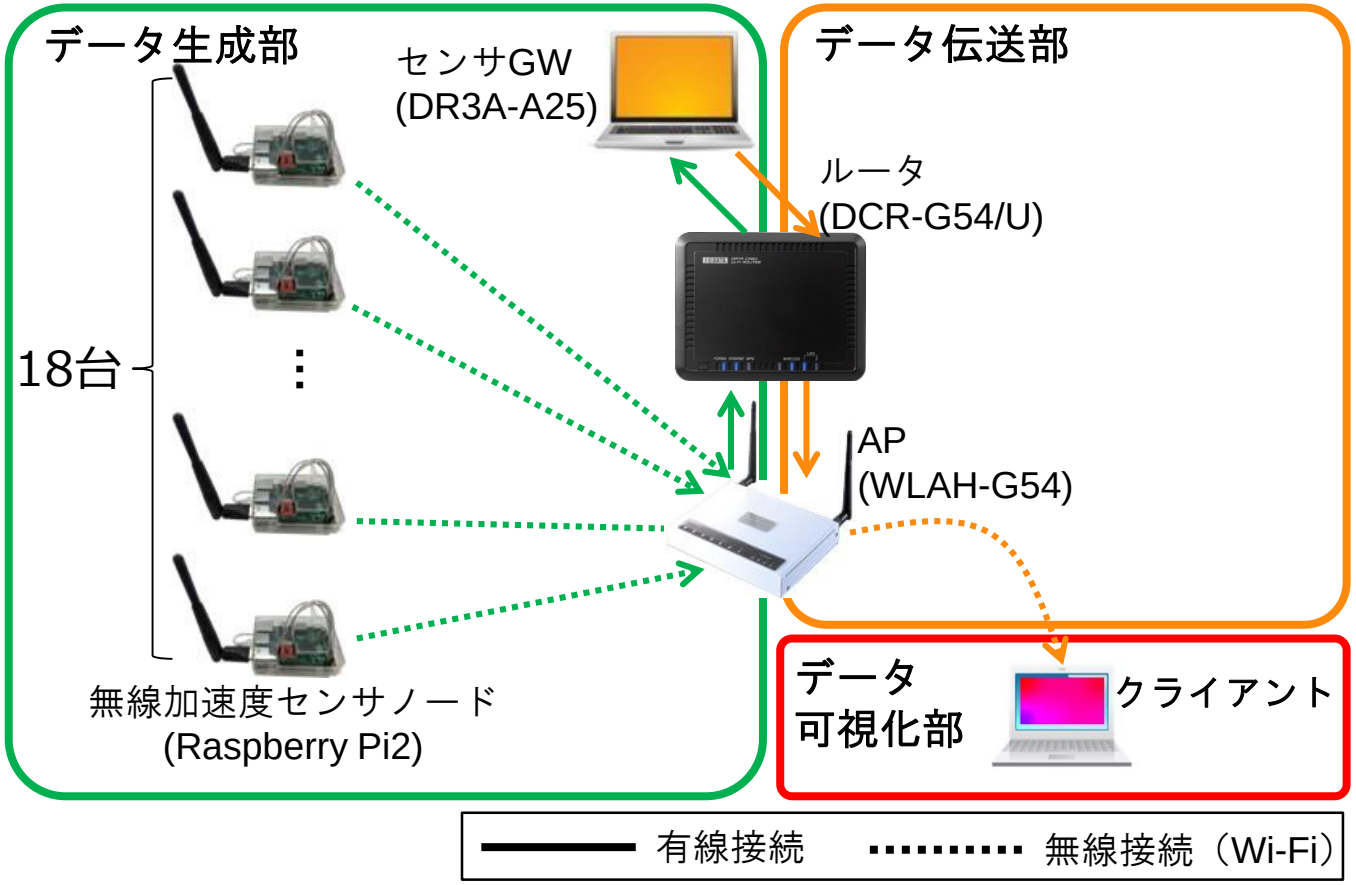
データ伝送
- 集約センサデータをセンサGWからクライアント端末へ伝送
- 伝送には軽量なWebSocket通信を採用

データ可視化
- クライアント端末にて集約センサデータを受信
- ウェブブラウザにてデータの空間補間と描画を行い可視化



プロトタイプシステム

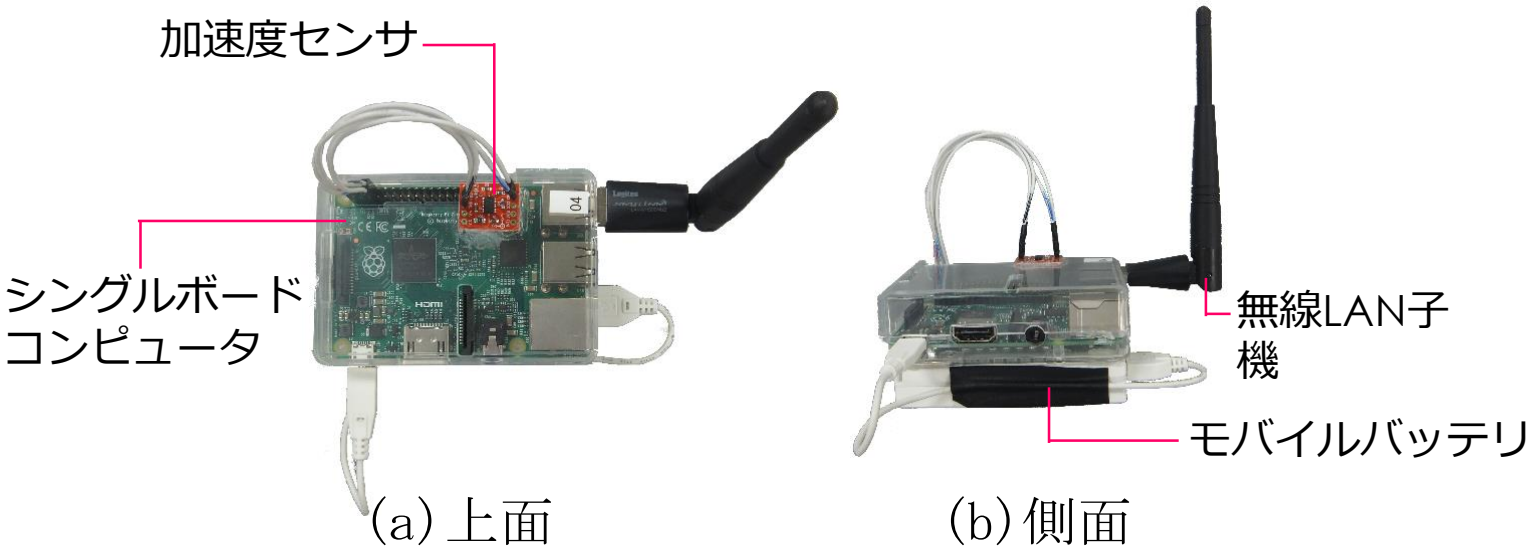
- マルチストリーミングセンサデータ向けリアルタイム空間補間可視化システム



無線加速度センサノード
構築手順

構築方法:センサノード[1/7]

- 外観

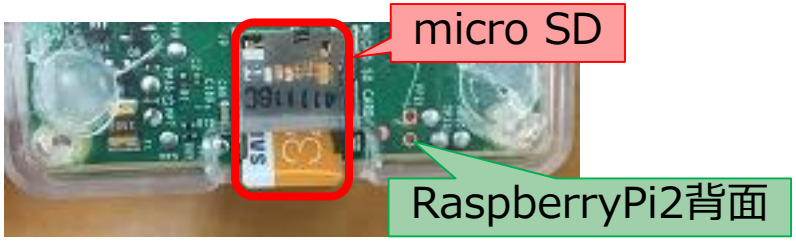


- 構成要素

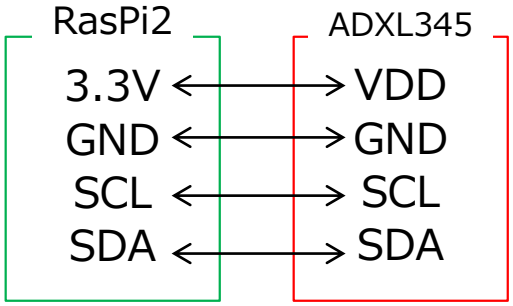
要素	品名・型番
シングルボードコンピュータ	Raspberry Pi2(ケース付)
micro SD カード	MB-MP32DA(32GB)
無線LAN子機	LAN-WH300NU2
加速度センサ	ADXL345
モバイルバッテリ	CHE-061

構築方法:センサノード[2/7]

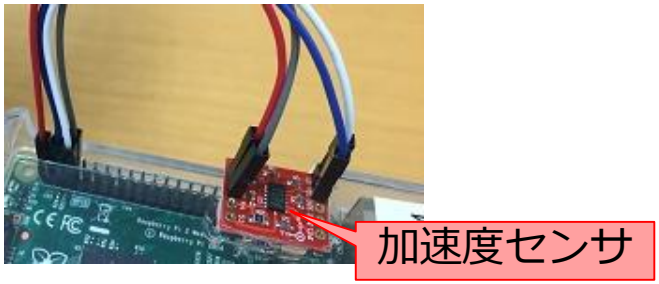
- RaspbeeyPi2へ各種機器を接続
  - OSインストール済みmicro SDカードの接続
    - OS：raspbian-jessie-lite
    - RaspberryPi2背面に挿入



  - 加速度センサ（ADXL345)のI2C接続
    - RaspberryPi2のGPIOとセンサの端子を接続



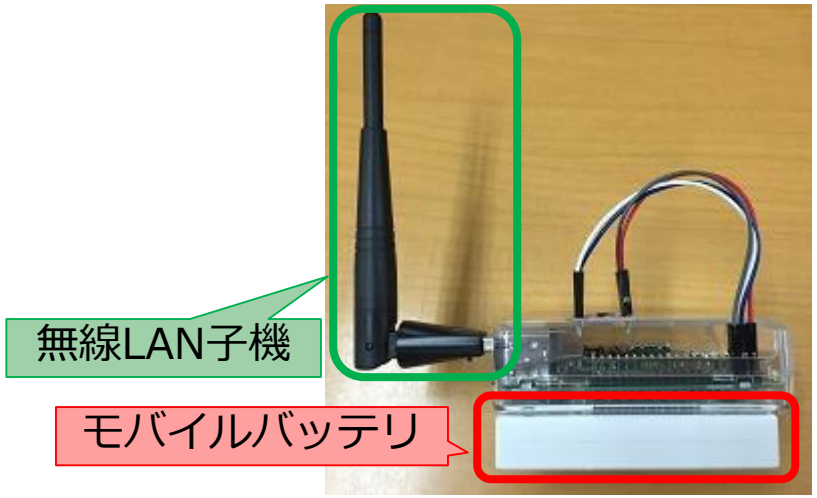
(a) GPIOの配線

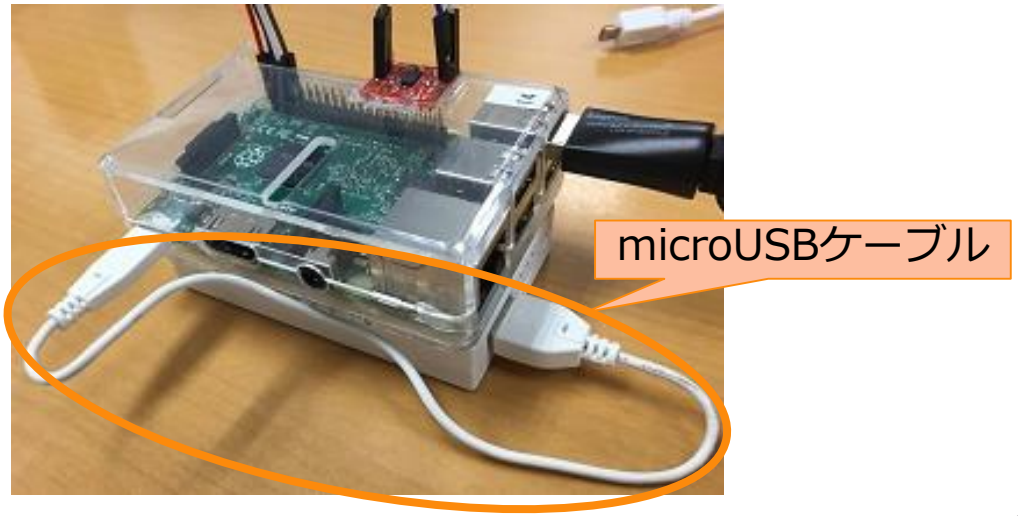


(b) Raspberry Pi2への接続

構築方法:センサノード[3/7]

- RaspbeeyPi2へ各種機器を接続
  - 無線LAN子機（LAN-WH300NU2）の接続
    - RaspberryPi2のUSB端子に接続
  - モバイルバッテリ（CHE-061）の装着
    - RaspberryPi2の背面にホットボンドを用いて装着
    - micorUSBケーブルを接続しモバイルバッテリから電源供給





構築方法:センサノード[4/7]

- RaspberryPi2の起動・ログイン
  - RaspberryPi2にディスプレイとUSBキーボードを接続
    - ディスプレイ接続にはHDMIケーブルを用いる
    - HDMIは起動前に接続しないと画面出力されないので注意
  - RaspberryPi2に電源を供給
    - 画面が表示され，起動が完了するとログイン状態となる
  - ログイン
    - User:pi
    - Pass:raspberry

構築方法:センサノード[5/7]

- 無線設定
  - IPアドレスの設定
    - /etc/network/interfaces を編集

interfaces(無線LANのIPに関係する部分のみ)

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.XX.YYY
    netmask 255.255.255.0
    network 192.168.XX.0
    broadcast 192.168.XX.255
    gateway 192.168.XX.1
    wireless-essid SSID {SSID}
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

address 192.168.XX.YYY ➡ 任意のIPアドレスに変更

  - 無線アクセスポイントの設定
    - 下記のコマンドを実行し，設定ファイルを更新

```
# wpa_passphrase {SSID} {PASS} >> /etc/wpa_supplicant/wpa_supplicant.conf
```

{SSID}:接続先のSSID
{PASS}:接続時のパスワード

構築方法:センサノード[6/7]

- 無線設定
  - ネットワークを再起動

```
# /etc/init.d/networking restart
```

  - 設定内容の確認
    - ifconfigとiwconfigを実行し，IPアドレスとSSIDを確認

```
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr {MAC ADDRESS}
          inet addr:192.168.XX.YYY  Bcast:192.168.XX.255  Mask:255.255.255.0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          （中略）

# iwconfig wlan0
wlan0     IEEE 802.11bg  ESSID:{SSID}  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.462 GHz  Access Point: {MAC ADDRESS}
          （中略）
```

IPアドレス

接続先SSID

構築方法:センサノード[7/7]

- センサノードプログラムの準備
  - プログラム確認

プログラムを配置したディレクトリに移動

```
# cd {センサノードプログラムディレクトリ}
# ls
Makefile           nodemain.c          setting.h
make_send_data.h   README.txt          udp_setup.c
mygettimeofday.c   sensor_io_handle.c  udp_setup.h
mygettimeofday.h   sensor_io_handle.h
```

  - プログラム設定
    - README.mdを参考にsetting.hを編集
  - コンパイル

```
# make
(中略)
sensor_io_handle.o nodemain.o mygettimeofday.o udp_setup.o -lpthread -lm
# ls nodemain
nodemain
```

実行ファイルの生成を確認

センサゲートウェイ
構築方法

構築方法:センサゲートウェイ[1/2]

- センサゲートウェイ端末
  - プロトタイプではDR3A-A25（ONKYO製）で動作確認

項目	型番・仕様等
型番（製造元）	DR3A-A25(ONKYO)
OS	Ubuntu 14.04.4 LTS 64bit
プロセッサ	Intel Core i5-2430M
クロック周波数	2.40GHz
コア数（論理コア数）	2 (4)
メインメモリ	8GB

- ゲートウェイプログラムの準備
  - gatewayフォルダ内のプログラムを端末内の任意の場所へコピー
  - setting.hを編集し，設定
    - 設定方法はREADME.mdを参照
  - コンパイル
    - makeコマンドを実行すると，実行ファイル「gatewaymain」が生成

構築方法:センサゲートウェイ [2/2]

- センサゲートウェイプログラム一覧

gateway
- client_manager.cpp
- client_manager.h
- create_thread_for_client.cpp
- create_thread_for_client.h
- datastructure.h
- gatewaymain.cpp
- get_max_sock.cpp
- get_max_sock.h
- Makefile
- mygettimeofday.cpp
- mygettimeofday.h
- my_json_encode.cpp
- my_json_encode.h
- recv_frame_from_client.cpp
- recv_frame_from_client.h
- recv_frame_from_client_thread.cpp
- recv_frame_from_client_thread.h
- ringbuffer.cpp
- ringbuffer.h
- send_data_thread.cpp
- send_data_thread.h
- send_to_client_thread.cpp
- send_to_client_thread.h
- setting.h
- set_fds.cpp
- set_fds.h
- tcp_listen.cpp
- tcp_listen.h
- udp_listen.cpp
- udp_listen.h
- WS_accept_thread.cpp
- WS_accept_thread.h
- ws_handshake.cpp
- ws_handshake.h
- cwebsocket
  - build_arduino_library.sh
  - client.html
  - LICENSE
  - README.md
  - lib
    - aw-base64.h
    - aw-sha1.h
    - websocket.c
    - websocket.h

クライアント環境
構築方法

構築方法:クライアント[1/2]

- クライアント端末
  - プロトタイプではR732/E26GB（TOSHIBA）で動作確認

項目	型番・仕様等
型番（製造元）	R732/E26GB (TOSHIBA)
OS	Windows7 Professional 64bit
プロセッサ	Intel Core i5-3230M
クロック周波数	2.60GHz
コア数（論理コア数）	2 (4)
メインメモリ	8GB

  - ウェブブラウザを搭載する汎用端末であれば使用可能

- クライアントプログラムの準備
  - clientフォルダ内のプログラムを任意のウェブサーバ上にコピー
    - プロトタイプではセンサGW内のウェブサーバを構築
  - README.mdを参考にプログラムを設定

構築方法:クライアント[2/2]

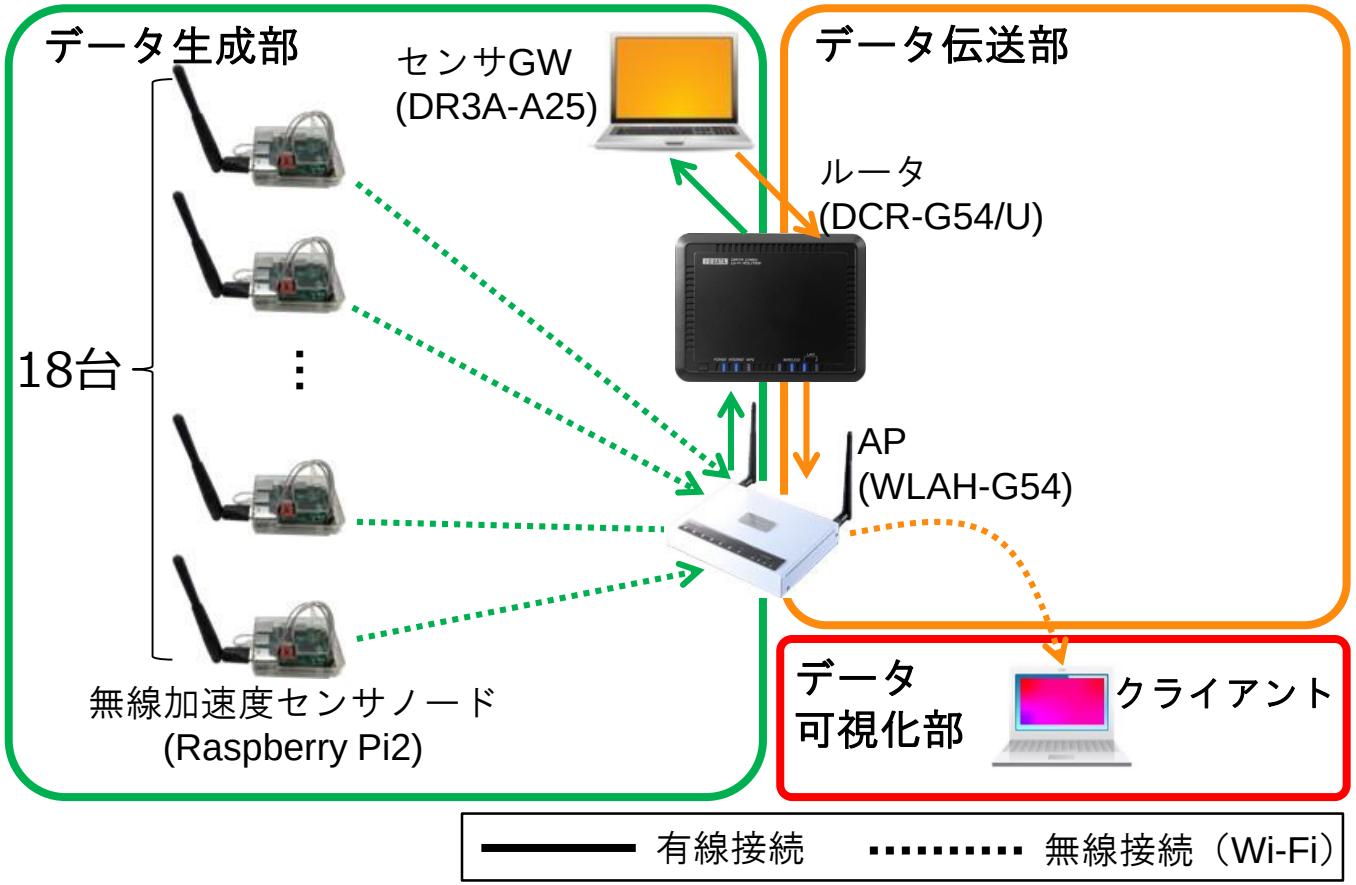
- クライアントプログラム一覧

```
client
├ client.html
├ README.txt
│
├─css
│  client.css
│
└─js
   jquery-2.2.4.min.js
   main.js
   webgl.js
   worker.js
```

システム起動手順

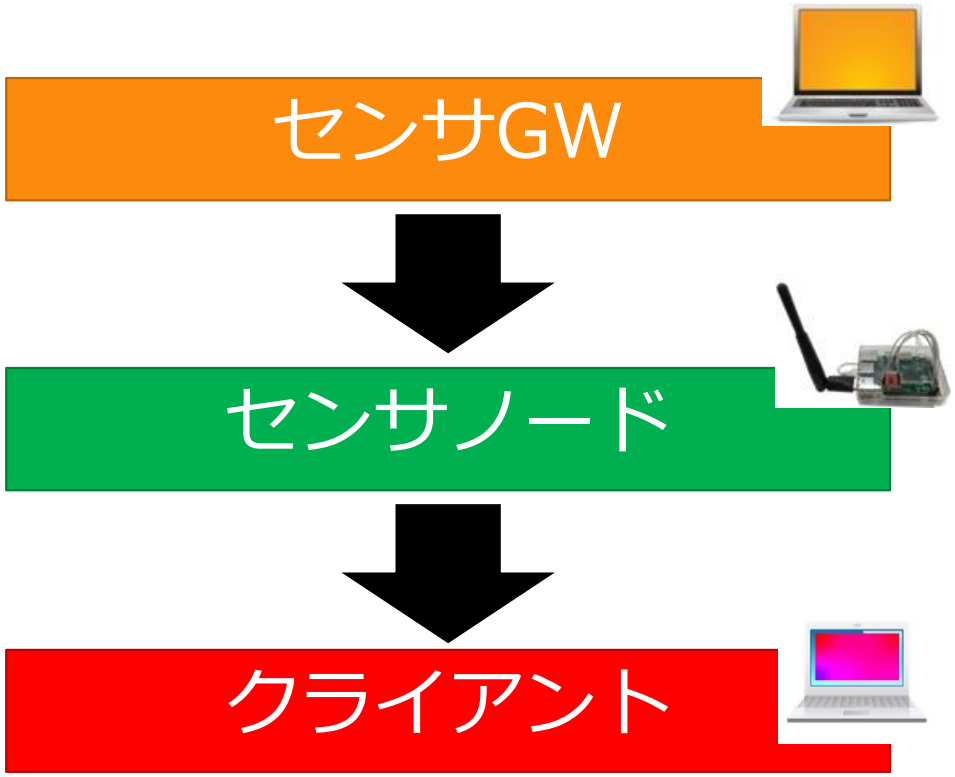
起動手順 [1/2]

- 下記のアーキテクチャのように各機器を接続



起動手順 [2/2]

- 起動順序



- 各プログラムの起動手順はREADME.txtを参照

動作例

動作例

- 室内環境における動作検証
  - 実験場所：研究室内 (縦90cm，横180cm)
  - 使用ノード数：18