Vol.2016-MBL-80 No.1
Vol.2016-CDS-17 No.1
2016/8/24

Bluetooth Low Energy を利用したセンサデバイスからのデータ収集方式

槌屋洋亮[†1]　石川憲洋[†2]

概要：近年，センサーデバイスにより収集されるデータの利活用へ注目が集まっており，センサーデータ収集方式についても様々な提案がなされている．しかしながら，その多くはスマートフォンに搭載されたセンサーからのデータ収集方式であり，その先に接続するセンサーデバイスについては考慮されない．また，スマートフォンと接続してデータを送信するセンサーデバイスも出回ってはいるが，製品ごとに独自に用意されるデータ蓄積方法の煩雑さや，搭載センサーの制約があり，利用者にとっては柔軟性に欠ける．この課題に対して，本論文では Bluetooth Low Energy (BLE)とスマートフォンを利用したセンサーデータ収集方式について提案する．具体的には，様々なセンサーに対応可能で BLE モジュールを搭載した Arduino センサーデバイスの試作，およびそのセンサーデバイスから BLE を介してデータを取得し，サーバへ蓄積するためのスマートフォンアプリの試作について述べる．本方式により，ユーザの目的に応じた柔軟でスケーラブルなセンサーデータ収集基盤の構築が可能となる．

キーワード：センサーデータ，センサーデバイス, Arduino，Bluetooth Low Energy，Mobile Crowd Sensing

1. はじめに

近年, 情報技術の分野ではセンサーデバイスにより収集, 蓄積されるデータの利活用への注目が集まっている．気温や湿度，屋内外の明るさ，場所の混雑度，騒音などを測定できるセンサーデバイスをインターネットへ接続させ，センサーデータを秒単位でクラウドサーバへリアルタイムに送信し, それらのデータを適切な形で可視化（Visualization）することで，日々刻々と変化する我々の社会や人間行動，生活環境について詳細に把握することが可能となる．それらのセンサーデバイスから取得されるデータを地域活性化や環境問題の解決などへ応用し，いわゆるスマートシティの形成のために利活用することが期待される．

センサーデータの利活用について注目が集まる中，センサーデータの収集方式についても様々な提案が行われている．中でもモバイルクラウドセンシング（Mobile Crowd Sensing）[1]は，スマートフォンをはじめとしたモバイルデバイスをクラウドサーバに接続し，センサーデータをリアルタイムで蓄積する，センサーデータ収集方式の新しいパラダイムとして注目されている．既に世界人口の 4 分の 1 以上が利用しているスマートフォンからのセンサーデータをクラウドサーバへリアルタイムに送信することで，気象データのモニタリングへの応用[2]や，旅行プランのレコメンデーションシステムへの応用[3]などが可能となっている．

しかしながら，モバイルクラウドセンシングをはじめとした既存の多くのセンサーデータ収集方式は，スマートフォンに搭載されたセンサーの利用を前提としたデータ収集方式である．センシングデバイスとしてスマートフォンの利用を前提としており，その先に接続するセンサデバイスについては考慮されない．また，スマートフォンと接続してデータを収集するセンサーデバイスもいくつか市販されているが，その多くは，製品ごとに独自に用意されているデータ蓄積方法が煩雑であり，デバイスの内部仕様やデータ送受信の方式などがオープンになっていないこと，搭載センサーの制約などから，利用者にとっては柔軟性に欠けている．

この課題に対して，本論文では，Arduino とスマートフォンを利用した Bluetooth Low Energy（BLE）によるセンサーデータ収集方式について提案する．具体的には，様々なセンサーに対応可能で BLE モジュールを搭載した Arduino センサーデバイスの試作，およびそのセンサーデバイスから BLE を介してデータを取得し, クラウドサーバへ送信するためのスマートフォンアプリの試作について述べる．本方式により，ユーザの目的に応じた柔軟でスケーラブルなセンサーデータ収集基盤の構築が可能となる．

2. 提案方式の概要

本論文で提案するセンサーデータ収集基盤のアーキテクチャを図 1 に示す．我々が提案する方式は，Arduino を利用してセンサーデバイスをユーザ自身で自作し，BLE を介してスマートフォンへ送信し，スマートフォンからインターネットを介してクラウドサーバにリアルタイムで送信，蓄積する．

本方式の特徴は次の 2 点である．第 1 に，センサーデバイスに Arduino をはじめとしたオープンソースハードウェアを採用することで，ユーザ自身の目的に応じた柔軟なセンサーデータの取得が可能となる点である．第 2 に，BLE を介してセンサーデバイスとスマートフォンの間でセンサーデータのやりとりを行うことで，センサーデバイスを直

†1 青山学院大学附置情報メディアセンター
Institute of Information and Media, Aoyama Gakuin University
†2 駒澤大学グローバル・メディアスタディーズ学部
Faculty of Global Media Studies, Komazawa Unviersity

Vol.2016-MBL-80 No.1
Vol.2016-CDS-17 No.1
2016/8/24

接ネットワークに繋げなくともスケーラブルなセンサーデータ収集基盤の構築が可能となる点である．

本論文では，Arduino センサーデバイスの作成手法，さらに制作した Arduino センサーデバイスから BLE を介してデータを取得しクラウドサーバ送信するスマートフォンアプリ，センサーデータを蓄積するためのクラウドサーバ構成方法について述べる．

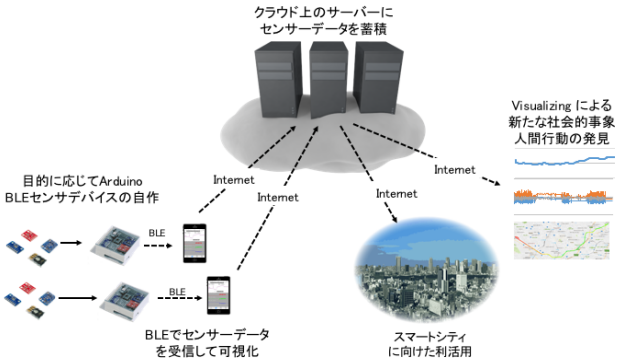


図 1 提案方式のアーキテクチャ

3. Arduino センサーデバイス作成手法

先ず，Arduino センサーデバイスの作成手法について述べる．Arduino は，オープンソースのデバイスプロトタイピングのためのプラットフォームである．一般的にデバイスの作成にはハードウェアの深い知識等が求められるが，Arduino はそのような専門知識がなくとも簡単に様々なデバイスを作ることを目的としている．他にデバイス作成で利用されるものとして Raspberry Pi が挙げられる．しかしながら，Raspberry Pi は電源供給が USB であり，動作させるには AC 電源アダプタを使うか，別途モバイルバッテリーが必要となる．また，Raspberry Pi の場合は，Raspbian などのオペレーティングシステムを搭載しているので，電源を落とす際にその都度シャットダウン処理を行わなければならない．Raspberry Pi と比較して，Arduino であれば乾電池やリチウムポリマー電池などで動作させることが可能であり，また内部の動作は全て Arduino IDE 上のプログラムによって制御することができる．また，Arduino にセンサーを接続する方法や，センサーを制御する Arduino プログラムも，多くはインターネット上にオープンソースとして公開されているので，移植やカスタマイズも容易に行える利点がある．

3.1 ハードウェア構成

Arduino はオープンソースのハードウェアなので，多くのメーカーが Arduino 互換機を製造している．用途に合わせて最適な Arduino を選択すればよいが，センサーデバイスを制作する場合は，搭載するセンサーの動作電圧と電源を考慮する必要がある．センサーデバイスは，屋内外への設置から，持ち運びながらの利用など様々な場面での利用が想定されるため，小型かつ消費電力が少なく，電池などを接続できる Arduino を選択した．該当する Arduino 互換機はいくつかあるが，例えば，Spark Fun 社製の Arduino Pro も利用可能である．

搭載するセンサーは，デバイスの作成者が「どのようなセンターデータを取得したいか」により決定すればよいが，搭載するセンサーの数や種類については，デバイス筐体の大きさやマイコンボードが動作する電圧を考慮する必要がある．使用するマイコンボードと同じ電圧で動作するセンサーモジュールを選ぶことが基本となる．その理由は，単にレベルコンバータなどを使わずに，センサーを接続する回路を簡略にするためである．

表 1 に，Arduino に搭載するセンサーの一例を挙げる．Arduino に搭載できるセンサーの種類は，基本的に動作電圧で分けることができる．3.3V で動作するセンサーには，気温センサー，湿度センサー，気圧センサー，照度センサー，騒音センサー，モーションセンサー，カラーセンサー，UV Index センサー，GPS などがある．一方，動作に 5V が必要なセンサーには，ダストセンサー，超音波距離センサー，ガスセンサー，水分センサーなどがある．3.3V で動作するセンサー類には持ち運びながら測定すると効果的なセンサーが多く，一方で 5V が必要なセンサーには，特定の位置に固定し，安定的な電源供給のもとで測定すると効果的なセンサーが多い．これらのセンサーを全て搭載することはできないので，この中から必要なセンサーを選択して組み合わせる必要がある．手順としては，先ず，センサーモジュールを選択し，ブレッドボードを使い回路を設計し，問題なく動作することを検証した後，各センサーモジュールをユニバーサル基板へはんだ付けし，Arduino へ接続する．

表 1 Arduino へ搭載するセンサーの例

電圧	種類	用途	モジュール例
3.3V	気温/気圧/湿度	気温・気圧・湿度の計測	Adafruit BME280
	照度	明るさの計測	Sparkfun TSL2561
	騒音	周囲の騒音を計測	Sparkfun INMP401
	モーション	3 軸の加速度・回転角度・地磁気を使い動きを測定	ストロベリーリナックス MPU9520
	カラー	物体の色や周辺環境の色温度を検出する	ストロベリーリナックス VEML6040
	UV	紫外線強度を計測する	Adafruit SI1145
	GPS	GPS データを受信する	GANMORE GMS6-CR6
5V	ダスト	周辺環境の空気の汚染具合（PM2.5 など）を計測	SHARP GP2Y1010AU0F
	距離	物体との距離を計測	Seeedstudio SEN136B5B
	ガス	水素やメタンガスなどを検知する．	Sparkfun Gas Sensor –MQ8 など
	水分	土壌中の水分を計測する	Sparkfun Soil Moisture Sensor

標準的な Arduino には無線通信機能が備え付けられていないため，デバイス間でデータの送受信を行う場合には，別途，通信モジュールを取り付ける必要がある．Arduino で利用出来る無線通信モジュールには Wi-Fi, Zigbee, 3G/4G, Bluetooth, BLE などがある．Wi-Fi は，転送速度が大きい利点はあるが，消費電力が大きい．ZigBee は Wi-Fi より消費電力が少なく，センサーネットワークを形成する際にはよく採用される．しかしながら，ZigBee の場合は，多くのスマートフォンでサポートされていないため，接続することができない．

これに対して，Bluetooth であればスマートフォンと直接接続し，データの送受信やスマートフォンを介してのセンサーデバイスの制御も可能となる．中でも，Bluetooth 4.0 で規定された BLE は，それまでの Bluetooth 3.0 よりも無線通信を，省電力かつ省コストで行うことができる．Arduino で利用出来る BLE モジュールはいくつかあるが，提案方式では，Arduino からのセンサーデータをスマートフォンへ送信するだけでよいので，例えば，BLE でシリアル通信を行うことができる浅草ギ研の BLESerial2 は，Arduino の標準ライブラリである Software Serial を利用して，簡単に BLE によるデータの送信を実装できる．

3.2 制御プログラム

ハードウェアを選定し，組み立てが完了したら Arduino の制御プログラムを開発する．Arduino の統合開発環境は Processing ベースで設計されており，記述言語は C 言語風の構文であるがソフトウェア開発に不慣れであっても容易にプログラミングが可能である．また，表 1 に示したセンサーモジュールからのデータを取得するためのサンプルコードやライブラリは，全てオープンソースとしてインターネット上から入手できる．Arduino 側では，各センサーから取得した値を特に加工せず，生データのまま BLE モジュールを介してスマートフォン側へ送信するプログラムを開発する．BLE モジュールとスマートフォンとのペアリングが開始されたら，各センサーデータを一定の秒間隔ごとにスマートフォンへ送信する．このとき，センサーのデータをどのようなフォーマットで送信するかという問題がある．いくつか有力なフォーマットはあるが，本試作では単純に，センサーデータを順番にカンマ区切り（CSV 形式）にして送信することとした．すべてのセンサーデータを送信した後で改行コードを送信する．これを一定の秒間隔で繰り返し，ペアリングが終了したらセンターデータの送信を終了する．

Arduino から BLE を介してセンサーデータを何秒間隔で送信するかは重要である．BLE は省電力とはいえ，データ送信の間隔が短ければ，通信のために消費する電力が増える．また，搭載するセンサーも多ければ多いほど消費電力が増える．消費電力が増える分，Arduino と接続する電池の持続時間が短くなる．図 2 は，搭載センサーの数とデータの送信間隔ごとに Arduino で組み立てたセンサーデバイスの電池使用時間を比較した評価結果である．サンプルとして，Sparkfun 社製のリチウムイオンポリマー電池 400mAh を使用し，照度（Sparkfun TSL2561），UV（Adafruit SI1145），モーション（ストロベリーリナックス MPU9520），気温・気圧・湿度センサー（Adafruit BME280）を搭載した Arduino センサーデバイス（フルセンサーとする）と，気温・気圧・湿度センサーのみを搭載したセンサーデバイスを用意し，スマートフォンへそれぞれ 1 秒間隔で送信した場合と 10 秒間隔で送信した場合の電池使用時間を測定した．フルセンサーを 1 秒間隔でデータを送信した場合は約 26 時間程度しか持たないが，BME280 のみを搭載し 10 秒間隔でデータを送信した場合は約 48 時間継続して利用可能である．充電容量の多い電池を利用すれば，更に利用時間を延ばすことが可能となる．

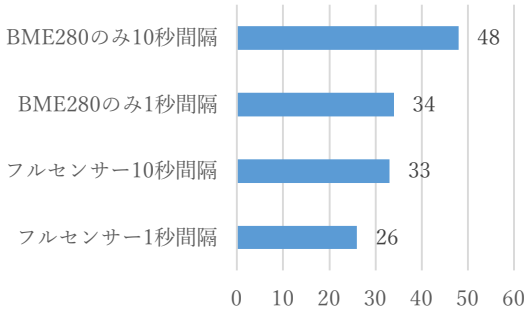


図 2　Arduino センサーデバイスの電池使用時間

3.3 筐体の作成

最後に，これまで試作した Arduino センサーデバイスを格納する筐体を用意する．最終的なデバイスの形状・大きさに合わせて適切な箱などを用意すればよいが，３D プリンターやレーザーカッターなど，いわゆるパーソナルファブリケーション[4]のための機材も普及しており，これらの機材を利用すれば，適切な用途にあわせた筐体を独自に制作することも可能である．３D プリンターを利用する場合は，３D CAD ソフトウェアを使って筐体の３D モデルを制作し，出力する．レーザーカッターを利用する場合は，Vector 描画ソフトウェアを利用して立体の平面図を作成，切削し，組み立てる．

いずれの方法でも筐体の制作は可能だが，３D プリンターは素材がプラスチック樹脂に限られるのに対して，レーザーカッターはアクリル板，ベニヤ板，プラスチックなどが利用できるため，素材選択の上で柔軟性が高い．一方，３D CAD ソフトウェアによるモデル作成は，2D での平面図の制作と比べると難易度は高い．レーザーカッターは高価ではあるが，最近では，機材の利用環境を提供している会員制工房などでも利用することができる．これらの会員制工房は，機材の使用方法も学べるので有効である．

3.4 センサーデバイスの試作

以上を踏まえ，本論文では，以下の通り Arduino センサーデバイスの試作を行った．Arduino 基盤のボードには Spark Fun 社製の Arduino Pro（ATMega328 3.3V/8MHz）を選択し，BLE センサーモジュールには浅草ギ研の BLESerial2 を使用した．搭載したセンサーモジュールは表 2 に示す．照度，気温・気圧・湿度，モーション（加速度・回転角度・地磁気），UV Index に加えて，照度センサー，カラーセンサー，騒音センサーのうち 1 つを追加搭載したデバイスを用意し，特定の場所に設置した場合も，移動中の場合でも柔軟に利用出来るようにした．筐体は，２D 平面図を作成し，レーザーカッターでアクリル板を平面図の通りに切断し組み立てることとした．レーザーカッターでアクリル板を加工して制作した筐体に，各種センサーを格納して試作したセンサーデバイスを図 3 に示す．

表 2 センサーデバイスに搭載するセンサーモジュール

種類	センサーモジュール
気温/湿度/気圧	Adafruit BME280
照度	Sparkfun TSL2561
紫外線強度	Adafruit SI1145
騒音	Sparkfun INMP401
9 軸モーション	ストリベリーリナックス MPU9520
RGB カラー	ストロベリーリナックス VEML6040

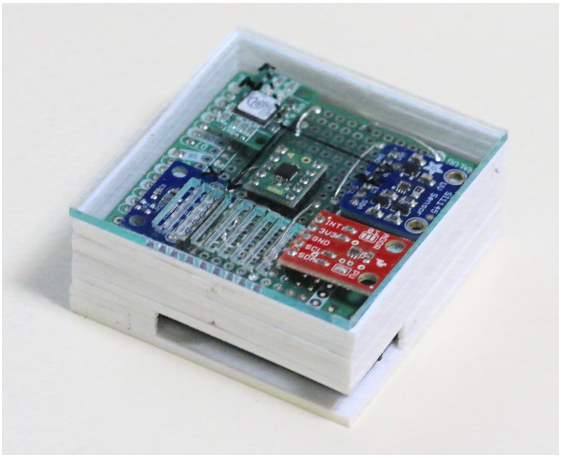


図 3　試作したセンサーデバイス

4. スマートフォンアプリの作成

前述の Arduino センサーデバイスから BLE を利用して送信されるセンサーデータをスマートフォン側で受信するアプリの実装方法について述べる．

4.1 基本的な機能

図 4 に，アプリの基本的な機能を示す．アプリの基本的な機能は，１）BLE を介したセンサーデータの受信，２）受信したセンサーデータのリアルタイムでの可視化，３）センサーデータを蓄積するクラウドサーバへのセンサーデータを送信，の３つである．

BLE を介してセンサーデバイスを受信する機能は，センサーデバイスとペアリングした後，デバイスからの各センサーデータを Arduino で設定された秒間隔で受信し，アプリ内で保持する．また，インターネット上のクラウドサーバにセンサーデータを送信する機能は，Wi-Fi ないしは 3G/4G などの無線データ回線を介してセンサーデータを指定されたクラウドサーバへ HTTP を用いて送信する．この時，サーバ側にはアプリからのセンサーデータを受け取り蓄積するサーバサイドプログラムを用意する．センサーデバイスとアプリとの間の BLE 通信, およびアプリとサーバとの間の通信は，双方ともにアプリをバックエンドで実行可能とする．アプリをバックエンドで実行することで，アプリを起動し，センサーとの BLE 通信を開始すれば，自動的に収集したセンサーデータがサーバに蓄積される．

以下に, Swift 言語を用いて試作した iOS 上で動作するアプリの主な機能について述べる．

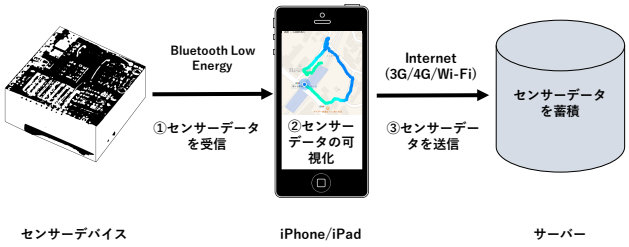


図 4　スマートフォンアプリの機能

4.2 BLE によるセンサーデータの受信機能

Arduino センサーデバイスからのデータは BLE で受信する．この時に課題となるのは受信するデータはどのセンサーデータかをどのように判別するかという点である．一般的なセンサーデバイスの場合，内部仕様や送信時のデータのフォーマット，制御するプログラムのソースコードなどが公開されていない場合が多く，センサーデバイスから送られてくるデータが何のセンサーデータかを判別することは困難である

これに対して，我々は Arduino によりセンサーデバイスを試作したため, センサーデバイスから BLE でデータを送信するフォーマットや送信間隔を自由に設定することができる．本試作では，3.2 にて述べた通り，Arduino 側からはセンサーのデータを順番にカンマ区切りにして送信され，すべてのセンサーデータを送信したところで改行コードが送信され，これを一定の秒間隔で繰り返される．従って，スマートフォンのアプリ側では，この送信フォーマットと

送信間隔に対応してデータを受信できるように実装すればよい．具体的には，アプリ側ではデータを Notify で受信し続け，受信したデータを文字列型の変数に格納し，改行コードを受信したら文字列を分割し，それぞれのデータを適切な型へ変換して配列に格納する．Arduino により BLE センサーデバイスを作成することの利点は，BLE によるセンサーデバイスとスマートフォンの間データの送受信のフォーマットや間隔などを，自由に設計することが可能となることにある．アプリ側は，iOS の場合には，Swift 言語より Core Bluetooth ライブラリを利用すれば BLE によるデータ受信プログラムを容易に実装することができる．

4.3 データ可視化機能

受信したセンサーデータをリアルタイムで可視化する機能は重要である．ユーザは，自分自身の行動と，現在いる（あるいは数分前に辿ってきた）場所の周辺環境の情報を，センサーデータを通じて克明に，リアルタイムに把握することが可能となる．但し，データ可視化には新鮮な着眼点や伝達力，効率性，効果的な表示方式などが必要であり，１つのセンサーデータのみを単純なグラフで表示するだけでは不十分である．一方，スマートフォンの画面は小さいため，インターフェイス上の制約も大きい．センサーデバイスからのデータをリアルタイムに可視化する画面として，折れ線グラフ形式で時系列の推移を表示する画面と，地図上に表示する画面を実装した．

センサーデータを地図上に表示する画面と，時系列のグラフ上に表示する画面を図５に示す．地図上に表示する画面では，センサーデータを取得した位置にマーカーを置き，センサーの値を HSV 色空間の値へ変換し，マーカーの色として設定する．これにより，ユーザは自分が歩いてきた場所の周辺の環境情報を直観的に把握することができる．位置情報には，センサーデバイスから送られてくる GPS データを利用するが，デバイスに GPS を搭載していない場合はスマートフォンに搭載されている GPS を利用する．センサーデータを時系列のグラフ形式で表示する画面では，過去数分間の気温や照度などの周辺環境についての時系列な変化についての折れ線グラフと，モーションセンサーのデータ（加速度，角速度，地磁気）の時系列な変化についての折れ線グラフの両方を同時に表示する．これにより，周辺の環境情報と，ユーザ自身の行動の時系列な関係を把握できる．

4.4 サーバへのデータ送信機能

BLE を介して Arduino センサーデバイスから受信したデータをクラウドサーバへ送信する際に重要な点は，そのセンサーデータがどのデバイスによって何時・何処で取得されたデータであるかを判別するためのデータを付して送信する一方で，データを送信する個人が特定されないようにすることである．単純にセンサーデータにデバイスの ID と日付と時刻，GPS 情報を付して送信すればよいが，この場合，デバイス ID とスマートフォンの所有者である個人を紐付けることが可能であるが，個人情報保護の観点から課題が残る．この課題を解決するためには，例えば，図５の経路図，時間的な連続性を保持するのであれば，「誰のデバイスから送信されたか」ではなく，「いつ・どこで使っていたデバイスから送信されたか」が判別できるための識別子が生成できればよい．例えば，Swift 言語であれば，アプリをセンサーと BLE でペアリングする度に，NSUUID 関数を用いて識別子を生成し，それを ID としてセンサーデータと時刻情報・位置情報に付して送信することが可能である．

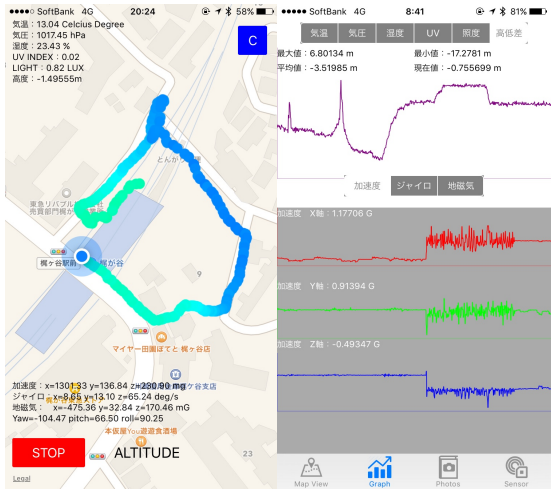


図 5　センサーデータ可視化画面

上述の ID，日時，GPS，センサーデータをクラウドサーバへ送信する．送信する際は，あらかじめ次章で述べるように，サーバサイドのプログラムを用意しておき，そのプログラムに対して HTTP POST メソッドにて送信する．この時，データを適切な送信間隔を設定して送信する．例えば，センサーデバイスから 1 秒ごとにデータを受信している場合，同時に 1 秒ごとにサーバへ送信するとインターネット回線に負荷をかける可能性がある．その場合には，間隔を開けてサーバへデータを送信し，その間はアプリ側でデータを保持し続ければよい．

5. クラウドサーバ

前述のスマートフォンアプリから送信されるセンサーデータを蓄積する，クラウドサーバの機能について述べる．先ず，クラウドサーバの仕様であるが，運用するデバイスの台数や蓄積するデータの規模によって，クラウドサーバのハードウェアをスケーラブルに構成する必要がある．本論文で述べた実験レベルであれば，レンタルサーバの利用

で十分であるが，データを蓄積するための容量には注意する必要がある．最近はテラバイト単位の外部記憶装置も安価に購入できるようになってきたが，センサーデータを蓄積するクラウドサーバを構築する際は，用途や蓄積するセンサーデータの規模を考慮しつつ，なるべく容量の大きい外部記憶装置を選択する必要がある．

上述のサーバに，センサーデータを蓄積し，必要に応じて検索して表示するための環境を設定する．図 6 にセンサーデータを蓄積するクラウドサーバの構成を示す．スマートフォンからのセンサーデータ受信を HTTP で行うために Apache などの Web サーバーソフトを使用し，受信したセンサーデータを蓄積ために，MySQL などのデータベース管理システムを使用する．また，具体的なプログラムを記述するために PHP をはじめとした Web アプリケーション言語を利用できるようにする．

サーバサイドでは，最低でも２つのプログラムを用意する．１つは，スマートフォンから送信されるセンサーデータを受信し，サーバ上のデータベースに格納するプログラムである．もう１つは，目的に応じてデータベースに問い合わせ，必要なセンサーデータを検索し，例えば，Web ブラウザ上に可視化するプログラムである．Web ブラウザに可視化する際は，各センサーデータには日時と位置情報が含まれているので，Google Map などを利用して地図上に可視化することが可能である．

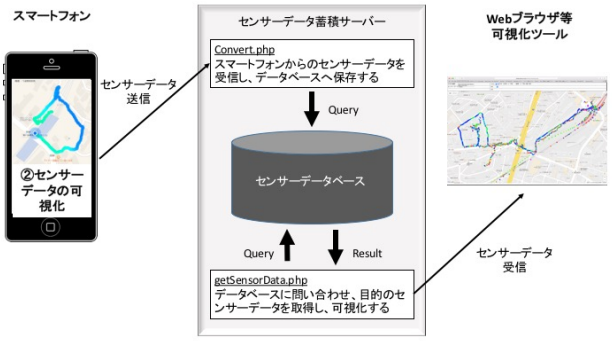


図 6　クラウドサーバの構成

6. 実証実験

以上，Arduino による BLE を利用したセンサーデバイスの試作，BLE を介してセンサーデータを取得し，リアルタイムで可視化するとともに，クラウドサーバへ送信するスマートフォンアプリの試作，センサーデータを蓄積して，利活用するクラウドサーバの試作について述べた．以下に，提案方式の有用性を検証するために行った，獲得できるセンサーデータの量の計測，蓄積されるセンサーデータの利活用に関する実証実験について述べる．

6.1 実験概要

実証実験は，BLE のデータ送信間隔の異なるサンプルを２つの Arduino センサーデバイスを試作して行った．１つは，1 秒間隔でデータを送信するデバイスで，これをデバイス A と呼ぶ．もう１つは，10 秒間隔でデータを送信するデバイスで，これをデバイス B と呼ぶ．どちらのデバイスにも，照度，気温・気圧・湿度，モーション（加速度・回転角度・地磁気），UV Index の各センサーを搭載した．なお、電源には両方とも図２の電池使用時間の計測に用いた Sparkfun 社製のリチウムイオンポリマー電池 400mAh を使用した。

スマートフォン側は，iOS 上で動作するアプリを Swift 言語を用いて試作した．今回の実験に利用したのは，iPhone 6s と第３世代 iPad mini である．iPhone 6s では BLE を介してデバイス A からのセンサーデータを 1 秒ごとに受信し，30 秒ごとにクラウドサーバへ送信する．iPad mini では，デバイス B からのセンサーデータを 10 秒ごとに受信し，約 1 分ごとにクラウドサーバへ送信する．

クラウドサーバは，LAMP 環境が提供されているレンタルサーバ上に構築した．センサーデータの蓄積は MySQL 上にて行い，スマートフォンから送信されてくるセンサーデータを MySQL へ保存し，目的のセンサーデータを検索し Google Map 上に可視化するサーバサイドのプログラムを PHP 言語にて実装した．

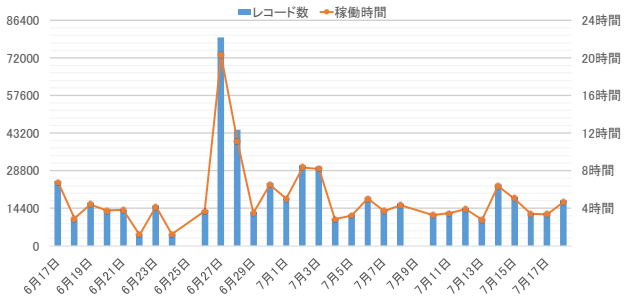
デバイス A については 2016 年 6 月 17 日から 7 月 18 日までの間のうち，6 月 25 日と 7 月 9 日以外の 30 日間の実証実験を行った．デバイス A の実証実験については移動中に常時稼働させることを基本とし，必要に応じてセンサーデバイスを特定の場所に 1 日もしくは半日（12 時間程度）固定し，センシングを行った．一方，デバイス B については，6/27，6/28，6/29，7/18，7/19，7/20，7/21 の 7 日間の実証実験を行った．デバイス B の実証実験については，特定の場所に，センサーデバイスを 1 日あるいは半日間，固定させて，常時稼働させることを基本とし，必要に応じて持ち運びながらのセンシングを行った．なお，実験に用いた Arduino センサーデバイスの構成と制御プログラム，スマートフォンアプリのソースコード，およびサーバサイドのプログラムは全て Git Hub にて公開しており，誰でも利用可能となっている[5].

6.2 データの蓄積量

図 7 に，デバイス A を利用した 30 日分のデータ蓄積量と稼働時間を，図 8 にデバイス B を利用した 7 日分のデータ蓄積量と稼働時間を示す．この２つのグラフにおいて，レコード数とは，サーバ上の MySQL に格納したセンサーデータの件数であり，１件あたり日時，緯度・経度，UV，気温・気圧・湿度，照度，3 軸加速度・回転角度・地磁気の各データが格納されている．デバイス A は 30 日で合計

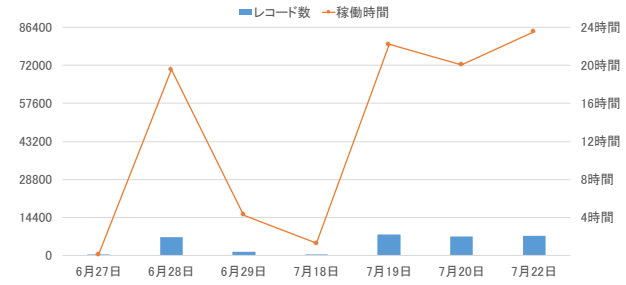
約 150 時間程度稼働させ，データベース上に約 56 万件を記録し，約 780 万（56 万×14 個）のセンサーデータを蓄積した．一方，デバイス B は 7 日間で合計約 91 時間稼働させ，データベース上に約 3 万件を記録し、合計で約 43 万（3 万件 x14 個）のセンサーデータを蓄積した．

当然ながら，BLE によるデータ送信間隔が異なるので蓄積されるデータ量もデバイス A の方が圧倒的に多い．一方で，図 2 にて示した通りデバイス B はデバイス A よりも長時間連続して稼働させることが可能で，実際，デバイス B の 1 日平均稼働時間は，デバイス A よりも長い結果となり、7/18 から 7/20 までの間はほぼ常時稼働させることができた。



合計レコード数	1日平均レコード数	合計稼働時間	1日平均稼働時間
563704	18790	150時間26分55秒	5時間00分54秒

図 7　デバイス A のデータ蓄積量と稼働時間



合計レコード数	1日平均レコード数	合計稼働時間	1日平均稼働時間
31112	4446	91時間00分51秒	13時間0分7秒

図 8　デバイス B のデータ蓄積量と稼働時間

6.3 相対高度と加速度

気圧センサーから取得される大気圧の値（hPa）を使用すると，高度を算出することができる．この時，特定の場所において測定開始位置を基準とした相対高度を計算して，可視化すれば，その場所および周辺地域の地形等を把握することができる．加速度センサーは，人間の行動を特定するために利用することができる．図 9 は，実験期間中の 7 月 4 日の朝にデバイス A から取得された気圧センサーのデータをもとに相対高度を算出し，Google Map 上に可視化したものである．高低差を HSV 色空間で表現し，測定開始位置（図 9 の右上）よりも低い場所を青色に，高い場所を赤色とする．相対高度の値によってマーカーの色が変化し，測定開始位置（緑色）と終了位置（赤色）の間で大きく高低差があることがわかる． 図 10 は，上述の同時刻に取得していた加速度センサーの値と，前述の相対高度を時系列グラフにまとめたものである．徒歩移動の時は，加速度センサーの振れ幅が大きいのに対して，電車移動の際は座席に座っていたので振れ幅が小さい．また，相対高度のグラフと照らし合わせると，例えば D の電車移動の際に，高度が大きく上昇していることが分かる．

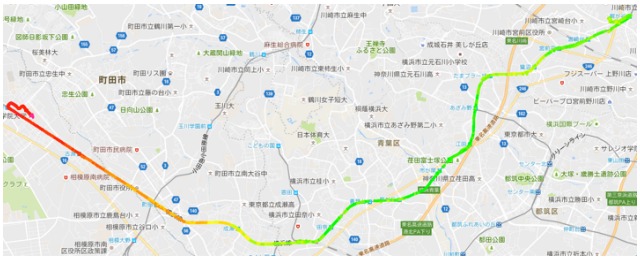


図 9 相対高度に変換したデータを表示した Google Map

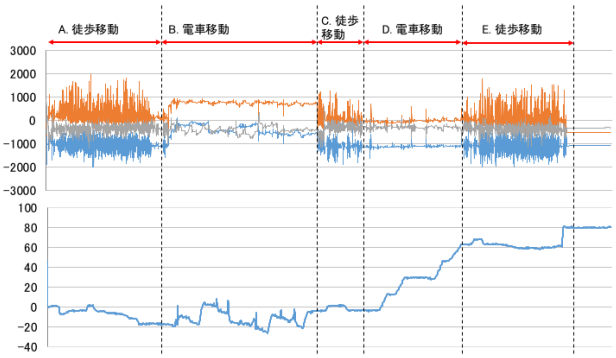


図 10　加速度と相対高度の時系列グラフ

6.4 照度

照度センサーは，周辺の明るさをルクス（Lux）単位で計測する．これを，夜間に外部で利用することで，周囲の明るさを基準としたセキュリティマップを作成し，夜間に暗くて危険な場所などを検出することができる．図 11 は，実験期間内の夜 20 時以降に，神奈川県内の住宅街にて，100 メートル圏内を対象として取得した照度センサーのデータを Google Map 上に表示したものである．日本工業規格の照度基準[6]における歩行者交通の維持照度値を参考に，5 lux 以上を明るい黄色に，1lux 未満を黒色となるように設定した．図 11 の下部の大通り付近は黄色い場所が多く，また右側のマンション入り口付近も比較的黄色い一方で，図 11 の左側の道路の黒色部分は，暗くて夜間は危険な地域であることが分かる．

図 11　照度データによるセキュリティマップの例

7. まとめと今後の課題

本論文では，Arduino とスマートフォンを利用した Bluetooth Low Energy（BLE）によるセンサーデータ収集方式とその試作, 及び実証実験について述べた. 具体的には，様々なセンサーに対応可能で BLE モジュールを搭載した Arduino センサーデバイスの試作，およびそのセンサーデバイスから BLE を介してデータを取得し, サーバへ送信するためのスマートフォンアプリの試作について述べた．1 ヶ月の実験を通じて約 780 万という多量のセンサーデータを蓄積し，明るさを基準とした夜間セキュリティマップの作成など，ユーザが知りたい情報を得るために利活用できることを検証した．実証実験により，提案方式が，ユーザ自身が求めるセンサーデータを柔軟に取得し，サーバに送信，蓄積して，利活用を行うための有効な方式であることを検証した．

今後の課題として，次の 2 点を挙げる．先ず 1 点目は，5V で動作するセンサーを試作し, センシングを行うことである．今回サンプルで制作した Arduino センサーデバイスは，表 1 で示したセンサーのうち 3.3V で動作するセンサーを中心にしている．スマートシティへの応用という観点からすると，ダストセンサーやガスセンサーを利用した周辺環境の汚染度合いのセンシングや，超音波距離センサーを用いた道路の混雑具合のセンシングなどは，よりニーズの高いセンサーデータと考えられる．２点目は，如何にして一般ユーザにインセンティヴを持たせ，モバイルセンシングへの参加を促すかの検討である．モバイルクラウドセンシングにおいて，一般ユーザからのデータ提供を如何に明示的あるいは暗示的に行うかは重要な課題であり，様々な研究が行われている[7][8]．今回の実験のように，センサーデバイスを常時携帯してセンシングを行う方式は，移動範囲内であれば大量のセンサーデータを蓄積することはできる．しかしながら，そのためにはユーザにセンサーデバイスを配り，ユーザに対して特定の時間帯に特定の場所でのセンシングに明示的な協力を求めなければならない．一方で，センサーデバイスを特定の場所に配置し，スマートフォンがセンサーデバイスに近づいたタイミングで BLE 通信を行い，センサーデータを取得しサーバに送信し，センサーデバイスから離れたら BLE を切断する方式が考えられる．この場合，用意するセンサーデバイスは少数に抑えつつ，アプリをスマートフォンにインストールさえすればセンサーデータを拾えるので，ユーザに対するモバイルセンシングへの協力も依頼しやすいと考えられる．

今後は，上述の２点の課題について検討することを念頭に，5V センサーデバイスの試作，および，固定したセンサーデバイスへの対応を念頭としたアプリ開発を行い，より大規模な実験を通じて，本論文にて示した提案方式の有効性を検証する予定である．

参考文献

[1] B. Guo, Z. Yu, X. Zhou and D. Zhang, "From participatory sensing to Mobile Crowd Sensing," *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, Budapest, 2014, pp. 593-598.
[2] V. Pankratius, F. Lind, A. Coster, P. Erickson and J. Semeter, "Mobile crowd sensing in space weather monitoring: the mahali project," in *IEEE Communications Magazine*, vol. 52, no. 8, pp. 22-28, Aug. 2014.
[3] Z. Yu, Y. Feng, H. Xu and X. Zhou, "Recommending travel packages based on mobile crowdsourced data," in *IEEE Communications Magazine*, vol. 52, no. 8, pp. 56-62, Aug. 2014.
[4] Neil Gershenfeld, "Fab: The Coming Revolution on Your Desktop – from Personal Computer to Personal Fabrication", New York: Basic Book, 2008.
[5] https://github.com/yskt0810/MobileSensorProject
[6] JIS 照度基準 https://www.gs-yuasa.com/gyl/jp/products/gs_html/shomei/technicaldata/pdf/technicaldata_5.pdf
[7] H. Ma, D. Zhao and P. Yuan, "Opportunities in mobile crowd sensing," in *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29-35, Aug. 2014.
[8] G. Cardone, A. Cirri, A. Corradi and L. Foschini, "The participact mobile crowd sensing living lab: The testbed for smart cities," in *IEEE Communications Magazine*, vol. 52, no. 10, pp. 78-85, October 2014.