IoTアプリケーションのためのSINETStreamベースAndroidセンサ情報収集アプリの開発

竹房 あつ子[1,2]　小林 久美子[1]　孫 静涛[1,†1]　合田 憲人[1,2]

概要：各種センサデータをクラウドに収集，蓄積し，機械学習で高度なデータ解析を行うIoTアプリケーションシステムの構築が可能になってきた．スマートフォンは，カメラ，GPS等の多様センサやバッテリが予め装備されており，IoTシステムへの活用が期待されるが，プログラム開発コストの高さから特に学術分野では十分に活用されていない．本研究では，スマートフォン向けのセンサ端末用アプリを開発し，研究・教育を目的としたIoTアプリケーションシステムの構築，利用促進の支援を目指す．我々は，広域データ収集・解析プログラム開発支援ソフトウェアパッケージSINETStreamを開発しており，Python/Java版に加えてAndroid版を公開している．本稿では，このAndroid版を利用してテキスト送受信アプリSINETStream Android Echo（Echo）とセンサ情報収集アプリSINETStream Android Sensor Publisher（Sensor）を開発する．特に，Sensorアプリは利用者のIoTアプリケーションでプログラムを開発することなく，そのまま利用されることを期待している．また，研究・教育用途での利用を支援するため，チュートリアルも開発し，IoTアプリケーションシステムの構築を体験できるようにする．

Development of SINETStream-based Android Sensor Data Collection Apps for IoT Applications

ATSUKO TAKEFUSA[1,2]　KUMIKO KOBAYASHI[1]　JINGTAO SUN[1,†1]　KENTO AIDA[1,2]

1. はじめに

広帯域，低遅延，長距離通信を可能にする第5世代移動通信システム「5G」の実用化とセンサ端末の小型化・安価化により，モバイルネットワークを介して各種センサから収集された大量な情報をクラウドに蓄積し，機械学習を用いた高度なデータ解析を可能にするIoT（Internet of Things）アプリケーションシステムの構築が可能になってきた．このようなIoTアプリケーションでは，開発コストや端末自体のコストの観点からRaspberry PiのようなLinuxベースの小型端末に必要なセンサを搭載して利用することが多い．一方で，スマートフォンはカメラ，GPS等の多様なセンサやバッテリが予め装備されていること，すでに多くの端末が普及していることから，IoTアプリケーションへの活用が期待されている．

国立情報学研究所（NII）では，学術情報ネットワークSINETをモバイル網に延伸させた「SINET広域データ収集基盤」（モバイルSINET）[1]により，センサ端末をSINETのVPNに直接接続させた安全なIoT実験環境の構築を可能にしている．また，応用研究分野の研究者によるIoTアプリケーションの開発を支援するため，広域データ収集・解析プログラム開発支援ソフトウェアパッケージSINETStream[2]を開発しており，Linux環境等で利用可能なPython/Java版[3], [4]とスマートフォンをセンサ端末として利用することを想定したAndroid版[5]を公開している．しかしながら，Linuxベースのシステムと比較するとスマートフォンの場合はプログラムの開発コストが高く，それにより特に学術分野では十分に活用されていない．

本研究では，SINETStream Android版を用いたアプリを開発し，研究・教育を目的としたIoTアプリケーショ

[1] 国立情報学研究所
National Institute of Informatics
[2] 総合研究大学院大学
The Graduate University for Advanced Studies (SOKENDAI)
[†1] 現在，日立製作所 研究開発グループ
Presently with Hitachi, Ltd., Research & Development Group

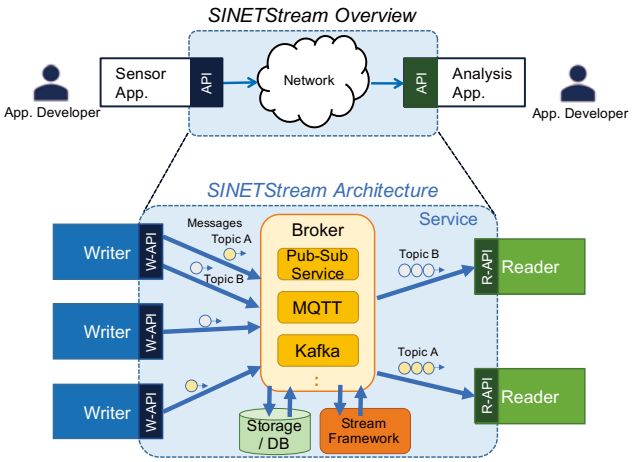

図 1　SINETStream の概念図.

ンシステムの構築，利用の促進を目指す．SINETStream Android 版は，IoT データの送受信のための機能を提供するコアライブラリと，Android 端末のセンサ情報の収集を支援するヘルパーライブラリで構成される．これらを利用して，Android 用のテキスト送受信アプリ SINETStream Android Echo（Echo と呼ぶ）とセンサ情報収集アプリ SINETStream Android Sensor Publisher（Sensor と呼ぶ）を開発する．Echo アプリでは，IoT システムのデータ収集における基本的な振る舞いを確認することができる．Sensor アプリでは，Android 端末をセンサとする IoT アプリケーションでのセンサプログラム開発が不要となり，IoT を活用した研究開発を加速させることができる．また，研究・教育用途での利用を支援するため，Echo と Sensor を用いたチュートリアルも開発，公開し，IoT アプリケーションシステムの構築，利用を簡単に体験できるようにする．

2. SINETStream の概要

広域データ収集・解析プログラム開発支援ソフトウェアパッケージ SINETStream の概要について説明する．SINETStream の API および機能の詳細は，公式サイト [2] および文献 [3], [4] を参照されたい．

2.1 SINETStream API

SINETStream では，トピックベースの Pub-Sub 型非同期メッセージングモデルを前提としている．図 1 に SINETStream の概念図を示す．SINETStream では，センサデータを送信する IoT デバイス側の Publisher プログラムを Writer，収集したデータを活用するサーバ計算機等で利用する Subscriber のプログラムを Reader と呼び，Writer と Reader で SINETStream が提供する API を実装することで図 1 の中央にあるメッセージブローカ（ブローカ）を介してセンサデータの送受信を可能にする．ブローカには，SINETStream に対応した任意のブローカソフトウェア，または MQTT (Message Queue Telemetry Transport)[6] ベースのクラウド Pub-Sub サービスを利用することができる．プログラムの可搬性を高めるため，各メッセージブローカとの通信に必要となるパラメータセットは別途定義できるようにしている．また，低遅延または高スループットな送受信処理を目的として，同期および非同期の Writer ／ Reader API を提供している．

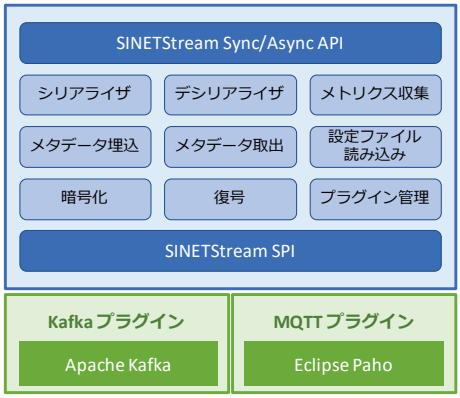

図 2　SINETStream のソフトウェア構成.

2.2 SINETStream の機能

SINETStream では，メッセージ送受信のための共通 API の提供の他，IoT アプリケーションシステムに必要なセキュリティ機能，メトリクス収集機能と，多様なブローカに対応するための SPI (Service Provider Interface) を提供している．図 2 に SINETStream のソフトウェア構成を示す．青枠内に示した部分は SINETStream 本体であり，API および SPI と，セキュリティ機能など IoT アプリケーションに必要とされる共通機能とプラグイン管理機能が実装されている．

セキュリティ機能として，認証・認可と通信・データの暗号化機能を提供している．認証・認可は，各ブローカ実装でサポートしている認証・認可機能を，SINETStream のコンフィグファイルまたは API から統一的に設定できるようにしている．公開鍵認証またはパスワード認証により，クライアントおよびサーバの認証が可能である．また，TLS による通信時の暗号化と SINETStream で提供しているデータ暗号化機能が利用できる．通信の暗号化は通信時のみ暗号化されるのに対し，データ暗号化はデータ収集サーバ側でも暗号化された状態でデータが可能されることになるため，より安全なデータ管理が可能になる．

メトリクス収集機能では，通信負荷をかけずにメッセージ送受信に関するメトリクス情報を収集することができる．SINETStream ライブラリ内で送受信メッセージ数，メッセージサイズ，エラー数をカウントすることで，ある期間における通信レート，最小／平均／最大メッセージサイズ，エラーレート等を提供することができる．利用者による測定用のデータ送信が必要ないため，通信負荷をかけずにメトリクス収集が可能になる．

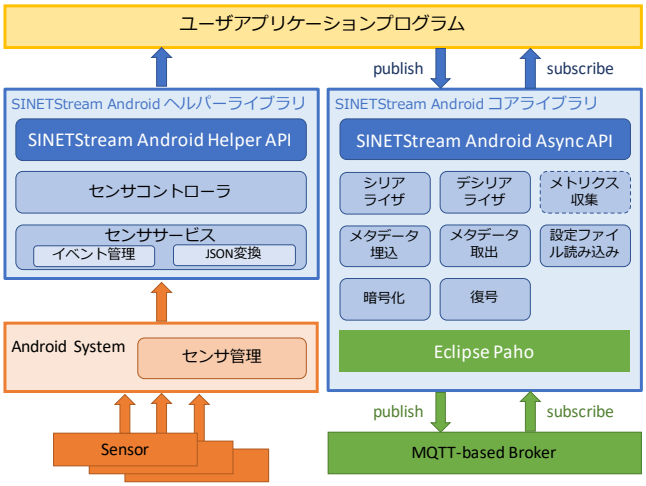

図 3 SINETStream Android 版のソフトウェア構成.

また，多様なメッセージブローカに対応できるようにするため，SPI を提供している．図 2 の緑枠内で示した部分は各ブローカ用のプラグインを表しており，SPI を介して独立したプラグインモジュールを組み合わせることができる．SINETStream v1.5 時点での Python ／ Java の実装では，MQTT ベースブローカと Apache Kafka[7], [8]（Kafka）をサポートしている．SPI を実装したプラグインを追加することで，エッジコンピューティング用のメッセージング基盤ソフトウェアなど，最新のメッセージブローカにも対応することができる．図 2 に SINETStream のソフトウェア構成を示す．青枠内に示した部分は SINETStream 本体であり，API および SPI と，セキュリティ機能など IoT アプリケーションに必要とされる共通機能とプラグイン管理機能が実装されている．緑枠内で示した部分は各ブローカ用のプラグインを表しており，SPI を介して独立したプラグインモジュールを組み合わせることができる．

3. SINETStream Android 版

Android 版は，SINETStream Python/Java 版で提供している IoT アプリケーションのためのデータ送受信機能の他，Android デバイスに装備されたセンサ情報の収集を容易にするためのライブラリも提供している [5]．**図 3** に SINETStream Android 版のソフトウェア構成を示す．図 3 右側のデータ送受信機能を提供するコアライブラリと，左側のセンサ情報収集を支援するヘルパーライブラリで構成されている．図中の矢印はデータの流れを表している．コアライブラリでは，下方向の矢印が Writer が publish する場合のデータの流れ，上方向の矢印が Reader が subscribe している場合のデータの流れを表す．ヘルパーライブラリでは，Android システムから各種センサデータを受け取り，適宜加工してユーザアプリケーションプログラムに送信する．以下でコアライブラリとヘルパーライブラリについて説明する．

3.1 SINETStream Android コアライブラリ

SINETStream Android コアライブラリは，SINETStream Python/Java 版と同等の機能を提供することを目的としている．ただし，Android はスマートフォンでの利用を前提とした OS であり，Android 用の Java では一部機能が制限されている．具体的には，非同期呼び出しとコールバックで実装する必要がある，Kafka の Android 用クライアントライブラリは提供されていない, という違いがある．よって，Android コアライブラリでは Python/Java 版で提供している機能と提供していない機能がある．図 3 右側で示すように，同期 Writer ／ Reader API は提供していない，SPI およびプラグインは非対応で Eclipse Paho[9] を用いた MQTT ブローカへの通信のみ対応している点が異なっている．また，暗号化・復号機能は現在実装中で SINETStream v1.6 リリースで公開予定であり，メトリクス収集機能は今後実装を検討している．

3.2 SINETStream Android ヘルパーライブラリ

ヘルパーライブラリは SINETStream Android 版固有の機能であり，Android 端末に装備された多様なセンサのデータ収集，加工を支援する．図 3 左側に，ヘルパーライブラリのソフトウェア構成の概要と Android システムのセンサ管理およびユーザプログラムとの関係を示す．ヘルパーライブラリは，コアライブラリ同様にユーザプログラムに対して非同期呼び出しとコールバックからなる API を提供し，Android システムのセンサの扱いを容易に行えるようにする．

ヘルパーライブラリは，センサコントローラとセンササービスの 2 つのモジュールで構成されている．センサコントローラは，Android 端末のセンサを制御するセンササービスのフロントエンドとして機能し，利用可能なセンサタイプの一覧の提供，センサデータの送出間隔の設定等を可能にする．センササービスは，Android 端末が装備する各種センサからセンサデータを受け取って蓄積し，設定された送出間隔でセンサデータを JSON 形式に加工して送出する．**図 4** に JSON 形式に加工されたセンサデータの例を示す．加工されたセンサデータでは，各センサで取得されたセンサデータに加え，Android 端末の情報が付与される．ヘルパーライブラリにより，複数センサのデータをまとめて 1 つのメッセージとして送信できるようになる．

4. SINETStream Android アプリの開発

応用研究分野の研究者に対して Android 端末を用いた IoT アプリケーションシステムの構築を支援するため，SINETStream Android のコアライブラリおよびヘルパーライブラリを用いたテキスト送受信アプリ Echo とセンサ情報収集アプリ Sensor を開発した．また，研究・教育用途での利用を支援するため，Echo と Sensor を用いたチュー

```
{
   "device": {
       "sysinfo": {
           "android": "11",
           "model": "Pixel 4",
           "manufacturer": "Google"
       },
       "userinfo": {
           "publisher": "user1@example.com",
           "note": "room1, bldg-A"
       },
       "location": {
           "latitude": "139.xxxxxx",
           "longitude": "35.xxxxxx"
       }
   },
   "sensors": [
       {
           "timestamp": "20200521T171528.094+0900",
           "type": "accelerometer",
           "name": "LSM6DSR Accelerometer",
           "values": [
               -4.1984100341796875,
               1.2285531759262085,
               5.224560737609863
           ]
       },
       {
           "timestamp": "20200521T171528.142+0900",
           "type": "light",
           "name": "TMD3702V Ambient Light Sensor",
           "value": 67.49675750732422
       },
           ...
   ]
}
```


図 4　JSON 形式に加工されたセンサデータ.

トリアルも開発，公開 [10] した．SINETStream Android 版および本研究で開発したアプリは，Android v8.0 以降に対応している．

4.1　テキスト送受信アプリ Echo

Echo は，IoT システムのデータ収集における基本的な振る舞いを確認することができるテキスト送受信アプリである．予め MQTT ブローカの接続情報を設定しておき，Android 端末のキーボードで入力した文字列を MQTT ブローカに送信すると，端末上に送信したメッセージが表示される．これにより，IoT システムのデータ収集における基本的な振る舞いを確認することができる．

図 5 に Echo アプリの状態遷移を示す．図中の a は Android のホーム画面，b は起動画面，c は初期画面，d は設定画面，e は Echo の主画面を表す．

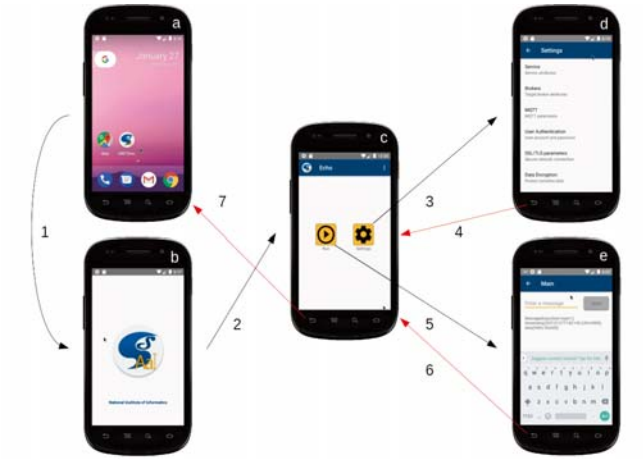

図 5　Echo アプリの状態遷移.

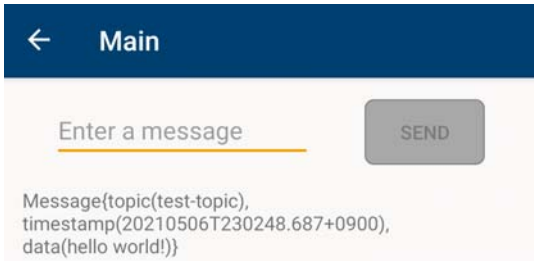

図 6　Echo アプリの e 主画面上部のスナップショット．メッセージ受信結果.

1. a ホーム画面で Echo アプリのアイコンを押下すると，アプリを起動する．
2. b 起動画面を表示後，c 初期画面に遷移する．
3. c 初期画面で右側の Settings ボタンを押下すると d 設定画面に遷移する．
4. d 設定画面で BACK ボタンを押下すると c 初期画面に戻る．
5. c 初期画面で左側の Run ボタンを押下すると，e 主画面に遷移する．
6. e 主画面で BACK ボタンを押下すると c 初期画面に戻る．
7. c 初期画面で BACK ボタンを押下すると a ホーム画面に戻る．

d 設定画面では，SINETStream のコンフィグファイル相当のパラメータ設定が可能である．具体的には，トピック名やブローカのアドレス，ポート番号等を設定することができる．また，e 主画面ではソフトウェアキーボードからテキストを入力して SEND ボタンを押すと，入力されたテキストを含むメッセージが MQTT ブローカに送信される．この際，メッセージは d 設定画面で指定されたトピック名で SINETStream Android コアライブラリの Writer API を介して MQTT ブローカに送信される．Echo アプリは指定されたトピックの Reader でもあるため，送信したメッセージを MQTT ブローカを介して受信し，その結果を e 主画面に表示する．図 6 に e 主画面の上部のスナップショット

図 7　Sensor アプリの主画面.

を示す．メッセージ受信結果では，入力したテキストデータだけでなくメッセージ全体が表示されている．メッセージには，トピック名，タイムスタンプ，入力したテキストデータが含まれている．

4.2　センサ情報収集アプリ Sensor

Sensor は，Android 端末に装備された多種センサの情報を収集するアプリであり，IoT アプリケーションシステムでそのまま利用可能となっている．予め MQTT ブローカの接続情報およびセンサデータ送出間隔を設定しておき，利用するセンサタイプを選択すると，指定された送出間隔で対象となるセンサデータが MQTT ブローカに送信される．

Sensor アプリの状態遷移は図 5 の Echo アプリとほぼ同様であるが，d 設定画面と e 主画面での操作が異なっている．d 設定画面では，SINETStream のコンフィグファイル相当のパラメータの設定が可能である他，センサデータの送出間隔やユーザ定義情報等の Sensor アプリ固有のパラメータが設定できる．e 主画面では，送出するセンサのタイプを指定できる．**図 7** は Sensor アプリの主画面であり，2 で SINETStream Android ヘルパーライブラリで認識されたセンサの一覧が表示されるため，利用したいセンサタイプを指定することができる．ただし，表示されるセンサの一覧は Android 端末の機種によって異なる．この他，図 7 の 1 の星印ではヘルパーライブラリのセンササービスが稼働中であることを示している．3 の RUN ／ STOP ボタンでは，センサデータの取得および MQTT ブローカへのメッセージ送信の開始／停止を行う．4 ではセンサデータ送信中に送信時刻，総送信メッセージ数が表示され，5 のリセットボタンではメッセージ送信停止時に統計情報をリセットすることができる．

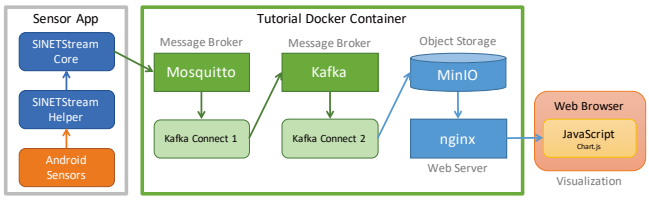

図 8　Sensor チュートリアルのサーバ用 Docker コンテナの概要.

4.3　Sensor を用いたチュートリアル

研究・教育用途での利用を支援するため，Echo と Sensor を用いたチュートリアルを開発した．Echo アプリでは，Writer と Reader の機能がアプリで実装されており，Python/Java 版のチュートリアルで提供されているサーバ用 Docker コンテナイメージ [11] を用いて MQTT ブローカを用意すれば動作を確認することができる．一方，Sensor アプリでは Writer 側の機能しか実装されておらず，動作確認には Reader 側の機能を持つプログラムが必要となる．そこで，Sensor アプリが送信したセンサデータを受け取って蓄積，可視化までを行う Sensor チュートリアルのサーバ用 Docker コンテナイメージも開発し，Android のセンサを活用した IoT アプリケーションシステムの構築，利用が体験できるようにした．本稿では，Sensor のチュートリアルについて紹介する．

Sensor チュートリアルの手順を以下に示す．

(1) Sensor チュートリアル用サーバを設定する．
(2) Android 端末でアプリの設定，実行する．
(3) 観測用端末で可視化結果を確認する．

まず，(1) ではサーバ計算機上で**図 8** に示す Sensor チュートリアル用のサーバ Docker コンテナを実行する．本コンテナでは，いずれもオープンソースソフトウェアである MQTT ブローカの Eclipse Mosquitto[12]，メッセージブローカ Kafka，オブジェクトストレージ MinIO[13]，Web サーバ nginx[14] がインストールされている．また，Kafka Connect 1 で Mosquitto に書き込まれたセンサデータを Kafka へ，Kafka Connect 2 で Kafka に書き込まれたセンサデータを MinIO へ蓄積するように設定した．MinIO は Amazon S3 (Simple Storage Service) 互換のオブジェクトストレージであり，nginx を介して指定されたセンサデータを取り出し，JavaScript で可視化できるようにした．別途処理プログラムを用意して Mosquitto からストレージにデータを格納する方法も可能であるが，本研究では Kafka ブローカを間に挟む構成とした．これは，Mosquitto では収集されたデータが永続化されないのに対して Kafka では一定期間永続化される点，Kafka Connect と呼ばれるツールを用いることで処理プログラムを用意しなくても上記のような処理ができる点による．

（2）では，4.2 節の操作により，アプリのインストール，センサデータ送出間隔の設定，センサタイプの選択，セン

図 9　Sensor チュートリアルでのセンサデータ可視化画面のスナップショット．上から歩数，照度，加速度のグラフが示されている．

サデータの送信処理を行う．(3) では，観測用端末で以下の URL にアクセスすると，図 9 の画面でセンサデータが取得できていることが確認できる．

`http://<server_address>/chart.html`

図 9 は，上から歩数，照度，加速度のセンサデータがグラフ化されたものである．表示されるグラフは必要に応じて追加・削除することができる．Sensor アプリを実行すると，定期的に指定したセンサのデータがグラフ上で更新されることを確認した．

5. 関連研究

Android スマートフォンを IoT アプリケーションシステムのセンサ端末として活用する試みは複数ある．文献 [15] では，患者の監視を目的として加速度センサ，温度センサ，心電図センサのデータを Bluetooth または NFC (Near Field Communication) 経由で Android 端末に収集して可視化するとともに，セントラルサーバに送信するマイクロコントローラを開発している．文献 [16] も eHealth を目的とし，患者に装着した心電図および心拍数のセンサを Wi-Fi 802.11 で Android 端末に接続し，GPS データとともに収集し，リアルタイムに可視化するシステムを構築している．これらのシステムでは，利用シナリオに応じて個別のセンサに特化したシステムが構築され，独自の方式でデータ収集等を行っているのに対し，本研究ではより広範な目的で利用可能なアプリとして開発されている．特に，Android システムが管理している複数センサに対応している点，サーバへのデータ収集は SINETStream により多種メッセージブローカに対応している点，SINETStream の持つ IoT システムに必要な各種機能が活用できる点で異なる．

Android 端末のセンサ情報をクラウドで収集・蓄積・解析し，その結果を活用する IoT システムの 1 つとして，自然災害による被害リスクを軽減することを目的とした早期警鐘システム [17] がある．このシステムでは，広域に分散している利用者が保有する多数の Android 端末で検知された揺れの情報をクラウドソーシングで収集し，クラウドで地震であると予測されると各 Android 端末の利用者に通知する．本研究で開発した Sensor アプリもこのような用途で活用されることが期待できる．

6. まとめと今後の課題

本研究では，SINETStream Android 版のコアライブラリと，Android 端末のセンサ情報の収集を支援するヘルパーライブラリを用いて，Android 用のテキスト送受信アプリ SINETStream Android Echo（Echo）とセンサ情報収集アプリ SINETStream Android Sensor Publisher（Sensor）を開発した．Echo アプリを用いることで，IoT システムのデータ収集における基本的な振る舞いを確認することが可能である．また，Sensor アプリは Android 端末をセンサとする IoT アプリケーションシステムでの利用が可能であり，IoT を活用した研究開発を加速させることができる．更に，研究・教育用途での利用を支援するため，Echo と Sensor を用いたチュートリアルも開発，公開した．Sensor チュートリアルでは，センサデータの収集，蓄積，可視化を可能にするサーバ用 Docker コンテナも開発し，本チュートリアルにより Android 端末を用いた IoT アプリケーションシステムの構築・利用が体験できるようにした．

今後は，Sensor アプリの利便性をより高めるとともに，セキュリティ機能やメトリクス収集機能など SINETStream Android 版の機能を拡充していく．また，実アプリケーションへの適用を行っていく．

謝辞　本研究にご協力いただいた数理技研の遠藤雅彦様，小泉敦延様，鯉江英隆様に深く感謝いたします．

参考文献

[1] SINET Wide Area Data Collection Infrastructure (WADCI), `https://www.sinet.ad.jp/wadci` (accessed 10-05-2021).
[2] SINETStream, `https://www.sinetstream.net/` (accessed 10-05-2021).
[3] 竹房あつ子，孫　静涛，藤原一毅，吉田　浩，合田憲人：IoT ストリームデータ処理のためのソフトウェアライブラリ SINETStream の開発，情報処理学会研究報告 2020-IOT-48，pp. 1–8 (2020).
[4] Takefusa, A., Sun, J., Fujiwara, I., Yoshida, H., Aida, K. and Pu, C.: SINETStream: Enabling Research IoT Applications with Portability, Security and Performance Requirements, *Proc. IT in Practice (ITiP) Symposium*

COMPSAC 2021 (To appear).

[5] 孫　静涛，竹房あつ子，藤原一毅，吉田　浩，合田憲人：IoT アプリ構築支援のための SINETStream Android 用プラグインの開発，マルチメディア，分散，協調とモバイル (DICOMO2020) シンポジウム論文集，pp. 421–427 (2020).

[6] MQTT (Message Queue Telemetry Transport), `http://mqtt.org/` (accessed 10-05-2021).

[7] Apache Kafka, `https://kafka.apache.org/` (accessed 10-05-2021).

[8] Kreps, J., Narkhede, N. and Rao, J.: Kafka : a Distributed Messaging System for Log Processing, *NetDB workshop 2011*, pp. 1–5 (2011).

[9] Eclipse Paho, `https://www.eclipse.org/paho/` (accessed 10-05-2021).

[10] SINETStream Android 版チュートリアル, `https://www.sinetstream.net/docs/tutorial-android/` (accessed 10-05-2021).

[11] SINETStream Python/Java 版チュートリアル, `https://www.sinetstream.net/docs/tutorial/` (accessed 10-05-2021).

[12] Eclipse Mosquitto, `https://mosquitto.org/` (accessed 10-05-2021).

[13] MinIO, `https://min.io/` (accessed 10-05-2021).

[14] nginx, `http://nginx.org/` (accessed 10-05-2021).

[15] Yi, W.-J., Jia, W. and Saniie, J.: Mobile Sensor Data Collector using Android Smartphone, *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 956–959 (online), DOI: 10.1109/MWSCAS.2012.6292180 (2012).

[16] Khalaf, A. and Abdoola, R.: Wireless Body Sensor Network and ECG Android Application for eHealth, *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, pp. 1–4 (online), DOI: 10.1109/ICABME.2017.8167526 (2017).

[17] Heryana, A., Nugraheni, E., Kusumo, B., Rojie, A. F. and Setiadi, B.: Applying Agile Methods in Designing an Earthquake and Landslide Early Warning System Application for Android, *2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pp. 80–84 (online), DOI: 10.1109/IC3INA.2017.8251744 (2017).