採択課題番号：IDEAS201601

普及型フィールドセンサによる OGC 標準データの災害情報サービスへの提供の研究

保科　紳一郎*，矢吹　益久*，本橋　元*，本多　潔**，竹島　喜芳**，竹本　勝政***

*鶴岡工業高等専門学校、**中部大学国際 GIS センター、***㈱ファルコン

1. はじめに

近年，「ゲリラ豪雨」が頻発し、都市部において多数存在する道路を深く掘ったアンダーパス部の冠水が頻発するようになり、物質・人的な被害の大きさが注目されるようになった。これらに関わる行政機関は、監視の必要な個所に測定機器を配置しているが、監視すべき地点は、数が多くそして広く分布している。一般的に、このような測定機器は高額であるため、必要十分な監視体制を構築するに至っていると言い難い。広い範囲に散在する測定機器による監視体制を補完するために、安価で、データ精度を許容範囲で保証し、野外での使用に耐えるフィールドセンサが実現できれば、多数の監視対象に対応できると期待される。

一方、センサ技術・通信技術の発展により機器の IoT(Internet of Things)化の進展が著しい。IoT に関わる情報の取り扱いでは、時刻・位置・測定値を継続的に取得する場合が多い。そして、先にあげたフィールドセンサはまさしく時刻・位置・測定値を継続的に取得するシステムと言える。このような時刻・位置情報を伴う情報を取り扱う際の標準的手法について、OGC(Open Geospatial Consortium) (参考文献 1)は様々な規定を提案している。OGC に準拠したデータの処理方法を採用することは、情報の収集・活用に大きなメリットがある。我が国における OGC を基にしたデータの標準化の取り組みとして、平成 26 年度末に、国の取り組みとして高度情報通信ネットワーク社会推進戦略本部（IT 総合戦略本部）・新戦略推進専門調査会農業分科会において、農業情報創成・流通促進戦略が示された。その際に、提案された農業情報の標準化（参考文献 2）の中には OGC を基にしたものがある。これらの農業情報の標準化の具体的内容は「個別ガイド」としてまとめられている。

本研究では、小型、低消費電力、高性能となってきたマイクロコンピュータを搭載したフィールドセンサから取得した情報を、OGC 標準の中に取り込むことを目的に、フィールドセンサの出力を、OGC 標準においてデータの収集および配信を行う SOS(Sensor Observation Service)サーバに対応できるようにすることを目標とする。本報告では、OGC に準拠して記述されたセンサ側の仕様や測定データを、OGC においてセンサデータの収集・配信を行う SOS(Sensor Observation Service)サーバに、登録・参照する手続きを、センサ側のアプリケーションを開発の視点から具体的手法に示す。

SOS 対応化されたフィールドセンサの開発を進めるとともに、その実用例としてアンダーパス用水位測定システムへの実装に取り組んでいる。平成 28 年 3 月より、愛知県春日井市においてアンダーパス部の水位測定を約一年間行った。本水位計の一年間の稼働を踏まえ、春日井市当局の担当者とのインタビューを行った。使用者側からの意見をいただくことができた。

2. SOS サーバへの対応

2-1. SOS API Ver.1.0 と Ver.2.0 の差異

本研究では、OGC に準拠する SOS サーバとの情報の送受信する手段として、SOS API を利用している。図 1 中に実線赤矢印で示されている情報の流れが、外部に公開されている SOS API によって実行される。近年、SOS Ver. 1.0 から Ver. 2. 0 へバージョンアップする際に、機能が拡張された。従来の Ver.1.0 では、GIS 等のアプリケーションは SOS サーバに対して Data Base のデータを取得する API

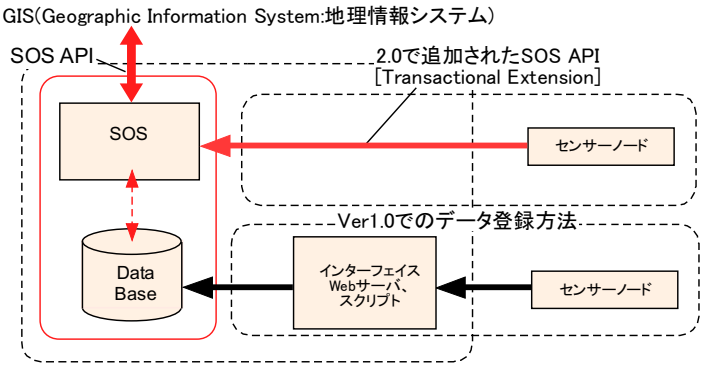

図 1　SOS 2.0 API の変更点

のみが用意されていた。一方、Ver.1 ではセンサ側の情報を直接 SOS サーバに入力する API が無いので、この場合、センサから情報を受信した際に web サーバ上で動作する GCI を用いて、SOS サーバが管理する Data Base に情報を書き込む。Ver. 2.0 で新たに拡張された機能（Transactional Extension）では、センサ側より SOS サーバへデータを送り出す API が整備された。この拡張された API に従えば容易にセンサ側からデータを SOS サーバへアップできる。(参考文献 3)

2-2. SOS サーバとのデータ送受信の検証用環境の構築

マイコンと SOS サーバ間の通信の検証には SOS サーバを用意する必要がある。本研究では、広く利用されている 52°North(参考文献 4)の SOS サーバを使用している。この SOS サーバのインストールは 52° North の web サイト上に公開されている。しかし、OS の種類やバージョンの違いによって、説明通りにインストールすることは容易ではない。近年、急速に拡大するクラウドサービスの中には IoT サービスに対応するために SOS サーバを提供するものもある(参考文献 5)。本研究では、無料で簡単に SOS サーバを含む GIS 環境を構築できる OSGeo-Live(参考文献 6)の Live DVD を利用した。Live DVD から PC を起動し、PC の HDD へ OS・デスクトップ環境・GIS の利用に便利なアプリケーション等を一括してインストールするため、しばしば問題となる OS・デスクトップ環境・アプリケーション間の設定の不整合によるトラブルを避けることができる。

SOS API は XML 形式、又は JSON 形式で表され、これらの表記方式で表されたデータを HTTP プロトコルの POST メソッドで SOS サーバの指定アドレスに送信する。マイコンから SOS API を用いて、SOS サーバと通信するプログラムの開発において、SOS API の表記だけでなく、HTTP 送信時のヘッダ情報なども含めて SOS サーバと正しく通信できるかを検証する必要がある。SOS API を用いて SOS サーバとのデータの送受信を検証するためには、HTTP プロトコルを用いて POST メソッドでデータを指定アドレスに送ることのできるツールを利用した。例えば、図 2 に示す 52°North SOS サーバに付属する Test Client は非常に使いやすい。このような SOS 付属の Test Client 以外にも、Web ブラウザにはその拡張機能として POST データの送信ツールが付属するのでそちらを利用することもできる。例えば図 3 はその一つである HTTP Resource Test である。HTTP Resource Test は Web ブラウザ FireFox で利用できるアドオンとして提供されるツールである。

HTTP プロトコルを用いて通信を行っているので SOS サーバと、Test Client や HTTP Resource Test との通信は、例えば、図 4 のように WireShark のようなキャプチャソフトによって、その詳細な通信内容を確認することができる。キャプチャソフトで得られた通信内容は、マイコンに SOS API を実装する際に、非常に参考になる。

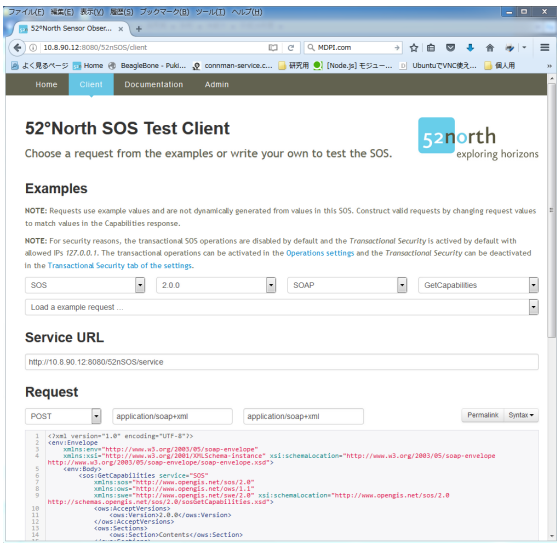

図 2 North52° SOS サーバ Test Client 起動画面

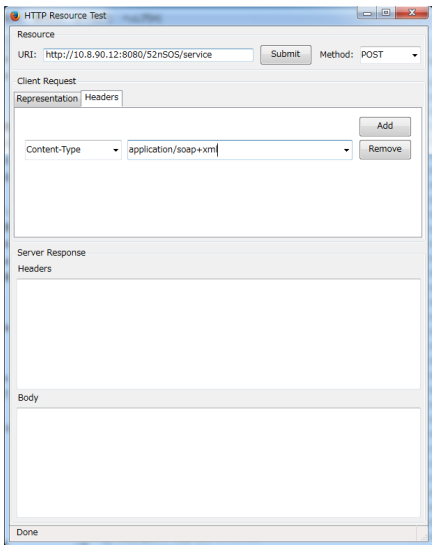

図 3 HTTP Resource Test 起動画面

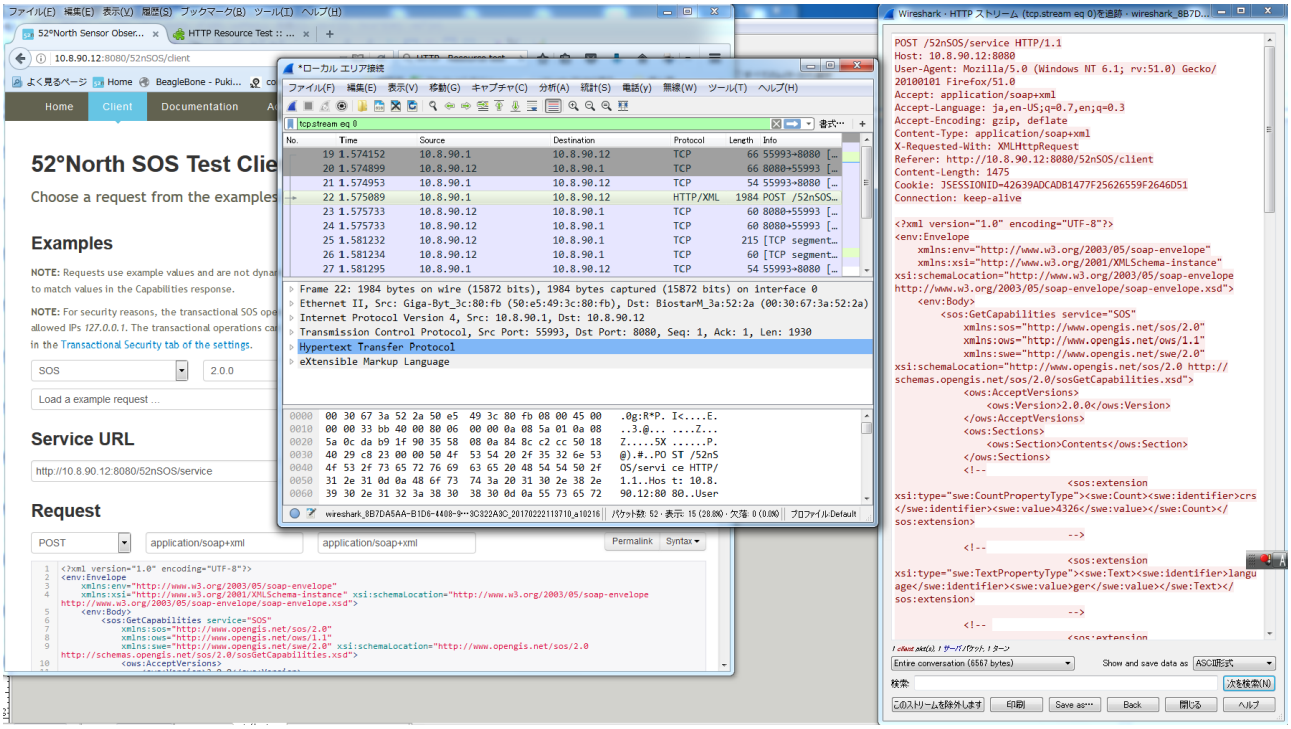

図 4 WinShark を使って SOS サーバ間との通信を取得

2-3. SOS サーバと API を使った通信の検証

2-3(a) SOS サーバとの通信手順

2-2 で示したツールを使って、SOS サーバとクライアント間の通信を検証する。実際にクライアントと SOS サーバ間の通信手順は図 5 のシーケンス図に示すように SOS API を使って進められる。

まず始めに、GetCapabilities を使って、クライアント側より SOS サーバに SOS サーバにアクセスするための諸情報を要求する。この要求に対する SOS サーバからの応答には、SOS API を使って情報をやりとりするに必要な情報の全てが入っている。以後の SOS API の送受信には GetCapabilities の応答に含まれる情報が使われる。

InsertSensor の要求では、センサの位置、名前、データの形式等のクライアントに係わる情報を SOS サーバに送る。SOS サーバは InsertSensor の要求に含まれる情報に基づいて、クライアント側から送られるセンサに係わる情報を保存する。

InsertObservation の要求では、SOS サーバにクライアント側から測定データを送信する。GetObservation の要求では SOS サーバに対して保存しているデータを要求する。

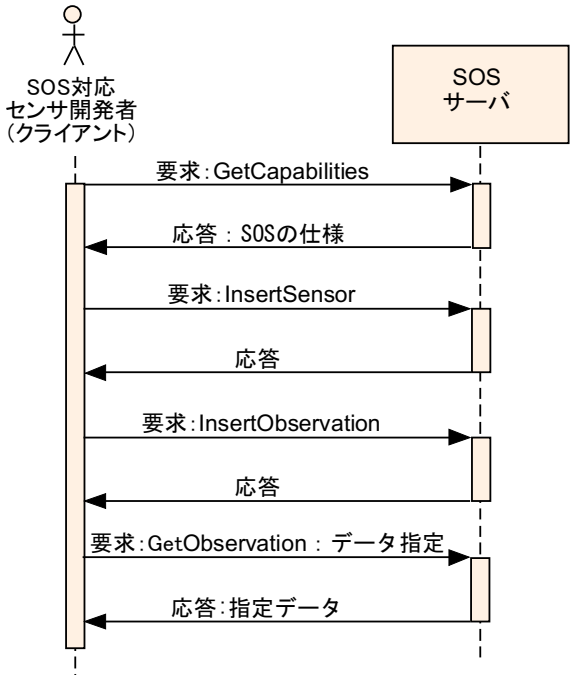

図 5 SOS サーバとの通信手順

SOS 対応センサ開発者は、少なくとも上記の 4 つの API を使い、センサの開発を行う。

2-3(b). GetCapabilities を使った SOS 内の情報の要求とその応答

図 6 に GetCapabilities の要求の例を示す。図 6 が実際に HTTP プロトコル・ボディ部として送信される GetCapabilities の実体となるデータである。GetCapabilities に限らず SOS API は XML 形式で記述される

ので、SOS API の先頭行に XML 宣言<?xml version="1.0" encoding="UTF-8"?>を配置し、具体的な内容は 2 行目以降に記述される。

この GetCapabilities の内容の中で、<ows:Sections>要素の内容である①~④は SOS サーバに要求する内容を示している。図に示した①から④以外にも要求できる内容はある。SOS へのデータをアップする際に必要な内容は主に①、④に対する応答の中に含まれる。①に対する応答には、SOS サーバで用意されている API の一覧、必要な HTTP ヘッダ、SOS サーバに登録されているセンサの一覧、センサから送られて来るデータの属性（温度、気温等）の情報を含んでいる。④に対する応答には SOS サーバに登録されているセンサの一覧が含まれる。

図 7 に GetCapabilities の応答の抜粋を示す。①は GetCapabilities の応答であることを示す。②は図 6 中の①に項目に対する応答であることを示す。③は SOS API InsertSensor が利用可能であることを示す。④-⑤は InsertSensor の要求を、SOAP メッセージ形式で送る際には、HTTP ヘッダに Content-Type: application/soap+xml をつける必要があることを示している。⑥は図 6 中の④の項目に対する応答であることを示している。⑦は以後に SOS サーバに登録されているセンサに関する情報が含まれていることを表す。⑧は SOS に登録されたセンサ名である。SOS API においてセンサを指定する場合は、<swes:procedure>センサ名</swes:procedure>と記述する。⑨-⑩は、このセンサから得られる情報を示している。センサから送られてくる測定項目は<swes:observableProperty>物理量</swes:observableProperty>と記述される。物理量の表記は任意に定義できるが、他のセンサで測定された同様な物理量と区別するために、物理量の定義を示した機関名などが付加される場合が多い。例えば、農業情報系分野では気温の表記を、http://www.kantei.go.jp/jp/singi/it2/senmon_bunka/shiryo/agro-env.html#air_temperature と定めている(参考文献 7)。⑪-⑫は、GetObservation などでデータを要求する際に、応答に含まれるデータの書式を示している。GetCapabilities の応答の中に数種類の<sos:responseFormat>書式</sos:responseFormat>が含まれることもあり、複数の書式が存在する場合はその一つを指定してデータの要求を行う。

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope   http://www.w3.org/2003/05/soap-envelope/soap-envelope.xsd">
    <env:Body>
        <sos:GetCapabilities service="SOS"
            xmlns:sos="http://www.opengis.net/sos/2.0"
            xmlns:ows="http://www.opengis.net/ows/1.1"
            xmlns:swe="http://www.opengis.net/swe/2.0"    xsi:schemaLocation="http://www.opengis.net/sos/2.0 http://schemas.opengis.net/sos/2.0/sosGetCapabilities.xsd">
            <ows:AcceptVersions>
                <ows:Version>2.0.0</ows:Version>
            </ows:AcceptVersions>
            <ows:Sections>
                <ows:Section>OperationsMetadata</ows:Section>      ←①
                <ows:Section>ServiceIdentification</ows:Section>  ←②
                <ows:Section>ServiceProvider</ows:Section>         ←③
                <ows:Section>FilterCapabilities</ows:Section>      ←④
                <ows:Section>Contents</ows:Section>
            </ows:Sections>
        </sos:GetCapabilities>
    </env:Body>
</env:Envelope>
```

図 6 SOS API　GetCapabilities の内容

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope …>
  <soap:Header/>
    <soap:Body>
      <sos:Capabilities version="2.0.0">  －①
        <ows:OperationsMetadata>  －②
          <ows:Operation name="InsertSensor">  －③
            <ows:DCP>
              <ows:HTTP>
                <ows:Post xlink:href="http://localhost:8080/52nSOS/service/soap">
                  <ows:Constraint name="Content-Type">  －④
                    <ows:AllowedValues>
                       <ows:Value>application/soap+xml</ows:Value>  －⑤
       …
     <sos:contents>  －⑥
        <sos:Contents>
        <swes:offering/>
          <swes:offering>
            <sos:ObservationOffering>  －⑦
              <swes:identifier>http://www.tsuruoka-nct.ac.jp/test/offering/0</swes:identifier>
              <swes:procedure>http://www.tsuruoka-nct.ac.jp/test/procedure/0</swes:procedure>  －⑧
              <swes:observableProperty>http://www.tsuruoka-
                 nct.ac.jp/test/observableProperty/temperature</swes:observableProperty>  －⑨
              <swes:observableProperty>http://www.tsuruoka-
              nct.ac.jp/test/observableProperty/water_depth</swes:observableProperty>  －⑩
              <sos:responseFormat>application/json</sos:responseFormat>  －⑪
              <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>  －⑫
...
```

図 7 SOS API　GetCapabilities に対する応答

2-3(c). InsertSensor を使ったセンサ登録の要求とその応答

図 8 に、SOS サーバにデータを登録する API InsertSensor の要求の例の抜粋を示す。①では、登録するセンサ名を示す。①に記述されたセンサ名は、SOS 上のセンサを参照する際に用いられる。②では、センサの入出力以外の情報、例えば、位置情報はこちらに記述される。③では、位置情報は EPSG コード 4326 で表現されることを示している。EPSG コード 4326 では測地系が WGS84 系であるので、位置情報を緯度、経度、高度で表現する際に Google マップで表示される座標値をそのまま用いることができる。④-⑥はセンサの位置を経度、緯度、高度で表現している。⑦では、このセンサからの出力を表している。ここでは temperature と表記されている。しかし、temperature と表記されていても何の温度であるか等の情報が無いため、他のセンサから得た temperature と単純に比較できない。そこで temperature がどのように定義された値であるかを明確にするために、⑧のようにメタ情報を加えて記述される。

図 9 に、InsertSensor の要求に対する SOS サーバの応答をしめす。登録が正常に行われれば SOS サーバに登録されるとセンサ名を返す。登録が正常に行われない場合は、その問題に係わる内容を返す。

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope・・・>
<env:Body>
  <swes:InsertSensor ・・・>
    <swes:procedureDescriptionFormat>http://www.opengis.net/sensorML/1.0.1</swes:procedureDescriptionFormat
    >
      <swes:procedureDescription>
```

```
      <sml:SensorML version="1.0.1">
        <sml:member>
          <sml:System>
            <sml:identification>
              <sml:IdentifierList>
                <sml:identifier name="uniqueID">
                  <sml:Term definition="urn:ogc:def:identifier:OGC:1.0:uniqueID">
                ①－<sml:value>http://www.tsuruoka-nct.ac.jp/test/procedure/1</sml:value>
                        ・・・
            <sml:capabilities name="featuresOfInterest">  －②
              <swe:SimpleDataRecord>
                <swe:field name="featureOfInterestID">
                  <swe:Text>
                     <swe:value> http://www.tsuruoka-nct.ac.jp/test/offering/1 </swe:value>
                     ・・・
            <sml:position name="sensorPosition">
              <swe:Position referenceFrame="urn:ogc:def:crs:EPSG::4326">  －③
                <swe:location>
                  <swe:Vector gml:id="STATION_LOCATION">
                    <swe:coordinate name="easting">
                      <swe:Quantity axisID="x">
                        <swe:uom code="degree"/>
                           <swe:value>139.79678750038147</swe:value>  －④
                      </swe:Quantity>
                    </swe:coordinate>
                      <swe:coordinate name="northing">
                        <swe:Quantity axisID="y">
                          <swe:uom code="degree"/>
                            <swe:value>38.70956778821735</swe:value>  －⑤
                             </swe:Quantity>
                           </swe:coordinate>
                      <swe:coordinate name="altitude">
                        <swe:Quantity axisID="z">
                          <swe:uom code="m"/>
                            <swe:value>26.33</swe:value>  －⑥
              ・・・
            <sml:outputs>
              <sml:OutputList>
                <sml:output name="temperature">  －⑦
                  <swe:Quantity definition="http://www.tsuruoka-
                  nct.ac.jp/test/observableProperty/temperature">  －⑧
                    <swe:uom code="Cel"/>  －⑨
                  </swe:Quantity>
                </sml:output>
            <sml:output name="water_depth">  －⑩
              <swe:Quantity definition="http://www.tsuruoka-nct.ac.jp/test/observableProperty/water_depth">
                <swe:uom code="cm"/>  －⑪
              ・・・
          </sml:SensorML>
      </swes:procedureDescription>
      ・・・
    </swes:InsertSensor>
  </env:Body>
</env:Envelope>
```

図 8 SOS API　InsertSensor の内容

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope・・・>
     <soap:Header/>
     <soap:Body>
       <swes:InsertSensorResponse>
         <swes:assignedProcedure>http://www.tsuruoka-nct.ac.jp/test/procedure/1</swes:assignedProcedure>
         <swes:assignedOffering>http://www.tsuruoka-nct.ac.jp/test/offering/1</swes:assignedOffering>
       </swes:InsertSensorResponse>
     </soap:Body>
   </soap:Envelope>
```

図 9 SOS API　InsertSensor の応答

2-3(d). InsertObservation を使ったデータの登録

図１０は、SOS サーバにデータの登録を要求する API InsertObservation の抜粋である。ここでは、温度[℃]、水位[cm]を同時に測定して、その結果を SOS サーバに登録する。SOS に送信する温度、水位毎に、①のように<sos:observation>データ</sos:observation>の間に記述する。①で登録する測定データは、⑤で示された気温である。②はデータの種類を示す。ここでは、OGC 標準に定められた OM(Observations and Measurements)（参考文献 8)において、「測定より得られたスカラー量」を表す OGC Name: http://www.opengis.net/def/observationType/OGCOM/2.0/OM_Measurement を指定している。③は測定時刻を表している。ここで phenomenonTime は測定対象となる時間範囲を示している。resultTime は測定値を確定した時刻である。厳密に言えば phenomenonTime と resultTime は異なるが、その時刻が十分に近い・同時である場合は同じ時刻を使用している。④は測定を行ったセンサを表す。⑤は測定値の種類を指定している。⑥は測定を行ったセンサの位置情報等の測定以外の情報を記述している。⑦は測定を行った位置を表す。センサの位置と測定を行った位置が同じであれば、InsertSensor で記載されたセンサの位置をそのまま使うことができる。ただし、センサの位置と測定を行った位置が異なる場合は測定を行った位置を記述する。⑧は測定結果 25.0℃を表している。⑨は<sos:observation>から始まるので、別の測定値についての記述がここからは始まっていること示している。ここから水位の測定結果について記述している。⑩は、先の気温の測定、水位の測定も同時に実施しているため、水位測定の phenomenonTime、resultTime は先の気温のそれと同じとなる。そこで、<om:phenomenonTime>、<om:resultTime>は href 属性を用いて、先に記述された phenomenonTime を参照している。⑪は測定結果が水位であることを示している。⑫は、<om:phenomenonTime>と同様に、先の<om:featureOfInterest>が気温のそれと同じなので、参照を用いて記述量を減らし見やすくしている。⑬は水位の測定結果が 1cm であることを表している。図 11 は、InsertObservation の要求が成功した際の SOS サーバからの応答である。

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope ・・・>
  <env:Body>
    <sos:InsertObservation ・・・>
      <sos:offering>http://www.tsuruoka-nct.ac.jp/test/offering/1</sos:offering>
      <sos:observation>  －①
        <om:OM_Observation gml:id="o1">
 ②―      <om:type xlink:href="http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement" />
          <om:phenomenonTime>  －③
            <gml:TimeInstant gml:id="phenomenonTime">
              <gml:timePosition>2001-01-10T00:00:00+09:00</gml:timePosition>
            </gml:TimeInstant>
          </om:phenomenonTime>
          <om:resultTime xlink:href="#phenomenonTime" />
          <om:procedure xlink:href="http://www.tsuruoka-nct.ac.jp/test/procedure/1" />  －④
          <om:observedProperty xlink:href="http://www.tsuruoka-
              nct.ac.jp/test/observableProperty/temperature" />  －⑤
```

```
        <om:featureOfInterest>  －⑥
          <sams:SF_SpatialSamplingFeature gml:id="ssf_test_feature">
            <gml:identifier codeSpace="">http://www.tsuruoka-
                 nct.ac.jp/test/featureOfInterest/1</gml:identifier>
            <gml:name>Tsuruoka</gml:name>
            <sf:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-
                OM/2.0/SF_SamplingPoint" />
            <sf:sampledFeature xlink:href="http://www.tsuruoka-nct.ac.jp/test/featureOfInterest/1" />
            <sams:shape>
              <gml:Point gml:id="test_feature_1">
                <gml:pos srsName="http://www.opengis.net/def/crs/EPSG/0/4326">38.70956778821735
                   139.79678750038147</gml:pos>  －⑦
              </gml:Point>
            </sams:shape>
          </sams:SF_SpatialSamplingFeature>
        </om:featureOfInterest>
        <om:result xsi:type="gml:MeasureType" uom="Cel">25.0</om:result>  －⑧
      </om:OM_Observation>
    </sos:observation>
    <sos:observation>  －⑨
      <om:OM_Observation gml:id="o2">
        <om:type xlink:href="http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement" />
        <om:phenomenonTime xlink:href="#phenomenonTime" />  －⑩
        <om:resultTime xlink:href="#phenomenonTime" />
        <om:procedure xlink:href="http://www.tsuruoka-nct.ac.jp/test/procedure/1" />
        <om:observedProperty xlink:href="http://www.tsuruoka-
nct.ac.jp/test/observableProperty/water_depth" />  －⑪
        <om:featureOfInterest xlink:href="#ssf_test_feature" />  －⑫
        <om:result xsi:type="gml:MeasureType" uom="cm">1.0</om:result>  －⑬
      </om:OM_Observation>
    </sos:observation>
  </sos:InsertObservation>
 </env:Body>
</env:Envelope>
```

図 10 SOS API　InsertObservation の要求

```
<?xml version="1.0" encoding="UTF-8"?>
  <soap:Envelope・・・>
    <soap:Header/>
    <soap:Body>
      <sos:InsertObservationResponse/>
    </soap:Body>
  </soap:Envelope>
```

図 11 SOS API　InsertObservation の応答

2-3(e). GetObservation を使った登録データの確認

図１２は、SOS サーバに登録されたデータ要求を行う API GetObservation の抜粋である。GetObservation では要求するデータを特定するために、様々な条件を課してデータの検索を行うことができる。①は、センサを指定している。指定されたセンサより測定されたデータを検索対象とする。②ではデータの項目を指定している。ここでは水位(water_depth)が検索対象となっている。③ではデータの検索範囲を時刻で限定している。④-⑤のように検索対象となる時刻の範囲を、開始時刻と終了時刻を指定して表す。⑥は SOS サーバからの返ってくるデータの書式を指定している。どのようなデータの書式が指定できるかは、図 7 中の⑫のように GetCapabilities の応答から読み取ることができる。

図１３は GetObservation の要求に対する SOS サーバの応答の抜粋である。図１２の⑥のように GetObservation の要求に記されたデータ書式に従ってデータが SOS サーバより返信される。①は、GetObservation 中の検索条件に適合したデータの記述がここからはじまることを表す。図１３に示された応答は、図１２の⑥の書式指定によって、データ一つ毎に、<sos:observationData>データ</sos:observationData>のようにして表される。ここでは検索条件に適合した水位データが一つだけ入っている。②-③は水位データの phenomenonTime と resultTime が示されている。図１２の InsertObservation 要求では、図１２の③のように phenomenonTime を 2001-01-10T00:00:00+09:00 と日本のタイムゾーン（JST）で指定しているが、応答は 2001-01-09T15:00:00.000Z と世界標準時（UTC）に変換されて返ってきている。④-⑤では、検索条件に一致しているセンサ名、データ項目が示されている。⑥では水位の値が 1.0cm であること示している。

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope・・・>
    <env:Body>
        <sos:GetObservation ・・・>
            <sos:procedure>http://www.tsuruoka-nct.ac.jp/test/procedure/1</sos:procedure>  －①
            <sos:observedProperty>http://www.tsuruoka-
              nct.ac.jp/test/observableProperty/water_depth</sos:observedProperty>  －②
            <sos:temporalFilter>  －③
                <fes:During>
                    <fes:ValueReference>phenomenonTime</fes:ValueReference>
                    <gml:TimePeriod gml:id="tp_1">
                        <gml:beginPosition>2001-01-09T00:00:00+09:00</gml:beginPosition>  －④
                        <gml:endPosition>2001-01-11T00:00:00+09:00</gml:endPosition>  －⑤
                    </gml:TimePeriod>
                </fes:During>
            </sos:temporalFilter>
            <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>  －⑥
        </sos:GetObservation>
    </env:Body>
</env:Envelope>
```

図 12 SOS API　GetObservation の要求

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope・・・>
     <soap:Header/>
     <soap:Body>
       <sos:GetObservationResponse>
         <sos:observationData>  －①
           <om:OM_Observation gml:id="o_2103">
             <om:type xlink:href="http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"/>
             <om:phenomenonTime>
               <gml:TimeInstant gml:id="phenomenonTime_2103">
                 <gml:timePosition>2001-01-09T15:00:00.000Z</gml:timePosition>  －②
               </gml:TimeInstant>
             </om:phenomenonTime>
             <om:resultTime xlink:href="#phenomenonTime_2103"/>  －③
             <om:procedure xlink:href="http://www.tsuruoka-nct.ac.jp/test/procedure/1"/>  －④
             <om:observedProperty xlink:href="http://www.tsuruoka-
                 nct.ac.jp/test/observableProperty/water_depth"/>  －⑤
             <om:featureOfInterest xlink:href="http://www.tsuruoka-nct.ac.jp/test/featureOfInterest/1"/>
             <om:result xmlns:ns="http://www.opengis.net/gml/3.2" uom="cm"
                 xsi:type="ns:MeasureType">1.0</om:result>  －⑥
```

図 13 SOS API　GetObservation の応答

```
            ・・・
      </soap:Body>
</soap:Envelope>
```

2-4. マイコンと SOS サーバ間の通信プログラム

図１４にフィールドセンサのハードウェアの構成図を示す。図１５に開発に用いている評価ボードを示す。図１５の評価ボードでは、センサ部(Arduino)とルータ部(BeagleBoneBlack)間の通信に UART を用いている。今回は、SOS サーバとルータ部との通信の検証のため、センサ部からのデータは使わずダミーのデータを使用して、ルータ部と SOS サーバ間で通信を行う。

BeagleBoneBlack は、安価であるがディスクトップマシンとしての必要最小限のハードウェアが実装されており、Arduino と異なり OS として Linux が用意されている。そのため、LinuxOS を搭載したパソコンで開発したプログラムを BeagleBoneBlack に移植することは容易である。

SOS サーバとフィールドセンサの間の通信を行う通信プログラムを Node.js 上で動作する Javascript 言語で開発する。Node.js は様々なプラットフォームに移植されており、Node.js 上で動作するアプリケーションは移植が非常に容易である。Node.js はインタプリタであるため、C 言語等のコンパイラ言語に較べて実行速度は遅くなるが、センサから上がってくるデータの速度が遅いため、Node.js の実行速度は問題とならない場合が多い。

Node.js 上で動作する通信検証用プログラムのソースファイルを図１６に示す。単純に SOS API を記述した XML ファイルを読み込んで、指定 SOS サーバに POST メソッドで送信するプログラムである。SOS サーバからの応答は標準出力に出力される。

本プログラムの実行例を図１７に示す。本プログラムはコマンドライン上から、図１０で示した InsertObservation の要求を表す XML 形式のデータを、InsertObservation.xml として XML ファイルにした。コマンドラインより送信テスト用プログラム Conncet_SOS.js に引数 InsertObservation.xml を渡して実行した。図１７中の赤字の部分が SOS サーバからの応答である。この応答は図１１と同じであることから、XML ファイル中に記述した InsertObservation の要求が SOS サーバに伝達されて、想定通りにデータのアップロードができたことを示している。図１６のような簡単なスクリプトであっても SOS API との通信の検証に十分に役に立つ。より複雑な動作や、XML の詳細な解釈が必要な場合は XML パーサの導入が必要と考えられる。

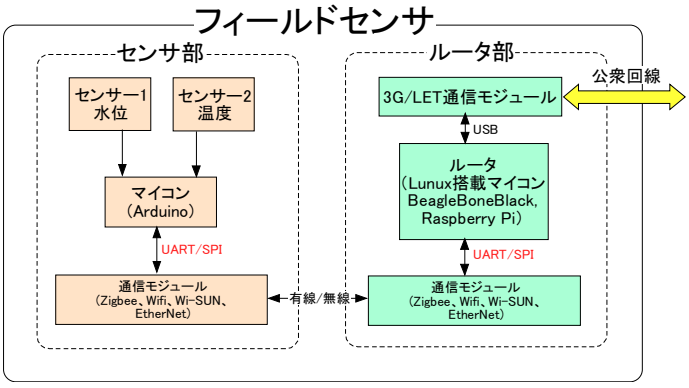

図 14 フィールドセンサ構成図

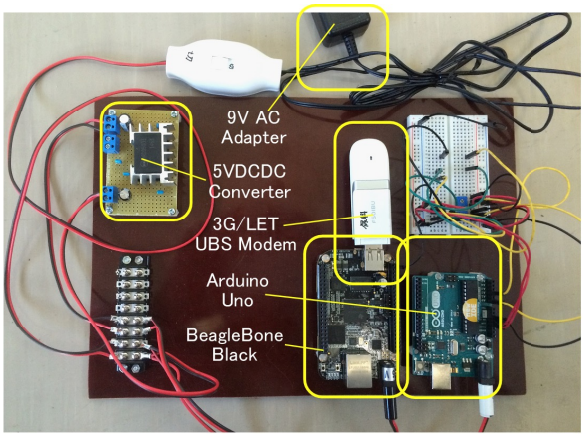

図 15　開発ボード

```
var fs = require('fs');              //ファイルアクセス用モジュール fs をロード
var request = require('request');    //HTTP クライアント用モジュール request をロード
var request_xml = fs.readFileSync(process.argv[2], 'utf8'); //引数で要求が記述されたファイルを渡す。
var headers = {
  'Content-Type'   : 'application/soap+xml' //要求に際して必要な HTTP ヘッダ GetCapabilities の応答に記述あり
};
var options = {
  url: 'http://10.8.90.12:8080/52nSOS/service', //SOS サーバの URL
```

```
  method  : 'POST',                              //送信方式 POST
  headers : headers,
  body    : request_xml                          //送信データ(xml 形式)
};
request(options, function (error, response, body) {  //HTTP POST メソッドでデータを送信
  console.log(body); //SOS サーバの応答内容を標準出力で表示
});
```

図 16　SOS サーバとの通信検証用プログラム(Conncet_SOS. js)

```
root@beaglebone:/opt/local# node Conncet_SOS.js InsertObservation.xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sos="http://www.opengis.net/sos/2.0"
xsi:schemaLocation="http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sosInsertObservation.xsd http://www.w3.org/2003/05/soap-envelope
http://www.w3.org/2003/05/soap-envelope">
  <soap:Header/>
  <soap:Body>
    <sos:InsertObservationResponse/>
  </soap:Body>
</soap:Envelope>
```

図 17　SOS サーバとの通信検証用プログラムの実行結果

3.　春日井市における水位計測システム

SOS 対応フィールドセンサの一例として、平成 28 年 3 月に春日井市勝川道風線地下道に、図１８のアンダーパス用水位計を設置した。装置の詳細は平成 27 度の成果報告(参考文献 9)に記述している。ここでは、運用状況と、実証試験から得られた検討課題について報告する。平成 28 年 3 月より運用を開始し、水位の測定データは中部大学を介して、春日井市の防災システムに組み込まれている。

図 18 アンダーパス用水位計の設置状況

平成 28 年 8 月 2 日に実際に道風線地下道にて冠水が生じた。そのときの地下道の監視画像を図１９に示す。図２０に、冠水した時刻を含む時間帯で測定された水位の変動の測定結果を示す。本アンダーパス用水位計には水位計の上部に監視カメラを設置しており、水位計周囲の状況を確認する際に用いられている。

図１９の監視カメラ画像では道路が冠水している状況が確認できる。水位計脇を通り抜ける車が水しぶきを上げて通り抜ける様子も確認できた。そのことから、監視カメラは周囲の状況の監視に十分に役立っていることが確認できた。

図 19 道路の冠水状況（8 月 2 日）

図２０において 16 時 20 分頃に急激に水位が上昇し、数分後今度は急激に水が降下している。道路の水位が上昇したため、アンダーパスに設置された排水ポンプが作動して排水をしたためと思われる。図１９で観測された冠水時刻と、グラフ上で冠水を観測した時刻がずれている。図１９の画面上の時刻はカメラ自体の時計で計時しており、その時刻を表示している。カメラの時刻調節をしておらず、本来の時刻とずれて表示されている。

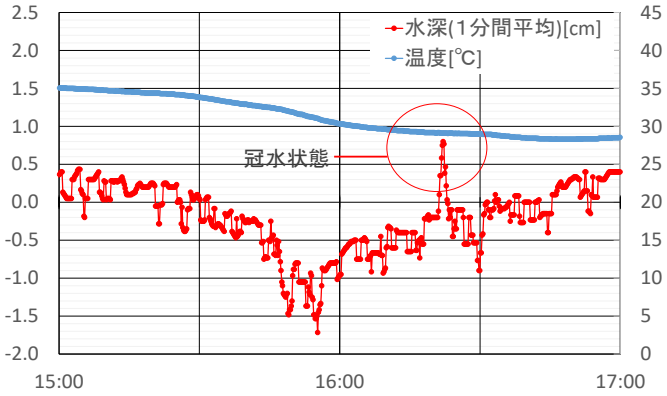

図 20 測定水位（8 月 2 日）

図２０より、急激に変化する水位を捉えることはできたが、冠水時以外でも水位が変動しているように観測されている。晴れている日にも同様に緩やかに水位が変動している様子が観測される。この緩やかな水位の変動は温度の変化と一緒に変動しており、温度の影響と考えられる。水位計の測定確度の向上のためには、この温度による、測定水位の変動を小さくする必要がある。

図２１は、平成 29 年 1 月 25 日中部大学国際 GIS センターにおいて催された春日井市との防災関連の会議での様子である。本会議に出席しアンダーパス用水位計について意見交換をすることができた。その席上で、市担当者から次のような意見をいただいた。1)水位計と一緒に設置しているカメラは非常に有効である。2)カメラの画像で現状を確認し、水位計の値は補助的な役割となっている。3)水位の測定精度は現状で十分である。課題として、1)水位の計測値に明らかな異常値が生じると誤報を発するので、異常値が発生しないようにして欲しい。2)カメラ画像は夜間では見えづらい。全般的には好意的な意見が多かった。平成 29 年度も水位測定システムを維持することで合意した。

図 21 中部大学における春日井市との防災関連会議

現在、水位測定システムのデータ送信先は、最終的に中部大学に設置した SOS である。その SOS サーバが Ver. 1.0 であることから、Ver. 2.0 のように直接 SOS サーバにデータをアップロードする事はできない。そこで、現在は中部大学の Web サーバ上で PHP スクリプトを起動して、SOS が参照している Data Base に直接データを書き込んでいる。

SOS 対応フィールドセンサの実証実験用に作成したアンダーパス用水位計であったが、思いのほか春日井市から高評価をいただき、有効に利用してもらうことになった。そのため、水位計は休みなく稼動し続けるために現状の維持が最優先とされている。その結果、SOS Version 2.0 への対応など大きな変更を加えることが困難となった。春日井市に設置した水位計を例として SOS 対応センサの開発を進めるのは難しくなったと思われる。今後は SOS 対応センサの開発は、センサ部と切り離して開発を進める必要があると思われる。アンダーパス用水位計自体は、春日井市に係わらず多くの地域の課題の解決に役立つものである。今後のアンダーパス用水位計の開発・改良が望まれる。

4. まとめ

本研究では、マイクロコンピュータを搭載したフィールドセンサから取得した情報を、OGC 標準の中に取り込むことを目的に、フィールドセンサの情報の出力を SOS(Sensor Observation Service)サーバに対応できるようにすることを目標とした。本報告では、センサ開発者の立場で SOS 対応センサを開発するための、SOS サーバとのデータ送受信の確認方法、および代表的な SOS API である GetCapabilities、InsertSensor、InsertObservation、GetObservation について、SOAP 表現の XML 形式データの具体例を示した。実際に今回紹介したツール、52° North SOS サーバ付属 Test Client や FireFox のアドイン HTTP Resource Test を使って API の送受信を確認した。SOS サーバとの送受信試験の成果を踏まえ、Linux 搭載マイコン BeagleBone Black 上で、Node.js を利用した簡単な SOS API 送信スクリプトを作成し、送受信が正しく行われていることを確認した。

近年の性能向上が著しいマイコンで十分に SOS クライアントとして動作できることを示した。現在の段階ではマイコンと SOS 間の通信の検証が終わった段階である。今後はセンサ～マイコン～SOS サーバと一続きとなる実証システムの開発が期待される。また、SOS にアップロードされたデータの有効利用のためにも SOS と GIS との連携についても課題である。

平成 28 年 3 月より、春日井市勝川道風線地下道で実施している。春日井市行政からは、本水位計システムを高く評価していただいた。約 1 年間の運用を通してアンダーパス用水位計の課題が挙げられた。今後は先にあげた課題の克服が必要と考えられる。

5. 謝辞

本研究は中部大学問題複合体を対象とするデジタルアース共同利用・共同研究 IDEAS201601 の助成を受けたものです。

6. 参考文献・データ

1. “Welcome to the OGC”, http://www.opengeospatial.org/ （2016 年 4 月 1 日）
2. 首相官邸 政策会議 高度情報通信ネットワーク社会推進戦略本部（ＩＴ総合戦略本部） 新戦略推進専門調査会分科会 取りまとめ等,
   http://www.kantei.go.jp/jp/singi/it2/senmon_bunka/nougyou.html (2016 年 4 月 1 日)
3. Arne Bröring, Christoph Stasch, Johannes Echterhoff, “OGC® Sensor Observation Service Interface Standard”, http://www.opengeospatial.org/standards/sos （2016 年 4 月 1 日）
4. North 52°, http://52north.org/ （2016 年 4 月 1 日）
5. CloudSence, http://cloudsense.com/ （2016 年 4 月 1 日）
6. OSGeo Live, https://live.osgeo.org/ja/ （2016 年 4 月 1 日）

7. 首相官邸 政策会議 高度情報通信ネットワーク社会推進戦略本部（ＩＴ総合戦略本部） 新戦略推進専門調査会分科会 取りまとめ等 別紙１１　環境情報のデータ項目 URI リスト.html, http://www.kantei.go.jp/jp/singi/it2/senmon_bunka/shiryo/agro-env.html （2016 年 4 月 1 日）
8. “Observations and Measurements - XML Implementation”, http://www.opengeospatial.org/standards/om （2016 年 4 月 1 日）
9. “普及型フィールドセンサによる OGC 標準でデータの災害情報サービスへの提供の研究”,文部科学省共同利用・共同研究拠点「問題複合体を対象とするデジタルアース共同利用・共同研究拠点」紀要、平成 27 年度成果報告,pp.17-20