International Environmental Modelling and Software Society (iEMSs)
7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA,
Daniel P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

Towards a low-cost, full-service air quality data archival system

Argyris Samourkasidis , and Ioannis N. Athanasiadis

Democritus University of Thrace
Xanthi, Greece
(argysamo@gmail.com, ioannis@athanasiadis.info)
http://eco.logismi.co

Abstract: We present our explorations towards a low-cost solution for creating an autonomous environmental data archival system. *AiRCHIVE* is a software platform for providing open access to sensor data with different ways, that account for machine interoperability. It is built with Raspberry Pi, a low-cost, pocket-size computer, and Air Pi a low-cost amateur sensory kit for air quality monitoring. Raspberry Pi with AirPi allowed us to easily capture raw sensor data and store them in a local database, and with *AiRCHIVE* we deployed "on sensor" a web server that provides with a set of services for data preprocessing and dissemination, including an implementation of OGC/SOS services and the OAI/PMH harvesting protocol.

Keywords: Environmental Data Archive; Sensor Observations; Sharing and Reuse; Semantic Web; Low-cost sensors; Open Archives; SOS; Metadata Harvesting

1 INTRODUCTION

Sensor networks provide with evidence not only to support science, but also for continuous surveillance, on-line decision making and citizen-centered services. At the same time, small-sized computers offer more capabilities. In sensor networks, the information is enhanced through a layered approach, that involves four steps: capture, storage, decision-making, and distribution [Athanasiadis and Mitkas, 2004]. The majority of environmental sensor networks consist of several sensor nodes that capture data from the environment, and a base station which gathers sensor data together (capture phase). Subsequently, data from different base stations are transferred to a centralized sensor network server (storage phase), so that data are further processed (i.e. through decision-making) and eventually become available over the World Wide Web or other means (distribution phase) [Harta and Martinez, 2006].

In this paper, we introduce *AiRCHIVE*, a system that provides the functionality of all four sensor layers in one credit-card sized "box", and comes with a respectively low cost. All sensor network functionality (sensor nodes, base station, and sensor network server) are implemented as services in *AiRCHIVE* for data capture, storage, decision-making (i.e. validation), and dissemination over the web. Furthermore, *AiRCHIVE* minimises communication costs, and employs standards for publishing environmental data. We adopted Sensor Observation Service (SOS), an *Open Geospatial Consortium* (OGC) standard to achieve interoperability [OGC, 2007], and built upon the Protocol for Metadata Harvesting (PMH), an *Open Archives Initiative* (OAI) standard for harvesting resource metadata over the web [OAI, 2002].

This work, contributes towards the vision of the *Semantic Sensor Web*, by adding semantic annotations to sensor data, which could be complemented with ontologies for interoperability, analysis, and reasoning over heterogeneous multimodal sensor data in order to provide enhanced meaning for sensor observations [Sheth et al., 2008]. Similar efforts of the *geospatial sensor web* community, where heterogeneous sources are made available as services [Chen et al., 2014]. Also, adds to the vision of *Semantic Modelling*, where

shared environmental resources are shared in ways that make model meanings explicit and meaningful, and automatic associations possible. Semantic annotation of environmental datasets and models is a prerequisite in order to explore the potential benefits of semantically explicit environmental modelling [Villa et al., 2009].

2 Background and Related work

2.1 Environmental sensor integration

Environmental sensor integration and management is an important component of environmental modelling. It was early noted that environmental monitoring is required to protect the public and the environment from toxic contaminants and pathogens that can be released into a variety of media. Thus accurate, inexpensive, long-term monitoring of environmental contaminants is needed using sensors that can be operated on site or in situ [Ho et al., 2005]. Undoubtedly, easy access to sensory information is of great added value and for a variety of users. Though, only recently, Mason *et al* [2014] presented with a centralised tool for managing, archiving, and serving point-in-time data in ecological research laboratories, that employes a service oriented architecture.

In this work, we aim to offer services for accessing sensory information, by providing a low cost solution that may run on sensor. We adopted SOS and O&M terminology, according to which an *observation* is an act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property and it has a *result*, which is an estimated value of an *observed property*. A *procedure* is used in making an *observation* and the result value is generated by a method, an algorithm, an instrument, a sensor, or a system. As an example, considering the procedure of observing temperature with an analog thermometer, the result may be $18°C$.

2.2 Raspberry Pi

The Raspberry Pi is a credit-card sized computer, equipped with a 700 MHz ARM processor, that weights 45 g. The Raspberry Pi 'Model B' has 512 MB of RAM, and Ethernet controller for wired internet and two USB ports. Instead of a hard disk it uses an SD card. It is also equipped with 26 GPIO pins (General Purpose Input Output) for connecting various low-level peripherals, and several manufacturers have released interface boards that are connected via the GPIO pins.
Raspberry Pi operates with Debian and Arch Linux ARM distributions, and programming tools are available for Python, which is the main programming language. It also supports several other popular programming languages, including Java and C/C++ [Upton and Halfacree, 2013]. It costs about 50 USD.

2.3 AirPi kit

The AirPi is an interface board for Raspberry Pi, that connects over GPIO pins and is equipped with amateur, low cost sensors of basic air quality and weather information. AirPi captures carbon monoxide, nitrogen dioxide, temperature, humidity, air pressure. It is accompanied with software libraries in Python for programming sensors and uploading sensor data to the internet. Similar two Raspberry Pi, the AirPi follows the open hardware philosophy, and it can be further extended with other sensors, including a GPS module [Dayan and Hartley, 2013]. It costs roughly 90 USD including the sensors.

3 System specification

Our goal with this work was to demonstrate that a sensor node can become an active archiver of its own recordings. Instead of being just a capturing device that pushes data on the web, we demonstrate that current low-cost hardware is powerful enough to provide archival services "on sensor". Thus, we assembled together a Raspberry Pi computer with the AirPi kit for sensing air quality and weather data, and developed a software system called *AiRCHIVE* which is effectively a web server running on Raspberry Pi and provides

with added-value services for making available sensor observations at real time, while it also operates as a permanent data storage of sensor data.

In order to demonstrate interoperability, we adopted the Sensor Observation Service (SOS) standard [OGC, 2007]. SOS defines a Web service interface which allows querying observations, sensor metadata, as well as representations of observed features in heterogeneous sensor systems.

3.1 User types and key requirements

User types. We identify three different types of users in *AiRCHIVE*.

- First is the system *owner*, who has full access both locally and from internet via SSH. As a system administrator is able to update system software, or restart the device.
- Second comes the *web user*, who has access to the system through a public webpage. The *web-user* submits queries to the system, using the Graphical User Interface (GUI), which responds with a visual representation of *observed properties*. He may be interested in current (real time) measurements, or historical ones.
- Third are the software agents that may contact the system as data harvesters. They interact with *AiRCHIVE* using different vocabularies and protocols to submit requests. One may follow the SOS protocol for sensory measurements, while another could use the *OAI-PMH*. They interact with the system with RESTful web-services over the `http` protocol.

Key requirements. *AiRCHIVE* works as a self-contained station for sensory data dissemination and archival that serves both real time access and long term storage of sensory data. We demonstrate the system with a low cost sensor node made by assembling the AirPi kit with Raspberry Pi. *AiRCHIVE* collects air quality measurements and stores them as *observed properties* in a time-stamped database. Subsequently, it makes them available over the web. At the same time, it has the capacity to perform Quality Assurance and Quality Control operations on the data collected. While in this work we dont discuss the quality assurance process in detail, we envision a hybrid expert system similar to those we have developed in the past [Athanasiadis et al., 2010; Athanasiadis and Mitkas, 2004]. While such automated systems almost always involve some kind of interaction with experts, this is considered an external process to the current system.

Any *web user* may access *AiRCHIVE*'s sensory data and perform queries. *Web users* may interactively explore data, and render them as graphs in their web browser, or download the corresponding datasets as files (i.e. in Comma Separated Value (CSV) format).

AiRCHIVE offers query services for harvesters that adopt either SOS or PMH standards. Implemented as RESTful services *SOS* queries return responses in O&M format. Queries in *OAI/PMH* may respond with one or more metadata profiles.

3.2 Abstract architectural design

AIRCHIVE is composed of five main components each providing different services, shown in Figure 1.

First comes the **data capture component** that actually collects sensed measurements from the sensory device. In our case, it interfaces AirPi sensors via the GPIO to Raspberry Pi, and contributes the result. This component is custom to hardware and sensor requirements, however the general behaviour remains the same: at regular time intervals it acquires the result from the sensor and makes it available to the system.

The result of the property observed by the sensor is subsequently stored in the **Database component**, along with a time stamp. The database component is also able to respond to queries.

A **data validation component** (optional) may intervene between *data capture* and *database* components. Its role is to apply some quality assurance/quality control process and identify hardware or sensor errors. Additionally, it could associate the measurement with a quality flag, by applying rules, or more empirical procedures (statistical, data driven, etc) [Athanasiadis et al., 2010]. Such a component is essential for ensuring data reliability and user confidence.

The **data processing component** is an intermediate layer between the relational database and the web server. It transforms arguments (submitted by users/harvesters with their queries) into appropriate database queries. It also works in the vice-versa manner, as it transforms database outputs into the format requested by the user.

Last, but not least, the **Web Server** serves three different user types. It provides the GUI for the human web user to submit queries for *observed properties*, and visualise graphs. As a Sensor Service server, interacts with harvesters submitting their queries in the SOS standard. As an PMH/OAI server disseminates preprocessed data on the Semantic Web, after a PMH harvester submits its query.

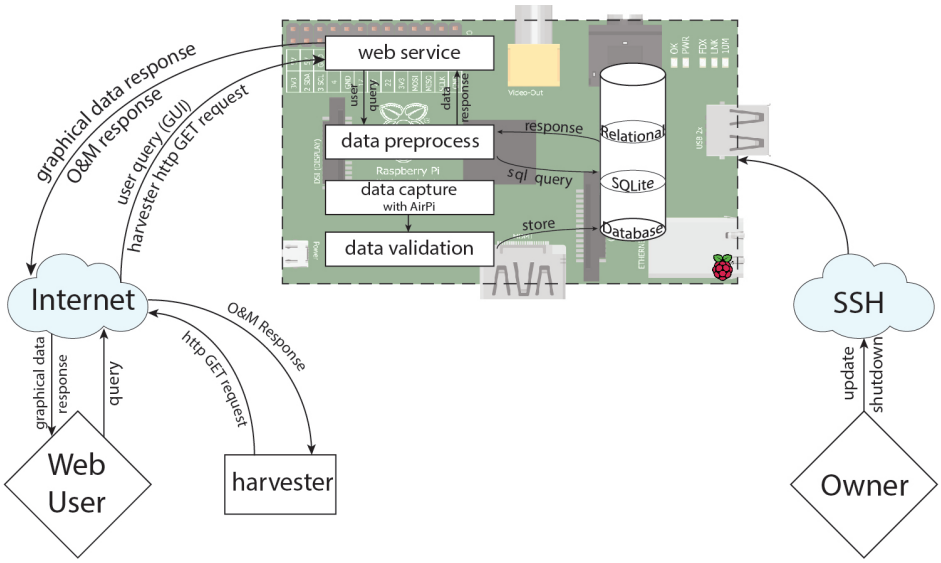

Figure 1. Abstract architecture

4 DETAILED DESIGN AND IMPLEMENTATION

4.1 System components

AiRCHIVE system components were implemented as services using Python. In the current status of implementation, we have not deployed the data validation component yet.

Data capture and Database components. AirPi comes with a Python module, which enables (using i2c and SPI buses) to read the sensors' results, print them on the system console, and upload them on a webserver. Based on this module, we developed a Python module, that stores sensor data in an time-stamped SQLite database. We also set the frequency that *measurements* are taken to be every five minutes.

The SQLite database stores primary data in a single table of six columns: `Timestamp`, `Carbon Monoxide`, `Nitrogen Dioxide`, `Temperature`, `Relative Humidity`, and `Noise`.

Data processing service. This service is the business logic layer that intervenes between database and web service. It is a Python module (implemented by us), which takes all arguments that a harvester agent

submits within its `HTTP GET` request, or a *human user* from GUI. Subsequently, these arguments are transformed into an SQL command, and are submitted directly to the database, using the SQLite Python library. This module also renders database outputs appropriately in the format requested by the harvester. When the harvester does not specify a *response format*, it applies the default formatting for every *observed property*. Data preprocessing service default response is in JSON format, which is the requirement of the the web service GUI.

Web service. The web service transforms *AiRCHIVE* to a Sensor Observation Service server and an OAI repository at the same time. It also provides graphical representations of the *results* for user requests. Web service handles *SOS* or *PMH* harvesters' arguments that are submitted through their queries.

In order to provide static and dynamic content, as web services in *AiRCHIVE*, we used Flask [Grinberg, 2014], a microframework for Python, that supports both for RESTful services and a template engine. Flask renders HTML templates using user arguments. We have developed the appropriate templates, so that *AiRCHIVE* is able to respond to SOS or OAI requests with *observations* that are stored in the database. Also, we developed templates for rendering the webpages of the system that enable user interaction. We also employed FLOT [Laursen and Schnur, 2007], a JavaScript plotting library for jQuery, for creating data graphs on the fly.

5 Installation and Operation

5.1 Installation

The *AiRCHIVE* implementation of both data capture and web services operate as Linux daemons, which start to operate after system boots. For minimising energy consumption while ensuring reliable result measurement, we decided to set the system to capture data every five minutes. The data capture service is triggered twelve times per hour. This is a design choice made based on the needs of the current system, while we have tested its operation with much smaller sampling frequencies. The data processing service is called when a user or a harvester submits a request to the web service, and responds with the corresponding result. Since Raspberry Pi runs on Linux distribution, the system owner may connect to it via Secure SHell (ssh) and update it, get a backup of the database or shut it down for energy or security reasons.

5.2 Providing services to harvester agents

Harvesters can submit queries to *AiRCHIVE*, which responds with formatted data. A harvester is a client application that issues *PMH* or *SOS* requests. Queries are expressed as either `HTTP GET` or `POST` methods. `POST` has the advantage of imposing no limitations on the length of arguments [OAI, 2002]. As we support for two types of harvesters we decided to allocate a different base URL for each. All requests consist of a list of keyword arguments, which take the form of `key = value` pairs. Arguments may appear in any order and multiple arguments must be separated by ampersands (`&`). A common `key` for both types of harvesters is `verb`.

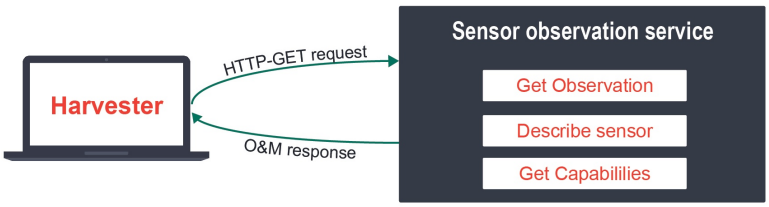

Figure 2. Harvester `HTTP GET` request

Use case 1: Machine interoperability with SOS.
Base URL for a SOS request is airchive.logismi.co/sos. The SOS core requirements class defines three operations and the corresponding verbs made available are:

- `GetCapabilities` is used to access to metadata and detailed information about the available operations.
- `DescribeSensor` enables querying of metadata about the sensors and sensor systems that are available.
- `GetObservation` is meant for accessing observations (`observedProperty` key) by selecting a sensor by a Uniform Resource Name (`Procedure` key), time duration (`eventTime` key) and the response format (`responseFormat` key).

For example, a typical `GetObservation` query is:

`http://airchive.logismi.co/SOS/?request=GetObservation&offering=Airchive:DUTH:DHT22 &observedProperty=Pressure&eventTime=2014-06-16/`

Use case 2: Machine interoperability with PMH.
Considering the Protocol for Metadata Harvesting, we adopted four operations. Available PMH verbs are:

- `Identify`: This verb is used to retrieve information about our repository. *AiRCHIVE*s' response includes: Repository base URL (`baseURL`), the earliest time that repository has records (`earliestDatestamp`) and administrators' email address (`adminEmail`).
- `ListIdentifiers` is used to retrieve available datasets hosted on *AiRCHIVE*. A typical request is :

  `http://airchive.logismi.co/oai/?verb=ListIdentifiers`
- `ListRecords` is used to harvest records (*measurements*) from a specified by harvester, sensor (*identifier*). Harvester can use optional arguments in its query, to permit selective harvesting of records based on time stamp. These optional arguments specify a lower (`from`) and an upper (`until`) bound for timestamp-based selective harvesting. The response format of the requested metadata can be specified with `metadataPrefix` key. When response's length exceeds 100 records, *AiRCHIVE* sends a `resumptionToken`, which is the flow control token returned by a previous `ListRecords` request that issued an incomplete list. `resumptionToken` can be used as an optional argument.
  A typical *ListRecords* request is :

  `http://airchive.logismi.co/oai/?verb=ListRecords&identifier=oai:Airchive:DUTH: DHT22&metadataPrefix=oai_dc&from=2014-03-29&until=2014-03-30&resumptionToken= L20140329R01`
- `ListMetadataFormats`: *AiRCHIVE* returns a list with the metadata formats that can be disseminated from our repository. We use an optional argument, `identifier`, that specifies the unique identifier of the item for which available metadata formats are being requested. If this argument is omitted, then the response includes all metadata formats supported by this repository.

5.3 Dissemination over the web

Use case 3 : Human user access via the webpages.
Web users submit queries to *AiRCHIVE* using the graphical user interface (GUI). Specifically, a user may select from a dropdown menu the *ObservedProperty* she is interested in, set a `temporalFilter` by specifing the time-window, and then the requested data are visualised with the help of FLOT framework. Additionally, those data can be extracted as comma-separated values (CSV) file.

Figure 3 shows the GUI response, when the user selects to visualize Carbon Monoxide concentrations for the period of the iEMSs conference (June 16-19, 2014).

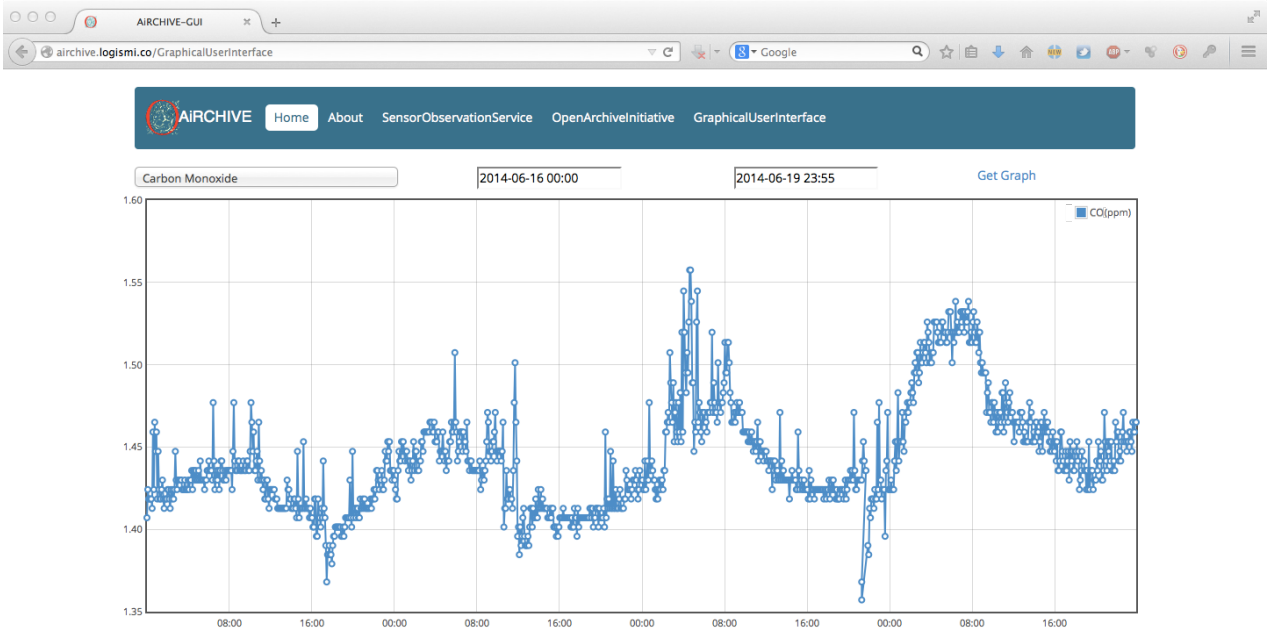

Figure 3. The GUI visualization. Using the FLOT framework, the user may create a graph to visualize Carbon Monoxide recordings for the period of the iEMSs conference (June 16-19, 2014).

6 Future Research

In this paper we presented a self-contained, low cost implementation of a sensor node that is able to archive data coming from a sensor network and make them available as services using standards, or over the web to end-users. Our experiments so far showed that while low cost sensors may not be very reliable and possibly not suitable for scientific experimentation, mini-computers are absolutely capable of serving those data over the web and provide with value-added services. While standardisation remains a challenge for ensuring data interoperability, web access is still relevant. We implemented both services with state-of-the-art tools, to experience only that the value of such a system is hindered only by network availability and sensor manufacturing quality.

Currently, we have identified as set of open issues to address in the near future. First is the selection of the datasets that are made available as resource under OAI/PMH. Currently, we offer monthly reports with summaries, but his could be easily extended. The dynamic nature of these services reveals the second issue, that of unique identifiers for sensor recordings and derivative products. Another issue is energy efficiency, as such nodes may are installed in remote locations and depend on less reliable energy sources. Future work could focus on optimising software to be less energy demanding. Another issue has to do with the fact that Raspberry Pi doesn't have a system clock as this would require a battery to power it. Time synchronises when Internet is available. This raises the issue on how to time-stamp *observed properties*, when Internet connectivity is not available. Finally, the data validation component is something we didn't work so far, despite the fact that sensors are prone to errors, and should be annotated with data quality attributes. However this is mostly a methodological problem and there are not significant challenges from an implementation point of view. Future work will focus to further extend *AiRCHIVE* and its functionality, and having the software tested, we will offer it under an open source license.

The current installation of *AiRCHIVE* is available online `http://airchive.logismi.co`.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013_FP7-REGPOT-2010-1, SP4 Capacities, Coordination and Support Actions) under grant agreement No 264226 (project title: Space Internetworking Center-SPICE). This presentation reflects only the authors views and the Community is not liable for any use that may be made of the information contained therein.

REFERENCES

Athanasiadis, I. N. and Mitkas, P. A. (2004). An agent-based intelligent environmental monitoring system. *Management of Environmental Quality*, 15(3):238–249.

Athanasiadis, I. N., Rizzoli, A. E., and Beard, D. (2010). Data Mining Methods for Quality Assurance in an Environmental Monitoring Network. In Diamantaras, K., Duch, W., and Iliadis, L., editors, *20th Intl Conf on Artificial Neural Networks (ICANN 2010)*, volume 6354 of *LNCS*, pages 451–456, Thessaloniki, Greece. Springer Verlag.

Chen, N., Wang, K., Xiao, C., and Gong, J. (2014). A heterogeneous sensor web node meta-model for the management of a flood monitoring system. *Environmental Modelling & Software*, 54:222 – 237.

Dayan, A. and Hartley, T. (2013). AirPI Air Quality and Weather Project. Available online: `http://airpi.es`.

Grinberg, M. (2014). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc.

Harta, J. K. and Martinez, K. (2006). Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191.

Ho, C. K., Robinson, A., Miller, D. R., and Davis, M. J. (2005). Overview of Sensors and Needs for Environmental Monitoring. *Sensors*, 5:4–37.

Laursen, O. and Schnur, D. (2007). FLOT: Attractive javascript plotting for jquery. Available online: `http://www.flotcharts.org`.

Mason, S. J., Cleveland, S. B., Llovet, P., Izurieta, C., and Poole, G. C. (2014). A centralized tool for managing, archiving, and serving point-in-time data in ecological research laboratories. *Environmental Modelling & Software*, 51:59 – 69.

OAI (2002). Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Available online: `http://openarchives.org`.

OGC (2007). Sensor Observation Service, Open Geospatial Consortium Standard. Available online: `http://www.opengeospatial.org/standards/sos`.

Sheth, A., Henson, C., and Sahoo, S. (2008). Semantic Sensor Web. *IEEE Internet Computing*, 12(4):78–83.

Upton, E. and Halfacree, G. (2013). *Raspberry Pi User Guide*. John Wiley & Sons.

Villa, F., Athanasiadis, I. N., and Rizzoli, A. E. (2009). Modelling with knowledge: a review of emerging semantic approaches to environmental modelling. *Environmental Modelling and Software*, 24(5):577–587.