Data Collector Service – Practical Approach with Embedded Linux

M. Saari*, P. Sillberg*, P. Rantanen*, J. Soini* and H. Fukai**

*Tampere University of Technology/Department of Software Engineering, Pori, Finland
**Keio University/Faculty of Policy Management, Fujisawa-shi, Japan

mika.saari@tut.fi

Abstract - Nowadays embedded systems are one of the most important application areas in information technology. Embedded systems are often used in life critical situations, where reliability and safety are more important criteria than performance. This paper presents a data collector service that has been developed based on embedded Linux, which operates as a key element in a larger intelligent alarm system. The target of this study was to test out how well a cost-efficient single-board computer could be used to gather sensory data, and how this data can be provided for the client over the public Internet. The paper describes the data collector service currently in use and its functionality and also gives a concrete example of how to utilize a microcontroller with an embedded Linux distribution. The paper presents one solution on how to utilize embedded systems for managing and controlling conditions in buildings and also environmental conditions in a smart and cost-effective way.

I. INTRODUCTION

Sensors are commonly utilized components including various kinds of warning and alarm systems. With the huge development of sensor technology, it has been possible to create minuscule, reasonably priced components, sensors and controllers with ultra-low power consumption. These kind of components are enabled for the rapidly improvement of the sensor networks and therefore it can be seen how sensor network implementations have been applied in numerous different fields of operation. A sensor network consists of single sensors, with the purpose of sensing the surroundings and to forward the collected data. There is a great variety of sensors available and their classification is based mainly on the features of the information collected. These kinds of physical features include temperature, humidity, brightness or air pressure.

In this research, one particular subject was to sense the condition changes in indoor spaces and data collection and entry related to their authentication as well as transmitting them forward. The objective of the research was to construct an automatic service – i.e. a data collector service – for collecting condition data that would, in turn, enable the collection and transmission of condition data for a backup system that exploits and analyzes measurement data as flexibly as possible in real-time. The starting point was the development of a system optimally attending to an independent, selected backup system for collecting, recording and transmitting condition data. The aim was to study and develop a solution that was as simple, reliable, cost-effective and easy to maintain as possible for the defined purpose. In this case, it was decided to implement the solution by adapting a microcontroller with embedded Linux for data collection and distribution. Embedded systems usually adopt embedded Linux as the operating system because of the numerous economic and technical benefits – the Linux kernel sources are well structured so that CPU-specific code is easy to find and is minimized. A prototype for realizing the service in question, which will be presented in detail in this paper, was created during the KiiauData research project in 2014.

The study presented here is part of the intensive collaboration between the Tampere University of Technology (TUT) in Finland and Keio University in Japan. The Global Environmental System Leaders (GESL) Program, ongoing in Keio University’s Shonan Fujisawa Campus (SFC), and the Alert system for detecting anomalous situations developed as one of its outcomes serve as the background of the research. One main part of the alert system under development is the data collector service which was developed in collaboration by the university partners in TUT Pori, enabled by the ongoing KiiauData (Smart analysis of property systems data) project. One of the main aims of this two-year (2013-2014) project, funded by TEKES [1], was to study potential new technologies for managing and controlling conditions in buildings in a smart way. The expected results of the project will enable providers of products and services in the built environment to form wider and more automated solutions both for new breakthroughs and recognized problems in the smart built environment. The data collector service presented here is one concrete example of the studied and piloted solutions produced during the joint project.

Related research in this specific area – i.e. utilization of a microcontroller with embedded systems – has been conducted by Rakesh et al. [2], for example, who have introduced a system which implements an embedded system for monitoring wireless sensor nodes and a camera installed inside a building for security surveillance. Toshniwal and Conrad [3] have studied how to make a cost-effective network-based sensor monitoring system which is portable for various applications. They have developed Linux-based systems based on desktop architecture with a sensor package, and also another system which used an embedded Single Board Computer.
(SBC) together with sensors. In addition, Cheng and Shen [4] have introduced a wireless sensor network communication terminal based on embedded Linux. Voinescu et al. [5] describe a device which can work as a network connection to a single board computer (BeagleBone or similar), where the target was to make an easy-to-use wireless networking device. Sawant et al. [6] studied a device that is capable of making file manager operations with two USB flash drives. Their study gives basic knowledge of using an ARM-based embedded Linux and touch screen. Banerjee et al. [7] proposed and implemented the design of a secure sensor node prototype. They built the prototype using a single board computer (Raspberry Pi in this case), accelerometer, and Bluetooth dongle. The above-mentioned studies deal with the same research area and have a very close connection to the specific research topic presented in this paper.

The following section (Section 2) briefly describes the background system and Section 3 gives a detailed explanation of how its first part – the data collector service – has been carried out. Section 4 includes a discussion and suggestions for future research on the topic and finally section 5 summarizes the study.

II. BACKGROUND – THE INTELLIGENT ALERT SYSTEM

The basis of the study is the alert system under development in Keio SFC. With this planned system it is possible to collect environmental data for different purposes. The system will make environmental sensing easier in various places by using small sensors, and there are many ways to utilize this collected data. One of the intended applications for using the system is for detecting anomalous situations. The aim of such a system is to handle environmental sensor data collected from multiple locations. In practice, it is not easy to understand the problems inherent in a given place just by looking over a graph of sensor data. Therefore, an alert system is need for interpreting the environmental changes in the space in question and associated problems.

This intelligent alert system consists of three (3) main parts shown in Fig. 1: the first part is the collection of environmental sensor data via the new data collector service. The second part is the detection of anomalous situations by utilizing the sensor data. The third part is sending the alert to where the situations were detected.

This particular system targets indoor spaces utilized by the public such as offices, meeting rooms, stations, trains, etc. In this case, the variables being sensed include temperature, light, and humidity. The system automatically collects the sensor data from sensors placed in various places, and analyzes changes over time. As environmental sensor data is collected for a given place over a long time-span, the results become more useful, not only by evaluating the results at one location but also by comparing the results from the sensor data from many locations. Each type of sensor data has four (4) feature values: location information, time, data type, and sensing value. The alert system registers the collected sensing data with an active database system in real time. The active database (upper left corner in Fig. 1) automatically reacts in response to detected state change rules that have been pre-defined by the users of the alert system. If an anomalous situation is detected as specified in the active database’s rule set, the database system sends alerts to the anomalous location. In this case, the role of the active database is to support rule definition, compare data between sensors, compare similar data at different time-points, and compare between different sensor types.

This paper deals with the first part of the alert system, i.e. the data collector service (lower left corner in Fig. 1), and does not describe the Intelligent Alert System as a whole. The paper presents the features and architecture, hardware and software components of the developed service, and also the physical connections between the components. The structure of the developed software used for data collection is also described.

III. DATA COLLECTOR SERVICE – IMPLEMENTATION

The architecture of the Data Collector Service is shown in Fig. 2. The purpose of the service is to measure light, temperature, and humidity. The sensors used are shown in Fig. 2 (right side). The data can be collected continuously, up to six times a minute. The collected data is then provided as a service for clients, who access the service using the Internet (left side of Fig. 2). In our use case the clients can also be other service providers.

The physical device itself is placed in a public space – TUT’s laboratory in this case – to collect data. Attention had to be paid to the physical size of the device as a device smaller than an ordinary PC is easier to install in a public area. In this implementation, the device must also have Ethernet or Wi-Fi capabilities in order to be remotely accessible. These two requirements lead to the use of a device with embedded Linux. The embedded Linux in this context means Linux that can run on ARM-based processors. The embedded Linux devices often have database and web server capabilities, or they can be easily added afterwards. The size of the device’s internal mass memory is not critical as long as the device has peripheral ports for external flash memory or Secure Digital (SD) cards.

The integral component of the Data Collector Service, the BeagleBone Black [8], is a low-cost, high-expansion focused SBC using an ARM Cortex-A8 based processor. It can host a Linux operating system and has a 10/100 Ethernet connection and a microSD connector, with 512
MB system memory and 2 GB of embedded MultiMediaCard (eMMC) memory.

The BeagleBone Black has two expansion headers, labeled P8 and P9, which allows the integration of BeagleBone electronics projects [8, 9], and in this research the features of these expansion headers were used. The sub system developed utilizes GND (Ground), 3.3V and 1.8V power, two GPIO (General Purpose Input/Output), and AIN (analog input) pins to drive light, humidity, and temperature sensors.

A. BeagleBone Black with Embedded Linux

The BeagleBone board comes with a pre-installed operating system called Ångström Linux, which is a stable and user-friendly distribution for embedded devices and is categorized in the Embedded Linux category. [10] Despite being designed for embedded devices, the Ångström Linux has many of the capabilities that can be found in other full-fledged Linux distributions, such as the X11 windowing system and a substantial amount of software packages in its package repositories.

Some basic Linux server hardening configurations were made to the Ångström Linux for more secure network operation. For example, the system time management was changed to use ntpdate [11], the X11 service was disabled, direct root user access was removed and a new basic user was created which could be used for remote Secure Shell (SSH) access. Python programming language was used together with an Adafruit-BeagleBone-Io-Python library, to utilize the I/O operations of the sensors [12].

The Data Collector Service itself does not utilize the data it collects, as this was to be done by a remote computer with more computing capacity. Thus the Data Collector Service only collects the data and serves it over the Internet for use or as input to the next part of the alert system. We chose to store the data to persistent memory so it could be later accessed by one or more clients. In this case there were no special requirements on how, when and how often the data should be delivered to the remote computer, so the decision was to implement relatively simple server software utilizing the client-server architecture. This way the consumer of the sensor data can decide the most convenient update cycle.

The Data Collector Service provides the data to the clients over a representational state transfer (REST) HTTP/GET [13] interface. Clients may access the interface for historical data from a chosen time interval, or if they choose to, poll periodically for the newest data. By using a short enough polling interval, it would be possible to get near real-time data from the service within the limitations of the computing performance of the BeagleBone platform. The number of new data points per minute would be limited by how often the sensors are read on the Data Collector Service.

The server software was deployed on an Apache Tomcat web server [14] and the sensor readings stored in a MySQL database [15]. In Fig. 2, these two components are called Web Service and Database, respectively. A third major software component called SensorApp also runs on the BeagleBone Black platform. The task of the SensorApp is to communicate with the Expansion headers which are used to drive the physical sensors attached to the BeagleBone Black.

The SensorApp and the Web Service running on top of the servlet container were written by the project team. The Database and Servlet Container were written by a third party. Ångström's own package manager provided MySQL, while Apache Tomcat and Java Virtual Machine (Oracle’s Java in this case) were installed using their latest available installation packages for Linux operating systems.

Fig. 2 also illustrates the directions of data flow which happen between the components. The Web Service can only read data from the Database, while the SensorApp has read and write access to the device’s GPIO headers in order to operate the sensors, and it also directly writes the collected data into the Database. For the completeness of the REST interface, the full CRUD (Create, read, update and delete) operations could have been implemented through the interface, but that would have increased the total code complexity of the system. In addition, it would have raised security issues, such as the malicious removal or modification of the collected data. Unauthorized access could be mitigated by the use of access control such as passwords or certificates. In the end, the prime interest was on collecting the data, so keeping outside access as "read only" was the most effective method in terms of computational capacity and code complexity.

Fig. 3 shows the architecture of the physical device with the sensors attached. BeagleBone expansion header
P9 was used to connect the sensors – a photoconductive cell (NSL-19M51 [16]) and humidity and temperature sensor (SHT11 [17]). The photoconductive cell is connected by using a typical application circuit [18]. The humidity and temperature sensor is connected to the BeagleBone by using a datasheet application circuit [17].

B. System Functionality

The web service provides the collected data through one read-only interface. The interface can be accessed by using a simple HTTP GET request. The default query without any parameters returns the ten latest data points. By using different parameters (such as begin_date, end_date, limit and paging) one can request a desired data set from the service. Fig. 4 shows the sequence of accessing the data from the web service. The Client connects to the device’s Web Service interface, which in turn retrieves the requested data from the Database. The data is then marshaled into XML format by utilizing JAXB (Java Architecture for XML Binding) annotations and sent back to the client.

As the data is stored to the device’s persistent memory, the data can be requested when needed and as often as needed. However, due to the limited computing resources of the platform, a request for a large data set may take a long time or even fail. The code for the web service was not specially written nor optimized for this use case, but merely as a generic proof-of-concept implementation. The marshaling of the XML output at least could have been done in a more memory-efficient way, or a different approach such as JSON serializing could have been used instead. An often-used paradigm is to limit the maximum measurement count to a known safe figure, and use the paging parameter to retrieve the rest of the results. The current implementation appears to cap at around 40000 measurements (or about five days of collected data) on the device, while a desktop machine could have been used instead. An often-used paradigm is to limit the maximum measurement count to a known safe figure, and use the paging parameter to retrieve the rest of the results. Because of the limited memory capabilities of the BeagleBone platform.

The sensor data is collected in a separate process to the web service. Fig. 5 is an illustration of one loop of the sensor reading process. Each loop corresponds to a single measurement point. The humidity and temperature sensor used has its own built-in circuit, and the measurements can take up to 80 or 320 milliseconds at the default accuracy (12 bits for humidity and 14 bits for temperature). Also, in order to avoid excess self-heating of the sensor, a maximum of one measurement per second at 12 bit accuracy should be made. [17] These limitations set a theoretical maximum of 15 measurements per minute at default accuracy (when the communication with the sensor is not taken into account). When a lower accuracy (20ms/8bit and 80ms/12bit) is used, a maximum of 60 measurements per minute can be achieved. For this application, it was decided to use the default accuracy and an interval of six measurements per minute.

As can be seen in Fig. 5, first the application reads the raw temperature value for the combined humidity and temperature sensor, and then pauses for a while to let the sensor cool down, after which the raw humidity value is read from the same sensor. Then the raw value from the photoconductive cell is read. The raw signal values have to be converted before they can be stored to the database by using the conversion formulas provided by the manufacturer. After the database has been updated, the application enters sleep mode to attain the desired measurement interval.

IV. DISCUSSION & FUTURE RESEARCH

This paper presents a data collector service, which utilizes BeagleBone Black development board with an embedded Linux distribution. The goal was to experiment how well a cost-efficient SBC can be used to gather sensory data, and how this data can be provided to the client over the public Internet. This goal was reached successfully, and the designed system was tested and found to work as planned. Nevertheless, the development process raised several improvement ideas, which could be realized in the future.

One of the issues is the packaging of the sensor system. The current version was a prototype version, and consequently the focus was on making the system functional, both by testing the sensor connections and readings, as well as benchmarking the functionality of the REST API and the software components. This limited the practical usability of the system, for example, making it unfit for use outdoors. In addition, the current software consists of various libraries, programming languages and components, and is as such slightly tricky to install. The software components could be packaged into a single application for easier installation. There is also the possibility to release the source code as an open source release.
The chosen sensor components could also be improved. The serial interface of SHT11 offers good power efficiency, but it cannot be addressed by standard I2C (Inter-Integrated Circuit) protocol, which would make programming tasks easier. In this use case the system is always provided with a continuous power supply and thus better programmability would be a major asset in future studies. In our case, the limited amount of sensor devices to be deployed makes the cost of the single board computer largely irrelevant, though it should be noted that there are other SBCs that are slightly cheaper, but still offer reasonable computing performance. One popular choice is the Raspberry Pi (e.g. [19]), which would work – specification-wise – equally well in this use case. Both of these boards offer excellent extension capabilities and can be expanded with additional sensors. The addition of multiple sensors raises another issue, which has not been studied in this research. It is unclear how well the board and the developed system would cope with a very large number of sensors. Also, in this system, all sensors are located very close to the actual board, and thus, the signal degradation can be thought negligible, but this is not necessarily true in large-scale monitoring systems. One example of this kind of system would be the monitoring of an entire apartment complex, where sensors are physically located very far apart and connected to the board by long wires or cables. In this case, it would be possible to deploy a multiple sensor system, but simply using multiple sensors with a single board is a more cost-efficient solution.

In addition to the aforementioned improvement ideas, our future research will focus on topics only briefly discussed in this paper, such as the utilization of the collected data using the designed REST API in various end-user applications.

As can be seen, there are many ways to continue and improve this study. However, the current service produces real-time data regularly and reliably for the benefit of the main system. The prototype developed has proven to be stable and reliable in practice. Work on building the final alarm system is currently ongoing at Keio University.

V. SUMMARY

The paper introduced a prototype system created for sensor data collection and transmission. The presented data collector service is part of a larger alarm system and provides sensor data for the main system. The aim of the study was to test how a single board computer can be used to gather sensory data and how this data can be provided to clients over the public Internet. The paper presented the features and architecture of the developed service, the used hardware and software components, the physical connections between the components, and also the structure of the software. The paper gives a concrete example of how to utilize a microcontroller with an embedded Linux distribution.

REFERENCES