Multidisciplinary
SCIENTIFIC JOURNAL OF
MARITIME RESEARCH

University of Rijeka
Faculty of Maritime
Studies Rijeka

Multidisciplinarni
znanstveni časopis
POMORSTVO

Logistics environment awareness system prototype based on modular Internet of Things platform

Saša Aksentijević[1], David Krnjak[2], Edvard Tijan[3]

[1] Aksentijevic Forensics and Consulting Ltd, Gornji Sroki 125a, Viškovo, Croatia
[2] Saipem SpA Croatian Branch, Alda Collonnella 2, Rijeka, Croatia
[3] University of Rijeka, Faculty of Maritime Studies Rijeka, Studentska 2, 51000 Rijeka, Croatia, e-mail: etijan@pfri.hr

ABSTRACT

Internet of Things (IoT) is a completely new paradigm of interconnected computing devices in the market segment that has started emerging from 2013, while trends have been recognized in 2014 and most predictions are related to the period until 2020. Anticipating widespread use of IoT technology in logistics chains reported by leading sector players, a dedicated logistics testbed IoT platform named MiOT is created and tested using Raspberry PI minicomputer, with research goal to evaluate possibilities of integration of the logistics IoT platform inside existing Windows corporate domains.

ARTICLE INFO

Preliminary communication
Received 22 November 2015
Accepted 18 December 2015

Key words:
Internet of Things
Logistics
Environment awareness system

1 Introduction

Internet of Things (IoT) is a new and upcoming paradigm related to networking of various applicative physical devices ("things"), as opposed to current situation where networking refers primarily to computer and network devices and peripherals. "Things" are embedded with electronics, software, sensors and connectivity that enable them to achieve functional value and exchange data with other devices and systems. They communicate over Internet and cover a variety of protocols, domains and applications.

Typical applications of IoT are various sensors or transponders used on farms or in search and rescue missions, automobiles with built-in sensors, biochips, wearable computers in any form and home or industrial automation systems. IoT technology in logistics is used to ensure quality of shipment conditions (monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes), item location (search of individual items in large areas like warehouses or harbors), storage incompatibility detection (warning emission on containers storing inflammable goods closed to others containing explosive material) and fleet tracking (control of routes followed for delicate goods like medical drugs, jewels or dangerous merchandises) [1].

IoT paradigm emerged due to convergence of various technologies approximately as of 2013, even though it has been in some its aspects discussed for decades and has been a topic of science fiction even longer than that.

At the end of 2014 there were 3,75 billion such devices already in use, and predictions state that at the end of 2015 there will be 4,88 billion [2] devices deployed globally, and corporations will spend more than 40 billion US$ for development of IoT solutions [3]. It was anticipated mid-2013 that 26 billion such devices will be connected to the Internet by 2020 [4]. Cisco and DHL have revised this number in July 2015 to 50 billion by 2020 [5]. Some analysts set these numbers at around 30 billion [6]. IoT devices are expected to exhibit similar exponential growth in consumer and business sectors. Cisco Systems expect the IoT market to reach a staggering value of 19 trillion US$ [7].

The convergence of various technologies will raise numerous questions, and industries logistics sector will certainly not be left out of the development. These questions can be divided in several categories that need to be addressed, among them the most important being information security, design, sustainability and environmental impact and privacy, autonomy and control.

In our research we have decided to tackle the following early stage IoT problems in logistics companies' IT environment and address some of the issues raised above by:

1. **Creating a theoretical testbed** IoT platform to be used specifically in logistics companies' research,
2. **Creating a working prototype** of the physical platform's application based on the testbed (in our case a sensor system to control overall environment), while respecting all prudential measures and best practices related to ICT security,
3. **Deploying the prototype in Windows domain**, as opposed to separating it outside the domain, but at the same time, implementing all information security rules imposed on devices connected to the domain. The reasoning for this is that in transitory period and while cross-industry standard is defined, there might be a need to join IoT devices to logistics companies' domains and IT function should have both skills and a standard set of practices available at hand how to manage them. Development related to deployment of logistics environmental awareness system inside corporate Windows domain will be a subject of further research.

We were led in our research by the following principles that are in line with expert panels' concerns explained above:

1. **Compliance** with best practice information security standards,
2. **Implementation of industry's best practices** using currently available protocols,
3. **3-way sustainability** approach:
   a) to use the electronic components with the lowest disposal footprint possible,
   b) to use implementation practices that achieve the largest financial and CO2 savings,
   c) to use the least number of the cheapest components possible,
4. **Open source design**: to facilitate overview of the code and discussion.

By April 2015, a testbed solution for usage of Internet of Things (IoT) platform inside Windows domain has been created. Working title of the platform is **MIoT – Modular Internet of Things**, thus achieving the primary research goal.

Besides the test bed solution, sensor electronics package prototype has been created along with 3-tiered program code, software support needed for data acquisition, storage, interpretation and visualization. Working name of the package is **LEAS – Logistics Environment Awareness System.**

However, the motivation for creation of both platforms was not purely technical, so it was decided early on in the project to evaluate technology in terms of overall financial and business impact. Schematic representation of interoperation of both systems and their main components is shown in the following figure.

Test bed and prototype were created bearing in mind the possibilities of further development and integration. They have been deployed in real-time environmental monitoring of a server room location supporting logistics chain activities of a multinational corporation in the period between April and November 2015 and have proven to be very stable in operation.

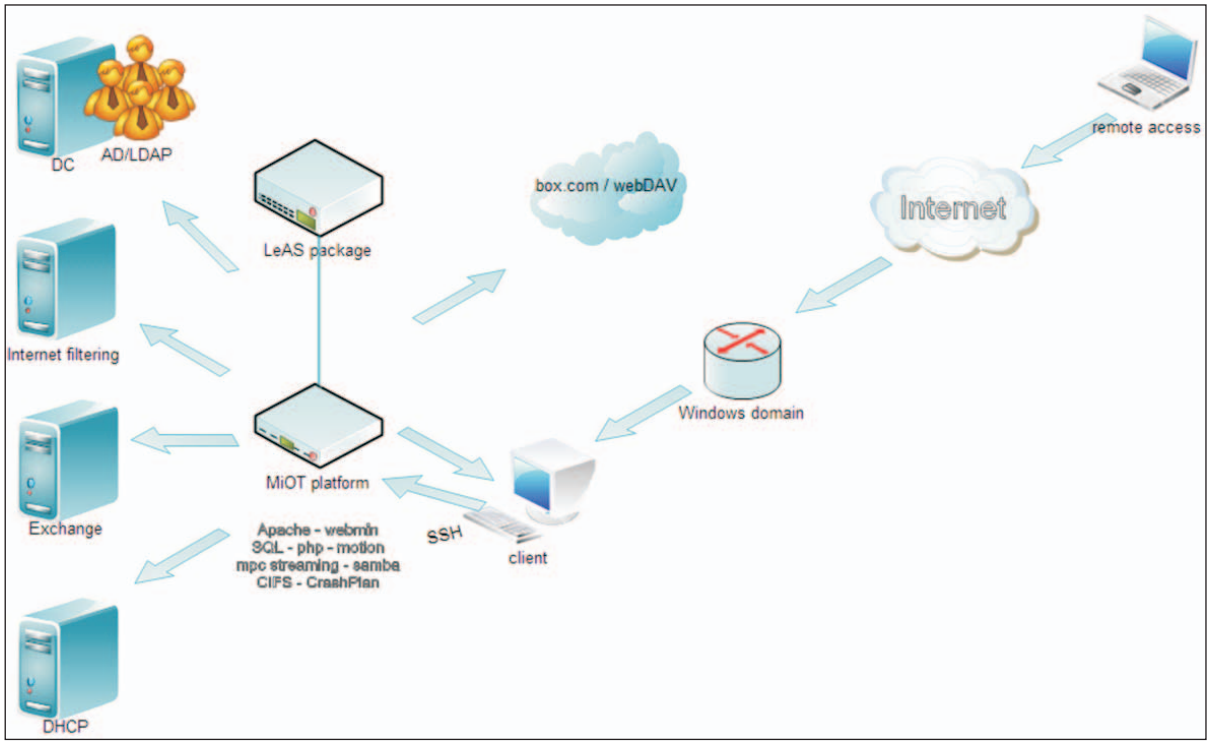

Figure 1 Schematic representation of MIoT and LEAS

2 Platform architecture description

MIoT/LEAS are:

1. Pioneer attempts to open the discussion about Internet of Things paradigm in the logistics companies,
2. Complete mini-computing solutions for further research and implementation,
3. A study in principles of Internet security, integration and sustainability of IoT devices in existing corporate Windows networks, and
4. Projects in progress used to learn and gather new experience.

MIoT/LEAS are not:

1. Definitive answers to challenges presented by near-future adoption of Internet of Things,
2. Attempts to define a way how to handle Internet of Things devices used in the logistics company outside of perimeter of logistics companies' networks or domains,
3. Suggestions for new developments in proprietary electronics used for industrial control.

2.1 Hardware

During initial investigation, several existing mini-computing platforms were considered, among them Beaglebone, BananaPi and Raspberry PI [8] platform (RPI in the text). Due to popular support, number of available resources and open code, it has been decided to proceed with detailed investigation of the feasibility of utilization of RPI. RPI is a credit-card sized full-capability mini-computing platform along with microcontroller able to control sensors and other external electronic circuits. Any model of RPI is able to run MIoT and LEAS, while we have used RPI B+ [9].

Led by principles of three-stage sustainability approach described in the Introduction, a basic set of hardware was obtained and a mini computer was assembled by simple addition of the heat sinks and electronics board in transparent acrylic enclosure.

USB 2.0 ports may serve any purpose or communicate with attached devices compliant with the standard. Successfully tested, but not used in the project were USB flash (pen) drives, USB hard drives, Bluetooth dongles and Wi-Fi dongles. Network connectivity was achieved by using standard Ethernet port and 10/100 cable connected to the corporate network. To use devices with higher power requirements, external power source via powered hub connection has to be provided. RPI can be powered from a simple USB high quality mobile phone charger rated at 1A or more, or from a USB powered port of another USB device. Both options were successfully tested.

Early on in the project, for portability, simplicity and security purposes it has been decided to run the RPI in a "*headless*" state [10], without physical monitor, keyboard or mouse and rely instead on remote connectivity. However, physical connectivity was tested and it works via HDMI connector or HDMI-DVI converter, and using standard USB keyboard and mouse over onboard USB ports.

2.2 Software

RPI uses mini or micro SD card for file system storage and the operating system. The first challenge was to find an adequate operating system to run RPI. Several candidates were evaluated, among them ARM Arch Linux, RaspBMC, RiscOS and Raspbian. Finally, Rasberry PI foundation's Raspbian operating system [11], contained on NOOBS distribution [12] was selected as the most viable candidate, and the selection proved to be adequate as the implementation was completed using that particular distribution.

Raspbian is foundation's fork of Debian Linux operating system, specially prepared for use with ARM processor contained in RPI. This particular operating system is known to be pre-delivered as hardened in terms of basic rules of security. The only port left open for external connection is port 22 used for secure SSH connections. Usage of this particular operating system provided researchers with a great deal of flexibility in additional hardening of the device for outside connections.

It has been decided early on in the project to provide two connectivity possibilities to access RPI:

1. **Using SSH** over direct serial connection (requires physical access to device) or over the network, and
2. **GUI (Graphical User Interface)**; for this particular purpose, Raspbian LXDE [13] Desktop was selected and installed.

Terminal services are used to connect to RPI's Graphical User Interface (GUI), namely, a lite terminal server is installed on the RPI in form of TightVNC terminal services. In order to toughen the connection as it is not secure in its initial state, initial SSH tunnel is initiated from the side of the RPI towards the client terminal machine.

After that, the connection is established by using the local host from the other side. This way, no unencrypted traffic flows between RPI server and connecting client. All management and connections are successfully tested between standard PC workstations running Windows operating systems inside Windows domain. Furthermore, there is a possibility to control RPI using encrypted terminal services (for example, *TeamViewer*) from mobile phones or a workstation inside Windows domain.

After successful remote control of the RPI, several methods for data exchange were tested and found to be fully functional, including CIFS protocol, Samba server, box.com cloud sharing and copying files to and from FAT32, NTFS or ext(x) using externally attached USB media.

Further to this, Apache, MySQL database and PHP servers (LAMP stack [14]) were also installed and were able to smoothly and concurrently run on the RPI, and Webmin GUI for system and server administration.

Modified Java 8 was successfully tested and was used on RPI. Another successfully tested but disabled (unused

functionality in the final setup of MIoT) relying on Java was cloud backup service from Code42 company called CrashPlan [15]. Bit Torrent Sync private cloud was also successfully tested (not to be confused with public *Bit Torrent* file sharing). Streaming packages *mpd* and *mpc* [16] were also used and tested (for example, for local Internet audio or radio streaming).

Further possibility of secure connection implementation consists of *OpenVPN* solution that was successfully ported to RPI. Finally, an attempt was performed to join RPI device to a Windows domain, and it was successful. This was achieved by using Kerberos 5 [17] and WINBIND Linux packages, so users logging on either remotely or locally to RPI need to have Windows domain account in order to access the MIoT device. These two facts present a major achievement because in this scenario:

1. It is possible to obtain **automatic recognition** of an IoT device on which domain administrators can impose a **set of rules** set by Microsoft's global policies, and
2. It is possible to obtain **audit logs** both on the device and the user actions.

The same is also relevant for remote connections, users using GUI and logging on remotely have to use Windows domain logon accounts in order to access the RPI device.

Exhaustive list of packages installed and tested on the RPI is quite long, so only the most important packages will be mentioned in the following list:

1. System, configuration and GUI preference tools
2. System utilities
   - Image Viewer,
   - *TightVNC* (Terminal services),
   - *Leafpad* (similar to notepad),
   - *Xarchiver* (archived file manipulator, similar to WinZip),
   - File manager,
   - Task manager, and
   - *WinSCP* (FTP client for RPI)
3. Office and productivity tools
   - Document viewer,
   - *xpdf* (PDF viewer), and
   - *Epiphany Web Browser* (able to run HTML 5 code, Youtube videos etc.)
4. Programming languages (both used in the research):
   - Python, and
   - C.

A complex application package used for environment monitoring via camera and/or microphone system called *motion* [18] was installed, tested and deployed on RPI. This application is highly configurable and used to monitor environment for motion and changes, take continuous or intermittent snapshots, detect movement according to preset criteria and create stills, shockwave animations or movies, offload them to a specific storage location, use multiple sources (cameras) and trigger certain actions. The system functions both in night and day conditions.

Local password policy is forced via PAM password policy while changes for the local password are enforced every 90 days, with the following parameters set through pam_cracklib.so.

Windows domain group for device management has been set, and only users from that group can log on to the device, further disabling unauthorized access to the MIoT platform. Login is currently allowed only to members of the domain group SID, namely SMH_G_RPI_login, by using pam_winbind.so.

Distribution and packages are maintained by Raspberry Pi Foundation and MIoT's operating system and dependencies can be upgraded either manually by using Internet connection, or they can be enforced using operating system and automatic updates patching by installed *unattended-upgrades* Debian package allowing for flexible silent background updating of the device itself.

All installed connection capabilities that are currently not in use are disabled by default, further strengthening the connection security of the platform.

Backup of the file system is achieved by using freeware tools *Win32DiskImager* and *DiskInternals Linux Reader* [19] on Windows workstations by reading or writing entire file system from SD card to and from a flat backup file. Final result is a hardened mini-computer device, integrated and controlled inside Windows domain and used by authenticated users having access to a viable and auditable Windows logon account. This is a best practice scenario usually encountered in logistics chain companies.

3 LEAS sensor package description

LEAS system may be used in any weather-proof space to gather information about the environment, store and alert about occurring events in the logistics process. With further weatherproofing of the casing design, the system may even be mounted externally or aboard vessels. Implementations based on MIoT platform (and especially sensory packages connected to it) should not be used for critical implementations with potential impact on material processes or personnel. Used sensors are entry-level and should be carefully calibrated and adjusted according to atmosphere requirements and applicable technical guidelines.

Deployed software code should be carefully evaluated. Limits related to its use and effects and functioning of MIoT platform should be taken into consideration prior to deployment.

3.1 Hardware

Goal of the LEAS prototype was to create an electronics sensor package controlled by program code run on MIoT platform described in the previous paragraph.

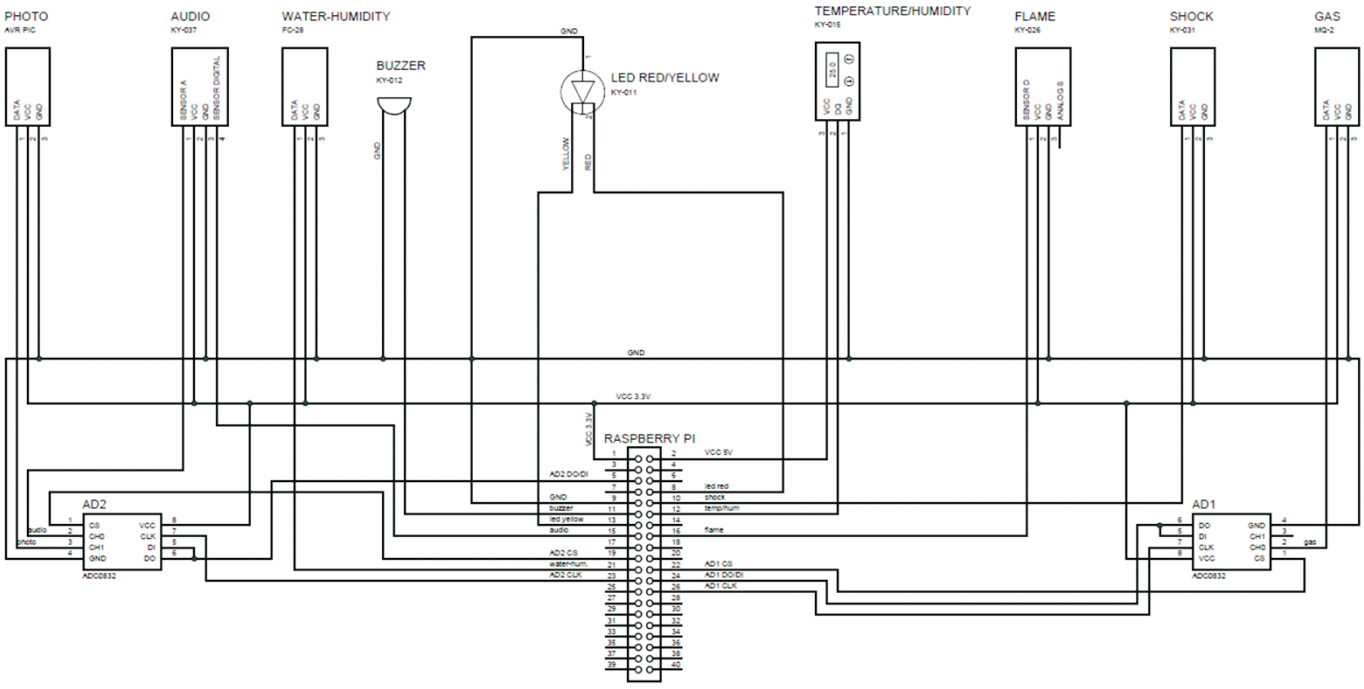

Figure 2 LEAS electronic circuit board layout

Components were carefully selected to be tough and reliable and the final choice was *Sunfounder* sensor kit [20]. To achieve speed of testing and assembly, two parallel stacked breadboards were used.

Considering that full environmental friendliness was one of the goals, there was no soldering, and no acids or other chemicals were used in preparation of final electronic circuitry. All elements are secured in place by using metal M2 and M3 Philips screws, nuts and bolts.

The circuits are communicating with MIoT using flat 40-pin data cable. Breadboards contain four power rails, two in each half of the breadboard motherboard. Three power rails from left side are powered with 3.3V while rightmost one is powered with 5V. Reason for this is the fact that some sensors require 3.3 V while some require 5 V. All ground rails are properly grounded. Color coding for wires is followed so all red wires carry electric current, all gray, dark or black wires are ground and other colors are signal wires. During testing, all rails had LED power indicators, while in the final assembly four LEDs and four resistors were removed, further simplifying the design.

There was a total of seven sensors used in the circuitry. Furthermore, there were three more elements used for auditory and visual warning. The list of used components and alerts is the following.

1. **Shock sensor** KY-031: detects movement of the LEAS package in any direction,
2. **Temperature/humidity sensor** KY-015: measures temperature and relative humidity,
3. **Flame sensor** KY-026: detects spark or flame by wavelength,
4. **Gas sensor** MQ-2: SnO2 industrial sensor that detects flammable gas, smoke, LPG, propane, butane, methane and H2,
5. **Photo sensor** AVR PIC: detects changes in illumination level,
6. **Sound sensor** KY-037: detects sudden change in noises,
7. **FC-28 Soil Humidity Detection Sensor Module:** selective A/D sensor that can be used to detect humidity in material, in this case used to detect water leak,
8. **Active buzzer** KY-012: used to announce auditory signal during increased levels of flammable gas,
9. **Red-yellow** LED KY-011:
   - red LED: visual alarm for fire alarm, and
   - yellow LED: visual alarm for movement detection.

Addressing of the header pins of the MIoT microcontroller in the program code was done in alignment with the library used to control pins, WiringPI. Ordinal header pins, functions (names) and BCM GPIO nomenclature is not in line with WiringPI. It is also possible to address separate pins, read and write them using GPIO library installed on the operating system, and not only by programming code in C using WiringPI library [21]. Main header pin layout conforms to the specification of the RPI B+ board.

Sensors that use interrupts in assembled circuitry are flame, shock, water and sound sensors. Other sensors are working in continuous loop reading mode. Readings of such sensors are put on hold while reading of the sensor triggering interrupt is in progress. Photo, sound and gas

sensors provide analog output while other sensors are digital. Water (soil humidity sensor) can be used both in digital and analog modes, but to avoid further wiring towards analog to digital converter, analog mode was selected for this particular sensor. Analog sensors require analog to digital converter chips in order to convert the signal to readable state. For this purpose, two ADC 08032 [22] chips were used. One chip is addressing one single sensor and the other channel is idle, while the other chip is addressing two sensors. Initially, there was one ADC per chip, but using this configuration enabled further simplification of the circuitry be removal of one chip and a bundle of jump wires.

Power consumption of the working board (for both MIoT and LEAS) is estimated at around 2 W [23]. The entire circuitry schematic diagram is shown in Figure 2.

Sensors' and ADC's input and output pins are controlled by cobbler board attached to the left breadboard. Cobbler board also provides electric power for the circuitry without requirements for external power source, adding to the simplification of the design.

One or more USB cameras (v1.1 or higher) can be attached to MIoT or a powered hub to gather environment motion and/or sound input. In test configuration, an affordable USB camera was adjusted in a way that infrared filter was removed so the camera may be used in daylight and night conditions under additional infrared illumination that is not visible to human eye. Current camera views can be accessed from MIoT minicomputer using Internet browser or from any computer on the same network segment, as the view itself is safely published using Apache web server [24].

3.2 Software

Software running on MIoT device used to control LEAS package consists of three modules: **sensor control engine, database layer and visualization layer.** Sensor control engine is used to gather data from the sensors, process alerts and write data to databases, database layer stores and serves data while visualization layer is used to visually inspect and interpret data. Selection of layers is similar to the usual best practice for division of IoT layers to 3-tiered division of perception, network and application. [25]

All software is written and compiled using RPI itself and installed native Python and C packages.

3.2.1 Sensor control engine – tier 1

Sensor control engine is written in programming language C and consists of 663 lines of code and uses various external libraries, among them the most significant one being open source WiringPI library used to control I/O pins of the MIoT microcontroller. Sensor control engine can run concurrently with visualization layer. The engine initially loads configuration file (plain text file) that defines main behavior of the program. Odd lines are treated as comment lines and even lines are entry values.

Explanation of the used initialization parameters is the following:

1. **Mathplotlib** [26] **redraw time** is a parameter shared with visualization layer described in more details in 3.2.3, and it defines time between updates of the screen with new data,
2. **Mail TO filed** is email address of the first alarm recipient,
3. **Mail CC field** is email address of the second alarm recipient,
4. **Temperature Min and Max** are trigger thresholds for low and high ends of allowed temperature,
5. **Gas sensor threshold** defines sensitivity of the gas sensor. Gas sensor is polled for new information every second,
6. **Interrupt write delay sec** defines delay time for sensors that use processor interrupt when triggered, namely flame, shock and sound sensors. During delay, writing of the new data is disabled,
7. **Mailing delay** in seconds is a delay between occurrences of sending alert email for the event. For example, in case that there is fire alert, an email is being sent out, but if the alert continues, emails are not sent anymore for a period of time defined in mailing delay parameter of the configuration file. Other types of alerts that are sent by email during that period are not suspended,
8. **Delay for temperature, humidity and illumination reading in sec** is a delay between readings for temperature, humidity and illumination.

Sensor readings are performed in a loop, once per second, and readings are recorded in a database and displayed in a terminal window according to logic taken from the configuration file. Log data related to email sending (SMTP activity) is shown in the terminal window, but not recorded in database.

If an alarm state is detected, indicating that temperature fell below or rose above set levels, flammable gas is detected or there was a knock, sudden change in sound level, water, spark or fire detection, an email is being sent to predefined email addresses with time and type of the event.

Concurrently, a motion daemon is run and single or multiple cameras connected to LEAS are gathering surrounding environment's information. The system is highly configurable and all possibilities of configuration will not be in details described in this paper. All detected motion in form of still pictures or video recordings, along with timestamp can be recorded on a local or remotely connected media and various triggers can occur depending on desired behavior.

Motion daemon can be configured to store only images when motion is detected or continuously, sensitivity level

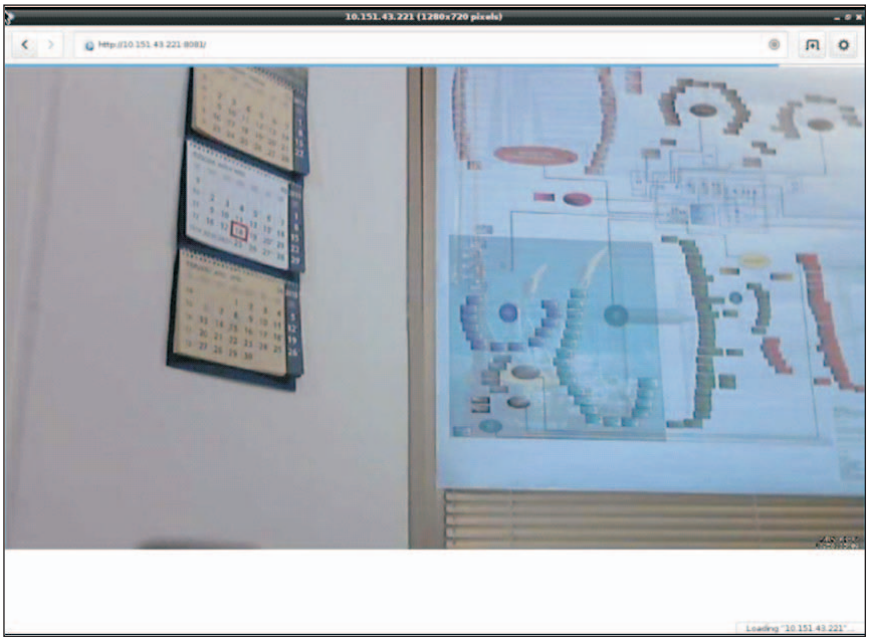

Figure 3 Live webcam view of motion daemon (max. camera resolution 720p – 1280×720 pixel)

can be set and picture can be divided into 9 equal segments and motion detection can be focused only on one of them or overall scene and according to pre-set percentage of pixel change. Images or automatically created Flash or video clip files can be saved to file system for later viewing. Current view of the camera(s) can be accessed over the network using regular Internet browser by accessing MIoT's IP address.

Motion function was also tested under night conditions and infrared illumination. Infrared light source was a small 3-LED light powered by 4.5V battery that was converted by soldering infrared LEDs instead of OEMs. Regular webcam was altered in a way that infrared filter screen was removed from the housing, turning webcam into a night vision device. Scene motion was successfully detected and results were satisfactory in complete darkness.

3.2.2 Database layer – tier 2

For simplicity, daily captured sensor data is stored in flat text files database that consists of three files using the following naming convention format:

1. *<year><month><day>_A_A.txt* – contains alarms activated by external events: fire, sparks, shock, gas leakage, sudden loud change in sound level and water detection. Events are stored in a way to contain date, time, type of event and measured level of event (if applicable). For example:
   2015-03-09 00:00:01 CEST; GAS detected, current level: 58
   2015-03-09 12:23:29 CEST; Sound Sensor ACTIVATED: 140
   2015-03-09 10:49:33 CEST; Knock detected!
   2015-04-21 11:52:00 CEST; Water detected!
2. *<year><month><day>_H.txt* – contains stored readings of external temperature and humidity, date and time of reading. For example:
   2015-03-09 00:00:02 CET: Humidity = 35.0 %
   Temperature = 22.0 *C (71.6 *F)
3. *<year><month><day>_I.txt* –contains date, time and level of illumination reading. For example:
   2015-03-09 10:48:14 CET: Current illumination: 87

Every day at midnight, a new set of three files is open and used to store new data while old sets of database files are maintained on a file system. Number of daily events kept in the database depends only on free space of the media. The system generates up to 600 kb of measured data per day, so monthly it requires around 18 MB of storage. Log database rotation is achieved using Linux cron job (an alternative would be using configurable settings inside Webmin [27]) and can be set according to own preferences. Linux cron job is run to prune all database entries older than 30 days.

3.2.3 Visualization layer – tier 3

Visualization layer is written in programming language Python [28] and consists of 344 lines of code, uses various libraries, and among them the most prominent being open source library for chart representation *Matplotlib*. Visualization layer is automatically drawing data for the current day after running the application and enables scrolling between days. Taking into consideration the relatively low raw processor power of the MIoT test bed platform and number of readings (plot points), it can take several minutes to render one entire day worth of events. Setting up less dense readings using configuration file will

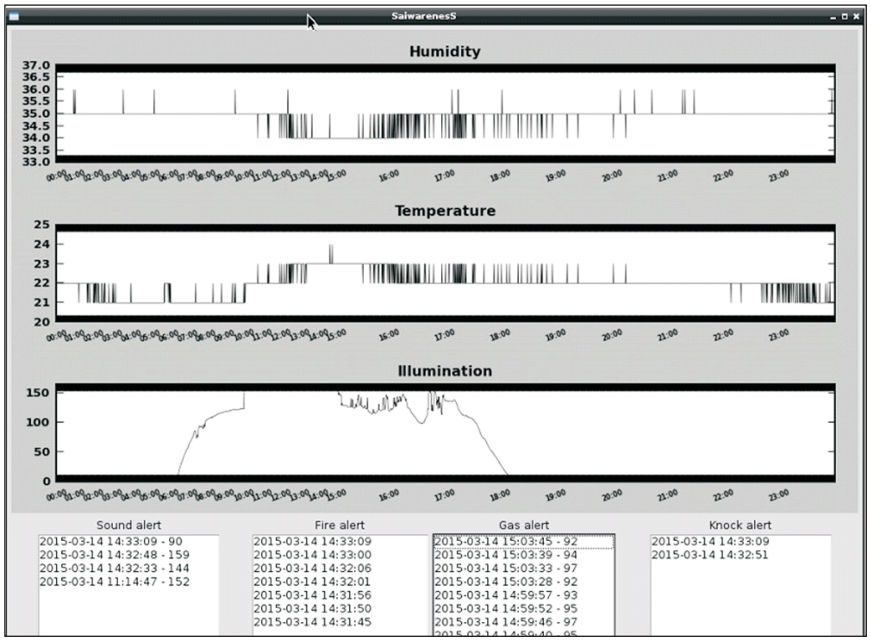

Figure 4 LEAS GUI (Graphical User Interface)

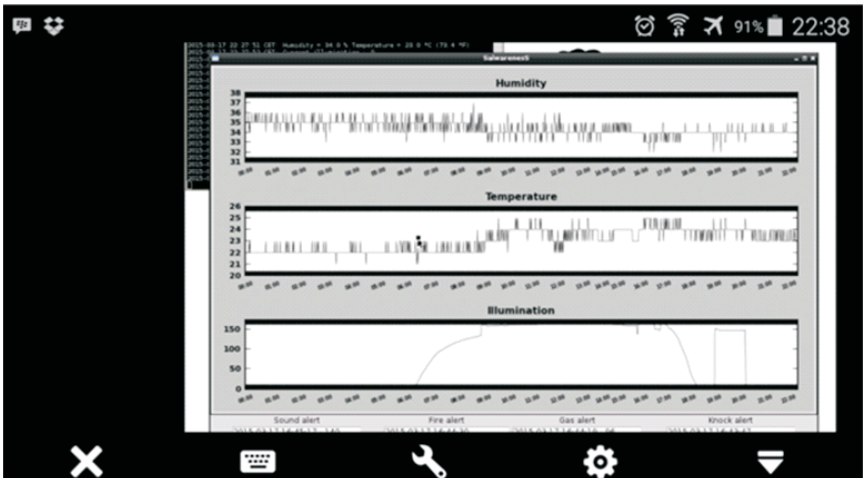

Figure 5 Smartphone visualization of LEAS

speed up rendering time. The same is valid also for rendering data of the previous or next days.

In the plot area, there are three charts representing temperature, humidity and illumination throughout the day and four box areas below them showing last eight alarms related to shocks (knocks), fire or sparks, sudden elevation in sound levels, detection of hazardous gasses and water detection.

Graphical representation of LEAS visualization layer is shown in the Figure 4.

Control over MIoT and LEAS is possible also using smartphone's GUI and functionalities, as shown in Figure 5.

As already mentioned in Introduction, access to LEAS GUI is also possible by using any smartphone capable to run terminal services and open access to the client from which secure connection is established towards MIoT platform.

LEAS visualization GUI contains some basic functions like navigation between days using left and right arrows, option to jump to data representing current day, "About" section and button used to exit from the GUI back to the operating system and LXDE desktop GUI.

4 Conclusion

On a macro level, the project is a result of recognition of an emerging trend, Internet of Things, which will be a hot topic in the next decade. Logistics companies have yet to align their practices with this new emerging trend and define a set of applicable rules for management of such

devices. With this step we wanted also to underline importance of this new technology paradigm inside logistics companies and enhance internal discussion on the topic.

On a micro level, we wanted to show possible applications and calculate some measurable savings related to just one such application. Real savings depend on perfusion of the platform's implementation in business realities of logistics companies. Opportunities are numerous and will be dictated by the perfusion of technology in the near future.

For the first part of our application, testbed IoT platform named MiOT, there was no traditional, previously used approach. Internet of Things is a completely new paradigm of interconnected computing devices. Currently, IoT is not a platform in widespread usage and it might have certain components scattered across industrial electronics solutions used in operations of the logistics companies and managed in a proprietary way or by implementing best practices for industry control electronics applications.

In the second part of our application, we have created a sensory package for logistics purposes, while a traditional approach would be the purchase and maintenance of proprietary package from the manufacturers, with pre-set number of sensors and functionalities. Annual cost of maintenance carries approximately 16 % of the investment cost in the traditional approach scenario. The packages are installed "as-is", with no possibility to intervene in the code, add functionalities or change them. On the positive side, such industrial solutions are reliable, made of high quality components, certified and easily replaceable in case of malfunction but expensive to purchase and maintain.

Industrial electronics are habitually segregated from logistics companies' business networks. We expect that miniaturization and developments in IoT segment will gradually change this design in the near future. With the creation of MiOT testbed, we have attempted a pioneer approach to create a platform that may be joined to a domain under the same set of rules applicable to user-assigned workstations and allow for secure and flexible computing. This platform is able to fill various server roles and be a microcontroller, something that currently requests heavy physical or virtual computing facilities and proprietary microcontroller packages.

Further to this, we wanted to create not only a testbed, but also a working prototype of a sensory package - LEAS - for use in environmental monitoring of logistics companies' operations, controlled by MiOT testbed. This particular package costs only a fraction of commercially available proprietary environment sensing equipment and due to extremely low electricity consumption leaves negligible financial and CO2 footprint during operations, while providing all necessary functionalities, email alerts, database event logging and graphical representation of data.

The actual benefit of MiOT testbed platform is the creation of an open source Linux OS image that can be used on Raspberry PI computing platform by anybody in logistics companies willing to learn, deploy or use Internet of Things solutions inside their networks in an efficient and sustainable way, until or in parallel with the time when the standard for management of IoT devices inside logistics companies is clearly defined.

The actual benefit of LEAS sensor package is the achieved compliance with a number of provisions of the best practices and information security policies in logistics companies, but also integral security as the sensor package is able to capture and monitor almost all environment information and can be further expanded in numerous ways by adding sensors.. Furthermore, LEAS is a complement to MiOT, where it has been proven that a useful package can be implemented in Windows domains in a safe and efficient manner by respecting all existing information security rules and practices.

In terms of social sustainability, special care was taken to use open source programs and to set the program code in a way to be able to release it as an open source code. Furthermore, by creating a testbed platform for IoT in the initial phase of the project, we have envisaged the possibility to further develop and disseminate knowledge and develop capabilities of IT personnel and electrical engineers that will be involved in implementation of logistic IoT platforms in the future.

References

[1] Vermesan, O., Friess, P., Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River Publishers, Aalborg, Denmark, p. 34.

[2] Velosa, A, Schulte W.R, Lheureux B.J., Hype Cycle for the Internet of Things, Gartner, August 2014, https://www.gartner.com/doc/3098434 [18.08.2015]

[3] Executive Summary Report, 2014 Conference Theme: Driving Digital Business, Gartner Symposium ITxpo 2014, 5-9 October 2014, Orlando, Florida, http://www.gartner.com/binaries/content/assets/events/keywords/symposium/sym25/gartner-sym24-executive-report2.pdf [18-08-2015]

[4] Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020, Press Release, Gartner, STAMFORD, Conn., December 12, 2013, http://www.gartner.com/newsroom/id/2636073 [18-08-2015]

[5] Macaulay J., Buckalew L., Chung, G., Internet of Things in Logistics – A collaborative report by DHL and Cisco on implications and use cases for the logistics industry, DHL Trend Research – Cisco Computing Services, p.5, [29-06-2015]

[6] Internet of Everything, ABIresearch, https://www.abiresearch.com/market-research/service/internet-of-everything/ [accessed 18-08-2015]

[7] Cisco CEO pegs Internet of Things as $19 trillion market, International Consumer Electronics Show, Las Vegas, 07th January 2014., http://www.bloomberg.com/news/articles/2014-01-08/cisco-ceo-pegs-internet-of-things-as-19-trillion-market [18-08-2015]

[8] Exploring Beaglebone – Companion Site for the Book by Derek Molloy (http://exploringbeaglebone.com), „Banana Pi – A Highend Single-Board Computer" (http://www.bananapi.org), Raspberry Pi Foundation https://www.raspberrypi.org [19th August 2015]

[9] https://www.raspberrypi.org/products/model-b-plus/ [19th August 2015]

[10] Munsell, A., Setting up the Raspberry Pi as a Headless Device, https://www.andrewmunsell.com/blog/setting-up-raspberry-pi-as-headless-device/ [19th August 2015]

[11] Raspbian, https://www.raspbian.org [19th August 2015]

[12] NOOBS Archives – Raspberry Pi, https://www.raspberrypi.org/blog/tag/noobs/ [19th August 2015]

[13] LXDE.org | Lightweight X11 Desktop Environment, http://lxde.org [19th August 2015]

[14] LAMP Stack – Web Stack (MySQL), https://www.turnkeylinux.org/lampstack [19th August 2015]

[15] Online Data Backup – Offsite, Onsite & Cloud – Crashplan Backup Cloud", http://www.code42.com/crashplan/ [19th August 2015]

[16] mpc – Music Player Daemon, http://www.musicpd.org/clients/mpc/ [19th August 2015]

[17] Ricciardi, F., The Kerberos protocol and its implementations, INFN – the National Institute of Nuclear Physics Computing and Network Services, Lecce, Italy, 26th November 2006, p. 4.

[18] Motion – Web Home, http://www.lavrsen.dk/foswiki/bin/view/Motion [19th August 2015]

[19] Freeware Linux Reader for Windows, http://www.diskinternals.com/linux-reader/ [19th August 2015]

[20] Sunfounder, http://www.sunfounder.com [19th August 2015]

[21] Wiring Pi – GPIO Interface library for the Raspberry Pi, http://wiringpi.com [19th August 2015]

[22] ADC08032, Texas Instruments, http://www.ti.com/product/adc08032 [19th August 2015]

[23] http://raspberrypi.stackexchange.com/questions/5033/how-much-energy-does-the-raspberry-pi-consume-in-a-day [19th August 2015]

[24] Apache http server project, http://httpd.apache.org [19th August 2015]

[25] Song, X., Huang L., Fenz, S., Internet of Things Applications in Bulk Shipping Logistics: Problems and Potential Solutions, Communications in Computer and Information Science Volume 312, 2012, p. 566.

[26] matplotlib, http://matplotlib.org [19th August 2015]

[27] Webmin, http://www.webmin.com [19th August 2015]

[28] Welcome to Python.org, https://www.python.org [19th August 2015]