Prototype for Multidisciplinary Research in the context of the Internet of Things

Miguel López-Benítez[a,*], Timothy D. Drysdale[b], Simon Hadfield[c], Mohamed Ismaeel Maricar[d]

[a]*Department of Electrical Engineering and Electronics, University of Liverpool, United Kingdom*
[b]*Department of Engineering and Innovation, The Open University, United Kingdom*
[c]*Centre for Vision, Speech and Signal Processing, University of Surrey, United Kingdom*
[d]*George Green Institute for Electromagnetics Research, University of Nottingham, United Kingdom*

Abstract

The Internet of Things (IoT) poses important challenges requiring multidisciplinary solutions that take into account the potential mutual effects and interactions among the different dimensions of future IoT systems. A suitable platform is required for an accurate and realistic evaluation of such solutions. This paper presents a prototype developed in the context of the EPSRC/eFutures-funded project "Internet of Surprise: Self-Organising Data". The prototype has been designed to effectively enable the joint evaluation and optimisation of multidisciplinary aspects of IoT systems, including aspects related with hardware design, communications and data processing. This paper provides a comprehensive description, discussing design and implementation details that may be helpful to other researchers and engineers in the development of similar tools. Examples illustrating the potentials and capabilities are presented as well. The developed prototype is a versatile tool that can be used for proof-of-concept, validation and cross-layer optimisation of multidisciplinary solutions for future IoT deployments.

Keywords: Internet of Things, hardware design, communications, data processing, prototyping, experimentation.

1. Introduction

Communication networks no longer connect just people, but are evolving into billions of interconnected smart devices (sensors, controllers, machines, autonomous vehicles, drones, etc.) (Atzori et al., 2014), with embedded electronics and a number of common basic functionalities (communications and networking protocols, operating systems and software) that enable automatic collection and exchange of data (possibly with little or no human intervention). This concept, known as the Internet of Things (IoT) (Atzori et al., 2010; Al-Fuqaha et al., 2015; Mattern and Floerkemeier, 2010), virtually allows any object to be sensed and controlled remotely across existing network infrastructure, creating unlimited opportunities for the integration of the physical world into automated computer-based systems.

IoT is seen as the next stage of the information revolution and, with an estimated 50 billion devices connected by 2020 (Evans, 2011), it is becoming a reality with the potential to revolutionise our lives through many new generation *smart* applications, such as smart cities, smart homes, smart e-healthcare, smart transportation, smart energy management, and smart security (Zanella et al., 2014), which will lead to improved efficiency and socio-economic benefits (Vermesan and Friess, 2013; Fleisch, 2010; Perera et al., 2014).

Future IoT systems will be multidisciplinary in nature and this will pose important challenges. Some relevant disciplines that will play a key role in future IoT systems are discussed below along with problems that still need satisfactory solutions:

1. *Hardware design.* Future IoT systems will require the ability to add IoT capability to almost any object, not only *smart* but also *dumb* objects. These interconnected devices need to be small and inexpensive. Novel hardware designs are required to enable ultra-compact wireless sensors (approximately the size of a human thumb or smaller). The size of the hardware associated to signal and data processing can be reduced with state-of-the-art nano scale technology. However, reducing the size of other components can be more challenging (this is particularly true for certain elements such as antennas, whose size can be constrained by the frequency of operation). New antenna designs that can be attached to almost any small object are required. Soft antennas and stretchable antennas are some particularly promising solutions, however they require a detailed research study in the context of IoT. Moreover, IoT devices need to be energy efficient. In industrial applications, IoT devices will often need to be able to run for at least ten years on a single battery. The battery size (and consequently its capacity) may be constrained by the size of the IoT object. Moreover, if there are to be billions or trillions of IoT devices then it

*Corresponding author.
Email addresses: M.Lopez-Benitez@liverpool.ac.uk (Miguel López-Benítez), tim.drysdale@open.ac.uk (Timothy D. Drysdale), s.hadfield@surrey.ac.uk (Simon Hadfield), mohamedismaeel.maricar@nottingham.ac.uk (Mohamed Ismaeel Maricar)

will be impossible to use batteries for every single IoT device (it would be impractical, costly, and unsustainable). Energy harvesting techniques are particularly relevant in the context of IoT (Kamalinejad et al., 2015; Gorlatova et al., 2015), which require novel tailored antenna and circuit designs (Bi et al., 2015; Piñuela et al., 2013).

2. *Communications.* A significant portion of IoT devices will be based on low-cost hardware with low computation capabilities and inaccurate clocks. Novel communication protocols with low complexity that can maintain reliable links in such scenario are required. Current communication networks are designed and optimised for the traffic generated by a moderated number of users of high data-rate services (e.g., video streaming, interactive gaming, enhanced web browsing). IoT is expected to introduce a much higher number of users (devices) generating low data-rate traffic, which will represent a radical change in current network traffic patterns. Existing network infrastructures and communication protocols may be inefficient for these new traffic patterns and new protocols and solutions will be required. Moreover, many devices will be connected to the IoT via wireless links based on a wide variety of radio access technologies. Forecasts of billions of IoT devices in the foreseeable future along with crowded frequency allocation charts claim for extremely efficient ways to access the spectrum. The introduction of spectrum sharing approaches based on dynamic spectrum access and cognitive radio constitutes a promising solution in the context of IoT, however this requires the identification of appropriate frequency bands of operation and the development of reliable mechanisms to enable interference-free coexistence among radio communication systems.

3. *Data processing.* With billions of devices generating data, efficient data processing methods are of paramount importance. Given the expected size and complexity of future date sets, traditional data processing approaches are unlikely to be suitable. Novel data processing solutions to extract relevant and useful information, possibly by looking at potential relations between apparently unrelated data, may lead to the discovery of surprising connections with the potential to provide new applications. In this context, it is necessary not only to review existing techniques from diverse domains (e.g., data mining, artificial intelligence, machine learning, database systems, statistics), analysing their suitability in the context of large-scale IoT, but also develop new tailored solutions.

When facing the above mentioned challenges, it is important to develop solutions that can address these problems not only individually but also from a multidisciplinary perspective, taking into account the possible mutual effects and interactions among the considered dimensions (i.e., hardware design, communications, and data processing) and providing a joint system optimisation. An accurate and realistic evaluation of such solutions requires a suitable platform. While mathematical analyses and software simulations may be suitable for the evaluation of certain individual aspects of the system, the development of an adequate prototype would enable a comprehensive and more realistic performance evaluation, including possible relations among multidisciplinary aspects of the system that would be difficult or impossible to capture with mathematical analyses or software simulations, and enabling a joint optimisation.

In this context, this paper presents a prototype developed in the framework of the EPSRC/eFutures-funded project "Internet of Surprise: Self-Organising Data". This prototype has been designed to effectively enable the joint evaluation of multidisciplinary aspects of future IoT systems. In particular, this paper provides an in-depth description of the hardware and software components, discussing design and implementation details that may be helpful to other researchers and engineers in the development of similar tools. Examples illustrating the potentials and capabilities of the developed platform are presented as well. The developed prototype provides researchers and engineers with a fully functional tool for proof-of-concept, validation and cross-layer optimisation of multidisciplinary solutions before bringing them to real IoT deployments.

The rest of this paper is organised as follows. First, Section 2 provides a general overview of the developed prototype. The three main parts or subsystems integrating the prototype are then described in detail in Sections 3, 4 and 5. Some examples illustrating the capabilities and studies enabled by the developed prototype are presented in Section 6. A discussion of the development process of the prototype along with problems faced and lessons learned that may be helpful in the development of similar tools is provided in Section 7. Finally, Section 8 summarises and concludes the paper. The abbreviations used in this paper are listed in Table 1.

2. Overview

The developed prototype is composed of three main parts or subsystems as shown in Fig. 1, namely an *IoT subsystem*, a *coexisting radio subsystem*, and a *spectrum monitoring subsystem*. The IoT subsystem emulates an IoT network composed of a number of IoT sensor nodes generating data that are gathered and processed by a central processing unit. The coexisting radio subsystem represents an additional radio communication system composed of a transmitter and two receivers, all of them operating in the same frequency band as the IoT subsystem (this subsystem may coexist in an interference-free manner with the IoT subsystem or generate certain interference patterns). Finally, the spectrum monitoring subsystem is used to monitor the spectral activity in the frequency band shared by the other two subsystems. Sections 3, 4 and 5 provide a more detailed description of each subsystem, including the motivation, design and hardware/software implementations.

3. IoT Subsystem

The IoT subsystem emulates an IoT network, including a number of nodes equipped with different types of sensors that

Table 1: List of abbreviations used in this paper.

ACLR	Adjacent Channel Leakage Ratio
ACPR	Adjacent Channel Power Ratio
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
CoAP	Constrained Application Protocol
CSI	Camera Serial Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSI	Display Serial Interface
FFT	Fast Fourier Transform
GPIO	General-Purpose Input/Output
I2C	Inter-Integrated Circuit
IoT	Internet of Things
IP	Internet Protocol
I/Q	In-phase/Quadrature
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardisation
MIMO	Multiple-Input Multiple-Output
MQTT	Message Queuing Telemetry Transport
PCB	Printed Circuit Board
PIR	Passive Infra-Red
PLL	Phase-Locked Loop
QPSK	Quadrature Phase Shift Keying
RGB	Red/Green/Blue
RSS	Received Signal Strength
SCL	Serial Clock Line
SDA	Serial Data line
SDR	Software-Defined Radio
SISO	Single-Input Single-Output
SPI	Serial Peripheral Interface
SRRC	Square Root Raised Cosine
SSID	Service Set IDentifier
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UHD	USRP Hardware Driver
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
XMPP	eXtensible Messaging and Presence Protocol

generate a diverse variety of data and a central processing unit that gathers and processes the data generated by the sensor nodes in order to extract relevant information. This is the main subsystem of the prototype since all IoT-related functions are implemented and evaluated in this subsystem.

3.1. Hardware Implementation

The central processing unit is a conventional computer that performs advanced data-processing operations on the raw data generated by the sensor nodes. This computer is the core of the IoT subsystem and where the intelligence is implemented (more details will be provided in Section 3.2).

The IoT sensor nodes are equipped with a wide variety of sensors that generate different types of data. The current implementation comprises four IoT sensor nodes (from Node A to Node D), however the number of nodes can be extended in a straightforward manner as the overall IoT subsystem is in fact

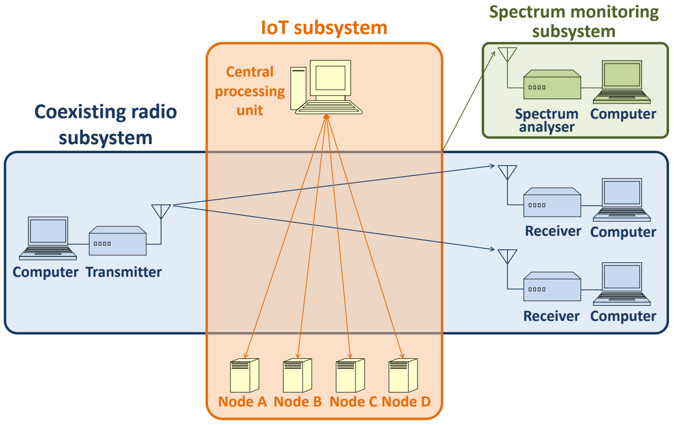


Figure 1: Overview of the developed prototype.

designed to be easily scalable (see Section 7 for a detailed discussion). Two options were considered in the design of the sensor nodes, namely a *scenario-specific* design with a particular application scenario in mind where the nodes would include a specific set of sensors as required by the considered scenario, and a *scenario-agnostic* design where the sensor nodes would include a generic set of sensors without any particular scenario in mind. The latter option was finally selected since it was considered to be adequate as a proof-of-concept of the envisaged IoT system and it would not constrain the applicability of the prototype in other possible studies as it might be the case of the former option. Consequently, the sensor nodes were equipped with an arbitrary set of sensors embracing a wide variety of physical parameters.

Fig. 2 shows one of the four implemented sensor nodes. Each sensor node is composed of six physical sensors (elements 1 to 6) connected to a Raspberry Pi minicomputer (element 7), which connects to the central processing unit by means of a USB wireless/WiFi adapter (element 8). The sensors included in each IoT node in the current implementation are as follows:

1. Momentary capacitive touch sensor based on the AT42QT1010 sensor (Adafruit 1374). This sensor provides a logical high output when touched by the user, and a logical low output otherwise. This sensor can be used to generate on/off patterns manually.

2. Toggle capacitive touch sensor based on the AT42QT1012 sensor (Adafruit 1375). This sensor alternates between high and low output states when touched by the user and can also be used to generate on/off patterns manually.

3. Passive infra-red (PIR) motion sensor HC-SR501. This sensor provides a logical high output when motion is detected within a range of 7 metres and a 120-degree angle. Once triggered the output remains high for an interval of 5 seconds (adjustable).

4. Light sensor based on the TSL2591 sensor (Adafruit 1980). This high-range luminosity sensor contains infra-red and full-spectrum diodes that can provide luminance

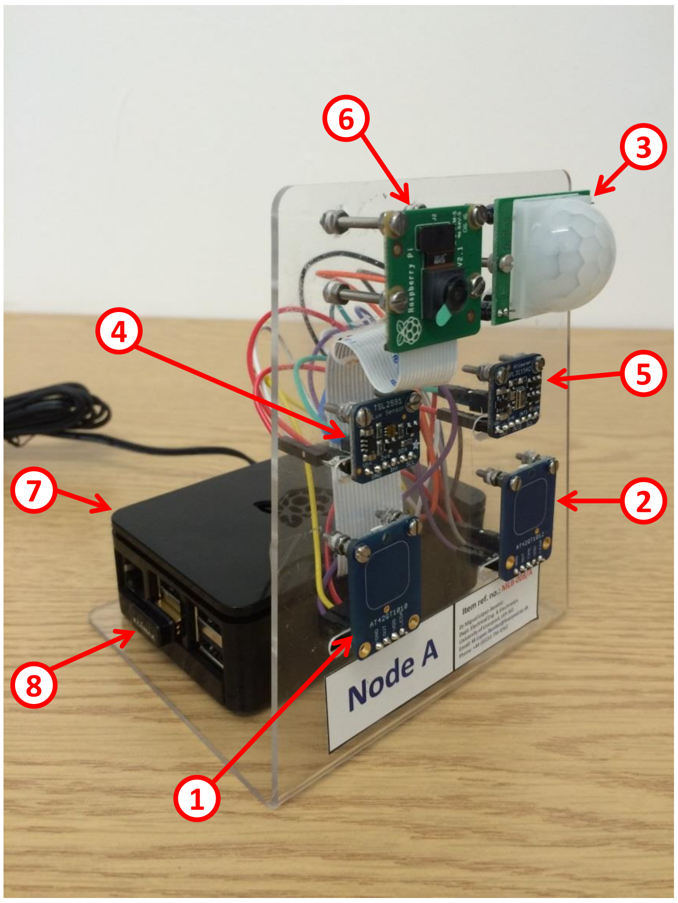


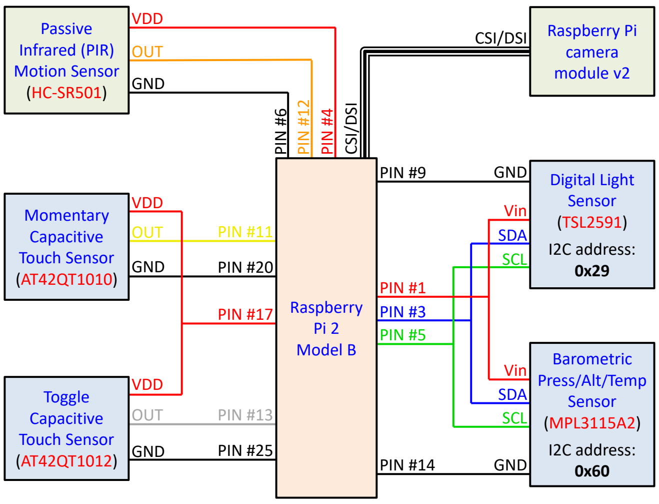
Figure 2: IoT sensor node.

measurements (in lux) of the infra-red and full-spectrum light. Both values can be used to compute the luminance in the human-visible spectrum (as the difference between both) and the corresponding luminosity perceived by the human eye (using an empirical formula). In practice, this physical sensor integrates four logical sensors.

5. Barometric pressure, altitude and temperature sensor based on the MPL3115A2 sensor (Adafruit 1893). This sensor contains a barometric pressure sensor that provides pressure measurements (in Pascals) and the equivalent altitude (in metres) along with temperature measurements (in Celsius).

6. Raspberry Pi camera module version 2, based on the Sony IMX219 8-megapixel image sensor. This module can provide 3280×2464 pixel static images and supports 1080p30, 720p60 and 640×480p90 video.

The six physical sensors are connected to a Raspberry Pi 2 Model B (element 7 in Fig. 2) as shown in Fig. 3. Except the camera module, which uses its own camera/display serial interface (CSI/DSI), the rest of sensors are connected via the general-purpose input/output (GPIO) pins. While sensors 1-3 provide simple logical binary (high/low) outputs and can be connected to standard logical input pins, sensors 4-5 are



Passive Infrared (PIR) Motion Sensor (HC-SR501)
VDD
OUT
GND
PIN #6
PIN #12
PIN #4
CSI/DSI
CSI/DSI
Raspberry Pi camera module v2
Momentary Capacitive Touch Sensor (AT42QT1010)
VDD
OUT
GND
PIN #11
PIN #20
PIN #17
Toggle Capacitive Touch Sensor (AT42QT1012)
VDD
OUT
GND
PIN #13
PIN #25
Raspberry Pi 2 Model B
PIN #9
PIN #1
PIN #3
PIN #5
PIN #14
Digital Light Sensor (TSL2591)
I2C address: 0x29
GND
Vin
SDA
SCL
Barometric Press/Alt/Temp Sensor (MPL3115A2)
I2C address: 0x60
Vin
SDA
SCL
GND

Figure 3: Connection of physical sensors to the Raspberry Pi 2 Model B.

based on the I2C bus protocol and are therefore connected to the SDA/SCL pins. All sensors are powered from 3.3-volt pins so that the logical outputs provide a voltage compatible with that of the GPIO pins (i.e., 0 volts for logical low and 3.3 volts for logical high), except the motion sensor which requires at least 5 volts (but provides a 0/3.3-volt output). Table 2 shows the maximum current consumption of each sensor. Due to the limitations of the internal 3.3-volt voltage regulator of the Raspberry Pi, the GPIO pins can safely draw a maximum of 50 mA distributed across all 3.3-volt pins (including input, output, and 3.3-volt power pins). The maximum current consumption of the sensors connected to these pins (1, 2, 4 and 5) is around 6 mA, and the current through any individual pin does not exceed the 16 mA limit. The motion sensor is powered via a 5-volt pin and the camera module is powered via the CSI/DSI interface; for these two sensors there is no current limit other than that of the main power supply. The maximum current consumption of the Raspberry Pi 2 Model B board under stress conditions (including USB devices) is 820 mA[1], which along with the camera module (250 mA), motion sensor (65 mA) and rest of sensors (6 mA), leads to a maximum consumption of 1141 mA per sensor node (2000 mA power supply is used).

IoT sensor nodes are identical and implemented as detailed above. The connectivity between the IoT nodes and the central processing unit is accomplished by means of IEEE 802.11 wireless links. To this end, the IoT nodes and the central processing unit are equipped with 2.4 GHz USB wireless adapters Edimax EW-7811UN (element 8 in Fig. 2), which offer a wide range of configuration options for a fine tuning of the radio opera-

[1]`http://www.raspberrypi.org/help/faqs/#powerReqs`

[*]Obtained by interpolating the consumption values provided in the datasheet (378.5μA@3V and 542.5μA@4V, i.e. 427.7μA@3.3V) and including the current consumption of the LED (1.6mA).

[†]Obtained by interpolating the consumption values provided in the datasheet (59μA@3V and 88μA@4V, i.e. 67.7μA@3.3V) and including the current consumption of the LED (1.6mA).

[‡]Maximum current consumption when actively sensing.

[§]Current consumption during acquisition and conversion.

Table 2: Maximum current consumption of each sensor.

Sensor	Maximum (mA)
Momentary touch sensor (1)	2.03 [†]
Toggle touch sensor (2)	1.67 [‡]
Light sensor (4)	0.4 [§]
Press. / alt. / temp. sensor (5)	2 [¶]
Motion sensor (3)	65
Camera (6)	250

tion conditions. The prototype offers the possibility to employ other antennas by means of an external adapter (Alfa Network AWUS036H), which provides an SMA connector for external antennas (the current implementation uses whip, microstrip and flexible PCB antennas but any antenna design can actually be employed). This feature is particularly appealing when evaluating a particular antenna design and its potential impact on other components of the system, both at the link and system level. For example, flexible antennas may be suitable for certain IoT devices (e.g., wearables) but their radiation pattern, directivity and other properties may be affected by the different positions to which they may be subjected, which may also have an impact on the quality of the wireless link (and interference patterns) and affect ultimately the data processed at the application layer.

3.2. *Software Implementation*

The IoT network can be configured in different ways in order to meet particular needs. Communication between the central processing unit and the IoT sensor nodes relies on Matlab features for remote access of Raspberry Pi computers, which is based on TCP/IP. This means that the IoT sensor nodes employed in the prototype could be virtually anywhere on the world and would be able to communicate with the central processing unit as long as they have a valid IP address. For convenience, the IoT network in the prototype is configured as a local ad-hoc network with private IP addresses where the central processing unit acts as an access point to which the IoT nodes are connected as terminals. This is a simple and convenient setup for experiments in a laboratory environment and moreover represents a realistic configuration since many future IoT devices are likely to be connected to the Internet via wireless access points or similar network infrastructures.

The central processing unit runs Microsoft Windows 7 operating system and employs the *Microsoft Virtual WiFi Miniport Adapter* technology to provide connectivity as an access point to the IoT sensor nodes. This technology enables a physical wireless adapter to be virtualised into several logical adapters, which can be useful for example to access several WiFi networks with the same physical adapter or to share an Internet connection with other wireless devices. This technology is used here to configure the central processing unit as an access point to which the IoT sensor nodes can connect, thus enabling the direct communication among nodes in the IoT network.

The central processing unit is configured as an access point by running the following two commands (as administrator):

```
1) netsh wlan set hostednetwork mode=allow
ssid=IoT-testbed key=eFutures
2) netsh wlan start hostednetwork
```

The first command configures the central processing unit as an access point hosting a network with name/service set identifier (SSID) `IoT-testbed` and access password `eFutures`. The second command starts the network. Once the hosted network has been started, any WiFi-equipped device will be able to connect to the network `IoT-testbed` (using the password `eFutures`) and communicate with other devices connected to the same network. This network is used to enable communication between the central processing unit and the IoT sensor nodes. For this to be possible, the IoT sensor nodes are configured to automatically search and connect to this network at startup. This is easily accomplished by adding to the file `/etc/wpa_supplicant/wpa_supplicant.conf` of the Raspbian operating system running on the IoT sensor nodes the following lines:

```
network={
ssid="IoT-testbed"
psk="eFutures"
}
```

This information will be employed by the IoT sensor nodes to connect automatically to the IoT network hosted by the central processing unit at startup. By default, this network will use private IP addresses in the range `192.168.137.0/24`, where the default gateway IP address `192.168.137.1` is assigned to the central processing unit (since Window's hosted network functionality is intended to share a computer's Internet connection with other wireless devices) and the rest of network nodes (i.e., the IoT sensor nodes) are assigned random IP addresses by the central processing unit via Dynamic Host Configuration Protocol (DHCP). To facilitate the identification of the IoT nodes in the network, they are configured with predefined host names following the format `RBPi2-Node-X` where `X` can be `A`, `B`, `C` or `D`. Since a Windows hosted network will have by default a domain name `mshome.net`, the IoT sensor nodes can be accessed remotely by their complete host/domain name `RBPi2-Node-X.mshome.net` regardless of the IP address assigned by the central processing unit - the Domain Name System (DNS) protocol will translate each host/domain name to the corresponding IP address. This approach removes the need to configure and manage fixed IP addresses, which may lead to some issues for example when adding or removing nodes.

Communication between the central processing unit and the IoT sensor nodes currently relies on Matlab functions for remote access of Raspberry Pi computers. This feature of Matlab simplifies the remote access to the IoT sensor nodes, however it has some limitations, in particular regarding the ability of the IoT nodes to initiate or trigger certain events and communications. While the current implementation based on Matlab's communications features is sufficient for experimentation and proof-of-concept, future plans to extend the prototype include the implementation of communication protocols that remove the above mentioned limitations such as the Message Queuing Telemetry Transport (MQTT) protocol. MQTT is an ISO standard (ISO/IEC PRF 20922) (ISO/IEC, 2016) messaging transport protocol based on a client-server publish-subscribe model that runs on top of TCP/IP and is light weight, open, simple, and designed to be easy to implement, which is particularly important in low-cost low-complexity IoT nodes where a small code footprint is required and/or network bandwidth may be limited. A variation for embedded devices on non-TCP/IP networks, such as ZigBee, is available as well (Stanford-Clark and Truong, 2013). Other protocols include Advanced Message Queuing Protocol (AMQP) (ISO/IEC, 2014; Vinoski, 2006), Constrained Application Protocol (CoAP) (Shelby et al., 2014), and eXtensible Messaging and Presence Protocol (XMPP) (Saint-Andre, 2011a,b, 2015, 2014b,a).

Once network connectivity is established, communication in the current implementation takes place on a point-to-point basis between the central processing unit and each of the IoT sensor nodes as depicted in Fig. 1. While it would be possible to emulate other network topologies (e.g., forwarding a message from one IoT sensor node to another via the central processing unit), this is currently not needed and, as mentioned above, the current implementation is sufficient for experimentation and proof-of-concept. Nevertheless, the implementation of communication protocols as the ones mentioned above may extend the prototype capabilities by enabling direct node-to-node links without involving the central processing unit.

Communication between the IoT sensor nodes and the central processing unit takes place via direct wireless links. In a real IoT network, certain fixed network infrastructure is likely to be present between both, which may result in packet delays and packet losses depending on the network congestion situation. A computer could be introduced in the prototype between the IoT sensor nodes and the central processing unit to capture TCP/IP packets sent by the IoT sensor nodes and selectively delay or discard some of them in order to reproduce network congestion situations. However, a simpler approach to achieve a similar effect is to selectively delay (in the central processing unit) the data reports from the IoT nodes before processing them, or discard them, according to a suitable packet delay/loss model (Lakshman and Madhow, 1997; Altman et al., 2005), which removes the need for an additional computer.

The central processing unit first establishes a TCP/IP connection with each IoT sensor node. Once an individual network connection is established with each node, the central processing unit configures (for each node) the GPIO pins taking into account the connections shown in Fig. 3, creates an I2C connection with the light and pressure/altitude/temperature sensors based on their I2C addresses (see Fig. 3), configures the I2C sensors (in particular: the gain for the light sensor can be configured as low/1x, medium/25x, high/428x and maximum/9876x; the integration time of the light sensor can be configured from 100 ms for bright light to 600 ms for dim light in increments of 100 ms; and the oversampling ratio for the pressure/altitude/temperature sensor can be configured from 1 to 128, which leads to minimum time intervals between samples from 6 to 512 ms), and finally creates a connection with the cameras and configures their parameters (resolution, image quality, rotation, horizontal/vertical flip, frame rate, brightness, contrast, saturation, sharpness, exposure/exposure compensation modes, and image effects).

Once IoT sensor nodes are configured, the central processing unit can start gathering data from the sensors. Each sensor node can provide camera snapshots and videos in addition to the reading from a total of ten sensors, namely momentary touch sensor state, capacitive touch sensor state, motion detection state, luminance (in lux) in the infra-red/human-visible/full spectrums and human eye perception (based on an empirical formula), pressure (in Pascals), altitude (in metres), and temperature (in Celsius). Note that this represents a network with an effective total of 40 sensors (distributed among 4 sensor nodes) in addition to the picture/video sensors of each node. In order to emulate different scenarios that can be found in IoT networks, the prototype offers the possibility to gather data in the following ways:

- *Synchronous full data reporting.* In this mode of operation, the central processing unit collects data periodically from all sensors in all nodes. A configurable data polling period determines the time interval between two consecutive data polling events. In every data polling event, the central processing unit requests a reading from each of the 10 sensors installed in each of the 4 IoT sensor nodes. This mode of operation emulates an IoT network where all sensors provide periodic data reports in a synchronous way.

- *Synchronous partial data reporting.* This mode of operation is similar to the synchronous full data reporting mode, with the exception that only a certain percentage (lower than 100%) of the sensors available in the network provide a data report in every periodic data polling event (if the percentage is configured as 100%, then this mode of operation is equivalent to the synchronous full data reporting mode). In this mode, the central processing unit computes first the number of sensors to be polled (i.e., the number of data polling events) based on the selected percentage of sensors and the total number of sensors available in the network. In each data polling event, one of the four IoT sensor nodes is selected randomly (by generating a random integer number from 1 to 4) and then one of the ten sensors available in that node is selected randomly (by generating a random integer number from 1 to 10). The process is repeated until the required number of sensors per data polling event is reached. This mode of operation emulates an IoT network where sensors provide periodic data reports in a synchronous way but not all sensors have new data to report in every data reporting event.

- *Asynchronous data reporting.* In this mode of operation the time instants of the data reporting events are individually decided for each sensor available in the network based on a Poisson point process with an individually configurable interarrival time for each sensor, or another suitable model. The central processing unit checks constantly the polling time scheduled for each sensor in the network. When the polling time for a particular sensor is reached, the sensor is polled (i.e., a data report is provided by the sensor) and the next polling time is generated based on the corresponding Poisson point process or the employed model. This mode of operation emulates an IoT network where sensors provide data reports asynchronously.

These three data reporting modes enable a wide variety of network traffic conditions and, subsequently, many interesting experiments. For example, consider the case of a temperature sensor that is configured to provide a new data report (in this case, a new temperature reading) every time the temperature differs from the last data report by $\pm 1^{\circ}$C. In this case a new data packet would be transmitted through the IoT network every time a relative temperature change of $\pm 1^{\circ}$C occurs. This setup would be useful in the development of traffic models for IoT networks associated to physical magnitudes such as temperature, pressure, light, human motion patterns, etc. Moreover, depending on the underlying communication protocol (e.g., MQTT over TCP/IP) this would also lead to a certain radio transmission pattern in the radio interface (which could be captured by the spectrum monitoring subsystem explained in Section 5) that can also be modelled and characterised for the development of efficient methods to avoid interference and enable spectral coexistence with other existing radio communication systems (as explained in more detail in Section 4). These are just some examples of the type of experiments in the context of IoT enabled by the developed prototype. Detailed examples will be provided in Section 6.

The data collected from the sensors available in the network is processed in the central processing unit, where the intelligence is implemented, using Matlab. Matlab was selected given its comprehensive set of powerful and versatile functions that enable sophisticated data operations and simplifies the processing of complex data sets. Some illustrative examples on novel data processing methods evaluated with the developed prototype are provided in Section 6.1.

When an experiment is finished, the prototype is shut down by following two steps. First, the IoT sensor nodes are shut down remotely from the central processing unit by running the following command for each node:

```
start /B plink.exe -l pi -pw raspberry
RBPi2-Node-X.mshome.net sudo poweroff
```

where `X` is replaced with `A`, `B`, `C` and `D`. This command uses Plink[2] (a free and open-source command-line network connection tool) to shut down each node individually. Finally, the IoT network is stopped by running the command:

```
netsh wlan stop hostednetwork
```

which stops the hosted network in the central processing unit.

[2]`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

4. Coexisting Radio Subsystem

As mentioned in Section 1, a significant amount of devices are expected to be connected to the IoT via wireless links using different types of radio access technologies such as IEEE 802.11 WLAN (WiFi) networks or mobile communication networks. Forecasts of billions of IoT devices in the foreseeable future indicate that there will be a high demand for spectrum, however the frequency allocation charts of most countries are currently overcrowded. In this context, spectrum sharing approaches based on dynamic spectrum access and cognitive radio techniques are particularly promising in the context of IoT and the availability of adequate platforms to assess the performance of new spectrum access approaches is of great importance. The developed prototype includes a coexisting radio subsystem operating in the same frequency band as the IoT subsystem, i.e., the 2.4 GHz Industrial, Scientific and Medical (ISM) band. While the IoT subsystem is constrained to the IEEE 802.11 communication standard, the coexisting radio subsystem relies on a flexible design based on Software-Defined Radio (SDR) devices that can virtually implement any radio communication technology. This subsystem can be configured to coexist in an interference-free manner with the IoT subsystem or generate certain interference patterns. This subsystem extends significantly the range of experiments that can be conducted with the developed prototype by introducing a spectrum-sharing dimension and enabling the study of interactions of an IoT network with other communication systems as well as the analysis of the resulting impact at higher layers.

4.1. Hardware Implementation

The coexisting radio subsystem is composed of one transmitter and two receivers. The transmitter (receiver) is implemented with USRP B200 (B210) SDR devices, respectively. While the USRP B200 is equipped with one transceiver, the USRP B200 includes two transceivers, which enables the implementation and evaluation of diversity reception techniques at the receivers. Moreover, one of the USRP B210 units can be used as a transmitter in order to introduce Multiple-Input Multiple-Output (MIMO) techniques (the two USRP B210 channels can be used as two synchronised transmitters and/or receivers, thus enabling a full 2×2 MIMO system, or higher order if synchronised with other units).

In reception, USRP devices capture and amplify the radio-frequency signal received by the antennas (dual-band 2.4/5 GHz $\lambda/2$ whip antennas, model Mobilemark PSKN3-24/55S), performs down-conversion to a lower intermediate frequency at which the signal is sampled at a configurable sample rate up to 61.44 MS/s with a resolution of 12 bits, and decimates the samples to baseband before sending them (in complex I/Q format) to a host computer via a USB connection. A computer implementing the receiver in software captures and processes the signal samples. In transmission, the computer implements in software a transmitter that generates signal samples, which are interpolated by the USRP to the appropriate intermediate frequency, converted to the analog domain and up-converted to the radio-frequency for transmission. USB 3.0 connections allow up to 56 MHz of instantaneous bandwidth in SISO (1x1) mode and up to 30.72 MHz in MIMO (2x2) mode (USB 2.0 connections allow a maximum rate of 8 MS/s and an approximate maximum usable bandwidth of 7-8 MHz).

When experiments are conducted in an indoor environment (in particular inside the same room), transmission/reception power limitations should be observed to ensure that all devices operate under safe conditions. The USRP B200/B210 can transmit at a maximum output power of 20 dBm and tolerates a maximum input power of -15 dBm. The receiver might be damaged if placed too close to the transmitter. A radio propagation model suitable to the scenario of operation can be used to determine the minimum separation distance between transmitter and receivers. According to the indoor pathloss model of the recommendation ITU-R P.1238-8 (ITU-R, 2015), for a frequency of operation of 2400 MHz (lowest point and worst case of the 2.4 GHz ISM band), a power loss coefficient of 30 (office environment at 2.4 GHz), and devices operating in the same floor, a distance of 1 metre (minimum distance for which the model is valid) provides an attenuation of 40 dB, which is greater than the minimum 35 dB required to guarantee a safe operation at maximum transmission power.

4.2. Software Implementation

USRP is a very flexible SDR platform that supports many popular software applications such as GNU Radio, LabVIEW, MATLAB and Simulink as well as the development of tailor-made solutions based on the API of the USRP Hardware Driver (UHD), which provides native support for C++. There exists a large number of open-source projects embracing a broad range of radio technologies, services and applications, which could be employed as a coexisting radio subsystem in the prototype.

For proof-of-concept studies and demonstrations, the coexisting radio subsystem currently implements in Matlab a simple communication system designed for the transmission of a unidirectional and constant flow of data. Source data are interpreted as a sequence of bytes that are converted into bits for transmission using Quadrature Phase Shift Keying (QPSK) modulation. Data are transmitted in frames of 200 bits. The first 26 bits are header bits used for frame synchronisation at the receiver. The header bits are obtained by oversampling by two a 13-bit Barker code (i.e., repeating the Barker code bits twice), which ensures the transmission of the Barker code on both in-phase (I) and quadrature (Q) components of the QPSK-modulated signal. The remaining 174 bits are used for data transmission and can be configured in a flexible manner.

Fig. 4 shows an example of frame format for the transmission of true-colour (24-bit) images (although the same format can be used or easily adapted for the transmission of any other type of data). A true-colour image with a height of N_h pixels and a width of N_w pixels can be represented as a $N_h \times N_w \times 3$ matrix containing three sub-matrices, each of which represents the corresponding values for the red (R), green (G), and blue (B) colour components for each pixel. Each pixel value is coded with 8 bits (numerical range from 0 to 255). Thus, the transmission of one pixel requires 24 bits. Each frame contains 5 consecutive pixels as shown in Fig. 4, which are read from the source image left-to-right and top-to-bottom. The 5 pixels (120 bits) are encoded with a Hamming(15,11) code to protect the information against transmission errors. This code has a rate of $11/15 \approx 0.73$ and outputs 165 bits, which constitutes the payload of the frame. The remaining 9 bits are padded with random bits. The payload and padding bits are scrambled over the data field of the frame using the generator polynomial $p(x^{-1}) = 1+x^{-1}+x^{-2}+x^{-4}$. The purpose of this is twofold. First, transmission errors typically occur in bursts of consecutive bits and the effectiveness of error-correcting codes in general decreases when the erroneous bits are close to each other. The descrambling performed by the receiver spreads the erroneous bits, thus increasing the effectiveness of the employed error-correcting code. Secondly, scrambling guarantees a balanced distribution of zeros and ones, which facilitates the timing recovery at the receiver.

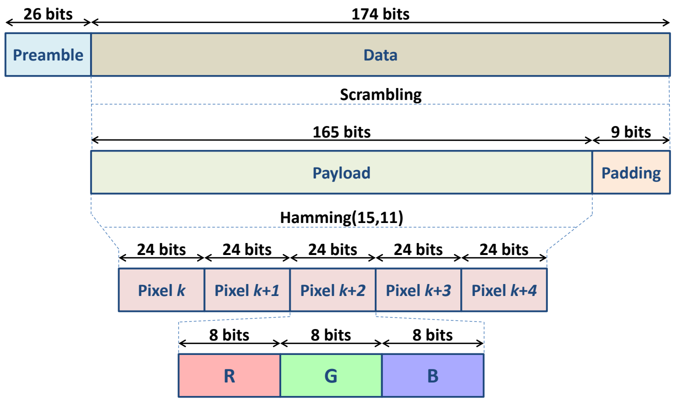


Figure 4: Link level frame format for the transmission of images.

Frames are modulated using QPSK (with Gray mapping), upsampled by a factor of 4, and filtered with a Square Root Raised Cosine (SRRC) filter (to minimise inter-symbol interference) with a roll-off factor of 0.5 and a span of 10 symbols (i.e., 40 samples). The filtered samples are sent to the USRP transmitter, which is configured to operate at a sample rate of 20 MS/s. Since this sample rate may be excessive for real-time operation with an average computer, the output of the SRRC filter is set at 200 kS/s and the samples are interpolated by a factor of 100. This leads to an effective data rate of $50 \cdot 10^3$ QPSK-modulated symbols per second (i.e., 100 kbps), which can be increased for powerful computers by reducing the interpolation factor. The USRP transmitter operates at a configurable frequency within the 2.4 GHz ISM band, which can be selected (along with the transmission power) to generate specific interference patterns.

In order to recover the original image, the receiver needs not only to revert the operations performed by the transmitter but also compensate the impairments of the wireless channel. To achieve this, the receiver includes Fast Fourier Transform (FFT)-based coarse frequency compensation, Phase-Locked Loop (PLL)-based fine frequency compensation, timing recovery with fixed-rate re-sampling and bit stuffing/skipping, frame synchronisation, and phase ambiguity resolution (implementation details are omitted for brevity, see Rice (2008) for details). Moreover, the receiver also needs to identify the beginning of a new image (i.e., the pixel on the top-left corner of the image). In order to facilitate this, the transmitter sends a predefined number of frames with the 120 bits of the message set to zero to indicate the beginning of a new image. The first *non-zero frame* corresponds to the first 5 pixels of the image. Note that this sequence of zeros is equivalent to 5 consecutive black pixels (the black colour is encoded as [0,0,0] in the RGB space). Therefore, an image with many black pixels may be incorrectly decoded at the receiver. To prevent this problem, all zero values in the original image are replaced with a value of 1 (the difference is not noticeable to the human eye).

The implemented coexisting radio subsystem has deliberately been kept as generic and simple as possible in order to enable the transmission of any type of data (images, text, files, etc.) in real-time (which can be achieved with an adequate configuration and using Matlab compiler to produce code for accelerated execution). While other more sophisticated systems are freely available, the implemented system is enough to investigate coexistence issues between IoT and legacy radio systems operating in the same frequency band and analyse its impact on the performance of both systems (from the physical to the application layer) as it will be shown in Section 6.

5. Spectrum Monitoring Subsystem

The purpose of the spectrum monitoring subsystem is to monitor and record (for off-line analysis) the spectral activity in the frequency band shared by the other two subsystems. This can be useful to understand how several systems and radio technologies coexist in the same frequency and identify particular interference patterns between systems. Spectrum analysers can provide relevant information such as effective channel occupied bandwidth, spectral masks, Received Signal Strength (RSS), channel power measurements for several radio technologies, Adjacent Channel Power/Leakage Ratio (ACPR/ACLR), etc. It can also be employed to understand how different systems react to interference (e.g., moving to a different radio channel) or associate certain errors observed in one system at a particular time instant with interference from another system that was active at the same time instant.

This subsystem does not require a specific implementation since a standard spectrum analyser can be employed. Some of the illustrative results provided in Section 6 were obtained using an Aaronia Spectran HF-60105 V4 X spectrum analyser with the same antenna used by the transmitter and receivers of the coexisting radio subsystem (dual-band 2.4/5 GHz $\lambda/2$ whip antenna, model Mobilemark PSKN3-24/55S). This is a low-cost USB spectrum analyser and despite some performance limitations it is sufficient for the purposes of the prototype.

6. Illustrative Examples

This section presents some examples illustrating the potentials and capabilities of the developed prototype in the study of future IoT systems. The developed prototype is flexible enough to embrace a wide variety of experiments in the context of IoT systems and the examples shown in this section is just a small subset of possible experiments. Other types of studies and experiments enabled by the developed prototype, although not shown here, are discussed in Section 1.

6.1. Smart IoT Data Processing

IoT systems will soon be overloaded by data from billions of IoT objects. Data sets in future IoT systems are expected to be so large and complex that traditional data processing approaches are deemed inadequate, thus claiming for innovative solutions to extract relevant and useful information. This first example shows how the prototype can be employed to explore new smart IoT data processing methods in order to extract meaningful information from the received data. The implemented system effectively counts with a high number of sensors as discussed in Section 3.2 (40 sensors distributed among 4 sensor nodes in addition to the picture/video sensors of each node). Fig. 5 shows a screenshot of one of the several visualisation screens implemented in the central processing unit of the IoT subsystem. Each subfigure shows in real-time the history of measurements reported by each sensor node for a particular parameter (the last 20 measurements for each sensor are shown). This shows how even a relatively low/moderate number of sensors can produce a significant amount of data and highlights the need and importance of developing adequate and efficient smart IoT data-processing methods.

In this illustrative example, a simple yet insightful method for data processing inspired by Information Theory is implemented. An important finding of Information Theory is that the amount of information conveyed by a message or event is inversely proportional to its probability of occurrence. Therefore the more *surprising* (i.e., unlikely) a message is, the more information it contains (Shannon, 1948a,b). In scenarios with high volumes of data the value will not come from the volume of traffic but from finding unexpected or *surprising* new trends or events. If the information is exactly as expected, then it likely needs no special handling. Only in unexpected scenarios is intervention required. If data processing is constrained to find information that is already known to look for, the potential for surprise is reduced and the full potential output from the data is not reached. However, by looking for emergent properties, surprising connections may be found.

Every sensor in an IoT system can be regarded as a source of messages that carry some information (for example, a temperature sensor will send reports/messages with the measured temperature values). Even if the measured magnitude is continuous (e.g., temperature), the sensor can only emit a finite discrete set of possible different messages that depends on the sensor capabilities (e.g., minimum/maximum measurable temperatures, resolution, and quantisation step). According to the well-known definition from Information Theory, the amount of information $\mathcal{I}$ carried by a message x_k from a set/alphabet $\{x_k\}_{k=1}^{K}$ of K mutually-exclusive messages is inversely proportional to

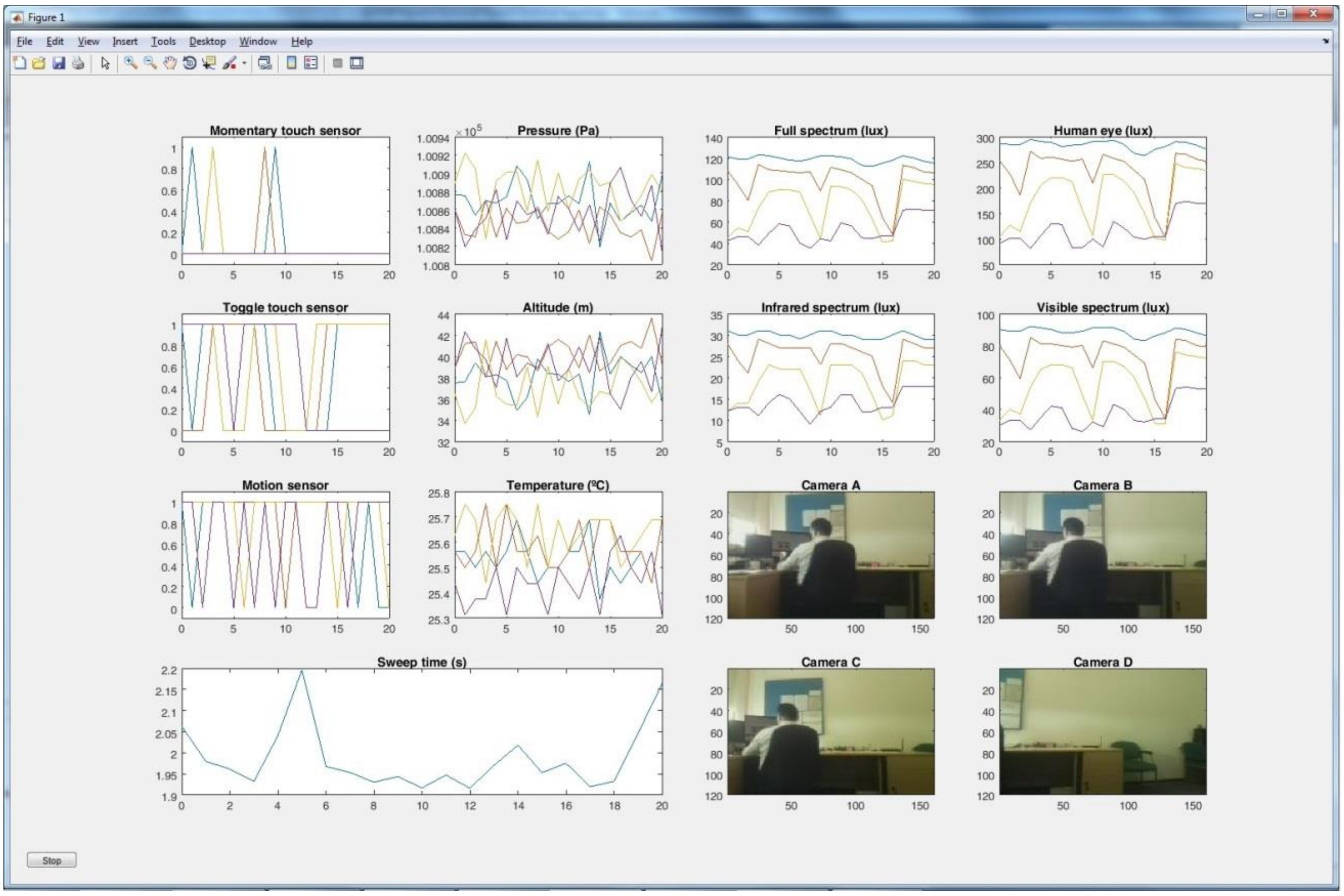


Figure 5: Screenshot showing the real-time monitoring of all the available sensor data.

its probability of occurrence $P(x_k)$ and is formally defined as:

$$\mathcal{I}(x_k) = \log_2\left(\frac{1}{P(x_k)}\right) \tag{1}$$

where $k = 1, \ldots, K$, and $\sum_{k=1}^{K} P(x_k) = 1$. The base of the logarithm in (1) determines the units of $\mathcal{I}$ (*bits* for base 2). Eq. (1) provides a quantitative definition of the amount of information carried by a message as a function of its probability of occurrence (e.g., the lower the probability of occurrence of a message is, the more information it conveys).

Eq. (1) considers the case of a single source of information (i.e., a single sensor that generates messages from a set/alphabet of K possible messages). The developed prototype implements multiple sensors, each of which can be seen as an individual source of information. To account for the multi-sensor case, (1) can be rewritten with a slightly different notation as:

$$\mathcal{I}(x_{m,n}) = \log_2\left(\frac{1}{P(x_{m,n})}\right) \tag{2}$$

where the index $m = 1, \ldots, M$ identifies an individual sensor (M is the number of sensors in the system, i.e., $M = 40$ in the current prototype implementation), and the index $n = 1, \ldots, N_m$ identifies the message sent by the mth sensor (N_m is the number of elements in the set of possible messages $\{x_{m,n}\}_{n=1}^{N_m}$ for the mth sensor; notice that each sensor can have a different set/alphabet size N_m). The modified version of (1) shown in (2) can be used to denote the amount of information carried by any message generated by any of the sensors in the IoT system.

The classical definition of information given in (2) is restricted to discrete sources (i.e., sources that can generate messages from a discrete set/alphabet). In sensors with discrete outputs (e.g., binary on/off sensors such as the touch and motion sensors) the application of this model is straightforward (e.g., $N_m = 2$ for binary sensors). In the case of sensors with continuous outputs such as temperature and pressure, the model of (2) cannot be applied directly. However it can be applied by discretising the continuous alphabet. To this end, the continuous range of output values of a sensor (e.g., the range of measurable temperatures or pressures) can be divided into a discrete set of N_m bins with a bin width w_m such that:

$$N_m = \left\lceil \frac{\max_n\{x_{m,n}\} - \min_n\{x_{m,n}\}}{w_m} \right\rceil \tag{3}$$

Therefore, a continuous output value x_m is classified into the nth bin when $x_m \in [\min_n\{x_{m,n}\} + (n-1)w_m, \min_n\{x_{m,n}\} + nw_m]$. The discretisation indicated in (3) enables the application of the model of (2) to sensors with continuous outputs.

The method here proposed defines a *surprisal* parameter in order to characterise the amount of information carried by a particular set of output values from the set of sensors available in an IoT system. Such characterisation is defined in terms of the probability of occurrence of such combination of outputs (the lower the probability of occurrence of a set of outputs, the higher the amount of information/surprisal will be).

Consider initially the outputs/messages x_{m_1,n_1} and x_{m_2,n_2} of two sensors with indices m_1 and m_2, respectively. The amount of information carried by the event of these two sensors jointly generating this pair of outputs/messages can be quantified by defining a *joint surprisal* parameter as:

$$\mathcal{S}_{\text{joint}}(m_1, m_2) = \log_2\left(\frac{1}{P(x_{m_1,n_1}, x_{m_2,n_2})}\right) \qquad (4)$$

where $P(x_{m_1,n_1}, x_{m_2,n_2})$ is the joint probability of simultaneous occurrence of output x_{m_1,n_1} in sensor m_1 and x_{m_2,n_2} in sensor m_2. While (2) quantifies the amount of information carried by any individual message generated by any of the sensors in the IoT system, (4) quantifies the amount of information carried by a pair of messages simultaneously generated by two individual sensors. Therefore (4) can be seen as an extension of (2) to the case of any two outputs/messages generated in the IoT system. This definition can be extended to an arbitrary number of sensors, leading to the more general definition:

$$\mathcal{S}_{\text{joint}}(\mathcal{M}) = \log_2\left(\frac{1}{P(x_{m_1,n_1}, \ldots, x_{m_{|\mathcal{M}|},n_{|\mathcal{M}|}})}\right) \qquad (5)$$

where $\mathcal{M}$ is a predefined set of sensor indices and $|\mathcal{M}|$ is the cardinality (number of elements) of the set. This *surprisal* parameter can be used to quantify how likely/expected or unlikely/unexpected the current state of the IoT system is in terms of the probability of occurrence of a particular set of outputs/messages from the set (or a subset) of IoT sensors. When a set of outputs/messages with a low joint probability is observed, this can be considered as a *surprising* event that very likely indicates the occurrence of a relevant event in the environment where the IoT sensors operate. This can be exploited to detect and identify situations that may require some type of special consideration and/or intervention. This novel smart data processing approach enables the extraction of relevant information from large volumes of data in a simple and efficient way.

It is worth noting that the definition given in (5) can lead to practical implementation problems even for a relatively moderate number of sensors. In a practical implementation, the joint probability for M sensors can be characterised by an M-dimensional histogram whose elements can be computed based on the history of past outputs. If each sensor can output N_m different values, the histogram would need to store $\prod_{m=1}^{M} N_m$ elements. For $N_m = N \ \forall m$ the size of the M-dimensional histogram would be of N^M elements (even for a small number of sensors and messages per sensor, the number of elements that would need to be processed, updated and stored would not only be huge but would also increase exponentially with the number of sensors, M, which obviously is not scalable and therefore difficult to implement in some practical scenarios). This problem can be overcome by introducing the assumption of independent sensors. If the sensors of the IoT system generate outputs/messages that are independent of each other, the joint probability in (5) can then be simplified to the product of the individual probabilities of occurrence of each message, which would lead to an *independent surprisal* parameter defined as:

$$\mathcal{S}_{\text{indep}}(\mathcal{M}) = \log_2\left(\frac{1}{\prod_{m\in\mathcal{M}} P(x_{m,n})}\right) \qquad (6)$$

When set $\mathcal{M}$ is selected as the set of all sensors available in the system (i.e., $\mathcal{M} = \{1, \ldots, M\}$), (6) can be expressed as:

$$\mathcal{S}_{\text{indep}}(\mathcal{M}) = \log_2\left(\frac{1}{\prod_{m=1}^{M} P(x_{m,n})}\right) \qquad (7)$$

The implementation of this alternative definition only requires an individual one-dimensional histogram for each sensor, and the total number of elements to be estimated and stored reduces to $\sum_{m=1}^{M} N_m$ (only MN elements in the example above, which is scalable and therefore easily implementable in practice).

The selection of one version of the parameter or the other is mainly determined by the number of sensors available in the system and how they provide data reports. For example, if the system operates in synchronous full data reporting mode where all sensors provide a data report in every data polling event (see Section 3.2 for details), the set $\mathcal{M}$ necessarily includes all sensors available in the system (i.e., $\mathcal{M} = \{1, \ldots, M\}$). In such a case, using (5) may be computationally infeasible and (7) would be a more convenient option. In synchronous partial data reporting mode and asynchronous data reporting mode, where not all sensors provide a data report simultaneously and only a subset of sensor outputs is relevant, (5) may be used if the number of considered sensors is low, otherwise (6) would be more convenient. If a small number of sensors is considered, then (5) may be used regardless of the data reporting mode.

Note that the probabilities in the denominators of (4)–(7) characterise the joint probability that the sensors in $\mathcal{M}$ output a particular set of values simultaneously. Those combinations of outputs that are more infrequent will have a lower joint probability and therefore the surprisal parameter $\mathcal{S}(\mathcal{M})$ will take a higher value. The variations in the value of $\mathcal{S}(\mathcal{M})$ can be used to automatically identify and react to relevant events.

The ability of the proposed data processing method to detect relevant events was assessed with an experiment conducted in an office environment where a person was working on a computer. This represents a static scenario with sporadic movements of the worker around the office. The window was equipped with blinds to control the light intensity in the room.

Fig. 6 shows a screenshot of a visualisation screen implemented in the central processing unit to show the real-time computation of the sensor data histograms and both versions of the surprisal parameter. The small subfigures on the top half of the screen show the (individual one-dimensional) histograms of the data reported by the sensors (4 sensor nodes, 10 parameters per sensor). Every histogram is updated every time a new data report (value) is received (the last value is shown as a vertical red line in each histogram). The histograms of sensors with binary on/off outputs (three first columns of figures on the left-hand side) are composed of $N_m = 2$ bins (note that the touch sensors

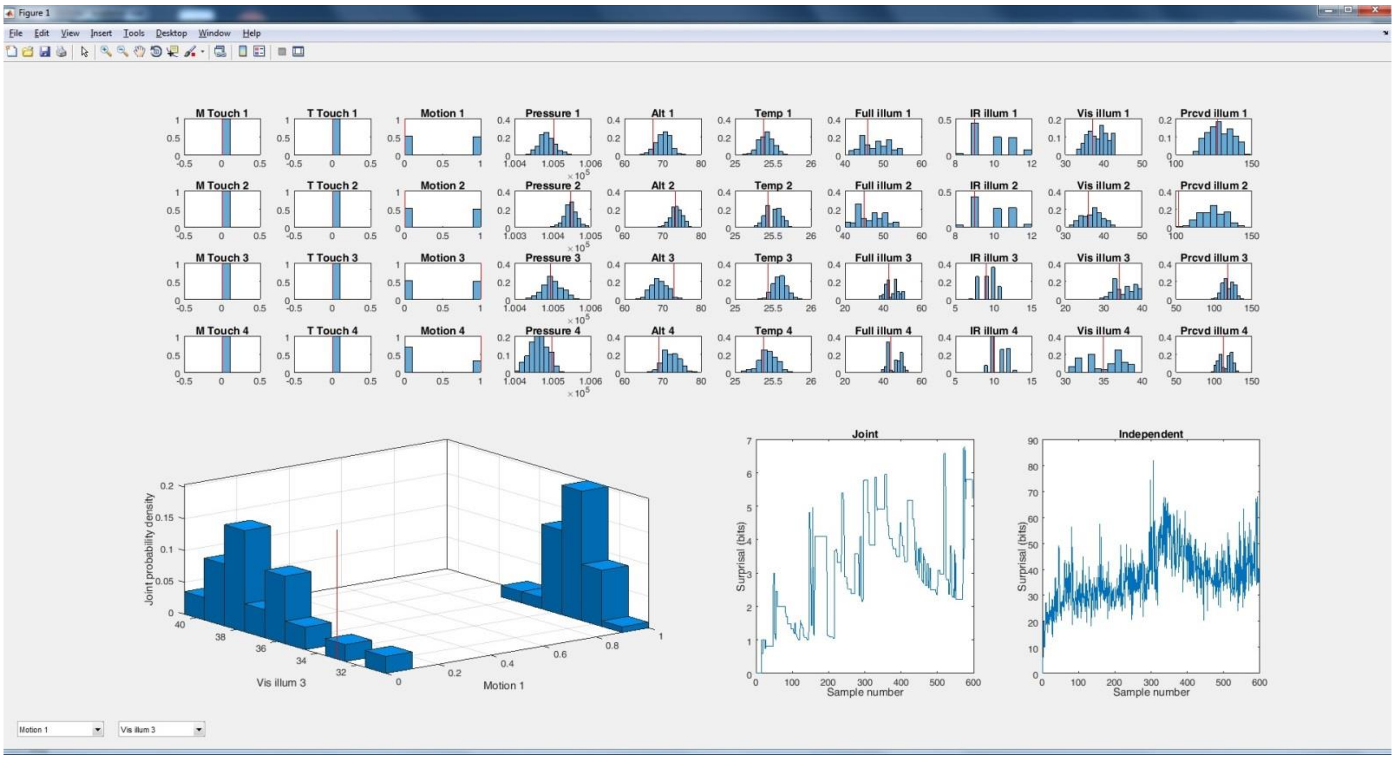


Figure 6: Screenshot showing the real-time computation of sensor data histograms and the surprisal parameter $\mathcal{S}(\mathcal{M})$.

were not operated in this example) and the rest of sensors (with continuous outputs all of them) are discretised to $N_m = 10$ bins. All histograms are normalised so that the value of every bin represents the probability of receiving a message whose value falls within that bin. These probabilities are used to compute the independent surprisal parameter $\mathcal{S}_{\text{indep}}(\mathcal{M})$ based on (6), where the set $\mathcal{M}$ is defined as the set of all sensors providing a valid data report in the last data polling event (the IoT system operated in synchronous partial data reporting mode with about 60% of the sensors providing a data report in every data polling event). The time evolution of $\mathcal{S}_{\text{indep}}(\mathcal{M})$ is shown in Fig. 6 in the figure labelled as *Independent*. Moreover, the joint 2-dimensional histogram/probability distribution of the motion sensor of node A and the visible light sensor of node C is computed (shown on the bottom-left corner of Fig. 6) and used to compute the joint surprisal parameter $\mathcal{S}_{\text{joint}}(\mathcal{M})$ based on (4)-(5). The time evolution of $\mathcal{S}_{\text{joint}}(\mathcal{M})$ is shown in Fig. 6 in the figure labelled as *Joint*. Fig. 7 shows a detailed view of how the surprisal parameters evolve over time in this experiment (the top figure also shows in red a moving-averaged version).

The experiment was divided into four phases, which are also indicated in Fig. 7. In the first phase of the experiment (sample numbers 0-295) the blinds were closed and the worker was sitting on the chair working on a computer (with sporadic movements to pick up papers and other objects on the desk). It can be observed that there is a transition at the beginning of the experiment until sample number ≈75 where the value of the surprisal parameters increases gradually until they reach a rather stable value. This transition is due to an initial learning phase where the system starts collecting samples from a completely unknown environment. In the rest of the first phase, the independent surprisal parameter varies within a bounded interval, with sporadic peaks that occur when some movement is detected. While the independent surprisal parameter shows some variation, the measurements received from the sensors fit reasonably well with the distribution of past values and therefore the independent surprisal parameter remains approximately constant, with some peaks when movement occurs (this trend is better appreciated in the moving-averaged sequence). On the other hand, the joint surprisal parameter shows more drastic variations every time some movement is detected. The reason is that this version of the surprisal parameter depends on only two parameters/sensors and is therefore more sensitive to the variation of one of these parameters (moreover, a joint histogram needs more observations to be estimated accurately and the joint distribution is not well modelled at this point). The worker remains static most of the time during the first phase of the experiment, and when some movement is detected this represents an *unusual* event since it does not fill well with the distribution of past values. As a result, the surprisal parameter increases suddenly as soon as the movement occurs.

In the second phase of the experiment (sample numbers 295-570) the worker was moving constantly around the room (the blinds remained closed as in phase one). A similar pattern can be identified in both surprisal parameters. The value of the surprisal increases suddenly as soon as the worker starts moving at the beginning of phase two. This sudden increase occurs because the new measurements reported by the sensors do not fit well with the distribution of past measurements (when the worker was static) and as a result they are initially seen as un-

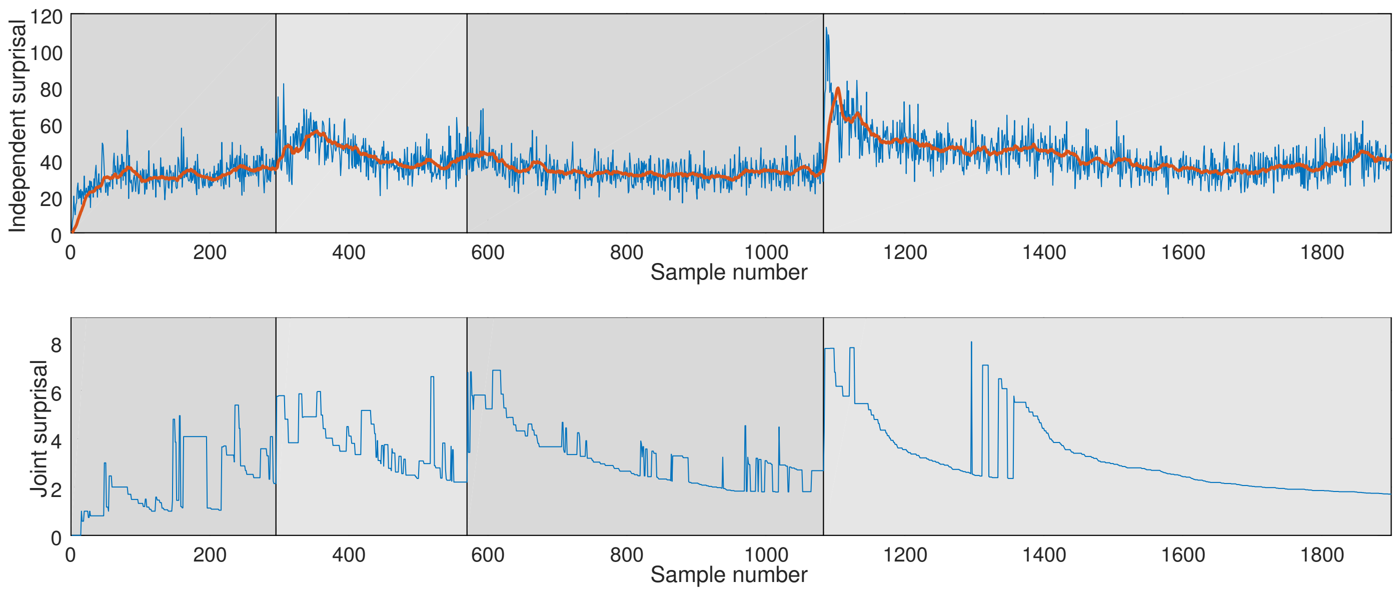


Figure 7: Time evolution of surprisal parameters.

likely events (because their corresponding probabilities in the histogram are low when they happen for the first time). When more similar measurements are reported by the sensors, the corresponding histograms are updated and their relative probabilities of occurrence increase, thus making them less unlikely and as a result the surprisal starts decreasing gradually. After certain time, this new situation becomes as frequent as the situation in phase one and the surprisal parameter decreases until it reaches a value similar/comparable to that of phase one.

In the third phase of the experiment (sample numbers 570-1083) the worker returns to the chair (with the blinds still closed), which is the same scenario as in phase one. One might expect that this event would not be detected by the system since it would in principle be a situation the system is already familiar with. However, Fig. 7 clearly shows this is not the case since the system immediately detects the event via the increase of the surprisal parameter. In the case of the independent surprisal parameter, the increase is not very significant (probably due to the existence of similar sets of measurements in the histograms) but in any case it is notable enough to be detectable. In the case of the joint surprisal parameter it is interesting to note that the increase is actually very significant, which can be explained by the fact that the new sets of measurements received while the worker was moving resulted in a reshaping of the original histogram so that when the worker comes back to the original situation of phase one the reported measurements are not as likely as they used to be in the first phase of the experiment. In any case, after some time, the surprisal parameter decreases as the system becomes more familiar with the new situation.

Finally, in the last phase of the experiment (sample numbers 1083-1900) the worker opens the blinds and returns again to a static position on the chair. As soon as the blinds open, the surprisal parameters increase suddenly and reach their highest peaks. This significant increase can be explained by the fact that this is the first time the system is exposed to a high light intensity and, even if the rest of parameters remain unaltered, the new values of light intensity reported by the sensors are so far from the distribution of past values that they are seen as very unexpected values and as a result the overall values of the surprisal parameters are affected. As in previous phases, as the system receives more measurements from the new scenario, the situation becomes less unlikely and the surprisal parameter decreases gradually (the peaks observed in sample numbers 1300-1400 were caused by the movement of some clouds, which led to some alternated periods of light/darkness).

From the obtained results it can be concluded that the proposed data processing method is capable to detect unexpected events in the physical world and react to them. The system keeps an updated version of the histogram of the values reported individually by each sensor and this information is exploited to determine how likely or unlikely a new set of data reports is. When an unexpected event occurs, the measurements reported by the sensors lead to a set of values with a low probability of occurrence and this helps the system identify such event and take actions automatically. Notice that the occurrence of a relevant event is essentially detected by means of a sudden increase in the value of the surprisal parameter. As observed in Fig. 7, the surprisal parameters show noticeable sudden increases at the beginning of every phase of the experiment, which indicates that the proposed approach can be used to detect the relevant events (i.e., changes of circumstances in the environment). However, there are some sudden increases (with different levels of importance) at other time instants as well, which might be due to some punctual events that are unlikely/surprising enough to generate these increases. This suggests that the proposed method could be extended with features such as processing of surprisal values to determine what events are more relevant and should trigger a reaction from the system (for example, based on the peak values, the duration of the peaks or the rate at which the peaks decrease), which is proposed as future work.

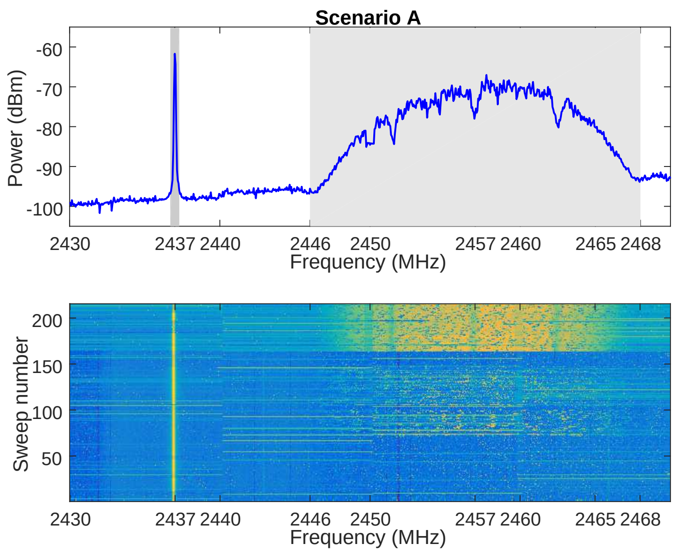


Figure 8: Coexistence Scenario A (without inter-system interference).

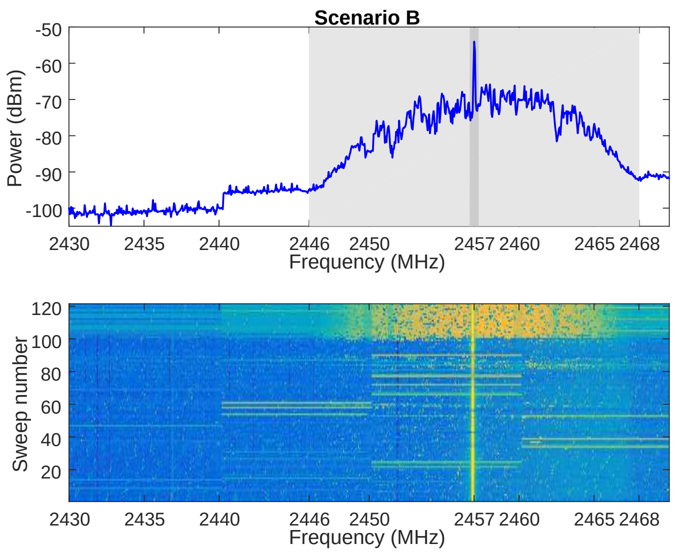


Figure 9: Coexistence Scenario B (with inter-system interference).

The proposed method, while based on rather simple principles, paves the way for the development of more sophisticated methods exploiting the full potentials of advanced fields such as data mining, artificial intelligence, or machine learning, among others. While the example shown in this section has not made use of the camera sensors, they can also be exploited in more sophisticated methods. For example the images captured by the sensors can be processed for *ego-motion* detection (i.e., the motion and rotation of the camera axes), which can be useful in applications where the sensors are worn or mounted on something that moves, the detection of people in front of the camera (in this case the camera would behave as an on/off sensor for the detection of human beings or objects), the identification of the type of environment (indoor, outdoor, urban, rural, etc.), or the extraction of descriptors that encode properties of the images as a whole. The range of possibilities is virtually unlimited and the developed prototype provides a suitable and flexible tool for a realistic evaluation of such innovative solutions in IoT systems.

The illustrative example discussed in this section not only demonstrates that it is possible to design smart data processing methods to extract meaningful information from large volumes of data and effectively detect relevant events in the context of IoT systems, but also shows the ability of the developed prototype to implement and evaluate such type of methods under configurable and controllable realistic scenarios.

6.2. Radio Coexistence

In the example presented in Section 6.1 the IoT subsystem operated without interference from other systems (i.e., the coexisting radio subsystem was deactivated). This section presents an example where the IoT subsystem coexists with the coexisting radio subsystem (and other radio systems) in the same frequency band. The spectrum monitoring subsystem is used to monitor and record the spectral activity of both systems in the selected band of operation (the 2.4 GHz ISM band).

Two different coexistence scenarios are considered in this experiment, which are illustrated in Figs. 8 and 9 (these figures were obtained with the spectrum monitoring subsystem). In both scenarios the IoT subsystem operates in channel 10 of the 2.4 GHz ISM band as defined by the IEEE 802.11 standard (2446-2468 MHz) and its operation can be divided into three phases. In the first phase (approximately the first third of sweeps shown in Figs. 8 and 9) the IoT network is established but the IoT subsystem remains inactive. In this phase there is no communication between the IoT nodes and the central processing unit, and therefore there is no spectral activity, except for the beacon signals that the central processing unit transmits periodically (some of which are captured by the spectrum monitoring subsystem). In the second phase (approximately the second third of sweeps) the IoT subsystem is active and operates in synchronous partial data reporting mode, with around 60% of the sensors sending data reports in every data polling event, which results in some intermittent usage of channel 10. In the third and final phase of the experiment (approximately the last third of sweeps) one of the IoT nodes sends video with a resolution of 1920×1080 pixels at 30 frames per second in real-time, which requires a high data rate, thus forcing an intensive (constant) usage of channel 10 as appreciated in Figs. 8 and 9.

The coexisting radio subsystem is configured to operate at two different radio frequencies. In scenario A the frequency of operation is 2437 MHz (center frequency of channel 6), which in this experiment is an idle channel (no other 802.11 networks were detected in this channel), while in scenario B the coexisting radio subsystem operates at 2457 MHz (i.e., the center frequency of channel 10). Therefore, in scenario A there is no interference between both radio systems, while in scenario B the interference is maximum. The coexisting radio subsystem transmits an image file in a cyclic loop, thus resembling the transmission of a video, with the exception that the transmitted image is always the same, which allows a more accurate evaluation of the impact of interference on the system performance.

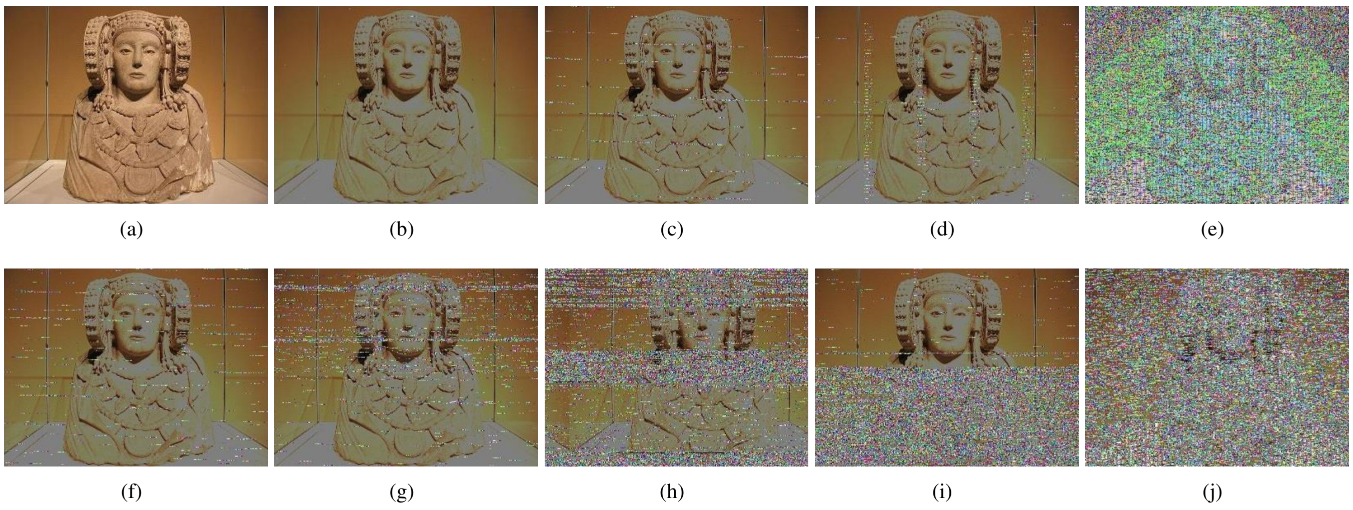


Figure 10: Image transmitted and received by the coexisting radio subsystem under various operation conditions.

The coexisting radio subsystem implements the physical layer features required for the successful transmission of data based on a QPSK modulation (as detailed in Section 4.2) but does not implement any interference prevention or interference management techniques thus making it very vulnerable to interference, as opposed to the IoT subsystem, which is based on IEEE 802.11 links and therefore counts with sophisticated methods to avoid and react to packet collisions.

This setup allows a detailed evaluation of the impact that the interference from an IoT system could have on other systems operating in the same frequency band, not only at the physical layer (e.g., bit, symbol or frame rates) but also at the application layer (e.g., visual quality of the received image).

During the experiment, the IoT subsystem did not appear to be affected by interference, at least to a significant extent. The data reported by all sensors were correct at all times. Only during some periods of interference the reception of data reports from the sensors experienced a noticeable delay (probably because of packet collisions), and only in one occasion the connectivity was lost during a moment of high interference but was recovered some time later. On the other hand, the impact of interference on the coexisting radio subsystem, which does not implement mechanisms as sophisticated as those included in the 802.11 standard, was much more noticeable.

Fig. 10 illustrates the impact of interference on the coexisting radio subsystem. Fig. 10(a) shows the transmitted image and the rest of figures show the image received under different operation conditions. Fig. 10(b) shows the image received in Scenario A when the overall activity in the 2.4 GHz ISM band was extremely low and there was virtually no interference from any other systems. Although the image does not show any noticeable artefacts, it is slightly darker than the original image, which could be explained by the transmission-reception process (in particular the up/down-sampling along with the use of SRRC filters). Figs. 10(c) and 10(d) show two images received in Scenario A with the IoT subsystem deactivated (first phase). Both figures show some lines with different numbers of consecutive incorrect pixels corresponding to frames received in error as a result of interference. Recall that pixels are transmitted from left-to-right and top-to-bottom, therefore interference leads to horizontal lines of consecutive erroneous pixels and the length of the line is proportional to the duration of the interference. Since the IoT subsystem was operating at a different frequency and with no ongoing communications, this interference was caused by other systems. While the interference pattern observed in Fig. 10(c) is rather irregular, the pattern in Fig. 10(d) appears to indicate that the interference was caused by a frequency hopping system, since interference intervals are observed at periodic time instants (when the interfering system hops to a frequency that overlaps with that of the coexisting radio subsystem) and with very similar durations (proportional to the time interval that the frequency hopping system remains in the same frequency). Since the coexisting radio subsystem and the IoT subsystem operate in different channels in Scenario A, one may expect they could never interfere with each other in such scenario. However this is not necessarily true. As a matter of fact, the image shown in Fig. 10(e), which corresponds to the third phase of the experiment (IoT subsystem transmitting video), indicates that the coexisting radio subsystem was severely affected by out-of-band emissions, or possibly by an increased noise floor, resulting from the transmission power of the IoT subsystem (17$\pm$1.5 dBm at maximum power).

Figs. 10(f)-10(j) correspond to images received in Scenario B. Fig. 10(f) was received in the first phase of the experiment (while the IoT subsystem was still inactive) and the interference observed was probably caused by other radio systems (in fact, Fig. 10(f) shows a relatively similar interference pattern as Fig. 10(c)). Fig. 10(g) was received in the second phase of the experiment, where the spectral activity of the IoT subsystem is higher as a result of the data reporting procedure; such activity leads to an increased level of interference, which can be clearly appreciated in Fig. 10(g). Fig. 10(h) shows the image received during an intermittent transmission of video in the IoT subsystem while Fig. 10(i) shows the image received when a continuous video transmission starts in the middle of the image reception; in both cases, the periods of interference can be clearly identified because the corresponding regions of the images are severely damaged. Moreover, in Fig. 10(h) the transmission of the image started during a period of interference, which affected the detection of the *zero frames* sent by the transmitter to help the receiver identify the beginning of an image. As a result, the beginning of the image (and consequently the whole image) was right-shifted. Finally, Fig. 10(j) shows the image received during a period of continuous video transmission. In this case the shape of the statue can be vaguely inferred, which indicates that the receiver was able to detect the *zero frames* (i.e., beginning of the image) but the rest of frames were severely corrupted by interference.

This illustrative example shows how the coexisting radio subsystem and the spectrum monitoring subsystems extend significantly the range of experiments that can be conducted with the developed prototype, introducing a spectrum-sharing dimension that enables the study of interactions between IoT networks and other communication systems and the analysis of the resulting impact not only at lower but also at higher layers of the system. This feature is particularly appealing in the development of reliable mechanisms to enable interference-free coexistence among radio communication systems in IoT systems.

7. Discussion

This section complements the description provided in the previous sections with a more general discussion of the development process of the prototype along with features and capabilities, problems and tradeoffs faced as well as lessons learned that may be helpful to other researchers in the development of similar tools. Some possible lines for future research are pointed out as well. Fig. 11 shows a generic prototype development flowchart, which is discussed in more detail below.

7.1. Objectives and Scenarios

The development starts with the definition of the objectives and potential scenarios of interest. This prototype is intended to enable a realistic evaluation and joint optimisation of potential relations and interactions of multidisciplinary aspects of IoT systems, in particular from the hardware (mainly in the radio front-end part), communications and data processing domains. Several potential IoT scenarios of interest have already been identified such as, for example, smart cities, intelligent buildings, transport, healthcare, energy/utilities, automotive, or asset tracking (see Ofcom (2015) for details). Nevertheless, this implementation is based on an *scenario-agnostic* design, which is a suitable approach for general proof-of-concept experimentation as discussed in Section 3. Moreover, a transition from a *scenario-agnostic* implementation to a *scenario-specific* one is usually more viable than a transition in the opposite direction.

7.2. Requirements

The developed prototype meets the common requirements typically set for IoT experimentation platforms (Gluhak et al., 2011), including realism of the experimentation environment (field measurements are performed and exchanged using real communication protocols), heterogeneity of IoT devices (currently 4 nodes with a total of 40 sensors of 10 different types), adequate scale (the current implementation is based on a small scale suitable for proof-of-concept in a laboratory setup, which can however be extended to include a significantly higher number of nodes as discussed below), mobility support (the nodes are connected with wireless links, can be operated with batteries if needed and rely on TCP/IP communications, which allows not only local mobility but also mobility at a larger scale), repeatability and replayability of experiments, and real end-user involvement in the experimentation cycle (see Section 6).

7.3. Scope and Architecture

The scope of a prototype plays a key role in the architecture required for its realisation. The overall architecture of the developed prototype is composed of three subsystems. The subsystems are independent and can operate without each other, which is a desirable feature in studies where the scope is in one particular domain (i.e., either an IoT network, a potential coexisting radio system, or a spectrum monitoring system). However the prototype has been designed bearing in mind scenarios where all three subsystem operate concurrently. This mode of operation is desirable in studies with a multidomain scope where the interest is in the potential interactions among aspects of different domains but without losing the level of detail provided by a detailed implementation of each subsystem/domain.

The prototype has been envisaged as the coordinated aggregation of three independent subsystems. This independence facilitates the development of each subsystem, which can be based on a tailored internal architecture that can be developed independently (except maybe for some readjustments required in the final integration of all subsystems). The IoT subsystem, which is the core of the prototype, relies on a two-tier architecture where the IoT device tier is directly attached to the server side (i.e., the central processing unit) by means of a point-to-multipoint topology. As discussed in Section 3.2, it is possible to employ packet delay/loss models (implemented in the central processing unit) to selectively delay or discard data packets from IoT nodes, thus emulating in software a three-tier hardware architecture where an intermediate gateway device tier is present. Small-scale prototypes as the one here presented are typically based on two-tier architectures while larger scale ones rely on the three-tier architecture model (Gluhak et al., 2011). The developed prototype, however, provides a flexible design that can emulate in software a three-tier architecture over a two-tier physical implementation.

The IoT device tier is currently based on a node-oriented homogeneous composition, where all IoT nodes, which are identical, are the entities that interact directly with the central processing unit by providing reports with data from their attached sensors. This approach simplifies the overall system operation since each IoT node manages locally its attached sensors, gathers the information and communicates with the central processing unit to provide data reports. Moreover, the extension of the prototype with more IoT nodes is straightforward as long as all

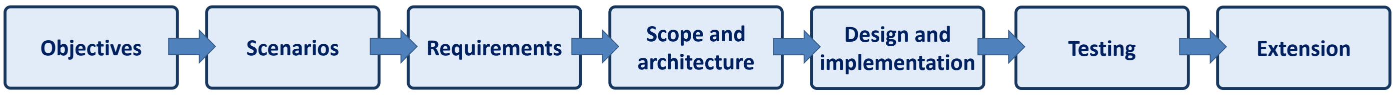


Figure 11: Prototype development process.

them are identical. The addition of a more heterogeneous set of IoT nodes (e.g., with different types of sensors or a different combination thereof) would pose some issues in this architecture, which highlights the importance of a standard reference architecture for IoT nodes in order to enable the abstraction of the particular details of the IoT nodes and their attached sensors and hence allow a flexible and scalable extension. For example, IoT nodes could provide an abstraction layer so that the central processing unit does not see a set of IoT nodes but the whole pool of available IoT sensors (regardless of which IoT node they are attached to). This shift from node-oriented to sensor-oriented architecture would enable a more flexible support of heterogeneous nodes/sensors but would require more complex procedures in the central processing unit in order to manage each individual sensor, which could lead to scalability issues. The study of this problem is proposed as future work.

7.4. Design and Implementation

A prototype is an initial implementation of a new system that provides a model on which the final implementation can be based. While a prototype is not expected to be a full final version of the system, it should certainly allow a designer to assess and validate the operation and performance of the new system under realistic conditions. This raises the important questions of how much of a real network should be implemented, how much should be emulated, and how much should be simplified or removed. Sections 3, 4 and 5 have provided a detailed description of the selected features implemented in the prototype. This is the phase of the development process where more important decisions had to be made and more problems were faced. A brief discussion of the most relevant lessons learned in this step are discussed below.

One important challenge is the design and implementation of IoT sensor nodes. A wide variety of commercial sensor nodes are available, typically based on a predefined set of sensors connected to a microcontroller that performs basic/no processing on the sensor data and is equipped with some basic communication capabilities. These designs are usually optimised for the particular application scenario they were envisaged for (in terms of performance, energy consumption, etc.), but offer very limited or null flexibility to changes or upgrades, which are typically performed by replacing the whole node. The IoT nodes implemented in the developed prototype are based on a more flexible and sustainable approach where each IoT node uses a generic low-cost minicomputer with a reasonable computational performance (e.g., Raspberry Pi or similar), running a mature software environment for which a wide variety of applications (for data processing, wireless and wired communications, etc.) are available, and a generic connection interface (GPIO) that can be reconfigured in software to support a broad range of sensor connectivity technologies (e.g., SPI, I2C, serial/UART). While this choice is suitable in a prototype for experimentation that may require frequent changes, it may also be a suitable approach for commercial application and deployment scenarios given its enormous flexibility. However, this improved flexibility is obtained at the expense of a lower performance that may lead to issues when handling simultaneously heterogeneous sensors with different timing requirements.

Another important aspect is the selection of the band of operation for IoT systems. ISM bands are the most popular due to the possibility of operating without a spectrum license and the wide range of connectivity technologies available, in particular in the 2.4 GHz band (e.g., WiFi, Bluetooth, ZigBee). However this band is becoming increasingly crowded, which makes difficult to perform experiments in a controlled environment in the case of a prototype and may lead to scalability problems in the case of real deployments. Other ISM bands at lower frequencies (e.g., 315/433 MHz or 868/900 MHz) may be less crowded but may differ among countries, offer a substantially lower amount of free capacity and have certain additional transmission constraints; on the other hand, ISM bands at higher frequencies (e.g., 5 GHz) may be less crowded but radio propagation conditions are less favourable and equipment/components are more expensive, which is not desirable in large-scale deployments. The 2G/3G/4G cellular bands have also been considered due to the wide deployment and coverage of mobile communication systems but suffer similar overcrowding problems. The selected band of operation also determines the potential interference patterns from other coexisting radio systems and therefore the design of the underlying communication protocols, which is an important aspect to consider and for which the developed prototype constitutes an excellent tool. Spectrum regulators have started considering alternative licence-exempt bands for IoT applications (see Ofcom (2015) for details); this will require dedicated studies in such bands and the developed prototype is a suitable tool to conduct this type of studies.

The robustness of the system in the presence of errors and exceptions is another important feature in future IoT systems, which are expected to integrate a large number of low-cost (and therefore with limited reliability) nodes. The approach adopted in the developed prototype is to detect and handle different types of errors from the central processing unit. Novel data processing methods such as the one proposed in Section 6.1 should be able not only to detect relevant genuine events but also different types of errors (e.g., malfunctioning sensors or misbehaving nodes). This can be done for example in combination with reputation or anomaly detection methods, which is a topic proposed for further future research.

Scalability is essential in any IoT prototype. The presented prototype has been developed to enable the study of scalability aspects in the three considered domains of interest, namely hardware design (e.g., energy-efficient devices and antennas, and circuits for energy harvesting), communications (e.g., protocols with low complexity that can maintain reliable links with inaccurate clocks, deal with frequent collisions and congestion situations arising from a high number of nodes, manage completely different network traffic patterns in an efficient way, and access/share spectrum efficiently), and data processing (e.g., innovative smart methods based on scalable solutions as the one proposed in Section 6.1). The developed prototype is an excellent tool for the study of these issues from both individual and multidisciplinary points of view.

7.5. Testing

A bottom-up testing methodology was proven to be the most suitable approach in the development of the presented prototype, starting from individual sensors, sensor-node connectivity, node-server communication, data processing in the central processing unit, coexisting radio subsystem, and finally spectrum monitoring subsystem. The testing methodology should include not only functional tests but also stress tests in all the above mentioned test stages in order to ensure an adequate operation under normal and extreme/peak operation conditions.

7.6. Extension

While the developed prototype has been initially deployed as a standalone platform, future extensions can be envisaged. The extent to which a prototype can be extended depends on its flexibility and scalability, two aspects that have been taken into account in the development process as discussed above. Possible future extensions to the developed prototype include for instance the addition of new local and remote nodes (based on communications over TCP/IP) as well as providing access to the prototype to the community at both individual and federated levels. Federation with other research facilities is necessary to achieve scale and add capabilities for experimentation that are not locally available, which is a further challenge to be tackled.

8. Conclusions

Current trends indicate that future communication networks will interconnect billions of smart devices capable of automatically collecting and exchanging data, thus enabling the direct integration of the physical world into automated computer-based systems equipped with intelligence and smart features. This concept, referred to as the Internet of Things (IoT), poses important challenges requiring multidisciplinary solutions that take into account the potential mutual effects and interactions among the different dimensions of future IoT systems. In this context, this paper has presented a prototype developed in the context of the EPSRC/eFutures-funded project "Internet of Surprise: Self-Organising Data", which constitutes a suitable platform for an accurate and realistic evaluation of IoT solutions. The prototype enables the joint evaluation and optimisation of multidisciplinary aspects of IoT systems, including aspects related with hardware design, communications and data processing. This paper has provided a comprehensive description of the developed prototype, including design and implementation details that may be helpful to other researchers and engineers in the development of similar tools. Several illustrative examples showing the potentials and capabilities of the prototype have been presented as well. The developed prototype is a versatile tool that can be employed for proof-of-concept, validation and cross-layer optimisation of multidisciplinary solutions before bringing them to real IoT deployments.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under eFutures project ref. RES/0560/7436 (application no. eFuturesSP15/003).

References

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., Fourth Quarter 2015. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys and Tutorials 17 (4), 2347–2376.

Altman, E., Avrachenkov, K., Barakat, C., April 2005. A stochastic model of TCP/IP with stationary random losses. IEEE/ACM Transactions on Networking 13 (2), 356–369.

Atzori, L., Iera, A., Morabito, G., October 2010. The Internet of Things: A survey. Computer Networks 54 (15), 2787–2805.

Atzori, L., Iera, A., Morabito, G., January 2014. From 'smart objects' to 'social objects': The next evolutionary step of the Internet of Things. IEEE Communications Magazine 52 (1), 97–105.

Bi, S., Ho, C. K., Zhang, R., April 2015. Wireless powered communication: opportunities and challenges. IEEE Communications Magazine 53 (4), 117–125.

Evans, D., April 2011. The Internet of Things: How the next evolution of the internet is changing everything. White paper, CISCO.
URL `http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf`

Fleisch, E., January 2010. What is the Internet of Things? - An economic perspective. White paper, Auto-ID Labs.
URL `http://cocoa.ethz.ch/media/documents/2014/06/archive/AUTOIDLABS-WP-BIZAPP-53.pdf`

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., Razafindralambo, T., November 2011. A survey on facilities for experimental internet of things research. IEEE Communications Magazine 49 (11), 58–67.

Gorlatova, M., Sarik, J., Grebla, G., Cong, M., Kymissis, I., Zussman, G., August 2015. Movers and shakers: Kinetic energy harvesting for the Internet of Things. IEEE Journal on Selected Areas in Communications 33 (8), 1624–1639.

ISO/IEC, May 2014. Advanced Message Queuing Protocol (AMQP) v1.0. PRF 19464, ISO/IEC.

ISO/IEC, June 2016. Message Queuing Telemetry Transport (MQTT) v3.1.1. PRF 20922, ISO/IEC.

ITU-R, July 2015. Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 100 GHz. ITU-R P.1238-8, ITU-R.

Kamalinejad, P., Mahapatra, C., Sheng, Z., Mirabbasi, S., Leung, V. C. M., Guan, Y. L., June 2015. Wireless energy harvesting for the Internet of Things. IEEE Communications Magazine 53 (6), 102–108.

Lakshman, T. V., Madhow, U., June 1997. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking 5 (3), 336–350.

Mattern, F., Floerkemeier, C., 2010. From the Internet of Computers to the Internet of Things. In: Sachs, K., Petrov, I., Guerrero, P. (Eds.), From Active Data Management to Event-based Systems and More. Springer-Verlag, Berlin, Heidelberg, pp. 242–259.

Ofcom, January 2015. Promoting investment and innovation in the internet of things. Statement, Ofcom.

Perera, C., Liu, C. H., Jayawardena, S., Chen, M., 2014. A survey on Internet of Things from industrial market perspective. IEEE Access 2, 1660–1679.

Piñuela, M., Mitcheson, P. D., Lucyszyn, S., July 2013. Ambient RF energy harvesting in urban and semi-urban environments. IEEE Transactions on Microwave Theory and Techniques 61 (7), 2715–2726.

Rice, M., 2008. Digital Communications: A Discrete-Time Approach, 1st Edition. Prentice Hall, New York.

Saint-Andre, P., March 2011a. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, IETF.

Saint-Andre, P., March 2011b. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 6121, IETF.

Saint-Andre, P., October 2014a. End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP). RFC 3923, IETF.

Saint-Andre, P., October 2014b. Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM). RFC 3922, IETF.

Saint-Andre, P., September 2015. Extensible Messaging and Presence Protocol (XMPP): Address Format. RFC 7622, IETF.

Shannon, C. E., July 1948a. A mathematical theory of communication. The Bell System Technical Journal 27 (3), 379–423.

Shannon, C. E., October 1948b. A mathematical theory of communication. The Bell System Technical Journal 27 (4), 623–656.

Shelby, Z., Hartke, K., Bormann, C., June 2014. The Constrained Application Protocol (CoAP). RFC 7252, IETF.

Stanford-Clark, A., Truong, H. L., November 2013. MQTT for sensor networks (MQTT-SN) Protocol specification. Tech. rep., IBM.

Vermesan, O., Friess, P., 2013. Internet of Things - Converging Technologies for Smart Environments and Integrated Ecosystems, 2nd Edition. River Publishers.

Vinoski, S., November 2006. Advanced Message Queuing Protocol. IEEE Internet Computing 10 (6), 87–89.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M., February 2014. Internet of things for smart cities. IEEE Internet of Things Journal 1 (1), 22–32.