IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON RELIABILITY IN SENSOR-CLOUD SYSTEMS AND APPLICATIONS (SCSA)

Received March 26, 2021, accepted April 3, 2021, date of publication April 9, 2021, date of current version April 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072168

Techniques to Improve Reliability in an IoT Architecture Framework for Intelligent Products

CIPRIAN M. COMAN[1,4], GIUSEPPE D'AMICO[2], ADRIAN V. COMAN[3], AND ADRIANA FLORESCU[4], (Senior Member, IEEE)

[1]Tesagon International SRL, 100029 Ploiesti, Romania
[2]Focus Innovazione srl, 72015 Fasano, Italy
[3]RID International Center SRL, 540089 Targu Mures, Romania
[4]Faculty of Electronics, Telecommunications and Information Technology, Politehnica University of Bucharest, 060042 Bucuresti, Romania

Corresponding author: Ciprian M. Coman (ciprian@tesagon.com)

This work was supported in part by MANUNET through the Executive Unit for Financing Higher Education, Research, Development and Innovation (UEFISCDI) under Grant COFUND-ERANET MANUNET III-POKET, and in part by the Apulia Region POR FESR-FSE 2014-2020 - Priority Axis 1 - Research, Technological Development, Innovation - Action 1.6 (Interventions for Strengthening the Regional and National Innovation System and Increasing Collaboration Between Companies and Research Structures and Their Enhancement) under Project MNET18/ICT-3415.

ABSTRACT Sensor-Cloud Systems (SCS) are designed to link sensor networks and cloud applications, in order to gather and process information. While this is a hot topic for both academia and industry, with a large number of implementations, not much effort was put towards analysing the reliability of these systems. This article presents an experimental implementation of a system in the field of intelligent products and explores reliability improving techniques in five main areas of the SCS: network communication performance, auto recovery, local backup, automated software testing and system security. They all play an important role in determining the level of reliability of the novel Internet of Things (IoT) architecture framework for intelligent products presented in the article. A new formula is proposed for assessing the reliability of a SCS based on metrics from each of the five areas. Metrics used to assess the system reliability are presented, along with comparisons between operating with improvement techniques and without them. The results show that the reliability of the implemented SCS is improved considerably by implementing a deliberate reliability policy using an original five level tiered approach.

INDEX TERMS Knowledge based systems, Internet of Things, reliability, sensor systems, sustainable development.

I. INTRODUCTION

In recent years, both academia and industry have pursued the topic of integrating sensors, networks, cloud and artificial intelligence into holistic systems, with goals such as automated decision making, knowledge generation or monitoring.

The concept of intelligent products was studied and developed since the beginnings of the internet, making use of Radio-frequency identification (RFID) [1] and handheld devices for communication [2]. Despite the continuously evolving technology, attempts have been made to define and classify intelligent products characterized by abilities such as identification, ambient sensing, communication, and computation. In a scientific survey, Meyer et all conclude the definition of smart products is diverse when focusing on different aspects of smart products and different lifecycle phases [3], but suggest a classification based on three orthogonal aspects: what is the level of intelligence of the product, where is the intelligence (or processing power) located, and whether the product is managed as a single entity or as an aggregation. At the same time, Meyer et all acknowledge the value created by using this technology across contexts, as in product life cycle management.

Sensors networks are used nowadays in various setups, wired or wireless, just monitoring or also actuating, standalone or connected into industry 4.0. Applications include composite material manufacturing [4], electric grid monitoring and control [5], [6], agriculture [7], smart homes [8], smart city [9] etc. There are also innovative approaches to sensor data manipulation and storage using blockchain technologies, thus providing decentralization, transparency and immutability [10], [11].

One paper published in 2013 is among the first to identify reliability as one of the important qualities of service

The associate editor coordinating the review of this manuscript and approving it for publication was Md Zakirul Alam Bhuiyan.

parameters of wireless sensor networks (WSN) [12]. Maktedar and Deshpande focus on the network aspects of the WSN reliability, and analyse the data packet transmission from the source node (sensor) to a sink node (processing unit). In order to evaluate reliability, the packet delivery ratio (PDR) and packet loss ratio (PLR) are measured. The packet size is shown to have an impact on reliability, smaller packet size yielding the best results. However, in this particular study, after a certain threshold, reliability increases slightly with packet size. The paper concludes with a hypothesis that network congestion plays a big role and reliability can be increased by controlling other parameters.

Faults caused by communication errors, unstable connectivity, sensor faults, etc. in WSN for structural health monitoring of infrastructure such as bridges or buildings, greatly affect the performance of the monitoring system [13]. Solutions are proposed to ensure a desired fault-tolerance, based on concepts such as backup sensors and decentralized computing. Simulations and a field experiment conducted on a building in Hong Kong show that reliability is an important factor in the success or failure of a sensors system. Also, the backup sensors were able to supplement the manually induced faults (e.g. main sensor removal).

Sensor-Cloud Systems and Applications (SCSA) are also used in large scale systems. Reference [14] present a novel approach for environmental monitoring and management that combines IoT, Cloud Computing, Geoinformatics, Remote Sensing (RS), Geographical Information System (GIS), Global Positioning System (GPS), and e-Science for environmental monitoring and management, with a case study on regional climate change and its ecological effects. The paper shows a correlation between precipitation, air temperature, leaf area index and crop production. The results showed that ever since 2014 such systems are valuable for the perception, transformation, processing, management, and sharing of multisource information in environmental monitoring and management.

Virtual representation and modelling play a significant role in consistently describing and evaluating SCSA. In 2014, [15] is one of the first to present a mathematical formulation of sensor cloud, including mapping an application to its physical resources and virtualization of the resources, which is very important for studying the behaviour of WSN-based applications in the sensor-cloud platform. The virtual model is used to assess performance metrics and cost effectiveness, showing that sensor-cloud architecture outperforms a traditional WSN, by increasing the sensor lifetime by 3.25% and decreasing the energy consumption by 36.68%. The dynamic virtual sensor provisioning scheme has to balance energy efficiency and quality of service (QoS) [16].

A review of industrial communication systems published in 2019 [17] assesses the characteristics of existing network technologies, such as configurations, typical traffic profiles, performance requirements and reliability. A complex standardization framework is introduced and performance is measured using three main indicators: delivery time, throughput and bandwidth. For networks based on a cycle, in which a master device periodically polls a set of slaves, two more indicators are used: minimum cycle time and jitter. Future perspectives are also discussed, suggesting that the technology will evolve considerably in the following years, particularly in the fields of time-sensitive networking, next-generation industrial wireless networks, industrial internet of things, 5G cellular networks and software defined networking.

Using a central controller in the cloud can lead to improved sensor availability and data reliability in remote monitoring. Using a function to check the data from all the sensors in an area provides three main advantages: fault detection, fault masking, energy management. However, a more intelligent pair-wise comparison algorithm is needed to make the proposed solution feasible [18], [19]. Cloud can also assist in stabilization of multi-agent sensor systems and it is possible to explicitly quantify the benefits of the cloud assistance in terms of stability performance as well as superior scalability performance with low access latency [20].

The security aspect of SCSA starts to be considered in various application contexts, ranging from industrial to home use. The evolution of industrial SCADA systems is presented as a cyber physical system (CPS) integration with the IoT utilizing cloud computing services, with their most prevalent applications in smart transportation, smart grids, smart medical and eHealthcare systems, and many more [21]. SCADA systems implementations present security challenges that can be addressed by employing existing best practices and recommendations for improving and maintaining security. Sajid *et al.* suggest network intrusion detection systems and keeping firewalls up to date, performing regular risk assessments and improving the clarity of security plans and their implementations, thus preventing the system from being accessed by unauthorized entities.

Security is also a concern for the world of intelligent devices for home automation. Patru *et al.* present solutions for connecting several devices into a single application while emphasizing hardware abstraction layers and Weave as a protocol for secure communication using OAuth 2.0 Authentication [8].

The cloud side of the SCSA is also a major contributing factor to the reliability of the entire system. An increasing number of CPS have been deployed in cloud platforms, and to accommodate numerous CPS applications, cloud datacentres often consist of a huge number of physical computation and storage nodes, and the number is still increasing [22]. Virtual Machine (VM) management and scheduling play an important role in enring a reliable service to the end user while optimizing energy and resource usage [23]. This kind of on-demand services were studied in the context of an unmanned aerial vehicle (UAV) system supporting big data acquisition. Modelling and simulations reveal the relationship between the acquisition rate of sensor data and the stability of the cloud-based UAV system [24].

Security of these distributed systems relies on authentication of any entity that inputs or accesses data on the cloud.

Authenticating the data exchange partner while preserving privacy raises a new problem for SCSA. Some user authentication protocols for WSNs have been proposed to address this issue [25]. A performance comparison of several algorithms is presented by Xiong Li *et al.* using an evaluation under the random oracle model and the NS-3 network simulator [26]. Another study focuses on privacy in the particular case of medical sensor networks, proposing a data splitting approach and storing pieces on several nodes [27]. In this case, even if one or a few of the nodes are compromised, the original data cannot be composed without the entirety of the pieces.

The community cloud represents a cloud system used by a large number of different applications, operated by a third-party cloud service provider. One important challenge of community clouds is resource allocation and task scheduling while ensuring application separation and data integrity. Most often this is done by using VMs to execute the submitted applications. Various algorithms exist to distribute applications to VMs taking into account the application owner, execution deadline and cost constraints [28].

Privacy and security requirements of IoT are different from any other types of communications, having to build trust between a large number of entities and systems. A security architecture framework was proposed [29], discussing best practices for six layers of the IoT technology stack.

Modern SCSA produce large quantities of data that falls into two main categories: production data and diagnosis data. Working with big heaps of data requires automated tools for data aggregation and data processing. In smart city applications it is often a challenging task to protect users' context privacy while guaranteeing accurate data analysis and prediction results after data fusion. One suggested approach for privacy-aware data fusion and prediction is based on the classic Locality-Sensitive Hashing (LSH) technique [9]. To facilitate reliability these applications also generate a lot of diagnosis data in the form of logs and events. Special Big Data database systems like Elasticsearch are needed to store and manage these logs, as well as query mechanisms and Machine Learning (ML) techniques to observe anomalies in the infrastructure and automatically alert the managers or administrators [30].

All the activities and trials using these concepts and technologies proved their value in numerous exciting applications. However, little attention has been paid to solving the genuine reliability concerns. In this paper we analyse various approaches to improve reliability, starting with theoretical considerations that provide the bases for understanding technology limitations and reliability issues, presented in Section II, followed by a new reliability score formula and a description of the experimental model implementation in Section III, test results as measured with the original reliability score in Section IV and conclusions on the original reliability policy presented in Section V.

II. THEORETICAL CONSIDERATIONS

A. NETWORK COMMUNICATION

Data transmission plays a crucial role in any sensor-cloud system application. The network is responsible for moving data from sensors to the cloud, sending commands to decentralized sensor hubs and communicating decisions to actuators.

The network communication subsystem can be seen as a stack of layers, each of them with specific objective. We briefly present here the layers and their characteristics that impact communication reliability. At the bottom of the stack is the physical layer, the media through which signals are transmitted as a raw stream of bits. There are two kinds of media: guided (metal wire and fibre optics) and wireless (terrestrial radio and satellites). Each medium has its own advantages and disadvantages given by natural limits (e.g. sound speed 300 m/s and light speed 300 Mm/s) or arbitrary influences (e.g. parasitic disturbances or noise).

No matter the medium, information is encoded by varying a physical property such as voltage or wave amplitude. This value can be represented as a function of time, $f(t)$, and analyzed mathematically. Using the Fourier series, we can encode digital data as an analogue signal and decode it at the receiver end, while maintaining good resilience to noise. Fourier transform error correction made Wi-Fi possible, and was patented by a group of Australian researchers [31].

In order to define the medium transmission capacity, we use the Claude Shannon's equation defining data rate of a noisy channel, based on Nyquist's equation for noiseless media (1). It takes into account the bandwidth (W) of a transmission channel, defined as the frequency range on which the signal does not suffer a weakening greater than a certain value (generally 3 dB, given that 3 decibels correspond to a weakening of the signal of 50 %).

$$C = W log_2 \left(1 + \frac{S}{N}\right) \tag{1}$$

The capacity (C) of a channel is the amount of information, in bits, that can be transmitted in a second on the channel having a band width W and a signal (S) to noise (N) ratio S/N.

Comparing the guided media, copper wire is widely used, it is the cheaper and widely known technology. However, fibre cables have higher bandwidth, lower attenuation and are less affected by corrosive chemicals. Fibre is also harder to tap by intruders.

Wireless transmission media are based on electromagnetic waves at various frequencies. In fact, they work similar to guided fibre optics, but using a lower frequency, in the radio waves spectrum, as shown in Fig.1. Free space optical (FSO) is also an option, but it suffers from atmospheric disturbances.

To compute the data rate in this case, we use (1) and the frequency band Δf calculated using (2), where c is the speed

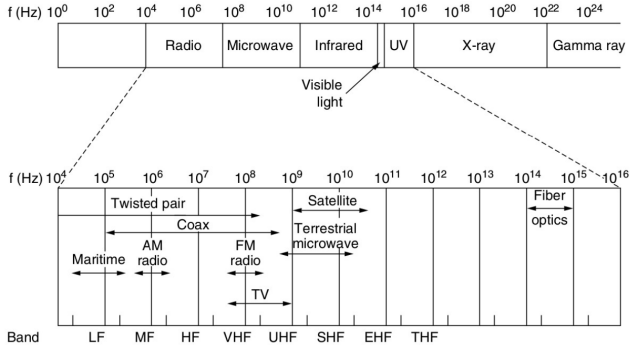


FIGURE 1. The electromagnetic spectrum used for communication [32].

of light and λ is the wavelength.

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2} \tag{2}$$

Communication satellites can provide internet access in remote locations. Traditionally, satellites were placed at high altitudes in order to reduce the required number of satellites to cover an area, but at the same time increasing latency and noise. However, we recently saw satellite constellations as low as 550 Km, rotating around the Earth every 91 minutes. There are also experiments with higher frequencies, in the K and V bands, despite their traditional issues with rain. As of March 2017, several companies filed with the US regulatory authorities plans to field constellations of V-band satellites in non-geosynchronous orbits to provide communications services [33].

The data layer is the second layer in the network stack. Its role is to translate between the raw bit stream from the physical layer and the packets used by the network layer, fulfilling the need to transmit data between two points linked by a direct point-to-point physical connection. Although at first the problem does not seem to be difficult, there are many challenges caused by errors in the bit stream from the physical layer, by the transmission delay and the finite data rate. There are several protocols designed to provide reliable and efficient communication. Error control is done using techniques such as flags to mark the start and end of a data frame, byte stuffing, bit stuffing, parity bits, checksums and means to request a frame retransmission when errors are detected at the recipient side. Flow control is required to prevent a fast sender from flooding a slow receiver.

A sliding windows mechanism is widely used to integrate flow control and error control. It ensures data frames are sent in a timely manner, with clear markers between frames, and allows time for error checking and acknowledgement receiving. While there are several protocols based on sliding frames, the internet uses the Point-to-Point Protocol (PPP) which is defined in Request for Comments (RFC) 1661 [34], RFC 1662 [35] and RFC 1663 [36]. As shown in Fig. 2, there is a general frame for transmitting data, and two particular frames used to bring lines up and negotiate protocol operating parameters.

FLAG	Address	Control	Protocol	Data	FCS	FLAG
0x7E	0xFF	0x03	2 bytes	(variable)	2 bytes	0x7E

Protocol values:

0x0021	General data
0xC021	Link Control Protocol (LCP)
0x8021	Network Control Protocol (NCP)

FIGURE 2. The PPP frame format.

The network layer is responsible for moving packets from the source to the destination, thus being the lowest level layer responsible for end-to-end communication. It uses software and protocols designed to route packets through several intermediate machines or routers. A dynamic map of the network is built using discovery and control messages, containing information on distance and congestion. A routing algorithm then decides the path each packet takes in order to optimize speed and network reliability.

Given the limited data rate of network communications, subnets can easily become congested, increasing delays and even discarding packets. Network design can most easily influence communication performance and reliability at this layer. Considering the size and number of packets sent from each sensor in the system, intermediate network equipment can be chosen to provide enough capacity and prevent congestion.

Another approach worth considering is the optimization of the links between sensors locations and the cloud server location, most likely located in different Autonomous Systems (AS). Various Internet Service Providers (ISPs) have different policies and partnerships at the internet routing protocol level, which can lead to unwanted detours and delays when routing packets between ASes.

The transport layer provides several services to the software applications, the most important of which is the host-to-host communication service, using the available lower layers, no matter which protocols of physical transmission media are used. It offers connection-oriented communication, reliability and flow control. There are two main transport protocols used in the internet: User Datagram Protocol (UDP), a single packet transmission protocol, and Transmission Control Protocol (TCP), a connection-oriented protocol.

TCP is based on sockets that take care of information segmentation in order to be sent in packets with limited data payload and make sure to reassemble the information stream at the receiver, despite packets possibly arriving in a different order then they were sent in. The protocol is also responsible for securing the communication channel, using a three-way handshake to establish and release the socket, as well as establish sequence numbers for data packets. The basic situation is show in Fig. 3, but the entire algorithm is more complex as the protocol has to account for lost or delayed packets that can resurface at an inopportune moment.

At transport level, there are several optimizations available to improve reliability and avoid congestion depending on the type of data stream that is sent. For example, a video sensor

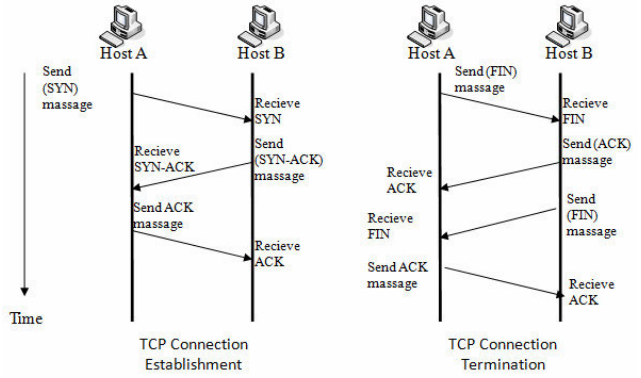


FIGURE 3. The base TCP session start and end [37].

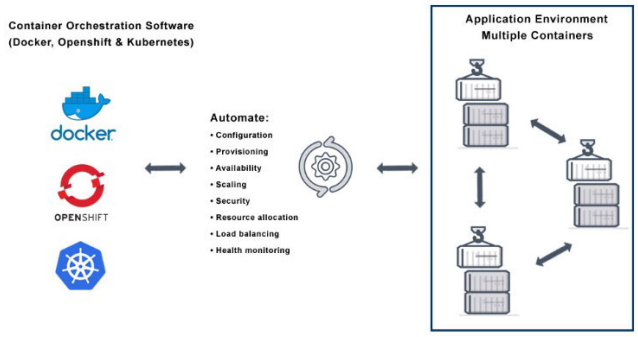


FIGURE 4. Cloud container orchestration architecture [38].

feed requires consistent bandwidth and can be optimized to follow the same route. On the other hand, atmospheric sensors send small packets at relatively large intervals of time and require a different approach.

Techniques to improve network performance also include: increasing processing power of the hosts, reducing the number of network packets by merging data, minimizing packet copying between software layers and avoiding timeouts.

B. AUTO RECOVERY

Automatic recovery refers to a system's ability to identify an error state and run a procedure to return to a nominal operating state. It plays and important role in increasing uptime and the proper operation of a system. For example, after the power was off, the device should be able to boot itself back up and resume normal operation.

Another effective and common technique is to power off and reboot the faulty module. In the case of sensor-cloud systems, a module can be the embedded controller reading the sensors and sending data to the cloud. When it is unable to acknowledge that the data was read and delivered to the cloud, it can be programmed to automatically perform a reboot procedure. Complete reboot is however an expensive procedure; a more efficient approach is to identify the malfunctioning process and restart only that process. It can be done using software techniques such as heartbeat and watchdog.

The automatic recovery also applies to the cloud part of the system. Using modern server software architectures such as orchestrated containers, presented in Fig. 4, one can ensure that there are enough running containers to serve the incoming requests. The orchestrator is responsible for creating and starting new containers, but also for monitoring their operation and shutting down misbehaving containers. In this way, the system reliability increases with no human interaction required.

C. LOCAL BACKUP

In a distributed system it can sometimes happen to temporarily loose connectivity between a node and the cloud. Even though real time data from the sensors is not be available in the cloud, it is still preferable to be able to access historical reports and graphs that include data for the disconnected period. To solve this request, local server controllers are fitted with storage capabilities where data is written while the network connection is down. When the connection is re-established, the controller should be able to process real time data from the servers, as well as synchronize backup data to the cloud.

When designing such a backup component, several metrics should be evaluated: storage capacity, read and write speed, network bandwidth. These metrics determine for how long the network can be offline before data starts being lost, as well as the minimum network requirements in order to process both real time data as well as backups over a required period of time.

Furthermore, the data has to be timestamped when it is acquired. This offers the ability to reorder the sensor data in chronological order, independent of the moment when it was transmitted to the cloud server. Precautions should be taken to ensure unique time values, taking into account summer and winter time, as well as sensors positioned in several time zones. The heterogenous nature of sensor networks raises new challenges for the time synchronization problem, however there are several solutions, using Network Time Protocol (NTP) [39], GPS [40] or post-facto messaging [41]. In order to deal with time zones and temporary changes, such as daylight-saving time, a time zone database (TZDB) was created by Arthur David Olson in the early 1980s to facilitate timekeeping on computer systems [42] and is currently hosted by IANA [43] and primarily maintained by Paul Eggert [44]. The W3C working group dedicated an article to guidelines and best practices for working with time in applications [45], including how to define and store time, represent zone offsets and comparing times.

D. AUTOMATED SOFTWARE TESTING

Today's technology evolves towards several digital systems working together in an ecosystem to solve a problem. All these interconnections create an extra level of complexity, as well as dependencies between systems. To ensure a reliable cooperation, the interactions between systems are defined in a strict way, and each software should take precautions to ensure that future developments do not impede the communication with other systems.

The interactions are defined as promises made to other software components regarding data format and communication protocols. To automatically enforce the documented agreement, the promises are translated into several tests. These tests are linked into test suits and are run automatically after any change is made to the code base. Running the test suits provides immediate feedback when a change affects one of the promises that were previously coded as a test.

Each test specifies the input data and the expected output for some usual situations as well as some corner cases. It is important to design the tests in such a way that all the code lines are evaluated; this is called code coverage. A test manager script is responsible for running the test and evaluating the output against the expected result.

There are several types of test methods commonly used in software development. They are designed to test different levels of system integration. Unit tests are used to verify the execution of a small part of code, usually a function or a class in Object Oriented Programming (OOP). Functional tests verify an entire use case, e.g. the data received from a sensor is routed to the correct processing function, the useful information payload is extracted and stored in the database. Acceptance tests in web applications are used to test the web interface used to access the cloud from a browser. They simulate a user clicking on buttons and scrolling the pages just like in real operational use [46].

There are many more approaches and methods of testing software and sensors applications. For example, coverage-guided fuzz testing helps discover bugs and potential security issues that other quality assurance (QA) processes might miss [47]. It sends random inputs to an instrumented version of the application in an effort to cause unexpected behaviour, such as a crash. Such behaviour indicates a bug that should be addressed.

These test approaches are automated using dedicated platforms linked to the source control management application. Popular options for software development automated testing management are Gitlab CI/CD and Jenkins, both open-source solutions that can be adapted to the requirements of any project.

Implementing and running automated software testing can quickly identify errors and prevent issues in the production implementations. By lowering the risk of software errors, the system reliability is increased.

E. SECURITY

Any SCSA contains several physical and software components, often distributed over a large area. This creates a lot of possible security breaches at various levels including sensor hardware, embedded software, network communications and cloud servers. Allowing third parties to intentionally or involuntary alter the normal operation of the system can lead to significant reliability issues. To prevent and protect against attacks there are several principals and techniques that can be used in the case of SCSA.

Network segregation is a common technique used to protect traffic while in transit over the internet. Physical network segregation is expensive because it requires a complete infrastructure just for a single system. Virtual segregation uses methods such as virtual local area network (VLAN) or virtual private network (VPN) to create a dedicated communication channel for the entities of a system. Virtual segregation is done at software level and it is often paired with end-to-end encryption to ensure data privacy even if an attacker accesses the virtual private network.

Automated continuous analysis can identify sensors that are malfunctioning based on comparisons between current readings and historic values, but also between values from neighbouring sensors [13].

Digital cyber-attacks are also carried out over longer periods of time. The attacker gains access using a vulnerability and creates a permanent way of access, while concealing his presence, in order to spy on the compromised system. These intrusions can be identified using log analysis to check for new connections from unknown sources as well as suspicions activities. Such logs are recorded by most modern computer software and can help with intrusion detection as well as forensic analysis, troubleshooting and compliance checking [48].

The security of modern systems is influenced a lot by the architecture, initial setup and configuration. It is important to provide access to the system on a need-to-know basis and create procedures on how and who should interact with the system. Proper management of passwords and other secrets is also important.

Another approach to improve security refers to the previous topic of automated testing. Vulnerability testing can be applied to components or the entire system and is performed automatically before deploying changes, but is also performed regularly on production systems.

III. MODEL

We designed and developed a Sensor-Cloud System (SCS), in order to create an IoT architecture as reference framework for intelligent products. Its goal is to transform information into knowledge along the product lifecycle, thus improving product and service quality, efficiency and sustainability.

In order to be able to measure and compare the reliability of SCS, a general score is needed. We propose a formula based on the five main reliability areas discussed in Section II, resulting in a sum of 5 factors R_x with their respective coefficients K_x (3).

$$R = K_n R_n + K_r R_r + K_b R_b + K_t R_t + K_s R_s \tag{3}$$

Each of the R_x reliability factors is a percentage with a value between 0 and 1, while the K_x coefficients can be distributed with the condition $\sum K_x = 10$, resulting in a maximum reliability score $R = 10$.

Taking into account the impact that each of the five areas has on the SCS, we established the following coefficients:

- Network $K_n = 4$
- Recovery $K_r = 1$
- Backup $K_b = 1$
- Testing $K_t = 2$
- Security $K_s = 2$

The network received the highest coefficient as it represents the backbone of the entire system information flow and can make or break the functionality of the SCS. K_n influences the factor measuring network latency, retransmission rate and bandwidth. The recovery and backup functions are nice to have and reduce data loss during exceptional scenarios, so they received a coefficient of one. The testing and security elements influence both the remote sensor hub and the central cloud servers and have the potential to disrupt functionality due to unintentional mistakes or intentional attacks on the SCS. As a consequence, each of the two coefficients received the value of two.

The network communication is the most important factor of the system. It is defined as

$$R_n = \frac{1}{3}\left(e^{-L} + (1-r) + \left(1 - \frac{B_{avg}}{B_{max}}\right)\right) \tag{4}$$

where L is the latency, r is the retransmission rate and B is the bandwidth.

The recovery reliability R_r is defined as uptime, or the percentage of the time the system was operating nominally.

The backup reliability is influenced by the storage capacity and the recovery time required to send all the data to the server, which is determined by the communication bandwidth already factored into R_n. Considering c the backup storage capacity and c_{max} the capacity of the data generated during the maximum outage period, we define:

$$R_b = \frac{c}{c_{max}} \tag{5}$$

The software testing reliability R_t is defined by the code coverage percentage.

The security component of the reliability score is measured based on the number of vulnerabilities v as defined in (6).

$$R_s = 1 - \frac{v}{10 + v} \tag{6}$$

The system architecture was adapted with feature flags to allow us to enable various reliability improving approaches and test them individually or in combinations. An example solution was implemented with the intent to create a modular system that best implements the concepts of plug & play and ease of installation.

The data capturing solution is inspired by modular industrial programmable logic circuits (PLCs), allowing for one or more specialized expansion modules that can be combined with a main module. In this way, highly efficient and cost-effective solutions can be rapidly implemented, leaving open any opportunities for future expansion. The architecture implemented is presented in Fig. 5, in an exemplary and non-exhaustive manner.

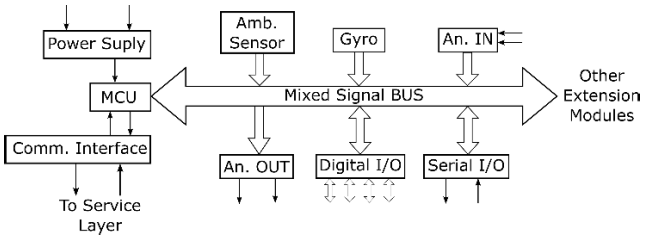


FIGURE 5. Sensor data acquisition architecture.

The power module is in charge of regulating the power supply voltages applied to the common bus and providing the correct pull-ups to the data lines. We opted for two integrated DC-DC Buck type regulators LM2670 from Texas Instruments, with fixed output voltage, with a maximum deliverable current of over 3A, one for 5V and the other for 12V. These particular regulators power the module with unregulated voltages between 15 and 40V, offering an efficiency of over 90%.

The module has been equipped with three resettable fuses as further protection compared to those already provided by the integrated regulator. One is positioned at the general power input, while the other two each supervise one of the two power lines of 5V and 12V.

The microcontroller unit (MCU) represents the brain of the acquisition device. It houses all the logic for the management of the slave modules as well as supervising the communication with the cloud servers. Given the modular concept for this application, the indispensable requirements are represented by the availability of the Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI) and serial communication buses. In addition to these, the MCU module must be able to communicate with the service layer. We chose to adopt a compact, open source, development and prototyping board based on ESP32 (ESP-WROOM-32) which integrates a Wi-Fi 802.11 b / g / n, a Bluetooth dual-mode, classic and Bluetooth low energy (BLE), and 34 General-purpose input/output (GPIO) ports. It supports data rates of up to 150 Mbps, has an antenna output power of 20.5 dBm for maximum range and a sleep mode consumption of only 5 μA. It also has interfaces for temperature sensors, touch sensors, Secure Digital (SD) card, Universal Asynchronous Receiver / Transmitter (UART), SPI, Secure Digital Input Output (SDIO), I2C, Light Emitting Diode (LED) Pulse-Width Modulation (PWM), Motor PWM, Infrared (IR), micro Universal Serial Bus (USB) connector, Boot button and reset button.

Notable features of this single board computer (SBC) include cryptographic hardware acceleration, storage devices management such as compact flash and SD cards, secure IEEE 802.11 connectivity, secure boot flash encryption, wide options for peripheral interfaces (34 GPIO). The board incorporates several modules as presented in Fig. 6. Each of them is further presented in the following paragraphs.

In the implementation presented here, the relevant features of the SBC are the communication functions together with the calculation capabilities in order to pre-process the

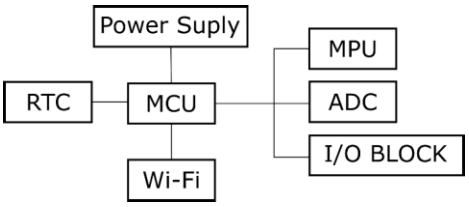


FIGURE 6. SBC modules diagram.

information to be sent to the service layer. The MCU coordinates all the modules to fulfil this goal.

It is essential that the data acquisition board is able to order sensor readings in chronological order. For this reason, a Real Time Clock (RTC) module was added with the goal of keeping current date and time information updated with extreme precision even in the absence of external power. A DS1302 chip by Maxim Integrated was chosen in this project which contains a real-time clock, calendar and 31 bytes of battery-backed Random-Access Memory (RAM). It communicates with the MCU via a serial interface and provides information such as seconds, minutes, hours, day, date, month and year. The month end date is automatically adjusted for months with less than 31 days, including corrections for leap year.

The DS1302 can interface with the MCU through synchronous serial communication. Only three connections are required to communicate with the clock or RAM: CE, I/O (data line) and SCLK (serial clock). Data can be transferred to and from the clock or RAM one byte at a time or in a burst of up to 31 bytes. DS1302 is designed to operate at ultra-low power and retain data and clock information at less than 1μW.

For some applications it may be useful to be able to detect and measure movements or vibrations in order to be able to evaluate their position in space, or through data analysis, detect any anomalies in operation and take the necessary steps to prevent downtime. For this purpose, a Motion Processor Unit (MPU) has been selected to detect movements and vibrations of the monitored product in real time. Due to low costs, robustness and wide diffusion and availability on the market, the MPU-6050 was opted for. It is an integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, a 3-axis accelerometer and a Digital Motion Processor (DMP), all in a small footprint of just $4\times4\times0.9$mm. An on-chip 1024-byte FIFO buffer helps reduce system power consumption by allowing the processor to read sensor data in bulk and then enter a low-power mode as the MPU collects more data. Communication with all registers of the device is performed using I2C at 400kHz.

For those situations when it is required to actively interact with the monitored device, or with some accessory connected to it, a special module was added to control small loads. MCP23017 from Microchip was used for its two 8-bit digital I/O banks that can be addressed separately via its I2C communication bus. To be able to control higher loads, SN75468 was added as a transistor array capable of handling high voltages and relatively high currents. This module gives the application considerable flexibility in terms of the possibility of use, while preserving the compactness and cost-effectiveness requirements of the solution.

To be able to measure analogue signals, an Analogue to Digital Converter (ADC) was connected. The chip ADS7828 was used, a 12-bit single-power, low-power data acquisition device that features an I2C serial interface, an 8-channel multiplexer, a sample and hold amplifier and an internal asynchronous clock. The combination of an I2C, two wire serial interface and power consumption make the ADS7828 ideal for this application.

To link the SBC to the cloud we opted for an application layer communication protocol most used in the home automation sector and, combining ease of use and descriptive power, Message Queue Telemetry Transport (MQTT). It works on top of the TCP / IP protocol stack and allows an exchange of messages through a publish-subscribe model: publishers publish messages to channels on which subscribers can listen and subscribe. The functioning of this mechanism is similar to what happens on platforms like Youtube: people create videos on their channels; users of the site, by subscribing to the channels, will then be able to know when the creators publish videos. The entities referred to up to now as a channel have a specific name in the MQTT protocol and are called topics.

This protocol creates a structured data flow, helping to integrate several data providers and several data readers (cloud servers, mobile applications, web applications), as shown in Fig. 7.

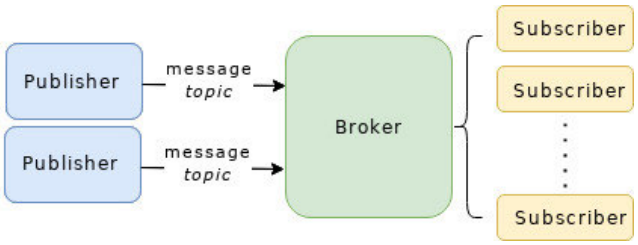


FIGURE 7. Relation diagram of a MQTT network.

An MQTT topic is a string that can represent a type of data of interest that is read and forwarded on the network, or it can represent a particular room of the environment that is going to be taken into consideration; they are simple to use tools but, at the same time, very versatile.

The entity that coordinates the activities of publishers and subscribers through the topics is the broker. The advantage of MQTT topics is that they can be organized in hierarchical structures, which describe in an optimal way the environments we take into consideration, for example, representing the set of sensors used, the type of data they capture, the product in which are placed.

For the demonstration of the concepts covered by this article, we used the open source broker Mosquitto; the installation package also includes two simple MQTT Clients, mosquitto_pub and mosquitto_sub, useful for verifying the operation of what was gradually implemented.

On the cloud server side, the information received is processed and stored using a MariaDB relational database, MySQL open source fork. The controller running in the cloud retrieves and indexes all the endpoints made available by the various devices with the respective information regarding the type, unit of measurement, bi-directionality of the communication and, subsequently, begins to read all the data published on the broker and to store them in an orderly manner in the appropriate table.

To make this data available to users, the cloud runs an implementation of the REST service protocol. The access parameter structure was created in such a way to facilitate modularity and extensibility. In order to demonstrate the properties of the system and its scalability, we present an example of obtaining historic data from one device within a specified time interval. It is sufficient to make a call structured as shown in Fig. 8.

```
http://2.226.154.26:8080/?dev_name=dev_01&ep_name=mag_x
&dt_from=2020-05-27&dt_to=2020-05-27&limit=100
```

FIGURE 8. REST request over HTTP.

As can be seen, the call is made up of the server address, the port and a series of parameters that characterize the request. In particular, the names of the parameters making up the request have been highlighted in blue:

- dev_name: is the name of the monitored product;
- ep_name: is the name of the endpoint sensor;
- dt_from and dt_to: indicate the time interval for which the data is to be obtained;
- limit: if specified, only the N most recent records will be returned from the specified interval.

An excerpt of a typical response to the above call is presented in Fig. 9.


```
▾ data:
    dev_short_desc:    "Demo Sensor"
    dev_name:          "dev_01"
    dev_sn:            "0001"
    dev_model:         "ESP WeMos D1"
    ep_name:           "mag_x"
    ep_type:           "uint16_t"
    ep_um:             "grad"
    num_records:       100
  ▾ historical:
    ▾ 0:
        data_ts:       "1590597159.671362"
        server_ts:     "2020-05-27 18:32:40"
        data_value:    "215"
    ▾ 1:
        data ts:       "1590597158.652334"
```


FIGURE 9. REST response in JSON format.

It should be noted that in the historical data, in addition to the measurement value, two other fields are indicated:

- data_ts: indicates in TIMESTAMP format the date and time of data acquisition;
- server_ts: indicates in DATETIME format the date and time when that data was entered in the Database.

All the code dealing with time values uses time synchronization with NTP. In order to account for time zone differences and daylight-saving time, all times are converted to Coordinated Universal Time (UTC) as early as possible, all the processing is done in UTC and only when necessary to display times to humans the information is convert to the user desired local time.

The experimental model also includes a web application used for real time monitoring, as well as data analysis. The application is based on the modern framework Angular 9 and has the ability to communicate with all the protocols used by the device and cloud, namely the MQTT protocol and the HTTP protocol, whose data are exposed in REST mode. A view from the application page displaying real time sensor values is presented in Fig. 10, showing a chart of real time acceleration values read from the MPU.

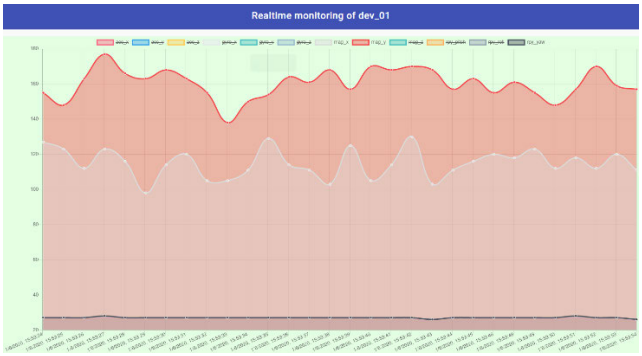


FIGURE 10. Real time sensor values in web application.

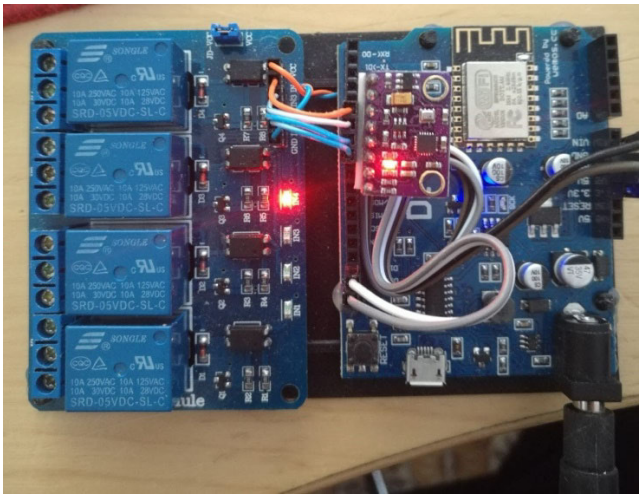


FIGURE 11. Experimental SBC with actuators.

For SBC with attached actuators, shown in Fig. 11, the application provides an overview of current state and the ability to change the state, as in Fig. 12.

All the software developed for this project was placed into a Gitlab server using the git version control system. Test were written to verify the correct functionality of the software and

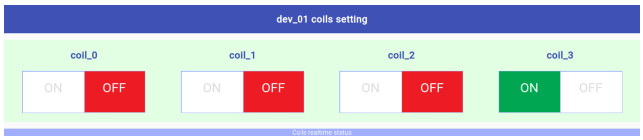


FIGURE 12. Actuators display in the web application.

an automated Continuous Integration (CI) process was set up using Gitlab in order to test and build new software versions.

IV. TESTS AND RESULTS

A. NETWORK RELIABILITY

The network communication is a complex problem, with a myriad of modules and protocols involved. At the same time, the sensor cloud systems use distributed equipment that is most often limited in terms of power usage, computational power, memory and storage. This leads to compromises at the network stack level, more explicitly, disabling features and protocols that add complexity but provide reliability improvements. Given this reality, it is important to evaluate the default values and options available for the modules used for network communication between sensors and the cloud.

To begin with, we explore the reliability of the physical transmission media. We connect a test controller with sensors to a computer over Wi-Fi using the standard IEEE 802.11g, and using an ethernet copper cable. The computer is running Wireshark, a network packet sniffer software. While sending the same number of sensor readings with similar data, as shown in Table 1, we notice a significant difference between the frame retransmission rate over wireless and cable.

TABLE 1. Frame retransmission rate.

Medium	Frames sent	Frames resent	Retransmit ratio
Wireless	6572	842	12.81%
Cable	6568	1	0.01%

While wireless networks are easier to setup, the cable is a more reliable option, with better throughput and less delay. This result is in line with the widely accepted common knowledge for the existing technology. However, the performance and reliability differences are expected to decrease with the advancement of wireless IEEE 802.11 standards and the introduction of the 5G protocols.

Next, we explore the Wi-Fi configuration options available on the ESP32 MCU board. All functions and capabilities are defined in the wifi_init_config_t structure that is passed to the esp_wifi_init function. Some options can also be changed using dedicated API functions.

By default, the module starts in soft AP mode, allowing up to 5 devices to connect to it. This behaviour creates noise as we only want to connect each device to a dedicated access point. Table 2 shows the measured bandwidth for the two modes. When setting the operation mode to station, the maximum tested bandwidth increased by 25%.

TABLE 2. Bandwidth by operation mode.

Operation mode	Bandwidth
Soft AP	28 Mbps
Station	35 Mbps

By reducing the number functions that the WiFi module has to provide, the performance improved, as expected. The MCU had to run less code and interruptions from wireless devices, while the wireless module sent less signals into the air. This change also has an impact on the power consumption, reducing the load on the power supply module.

In order to analyse the influence of the packet size on the packet delivery ratio over wireless connections, we created predefined packets and sent them over the test infrastructure described above, from the sensor controller to the computer running Wireshark. One hundred packets were sent for each of the tested packet sizes, as presented in Fig. 13, and an average PDR was computed using the packet analysis in Wireshark. We have found that a packet size between 10 and 50 bytes has a negligible influence on the PDR, as presented in Fig. 13, and we can also observe that the real network built for our project performed better than the simulation presented by [12], at 87% PDR versus 75% to 55% depending on packet size. This significant improvement and consistency can be attributed to the MCU configuration optimizations, but also to technological advances related to embedded devices and WiFi standards.

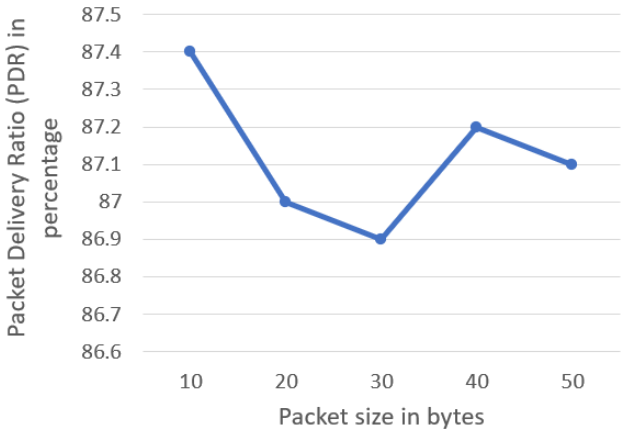


FIGURE 13. PDR as a function of packet size.

The AP connection parameters and credentials are saved by default both in memory and on the non-volatile flash card. This is preferrable in order to allow quick reconnection, including after reboot. Configuring the connection in static mode, providing the IP, gateway, subnet and DNS improved the start-up time by 2 seconds as opposed to DHCP. While this can increase the system implementation time, it is a good option when reboot time is important.

Wi-Fi radio signal power is regulated in most parts of the world and differs from one country to another. By default,

the ESP32 module is set to China, and to automatically try to obtain the country of the AP to which it is connected. In order to speed up network initialization, the module can be configured in manual mode. The transmitting power can also be set independently of the country settings, between 2dBm and 20dBm. Increasing the power will improve the wireless range and connection stability, but local radio regulations must also be respected.

The network module buffer size can also be altered using the initialization configuration structure. However, we did not see any notable change in performance by changing the buffers size 50% up or down from the default value.

As described above, network configurations can have a significant impact on the controller performance. Simply by changing some of the default config values according to the application setup, the network communication reliability can be increased.

The reliability score was computed for each scenario, considering the average required bandwidth 20 Mbps. The results presented in Table 3 reveal a variation of 0.29 between the best and worst tested scenarios.

TABLE 3. Network communication reliability.

Operation mode	Latency	Retransmit ratio	Bandwidth	R_n
Wireless Soft AP	0.247 s	13.78%	28 Mbps	0.643
Wireless Station	0.239 s	11.87	35 Mbps	0.699
Cable	0.001 s	0.01%	100 Mbps	0.932

B. AUTO RECOVERY

The test model was developed with auto recovery for several components. The embedded device is set to resume operations after power failures, to reboot itself when a sensor is not responding and to reboot with optional remote firmware update when requested by the cloud server. The test location experienced frequent short power interruptions and we recorded the recovery results in Table 4 as a comparison between a controller with auto recovery enabled and one that required manual intervention.

TABLE 4. Auto ecovery outage over two weeks.

Device	Event	Number of incidents	Missing data interval	R_r
Auto recovery	Power Outage	8	14 minutes	0.998
Manual recovery		8	23 hours	0.808
Auto recovery	System malfunction	4	9 minutes	0.998
Manual recovery		5	7 hours	0.941

Two identical sensor hubs were placed in a test room, only one having auto recovery enabled. Over a period of five days there were 8 power outages that resulted in 14 minutes of missing data for the hub with auto recovery, most of the period due to the power being out, and a short amount from the bootup sequence duration. The average boot time is 31ms and a total of 846ms are needed to connect to the wireless network and establish a connection with the cloud. The second hub had to be manually started up and reconnected to the cloud. Since some of the incidents occurred at night when no human was around to tend to the issue, the total missing data interval was over 19%.

A similar difference was observed for system malfunction incidents. The basic script created to read the sensors and send the data to the cloud would sometimes die or get stuck. The auto recovery function used a watchdog task to check the status of the process running the script, and automatically restart it when necessary. The missing data interval was again reduced by an order of magnitude, from hours to minutes.

The cloud component also benefitted from auto recovery techniques. It uses Docker containers orchestrated by Kubernetes and was configured to monitor and manage containers automatically. The tests were successful in starting one container for each software component in the cloud (database, sensors endpoint and web application) and then creating more incoming data processing containers as we connected more sensors. Over the two weeks period of the test presented in Table 1, we saw a number of three incidents related to non-responsive containers, all of them being automatically detected and resolved by Kubernetes. Our assessment concludes that modern cloud software architectures using microservices, containers and orchestrators has evolved significantly during the last years and can perform reliably in production environments.

C. LOCAL BACKUP

The experimental setup is fitted with a local backup system using a non-volatile memory card. Whenever a sensor controller cannot reach the cloud due to network issues or cloud service outage, it writes messages containing the information from sensors in a local file. The MQTT packets are serialized as UTF-8 text to be sent to the cloud later.

When designing a local backup feature, it is important to correctly size the storage capacity and transfer speeds. Most sensor readings provide one or more decimal values. The experimental setup uses one motion sensor, one temperature sensor, one humidity sensor with reading interval set at one minute. The backup file size for these 3 sensors over 10 minutes was 49.02 Mb. The resulting average packet size for one sensor is 1634 kb, while the average bit rate for the device is 81.7 kbps. Generating data at this bitrate, results in a theoretical 841 MB per day, which was verified in practice, leading us to conclude that these sensors generate data at a constant bitrate. While storage options available today far outperform these requirements, we should note that the data was not optimized for storage and much lower bitrates can be achieved by encoding the sensor values with less message structural data.

Without any local storage the backup reliability indicator is 0. In the scenario described above, considering a seven-day maximum outage and an installed storage memory of 4GB, the R_b is 0.695.

The addition of video streaming capabilities to the system generates a significant bitrate increase. Many popular video encoders on the market use variable bitrate encoding. Depending on the level of motion in your video content and your keyframe interval, the actual encoded bitrate of the stream varies. Table 5 presents recommended values for the popular h.264 video codec.

TABLE 5. Audio – Video Streaming Bitrates.

Quality	Resolution	Video bitrate	Audio Bitrate
SD	640 x 480	1000 - 1500 kbps	96 kbps
HD	1280x720	1500 - 4000 kbps	128 kbps
Full HD	1920x1080	4000 - 8000 kbps	192 kbps
Ultra HD	3840x2160	8000 - 14000 kbps	192 kbps

To test the local backup, we used a memory card rated at 8GB capacity and UHS Class I. The controller was linked to 4 sensors, one being a Full HD camera recording an area with limited movement. The internet connection to the cloud was interrupted to force the local backup. The storage filled up to 90% in 3 hours and 46 minutes, resulting in an average bitrate of 4247 kbps.

The internet connection was re-established over a local Wi-Fi network and further over ethernet and fibre optics to the cloud. The application started to receive live data, while the backup required 64 minutes to be transferred and made available on the cloud which corresponds with the theoretical limit of 22 Mbit/s average throughput for the 802.11 Wi-Fi standard.

The approach presented here should be adapted to the required bitrate of each application and to the bitrate capability of the Wi-Fi, taking into consideration possible equipment upgrades using the latest IEEE 802.11 standard.

D. AUTOMATED SOFTWARE TESTING

Initial development of the embedded software, cloud scripts and web application did not use any method of code testing. Each feature was documented and implemented in a separate git branch. We measured several indicators to help estimate the impact of automated testing: duration of review, changes made during review, number of issues identified after merging the feature branch.

Comparative results between no tests and using automated testing are presented in Table 6. The average test code coverage was 68% and we anticipate that the results would improve even more with better code coverage. The testing reliability indicator R_t increased in this case from 0 to 0.68.

Note that while writing tests takes some time, the overall time required for development, identifying issues and fixing them was decreased and required the involvement of fewer people.

It is clear that automatically testing all changes made to the software code base can reduce the number of errors and the time required for development and QA. We conclude that the reliability of the entire system was significantly increased by ensuring the deployed software more strictly conforms to technical requirements and provides legal compliance.

TABLE 6. Development Quality Assessment.

Indicator	Value no test	Value with tests
Duration of review (days)	5	2
Changes made during review (commits)	6	1
Number of issues post merge (bug reports)	3	1
Integration issues (communication between modules)	2	0

E. SECURITY

At the sensor component level, we implemented embedded code protection using secure boot flash encryption. Furthermore, sensor controllers are placed inside appliances or machines and inside buildings that have some level of physical protection and access control.

At the network level we implemented communication encryption using cryptographic hardware acceleration at both ends. This ensures data privacy and security while traveling on third party communication infrastructure.

The precaution methods described above represent a significant deterrent to criminal activities. While it is difficult to measure the impact of these security measures, it is clear that they make it more difficult and costly for attackers to infiltrate the SCS and to extract data or alter data without leaving traces. In term, reducing the risk of attacks has a positive influence on the reliability of the SCS. As stated in [29], it is important to use a well-defined procedure to enhance security at several layers.

To further test for vulnerabilities and secure the system we used a set of predefined penetration tests [49] to become resilient against cyber-attacks. We ran a number of tests targeting the infrastructure and the cloud web application and identified a total number of 2 high security risks, 12 medium risks and 18 low security risks, as shown in Fig. 14.

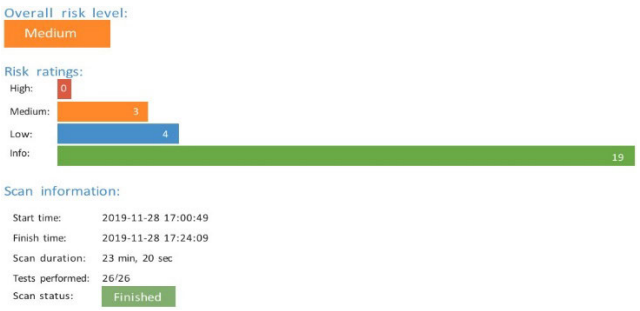


FIGURE 14. Vulnerability scan report for the web application.

The vulnerabilities are ranked based on the impact they can have on the system. High risk are vulnerabilities that can be exploited to execute code on the system, medium risk vulnerabilities can lead to exposed data, while low vulnerabilities can cause system slowdown. Info rated risks refer to updates

available for external libraries that can improve the system performance.

Following information and instructions provided by the test platform, we managed to fix all high and medium security risks, but ignored 2 low risks coming from external libraries used in our development. These low risks were evaluated as extremely difficult to exploit by an outside attacker, but also difficult to fix and sustainably update the external libraries periodically.

By using the reliability policy, the vulnerabilities were identified and fixed. The security reliability indicator has improved from 0.238 to 0.833.

The same suite of tests was set to run every month on our experimental system. Over time, we discovered a continuous evolution of the known software vulnerabilities and possible attack strategies, but also timely updates for the third-party software to mitigate those vulnerabilities.

While it is difficult to empirically evaluate the implications that security has over reliability, it is safe to say that in the event of a successful attack the system disruption can be severe. Consequently, we reiterate the best practice of monitoring for vulnerabilities and updating software components on a regular maintenance schedule.

The final scores for each reliability indicators are presented in Fig. 15. Overall, summing up all the reliability indicators according to the formula, the score R varies between 3.856 in the worst case when no optimisations were considered and 8.447 when the reliability policy was applied and several improvements techniques were used in each of the five key areas.

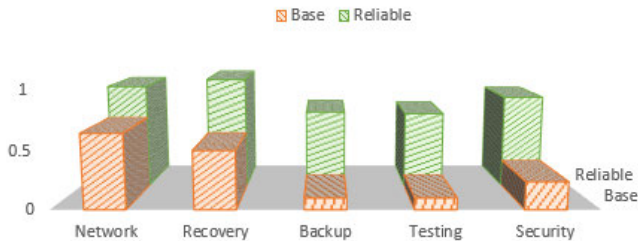


FIGURE 15. Reliability improvement results.

V. CONCLUSION

As the industry matures, the reliability concerns around IoT applications grow. Once a project goes past the proof-of-concept phase, users, developers, owners and stakeholders in general become aware of glitches, delays, incorrect data, lack of consistency and other reliability topics.

This article presents a modular and flexible IoT setup, designed for a product knowledge and lifecycle management solution. A fundamental feature is the ability to harmonize the information exchange format to facilitate data acquisition for control dashboards and more sophisticated distributed-cloud systems.

Five topics with impact on system reliability were discussed and a novel, tiered, result driven, reliability control and improvement approach for IoT is put forward. It is a new policy, proposed to ensure the reliability of a SCS by taking action in five main areas. Network communication is a fundamental part of such systems and several improvement steps with proved results are presented. Inevitably, unexpected situations do appear at some time and we present solutions to alleviate the problems caused with techniques such as local backup and auto recovery. Automated software testing and continuous integrations are discussed and their impact on software quality and reliability is evaluated. Security concerns are also treated in the article and an automated approach to continuous security testing is presented.

A new formula is presented in order to measure and compare reliability of complex SCS. The 10-points scale reliability score has increased by over 4.5 points in the case of the implemented test model that is presented in this paper, proving the efficacy of the reliability policy based an original five level tiered approach.

While the results are very promising, one should also consider the extensibility of this model. As technology evolves, more than five topics can be taken into account. The scoring formula includes coefficients that were arbitrarily selected for this particular experiment, based on arguments presented in Section III. With more experimentation on various use cases, better coefficients might be established for some categories of applications.

Taking advantage of the experience gained building and testing the system for the POKET research project, the team is looking forward to working in the future to create a solid software framework for SCSA, with built in reliability features, to be released as open source.

ACKNOWLEDGMENT

The authors would like to thank our employing companies, RID International Center, Focus Innovazione and Tesagon International for supporting the POKET research project and the publication of this article.

REFERENCES

[1] R. Angeles, "RFID technologies: Supply-chain applications and implementation issues," *Inf. Syst. Manage.*, vol. 22, no. 1, pp. 51–65, Dec. 2005, doi: 10.1201/1078/44912.22.1.20051201/85739.7.

[2] R. L. Albuquerque, J. F. Hübner, G. E. D. Paula, J. S. Sichman, and G. L. Ramalho, "KSACI: A Handheld device infrastructure for agents communication," in *Intelligent Agents VIII*, vol. 2333, J.-J. Ch. Meyer and M. Tambe, Eds. Berlin, Germany: Springer, 2002, pp. 423–435.

[3] G. G. Meyer, K. Främling, and J. Holmström, "Intelligent products: A survey," *Comput. Ind.*, vol. 60, no. 3, pp. 137–148, Apr. 2009, doi: 10.1016/j.compind.2008.12.005.

[4] C. M. Coman, A. Florescu, and G. Stigliano, "Distributed sensors array for composite materials manufacturing quality assurance," in *Proc. 11th Int. Symp. Adv. Topics Electr. Eng. (ATEE)*, Bucharest, Romania, Mar. 2019, pp. 1–6, doi: 10.1109/ATEE.2019.8724867.

[5] C. M. Coman and A. Florescu, "Electric grid monitoring and control architecture for industry 4.0 systems," in *Proc. Int. Symp. Fundamentals Electr. Eng. (ISFEE)*, Bucharest, Romania, Nov. 2018, pp. 1–6, doi: 10.1109/ISFEE.2018.8742435.

[6] C. M. Coman, A. Florescu, and C. D. Oancea, "Improving the efficiency and sustainability of power systems using distributed power factor correction methods," *Sustainability*, vol. 12, no. 8, p. 3134, Apr. 2020, doi: 10.3390/su12083134.

[7] A. Zarnescu, R. Ungurelu, M. Secere, C. M. Coman, and G. Varzaru, "Putting Internet-of-things at the service of sustainable agriculture. Case study: Sysagria," *Lucrari Ştiinţifice*, vol. 62, no. 1, pp. 9–14, 2019.

[8] I.-I. Patru, M. Carabas, M. Barbulescu, and L. Gheorghe, "Smart home IoT system," in *Proc. 15th RoEduNet Conf., Netw. Educ. Res.*, Bucharest, Romania, Sep. 2016, pp. 1–6, doi: 10.1109/RoEduNet.2016.7753232.

[9] L. Qi, C. Hu, X. Zhang, M. R. Khosravi, S. Sharma, S. Pang, and T. Wang, "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Trans. Ind. Informat.*, vol. 17, no. 6, pp. 4159–4167, Jun. 2021, doi: 10.1109/TII.2020.3012157.

[10] B. C. Florea, "Blockchain and Internet of Things data provider for smart applications," in *Proc. 7th Medit. Conf. Embedded Comput. (MECO)*, Budva, Montenegro, Jun. 2018, pp. 1–4, doi: 10.1109/MECO.2018.8406041.

[11] B. C. Florea and D. D. Taralunga, "Blockchain IoT for smart electric vehicles battery management," *Sustainability*, vol. 12, no. 10, p. 3984, May 2020, doi: 10.3390/su12103984.

[12] P. P. Maktedar and V. S. Deshpande, "Interpretation of reliability in wireless sensor networks," in *Proc. Int. Conf. Cloud Ubiquitous Comput. Emerg. Technol.*, Pune, India, Nov. 2013, pp. 104–107, doi: 10.1109/CUBE.2013.54.

[13] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Deploying wireless sensor networks with fault-tolerance for structural health monitoring," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 382–395, Feb. 2015, doi: 10.1109/TC.2013.195.

[14] S. Fang, L. D. Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan, and Z. Liu, "An integrated system for regional environmental monitoring and management based on Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1596–1605, May 2014, doi: 10.1109/TII.2014.2302638.

[15] S. Misra, S. Chatterjee, and M. S. Obaidat, "On theoretical modeling of sensor cloud: A paradigm shift from wireless sensor network," *IEEE Syst. J.*, vol. 11, no. 2, pp. 1084–1093, Jun. 2017, doi: 10.1109/JSYST.2014.2362617.

[16] T. Ojha, S. Misra, N. S. Raghuwanshi, and H. Poddar, "DVSP: Dynamic virtual sensor provisioning in sensor–cloud-based Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5265–5272, Jun. 2019, doi: 10.1109/JIOT.2019.2899949.

[17] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G," *Proc. IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019, doi: 10.1109/JPROC.2019.2913443.

[18] B. Y. Ooi, W. L. Beh, W.-K. Lee, and S. Shirmohammadi, "A cloud system to improve sensor availability and data reliability in remote monitoring," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, Houston, TX, USA, May 2018, pp. 1–6, doi: 10.1109/I2MTC.2018.8409811.

[19] B. Y. Ooi, W. L. Beh, W.-K. Lee, and S. Shirmohammadi, "Using the cloud to improve sensor availability and reliability in remote monitoring," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 5, pp. 1522–1532, May 2019, doi: 10.1109/TIM.2018.2882218.

[20] S. Cai and V. K. N. Lau, "Cloud-assisted stabilization of large-scale multiagent systems by over-the-air-fusion of IoT sensors," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7748–7759, Oct. 2019, doi: 10.1109/JIOT.2019.2901576.

[21] A. Sajid, H. Abbas, and K. Saleem, "Cloud-assisted IoT-based SCADA systems security: A review of the state of the art and future challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016, doi: 10.1109/ACCESS.2016.2549047.

[22] X. Xu, X. Zhang, M. Khan, W. Dou, S. Xue, and S. Yu, "A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 789–799, Apr. 2020, doi: 10.1016/j.future.2017.08.057.

[23] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, Mar. 2020, doi: 10.1007/s11280-019-00684-y.

[24] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, and Y. Ren, "Stability of cloud-based UAV systems supporting big data acquisition and processing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 866–877, Jul. 2019, doi: 10.1109/TCC.2017.2696529.

[25] M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Revisiting context-based authentication in IoT," in *Proc. 55th Annu. Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2018, pp. 1–6, doi: 10.1145/3195970.3196106.

[26] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, "A robust ECC-based provable secure authentication protocol with privacy preserving for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3599–3609, Aug. 2018, doi: 10.1109/TII.2017.2773666.

[27] M. Z. A. Bhuiyan, M. Zaman, G. Wang, T. Wang, and J. Wu, "Privacy-protected data collection in wireless medical sensor networks," in *Proc. Int. Conf. Netw., Archit., Storage (NAS)*, Shenzhen, China, Aug. 2017, pp. 1–2, doi: 10.1109/NAS.2017.8026872.

[28] K. Dubey, M. Y. Shams, S. C. Sharma, A. Alarifi, M. Amoon, and A. A. Nasr, "A management system for servicing multi-organizations on community cloud model in secure cloud environment," *IEEE Access*, vol. 7, pp. 159535–159546, 2019, doi: 10.1109/ACCESS.2019.2950110.

[29] M. Obaidat, M. Khodiaeva, S. Obeidat, D. Salane, and J. Holst, "Security architecture framework for Internet of Things (IoT)," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, New York City, NY, USA, Oct. 2019, pp. 0154–0157, doi: 10.1109/UEMCON47517.2019.8993096.

[30] V.-A. Zamfir, M. Carabas, C. Carabas, and N. Tapus, "Systems monitoring and big data analysis using the elasticsearch system," in *Proc. 22nd Int. Conf. Control Syst. Comput. Sci. (CSCS)*, Bucharest, Romania, May 2019, pp. 188–193, doi: 10.1109/CSCS.2019.00039.

[31] J. D. O'Sullivan, G. R. Daniels, T. M. P. Percival, D. I. Ostry, and J. F. Deane, "Wireless LAN," U.S. Patent 5 487 069 A, Jan. 23, 1996.

[32] A. S. Tanenbaum, *Computer Networks*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2003, p. 101.

[33] C. Henry, "FCC gets five new applications for non-geostationary satellite constellations," *Space News*, Mar. 2017. Accessed: Nov. 17, 2020. [Online]. Available: https://dev.spacenews.com/fcc-gets-five-new-applications-for-non-geostationary-satellite-constellations/

[34] W. Simpson. *The Point-to-Point Protocol (PPP)*. Accessed: Nov. 17, 2020. [Online]. Available: https://tools.ietf.org/html/rfc1661

[35] W. Simpson. *PPP in HDLC-Like Framing*. Accessed: Nov. 17, 2020. [Online]. Available: https://tools.ietf.org/html/rfc1662

[36] D. Rand. *PPP Reliable Transmission*. Accessed: Nov. 17, 2020. [Online]. Available: https://tools.ietf.org/html/rfc1663

[37] *Manual: Connection Oriented Communication (TCP/IP)—CableFree RadioOS*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.cablefree.net/support/radio/software/Manual:Connection_oriented_communication_(TCP/IP)

[38] *Container Orchestration Diagram*. Accessed: Nov. 16, 2020. [Online]. Available: httpzakis://avinetworks.com/wp-content/uploads/2018/12/container-orchestration-diagram-1.png

[39] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Proc. 18th ACM Symp. Operating Syst. Princ.*, Oct. 2001, pp. 146–159.

[40] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications*. Norwood, MA, USA: Artech House, 1996.

[41] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proc. Int. Parallel Distrib. Process. Symp. (IPDPS) Workshop Parallel Distrib. Comput. Issues Wireless Mobile Comput.*, Apr. 2001, Art. no. 30186b.

[42] S. K. Mani, P. Barford, R. Durairajan, and J. Sommers, "What time is it: Managing time in the Internet," in *Proc. Appl. Netw. Res. Workshop*, Montreal, QC, Canada, Jul. 2019, pp. 37–44, doi: 10.1145/3340301.3341125.

[43] *IANA—Time Zone Database*. Accessed: Nov. 24, 2020. [Online]. Available: https://www.iana.org/time-zones

[44] *Time Zone Database and Code*. Accessed: Nov. 24, 2020. [Online]. Available: https://github.com/eggert/tz

[45] *Working With Time Zones*. Accessed: Nov. 24, 2020. [Online]. Available: https://www.w3.org/TR/timezone/

[46] *01-Introduction-Codeception-Documentation*. Accessed: Nov. 25, 2020. [Online]. Available: https://codeception.com/docs/01-Introduction

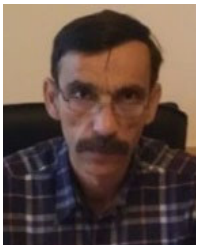
[47] *Coverage Guided Fuzz Testing | GitLab*. Accessed: Nov. 25, 2020. [Online]. Available: https://docs.gitlab.com/ee/user/application_security/coverage_fuzzing/

[48] O. Awotipe, "Log analysis in cyber threat detection," *Creative Compon.*, Jan. 2020. [Online]. Available: https://lib.dr.iastate.edu/creativecomponents/468

[49] Pentest-Tools. *Powerful Pentesting Tools, Easy to Use*. Accessed: Sep. 16, 2020. [Online]. Available: https://pentest-tools.com/home

CIPRIAN M. COMAN received the bachelor's degree in computer science and engineering and the M.S. degree in information technology management from the Politehnica University of Bucharest, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. focusing on IoT technologies for industry 4.0. From 2009 to 2014, he was a Teaching Assistant for the applied electronics optional course, algorithms design course, and software project management course with the Politehnica University of Bucharest. He is also leading an invocative ICT company, Tesagon International. He has experience working as a researcher in several projects, as well as industry experience as a software engineer, a system architect, and a manager. His research interests include the IoT, automation, digital industry, clean energy, cloud systems, web technologies, artificial intelligence, and blockchain.

GIUSEPPE D'AMICO was born in Fasano, Italy, in 1980. He received the M.S. degree in electronics engineering with the Politecnico di Bari, in 2006. From 2007 to 2018, he has been a Project Manager/PMO with Accenture S.p.A. From 2019 to 2020, he worked as a PM of research projects with Horus srl. Since 2020, he has been a PM of various initiatives, including many research projects with Focus Innovazione srl, Fasano. He worked on the design of quite complex security systems to protect large photovoltaic fields and to electrical systems design (low and medium voltage). He is an expert in design of electrical and safety systems. His research interests include ICT field, such as technological renewals of data centres, technology changes in IT infrastructure, virtualizations, data centre consolidations, technological upgrades, and redesign and implementation of environment configuration.



ADRIAN V. COMAN received the degree in engineering and the Ph.D. degree in aerospace engineering from the Military Technical Academy, Bucharest, Romania, in 1983 and 1995, respectively. From 1983 to 1990, he worked as a Researcher and a Main Researcher with ICPEM Ploiesti. From 1990 to 1993, he was an Assistant and Lecturer with the Military Technical Academy Bucharest. He continued to work as a Researcher and a Lecturer with the Military Technical Academy Bucharest, ICPEM Ploiesti, ROMARM, and Electromecanica Ploiesti, in 2007. He started a private research organization, in 2007, where he works as a Researcher and a Project Manager, currently named RID International Center. He was part of over 30 projects as a project manager or a researcher. His research interests include rocket engines, satellites, supercapacitors, sensors and data acquisition, and the IoT.

ADRIANA FLORESCU (Senior Member, IEEE) received the degree in engineering and the Ph.D. degree in electronics from the Politehnica University of Bucharest (PUB), Romania, in 1988 and 2001, respectively. Since 1993, she has been working with the Department of Industrial Electronics and Informatics, Faculty of Electronics, Telecommunications and Technology Information, PUB. She has been a Professor, since 2015, and a Ph.D. Coordinator, since 2016. She has published more than 100 publications, one patent, and ten books in these fields. Her research interests include power electronics, renewable energy conversion and artificial intelligence and currently electric vehicles, medical electronics, and the IoT. She has been the IEEE member of the Power Electronics Society, since 1995, and the Social Chair in the Executive Committee of IEEE Romanian Section, since 2013.