Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (36^{th}cycle)

Security and Reliability in Pervasive Computing

By

Pietro Chiavassa

******

Supervisor:
Prof. Filippo Gandino

Doctoral Examination Committee:
Prof. Javier Vales Alonso, Referee, Universidad Politécnica de Cartagena
Prof. Gianluca Cena, Referee, CNR-IEIIT
Prof. Maurizio Rebaudengo, Politecnico di Torino
Prof. Valentina Gatteschi, Politecnico di Torino
Ph.D. Alberto Scionti, Fondazione LINKS

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Pietro Chiavassa
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my family and to my friends, who supported me through these years.

Acknowledgements

I want to thank my family and friends for their help and support during these years. I also want to express my gratitude to my colleagues and supervisors who have accompanied me on this journey.

Abstract

The thesis presents research work exploring security and reliability in pervasive IoT systems. At first, these topics are closely analyzed in relation to state-of-the-art IoT technologies and infrastructures. Subsequently, two main fields of application are investigated: Particulate Matter (PM) monitoring with low-cost light-scattering sensors and secure intermittent computing (ImC) for energy harvesting systems.

Low-cost light-scattering PM sensors are often proposed as an IoT solution for the creation of dense monitoring networks. These devices are a miniaturization of traditional full-size optical particle counters (OPCs) and nephelometers. However, due to intrinsic technological limitations, they are often considered to be unreliable and imprecise. Multiple monitoring campaigns are conducted by placing a large number of low-cost PM sensors at an official monitoring site in the city of Turin, Italy. At first, the collected measurements are compared to the high-precision instruments of the Metropolitan City of Turin, to understand the benefits that their integration into the official monitoring network could provide. Secondly, a pipeline that performs failure detection, filtering, and calibration of these sensors is presented and evaluated on the collected data. Then, it is analyzed how the introduction of a duty cycle in their operation affects measurement quality. Finally, an improved version of a low-cost monitoring station is designed, together with the entire IoT infrastructure, to transmit, store, and visualize the collected measurements. The infrastructure also provides data validation functionalities by storing the hash of the collected data inside a blockchain.

In the context of intermittent computing, instead, a secure checkpointing utility is designed and tested on a target architecture supporting a Trusted Execution Environment (TEE): ARM Truszone for Cortex-M. Differently from other state-of-the-art approaches, the entire security chain of the platform is considered. This results in a solution that is directly applicable without the need for custom or hardware

non-available MCU features. The performance of the utility is compared with other state-of-the-art works, by also evaluating the lifetime of the device.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Pervasive computing and IoT

The term "ubiquitous computing" was first introduced by Mark Weiser in 1988 and further discussed in his 1991 article published in Scientific American [12]. Ubiquitous computing was described as the disappearance of silicon devices in the environment, allowing people to use them unconsciously or without being aware of their presence, while focusing on their goals and not on the information overhead of the technology.

Additional terms, such as "Internet of Things (IoT)" and "pervasive computing", were later introduced to highlight some of its aspects or technological characteristics. The term "Internet of Things", which is most commonly used nowadays, refers to physical objects that are embedded with sensors, software, and network connectivity, allowing them to collect and share data [13]. In the context of this work, the term "pervasive computing" is mainly intended as a synonym for IoT. Nonetheless, the pervasive component is a fundamental aspect of the research contributions presented in this work.

Even in the early work of Mark Weiser, problems such as information privacy and security were already discussed. In the world of today, these problems are magnified by the development and diffusion of IoT devices, which also introduced newer and more complex challenges.

1.2 State of the art

Pervasive computing is a complex and heterogeneous field. This section provides a background on the state of the art in IoT and the current trends. The subject is analyzed by focusing on orthogonal perspectives. First, reasons and applications of IoT technology are introduced. Secondly, IoT is presented from an architectural point of view, highlighting the different system components and the information flow between them. Finally, the role of the different enabling technologies involved in pervasive systems is discussed.

1.2.1 Motive and applications

Providing internet connectivity to physical objects allows the creation of a network of smart devices that both monitor and interact with the environment in which they are located. The objective of this computational approach is to improve the efficiency of business processes, allow data-driven decision-making, reduce costs, and improve customer experience [13]. Efficiency can be improved by constant and pervasive monitoring of equipment, in order to identify faults and inefficiencies and reduce downtime. Collected data can be analyzed to extract essential information and define either short or long-term strategies to increase productivity and solve critical challenges. Reducing costs, avoiding energy waste, and increasing sustainability are all goals that IoT tries to achieve.

IoT technologies are exploited in a multitude of application fields [14]:

- **Healthcare**. Patient data is collected, stored, and analyzed in real-time to assist doctors in medical diagnosis and reduce human error. The everyday monitoring of biological signals via wearable devices has the potential for earlier diagnostic and guidance of treatment [15].
- **Smart Grids**. The status of the electrical grid is monitored, transmitted, and analyzed in real time. Smart and interconnected consumer appliances and storage devices can be used for peak-shaving and dynamic grid optimizations [16].
- **Transportation**. Data from the transportation systems is used to develop services for traffic control, traffic prediction, and emergency response. Route

optimizations can achieve a reduction in fuel and energy consumption, leading to higher sustainability. Vehicle-to-vehicle and vehicle-to-infrastructure communication increase safety and provide technological support for full-autonomous vehicles [17].

- **Smart Buildings**. Via the constant monitoring of indoor environmental parameters, such as air pollutants, noise levels, and thermal and lighting conditions, the well-being of occupants can be guaranteed [18]. The collected data can be used to model and improve the energy efficiency of buildings.
- **Smart Cities**. Data collected by smart sensors is used to optimize and extend public services. Examples are the management of car parking, street lighting, waste collection, and noise level monitoring [19]. Monitoring of outdoor air pollutants with low-cost sensors is also a common field of application.
- **Agriculture**. Common techniques in precision agriculture are intelligent soil sensing for water optimization [20] and multi-spectral imaging of crop fields for the detection and targeted intervention against insect infestations and plant diseases [21]. IoT sensors and blockchain technology can also assure product traceability and perform thermal monitoring of the supply chain process for perishable items (cold chain) [22].
- **Industry**. Industrial IoT is used for improving worker safety, increasing production uptime via predictive maintenance of machinery, helping to ensure regulatory compliance, and accelerating response times with real-time collection and processing of operational data [23].

1.2.2 Architecture

IoT systems are usually organized according to a layered architecture, which follows the flow of data and information. Even if the number of actual layers is dependent on the adopted technologies and the specific use case, it is still possible to identify a general structure. From the bottom up, the following layers are encountered:

IoT sensors The bottom layer of the architecture represents IoT sensors [24]. Typically, these are standalone devices or physical objects with added computational and

connectivity capabilities. The computational element is constituted by a microcontroller or a single-board computer, depending on the required processing power. This component is needed to manage sensors and actuators, which are used to collect data and perform actions on the physical world. The collected data can be temporally processed and stored on the device, to be finally transmitted to a cloud server or data center. Power management is also of crucial importance for IoT devices, since they cannot always rely on a connection to the power line. Either batteries or battery-less energy harvesting solutions are utilized to overcome this limitation. In the latter case, the power required for operation is extracted from the surrounding environment and used without the support of long-term and high-capacity energy storage.

Sensor networks IoT sensors are often organized in sensor networks. Connectivity is usually achieved via wireless technologies, but wired solutions are also present. The former allows for more flexible deployments, in-movement operation, and installation in remote or hard-to-reach locations. Low-power wide-area networks (LPWAN) allow long-range wireless communication of sensors. Wireless local area networks (WLANs) and personal area networks (WPANs) are used when faster speeds are preferred over a longer transmission range. Ultra short-distance communication technologies, such as RFID and NFC, are also common in IoT systems. Star and mesh network topologies are the most common. Gateway devices are usually positioned at the center of the star or positioned in one or more nodes, providing communication to the central server infrastructure [24]. The multitude of protocols and technologies will be further addressed in the following sections.

Communication layer The communication layer provides a connection between sensor networks and a centralized server infrastructure. The communication technologies used in this layer depend on the type of sensor network and application scenario. For example, in an industrial context, the communication layer can be the LAN of the factory. In smart cities, instead, cellular technologies such as 5G are very common [25]. In this case, the layer implements the radio access network (RAN), which connects the end devices (e.g., IoT gateways) and the base radio stations. Subsequently, the stations are wire-connected between them and to the core network of the operator, allowing internet access. In edge computing approaches, the communication layer can host computational resources to reduce service latency and network load [26].

Server infrastructure The data generated by IoT sensors requires a processing infrastructure to extract useful information and provide valuable services. Cloud infrastructures are the most common nowadays, where containerized applications are seamlessly and dynamically deployed over a distributed network of hardware components. The hosting and management of these infrastructures can be carried out internally by an organization (private cloud) or outsourced to an external company (public cloud). Hybrid approaches (hybrid cloud) are also possible. Cloud platforms provide services to their users with different levels of abstraction over the underlying architecture [27]. Infrastructure-as-a-service (IaaS) models only offer managed hardware and virtualization technologies. Containers-as-a-service (CaaS) provides a managed service for running a container orchestration platform and deploying containerized applications. In platform-as-a-service (PaaS) models, all the tools to develop, deploy, run, and manage apps are already included. Function-as-a-service (FaaS) allows automatic scaling and execution of user-defined code functions in response to events. Software-as-a-service (SaaS) is the highest abstraction layer, where all software applications are already installed and managed for the user.

Software services IoT software platforms provide device management, multi-tenancy support, data visualization, data storage, and event-based workflows via rule engines. Subsequently, data mining and machine learning techniques are used to extract value from the collected data. This allows to perform predictions, understand trends, identify anomalies, achieve optimizations, and define data-driven strategies.

1.2.3 Technologies and trends

Research and technological innovation is being carried out at every level of the IoT stack. This section discusses some of the most current technologies and trends in the field of IoT. Topics are selected based on their relevance with the research work that will be presented in this thesis, and according to the most popular articles in the IEEE Internet of Things Journal [28]. They are presented in a bottom-up order, following the structure of the IoT architecture introduced in Section 1.2.2.

Intermittent computing (ImC)

Intermittent computing (ImC) systems are characterized by short periods of program execution separated by reboots [29]. ImC devices are usually powered by energy harvesters (EH), which extract energy from the surrounding environment. The objective of ImC is to ensure the forward progress of the system's operation. Computation is generally performed according to the charge-operate-die cycle, which is imposed by the lack of permanent and high-capacity energy storage. Capacitors or super-capacitors are commonly used as temporary energy buffers. The objective of this approach is to make IoT systems and sensor networks truly pervasive, alleviating maintenance costs, e.g., no battery replacement is required, and enabling a deploy-and-forget approach. Removing the need for batteries further increases the sustainability of these types of systems. Challenges in this field are ensuring atomic execution, tracking time, saving peripheral state, avoiding data inconsistencies, and allowing forward progress. Solutions are searched from either the software or hardware point of view. The objective is to create an efficient mechanism for the checkpoint and restoration of the device's state, trying to minimize the additional energy overhead. Non-volatile memory (NVM) technologies are of essential support for the persistence of the device's state between reboots.

Communication protocols

IoT introduces dedicated communication protocols at each level of the OSI and TCP/IP models [24]. Adopting wireless technologies, these protocols are mainly used to create sensor networks with different topologies, geographical extensions, and communication speeds. Gateways operating in these networks interface these specialized protocols and the traditional internet stack. Nonetheless, more general-purpose technologies are still utilized for use cases without specialized requirements or constraints.

Communication protocols for wireless sensor networks can be subdivided according to the range of communication, which affects the geographical extension of the networks that can be realized:

- Personal area networks (PANs)
- Local area networks (LANs)

- Low-power wide-area networks (LPWAN)

Some of the most widespread communication protocols for PANs in IoT are:

- **Zigbee**. Zigbee is a wireless mesh network standard targeted at battery-powered devices [30]. Tree and star network topologies are also supported in addition to mesh. It is mainly used in the field of home automation and utility metering. It is based on the IEEE 802.15.4 standard and uses unlicensed ISM bands: 2.4GHz (global), 915Mhz (Americas), and 868Mhz (Europe). It achieves a maximum transfer speed of 250 kbit/s (2.4GHz, 16 channels) at the physical layer. The communication range is between 10 to 100 meters, which increases up to 1 km if sub-gigahertz frequency bands are used [31]. Zigbee also adds a network and application layer on top of IEEE 802.15.4. At the network layer, the Ad hoc On-Demand Distance Vector (AODV) routing protocol is used. Three types of Zigbee devices are present: Zigbee Coordinator (ZC), Zigbee Router (ZR), and Zigbee End Device (ZED). There is only one ZC in each network, which stores all the network's configuration and acts as a bridge to other networks. ZRs can both route data to other devices and accomplish application duties like ZEDs. ZEDs, unlike ZCs and ZRs, are generally battery-powered and operated in sleep mode most of the time to save power.

- **Thread**. Thread [32] is a networking protocol for low-power IoT systems, built for IEEE 802.15.4 wireless mesh networks operating at 250 kbps in the 2.4 GHz band [33, 34]. It uses 6LoWPAN and supports IPv6 addressing. Thread can be used to realize self-healing mesh networks of hundreds of nodes, where all devices are authenticated and communication is encrypted. To this end, UDP is used for messaging between devices, while TCP is supported for application-layer communications. Spread-spectrum techniques are also used to mitigate interference. It is designed to enable long-term operation of battery-powered devices that can sleep to reduce energy consumption. Nodes can be either Routers or End Devices. Routers keep their transceiver on at all times, being in charge of forwarding packets and enabling other nodes to join. End Devices, instead, can power off their transceiver to reduce power, don't forward traffic, and primarily interact with a single Router. In some conditions, End Devices can become Routers or vice versa. Each network has

a self-elected Thread Leader and a Border Router to forward information to non-Thread networks.

- **Bluetooth low energy (BLE)**. BLE [35] utilizes the unlicensed 2.4 GHz ISM band. The data rate at the physical layer is 1 Mb/s, and the coverage is in the tens of meters range [36]. To reduce power consumption, all devices can sleep when not transmitting data. BLE is typically used in wearable devices (e.g., health and fitness trackers) [24]. Starting from the bottom, the protocol stack is vertically divided into the controller and host portions. The former is usually implemented on the Bluetooth module SoC, while the latter is managed by the OS or bare metal firmware. Devices act as either clients or servers. Servers expose attributes organized as key-value pairs that the client can act on. In the host stack, this is implemented at the Attribute Protocol (ATT) layer. Above it, the Generic Attribute Profile (GATT) layer defines high-level data types based on attributes: characteristics, services, and descriptors. Finally, the Generic Access Profile (GAP) layer defines the roles and operational modes of BLE devices. Additional host protocols are the Security Manager Protocol (SMP) and the Adaptation Protocol (L2CAP). SMP handles security procedures (such as paring), while L2CAP acts as an intermediary between the host and controller stacks. BLE also supports mesh networking but requires a dedicated host stack.

At the LAN level, the most common protocol in IoT is Wi-Fi:

- **Wi-Fi**. The term Wi-Fi refers to a family of wireless network protocols based on the based on the IEEE 802.11 family of standards [37]. It has applications in homes, small offices, public places, and industrial environments to provide Internet access to a multitude of different types of devices. The Wi-Fi trademark is owned by the Wi-Fi Alliance [38], which controls the "Wi-Fi Certified" logo. Wi-Fi is designed for seamless connectivity with Ethernet (IEEE 802.3) [39]. Currently, supported radio bands are 2.4 gigahertz UHF, 5 gigahertz SHF, and 6 gigahertz SHF. These bands are further partitioned into multiple channels. Wi-Fi 7, which is in the latest stages of development, has a maximum channel bandwidth of 320 Mhz and supports up to 46 Gb/s of peak data rate [40]. It is designed to reduce latency and jitter and to be suitable for latency-sensitive applications. In addition, it adds multi-link operation, enhanced QoS management, and multi-access point (AP) coordination.

For what concerns LPWANs, common protocols are:

- **LoRa/LoRaWAN**. LoRa is a proprietary spread spectrum modulation technique derived from chirp spread spectrum (CSS) technology [41], currently owned by Semtech. The LoRaWAN standard, maintained by the LoRa Alliance [42], defines a link layer protocol built on top of the LoRa physical layer. LoRaWAN uses the license-free sub-gigahertz frequency bands, which are indicated in the Regional Parameters specification. LoRaWAN uses a star-of-stars network topology, where all devices communicate with a gateway connected to a central server. LoRaWAN baud rates range from 0.3 kbps to 50 kbps. Communication is always bi-directional, and end-device classes (A, B, or C) are defined to provide support for the downlink latency requirements of the targeted use case.

- **Sigfox**. Sigfox [43] is LPWAN technology that uses D-BPSK (Differential Binary Phase-Shift Keying) modulation, for which the message has a fixed bandwidth of 100Hz [44]. It uses sub-gigahertz frequency bands: 868MHz and 915MHz for Europe and the U.S. region, respectively. Transmission speeds reach 100 bps, and the range is tens of kilometers. Sigfox employs a star topology, where devices are directly connected to a base station.

- **Nb-IoT**. Narrowband Internet of Things (NB-IoT) is a cellular technology introduced in the 13th Release of the 3rd Generation Partnership Project (3GPP) [45]. It requires a minimum bandwidth of 180 kHz, providing bi-directional communication [46]. NB-IoT can be deployed either as a standalone carrier or inside the LTE spectrum. In the latter case, it is positioned in the LTE carrier or in the guard band. Data rates are around 160–200 kbps for the uplink and 160–250 kbps for the downlink [47]. For urban and suburban environments, the range is 1-8 km and 25 km, respectively.

- **LTE-M**. LTE-M was introduced for devices that have limited capabilities w.r.t. to the ones that support standard LTE [48]. The objective is to increase coverage, support a large number of connected devices, and allow for long-term battery operation, while maintaining latency below ten seconds. LTE-M carriers, having a bandwidth of up to 20 MHz, are divided into a number of narrowbands. Two categories of devices are defined: Cat. M1 and Cat. M2. Cat. M1 devices have a bandwidth of 1.4 MHz, an uplink data rate of

3 Mbps, and a downlink data rate of 1 Mbps. Cat. M2, instead, reaches 5 MHz of bandwidth, 7 Mbps uplink data rate, and 4 Mbps downlink data rate. Communication is usually half-duplex, but full-duplex is also supported.

Specific network layer and routing protocols are also available:

- **6LoWPAN**. 6LowPAN is an adaptation protocol between IPv6 and lower layer technologies, typically IEEE 802.15.4 [36]. It targets resource-constrained devices that utilize low-power and lossy links. The objective is to reduce the size of the IP datagrams and remove redundant fields. This is achieved with a compression and fragmentation mechanism.
- **RPL**. The IPv6 routing protocol for low-power and lossy networks (RPL) is often used in IoT sensor networks [36]. It supports multipoint-to-point, point-to-multipoint, and point-to-point multi-hop communications. Both global and local repair mechanisms are also defined.

Similarly, new application layer protocols are also introduced:

- **CoAP**. The Constrained Application Protocol (CoAP) is an application layer protocol for constrained networks and nodes, typical of IoT systems [49]. It is based on the REST architectural style, and it is designed to work over UDP. Its design starts from the standard web technologies: HTML, HTTP/REST, and URIs. In CoAP, HTML can be easily replaced by specialized data formats, depending on the application. For what concerns HTTP, it is too expensive in terms of both code space and network usage. Differently from HTTP, CoAP also needs to handle packet re-transmission, which is missing from UDP. This problem is solved by introducing a four-byte binary header, followed by a sequence of options, leading to a total size of 10 to 20 bytes for a typical request. CoAP preserves four request methods from HTTP: GET, PUT, POST, and DELETE. Response codes are also inherited from HTTP but are encoded on a single byte. Resources are also still identified via URIs. Finally, the realization of CoAP-HTTP intermediaries is possible without encoding specific application knowledge.
- **MQTT**. MQTT [50] is an application layer protocol based on a publish-subscribe pattern that is generally used on top of TCP. Clients can define

topics that act as communication channels for sending messages. A client can subscribe to one or more topics and receive messages that other clients publish on them. A central broker is in charge of receiving and delivering messages to all clients. Messages are also characterized by a Quality of Service (QoS) parameter, which can be set to either zero, one, or two. Higher QoS values increase the reliability of the message transmission but introduce additional overhead. Messages are delivered either at most once (QoS=0), at least once (QoS=1), or exactly once (QoS=2). The protocol can be either stateless or stateful. In the latter case, both the broker and the clients maintain session information. The broker also stores messages to be delivered to disconnected clients.

5G technology

5G is the 5th generation standard for cellular network technology, developed by 3GPP [45]. In the IoT architecture presented in Section 1.2.2, it is usually positioned at the communication layer and used to connect IoT gateways to cloud and edge services [25]. It introduces the 5G New Radio (5G NR) [51] and its radio access technology (RAT). 5G NR utilizes frequencies both below 6GHz and in the 20-100 GHz (millimeter wave). It uses Orthogonal frequency-division multiplexing (OFDM) with scalable numerology, i.e., the possibility of adjusting sub-carrier spacing (SCS). An increased number of antenna arrays is used to improve 3D beamforming and multiple-input and multiple-output (MIMO) capabilities. One of the limitations of 5G is that millimeter wave propagation tends to be mostly in line-of-sight and can be easily blocked objects.

Edge computing

The shift towards the prevalent adoption of mobile terminals and emerging Internet of Things (IoT) technologies has put a toll on mobile and wireless networks. These networks must overcome problems of low storage capacity, high energy consumption, low bandwidth, and high latency of the connected devices [26]. The traditional paradigm of a centralized cloud computing infrastructure cannot support real-time applications and handle the increasing number of connected devices. As a solution,

edge computing introduces the idea of moving computation and data storage closer to the sources of data [52].

Multi-access Edge Computing (MEC), formerly known as Mobile Edge Computing and standardized by the European Telecommunications Standards Institute (ETSI) [53], is one of the most common strategies. It involves moving the aforementioned resources within the radio access network (RAN). By adopting the MEC, mobile network operators will also become application service providers. The main characteristics of this approach are [54]:

- **On-Premise.** Services can run isolated from the rest of the network, providing greater resilience.
- **Proximity.** Services are close to the data source and may have direct access to the devices.
- **Lower latency.** Service proximity reduces latency to support real-time applications and helps in preventing traffic congestion.
- **Location awareness** Services are able to localize connected devices, opening the possibility for newer business use cases.
- **Network context information.** Real-time network data can be used to offer context-related user services.

MEC can support a variety of applications, such as smart vehicles, augmented reality, content delivery and caching, smart building control, and real-time video analysis.

Digital twins

The idea of digital twins is the creation of a virtual counterpart of physical objects. It requires the presence of three components: the physical object, the virtual twin, and a mapping between the two [55]. Its implementation also requires bi-directional communication between the two entities. The digital twin continuously adapts from the data received from the physical object and returns feedback to optimize its operations. In some cases, the design of the physical object and its virtual twin can be carried out in parallel via data engineering.

Digital Twin Networks (DTNs) are the next step in this approach. Multiple physical-virtual twin pairs are introduced. In addition to the physical-to-virtual (P2V) communication between the physical object and its virtual counterpart, physical-to-physical (P2P) and virtual-to-virtual (P2V) connections are also introduced. This results in the creation of an information-sharing network for real-time collaboration.

Key underlying technologies in this approach are cloud and edge computing, which provide the resources and infrastructure for managing virtual objects and their communication. Data processing techniques are also essential to manage the high volume of multi-source high-noise information. The first step involves handling missing values, redundancy, and conflicts. Subsequently, data fusion techniques can be used to reduce dimensionality, to find physical objects with high similarity (matching), and to integrate the experience of other physical objects (extension).

Examples of applications are the simulation of production processes, the prediction of equipment failures, the analysis of network traffic, and the creation of a virtual view of cities and transportation systems [55].

Blockchain

Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. An asset can be tangible (a house, car, cash, land) or intangible (intellectual property, patents, copyrights, branding) [56]. Smart contracts can also be stored in the blockchain, allowing the execution of automatic rules and behaviors (e.g., performing payments when agreed-upon conditions are met).

Single transactions are grouped together to form blocks. Each block contains the cryptographic hash of the previous one, hence creating a chain. According to an iterative process, each block confirms the integrity of the previous one until the start of the chain is reached. The addition of a new block to the chain is controlled by running a decentralized consensus algorithm, which prevents any central entity from controlling the chain unless it gains control of more than 50% of the peers participating in the network [57].

Examples of applications of blockchain technologies in IoT are freight transportation, component tracking and compliance, and the logging of operational maintenance data [58].

1.3 Security and reliability

Security and reliability are concepts that are orthogonal to the entire IoT architecture and technology stack. Starting from literature studies, this section presents an overview of security and reliability challenges in the field of IoT.

1.3.1 Security

Starting from the bottom of the IoT architecture, the first layer that is encountered is the hardware devices. Ensuring their security is complicated by limitations in terms of communication, computation, and storage capabilities. When considering highly pervasive systems, such as ImCs, energy becomes an additional constraint. In these scenarios, the device hardware can also be physically accessed by attackers [59]. Finally, security is often seen as an afterthought in the design of IoT devices [36]. Possible attacks can target vulnerabilities in the device's software, exploit side channel information (electromagnetic signals, timing of operations, energy consumption), add hardware trojans (modification to circuits), and tamper the device package to extract keys, modify software, and reverse engineer patented designs.

To address these problems, some MCUs and CPUs in single-board computers provide basic security features, such as secure boot, memory protection via OS-manged memory protection units (MMUs) and memory management units (MPUs), random number generators, and dedicated crypto units. However, the increasing complexity of computing platforms introduces the need for execution environments that are capable of isolating security-sensitive applications [60]. A common approach is the use of Trusted Execution Environments (TEE), which enable the creation of hardware-isolated areas where to securely run mutually distrusting applications. The objective is to ensure confidentiality and integrity of data and code. The trusted computing base (TCB) can rely entirely on hardware or a small software component. TEEs are available for all use cases, from simple IoT devices to data center applications. Commercial examples of this are ARM Trustzone [61, 62], intel SGX [63], and AMD-SEV [64].

By moving to the next layer, sensor networks and protocols are encountered. Common attacks at this level are eavesdropping of communication, injection of fraudulent traffic, and routing attacks. Examples of vulnerabilities are documented

in [36]. In Zigbee, the discovery of cryptographic keys used for secure communication is a common attack vector. Acquiring this information is possible when unsafe key-sharing mechanisms are adopted or via known plaintext attacks due to the repeated encryption of the same control messages. BLE, instead, presents vulnerabilities in some of its device pairing procedures. 6LoWPAN can be affected by attacks against the RPL protocol and border routers. Nodes in a LoRaWAN network must all share the same keys to support multicast messages. Class B devices also do not encrypt beacon messages, so they cannot distinguish between authentic and malicious ones.

Cloud computing and edge computing are the most vulnerable technologies in the communication and infrastructure layers. Cloud computing security is mostly in the hands of third-party suppliers, which have to be trusted. In addition, vulnerabilities of shared hardware and virtualization technologies can break isolation between software applications running in the same infrastructure [65]. Deployment of computing services at the edge stresses the network management policy and increases vulnerability from denial of service (DoS) attacks [26]. Physical access to edge computing servers is also less protected than in traditional data centers.

1.3.2 Reliability

Reliability is a very general term when used in such a vast field as IoT. In this work, we refer to the reliability of data and information during collection, transmission, processing, and storage.

The increasing popularity of IoT technology has been fueled by cost-effective sensing and monitoring solutions [66]. Low-cost sensors are often a miniaturization of traditional full-size devices, which exploit the same fundamental technologies [2]. This process comes with some drawbacks, such as reduced accuracy and precision, and increased possibility of faults and anomalies. This is worsened by the operating condition of pervasive devices, which are often left unattended in remote locations and subjected to the elements. Therefore, the reliability of sensor measurements becomes essential in safety-critical contexts, such as industrial environments, and when making mission-critical data-driven decisions. Current works explore better sensor designs, sometimes enabled by intelligent machine learning and computational-based

techniques [67]. The development of effective sensor calibration models is also of vital importance [68].

Reliability in data transmission, instead, is strongly correlated to the security of the transmission protocols involved. It is essential that the integrity and the authenticity of information are preserved in this phase. This is further discussed in Section 1.3.1.

Data processing is often performed at the infrastructure layer, where more computational power is available. When real-time intervention and corrections are necessary, edge computing approaches can be adopted. Some simpler strategies can also be deployed directly on IoT devices. A large amount of data is analyzed to develop calibration models, refine anomaly detection techniques, and extract information intelligence. While calibration models can be very specific to the field of application, anomalies and anomaly detection techniques can be better classified. Anomalies can be related to the monitored system, e.g., a production line, a piece of equipment, or the IoT device itself. Point anomalies are characterized by a short time duration, after which the system returns to its normal state. Contextual anomalies, instead, are defined as the variation from the expected pattern of a time series. Collective or pattern anomalies are a series of observations that are anomalous with respect to the rest of the data [66]. Information intelligence, instead, is important for the definition of business strategies and goals. To this end, reliable visualization and representation of information are also essential aspects [69].

For what concerns the storage of reliable data and information, blockchain is one of the main innovative technologies. This is due to the immutability of the stored transactions and the decentralized control of the chain. Fraud and unauthorized activity from mutually distrusting entities can be prevented, and greater transparency can be achieved.

1.4 Thesis structure

This thesis presents and analyzes two different research projects spanning the entire IoT stack. At each layer, different technologies and solutions are investigated. Security and reliability aspects remain at the center of all the presented research, and will be discussed in detail throughout this document.

More in detail, Chapter 2 presents work on particulate matter monitoring using low-cost light-scattering sensors. In this context, the reliability aspects consist in evaluating and ensuring data quality. For what concerns security, an IoT infrastructure that provides integrity verification for sensor measurements is presented. Chapter 3, instead, investigates security in intermittent computing systems. A utility for secure checkpointing is implemented and evaluated. Finally, Chapter 4 discusses the conclusions of the presented work.

Chapter 2

Low-cost particulate matter monitoring

This chapter presents the research work conducted on particulate matter (PM) monitoring with low-cost light-scattering sensors.

2.1 Introduction

Particulate matter (PM) is an air pollutant with significant effects on human health. PM is composed of microscopic particles suspended in the air that can penetrate deeply into the respiratory system and enter the bloodstream. It can be responsible for a series of health problems such as cardiovascular diseases, strokes, and lung cancer [70]. It is estimated that outdoor air pollutants, also comprising PM, were responsible for 4.2 million premature deaths in 2019 [71].

The chemical composition of PM can be varied, with major components being sulfates, nitrates, ammonia, sodium chloride, black carbon, mineral dust, and water [71]. PM is usually classified according to particle diameter. PM10, PM2.5, PM1.0, and PM0.1 consider particles with a diameter lower than 10 μm^3, 2.5 μm^3, 1 μm^3, and 0.1 μm^3, respectively. Particle size is an important characteristic since it determines where PM is able to deposit inside the human respiratory system [72, 73]. PM levels are usually expressed as number or mass concentrations. For mass concentration, $\mu g/m^3$ is generally used.

Monitoring of particulate matter is traditionally performed by environmental agencies using networks of fixed stations. Official stations are highly regulated in terms of their position, deployment density, and adopted monitoring equipment. This approach provides highly precise and accurate measurements. However, due to their size and the high cost of instrumentation, their deployments are quite sparse.

Low-cost light-scattering sensors are often presented as a possible solution to this problem. These sensors are a miniaturization of traditional optical particle counters (OPC) and nephelometers, which makes them suitable for IoT applications. Due to their reduced cost, size, and power consumption, they could enable the creation of much denser monitoring networks. In addition, they generally offer higher sampling rates than traditional instruments. PM is sampled every few seconds, whereas reference instruments may take several minutes or hours to provide a valid measurement, depending on the adopted technology.

Despite their growing adoption, they are still considered to be unreliable, inaccurate, and imprecise. One of the main reasons is that their measurement process involves several approximations and assumptions. The most notable is about PM composition, which determines the density and Refractive Index (RI) of the analyzed particles. The limited percentage of detected particles and the reduced size ranges are also known limitations of light-scattering technology, which are further accentuated by the miniaturization process. Readings of low-cost light-scattering sensors can also be greatly affected by high humidity levels. Finally, a connection to the power line is still required for continuous operation, which is mostly due to the energy consumption of the intake fan.

The main objective of this research work is to study and improve the reliability of low-cost light-scattering sensors. Long-term monitoring campaigns are conducted by placing a multitude of low-cost PM sensors near an official monitoring station in the City of Turin, Italy. The data from both the sensors and the reference station is collected and analyzed.

At first, the correlation and error between the sensors' measurements and the reference station are compared with the correlation and error between the reference station and the other stations in the official network. The objective is to understand whether the integration of low-cost sensors into the existing monitoring infrastructure can provide additional benefits and insights.

Secondly, the collected measurements are used to design a data processing pipeline comprising multiple stages: fault detection, filtering, outlier removal, and calibration. This approach is intended to increase reliability in large-scale deployments where the sensor data volume is too extensive for manual analysis.

Then, it is evaluated how a reduction of the power-on time of the light-scattering sensors can affect measurement quality. The objective of this approach is to save power and increase the lifetime of the devices. Different duty cycles are simulated by resampling the collected data, and the information loss is analyzed w.r.t. both the original signal and the official reference.

Finally, a backend infrastructure is created to support the transmission, storage, visualization and retrieval of data generated by low-cost IoT sensors. Verification of measurement integrity and authenticity is also provided via blockchain technology.

2.2 Background

This section starts by discussing the main technologies for PM monitoring. Then, it presents the area under analysis, i.e., the Metropolitan City of Turin in Italy, discussing pollution sources, seasonality trends, and its official monitoring infrastructure.

2.2.1 PM monitoring technologies

This section provides an overview of the main instrument technologies used for particulate matter monitoring [74].

Gravimetric measurements

Gravimetric instruments derive the PM concentration in an air sample by measuring the weight of its deposit on a filter. The air is drawn in via a vacuum pump and passes through specifically designed inlets. The inlets must be able to collect air samples that are representative of the monitored environment in terms of concentration and size distribution of particles [75]. Inlets can also implement cut points for standard particle sizes, such as PM10 and PM2.5 [76]. Particle size selection can also be

accomplished in multiple stages by using cyclones and virtual impactors, which exploit the inertia of the particles to achieve separation [75]. Maintaining a precise and accurate flow rate is essential in these processes.

Finally, the sample is collected on the filter. Some instruments contain multiple filter cartridges that are automatically swapped during sampling. Dirty filters are then taken to a laboratory, where they are weighted according to standard procedures. The weight difference from the initial clean filter provides the mass of the particulate, which is then divided by the analyzed air volume to compute its concentration. The collected PM sample can also be analyzed to derive its chemical composition.

The drawback of this approach is related to the cost of the instrumentation and to the related operational and maintenance expenses. In addition, gravimetric instruments cannot achieve high sampling frequencies due to the limited number of available filters.

TEOM devices

These devices use a tapered element oscillating microbalances (TEOM) to automatically and continuously weigh the PM sample [77, 78]. The filter is positioned on top of a glass tube that is maintained in constant oscillation. The increasing weight of the PM deposit is mathematically correlated to the oscillation frequency of the system, which can be measured electronically. Since no filter conditioning is possible before measurement, the air sample can be heated to remove humidity that would otherwise contribute to the measured weight. The advantages of this approach are to increase the sampling frequency and reduce manual operations.

Beta attenuation

Beta attenuation monitors use a different principle for measuring PM concentrations [79, 80]. Instead of a single filter, a filter tape is used to collect the PM samples. A low-energy radioactive source (e.g., carbon-14) generates a beta radiation that is initially shined on an empty spot on the tape. A detector, i.e., a Geiger-Muller counter, is positioned on the other side of the tape to measure radiation intensity. The same process is repeated after collecting the PM sample at the same location on the tape. The attenuation of beta radiation from an object is only dependent on its

mass, and not on other factors such as chemical composition, optical properties, and density. Consequently, the ratio between the two beta-ray counts is used to evaluate the mass of the PM deposit. As for TEOM systems, a heater is used to remove humidity from the air sample, since it would contribute to the total measured weight. Also, in this case, the advantages consist in increasing the sampling frequency and reducing manual intervention.

Light-scattering

Light-scattering technologies [81, 82] for counting particle concentrations have been around for decades and used in full-size monitoring instruments. However, in the past years, miniaturized versions of the same devices have become available for IoT applications due to their low cost and low power consumption. Two different sensor technologies are found: optical particle counters (OPCs) and nephelometers. OPCs are able to detect single particles, while nephelometers analyze the entire air sample as a whole. In both technologies, a laser beam is shined on the air sample, and the intensity of the scattered light is measured at specific viewing angles via photodetectors. Low-cost sensor manufacturers often do not disclose which technology is adopted by their devices.

OPCs OPCs are able to count single particles by detecting light pulses generated by their scattering. A calibration curve can be defined that correlates the intensity of the pulse and the diameter of the particle. This is achieved by using a calibration aerosol with known optical properties and particle size. The resulting curve also depends on specific characteristics of the monitoring device, such as the wavelength of the light source and the angle of observation. Using Mie theory, which describes the scattering properties of perfectly spherical particles of known refractive index, it is possible to analytically generate the calibration curves for a specific sensor design [81]. This can help in understanding sensor behavior and also in finding solutions that provide a monotonic relationship between the intensity of scattered light and particle diameter, avoiding measurement ambiguity.

Particles are counted by assigning them to size bins, which are defined in terms of particle diameter. The size of bins is chosen as a trade-off between the resolution and uncertainty of the assignment. Larger bins are useful, for example, when portions of the calibration curve are not monotonic: the size is chosen to include all

ambiguous measurements in the same bin. Instead, the range of diameter for which particles are detected is usually limited by costs. Detecting smaller particles requires higher-quality optics and photodetectors.

The last step in the process requires the conversion of particle counts to standard mass concentrations, e.g., PM1.0, PM2.5, and PM10. After a certain acquisition time, which can be either predefined by the manufacturer or chosen by the user, the particle count for each bin is obtained. By assuming spherical particles, the total particle mass can be derived with the following formula (from [81, 83]):

$$PM = \rho \sum_i N_i \frac{\pi}{6} D_i^3 \tag{2.1}$$

where ρ is the assumed density of particles, N_i is particle count for bin i and D_i is the geometric mean diameter for the bin [81]. Alternatively, [83] provides a formula for computing a volume-weighted diameter:

$$D_i = LB_i \left[\frac{1}{4} \left(1 + \left(\frac{UB_i}{LB_i} \right)^2 \right) \left(1 + \left(\frac{UB_i}{LB_i} \right) \right) \right]^{\frac{1}{3}} \tag{2.2}$$

where UB_i and LB_i are, respectively, the upper and lower bound of the bin. Finally, mass concentrations can be computed by dividing total particle mass by the analyzed aerosol volume.

Nephelometers Nephelometers measure the intensity of the scattered light from the entire air sample [81]. A wider range of angles is used for the measurement w.r.t. OPCs. The intensity of the scattered light is compared with a reference mass measurement to compute a calibration factor. This factor is used to convert the measured intensity to the total particle mass, which can be divided by the analyzed volume to compute standard mass concentrations.

Limitations These devices have multiple limitations, some depending on the underlying technology, some related to the low-cost implementations. The calibration of both OPCs and nephelometers is sensitive to the characteristics of the aerosol used during the calibration procedure [81]. The aerosol can be either synthetic, such

as polystyrene latex particles [84, 85], or representative of real-world sources, such as cigarette smoke [86].

If the measured particles have a different Refractive Index (RI), it can greatly affect the efficacy of calibration. In addition, the particle size distribution of the analyzed aerosol plays an important role. In OPCs this is a problem if a significant amount of particles is below the minimum detected size. For nephelometers, instead, if the particle size distribution changes, the calibration coefficient that relates scatter intensity and total particle mass must be corrected.

Low-cost OPCs only detect a small percentage of the particles in the air sample, and the actual number is derived via statistics and extrapolations [87]. The count density of large particles is also much lower than that of smaller ones. In order to acquire enough data to estimate their number, the sensors should integrated over long time intervals (hours or days). As a consequence, size fractions that include bigger particles, such as PM4 and PM10, are not measured directly but derived from PM1.0 and PM2.5 [86, 87]. Finally, when converting particle counts to particle mass, OPCs assume particles to be spherical and of known density, which is not the case in real-world scenarios.

Low-cost light-scattering sensors are negatively affected by high levels of humidity [88–91]. Particles are subjected to hygroscopic growth, meaning that their size increases due to water intake, resulting in an overestimation of particle mass. This process also affects the value of their Refractive Index (RI). To avoid this problem, high-precision devices heat the air sample before conducting the measurement [92].

Due to all this, some manufacturers declare the factory calibration precision as a device-to-device variation [87], which indicates the deviation of an individual low-cost sensor w.r.t. the average of a group of devices. This makes the calibration precision independent from an individual reference.

Low-cost devices use a small fan to draw in the air sample. The need to power these mechanical components requires a connection to the power line for high-frequency PM monitoring. In addition, the fan speed can be subjected to variations, affecting the airflow rate through the device and leading to inaccurate measurements. Some low-cost sensors are able to monitor and correct the speed dynamically to prevent this problem [84, 85]. Finally, due to the exposure to the elements, low-cost sensors are also affected by possible faults and anomalies [93].

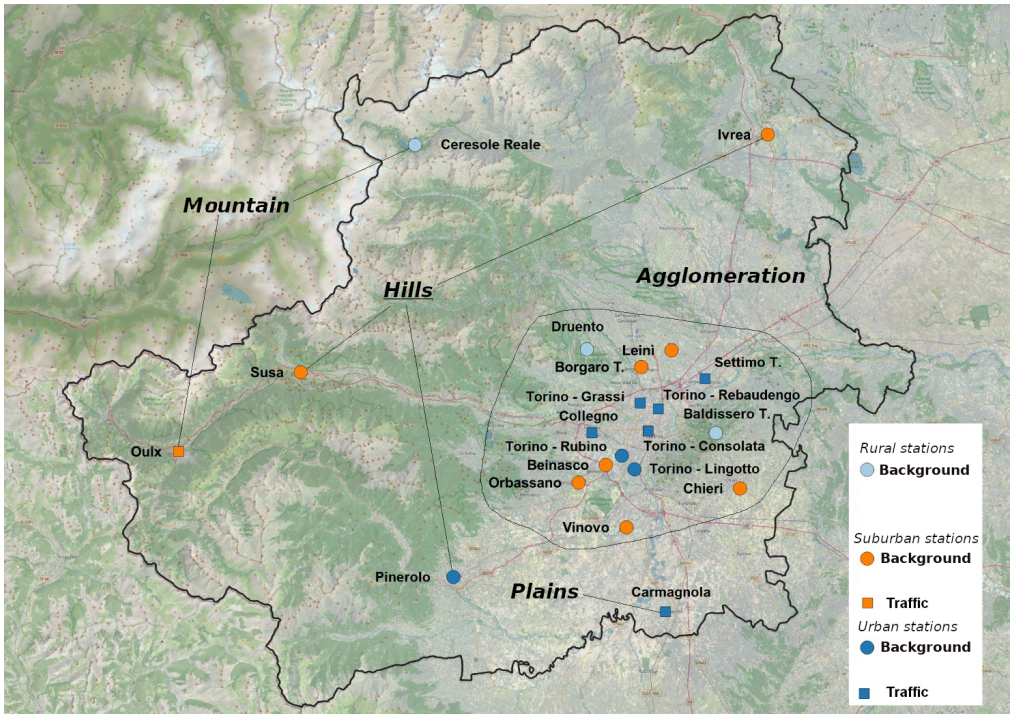


Fig. 2.1 PM monitoring network of ARPA Piemonte in the Metropolitan City of Turin [1, 2].

2.2.2 Area under consideration

The area under consideration is the Metropolitan City of Turin [94] (formerly known as the province of Turin), located in northern Italy in the Piedmont region. It is an administrative division that comprises 312 municipalities with a total area of 6827 km^2 and over 2 million inhabitants. Its capital is the City of Turin [95], which covers an area of 132 km^2 and has a population of 800.000. Being part of a metropolitan city, it borders other large municipalities on the north, east, and west sides.

Official Monitoring Network

In Italy, the monitoring of air pollution is regulated by the legislative decree 155/2010, which is an actuation of the European Directive 2008/50/EC. The directive defines objectives for data quality, location, minimum number of sampling points, and measurement methods. For what concerns particulate matter, only PM2.5 and PM10 are considered by the directive. The reference method for sampling PM10 and PM2.5 concentrations is gravimetric measurements, according to the EN12341:2014 stan-

dard. A member state can utilize other techniques only if it provides a demonstration of equivalence that must be approved by the Commission.

In the Metropolitan City of Turin, the official air pollution monitoring network [96] is managed by ARPA Piemonte [97], the regional agency for the protection of the environment of the Piedmont region. The network of stations that measure PM is shown in Figure 2.1. The stations are classified according to two parameters. The first is the level of urbanization of the area in which they are located, which can be either urban, suburban, or rural. The second is the prevalent source of pollution, which can be either background or traffic [98]. This classification is highlighted in Figure 2.1. The monitoring stations employ both gravimetric and beta attenuation instruments. The former sample PM once per day, while the latter once per hour. Measurements are accessible online [99], and air pollution reports are redacted for each year [100].

PM sources and trends

In order to discuss this topic, it is important to make a distinction between primary and secondary particulate [101]. Primary particulate is generated and directly emitted into the atmosphere by its source. Secondary particulate, instead, is generated in the atmosphere by chemical reactions from its precursors. In addition, due to transport and diffusion phenomena, PM can also be emitted or generated in a different location than the one of observation. For this reason, only considering emission sources does not provide a complete picture.

ARPA Piemonte utilizes a technique of source apportionment [102] that allows for the quantitative estimation of the contribution of the different PM sources, not in terms of their emission but in terms of the overall resulting concentration in the atmosphere. Source apportionment can be performed either via mathematical modeling or analytically. In the first case, chemical and pollution transport models are adopted [103, 104]. In the second, the chemical composition of the PM samples is studied. Unfortunately, the main limitation of this latter approach is the difficulty of assigning chemical profiles to a specific source, and therefore, it is often used as validation of the former.

Studies conducted by ARPA Piemonte [105–107] identify vehicular traffic and domestic heating as the main sources of PM10 in the Metropolitan City of Turin.

In Turin, traffic and domestic heating contribute, respectively, 38% and 49% of the total PM10 concentration (both primary and secondary) [108]. The rest is mainly attributed to industry, agriculture, and farming.

The contribution of domestic heating to primary PM10 is mostly due to the incineration of wood and pellet fuel. Sources are mostly located outside the urbanized areas, where district heating is less common. Domestic heating is also responsible for NOx (nitrogen oxide) emissions, which is a precursor of secondary particulate. In this case, contributing fuels are not only wood and pellets but also methane.

Traffic is the main source of primary PM10 inside the municipality of Turin. This is caused by exhaust emissions of vehicles, tire wear, and resuspension of particulate depositions from the road surface. If secondary particulate is considered, traffic is also the main contributor to NOx emissions in the city.

Diesel vehicles are often at the center of attention since they significantly contribute both via direct emissions of particulate and, indirectly, via the production of NOx [109]. This effect is mitigated in newer vehicles with the introduction of the latest European Emission Standards [110], which enforce strict limitations on vehicle emissions. In addition to the anti-particulate filter, technologies like Ad-Blue are used in diesel vehicles to reduce their contribution to NOx. However, lots of older diesel vehicles are still in use, for which the city imposes traffic restrictions [111].

The geographical properties of the area also influence the dispersion of PM. The Metropolitan City of Turin is surrounded by mountains on the north and west and by hills on the south and east, favoring air stagnation. In the winter period, air circulation is also prevented by the meteorological phenomenon of thermal inversion [112, 113]. Normally, temperature constantly decreases with increasing altitude. Thermal inversion, instead, consists in the formation of a layer of warmer air on top of the cold air, which prevents the dispersion of air pollutants.

Figure 2.2 represents daily averages PM2.5 measured at the station of Torino Rubino for a period of over one year. The average concentration of the pollutant starts increasing during autumn and remains high until spring. For the rest of the year, PM2.5 concentrations are lower. This trend is mainly due to the combined effect of thermal inversion and the use of domestic heating in colder months. Other meteorological phenomena also affect the presence of PM. Figure 2.2 shows how precipitation events always correspond to a reduction of PM levels. Wind is also important in helping with the dispersion of pollutants. A typical wind of the area

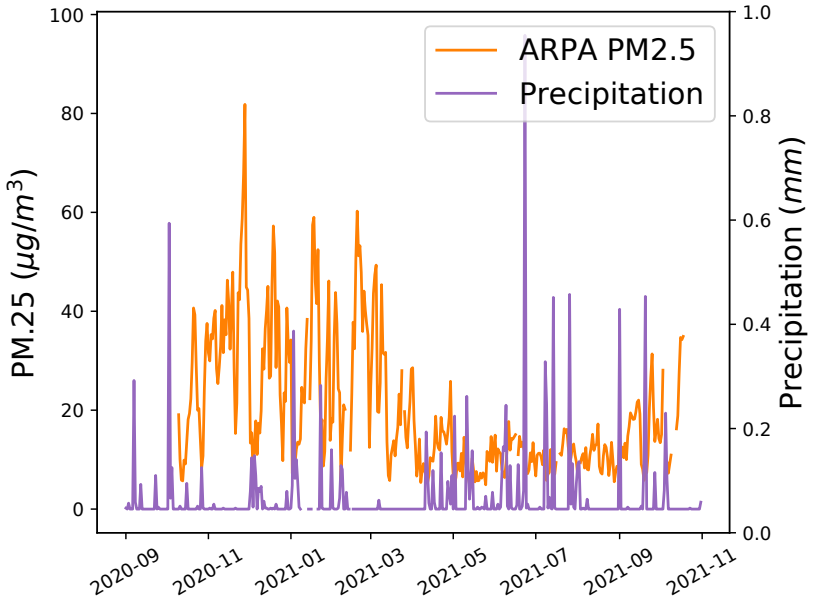


Fig. 2.2 Daily PM2.5, and precipitations, measured by the Torino-Rubino station [3].

under consideration is Fohn (or Foehn) [114–117], which also manifests during the winter period. Its effects are strong wind (speed greater than 1.5 m/s), increasing temperature, and low relative humidity (less than 40%).

2.3 Related works

Low-cost light-scattering sensors are often considered as a solution for increasing the spatial and temporal granularity of PM measurements. Due to their power efficiency and lower cost, they could be adopted for the creation of dense monitoring networks. Multiple studies are conducted in the literature about low-cost PM sensors, ranging from theoretical studies and laboratory evaluations to in-field performance assessments.

In [82], a survey is conducted about the best practices in terms of calibration models and metrics for data quality evaluation.

In [81], a physics-based approach is used to simulate low-cost light-scattering sensors to study the limitations of the technology. The analysis shows that the quality

of their measurements is affected by the optical properties of PM, such as light absorption, particle size distribution, and high levels of relative humidity.

In [118], an evaluation is conducted in a laboratory-controlled environment by placing sensors inside an acrylic glass chamber. Detection limits, linearity of response, and precision are the chosen metrics for the analysis. The influence of PM composition, particle size, relative humidity, and temperature is also considered.

Field evaluations are also common the the literature [88, 93, 119, 120, 83], where the effectiveness of low-cost light-scattering sensors is studied by comparing them with high-precision reference instruments.

In [88], a long-term field evaluation is carried out in the city of Bologna. The sensors are compared to a high-precision light-scattering monitoring device, the MetOne Profiler 212. The objective is to analyze the effects of seasonal variability, time resolutions, and meteorological conditions. The study concludes that low-cost PM sensors were mostly affected by high humidity levels and by the presence of mineral dust, while still remaining extremely informative about air quality conditions.

In [93], a long-term evaluation, lasting more than one year, is conducted by placing low-cost PM sensors from different manufacturers at multiple locations in the city of Southampton, UK. Sensors were compared to monitoring stations present in the area.

In [119], low-cost air quality monitoring stations were located at three different official monitoring sites in Santiago, Chile. The quality of PM and relative humidity measurements is evaluated. The effects of humidity on low-cost light-scattering sensors are also studied.

In [120], low-cost light-scattering sensors are compared with the TEOM 1400a gravimetric instrument. It is noticed that they overestimated PM2.5 concentrations, requiring a dedicated calibration for the specific deployment environment, instead of relying on the one provided by the manufacturer. Sensor failures were also common, so multiple units of each sensor type were deployed.

Finally, in [83], low-cost light-scattering sensors are evaluated against high-precision instruments adopting different technologies. A model for humidity correction is also introduced.

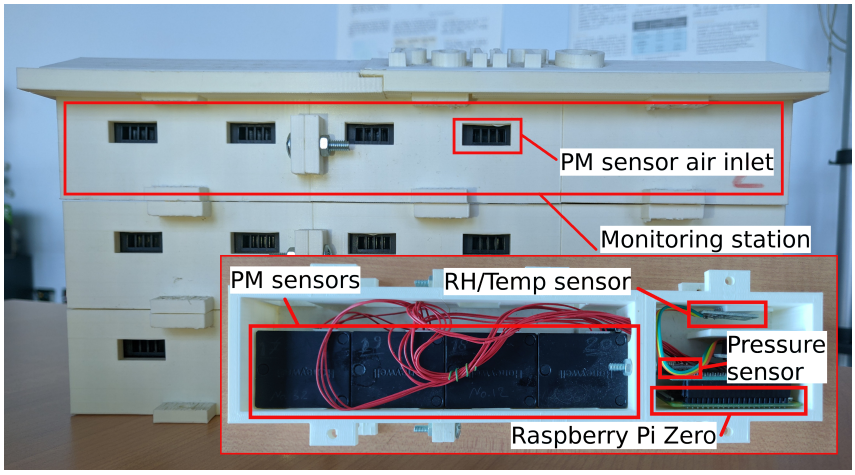


Fig. 2.3 Low-cost monitoring station (originals design) with details of components [2].

2.4 Monitoring system

Two designs for a low-cost PM monitoring station were created, both integrating low-cost light-scattering sensors. Multiple stations were built for each design and used in different monitoring campaigns. This section describes the characteristics of the stations and presents the monitoring campaigns.

2.4.1 Monitoring stations

Original design The first monitoring station design is shown in Figure 2.3. It is the same that was used in earlier work [4]. The hardware is enclosed by a 3D printed case, which is made in ABS plastic to better endure the elements. The case is designed to allow the stacking of multiple stations, that can be secured via nuts and bolts. The topmost station in the stack is covered by a roof-shaped element. Each station is equipped with the following components:

- 4 Honeywell HPMA115S0-XX low-cost light-scattering PM sensors [86].
- DHT22 temperature and relative humidity sensor [121].
- Bosch BME280 pressure sensor [122].
- DS3231 Real Time Clock (RTC) module [123].

- Raspberry Pi Zero W [124].

A small custom PCB is used to facilitate the interconnection of the different components. The Raspberry Pi board, which is flashed with an open-source Linux distribution, is in charge of managing all the installed hardware. It uses the RTC module to retrieve the correct time after a possible power loss.

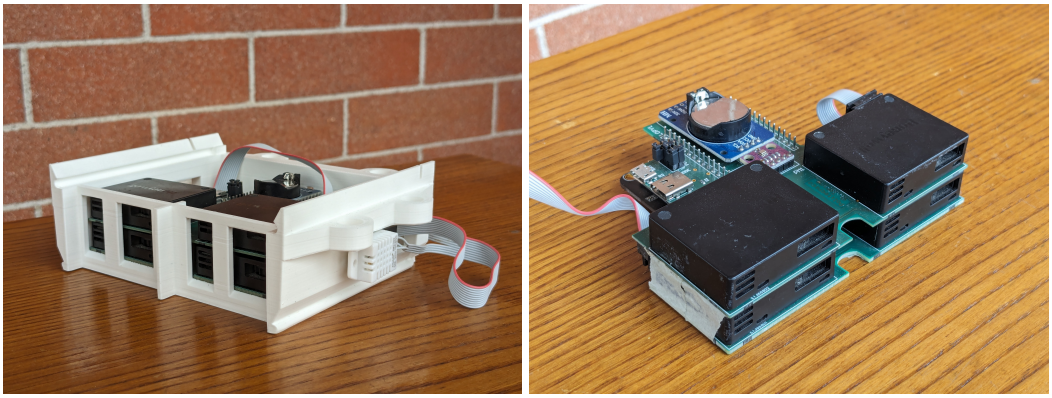
The components are divided into two separate enclosed areas. The biggest contains the four PM sensors and provides openings on both sides for their inlets and outlets. It was decided to use four PM sensors to have redundancy in the system in case of sensor failure and to detect measurement anomalies. The smallest enclosed area, instead, contains all the remaining components. It also provides an opening for the temperature and relative humidity sensor.

The system stores the acquired measurements on the MicroSD card of the Raspberry Pi. A new measurement file is created every day for each sensor. A dedicated software program acquires data from the sensors and stores them in the appropriate file. The program is configured to run continuously and to restart in case of crashes. At certain time intervals, which can be configured, the board tries to connect to a predefined Wi-Fi network. If the connection is successful, it synchronizes the RTC via the remote Network Time Protocol (NTP) [125]. Subsequently, it uses the Unix *rsynch* [126] utility to upload the new measurement files to a remote server. The connection with the remote server is performed via the SSH protocol [127], which also provides integrity and confidentiality to the transmitted information.

For what concerns the HPMA115S0-XX PM sensor, it provides measurements for both PM10 and PM2.5. However, only PM2.5 concentrations are measured directly, while PM10 is estimated from PM2.5 with a proprietary algorithm. The manufacturer also does not specify whether the sensor acts as an OPC or a nephelometer. The sensor's datasheet provides the following information about the accuracy of PM2.5 readings:

- $\pm$ 15 μg/m^3 from 0 μg/m^3 to 100 μg/m^3.
- $\pm$ 15% from 100 μg/m^3 to 1000 μg/m^3.

Unfortunately, these accuracy values are only valid at a temperature of 25°C $\pm$ 5°C. Also, operating conditions require that relative humidity stays between 0%



(a) Exterior. (b) Internal hardware.

Fig. 2.4 Low-cost monitoring station (updated design) with details of components.

and 95% (non-condensing). However, the outdoor conditions in which the stations are meant to be used are fairly different. In Turin, the temperature can go below 0°C in winter and over 30°C during summer. Heavy rains and condensing humidity conditions are also common. Finally, the aerosol used for factory calibration is cigarette smoke, which has characteristics that are different from those of the air in an urban environment.

Updated design The second design for the low-cost monitoring station is shown in Figure 2.4. The case, shown in Figure 2.4a, is 3D printed using ABS plastic to better endure long-term outdoor deployments. It is possible to stack multiple stations by sliding them on top of each other. The case also provides some lateral mounting holes to secure the stations together. The topmost station in a stack can be covered with a flat plastic panel. The case presents one main enclosed area where the internal hardware, shown in Figure 2.4b, can be positioned. Each station is equipped with the following components, which are connected to a custom PCB:

- 4 Honeywell HPMA115C0-003 low-cost light-scattering PM sensors [86].
- DHT22 temperature and relative humidity sensor [121].
- Bosch BMP280 pressure sensor [128].
- PA1010D GPS unit [129].
- DS3231 Real Time Clock (RTC) module [123].

- Pycom LoPy4 [130] or FiPy [131] development modules.
- Pycom Expansion Board 3 [132].

The system can be managed by either the Pycom LoPy4 or the FiPy development modules. They are both based on the ESP32 microcontroller and integrate Wi-Fi (802.11 b/g/n at 2.4 GHz), Bluetooth (v4.2, BLE), SigFox, and LoRa. The FiPy module also supports mobile connectivity through LTE Cat. M1 and NB-IoT networks, providing a slot for a nano SIM. The modules are connected to an expansion board that provides GPIO ports, LEDs, buttons, a USB port, and a Micro-SD card slot.

As in the previous design, four low-cost light-scattering sensors are used. In this case, however, the compact surface-mounted version of the sensor is chosen. This newer model has both inlet and outlet on the same side, which facilitated the design of the enclosure. In addition to PM2.5 and PM10, the sensor can also measure PM1.0 and PM4.0. As for PM10, these values are only estimated from PM2.5 with a proprietary algorithm. The manufacturer also does not specify whether the sensor acts as an OPC or a nephelometer. The declared accuracy for PM2.5 is the same as the full-size model:

- $\pm$ 15 μg/m^3 from 0 μg/m^3 to 100 μg/m^3.
- $\pm$ 15% from 100 μg/m^3 to 1000 μg/m^3.

For the compact model, the datasheet also reports the accuracy for PM1.0, PM4.0, and PM10:

- $\pm$ 25 μg/m^3 from 0 μg/m^3 to 100 μg/m^3.
- $\pm$ 25% from 100 μg/m^3 to 1000 μg/m^3.

All these values are still measured at 25°C $\pm$ 5°C. Operating conditions still require that relative humidity stays between 0% and 95% (non-condensing). Finally, it is important to mention that PM1.0 and PM4.0 measurement collection was disabled, since high-precision reference instruments were never available for these size fractions.

The DTH22 sensor and the GPS unit are both installed outside the case. The reason for this decision is to prevent erroneous temperature readings due to the heat coming from the hardware components, and to avoid interference of the GPS signal when stacking multiple stations.

The FiPy and Lopy4 development boards run a customized version of Micropython [133]. Micropython is an implementation of the Python 3.x programming language, which is optimized for resource-constrained microcontrollers. Therefore, the device runs a Python interpreter, and the firmware can be programmed just by uploading Python scripts.

The software developed for this project defines a separate interrupt for each sensor, in order to allow for different sampling times. Interrupts are queued and served according to the order of arrival, and, for this reason, interrupt routines must be kept concise. The routines are in charge of retrieving the measurements from the sensors to proceed with further processing. Sensor measurements can either be stored on the MicroSD or transmitted via the different communication technologies available on the device. In the context of this work, the devices are always connected to a Wi-Fi network and transmit data to a central MQTT broker.

2.4.2 Monitoring campaigns

Multiple monitoring campaigns were conducted to collect real-world data with the low-cost light-scattering PM sensors. Most of the work presented in this thesis is focused on the second monitoring campaign and the crossroad deployment. However, the other campaigns are still discussed for completeness.

First campaign

The first data acquisition campaign is the one described in earlier work [4]. The monitoring system was composed of twelve low-cost monitoring stations that followed the original design, as described in Section 2.4.1. The assembled system is shown in Figure 2.6.

The low-cost system was positioned on top of the official ARPA monitoring station of Torino Rubino, shown in Figure 2.5, near the inlets of official gravimetric and beta attenuation instruments. The station of T. Rubino, located in a public park,

Fig. 2.5 ARPA monitoring station of Torino Rubino [4].

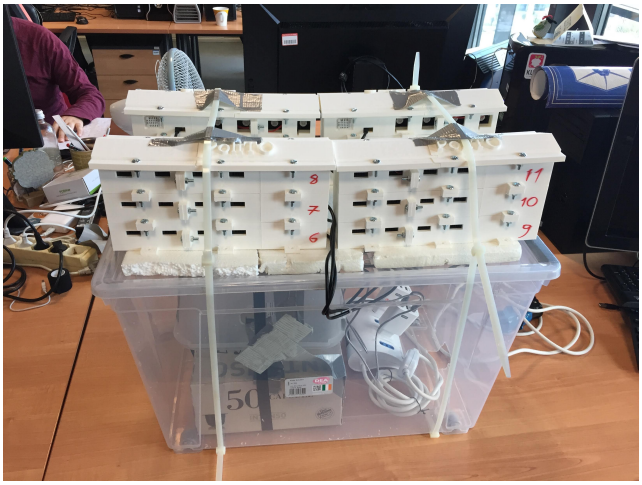


Fig. 2.6 Assembled monitoring system (first design).

is classified as a background urban station. Its position inside the Metropolitan City of Turin is reported in Figure 2.1.

The experiment lasted for a period of 5 months, from October 2018 to February 2019. The low-cost light-scattering sensors were configured to sample PM2.5 and PM10 every second, while temperature, humidity, and pressure measurements were acquired every 3-4 seconds. The official data of the reference instruments was also collected for the same period. The beta attenuation monitor samples PM2.5 and PM10 every hour, while the gravimetric instrument only provides one measurement per day. The dataset containing the collected data can be found at [134].

Second campaign

The second campaign was conducted with the same setup as the first one, but fourteen low-cost stations were deployed instead of just twelve. The location of the experiment was still the T. Rubino station. This campaign lasted for more than one year, from October 10, 2020, to November 1, 2021. This allowed for the study of the performance of the low-cost sensors by also taking into account the seasonality of PM concentrations. The sampling frequency of both the low-cost and the official sensors was the same as in the previous campaign. The dataset containing the collected data can be found at [135].

Third campaign

The third campaign was again performed at T. Rubino, this time using the newer design of the low-cost stations. Six stations were deployed for a period of seven months, from November 2022 to May 2023. PM2.5 and PM10 were sampled every second, while temperature, humidity, and atmospheric pressure every 2-5 seconds. The sampling time of the official instruments was unchanged: one hour for the beta-attenuation monitor and one day for the gravimetric device. The dataset[1] of the collected measurements can be found at [136].

[1]Currently, only an aggregated version of the dataset has been published.

(a) Integration test. (b) Crossroad installation.

Fig. 2.7 Setup of crossroad experiment [5].

Crossroad experiment

The last monitoring campaign consisted in deploying two low-cost monitoring stations (newer design) near two high-traffic areas in the City of Turin. The stations were mounted on poles installed at the crossroads between Via Ventimiglia and two other streets: Corso Spezia and Corso Maroncelli. They are integrated with a system developed by Fondazione Links [137] for a project on autonomous driving. This is done to provide power and internet access to the low-cost PM monitoring system. Each station connects to a local Wi-Fi network to access a 5G gateway.

The experiment started in June 2023 and is still in progress. The sampling time of the different sensors was configured as in the third campaign. However, the GPS was also activated in this scenario. It provides latitude, longitude, and altitude information every 60 seconds. The current objective of this measurement campaign is to ensure the correct functionality of the system, together with the backend infrastructure, which will be discussed in the following section.

2.5 Data quality evaluation

This analysis evaluates the performance of low-cost PM sensors w.r.t. to official high-precision instruments. The objective is to understand the benefits they can provide if integrated into a preexisting official monitoring network. The study is carried out on the data collected during the second monitoring campaign since an entire year of measurements is available. Results of this work are published in [2].

2.5.1 Dataset

The dataset for this experiment is created starting from the data collected during the second campaign, presented in Section 2.4.2, where fourteen low-cost monitoring stations were positioned a official station T. Rubino for over one year. After an initial data exploration phase, only the measurements collected from November 1, 2020, to November 1, 2021, are included, to consider a period of exactly 12 months. For this analysis, only the PM2.5 measurements of the 56 low-cost sensors are considered, since the concentration of the other size fractions is not measured directly but only estimated from PM2.5.

The data of all the other ARPA stations in the Metropolitan City of Turin was also collected for the same period. It is composed of daily PM2.5 and PM10 measurements of the gravimetric and beta instruments, with the exception of T. Rubino, where the beta sensor also provides hour concentrations. In order to allow for comparisons, hour and day aggregations of PM2.5 measurements are computed for each low-cost sensor.

2.5.2 Data exploration

A preliminary analysis is conducted by analyzing the behavior of the low-cost sensors during the measurement campaign.

Faults and anomalies

The data of the 56 low-cost PM sensors is manually inspected to detect possible problems during device operation. The identified faults and anomalies are summarized

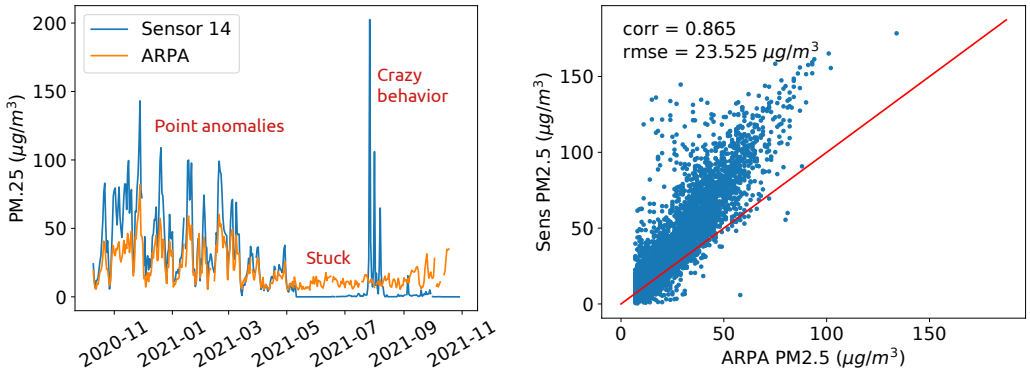


Fig. 2.8 Faults and anomalies of sensors 14 (PM2.5), compared to the beta reference of T. Rubino on day averages (left). Point anomalies of sensors 14 (PM2.5) with permanent faults removed, compared to the beta reference of T. Rubino on hour averages (right) [3].

in Figure 2.8. A distinction can be made between permanent faults and temporary anomalies. Two main types of permanent faults are observed. The first type is devices getting stuck in the lower range of the measurement scale (0 or 1 $\mu g/m^3$). The second type is non-deterministic behavior, which is characterized by extremely noisy or completely incorrect measurements. For what concerns temporary anomalies, instead, they are usually point anomalies. In this case, the sensor provides incorrect readings for a short period of time.

Since no issues were encountered during data logging or transmission, the reason for permanent faults can be attributed to malfunctions in the optical components of the sensor. For what concerns point anomalies, instead, they are usually characterized by high-frequency changes, often impulsive, that are maintained for several minutes. These can be attributed to elements getting stuck inside the sensor for a period of time of random duration.

In conclusion, this analysis shows that ten sensors (18%) failed at the beginning of the experiment, 33 sensors (59%) failed during the experiment, and 13 (23%) worked correctly throughout the entire campaign.

Seasonality

Figure 2.9 depicts the seasonality of the low-cost sensors' behavior. Pearson correlation and relative error between the median of all working low-cost sensors (PM2.5)

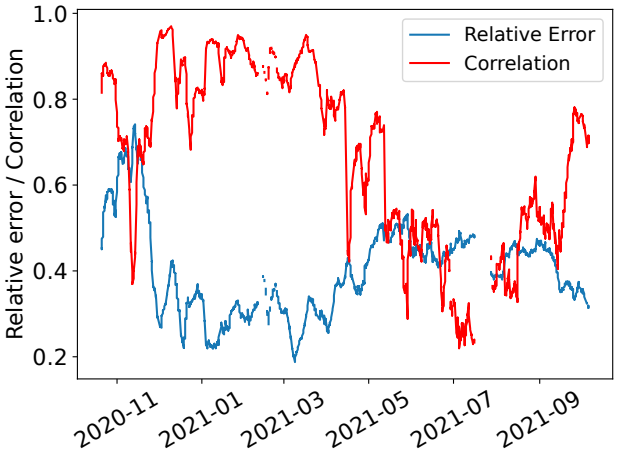


Fig. 2.9 Correlation (Pearson) and relative error between the median of all working low-cost sensors (PM2.5) and the reference beta instrument of T. Rubino. Evaluation on day averages, using a fourteen-day sliding window.

and the reference beta instrument of T. Rubino is computed throughout the year. The evaluation is performed on day averages, considering a fourteen-day sliding window to compute the two metrics. As can be seen, the correlation is higher during the winter period and becomes lower during summer. The relative error, instead, has the opposite behavior. This can be explained by the seasonality of PM2.5 concentrations in the City of Turin (Figure 2.2), which are higher in winter and decreases in warmer months. According to the sensor's datasheet, the measurement accuracy is $\pm$ 15 μg/m^3 for PM2.5 concentrations below 100 μg/m^3. During summer, the measurement error of the sensors becomes comparable with the values being measured, as depicted in the curve or relative error. Therefore, the low-cost sensors are not able to provide insightful information about changes in PM levels, leading to a lower correlation with the reference.

2.5.3 Preprocessing

In order to evaluate the performance of the low-cost light-scattering PM sensors, the data related to extensive sensor malfunction is removed. This is because, in a real-world scenario, faulty sensors would be identified and replaced, and therefore, they would not contribute to defining the data quality of the monitoring system.

The median of all the low-cost sensors is computed for the entire measurement period. For each sensor, its Pearson correlation with the sensors' median is calculated. If this value is lower than 0.8, the sensor is discarded. At the end of this step, 22 sensors are removed from the analysis.

Some of the remaining 34 sensors got stuck before the end of the monitoring campaign, providing readings close to zero. Starting from the end of each sensor's time series, values are removed until the first measurement above the threshold of 2 $\mu g/m^3$ is encountered. At this point, the median of all the remaining low-cost PM sensors is computed again to generate a single time series representative of the entire low-cost monitoring system.

2.5.4 Methodology

The objective of this analysis is to assess the benefits of integrating low-cost light-scattering PM sensors in an official monitoring network composed of high-precision reference instruments. Multiple analyses are carried out, starting from the comparison of the official instrument inside the Metropolitan City of Turin, considering devices positioned at different locations and measuring different size fractions. Then, the low-cost PM2.5 measurements are compared with ones of the official instruments, also for different size fractions, to understand in which scenario light-scattering devices can provide additional information.

The metric that is chosen to compare two different monitoring instruments is the Pearson correlation coefficient, evaluated on the whole year for daily measurements. This is chosen because the compared instruments can be located at different locations and can measure different size fractions. Since they are effectively monitoring different quantities, it is not meaningful to estimate the error between two devices. Instead, the correlation coefficient provides information on the capability of one instrument to estimate the values of the other. This is useful for cases in which the second instrument is not present. In the last step of the analysis, Root Mean Squared Error (RMSE) is also used to evaluate the error between two devices when calibrating one device to reproduce the measurements of the other. This error is computed on the whole year and on daily averages, using a 30-day sliding window with a one-day shift to select the calibration period. The calibration model of choice is simple linear regression.

The following comparisons are conducted:

1. Reference instruments at the same monitoring site:
    - Correlation between gravimetric and beta attenuation devices.
    - Correlation between PM2.5 and PM10.
2. Reference instruments at different urban sites (City of Turin):
    - Correlation of PM2.5 measurements.
    - Correlation of PM10 measurements.
    - Correlation between PM2.5 and PM10.
3. T. Rubino and stations outside the City of Turin (reference instruments):
    - Correlation of PM2.5 measurements.
    - Correlation between PM2.5 and PM10.
4. Low-cost sensors and beta attenuation at T. Rubino.

Then, the results of all these evaluations are compared together to provide a comprehensive picture of different findings. The RMSE analysis is conducted by calibrating the PM2.5 reference instruments inside the City of Turin and the low-cost sensors to target the reference instruments of T. Rubino. Finally, the precision and accuracy of the low-cost sensors are evaluated against the T. Rubino reference across the range of measured PM2.5 concentrations.

2.5.5 Analysis

Reference instruments at the same monitoring site

This section presents comparisons between reference instruments located at the same monitoring site.

Table 2.1 Correlation between gravimetric and beta attenuation instruments at the same monitoring station [2].

Station	PM size fraction	Correlation
T. Rubino	10	0.984
T. Lingotto	10	0.987
T. Lingotto	2.5	0.989
Borgaro	10	0.990
Chieri	2.5	0.993
Settimo	10	0.990

Correlation between gravimetric and beta attenuation devices. Table 2.1 shows the correlation between gravimetric and beta instruments located at the same station, measuring the same PM size fractions. As it is possible to see, only a few stations in the Metropolitan City of Turin are equipped with both instrument technologies. Since high-precision devices measuring the same quantities are compared, it is expected that they provide extremely correlated measurements. This is confirmed by the results, which show correlations between 0.984 and 0.993 with an average of 0.989. These values will be considered as a benchmark for the rest of the analysis.

Correlation between PM2.5 and PM10. Table 2.2 shows the correlation between PM2.5 and PM10 reference instruments located at the same monitoring site. This analysis can be useful in understanding whether it is possible to estimate PM2.5 from PM10 or vice versa, when only one size fraction is measured. As can be seen, there are seven monitoring stations in the Metropolitan City of Turin providing measurements for both size fractions. The first three stations that are listed, i.e., T. Rubino, T. Lingotto, and T. Rebaudengo, are urban stations inside the City of Turin. The others belong to different municipalities in the Metropolitan City of Turin. Settimo is an urban station, Chieri, Ivrea, and Boargaro are suburban stations, and Ceresole is a rural station. For what concerns the results, most stations maintain a high correlation between PM2.5 and PM10 measurements, even if lower than the benchmark defined in the previous paragraph. The only outlier is Ceresole, for which the correlation between the two size fractions is much lower. This can be explained by the fact that this is a rural station residing in a natural park in the mountains, where atmospheric conditions can be significantly different.

Table 2.2 Correlation between PM2.5 and PM10 measurements collected at the same monitoring site [2].

Station	PM2.5 Sensor	PM10 Sensor	
		Gravimetric	β
T. Rubino	β	0.943	0.948
T. Lingotto	Gravimetric	0.986	0.972
T. Lingotto	β	0.960	0.935
T. Rebaudengo	β		0.939
Borgaro	Gravimetric	0.981	
Borgaro	β	0.936	0.937
Ceresole	β		0.875
Chieri	β		0.951
Chieri	Gravimetric		0.987
Ivrea	Gravimetric	0.986	0.949
Settimo	β	0.947	0.936
Settimo	Gravimetric	0.975	

Reference instruments at different urban sites (City of Turin)

In this section, the correlations between instruments located at different urban sites inside the municipality of Turin are evaluated. The stations considered are T. Rubino, T. Lingotto, T. Consolata, T. Grassi, and T. Rebaudengo. The first two are background stations, while the others are traffic ones.

Correlation of PM2.5 measurements. The result of this evaluation is shown in Table 2.3. The correlation between each pair of stations is computed. Comparisons of instruments located in the same station are excluded, since they were already discussed previously (see Table 2.1). Correlation is between 0.955 and 0.969 with an average of 0.961, which is comparable with the results presented in Table 2.2 for different size concentrations measured at the same site. Table 2.4 presents the average PM2.5 concentration measured by the same instruments over the entire experiment campaign. Despite the good correlation, there are significant differences in the average concentrations between the monitoring sites, which are accentuated when comparing traffic and background stations. In fact, the traffic station of T. Rebaudengo measured an average PM2.5 concentration 3.5 $\mu g/m^3$ higher than the other background sites.

Table 2.3 Correlation between measurements of PM2.5 of urban monitoring stations inside the City of Turin [2].

		T. Lingotto (β)	T. Rebaudengo (β)
	T. Lingotto (Gravimetric)		0.967
T. Rubino (β)	0.956	0.969	0.957
			0.955

Table 2.4 Yearly average PM2.5 concentration measured by urban monitoring stations inside the City of Turin [2].

Station	Average PM2.5 (μg/m^3)
T. Rebaudengo (β)	23.3
T. Lingotto (Gravimetric)	19.3
T. Lingotto (β)	18.3
T. Rubino (β)	19.8

Correlation of PM10 measurements. A similar analysis is also conducted for PM10 measurements. The resulting correlations, shown in Table 2.5, are between 0.945 and 0.981, with an average of 0.964. The range of correlations is larger than for PM2.5, mainly due to the presence of more stations monitoring PM10. The average correlation, however, is very similar.

Correlation between PM2.5 and PM10. Table 2.6 shows the correlation between measurements of PM2.5 and PM10 collected at different urban sites in the City of Turin. The correlation is between 0.913 and 0.959, with an average of 0.933, which is lower than what is found when comparing the same size fractions.

Table 2.5 Correlation between measurements of PM10 of urban monitoring stations inside the City of Turin [2].

					T. Rebaudengo (β)
			T. Lingotto	**T. Lindotto** (β)	0.957
		T. Grassi	(Grav.)		0.954
	T. Consolata	(Grav.)	0.945	0.949	0.959
	(Gravi.)	0.970	0.979	0.979	0.965
T. Rubino (β)	0.976	0.946	0.976	0.977	0.949
T. Rubino (Grav.)	0.977	0.946	0.979	0.981	0.950

Table 2.6 Correlation between measurements of PM2.5 and PM10 collected at different urban sites inside the City of Turin [2].

PM2.5 Stations	**PM10 Stations**						
	T. Rubino (Grav.)	T. Rubino (β)	T. Consolata (Grav.)	T. Grassi (Grav.)	T. Lingotto (Grav.)	T. Lingotto (β)	T. Rebaudengo (β)
T. Rubino (β)			0.944	0.943	0.952	0.935	0.919
T. Lingotto (β)	0.921	0.913	0.930	0.920			0.914
T Lingotto (Grav.)	0.959	0.955	0.959	0.926			0.937
T. Rebaudengo (β)	0.914	0.917	0.936	0.936	0.943	0.925	

T. Rubino and stations outside the City of Turin (reference instruments)

In this section, the correlation between T. Rubino and other official monitoring stations outside the City of Turin is evaluated. Due to the large number of monitoring stations, only the PM2.5 beta attenuation instrument is considered for T. Rubino. First, the comparison is performed on PM2.5 concentrations. Then, PM2.5 measurements at T. Rubino are correlated with PM10 measurements of the other stations.

Correlation of PM2.5 measurements Table 2.7 shows the results of the comparison between the PM2.5 concentrations measured by the reference devices. The correlation is between 0.032 and 0.968, with an average of 0.846. Without considering the outlier represented by the rural station of Ceresole, which is situated in a fairly different environment, the lowest correlation and the average would be, respectively, 0.886 and 0.937. In any case, results are worse than what is obtained in the comparison between PM2.5 measurements inside the City of Turin (shown in Table 2.3): variance is higher, and the average correlation is lower.

Correlation between PM2.5 and PM10 The correlation between PM2.5 measured at T. Rubino and PM10 measured in monitoring sites outside the municipality of Turin is reported in Table 2.8. The correlation is between 0.095 and 0.941, with an average of 0.769. The less correlated station is again Ceresole, whose PM10 measurements appear to be completely unrelated to PM2.5 concentrations at T. Rubino. Results are significantly worse than the ones obtained by correlating PM2.5 and PM10 inside the City of Turin (Table 2.6). Similarly, correlations are also worse

Table 2.7 Correlation of PM2.5 measurements between T. Rubino and monitoring sites outside the municipality of Turin [2].

Station	Sensor	Correlation
Baldissero	β	0.968
Borgaro	Gravimetric	0.886
Borgaro	β	0.960
Ceresole	β	0.032
Chieri	Gravimetric	0.910
Chieri	β	0.949
Ivrea	Gravimetric	0.925
Leini	β	0.948
Settimo	Gravimetric	0.941
Settimo	β	0.944

than in the previous experiment, which compares PM2.5 measurements of T. Rubino to PM2.5 measurements of stations outside Turin (Table 2.7).

Low-cost sensors and beta attenuation at T. Rubino

In this section, the low-cost light-scattering sensors positioned at T. Rubino are compared with the beta attenuation instrument installed at the station. Table 2.9 presents the results. The sensors listed in the table are the ones that worked for the entire measurement campaign. The median of the low-cost sensors, instead, is computed considering the 34 sensors that are selected during the data preprocessing phase described in Section 2.5.3. Since hour measurements are available from the beta instrument of T. Rubino, correlations are computed using both day and hour averages.

A strong correlation (higher than 0.9) is achieved on daily averages between the light-scattering sensors and the reference. Instead, when considering hour averages, the correlation drops below 0.9. However, it is interesting to notice that the correlation of the eight light-scattering sensors with the median is not affected by the change in sampling rate. Therefore, sensors still agree with each other on hour averages, but not with the reference. This seems to indicate that the reduced correlation for hour averages is a technological limitation of the sensors, and not due to imprecisions of single devices. This conclusion can also be reached by observing the reduction in

Table 2.8 Correlation between PM2.5 measured at T. Rubino and PM10 measured at monitoring sites outside the municipality of Turin [2].

Station	Sensor	Correlation
Baldissero	β	0.793
Beinasco	β	0.941
Borgaro	Gravimetric	0.916
Borgaro	β	0.908
Carmagnola	Gravimetric	0.899
Ceresole	β	0.095
Chieri	β	0.925
Collegno	Gravimetric	0.922
Druento	Gravimetric	0.692
Ivrea	Gravimetric	0.834
Ivrea	β	0.909
Leini	β	0.927
Oulx	Gravimetric	0.168
Pinerolo	β	0.783
Settimo	Gravimetric	0.931
Settimo	β	0.919
Susa	Gravimetric	0.505

Table 2.9 Correlation of between PM2.5 light-scattering sensors that worked for the entire experiment and the T. Rubino reference [2].

Sensor ID	Day		Hour	
	T. Rubino	Median	T. Rubino	Median
25	0.954	0.997	0.881	0.995
29	0.920	0.935	0.824	0.920
145	0.890	0.885	0.811	0.882
147	0.932	0.975	0.854	0.959
156	0.953	0.987	0.871	0.975
158	0.939	0.983	0.858	0.973
167	0.914	0.979	0.854	0.976
211	0.916	0.975	0.843	0.967
Median	0.952	1.000	0.881	1.000

correlation between the median and the reference when moving from day to hour averages.

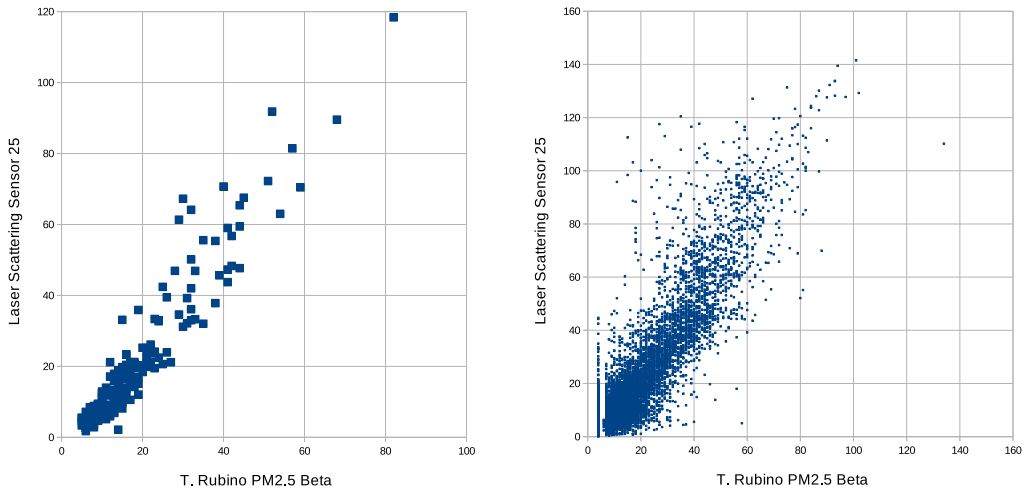
Scatter plots of light-scattering sensors over the T. Rubino beta instrument are shown in Figure 2.10 for both day and hour averages.

Summary

A summary of all the different analyses is reported in Figure 2.11 and in tabular form in Table 2.10.

As expected, the highest correlation is between instruments of different technologies (beta and gravimetric), measuring the same PM size fraction at the same monitoring site. For what concerns the correlation between PM10 and PM2.5 at the same site, the variance is much greater between the different stations. This is probably due to local phenomena and characteristics of the monitored location that caused the ratio between the two size fractions to change during the monitoring campaign. Therefore, even if, in some cases, a strong correlation is found, using PM2.5 to estimate PM10 or vice versa can result in wrong assessments.

The correlation between instruments positioned in the same urban area, the City of Turin, and measuring the same size fraction is very high. Nonetheless, it is lower than the benchmark correlation given by reference instruments at the same location. For PM10 there is more variance in the results, but this is due to the higher number of



(a) Day averages.

(b) Hour averages.

Fig. 2.10 Scatter plot of PM2.5 light-scattering sensor 25 over PM2.5 T. Rubino (μg/m^3) [2].

instruments measuring this size fraction. When correlating PM10 and PM2.5 inside the City of Turin, good results are still found, even if they are lower than correlations between same size fractions. Instead, comparing sensors outside the City of Turin with the station of T. Rubino leads to lower correlations and higher variance in the results.

For what concerns low-cost light-scattering sensors, they achieve a good level of correlation with the station of T. Rubino. However, they are below the benchmark value of correlation between gravimetric and beta attenuation instruments in the same location. Their performance is similar to the one of instruments located in the same urban environment, also considering correlations between different size fractions. Therefore, if other high-precision instruments are present in the same urban environment, their usefulness is reduced when measuring daily averages. Nonetheless, their contribution could be essential in places where official instruments are missing or very sparse. Finally, low-cost light-scattering sensors enable instantaneous sampling, which is missing from most official monitoring sites. However, as shown for hour averages, their correlation with the reference can decrease when increasing the sampling frequency. Despite this, they could still provide interesting insights on short-lived emission phenomena.

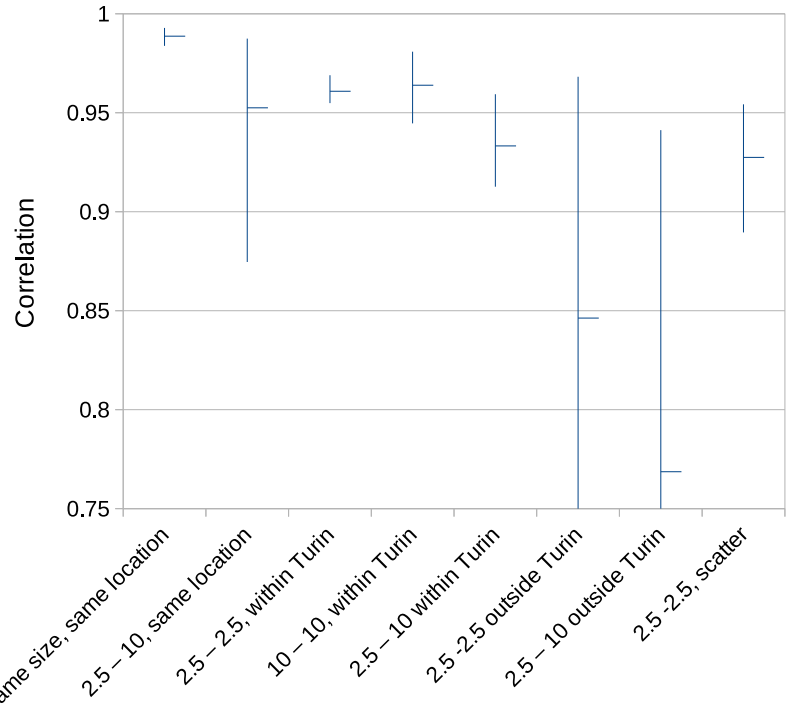


Fig. 2.11 Mean, max, and average correlations for each analysis [2].

Table 2.10 Mean, max, and average correlations for each analysis [2].

Size and Location	Mean	Max	Min
same size, same location	0.989	0.993	0.984
2.5 – 10, same location	0.952	0.987	0.875
2.5 – 2.5, within Turin	0.961	0.969	0.955
10 – 10, within Turin	0.964	0.981	0.945
2.5 – 10 within Turin	0.933	0.959	0.913
2.5 -2.5 outside Turin	0.846	0.968	0.032
2.5 – 10 outside Turin	0.769	0.941	0.095
2.5 -2.5, scatter	0.927	0.954	0.890

RMSE analysis

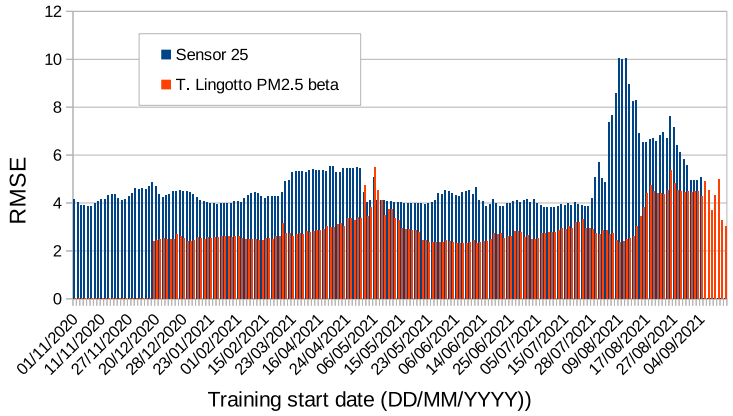
In this section, the best performing low-cost PM sensor (sensor 25) and the official instruments inside the City of Turin that measure PM2.5 are calibrated to target the values produced by the PM2.5 beta attenuation device of T. Rubino. The capability of the different calibrated instruments to predict the beta sensor at T. Rubino is evaluated via RMSE and compared.

Starting from the dataset containing the daily measurement of the different sensors, all days are removed for which at least one of them did not produce any value. Data is not removed from all sensors if the missing values of one of the sensors are exactly at the start or at the end of the measurement campaign. Then, a simple linear regression is trained multiple times for each sensor, using a 30-day rolling window with one-day shifts to select the training set. For each training model, the sensor is calibrated for the entire period of the campaign, and its RMSE with T. Rubino reference is computed.

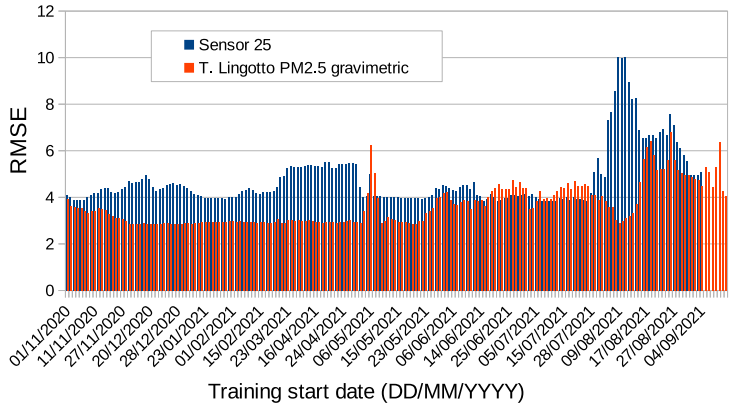
The results are shown in Figure 2.12. Sensor 25 is compared with the beta attenuation instruments of T. Lingotto and T.rebaudengo, and with the gravimetric instrument of T. Lingotto. Data is missing at the beginning of the campaign for the beta instrument of T. Lingotto, since this device was not available yet. The graphs show that low-cost sensor 25 has an almost constant error with T. Rubino independently of the calibration period, except during summer. This is in line with what was observed in Section 2.5.2: low-cost sensor measurements during the summer period are worse, since their error becomes comparable with the value being measured, and this is reflected in the resulting calibration model.

The RMSE of the beta sensor of T. Lingotto is almost constant and lower than the one of the light-scattering device. The same can also be said for the gravimetric sensor at T. Lingotto; however, in this case, it seems to perform worse than the beta sensor at the same location. The instrument of T. Rebaudengo is really close in terms of error to sensor 25, but it is not affected by the calibration problems during summer.

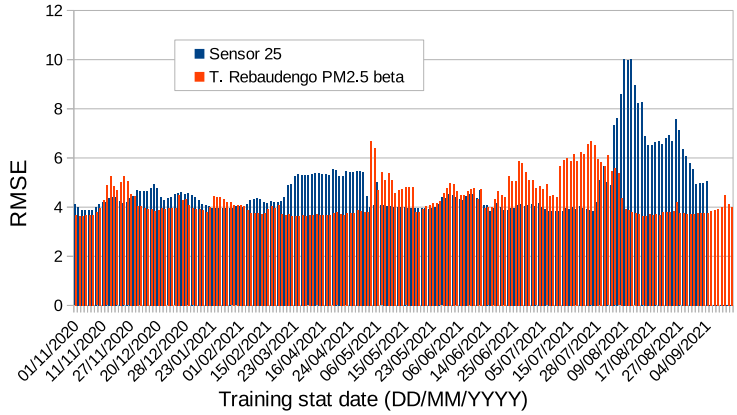
Finally, these images also show the importance of choosing the right training period for the calibration model, since it can significantly affect the efficacy of the resulting calibration.



(a) T. Lingotto β and light-scattering sensor 25.



(b) T. Lingotto gravimetric and light-scattering sensor 25.



(c) T. Rebaudengo β and light-scattering sensor 25.

Fig. 2.12 RMSE (μg/m^3) comparison between different instruments calibrated on T. Rubino [2].

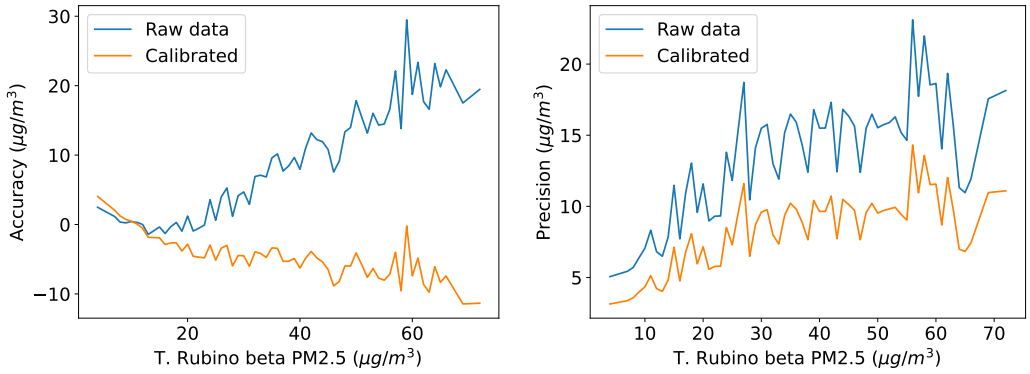


Fig. 2.13 Precision and accuracy on hour averages of light-scattering sensor 25 over T. Rubino beta [2].

Precision and accuracy of light-scattering sensors

Accuracy and precision of the calibrated and non-calibrated low-cost PM sensor 25 are computed w.r.t the beta attenuation instrument of T. Rubino on hour averages. The results are shown in Figure 2.13. The sensor calibration is performed using a simple linear regression model trained on the first 30 days of the measurement campaign.

Accuracy is computed as the difference between the average reading of the sensor for a certain value of the reference and the value of the reference itself. Precision, instead, is computed as the standard deviation of the sensor's readings for a specific value of the reference. From the graphs, it can be seen that the calibration improves both the accuracy and precision of the low-cost sensors for almost all values of the reference. In addition, the calibrated sensor satisfies the accuracy declared on the datasheet ($\pm$ 15 μg/m^3).

2.5.6 Conclusions

This analysis evaluates the performance of low-cost light-scattering PM sensors in relation to other official instruments available in the Metropolitan City of Turin. The objective is to understand the contribution that low-cost PM sensors can provide when an official monitoring network is in place.

Results show that, on day averages of PM2.5 concentrations, low-cost light-scattering sensors have a correlation with high-precision beta instruments of the station in which are located similar to the correlation of other official instruments situated in the same urban environment, i.e., the City of Turin. Therefore, they can be useful when the official monitoring sites are very sparse or missing.

Low-cost light-scattering sensors also have the benefit of providing higher frequency sampling, which is not available for beta attenuation and gravimetric devices. However, when the sampling rate is increased, they become less precise and accurate. This is evaluated on hour averages, where they still reach a good correlation with reference, even if it is significantly lower than for daily measurements. Finally, when calibrated with a simple linear model, their accuracy on hour concentrations is in line with what is specified by the manufacturer.

Calibration is also an interesting topic in the presented analysis. Selecting the correct training period is essential to ensure the efficacy of the calibration. Seasonality is a big factor in this: low-cost sensors are very noisy in summer due to the high relative error with the reference, which leads to bad calibration models if this period is used for training.

Future works should focus on evaluating their performance at higher sampling rates, for example, using TEOM devices or high-precision light-scattering instruments as reference. Low-cost sensors should also be evaluated on shorter time intervals to better understand the effect of seasonality and meteorological changes. Calibration models should be devised that take into account the sensitivity of low-cost sensors to different environmental factors. Selection of calibration period, duration, and re-calibration frequency, should all be further investigated. Fault and anomaly detection techniques should be developed to automatically detect wrong measurements and broken sensors.

Finally, different models of low-cost PM sensors, also from different manufacturers, should be evaluated. To this end, preliminary results on the third monitoring campaign show improved performance of the light-scattering sensors w.r.t. the presented analysis. This could be due to the different technological implementation of the low-cost PM sensors, or to the different environmental conditions to which they were subjected during the experiment.

2.6 Improving data quality

In this section, a data processing pipeline is introduced to improve the data quality of low-cost light-scattering PM sensors. The objective of this work is to identify a series of processing steps to be applied to the sensors' readings to obtain accurate measurements even in the presence of faults and anomalies. The process should require as little human supervision as possible since, in real-world deployments, the amount of data would be too large to be analyzed manually. In addition, algorithms and models are chosen to preserve the explainability of the obtained results. In the context of PM monitoring this is critical, since measurements could be used to define air quality policies and urban development plans. Results of this work are published in [3].

2.6.1 Dataset

The dataset is built starting from the data collected during the second monitoring campaign, presented in Section 2.4.2, where fourteen low-cost monitoring stations were positioned a official station T. Rubino for over one year, starting from October 10, 2020. For this analysis, only the PM2.5 measurements of the 56 low-cost light-scattering sensors are considered, since the concentration of the other size fractions is not measured directly but only estimated from PM2.5. Relative humidity readings from the low-cost monitoring stations are also included in this work. Data from low-cost sensors is not yet aggregated at this stage.

The hour PM2.5 measurements of the beta attenuation monitor of the ARPA station of T. Rubino were also collected to be used as a reference. This instrument does not provide reliable measurements below 4 $\mu g/m^3$. Therefore, all readings below this threshold are removed for this device.

2.6.2 Data exploration

This section discusses the calibration issues that this work tries to address and the statistical properties of the collected data.

Calibration issues

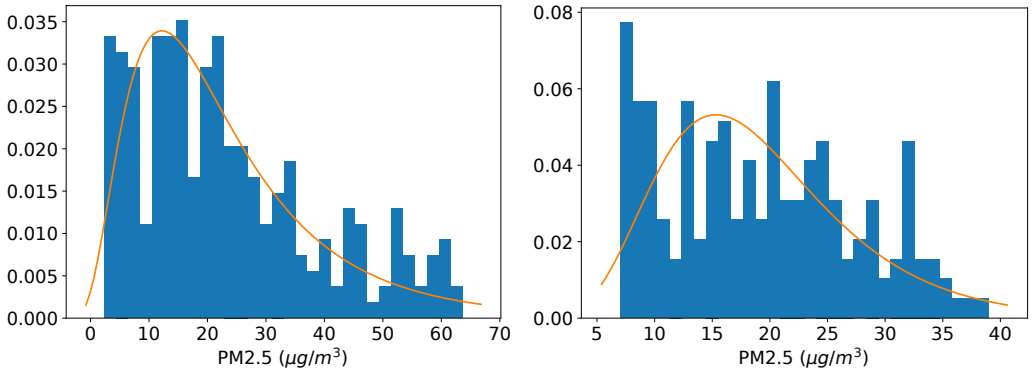
Previous work evaluated different calibration models for low-cost light-scattering sensors [4]. It was concluded that Multivariate Linear Regression, selecting PM2.5 and relative humidity as independent variables, is an effective and relatively simple choice to achieve this objective. The inclusion of humidity helps in mitigating the negative effects that this quantity has on measurement accuracy, which were discussed in Section 2.2.1. However, the effect of high relative humidity is not linear [88, 90, 91], so the model cannot fully compensate for it.

Another observation is that, even if measurements related to permanent sensor fault are removed, the presence of point anomalies in the training set leads to an overcompensation of the correction performed by the resulting calibration model. In fact, excessively small coefficients are often selected for PM2.5. This work tries to address this limitation by removing outliers from the training set. This is also supposed to lead to more robust calibration models, even in the presence of occasional anomalies in the sensor's readings.

For this analysis, the first three weeks of the data acquisition campaign, from October 10 to October 31, 2020, are selected as training period. While a single training period can affect the generality of the results, it helps in reducing the scope and complexity of the overall evaluation. Being in autumn, the calibration is not harmed by the issues that the sensors encounter during the summer months (see Sections 2.5.2 and 2.5.5).

Data distribution

Literature indicates that PM concentrations often follow a log-normal distribution [138, 139]. This characteristic is also present in the data collected during the calibration period, for both low-cost light-scattering sensors and the ARPA reference, as shown in Figure 2.14. This property is exploited by the processing pipeline to filter and detect outliers, as will be presented in the following.



(a) Sensor 14

(b) T. Rubino beta instrument

Fig. 2.14 Distribution of hour averages of $PM_{2.5}$ during first three weeks of the campaign (calibration period) [3].

2.6.3 Proposed pipeline

The proposed pipeline is shown in Figure 2.15. The output of this process should be calibrated and reliable sensor measurements, even in the presence of faults and anomalies discussed in Section 2.5.2.

The first step is the detection and removal of measurements related to permanent sensor faults. A simple failure detection algorithm is tuned and evaluated on the raw readings of the low-cost PM sensors. In the second phase, different filters are tested to mitigate the effects of sensor noise and point anomalies. Subsequently, a calibration model is trained on hour averages of PM2.5 and relative humidity measurements of the low-cost monitoring stations, automatically excluding outliers from the procedure. The performance of the resulting per-sensor calibration is evaluated w.r.t the ARPA reference. Finally, the median of the four calibrated low-cost PM sensors in each station is used as the best estimator of PM2.5 concentrations, and compared to the ARPA reference.

In the following of this section, each step in the pipeline is discussed in detail.

Failure detection

A simple algorithm is implemented to detect permanent low-cost PM sensor faults characterized by reading getting stuck in the lower range of the measurement scale (see Section 2.16). Figure 2.16 depicts the adopted approach. A rolling window is

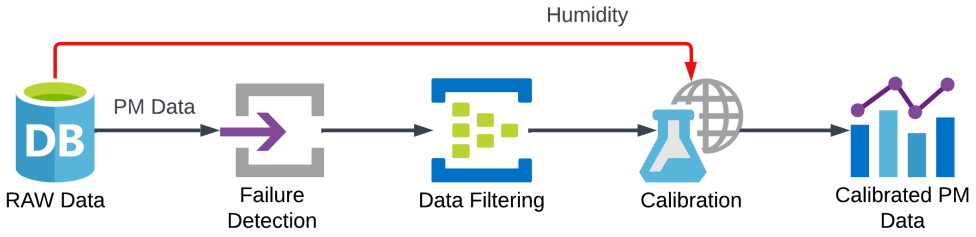


Fig. 2.15 Proposed data processing pipeline [3].

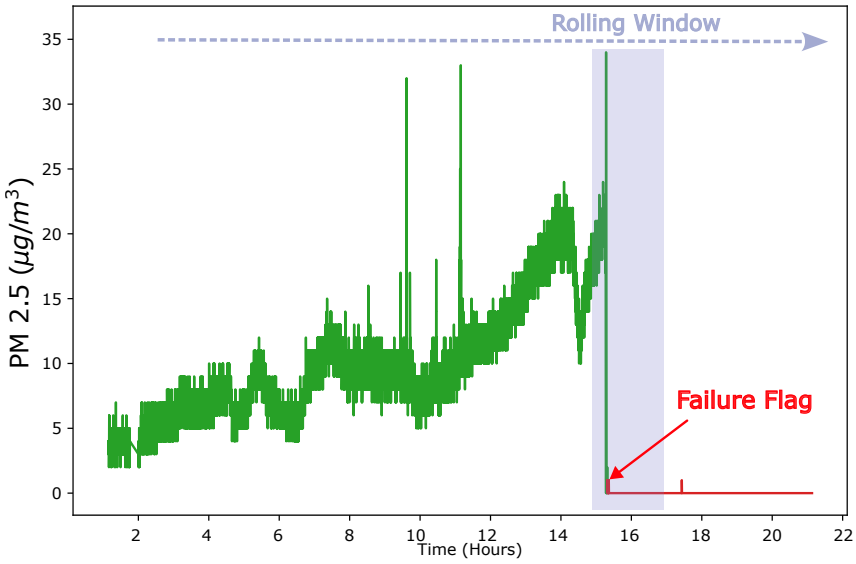


Fig. 2.16 Fault detection algorithm [3].

used to select different sets of contiguous measurements. If the PM2.5 readings of the sensor stay below the 2 $\mu g/m^3$ threshold for the entire length of the window, the sensor is marked as broken. This approach is applied to raw sensor measurements, with the window shifting for every new data point that is acquired. The window size, instead, is a tunable parameter of the algorithm. Tests are made using window sizes ranging from 1 to 12 hours, to maximize the percentage of detected faults and minimize false positives.

Filtering

The next step in the pipeline consists in filtering out point anomalies. As discussed in Section 2.5.2, they are characterized by high-frequency noise and impulsive changes.

Sometimes, these rapid and erroneous changes in the measured PM2.5 concentrations can be maintained for multiple minutes. This behavior can be attributed to elements getting stuck inside the sensors.

Filtering is performed on the raw per-second reading of the low-cost PM sensors, after permanent faults are removed in the previous step. The performance of the filters, instead, is evaluated on hour averages, to match the sampling frequency of the T. Rubino reference.

Three different filters are tested:

- **Low pass filter** In order to ensure that the signal remains unchanged in the bandpass, and to avoid alteration in the signal's shape due to a stringent cut-off band, an eighth-order Butterworth IIR filter is selected. The cut-off frequency is chosen according to a previous work [140].
- **Median filter** This is chosen as a non-linear filter. The kernel size is also defined according to a previous work [140].
- **Z-score filter** This filter is introduced to handle anomalies lasting for an undefined period of time, probably due to elements getting stuck inside the sensors. Due to their random duration, it is hard to define cut-off frequencies and kernel sizes for the low-pass and median filters to correctly remove them. The Z-score filter removes outliers based on the statistical distribution of the sampled measurements. At first, the mean and standard deviation of the sensor's readings are computed. Then, the Z-score (z) of each data point (x) is evaluated as its distance from the mean (μ), divided by the standard deviation(σ):

$$z = \frac{x - \mu}{\sigma} \tag{2.3}$$

  In simple terms, its absolute value indicates how many standard deviations a measurement is away from the sample mean. Filtering can be achieved by setting a maximum threshold for the absolute value of the Z-score of the data points. However, this process assumes that the measurements are normally distributed. This is not the case, as discussed in Section 2.6.2, since they follow a log-normal distribution. For this reason, the logarithm of the sensor's measurements is taken before applying the filter. To account for seasonality,

the filter is separately applied to non-overlapping seven-day windows. A z-score threshold of two is selected for this analysis.

Calibration model

Multivariate linear regression, using PM2.5 and relative humidity as independent variables [4], is the calibration model chosen for this step. Training is performed on data collected during the first three weeks of the measurement campaign, from October 10th to October 31st, 2020. Both PM2.5 and relative humidity measurements from the low-cost stations are aggregated to hour averages to match the granularity of the reference instrument, i.e., the beta attenuation monitor of T. Rubino. The calibrated model is then evaluated on the remaining part of the campaign. Both training and test data come from the previous stages of the pipeline.

However, as discussed in Section 2.6.2, multivariate linear regression is affected by the presence of outliers, in this case corresponding to point anomalies in the low-cost PM2.5 measurements. Even if this aspect is already addressed in the previous stages of the pipeline, during the training of the calibration model, which is usually performed before deployment, the data from the reference instrument is also available. This can be exploited to further improve outlier detection, excluding anomalous data points from the training set and leading to a more reliable calibration procedure. The objective of this approach is the automatic calibration of low-cost PM sensors without the need for extensive human supervision. This is useful in real-world scenarios, where the quantity of collected data is too high to be analyzed manually.

The adopted approach is based on a Multivariate Gaussian Model (GM). This model is fitted to the 2-dimensional data points of the training set, composed of the low-cost PM2.5 measurements and the corresponding reference values. A one-parameter probability distribution for the 2-dimensional data points can be defined in function of the Mahalanobis distance from the sample mean:

$$dist(x_i) = \sqrt{(x_i - x_{mean})\Sigma^{-1}(x_i - x_{mean})^T} \tag{2.4}$$

where Σ is the covariance matrix of the multivariate Gaussian distribution. Intuitively, the farther a data point is from the sample mean, the lower is its probability

of having been measured. Therefore, outliers can be removed by excluding data points that are at a farther distance from the center of the distribution. Different approaches can be used to select this distance. A simple approach is to exclude a fixed percentage of the less probable (or more distant) data points. However, this method poorly generalizes to the specific case of each sensor. Therefore, the threshold is defined in terms of a probability. Given the distance of a datapoint x_i, the probability of measuring a value x at a greater distance is [141]:

$$p(x|dist(x) \geq dist(x_i)) = e^{-(dist(x_i)^2)/2} \tag{2.5}$$

This formula provides a way to convert the distance of a data point from the sample mean to its probability of having been measured. When the distance of each data point is converted to a probability, filtering can be implemented by defining a threshold.

However, as discussed in Section 2.6.2, the data distribution of both the sensors' data and the reference follows a log-normal distribution. Therefore, the hour measurements of both the sensors and the reference residing in the training set are fitted with a log-normal distribution using the scypi [142] software package (scipy.stats.lognorm.fit). This allows for the estimation of the shift parameter of the distribution, which can be used to perform a more precise transformation of the data points to make them follow a Gaussian distribution:

$$normal = ln(lognormal - shift) \tag{2.6}$$

The outlier detection process can now be applied to the transformed data. In order to ensure the correctness of the log-normal-to-normal transformation, a normality test can also be conducted on the transformed data. Finally, measurements can be transformed back and used for the training of the Multivariate Linear Regression.

In the presented work, a probability threshold of 5% is used for all sensors. In addition, before starting the calibration process, low-cost PM sensors with a correlation with the reference lower than 0.65 during the training period are excluded. The reason is that if a sensor is already broken during calibration, it should not be considered for deployment.

2.6.4 Results

This section presents the evaluation of every step in the proposed pipeline.

Failure detection

Six different window sizes are tested for the failure detection algorithm: 1, 2, 4, 8, and 12 hours. The algorithm is considered to be successful if the failure is recognized in less than two weeks from the moment it manifests. Results are shown in Tables 2.11 and 2.12. Accuracy, Precision, Recall, and F1-Score are evaluated as follows:

$$accuracy = \frac{TruePositives + TrueNegatives}{Positives + Negatives} \tag{2.7}$$

$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.8}$$

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.9}$$

$$F1_{score} = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.10}$$

As can be seen, smaller window sizes increase the probability of False Positives, while large ones increase the probability of False Negatives. The 12-hour window is selected as the best solution, since it provides better accuracy and precision with only a small reduction in the recall. It is also important to note that the False Negatives are mainly due to sensors consistently measuring highly noisy data before getting stuck at the bottom of the measurement scale. Detecting these types of faults is more difficult and would require a better understanding of the underlying cause.

Filtering

Filters are applied to the per-second data produced by the PM2.5 sensors. Their performance is evaluated on hour averages w.r.t. to the T. Rubino reference using multiple metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE),

Table 2.11 Window size evaluation for failure detection: True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN) [3].

Window size (hours)	TP	FP	FN	TN
2	28	13	7	8
3	31	7	9	9
4	32	5	9	10
8	32	3	10	11
12	32	0	11	13

Table 2.12 Window size evaluation for failure detection [3].

Window size (hours)	Accuracy	Precision	Recall	F1-Score
2	0,643	0,683	0,800	0,737
3	0,714	0,816	0,438	0,570
4	0,750	0,865	0,780	0,821
8	0,768	0,914	0,762	0,831
12	0,804	1,000	0,744	0,853

Person correlation (r^2), and the percentage of improved/worsened measurements. Results are shown in Table 2.13, where the metrics are averaged between all the considered sensors. Among all, the Z-score filter achieves better performance in terms of reduced RMSE and increased correlation. It was also tried in combination with the median filter, which also achieved good performance, but the combined result was not satisfactory. Therefore, the Z-score filter is the one selected to be used in this step of the pipeline.

Table 2.13 Filter evaluation by comparing hour averages with T. Rubino. Metrics are averaged between the different sensors [3].

Filter	RMSE ($\mu g/m^3$)	MAE ($\mu g/m^3$)	r^2	Improved (%)	Worsened (%)
Raw Data	18,642	13,226	0,810	-	-
Low Pass Filter	18,724	13,268	0,808	1,160	0,553
Median Filter	18,558	13,162	0,812	0,592	0,178
Z-Score	18,042	13,176	0,822	2,161	2,151
Median + Z-Score	18,443	13,169	0,815	1,873	2,320

Table 2.14 Comparison between different calibration models. Metrics are averaged between sensors [3].

Calibration model	RMSE ($\mu g/m^3$)	MAE ($\mu g/m^3$)	r^2	R^2
PM + Hum	9,608	7,149	0,857	0,670
PM + Hum + GM	9,125	6,744	0,857	0,692
Filtered PM + Hum	9,185	6,960	0,865	0,677
Filtered PM + Hum + GM	8,976	6,681	0,866	0,700

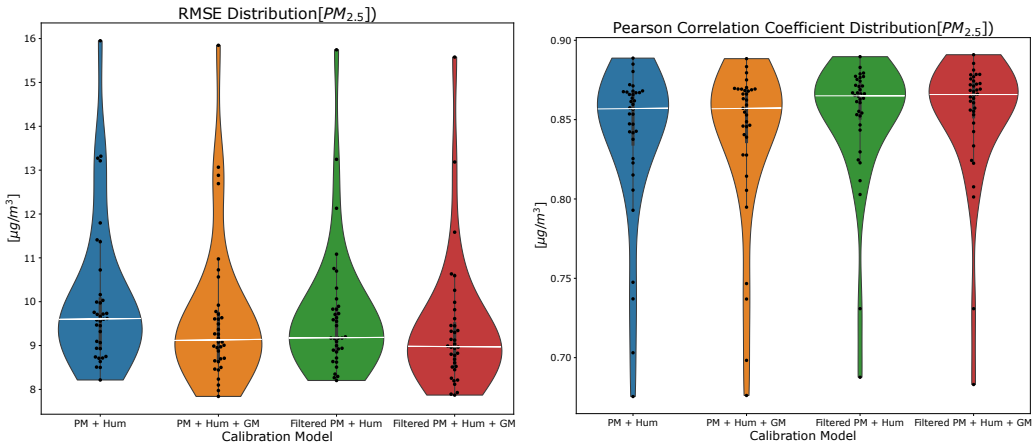
Calibration

The calibration models are evaluated by excluding sensors exhibiting permanent faults that were not detected during the failure detection phase. This is done to independently assess the efficacy of the calibration. In any case, the combined performance will be presented in the next section.

The base reference model is a Multivariate Linear Regression using PM2.5 and humidity as independent variables (PM + Hum). The model is then evaluated on Z-score filtered data (Filtered), used in both training and testing, and by removing outliers from the training set via the Multivariate Gaussian Model (GM). These two approaches are tested both separately and together. Results are shown in Table 2.14 and Figure 2.17.

Table 2.14 presents a global overview of the improvements that are achieved over the reference model (PM + Hum). Metrics are averaged between sensors. It can be seen that both approaches, if used independently, are able to reduce RMSE and MAE, while also increasing correlation and the coefficient of determination (R^2). However, the combination of the two achieves the best result.

The violin plots in Figure 2.17 show that the proposed solution also helps in limiting the variance of correlation and RMSE between the different sensors. This highlights the efficacy of filtering and outlier removal (GM) in handling sensor anomalies, leading to a more stable and resilient calibration.



(a) RMSE (b) Correlation

Fig. 2.17 RMSE and correlation distribution for each calibrated sensor (black dots). The white lines represent the sensors' average [3].

Overall evaluation

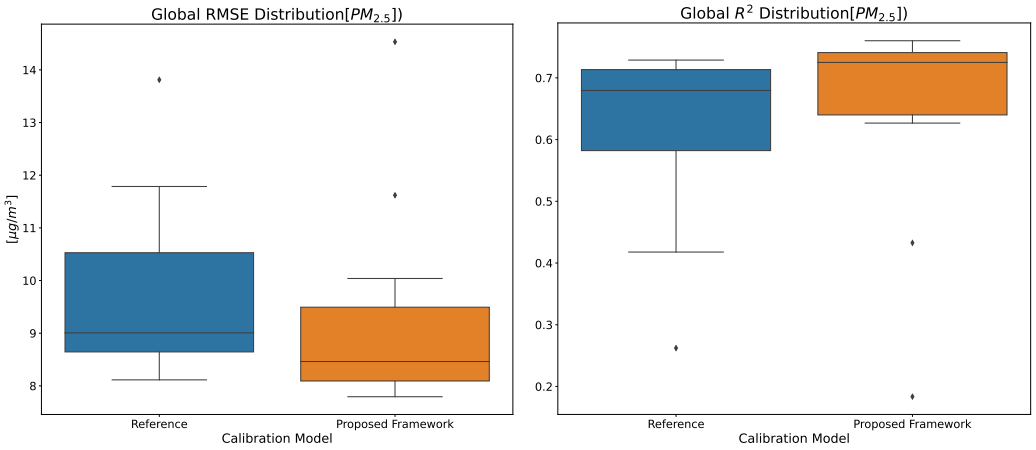
The overall evaluation of the pipeline is conducted by considering each low-cost monitoring station as a whole. The median of the hour averages of the installed PM2.5 sensors is taken for each station. This is done after each of the sensors has been processed by the entire pipeline. Finally, the resulting PM2.5 measurement time series of each station is compared to the reference. Similarly, base reference time series are generated for each monitoring station as the median of their sensors, which are calibrated using standard Multivariate Linear Regression.

Table 2.15 shows the average performance achieved by the different stations for both the base reference (PM + HUM) and the proposed pipeline. As it can be seen, the proposed approach manages to improve all metrics. In addition, thanks to the four sensor redundancy, the resulting performance is equivalent or superior to what is achieved by calibrating the single sensors (see Table 2.14).

By analyzing the box plots in Figure 2.18, it can be seen that variance in the sensors' performance is also reduced by the introduced pipeline. However, some bad-performing stations are still present. This is due to faults and anomalies occurring for more than two low-cost PM sensors in the same station, that are not identified and corrected by the pipeline. Even if infrequent, these problems should be addressed in future work.

Table 2.15 Comparison between reference and proposed pipeline. Metrics are averaged between stations [3].

Calibration model	RMSE $(\mu g/m^3)$	MAE $(\mu g/m^3)$	r^2	R^2
PM + Hum	9,006	6,757	0,858	0,654
PM + Hum + GM Filtered	8,462	6,241	0,868	0,725



(a) RMSE (b) R^2

Fig. 2.18 RMSE and R^2 distribution for the different monitoring stations: comparison between reference and proposed pipeline [3].

2.6.5 Conclusions

A data processing pipeline to improve the data quality of low-cost light-scattering PM sensors is introduced. The objective of this solution is to define a series of processing steps to apply to the sensors' readings to obtain accurate measurements even in the presence of faults and anomalies. The process should require as little human supervision as possible since, in real deployment, the amount of collected data can be too high to be analyzed manually. In this study, the measurements collected during the second monitoring campaign, described in Section 2.4.2, are used. The proposed solution also focuses on the explainability of the adopted models, since transparency is essential in the context of air quality monitoring.

The pipeline is designed to achieve multiple objectives. At first, it detects and removes permanent sensor failures via a simple threshold algorithm. Secondly, high-frequency noise and point anomalies are removed by a carefully selected filtering algorithm. In the subsequent step, calibration is performed by automatically identifying and removing the remaining outliers in the training set. Finally, the redundancy provided by the four low-cost PM sensors in each station is exploited by taking their median.

Results show that the proposed approach is able to increase the reliability of low-cost PM2.5 measurements across multiple metrics. In addition, the variance in performance is reduced between both sensors and overall stations, showing the efficacy of the solution in handling failures and anomalies.

Future works should further challenge the effectiveness and generality of the proposed approach by testing on different calibration periods. In addition, the effects of environmental conditions, especially the ones related to relative humidity, should be carefully considered and better characterized in the calibration model. Studies should also be performed in different scenarios, e.g., high-traffic areas, where the statistical distribution of the data might change. Different models of low-cost PM sensors, also from different manufacturers, should be considered. Finally, neural-network-based models should also be tested and compared, even if they preclude the explainability of the obtained results.

2.7 Duty cycle evaluation

In this section, it is analyzed how the introduction of a duty cycle, with the objective of reducing wear and power consumption, affects the measurement quality of low-cost light-scattering sensors. Starting from the data collected during the second monitoring campaign, presented in 2.4.2, sensor measurements are re-sampled by simulating different duty cycles. The obtained signals are compared to both the original ones and to the high-precision ARPA reference. Results are published in [6].

2.7.1 Dataset

The dataset contains the per-second measurements of the low-cost monitoring stations during the first 6 months of the second monitoring campaign, described in 2.4.2. In this campaign, fourteen low-cost monitoring stations were positioned at the official station T. Rubino, in the City of Turin, Italy. Being a background location, PM levels are not expected to change rapidly. For this analysis, only PM2.5 measurements are considered, since the other size fractions are not measured directly but are estimated from PM2.5. Low-cost relative humidity measurements are also included in the dataset. The hour measurements of the high-precision beta attenuation monitor of the ARPA station are collected for the same period.

2.7.2 Pre-processing

The first step is to remove faulty light-scattering sensors, since their presence would conceal the effect of introducing a duty cycle. Hour averages of the PM2.5 measurement of each low-cost sensor are computed, and compared with the ARPA reference.

Only sensors with a correlation higher than 0.82 with the official instrument are included in the analysis. This threshold provides a good trade-off between the number of discarded sensors and the quality of the measurements. A visual inspection is also carried out by plotting the hour averages of the measurements against the reference, and looking for evident anomalies of faults. Sensors that did not work properly, even for just a portion of the measurement campaign, are removed from the analysis. From a total of 56 PM sensors, only 24 are selected.

This work wants to analyze both raw and calibrated data. For this reason, a Multivariate Linear Regression model, using both PM2.5 and relative humidity as independent variables, is adopted for calibration. The model is trained on the hour averages of the low-cost sensors' measurements, targeting the ARPA reference. The training set consists of the first two weeks of the measurement campaign.

The per-second PM2.5 measurements of the low-cost sensors are calibrated using the trained model. However, this also requires having relative humidity measurements for every second. Since the low-cost humidity sensors only provide readings every 3-4 seconds, the missing measurements are filled using the last known value.

Finally, all data corresponding to the training period is removed from the analysis. In order to maintain a similar number of samples between the different months, the entire month of October, containing the data used for training, is discarded.

2.7.3 Methodology

The per-second PM2.5 measurements of the low-cost PM sensors are re-sampled to simulate the introduction of a duty cycle. The duty cycle and the length of its period are selected as follows:

$$T = 2^x seconds \tag{2.11}$$

$$DC = 2^y seconds/T \tag{2.12}$$

By varying **x** and **y**, multiple combinations of duty cycles and period lengths are selected. **x** is varied in the range [0, 10] to ensure that at least one sample is taken every hour. **y**, instead, is varied in the range [0, **x**). This methodology for selecting periods and duty cycles simulates exponential changes in the operating time of the PM sensors. As a consequence, a broader exploration of the search space is achieved.

Finally, for each sensor and for each combination of duty cycle and period, the hour averages are recomputed. The resulting measurement time series are compared with the ARPA reference using both RMSE and Pearson correlation, and the results are averaged between sensors. The analysis is carried out for both raw and calibrated

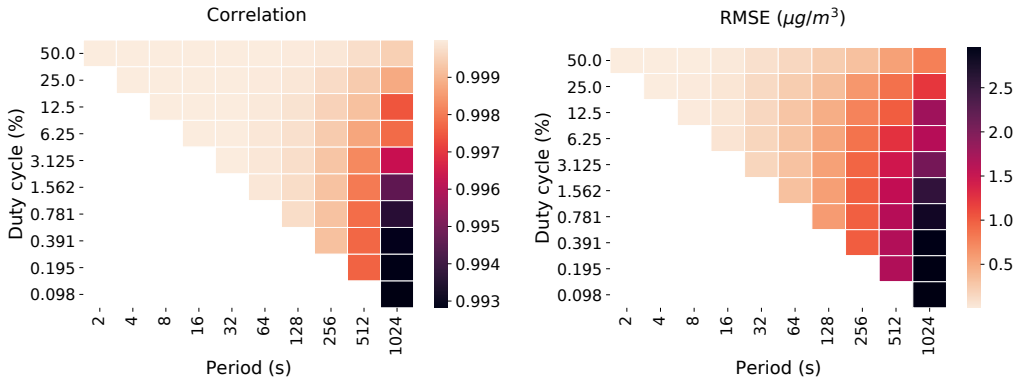


Fig. 2.19 Correlation and RMSE with respect to the raw full data for different periods and duty cycles [6].

data. The same evaluation is then repeated by partitioning the data according to the different hours of the day and the different months.

As a final note, it is important to mention that the actual sensors require six seconds after startup to provide reliable measurements. Therefore, the actual duty cycle will be higher than the simulated one.

2.7.4 Results

Figure 2.19 reports the correlation and RMSE evaluated between the original and the re-sampled sensor data, considering all the different combinations of period and duty cycle. When increasing the period and reducing the duty cycle, the information loss is clearly visible for both the considered metrics. However, even in the worst case, the variations of correlation and RMSE are minimal (0.007 and $3\mu g/m^3$, respectively).

Figure 2.20 compares the re-sampled signals to the ARPA reference using the same metrics. The trends remain the same, even if some exceptions are present. As expected, the overall correlation and RMSE w.r.t the reference are lower than with the original signal. However, their variation is reduced across the different combinations of duty cycles and periods: the range of variation of the correlation decreases from 0.007 to 0.003, while the one of the RMSE from $3\mu g/m^3$ to about $0.2\mu g/m^3$. This is probably due to the fact that the inaccuracy of the low-cost sensors conceals the information loss induced by the duty cycles.

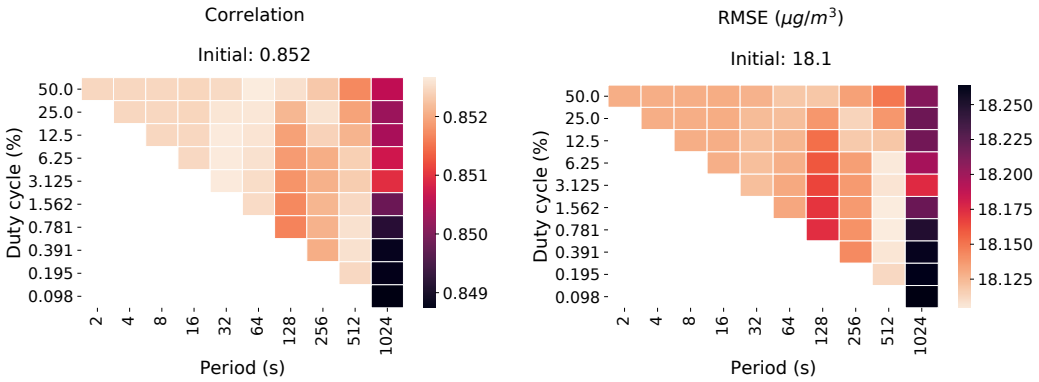


Fig. 2.20 Correlation and RMSE of raw data with respect to the ARPA reference for different periods and duty cycles [6].

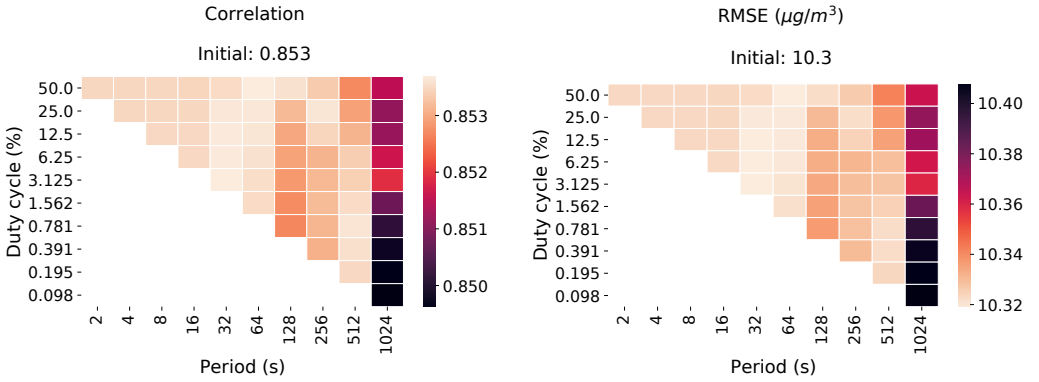


Fig. 2.21 Correlation and RMSE of calibrated data with respect to the ARPA reference for different periods and duty cycle [6].

The situation remains similar after applying the calibration (Figure 2.21). Calibration increases the overall accuracy, reducing RMSE, but does not affect precision, since the correlation remains similar. Concerning the range of variation of correlation and RMSE, there are no significant changes from the scenario without calibration. Correlation varies by 0.003 between all the considered combinations of duty cycles and period, while RMSE by $0.2 \mu g/m^3$.

The second part of the analysis, instead, focuses on evaluating the effects of the different duty cycles and periods during different months and hours of the day. The objective is to understand whether a dynamic duty cycle schedule could be effective in reducing power consumption while maintaining data quality. In fact, the speed at which PM2.5 concentrations vary can be affected by the different environmental conditions that are encountered during the day or the year (e.g., peak hour traffic and

seasonal changes). These analyses are carried out using calibrated data and ARPA as a reference.

Figure 2.22 shows that the effect of introducing a duty cycle is negligible w.r.t the variation of correlation and RMSE during the day. In addition, it is possible to see that correlation notably worse from midnight to 9 a.m.. This is probably due to the harsher environmental conditions that are encountered, i.e., higher humidity levels and lower temperatures. For what concerns RMSE, the increase in measurement error in certain hours is consistent with the increase of PM2.5 levels during the same hours.

Figure 2.23 shows the results of the analysis conducted over different months. Also, in this case, the influence of the duty cycle is negligible w.r.t to the variation in correlation and RMSE between the different months. The month of November represents an outlier in terms of both correlation and error, probably due to the very high humidity levels that were encountered.

2.7.5 Conclusion

In conclusion, results show that, even when the power-on time of the sensors is substantially reduced by the introduction of a duty cycle, the effects on hour measurement quality are negligible w.r.t. the original signal and the high precision reference. This is also true when different months or hours of the day are separately considered. In fact, the variation of correlation and error during the day and the year greatly exceeds the imprecision added by the duty cycle operation.

Therefore, the study shows that the reduction of the sensor duty cycle can be a viable solution to increase device lifetime, reduce maintenance costs, and limit power consumption. This could result in wider and more manageable sensor deployments.

A limitation of this work is that it only considers a low-traffic environment. Future works should validate the results presented in this study in high-traffic conditions.

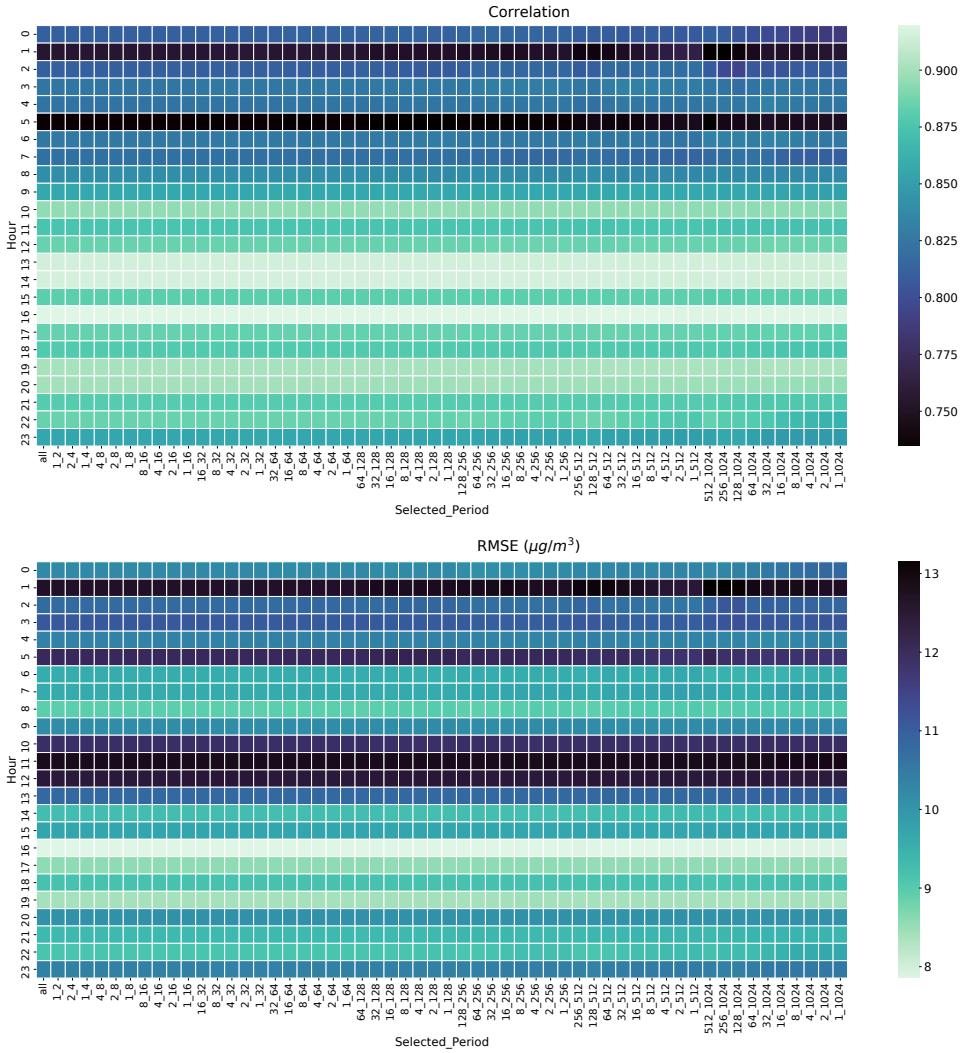


Fig. 2.22 Correlation and RMSE of calibrated data with respect to the ARPA reference for different hours of the day [6].

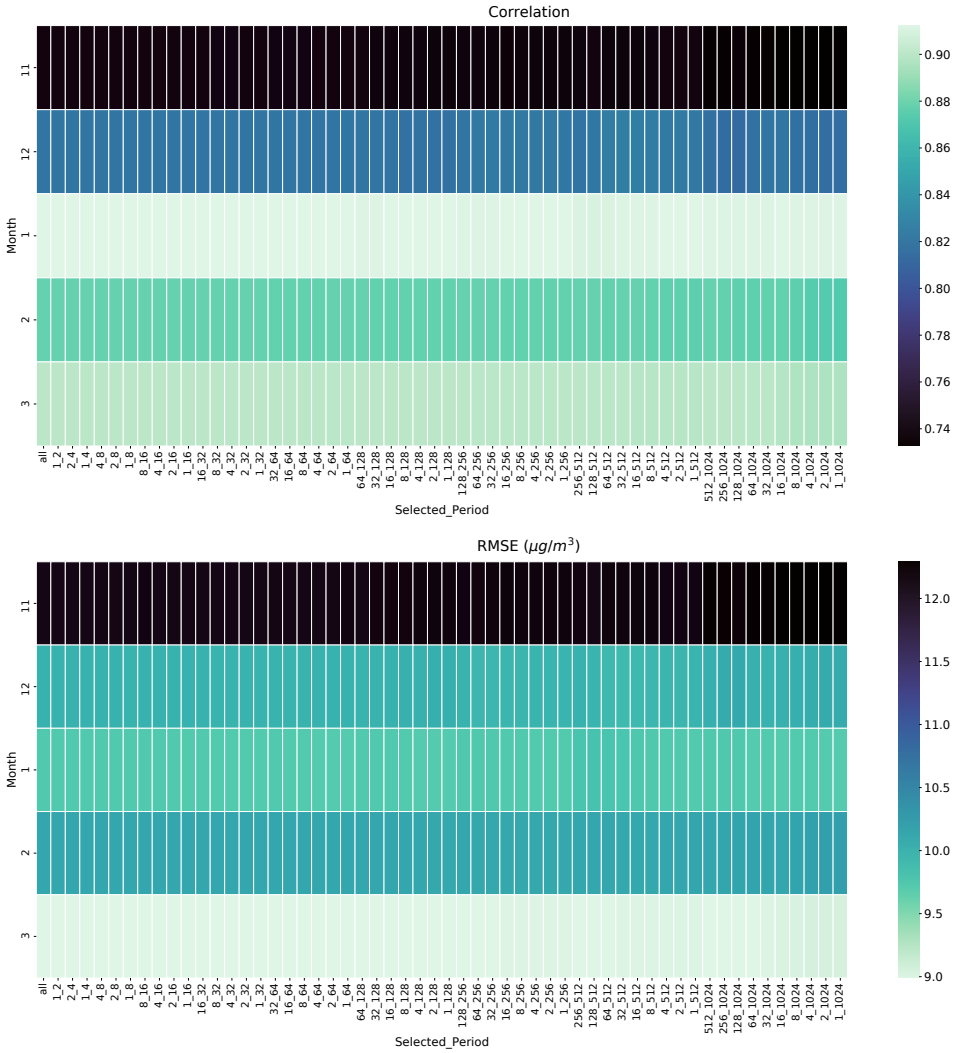


Fig. 2.23 Correlation and RMSE of calibrated data with respect to the ARPA reference for different months [6].

2.8 Backend infrastructure

A backend infrastructure is created to support the transmission, storage, visualization, and retrieval of data generated by the low-cost monitoring stations that implement the new design (see Section 2.4.1). However, it can also be adapted to work with any IoT device that is able to send data via the MQTT protocol. This section starts by describing the base architecture. Then, it shows how it is customized and integrated with external services for the crossroad experiment described in Section 2.4.2.

2.8.1 Architecture

Figure 2.24 depicts the backend architecture that is designed to support efficient data collection, storage, visualization, and retrieval. All software services are deployed as Docker [143] containers. Docker allows the creation of portable software applications that can be easily deployed, managed, and replicated across various environments.

The low-cost monitoring stations send their measurements via MQTT messages that are received by the MQTT Broker. A dedicated client connects to the broker to receive and parse the messages of the stations. Subsequently, the extracted measurements are inserted into the database. Software schedulers process and enrich the stored information at regular intervals, by computing measurement averages and by accessing external services (e.g., to retrieve meteorological data). The Web-API is designed to support visualization and data analysis use cases, by providing endpoints to query the stored measurements. Finally, a dashboarding application is configured for real-time monitoring of the data sent by the low-cost stations.

Broker

The MQTT broker is in charge of receiving and delivering MQTT messages to all connected clients. The MQTT protocol is described in Section 1.2.3. It operates via a publish-subscribe pattern, where communication channels are represented by topics of interest. Clients can define topics and publish messages on them. Messages are then received by all the clients subscribed to the corresponding topics.

In this architecture, the low-cost monitoring stations use pre-defined topics to publish the collected measurements. In order to reduce overhead, messages are

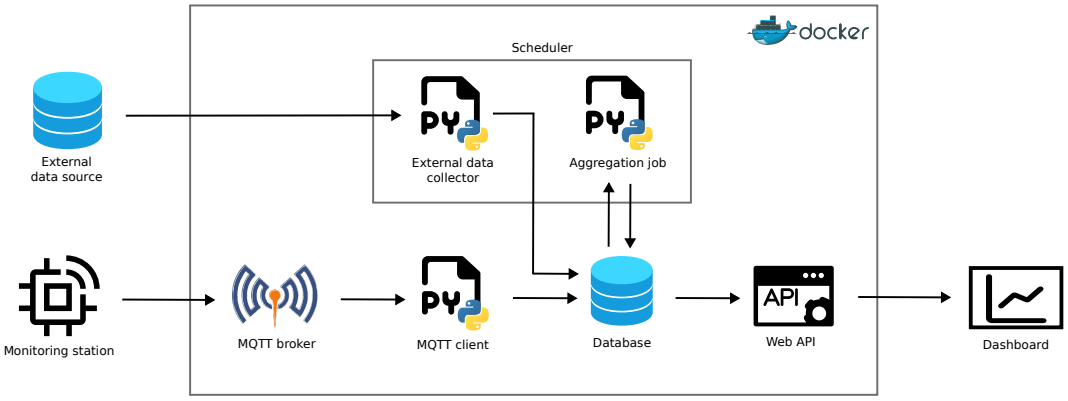


Fig. 2.24 Backend architecture [7].

sent with a Quality of Service (QoS) level equal to zero. Therefore, there are no guarantees on their delivery. Higher QoS levels could be used to transmit control messages to and from the monitoring stations, but this functionality has not yet been implemented.

Due to its simplicity and ease of configuration, Mosquitto [144] was selected as the MQTT broker for this infrastructure. However, some limitations were encountered during development, mostly related to user management and authorization.

MQTT Client

An MQTT client, developed in Python [145] using the paho-mqtt library [146], connects to the broker and retrieves the messages coming from the low-cost monitoring stations. In order to achieve this, the client obtains information about the deployed stations from the database and subscribes to their topics. When a measurement packet is received, the client parses its content and inserts the contained measurements in the database.

Database

The Database Management System (DBMS) adopted by the infrastructure is MySQL [147]. Being a well-documented and established open-source solution, it was selected to simplify the development process. However, time-series-oriented databases, such as Timescale [148] and Influx DB [149], should also be evaluated in the future.

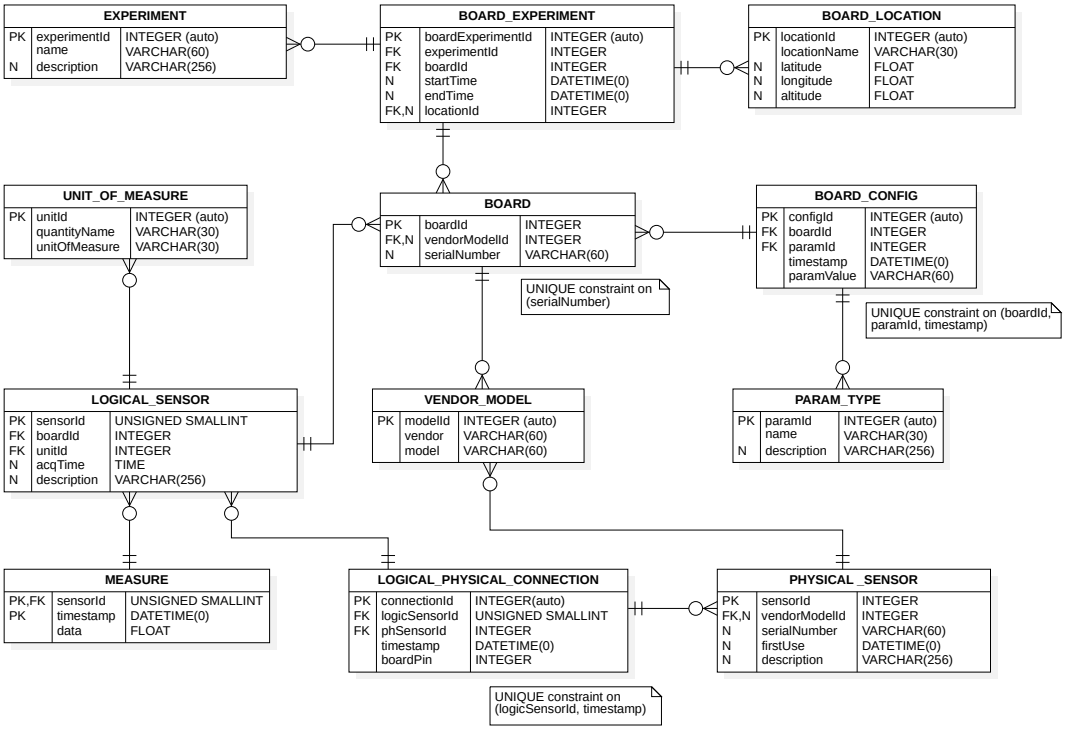


Fig. 2.25 ER diagram of the database [7].

The database schema, reported in Figure 2.25 as an ER diagram, is designed to support a general data-collection use case for IoT systems. The *BOARD* table represents an IoT device containing multiple sensors. Each *BOARD* is characterized by a serial number and a vendor, which is externally referenced via a foreign key from the *VENDOR_MODEL* table. Each IoT device generates multiple time series of measurements. These are represented by the *LOGICAL_SENSOR* table. This table references the *BOARD* that generated the time series via a foreign key, and contains additional properties such as the sampling frequency and the quantity being measured. The measured quantity is described by the *UNIT_OF_MEASURE* table, which is also referenced via a foreign key. Each *UNIT_OF_MEASURE* contains the name of the measured quantity and the measurement unit in which it is expressed.

The *MEASURE* table is a fundamental component in the design. It is used to store the single acquisitions of each *LOGICAL_SENSOR*. Each measurement is expressed as a tuple in the format *(sensorId, timestamp, data)*. The composite primary key of the table is defined by the tuple *(sensorId, timestamp)*. Due to its lower cardinality, *sensorId* is selected as the first column in the composite key, providing a speedup when filtering for this parameter.

Since each single measurement results in an entry in the *MEASURE* table, it is of great importance to reduce its size as much as possible to save disk space. For this reason, *sensorId* is chosen to be an *UNSIGNED_SMALL_INT*, an *data* to be expressed in single precision, occupying respectively two and four bytes. The *timestamp* of the measurement, instead, is represented by a *DATETIME* data type with no fractional part for seconds, occupying only five bytes. This data type also does not execute timezone conversion before storage and retrieval, which is useful since the entire infrastructure always expresses time in UTC. In total, only eleven bytes are used to save each measurement. Unfortunately, this optimization results in a limitation in terms of measurement precision, number of *LOGICAL_SENSOR*s, and sampling frequency. However, according to the use case, the size of the data types can be changed.

The *PHYSICAL_SENSOR* table represents the sensor hardware that produces the measurement time series described in the *LOGICAL_SENSOR* table. Among other things, this table provides information about the age of the sensor via the *firstUse* column. *PHYSICAL_SENSOR*s and *LOGICAL_SENSOR*s are related via the *LOGICAL_PHYSICAL_CONNECTION* table. This allows for the representation of sensors that measure multiple quantities and the tracking sensor replacements due to failures. In case of replacement, the new sensor will act as the same *LOGICAL_SENSOR*(s).

The *BOARD_CONFIG* table is used to track the configuration of different board parameters, represented by the *PARAM_TYPE* table. Instead, the *EXPERIMENT*, *BOARD_EXPERIMENT*, and *BOARD_LOCATION* tables are used to track the usage of the IoT devices across different measurement campaigns and deployments. To express device movement, it is possible to keep the *locationId* unspecified in the *BOARD_EXPERIMENT* table, and define logical sensors representing the time series of device positions. These time series can coincide with the log of a GPS sensor, if installed on the device.

Figure 2.26 shows the ER diagram of additional tables used to store data aggregations. The *FIVE_MIN_AVG_MEASURE* and *HOUR_AVG_MEASURE* tables contain measurements aggregated, respectively, over five minutes and one hour. In addition to the average, the maximum, and the minimum, the standard deviation of the aggregated data points can also be computed and stored. The identifier of the aggregated time series (*sensorId*) is the same as for the original *LOGICAL_SENSOR*.

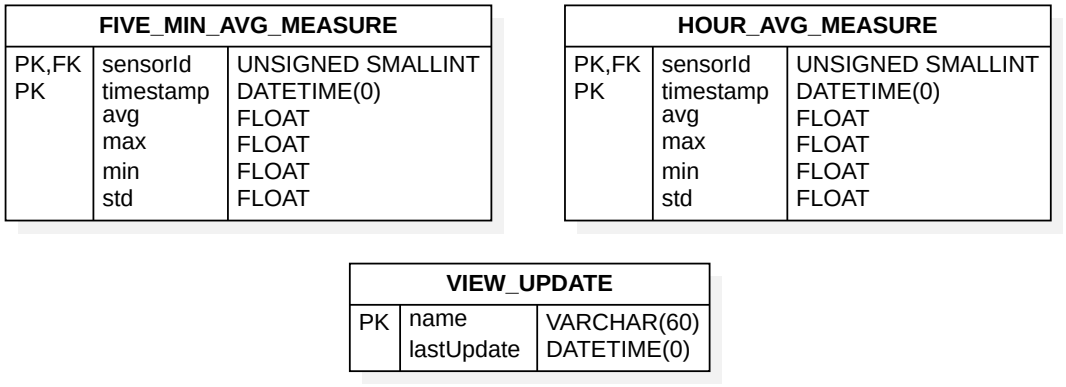


Fig. 2.26 ER diagram of aggregation tables [7].

Finally, the *VIEW_UPDATE* table records the time at which each aggregation table was last updated.

Scheduler

Multiple schedulers are defined to carry out specific operations at regular time intervals. These are implemented using the APScheduler [150] Python library. Their main task is to continuously update the data aggregations that are stored in the *FIVE_MIN_AVG_MEASURE* and *HOUR_AVG_MEASURE* tables. The aggregated measurements are useful in reducing the load on the API and database when downloading historical data for analysis and visualization. From the *VIEW_UPDATE* table, it is also possible to recompute past aggregations by changing the *lastUpdate* field related to the aggregation table of interest, by setting it to an older timestamp. All more recent aggregations will be recomputed and updated. Finally, schedulers can be used to retrieve information from external sources, e.g., a weather API for reference meteorological conditions.

Web API

The web API, implemented using Flask [151] Python library, allows external services and users to access the information stored in the database. Endpoints are defined to retrieve either raw or aggregated data. In addition, the API can perform specific second level aggregations, adapt the format of the stored data to the designated

visualization service, and limit the amount of measurements that can be queried in a single invocation.

Dashboard

The dashboarding application utilized by the system is Grafana [152]. It is an open-source tool for real-time data visualization and analytics. Grafana is used to achieve two main objectives. The first is to monitor the data received from the IoT devices, in order to understand whether there are problems related to data collection or transmission. Condition-based rules can also be defined to trigger alarms for system administrators. For this used case, Grafana is configured to query the database directly.

The second objective is to provide data visualization to end users. Graphs updating in real-time can be created and embedded in custom web applications. For this use case, Grafana is configured to access ad-hoc API endpoints.

2.8.2 Crossroad experiment

The crossroad experiment is conducted within a project involving external partners. Therefore, the architecture needs to be adapted for the specific use case, which requires its integration with external services. The objective of this work is to implement a proof-of-concept infrastructure that exploits blockchain technologies to provide validation for the collected data. The main idea is to group measurements in blocks, compute their hash values, and store the result inside a blockchain. Every entity or end user that gets in possession of the data related to the experiment can verify its validity by querying the blockchain. The overall architecture is shown in Figure 2.27.

Low-cost monitoring stations measure PM2.5 concentration at various points of interest. Every minute, an MQTT message containing the measurements is sent to the broker via a 5G cellular network. An MQTT client subscribes to the relevant topics on the broker to receive the measurements in real time, saving them in the database. The measurements are made accessible to stakeholders through a Web API.

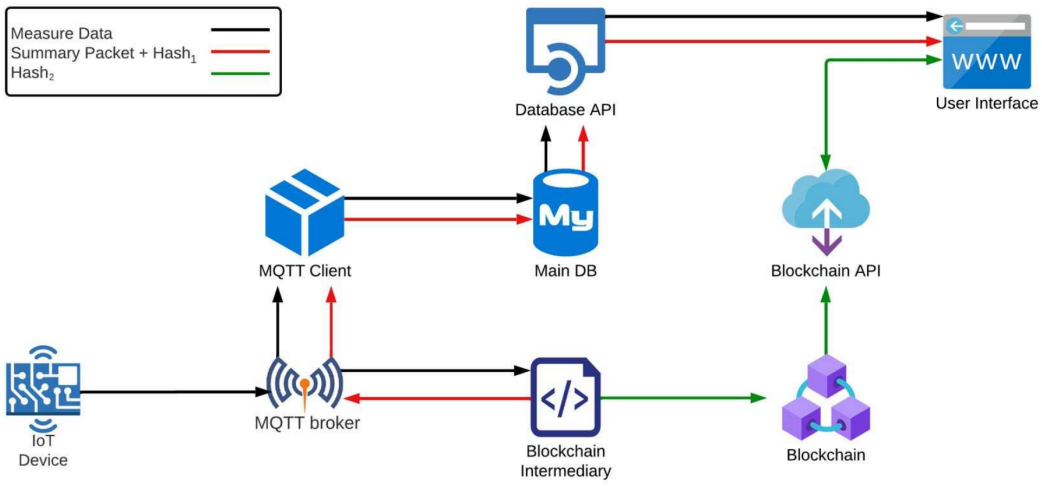


Fig. 2.27 Backend architecture for crossroad experiment [5].

A software application, acting as an intermediary for the blockchain, also subscribes to the broker to receive real-time measurements. For each board, it concatenates the measurements received during the hour and calculates their hash. The hash of the measurements, along with additional metadata, forms an hourly summary packet. The hash of this summary packet is inserted into the blockchain. The summary packet is then sent via the MQTT broker to the MQTT client, which saves the summary information in the database.

The user receiving the measurements and metadata through the Web-API can reconstruct the summary packet and verify the presence of the hash in the blockchain. To do this, a reference implementation of the verification client is provided. To check the presence of the hash in the blockchain, a dedicated API is created.

Board messages

Every minute, each station sends an MQTT message containing the collected measurements to the broker. The payload of the message is a JSON object structured as shown in Listing 1.

The meaning of the fields is the following:

- *block_ts* is the timestamp at which the packet is sent, expressed in Unix time.
- *board_id* is the numerical identifier of the station.

```
{
    "block_ts": integer,
    "board_id": integer,
    "data": string
}
```

Listing 1 Payload of MQTT messages sent by stations.

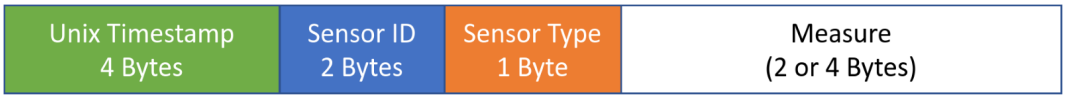


Fig. 2.28 Binary format for measurement.

- *data* is the base64 encoding of the *binary measurement packet*.

The tuple (*block_ts*, *board_id*) is used to uniquely identify the *binary measurement packet*. The *binary measurement packet* is composed by the concatenation of measurements expressed in the binary format detailed in Figure 2.28.

All fields are formatted in network byte order (Big Endian) for compatibility across different architectures. All fields are numeric and contain unsigned integers, except for *SensorType*, which contains an ASCII character. The meaning of the fields is the following:

- *Unix Timestamp*: measurement timestamp in Unix time.
- *Sensor ID*: identifier of the *LOGICAL_SENSOR*.
- *Sensor type*: identifies which is the quantity being measured and the respective encoding.
- *Measurement*: measured value, encoded according to the measurement type to be expressed as an unsigned integer.

The *Sensor Type* for each measured quantity is shown in Table 2.16. In addition, the table shows the formulas that are used to convert measurements of different *Sensor Type*s into unsigned integer values.

Table 2.16 Encoding for measured quantities.

Quantity	Sensor Type	Length (bytes)	Conversion formula	Range	Resolution	Unit
Temperature	'T'	2	$(x+40)\times 100$	-40 to 80	0.01	Celsius
Humidity	'H'	2	$x\times 100$	0 to 100	0.01	%
PM2.5	'P'	2	x	0 to 10000	1	μg/m^3
Pressure	'A'	4	$x\times 10000$	300 to 1100	0.001	hPa
GPS coord.	'G'	4	$(x+180)\times 10^6$	-180 to 180	10^{-6}	Degrees

Summary packets

The Blockchain Intermediary creates a measurement file for each station by concatenating the *data* fields of all the board messages received in one hour. The measurement file is only needed to create the summary packet, and then it is discarded. The summary packet is sent to the MQTT client, which saves the stored information inside the database. Finally, the hash of the summary packet is inserted in the blockchain. The format of the summary packet created by the Blockchain Intermediary is shown in Listing 2.

```
{
    "created": <string_timestamp>,
    "board_id": <station identifier>,
    "timestamps": [ <array of block_ts> ],
    "size": <size in bytes of file>,
    "number": <number of payloads collected in the file>,
    "ip": <IP address of board>,
    "hash": <SHA-256 hash of file>
}
```

Listing 2 Summary packet

The meaning of the different fields is the following:

- *created*: the timestamp at which the summary packet is created.
- *board_id*: the identifier of the station that generated the board messages included in the summary.

- *timestamps*: the array of ordered *block_ts* fields from the board messages included in the summary.
- *size*: the size in bytes of the measurement file.
- *number*: the number of board messages received from the station during one hour.
- *ip*: the IP address of the station sending the board messages.
- *hash*: the hash of the measurement file.

Database tables

Figure 2.29 shows the changes that were made to the database tables to support the measurement verification procedure via the blockchain. These allow for the reconstruction of the board messages coming from a station and, subsequently, for the computation of the hash of the measurement file related to each summary packet. The obtained hash values are compared to the ones contained in the summary packets received from the Blockchain Intermediary. Finally, the hash of each summary packet can be computed to check its presence in the blockchain.

The *PACKET_MEASURE* table substitutes the *MEASURE_TABLE* by adding information about the *PACKET* (board message) that contains the measurement and the position of the measurement inside the packet. The *PACKET* table, instead, represents a board message coming from a station. It references via foreign key the *BOARD* table representing the station, and contains the timestamp at which the board message is sent (*block_ts*).

The *PACKET_SUMMARY* table stores the the received summary packets. The *PACKET_CONNECTION* table, instead, is used to link the *PACKET_SUMMARY* to the contained *PACKETs*. This relation is not implemented via foreign key, because board messages and summary packets can arrive in any order and at different times. Therefore, *PACKET_CONNECTION* stores the timestamp of each board message contained in the summary. The timestamp, together with the station identifier stored in the *PACKET_SUMMARY* table, uniquely identifies the board message inside the *PACKET* table. Finally, all *PACKET*s contained in the *PACKET_SUMMARY* can be re-ordered using the *packetPosition* field stored in the *PACKET_CONNECTION* table.

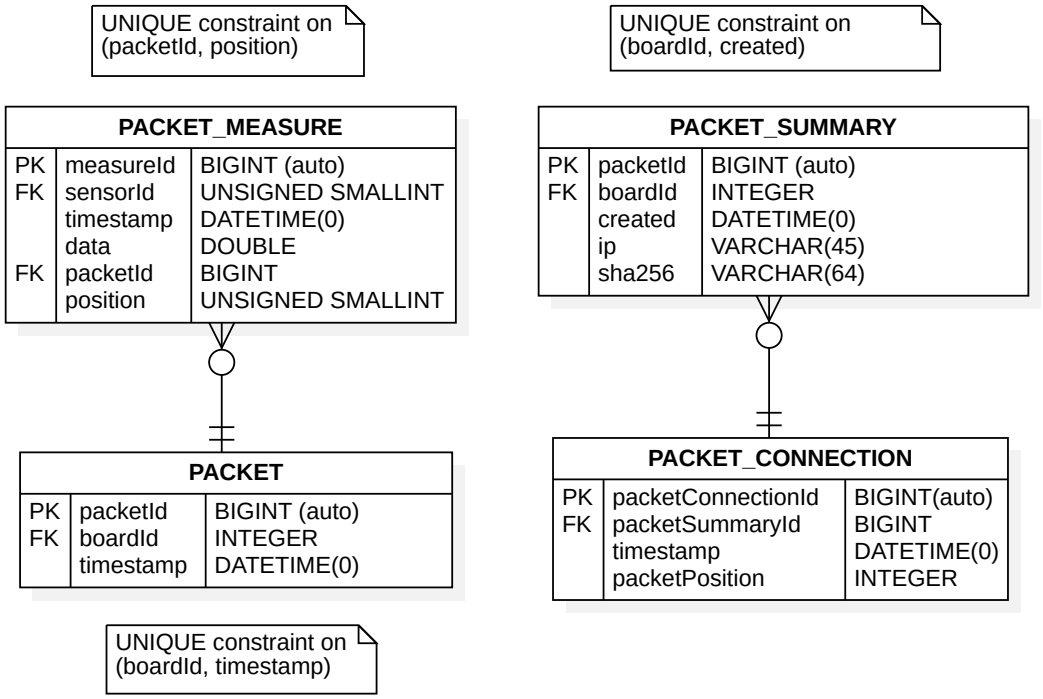


Fig. 2.29 ER diagram of tables for the crossoroad experiment [7].

Chapter 3

Security in Intermittent Computing Systems

3.1 Introduction

Intermittent computing (ImC) is a computation paradigm for systems that operate by using unreliable energy sources. More specifically, it involves checkpointing the state of the system before power loss. It is often adopted by IoT sensors that are powered via energy harvesters, which extract the required energy from the environment in which they are located. The objective is to reach true pervasivity and scalability of IoT deployments. Being often unattended and supporting limited security features, ImC devices are an easy target for attackers.

This work presents a utility to securely save and restore the state of an ImC device. The solution is implemented on a microcontroller (MCU) supporting ARM TrustZone for Cortex-M, a Trusted Execution Environment (TEE) for low-power MCUs. Finally, the utility is evaluated in terms of computational overhead and device lifetime and compared to other state-of-the-art solutions. Results are published in [11].

3.2 Background

This section presents a background on concepts and technologies adopted in this work.

3.2.1 Intermittent computing

Intermittent computing (ImC) systems are characterized by short periods of program execution separated by reboots [29]. ImC devices are usually powered by energy harvesters (EH), which extract energy from the surrounding environment. Typical sources are solar radiation, wind, vibrations, and electromagnetic signals. Temporary energy storage solutions, such as capacitors and super-capacitors, are usually preferred to traditional batteries. The reason for this choice is to increase device lifetime and reduce the maintenance effort, enabling a deploy-and-forget approach. In addition, batteries can be harmful to the surrounding environment if not properly maintained.

Due to the lack of permanent and high-capacity energy storage, these devices operate according to the cycle depicted in Figure 3.1. The device starts off with an empty energy buffer, which is replenished by the harvesting circuit. When the buffer is full, the system powers on to perform computation, draining the available energy. Eventually, the device shuts off again, and the cycle restarts.

In this context, the objective of ImCs is to ensure the forward progress of computation. This requires preserving the volatile state of the device between power losses. Different challenges are encountered, such as being able to track time, ensure atomic execution of tasks, prevent wasted computation, and avoid data inconsistencies due to uncompleted operations. Efficient intermittent computing solutions need to limit the energy overhead required for the creation and restoration of checkpoints.

State-of-the-art

Solutions [8, 29] are investigated from either the software or hardware points of view. From the hardware side, different non-volatile memory (NVM) technologies, such as FRAM, STT-RAM, PLM, and RRAM, are exploited. Caching solutions are introduced to reduce energy overhead related to frequent memory access. Non-

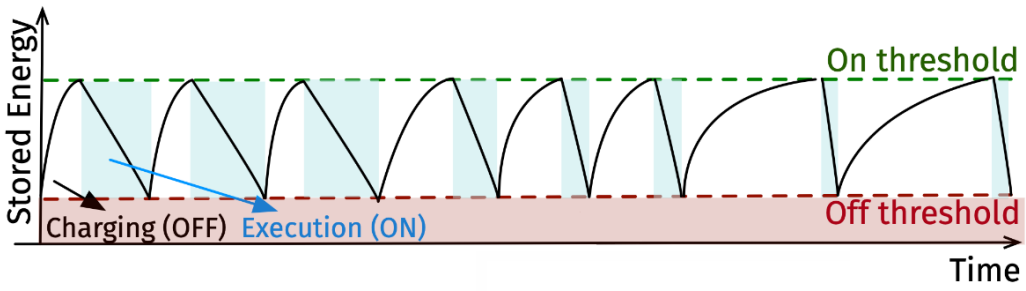


Fig. 3.1 Charge-operated-dye cycle [8].

volatile processors (NVPs) and application-specific integrated circuits (ASICs) are also being studied and developed. Finally, time-tracking solutions are implemented by measuring the discharge time of dedicated capacitors.

From the software side, instead, the focus is on developing efficient techniques to persist the state of the device, while maintaining atomicity and data consistency. Apart from standard checkpoints, task-based execution models are common: the software program is subdivided into atomic tasks whose execution does not exceed the energy available in the capacitor. Information is then exchanged between consecutive tasks using non-volatile channels. Ad-hoc compiler techniques are adopted to perform task subdivision, in order to reduce the overhead for the programmer. Deciding when to checkpoint is also a common problem. Static approaches exploit compiler optimization to identify possible checkpoint locations in the code. Dynamic solutions, instead, either perform checks on the remaining energy level or use timers that are adjusted via a trial-and-error process. Approximate computing solutions reduce the precision of the computation as a trade-off for energy-efficiency gains [153].

Platforms

The main limitation of custom hardware solutions is that their development is often limited to only research prototypes. Therefore, it is unlikely that they reach widespread adoption. However, some MCU platforms that support intermittent computing features are available on the market. The MSP430FR family of microcontrollers from Texas Instrument (TI) is the one that is most commonly used [29]. MSP430FR devices feature integrated FRAM memory and provide the Compute

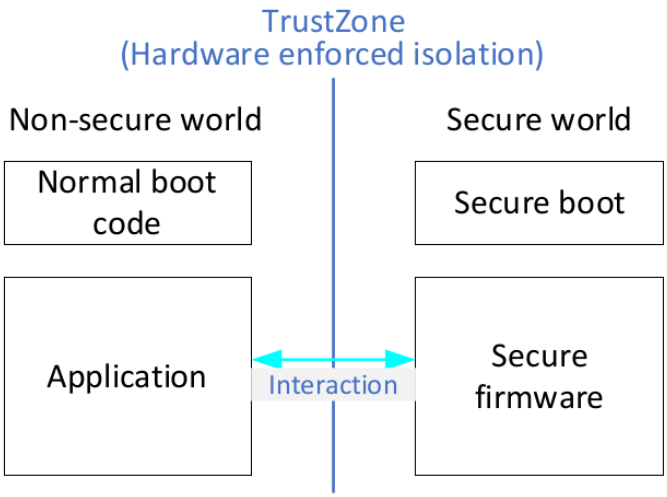


Fig. 3.2 TrustZone for Cortex-M isolation [9].

Through Power Loss (CPTL) [154] utility for persisting the state of CPU registers and peripherals.

3.2.2 ARM TrustZone for Cortex-M

ARM TrustZone for Cortex-M [62] is a Trusted Execution Environment (TEE) [155] for MCUs. It provides hardware isolation by dividing the memory of the device into two worlds: secure (S) and non-secure (NS). Code residing in non-secure world can only access non-secure memory, while code residing in secure world can access all the memory of the device [156]. Figure 3.2 represents these concepts.

At boot, the flow of execution starts from the secure world, which configures the TrustZone isolation before jumping to non-secure. Then, non-secure software can access secure services, implemented as functions, via specific entry points. Entry points are located in non-secure callable (NSC) memory, which is specifically reserved for this functionality. Trusted Firmware-M [157] provides a reference implementation of secure services.

The isolation provided by TrustZone is orthogonal to the one that is traditionally achieved via memory protection units (MPUs), which are managed by privileged code (e.g., Real-Time operating systems). In fact, privilege levels are usually available for both worlds, and the MCU can be independently configured in each of them [158].

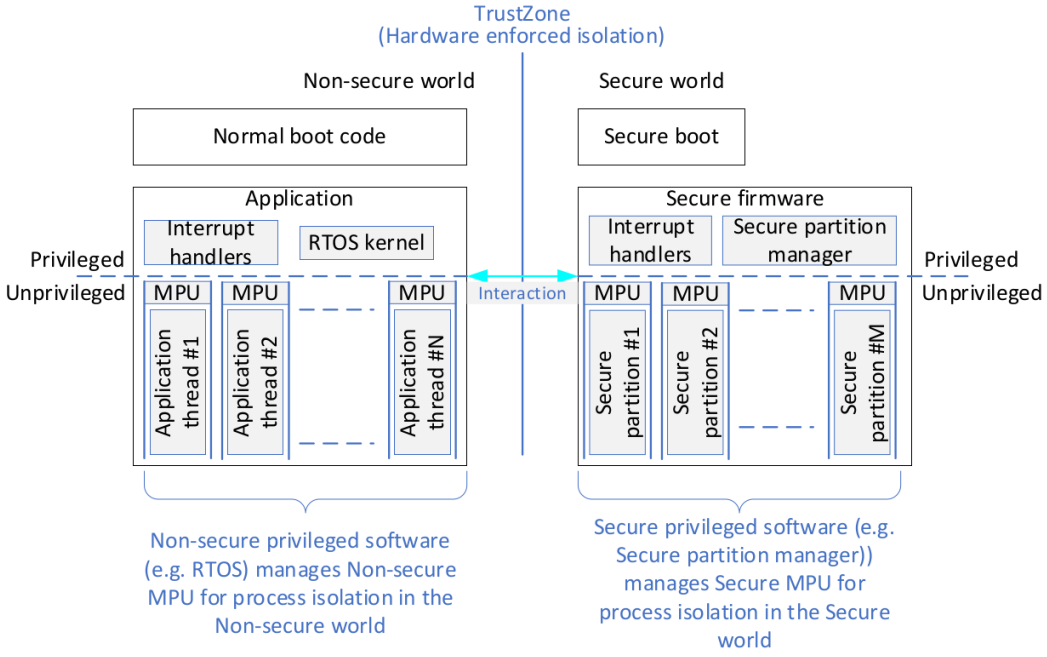


Fig. 3.3 Example of software architecture with Trusted Firmware-M [9].

Consequently, TrustZone prevents attackers who gain access to privileged execution from controlling the entire device. If this happens, secure world's data and services are still protected by the additional isolation. Finally, it also has the advantage of reducing the size of the Trusted Computing Base (TCB), which only includes secure-world software. These concepts are highlighted in Figure 3.3.

Implementation details

Two main components, residing in the Cortex-M core, are in charge of managing the TrustZone isolation: the Implementation Defined Attribution Unit (IDAU) and the Security Attribution Unit (SAU) [159]. The two units are used to specify the security attribution (S, NS, or NSC) of address space regions. The IDAU configuration is fixed in hardware by the MCU vendor, while the SAU can be programmed through code. The SAU configuration is usually performed at boot in secure world. If the two units specify a different security attribution for the same address space region, the stricter prevails. IDAU and SAU also block non-secure code from accessing secure regions. In all other cases, they mark the bus transaction with the security attribute of the accessed region.

In TrustZone systems, due to hardware design constraints, each peripheral is mapped twice in the address space (aliasing) to reside in both secure and non-secure regions, according to the default partition defined by the IDAU. Therefore, peripherals must be individually configured to only accept secure or non-secure transactions (not both). This configuration can only be carried out via secure code. Memory peripherals, such as FLASH and SRAM, can also be split between the two worlds. Access control is implemented by gates positioned in front of the peripherals, which check the security attribute of the incoming transaction. The design of the gates and of the hardware components for their configuration is left to MCU vendors.

Another problem that needs to be addressed is the presence of multiple bus masters, such as DMA controllers. IDAU and the SAU, residing in the MCU core, are bypassed by these hardware components. Therefore, they could easily be used to access secure memory from non-secure world. One of the solutions is to exclusively assign these masters to either secure or non-secure world. In this way, the issued transactions are marked according to their security attribution and can be filtered by the gates positioned in front of the peripherals.

3.3 State of the Art

Checkpointing solutions for ImC systems are extensively covered in the literature [160–166]. However, the security aspect is often overlooked. The work presented in [167] discusses security vulnerabilities related to checkpoints. The attacker is assumed capable of reading the internal memory of the device, an MSP430FR. This is achieved via an unprotected JTAG interface or by adopting sophisticated chip probing techniques. Different attacks targeting the AES128 library of the system are presented, leading to the retrieval of the secret key.

The Secure Intermittent Computing Protocol (SICP) [168, 169] acted as a starting point for the work presented in this chapter. SICP guarantees the integrity and freshness of checkpoints while also providing confidentiality to data specified by the user. In addition, it ensures the atomicity of checkpointing operations to avoid leaving the system in an undefined state. The protocol is implemented on an MSP430FR microcontroller, using the available FRAM as checkpoint storage. The attacker model allows to read and write the internal memory of the device. For this reason, the secret key and the nonces of the checkpoint must be stored in tamper-free

memory. However, tamper-free memory is not available on the device, and, instead, it is simulated using Intellectual Property (IP) encapsulation. This security feature, managed by the MPU, defines a memory section where the contained data can only be accessed by the code executed from the section itself. In addition, the full security chain of the platform is not considered in this work. For example, even in the presence of tamper-free memory, an attacker with read and write capabilities on the internal memory could modify the firmware to access or use the stored secrets. Only secure boot or checkpointing services implemented inside tamper-free memory could solve this problem.

Another interesting work is the one presented in [170], which exploits TrustZone for Cortex-M to persist and secure the state of non-secure world. This is achieved by copying the checkpoint inside FLASH memory residing in secure-world. Since the attacker model does not allow to read and write the internal memory of the device, the isolation provided by TrustZone protects the checkpoint from malicious firmware running in non-secure world. However, a full checkpoint utility for saving and restoring the state of non-secure world is not provided. In addition, FLASH memory only endures a limited amount of erase cycles, greatly reducing the lifetime of the system.

3.3.1 Contributions

This work addresses the combined limitation of previous works, proving the following contributions:

- A complete checkpoint utility is implemented for a TrustZone-enabled MCU, allowing for the saving of the state of secure world (non-secure world is currently not checkpointed).
- TrustZone for Cortex-M is used to protect the checkpoint utility and the related data from malicious firmware running in non-secure world.
- The full security chain of the target platform is considered, and no missing hardware features are required.
- Device lifetime is preserved by using FRAM memory to store the checkpoints.

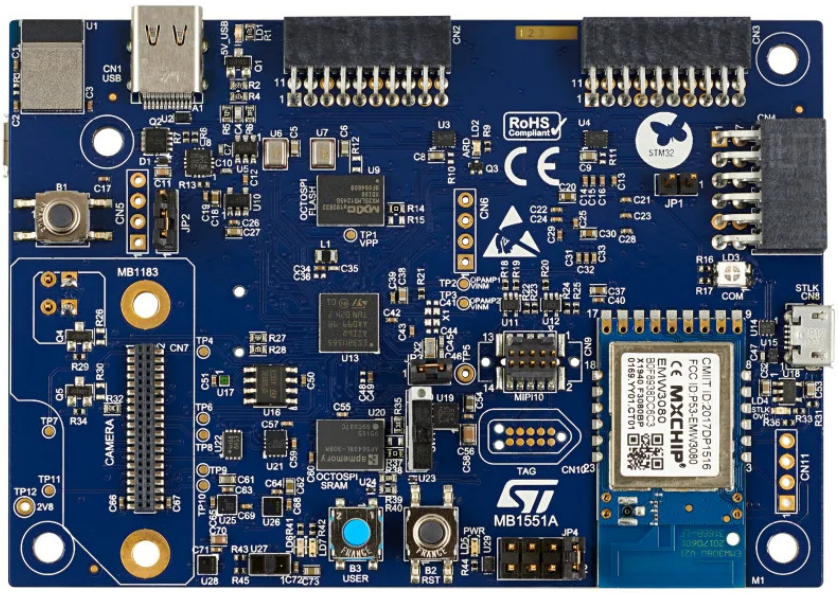


Fig. 3.4 B-U585I-IOT02A Discovery kit [10].

3.4 Proposed solution

This section introduces the target platform, establishes a security objective, defines the attacker model, and describes the implementation of the checkpoint utility.

3.4.1 Target platform

The target platform that is selected for this work is the STM32U5 MCU from STMicroelectronics. Specifically, the solution is developed on the B-U585I-IOT02A Discovery kit, which integrates an STM32U585AII6Q microcontroller. Figure 3.4 shows a picture of the board.

The MCU core is an ARM Cortex-M33 with 2 Mbytes of internal flash memory divided into two banks, and 786 Kbytes of SRAM. In addition, the STM32U5 MCU integrates several security features (only the ones relevant to this work are listed):

- ARM TrustZone for Cortex-M.
- MPU for privilege-based memory isolation.
- Random Number Generator (RNG).
- AES module supporting AES-GCM with 256-bit key size.
- Secure AES module (SAES) for side channel protection and key sharing features.
- Readout protection (RDP) configurable on three levels. This is used to disable static board configuration (option bytes) and debug features, while also fixing the boot entry point.

The 8 kB pages of the internal FLASH memory only endure 10.000 erase cycles, with 256 kB for each bank that can reach 100.000. If one checkpoint per capacitor discharge is considered, the internal FLASH cannot be used to save the device state without harming the lifetime of the system. For this reason, an external FRAM memory is connected to the board via the SPI interface. The memory chip that is adopted is the Infineon CY15B102QN. It features a 40MHz SPI interface, 2MB of storage, low-power modes, and can endure up to 10^{15} write cycles. However, as it will be discussed later, the internal FLASH memory is still used to store the nonces of the checkpoints, affecting the lifetime of the device.

3.4.2 Security objectives

This work wants to achieve the following security objectives:

- **Information security**: integrity, authenticity, and confidentiality of data stored in checkpoints.
- **Freshness**: the creation of a new checkpoint invalidates the previous ones, preventing replay attacks.
- **Atomicity**: the system cannot enter an undefined state due to a power loss during checkpoint creation.
- **Unclonability**: checkpoint information cannot be used to clone the device.

Similarly to the other state-of-the-art solutions, forward progress is not preserved in the presence of an attacker.

3.4.3 Attacker model

The attacker is assumed to have physical access to the device, granting him the following abilities:

- The attacker can start and stop the device by controlling the supplied power.
- The attacker can run malicious code in non-secure work.
- The attacker can read and write content on the external FRAM memory.
- The attacker can connect to unprotected debug ports.

The attacker is also subjected to the following limitations:

- The attacker cannot read the internal memory of the device (e.g., via expensive chip-probing techniques).
- The attacker cannot run arbitrary code in secure world.
- Side channel attacks (e.g., differential power analysis) are not considered.

3.4.4 Implementation

This section describes the implementation of the secure checkpointing utility for saving the state of secure world. The utility is provided as MIT-licensed open source code [171].

3.4.5 Memory layout

A customized memory layout is created by modifying the linker script of the secure-world image. The result is shown in Figure 3.5.

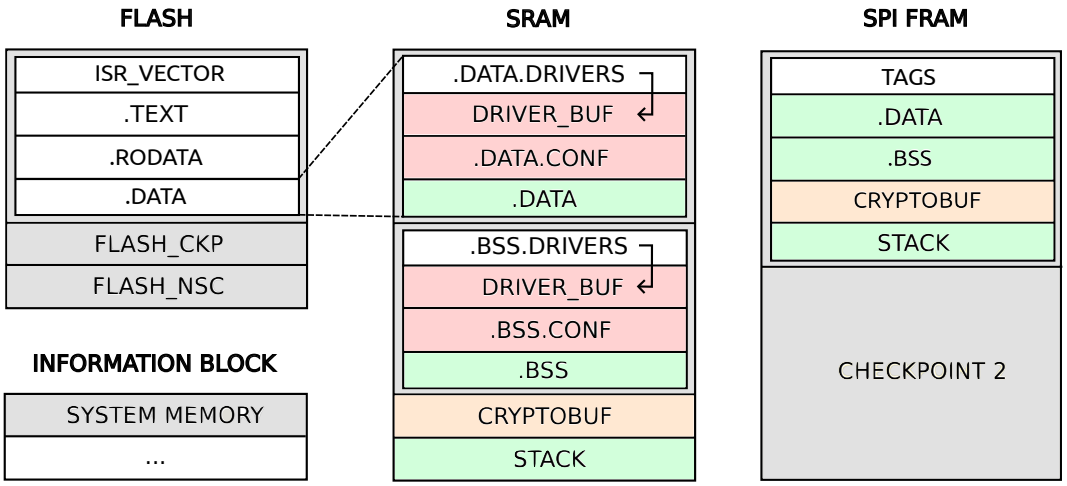


Fig. 3.5 Custom memory layout, showing associated data (green), plaintext (red) and ciphertext (orange) [11].

The structure of the FLASH memory attributed to secure world is changed by reserving space for the *FLASH_CKP* area. This area is used to store the nonces of the checkpoints. The size is set to 256 kB, because it is the maximum amount of FLASH memory that can endure up to 100.000 erase cycles.

The structure of the *data* and *bss* sections is also modified. The *data* section, initially stored in FLASH memory, is copied inside the SRAM at boot. It contains the initialized global variables (both global and static) of the program. The *data* section is now divided into different parts. *data.drivers* contains the variables used by the drivers (HAL library) of the board. Since its content may be changed by the checkpoint routine itself, *driver_buf* is created to backup its state. This prevents data inconsistencies during checkpoint creation and restoration. *data.conf*, instead, stores variables for which the programmer wants to preserve confidentiality. Variables can be assigned to these areas via compiler attributes (*__attribute__((section("name")))*). Instead, for third-party code or binary files, it is possible to place the variables of the compiled object files into a specific section via the linker script.

A similar subdivision is also performed for the *bss* section, which contains uninitialized variables. Finally, the *cryptobuf* section stores the ciphertext of confidential memory areas. The *driver_buf* area of both the *data* and *bss* sections is considered confidential. The content of the stack, instead, is not encrypted. This is a limitation of the current implementation.

The external SPI memory is divided into two parts to store two checkpoints. This is needed to prevent corruption of the previous checkpoint if a power loss occurs during the creation of the new one.

Finally, Figure 3.5 shows the information block. This is a memory area that resides in the first flash bank. The first 32 kB of the information block contains system memory, which is an immutable section reserved for STMicroelectronics. The device's 256-bit Root Hardware Unique Key (RHUK) is stored in this section. The key is not accessible from code, and it is wired directly into the SAES crypto unit. The RHUK is only used to generate Derived Hardware Unique Keys (DHUKs), which are the ones actually used for cryptographic operations. The key derivation procedure is parametrized by the TrustZone attribution of the SAES module (secure or non-secure) and the key utilization mode (normal, wrapped, shared).

3.4.6 Checkpoint utility

The checkpoint utility is constituted by two main operations: SAVE and RESTORE. The former is in charge of creating and persisting the checkpoint, while the latter is used to restore the state of the device. SAVE and RESTORE can be called from both privileged and non-privileged software via system calls, making them always run in privileged mode. Upon exception entry, the hardware saves the relevant CPU registers on the stack, allowing SAVE to preserve the CPU state by only persisting the stack. The SAVE routine can be called either periodically or just before a power loss. Finally, when RESTORE finishes its operations, it simulates an exception exit from the SAVE system call that created the checkpoint.

SAVE

The pseudocode of the save routine is shown in Algorithm 1. The first task that is carried out by the routine is to copy the global and static variables used by the device drivers (STM32 HAL Library). This is needed because SAVE uses the drivers to perform its operations, and could checkpoint them in an undefined state. For this reason, the *data.drivers* and *bss.drivers* sub-sections are copied in the respective *driver_buf*s (1-2).

Algorithm 1 SAVE [11]

1: $driver\_buf_{.data} \leftarrow .data.drivers$
2: $driver\_buf_{.bss} \leftarrow .bss.drivers$
3: $IV \leftarrow generateIV()$
4: $ad \leftarrow .data$
5: $pt \leftarrow driver\_buf_{.data}|.data.conf$
6: $tag_1, ct \leftarrow AES\_GCM(IV, ad, pt)$
7: $cryptobuf.append(ct)$
8: $ad \leftarrow .bss$
9: $pt \leftarrow driver\_buf_{.bss}|.bss.conf$
10: $tag_2, ct \leftarrow AES\_GCM(IV+1, ad, pt)$
11: $cryptobuf.append(ct)$
12: $ad \leftarrow stack$
13: $tag_3 \leftarrow AES\_GCM(IV+2, ad)$
14: $tags \leftarrow tag_1|tag_2|tag_3$
15: $writeSPI(tags|.data|.bss|cryptobuf|stack, ckp)$
16: $writeFlash(IV, ckp)$
17: **if** $firstNonceInNewPage()$ **then**
18: $\quad erasePreviousPage()$
19: **end if**
20: $ckp.toggleLocation()$
21: $flash.setNextNonceSlot()$

Then, the Random Number Generator (RNG) is used to generate the 96-bit initialization vector (IV) for the cryptographic primitive (3), i.e., AES-GCM. This value also acts as part of the nonce for the checkpoint to ensure freshness.

In the next steps, AES-GCM is called three times. The first invocation is used to authenticate the *data* section, while also providing confidentiality to *data.conf* and *driver_buf* (4-6). Information that only requires authentication is provided to the primitive as associated data (*ad*). Confidential information, instead, constitutes the plaintext (*pt*). The results of this operation are the authentication tag of the entire *data* section and the ciphertext of its confidential sub-sections, which is stored in the *cryptobuf* (7).

Similar operations are repeated for the *bss* section (8-11) and the stack (12-13). For what concerns the stack, its content is currently only authenticated. IV is increased at each invocation of the cryptographic primitive, since reusing the same value breaks the security properties of AES-GCM.

At this point, the checkpoint is stored on the external memory (14-15) at the current checkpoint location (*ckp*). It is composed by the concatenation of the three authentication tags, the non-confidential part of the *data* and *bss* sections, the *cryptobuf* and the *stack*.

A 128-bit nonce is created by concatenating the 96-bit IV and the 32-bit address, which identifies the location of the checkpoint on the external memory (*ckp*). Subsequently, this value is written to FLASH, inside the *FLASH_CKP* region (16). As soon as the nonce is saved, the older checkpoint is invalidated. Due to the limitations of FLASH memory, nonces are not overwritten, since this would require the erasure of the entire flash page, but they are appended one after the other. Obviously, only the latest nonce is considered valid. It is important to note that including the address of the checkpoint in the nonce does not require additional space, since the minimum write size for the FLASH memory is 128 bits.

In the final steps, the routine checks whether the new nonce was written on a new flash page (17-19). If this is the case, it erases the previous one. This maintains a circular buffer to store the new nonces. Finally, the locations to store the next nonce and the next checkpoint are updated (20-21).

Two important considerations are required. The first is that the MCU vendor does not specify whether memory writes to FLASH memory are atomic. So, in case of a power failure, it may be possible that the stored nonce is corrupted. However, this does not represent a security vulnerability because it only results in the invalidation of all checkpoints, requiring the system to restart the computation. Nonetheless, this could impede forward progress during normal operations.

The second consideration is related to the use of a random number as IV, instead of a simple counter. Using a counter would introduce a security vulnerability, because the attacker can read the checkpoint stored on the external memory and stop the computation before the nonce is written to flash. By restarting the devices multiple times, the attacker could collect multiple AES-GCM messages generated with the same IV. This information can be used to break the security properties of AES-GCM.

RESTORE

The pseudocode of the RESTORE routine is shown in Algorithm 2. The routine starts by loading the key used to secure the checkpoints (checkpoint key) inside the

Algorithm 2 RESTORE [11]

1: $unwrapSecretKey(key)$
2: $IV, ckp \leftarrow readFlash()$
3: $.data, .bss, cryptobuf \leftarrow readSPI()$
4: $ad \leftarrow .data$
5: $ct \leftarrow cryptobuf.at(driver\_buf_{.data}|.data.conf)$
6: $tag_1, pt \leftarrow AES\_GCM^{-1}(IV, ad, ct)$
7: $driver\_buf_{.data}|.data.conf \leftarrow pt$
8: $ad \leftarrow .bss$
9: $ct \leftarrow cryptobuf.at(driver\_buf_{.bss}|.bss.conf)$
10: $tag_2, pt \leftarrow AES\_GCM^{-1}(IV, ad, ct)$
11: $driver\_buf_{.bss}|.bss.conf \leftarrow pt$
12: $stack \leftarrow readSPI()$
13: $ad \leftarrow stack$
14: $tag_3 \leftarrow AES\_GCM^{-1}(IV, stack)$
15: $tags \leftarrow readSPI()$
16: **if** $tags! = tag_1|tag_2|tag_3$ **then**
17: $abort()$
18: **end if**
19: $.data.drivers \leftarrow driver\_buf_{.data}$
20: $.bss.drivers \leftarrow driver\_buf_{.bss}$
21: **if** $firstNonceInNewPage()$ **then**
22: $erasePreviousPage()$
23: **end if**
24: $ckp.toggleLocation()$
25: $flash.setNextNonceSlot()$

AES module (1). This operation is also performed at the first boot of the device, since the key is also needed by the SAVE routine. The checkpoint key is normally stored as a global variable inside the software image of secure world, after being encrypted with the DHUK. The SAES unit, configured in shared mode, performs both key decryption and loading.

The next step is to retrieve the most recent nonce from FLASH memory (2). The first 128 bits of each FLASH page inside *FLASH_CKP* are read to identify which is the page that contains it. Then, a binary search is conducted inside the page to find the latest nonce.

The *cryptobuf* and the non-confidential parts of the *data* and *bss* sections are read from external memory and restored (3). The *cryptobuf* is decrypted, and the

obtained plaintext is stored at its original location (4-11). This includes confidential information from both the *data* and *bss* sections. The authentication tags for both sections are also computed. Similarly, the stack is restored from external memory, and its authentication tag is generated (12-14).

The three authentication tags are compared with the ones stored on the external memory, to see whether the checkpoint was modified (15-18). Subsequently, the driver buffers are copied inside their original locations (19-20). At this point, if the nonce of the restored checkpoint is in a newly created flash page, the routine checks if the previous page was erased (21-23). If this is not the case, e.g., for a power loss during the SAVE routine, it is erased. Finally, the locations to store the next nonce and the next checkpoint are updated (24-25).

3.4.7 Security chain

The security of the device relies on the RHUK, which is stored in system memory (flash bank 1). This is the reason why the attacker model excludes the possibility of reading and writing the internal memory of the device. If this was allowed, the RHUK and the encrypted checkpoint key could be extracted. Then, the checkpoint key could be decrypted and used to access the confidential information stored in the checkpoints. In any case, due to the high cost of the instrumentation required to perform chip-probing attacks, this assumption can be considered reasonable.

The second step is to provide isolation between the secure and non-secure worlds. TrustZone is set up in two phases. The first requires the configuration of option bytes, which are stored in the information block (first flash bank) and can be accessed via debug features. Once they are configured, debug access must be prevented by setting the Readout Protection (RDP) level to two. This also disables debug access to the firmware. In the second phase, the device boots in secure world, and the secure firmware finalizes the configuration of Trustzone.

At this point, malicious non-secure software is isolated from secure word. This provides integrity, authenticity, and confidentiality of both data and software residing in secure world. This includes the checkpoint key, which is also encrypted, and the nonces. Checkpoints are secured and stored on the external SPI memory by the SAVE routine. Since both the key and the nonces are protected, the authenticity, integrity, confidentiality, and freshness of checkpoints are ensured.

Finally, the SPI interface connected to the external memory can be assigned to secure world to make it inaccessible from non-secure software. Secure boot services could also be configured, but they would increase the energy overhead required during startup.

3.5 Evaluation

In this section, the checkpoint utility is evaluated in terms of device lifetime and computational overhead.

3.5.1 Lifetime evaluation

Device lifetime is an important aspect for intermittent computing systems, since their objective is to achieve a deploy-and-forget operational paradigm. In this work, this is addressed by using external FRAM memory to store the checkpoints. Differently for FLASH, this memory technology is not so severely limited in terms of wear, allowing it to reach a much higher number of write cycles. However, in the proposed solution, internal FLASH is still used to store nonces, which reduces the lifetime of the device. By considering a nonce buffer of 256 kB, which is the maximum FLASH size that can sustain up to 100.000 erasures, a 128-bit nonce, and one checkpoint per second, the lifespan of the device can reach 52 years.

3.5.2 Overhead evaluation

The objective of this evaluation is to measure the overhead introduced by the creation and restoration of checkpoints. This is important because the energy used by SAVE and RESTORE is subtracted from the domain application.

As a first step, the total time overhead of each routine is measured. This is done by varying the memory size of the secure application that is checkpointed. The test is repeated two times, first by considering the data of the application as non-confidential, then by making it entirely confidential. In the second step, instead, the contribution of each operation inside the routine is separately evaluated, in order

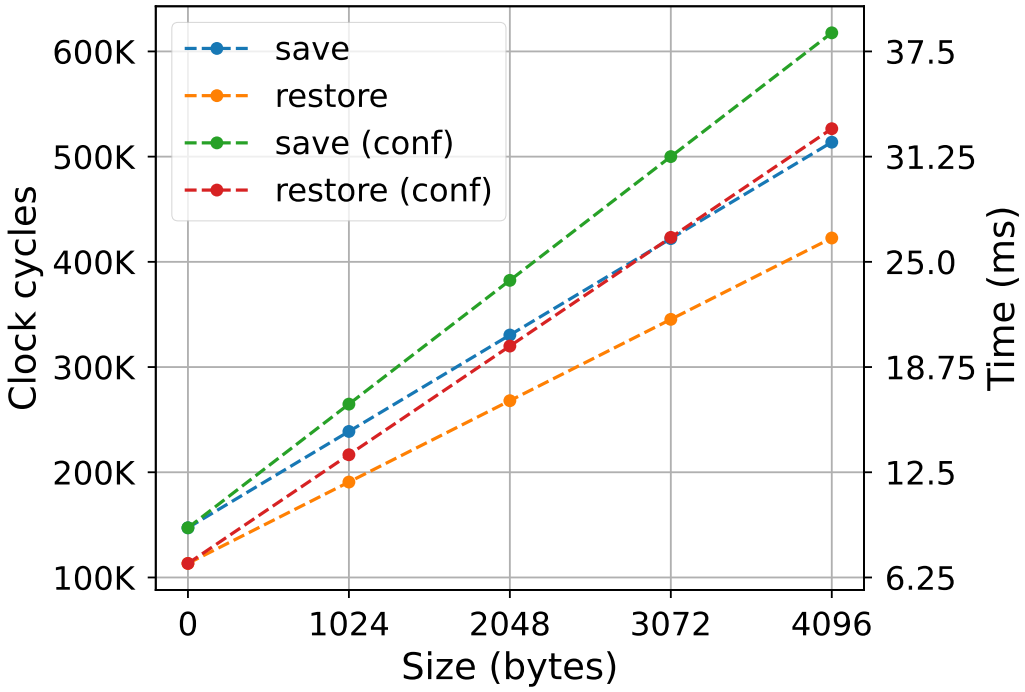


Fig. 3.6 Comparison of SAVE and RESTORE overhead when varying application size and introducing confidentiality [11].

to understand which are the most demanding ones. Also, in this case, both the non-confidential and fully confidential scenarios are tested.

The overhead is measured in terms of clock cycles, which are also converted to milliseconds given the 160 MHz clock frequency of the MCU. To achieve this, the DWT cycle counter is read before and after the monitored operation. Then, a message containing the difference is sent via the UART interface. Measurements are repeated for a minimum of ten times and then averaged.

Figure 3.6 shows the results of the first evaluation. The memory size of the domain application is simulated by an array of variable sizes placed inside the *bss* section. A null memory size indicates the overhead for saving the state of the checkpoint utility itself. It can be seen that the overhead increases linearly with application size. In addition, adding confidentiality reduces the performance of both routines. This is due to the design characteristics of AEG-GCM, which is slower when the data needs to be encrypted instead of only authenticated. SAVE also appears to be slower than RESTORE.

Table 3.1 Benchmark of SAVE [11].

Operation	Cycles	Time (ms)
Authenticated encryption	51777 (103745)	3,236 (6,484)
Copy drivers	11366	0,710
Write nonce	20409 (20408)	1,276
IV generation	451	0,028
Page erase*	274	0,017
SPI R/W	245883 (245884)	15,368
Others	275 (277)	0,017
Total	330435 (382405)	20,652 (23,9)

Table 3.2 Benchmark of RESTORE [11].

Operation	Cycles	Time (ms)
Authenticated encryption	52077 (104044)	3,255 (6,503)
Copy drivers	11325	0,708
Decrypt key	2697	0,169
Read nonce	586	0,037
Page erase*	274	0,017
SPI R/W	200677 (200675)	12,542
Others	378 (334)	0,024 (0,021)
Total	268014 (319935)	16,751 (19,996)

The contribution of the single operations to the overhead of the routines is shown in Figures 3.1 and 3.2. The evaluations are performed considering an application size of 2kB. Values inside round brackets refer to the full-confidential scenario. It is important to mention that page erase operations are performed only when the page is completely filled with nonces. Therefore, their contribution is divided by the number of nonces that fit on a page, i.e., 512. Results show that the main contributions to the overhead are external memory access and authenticated encryption. A significant overhead also comes from copying the device drivers, but this is probably due to poor code optimization.

The main difference between the overhead of the two routines is given by external memory access and the programming of the nonce in FLASH (for the SAVE routine). For what concerns the effect of confidentiality, the only significant difference is given by the cryptographic primitive.

State-of-the-art comparison

It is interesting to compare the results obtained with those achieved by the SICP protocol on the MSP430FR. When providing no security and considering checkpoints with sizes ranging from 468 to 3198 bytes, the MSP430 takes less than one millisecond to perform both save and restore. Since variables are stored in FRAM, which is non-volatile, they don't need to be copied to another memory. The small overhead, instead, is probably due to saving the state of the peripherals and CPU registers. This is achieved via the CTPL utility provided by Texas Instruments.

When authentication is introduced, the overhead reaches 20-73 ms, which is similar to what is obtained in this work. However, the MSP430 does not have to perform a memory copy since it only computes and saves the authentication tag. If the contribution of SPI R/W operation is removed from the results achieved in this work, the STM32 hardware shows better performance.

When full confidentiality is introduced, the SICP overhead becomes of 68-380 ms. This is due to the additional cryptographic operations and the need to save both the ciphertext and the restored plaintext in memory.

As a final note, it is important to understand that time is not the best metric for the evaluation of the performance of these implementations. In fact, ImC systems are constrained in terms of energy. Therefore, the best solution is not the fastest but the one that requires less energy. In the presented scenario, the SM32U5 MCU runs at a frequency of 160 MHz, against the 8 MHz of the MSP430FR. This for sure leads to higher power consumption and wasted clock cycles while waiting for SPI and cryptographic operations to complete. Therefore, future work should focus on evaluating and comparing the efficiency of the proposed solution via power analysis.

3.6 Conclusions

Starting from the works on the SICP protocol [168, 169], a secure intermittent computing solution is developed for the ARM Cortex-M platform. TrustZone for Cortex-M is used to enforce isolation between a non-secure world, containing potentially vulnerable software, and secure work, containing trusted services, among which the checkpoint utility itself.

In order to guarantee the device's lifetime, an external SPI FRAM memory is used to store the checkpoints. This mitigates problems related to the wear of the internal FLASH memory. A small portion of the internal FLASH is still used to store nonces, which are needed to prevent the replay of older checkpoints. Fortunately, this only has a limited impact on the lifetime of the device.

The overhead introduced by the utility is evaluated and compared with the SICP implementation, reaching similar performance. Benchmarks also show that communication to the external memory and cryptographic operations provide the biggest contribution to the overhead.

The main limitation of this work is that checkpointing of non-secure world is not currently supported. Confidentiality is also not provided for the content of the stack. In addition, due to the fragmentation created by dynamic memory allocations, the state of the heap is not persisted. Similarly, the state of device peripherals is also not checkpointed. Future works will address these limitations, while also focusing on testing the power efficiency of the secure checkpoint utility in real use case scenarios.

Chapter 4

Conclusions

The work presented in this thesis investigates security and reliability in pervasive computing systems. Two main fields of application are considered: low-cost Particulate Matter (PM) monitoring with low-cost light-scattering sensors and secure intermittent computing (ImC) for energy harvesting devices. In the context of PM monitoring, the reliability aspect consists in evaluating and ensuring data quality. The security aspect, instead, is explored by designing an IoT infrastructure that provides integrity and authenticity verification of sensor measurements. For what concerns ImC systems, security is investigated by developing a software utility for the creation, restoration, and protection of checkpoints.

PM monitoring

Low-cost light-scattering sensors are often proposed as a solution for the creation of dense monitoring networks. However, due to intrinsic technological limitations, their measurements are often unreliable and imprecise. Multiple measurement campaigns are conducted by placing a multitude of these devices at an official monitoring site in the city of Turin, in Italy.

At first, data quality is evaluated by comparing them to the official monitoring network of fixed stations of the Metropolitan City of Turin. It is concluded that, for daily PM2.5 measurements, they have a correlation with the high-precision monitoring instruments at the same monitoring site, similar to the correlation between the different urban sites in the same area. Therefore, in this context, they provide

additional benefits if high-precision instruments are absent or very sparse. In addition, they are also evaluated on hour measurements, highlighting a loss of correlation at higher sampling rates.

A data processing pipeline that performs failure detection, filtering, and calibration of low-cost PM sensors is proposed and evaluated. Results show that data quality is improved with respect to the reference, considering multiple metrics.

The effect of introducing a duty cycle for low-cost PM sensor operation is analyzed in terms of measurement quality. Results show that, in a non-traffic area, the sampling frequency can be greatly reduced without affecting measurement quality w.r.t. to the official reference.

An IoT infrastructure for data transmission, storage, and visualization is developed. It also provides verification of measurement integrity and authenticity, by storing the hash of blocks of measurements inside a blockchain.

Secure ImC

A checkpointing utility for secure intermittent computing is realized, considering the entire security chain of the target platform and exploiting ARM Trustzone for Cortex-M functionalities. TrustZone is used to enforce isolation between a non-secure world, containing potentially vulnerable software, and secure work, containing trusted services, among which the checkpoint utility itself. In order to guarantee the device's lifetime, an external SPI FRAM memory is used to store the checkpoints. This mitigates problems related to the wear of the internal FLASH memory.

The overhead introduced by the utility is evaluated against the state-of-the-art, reaching comparable performance. Benchmarks also show that communication to the external memory and cryptographic operations provide the biggest contribution to the overhead. Limitations are still present, such as the lack of support for saving the state of non-secure world, and will be addressed in future works.

References

[1] ARPA Piemonte. La qualità dell'aria in Piemonte rapporto 2021. https://www.arpa.piemonte.it/pubblicazione/qualita-dellaria-piemonte-rapporto-2021.

[2] Filippo Gandino, Pietro Chiavassa, and Renato Ferrero. Measuring Particulate Matter: An Investigation on Sensor Technology, Particle Size, Monitoring Site. *IEEE Access*, 11:108761–108774, 2023. Conference Name: IEEE Access.

[3] Gustavo Ramirez-Espinosa, Pietro Chiavassa, Edoardo Giusto, Stefano Quer, Bartolomeo Montrucchio, and Maurizio Rebaudengo. Improving Data Quality of Low-Cost Light-Scattering PM Sensors: Towards Automatic Air Quality Monitoring in Urban Environments. *IEEE Internet of Things Journal*, pages 1–1, 2024. Conference Name: IEEE Internet of Things Journal.

[4] Bartolomeo Montrucchio, Edoardo Giusto, Mohammad Ghazi Vakili, Stefano Quer, Renato Ferrero, and Claudio Fornaro. A Densely-Deployed, High Sampling Rate, Open-Source Air Pollution Monitoring WSN. *IEEE Transactions on Vehicular Technology*, 69(12):15786–15799, December 2020. Conference Name: IEEE Transactions on Vehicular Technology.

[5] Ramirez Espinosa and Gustavo Adolfo. IoT for Urban Sustainability in Smart Cities. *Doctoral thesis Polito*, October 2023.

[6] Pietro Chiavassa, Filippo Gandino, and Edoardo Giusto. An investigation on duty-cycle for particulate matter monitoring with light-scattering sensors. In *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6, September 2021.

[7] Alberto Barbaro, Pietro Chiavassa, Virginia Isabella Fissore, Antonio Servetti, Erica Raviola, Gustavo Ramírez-Espinosa, Edoardo Giusto, Bartolomeo Montrucchio, Arianna Astolfi, and Franco Fiori. Data Acquisition, Processing, and Aggregation in a Low-Cost IoT System for Indoor Environmental Quality Monitoring. *Applied Sciences*, 14(10):4021, January 2024. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[8] Kasım Sinan Yıldırım. The today and future of intermittent computing for sustainable sensing. https://community.arm.com/arm-research/m/resources/990.

[9] Trusted firmware-m technical overview. https://www.trustedfirmware.org/docs/TrustedFirmware-MTechnicalOverviewQ1-2023.pdf.

[10] STMicroelectronics. B-U585I-IOT02A - Discovery kit for IoT node with STM32U5 series. https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html.

[11] Pietro Chiavassa, Filippo Gandino, Renato Ferrero, and Jan Tobias Mühlberg. Secure Intermittent Computing with ARM TrustZone on the Cortex-M. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 160–168, July 2024. ISSN: 2768-0657.

[12] WeiserMark. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, July 1999. Publisher: ACM PUB27 New York, NY, USA.

[13] IBM. What is the internet of things? https://www.ibm.com/topics/internet-of-things.

[14] Ritika Lohiya and Ankit Thakkar. Application Domains, Evaluation Data Sets, and Research Challenges of IoT: A Systematic Review. *IEEE Internet of Things Journal*, 8(11):8774–8798, June 2021. Conference Name: IEEE Internet of Things Journal.

[15] Duarte Dias and João Paulo Silva Cunha. Wearable Health Devices—Vital Sign Monitoring, Systems and Technologies. *Sensors*, 18(8):2414, August 2018.

[16] Energy Independence and Security Act of 2007 - TITLE XIII—SMART GRID.

[17] Wang Tong, Azhar Hussain, Wang Xi Bo, and Sabita Maharjan. Artificial Intelligence for Vehicle-to-Everything: A Survey. *IEEE Access*, 7:10823–10843, 2019. Conference Name: IEEE Access.

[18] Virginia Isabella Fissore, Silvia Fasano, Giuseppina Emma Puglisi, Louena Shtrepi, and Arianna Astolfi. Indoor Environmental Quality and Comfort in Offices: A Review. *Buildings*, 13(10):2490, October 2023. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[19] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, 1(1):22–32, February 2014. Conference Name: IEEE Internet of Things Journal.

[20] Yihan Xu, Mikhail Smirnov, Michael C. Kohler, Ziqian Dong, Reza K. Amineh, Fang Li, and Roberto Rojas-Cessa. A Survey of Wireless Soil Sensing Technologies. *IEEE Access*, 12:12010–12038, 2024. Conference Name: IEEE Access.

[21] Daniel Caballero, Rosalba Calvini, and José Manuel Amigo. Hyperspectral imaging in crop fields: precision agriculture. In José Manuel Amigo, editor, *Data Handling in Science and Technology*, volume 32 of *Hyperspectral Imaging*, pages 453–473. Elsevier, January 2019.

[22] Chen Yang, Shulin Lan, Zhiheng Zhao, Mengdi Zhang, Wei Wu, and George Q. Huang. Edge-Cloud Blockchain and IoE-Enabled Quality Management Platform for Perishable Supply Chain Logistics. *IEEE Internet of Things Journal*, 10(4):3264–3275, February 2023. Conference Name: IEEE Internet of Things Journal.

[23] Cisco. What Is Industrial IoT (IIoT)? https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-industrial-iot.html.

[24] IBM. Getting started with IoT development. https://developer.ibm.com/learningpaths/iot-getting-started-iot-development.

[25] Lalit Chettri and Rabindranath Bera. A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems. *IEEE Internet of Things Journal*, 7(1):16–32, January 2020. Conference Name: IEEE Internet of Things Journal.

[26] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018.

[27] Google. PaaS vs IaaS vs SaaS: What's the difference? https://cloud.google.com/learn/paas-vs-iaas-vs-saas.

[28] IEEE Internet of Things Journal. https://ieee-iotj.org/.

[29] Sumanth Umesh and Sparsh Mittal. A survey of techniques for intermittent computing. *Journal of Systems Architecture*, 112:101859, January 2021.

[30] Zigbee | Complete IOT Solution. https://csa-iot.org/all-solutions/zigbee/.

[31] Zigbee FAQs | Frequently Asked Questions. https://csa-iot.org/all-solutions/zigbee/zigbee-faq/.

[32] Thread Group. https://www.threadgroup.org/.

[33] OpenThread. What is Thread? https://openthread.io/guides/thread-primer.

[34] Thread Group. Thread network fundamentals. https://www.threadgroup.org/Portals/0/documents/support/Thread%20Network%20Fundamentals_v3.pdf.

[35] Bluetooth. The Bluetooth Low Energy Primer, May 2022. https://www.bluetooth.com/wp-content/uploads/2022/05/the-bluetooth-le-primer-v1.2.0.pdf.

[36] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal*, 6(5):8182–8201, October 2019. Conference Name: IEEE Internet of Things Journal.

[37] Wikipedia. Wi-Fi. https://en.wikipedia.org/w/index.php?title=Wi-Fi.

[38] Wi-Fi Alliance. https://www.wi-fi.org/.

[39] Wikipedia. Ethernet. https://en.wikipedia.org/w/index.php?title=Ethernet.

[40] Cheng Chen, Xiaogang Chen, Dibakar Das, Dmitry Akhmetov, and Carlos Cordeiro. Overview and Performance Evaluation of Wi-Fi 7. *IEEE Communications Standards Magazine*, 6(2):12–18, June 2022. Conference Name: IEEE Communications Standards Magazine.

[41] Semtech. LoRa PHY. https://www.semtech.com/lora/what-is-lora.

[42] LoRa Alliance. https://lora-alliance.org/.

[43] Sigfox. Sigfox 0G Technology. https://www.sigfox.com/.

[44] Alexandru Lavric, Adrian I. Petrariu, and Valentin Popa. Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions. *IEEE Access*, 7:35816–35825, 2019. Conference Name: IEEE Access.

[45] 3GPP - The Mobile Broadband Standard. https://www.3gpp.org/.

[46] Y.-P. Eric Wang, Xingqin Lin, Ansuman Adhikary, Asbjorn Grovlen, Yutao Sui, Yufei Blankenship, Johan Bergman, and Hazhir S. Razaghi. A Primer on 3GPP Narrowband Internet of Things. *IEEE Communications Magazine*, 55(3):117–123, March 2017. Conference Name: IEEE Communications Magazine.

[47] Jiming Chen, Kang Hu, Qi Wang, Yuyi Sun, Zhiguo Shi, and Shibo He. Narrowband Internet of Things: Implementations and Applications. *IEEE Internet of Things Journal*, 4(6):2309–2314, December 2017. Conference Name: IEEE Internet of Things Journal.

[48] Rapeepat Ratasuk, David Zhou, and Rajnish Sinha. LTE-M Coexistence Within 5G New Radio Carrier. pages 224–228, September 2020.

[49] Carsten Bormann, Angelo P. Castellani, and Zach Shelby. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*, 16(2):62–67, March 2012. Conference Name: IEEE Internet Computing.

[50] MQTT - The Standard for IoT Messaging. https://mqtt.org/.

[51] Federica Rinaldi, Alessandro Raschellà, and Sara Pizzi. 5G NR system design: a concise survey of key features and capabilities. *Wireless Networks*, 27(8):5173–5188, November 2021.

[52] Wikipedia. Edge computing. https://en.wikipedia.org/w/index.php?title=Edge_computing.

[53] ETSI - Welcome to the World of Standards! https://www.etsi.org/.

[54] ETSI. Mobile-Edge Computing – Introductory Technical White Paper. https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf.

[55] Yiwen Wu, Ke Zhang, and Yan Zhang. Digital Twin Networks: A Survey. *IEEE Internet of Things Journal*, 8(18):13789–13804, September 2021. Conference Name: IEEE Internet of Things Journal.

[56] IBM. What Is Blockchain?, July 2021. https://www.ibm.com/topics/blockchain.

[57] Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/en/bitcoin-paper.

[58] IBM. What is IoT with Blockchain?, July 2021. https://www.ibm.com/topics/blockchain-iot.

[59] Archanaa S. Krishnan and Patrick Schaumont. Exploiting security vulnerabilities in intermittent computing. pages 104–124, 2018.

[60] Antonio Muñoz, Ruben Ríos, Rodrigo Román, and Javier López. A survey on the (in)security of trusted execution environments. *Computers & Security*, 129:103180, June 2023.

[61] Arm Ltd. TrustZone for Cortex-A – Arm. https://www.arm.com/en/technologies/trustzone-for-cortex-a.

[62] Arm Ltd. TrustZone for Cortex-M – Arm. https://www.arm.com/technologies/trustzone-for-cortex-m.

[63] Intel. Intel Software Guard Extensions (Intel SGX). https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/software-guard-extensions.html.

[64] AMD. AMD Secure Encrypted Virtualization (SEV). https://www.amd.com/en/developer/sev.html.

[65] Meltdown and Spectre. https://meltdownattack.com/.

[66] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, July 2020. Conference Name: IEEE Internet of Things Journal.

[67] Zachary Ballard, Calvin Brown, Asad M. Madni, and Aydogan Ozcan. Machine learning and computation-enabled intelligent sensor design. *Nature Machine Intelligence*, 3(7):556–565, July 2021. Publisher: Nature Publishing Group.

[68] Francesco Concas, Julien Mineraud, Eemil Lagerspetz, Samu Varjonen, Xiaoli Liu, Kai Puolamäki, Petteri Nurmi, and Sasu Tarkoma. Low-Cost Outdoor Air Quality Monitoring and Sensor Calibration: A Survey and Critical Analysis. *ACM Transactions on Sensor Networks*, 17(2):20:1–20:44, May 2021.

[69] Vishal Kumar Choudhary, Gulshan Kumar, Saurabh Suman, and Abhishek Kumar. A Comprehensive Review on Data Visualization Techniques for Real-Time Information. In *2024 International Conference on Inventive Computation Technologies (ICICT)*, pages 866–871, April 2024. ISSN: 2767-7788.

[70] WHO. Types of pollutants. https://www.who.int/teams/environment-climate-change-and-health/air-quality-and-health/health-impacts/types-of-pollutants.

[71] WHO. Ambient (outdoor) air pollution. https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health.

[72] Vikram Rao and William Vizuete. Chapter One - Why particulates matter. In Vikram Rao and William Vizuete, editors, *Particulates Matter*, Emerging Issues in Analytical Chemistry, pages 3–21. Elsevier, January 2021.

[73] Dean E. Schraufnagel. The health effects of ultrafine particles. *Experimental & Molecular Medicine*, 52(3):311–317, March 2020. Publisher: Nature Publishing Group.

[74] Simone Simões Amaral, João Andrade De Carvalho, Maria Angélica Martins Costa, and Cleverson Pinheiro. An Overview of Particulate Matter Measurement Instruments. *Atmosphere*, 6(9):1327–1345, September 2015. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[75] C. W. Haig, W. G. Mackay, J. T. Walker, and C. Williams. Bioaerosol sampling: sampling mechanisms, bioefficiency and field studies. *Journal of Hospital Infection*, 93(3):242–255, July 2016.

[76] Met One. E-FRM Particulate Sampler. https://metone.com/products/e-frm-particulate-sampler/.

[77] Wikipedia. Tapered element oscillating microbalance, April 2024. https://en.wikipedia.org/w/index.php?title=Tapered_element_oscillating_microbalance.

[78] Thermofisher. 1405-D TEOM™ Continuous Dichotomous Ambient Particulate Monitor. https://www.thermofisher.com/order/catalog/product/TEOM1405D.

[79] Wikipedia. Beta attenuation monitoring, July 2021. https://en.wikipedia.org/w/index.php?title=Beta_attenuation_monitoring.

[80] Thermofisher. 5014i Beta Continuous Ambient Particulate Monitor. https://www.thermofisher.com/order/catalog/product/5014I.

[81] David H. Hagan and Jesse H. Kroll. Assessing the accuracy of low-cost optical particle sensors using a physics-based approach. *Atmospheric Measurement Techniques*, 13(11):6343–6355, November 2020. Publisher: Copernicus GmbH.

[82] Michael R. Giordano, Carl Malings, Spyros N. Pandis, Albert A. Presto, V. F. McNeill, Daniel M. Westervelt, Matthias Beekmann, and R. Subramanian. From low-cost sensors to high-quality data: A summary of challenges and best practices for effectively calibrating low-cost particulate matter mass sensors. *Journal of Aerosol Science*, 158:105833, November 2021.

[83] Leigh R. Crilley, Marvin Shaw, Ryan Pound, Louisa J. Kramer, Robin Price, Stuart Young, Alastair C. Lewis, and Francis D. Pope. Evaluation of a low-cost optical particle counter (Alphasense OPC-N2) for ambient air monitoring. *Atmospheric Measurement Techniques*, 11(2):709–720, February 2018. Publisher: Copernicus GmbH.

[84] Alphasense. OPC-N3. https://www.alphasense.com/products/view-by-target-gas/particulates-optical-particle-counters.

[85] Alphasense. OPC-N3 Software and Manual. https://www.alphasense.com/downloads/software.

[86] Honeywell. HPM Series Particulate Matter Sensors. https://sps.honeywell.com/us/en/products/advanced-sensing-technologies/healthcare-sensing/particulate-matter-sensors/hpm-series.

[87] Sensirion. SEN50. https://sensirion.com/products/catalog/SEN50.

[88] Erika Brattich, Alessandro Bracci, Alessandro Zappi, Pietro Morozzi, Silvana Di Sabatino, Federico Porcù, Francesca Di Nicola, and Laura Tositti. How to Get the Best from Low-Cost Particulate Matter Sensors: Guidelines and Practical Recommendations. *Sensors*, 20(11):3073, January 2020. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[89] P. Zieger, R. Fierz-Schmidhauser, E. Weingartner, and U. Baltensperger. Effects of relative humidity on aerosol light scattering: results from different European sites. *Atmospheric Chemistry and Physics*, 13(21):10609–10631, November 2013. Publisher: Copernicus GmbH.

[90] Andrea Di Antonio, Olalekan A. M. Popoola, Bin Ouyang, John Saffell, and Roderic L. Jones. Developing a Relative Humidity Correction for Low-Cost Sensors Measuring Ambient Particulate Matter. *Sensors*, 18(9):2790, September 2018. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[91] M. D. Petters and S. M. Kreidenweis. A single parameter representation of hygroscopic growth and cloud condensation nucleus activity. *Atmospheric Chemistry and Physics*, 7(8):1961–1971, April 2007. Publisher: Copernicus GmbH.

[92] Palas. Fidas 200. https://www.palas.de/en/product/fidas200.

[93] Florentin M. J. Bulot, Steven J. Johnston, Philip J. Basford, Natasha H. C. Easton, Mihaela Apetroaie-Cristea, Gavin L. Foster, Andrew K. R. Morris, Simon J. Cox, and Matthew Loxham. Long-term field comparison of multiple low-cost particulate matter sensors in an outdoor urban environment. *Scientific Reports*, 9(1):7497, May 2019. Number: 1 Publisher: Nature Publishing Group.

[94] Wikipedia. Metropolitan City of Turin, May 2024. https://en.wikipedia.org/w/index.php?title=Metropolitan_City_of_Turin.

[95] Wikipedia. Turin, June 2024. https://en.wikipedia.org/w/index.php?title=Turin.

[96] Città Metropolitana di Torino. Stazioni di monitoraggio. http://www.cittametropolitana.torino.it/cms/ambiente/qualita-aria/rete-monitoraggio/stazioni-monitoraggio.

[97] Arpa Piemonte. https://www.arpa.piemonte.it.

[98] Qualità dell'aria in Piemonte - Rete di monitoraggio. https://aria.ambiente.piemonte.it/valutare-aria/monitoraggio.

[99] Qualità dell'aria in Piemonte - Accesso dati misurati. https://aria.ambiente.piemonte.it/qualita-aria/dati.

[100] Città Metropolitana di Torino. Relazioni annuali sulla qualità dell'aria: "Uno sguardo all'aria". http://www.cittametropolitana.torino.it/cms/ambiente/qualita-aria/dati-qualita-aria/relazioni-annuali.

[101] ARPA Piemonte. Gli Inquinanti Primari e Secondari. https://www.arpa.piemonte.it/sites/default/files/media/2023-11/Inquinantiprimari_e_secondari3.pdf.

[102] ARPA Piemonte. Il Source Apportionment. https://www.arpa.piemonte.it/sites/default/files/media/2023-11/source_apportionment3.pdf.

[103] ARPA Piemonte. I modelli di qualità dell'aria. https://www.arpa.piemonte.it/scheda-informativa/modelli-qualita-dellaria.

[104] ARPA Piemonte. Modelli CTM di Chimica e Trasporto. https://www.arpa.piemonte.it/sites/default/files/media/2023-11/Modellistica3.pdf.

[105] ARPA Piemonte. Inquinamento da particolato PM10: le sorgenti. https://www.arpa.piemonte.it/notizia/inquinamento-particolato-pm10-sorgenti.

[106] ARPA Piemonte. Inquinamento da particolato PM10: il trasporto su strada. https://www.arpa.piemonte.it/notizia/inquinamento-particolato-pm10-trasporto-strada.

[107] ARPA Piemonte. Inquinamento da particolato PM10: il riscaldamento domestico. https://www.arpa.piemonte.it/notizia/inquinamento-particolato-pm10-riscaldamento-domestico.

[108] Comune di Torino. Raccolta di domande frequenti (faq) sul tema della qualità dell'aria. http://comune.torino.it/torinosostenibile/documenti/200904_FAQ%20Aria_v2.pdf.

[109] Comune di Torino. Rispetto al traffico veicolare, come mai si bloccano prevalentemente i veicoli diesel? http://www.comune.torino.it/ambiente/aria/faq_aria/rispetto-al-traffico-veicolare-come-mai-si-bloccan.shtml.

[110] Wikipedia. European emission standards, June 2024. https://en.wikipedia.org/w/index.php?title=European_emission_standards.

[111] Comune di Torino. Limitazioni del traffico in Torino. http://www.comune.torino.it/ambiente/aria/limitazioni-traffico/index.shtml.

[112] Wikipedia. Inversion (meteorology), April 2024. https://en.wikipedia.org/w/index.php?title=Inversion_(meteorology).

[113] ARPA Piemonte. Perché il PM10 è elevato solo nel periodo autunno/inverno? https://www.arpa.piemonte.it/faq/perche-pm10-elevato-solo-nel-periodo-autunnoinverno.

[114] Wikipedia. Foehn wind, January 2023. https://en.wikipedia.org/w/index.php?title=Foehn_wind.

[115] ARPA Piemonte. Foehn ed Effetti. https://www.arpa.piemonte.it/export/sites/default/pubblicazioni/pdf/6_MeteoVetta_foehn.pdf.

[116] ARPA Piemonte. Prime valutazioni sulla qualità dell'aria nel 2023: trend in miglioramento. https://www.arpa.piemonte.it/notizia/prime-valutazioni-sulla-qualita-dellaria-nel-2023-trend-miglioramento.

[117] ARPA Piemonte. Analisi del rapporto tra PM10 e PM2.5 nel Piemonte sud-occidentale. https://agenda.infn.it/event/6377/contributions/62806/attachments/45514/53875/Poster_43.pdf.

[118] Yang Wang, Jiayu Li, He Jing, Qiang Zhang, Jingkun Jiang, and Pratim Biswas. Laboratory evaluation and calibration of three low-cost particle sensors for particulate matter measurement. *Aerosol Science and Technology*, 49(11):1063–1077, 2015.

[119] Matías Tagle, Francisca Rojas, Felipe Reyes, Yeanice Vásquez, Fredrik Hallgren, Jenny Lindén, Dimitar Kolev, Ågot K. Watne, and Pedro Oyola. Field performance of a low-cost sensor in the monitoring of particulate matter in Santiago, Chile. *Environmental Monitoring and Assessment*, 192(3), February 2020.

[120] Marek Badura, Piotr Batog, Anetta Drzeniecka-Osiadacz, and Piotr Modzel. Evaluation of low-cost sensors for ambient PM 2.5 monitoring. *Journal of Sensors*, 2018, 2018.

[121] SparkFun. Dht22 - datasheet. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[122] Bosch Sensortec. Humidity Sensor BME280. https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/.

[123] Analog Devices. DS3231 Datasheet and Product Info. https://www.analog.com/en/products/ds3231.html.

[124] Raspberry Pi Ltd. Buy a Raspberry Pi Zero W. https://www.raspberrypi.com/products/raspberry-pi-zero-w/.

[125] Wikipedia. Network Time Protocol, May 2024. https://en.wikipedia.org/w/index.php?title=Network_Time_Protocol.

[126] Wikipedia. rsync, May 2024. https://en.wikipedia.org/w/index.php?title=Rsync.

[127] Wikipedia. Secure Shell, June 2024. https://en.wikipedia.org/w/index.php?title=Secure_Shell.

[128] Bosch Sensortec. Pressure Sensor BMP280. https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/.

[129] CDtop Technology. PA1010D. https://www.cdtop-tech.com/products/pa1010d.

[130] Pycom. LoPy 4. https://docs.pycom.io/datasheets/development/lopy4/.

[131] Pycom. FiPy. https://docs.pycom.io/datasheets/development/fipy/.

[132] Pycom. Expansion Board 3. https://docs.pycom.io/datasheets/expansionboards/expansion3/.

[133] MicroPython - Python for microcontrollers. http://micropython.org/.

[134] Bartolomeo Montrucchio, Edoardo Giusto, Mohammad Ghazi Vakili, Stefano Quer, Renato Ferrero, and Claudio Fornaro. A densely-deployed, high sampling rate, open-source air pollution monitoring wsn, 2020. https://dx.doi.org/10.21227/m4pb-g538.

[135] Pietro Chiavassa, Edoardo Giusto, Gustavo Ramirez, Mohammad Ghazi Vakili, Stefano Quer, Filippo Gandino, Bartolomeo Montrucchio, and Maurizio Rebaudengo. Dataset for "Improving data quality of low-cost light-scattering PM sensors: Towards automatic air quality monitoring in urban environments", September 2023. https://zenodo.org/records/8329133.

[136] Pietro Chiavassa, Edoardo Giusto, and Lorenzo Bergadano. Dataset for "Q-SCALE: Quantum Sensor Calibration for Advanced Learning and Efficiency", April 2024. https://zenodo.org/records/11068317.

[137] Fondazione LINKS - soluzioni tecnologiche dal laboratorio al mercato, October 2021. https://linksfoundation.com/en/.

[138] David Hasenfratz, Olga Saukh, Christoph Walser, Christoph Hueglin, Martin Fierz, Tabita Arn, Jan Beutel, and Lothar Thiele. Deriving high-resolution urban air pollution maps using mobile sensor nodes. *Pervasive and Mobile Computing*, 16:268–285, January 2015.

[139] Brian Rumburg, Richard Alldredge, and Candis Claiborn. Statistical distributions of particulate matter and the error associated with sampling frequency. *Atmospheric Environment*, 35(16):2907–2920, June 2001.

[140] Gustavo Ramirez Espinosa, Bartolomeo Montrucchio, Filippo Gandino, and Maurizio Rebaudengo. Frequency Analysis of Particulate Matter in Urban Environments under Low-cost Sensors. In *2021 International Conference on Computer Communication and Artificial Intelligence (CCAI)*, pages 97–105, May 2021.

[141] N-dimensional cumulative function, and other useful facts about gaussians and normal densities. https://upload.wikimedia.org/wikipedia/commons/a/a2/Cumulative_function_n_dimensional_Gaussians_12.2013.pdf.

[142] SciPy. https://scipy.org/.

[143] Docker: Accelerated container application development. https://www.docker.com/.

[144] Eclipse mosquitto: An open source mqtt broker. https://mosquitto.org/.

[145] Python programming language. https://www.python.org/.

[146] Eclipse paho: Mqtt python client library. https://pypi.org/project/paho-mqtt/.

[147] Mysql. https://www.mysql.com/.

[148] Timescale: Postgresql ++ for time series and events. https://www.timescale.com/.

[149] Influxdb. it's about time. https://www.influxdata.com/.

[150] Apscheduler: Advanced python scheduler. https://apscheduler.readthedocs.io.

[151] Flask. https://flask.palletsprojects.com.

[152] Grafana — the open platform for analytics and monitoring. https://grafana.com.

[153] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. Approximate computing: A survey. *IEEE Design & Test*, 33(1):8–22, 2016.

[154] Texas Instruments. TIDM-FRAM-CTPL reference design | TI.com. https://www.ti.com/tool/TIDM-FRAM-CTPL.

[155] Moritz Schneider, Ramya Jayaram Masti, Shweta Shinde, Srdjan Capkun, and Ronald Perez. Sok: Hardware-supported trusted execution environments. *arXiv preprint arXiv:2205.12742*, 2022.

[156] Arm Ltd. Trustzone technology for armv8-m architecture. https://developer.arm.com/documentation/100690/latest.

[157] Trusted Firmware. TrustedFirmware-M (TF-M). https://www.trustedfirmware.org/projects/tf-m/.

[158] Arm Ltd. Armv8-m memory model and memory protection user guide. https://developer.arm.com/documentation/107565/latest/.

[159] Arm Ltd. Trustzone technology microcontroller system hardware design concepts user guide. https://developer.arm.com/documentation/107779/latest.

[160] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems. *IEEE Embedded Systems Letters*, 7(1):15–18, March 2015. Conference Name: IEEE Embedded Systems Letters.

[161] Domenico Balsamo, Alex S. Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. Hibernus++: A Self-Calibrating and Adaptive System for Transiently-Powered Embedded Devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(12):1968–1980, 2016. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[162] Saad Ahmed, Naveed Anwar Bhatti, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. Fast and Energy-Efficient State Checkpointing for Intermittent Computing. *ACM Transactions on Embedded Computing Systems*, 19(6):45:1–45:27, September 2020.

[163] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. QUICKRECALL: A Low Overhead HW/SW Approach for Enabling Computations across Power Cycles in Transiently Powered Computers. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 330–335, January 2014. ISSN: 2380-6923.

[164] Naveed Anwar Bhatti and Luca Mottola. HarvOS: Efficient Code Instrumentation for Transiently-Powered Embedded Sensing. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 209–220, April 2017.

[165] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: system support for long-running computation on RFID-scale devices. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ASPLOS XVI, pages 159–170, New York, NY, USA, March 2011. Association for Computing Machinery.

[166] Saad Ahmed, Naveed Anwar Bhatti, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. Fast and energy-efficient state checkpointing for intermittent computing. 19(6), sep 2020.

[167] Archanaa S. Krishnan and Patrick Schaumont. Exploiting security vulnerabilities in intermittent computing. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 104–124, Cham, 2018. Springer International Publishing.

[168] Archanaa S. Krishnan, Charles Suslowicz, Daniel Dinu, and Patrick Schaumont. Secure intermittent computing protocol: Protecting state across power loss. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 734–739, 2019.

[169] Archanaa S. Krishnan and Patrick Schaumont. Benchmarking and configuring security levels in intermittent computing. *ACM Trans. Embed. Comput. Syst.*, 21(4), sep 2022.

[170] Hafiz Areeb Asad, Erik Henricus Wouters, Naveed Anwar Bhatti, Luca Mottola, and Thiemo Voigt. On securing persistent state in intermittent computing. In *Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, ENSsys '20, page 8–14, New York, NY, USA, 2020. Association for Computing Machinery.

[171] Pietro Chiavassa. ptrchv/STM32-IntermittentSecurity, June 2024. https://github.com/ptrchv/STM32-IntermittentSecurity.