SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I5P106

Volume 11 Issue 5, 53-59, May 2024

Original Article

Efficient Data Logging for One Wire Protocol Sensor in IoT: A Hardware-in-the-Loop (HIL) Approach

Patnaikuni Dinkar R. Patnaik[1], Sachin R. Gengaje[2]

[1,2]*Department of Electronics Engineering, Walchand Institute of Technology, Maharashtra, India.*

[1]*Corresponding Author : pdrpatnaik@gmail.com*

Received: 05 March 2024 Revised: 05 April 2024 Accepted: 03 May 2024 Published: 29 May 2024

Abstract *- The Internet of Things (IoT) has changed how one uses technology by using many sensors that create large sets of data. The integrity and efficiency of IoT systems are very important for their widespread adoption, and Hardware-in-the-Loop (HIL) testing has emerged as a key tool in nearly achieving this. This research paper offers a brief exploration of HIL testing within the scope of IoT engineering, exploring its practical applications and associated methodologies. This article empirically describes two distinct data processing methods, Method-I and Method-II, providing insights into the adept tradeoff between CPU-intensive algorithms and the efficiency of one-pass processing with Welford's Algorithm. Additionally, the paper introduces an innovative schema-based approach for sensor data storage using both XML and JSON formats, enabling efficient and versatile data storage and retrieval, particularly when guided by Probability Density Functions (PDFs). While XML's verbosity and rigidity are acknowledged, the choice of format is contextual and application-specific. Overall, this research advances our understanding of effective data processing and storage in IoT environments while showcasing the potential of schema-driven, PDF-based data storage, reproduction, and schema design.*

Keywords *- Hardware-in-the-Loop (HIL) Testing, Sensor data logging and processing, Welford's algorithm, Running standard deviation and average, Sensor data storage, XML schema, JSON schema, Probability Density Function (PDF), Schema-based data storage, Sensor data reproduction.*

1. Introduction

IoT devices are becoming increasingly important, covering domains ranging from home automation to industrial control systems. The proper functioning of these devices relies on rigorous testing and data analysis.

Hardware-in-the-Loop (HIL) testing is a critical approach for evaluating IoT systems. However, the current systems lack a detailed analysis of the underlying embedded system that deals with signal capture, process, and reproduction.

Therefore, there is a need to identify the efficiency of the HIL for any given application that requires an efficient approach to reproduce the realistic signal behaviour to emulate the near-real-time or real-time signal data at the right moment that deals with polling every moment of a signal for the entire system without failure which often is a CPU intensive job for an embedded system.

The following section discusses the relevant methods used to experiment and validate an efficient methodology to reproduce the sensor signal in the perspective of reproducing a sensor signal specifically designed to implement HIL for an IoT application.

1.1. What is Hardware-in-the-Loop

Hardware-in-the-Loop (HIL) testing is a methodology used to test and validate IoT systems by simulating real-world conditions. In this environment, actual hardware components, such as sensors, actuators, and controllers, are integrated with simulation models [1]. This amalgamation facilitates the evaluation of the system's performance in a controlled yet realistic setting.

1.2. Hardware-in-the-Loop (HIL) Testing

Hardware-in-the-Loop (HIL) testing is a methodology used to test and validate IoT systems by simulating real-world conditions. In this environment, actual hardware components, such as sensors, actuators, and controllers, are integrated with simulation models [2]. This conjunction facilitates the evaluation of the system's performance in a controlled yet realistic setting.

1.3. The HIL Environment

The HIL environment consists of both hardware and software components. Hardware components include IoT devices like sensors, microcontrollers (such as the Raspberry Pi Pico), and actuators. These components are connected to a testing rig, which allows for data exchange and interaction

with the IoT system. The software part comprises simulation models that emulate the behavior of the physical world.

2. Data Logging for Sensors

Data logging for sensors involves a systematic process that begins with selecting suitable sensors for specific measurements and ensuring their proper installation and calibration. A compatible data logger or data acquisition system is chosen, configured to capture data at the desired intervals, and powered adequately for continuous operation.

Sensors are connected to the data logger, accounting for environmental conditions, and real-time monitoring, if needed, is established. Data is stored either in the data logger's memory or external storage, with retrieval options, and subsequently analyzed for insights. Maintenance, calibration, security, and data management considerations are crucial for accurate and secure data collection and storage. This process facilitates informed decision-making across various applications, from scientific research to industrial processes and IoT deployments.

Data logging with a microcontroller like the Raspberry Pi Pico involves interfacing sensors with the GPIO pins of the Pico, using libraries and programming languages like Python to read sensor data at specified intervals. The collected data can be stored in a local file or transmitted to external storage or cloud services.

The Pico's built-in hardware timers can help in precise data sampling, and you can implement power-saving strategies to extend battery life for remote deployments. This approach enables cost-effective and versatile data logging solutions for a wide range of applications, such as environmental monitoring, home automation, and industrial control systems.

Microcontrollers store data using various methods, including internal memory (such as Flash or EEPROM) for non-volatile storage of program code and configuration settings, RAM for temporary data storage during program execution, and external storage devices like SD cards or EEPROM modules for larger datasets. Additionally, some microcontrollers support communication with external storage through interfaces like SPI or I2C. Storing data in registers, arrays, or variables within the microcontroller's memory is common for in-program data manipulation. At the same time, external storage is ideal for more extensive or long-term data-logging applications.

The choice of storage method depends on factors like data volume, retention requirements, and power constraints, it is equally important to assess the complexity involved in storing the data logged and also the amount of load that a microcontroller is capable of withstanding.

3. Data Processing Methods

Efficient data processing is essential for extracting meaningful insights from IoT sensor data. This paper presents two distinct methods for data processing: Method I and Method II.

The complexity surrounding sensors discussed in Valencia, Goswami, and Goossens (2019) is not explicitly outlined in the provided passages. The paper delves into platform-aware control design flows, mechanical setups, electrical circuits, MATLAB and Hardware-in-the-Loop (HIL) experiments, Quality of Control (QoC) analysis, and design guidelines [3]. Although sensors are mentioned, there is no specific focus on their complexity.

In contrast, Gis, Büscher, and Haubelt (2021) elaborate on the complexity of sensors, particularly in evaluating entire systems, including inertial sensors, especially in safety-critical systems [4]. They introduce a Sensor-in-the-Loop architecture to address reproducibility challenges, enabling real-time injection of sensor data directly into the hardware, ensuring repeatable and reproducible results with lower jitter compared to traditional methods.

Similarly, Gis, Büscher, and Haubelt (2020) discuss challenges in software development for smart inertial sensors due to hardware limitations and lack of reproducible testing options, proposing a Sensor-in-the-Loop architecture as a solution [5, 6].

Additionally, the complexity of digital sensors in a Hardware-in-the-Loop (HIL) testing framework is highlighted in Kalyan et al. (2023), specifically focusing on accelerometer sensors BMA280 and BMC150 [7].

Overall, while Valencia et al. (2019) provide insights into control design flows and experiments without emphasizing sensor complexity, Gis et al. (2020, 2021) and Kalyan et al. (2023) address challenges related to sensor evaluation and testing within HIL frameworks, proposing innovative solutions to mitigate these complexities [8].

3.1. Method-I: CPU-Intensive Algorithm

Method I employs a computationally intensive algorithm. It involves recalculating the standard deviation and running the average every time new data is added to the existing data stream [9, 10]. This method is suitable for scenarios where real-time processing and immediate feedback are paramount, but it comes at the cost of high CPU utilization.

However, the CPU-intensive nature of this Algorithm may pose challenges in resource-constrained IoT devices. It demands a significant amount of processing power, which could lead to increased energy consumption and potential overheating issues in certain scenarios [11].

3.2. Method-II: One-Pass Data Processing with Welford's Algorithm

Method II takes advantage of Welford's Algorithm to calculate standard deviation and running average efficiently. Unlike the two-pass mechanism of Method I, this method processes the entire data set in a single pass 12, 13]. It enables the continuous calculation of standard deviation and running average in real-time, making it particularly suitable for applications with resource constraints.

Welford's Algorithm is a smart way to find the average and spread of numbers as one gets them, and this itself is a novelty, without needing to remember all the numbers we have seen [14-16]. It is really handy when we are dealing with many numbers and do not have much memory to spare. This method keeps track of three things: how many numbers we have seen so far (n), what the average is (M), and how spread out the numbers are (S).

When a new number comes in, it uses simple math to update these three values. It is like keeping a running tally of the average and spread as one goes along, which is super helpful for real-time calculations and when one cannot store a huge list of numbers in our computer's memory. One of the key advantages of Method II is its ability to maintain continuous calculations of standard deviation and running average in real time. This makes it particularly suitable for IoT applications where resource constraints are a concern. By processing the data more efficiently, Method-II also minimizes CPU utilization and energy consumption.

This paper presents the data logging and processing using the usual approach. It then later compares that with another approach as described above, using the statistical method of sensor data storage using a Raspberry Pi Pico microcontroller and a DHT 22 temperature and humidity sensor, which uses a one-wire protocol.

4. Methodology

In this section, the detailed methodology used to interface the DHT22 sensor with the Raspberry Pi Pico, a commonly used microcontroller in IoT applications and two methods of logging sensor data practically are discussed. The DHT22 sensor provides temperature and humidity readings. It communicates via a standardized protocol, i.e., a wire protocol, making it suitable for integration with IoT devices through a simple one-wire interface, thereby limiting the number of wires needed to interface many sensors.

4.1. Interfacing the DHT22 Sensor with Raspberry Pi Pico

Figure 1 below represents the Raspberry Pi Pico and the DHT22 sensor connected using the one-wire protocol to capture data on temperature and humidity. The "DATA" wire serves as the communication link between the Pico and the sensor, while the "GROUND" connection ensures a common reference for both devices. The DHT22 sensor is responsible for measuring temperature and humidity and sending this data to the Raspberry Pi Pico for processing or logging. The following interface was simulated using an online tool available at the link wokwi.com, which is a widely used tool to simulate IoT projects through a browser. The same tool would be used further to simulate the interfacing of the DHT 22 sensor with a Raspberry Pi Pico for demonstration.

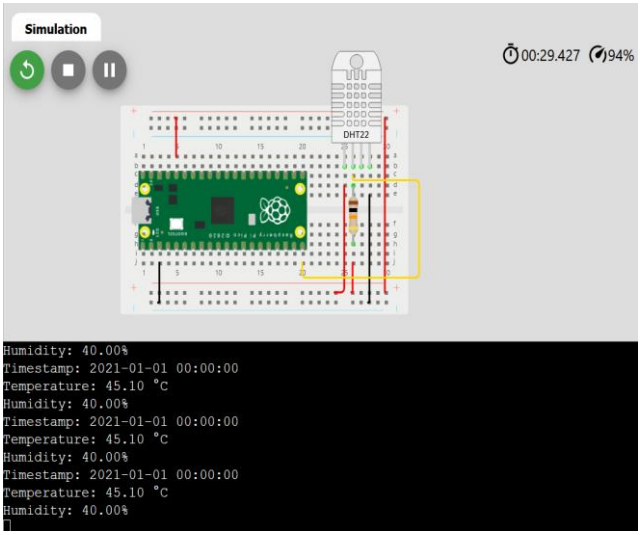

Fig. 1 Raspberry Pi Pico interfaced with DHT 22 sensor for data capture

As can be clearly seen in the figure above, the Raspberry Pi Pico is programmed so as to display Timestamp, Temperature and Humidity, which is further stored in the form of an array and later retrieved to store in the form of a CSV file using appropriate external tools which is beyond the scope of this paper. However, certainly, this method is too time-consuming and the chances of inserting noise into data increases.

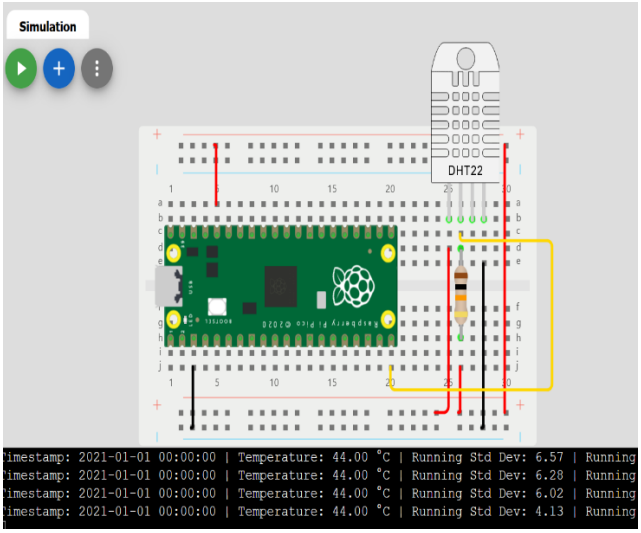

Fig. 2 Raspberry Pi Pico interfaced with DHT 22 showing running standard deviation and average

Followed by this, as shown above in Figure 2 an experiment was conducted to reprogram the Raspberry Pi Pico to calculate the running standard deviation and average for temperature and skipped the humidity reading for simplicity and demonstration purposes. Here, it is very clear that the use of Welford's Algorithm greatly reduces the CPU intensiveness of the calculation process for standard deviation and average while the temperature data stream.

As can be seen in the lower part of Figure 2 above, whenever the temperature varies, the standard deviation is calculated. If the same temperature is maintained for a while, the deviation tends towards zero value while the average approaches the true value. This stands valid with respect to the rules of probability and statistics and, hence, is a novel way presented in this paper to store the behaviour of the sensor data for a given experiment of logging the temperature.

Further, this data is stored in the form of a Probability Distribution Function (PDF) of the sensor data, which can later be used to reproduce a similar behavior exhibiting sensor waveform. However, it is again very important to note that complete regeneration of the entire signal is almost not possible merely on the basis of the PDF available for a given signal and it only stores and gives information about the statistical contribution and potentially leaves some essential details about the signal structure for each value for a given span of consideration.

4.2. Schema Definition for Sensor Data

A sensor schema is like an organized and agreed-upon template for recording information about sensors. This template includes important facts about sensors, such as what kind they are, where they are placed, their special identification numbers, how they are calibrated, the way they store data, and other important details. These sensor schemas are really important because they help keep all the sensor data neat and organized. It is like having a set of rules that make sure everyone records sensor information in the same way. This makes it easier for people to find, use, and share sensor data, no matter what kind of systems or fields they are working in, and, of course, for a microcontroller, too.

Basically, there are two formats for defining a sensor schema, which are discussed here, namely JSON and XML. Here is an example of a DHT22 sensor data schema defined using XML and JSON formats which will be compared for its benefits later in this section.

Table 1. JSON schema for DHT22 sensor data

```
{
  "sensor_signal": {
    "sensor": {
      "name": "DHT22",
      "model": "AM2302",
      "manufacturer": "Adafruit"
    },
    "location": {
      "latitude": 40.7128,
      "longitude": -74.0060,
      "altitude": 10,
      "description": "New York City"
    },
    "data": {
      "temperature": {
        "value": 25.0,
        "unit": "Celsius"
      },
      "humidity": {
        "value": 50.0,
        "unit": "Percentage"
      }
    },
    "timestamp": "2023-09-25T14:30:00Z"
  }
}
```

Table 2. XML schema for DHT22 sensor data

```
<sensor_signal>
  <sensor>
    <name>DHT22</name>
    <model>AM2302</model>
    <manufacturer>Adafruit</manufacturer>
  </sensor>
  <location>
    <latitude>40.7128</latitude>
    <longitude>-74.0060</longitude>
    <altitude>10</altitude>
    <description>New York City</description>
  </location>
  <data>
    <temperature>
      <value>25.0</value>
      <unit>Celsius</unit>
    </temperature>
    <humidity>
      <value>50.0</value>
      <unit>Percentage</unit>
    </humidity>
  </data>
  <timestamp>2023-09-25T14:30:00Z</timestamp>
</sensor_signal>
```

4.3. Sensor Schema for PDF of Sensor Data

The Probability Density Function (PDF) serves as a mathematical tool for characterizing the statistical properties of a random variable's potential values. It offers insights into the likelihood of various values occurring within the variable's defined range. Through the process of integrating this function over a specific interval, this gains the ability to quantify the probability of the random variable taking on a value that lies within that particular interval.

In more precise technical terms, the PDF operates as a formal representation of the probabilities associated with different potential outcomes of a random variable. It provides a continuous description of how these outcomes are distributed across the variable's feasible range. When one performs integration on the PDF over a defined range, it corresponds to a probabilistic calculation. This fundamental concept in probability theory and statistics is now utilized to form the schema of the DHT22 sensor data, as shown in the tables below.

Table 3. JSON schema for DHT22 sensor data PDF

```
{
 "sensor_signal": {
  "temperature": {
   "mean": 25.0,
   "std_deviation": 2.0,
   "welford_variance": 0.0,
   "welford_mean": 0.0,
   "sample_count": 0
  },
  "humidity": {
   "mean": 50.0,
   "std_deviation": 5.0,
   "welford_variance": 0.0,
   "welford_mean": 0.0,
   "sample_count": 0
  }
 }
}
```

In this XML schema, elements for welford_variance, welford_mean, and sample_count under both temperature and humidity to keep track of the statistics calculated using Welford's Algorithm are added. Initially, these values are set to 0.

Table 4. XML schema for DHT22 sensor data PDF

```
<sensor_signal>
<temperature>
<mean>25.0</mean>
<std_deviation>2.0</std_deviation>
<welford_variance>0.0</welford_variance>
<welford_mean>0.0</welford_mean>
<sample_count>0</sample_count>
</temperature>
<humidity>
<mean>50.0</mean>
<std_deviation>5.0</std_deviation>
<welford_variance>0.0</welford_variance>
<welford_mean>0.0</welford_mean>
<sample_count>0</sample_count>
</humidity>
</sensor_signal>
```

In this JSON schema, similar to the XML schema, additional fields for welford_variance, welford_mean, and sample_count under both temperature and humidity to keep track of the statistics calculated using Welford's Algorithm are added. Initially, these values are set to 0.

4.4. Advantages of Defining Sensor Data through Schema

Firstly, defining any data through schema has the added advantage of getting the schema definition validated through appropriate tools, which in turn validates the definitions of the application as a whole in cases where the sensor data is prime.

4.4.1. Advantages of JSON

- Lightweight and user-friendly format for both humans and machines.
- Supported in modern programming languages, ensuring versatility.
- Native compatibility with JavaScript, commonly used in web and IoT development.
- Excellent choice for straightforward data exchange between systems.

4.4.2. Advantages of XML

- Offers structured representation for complex data hierarchies.
- Supports namespaces and validation, which is vital for data interchange standards.
- Widely used in applications requiring precise document structure and data validation.

- Self-descriptive with tag names and attributes, enhancing human readability in certain scenarios.

XML Schema tends to be verbose and challenging to compose and comprehend when compared to JSON Schema. It leans towards being inflexible and less adaptable, which constrains your ability to define data and logic creatively. Furthermore, it does not show the same level of popularity as JSON or other formats that align better with web applications. However, the reader of this paper needs to note that depending on what the end application is one must choose the format.

5. Signal Reproduction for the HIL System

To reproduce the signal for temperature and humidity based on the sensor schema created, the Pico was reprogrammed, and the MicroPython code that generates synthetic data using Welford's Algorithm was incorporated in the code, as can be seen in Figure 3 below (left half).

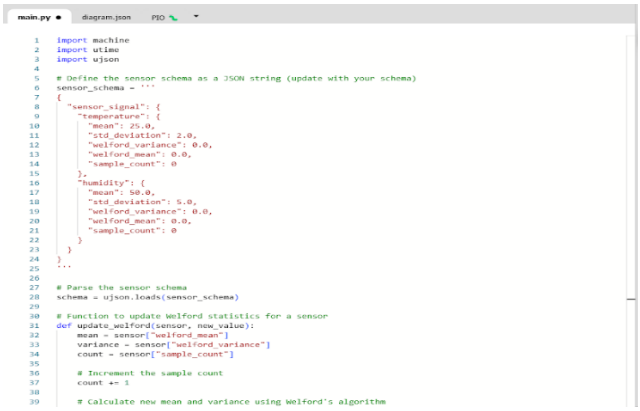

Fig. 3 Raspberry Pi Pico reproducing DHT22 sensor data using PDF schema and synthetic data

Since MicroPython does not include extensive data plotting libraries, the focus was on generating and printing the synthetic data. This data can later be transferred to a computer for detailed plotting if needed. This experiment was only to demonstrate the reproduction of sensor data and does not deal with the accuracy of the data thus reproduced. However, the plotted over an Excel plotter showed the results to be quite similar as these were not completely relying on the randomness of the sensor data but a gradual and steady change in the environmental parameters in the sensor's vicinity.

This code initializes the sensor schema, updates Welford statistics for temperature and humidity, generates synthetic data, which can even be replaced with real sensor data if there is a need for accuracy, and prints the data along with the final Welford statistics.

6. Conclusion

In conclusion, this research paper has delved into the pivotal role of Hardware-in-the-Loop (HIL) testing within the domain of IoT engineering, exploring its practical applications and methodologies. It has further examined two data processing methods, Method-I and Method-II, providing insights into the balance between CPU-intensive algorithms and the efficiency offered by one-pass processing via Welford's Algorithm.

Additionally, the paper introduced a schema-based approach for sensor data storage using both XML and JSON formats, paving the way for efficient PDF-based data storage and retrieval. While XML's verbosity and rigidity were acknowledged, the choice between XML and JSON depends on the specific demands of the application.

In essence, this research contributes not only to the understanding of effective data processing and storage in IoT environments but also to the utilization of PDF-based storage schemas, offering practical insights into sensor data logging, reproduction, and robust schema design.

References

[1] Igor Pintaric et al., "Flexible HiL Interface Implementation for Automotive XiL Testing," *2021 IEEE Vehicle Power and Propulsion Conference (VPPC)*, Gijon, Spain, pp. 1-3, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Farshideh Kordi et al., "Poster: Conceptual Design for FPGA Based Artificial Intelligence Model for HIL Applications," *2023 IEEE Symposium on Computers and Communications (ISCC)*, Gammarth, Tunisia, pp. 1-3, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Juan Valencia, Dip Goswami, and Kees Goossens, "Comparing Platform-Aware Control Design Flows for Composable and Predictable TDM-Based Execution Platforms," *ACM Transactions on Design Automation of Electronic Systems*, vol. 24, no. 3, pp. 1-26, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Angga Wahyu Aditya et al., "Implementation of the In The Loop (Mil) Model and In The Loop (Hil) Hardware as Practical Support Means," *Electrical, Electronics and Telecommunications Engineering-B30*, vol. 4, pp. 1-6, 2019. [Google Scholar] [Publisher Link]

[5] Daniel Gis, Nils Büscher, and Christian Haubelt, "Investigation of Timing Behavior and Jitter in a Smart Inertial Sensor Debugging Architecture," *Sensors*, vol. 21, no. 14, pp. 1-25, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Daniel Gis, Nils Büscher, and Christian Haubelt, "Advanced Debugging Architecture for Smart Inertial Sensors Using Sensor-in-the-Loop," *2020 International Workshop on Rapid System Prototyping (RSP)*, Hamburg, Germany, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Dusarlapudi Kalyan et al., "Model-Based Design Accelerometer Control System in EV-ECU for HIL Testing," *2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bengaluru, India, pp. 150-154, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] B. Aravind Krishnan, and Anju S. Pillai, "Digital Sensor Simulation Framework for Hardware-in-the-Loop Testing," *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kerala, India, pp. 813-817, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[9] Henrique Magnag et al., "HIL-Based Certification for Converter Controllers: Advantages, Challenges and Outlooks (Invited Paper)," *2021 21st International Symposium on Power Electronics (Ee)*, Novi Sad, Serbia, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Antonio Parejo et al., "Raspberry Pi-Based Cluster Network for the Emulation of Sensor Networks in Remote Teaching," *2022 Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (XV Technologies Applied to Electronics Teaching Conference)*, Teruel, Spain, pp. 1-5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] James T. Meech, and Phillip Stanley-Marbell, "An Algorithm for Sensor Data Uncertainty Quantification," *IEEE Sensors Letters*, vol. 6, no. 1, pp. 1-4, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Georgios Kokkinis et al., "High-Speed, Real Time Sensor Data Acquisition and Transfer Based on the Raspberry Pi Single Board Computer," *2023 International Balkan Conference on Communications and Networking (BalkanCom)*, İstanbul, Turkiye, pp. 1-4, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Sheikh Badar ud din Tahir, Ahmad Jalal, and Kibum Kim, "Daily Life Log Recognition Based on Automatic Features for Health Care Physical Exercise via IMU Sensors," *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, Islamabad, Pakistan, pp. 494-499, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Wenbing Zhao et al., "A Blockchain-Facilitated Secure Sensing Data Processing and Logging System," *IEEE Access*, vol. 11, pp. 21712-21728, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Zhan Zhang et al., "Efficient Hardware Redo Logging for Secure Persistent Memory," *2021 IEEE 23rd International Conference on High-Performance Computing & Communications, 7th International Conference on Data Science & Systems, 19th International Conference on Smart City, 7th International Conference on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Haikou, China, pp. 41-48, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Andrey A. Efanov, Sergey A. Ivliev, and Alexey G. Shagraev, "Welford's Algorithm for Weighted Statistics," *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, Moscow, Russia, pp. 1-5, 2021. [CrossRef] [Google Scholar] [Publisher Link]