SENSIBUS: a Novel One Wire Protocol for Smart Sensors

Leonardo Balocchi
Department of Engineering
University of Perugia
Perugia, Italy
leonardo.balocchi@studenti.unipg.it

Michele Vitelli, Mario Molinara
Dept. of Electrical and Information Engineering
University of Cassino and Southern Lazio
Cassino, Italy
{michele.vitelli; molinara}@unicas.it

Stefania Bonafoni, Luca Roselli
Department of Engineering
University of Perugia
Perugia, Italy
{stefania.bonafoni; luca.roselli}@unipg.it

Simone Contardi, Iacopo Nannipieri
Sensichips s.r.l.
Aprilia, Italy
{simone.contardi; iacopo.nannipieri}@sensichips.com

***Abstract*— This paper introduces a novel communication protocol tailored for multi-sensor readout applications, offering enhanced implementation efficiency and faster readout speeds compared to current protocols in use. Dubbed Sensibus, this protocol operates as an asynchronous master-slave system, drawing inspiration from the Maxim One Wire protocol while refining addressing capabilities. The protocol underwent successful implementation on a cable and was deployed alongside the Sensiplus chip for distributed measurements, demonstrating its effectiveness. This advancement unlocks new possibilities for distributed sensing applications, facilitating swift measurements for real-time applications like the battery monitoring problem presented in this article. Moreover, its straightforward nature makes Sensibus easily deployable and cost-efficient, aligning seamlessly with IoT standards.**

***Index Terms*—Sensibus, Sensiplus, Single Wire Protocol, Distributed Sensors, Battery Monitoring**

I. Introduction

In the rapidly evolving field of Internet of Things (IoT), communication protocols play a critical role in integrating and facilitating the functionality of connected devices. The exponential growth of IoT applications has highlighted the need for effective, reliable and scalable communication methods between integrated circuits (ICs). Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) have emerged as industry standards, facilitating complex device interactions in constrained environments [1]. SPI, or Serial Peripheral Interface, is known for its high-speed data transfer capabilities. Developed by Motorola, SPI operates on a master-slave architecture, offering synchronous and full-duplex communication with distinct advantages in speed and data integrity [2]. Conversely, I2C, developed by Philips Semiconductors (now NXP Semiconductors), is a synchronous multi-master, multi-slave bus system [3] [4]. It is known for its minimal wiring requirements and inherent flexibility, making it a common choice for low-speed peripheral communication within complex electronic systems.

This work was supported in part by the the Italian Ministry of University and Research (MUR), in the frame of the "PON 2022 Ricerca e Innovazione" action.

However, as IoT systems become more complex and the number of sensors increases, traditional protocols reach their limits, particularly in terms of wiring complexity and circuit requirements [5]. The One-Wire protocol developed by Maxim Integrated offers a compelling alternative. Operating with a single data wire plus ground for communication, it allows power and data to be transmitted over the same wire, significantly reducing wiring complexity and circuit footprint [6] [7] [8].

Our proposed enhancements to Maxim's One-Wire communication protocol target specific application deficiencies: in particular, the important broadcast and multicast features, native to One-Wire and not to SPI and I2C, were retained, but the data reading part from multiple sensors was also improved. Therefore, the primary focus is on improving data reading from multiple sensors, addressing the limitations of current single-wire protocols. Traditionally, broadcast and multicast protocols are suitable for sending instructions to an array of sensors, but the efficiency of retrieving data - or 'readcast' - from multiple sensors simultaneously remains suboptimal. Our proposed solution introduces efficient hardware mechanisms for addressed communication with multiple slave devices, supporting multicast modes for both write and read operations. So in this work we present a system that allows dynamic changes in communication speed and addressing size, which are essential for unambiguous device identification in complex IoT ecosystems. This enhanced flexibility in communication settings will meet the diverse requirements of advanced IoT applications, ranging from simple home automation systems to complex industrial networks.

II. SENSIBUS PROTOCOL

The Sensibus (SB) protocol is a flexible asynchronous communication system that uses a single data line in open collector mode, supporting a master-slave setup for multiple devices. It supports different addressing methods: unicast for one-to-one communication, broadcast for one-to-all write operations, and multicast/readcast for group communication (write/read operations). Specific addresses are required for unicast and multicast operations, while no address is required if only one slave is present. Broadcast ensures that all slaves receive the same message at the same time. Multicast and Readcast address specific groups for writing and reading, respectively, with addresses reserved for these methods. The following sections provide a detailed discussion of each addressing mode.

A. Bit Timeslot

The SB protocol features a single master that can connect to unlimited slave devices. The master initiates the minimum receive/transmit bit cycle within a single time slot (TS). The duration of a TS is customizable, and to facilitate this, the slave device must include a memory register that contains the necessary information. In the current implementation, a 10 MHz base clock and an 8-bit register allow the TS to be programmed from 102.4 µs down to a minimum of 0.4 µs. At power-up, the default start value is the slowest setting. Each transmitted bit in the SB passes through four phases within a TS, as shown in Figure 1.

- *TS Start*: The master initiates the start of a time slot (TS) by lowering the data line for the first $\frac{1}{4}$ of the TS duration. This low state indicates the start of the TS. The data line must remain low for at least $\frac{1}{8}$ of the TS to be considered valid. If this criterion is not met, the TS is aborted.
- *TS Data Setup*: During the period from $\frac{1}{4}$ to $\frac{1}{2}$ of the TS, the data must be presented and maintained till the end of the TS (from either the master or slave).
- *TS Data Strobe*: At $\frac{3}{4}$ of the TS duration, the receiving device (either master or slave) samples the data present on the line.
- *TS End*: In the last $\frac{1}{4}$ of a TS, after the receiving device has captured the data, the data line is released and goes high. This event concludes the TS and the master can initiate the next transaction when it detects that the line has reached a high level.

The protocol's asynchronous nature enables the transmission of successive bits without the need to meet precise time constraints, while still maintaining the context of the communication.

B. Device Identification

The protocol introduces a unique timing sequence called Start Communication (StartCom) to initiate each new transmission on the bus. The master lowers the data line and holds it in a low stable state for more than 1.5 times the maximum time slot period (TSmax) that can be set. The master then releases the line to a high state. In response, all active

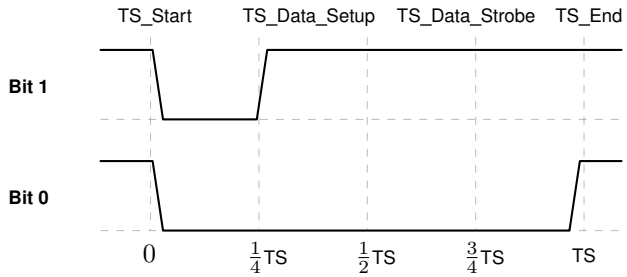

Fig. 1. This is a definition of a timeslot structure for transmitting or receiving bits. The first line shows the structure when the bit is equal to 1, and the second line shows the structure when the bit is 0. The time axis evolves in steps of $\frac{1}{4}$ of the timeslot.

slaves on the bus send an Alive message by dropping the data line after $\frac{1}{4}$ of the TSmax period and holding it low for $\frac{1}{2}$ of the TSmax duration, based on the TS interval set in the slave (Figure 2). Additionally, the protocol incorporates a hot insertion feature to detect new devices on the communication channel. Each slave utilizes a power-on reset that lasts 3.75 times the TSmax duration, followed by an Alive message to indicate its presence. This dynamic detection enables the master to identify newly added devices. The master can then acquire the device ID using an approach similar to the One Wire protocol.

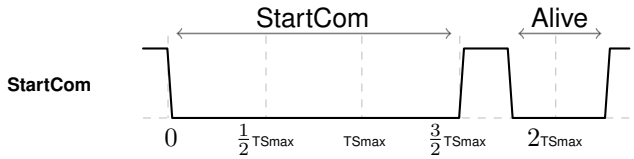

Fig. 2. Definition of the StartCom special bus transaction. The master keeps the line low for 1.5 times the maximum timeslot (up to $\frac{3}{2}$ TSmax). After an additional $\frac{1}{4}$TSmax, the slaves on the channel respond by keeping the line low for $\frac{1}{2}$ of the maximum timeslot. The time axis evolves in steps of $\frac{1}{2}$ of the maximum timeslot.

C. Write and Read Operations

In the Sensibus protocol, the register address indicates whether an operation involves writing to or reading from a memory register. The size of the register address is 8 bits in the current implementation, with the most significant 6 bits identifying the memory area (sub-address) and the remaining 2 bits defining the access type, as shown in Figure 3. Specifically, the bit at position 0 indicates the operation type, where 1 indicates a read operation and 0 indicates a write operation. The bit at position 1 serves the same purpose as configuring single or multiple access. Single access refers to operations involving only one register or, more precisely, the minimum memory location that can be accessed. Multiple access allows the master to control multiple exchanges of registers within the same transaction at its discretion. To function properly, slave devices automatically increment the reference to the selected internal register area (sub-address) after each exchange. This allows sequential and incremental operations on all available registers.

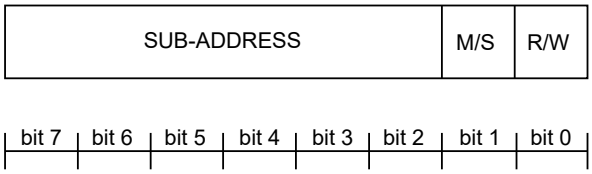

Fig. 3. Structure of the register address. The least significant bit indicates whether the operation is write (W) or read (R). Bit 1 indicates whether single (S) or multiple (M). The remaining bits, from bit 2 to bit 7, identify the register or memory area on which the operation is to be performed.

D. Addressed Modes

In the Sensibus protocol, devices have at least two identifier types with a maximum length of 48 bits, which are actively used in different communication modes. The protocol allows devices to be configured to use a subset of identifiers for more efficient communication. There are three addressing mode configurations available:

- *NoAddress*: Used when a single slave device is on the bus, omitting any reference to the identifier in each transaction.
- *ShortAddress*: Uses an 8-bit portion of the address, providing efficient communication while still uniquely identifying devices or a set of devices.
- *FullAddress*: Uses the entire 48-bit identifier for communication.

In addition, the protocol presents the following usage options:

- *Unicast*: Any device with a unique Device ID (DevID) can communicate in unicast mode, allowing specific messages for both read and write operations without confusion.
- *Broadcast*: Two reserved addresses (0x0000000000000000 and 0x000000000001) are used for broadcast write mode. The former defaults to the slowest bit timeslot, reaching all devices on the channel, while the latter operates at the current speed of the configured devices.
- *Multicast*: Reserved DevID addresses (0x0000000002 through 0x0000000012) are used as Multicast Device ID addresses (MDevID) for group communication. Each slave device can be configured to respond to any of these addresses, allowing simultaneous write operations for up to 16 independent clusters of devices.
- *Readcast*: This unique read system allows multiple chips to respond to a single master read request. In this mode, the DevID is not used directly but serves as an internal counter to optimize read accesses based on the number of devices present. For example, a multicast group of N chips can be set up to respond simultaneously to a shared MDevID. The DevIDs are not used and can be temporarily overridden for each device with a consecutive number ranging from 0 to N-1. This serves as a reference to determine when each slave should send their response to the communication channel. As a result, this allows for the concatenated response of all the chips within the multicast group to be obtained with just one read access to the communication channel.

E. Communication Sequence

In conclusion, the communication structure using Sensibus is defined by these steps (Figure 4).

- *StartCom*: The Master initiates communication by asserting this sequence.
- *Send_DevID*: The master transmits the DevID of the selected slave device from the most significant bit to the least significant bit.
- *Send_RegAddr*: Then, the Master sends the Register Address Byte, following the same convention as the other serial interfaces.
- *Write_Byte or Read_Byte*: The data to be written in the indicated register (Write_byte) is transmitted by the master, or the contents of the register are sent by the slave (Read_byte).

III. Example of Use

The described protocol has been used and tested in an application as a proof of concept. The Sensiplus sensor, the only one currently using this type of bus, was selected for the specific application. The Sensiplus microChip (SPC), developed jointly by Sensichips s.r.l. and the Department of Information of the University of Pisa [9], is a proprietary micro-analytical sensor platform with on-chip sensing capabilities. It features multiple internal and external ports and a versatile analog front end that enables Electrochemical Impedance Spectroscopy (EIS) measurements on both internal and external sensors. Characterized by low power consumption (less than 1.4 mW), small size (1.5 mm x 1.5 mm), and potentially low cost (less than 10 USD in volume production), the SPC lacks processing power and is therefore complemented by a Micro Control Unit (MCU) with communication capabilities (wired or wireless). The connection between the SPC and the MCU is established using the Sensibus protocol. A device equipped with this technology is highly versatile and can be applied in various fields. For instance, it can be used to create IoT systems for optical spectroscopy [10] or end-to-end systems for pollutant detection in wastewater [11]. To evaluate the protocol capabilities, the authors selected the Sensiplus-based device known as the Cell Battery Management Unit (CMU), which is designed specifically for battery engineers and researchers. The detailed measurement capabilities of these devices are described in [12]. In this work, these capabilities have been exploited to build a prototype of a distributed sensing system for use in an automotive context where all-electric mobility is the focus (Figure 5). These vehicles are based on energy cell-based storage systems organized in modules, where the basic element is the single energy cell. The idea is to apply a sensor to each of these cells, this kind of approach easily reaches a large number of sensing devices that need to be orchestrated quickly and efficiently. A schematic example of this type of architecture is shown in Figure 6. A protocol for creating

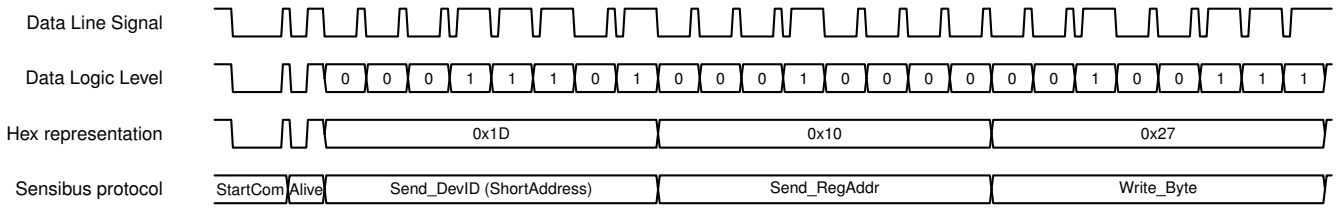

Fig. 4. This is an example of a full *Unicast* Sensibus transaction in *ShortAddress* mode. It consists of a write operation on a device with 0x1D as the least significant 8-bit of the DevID, a memory register identified by 0x10, and a payload equal to 0x27.

clusters of independent devices and efficiently querying them is critical in these applications. To read data extracted from sensors, it is necessary to take simultaneous readings at a given instant by sending a command simultaneously to all sensors in each module. This reduces measurement times and ensures that readings are taken at the same time instant. The ultimate goal of the prototype is to create datasets that accurately represent the operating state of the system. This will enable the development of methods and models to address open challenges in the automotive and smart energy application fields, such as estimating characteristic parameters of energy cells. Identifying their life state will promote the circular economy and can reduce the environmental impact of the transition to renewable energy.

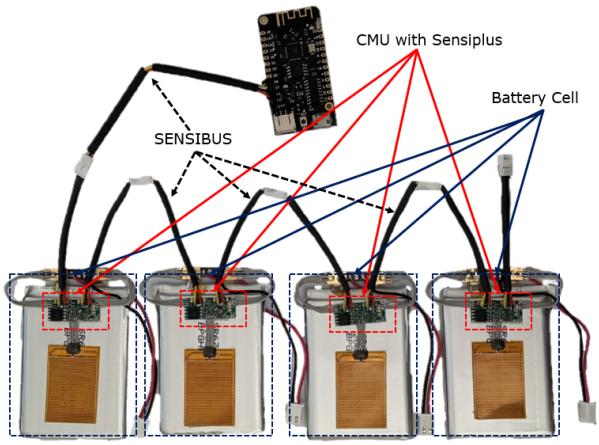

Fig. 5. Prototype application example of monitoring several cells within a battery pack, with as many CMUs and Sensiplus as cells to be monitored.

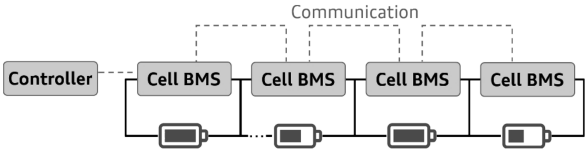

Fig. 6. Working principle of monitoring several cells within a battery pack, with as many Battery Management System (BMS) as there are cells to be monitored. Each BMS inside contains a CMU with its Sensiplus.

IV. Conclusions

In this study, we introduced the Sensibus protocol as an improvement in the domain of multi-sensor readout systems, highlighting its capabilities for multicast, unicast, and broadcast communication, contributing to better implementation efficiency and faster readout speeds. Built on an asynchronous master-slave architecture inspired by the Maxim One Wire protocol, Sensibus effectively enhances addressing capabilities. Its deployment with the Sensiplus chip for distributed measurements demonstrates its practicality, suggesting its potential for broader distributed sensing applications. However, challenges such as scalability in extensive sensor networks and adaptability in diverse IoT environments need addressing. Future work should focus on its performance in low-power, long-range scenarios to align with the evolving IoT standards and fully capitalize on its potential for IoT communications.

Acknowledgment

This work was supported in part by the Italian Ministry of University and Research (MUR), in the frame of the "PON 2022 Ricerca e Innovazione" action, in part by the Horizon Europe MATISSE with (GA 101056674), in part by the Project ECS 0000024 "Ecosistema dell'innovazione - Rome Technopole" financed by EU in NextGenerationEU plan through MUR Decree n. 1051 23.06.2022 PNRR Missione 4 Componente 2 Investimento 1.5 - CUP H33C22000420001, and in part by MOST, the Italian National Center for Sustainable Mobility, supported this research activity, funded by the Italian Ministry of University and Research in 2022-2025 (CUP H38H22000300001).

References

[1] R. R. Pahlevi, M. Abdurohman *et al.*, "Fast uart and spi protocol for scalable iot platform," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2018, pp. 239–244.

[2] P. Visconti, G. Giannotta, R. Brama, P. Primiceri, R. de Fazio, and A. Malvasi, "Operation principle, advanced procedures and validation of a new flex-spi communication protocol for smart iot devices," *International Journal on Smart Sensing and Intelligent Systems*, vol. 10, no. 3, pp. 1–45, 2017. [Online]. Available: https://doi.org/10.21307/ijssis-2017-222

[3] L. N. Pintilie, T. Pop, I. C. Gros, and A. M. Iuoras, "An i2c and ethernet based open-source solution for home automation in the iot context," in *2019 54th International Universities Power Engineering Conference (UPEC)*. IEEE, 2019, pp. 1–4.

[4] T. Fischer, D. Pirker, C. Lesjak, and C. Steger, "Tinyi2c-a protocol stack for connecting hardware security modules to iot devices," in *2020 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*. IEEE, 2020, pp. 1–7.

[5] P. Visconti, G. Giannotta, R. Brama, P. Primiceri, A. Malvasi, and A. Centuori, "Features, operation principle and limits of spi and ic communication protocols for smart objects: A novel spi-based hybrid protocol especially suitable for iot applications," *International Journal on Smart Sensing and Intelligent Systems*, vol. 10, no. 2, pp. 1–34, 2017. [Online]. Available: https://doi.org/10.21307/ijssis-2017-211

[6] M. V. K. Reddy, K. S. Achyuth, and S. Babu, "Patient health monitoring system using iot," in *2022 8th International Conference on Smart Structures and Systems (ICSSS)*. IEEE, 2022, pp. 01–04.

[7] N. Al Bassam, S. A. Hussain, A. Al Qaraghuli, J. Khan, E. Sumesh, and V. Lavanya, "Iot based wearable device to monitor the signs of quarantined remote patients of covid-19," *Informatics in medicine unlocked*, vol. 24, p. 100588, 2021.

[8] S. Basu, S. Saha, S. Pandit, and S. Barman, "Smart health monitoring system for temperature, blood oxygen saturation, and heart rate sensing with embedded processing and transmission using iot platform," in *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*. Springer, 2020, pp. 81–91.

[9] A. Ria, M. Cicalini, G. Manfredini, A. Catania, M. Piotto, and P. Bruschi, "The sensiplus: A single-chip fully programmable sensor interface," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2021, pp. 256–261.

[10] A. Ria, A. Motroni, F. Gagliardi, M. Piotto, and P. Bruschi, "An iot sensor platform for led-based optical spectroscopy," *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–4, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260385740

[11] L. Gerevini, G. Cerro, A. Bria, C. Marrocco, L. Ferrigno, M. Vitelli, A. Ria, and M. Molinara, "An end-to-end real-time pollutants spilling recognition in wastewater based on the iot-ready sensiplus platform," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 1, pp. 499–513, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157822004360

[12] G. Manfredini, A. Ria, P. Bruschi, L. Gerevini, M. Vitelli, M. Molinara, and M. Piotto, "An asic-based miniaturized system for online multi-measurand monitoring of lithium-ion batteries," *Batteries*, vol. 7, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2313-0105/7/3/45