WhiteBus: A Platform Independent Plug-and-Play Interface for IoT Infrastructures

Galal Hassan, Abdulmonem M. Rashwan, Hossam S. Hassanein
School of Computing
Queen's University, Kingston, ON, Canada
Email: {ghassan, arashwan, hossam}@cs.queensu.ca

***Abstract*—The IoT industry involves a wide diversity of sensing and control transducers, each using a different type of interfacing and operation technology. Such diversity creates a wide range of challenges to large-scale IoT development and deployment. Despite the numerous research efforts in current literature that aim to provide solutions for IoT transducer interfacing, there remains a massive lack of critical features that enable the broad adoption for large-scale applications. In this paper, we introduce a dynamic transducer interface, WhiteBus, that has the simplicity of bare-metal interfacing for IoT developers yet offers low-cost transducer integration flexibility for the manufacturers. WhiteBus exposes multiple peripherals from an IoT device to the connected transducers using a compact bus interface, mapping only the peripherals required by each transducer. Thus allowing manufacturers to reduce the time to market and reach a larger adoption audience without the need for integrating complex interfacing controllers, instead, only adding a memory unit to the transducer. We illustrate WhiteBus architecture and operational flow while outlining how it simplifies the integration and improves the cost-effectiveness for both IoT transducer manufacturers and platform developers. We then qualitatively compare WhiteBus with selected interfacing efforts for IoT, illustrating their essential differences and noting WhiteBus applicability in the IoT industry.**

***Index Terms*—IoT; PnP; Smart Sensors; Physical Bus; Configurable Interface; Multiport Interface; I/O Peripherals; Peripherals Multiplexing; USB; IEEE 1451;**

I. Introduction

Thousands of companies are increasingly interested in the IoT industry every year, causing an expected growth of around 20 billion IoT devices come 2020 [1]. The current IoT market is monopolized by IoT manufacturers that develop solutions, prototypes, and proprietary Software Development Kits (SDKs) [2], creating an increased lack of interoperability between the different platforms. Such disconnect throttles the advancement of IoT and in turn delays the realization of the fourth industrial revolution known as Industry 4.0 (I4.0) [3].

In 1996, the Universal Serial Bus (USB) revolutionized the interfacing between personal computers and peripheral devices. The self-configuration feature reduced the users involvement to get a peripheral device up and running, making it the industry standard used by most peripheral and computer manufacturers [4]. Another feature that proved lucrative for users was Hot Plugging, where a USB device does not require a system reboot after being plugged or unplugged [5].

Unfortunately, USB is very resource demanding, making it impractical for transducer manufacturers [6]. Therefore, manufacturers often use a lower rate, simpler interface bus. Some of the popular choices among manufacturers are I^2C, SPI, UART, Analogue and Digital are some of the popular choices among manufacturers [7]–[10]. Using one of these interfaces, manufacturers are restricting an IoT system developer with their choice of communication bus for transducer interfacing. Hence, complicating the development of a new IoT system, reducing the flexibility of using different sensors post-production and intensifying the risk of investment for shareholders. Therefore, having the ability to seamlessly plug and unplug a transducer without the need to reconstruct the system architecture and re-interface with the IoT devices microcontroller unit (MCU) would reduce the development time and stakeholders risk significantly. Due to the diversity of interfaces and transducer manufacturers, a gap between plug-and-play (PnP) functionality and IoT applicability exists. However, PnP capability is essential for IoT as it will provide researchers with a simple prototyping mechanism, thus accelerating the research and development of IoT to realize the vision of Industry 4.0.

In response to this gap, we introduce a PHY layer interface bus (WhiteBus) that provides PnP capability tailored for IoT transducers. Our bus combines multiple interfaces into a single multiplexed interface that is compatible with a vast range of transducers both digital and analogue. WhiteBus initially introduces four primary modes of operation for ease of adoption in the industry namely, red, yellow, magenta, and orange. Each of these modes supports one or more of the standard interfaces, making WhiteBus easily backwards compatible with current transducers. Our research also enables PnP capability through the ability to identify sensors once they are connected using an onboard memory unity. Finally, the ability to dynamically multiplex the interfaces on-demand maximizes the utilization of all the IoT device MCUs peripherals, improving the flexibility and scalability of an IoT system.

The remainder of this paper is organized as follows. Section II outlines a few standards and efforts on implementing a PnP interface. Section III introduces WhiteBus outlining its architecture and operational flow while illustrating its benefits to both sensor manufacturers and IoT system designers. Section IV qualitatively compares WhiteBus against selected PnP and interfacing efforts in the literature. Section V outlines the challenges and open concepts. Finally, Section VI concludes the paper and provides an outline of future directions.

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

II. Background

Developers and system designers use a heterogeneous variety of tools and sensors to create their IoT systems. Such diversity reduces the flexibility of substituting or upgrading sensors connected to an IoT system. Therefore, a need for a PnP solution that is optimized for IoT is born [11]. In the quest towards finding a PnP solution that simplifies the design efforts of both the manufacturers and IoT platform designers, we review the efforts put forth in the literature regarding the overhead incurred on the designing and manufacturing processes.

A. Universal Serial Bus

USB has been the standard for PnP functionality since its introduction in the early 1990s [4]. However, such technology did not spread into the IoT and embedded system realms due to its impracticality on such resource-limited platforms [12]. USB requires a dedicated chip/microcontroller on the sensor to be connected with the IoT system, introducing a higher level of complexity for both the manufacturer and the system designer [13]. Moreover, such a requirement increases the cost and energy consumption of using USB interface. In order for a system designer to integrate USB interfacing, a driver must be developed for each separate sensor, given that the MCU on the IoT system supports USB natively [13].

B. IEEE 1451

In an effort to standardize a PnP solution for analogue transducers, the IEEE 1451 standard was introduced at the beginning of the new millennia [14]. The standard proposed a unified bus for connecting analogue transducers by requiring a memory unit to be attached to each transducer. The memory unit holds a Transducer Electronic Data Sheet (TEDS) that is responsible for identifying the transducer as well as providing the Network Capable Application Processor (NCAP) with specific parameters to be used in calculating the sensed data. Although being still active, the standard was only adopted by a handful of sensor manufacturers. A key reason is that the initial design was aimed at analogue transducers, which only a handful of manufacturers support. The standard required all transducer manufacturers to integrate a memory unit loaded with a TEDS file along each of their transducers. Not only was this expensive at the time, but it was also impractical given that memory units had far less capacity and were not as miniaturized as they are today. The NCAP is developed by a system designer, and out of the scope of the sensor manufacturer, hence the signal conditioning and processing are not implemented by the manufacturers. This standard intensifies the necessary development from the system designers side while preventing the manufacturers from ensuring the intended implementation of their sensors [15].

C. mikroBUS

As an alternative to USB interface for embedded systems, researchers at MikroElektronika proposed a standard (mikroBUS) for add-on boards. The standard describes a pin configuration layout to be used by add-on board manufacturers, consisting of 2x8 female headers. MikroBUS is now a popular choice among manufacturers in order to support add-on functionality on embedded systems. However, in order to comply with the standard, sensor manufacturers are required to use the full 2x8 headers in the add-on board even if not all pins are required. Such requirement increases the cost and size of the sensors, as well as adds a considerable overhead on the manufacturers. From the system designers perspective, using mikroBUS either limits the design options to platforms that support the interface or requires the designer to integrate the standard interface as part of the design process. Not only do complicate the design procedure, but they also require the designer to write code for each board to be connected.

D. Other Efforts

Efforts that aim to provide PnP functionality for IoT devices and embedded systems require the use of an extra microcontroller to implement the communication protocol. In [16], the authors propose a modular PnP architecture technology that is similar to USB but optimized for embedded systems. In their research, they require each sensor or actuator to include a dedicated driver chip (microcontroller) in order to implement the communication protocol for the proposed interface bus. Their effort has the benefit of improvement over USB for embedded systems and IoT devices. However, the overhead incurred on the sensor manufacturers remains an issue.

The reviewed efforts, seem to implement a partial solution towards PnP functionality that is optimized for IoT platforms. However, a significant burden is incurred on both the system designer and the sensor manufacturers in order to implement the proposed solutions. Nonetheless, the efforts that seem to offer a full PnP capability are not optimized for IoT or embedded system applications. WhiteBus aims to expose the current standard interfaces to the sensors through a unified bus, significantly reducing the burden on the designers and manufacturers. Such property combines the simplicity of operation in USB with the simplicity and flexibility of design in bare-metal interfacing.

III. WhiteBus

WhiteBus is a dynamic interface bus that combines different popular interface buses into a single dynamically modifiable bus. Such flexibility enables PnP functionality that is optimized and targeted at IoT applications while maintaining platform independence.

Our interface bus aims to minimize the burden on system developers and maximize the return on investment for the companies; this is possible due to the ability to develop a genuinely generic IoT device that can be customizable during runtime by connecting different sensors. Developers can use any sensor on the market from any manufacturer and connect it to their IoT device with minimal configuration.

There are three main parts to WhiteBus, the dynamic WhiteBus Universal Serial Interface (WBUSI), the WhiteBus

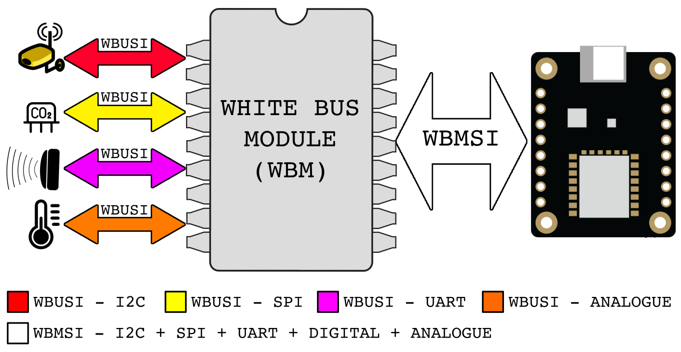
WBUSI
WBUSI
WBUSI
WBUSI
WHITE BUS MODULE (WBM)
WBMSI
🟥 WBUSI - I2C 🟨 WBUSI - SPI 🟪 WBUSI - UART 🟧 WBUSI - ANALOGUE
□ WBMSI - I2C + SPI + UART + DIGITAL + ANALOGUE

Fig. 1. Illustration of the WhiteBus Architecture Concept.

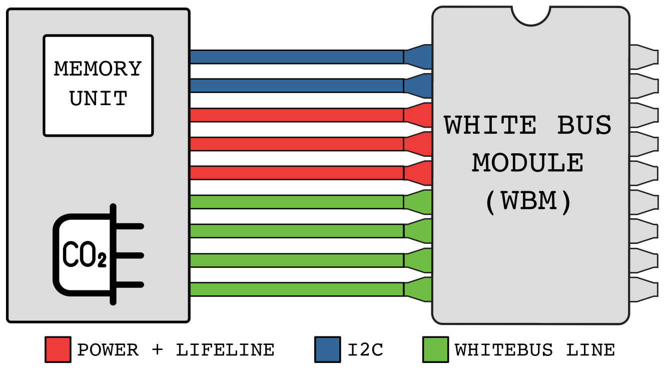

Fig. 2. Details of a White Bus Interface.

Module (WBM), and the WhiteBus Master Serial Interface (WBMSI). Each part is discussed further below.

A. Architecture Overview

In order to add WhiteBus support to current sensors and sensor nodes, a WBM needs to be used. Fig. 1 illustrates how a WBM facilitates interfacing between a generic MCU and multiple sensors through WBUSI and WBMSI. One of the motivations behind developing WhiteBus is to facilitate PnP functionality. Therefore, sensor identification is a crucial requirement. A concept similar to the TEDS from the IEEE 1451 standard is to be used in a sensor in order to be supported by a WBM, in which a memory chip attached to the sensor contains a descriptor file (written by the sensor manufacturer). The descriptor file is used by WhiteBus to identify the type of sensor, manufacturer, interface, as well as other metrics to be used by the MCU. Such information is useful to be able to map and address the correct interfaces to the connected sensor.

A WhiteBus Universal Sensor Interface (WBUSI) consists of nine lines, illustrated in Fig. 2. Since each sensor houses a memory chip, it is essential that all memory chips use a I^2C interface, hence the SDA and SCL lines are fixed for all sensors and are used to identify the connected sensor. The LifeLine is used to detect connection and disconnection of the sensors, while four WhiteBus Lines (WBL) are used to support the different modes of operation. Common power supply lines are included in operating the sensor.

The WhiteBus Module (WBM) consists of three modules: address translation, interface mapping, and sensor mapping. Fig. 3 illustrates the internal architecture of a WBM. The function and operation of each module is described as follows:

- **Address Translation:** The address translator receives serial data from the master and converts it into parallel data that is fed to the interface mapper and sensor mapper. The translation operation divides the received address into different parts and uses a lookup table to match each part to its appropriate function.
- **Interface Mapping:** The interface mapper receives multiple different interfaces from the master in order to be connected with the appropriate sensors, as well as an interface map code from the address translator. The interface mapper then selects the appropriate interfaces based on four distinct interfacing modes, magenta, orange, yellow, and red. Each of these modes involves a combination of the input interfaces to be connected to the input of the sensor mapper.
  Red Mode: I^2C and LifeLine
  Yellow Mode: SPI, I^2C and LifeLine
  Magenta Mode: UART, I^2C and LifeLine
  Orange Mode: A/D, I^2C and LifeLine
- **Sensor Mapping:** The sensor mapper receives a single white bus interface from the interface mapper and outputs multiple white bus interfaces, each to be connected to a different sensor. A sensor map code received from the address translator is used to activate one of the sensors white bus interfaces with the input white bus interface.

Finally, the WhiteBus Master Serial Interface (WBMSI) consists of 15 lines. The interface connects the WBM with the common serial interfaces on an IoT device in order to connect the IoT device to any sensor. The interfaces supported by a WBM are I^2C, SPI, UART, analogue and digital input/output. WBMSI allows unconnected interfaces such that if an IoT device does not have one of the interfaces mentioned earlier, it is still able to utilize the benefits of WhiteBus.

B. Operation Concept

An IoT system that uses WhiteBus would connect an IoT sensor node with multiple sensor elements using a WBM. A WhiteBus system is based on a star topology with a master initiating and controlling all communications with the slaves. In an IoT application, the sensor node and sensor elements are the master and slaves respectively. The master schedules the communication with all the current connected sensors and ensures that the time slot each sensor receives is fair.

Any PnP system requires three features, the ability to detect connection and disconnection, the ability to identify the connected sensor, and driver support for possible sensors to be connected [17]. Current IoT sensor nodes have driver implementations for the supported sensors, but no method to easily plug and unplug these sensors without significant reprogramming. Therefore, WhiteBus provides a solution to support the other features.

The master keeps a list of all currently connected sensors and schedules the communication with each of them. To

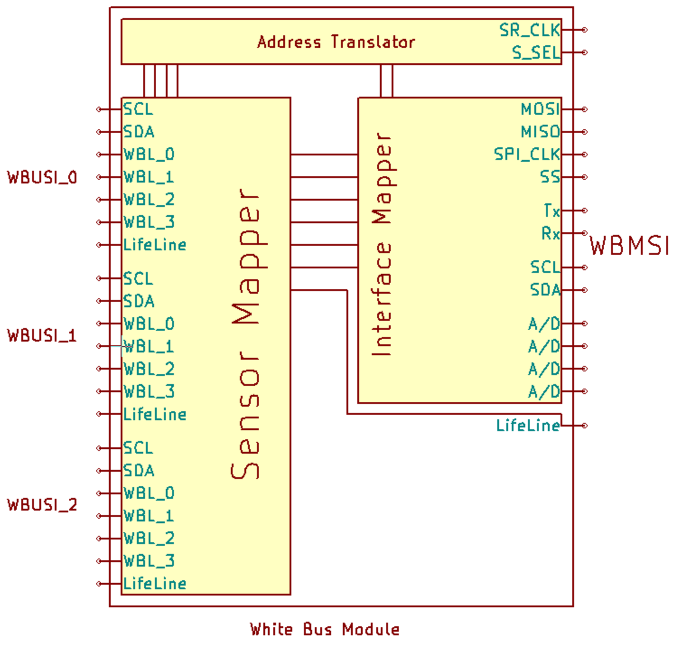

Fig. 3. The internals of a White Bus Module.

initiate the connection with one of the sensors, the master requests the connection from the WBM which in turn maps the correct interfaces to the requested sensor, providing the master with a direct connection to the sensor. Afterwards, in order to detect connection or disconnection, the master polls the LifeLine connected to the sensor in question.

In order to identify the connected sensor, the master communicates with the memory embedded on the sensor using the I^{2}C interface of WhiteBus and reads the descriptor file preloaded by the manufacturer. The descriptor file, similar to TEDS, contains information about the sensor manufacturer, the type of sensor, the interface required, the type of data the sensor provides, and a description of any required calculations to be performed on the sensor readings. Once the descriptor file is read, the master allows the sensor driver to communicate directly with the sensor, making the WBM transparent to the sensor driver. Fig. 4 illustrates the operation flow of a WhiteBus system.

IV. System Assessment

In order to ensure the applicability of WhiteBus, we use four key properties to compare it with selected PnP and interfacing efforts qualitatively as shown in Table 1. We focus on key differences such as the cost/complexity to integrate, the impact on the IoT node energy consumption, the integration/development process, the sensor scalability, and PnP support.

A. Energy Impact

One of the major considerations for any IoT device is the energy consumption and lifetime of the device. Therefore, it is crucial to ensure that the benefits brought by using WhiteBus will not impact the overall lifetime of the device. We discuss the primary sources of energy consumption when using an interface bus, and evaluate the expected impact of using WhiteBus.

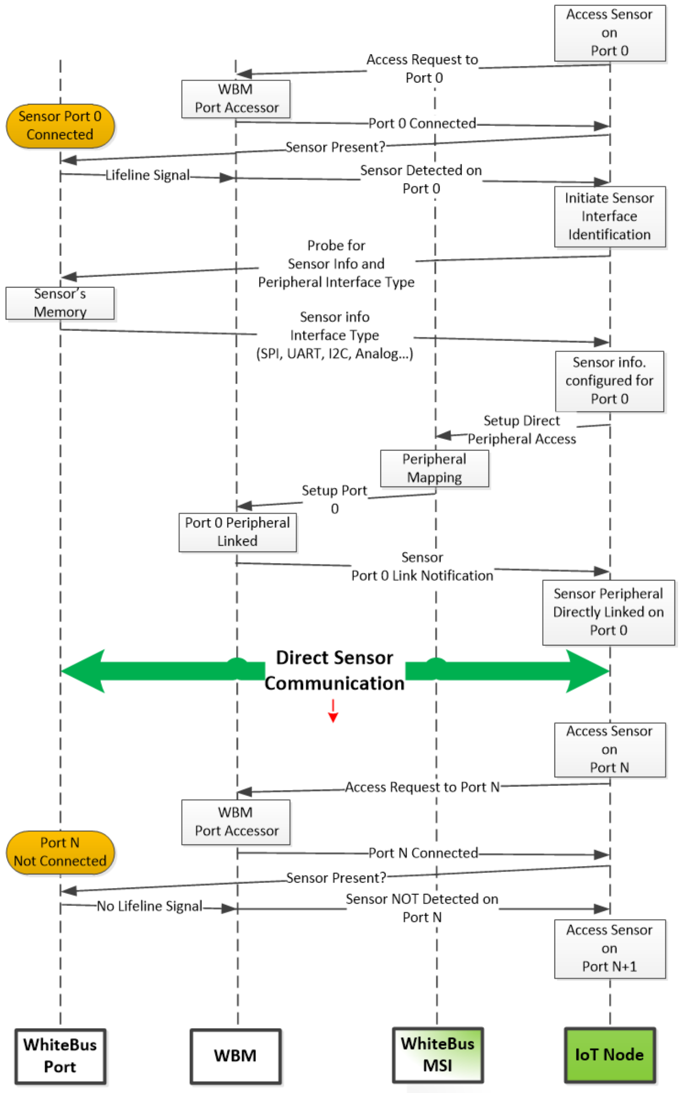
Access Sensor on Port 0
Access Request to Port 0
WBM Port Accessor
Sensor Port 0 Connected
Port 0 Connected
Sensor Present?
Lifeline Signal
Sensor Detected on Port 0
Initiate Sensor Interface Identification
Probe for Sensor Info and Peripheral Interface Type
Sensor's Memory
Sensor info Interface Type (SPI, UART, I2C, Analog...)
Sensor info. configured for Port 0
Setup Direct Peripheral Access
Peripheral Mapping
Setup Port 0
Port 0 Peripheral Linked
Sensor Port 0 Link Notification
Sensor Peripheral Directly Linked on Port 0
Direct Sensor Communication
Access Sensor on Port N
Access Request to Port N
WBM Port Accessor
Port N Not Connected
Port N Connected
Sensor Present?
No Lifeline Signal
Sensor NOT Detected on Port N
Access Sensor on Port N+1
WhiteBus Port
WBM
WhiteBus MSI
IoT Node

Fig. 4. Operation Flow of WhiteBus.

When a developer decides not to use a bus system, each sensor is connected to the IoT device directly through one of the interfaces provided by the MCU of the device. Such a bare-metal method entails the design of the circuitry used for each interface to be implemented by the developer. Hence, energy impact of interfacing a specific sensor relies on the developers hardware design. For our research, we assume that the developers hardware design is optimized and does not involve current leakage issues. We use this method as a benchmark for comparison with other interface bus systems.

MikroBuss energy impact results from the manufacturers hardware design of the sensor board. A best-case scenario would be equivalent to the bare-metal method in which the MikroBus sensor board is optimized regarding hardware design.

In order to use USB on an IoT device, a microcontroller is required on the sensors side, which entails a higher energy impact. USB also uses a 5v level instead of the common 3.3v used in IoT and embedded systems.

Since the authors in [16] do not measure the energy impact

TABLE I
QUALITATIVE ASSESSMENT OF SELECTED BUSES

		Bare-Metal	**USB**	**WhiteBus**	**mikroBUS**
Energy Impact		LOW	HIGH	MEDIUM	LOW
Cost & Complexity	Manufacturer	LOW	HIGH	MEDIUM	MEDIUM
	System Designer	HIGH	HIGH	MEDIUM	MEDIUM
Sensor Scalability		LIMITED	HIGH	HIGH	LIMITED
PnP Support[1]		NO	YES	YES	NO

of their proposed research, we assume that the use of an FPGA device would drastically elevate the energy consumption.

With WhiteBus, energy consumption in a best-case scenario is equivalent to that of the regular bare-metal implementation, along with the additional energy consumption incurred by the presence of the WBM. Such an impact on energy consumption, however, remains marginally less than that of USB, because WhiteBus does not require any extra microcontrollers.

B. Cost and Complexity

The complexity sustained from an interfacing method directly affects the cost of implementation for both sensor manufacturers and system designers. Therefore, it is essential to design an interface bus with reduced complexity in mind. We discuss the complexity of each interfacing method to evaluate the complexity of WhiteBus qualitatively.

A bare-metal interfacing method has the lowest complexity for sensor manufacturers, in which a manufacturer can design a sensor using any interfacing system and delegate all the development complexity solely on the developer. However, this results in a daunting list of tasks on the system designers side, significantly shifting the complexity and cost away from the manufacturer towards the designer. Such a task list usually involves the design of the circuitry required for each type of interface. By requiring a specific bus interface on both the sensor and IoT system, mikroBUS and USB divide the complexity and cost between the system designers and the sensor manufacturers. However, mikroBUS provides a combined bus interface that includes dedicated pins of the common peripherals, while USB uses a standardized bus along with sensor drivers to provide a universal abstraction of data/control regardless of the sensors native peripherals. Therefore, mikroBUS requires large pin-out footprint while USB requires a microcontroller capable of its protocol on both the sensors side and the IoT systems side.

WhiteBus requires an EEPROM for identification purposes to be integrated on the sensor, hence, shifting most of the burden on the IoT system designers side. Such design choice provides flexibility to the sensor manufacturers while maintaining the same level of control on the designers side. WhiteBus also reduces the burden on the designer with the use of the WBM, in which the peripheral mapping is abstracted from the system designer and handled by the WBM. Another benefit of using a WBM is the reduction of pins compared to that of mikroBUS.

[1]PnP capabilities for both USB (Section II) and WhiteBus (Section III) have been discussed earlier.

C. Sensor Scalability

One of the significant benefits of using an interface bus is the ability to connect a greater number of peripherals than what the system hardware is capable of supporting. This is one of the key reasons behind the popularity of USB [4].

A bare-metal approach limits the sensor scalability of an IoT device to the hardware interface capabilities. Therefore, the use of a bus system increases the number of sensors the IoT device can interact.

The mikroBUS standard enforces the use of standardized pin-mapped headers on the sensor and the IoT device. In some cases, some of the peripherals are not shared between sensor boards (i.e. I^2C), therefore, in a best-case scenario, the mikroBUS scalability is equivalent to that of bare-metal implementation.

USB and WhiteBus use a similar approach for scalability, in which a master schedules different time-slots between the connected slaves. Regarding the bus interface, USB uses hubs to extend the number of slaves connected at a particular time, while WhiteBus utilizes a WBM that supports a limited number of connected slaves at a time using the WBUSI ports. In order to increase the number of connected slaves, a different WBM with a more significant number of WBUSI ports would be needed.

V. CHALLENGES & FUTURE DIRECTIONS

IoT microcontrollers are diverse and can have from six pins to hundreds of pins [18]. Therefore, controllers with a low number of pins usually lack the support of all the peripherals supported by WhiteBus (e.g., lack of SPI or UART functionality). Moreover, some microcontrollers lack the support of peripheral logic, due to design cost constraints regardless of the number of pins. Since WhiteBus is designed to provide peripheral mapping and multiplexing to sensor ports, it is limited by the available peripherals on the microcontrollers. Also, the support of I^2C is crucial to the operation of WhiteBus as it is used to access the sensors memory units.

Fortunately, the implementation of the software-based I2C driver is simple and has a minimal footprint on most microcontrollers. This implementation would allow WhiteBus to function effectively even with the the lack of other peripherals. The developers, in this case, can read the sensor interface information and determine the feasibility of supporting such a sensor. Developers are also able to incorporate the use of other software-based peripheral drivers, and then link the corresponding pins to the WhiteBus MSI, given that the microcontrollers can support them.

Another challenge lies in the diversity of voltage levels in sensors, the common 3.3v level used in IoT is not supported by every sensor. Such a difference in voltage level requires a power management unit to supply the required supply voltages. We believe that the sensor manufacturer should implement the voltage level change circuitry into their sensors, which in turn increases the cost, however, remains a better option than incorporating a microcontroller for interfacing in systems such as USB.

A third challenge relates to the advanced sensors that usually incorporate multiple feedback (i.e., interrupt) lines. WhiteBus limits the number of feedback lines available to the sensor based on the peripheral mode supported by the sensor, for instance, the magenta and red modes provide only two lines to be used for feedback. Sensors that require multiple feedback lines may require the manufacturer to integrate an I^2C to I/O bridge in order to expose the feedback lines over the same I^2C bus used by sensors memory, however, with a different address. The sensor EEPROM will include, in this case, the identification and address for the I^2C to I/O bridge. It is the responsibility of the IoT developers to pull the feedback status via the bridge to check for the sensor status.

VI. Conclusion

In order to excel the vision of Industry 4.0 rapid development in IoT is required. One of the factors that throttle the advancement in IoT is the lack of standardization in the industry. Therefore, a solution such as PnP is in demand to overcome this heterogeneity.

In response to that demand, we proposed a dynamic peripheral interface (WhiteBus) that simplifies the integration of transducers to any IoT system. WhiteBus combines the peripheral interfaces available on an IoT device and connects only the peripheral a specific transducer uses, abstracting the actual interface to the developer and enabling PnP functionality. Our research requires an EEPROM to be integrated with the transducer similar to the IEEE 1454 standard, allowing the IoT device to identify the connected transducer.

We compared WhiteBus to notable systems in the literature and found that WhiteBus provides a good balance between scalability, complexity, and power consumption. We also noted that WhiteBus introduces some challenges, for instance, when using limited functionality microcontrollers, we proposed the use of software-defined peripherals. The use of sophisticated transducers might require extra digital feedback connections that are not supported by WhiteBus. Hence, we proposed the use of additional simple circuitry on the transducer to seamlessly support WhiteBus.

Our research only scratches the surface of possibilities with WhiteBus, hence we consider some future directions that would improve the interface. It is possible to add self-configurable PnP functionality to WhiteBus by migrating the driver software from the IoT device to the transducer memory unit. Another ambitious direction towards simplifying sensor integration is implementing an interface translation unit within the WBM. The purpose of this unit is to allow any sensor to interface with any IoT device regardless of the available interfaces.

VII. Acknowledgement

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: STPGP 479248.

References

[1] M. Hung, “Leading the iot, gartner insights on how to lead in a connected world,” *Gartner Research*, pp. 1–29, 2017.

[2] A. Polianytsia, O. Starkova, and K. Herasymenko, “Survey of hardware iot platforms,” in *Problems of Infocommunications Science and Technology (PIC S&T), 2016 Third International Scientific-Practical Conference*. IEEE, 2016, pp. 152–153.

[3] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, “Industry 4.0: The future of productivity and growth in manufacturing industries,” *Boston Consulting Group*, vol. 9, 2015.

[4] P. Merolla, J. Arthur, and J. Wittig, “The usb revolution,” *The Neuromorphic Engineer*, vol. 2, no. 2, pp. 10–11, 2005.

[5] U. S. B. Specification, “Revision 1.0,” 1996.

[6] D. Anderson and D. Dzatko, *Universal Serial Bus System Architecture*, 2nd ed., ADDISION-WESLEY, Ed. MindShare INC, 2001.

[7] *Digital 16bit Serial Output Type Color Sensor IC*, ROHM Semiconductor, 5 2016, rev. 2.

[8] *Thermal Sensor with SPI Interface*, MICROCHIP, 2012.

[9] *Intelligent 9-axis absolute orientation sensor*, BOSCH, 6 2016, rev. 1.4.

[10] *Heavy Duty Pressure Transducers - Sensing and Internet of Things*, HONEYWELL, 2018, issue B.

[11] M. Suárez-Albela, P. Fraga-Lamas, T. M. Fernández-Caramés, A. Dapena, and M. González-López, “Home automation system based on intelligent transducer enablers,” *Sensors*, vol. 16, no. 10, p. 1595, 2016.

[12] S. Pitzek and W. Elmenreich, “Plug-and-play: Bridging the semantic gap between application and transducers,” in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, vol. 1. IEEE, 2005, pp. 8–pp.

[13] G. Liao, Q. Lu, and W. Zhang, “Design of reusable software for usb host driver in embedded system,” in *Computing, Control and Industrial Engineering (CCIE), 2010 International Conference on*, vol. 1. IEEE, 2010, pp. 312–315.

[14] T. R. Licht, “The ieee 1451.4 proposed standard,” *IEEE Instrumentation & Measurement Magazine*, vol. 4, no. 1, pp. 12–18, 2001.

[15] D. Hernández-Rojas, T. Fernández-Caramés, P. Fraga-Lamas, and C. Escudero, “A plug-and-play human-centered virtual teds architecture for the web of things,” *Sensors*, vol. 18, no. 7, p. 2052, 2018.

[16] M. Pancani, “Architecture for plug-and-play modular technology,” *2015 NCUR*, 2015.

[17] K. Mikhaylov, T. Pitkaäho, and J. Tervonen, “Plug–and–play mechanism for plain transducers with wired digital interfaces attached to wireless sensor network nodes,” *International Journal of Sensor Networks*, vol. 14, no. 1, pp. 50–63, 2013.

[18] *6-Pin, 8-Bit Flash Microcontrollers*, MICROCHIP, 2014.