IOP Conf. Series: Journal of Physics: Conf. Series **870** (2017) 012002 doi:10.1088/1742-6596/870/1/012002

Legacy to Industry 4.0: A Profibus Sniffer

Fesseha Tsegaye Mamo[1, a], Axel Sikora[1,a], Christoph Rathfelder[1,a]

[1]Hahn-Schickard, Software Solutions, 78052 Villingen-Schwenningen, Germany

f.mamo@hahn-schickard.de; christoph.rathfelder@hahn-schickard.de; axel.sikora@hs-offenburg.de

Abstract. Legacy industrial communication protocols are proved robust and functional. During the last decades, the industry has invented completely new or advanced versions of the legacy communication solutions. However, even with the high adoption rate of these new solutions, still the majority industry applications run on legacy, mostly fieldbus related technologies. Profibus is one of those technologies that still keep on growing in the market, albeit a slow in market growth in recent years. A retrofit technology that would enable these technologies to connect to the Internet of Things, utilize the ever growing potential of data analysis, predictive maintenance or cloud-based application, while at the same time not changing a running system is fundamental.

1. Introduction

Legacy industrial communication protocols are proved robust and functional. During the last decades, the industry has invented completely new or advanced versions of the legacy communication solutions. However, even with the high adoption rate of these new solutions, still the majority industry applications run on legacy, mostly fieldbus related technologies. Despite its age, Profibus is one of those technologies which still are very popular in the market. [1] A retrofit technology enabling these technologies to connect to the Internet of Things, would open the ever growing potential of data analysis, predictive maintenance or cloud based applications, while at the same time not changing a running system.

In order to achieve the above described technology, it is necessary to adapt existing communication protocols which are not capable of seamless interconnection to IP based networks. Profibus is one of these protocols which rely on RS485 physical medium for majority of its use cases [2]. In this paper we describe a technique to demonstrate such a possibility of physical and logical interconnections. The demonstrator focuses on a Profibus sniffer that passively collects data from the communication bus, without interfering with the system, and that forwards relevant data to a server for storage and further analysis and manipulation of the collected data.

This paper initially takes a look at comparable solutions and the Profibus protocol. In later sections the specific setup, hardware and software solutions from this work are described in brief.

2. Project Background

The objective of the presented work is part of a larger project. The project named in short NIKI4.0 (Nicht-disruptives Kit für die Evaluation von Industrie 4.0) [3] is a cooperative project between Hahn-Schickard [4], Institute of reliable embedded systems and communication (IvESK) [5] at the University of Applied Sciences Offenburg, and Karlsruhe Forschungszentrum Informatik (FZI) [6]. Many industrial partners also take part in the project. The project is funded by the Baden-Württemberg-Stiftung [7].

The project aims at equipping industries with a retrofit solution by fitting production sites with different kind of sensors, information monitoring tools (Profibus sniffer), and extended visualization

[a] f.mamo@hahn-schickard.de, axel.sikora@hahn-schickard.de, christoph.rathfelder@hahn-schickard.de

mechanisms. With such a system in place it will be possible to collect valuable information that helps in better understanding and optimizing the operations of the facilities.

3. State of the Art

There currently exist few Profibus protocol sniffers. These sniffers focus on capturing raw Profibus data and analyzing the electrical behaviors of the network bus for troubleshooting purposes. Few examples are the ProfiTrace [8] and ibaBM-DP [9]. In addition to data sniffing these devices also provide analysis tools for the electrical signal on the network bus.

Using Wireshark [10] for sniffing communication protocols is the general approach. However, there are no solutions for Profibus protocol sniffing that is based on Wireshark.

4. Profibus Overview

Profibus is defined by a collection of IEC standards (cf. Figure 1) and it is one of the most successful and mostly used industrial communication protocols. Profibus operates on different physical mediums addressing the requirements in different industrial environments and use cases. Profibus defines the specification for the Fieldbus Data Link layer, the Application layer, and the Application profiles. Figure 1 summarizes the components of Profibus as represented according to the OSI model.

Profibus User Association (PNO) [11], through the years, has adopted multiple engineering tools that make installation and commissioning of Profibus devices easy. General Station Description (GSD) [12] and Electronic Device Description (EDD) [13] are examples of such solutions that describe the capabilities of a Profibus device in a generic way. They describe the hardware capabilities of a Profibus slave device starting from supported speeds to the number of input outputs on a slave module.

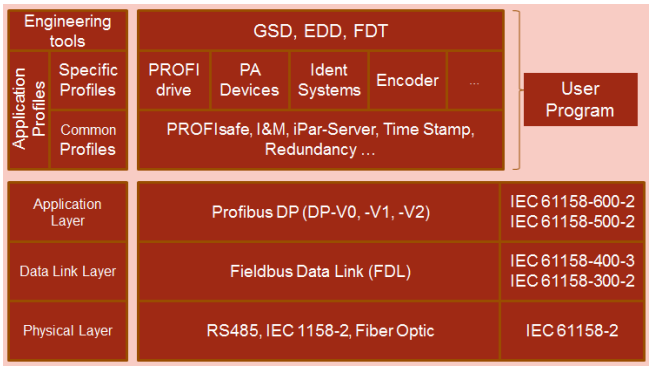

Figure 1. Profibus specification and tools

The Profibus data link layer, named the Fieldbus Datalink Layer (FDL), provides data transmission services that serve different purpose. These services for data transmission are:

- SDN - Send Data with No acknowledge
- SDA - Send Data with Acknowledge
- SRD - Send and Request Data
- CSRD - Cyclic Send and Request Data
- MSRD - Send and Request Data with Multicast Reply
- CS - Clock Synchronization

These transmission services utilize five types of packet structure. These are the telegrams without data (SD1), telegrams with variable length (SD2), telegrams with fixed data (SD3), the token telegram (SD4) and the short confirmation (SC). Actual information with payload can only be transmitted in SD2 and SD3 Fieldbus Datalink Layer (FDL) packets. The structure of these packets is shown below.

The following list of information can be acquired from a Profibus data bus:

- Diagnosis
- I/O Data
- Alarm
- Process Data
- LR (Load Register) Data
- Context
- Function Invocation Data

Of the above mentioned, diagnosis, alarm, status and I/O data are planned to be retrieved in this research. The GSD file of a Profibus device provides module information which is useful to extract meaningful I/O information from the gathered (sniffed) data.

5. System setup

One of the physical mediums supported by Profibus, as shown in Figure 1, is RS485. Since this is the mostly used physical media this work focuses on Profibus networks using RS485. It is a two wire pair differential system. Profibus slave devices provide a diagnosis port that enables monitoring or diagnosis devices to connect to it, as shown in Figure 2. Using this port and a carefully designed hardware to collect data from this port, all data on the Profibus network can be sniffed. The hardware setup is discussed in later sections (cf. ch. 6.1) of this paper.

Figure 2. SIMATIC ET200M –IM 163-2 Profibus slave diagnosis port

After the data collection, filtering of relevant traffic is the second step. Relevant traffic in this demonstrator includes diagnosis, alarm, status and I/O data. In this demonstration, once the packets are received at the end of the RS485 interface of the sniffer, the next task is to filter the SD2 and SD3 type packets. After this step modelling of the PDU is needed before transmitting to the gateway. The data is modelled using the IPSO object model; represented using either JSON or XML and the data is transmitted using CoAP which is the transport mechanism in LWM2M [14].

The Profibus sniffer collects data by tapping into the Profibus slave diagnosis port as shown in Figure 3. The data is then provided as a stream for further processing relevant information.

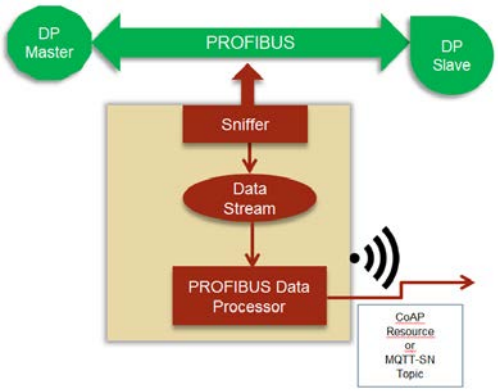

Figure 3. Profibus sniffer

6. Demonstrator

6.1 Hardware setup

Profibus specification supports speeds of up to 12 Mbps. Therefore the RS485 transceiver hardware should be able to operate up to such speed. However normal off the market RS485 to UART converters do not support such speed. In majority of Profibus devices such speeds are possible by using a targeted ASIC designed hardware.

In this research, though, a system with minimum complexity and affordable cost is attained by using the TI Sitara AM3358 processor [15] that provides the programmable real-time core (PRU) for industrial communication protocols, Figure 4. The PRU provides a UART interface with a dedicated clock of 192MHz that can operate on speeds up to 50Mbps.

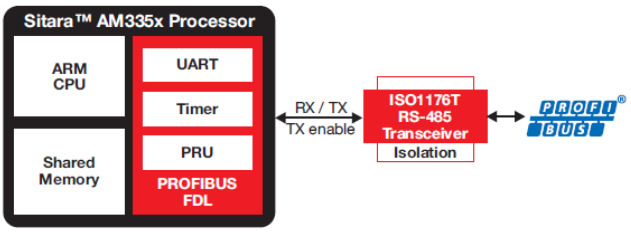

Figure 4. TI Sitara Processor [15]

Such a processor is found in the Beaglebone Black open source hardware development kit. The Profibus sniffer is built by combining the Beaglebone Black's PRU UART port with an external RS485 to UART transceiver module. This setup is shown in Figure 5.

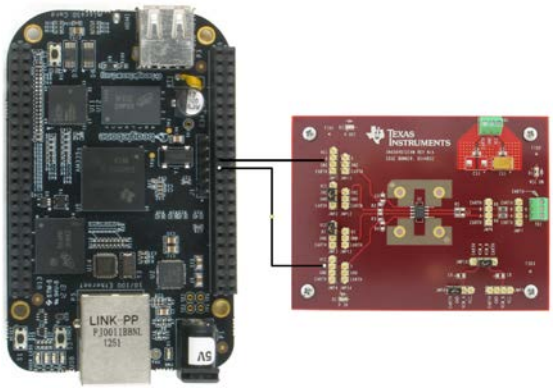

Figure 5. Sniffer Hardware

6.2 Software

DP or Decentralized Peripherals deals with decentralized I/O modules. In simple terms there are an array of input and output modules represented by single bytes and bits, representing a 1 as ON and a 0 as OFF. Different data types are described by the Profibus DP profile.

Using the sniffer hardware, an output of raw data is provided at the Linux subsystem level. This raw Profibus data has to be then fed to a parser to take out the interesting information as described in section 4 of this paper. A sample raw data with description is shown below in Figure 6.

The first packet is a packet with variable data length with start delimiter (SD) 0x68. It is a request packet, identified by the function code (FC=0x21), destined to the destination address of 3, which in this setup is a Slave device. The sender is a Master device with the address 2. The master is requesting IO data for the slot 0x0B and index of 0x04. The length of the requested IO data is signified by the following zero bytes. The 0x91 byte is the Frame Check Sequence and the last byte 0x16 is the fixed value End Delimiter field.

Figure 6. Raw Profibus data

In the case of a distributed object structure for Profibus, the instance should act as the address of the Profibus device.

e.g. /3500/2/6001 represents diagnosis information from the Profibus device with address 2

If the device in the above example reports multiple diagnosis information, a rolling amount of instances of the diagnosis data should be stored. There should be a limit to the stored instance amount in order not starve out memory.

The first approach of representing Profibus data is in a modular, functional oriented way. This means creating a separate object representation based on the type of information contained. The second type of object representation is a device based approach.

Table 1. Profibus diagnosis data according to IPSO resource model

Object	Object ID	Object URN	multiple instances?
Profibus Diagnosis	3500		Yes

res name	res ID	access type	multiple instances	M/ O	type
dev add	6000	R	No	M	Float
ident num	6001	R	No	M	Float
diag status	6002	R	Yes	M	String
identifier-based diag.	6003	R	Yes	O	String
channel-based diag.	6004	R	Yes	O	String
device-based diag.	6005	R	Yes	O	String

7. Summary

This paper and the work done under this demonstration are a proof of concept for integrating legacy communication protocols to the ever growing and demanding necessity of connecting things to the Internet.

While connecting legacy systems to the Internet seems a fairly uncomplicated task, seamlessly mapping the semantics of such systems is a very challenging task. This is so because of the simplified roles of such kind of field devices and that the actual logic of such systems runs on external systems such as PLCs. To give an example, a Profibus slave device could have multiple digital inputs and outputs which are bit wise programmed for a given functionality on the PLC. Therefore even if it is possible to get these process values by tapping on the network bus, it is impossible to understand the meaning of it unless the context of each bits and bytes are known from the PLC.

With that in mind, it is still possible to collect standard information being described by the Profibus specifications, which are context independent such as diagnosis, alert and status information.

References

1. PI Figures Specification 2014, http://www.profibus.com/nc/pi-organization/regional-pi-associations/south-east-asia/downloads-a nd-media/downloads/2015-pi-seminar/download/19049/
2. http://www.smar.com/en/profibus
3. http://fzi-forschungszentrum-informatik.github.io/NIKI40/
4. http://www.hahn-schickard.de
5. http://ivesk.hs-offenburg.de
6. https://www.fzi.de/startseite
7. https://www.bwstiftung.de/startseite
8. http://www.procentec.de/profitrace2/
9. http://www.iba-ag.com/de/produkte/produktkatalog/13121001-ibabm-dp/
10. https://www.wireshark.org
11. http://www.profibus.com/pi-organization/about-pi/
12. Profibus GSD files, http://www.profibus.com/products/gsd-files/
13. EDDL Specification; Specification for PROFIBUS Device Description, vol. 2
14. Open Mobile Alliance Lightweight M2M, http://www.openmobilealliance.org/wp/Overviews/lightweightm2m_overview.html
15. Profibus on Sitara Processors whitepaper, http://www.ti.com/lit/wp/spry155b/spry155b.pdf