Rapid prototyping framework for automation applications based on IO-Link

V. Chavez[1], F. Cruz[2], M. Ruchay[3], J. Wollert[4]

Abstract

The development of protype applications with sensors and actuators in the automation industry requires tools that are independent of manufacturer, and are flexible enough to be modified or extended for any specific requirements. Currently, developing prototypes with industrial sensors and actuators is not straightforward. First of all, the exchange of information depends on the industrial protocol that these devices have. Second, a specific configuration and installation is done based on the hardware that is used, such as automation controllers or industrial gateways. This means that the development for a specific industrial protocol, highly depends on the hardware and the software that vendors provide. In this work we propose a rapid-prototyping framework based on Arduino to solve this problem. For this project we have focused to work with the IO-Link protocol. The framework consists of an Arduino shield that acts as the physical layer, and a software that implements the IO-Link Master protocol. The main advantage of such framework is that an application with industrial devices can be rapid-prototyped with ease as its vendor independent, open-source and can be ported easily to other Arduino compatible boards. In comparison, a typical approach requires proprietary hardware, is not easy to port to another system and is closed-source.

Keywords

Rapid-prototyping, Arduino, IO-Link, Industrial Communication

1 Introduction

Integrating industrial devices such as sensors and actuators in rapid-prototyping applications or proof-of-concepts is not always straight forward. These devices transmit data typically through an industrial communication protocol. To access the information of these protocols an automation controller or industrial gateway is normally required. Their use involves a specific hardware and software that is dependent on the vendor. For rapid-prototyping scenarios, this means that developing an application depends on the hardware that is used and cannot be replicated easily with another vendor. In addition, the hardware and software provided by these vendors is not always modifiable, and adding extra features is not possible.
In this paper, we present the development of a framework that intends to address these issues and provide a rapid-prototyping framework with industrial sensors and actuators. The scope of this work is oriented towards the development of rapid-prototyping applications with the IO-Link industrial communication protocol. The frame of reference for this work is the Arduino platform which is oriented to rapid-prototyping and is vendor neutral. In the following sections the motivation for this work

---

[1] Fachhochschule Aachen
[2] Fachhochschule Aachen
[3] Fachhochschule Aachen
[4] Fachhochschule Aachen

is explained, an Introduction to IO-Link is given and the design of the framework is explained. At the end our preliminary results are shown and a conclusion and future work is provided.

2 Motivation

Our interest in developing a rapid-prototyping framework for sensors and actuators based on IO-Link is two-fold. Firstly, at our university we develop low-cost mechatronic projects with students and we have noticed the lack of open-source tools to integrate industrial sensors and actuators easily without being dependent of a specific vendor solution. Secondly, we have hands-on experience with IO-Link from the development of an Arduino based IO-Link device framework [1], and are interested in expanding it to have a complete industrial prototyping ecosystem that is accessible, hackable and easy to use for students. In addition, we intend to work on this framework and adapt it for use with the ethernet 10BASE-T1L PHY layer as part of our research from [2].

3 Related work

As far as we know, there are two rapid-prototyping environments for industrial IO-Link devices available. There is a commercial evaluation board from STMicroelectronics, compatible with the STM32 nucleo microcontroller and the Arduino UNO. It consists of an IO-Link master transceiver board that includes a software demo evaluation, and a proprietary graphical user interface that can load an IO-Link description file and view the process data of an IO-Link device. This framework is suitable to learn in general how IO-Link. However, as it consists of a closed-source software, modifications and adaptions are not possible.

The second alternative is an open-source IO-Link Master shield/hat [3]. This project consists of an electronics board with an IO-Link Master transceiver compatible with Arduino and Raspberry Pi. It also includes a software that explores some of the features of the IO-Link protocol. This project seems promising as it’s the only open-source project alternative for IO-Link but at the moment lacks IO-Link features such as parametrization of devices (on-request data) and diagnostics information (IO-Link events). From these two options we see that there still exists a gap in a framework that incorporates of all the main features of IO-Link, is open-source and can easily be adapted to any other development environment.

4 What is IO-Link?

IO-Link is a standardized single-drop digital communication interface technology (SDCI) for sensors and actuators, part of the IEC61131-9 specification [4]. IO-Link consists of at least two components, one IO-Link master and one or more IO-Link devices. The IO-Link master initiates the communication and configuration of any IO-Link device that is connected to it. In the other hand, an IO-Link device is considered the source of information about a process (e.g., sensors) or the end device to control a process (e.g., motor). The physical IO-Link interface consists of a three-wire signal consisting of two wires (L+, L-) for power supply and one wire (C/Q) for communication. The C/Q line transports information as a serial protocol that can communicate up to a speed of 230.4 kbit/s.

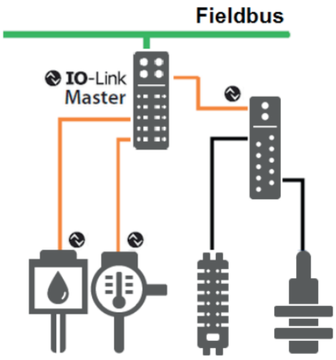


Figure 1 Typical IO-Link topology [5]

The main advantage of IO-Link is that it allows to connect sensors and actuators as plug-and-play devices. The exchange and management of information is done automatically by the IO-Link master and can be configured to download a configuration to the IO-Link device in case of replacement. The information and configuration of an IO-Link master can be accessed through its so-called Standardized Master Interface (SMI). This interface consists of a set of services that allow to control the IO-Link master, get information from the connected devices and configure it. Typically, this interface is internally built to connect over a fieldbus network or a proprietary protocol as part of a Programmable Logic Controller (PLC) module.

5 Framework

One of the most well-known prototyping frameworks for electronics and do-it-yourself projects is the Arduino open-source electronic prototyping platform [6]. Due to its compatibility with different microcontrollers, easy integration to different ecosystems and its constant development, we have chosen to develop this project based on this platform. In the following subsections, we present a view on the design process for the framework and the design requirements based on the IO-Link specification version 1.1.3 [7].

5.1 Hardware Requirements

The first step of the design process consists of identifying the hardware components that are needed to develop an IO-Link Master. One of the first, and most important components is the IO-Link Master transceiver. This transceiver acts as a bridge between the IO-Link physical layer and a microcontroller that implements the communication stack. For this prototype we selected an LTC2874 transceiver from Analog Devices. This transceiver supports communication up to four IO-Link devices, automatic generation of the wakeup signal, and configuration via a SPI interface [8].

Another part required according to the specification, is the use of 2048 bytes of non-volatile memory per IO-Link device for the so-called Data Storage (DS). For this part, we selected a 25AA128T EEPROM memory from Microchip with 16 KB of storage. The next part that is needed for the design is the microcontroller that will drive the communication logic and interact with the other components. Before discussing the selection of the microcontroller, we will discuss the hardware requirements that it needs to have according to the IO-Link specification.

The IO-Link communication relies on the principal of a universal asynchronous receiver-transmitter (UART). The maximum baud rate required for IO-Link communication is 230.4 Kb/s. The second hardware requirement is the use of timers to measure timeouts and events. Not all of the timers, are used at the same time and an analysis was done to verify the maximum number of timers required per IO-Link device. The use of timers can be divided in two phases. Firstly, there is the initialization phase, which consists of detecting whether an IO-Link device is connected or not. Secondly, there is the communication phase which can be in one of the three IO-Link states known as startup, pre-operate and operate.

For the first phase, a total of four timers were identified, according to the specification these are Tren, Tdwu, Tdmt and Tsd. By checking their usage in a timeline, it was found that two of them run sequentially (Tren, Tdwu) and the two others run in parallel (Tdwu,Tsd). To minimize the number of timers required for this phase, it was found that Tren and Tdwu can be assigned to one timer, and Tdwu and Tsd to another (see Figure 2).

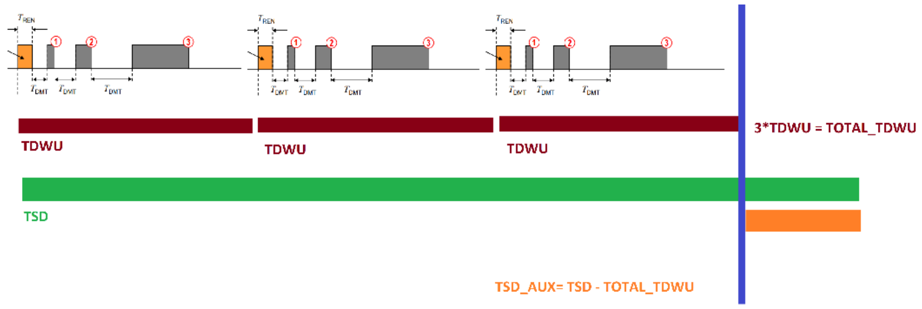


Figure 2 Visual representation of timers in the initialization phasee

The second phase, requires three timers: TMseq, Tcyc, and TIsdu. After analysing their usage in a timeline (see Figure 3), it was concluded that they have to be run in parallel. As both of these phases do not run in parallel, the maximum number of hardware timers needed is three.

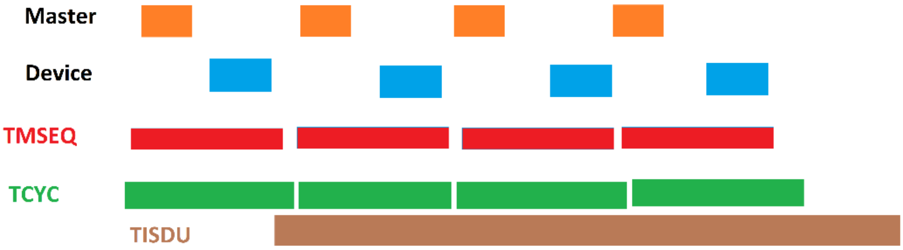


Figure 3 Visual representation of timers in the communication phasee

Based on the aforementioned hardware peripherals, the microcontroller could now be selected. As our focus is to work with the Arduino framework, the microcontroller has to be compatible with one of the available Arduino development boards. The main requirements for this board can now be discussed. The board should have four UARTs (four IO-Link devices) and three timers per IO-Link device. To the best of our knowledge one of the most suitable boards for this is the Arduino Due. The Arduino Due board has one UART, four USARTs, nine timers and a SPI peripheral. This board is based on the ATSAM3x8e microcontroller from Atmel (now Microchip), with an ARM Cortex-M3 CPU, a clock frequency of 84 MHz, 512 KB of flash memory, and 96 KB of SRAM.

5.2 Hardware development

The framework for this project involved the development of a custom board so-called Arduino shield that can be connected on top of the Arduino Due and provide access to the external hardware to communicate via IO-Link. The main goal of the shield is to have access to the IO-Link Master transceiver, the non-volatile memory for DS and four IO-Link device ports with M12-4A female connectors. Additionally, we added an ethernet MAC-PHY interface ENC424J600 from Microchip for the future addition of ethernet connectivity.

IO-Link requires a 24V power supply for the transceiver and the IO-Link devices. As the Arduino Due only supports 12 volts maximum for voltage input, a 24V voltage regulator was included in the shield. A block diagram of the components that were integrated into this IO-Link Master shield is seen in Figure 4. The blocks colored with blue are the main hardware components of the shield, while the greeen blocks represent the communication interfaces that the shield provides for the Arduino Due, and the yellow blocks are the external communication interfaces. A preliminary view of the complete PCB assembly is seen in Figure 5. In this view we can see the addition of status LEDs, the positioning of the IO-Link Device connectors on the right side and the header for stacking the Arduino Due on the bottom side.

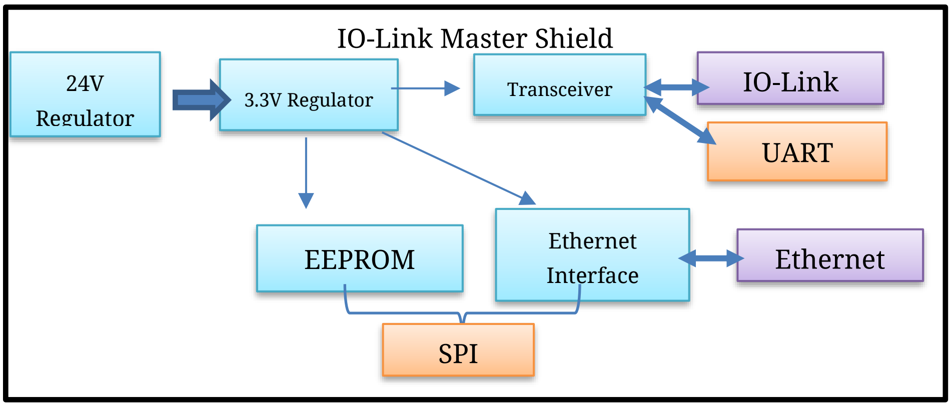


Figure 4 Block diagram view of IO-Link Master shield

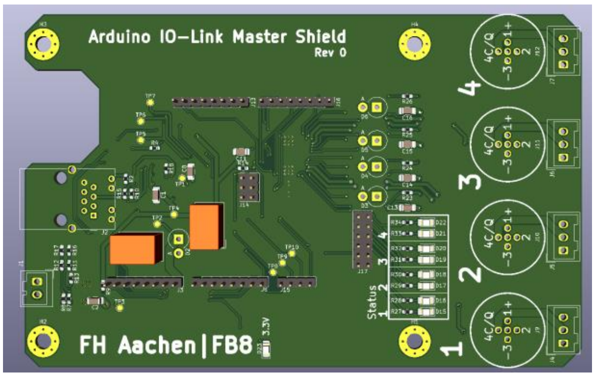


Figure 5D View of PCB Assembly

5.3 Software Design

The objective of the framework kept in mind compatibility with the Arduino Software Development Kit (SDK) and flexibility to extend it to other Arduino boards in the future. In this section the software design, its implementation and its focus for the Arduino platform are discussed. The main premise for the software design was that the framework is intended for rapid-prototyping applications and easy customization. For this reason, we followed a layered style architecture due to its simplicity and ease of development. The software implementation can be divided into three parts: the communication stack, the Hardware Abstraction Layer (HAL), and the Application Programming Interface (API).

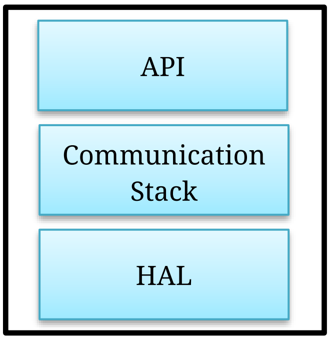


Figure 6 Overview of Software Architecture

The lowest layer in this design is the HAL. The purpose of the HAL is to access the hardware peripherals independent of the Arduino board that is used. The main components modelled for the HAL were the SPI, the UART, the EEPROM, the IO-Link Master transceiver and the timers. The HAL interacts directly with the communication stack, which requests access to the peripherals for the IO-Link communication.

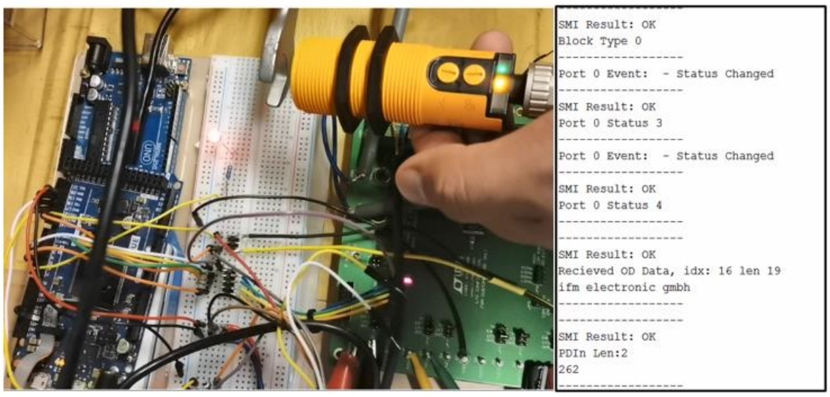
The second layer consists of the communication stack. In this layer the communication layers from the IO-Link specification v1.1.3 were implemented. The hardware access is done as mentioned through the HAL and any events from the stack or requests are interchanged with the API layer.

The last layer is the API. In this layer the user can send requests to the API which will redirect them to the communication stack. In addition, confirmation of the requests and notifications from the communication stack are processed and sent to the user. The API is based on the IO-Link SMI, which represents a standard way of accessing the services of the IO-Link Master and its IO-Link device ports.

6 Preliminary results

For the first iteration of this project, we did not include the integration of the Arduino shield due to the current chip shortage problematic. Instead, we only tested the software and used a LTC2874 development board for the IO-Link communication. The tests for this first prototype included verifying that an IO-Link device could go into operate mode and exchange process data, reading its vendor name, and notification of IO-Link events.

For these tests we used two IO-Link devices: the first one is a O5D150 distance sensor from ifm Electronic, and a KI6000 capacitive sensor from the same company. The verification process consisted of using the API to send requests to set an IO-Link port in IO-Link mode and read its process data, serial number and produce an IO-Link event. The results of these operations were logged and sent through the serial monitor port of the Arduino IDE (see Figure 7)



```
SMI Result: OK
Block Type 0
------------------
Port 0 Event:  - Status Changed
------------------
SMI Result: OK
Port 0 Status 3
------------------
Port 0 Event:  - Status Changed
------------------
SMI Result: OK
Port 0 Status 4
------------------
------------------
SMI Result: OK
Recieved OD Data, idx: 16 len 19
ifm electronic gmbh
------------------
------------------
SMI Result: OK
PDIn Len:2
262
------------------
```

Figure 7 Tests with an IO-Link Device

Besides testing these features we also measured the average latency of the processing time of the communication stack and the reception of process data. For the average processing time with one port active it was found that in average it took 40 microseconds the execution of the stack. The reception of process data through the API starting from the reception of the IO-Link device data through the UART took about 130 microseconds. The asynchronous read request from the API to the communication stack was in average 100 microseconds.

7 Conclusions

The preliminary results of this work allowed us to show the feasibility of a rapid-prototyping framework for applications with industrial sensors and actuators. In particular, we decided to use as a reference frame the Arduino platform for our development. The advantage of this platform is that its open-source and there is support for wide variety of components that can be added to the Arduino framework such as data logging, wireless connectivity, and embedded sensors. In comparison with a traditional approach, were a PLC is required to get information from a sensor, the Arduino platform is neutral and compatible with development boards. In addition, configuring a project for a PLC, setting up the communication and extracting the information for a simple application is more cumbersome than just using an Arduino and loading a library. The main advantage here is that this approach can be deployed faster to test a prototype and integrate it with other components due to the flexibility that Arduino offers and its constant development.

It has to be noted that the framework is not optimized for performance and rather uses a more simplistic design for ease of use and portability with other Arduino development boards. The main reason for this is that the purpose of the work is for rapid prototyping and not for deployment as a commercial product where optimization and performance matter, such as by using an event-driven architecture and a real time operating system. The current implementation is not finished and will be further improved to test that it works with more than one IO-Link device and to fix any bugs in the software. In addition, tests with the IO-Link Arduino shield will be made.

8 References

[1] V. Chavez and J. Wollert, “Arduino based Framework for Rapid Application Development of a Generic IO-Link interface,” pp. 21–33, 2020.

[2] V. Chavez, P. Saurabh, and J. Wollert, “10BASE-T1L based connection for field devices in Cyber-

Physical Systems: A Proof of Concept," in *Embeddd World Conference*, 2021.
[3] T. Gammeter, J. Lehmann, P. Frei, and M. Gafner, "openiolink/io-link-master-shield-hat-sw Software for the IO-Link Master Shield/Hat for Arduino and Raspberry Pi." [Online]. Available: https://github.com/openiolink/io-link-master-shield-hat-sw. [Accessed: 01-Oct-2021].
[4] International Electrotechnical Commission, "IEC 61131-9:2013." [Online]. Available: https://webstore.iec.ch/publication/4558. [Accessed: 01-Jan-2021].
[5] IO-Link Community, "IO-Link System Description," 2018.
[6] "Arduino." [Online]. Available: https://www.arduino.cc/. [Accessed: 01-Oct-2021].
[7] IO-Link Community, "IO-Link Interface and System v1.1.3," 2019.
[8] Analog Devices, "LTC2874 Datasheeet and Product Info." [Online]. Available: https://www.analog.com/en/products/ltc2874.html. [Accessed: 01-Oct-2021].