Research Project and Seminar

Information and Communication Systems

Development of an embedded communication hub for sensor data acquisition in a robotic system

by

Juan Carlos Reyes Andrade

September 2020

Supervised by

MSc. Jannick Brockmann
Head of Electronics at Han's Robot Germany

First Examiner | Prof. Dr.-Ing. Bernd-Christian Renner
Research Group smartPORT
Hamburg University of Technology

Research Group smartPORT
Hamburg University of Technology
TUHH

Acknowledgment

This is the place to thank all the people involved with your thesis / project. Examples would be your family, friends, and of course your supervisor. The acknowledgement will not have any influence on your grade; however, we think it is good style to have an acknowledgement in your thesis.

Abstract

The abstract of your thesis goes here. There may be formal requirements on it that can be found in the corresponding examination guidelines (Prüfungsordnung). If there are none, ask your supervisor. As a rule of thumb, the abstract should be concise and focused. It is not a shortened introduction to your work. We also suggest that—if an abstract is not required—only write one if it is really well done.

Table of Contents

Chapter 1

Introduction

This document describes the different stages through the development of an embedded communication hub for sensor data acquisition in a robotic system, within dis document this prototype will be refered as *Axis Communication Hub* or ACB. During this chapter a brief introduction to the Real-Time Ethernet (RTE) industrial networks is presented, as well as a summary of the standards involved with comments about how they are related to each other. Moreover, the usage of these RTE industrial protocols in embedded applications and its relation to the Industiral Internet of Things (IIoT) necessities is briefly introduced. Finally in this chapter, a brief comparison of the openess of these protocols and how this is related to the development of devices is presented. The second chapter shows a summary of the state of the art regarding the possibilities for developing open source projects according to the degree of openess of an RTE communication protocol. This has focus on EtherCAT devices as it is within the scope of this Research Project and shows advantages that will be detailed as the reader reads through this document. Afterwards, the third chapter deals with the goal of the Research Project and its proposed solution. a summary of technical specifications, the hardware available, software structure and an overall prioritization of the goals is also included. Later on, during the fourth chapter, the main points related to the implementation is presented. The overall results are discussed in chapter five, where the reader can find comments about the implementation and test challenges. As part of the conclusions chapter, a proposed list of points for further development is discussed. Finally, extra information focused on the technical details of the implementation can be found within the appendixes.

1.1 The need of RT within industrial environments

During the last years an increase in the usage of the Ethernet-based fieldbuses within industry has been recorded. This, with no surprises, shows the expected adaptation of the industrial automation to the IT infrastucture, which is fundamental for the *Industrie 4.0* paradigm and

its consequent huge amount of data to be monitored, analyzed and controlled, dealing at the same time with differnt time constraints and interconnectivity among the different layers of an industrial system and their devices. Having in mind the current *automation pyramid* with direct access from the top to the bottom[SKJ18], see figure 1.1, it is understandable that several technologies providing this access have been meeting each other comming either from the top or the down levels, at the point that they offer similar features regarding data access and security. Each of them with their own development history, alliances and, therefore, standards. Comming from top-level-related frameworks there is, e.g., the OPC UA project; whereas names like Profinet, DeviceNET, EtherCAT, Powerlink, etc, come from the fieldbus side -lowest level-. All of them have developed in an individual way as response of market, however meeting in the late decade through the necesity for unified standards to improve interoperability between the incredible number of projects. This happens at a time were information, technical as well, and development tools have become even more available and open to the end-user. Leading now then to a situation where the private initiatives are not any longer the full owners of the technology development.

Another line of work, closely related to interoperability, is the Real Time (RT) applications in their both versions with *hard* and *soft* requirements. Nowadays, there is an increasing number of applications in robotics that demand control loops and device chains that demand hard real time performance. Although, this requirements are more typical at the device level, such as, robots, cncs, etc. They all now face the IIoT requirements; hence, their networks should meet as well certain degree of RT capability. Moreover, synchronization of time sensitive systems within manufacturing lines, for instance, has been addressed for years by the RTE protocols and now these sort of features are increasingly been demanded as well at upper levels IT levels.

The current automation industry has a record of many competitors and closetechnologies, as natural consequence for specific processes requirements (depending on the industry), but also as a response of market strategies. Nevertheless, the serach for standardization can be tracked

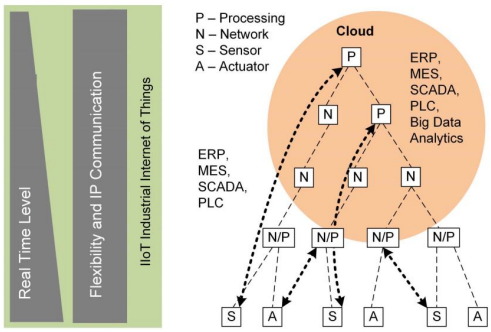


Figure 1.1: *Industrie 4.0* architechture. Industrial Internet of Things.[SKJ18]

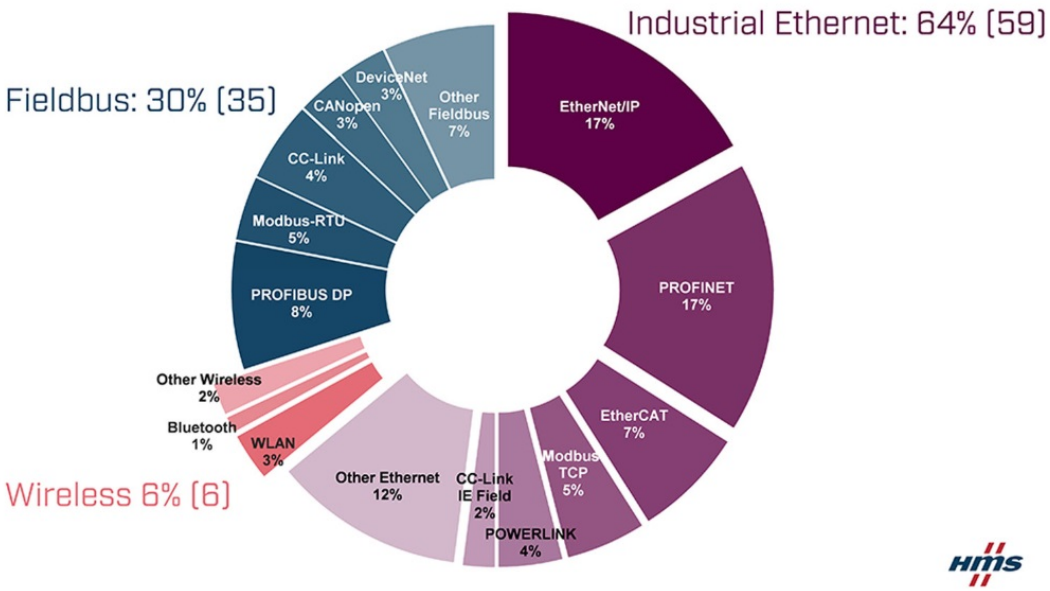


Figure 1.2: Industrial network market shares 2020 according to HMS Networks.[Car20]

back to the 80s, as the fieldbuses were standardized by the International Electrotechnical Commission (IEC). Continuing after the Ethernet took its place within the industry. As an important note, during the last two years, according to the HMS Industrial Networks' annual study, the total market shares of new industrial nodes in factory automation increased for the Industrial Ethernet from 52% to 64%; as the commonly called fieldbuses decreased in the same period from 42% to only 30%; finally the industrial wireless remained around the 6%, see figure 1.2. [Car20]

It is yet worthy to mention that the name *Industrial Ethernet* is used only as a generalization for the group of protocols that historically developed on IEEE's Ethernet specification; even though, they all are almost no further compatible with each other -as they have modified Media Access Control (MAC) Layers-. More details about this differences will be addressed in the following chapters.

As history shows, vendor protected technology have its limit when there are plenty of possibilities for automation technologies, even if they are in ongoing development. For instance, as happened during the lifetime of the Open Platform Communications OPC -predecessor of OPC UA-, that was started only upon *Microsoft Windows* and as the time went by, the emerging needs made it change to use open standards and a multiplatform approach.

To introduce the reader to a common ground regarding standardization, the following chapter will present a brief summary of the standars that are of interest for anyone who wants to start developing using industrial interfaces.

1.2 Industrial standards and the TSN initiave

This section is intended to provide the starting developer a rough but useful reference of the standards related to industrial communication networks. If the reader has already good knowledge of these standards, this section can be easily skipped and continue with the next chapter. First of all, due to the historical and technological process of innovation within the information and communication systems, several parties have been related and, at some extention have merged results, bringing out an interconnected set of norms that thrive continuosly onto a global standardization.

The following list is intended to be a quick reference to the current standards for Ethernet, legacy and current fieldbuses, Time-Sensitive Networks in their american and international iniciatives. This way, the reader has a roadmap to be taken into account for deeper research within the industrial applications. Furthermore, information related to the similar standardization processes between the IEC, ISO and IEEE, and their unavoidable cooperation, can be read in [JDH02].

ISO/IEC/IEEE 8802-3:2015 Revision of the Ethernet standard for half and full-duplex communication up to $100Mbps$. Originally published by american IEEE 802-3 in 1985 and accepted internationaly in 1989. The last revision 8802-3-2017/Amd 10-2019 includes MAC controls for 200 and 400 Gbps.[ISO17] After 2019 The name Ethernet is not longer used, instead CSMA/CD or a reference to the corresponding ISO standard 8802.3 is the formal name.

IEC 61158:1999-2000 First international fieldbus standard published in 1999, where 8 *Types* of fieldbuses were introduced addressing the Physical Layer (PhL), the services and protocols of Data Link Layer (DLL) and Application Layer (AL). Some included brand names were the following: H1/HSE/H2, ControlNet, EtherNET/IP, Profibus, Profinet, Interbus. This standard has an interesting story concluding with the signing of the *Memorandum of Understanding* by the main contenders to put end to the fiedlbus war.[Fel02] Its most updated version in 2019 counts with 26 Types of protocols and grouped them as fieldbus Communication Profile Families (CPF).

IEC 61784-Part 2:2008 It is an extension for the CPs capable of RT that are based on the IEEE 8802-3 standard (commonly named Ethernet). Commercial names included are the next: EtherCAT, Profinet, Ethernet/IP, Ethernet Powerlink, and Modbus TCP.[VZS19] The SERCOS CPF is highlighted, since its third version is, altogether with the EtherCAT profile, the fastest one in the list; providing as well a more efficient use of the available bandwidth with an open source resources. IT shows even advantages over CAN devices due to its original design intended for hard RT motion control.[Sta00][Sta20] This is a

very interesting Hard Real Time capable protocol that might need further development which is not included in the scope of this Research Project.

IEEE 802.1A/B/C/D/Q Time-Sensitive Networking standards is an initiative to improve the IEEE 802-3 in order to meet the industrial real time requirements, which story can be tracked back to 2005, as the IEEE 802-3 group was merged with the IEEE 802.1 Audio Video Bridging Task Group and started to work for industrial environments. This a response to the vast alternatives of the RTE CPs. About 60 individual IEEE standards oriented to improve the ISO/OSI layer 2, including 13 focused on its security, are within the scope of the TSN project. [VZS19] The mentioned project covers the lower layers of the communication system, whereas the upper ones, representation and transfer of data, is addressed by OPC UA. Moreover, it is important to mention that this is an on-going project and still around 40% of its standars are in draft or preparation phase.[IEE20]

IEC/IEEE 60802 TSN Profile for Industrial Automation is the stand alone TSN base standard that will include the common advancements from IEC SC65C/WG18 and IEEE 802 work groups mentioned in the previous item.[II20]This is an on-going project started around 2017, still being in a draft phase. Since this will be the international standard, it would be the equivalent to the effort once given during the creation of the IEC 61158 for the legacy fieldbuses.

IEC 62541:2016-2020 Set of IEC standards for OPC UA. Individually, the IEC 62541-14:2020 defines the OPC Unified Architecture (OPC UA) PubSub communication model. It defines an OPC UA publish/subscribe pattern which complements the client server pattern defined by the Services in IEC 62541-4. IEC TR 62541-1 gives an overview of the two models and their distinct uses.[IEC20] Quoting: *OPC UA is a client-server communication protocol for industrial use cases without hard realtime requirements. The new PubSub extension of OPC UA adds the possibility of many-to-many communication based on the Publish / Subscribe paradigm. In conjunction with the upcoming Time-Sensitive Networking (TSN) extensions of Ethernet, OPC UA PubSub aims to also cover time-deterministic connectivity.*[PERK18]

Chapter 2

State of the art

This chapter introduces some current applications mainly foucsed on robotics, since this area is closely related to the environment with which the ACB will be interacting. Robotics sees various advantages from the RT capable communication protocols, when it comes to integrate motion controllers and any other industrial peripheral. Afterwards, an overview about industrial development framworks is given, yet focused not on the RT interfaces, but the specific software. The latter is of great importance, for the Real Time Operative Systems (RTOS) are a corner stone for embedded systems that need to provide a deterministic service within their environment. Finally, an overview of the EtherCAT protocol is included.

2.1 Current applications

As rapidly mentioned in 1.2, the SERCOS motion control interface has been standardized within the CPFs of IEC 61784-Part 2. Furthermore, it has been even integrated to EtherCAT as a compatible CP. This service is available within the DLL and AL and is called Sercos over EtherCAT (SoE), which provides access to motion controllers under the SERCOS specifications and, consequently, offers interoperability within its own RT features and the latter's hard RT capabilities. An example of this compatibility is presented in [XJY11], it shows that a jitter of 30 microseconds is feasible in a control loop while the Master uses the SoE service.

Another interesting application has been the characterization of an EtherCAT Master within a Real Time control loop for Servo Motors, which run CAN devices over EtherCAT (CoE service). The implementation of the Master device ran on different open source RT OS based on Linux, namely, Xenomai and Linux with the *RT_PREEMPT*. It was concluded that both of the apporaches were capable to achieve update periods of $1ms$, and Jitter around 1.15 microseconds. Moreover, Xenomai could average execution times around 100 microseconds[DC17].It is worth to mention that the Master device executed on top of both kernels, the IgH EtherCAT Master, the latter mentioned in 2.3.

The characterization and optimization of performance of different RTE profiles within TSN is the current topic as the TSN standards and the RTE commisions are still working together, as briefly mentioned in 1.2. In [MGSR20] are presented simulations of TSN topologies with EtherCAT and SERCOS data frames, where the QoS is addressed and evaluated through the usage of SDN (Software-defined Network) for TSN switches. The approach of this project is to test different scheduling features given fixed Cycle Times of the Data frames, which were proposed to be similar to the current applications in both technologies. In this manner the importance of an unifed network that supports diferent protocols is highlighted, but further research in this topic, including tests with other RTE Data Frames is still open.

Besides robotics, a recent industrial application concerning distributed control and monitoring based on EtherCAT open protocol can be seen in this article [Nor20].

Addressing now the usage of open source tools, OS and RTE Protocols, for development of complex robotic systems, in [MHF+20] is presented a *Motion Planning for Quadrupedal Locomotion*. This is roughly composed, besides the hydraulical actuators, mechanics and other peripherals, of two PCs on board with RT capabilities and shared memory. RT Linux (Xenomai) runs on both of them and take care of different levels of the control threads at two different rates depending on the tasks, namely 1 kHz and 250 Hz. The former rate is used for communicating with the motor controllers over EtherCAT interfaces.

Currently *Han's Robot Germany GmbH* focuses on enhancing robots' cognitive abilities by developing in the fields of environment perception, drive technologies, control theory, material science, mechanical design and artificial intelligence. Interfaces within the robotic system rely on various industrial protocols to make its interoperability one of the key features. For instance, current motor drivers are linked over internal EtherCAT chain to the main controller[Han20].

The above mentioned applications are just a tiny number of examples that shows the importance of an already standardized open industrial communication protocol, within a broad set of fields that cannot be completely covered in the scope of this document. Nevertheless, it paves the road to understand why generating the know-how to any of the mentioned technologies, represents a high-impact resource for any research or development group, regardless of its commercial or academic purposes.

2.2 An overview about the RT capable SW in robotics

As the previous section was being written, several resources and examples showed the current usage of RT-oriented open source software an its community. Since this Research Project has a goal of introducing the reader a roadmap for RTE communication interfaces and its applications, it was worth adding this section to summarize the RT Software for development in robotics.

As mentioned in [DC17] and [MHF+20],the usage of middlewares within the field of robotics is growing and it relies on robot software that exists between the application and a RT capable OS. In [JYJP20] a list of requirements is provided to address these middlewares and how to consider them Real Time Robot Software Platforms. The mentioned list can be interesting to start getting familiar with the capabilities and features of the so-called Robot Software. The list of requirements is as follows:

- R1: Support data exchange
- R2: Real time support (strict period execution and sporadic performance support)
- R3: Supports thread and process types for user defined programs
- R4: Easy configuration of applications (robot control SW, PLC SW, vision inspection SW, non-real-time SW, etc.)
- R5: Support multiple periods.
- R6: Threads or processes running in the same period are classified by priority.
- R7: Check and handle the event through the event handler.

Common names for different projects aiming to create this development frameworks are the following: Common Object Request Broker Architecture (CORBA), Real-Time CORBA (RT-CORBA), Data Distribution Service (DDS), OPC Unified Architecture (OPC-UA), Robot Operating System (ROS), Open Platform for Robotic Services (OPRoS) OpenRTM (open robotics technology middleware), open robot control software (OROCOS), XbotCore, and real-time middleware for industrial automation devices (RTMIA). Further comments and comparison between their features can be seen in the previously referenced paper. As to what concerns to this document, only some of them will be roughly commented as they ended up being somehow related to the RTE profiles[MLH+17].

OPC UA As frequently mentioned before, this is an open standard for data sharing among nodes within industrial networks and has been considered in some projects related to robotics. Neverthless, it is important to highlight that this is no considered a full middleware, since it only provides a protocol to control the exchange of data between nodes, a good degree of reliability and security. However, it does not provide RT capabilities to the system only compatibility. Hence, it needs an operative system and the consequent lower layers capable of RT scheduling and communication, concerning the latter the TSN set for protocols is an example.

ROS/OROCOS/OpenRTM These are projects that aim to create a suitable middleware for robots by implementing Xenomai or Linux operating systems. ROS prioritizes the final

user, avoiding in the way some fine-grained features due to its difficuty, therefore having sometimes issues to meet the hardt RT requirements. Orocos has further improved its comaptibility, similarly to OpenRTM.

CODESYS and TwinCAT To fully meet compatibility with the industry, the so-called PLC Software has been also used in open robotics. These applications roughly need to run both, the robot functional blocks and the robot tasks. For further details on it the following references can be reviewed [MLH+17] and [MLT18].

xbotcore This is an attempt to provide of a highly compatible open middleware for industrial robotics, it runs over the Xenomai and uses a SOEM stack to interface with any compatible industrial device, recall 2.3. Applications has been already mentioned in 2.1, which have reached control loops down to 1khz for 33 axes [MLH+17].

RTMIA RTMIA middleware + Linux or Xenomai but used open PLC running parallely[?] « This needs further reading [?]

The previous information was presented only to draw an idea for the non-familiar reader about the applications and, since this topic is in on going development and, furthermore, many other plataforms are addressing similar challanges are arising; the reader is invited to go deeper into these topics, for instance, by reviewing this resource [JYJP20].

2.3 Approaching openness within the RT protocols

Among the industrial standards mentioned in 1.2, there are some related initiatives to include a certain degree of open source software to improve the development of applications. The following is a brief list of a few interesting references to them. However, as expected, most of the software stacks for industrial communication systems are commercial and provided by third-party companies.

OSADL Open Source Automation Development Lab eG (OSADL):It is a German group that intends to lead the development of open source development for industrial automation. Closely related in the developing of OPC UA and other Linux features for industrial applications.

open62541 Within the official scope of OPC UA, there is this Certified SDK project that is within its second phase, at which it is expected from the research and industrial community to develop applications to test its performance. Moreover, as the TSN specification is of huge importance, a set of enhancements for the *open62541* project were developed by *Fraunhofer IOSB* and a serie of patches for the Linux kernel has

been released to make it a RT compatible.[OSA19] The OPC UA is developed under GPL 2.0 license and due to its current phase implies a further adaptation for the physical node, e.g., ARM arquitechtures to make them compatible with the mentioned patches.

SOEM/SOES RT Labs Industrial development group focused on Software Stacks for industrial protocols. Among their commercial communication stacks there are software stacks under GPL for EtherCAT Master and Slave devices SOEM/SOES*. [RT-20]

Sercos Stacks Sercos III technology is able to be operated in a common TSN-based network [Ser18], since its development group is working closing together with the TSN group. They also made available open source software dedicated for development of master and slave devices, namely: Common Sercos Master API (CoSeMa), Sercos Internet Protocol Services (IPSS), and The Sercos III SoftMaster, the later even allows the host to use any standard Ethernet controller.[Ser20] It is important to mention that there are testing tools to certificate those devices and achive a Safety Integrity Level 3 (SIL-3).

EtherNet/IP Stacks There are several commercial stacks that comply with the Open DeviceNet Vendors Association's (ODVA) EtherNet/IP specification. *OpEner* is an open source alternative which targets PCs with a POSIX operating system and a network interface. Examples are provided for integration only in Linux and Windows in [OpE20]; nevertheless, a variation for embedded systems has been presented for an STM32 microcontroller. In the mentioned project, more tools had to be adapted, for instance, STM32F4x7 Ethernet Driver v1.1.0, lwIP v1.4.1 (TCP/IP Stack), MicroHTTP v5.1.0.1, a patched version of OpENer v1.2., among others.[emb20]

IgH EtherCAT Master This is a bundle of libraries to give a Linux host (LinuxCNC for example) EtherCAT Master features, it is developed under the GPLv2 license. An interesting example of this open source resource within an Airbus Test Rig can be reviewed in the following reference [HWHP].

As the scope of this Research Project is only focused on the indsutrial communication profiles capable of RT and, so far has been clear how the EtherCAT is a reliable one, yet open and significantly considered in the industry -recall -2.1-; it makes sense to present an introduction to the protocol itself. This way it is easier to go sensibly to the implementation of what is one of the basic chain-elemnts in what could become a very complex application: an EtherCAT Slave device with open source elements.

2.3.1 EherCAT protocol

As mentioned in the introduction, EtherCAT is an industrial Communication Profile developed by Beckhoff that is standardized in the IEC 61588 under the RTE CPF. The development

within this company is oriented to the use of open standards to increase its impact within the indsutry, but not only reduced to it but the overall field of smart cities[?], in a certain degree this apporach eliminates the need for many expensive "black boxes"[Bec13]. This implies that the interoperability of devices is almost guaranteed, at least from the specification perspective, not only for private development centers but also for any other developer that follow the standards; if the standards are of public access, then this is a mean of empowerment of any group that might be willing to create its own industrial-compatible technologies.

The OSADL emphatized in 2008 [Bor08], for example, a vision for leading the integration of opensource in the industry by using the Linux Kernel as a certified Industrial RT (IRT) operative system for industrial embedded applications. Back in that day, Beckhoff was involved in that discussion representing the contrary model. Nonetheless, in the last months the same company has apparently retaken the opensource iniciative by the introduction of the FreeBSD compatible version of the TwinCAT Runtime[Bec20b].

In comparison with other RTE profiles, EtherCAT has shown a higher performance, more flexible topology and lower costs than other ethernet fieldbus technologies. This protocol applies a master-slave mode, in which the master device uses standerd 100BASE-TX ethernet adapter and the ESC (EtherCAT Slave Controler), that implements a EtherCAT IP (intellectual property) core within an ASIC or a FPGAto process the frames. As the working cycle starts, the EtherCAT master publishes a frame encapsulated a standardized 8802.3 frame. When the it reaches an ESC, it analyses the address and location on the frame, decides which parts of it are useful sections and then reads or writes data on it. As the read-write operation finishes, the Working Counter (WKC) at the end of the frame is added by one, this way the data on the frame has been processed. This cycle repeats for each ESC within the topology.EtherCAT supports almost all kinds of topology structure, such as ring, line, star and tree. The transmission speed of EtherCAT is fixed to 100 Mbit/s with full duplex communication. The EtherCAT network is able to connect maximally 65535 devices via switch and media converter. The EtherCAT system can update 1000 I/Os in just 30 microseconds or exchange 1486 byte contents in 300 microsecond. [Bec17][XJY11]

Important to highlight is that other CPs are also integrated as services inside the protocol, as previously mentioned for the SERCOS specification in 2.1. Other examples of these integrated CPs are File over EtherCAT (FoE) or Ethernet over EtherCAT (EoE), which make possible to support a wide variety of devices and application layers in the same network[Bec20a]. A complete list of the communication profiles that are on hand through the protocol's mailboxing is given below, as to the overall layered integration can be seen in figure 2.1.

- CoE: CAN application protocol over EtherCAT
- SoE: Servo drive profile, according to IEC 61800-7-204 (SERCOS protocol)

- EoE: Ethernet over EtherCAT
- FoE: File Access over EtherCAT (HTTP,FTP,etc)
- AoE: Automation Device Protocol over EtherCAT (ADS over EtherCAT)

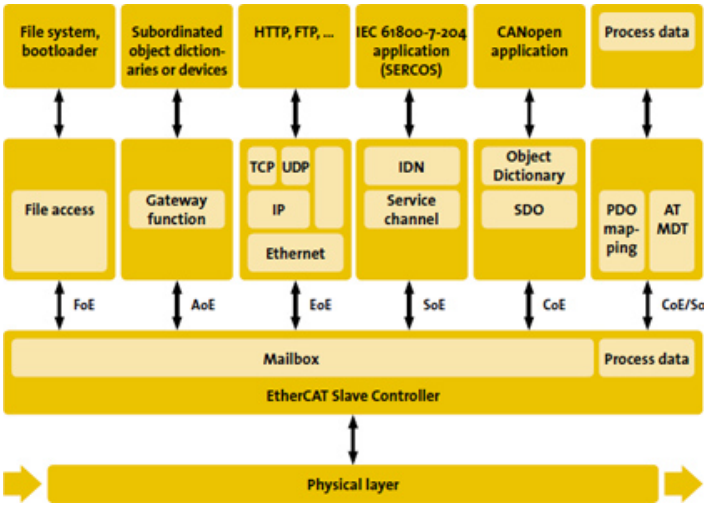


Figure 2.1: Different communication profiles can coexist in the same system.

An EtherCAT device with switchport properties usinge EoE would be the equivalent of the TSN compliant switches, since they would insert any non time-senstitive TCP/IP fragment into the EtherCAT traffic preventing in this way the real time properties from being affected. Furthermore, the architecture of the protocol itself and its early cooperation with the IEEE 802.1 group and the OPC Group ensure its continuos compatibility with the standardization of TSN, OPC UA and the IoT paradigm.[Bec20a]

A slave device must comply at least with CoE and the Mailbox, whereas the Master may comply with all the communication profiles. This of course needs to be suited to the requirements of the application and the degree of tlexibility that is to be achieved. Consequent certification process should adhere to Beckhoff's specifications.[Bec18]

Having presented this brief summary of the EtherCAT technology, the reader may continue to the following chapters. More detailed information of the protocol itself that was needed to understand the function of the SOES library can be read in the section 4.4.

Chapter 3

Solution proposal

This chapter is intended to give the reader a summary of the proposed solution and its structure, as well as the technical specifications that were given and taken into account. Any other constraint not mentioned here was adjusted or set while being within the design-test loop, as consequence of the prototype nature of the project.

3.1 Technical specifications

In table 3.1 the requirements for this prototype and their state after the implementation can be observed. This comparison helps as a good summary of the overall achievments that are further detailed in the next sections.

3.2 Available Hardware

In this part it is presented the base hardware that was available to develop the prototype. The microcontroller (MCU) was chosen due to its active community, resources and current on-develop projects. Other MCUs were considered since the overall characteristics are somewhat similar and generic -regarding peripherals like serial interfaces or dma, processing power and memory, for instance-. MCUs from Infineon [Inf18][Gro04] and Texas Instruments [Son20] were good possible candidates; however, the basic familiarity with the STM32CUBE IDE and the related ST technology was a crucial factor, since the learning curve is not negligible when it comes to develop any firmware at a fair level, even more when it deals with other to-learn* technologies, for instance, development with RTOS, modification of open libraries, EtherCAT protocol and the Network Controller chip.

Regarding the Network Controller, the LAN9252 belongs to a set of ASICs that are verified and certified by Beckhoff GmbH. For a further reference for other alternatives visit [Gmb20]. The LAN9252 integrates a so-called EtherCAT Slave Controller (ESC) and it represents a good alternative to the Beckhoff's original ASIC ET1100. This way, the basic hardware is there to

Feature	Requirement	Implementation	Remark
Upper layer interface	Ethernet/EtherCAT compatibility. Non-safety relevant. Services and synchronization: tbd	EtherCAT slave. Services: Mailboxing and CoE Synchronization: Free Run and SM	✓ FoE and SD synchronization possible in the medium term.
Display/signaling	LED stripes with serial interface: WS2812 2 Ch	2-4 Ch modifiable in SW Animation capable	✓ Chs can increase up to number of DMA-Timers (8*)
Temperature	Data interface for 1-Wire bus	1-Wire Master 15 sensor in bus	✓ 6 Sensors simultaneously tested
PCB	PCB Prototype Layout and size: tbd	Attachable PCB for LAN9252-EVB-SPI. Size: 55mm$x$38mm	✓ Second layout with both chips included posible in the short term.
Safety	n.a.	Non-safety relevant for this prototype	– SFoE could be researched in the long term.
Extra interface	SPI or I2C interface for current/IMU/black channel	Extra SPI considered in PCB and SW JTAG/SWD compatible interface	✓
Speed/position	Possible interface of BISS-C type	Not required for this prototype	–
Refresh data cycle	tbd	No hard RT deadlines. Deterministic refresh cycle of ~10ms by RTOS. Timeout faults handling.	⊕
Data structure	tbd	Functional and parametrization data structure as Object Dictionary. Standard ESI file.	✓
FW programming	n.a.	CMSIS - FreeRTOS for thread, event and time management.	⊕

Table 3.1: Technical specifications

STM32F446ZE	LAN9252
ARM®32-bit Cortex®-M4 + FPU + Chrom-ART™ Accelerator Up to 180MHz CPU 512 kB of Flash 128 KB of SRAM	EtherCAT slave controller with 3 FMMUs and 4 SyncManagers Distributed clock support 4KB of DPRAM
General-purpose DMA Up to 17 timers Up to 4 × I2 C interfaces Up to 4 USARTs/2 UARTs Up to 4 SPIs 2 × CAN (2.0B Active) USB 2.0 full-speed device/host/OTG	100Mbps Ethernet transceivers compliant with IEEE 802.3/802.3u (Fast Ethernet) 8/16-Bit Host Bus Interface, indexed register or multiplexed bus SPI/Quad SPI Digital I/O Mode Multifunction GPIOs
LQFP64, LQFP100 and LQFP144 packaging	Pb-free RoHS compliant 64-pin QFN or 64-pin TQFPEP packaging

Table 3.2: Summary of the characteristics of both STM32F446ZE and LAN9252 used in the prototype.

fulfill *Han's Robot Germany*'s proposal for developing industrial compatible devices that could enhance the prototyping process within the electronics department. Moreover, the mentioned ASIC has a wide compatible control interface that make it be suitable to any microcontroller with which the developer has experience. The table 3.2 lists main characteristics of the above mentioned hardware.

(a) NUCLEO-STM32F446ZE (b) LAN9252-EVB-SPI

Figure 3.1: Evaluation boards for prototyping.

3.3 PCB proposal

As it can be seen in the Fig. 3.1(b), the evaluation board counts with on-board male pins, this was taken as an advantage and the PCB to be designed consisted on a plugable PCB that would be mounted on top of it, increasing minimaly the volume already occupied by the evaluation board. This idea needed to be designed taking into account the minimum of components based

on the Nucleo-STM32F446ZE original design and the requirements of the LAN9252. This means that it had to provide, both 5V and 3.3 V power supply, physical ports for the prioritized communication capabilities, minimaly SPI, One-Wire JTAG and the LED ports according to the technical specifications 3.1.

3.4 Software structure

The final proposal for the structure of the embedded system can be seen in fig. 3.2. The functional blocks represented in grey or dashed lines are mainly components that are planned not to be modified at all or not in deep, because of either its complexity or its given reliability. This means its functionality is almost granted. Nevertheless, the progress relies on documentation that can be either good or poor, for instance, TwinCAT has good resources, whereas SOES does not. Another thing to keep in mind is the abreviation DSM (Device's State Machine) which is used in this document as substitute for State Machine, such that it is not confused with Synchronization Manager, defined as well by Beckhoff as SM.

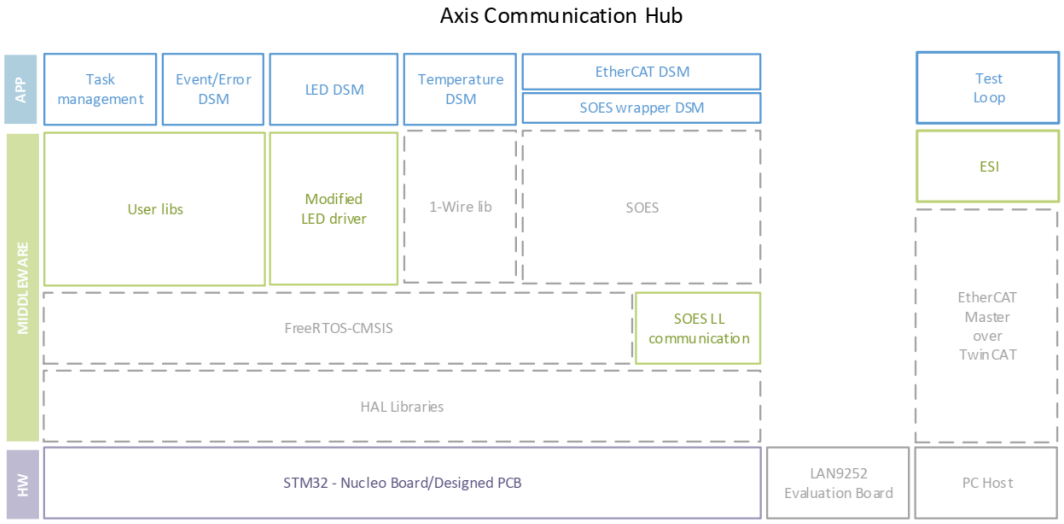


Figure 3.2: Layered strcuture of the proposed functional blocks.

Chapter 4

Implementation

The following chapter will documet the different functional modules that were implemented according to the proposal. Most of them interact with each other through the SMs, which has been added as well as a module for its understanding.

4.1 LED Control

The robot counts" with at least two LED rings that notify to the user the overall state of each access, this is considered only a visual aid which means that is not critical for the system. Currently there is a control for them using Arduino* devices. One of the goals of this prototype is to integrate the control of the LED rings within a single board; therefore, different apporaches were taken into account depending on the stages and experience in devloping on STM32 chips and its IDEs. The following stages were carried out:

4.1.1 Technical background

Here comes one picture of the LED ring and a very brief description of the LED and its interface, is it interface or protocol?-> interface Also a basic description of DMA in STM32

4.1.2 Development*

First control tests Learning the basics of the interface used to control one LED

Writing code for one LED control Using the peripherals of the MCU, simple routines were written to set different basic colors in RGB. The peripherals used were a PWM channel and one timer to keep control of the timing.

Exploration of libraries Once the basic communication was understood, it was clear that the usage of libraries would be more practical since the control demanded cofnigurable number of LEDs and updating all of them at once, even with the possibility of including

effects. From this exploration around STM32, the codes were mainly divided in two types: libraries for 8/16 bits processors that whose main task was controlling the LEDs, and a pair of libraries that used DMA Module within 32-bit processor. The latter was divided into two different approaches, namely rewritting an unique buffer for only one PWM channel and another highly focused on performance by divinding a circular buffer and interrupting the main processor only to update parts of it as the data was sent.

First library selection Due to the unexperience working with DMA modules and as the LED control was not of a high priority as the EtherCAT implementation, it was decided to run first one of the basic libraries to achieve a multi LED control and look for its adequating to the Axis Communication Board*.

Trial of adequation of the library into the SM task *here comes the explanation of why that library did not match with a multitask apporach*

Second library selection As the first library was able to control a set of 20 LEDs, some drawbacks were found in the approach, since the processor was focused only on polling Timers states. Event though, this first trial helped improving the overall understanding of the host MCU and the LEDs, it was not suitable to multi-tasking. Moreover, usage of DMA became clearer and it was decided to improve the approach by selecting a 32-bit processor based library. The Library* was then included and following the guidelines of the GNU* license it was modified to use two independant PWM generators -easily extendable* to four-. Furthermore, the main logic* of the library was divided such that its execution becomes non-sequential but triggered by events. The new interactions with other parts of the code can be seen in following subsections for SM*s.

4.2 Temperature acquisition

4.2.1 Technical background

Here comes the image of the sensor, datasheet and summary of translating the one-wire protocol to usart

4.2.2 Development

First readouts Within the project's schedule this was the first set of test, so through them the IDE stm32cubei was learnt, along with the general configuration of the MCU. First trials* included set up of internal clocks, interruptions, GPIO basic usage, basic configuration of freeRTOS in STM32 *here comes a reference to a future subsection where the structure and relation of freeRTOS and CMSIS is explained*; as well as the

introduction to the one-wire protocol. The first approach, similar to the one of the LED, was to understand the sensor and its protocol. Therefore, self written functions were tested based on peripherals, namely two timers, through which the timing of the data streams are controled as wel as the duration of ones and zeros. This approach worked as intended but it was knew from the beginning that does not match the multi-tasking and that further was needed. This implementation was able only to access in a generic way to the temperature conversion, and further functions were needed to access the sensor's ROM needed e.g. for identification.

Readouts as a task/FreeRTOS first tests Short after the working code was used to do the first tests with the RTOS, in this manner the code was translated as a Task (Thread as used by CMSIS) and some features like prioritization, task attributes, task handling and signals were tested with other generic functions, e.g. clocks and pwm generators or blinking LEDs. *here could come a reference to the captures of signals and PWM being generated and interrupted*. However, this implementation was not able to handle multiple one-wire devices due to its abscense of CRC** comparison.

Integration of library Finally, it was decided to adapt one open source library designed for STM32 processors. This is based on the principle that UART speeds @9600 and @11200 bps suits the One-Wire timing, such that the detection of One-Wire devices and communication process can be downloaded to hardware already included in many general-purposes processors using USART. The integration of this library is from design compatible with RTOS, namely with CMSIS-RTOS within the smt32cube development environment.

4.3 Extra SPI Service/Auxiliar

An extra SPI apart from the one that is used to interface the MCU host to the LAN9252 Evaluation Board was included in the design, mainly for comming applications and development that may include interface SPI sensors, for instance, IMU sensors. The DMA streams and pins had to be taken into account to avoid overlapping functions** with other functionalities. This section will also wrap the auxiliar functional block such that it can be addressed similarly to the proposed structure in Fig. 3.2.

4.4 EtherCAT Slave communication: SOES adaptation

This functional module is the one with highest priority, therefore most of the effort given was focused not only on the library itself but the protocol and the hardware commissioning. Hence,

this section was extended and divided as follows: first, more technical details are presented regarding the EtherCAT specification, such that the SOES features are better understood. As to the next subsection, some constraints for the prototype are commented and the available hardware is explained*. Once summarized, the main points of the implementeation are presented.

4.4.1 EtherCAT data consistency and constraints for design

This subsection describes both the features with which the *Axis Communication Board* has compatibility, and a summary of the mechanism that the protocol implements at the low level to work with the data exchange between Master and Slave. The constraints that were set were part of a live process that ran all along the learning process of the protocol itself. This is important to mention, since the understanding of the protocol leads to a sinful selection of the features that a device should have implemented. Therefore, the understing process was a natural consequence from the integration of the SOES library. It is presented however before, such that it is understandable what features became priorities and which others are now presented as proposals.

The reader may recall the set of Communication Profiles that are available within the EtherCAT fieldbus, see 2.3.1. From them, the Mailbox and CoE are the main features with which the Axis Communication Board works. Leaving aside for future integration the FoE and EoE, the forme would make possible update the device by sending a firmware binaries to the device's bootloade; whereas the latter would make the ACB* accessible for any IT tool based on TCP/IP.

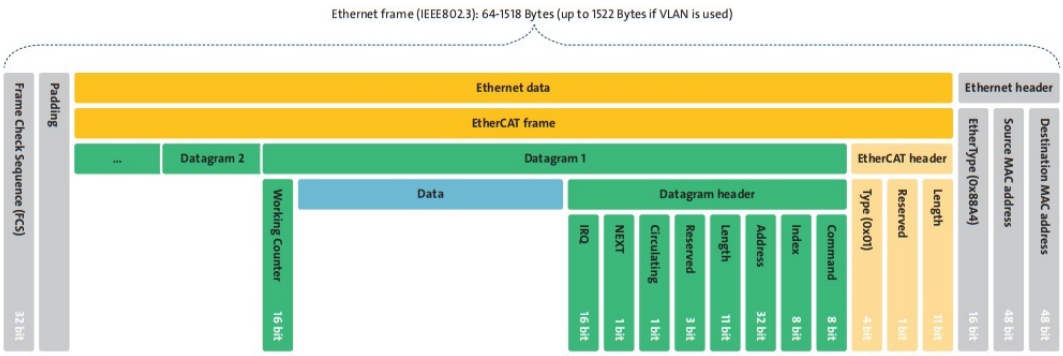


Figure 4.1: EtherCAT Datagram within the Ethernet Frame. Source: ETG.1000.4 - EtherCAT frame structure.

The data frame and the Synchronization Managers

Besides the challange of setting up the hardware and basic firmware for a correct data transmission between ESC and the host MCU; the description of the EtherCAT Slave device is a task that demands, at least, a basic understanding of the data frame exchange and how the protocol demands its synchronization. From here on, the following topics are going to be summarized: Synchronization modes and managers. Whenever there are Real Time constraints, and the device takes part of a control loop, synchronization modes are needed to be set correctly between the Master and any Device present. For this task the Distributed Clocs (DC) are need to be synchronzied. [?][?]

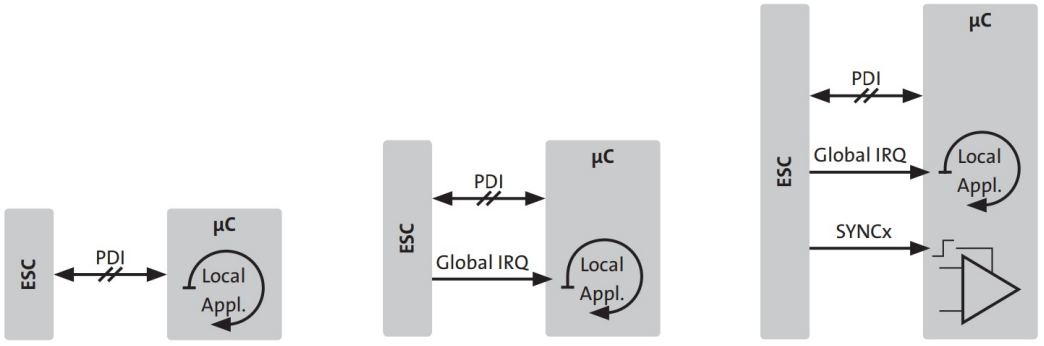
There are three synchronization modes:

Free Run Application is triggered by local clock and runs independently from EtherCAT cycle.

SM-Synchronous Application is synchronized whenever there are process data being written to the Synchronization Manager 2 (SM2). Moreover, any event generated by the Master is mapped onto an internal register or physically triggering an IRQ Pin of the ESC.

DM-Synchronous Within this synchronization mode the frame jitter can be even reduce down to *ns* and use two different synchronization units within the ESC, namely the SM2 and SYNC/LATCH UNIT.

The scope of this prototype covers the Freerun and SM-Synchronous mode, as they are the basic ones for communication Master-Slave. A graphical representation of them is shown in figure 4.2.



(a) Free Run (b) SM-Synchronization (c) DC-Synchronization modes

Figure 4.2: Synchronization modes defined in ETG.1000. Source [?]

SMXs (Synchronization Managers 1,2,3 ...) coordinate access to the ESC memory from both sides, EtherCAT and Host MCU (PDI). In case of process data communication it ensures

that process data can always be written to the memory by EtherCAT and can always be read by PDI side and vice versa (3-buffer mode). SyncManager 2/3 length is equal to the Data Object lengths defined for receive and transmit data chunks respectively. [?] The mapping of the process data objects within the Ethernet Frame can be seen in figure 4.1 and 4.3. The correct setup of the SMXs ensure the consistency of the data and needs to be linked correctly depending on the specifactions of each type of ESC, and SW Stack that are being used, this information is also linked to the CoE Object Dictionary (OD) and EtherCAT Slave Information (ESI) file.

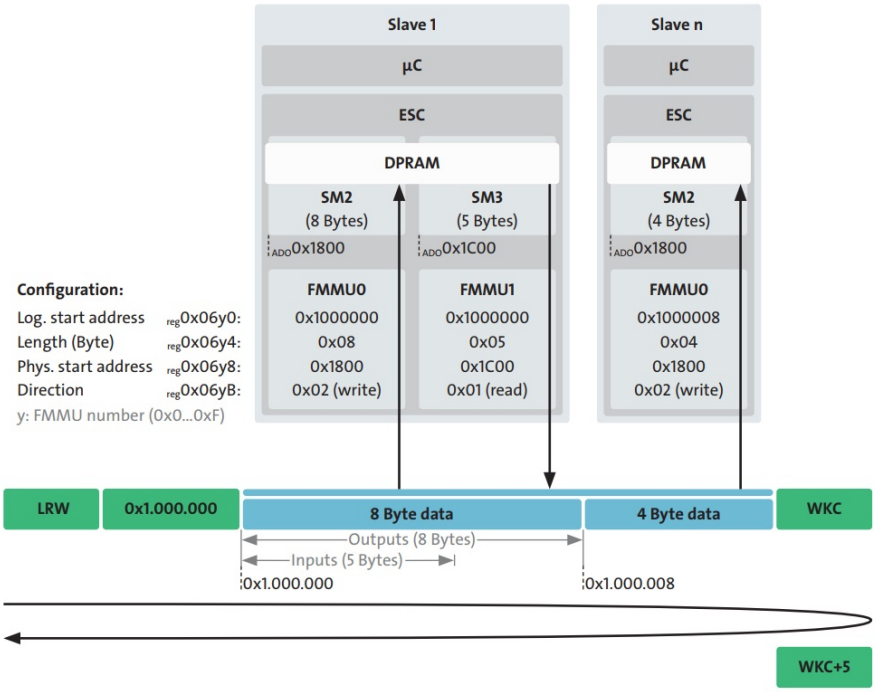


Figure 4.3: Depending on the different states of the Slave, there will be different data frames being exchanged with the Master. The above one corresponds to the Proces Data Object which is updated continuosly by the SM2/3 during Operation State. Referece: ETG.1000.6-SDO

4.4.2 SOES

As briefly commented in section 2.3, the types of licenses allow open development and integration of software. SOES software stack was written in C and published based upon the GPLv2, which is a Copyleft License. However, the tools developed by the Open EtherCAT Society which support the design, implementation and certification of EtherCAT slaves using the mentioned stack are comercial ones. A significant part of the challenge covered by this Project Research was to achieve the EtherCAT Slave functionality in the prototype without

Features	Requirements
EtherCAT State Machine	Build up the SII-EEPROM Data-Layout
Mailbox Interfaces	Create the ESI-file
CoE	Port libraries to the STM32 using HAL drivers
FoE + bootstrap template	Use FreeRTOS for scheduling (Hardware Requirements $RAM > 64KB$)

Table 4.1: Features of SOES library and the overall requirements to make it work.

those tools, as the protocols are open. In table 4.1 can be seen the main features abstracted and available in the stack, as well as the overall tasks to carry out for a device to work properly.

4.4.3 EtherCAT Slave Controller (ESC): LAN9252

As part of the available hardware introduced in 3, the LAN9252-EVB-SPI is an evaluation kit for the ASIC LAN9252 manufactured by Microchip. This IC is an EtherCAT Slave Controller with 4K bytes of Dual Port memory (DPRAM) and 3 Fieldbus Memory Management Units (FMMUs). Each FMMU performs the task of mapping logical addresses to physical addresses. The EtherCAT slave controller also includes 4 SyncManagers to allow the exchange of data between the EtherCAT master and the local application.[?]As briefly summarized in 4.4.1, each SMX direction and mode of operation is configured by the EtherCAT master. Two modes of operation are available: buffered mode or mailbox mode. In the buffered mode, both the local microcontroller and EtherCAT master can write to the device concurrently. The buffer within the LAN9252 will always contain the latest data. If newer data arrives before the old data can be read out, the old data will be dropped. In mailbox mode, access to the buffer by the local microcontroller and the EtherCAT master is performed using handshakes, guaranteeing that no data will be dropped. The overall structure of the ASIC can be seen in 4.4.

4.4.4 Development

Once explained the general information regarding the Communication Profile, the library and the hardware, the following lines will enlist and expose some of the most relevant information during the integration.

Porting the low-level functions of the library THIS

First tests with most basic read funcions over spi THIS

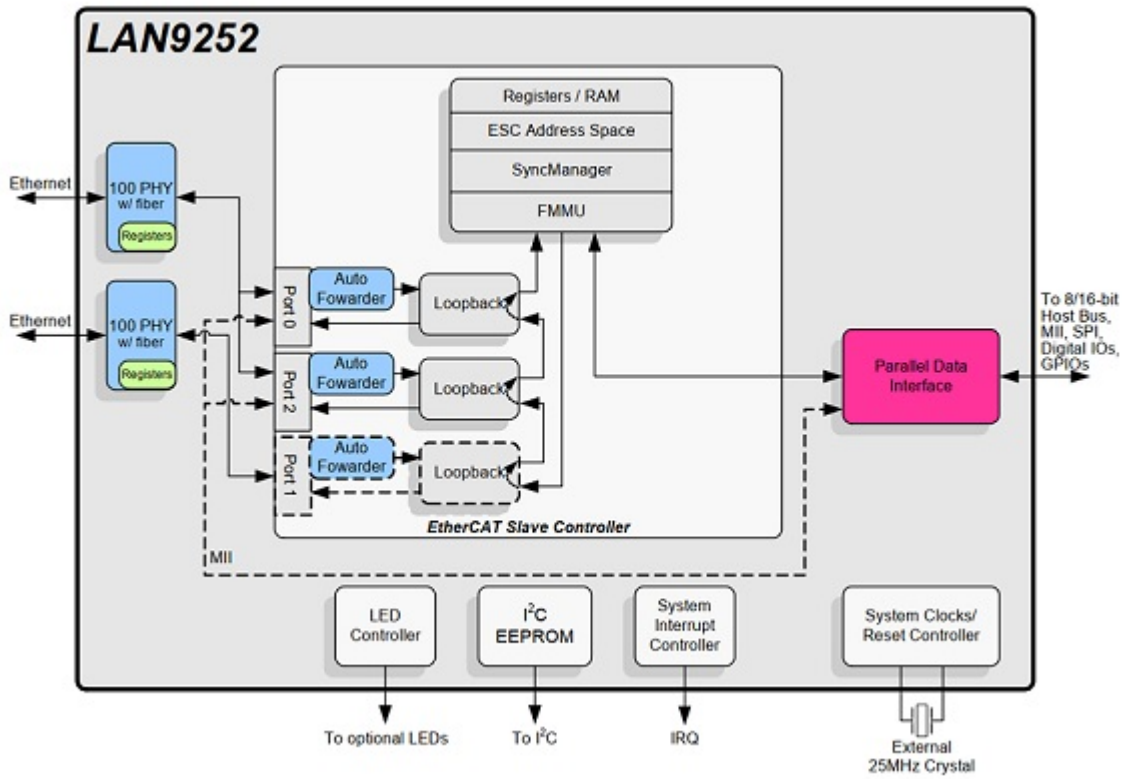


Figure 4.4: Internal structure of the LAN9252, highlighting the PDI which was selected for this application to be SPI

Selecting the features to be implemented on MCU Host according to specification and complexity THIS

Second tests with read/write functions for directly addressed registers THIS

Third tests with the EtherCAT Master At this stage a compilant EtherCAT Master was configured through a PC running *TwinCAT 3*. In order to ensure a reliable configuration two different EtherCAT devices were connected synchronizing their data with the Master. Namely, a comercial 3-Phase Motor Controller (*ELMO Controller*) and an in-house multi-protocol end effector tool. For those different data structures were declared and very simplistic update loops were programmed within the XAE environment using *SText***** programming language.

Creation of an ESI file and flashing **Mention that the existing information is either for the Beckhoff's ET1100 or PIC32 (in the case of the LAN9252 set of APIs).

Object dictionary **Mention that according to the standard *mention the standard for ESI files** and object dictionary was created matching to the one contained in the ESI file, but mapped according to the few documentation available of SOES.

Fourth tests: running the flashed device Longer tests and configuration loop due to the deepening on EtherCAT protocol. Refer to the the SMs characteristics**** Constant comparisons between the data read by the Master *this could have another image from the mapping access through XAE* and the data received by the MCU host.

4.5 Device's State Machines (DSMs)

In order to have a deterministic behaviour of the embedded system, a set of State Machines (DSM to not be confused with Synchronization Manager) were proposed and implemented as part of the project library. The DSMs follow a case comparator approach, since it was simple, yet effective and flexible enough, to work during the prototype. This characteristics was very important, since the DSMs structures were in constant change as the integration of new libraries and the functionalities developed. The proposed DSMs are as follows:

Event Handler Its purpose is to react to notifications or errors that could appear within other SMs and update the state of the LED rings in accordance.

Temperature Initializes and runs the temperature related functions.

ECAT Initializes the EtherCAT communication and activates the SOES App. It is important to mention, that this DSM is rather a synchronization of two state machines, in order to adapt the native infinite loop the SOES library is based on, to the DSMs of the system, see figure 4.5.

LED Initializes and updates periodically the RGB value of the LED Rings.

***Here come all the SMs diagrams

4.5.1 Scheduling

All the SMs and two super states are implemented as Threads using CMSIS-RTOS on STM32. All threads have fixed priorities and the timeslot assigned to each Thread is control by a OS-native delay function, which allows the scheduler to allocate CPU resources to the next priority tasks. The only time constraint is defined as desired so it is no hard real time constraints* and the overall execution follows a best effort approach. This, however, opens the door to further improvement in the sense of characterizing the task durantion. For instance, since it is quite demanding to know the exact precise execution time of each Task and without that is not

possible to think about optimizing the OS configuration -if required-, the following is a list of proposed activities that could take place in future stages of the project (out of scope)****. **This is related to a calculation of the Utilization factor that is helpful for more demanding design cases, e.g., while considering heat sinks for processors within enclosed devices.

Execution Time estimation per task each task can be isolated by software and through adding a piece of code to toggle a free GPIO at the end of the thread, a signal can be traced with a fair digital analizer (**include example of one). Omiting the rather small HAL overhead added with the GPIO control, an estimation could be achieved.

Live thread tracing a trace debugging like SEGGER SystemView**reference to fastbitlab** can be used to debug freeRTOS applications running on ARM Cortex Mx based Microcontroller such as STM32Fx. With this tool it could be possible to have at runtime a trace of the thread allocations, knowing in consequence the duration of the threads. **Further information regarding this methodology is needed.

Optimization of threads By knowing the WCET of each thread an optimization of the utilization could be carried out by using differnet OS-native features to improve the scheduling, as long as the application demands it.

Here comes a table with the priority for each task
RAte monotonic oder deadline monotonic???
EDF —-Scheduling , is this for dynamic priorities?

4.6 PCB

In this section is described the main stages regarding the manufacturing of the PCB. The schematics and PCB layout can be consulted within the annex X A

4.6.1 Development

Design *Altium* PCB design software was used to deliver the *Axis Communication Hub* prototype. During the process the designs from

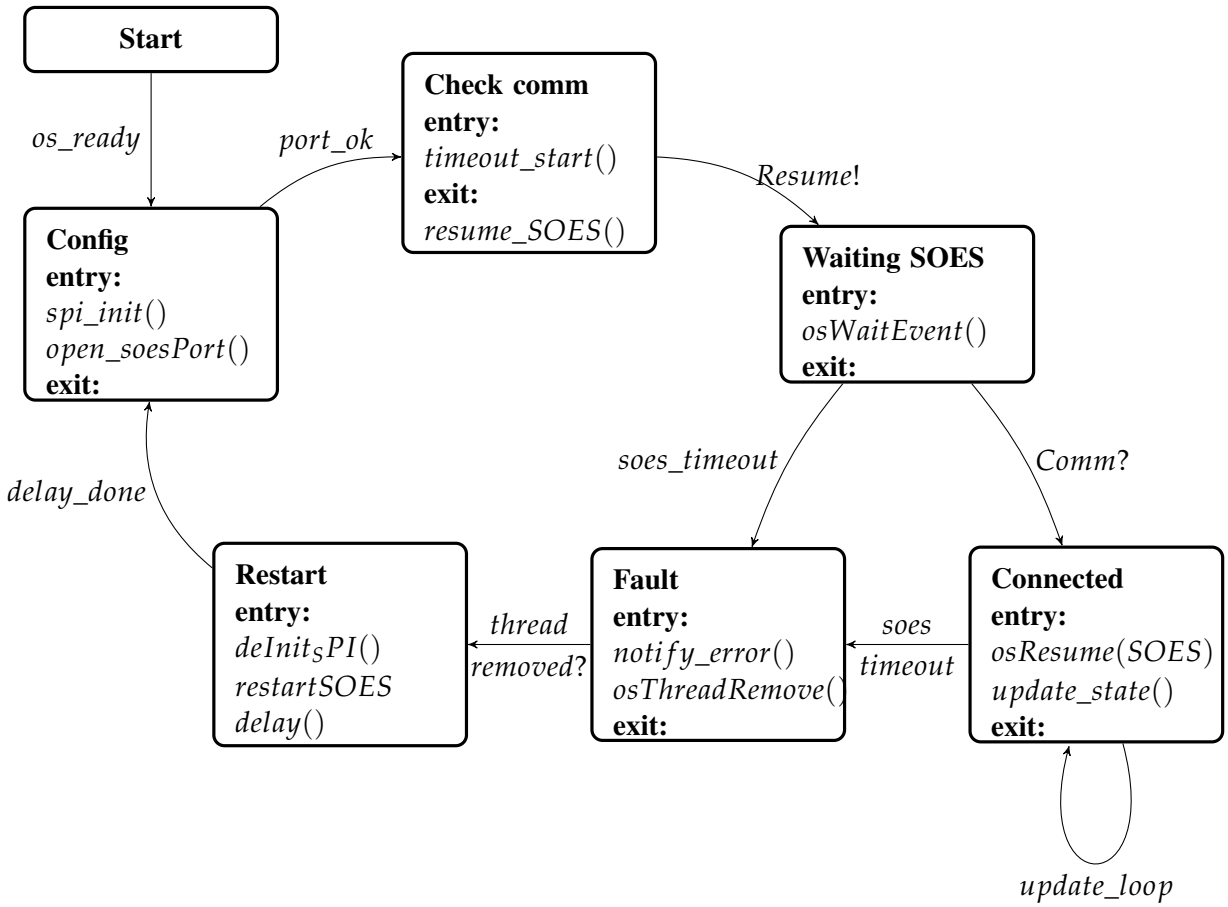
Manufacturing

Testing The overall integrity and functioning of the Overall power-on and SWD-Programming of the STM32 MCU via SWD/JTAG connector on-board, SPI connections/communication with LAN9252 Evaluation Board. Readout of directly addressed memory space, specificaly test and ID register; PWM Outputs over the two channels for WS2812 LED

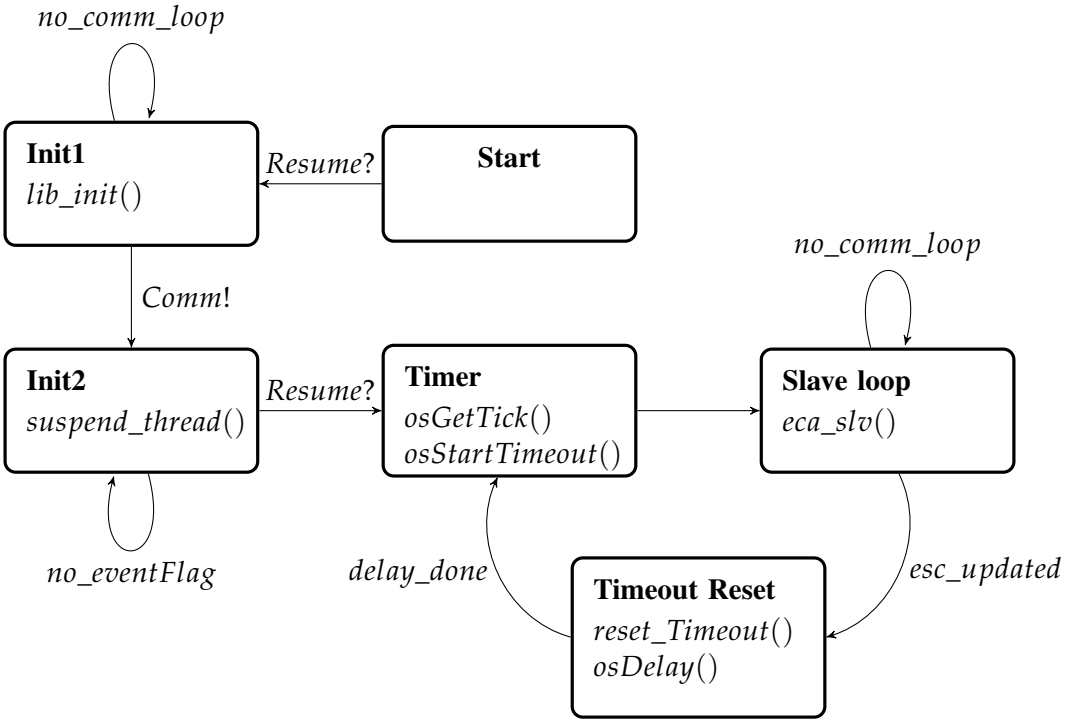
control, 1 per physical connector and 1-wire connection physycally tested by reading 1-wire sensors, as detailed in 4.2.

Here comes an image of the PCB mounted on top of the LAN9252 Evaluation Board

** Question. What would happen if the synchronization mode within an EtherCAT network wants to be integrated to those of a TSN???



(a) Synchronization state machine



(b) SOES application state machine

Figure 4.5: State machines for EtherCAT slave functionality

Chapter 5

Results

In this chapter it is presented a set* of evidences from the final results. And it is summarized the challenges and further improvements that can be considered in later versions of the *Axis Communication Hub*.

5.1 Photographic evidences*

Wireshark trace **Insert captures of the EtherCAT frames/datagrams for initialization or a change of state INIT-OP

TwinCAT **Insert a capture of XAE showing the EtherCAT Slave within the EtherCAT tree. Moreover, the data types declared in SText to update in free-run mode the values sent by the Slave device.

Board peripherals The figure that shows two LED strips and 10 temperature sensor connected to the Board. **inserts the figure**

5.2 Challenges and accomplishments

5.2.1 LED Control

An open point which did not represent an obstacle for the project, but keeps the door open to further improvement is the possibility of using more than one channel per PWM generator. The reason was this was not implemented lays on the time required for testing. The straightforward option is use one DMA channel and one PWM Channel per ring, as the MCU host contains several DMA channels* *here comes a reference to a table* this does not represent a problem. Nonetheless, it could be argued that with the same PWM X generator more channels could be controlled if they are updated in a row; this is, updating the Y number of LEDs within the Ring1 using DMA stream attached to PWM X, then switching the channel to CH+1 and, using the same DMA stream, update the Ring 2 and so on. This approach would imply

apparently longer times but if the Rings are taken as a fraction of the same buffer then it's the same time*. This could also lead to the discussion about using two streams at the same time for a total of 8 Rings with only two PWM generators and two DMA streams, since they rely on independent hardware -one of the main characteristics of the DMA modules-. The optimization of this LEDs control is not part of the scope of this project but the following links resulted interesing.*add the link*

*Mention the priority of the functionalities « important, maybe in introduction

5.2.2 Temperature acquisition

here comes the problematic about the parasistic power supply and the polling of the hardware status, this means, there is more space for improvement with YIELD or timeouts from OS funcitons.***

5.2.3 SOES

Understand the variations between Little/Big Endian within buses. That the library is based on polling by one by one the unique FIFO register of the ESC, such that the PDOs can be written and read in the PD RAM, as it was thought that a direct access to the RAM would permit the User Address Space of the MCU to be extended and the access then would be transparent for manipulating the data, and that the SM would permit only the safe access from the Master device to read/write the same address space. It could be then understood that there is a significant delay that van be even measure, that in some papers is determined as "Stack Delay" which in this case *Reference the paper of the design of the communication/delay analysis*. The advantages of reducing it through the implementation of the SQI or the HSI interface -which at the end does not have that much purpose since for achieving less delays the FPGA or ASIC maybe implemented-. The question, is it still sensible to develop EtherCAT devices with this approach or should it be switched completely to the FPGA and ASIC approach. An analysis considering costs and other things could be then carried out. *** **Making available the Szstem Interrupt Controller without polling would also increase the performance of the application, this can be aimed in next versions where physically the the pins are connected to the Host MCU. **A brief analysis of the bandwidht achieved as the bandwidth necessarz to make sensible a change of PDI would make sensce, since the HBI****reference to internet or anz specification of HBI*** is mainly foucsed to Board-To-Board connections or communication between ASICS, hence 4Gbps per pin die-to-die connectivity with low latency may not be needed at all *https://www.synopsys.com/designware-ip/interface-ip/die-to-die.html**. **Important to mention is the HBI only helps exchanging process data, so it could be implemented for applications where data chunks are needed to be transmitted at high bandwidth since it is a direct fifo access to the ESC's SRAM. **Here should be also

commented the problems by reading and writing, mainly writting due to the noisy medium (cabling).

5.2.4 State Machines and RTOS

5.2.5 PCB and hardware

5.2.6 Challenges

The overall functionalities of the PCB were achieved from the first attempt, leaving only one disadvantage that appeared until the tests for the readout of temperature sensors were in the final stage. According to the library integration process commented in 4.2, during the first month a rather simplistic approach was coded to have access to the temperature values. The final library that was included made use of the UART peripheral in a full duplex mode, which means that two independant pins were explicetely needed, namely RX and TX. Since, within the first approach only one GPIO was being driven, the current physical routing of the board was not appropiate to test properly the one-wire communication. Nonetheless, arrangements werd carried out to use the UART RX/TX pins available in the JTAG connector. This approach was marely for testing and will be corrected in any further development out of the scope of this Project Research.

SPI communication

During the final tests some communication errors appeared that were tracked back to the following conditions:

Voltage drops The on-board Low-Dropout voltage regulator* LDO that provided 3.3 V had random failures, where the output voltage was from 3.28 to 3.29 V. This caused that the LAN9252 chip was not always ready to work as intended. This behaviour increased from not-happening to failed completely to start. The mentioned condition obly affected the ESC functionalities, whereas the host MCU worked realiably. To revert this failure mode, an external 3.3V power source was used to continue the current tests.

Noisy channels While troubleshooting the communication problems mentioned above, an SPI communication test was carried out, such that the effects of different configurations could be characterized. It was concluded that the higher frequencies the SPI was set up, the greater communication faults appeared, leading to error states in the internal SMs. Moreover, even within rhe stable range, the higher frequencies were not implying shorter execution times. A summary of this observations are presented in the table **??**.

Regarding the no-improvement of the execution time at a high-speed SPI configuration, the figure **??** shows a relatevely constant software delay within the words being sent over the

SPI bus. Therefore, it does not matter how fast the words are sent, for this Software delay is always present *write the order of the delay*. This emergent characteristic of the system could be optimized as long as it is determined, whether its impact is critical on the overall throughput of the system. The reader should take into consideration the rather small amount of data and the refresh speed of the overall system. However, this could be a stating point for improvement of the SOES stack, for those SW delays can be approached by using a fixed sending buffer through DMA instead of the usage of the native* API functions that send word by word. A further evaluation of the worthiness* of the library changes may be needed.* The above mentioned challanges are addressed within the proposals for improvement presented in the section 6.1.

Chapter 6

Conclusions and further development

Overall status: Board and basic SOES features ready for further EtherCAT development.

Even though, there have been delays due to the libraries being ported, and channels being noisy before having the PCB working, we're in a stage where we can ensure that the PCB manufactured and ported libraries allow to further develop an EtherCAT compatible application. Therefore, in the following calendar days the following tasks will be priorized to complete the main tasks:

6.1 Further development

**Inserts probably the challenges in the same strutcture as in the implementation part.

Bibliography

[Bec13] Beckhoff Automation GmbH. Urban Automation PC-based Control for Urban Environments. https://m.beckhoff.com/english/applicat/urban.htm, April 2013. Last visited: 16/09/2020.

[Bec17] Beckhoff Automation GmbH. EtherCAT Slave Controller: Technology. https://www.beckhoff.com/english/downloadfinder, February 2017. Last visited: 16/09/2020.

[Bec18] Beckhoff Automation GmbH. ETG.2200 V3.1.0. https://www.beckhoff.com/english/downloadfinder, November 2018. Last visited: 16/09/2020.

[Bec20a] Beckhoff Automation GmbH. EtherCAT - the Ethernet Fieldbus. https://www.ethercat.org/en/technology.html, January 2020. Last visited: 16/09/2020.

[Bec20b] Beckhoff Automation GmbH. TwinCAT/BSD: operating system for Industrial PCs. https://www.beckhoff.com/english.asp?highlights/twincat-bsd/default.htm, June 2020. Last visited: 16/09/2020.

[Bor08] Detlef Borchers. Open Source trifft die Industrie. https://www.heise.de/newsticker/meldung/Open-Source-trifft-die-Industrie-201855.html, April 2008. Last visited: 16/09/2020.

[Car20] T. Carlsson. Industrial network market shares 2020 according to HMS Networks. https://www.anybus.com/about-us/news/2020/05/29/industrial-network-market-shares-2020-according-to-hms-networks, May 2020. Last visited: 11/09/2020.

[DC17] Raimarius Delgado and Byoungwook Choi. Real-time servo control using ethercat master on real-time embedded linux extensions. *International Journal of Applied Engineering Research*, 12:1179–1185, 11 2017.

[emb20] emb4fun. emb4fun's OpENer patch homepage. https://www.emb4fun.de/archive/chibiosopener/index.html, January 2020. Last visited: 11/09/2020.

[Fel02] M. Felser. The fieldbus standards: History and structures. 2002.

[Gmb20] Beckhoff Automation GmbH. EtherCAT Slave Controller Overview. https://www.ethercat.org/en/downloads.html, January 2020. Last visited: 17/09/2020.

[Gro04] G. Gross. Infineon to pay 160M fine for DRAM price-fixing. https://www.computerworld.com/article/2566841/infineon-to-pay–160m-fine-for-dram-price-fixing.html, September 2004. Last visited: 17/09/2020.

[Han20] Han's Robot Germany. Homepage. http://hansrobot.de/home, September 2020. Last visited: 16/09/2020.

[HWHP] W. Hagemeister, Weber, R. Hacker, and F. Pose.

[IEC20] IEC. OPC unified architecture - Part 14: PubSub. https://webstore.iec.ch/publication/61108, July 2020. Last visited: 11/09/2020.

[IEE20] IEEE. Time-Sensitive Networking (TSN) Task Group. https://1.ieee802.org/tsn/, January 2020. Last visited: 11/09/2020.

[II20] IEEE and IEC. IEC/IEEE 60802 TSN Profile for Industrial Automation. https://1.ieee802.org/tsn/iec-ieee-60802/, January 2020. Last visited: 11/09/2020.

[Inf18] Infineon Technologies AG. XMC4700 / XMC4800 Datasheet. https://www.infineon.com/cms/en/product/, September 2018. Last visited: 17/09/2020.

[ISO17] IEEE ISO, IEC. 8802-3:2017 Part 3: Standard for Ethernet. https://www.iso.org/standard/72048.html, 2017. Last visited: 11/09/2020.

[JDH02] B. C. Johnson, D. G. Dunn, and R. Hulett. A comparison of the ieee and iec standards processes. In *Record of Conference Papers. Industry Applications Society. Forty-Ninth Annual Conference. 2002 Petroleum and Chemical Industry Technical Conference*, pages 1–12, 2002.

[JYJP20] Sanghoon Ji, Donguk Yu, Hoseok Jung, and Hong Seong Park. Real-time robot software platform for industrial application. In Antoni Grau and Zhuping Wang, editors, *Industrial Robotics*, chapter 6. IntechOpen, Rijeka, 2020.

[MGSR20] M. A. Metaal, R. Guillaume, R. Steinmetz, and A. Rizk. Integrated industrial ethernet networks: Time-sensitive networking over sdn infrastructure for mixed applications. In *2020 IFIP Networking Conference (Networking)*, pages 803–808, 2020.

[MHF+20] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini. Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics*, pages 1–14, 2020.

[MLH+17] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis. Xbotcore: A real-time cross-robot software platform. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 77–80, 2017.

[MLT18] L. Muratore, B. Lennox, and N. G. Tsagarakis. Xbotcloud: A scalable cloud computing infrastructure for xbot powered robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.

[Nor20] M. Norris. A Cannabis Equipment Supplier Turns to Automation to Extract CBD at Commercial Scale. https://www.automationworld.com/home/article/21173096/a-cannabis-equipment-supplier-turns-to-automation-to-extract-cbd-at-commercial-scale, August 2020. Last visited: 14/09/2020.

[OpE20] OpENer community. OpENer - Open Source EtherNet/IP(TM) Communication Stack. http://eipstackgroup.github.io/OpENer/index.html, January 2020. Last visited: 11/09/2020.

[OSA19] OSADL. Building an Open Source OPC UA/TSN Ecosystem. https://www.osadl.org/uploads/media/OSADL-OPC-UA-TSN-Open-Source-Ecosystem-LoI-2nd-edition-V6.pdf, January 2019. Last visited: 11/09/2020.

[PERK18] Pfrommer, Ebner, Ravikumar, and Karunakaran. Open source opc ua pubsub over tsn for realtime industrial communication. 07 2018.

[RT-20] RT-LABS. EtherCAT Software Stacks. https://rt-labs.com/products/ethercat/, January 2020. Last visited: 11/09/2020.

[Ser18] Sercos. Ethernet TSN heralds a new era of industrial communication. https://www.sercos.org/technology/sercos-in-connection-with-tsn-opc-ua/sercos-in-connection-with-tsn/, January 2018. Last visited: 11/09/2020.

[Ser20] Sercos. Driver Software. https://www.sercos.org/technology/implementation/driver-software, January 2020. Last visited: 11/09/2020.

[SKJ18] S. Schriegel, T. Kobzan, and J. Jasperneite. Investigation on a distributed sdn control plane architecture for heterogeneous time sensitive networks. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–10, 2018.

[Son20] Maneesh Soni. EtherCAT on Sitara Processors. Technical report, Arm Microprocessor Group Texas Instruments, April 2020. Last visited: 17/09/2020.

[Sta00] DN Staff. SERCOS picks up the pace. https://www.designnews.com/sercos-picks-pace, August 2000. Last visited: 11/09/2020.

[Sta20] SERCOS Staff. SERCOS performance. https://www.sercos.org/technology/advantages-of-sercos/performance/, January 2020. Last visited: 11/09/2020.

[VZS19] S. Vitturi, C. Zunino, and T. Sauter. Industrial communication systems and their future challenges: Next-generation ethernet, iiot, and 5g. *Proceedings of the IEEE*, 107(6):944–961, 2019.

[XJY11] Hu Xing, Huan Jia, and Liu Yanqianga. Motion control system using sercos over ethercat. *Procedia Engineering*, 24:749–753, 12 2011.

Appendix A

PCB drawings and layout

here comes the electrical diagrams and the pcb layout exported by Altium

Appendix B

Source Code?

This appendix shows the source code which was self written SMs and auxiliar wrappers. For any other code, namely the one-wire and soes library, since they are under the GNU* license, it is easy that the reader look for oneself the code:

LED WS2812b Driver Original version only for one channel and suited for **add the ST family** « ** mention in the challenges 4.1 that there was a DMA incompatibility within this two families. **refer to the porting guide provided by ST.

One-Wire generic driver Original compatible version which includes a sample that runs in a infinite while-loop.

SOES Library Original version without the low level functions. « *** Mention that originally the support for ST is not provided but only for: **look the processors natively supported. 4.4.

**Add code