Data acquisition system for fine surface process monitoring

Nicolle Nonato

Dissertation presented to the School of Technology and Management of Bragança to obtain the Masters Degree in Industrial Engineering in Double Degree program with UTFPR.

Work oriented by:

Prof. Dr. José Luís Sousa de Magalhães Lima

Prof. Dr. Nelson Rodrigues

Prof. Dr. Fábio Luiz Bertotti

Bragança

2020

Acknowledgement

I want to thank all the teachers, partners, and leaders on the On Surf project, my advisors, professor José Lima, Nelson Rodrigues and professor Fábio Bertotti, the research center CeDRI, the IPB, and the school ESTIG. I also want to thank my colleagues who accepted this challenge with me, João and Marcos.

Furthermore, my family and friends that accompanied the development of the project and supported me all the time.

Abstract

Thin surface manufacturing processes are known worldwide and used in a variety of applications. Among the various manufacturing processes, metal stamping and plastic injection are indispensable techniques for the success of companies in this field, where innovation is essential to remain competitive. In this sense, several thin surface optimization procedures are explored in the On-Surf project. This work focuses on a sensory data acquisition system on these surfaces, meeting the requirements of modularity and scalability, meaning that they can be easily expanded or contracted, depending on the number of sensors required. The designed system consists of several modules, one dedicated to signal conditioning to temperature and pressure sensors, the other has as purpose acquiring data and sending this data to a hub via Ethernet UDP communication, using the Olimex ESP32-PoE-ISO as an embedded system. Finally, Raspberry Pi was chosen as a data hub module to aggregate the information received by ESP32 modules. For validation, several tests were performed to verify the structure robustness. As result, each conditioning module can support up to eight sensors simultaneously, the signal of the sensors was acquired by the embedded system at 1 kHz sampling rate, and UDP transmission reached a communication throughput of 2000 packets per second.

Keywords: On-surf; modular; scalable; data acquisition; timestamping.

Resumo

Os processos de manufatura em superfícies finas são conhecidos mundialmente e utilizados em uma variedade de aplicações. Entre os vários processos de manufatura, a estampagem metálica e a injeção plástica são técnicas indispensáveis para o sucesso das empresas nesse campo, onde a inovação é essencial para se manter competitivo. Nesse sentido, vários procedimentos de otimização em superfícies finas são explorados no projeto On-Surf. Este trabalho enfoca um sistema de aquisição de dados sensoriais nessas superfícies, atendendo aos requisitos de modularidade e escalabilidade, conforme o número de sensores necessários. O sistema projetado consiste em vários módulos, um dedicado ao condicionamento de sinal; o outro tem como objetivo a aquisição de dados e enviá-los para um concentrador via comunicação Ethernet, usando o Olimex ESP32-PoE-ISO como sistema embarcado. Finalmente, o Raspberry Pi foi escolhido como um módulo de concentrador de dados para agregar as informações recebidas pelos módulos ESP32. Para validação, vários testes foram realizados para verificar a robustez da estrutura. Como resultado, cada módulo de condicionamento suporta oito sensores simultaneamente, o sinal dos sensores foi adquirido pelo sistema incorporado a uma taxa de amostragem de 1 kHz e a transmissão UDP alcançou uma taxa de transferência de dados de 2000 pacotes por segundo.

Palavras-chave: On-surf; modular; escalável; aquisição de dados; timestamping.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and motivation

Thin surface manufacturing processes are known worldwide and used in a variety of applications such as automotive and agricultural machinery, kitchen appliances, surgical instruments, and consumer appliances. Among the forming processes, there are mechanical stamping and plastic injection, both today popular. Both occur in cold-pressing operations and may be cut or deformed.

Metal stamping can present some problems, including ruptures, ripples, wrinkles and spring return, as well as in the plastic injection process which can present some complications, such as frost spots and plastic suction. Such problems have consequences for the manufacturer as the machinery requires replacement or maintenance. Thus, production must stop to make certain adjustments, wasting time and therefore money.

Within the project, a module with remote access is proposed for the acquisition of temperature and pressure data and subsequently, the available data are provided to factory departments that employ metal stamping or plastic injection processes.

This dissertation is part of the On-Surf project, a national project in Portugal that aims at innovative solutions in the surface engineering area. The project includes two other dissertations developed at IPB. One is about the signal conditioning system that comes from the sensors and provides the signal for the data acquisition module, which can be called Analog Front End of the project. The other dissertation addresses the data hub, its connection, and transmission with ESP32, posting of data to the database, and linking with the graphical visualization. The On-Surf project aims to develop thin surface modification processes and focuses on sensed surfaces for applications in metal stamping and plastic injection molds.

1.2 Objectives

The project's contributions include the development of a system with modular and scalable architecture capable of measuring temperature and pressure values in thin surface molds, intending to optimize the manufacturing processes using these surfaces, and promoting better control quality of the final product. It is also expected to track back the manufacturing process, store the data and make it available for remote access.

The development of this work should address the identification of the problem related to data acquisition on thin surfaces, the definition of the system architecture, the design and implementation of software and hardware. Then, the prototype needs to be validated and the work documented.

1.3 Document Structure

This dissertation is divided into six chapters and it has two appendices. This chapter deals with a brief introduction containing background, project context, and objectives. In Chapter 2, the state of art is a collection of important sources about the work. Metal forming, data acquisition systems, and modular systems are some of the important approached concepts in this project. Chapter 3 presents the methodology, approach and modeling of the project. In chapter 4 it is explained how the work was developed, the stages of the work, the hits, and misses. Chapter 5 presents the evaluation of results, whether the work went well, and how to adjust errors. Finally, in Chapter 6, the conclusions are presented, where the main objectives and the original schedule are evaluated and emphasized. Chapter 6 also points out some future work. On appendix A there is the code developed for the project.

Chapter 2

State of Art

2.1 Material Formability

Formability is a feature of the material which refers to the ease of metal to take shape without breaking or fracturing. It is an essential property for the material to undergo a metal forming process [1].

According to [1], the optimization of the sheet metal forming process can be accomplished through the forming limit diagram (FLD), which is a very effective method. The diagram is developed by etching a grid of circles on the surface of a sheet metal. Then the sheet metal is exposed to damage and it is made a graph which shows the major strain versus minor strain. This plot is called Keeler-Goodwin forming limit diagram, and it gives the limiting deformations corresponding to safe deformations.

The prediction of wrinkling is important for the design of stamping and deep-drawing processes. Wrinkling is unacceptable in the outer skin panels where the final part appearance is crucial [1].

As referred by the authors in [1], to obtain the best quality in the final product, one methodology adopted is the optimization of process parameters by adding

metal forming reference parameters as responses. The temperature and pressure are parameters directly related to the quality of the final product.

2.2 Flexibility in Manufacturing Processes

Metal forming is a manufacturing process that employs the creation of parts from the deformation of a metal body. Plastic injection molding is a plastic processing technology for producing parts by injecting molten material into a mold.

In sheet metal forming, some parameters are taken into account to obtain the best result in the final product to avoid mistakes. There are two distinct types of parameters, geometrical and material. In this project, only two of the geometrical parameters are studied, which are the blank temperature and the blank holding pressure.

Such as in sheet metal forming, plastic injection also has some parameters that must be taken into account. In the present project, the same parameters are analyzed in the metal forming process and plastic injection, which is temperature and pressure.

The manufacturing sector needs flexibility due to the advent of a new industrial revolution, the so-called fourth industrial revolution or Industry 4.0. Conventional mass production is changing, needs more flexibility and on-demand manufacturing, which brings new technological challenges [2].

As explained by the author in [2], [3], for mold-based manufacturing processes such as casting and metal molding, flexibility can be a challenge. However, in manufacturing processes such as machining and welding, this flexibility is inherent. Thus, three key factors to make the industry more flexible according to Industry 4.0, are connectivity, intelligentization, and automation, as illustrated in Figure 2.1.

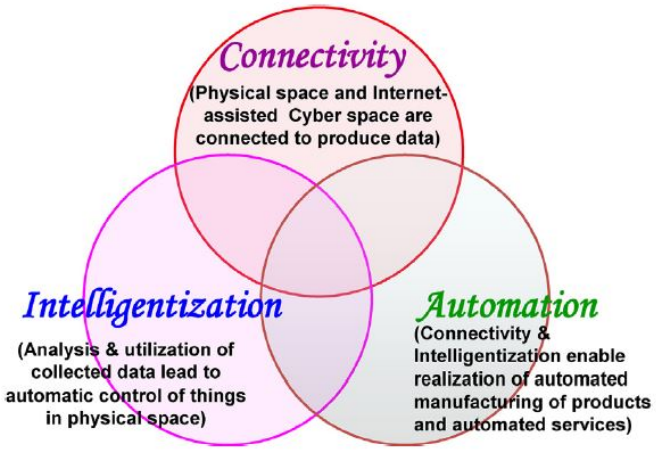

Figure 2.1: Industry 4.0 key factors [2]

This project gets together the key factors, connectivity by producing data through the sensors in the mold and the cloud connection, intelligentization by using collected data to control temperature and pressure parameters in metal forming and plastic injection processes, and automation by enabling automated product manufacturing by metal forming and plastic injection processes.

The three key factors are related to Cyber-Physical Systems (CPS), which are defined by [4] as systems involving the physical world and its processes having a strong connection with computational entities. These systems are characterized by three main characteristics:

- Intelligence or smartness: provides the ability for elements to acquire information from their surroundings and act autonomously;
- Connectedness: describes the ability to set up and use connections to other elements of the system, for cooperation and collaboration;
- Responsiveness: response towards internal and external changes.

Cyber-Physical Systems are the link between the real world and the virtual

world. These systems are used wherever complex physical systems need to communicate with the digital world to allow their performance to be optimized and their efficiency to be improved.

2.3 Data Acquisition Systems

As the authors explain in [5], data acquisition is called a process in which a computer acquires digitized analog data (converted to digital). The acquisition of a single data point is called "sampling" the analog signal and the value of the data point is called "sample". There are several applications for this process. In a storage application, such as digital audio or video recording, the internal microcomputer will store the data and then transport them to a digital-to-analog converter (DAC) at a later time to reproduce the original analog signal. In a process control application, the computer can examine the data or perform computations on them to determine what control outputs to generate [5].

Data acquisition systems are designed to acquire data and convert it into a format that can be viewed and manipulated on a computer. Figure 2.2 represents the operation of a DAS. Initially, the sensors must collect the data, the signal conditioning circuit converts this data to a suitable analog signal, such as voltage or current. Then the analog signal goes to an analog-to-digital converter and the output signal becomes a digital form. Finally, the digital signal is accessed and controlled by data acquisition software on a computer.

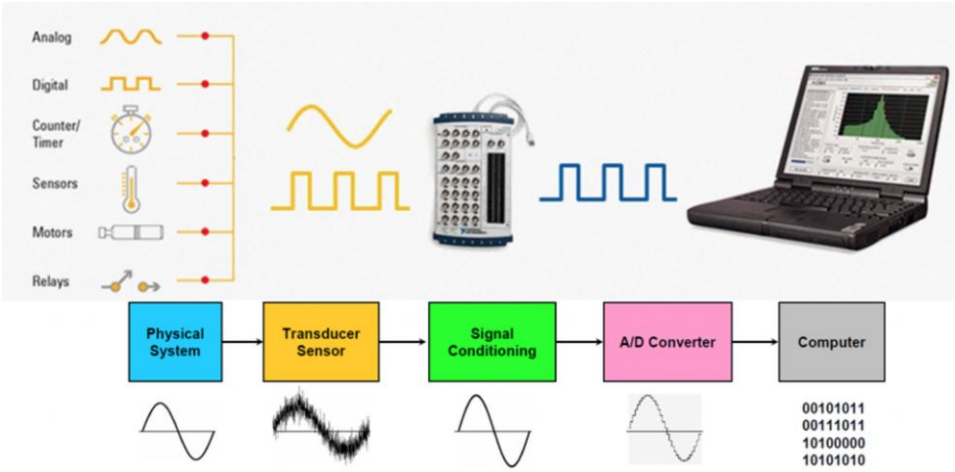

Figure 2.2: Data Acquisition System [6]

The field of DAS is complicated but takes place in almost every industrial process to optimize and improve the production and get a better quality in the final product. Since the first data acquisition system, new technologies have been introduced in this area with smaller and simpler systems.

Currently, due to the studies and technologies launched in the field of DAS, it is possible to measure and analyze almost any physical system. There are many types of systems, but in general, they have four components, the sensors used to collect data from physical systems, the signal conditioning circuitry, the analog-to-digital converter (ADC), and the computer on which the data can be visualized and analyzed [7].

A major type of data acquisition system is the supervisory control and data acquisition system (SCADA), included in this system are the personal computer (PC) and the universal serial bus (USB), two other essential types of data acquisition systems. PC data acquisition involves all systems and devices that require connection to a host computer, which is most systems. USB data acquisition is a type of PC data acquisition but can be considered a method of its own. It uses

a serial bus as a staple to connect and maintain conversation between data acquisition devices and their host controllers. The USB method has some benefits, among them, has higher bandwidth and the ability to provide power to peripheral devices [7].

SCADA are complex systems that use countless devices and programming to control and manipulate operations. Multiple sites spread over long distances can be supervised and controlled with these systems, but they can be vulnerable to cyber-attacks. Devices used in SCADA systems include computer ports, graphical user interfaces, and network data communications; proportional integral derivative (PID) controllers or programmable logic controllers (PLC) can also be used to interface with machines or process plants [7].

2.3.1 Sensors

Sensors are devices that detect and responds to a signal, this signal can be produced by energy, motion, force, or it can be intrinsic to the environment, such as temperature. Typically, sensors convert a signal into an electrical - analog or digital - readable output [8]. For example, there are many types of temperature sensors, the thermistor represents a variation on the electric resistance, on the other hand, the thermocouple gauge works with a temperature gradient and it returns a potential difference, a voltage.

The development of any DAS should start with the physical property being measured and exactly what type of data is needed to be collected. However, it is important to know that any sensor included in the process can affect the system.

The temperature sensor used in this project is a PT100 RTD sensor. It is a platinum-resistant thermometer (PRT) whose resistance varies with ambient temperature. The name given to the sensor is equivalent to the resistance value at 0°C, in this case, the PT100 has a resistance of 100 ohm at 0°C. This sensor

was chosen due to its similarity to the developed real temperature sensor, has a sensitivity of 0.1°C and works in high-temperature ranges [9].

The sensor used to collect pressure data is the strain gauge, which converts the deformation (strain) of an object into electrical resistance. The variation of resistance is very small and difficult to measure, so it should be used Wheatstone bridges and the bridge output voltage measured. Strain or stress are the results of an object subjected to external forces. As a technical term, "strain" is divided by deformation and traction, defined by a positive or negative sign. Thus, strain gauges can be used to capture expansion and contraction [10].

2.3.2 Signal Conditioning

Applications involving environmental or structural measurements, such as temperature sensors or strain gauges, demand signal conditioning for the data acquisition device accurately measure the signal. If the signal conditioning is not suitable, the system can have problems in the process. However, it should be considered that each sensor needs especial signal conditioning, which can diversify widely in functionality depending on the sensor. It is necessary to understand the circuitry required to ensure accurate measurement.

Both sensors, temperature and strain gauge, require excitation to operate. The RTD or temperature sensor requires amplification and low pass filters for suitable operation, whereas the strain gauge needs the Wheatstone bridge. To optimize the work, there are integrated circuits (ICs) with these functions and the necessary conditioning, several models of ICs can provide amplification, calibration and temperature compensation.

When building a signal conditioning system, some variables are key factors that contribute to the success of the whole project. According to NI instruments, these variables are [11]:

- Integration is the ability of the signal conditioning system to integrate easily with the rest of the system. The interaction between different components of the measurement chain helps characterize expected results and troubleshoot unexpected ones;

- Connectivity must be a wide variety of options. As the application changes, so does the need for different connectivity. Creating new connectivity devices for changing test requirements can become unmanageable over time as technology requirements change;

- Expandability is the flexibility needed. By architecting the system in a modular way, it has more flexibility to change and expand both channel count and signal mix. Architectures with high levels of cross-functional reliance can require a massive overhaul to make even relatively minor alterations;

- Isolation is necessary because some signals can compromise the integrity of the entire data acquisition system;

- Bandwidth has to be high enough to handle the data throughput needed and accommodate future channel-count growth;

- Software can minimize project costs by developing in an environment designed specifically for this type of application because much of the total cost of a test and measurement system is in developing applications when resources and engineering time are required, development and testing;

- Configuration and installation should be easy. An ideal signal conditioning system polls the hardware, reports which equipment is present, and provides a software interface for assigning all settings;

- Calibration must be done periodically to make the most accurate measurements possible. Most measurement devices are calibrated at the factory, but the accuracy drifts over time;

- Maintenance in case the engineer who designed the system moves to a different company or project. It is necessary to compile all of the system information into a formal document. Troubleshooting the system, adding new functionality, or duplicating the system can be nearly impossible without detailed documentation.

In addition to signal conditioning, the acquired signals must be processed and analyzed by a computerized system.

2.3.3 Computer Hardware and Software

The computerized system is necessary for collection and analysis, once the signal has been amplified and cleaned up. This computerized system can be a microcontroller, which already has the analog-to-digital converter and all the work is embedded. Choosing the microcontroller is difficult, characteristics should be right to the job, such as the RAM, the ADC, the communication.

There are several considerations to be made since all the microcontroller features depend on the project. Microcontroller industries talk about making a list, so some of these considerations are about power, it should be considered whether the project is running on batteries or not. Speed is very important if the project crunch a lot of numbers or gather huge amounts of data, also the sampling rate and how many ADCs exist. Naturally, the price of the board depends on the budget.

Communication is fundamental if the project intends to connect with other devices. Among various communication protocols and technologies, Ethernet is one of the most well-known, also serial interfaces like USB, serial peripheral interface

(SPI), inter-integrated circuit (I2C), parallel communication, WiFi and Bluetooth.

2.4 Modular and Scalable Systems

2.4.1 Modular Systems

Modularity is the ability of a system to separate its components and how they can be recombined and refers to both the coupling between components and the degree to which the "rules" of the system architecture allow (or prohibit) mixing and matching of components [12]. All systems can be considered modular to some extent, since very few systems have a fixed architecture where the components are inseparable and can not be rearranged [13].

Several systems tend to migrate to increasing modularity. Systems that originally were strictly aggregated may be separated, having their components coupled and blended, allowing much greater flexibility [13].

In [13], the author uses the personal computers as an example of modularity. Computers were originally introduced as complete packages (such as Intel's MCS-4, Kenback-1, Apple II or Commodore PET), but quickly evolved into modular systems that allow mixing and combination of components from different suppliers [13]. This development trend of modular systems is growing in many fields, as well as home appliances, which have been developing innovative products such as cooktops, which used to be stoves with oven, gilder, and coif, today are only stoves, and the oven is sold separately.

In [13], the author claims that modularity helps to boost the system flexibility because it provides an exponential increase of possible configurations for a certain set of inputs. So, referring to modular data acquisition systems (DAS), it can be defined by a 'chassis' that contains some 'modules' [13].

The modular architecture allows single systems to be built by recombining their

modules. There are many advantages to using modular systems in custom units, including cheaper and faster development, reliability and a robust upgrade and modification path.

2.4.2 Scalable Systems

Scalability is another desirable attribute when developing or designing a system. The concept is related to the ability of a system to adapt an increasing number of elements, objects or data, to perform increasing process volumes and/or to be susceptible to expansion [14].

Trying do define scalability, the author cites an observation by Greg Papadopoulos of Sun Microsystems that the amount of online storage on servers is expanding faster than Moore's law [14].

To scale a system or a system component, it is essential to know the data structure models used and the algorithms implemented and how the components communicate. Repeated activities cause waste and may fail the attempt to scale the system. Scalable systems can develop well with low load, but with an increasing load, they undergo substantial degradation. Ethernet does not have load scalability because of the inefficiency in bandwidth usage, which is caused by the high collision rate at heavy loads [14].

According to [14], scalability is classified in 4 types: load scalability; space scalability; space-time scalability and structural scalability. A system can have more than one scalability types. Space-time scalability, for example, has the space scalability intrinsic. Furthermore, there is scalability over long distances that embraces distance scalability and speed/distance scalability [14].

A very important application for scalable systems used in this project is the ability of a data acquisition system to process and analyze the number of data provided, increasing or decreasing over time.

Allied to modular and scalable systems, it is very important the security of the system and the documentation. This digital documentation is together with timestamping, a way to permanently preserve the digital signature. When programming, a timestamp should always be added to prevent future issues, such as the confidence that that document belongs to that person, and if it was changed, it was changed by someone else, among other issues that may arise.

2.5 Internet of Things and Industry 4.0

Related to the task of building and developing a scalable modular system, it is necessary to create communication between these modules with integrated identification, detection and/or actuation skills. Internet-of-Things (IoT) is related to digital and physical environments that can be interconnected through appropriate communication technologies. This project can have devices capable of gathering and transmitting data between them over a network without requiring human-to-human interactions. This ability is directly related to the Internet-of-Things and defines the system as an IoT system.

In [15], the authors address that the conventional concept of the Internet will change and the use of the Internet will happen as a global platform to allow intelligent machines and objects to communicate, dialogue, calculate and coordinate. However, the vital role of the Internet continues to be the main support for the sharing and dissemination of global information, linking computing and communication skills to physical environments through a wide range of resources and technologies [15]. The possibility of change is possible because of the electronics integration into everyday physical objects, making them "smart" so they are included in the global cyber-physical infrastructure [12]. It would open up new opportunities for the Information Technology (IT) sector.

According to [15], the term Internet-of-Things can be described in three parts: the network resulting from the connection between smart objects and Internet technologies, the set of necessary technologies (including sensors, actuators, etc.) and lastly, the applications and services that use these technologies to open new business and market opportunities [15].

Closely related to the Internet-of-Things, there is the fourth industrial revolution or Industry 4.0. Among the main objectives of Industry 4.0, is achieving the highest level of operational efficiency and productivity, as well as automation.

As state by Lu in [16], the top five features of Industry 4.0 are scanning, optimizing and customizing production; automation and adaptation; man-machine interaction (MMI); value-added services and business, and automatic exchange of data and communication.

The author Lu also cites some definitions, in [16], made by other authors, according to the Industrial Internet Consortium, Industry 4.0 is *"the integration of machines and complex physical devices with network sensors and software used to predict, control and plan better social and business results."* Henning and Johannes, from the German academy of engineering sciences, define Industry 4.0 as *"a new level of organization and value chain management throughout the product life cycle."* and Hermann, from the 49th Hawaii International Conference on System Sciences, define by *"a collective term for technologies and concepts of value chain organization."*

Industry 4.0 brings interconnection and automation to the traditional industry in a facilitated way. Among the main goals of Industry 4.0 are mass customization of manufactured products, automatic adaptation, and loosening of the production chain, facilitating communication between parts, products, and machines, applying man-machine interaction paradigms (MMI), optimizing production in smart factories and provide new technologies and business opportunities for interaction

in the value chain. The revolution brings disruptive innovations to supply chains, business models, and business processes. At the core of Industry 4.0 are real-time production, service orientation, modularity, decentralization, interoperability, and ultimately virtualization [16].

Further, as referred by [16], Industry 4.0 can be summarized as an integrated, tailored, optimized, service-oriented and interoperable manufacturing process, correlated with algorithms, big data, and high technologies.

The major advantage of Industry 4.0 is interoperability that according to the Merriam Webster dictionary is the *"ability of a system (such as a weapons system) to work with or use the parts or equipment of another system."*

Interoperability, as shown in Figure 2.3, can be divided into four parts: operational or organizational, systematic (applicable), technical, and semantic.

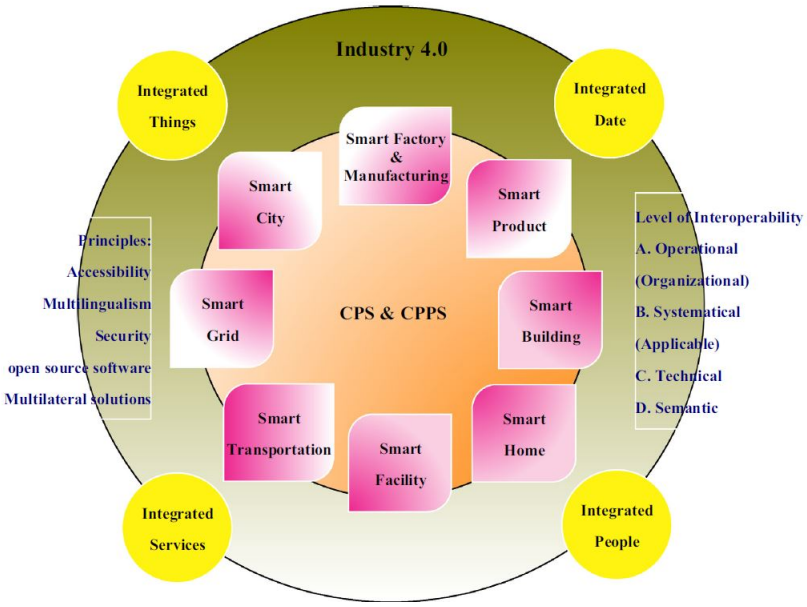

Figure 2.3: Framework of interoperability in Industry 4.0 [16]

Operational or organizational interoperability encompasses general concepts, standards, or relationships within Cyber-Physical Systems and Industry 4.0. The

guidelines, methodologies, domains, and models are identified by systematic interoperability. Technical interoperability links tools and structuring platforms, IT systems and related software. Finally, semantic interoperability is responsible for exchanging data between different groups of people, malicious application packages and companies. This structure is the foundation for Industry 4.0 and CPS, which helps in profit and effectiveness.

2.6 Network Communication Protocols

A network is a group of computers, servers, peripherals, or other devices linked to each other to enable data sharing, exchange files, or allow electronic communications. There are many types of networks in use, such as LANs, WLANs, Intranet, Internet, and others.

LANs or Local Area Networks are usually limited in a geographic area, like an office or a college. On the other hand, WLANs are used to exchange data via the air. WLANs don't need to be connected to a cable, this gives mobility to the connection and enables it to increase productivity. WiFi is included in the WLAN protocols. But, compared to the Ethernet, WiFi is slower and more susceptible to attacks, therefore it is preferred to use Ethernet on this project, due to its data transmission speed, lower electrical noise generation and security.

The Intranet is a private network within an organization that conforms to the same standards as the Internet, accessible only by members of the organization, employees or third parties with access authorization. The main purpose of an intranet is to share information about the organization and computing resources among users. Among the LAN technologies, the most popular LAN technology today is the Ethernet. Systems that use Ethernet communication divide data streams into packets or frames. The frames include source and destination address

information, as well as mechanisms used to detect errors in transmitted data and relay requests.

As elucidated in [17], Ethernet has a good balance between speed, cost, and ease of installation. These benefits, coupled with broad market acceptance and the ability to support virtually all popular network protocols, make Ethernet an ideal networking technology for the clients.

In an Ethernet communication protocol, packets are sent from sender to receiver, there are two well-known protocols to send such packets and achieve the desired communication. One is UDP or User Datagram Protocol, the other is TCP or Transmission Control Protocol. The main difference between the two protocols is that TCP has reliability, has an error checking mechanism and confirms that the receiver received the packet, while UDP, although unreliable, is easier and faster to complete the transmission[18].

Transmission Control Protocol/Internet Protocol or TCP/IP are internet protocols that enable interconnection between devices. They specify how data is exchanged, providing end-to-end connections that identify how it should be packaged, addressed, transmitted, routed, and received at the destination.

Some examples of internet transmissions that allow devices to connect and communicate are Unicast, Broadcast, and Multicast. The term "cast" means that a packet flow is being transmitted to the recipient(s) from the sender(s) through the communication channel. Unicast is the most common form of data transfer and consists of a one-to-one transmission, there should be a participation of a single sender and a single recipient, with determined IPs. Broadcast is a kind of one-to-all transmission that can be a data transfer in the destination address of the sender, called Limited Broadcast Address, or the data transfer can be to all the devices over the other network, referred to as Direct Broadcast Address in the datagram header for information transfer[19].

The communication used in this project is Multicast, a type of group communication in which the sender forwards a message to the Multicast group and the recipient, connecting to the same Multicast group, captures the message. Multicast can be a transmission of one/many senders to one/many recipients. Multicasting allows single direct copies of the data stream server to be simulated and routed to the hosts that request it. The delivery of the packages is made more safely [19]. Another LAN technology is Power over Ethernet (PoE), which enables network cables to carry electrical power. It has many benefits, such as saving time and cost, because it doesn't need another cable just for power, also flexibility, security, reliability, and scalability. The IEEE 802.3af standard regulates the concept of PoE. This standard describes that the power supply must be between 44Vdc and 57Vdc (48Vdc is the rated voltage) and the signal power must be a maximum of 15.4W [20].

2.7 Open Source Hardware Boards

The embedded system used in this project is an open-source hardware and software board with an ESP32 microcontroller. According to the Open Source Hardware Association, Open Source Hardware (OSHW) is a term used to establish that your project and design are released to the public so that anyone can make, modify, distribute or use the devices. This term covers only material, concrete devices such as machines or apparatus [21].

Anyone using an OSHW license to produce items should make it clear that the products are not manufactured, modified, sold, or warranted by the original designer, and must not use trademarks either. The term can also be expanded to Open Source Software (OSSW), in which a source code from a computer software is released under a license similar to the OSHW [21].

Some companies are benefiting from open source licensing and are developing boards of all types with numerous functions. These boards make life easier for researchers, college students, and other businesses as well.

2.8 Programming Software

Programming software is a program that helps the programmer develop other software. Examples of programming software are compilers, assemblers, debuggers, interpreters, etc. Integrated development environments (IDEs) are combinations of all these software.

An example of an IDE is Arduino, an open-source software that makes it easy to create firmwares and upload them to the boards. Arduino is so flexible and can be used for many kinds of boards, including Olimex boards. The Arduino compiler/IDE accepts both C and C++ languages.

Chapter 3

Materials and Methods

This project aims to develop a data acquisition system that comprises a set of modules, each being an embedded system that allows the connection of up to four temperature sensors and four pressure sensors. The system is in the context of the On-Surf project to be implemented in molds used for metal stamping and plastic injection.

The sensors may be connected to a signal conditioning IC, and this one connected to an ESP32 hardware board. Then, there is one I/O expander linked to each ESP32 board. These elements compose one module. Thence, there is data transmission between ESP32 and Raspberry PI via Ethernet.

3.1 Project Structure

The project structure, represented in Figure 3.1, is modular and scalable, with the possibility of being expanded or contracted according to the demand of sensors.

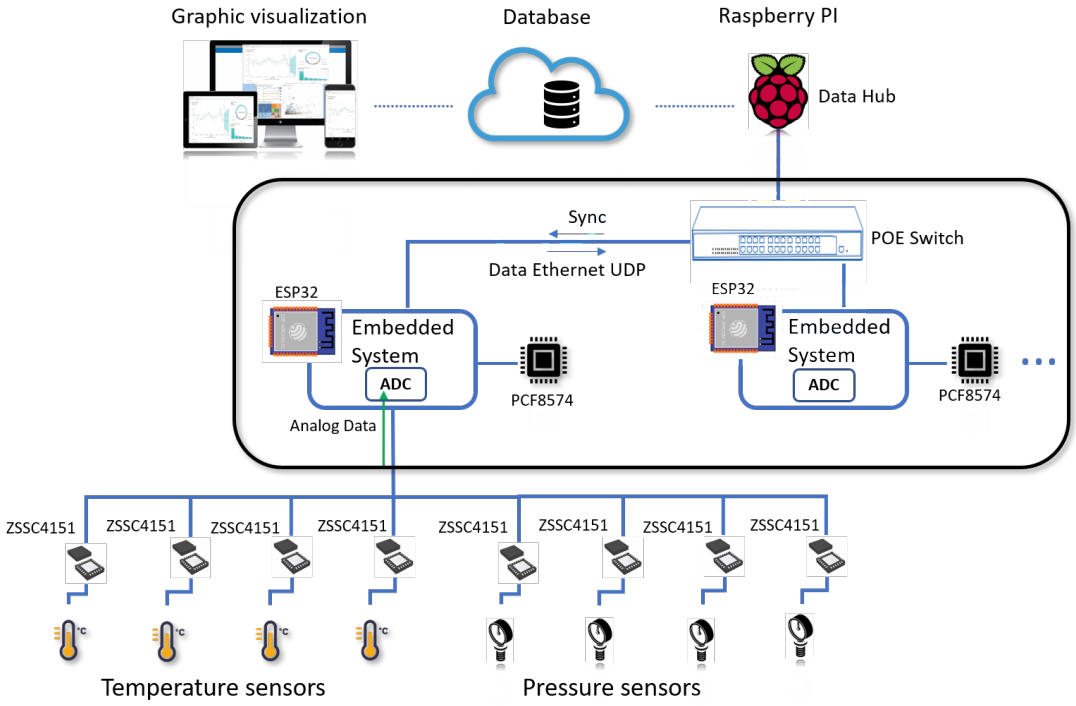

Figure 3.1: Project Structure

Each signal conditioner ZSSC4151 support only one sensor, therefore the system must have up to eight sensor signal conditioners connected to one embedded system, the ESP32 board. This connection is related to the ADC data acquisition on the board. Also linked to the ESP32 board is an I/O expander PCF8574 with the task of addressing an IP octet for each module. The data is transmitted from the ESP32 board to the Raspberry PI via Ethernet, mediated by a PoE switch for power and data transmission. The communication via Ethernet approaches the multicast protocol and the timestamp synchronization.

This thesis focuses mainly on the circled part of Figure 3.1, the embedded system with the ESP32 board and configurations, and the I/O expander. Next, each component of the proposed architecture will be detailed.

3.2 Sensor Signal Conditioner ZSSC4151

The ZSSC4151 Sensor Signal Conditioner (SSC) is an integrated circuit that contains high precision amplification and specific correction of bridge sensor signals. It has a digital compensation for sensor swap, sensitivity, temperature drift, and nonlinearity that may be obtained through an internal microcontroller [22].

The ZSSC4151 is adjustable for almost all types of bridge sensors, for example, the strain gauge. Measured values can be provided on the analog or digital voltage output. The specific sensor and ZSSC4151 can be digitally calibrated, causing noise to decrease. A block diagram of the device with its main characteristics is in Figure 3.2.

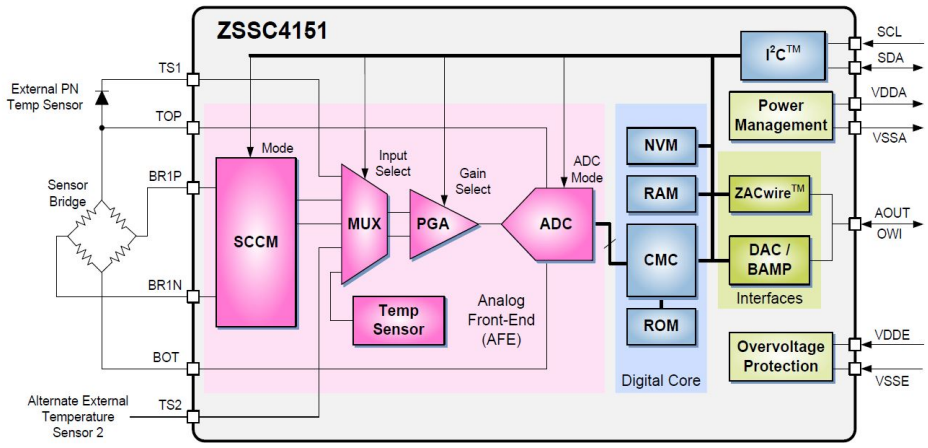

Figure 3.2: ZSSC4151 Block Diagram [22]

This signal conditioner was chosen because it has an I2C interface, which has the advantage of supporting multiple slaves, which meets the needs of the project. Besides, it can be reconfigured by the software provided by the manufacturer, has adjustable gain and easy access to RAM.

3.3 I/O Expander PCF8574

The PCF8574 device can provide a remote I/O expansion through the I2C interface. The I/O device does not have internal configuration, so, it reads or writes the device's I/O directly after sending the address. It can notify the microcontroller if there is incoming data on its ports by sending an interrupt signal [23].

The PCF8574 device has a unique address provided by the ports A0, A1, A2. It may operate as a switch to the ESP32 board. The expander is used as a slave to the ESP32 master to address a module, consisting of an ESP32 board and up to eight ZSSC4151. The device works as a switch to provide a 7-bit address through the I2C interface. This address is converted to decimal and defines one octet of the master's IP address.

3.4 Open Source board ESP32-PoE-ISO

The open-source board for this project is the Olimex ESP32-PoE-ISO, open-source hardware and software, manufactured by Olimex, a Bulgarian company that manufactures boards and develops various software.

The ESP32 microcontroller is good for Internet-of-Things applications with a processor up to 240MHz, 34 programmable GPIOs, 12bit SAR-ADCs with 18 channels, and supports I2C, I2S, SPI, and other peripherals.

The board has Power Over Ethernet, a 3000VDC galvanic insulation from Ethernet power, a UEXT connector, a battery connector, user and reset buttons. It has only 12 ADCs channels and considerably fewer GPIOs, but it fits well into this project. As programming software for the board, the Arduino IDE has open-source codes that can be easily found on the internet.

3.5 PoE Switch

The PoE is a network switch that supports power and data transmission over one Ethernet cable at the same time, which brings more flexibility, reliability, and cost-efficiency. With the importance of having two devices connected to the network all the time, a PoE switch is required, with lower infrastructure costs, a simplified installation, enhanced deployment flexibility, and centralized control.

3.6 Raspberry PI

The Raspberry PI is a small single-board computer. It runs Linux, a mouse and a keyboard can be connected to it, also a monitor. It provides a set of GPIO pins that can be used to control electronic components and explore IoT.

The model included in this project, the Raspberry PI 3 B+ has a CPU up to 1.4GHz, 17 GPIO pins and connector for external peripherals. The Raspberry PI functioning as a data hub receives data from the ESP32 board and sends it to a database.

3.7 Data Acquisition

The ESP32 ADC is used due to the signal provided by the ZSSC which is an analog signal, the use of the ZSSC is recommended in common mode, with the analog output.

The ESP32-PoE-ISO board has twelve ADC channels available, but it is required only eight sensors. The sampling frequency must be set to 1kHz because it was established as a requirement by the On-Surf project partner industries.

There is a specified library to use the ADC, the sensor pins are configured for input and the main function used to read analog values is Arduino's "analogRead

(analogPin)", with analogPin being the input pin where the sensor is connected.

The sensors designed for the project were not ready in time, so other sensors were used for tests, such as potentiometers, PT100 for temperature and others available in the laboratory.

3.8 Communication network via Ethernet

To communicate both ESP32 board and Raspberry PI, a LAN was created through Ethernet. To establish communication via Ethernet, messages are being sent over User Datagram Protocol (UDP), a protocol that establishes low-latency and loss tolerating connections between applications on the internet. UDP was chosen for the project because there is no need to guarantee the sending of the message. The task focuses mainly on sending multiple data packets with the fastest protocol.

Raspberry PI sends a time synchronization message to the ESP32 board to initialize the communication, receiving this message the board starts to send packets in response, with the acquired data and the timestamp.

Chapter 4

Development

This chapter describes every addressed methodology used in this project with details, relevant points as well as difficulties found on the way, and innovative solutions that helped to solve the problems.

Experimental tests were done in parts and a separate code was developed for each part, Ethernet connection, ADC configuration, I2C configuration, timestamp synchronization, and packet transmission.

The codes were tested separately and merged. It was used potentiometers as sensors for the tests. Finally, all previously tested codes were merged into one code (Appendix A).

4.1 Ethernet Communication

The use of Ethernet on this project is due to its data transmission speed, lower electrical noise generation, and security. To connect the ESP32 board to the Ethernet network, the software libraries "AsyncUDP.h" and "ETH.h" have been employed, both downloaded from Github in the ESP32 folder for Arduino. The library "ETH.h" is based on the WiFi library from Arduino, it enables the board

to connect to the Ethernet.

WiFi is turned off to avoid interference with the ADC2, it can only be used when the WiFi driver has not started. Then, the module's address is assigned to the board IP and establishes a connection with a multicast group. Connected to the group, once the board receives a packet, it starts sending packets via unicast. Multicast protocol was chosen to avoid sending unnecessary packets and to guarantee the synchronization of communication with several modules at the same time.

Packets are sent via UDP and have the data concatenated with other important information in the message. The connection to Ethernet can be seen in the following code 4.1.

```
WiFi.mode(WIFI_OFF);    //WiFi turned off

IPAddress ip(194,210,1,address_dec); //Board IP address
IPAddress gateway(194,210,1,1);
IPAddress subnet(255,255,0,0);
ETH.begin();
ETH.config(ip, gateway, subnet);

/*Multicast group*/
if(udp.listenMulticast(IPAddress(224,3,29,71), 10000))
{
   udp.onPacket([](AsyncUDPPacket packet)
   {
      remoteIP = packet.remoteIP();
      remotePort = packet.remotePort();
      run=1;
    });
 }
```

Listing 4.1: Connection to Ethernet

4.2 Analog-to-Digital conversion

The embedded system receives an analog signal from the signal conditioner, the ADC resolution is 12 bits, the sampling frequency is set at 1kHz and all the sensors are defined as inputs.

The function used to read analog values is "analogRead(Analog_pin)", the result is converted to a string type and sent to the data-hub. An example of how to read data from potentiometers is in code 4.2.

```
for(int i=0;i<8;i++){                    //8 sensors
    val[i]=analogRead(Analog_pin[i]);     //Read
    volt[i] = val[i]*(3.30/4095);         //Convert to Volts
    dtostrf(volt[i], 5, 2, datanow);      //Float to String
}
```

Listing 4.2: Reading analog values

4.3 Data Transmission

ESP32 sends two packets of messages, both to the same IP and port, one packet has the data currently acquired and the second has the old data, identical to the previously sent. This method is to ensure redundancy and prevent packet loss along the way.

In the redundancy method, the next packet to be sent always has the same data as the previous one, plus new data. So, if a packet is lost, the data from the lost packet will be in the next packet sent.

The interaction among components (i.e., ESP32 and Raspberry PI) is crucial, requiring that the components use the same knowledge representation [24]. In this way, in both packets, concatenated in the message, there is a data-related ID counter, counting the number of messages sent that helps keep track of the packets.

Between the data are used separators to facilitate the message visualization and to separate each data and timestamp. The timestamp sent is received from the data-hub to help synchronize the time in the ESP32 board with the Raspberry PI.

The messages have a lot of information attached and they have to be a string type because the Raspberry PI receives the messages in a string. The data ID and the timestamp are transmitted concatenated with the data in the code 4.3.

```
sprintf((char*)bufferData, "%lf", t1);    //Timestamp
sprintf((char*)IDdata, "%ld", countID++);
strcat((char*)bufferData, "$$");
strcat((char*)bufferData, (char*)IDdata);  //Data ID

AMessage.write((const uint8_t*)bufferData, strlen((const char
    *)bufferData));    //Write the message

udp2.sendTo(AMessage, remoteIP, remotePort); //Send the packet
```

Listing 4.3: Data currently acquired concatenated with data ID.

4.4 Timestamp Synchronization

To synchronize the times between the ESP32 board and the Raspberry PI, the chosen method was sending a timestamp message from the Raspberry PI to the ESP32, and the board would synchronize its time according to the timestamp received. This method was preferred because there is no guarantee that the system will be connected to the internet all the time and this eliminates the use of the Network Time Protocol (NTP) server. NTP allows devices connected to the network to synchronize clock times by receiving time from a server [25].

A Mutex is used to synchronize the timestamp and ensure that one action does not occur before another. In each action, there is a set time to wait for Mutex, if

the Mutex timeout expires, the action will not occur. With this set time, it has a possibility of the timestamp delaying in this time.

A Mutex is a mutually exclusive flag or a binary semaphore, including a priority inheritance mechanism, intending to synchronize threads. It acts as a gatekeeper to a section of the code blocking some threads and allowing just one at a time. The first block of the code attempting access has to set the Mutex, once that block is complete, the Mutex is released and the second block can have access and so on. The function of a Mutex is illustrated in Figure 4.1.

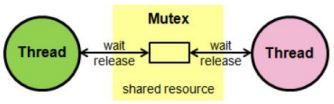

Figure 4.1: Function of a Mutex [26]

As soon as the ESP32 board receives the timestamp sync message, the message is checked bit by bit and saved to a variable timeSync. Then, it is used the function "esp_timer_get_time()" to get the time in microseconds since esp_timer_init is called, and saved to a variable t0. Finally, the Mutex is released. This mechanism is in the code 4.4

```
if(xSemaphoreTake(xSemaphore,(TickType_t)pdMS_TO_TICKS(500)) ==
    pdTRUE){                          //Wait for have the mutex
     mdata = (char*)packet.data();
     if(checkSringSync(mdata))  //Check the string
     {
          timeSync = atof(mdata);
     }
     t0 = esp_timer_get_time()/1000000.0;
     xSemaphoreGive( xSemaphore );    //Release the Mutex
}
```

Listing 4.4: Mutex used to synchronize the timestamp.

The new timestamp in code 4.5 is determined by the sum of the old timestamp with the result of the function "esp_time_get_time()" and the subtraction of t0.

```
    t1 = timeSync + esp_timer_get_time()/1000000.0 - t0;
```

Listing 4.5: Equation to obtain the timestamp

4.5 Module Address

The I2C communication is configured for the ESP32-PoE-ISO board to be the master and the I/O expander PCF8574 the slave. Each slave has a unique 7-bit address and works as a switch.

The code 4.6 for an I2C scanner is used to identify the PCF8574 address, once this address is explicit, it addresses one octet in the board's IP address.

```
void Scanner()
{
    for (byte i=8; i<120; i++)
    {
        Wire.beginTransmission (i);        //Begin I2C transmission
        if (Wire.endTransmission() == 0)  //Receive 0 = success
        {
           address_dec = (int)i;     //Module address -> IP octet
        }
    }
}
```

Listing 4.6: Code for an I2C scanner.

The pins for I2C communication in the ESP32 board are the GPIO 13 for SDA or serial data, and GPIO 16 for SCL or serial clock. It is needed to use the "Wire.h" library from Arduino, to start the transmission, the function "beginTransmission()" is used.

4.6 Code Fluxogram

To explain the code developed in a better way, a flowchart was set up, which is in figure 4.2, with the main parts of the code, divided between setup and loop.

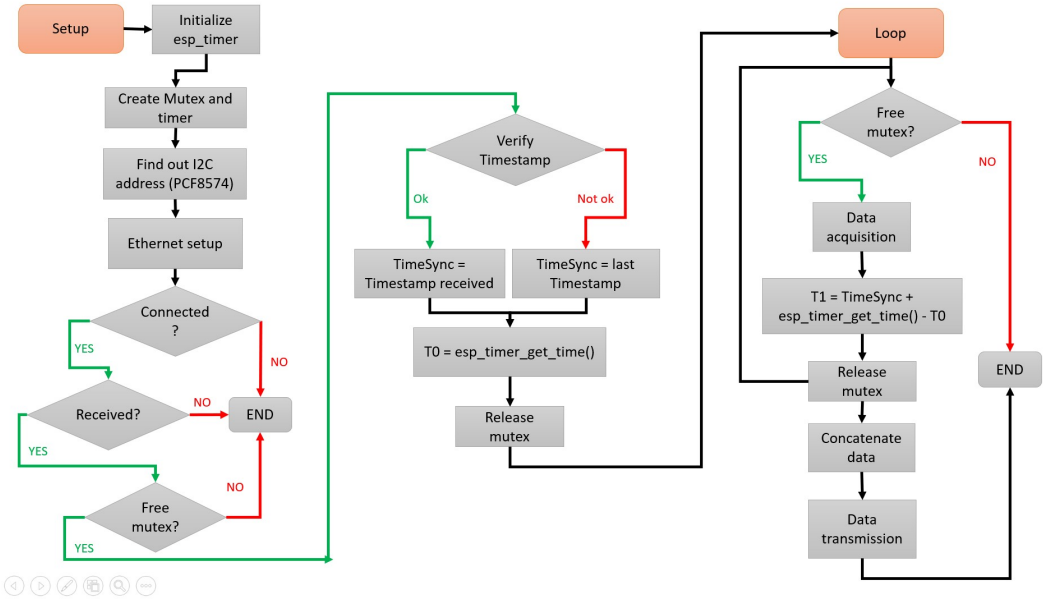

Figure 4.2: Code flowchart

Chapter 5

Results

This chapter presents the experimental tests performed to verify if the developed project meets the expected objectives and solves the proposed problem. It also addresses problems faced during the project development and the proposed solution.

5.1 Data Transmission

For data transmission, it is essential to ensure data sending and avoid packet loss. Monitoring baud rate and throughput are crucial. Also, the messages must contain all necessary information and timestamp.

To perform data transmission tests, Wireshark software was used to monitor the Ethernet network through which the ESP32 sends data. Wireshark is a widely used open-source network protocol analyzer that captures packet traffic on the network and analyzes them accurately. It is used by many companies, government agencies, and educational institutions [27].

For data transmission tests, were used an ESP32-PoE-ISO board, a PCF8574 as a switch, and the UDP stream was tested with the ESP32 board sending data

to a computer via Ethernet. The UDP filter has been applied to Wireshark tests because only packets transmitted via UDP matter. The setup used is in figure 5.1

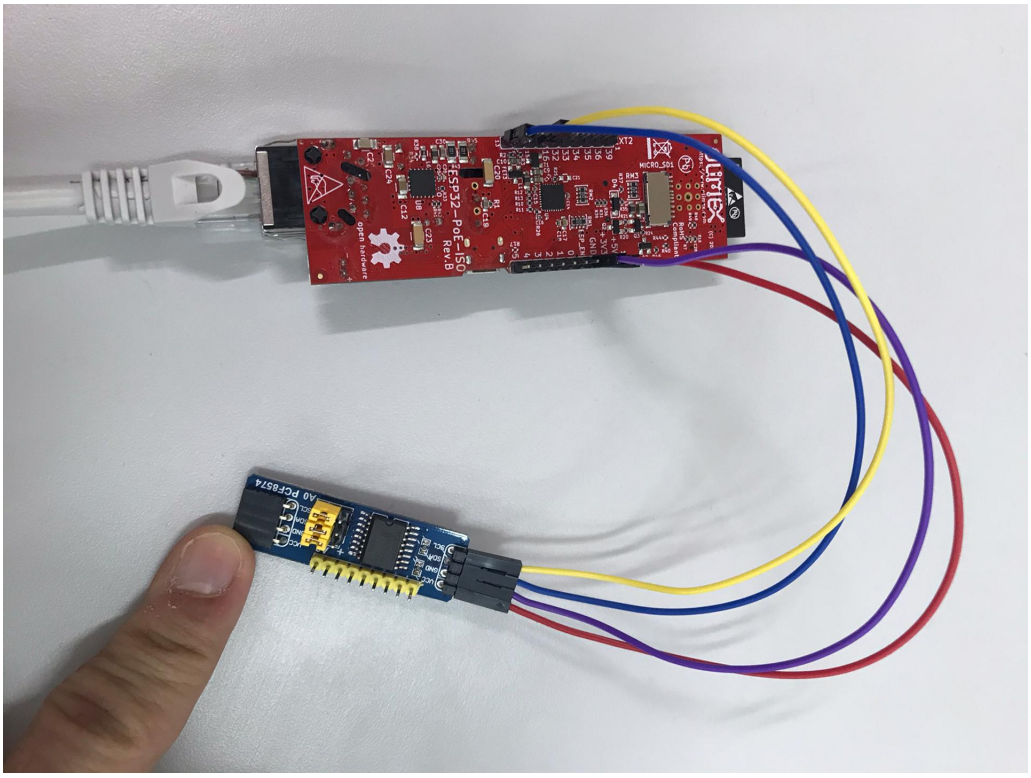

Figure 5.1: ESP32 and PCF8574 for data transmission tests

5.1.1 Statistics

There is a UDP stream analysis in Figure 5.2. Statistics show the number of packets, the average packet per second (pps), the time span, the average packet size in bytes, the number of bytes, the average bytes per second, and the bits per second (bps).

Measurement	Captured
Packets	4201723
Time span, s	2100.059
Average pps	2000.8
Average packet size, B	124
Bytes	519733731
Average bytes/s	247 k
Average bits/s	1979 k

Figure 5.2: Statistics from Wireshark test

Analyzing these statistics, the packet rate per second is defined as "network throughput" and is the message transmission rate on a communication channel. The average bit per second is also called bandwidth and is the amount of data that can be transmitted from one point to another over the network over a specific time. Baud rate refers to what information is transmitted over a communication channel, if the information unit is equal to one bit, baud rate and bandwidth are the same [28].

Analyzing bandwidth and network throughput statistics, bandwidth is the maximum amount of data that can be transmitted through a communication channel, while throughput is the amount of data that passes through the channel successfully [29]. A comparison can be made between throughput and bandwidth, 2000.8 pps with 124 bytes each is 248,099.2 bytes/s in total. Compared with the average 247k bytes/s, the value reaches 99.6%.

The average network throughput in packets per second is represented in the I/O graph from Wireshark in Figure 5.3. The axis X is time in seconds, and the axis Y is packets per second. The maximum value obtained for the throughput is 2221 pps at time 1974 seconds.

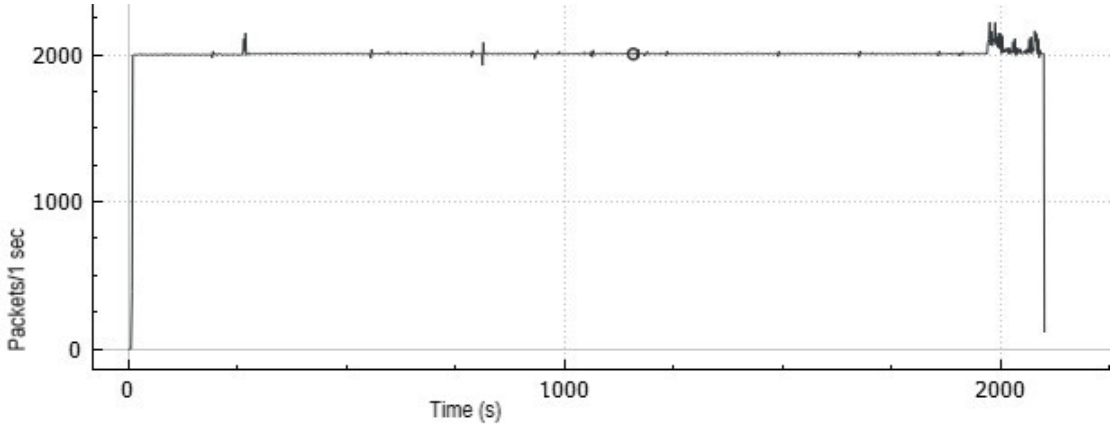

Figure 5.3: I/O Graph from Wireshark

The test results presented are satisfactory and show a well-designed architecture with the ability to achieve the proposed objectives. However, it should be borne in mind that testing with more sensors, more ESP32 boards is still needed and thus ensuring a modular and scalable system.

5.1.2 Message composition

Sent messages are composed of the acquired data from the sensors, the timestamp, and a data ID. There are always two identical messages, the current message and the previous one, to ensure redundancy in sending data.

Figure 5.4 shows a capture of six messages received by the recipient, the message composition can be verified as described, also the redundancy. In this test, the results from the ADC are simulated because the system was not tested with actual sensors. The information highlighted in Figure 5.4 is the timestamp between dollar signs in blue and the data ID in yellow.

```
3.30$$ 3.30$$ 0.49$$ 0.24$$ 3.30$$ 1.69$$ 0.80$$ 3.30$$1578580078.161059$$3
3.30$$ 3.30$$ 0.49$$ 0.24$$ 3.30$$ 1.69$$ 0.80$$ 3.30$$1578580078.162046$$3
3.30$$ 3.30$$ 0.45$$ 0.21$$ 3.30$$ 1.08$$ 0.87$$ 3.30$$1578580078.162046$$4
3.30$$ 3.30$$ 0.45$$ 0.21$$ 3.30$$ 1.08$$ 0.87$$ 3.30$$1578580078.163062$$4
3.30$$ 3.30$$ 0.38$$ 0.18$$ 3.30$$ 0.00$$ 0.94$$ 3.30$$1578580078.163062$$5
3.30$$ 3.30$$ 0.38$$ 0.18$$ 3.30$$ 0.00$$ 0.94$$ 3.30$$1578580078.164050$$5
```

Figure 5.4: Message composition from Wireshark

The module address is defined in the last octet of the IP address by the PCF8574 address. Figure 5.5 was extracted from the recipient and shows three portions of messages with the IP address of the board and the module address in green, which is 32.

```
$$1578580659.738864$$581371" from ('194.210.1.32', 49153)
$$1578580659.739856$$581371" from ('194.210.1.32', 49153)
$$1578580659.739856$$581372" from ('194.210.1.32', 49153)
```

Figure 5.5: Message with the module address

The message contains all the necessary information, is properly written to be sent and well-concentrated in the database. Message redundancy is guaranteed by sending two identical messages to retrieve information from the previous package, in case it gets lost.

5.2 Data Acquisition

To perform data acquisition tests, values were acquired from two potentiometers connected to the ESP32 ADC. The ADC capture resolution chosen was 12 bits and the working voltage 3.3V. The full-scale digital value is 4095, so the step value is 3.3 divided by 4095. To compare only analog values, the solution is to multiply the step by the acquired binary value.

The voltages resulting from the data acquisition tests show a slight variation in the data obtained by the ADC, as shown in Figure 5.6. The second decimal place varies between a range of four values, which is not expected. This variation

is ignored because the digits are not considered significant for temperature and pressure values on thin surface molds.

```
1.61##ID1$$ 1.77##ID2$$
1.61##ID1$$ 1.76##ID2$$
1.61##ID1$$ 1.76##ID2$$
1.61##ID1$$ 1.74##ID2$$
1.61##ID1$$ 1.74##ID2$$
1.62##ID1$$ 1.75##ID2$$
1.62##ID1$$ 1.75##ID2$$
1.61##ID1$$ 1.76##ID2$$
```

Figure 5.6: Data obtained from sensors

The temperature values recorded in a mold factory, partner of the On-Surf project, were approximately 70^oC, while the pressure values tend to change according to the material flow tension, however, they are all in order of megapascals (MPa). Thus, a resolution of 1^oC for temperature and 1MPa for pressure can be considered sufficient for the application. Therefore, it is concluded that the results of the data acquisition tests are considered adequate since the variation is not significant.

However, the system must be modified correctly according to the sensors used for acquisition. Still, to be tested with eight sensors and eight signal conditioners, the code must also be modified and the equations made in the ESP32 itself for better visualization of the data.

Chapter 6

Conclusions and future work

With the completion of the project, the completion of the objectives was verified. A scalable and modular system was developed, divided into modules with the ability to be expanded or contracted as needed. The system can temporarily identify data samples using timestamp synchronization. The data acquisition was performed by the embedded system, as well as data processing. Finally, the data transmission to the Raspberry PI data hub came with satisfactory throughput and bandwidth; then, the data hub receives this information and makes it available in a remote access database.

For future work, the three works developed within the project On-Surf should be joined, the data acquisition must be modified to test with the sensors developed for the project, also calibration of the system to minimize measurement errors. Graphic visualization should be planned and developed, and testing should be performed during metal stamping and plastic injection processes. Another work that is being developed under the On-Surf project is the construction of virtual sensors, to facilitate analyses without the need for the actual sensor.

Bibliography

[1] B. T. Y, B. S. S, S. S. Bobade, and T. Y. Badgujar, "a State of Art in a Sheet Metal Stamping Forming Technology - an Overview," *International Journal of Advance Research and Innovative Ideas in Education*, no. 3, pp. 3760–3770, 2017.

[2] D. Y. Yang, M. Bambach, J. Cao, J. R. Duflou, P. Groche, T. Kuboki, A. Sterzing, A. E. Tekkaya, and C. W. Lee, "Flexibility in metal forming," *CIRP Annals*, vol. 67, no. 2, pp. 743–765, 2018, ISSN: 17260604. DOI: 10.1016/j.cirp.2018.05.004.

[3] N. Rodrigues, P. Leitão, and E. Oliveira, "Dynamic Composition of Service Oriented Multi-agent System in Self-organized Environments," in *Proceedings of the 2014 Workshop on Intelligent Agents and Technologies for Socially Interconnected Systems - IAT4SIS '14*, 10.1145/2655985.265599: ACM Press, 2014, pp. 1–6, ISBN: 9781450328906. DOI: 10.1145/2655985.2655990. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2655985.2655990.

[4] H. Hagenah, R. Schulte, M. Vogel, J. Hermann, H. Scharrer, M. Lechner, and M. Merklein, "4.0 in Metal Forming – Questions and Challenges," *Procedia CIRP*, vol. 79, pp. 649–654, 2019, ISSN: 22128271. DOI: 10.1016/j.procir.

2019.02.055. [Online]. Available: https://doi.org/10.1016/j.procir.2019.02.055.

[5] N. W. R. Tocci and G. Moss, *Digital systems: principles and applications*. Pearson, 2017, vol. 12.

[6] D. D. A. Blog. (2017). Types of data acquisition systems, [Online]. Available: https://daqifi.com/blog/types-of-data-acquisition-systems/.

[7] D. A. Systems. (2019). Data acquisition systems, [Online]. Available: https://www.dataacquisitionsystems.com/.

[8] E. Garage, *Different type of sensors*, https://www.engineersgarage.com/article_page/sensors-different-types-of-sensors/, [Online; accessed 26-December-2019].

[9] Beamex, *PT100 Temperature sensor*, https://blog.beamex.com/pt100-temperature-sensor, [Online; accessed 26-December-2019].

[10] Omega, *Strain Gauge*, https://www.omega.co.uk/prodinfo/StrainGauges.html, [Online; accessed 26-December-2019].

[11] N. instruments, "The Engineer ' s Guide to Signal Conditioning," [Online]. Available: https://download.ni.com/evaluation/signal%7B%5C_%7Dconditioning/20712%7B%5C_%7DBenefits%7B%5C_%7Dof%7B%5C_%7DIntegrated%7B%5C_%7DSC%7B%5C_%7DWP%7B%5C_%7DHL.pdf.

[12] N. Rodrigues, E. Oliveira, and P. Leitao, "Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems," *Computers in Industry*, vol. 101, pp. 81–90, Oct. 2018, ISSN: 01663615. DOI: 10.1016/j.compind.2018.06.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361517306991%20https://linkinghub.elsevier.com/retrieve/pii/S0166361517306991.

[13] A. Schilling, "Toward a General Modular Systems Theory and Its Application To Interfirm," *Management*, vol. 25, no. 2, pp. 312–334, 2008.

[14] A. B. Bondi, "Characteristics of scalability and their impact on performance," pp. 195–203, 2004. DOI: 10.1145/350391.350432.

[15] N. Shahid and S. Aneja, "Internet of Things: Vision, application areas and research challenges," *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, vol. 10, no. 7, pp. 583–587, 2017, ISSN: 1570-8705. DOI: 10.1109/I-SMAC.2017.8058246. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2012.02.016.

[16] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017, ISSN: 2452414X. DOI: 10.1016/j.jii.2017.04.005. [Online]. Available: http://dx.doi.org/10.1016/j.jii.2017.04.005.

[17] I. Lantronix, *Ethernet Tutorial Networking*, https://www.lantronix.com/resources/networking-tutorials/ethernet-tutorial-networking-basics/, [Online; accessed 16-December-2019].

[18] C. Hoffmann, *What's the Difference Between TCP and UDP?* https://www.howtogeek.com/190014/htg-explains-what-is-the-difference-between-tcp-and-udp/, [Online; accessed 27-December-2019].

[19] G. F. Geeks, *Difference between Unicast, Broadcast and Multicast in Computer Network*, https://www.geeksforgeeks.org/difference-between-unicast-broadcast-and-multicast-in-computer-network/, [Online; accessed 27-December-2019].

[20] IEEE, *802.3af-2003 - ieeee standard for information technology*, https://standards.ieee.org/standard/802_3af-2003.html, [Online; accessed 05-January-2020].

[21] O. S. H. Association, *Definition*, https://www.oshwa.org/definition/, [Online; accessed 27-December-2019].

[22] I. Integrated Device Technology, "Automotive Sensor Signal Conditioner with Analog Output, ZSSC4151 Datasheet," *Datasheet*, p. 2, 2016.

[23] T. Instruments, "PCF8574 Remote 8-Bit I/O Expander for I2C Bus," *Datasheet*, p. 11, 2015.

[24] P. Leitao, N. Rodrigues, C. Turrin, A. Pagani, and P. Petrali, "GRACE ontology inteGrating pRocess and quAlity Control," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 4348–4353. DOI: 10.1109/IECON.2012.6389189.

[25] WhatIs, *What is network time protocol*, https://searchnetworking.techtarget.com/definition/Network-Time-Protocol, [Online; accessed 06-January-2020].

[26] I. D. D. Mbed, *Mutex*, https://os.mbed.com/docs/mbed-os/v5.9/reference/mutex.html, [Online; accessed 29-December-2019].

[27] Wireshark, *About*, https://www.wireshark.org/, [Online; accessed 06-January-2020].

[28] V. K.Garg and Yih-ChenWang, *The Electrical Engineering Handbook*. Academic Press, 2005, vol. 1.

[29] T. Keary, *Throughput vs bandwidth: Understanding the difference plus tools*, https://www.comparitech.com/net-admin/throughput-vs-bandwidth/, [Online; accessed 06-January-2020].

Appendix A

Developed Code

```
#include "Arduino.h"
#include "ETH.h"
#include "AsyncUDP.h"
#include "esp_timer.h"
#include "driver/adc.h"
#include "Wire.h"

AsyncUDP udp;
AsyncUDP udp2;
volatile static bool eth_connected = false;

int Analog_pin[] = {36, 39, 32, 33, 34, 35, 4, 0};  //GPIOs
float val[8];
float volt[8];

int run = 0;
IPAddress remoteIP;
uint16_t remotePort;

double timeSync;
```

```
double t0;
double t1=0.0;
double oldTS=0.0;

char * mdata=NULL;

int address_dec;

static void periodic_timer_callback(void* arg);

SemaphoreHandle_t xSemaphore = NULL;

int checkSringSync(char *p){  //Check the timestamp syntax
   received from master  xxxxxxxxxx.yyyyyy
  int i;
  for(i=0; i<10; i++) {
    if(isdigit(p[i]==0))
    return 0;
  }
  if(p[10]!='.')
    return 0;
  for(i=11; i<=16; i++) {
    if(isdigit(p[i]==0))
    return 0;
  }
  p[17] = 0;
  return 1;
}

void Scanner()
{
  for (byte i=8; i<120; i++)
  {
```

```
    Wire.beginTransmission (i);        //Begin I2C transmission
    if (Wire.endTransmission() == 0)  //Receive 0 = success
    {
      address_dec = (int)i;
    }
  }
}

void setup() {
 esp_timer_init();
 xSemaphore = xSemaphoreCreateMutex();
 if( xSemaphore != NULL )
 {
    //The semaphore was created.
 }

 const esp_timer_create_args_t periodic_timer_args = {
    .callback = &periodic_timer_callback};

 esp_timer_handle_t periodic_timer;
 ESP_ERROR_CHECK(esp_timer_create(&periodic_timer_args, &
    periodic_timer));

 /* Start the timers */
 ESP_ERROR_CHECK(esp_timer_start_periodic(periodic_timer, 1000));
      //Define the sampling frequency in microseconds

 Serial.begin(115200);
 Wire.begin(13,16);
 WiFi.setAutoConnect(false);
 WiFi.setAutoReconnect(false);
 WiFi.disconnect();
```

```
 Scanner();

 WiFi.mode(WIFI_OFF);
 btStop();

 IPAddress ip(194,210,1,address_dec); //Address PCF
 IPAddress gateway(194,210,1,1);
 IPAddress subnet(255,255,0,0);
 ETH.begin();
 ETH.config(ip, gateway, subnet);

 for(int i=0;i<8;i++) {
     pinMode(Analog_pin[i], INPUT);
 }
 if(udp.listenMulticast(IPAddress(224,3,29,71), 10000)) {
    udp.onPacket([](AsyncUDPPacket packet) {
      remoteIP = packet.remoteIP();
      remotePort = packet.remotePort();
      if(xSemaphoreTake(xSemaphore,(TickType_t)pdMS_TO_TICKS(500))
          == pdTRUE){ //Wait until 500ms for have the mutex
        oldTS = timeSync;
        mdata = (char*)packet.data();
        if(checkSringSync(mdata)){   //Check the string
            timeSync = atof(mdata);
        }
        t0 = esp_timer_get_time()/1000000.0;
        xSemaphoreGive( xSemaphore ); //Release the mutex
      }
      run=1;
    });
 }
}
bool sendFlag = false;
```

```
static void periodic_timer_callback(void* arg){ //Callback fired
    within a period of 5ms
  sendFlag = true;
}
int64_t countID=1;
int64_t countID2=0;
uint8_t bufferData[2000];
uint8_t bufferData2[2000];
uint8_t IDdata[500];
uint8_t IDdata2[500];
static char datanow[2000];
static char datapast[2000];
void loop() {
  AsyncUDPMessage AMessage;
  AsyncUDPMessage AMessage2;
  if(run==1) {
    if(sendFlag) {
      if( xSemaphoreTake( xSemaphore, ( TickType_t ) pdMS_TO_TICKS
          (500) ) == pdTRUE ) { //Wait until 500ms for have the
          mutex

        for(int i=0;i<8;i++) {                //Read sensors
            dtostrf(volt[i], 5, 2, datapast);
            strcat((char*)datapast, "$$");

            AMessage2.write((const uint8_t*)datapast, strlen((
                const char*)datapast));   //Sending old data

            val[i]=analogRead(Analog_pin[i]);
            volt[i] = val[i]*(3.30/4095);
            dtostrf(volt[i], 5, 2, datanow);     //Float to String
            strcat((char*)datanow, "$$");
```

```
            AMessage.write((const uint8_t*)datanow, strlen((const
                char*)datanow));      //Write the message
          }
        t1 = timeSync + esp_timer_get_time()/1000000.0 - t0;
        xSemaphoreGive( xSemaphore );  //Release the mutex
       }

     sprintf((char*)bufferData, "%lf", t1);   //Timestamp
     sprintf((char*)IDdata, "%ld", countID++);
     strcat((char*)bufferData, "$$");
     strcat((char*)bufferData, (char*)IDdata);  //Data ID

     sprintf((char*)bufferData2, "%lf", t1);   //Timestamp
     sprintf((char*)IDdata2, "%ld", countID2++);
     strcat((char*)bufferData2, "$$");
     strcat((char*)bufferData2, (char*)IDdata2);

     AMessage.write((const uint8_t*)bufferData, strlen((const char
        *)bufferData));   //Write message
     AMessage2.write((const uint8_t*)bufferData2, strlen((const
        char*)bufferData2));

     udp2.sendTo(AMessage2, remoteIP, remotePort);
     udp2.sendTo(AMessage, remoteIP, remotePort); //Send message

     bufferData[0] = 0;
     bufferData2[0] = 0;
     sendFlag = false;
    }
  }
}
```