Politecnico di Torino

Master's Degree Course in Mechatronic Engineering

Use of actuators for advanced HMI in the aerospace sector

Candidate:

Fabio Filannino

Supervisors:

Prof. Marcello Chiaberge

Eng. Mario Nigra

Eng. Massimo Tarro Genta

Academic year 2023/2024

Index

1 Introduction **1**

1.1 Context and motivation . . . 1

1.2 Objectives of the thesis . . . 2

1.3 Structure of the thesis . . . 2

2 State of the art **4**

2.1 Senses and sensory perceptions . . . 4

2.1.1 The sense of touch in aerospace applications . . . 5

2.2 HMI . . . 6

2.2.1 Evolution of HMIs . . . 6

2.3 Wearable devices . . . 8

2.3.1 Haptic HMIs . . . 8

2.4 Previous studies and existing solutions . . . 9

2.4.1 Haptic feedback gloves . . . 10

2.4.2 Smart jackets with actuators . . . 11

3 Theory of actuators **15**

3.1 Sensors and actuators . . . 15

3.2 Operating principles of actuators . . . 17

3.2.1 Haptic feedback actuators . . . 17

3.2.2 Thermal actuators . . . 19

3.2.3 Multisensory actuators . . . 20

3.3 Selection of actuators . . . 21

3.3.1 Vibration actuators . . . 21

3.3.2 Peltier modules . . . 21

3.3.3 Heating elements . . . 23

3.3.4 Support actuators . . . 23

4 Design of the feedback system **27**

4.1 System specifications and design . . . 28

4.1.1 Functional requirements . . . 28

4.1.2 Hardware components 28
4.1.3 Software integration 32
4.2 Implementation of the feedback system with the flight simulator 33
4.2.1 System architecture overview 33
4.2.2 Algorithm to filter parameters 34
4.2.3 Raspberry Pi algorithm 40
4.2.4 Relays control 44
4.2.5 LED strip control 46
4.2.6 LED matrix control 50

5 Simulation and experimentation **54**
5.1 Experimental validation 54
5.2 Performance evaluation 56
5.2.1 Responce time and latency 56
5.2.2 Accuracy of feedback 57
5.2.3 User feedback 59
5.3 System limitations 60
5.3.1 Feedback delay 60
5.3.2 Limited feedback customization 61
5.3.3 System scalability 61
5.3.4 Flight simulator limitations 62

6 Conclusions **63**
6.1 Summary 63
6.2 Future work and developments 64
6.2.1 Wireless connectivity and wearability 65
6.2.2 Machine Learning for personalized feedback 66
6.3 Final thoughts 66

Bibliography **68**

List of Figures

Figure 2.1: Early analog instruments in aircraft cockpits. 7
Figure 2.2: FinGAR. 11
Figure 2.3: Sense-Roid. 12
Figure 2.4: Teslasuit. 13
Figure 2.5: Body–conforming tactile jacket. 13
Figure 2.6: EmoJacket. 14

Figure 3.1: Relation between sensor, controller and actuator. 16
Figure 3.2: ERM motors. 18
Figure 3.3: Peltier module. 19
Figure 3.4: Performance curve of Peltier module. 22
Figure 3.5: Adhesive heating plate. 23
Figure 3.6: Small wearable LEDs. 24
Figure 3.7: LED matrix. 25
Figure 3.8: LED strip. 26

Figure 4.1: Wearable ERMs. 29
Figure 4.2: ATX alimentation. 30
Figure 4.3: Photos of the soldering done to connect all the components. 30
Figure 4.4: Switch used to connect and protect all the cables. 31
Figure 4.5: Raspberry Pi connected to the two relays and the switch. 31
Figure 4.6: MQTT network. 32
Figure 4.7: Parameters chosen. 34
Figure 4.8: Configurable file. 42

Figure 5.1: Flight simulator. 55
Figure 5.2: LED strip in no roll situation. 58
Figure 5.3: LED strip in turn sitation. 58
Figure 5.4: LED matrix in action. 59

Chapter 1

Introduction

1.1 Context and motivation

The aerospace sector stands as a pinnacle of technological advancement, driving progress in transportation, defense, and space exploration. Continuous innovation in the sector ensures safety and efficiency, while also improving user experience and operational effectiveness. The aerospace industry is demanding the latest technology to design its systems, and with strong safety regulations that require high reliability.
Human-Machine Interfaces (HMI) in aerospace have advanced from mechanical based systems toward advanced digital based systems. Current and updated HMI designs are very relevant in operation of the aircraft because they offer important data to pilots and operators besides being control interfaces. Thus, despite the progress that has been made, the existing HMI systems have issues like latency, accuracy, and users' feedback that can affect the systems' performance as well as safety.

The motivation for this study arises from the need to address specific limitations of existing HMI systems. Therefore, the application of actuators and their integration into HMI systems could improve performance, safety, and user experience in aerospace operations where everything is potentially decisive.
The main idea of this research is to manufacture innovative and portable electro-mechanical actuating components and to establish an effective feedback circuit. Nowadays the use of new inverse transducers is restricted only to the field of simulation, in real cockpits only displays are used, but the development and evolution of advanced HMI and integrated systems suggests that research such as the one conducted in this thesis work could be the future of the aerospace sector.

1.2 Objectives of the thesis

The primary objective of this thesis is the study and implementation of actuators to provide sensory feedback to aircraft pilots. The feedback aims to convey to the pilot particular conditions or states in which the aircraft is located, implementing a new HMI communication channel.

The actuators used in this research are categorized into two main types: wearable actuators and support actuators.
Wearable actuators are designed to be worn by the pilot, providing direct tactile feedback. They can communicate critical information about the aircraft's state through vibrations, pressure, or other haptic signals. This direct feedback mechanism aims to enhance the pilot's situational awareness and response time to various flight conditions.
Support actuators, such as LEDs, serve as visual indicators within the cockpit. These actuators can display information about the aircraft's status and give directional cues. By using a combination of visual signals and patterns, these actuators aim to provide clear and immediate communication of essential data and instructions to the pilot.

The scope of this thesis also includes the development of a feedback system that is configurable and editable. The system will be controlled by a Raspberry Pi which will have the purpose of turning on certain actuators depending on the state of the aircraft.
This system will be designed to allow customization of the feedback, enabling adjustments according to the specific needs of different flight scenarios or preferences. It should be user-friendly, allowing pilots or technicians to easily modify the feedback parameters.

1.3 Structure of the thesis

The first thing done for the thesis work was to provide a comprehensive review of the existing literature and current research related to human senses and perceptions, wearable devices and HMI systems in aerospace. It identifies the gaps in current research that this thesis aims to address.
The next step was to delve into the fundamental principles of actuation, as well as the choice of inverse transducers that would be used. Following this, the design process of the feedback system has been outlined. in particular, the technologies and methodologies used to develop the feedback system, starting from the power supply of the devices up to their connection and control, through a Raspberry Pi and an MQTT network.

After that, a long activity of experimentation, collection and analysis of the results was carried out, also considering the limitations of the system. In the end, the final chapter summarizes the main findings of the research. It discusses the implications of the work for the aerospace industry, what the scenarios and use cases might be, and suggests potential directions for future research, highlighting areas where further investigation could continue to improve HMI systems.

Chapter 2

State of the art

The integration of electro-mechanical actuators (EMAs) in advanced HMIs for the aerospace sector represents a significant technological evolution.
This chapter delves into the current state of research, development, and application, starting with a brief review of the human senses and sensory perceptions. In particular, we will focus on the advantages of using the sense of touch to communicate compared to the more common sense of sight and hearing.
Afterwards, we explored into the topic of the human-machine interface and its evolution, up to the advanced, intelligent and wearable devices designed and existing in the real world.

2.1 Senses and sensory perceptions

One of the main functions of the human mind is to know and enter into relationship with the surrounding environment, and to do this it uses the sense organs, which allow us to perceive reality.
Understanding human senses and sensory perceptions is fundamental to developing effective systems in the aerospace industry. In aerospace applications, the senses involved mainly include sight and hearing, but also senses such as touch and proprioception (the sense of body position and movement) have been the focus of research in recent years. Each of these senses plays a crucial role in how pilots interact and respond to their environment and the aircraft's systems.

Vision is the primary sense for pilots, it provides critical information about the environment, including the horizon, other aircraft, and potential hazards. Modern cockpit displays utilize high-resolution screens, Head-Up Displays (HUDs), and Enhanced Vision

Systems (EVS) to present vital flight data and environmental information. These systems enhance situational awareness, especially in low-visibility conditions, by projecting real-time data onto the pilot's field of view.
Hearing plays a vital role in communication and alerting pilots to critical situations. The cockpit environment is equipped with advanced audio systems, including 3D audio and noise-canceling technologies, which ensure clear communication between pilots and air traffic control, as well as within the cockpit crew. Auditory alerts and warnings are designed to be distinctive and easily recognizable, enabling quick response to potential issues.

2.1.1 The sense of touch in aerospace applications

The sense of touch is a critical sensory modality that could provide direct and immediate feedback from physical interactions. In aerospace applications, leveraging the tactile perception through haptic feedback can significantly enhance pilot performance and safety. It could create a more intuitive and natural interaction, especially beneficial in scenarios requiring hands-on engagement with machines or systems.
Touch-based feedback is inherently intuitive. Humans are naturally adept at using touch to interact with their environment from an early age. This familiarity makes haptic interfaces easy to learn and use, reducing the cognitive load on pilots.
One of the most significant advantages of touch over other senses is the speed of perception. The human skin can detect and differentiate between two separate touch events within about 5 milliseconds. In contrast, the visual system typically requires up to 25 milliseconds to perceive and process changes. This faster perception speed means that haptic feedback can provide almost instantaneous alerts and cues, allowing for quicker response times in critical situations. In high-stakes environments like aviation, this rapid response capability can be crucial for avoiding accidents and maintaining control.
Furthermore, cockpits are information-dense environments, often requiring pilots to process numerous visual and auditory cues simultaneously. This can lead to sensory overload, where the ability to process important information is compromised. Integrating touch-based feedback helps distribute the sensory load more evenly, allowing pilots to use their sense of touch to receive critical information without overwhelming their visual and auditory channels. This multisensory approach can enhance situational awareness and improve overall performance.

Haptic feedback, or the use of touch-based sensations to convey information, is increasingly being used in aerospace applications not just through controls and interfaces, but also directly on the pilot's body. This approach can significantly enhance situational awareness and improve response times by providing intuitive directional cues.

Wearable haptic devices, such as vests, belts, and bands, are designed to deliver tactile feedback to specific areas of the body. These devices can produce a variety of sensations, including vibrations, pulses, pressures and hot or cold feedback, to communicate directional information. For instance, a vibration on the left side of a pilot's body can indicate a necessary turn or attention in that direction. In emergency situations where quick and decisive action is required, haptic feedback can provide unambiguous directional cues that cut through the noise of visual and auditory alerts. For instance, if a pilot needs to make an evasive maneuver to avoid a collision, a strong vibration on the appropriate side of their body can provide a clear and immediate indication of the necessary action. This can significantly reduce reaction times and improve safety outcomes.
Haptic feedback can also be particularly useful in cases where the pilot is distracted and does not see or hear other information being communicated. Pilots may already be too busy due to the high volume of information communicated to them via sight or hearing, or they may miss critical alerts during high-stress situations. In such cases, a tactile alert can immediately capture the pilot's attention and convey important information. This ensures that critical warnings are noticed and acted upon promptly, even if other sensory channels are compromised.

2.2 HMI

In this section we will comprehensively address the aspect of controlling the avionics system and its components by the pilot, as well as data visualization. This part of avionics is defined as the Human-Machine Interface (HMI) and has become increasingly important as the complexity of avionics systems has grown. HMIs play a crucial role in the aerospace sector, acting as the bridge between pilots and aircraft systems. Effective HMIs are designed to facilitate seamless interaction, enhance situational awareness, and reduce the cognitive workload on pilots.

2.2.1 Evolution of HMIs

The design of HMIs in aviation has evolved significantly over the decades. Early cockpits were equipped with analog gauges and mechanical instruments, requiring pilots to manually interpret data and make adjustments. For example, since the performance of an aircraft, indeed its very sustenance, depends on the airspeed, it was soon clear that it was necessary to have an instrument on board that provided this information. The other instruments that first appeared on the scene were the inclinometer, the tachometer and the compass.

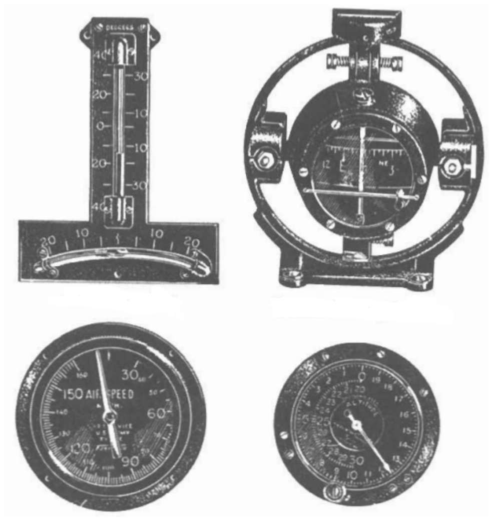


Figure 2.1: Early analog instruments in aircraft cockpits.

As the complexity of the avionics systems increased, the number of information to be presented to the pilot also increased and consequently also the number of instruments to be installed on the dashboard, until the available space was completely saturated. He soon became aware of the fact that a pilot did not need all the information he had available at the same time; indeed, he used some of it only in very specific, and limited, flight phases. The focus then shifted towards developing tools that were no longer dedicated to a single piece of information but could present the pilot with the data required at a specific moment. This led to the development of Multi-Function Displays (MFDs).
Subsequently, there arose a need to ensure that the pilot did not have to look away from the flight path to check the dashboard instruments. The only solution was to project flight parameters and navigation information onto a transparent screen placed in front of the pilot's eyes. Thus, the Head-Up Display (HUD) was born.

In recent years, the focus has been on integrating more advanced technologies to create more intuitive and responsive HMIs. The adoption of touchscreens has made interactions more direct and user-friendly, while voice recognition systems allow pilots to issue commands without taking their hands off the controls. These advancements have contributed to making cockpits more ergonomic and reducing the cognitive load on pilots.

2.3 Wearable devices

The increasing complexity and cost of aerospace systems necessitate sophisticated HMI solutions that facilitate interaction between humans, computer systems, and mechanical components. These advanced HMIs leverage innovative technologies that go beyond traditional screens and touchscreen displays, incorporating augmented reality (AR) and virtual reality (VR) to provide voice, visual, and tactile information.
Wearable devices, such as smartwatches, smart bands, AR headsets, exoskeletons, and 3D gesture scanners, are increasingly being integrated into HMI systems. These devices are designed to support pilots in their tasks, communicate with them and improve the flight experience. They provide innovative and bidirectional information exchange, increase safety in the cockpit and the efficiency of communications with the surrounding environment. Unlike traditional large and rigid electronic devices, human–machine interface systems incorporating flexible and wearable components represent an inevitable trend for the future. Developing efficient strategies for designing the material microstructures of wearable sensors and electronic devices with superior mechanical flexibility and stretchability is crucial for achieving effective, intuitive, and control of new HMI channels.

2.3.1 Haptic HMIs

Haptic human-machine interfaces (HHMI) combine tactile sensations and feedback to enable close interaction between humans and machine, providing immersive experiences and innovative flight modes. Significant progress has been made as new materials have been developed with better flexibility, durability, and sensitivity, allowing for the creation of more comfortable and reliable wearable.
HHMIs are composed of several essential components that work together to provide tactile feedback and ensure effective interaction between users and machines:

- Wearable sensors: these sensors detect physical and electrophysiological stimuli with high accuracy. Modern advancements have improved their softness, functionality, reliability, and selectivity, making them suitable for continuous wear and real-time monitoring;

- Soft actuators: Soft actuating systems provide high-quality haptic feedback by precisely controlling force, displacement, frequency, and spatial resolution. These actuators can mimic various tactile sensations, such as vibrations, pressure, and texture, to convey information or alerts to the user;

- Control systems: The integration of advanced control systems allows for the precise manipulation of sensors and actuators. These systems ensure that the haptic feedback is timely and accurately corresponds to the user's actions or the machine's status.

Artificial intelligence and machine learning algorithms are also integrated into HHMIs to improve their responsiveness and adaptability. These technologies can predict user actions, optimize feedback mechanisms, and enhance overall system performance.
HHMIs could have a wide range of applications in aerospace. For instance, haptic feedback can alert pilots to critical flight parameters or environmental changes without requiring them to look away from their primary tasks. Moreover, it could also be used in training or simulation, increasing immersion and realism, providing feedback that mimics real-word sensations, as we will see later in the course of this thesis work.

2.4 Previous studies and existing solutions

In recent years, the aerospace sector has seen a growing interest in integrating wearable devices, they have been increasingly utilized in the aerospace industry to support various functions such as health monitoring, navigation, and situational awareness. These devices provide real-time data acquisition and feedback, helping pilots and crew members to maintain optimal performance under demanding conditions.
One notable example is the development of smart jackets and other wearable health monitors designed to track vital signs and physiological parameters. These devices can monitor heart rate, respiration, and stress levels, providing crucial information that can help prevent fatigue and enhance pilot decision-making. For instance, the "LifeVest" is a wearable defibrillator that continuously monitors the user's heart activity and can deliver a treatment shock if a life-threatening rhythm is detected.
Augmented Reality (AR) and Virtual Reality (VR) headsets are also becoming increasingly common in the aerospace field, especially in training. These headsets can overlay critical flight information directly into the pilot's field of view, reducing the need to glance down at instrument panels.

A Dutch university conducted an experiment on the use of active control devices to make flying safer and analyzed the main results of a haptic feedback evaluation.
International aviation safety organizations, such as the European Union Aviation Safety Agency (EASA) and the International Air Transport Association (IATA), recognize loss of control in flight as one of the main risk factors responsible of most aviation fatalities. A significant factor contributing to this loss of control is insufficient monitoring of critical flight parameters and automation modes. To maintain and improve safety standards,

experiments were carried out using three types of sensory feedback.
The first concept used both force feedback and vibro-tactile alerts, producing promising, but not conclusive, results. The second experiment used asymmetric vibrations to provide directional warning signals, which did not result in improved performance upon initial use, but did produce an improved learning rate for the task. The third system used force feedback to physically guide the pilot away from the limits of the flight envelope that could have led to a loss of control of the aircraft, which produced safety improvements since first use, but created addiction: the pilot's performance worsened immediately after the removal of this technique.
In summary, the experiments highlighted the potential of haptic feedback systems to enhance flight safety by providing more intuitive and immediate warnings to pilots. While force feedback and vibro-tactile alerts showed promise but lacked conclusive results, asymmetric vibrations improved learning rates over time. The use of force feedback proved effective in guiding pilots away from dangerous flight envelope limits, though it also led to dependency issues. These findings suggest that combining these haptic feedback techniques could offer a balanced approach to improving pilot performance and safety in aviation.

2.4.1 Haptic feedback gloves

One innovative solution is the use of haptic feedback gloves, which allow pilots to feel tactile sensations corresponding to their interactions with controls and instruments. These gloves can simulate the feeling of pressing buttons, turning dials, or even experiencing environmental conditions like turbulence.
"Finger Glove for Augmented Reality" (FinGAR), provides an example of a more advanced haptic technology that uses combined electrical and mechanical stimulation for selective activation of the sensory mechanoreceptors in skin, which results into tactile feedback points on virtual objects. A DC motor and a series of electrodes is used to achieve this tactile feedback. They form a network with three fingers (thumb, index finger and middle finger), that provide feedback through four different stimulation modalities: the DC motor generates high frequency vibrations and shear deformation across the entire finger, while electrodes allow pressure and low-frequency vibrations at high spatial resolution.

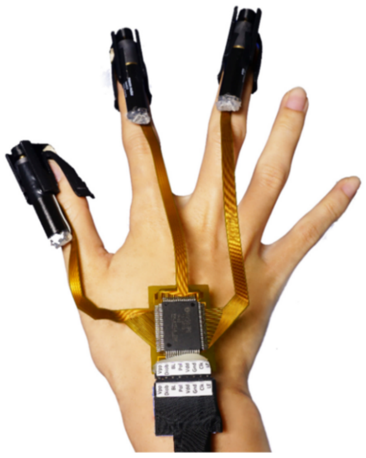


Figure 2.2: FinGAR.

More than 1000 people took part in the experiment, and most of them were surprised by the tactile feedback, some even said they had difficulty understanding whether that sensation was due to the glove or if they were actually touching something with their fingers. This wearable glove has potential application in assisting product design but also can be used for tactile feedback in teleoperation or augmented reality.

2.4.2 Smart jackets with actuators

Smart jackets equipped with actuators could represent an innovative approach in wearable technology, particularly in the aerospace sector. These jackets are designed to provide haptic feedback to pilots, by delivering tactile cues directly to the body.
This subsection reviews existing smart jacket technologies and presents some examples of them, although not yet used in aerospace, in fact nothing similar exists yet, but the final product imagined by this thesis work, which can be seen as a first step towards a new field of studies and research, aims to design a wearable and intelligent jacket that could revolutionize the way where pilots interact with their aircraft, providing them with an intuitive and immediate understanding of critical flight information through the sense of touch.

- Sense-Roid: this jacket, developed by a Japanese university, is made up of vibrators and artificial muscles to reflect a movement of the wearer and give it the sensation of wrapping and tightening it.
  They adopted some actuators, which are a type of artificial muscle, that expand in the direction of the diameter and contracts axially by sending air using an air compressor. These expansions and contractions compress the user's body, achieving a natural squeezing sensation, as shown in Figure 2.3.

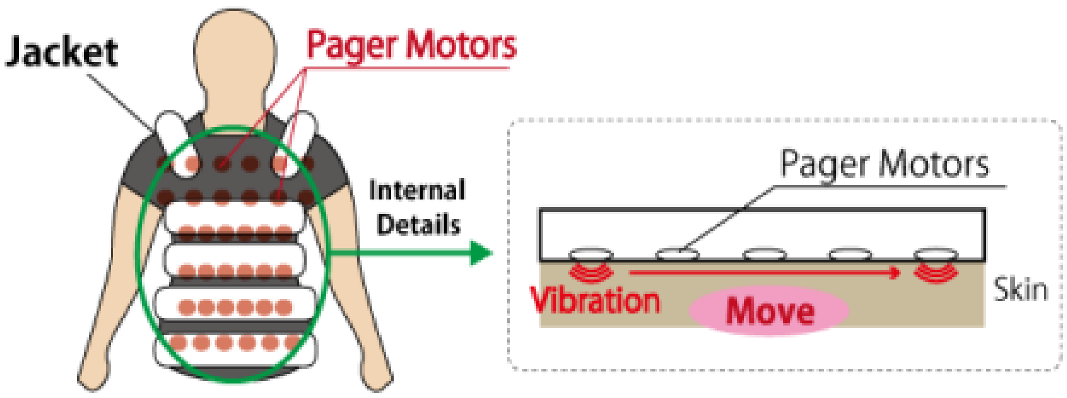


Figure 2.3: Sense-Roid.

Air flow to the actuator is controlled by a valve attached to servo motors. The motion of the servo motors is controlled by a microcomputer. This type of jacket could be used by the pilot to communicate particular conditions of the aircraft, obviously without exaggerating the sensation of squeezing.

- A smart jacket with portable thermoelectric cooling system: the jacket utilizes Peltier modules and a water circulation system to efficiently transfer heat. Cooling jackets are now widely used in hot environments to help control body temperature and enhance both physical and mental performance. By preventing body temperature from reaching dangerous levels, it could also avoid situations that would lead to dehydration and illness and it could help the pilot in extreme cases.

- Teslasuit: this feedback system utilizes electro-muscle stimulation (EMS) and transcutaneous electrical nerve stimulation (TENS) to recreate a variety of realistic sensations. Thanks this technology, the suit can deliver physical feedback corresponding to visual simulations through immersive reality devices.
  This wearable device can provide more immersive experiences, build muscle memory, autocorrect technique, and build deep learning environments. It exploits electrical stimulation to simulate a range of real-life feelings and sensations.
  At the 2019 Spanish and British Grand Prix it was used in the world's first multi-sensory driving experience offering an unprecedented level of immersion. Similarly it could also be used in a flight simulation.

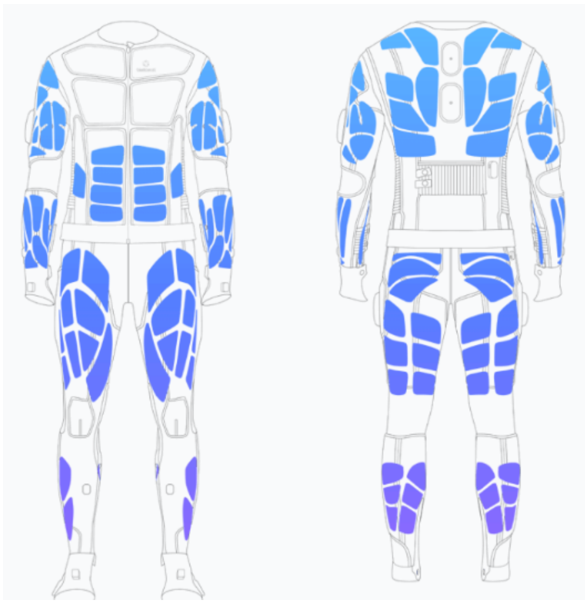


Figure 2.4: Teslasuit.

- A body–conforming tactile jacket to enrich movie viewing: a wearable tactile jacket that is used to deliver movie–specific tactile stimuli to the viewer's body that are specifically targeted to influence the viewer's emotions. In the same way it could also be used to create an HMI channel.
  This body–conforming jacket contains 64 tactile stimulators. For the actuators they have opted for pancake–shaped (coin type) generic excentric rotating–mass (ERM). This ones are glued on either of two types of custom–made PCB's. The connections between the motor PCBs and the driver PCB are made with thin flexible wires, as shown in Figure 2.5 where are presented the outer lining (left) and the jacked turned inside-out (right)

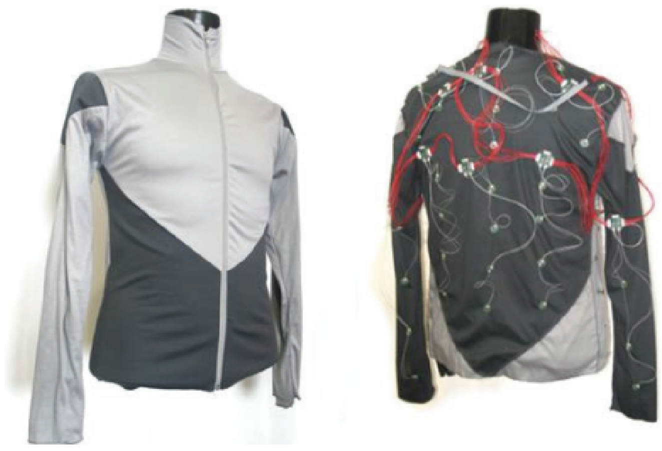


Figure 2.5: Body–conforming tactile jacket.

- EmoJacket: the jacket contains an embedded computing system and haptic actuators that are programmed to affect its user emotionally and improve their immersion in gaming and movie watching. The prototype is composed of 6 affective haptic components: chest and neck vibration, neck warmth, heartbeat simulation, arms vibration and shivering.
  This jacket is similar to the previous ones, in fact it combines the characteristics of the jacket with the Peltier modules and the one with ERM motors, using both vibrational actuators and thermoelectric coolers, distributing them along the sleeves and also at the height of the chest, shoulders and neck.
  In particular, we can note the use of temperature sensors in such a way as to always have the heat or cold sensation that is being transmitted under control.

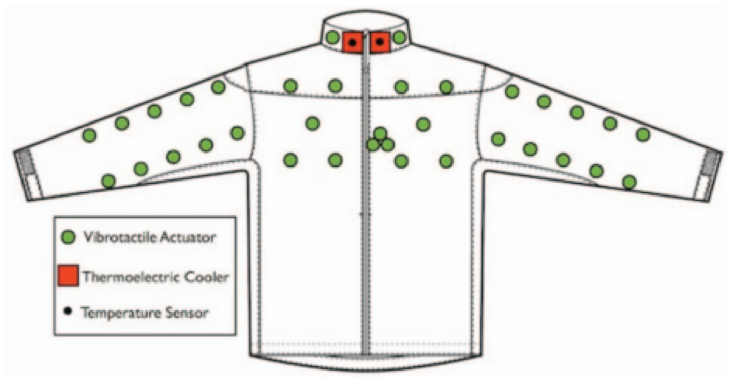


Figure 2.6: EmoJacket.

These smart jackets seen so far were taken as a reference for the planning and design of the wearable actuator system, especially the use of vibrational motors and Peltier cells. They will be used to communicate with the pilot and convey to him some conditions of the aircraft.

Chapter 3

Theory of actuators

Actuators are critical components in modern aerospace systems, playing a key role in the operation and control of various subsystems. In addition to traditional hydraulic or pneumatic actuators, electro-mechanical actuators (EMA) will be studied which convert electrical energy directly into mechanical movement, offering numerous advantages such as greater efficiency, reduced weight and greater precision. The interest in EMAs has rapidly grown over the years, especially considering the trend that the aerospace industry has taken toward sophisticated and more automated systems.

This chapter delves into the fundamental principles behind the operation of actuators. We will explore the fundamental concepts, such as the conversion of electrical energy into another form of energy or vice versa, the various types of actuators and their performance characteristics. Furthermore, we will examine the advantages and limitations of different designs, providing a complete understanding of their role and the choice of actuators that will be used for this thesis work.

3.1 Sensors and actuators

Sensors and actuators, and more generally transducers, are the electrical components that allow electronic systems to detect and act on physical parameters of daily life.
A sensor is defined as a device that transforms a physical quantity into an electrical quantity. Instead, an actuator is defined as a device that transforms an electrical quantity in a physical quantity.
A transducer is a more complex electrical or electronic device which contains a sensor plus an interface circuit. If instead of the sensor the system was composed of an actuator then it would be called an inverse transducer. In the following, however, these terms will

be used interchangeably to indicate a device that interfaces on one side with the physical system to be controlled and/or on the other with an electronic control system.

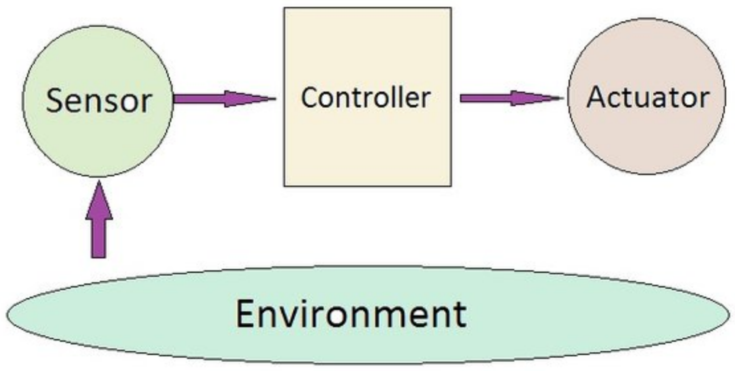


Figure 3.1: Relation between sensor, controller and actuator.

Sensors play a crucial role in electro-mechanical systems by detecting various physical parameters and converting them into electrical signals that can be processed. In aerospace applications, sensors are used to monitor a wide range of variables:

- Temperature sensors: Temperature sensors, such as thermocouples and resistance temperature detectors (RTDs), monitor the thermal conditions of critical components, including engines, avionics, and electrical systems. Proper temperature management is vital to prevent overheating, maintain efficiency, and ensure the longevity of the aircraft's systems;

- Position sensors: Position sensors, including potentiometers, encoders, and resolvers, are used to determine the exact position of moving parts within the aircraft, such as control surfaces or landing gear. These sensors provide feedback to the control system, enabling precise adjustments and maintaining the desired flight trajectory;

- Speed sensors: Speed sensors, including tachometers and accelerometers, measure the rotational or linear speed of components. These sensors are also critical for navigation;

- Force and pressure sensors: Force and pressure sensors detect the loads and pressures exerted on various components, such as hydraulic systems, fuel lines, and airframes. These sensors help in monitoring structural integrity and ensuring that the aircraft can endure the efforts encountered during flight.

Actuators are the components that convert electrical signals, received from the control systems, into a measurable physical quantity, such as motion, force, temperature, light, or sound. In this study, they will be used to communicate with the pilot, providing sensory feedback. They are essential in a wide range of applications where it is necessary to control or manipulate the physical world in response to input signals, in this case study they will come from the flight simulator, in a possible real and future application instead they will come from the aircraft's sensors, such as those described above.

3.2 Operating principles of actuators

Understanding the operating principles of different types of actuators is essential for selecting the appropriate actuator for specific tasks. This section will discuss the operating principles of various actuators, categorized by their types.

3.2.1 Haptic feedback actuators

This kind of actuators are devices that provide tactile or sensory responses to users through the sense of touch. The main types are now reported:

- Vibration: These actuators are used to deliver sensations of vibration to users.
    - Eccentric Rotating Mass (ERM) Motor: These are among the most common vibrational actuators and are often used in mobile phones, wearable devices, and game controllers. They operate by rotating an off-center weight, which generates a tangential vibration when the actuator is mounted on a surface. They are shown in Figure 3.2;
    - Linear Resonant Actuator (LRA): LRAs are more compact and efficient than ERMs, offering quicker response times. They are also used in mobile devices, such as smartphones and smartwatches, to provide more precise tactile vibrations and subtle sensations. An LRA is a vibration motor that generates an oscillating force along a single axis;
    - Vibration Disc Actuators (DRV): These vibrational actuators are designed to provide even vibrations across a wider surface area. They are used in applications requiring tactile feedback over a broad area, such as in driving simulator seats or automotive touchscreens.

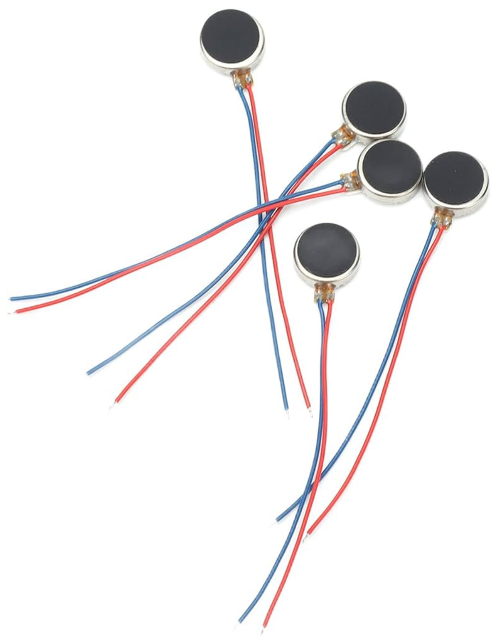


Figure 3.2: ERM motors.

- Piston-type actuators: These actuators can create physical movements, such as pushing or pulling an object. For example, in flight simulator joysticks, actuators can simulate push or pull effects during flight.
  They can be divided according to the fluid used to produce the movement. Hydraulic actuators, for example, use incompressible fluids, such as oil, and represent the best option in terms of power and are the ideal solution for moving heavy loads. Pneumatic actuators, on the other hand, use compressed air and are chosen mainly for their speed.

- Piezoelectric actuators: These are devices that exploit the piezoelectric properties of certain materials to generate movement or deformation when an electrical voltage is applied.
  Piezoelectricity is the property of some materials, such as quartz or some ceramics (e.g., PZT), to generate an electric charge in response to a mechanical stress, and vice versa. When an electric field is applied to these materials, they deform (expand or contract), producing mechanical motion.
  Piezoelectric actuators can produce extremely precise linear motion, often in the micron or submicron range, depending on the design and applied voltage. This precision makes them ideal for applications where high motion resolution is required.

3.2.2 Thermal actuators

Although thermoception could be considered in the sense of touch, it will be analyzed a type of devices which, unlike the previous ones, provide tactile feedback through the conversion of electrical energy into thermal energy, therefore they do not generate any mechanical movement. Here are some of the most popular ones:

- Heating elements: They are among the most common thermal actuators. They are found in home appliances too. Electrical energy is converted into heat through a resistive wire (such as nichrome) that heats the air or an object;
- Cartridge resistors: These are small cylindrical devices that generate heat and are used in applications requiring localized heating;
- Thermoelectric modules: They are devices that convert electrical energy into a temperature difference, thus generating both heat and cold. An example of these are Peltier cells.
  Peltier cells exploit a physical principle that takes its name from the physicist (Jean Charles Athanase Peltier) who discovered it.
  The Peltier effect is the inverse phenomenon of the Seebeck effect, which is in fact used in some sensors such as thermopiles and thermocouples. When an electric current flows through the junction connecting two dissimilar materials, it emits or absorbs heat per unit of time in the junction to balance the difference in chemical potential of the two materials. The voltage difference is then transformed into a temperature difference. A Peltier cell is composed of numerous thermoelectric elements connected in series electrically and in parallel thermally, inserted between two ceramic plates. Therefore, when there is a difference in voltage, one plate heats up and the other cools down.

Figure 3.3: Peltier module.

These modules are used for cooling electronic components or for creating small portable refrigerators. However, as seen in EmoJacket or the smart jacket with thermoelectric cooling system examples in paragraph 2.4.2, their application has also been studied for contact with human skin to provide sensory feedback.

3.2.3 Multisensory actuators

In this section we will talk about devices that provide feedback involving other senses, like sight and hearing. This type of actuator is undoubtedly the most widespread and sometimes they are not even considered as such, however they convert electrical energy into light and sound and are used for communication, especially in the aerospace sector, so they will be considered in this thesis project.

Visual actuators use the sense of sight and involve the use of light and visual signals to convey information. These actuators often use LEDs (Light-Emitting Diodes). LEDs have become ubiquitous in modern technology due to their energy efficiency, durability and versatility. They could be used to signal alerts, status changes, or warnings. For example, LEDs could change color to indicate different operating states or flash rapidly to attract attention during critical situations.
In more advanced systems, visual actuators can be part of augmented reality (AR) displays, where information is superimposed on the user's field of view, improving situational awareness without requiring them to look away from their task.

Auditory actuators utilize sound to communicate information, ranging from simple beeps and alarms to complex auditory signals or spoken instructions.
Auditory feedback is especially useful in environments where visual attention may be focused elsewhere. Sound can be used to provide immediate alerts, guide users through tasks, or confirm actions.

Though less common, olfactory actuators are designed to engage the sense of smell by releasing specific scents in response to certain triggers. However, the potential for olfactory actuators exists in scenarios where smell could serve as an additional alert mechanism or as part of a multisensory experience in training simulations.
For example, an olfactory actuator could be used in a training environment to simulate the smell of smoke or burning material, helping to create a more immersive and realistic experience during emergency drills.

3.3 Selection of actuators

The actuators chosen for this system were selected based on their ability to provide precise, responsive, multi-sensory feedback, meeting the specific needs of the application.
The actuators selected include vibration actuators, Peltier cells, heating plates and LEDs, each chosen for their unique characteristics and contribution to the overall functionality of the system.

3.3.1 Vibration actuators

Among the various types of haptic actuators available, the ERM vibration ones were selected. They are chosen for their simplicity, reliability, and ability to deliver strong, consistent vibrations. They are ideal for applications where clear, unmistakable feedback is required, such as alerting users to critical events or guiding them through a task.
LRAs would also be great and perhaps even better because they have faster response times, however unlike eccentric rotating mass motors, linear resonant actuators rely on AC voltage to drive the coil which is pressed against a moving mass attached to a spring. Although the frequency and amplitude of the linear resonant actuator can be adjusted by changing the AC input, the actuator must be driven at its resonant frequency to generate significant force, for this reason ERMs are simpler to use and power and were chosen for this project.
DRVs, on the other hand, were discarded because they are used to generate vibrations over larger surfaces.

The ERM used is a small coin vibration motor, 2 mm thick and 10 mm in diameter.
It is powered up to a voltage of 3V DC and is able to generate a good vibration, certainly perceptible even in a cockpit if in contact with the pilot.

3.3.2 Peltier modules

Peltier modules were chosen to provide sensations of cold. In fact, as written in the previous section, when they are powered, a temperature difference is generated between the two faces of the cell, one becomes cold and the other hot.
The great advantage of this device is the speed with which the temperature changes, in fact as soon as the current flows, you can immediately perceive the cold. Peltier elements offer precise control, allowing the system to quickly and accurately adjust the temperature based on user interaction or system requirements. This responsiveness is essential for

applications where immediate feedback is critical, such as in safety systems or interactive simulations

Furthermore, considering the space constraints due to the fact that they will have to be worn, it is essential to use small and easy to handle devices. Their small size allows them to be easily integrated into various form factors without significantly increasing the device's bulk or weight. This is particularly important in applications where portability and user comfort are priorities. The chosen device is 15 x 30 x 30mm in size, 4mm thick, so absolutely small and suitable.

For the project of this thesis, the device will be powered by a 3V power supply and, as can be seen in the diagram shown in Figure 3.4, it will be able to provide a temperature difference between the two faces of about 20/30 degrees, thus making immediately perceptible, both the hot plate and the cold one that will be placed in contact with the skin.

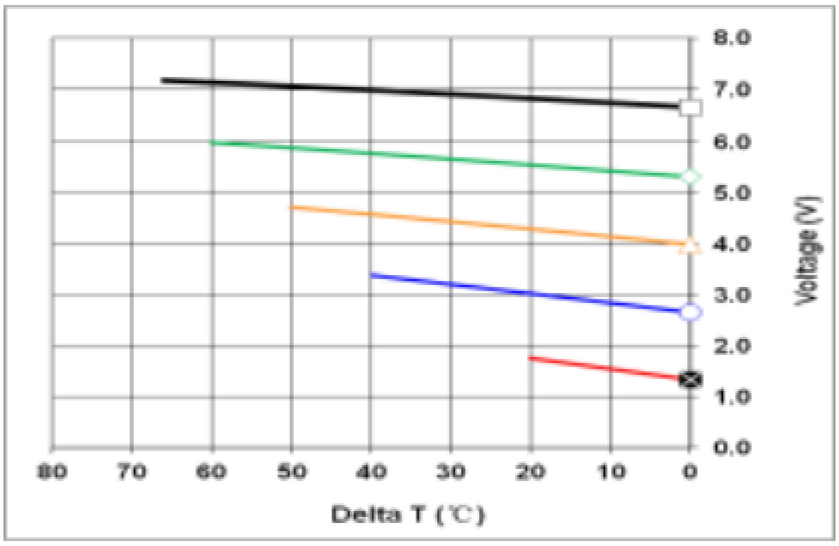


Figure 3.4: Performance curve of Peltier module.

It is important to note that the temperature difference between the two faces remains constant, but not the temperatures. In fact, if it is powered for long periods, both faces of the Peltier cell can reach high temperatures, while maintaining the differential.

For this reason, to generate a feeling of cold, a heat sink is usually used to keep the temperature of the cold cell constant. In this thesis work, for simplicity, the cell will be powered for a short duration, preventing the cold side from overheating.

3.3.3 Heating elements

Since the Peltier cell will be used exclusively to provide a sensation of cold, as for the actuator capable of providing heat, an adhesive heating plate was purchased.
Adhesive heating plates are designed to provide direct and uniform heat distribution across their surface. This characteristic is essential for applications where consistent and predictable thermal feedback is required. The even heat distribution ensures that users experience the intended temperature changes without hot spots or uneven warming, which could lead to discomfort or ineffective feedback.
Another advantage of these devices is their ease of integration into various devices and surfaces. Their adhesive backing allows them to be securely attached to a wide range of materials, including textiles, metals, and plastics. This flexibility in application makes them ideal for use in wearable technology, where they can be seamlessly incorporated into clothing or accessories without the need for complex installation procedures.
The chosen plate is rectangular in shape, measures 30 x 90 mm and can be powered up to 12 V. It can reach a rather high but not dangerous temperature in contact with the skin, resulting perceptible but not annoying.

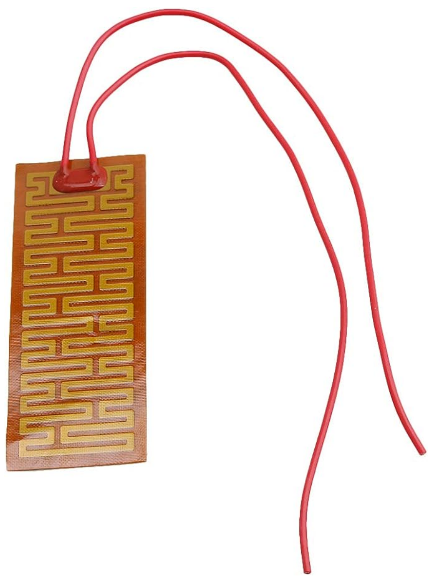


Figure 3.5: Adhesive heating plate.

3.3.4 Support actuators

In the context of improving safety in aerospace applications, especially in the cockpit environment, some actuators, which we can be called “support”, such as small wearable LEDs, LED matrix and LED strips play a crucial role. Although they may seem redundant given that there are already cockpit displays and lights that provide essential information to pilots, the addition of these support actuators provides an additional level of reliability and functionality, which is critical in complex and high-stress scenarios. These components not only increase safety, but also provide additional visual cues that can be instrumental in guiding the pilot's actions and improving situational awareness.

Small wearable LEDs are compact, lightweight, and can be integrated into various parts of the pilot's gear, such as gloves, helmets, or flight suits. These LEDs serve as an immediate and easily noticeable source of visual feedback, alerting the pilot to critical system statuses or operational conditions. For instance, wearable LEDs can be programmed to change color or blink in response to specific alerts, such as engine issues or navigational warnings.
The portability and proximity of these LEDs to the pilot ensure that alerts are impossible to miss, providing a personal and direct line of communication between the aircraft systems and the pilot. The strategic placement of these LEDs on wearable gear means that the pilot can receive alerts without needing to shift focus from their primary task. The LEDs' proximity to the pilot ensures that the alerts are not only visible but also impossible to ignore, providing an extra layer of safety.
This redundancy is particularly valuable in situations where cockpit displays might be obscured or if the pilot's attention is focused elsewhere.

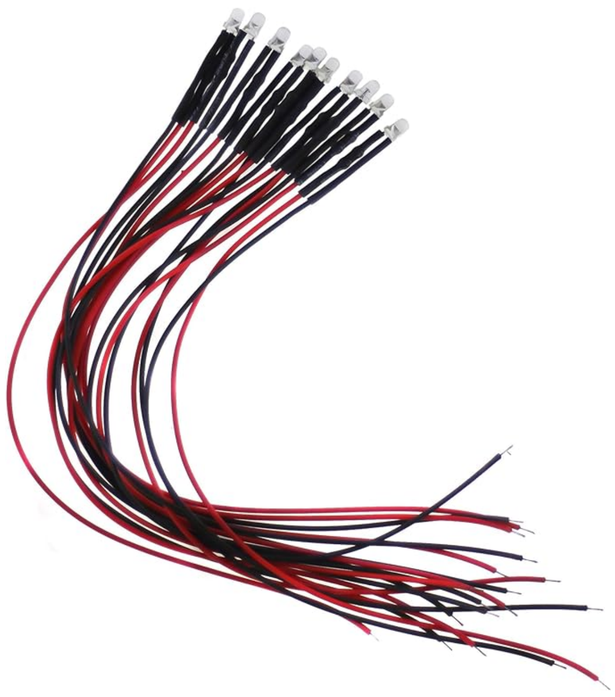


Figure 3.6: Small wearable LEDs.

An LED matrix is a grid of LEDs that can display various patterns, symbols, or even alphanumeric characters. In a cockpit setting, a LED matrix can be used as an auxiliary display to convey critical information visually, such as directional indicators, status alerts, or operational commands.
The flexibility of the LED matrix allows it to be programmed for different functions depending on the flight phase or specific situations. For example, during navigation, the matrix can display arrows or symbols to indicate the correct heading or alert the pilot to course deviations; in the event of an engine malfunction or fuel level anomaly, the matrix could flash symbols such as warning triangles or specific icons representing the affected systems.
For this thesis work, it was thought to use it to "draw" arrows to the left or right, making them flash. Furthermore, it is possible to vary both the color of these arrows and the speed with which they flash.

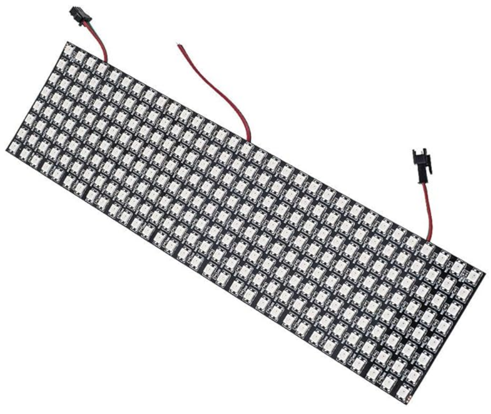


Figure 3.7: LED matrix.

LED strips provide a continuous line of light that can be installed in various parts of the cockpit to provide directional guidance or highlight specific areas. These strips can be programmed to illuminate in different colors or sequences, providing intuitive guidance to the pilot. For example, an LED strip positioned along the edge of the control panel can illuminate in the direction the pilot should be heading, providing a clear and immediate visual cue for navigation.
In addition to directional guidance, LED strips can act as status indicators, illuminating in specific colors to reflect the status of various systems (for example, green for normal operation, yellow for caution, red for critical warnings).

Figure 3.8: LED strip.

Using LED strips in the cockpit also improves visibility in low-light conditions, contributing to safer operations, especially during nighttime or adverse weather conditions.
This form of redundancy ensures that even if the primary displays fail or become difficult to read, the pilot can still rely on the LED strip for essential information.
The idea for this project is to use the strip around the Visual Display, by turning on the LEDs and sliding them towards a direction as needed.
An idea of how the LED strip could be used is presented in Figure 3.8, i.e. attached to a fixed and rigid part in the cockpit as in the image it is on a wall.

Chapter 4

Design of the feedback system

This chapter details the design and implementation of the feedback system, which integrates various actuators into a cohesive setup that enhances the pilot's interaction with the flight simulator.
Actuators such as ERMs, Peltier modules, heating plates and wearable LEDs are controlled by a Raspberry Pi via two relays that turn certain actuators on or off when certain conditions occur, in order to communicate with the pilot during the simulation.
The other actuators, the support ones, that are the LED matrix and the LED strip, are instead intelligent devices and therefore they are not simply turned on or off, but it is also established how to turn on, the color they must assume and the animation they must replicate. For this reason they cannot be connected to relays, which are instead simple switches, but directly to the Raspberry Pi.
The Raspberry Pi acts as a client in an MQTT network. It turns on the actuators depending on the message it receives from another client in the network, that is the flight simulator.
This chapter outlines the detailed approach taken to design, develop, and implement the entire system, from the requirements to the choice of hardware and software components used. The chapter also delves into the power supply considerations, detailing how the components are powered reliably to prevent any failure during operation.
The final section of the chapter focuses on the integration of the feedback system with a flight simulator. This involves setting up the system to continuously receive and monitor flight parameters, such as speed, altitude, or engine performance. When these parameters exceed predefined thresholds, the system triggers the corresponding actuators, such as haptic devices for tactile feedback or LEDs for visual alerts. In particular, it will be a focus on the algorithms used for the feedback system.

4.1 System specifications and design

This section delves into the detailed specifications and design choices that guide the development of the feedback system. Moreover, this section elaborates on the system's requirements, the selection of components, both hardware and software.

4.1.1 Functional requirements

The feedback system is designed to meet some specifications, the most important one is that it must provide feedback through multiple sensory channels (tactile, visual, and thermal) to ensure comprehensive situational awareness. Each sense is targeted to enhance different aspects of the pilot's interaction with the simulator, thereby improving reaction times and decision-making under stress. The pilot must already be prepared and know what information corresponds to a certain sensation, for example if he is given heat feedback, it could mean that the engine temperature has reached a critical level, indicating the need to reduce power or perform an engine cool-down.
Another important attribute that the system must have is to allow easy adjustments of the feedback settings. It must be configurable, it must be possible to vary which actuator to turn on, under what condition and for how long. Also, for the LED matrix and LED strip, you must require changes in intensity, frequency, color and animation.
The system must be designed to adapt to these changes with minimal reconfiguration and open to anyone who uses it, not just the designer. Furthermore, it must be scalable, allowing for future improvements or the addition of new actuators, if necessary.

The feedback system must integrate perfectly with the flight simulator. This includes acquiring real-time data from the simulator and immediately activating the appropriate feedback mechanisms without noticeable delays.

4.1.2 Hardware components

The selection of hardware components is a critical phase in the design process. Each component must meet the functional requirements and work harmoniously within the overall system architecture.

- Actuators: The devices chosen for this project have already been reported in the previous chapter. In particular, 8 vibrational actuators were used, divided on both arms. An example is shown in Figure 4.1.

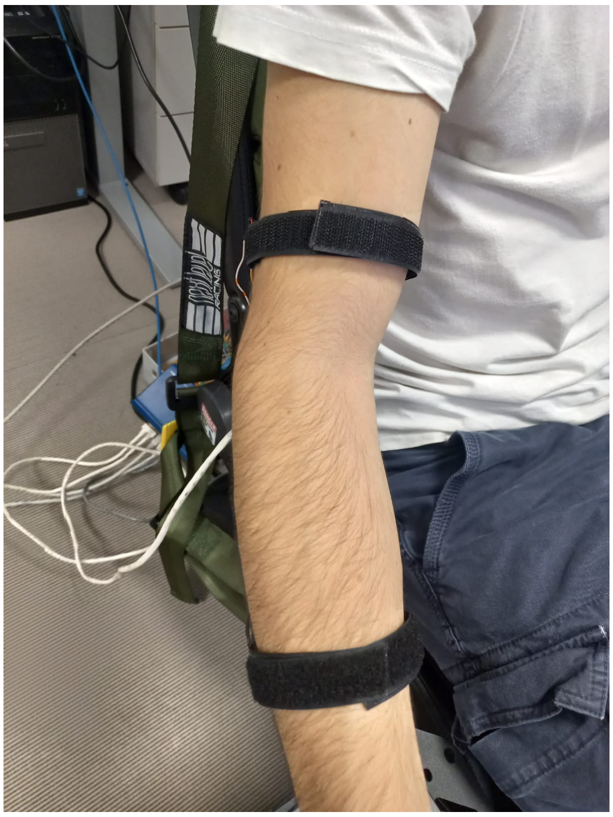


Figure 4.1: Wearable ERMs.

In addition, 2 Peltier cells and 2 heating plates that can be attached near the collar of a jacket or suit.
There are 4 LEDs too and they can be distributed on the pilot's body. As for the support actuators, the LED matrix was positioned near the screen that acts as the Primary Flight Display or Navigation Display.
The LED strip is positioned around the Visual Display that represents the external environment seen from the cockpit, that is, what a pilot would see through the windshield of the plane during flight.

- Control Unit: A microcontroller (e.g. Raspberry Pi) that processes input from the simulator and controls the actuators.
  However, between the simulator and the Raspberry Pi, a computer has been used that will work as a "filter". In fact, since the simulator will send a lot of parameters that will be updated in real time, leaving them all managed by the Raspberry Pi can be risky. For this reason, a PC has been added that receives the parameters from the simulator and, only if they pass certain conditions, will communicate them to the Raspberry Pi.

The actuators are connected via two relays in order to turn them on. They simply work as electromagnetic switches, operated by a relatively small electric current that can turn on or off a much larger electric current, in this case, the actuators.

- Power supply: To power the actuators that were used in this project, it was chosen to use an ATX power supply like the one in Figure 4.2.

Figure 4.2: ATX alimentation.

In addition, all the power cables were soldered and enclosed in an old switch case so that they could be connected simply using an RJ45 port and kept protected, as you can see in the next figures.

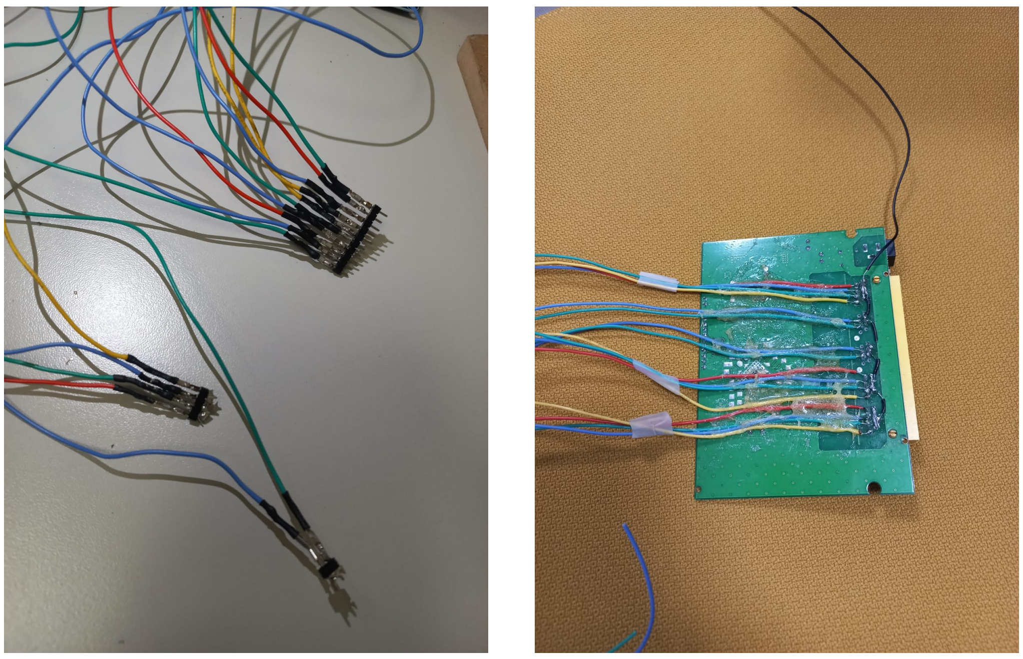


Figure 4.3: Photos of the soldering done to connect all the components.

Figure 4.4: Switch used to connect and protect all the cables.

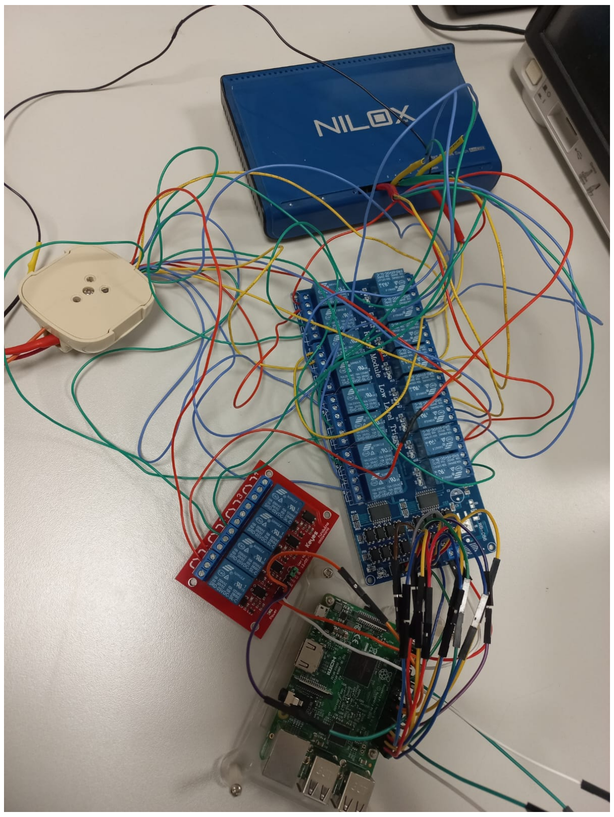


Figure 4.5: Raspberry Pi connected to the two relays and the switch.

4.1.3 Software integration

The software component of the feedback system is responsible for controlling the actuators in real-time, based on data received from the flight simulator.
The feedback system's software architecture is built on the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is an OASIS standard messaging protocol for the Internet of Things. It is designed as an extremely lightweight publish/subscribe messaging transport. It is particularly well-suited for applications where bandwidth is limited, latency is critical, or there are many devices (clients) that need to communicate. MQTT enables efficient and reliable communication between the flight simulator, the central PC (the "filter") and the Raspberry Pi that controls the actuators. The MQTT broker is the central hub that manages message distribution between devices. In this case, Mosquitto will be used as the broker. The broker receives messages from clients that publish them, and then it distributes these messages to the appropriate clients that have subscribed to those messages.
The clients can be any device or application that connects to the broker. Each client can act as a publisher, subscriber, or both. The messages that are published and received by each client are composed of a topic, which categorizes the messages, and a packet that contains the information to be exchanged.
The mqtt network of the system in question is represented by the following figure:

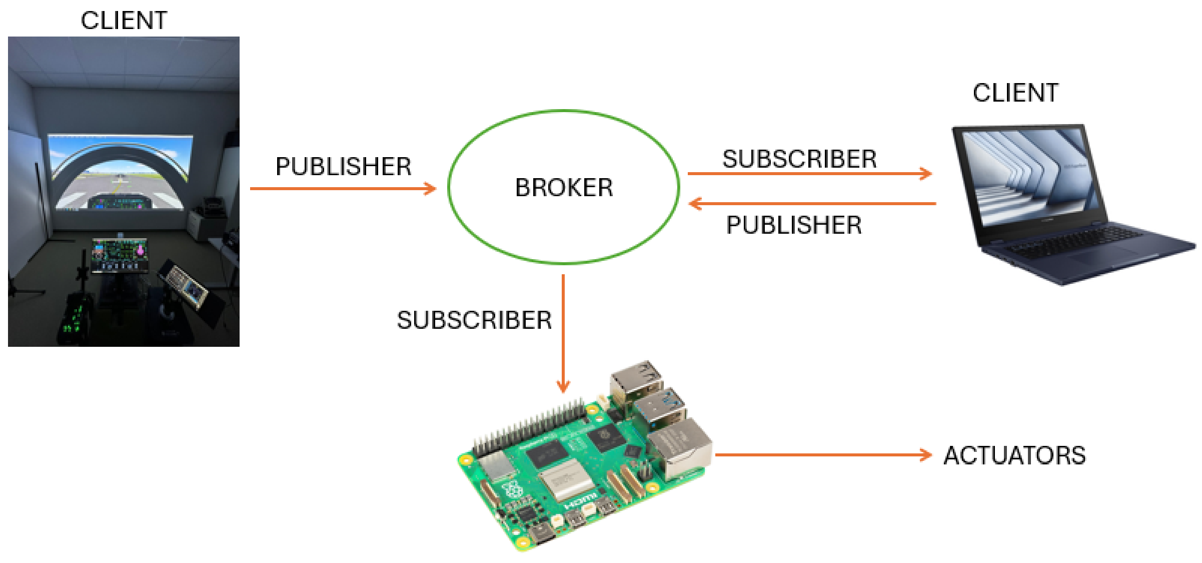


Figure 4.6: MQTT network.

The MQTT communication flow is now reported:

1. Flight simulator to PC: The flight simulator continuously publishes flight data to the PC, which listens (subscribes) to these updates in real-time;
2. PC processing: The PC analyzes the incoming data, determines the necessary feedback actions, and generates appropriate messages;
3. PC to Raspberry Pi: The PC publishes these messages to the MQTT network, where they are received by the Raspberry Pi;
4. Raspberry Pi to actuators: The Raspberry Pi, upon receiving the messages, triggers the corresponding actuators, providing the pilot with real-time tactile, thermal, or visual feedback.

4.2 Implementation of the feedback system with the flight simulator

The implementation of the feedback system involves integrating the various actuators with a flight simulator through a Raspberry Pi and controlling them via MQTT protocol. The system is designed to provide real-time feedback to the pilot based on flight conditions, enhancing the simulation experience by making it more immersive and realistic, thanks to the wearable devices.
There will be a careful analysis of the algorithms written to allow the integration of the entire feedback system, starting from the parameters sent by the flight simulator, the configurable data that can be chosen by any user and that will allow you to choose which actuators to turn on and for how long they can remain on.
Afterwards, the algorithm that will work as a filter will be analyzed and finally the one in the Raspberry Pi that will control the actuators through the GPIO (General Purpose Input Output) ports.

4.2.1 System architecture overview

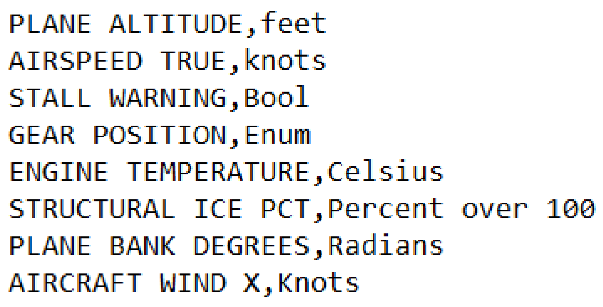
The feedback system is built around a Raspberry Pi, which serves as the main controller for the actuators. The Raspberry Pi is connected to a series of actuators, including vibration motors, Peltier modules, heating plates, LED, LED matrices and strips, which are

used to provide tactile, thermal, and visual feedback, respectively.
The flight simulator publishes various flight parameters in real-time (such as altitude, speed, or engine temperature) to specific MQTT topics. A PC, that works as a filter, subscribes to these topics, processes the data, and if they exceed certain threshold values, it communicates it to the Raspberry Pi.
Therefore, the Raspberry PI triggers the appropriate actuators based on predefined conditions.

4.2.2 Algorithm to filter parameters

In the design of the feedback system, an essential component is the algorithm used to filter and process the flight parameters received from the simulator. This algorithm ensures that only relevant and significant data triggers the actuators, preventing unnecessary activations and enhancing the overall accuracy and reliability of the feedback system.
To avoid overloading the Raspberry Pi, these algorithms are run by a PC that works as a "filter", as already written in the previous section.

First, it was necessary to choose which parameters must be communicated and controlled. In the example shown in Figure 4.7, there are the parameters chosen for the experiment done for this thesis work, however it is allowed to remove and add parameters, simply by modifying a file and inserting the name of the new parameter and its unit of measurement, taken from the flight simulator.



```
PLANE ALTITUDE,feet
AIRSPEED TRUE,knots
STALL WARNING,Bool
GEAR POSITION,Enum
ENGINE TEMPERATURE,Celsius
STRUCTURAL ICE PCT,Percent over 100
PLANE BANK DEGREES,Radians
AIRCRAFT WIND X,Knots
```

Figure 4.7: Parameters chosen.

Thus, the data communicated by the simulator including the plane altitude; the airspeed, which is a measure of the aircraft's speed relative to the surrounding air mass; the gear position that indicates the position of the landing gear from 0 to 1 (2 if it is unknown); the parameter about the stall condition; the engine temperature; the amount, in percentage, of ice on aircraft structure; the measurement of the bank angle; the wind component in aircraft lateral axis.

After choosing the parameters to send, you need to decide the threshold values that they must exceed to activate a certain condition and communicate to the pilot through the actuators.
For this thesis work, it was thought to use the information relating to the speed, the altitude of the aircraft and the position of the landing gear to warn the pilot that a possible take-off or landing operation is finished and therefore that it is necessary to modify the position of the landing gear. Thus, the threshold values that will be chosen by the user refer to the height and speed of the aircraft, as well as which actuators (referring to the Raspberry Pi GPIO port numbers) to turn on and for how long. In this case certain ERM motors will turn on according to the take-off or the landing situation.
As for the engine temperature, the value in degrees Celsius was chosen, above which, the two heating plates will be turned on. Similarly, the percentage value present in the aircraft structure was chosen to turn on the Peltier cells and generate the sensation of cold.
The stall condition, which is already communicated by the simulator and therefore does not require a threshold value, given that it is one of the most dangerous situations a pilot can find himself in, is communicated by turning on both all the vibration motors and the wearable LEDs.

As for the LED matrix and LED strip, the aim was to use them to communicate a directionality indication to the pilot. Among the available variables, the bank angle and the lateral component of the wind were chosen. Depending on the size of these two parameters, the user can choose the color of the LEDs and the speed at which they turn on. For example, if the plane reaches a dangerous bank angle tilted to the left, the LED strip will turn on its red LEDs and give an effect of movement to the left, increasing the speed depending on the danger of the bank angle. The matrix, on the other hand, will light up and flash, forming arrows in the same direction in which the wind will hit the plane, with a proportional speed.

To receive and listen to the parameters published by the simulator, analyze them and tell the Raspberry Pi to turn on certain actuators in a certain way, the following code on Python is executed:

```python
# Read the configuration file
config_file = r'C:\Users\demo.gcap\Desktop\configuration.txt'
config_vars = {}

with open(config_file, 'r') as file:
    for line in file:
        line = line.strip()
        if not line:
            continue
        name, value = line.split(',')
        name = name.strip()
        value = value.strip()
        try:
            value = int(value)
        except ValueError:
            pass
        config_vars[name] = value

# Define configuration variables
gear_height = config_vars.get('gear_height')
gear_speed = config_vars.get('gear_speed')
gear_pin1 = config_vars.get('gear_pin1')
.
.
gear_time = config_vars.get('gear_time')
.
.
.
```

This first part of the code read from a configurable file all the threshold values about the parameters. For example, the altitude and speed of the aircraft required to initiate the gear condition, as well as the pins to activate for communication and the maximum duration.
The same procedure is repeated for all parameters and all actuators to be turned on under certain conditions.

```python
sub_broker = 'localhost'
pub_broker = 'localhost'
port = 1883
sub_topic = "data/#"
pub_topic = "pin/"
# Generate a Client ID with the subscribe prefix.
sub_client_id = f'subscribe-{random.randint(0, 100)}'
pub_client_id = f'publish-{random.randint(0, 100)}'

def connect_mqtt(broker, port, client_id) -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

This code snippet sets up some basic configurations for the MQTT client. First, it specifies the broker address for the publishing and subscription client. After that, it defines the port number to connect to the MQTT broker.
About the topics, the subscription one is "data/#", where # is a wildcard that matches any sub-topic. This means that the parameters from the flight simulator starts with "data/" and the code reads all the topics starting like that. The publishing topic starts with "pin/".
The following two rows generate unique client IDs for subscribing and publishing.
In the last part of this code there is a function designed to create and configure an MQTT client, establish a connection to the MQTT broker, and return the connected client.

```python
# Dictionary to store second-level topic values with default values
second_level_values = {
    "PLANEALTITUDE": float('inf'),
    "AIRSPEEDTRUE": float('inf'),
    "STALLWARNING": float('inf'),
    "GEARPOSITION": float('inf'),
    "STRUCTURALICEPCT": float('inf'),
    "ENGINETEMPERATURE": float('inf'),
    "PLANEBANKDEGREES": float('inf'),
    "AIRCRAFTWINDX": float('inf'),
}
```

This part of the code stores the values coming from the flight simulator.

After that, all the functions that check whether certain conditions have been met have been written. As an example, only the one related to the landing situation in the case where the landing gear has not been extended is reported.

```python
was_landing = False
def check_LANDING(pub_client):
    global was_landing  # Use the global state variable

    # Check specific conditions
    landing_condition = (
        second_level_values["PLANEALTITUDE"] < gear_height and
        second_level_values["AIRSPEEDTRUE"] < gear_speed and
        (second_level_values["GEARPOSITION"] == 2 or
        second_level_values["GEARPOSITION"] = 0)
    )

    if landing_condition:
        if not was_landing:
            # Action to perform if the conditions are met and were not met before
            print("You are landing, extend the landing gear!")

            decision_time = str(gear_time)
            for pin in [gear_pin1, gear_pin2, gear_pin3, gear_pin4]:
                topic_on = pub_topic + f"{pin}"

                pub_client.publish(topic_on, decision_time)
            was_landing = True
            print("Messages published to turn on the pins for the landing condition")

    else:
        if was_landing:
            # Action to perform if the conditions are not met and were met before
            for pin in [gear_pin1, gear_pin2, gear_pin3, gear_pin4]:
                topic_off = pub_topic + f"{pin}"
                pub_client.publish(topic_off, "OFF")
                print(f"Message published to turn off pin {pin}")

            was_landing = False
```

First of all, there is a global variable that keeps track of whether the landing conditions were met during the previous check. Initially, it is set to "False". The following block defines "landing_condition" ,a boolean variable that evaluates to "True" if all the following conditions are met:

- The plane's altitude is below a certain threshold;
- The true airspeed is below a certain threshold;
- The landing gear is either fully retracted (GEARPOSITION $==$ 0) or its position is unknown (GEARPOSITION $==$ 2).

When these conditions are met and were not met before, for each selected pin (actuator), an MQTT message is published to the topic "pin/pin number", sending how long can it

stay on at most. A maximum power-on time is considered for safety reasons: if something goes wrong and there is an interruption in communication, an actuator could remain on continuously and could become dangerous, which is why a maximum power-on time is established.
Otherwise, if the conditions were no longer met, the actuators would be turned off through a message with the word "OFF", as can be seen in the next block.
After that all other functions related to checking the conditions of the other actuators were designed.
As for the support actuators, it was thought to use a topic of the form "pin/pin number/turning on speed/turning on color" to communicate how the LEDs had to turning on, and the package containing the direction to switch on.

```python
def subscribe(sub_client, pub_client):
    def on_message(client, userdata, msg):
        # print(msg.payload.decode())
        try:
            # Extract the second level of the topic
            topic_levels = msg.topic.split('/')
            if len(topic_levels) > 1:
                second_level = topic_levels[1]
                # Convert the payload from string to double
                payload_value = float(msg.payload.decode())
                # Update the dictionary with the new value
                second_level_values[second_level] = payload_value
                # Check conditions after the update
                check_LANDING(pub_client)
                check_FLY(pub_client)
                check_ICE(pub_client)
                check_HEAT(pub_client)
                check_STALL(pub_client)
                check_ROLLING(pub_client)
                check_WIND(pub_client)
            else:
                print("Invalid topic structure")

        except ValueError:
            print(f"Failed to convert payload to float: {msg.payload.decode()}")
        except Exception as e:
            print(f"Unexpected error: {e}")

    sub_client.subscribe(sub_topic)
    sub_client.on_message = on_message
```

This block of the code contains the subscribe function for messages on a specific topic and process those messages when they arrive. The incoming message data is stored, and various condition checks are triggered. The function includes also error handling to manage issues like invalid payload data or unexpected exceptions.

```python
def run():
    # Connect to the broker for subscription
    sub_client = connect_mqtt(sub_broker, port, sub_client_id)
    # Connect to the broker for publication
    pub_client = connect_mqtt(pub_broker, port, pub_client_id)
    # Set up the subscription
    subscribe(sub_client, pub_client)
    # Start the subscriber client's loop
    sub_client.loop_forever()
```

The "run" function is responsible for initializing the MQTT clients for both subscribing and publishing, setting up the necessary subscriptions, and starting the MQTT loop to handle incoming messages continuously.

In conclusion, the algorithm used as a filter has the task of continuously receiving the parameters arriving from the simulator, which can be modified by the user. After that, it collects the threshold values useful for checking certain conditions, the actuators to turn on, the time and other optional characteristics, also these can be modified by the user. Then, there is the actual programming part, in which all the conditions are verified through a Python code, and any MQTT messages are published.

4.2.3 Raspberry Pi algorithm

This subsection outlines the algorithm implemented on the Raspberry Pi to manage the activation and deactivation of various actuators based on the flight parameters received through the MQTT network.
The control algorithm is designed to efficiently translate the incoming data from the PC into specific commands that operate the actuators.
The PC has already analyzed the parameters coming from the simulator and will publish via MQTT, a new message that will be listened to by the Raspberry Pi. This message will have a topic and a specific package that will direct the control unit to command the GPIO port connected to the actuator to be turned on.

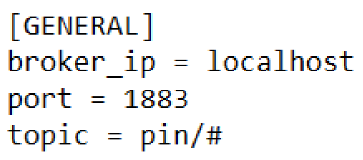
A Python code has also been programmed for the Raspberry Pi, however for simplicity and to avoid making a single code that is too long and heavy, it has been divided. The main code is the one that connects to the MQTT broker and subscribes. It receives a message, analyzes it, and based on the form of the topic it can understand whether to call a function to control the relay, the LED matrix or LED strip.
Below is the code that was implemented:

```python
import random
from paho.mqtt import client as mqtt_client
import RPi.GPIO as GPIO
import configparser
import os
import sys
import threading
from relay_control import setup_pins, handle_relay_message
from led_strip_control import handle_led_strip_message, turn_off_leds
from led_matrix_control import handle_led_matrix_message, turn_off_matrix

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# GPIO Pin Setup
pins = setup_pins()
```

This script is setting up the environment for controlling various hardware components (relays, LED strips, LED matrices). It imports all the libraries that will be used to manage the devices.
After that, it initializes the GPIO port pins that will be used.

```python
broker = ""
port = 0
topic = ""
```

This part of the code indicates that it is intended to connect to an MQTT broker, but the actual connection details are not yet provided.
This is done with the goal of making the whole system modifiable. In fact, the Raspberry Pi could connect to any MQTT network simply by modifying the following file, which will be called later in this code.



```
[GENERAL]
broker_ip = localhost
port = 1883
topic = pin/#
```

Figure 4.8: Configurable file.

If the Raspberry Pi needs to connect to another network or listen to another topic, just edit this file without changing the code.

After that, the client has to sets up and connects to the broker. As seen before, the code is the following:

```python
# Generate a Client ID with the subscribe prefix.
client_id = f'subscribe-{random.randint(0, 100)}'

def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Once connected, the Raspberry can proceed with MQTT operations, in this case subscribing.

The following part is very important. It is used to initialize a dictionary that will track threads and their associated stop events for different GPIO pins. In this application, where each GPIO pin might be handled by a separate thread (for instance, handling interrupts or periodic tasks), this dictionary is crucial to manage these threads and their associated stop signals.
In the functions that control the actuators it is needed to check if a GPIO pin (and the relative actuator) is already turned on and if it needs to be turned off.

```python
# Dictionary to track stop events and active threads
pin_threads = {key: {'thread': None, 'stop_event': None} for key in pins}
```

```python
def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        topic_parts = msg.topic.split('/')
        payload = msg.payload.decode()

        if len(topic_parts) == 2 and topic_parts[0] == 'pin':
            handle_relay_message(pins, pin_threads, topic_parts[1], payload)

        elif len(topic_parts) == 4 and topic_parts[0] == 'pin' and
        topic_parts[1] == 'strip':
            handle_led_strip_message(topic_parts, payload)
            print(topic_parts,payload)

        elif len(topic_parts) == 4 and topic_parts[0] == 'pin' and
        topic_parts[1] == 'matrix':
            handle_led_matrix_message(topic_parts, payload)
            print(topic_parts,payload)

    client.subscribe(topic)
    client.on_message = on_message

def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

def read_config():
    global broker
    global port
    global topic
    script_path = os.path.abspath(os.path.dirname(sys.argv[0]))
    config_path = os.path.join(script_path, "config/sub.ini")
    sub_parser = configparser.ConfigParser(interpolation=configparser.ExtendedInterp...
    sub_parser.read(config_path)
    broker = sub_parser.get('GENERAL', 'broker_ip')
    port = sub_parser.getint('GENERAL', 'port')
    topic = sub_parser.get("GENERAL", 'topic')
```

The subscribe function sets up the MQTT client to handle incoming messages. It splits the topic in more parts and according to the number or the name of those, it recognizes which actuator the message refers to and consequently calls the function that can control it, sending it topics and packages.
The next one starts the subscribe MQTT client and runs its main loop.
The last function reads configuration settings from a file. As written before, it receives

the connection details to the MQTT network.

The main code is finished, as seen above, it calls control functions for certain actuators. These will be analyzed in the next paragraphs.

4.2.4 Relays control

To control the actuators, it was necessary to use two relays, both connected to the Raspberry Pi through the GPIO ports.
The devices to be controlled by the relays are:

- 8 ERM vibration actuators;
- 2 Peltier cells;
- 2 adhesive heating plates;
- 4 wearable LEDs.

The code used is the following:

```python
import RPi.GPIO as GPIO
import threading
import time

def setup_pins():
    pins = {
        '4': {'gpio': 21, 'active_state': GPIO.LOW},
        '2': {'gpio': 26, 'active_state': GPIO.LOW},
        '1': {'gpio': 6, 'active_state': GPIO.LOW},
        '3': {'gpio': 16, 'active_state': GPIO.LOW},
        '12': {'gpio': 19, 'active_state': GPIO.LOW},
        '11': {'gpio': 20, 'active_state': GPIO.LOW},
        '10': {'gpio': 27, 'active_state': GPIO.LOW},
        '9': {'gpio': 17, 'active_state': GPIO.LOW},
        '8': {'gpio': 5, 'active_state': GPIO.LOW},
        '7': {'gpio': 0, 'active_state': GPIO.LOW},
        '6': {'gpio': 7, 'active_state': GPIO.LOW},
        '3b': {'gpio': 1, 'active_state': GPIO.HIGH},
        '1b': {'gpio': 11, 'active_state': GPIO.HIGH},
        '15': {'gpio': 25, 'active_state': GPIO.LOW},
        '13': {'gpio': 8, 'active_state': GPIO.LOW},
        '2b': {'gpio': 22, 'active_state': GPIO.HIGH},
    }
    for pin_info in pins.values():
        GPIO.setup(pin_info['gpio'], GPIO.OUT)
        GPIO.output(pin_info['gpio'], not pin_info['active_state'])
        # Set all pins to inactive state
    return pins
```

The function setup_pins is responsible for configuring and initializing GPIO pins on Raspberry Pi. It it possible to notice that some pins are initialized as "LOW", while those of another relay, indicated by the letter "b", are initialized as "HIGH". This is because the two relays are designed to be activated with different logic.
However, all pins are set to the inactive state, to prevent any turning on when they are initialized, otherwise it would activate the actuator and give feedback to the pilot.

```python
def handle_relay_message(pins, pin_threads, pin_key, payload):
    if pin_key in pins:
        pin_info = pins[pin_key]
        gpio_pin = pin_info['gpio']
        active_state = pin_info['active_state']

        if payload.isdigit():
            duration = int(payload)

            def activate_pin(stop_event):
                GPIO.output(gpio_pin, active_state)
                print(f"Pin {pin_key} (GPIO {gpio_pin}) ON for {duration} seconds")
                for _ in range(duration * 10):
                    if stop_event.is_set():
                        break
                    time.sleep(0.1)
                GPIO.output(gpio_pin, not active_state)
                print(f"Pin {pin_key} (GPIO {gpio_pin}) OFF after {duration} seconds")

            if pin_threads[pin_key]['thread'] is not None:
                pin_threads[pin_key]['stop_event'].set()
                pin_threads[pin_key]['thread'].join()

            stop_event = threading.Event()
            thread = threading.Thread(target=activate_pin, args=(stop_event,))
            pin_threads[pin_key] = {'thread': thread, 'stop_event': stop_event}
            thread.start()

        elif payload == 'OFF':
            if pin_threads[pin_key]['thread'] is not None:
                pin_threads[pin_key]['stop_event'].set()
```

```python
60                  pin_threads[pin_key]['thread'].join()
61              GPIO.output(gpio_pin, not active_state)
62              print(f"Pin {pin_key} (GPIO {gpio_pin}) OFF")
```

This is the function designed to control GPIO pins connected to relays based on messages received. First, it verifies that the pin_key corresponds to a valid GPIO pin defined in the pins dictionary.
Then, since the package of the received message contains the maximum duration for the pin to be turned on, it converts it into a variable called "duration". Therefore, it activates that pin (which is specified in the "pin/pin_number" topic) for that duration and then checks every 0.1 seconds whether a stop event should be set. At the end of the duration time, or if a stop event is set, it turns off the actuators connected to that pin.
The next two blocks, from rows 48 to 55, they respectively stop the thread by setting the stop event and wait for the thread to finish, and create a new stop and thread event to handle the activation of the relay.
The last part of the code manages what happen if the payload is 'OFF' (when arrives the message to turn off an actuator). It stops any existing thread controlling the pin, turns the relay off by setting the GPIO pin to its inactive state, and prints a confirmation message.

4.2.5 LED strip control

The function that controls the LED strip is called when the bank angle of the airplane reaches dangerous degrees of inclination. When this happens, the PC sends a message with a topic of the form "pin/strip/speed/color" with the package "direction".
According to the bank angle, the speed can be low, medium or high. The color is chosen by the user indicating it with the RGB format in the configurable file discussed in the previous paragraph. The direction changes according to the simulation, it could be "left" or "right" and the LEDs in the strip will light up and "slide" towards that direction.

Below is reported the example of Python code used on the Raspberry Pi to control the strip:

```python
import time
from rpi_ws281x import PixelStrip, Color
import threading
import signal
import sys
```

```python
# LED matrix configuration
LED_COUNT = 300  # Total number of LEDs in your matrix
LED_PIN = 13    # The GPIO pin to which your matrix is connected (change if necessary)
LED_FREQ_HZ = 800000  # Signal frequency in hertz (usually 800kHz for WS2812)
LED_DMA = 10  # DMA channel to use for generating the PWM signal
LED_BRIGHTNESS = 20  # Brightness of the LEDs (0-255)
LED_INVERT = False    # Set to True if the signals should be inverted
LED_CHANNEL = 1       # DMA channel to send data to (0 for the matrix, 1 for the strip)

# Create the PixelStrip object
strip = PixelStrip(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT, LED_BRIGHTN....

# Initialize the LED matrix
strip.begin()

# Global stop event
stop_event = threading.Event()
current_thread = None
```

In the first part, the code configures and initializes the WS2812 LED strip. After that, it sets up a global stop event for managing concurrent LED control threads.

```python
def movingBlock(strip, color, block_size, start_pos, wait_ms, direction, stop_event):
    if direction == 'right':
        step = -1
        end_pos = -block_size
    elif direction == 'left':
        step = 1
        end_pos = strip.numPixels() + block_size
    else:
        return

    while not stop_event.is_set():
        for i in range(start_pos, end_pos, step):
            if stop_event.is_set():
                break
            for j in range(block_size):
                if 0 <= i + j < strip.numPixels():
                    strip.setPixelColor(i + j, color)
            strip.show()
            time.sleep(wait_ms / 1000)
            for j in range(block_size):
                if 0 <= i + j < strip.numPixels():
                    strip.setPixelColor(i + j, Color(0, 0, 0))  # Turn off LED
```

In this block, the movingBlock function is defined to create an animation effect on an LED strip, where a block of LEDs (a group of consecutive LEDs) lights up and moves either to the left or right along the strip. The animation continues until a stop event is triggered. When this happens, all the lights are turned off.

```python
def handle_led_strip_message(topic_parts, payload):
    global stop_event, current_thread
    try:
        # Extract parameters from the topic
        strip_speed = topic_parts[2]
        strip_color = topic_parts[3]
        color_R = int(strip_color.split(',')[0])
        color_G = int(strip_color.split(',')[1])
        color_B = int(strip_color.split(',')[2])
        # Set wait_ms based on speed
        if strip_speed == 'low':
            wait_ms = 150  # low
        elif strip_speed == 'medium':
            wait_ms = 50  # medium
        elif strip_speed == 'high':
            wait_ms = 10  # high

        # Stop the current thread if it exists
        if current_thread is not None:
            stop_event.set()
            current_thread.join()

        # Reset the stop event
        stop_event = threading.Event()

        if payload == "NO ROLL":
            direction = "NO ROLL"
            # Turn off all LEDs before turning on new ones
            for i in range(strip.numPixels()):
                strip.setPixelColor(i, Color(0, 0, 0))
            strip.show()
            # Turn on the central LEDs
            for i in range(140, 161):
                if 0 <= i < strip.numPixels():
                    strip.setPixelColor(i, Color(color_R, color_G, color_B))
            strip.show()
        else:
            # Determine direction based on payload
            if payload == "right":
                direction = 'right'
            elif payload == "left":
                direction = 'left'
            else:
                print("Error: Invalid payload for direction.")
                return

        # Create and start a new thread for the movingBlock function
        current_thread = threading.Thread(target=movingBlock, args=(strip, Color(...
        current_thread.start()

    except IndexError:
        print("Error: Invalid topic format for the LED strip.")
```

Briefly, the above block of the code extracts parameters from the message topic and payload to determine the behavior of the LED strip, such as turning on specific LEDs or running an animation. The function also manages threading to ensure that only one animation runs at a time, and it can stop the current animation if a new one is triggered.
It just needs to be clarified that if the payload is "NO ROLL", it means that the bank angle is close to 0, therefore the aircraft maintains a rather horizontal attitude. All LEDs on the strip are turned off first; then, a specific range of LEDs (from index 140 to 160, the central ones) are turned on with a specified color. This creates a static display of a block of LEDs in the middle of the strip.

```
def turn_off_leds():
    """Turns off all LEDs on the strip."""
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, Color(0, 0, 0))
    strip.show()

def signal_handler(sig, frame):
    """Handles the SIGINT signal to turn off the LEDs."""
    #print("Interrupt received. Turning off LEDs...")
    stop_event.set()
    if current_thread is not None:
        current_thread.join()
    turn_off_leds()
    sys.exit(0)

# Register the signal handler
signal.signal(signal.SIGINT, signal_handler)
```

The code ends with the function used to turn off all the lights and another one used to handle the "SIGINT" signal, which is typically sent when the user interrupts the program by pressing "Ctrl+C". It ensures that the LEDs are turned off and that the program exits gracefully.

4.2.6 LED matrix control

This function is used to control le LED matrix. Among the variables available by the simulator, it was decided to use the lateral component of the wind blowing on the aircraft to have a parameter to communicate the directionality of.
The MQTT message received by the Raspberry Pi has a topic of the form "pin/strip/speed/color" with the package "direction" like the one discussed in the previous paragraph.

The script used to handle the actuator is similar to the LED strip one, as you can see in the code reported below:

```python
import time
from rpi_ws281x import PixelStrip, Color
import threading
import signal
import sys

# LED matrix configuration
LED_COUNT = 256  # Total number of LEDs in your matrix
LED_PIN = 18    # The GPIO pin to which your matrix is connected (change if necessary)
LED_FREQ_HZ = 800000  # Signal frequency in hertz (usually 800kHz for WS2812)
LED_DMA = 10  # DMA channel to use for generating the PWM signal
LED_BRIGHTNESS = 20  # Brightness of the LEDs (0-255)
LED_INVERT = False    # Set to True if the signals should be inverted
LED_CHANNEL = 0       # DMA channel to send data to (0 for the matrix, 1 for the strip)

# Create the PixelStrip object
strip = PixelStrip(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT, LED_BRIGH...

# Initialize the LED matrix
strip.begin()

# Global stop event
stop_event = threading.Event()
current_thread = None
```

The only difference to note compared to the previous one is the number of "LED_CHANNEL". When controlling both a LED matrix and a LED strip simultaneously on the same Raspberry Pi using the same ("rpi_ws281x") library, you need to use different DMA channels (LED channels 0 and 1) to avoid conflicts in signal generation. This way, both the matrix and the strip can be controlled simultaneously without interfering with each other.

To generate the "drawing", it was chosen to manually compose the pattern of lit LEDs, as can be seen in the following piece of code:

```python
pattern1 = [
    "00111100",
    "00111100",
    "00111100",
    "00111100",
    "00111100",
    "11111111",
    "01111110",
    "00111100",
    "00011000",
]
```

This is one of the patterns used, the others are done in the same way but form up to three arrows. The chapter concerning the tests will make you understand the design that is made by the matrix.

```python
def turn_on_matrix(color,wait,direction):
    if direction == 'right':
        while not stop_event.is_set():
            for y in range(8):  # 8 rows
                for x in range(9):  # 9 columns
                    index = y * 9 + x
                    if pattern1[index // 8][index % 8] == '1':
                        strip.setPixelColor(index, color)
                    else:
                        strip.setPixelColor(index, Color(0, 0, 0))  # Turn off LED
```

This function is designed to animate a LED matrix by lighting up certain LEDs, according to the patterns. In the original code different patterns turn on in a sequential manner. The function will continue to animate the LED matrix until a stop event ("stop_event") is set.

The same is done when the direction is "left" too.

```python
def handle_led_matrix_message(topic_parts, payload):
    global stop_event
    global current_thread

    try:
        # Extract parameters from the topic
        matrix_speed = topic_parts[2]
        matrix_color = topic_parts[3]
        color_R = int(matrix_color.split(',')[0])
        color_G = int(matrix_color.split(',')[1])
        color_B = int(matrix_color.split(',')[2])
        color = Color(color_R, color_G, color_B)

        # Set wait_ms based on speed
        if matrix_speed == 'low':
            wait = 500/1000  # low
        elif matrix_speed == 'medium':
            wait = 100/1000  # medium
        elif matrix_speed == 'high':
            wait = 50/1000  # high

        # Stop any running thread
        if current_thread and current_thread.is_alive():
            stop_event.set()
            current_thread.join()

        stop_event.clear()

        if payload == "NO WIND":
            direction = "NO WIND"
            # Turn off all LEDs
            for i in range(strip.numPixels()):
                strip.setPixelColor(i, Color(0, 0, 0))
            strip.show()

        else:
            # Determine direction based on payload
            if payload == "right":
                direction = 'right'
            elif payload == "left":
                direction = 'left'
            else:
                print("Error: Invalid payload for direction.")
                return

        def run_matrix():
            while not stop_event.is_set():
                turn_on_matrix(color, wait, direction)
                time.sleep(wait)
            # Turn off LEDs after thread is stopped
            turn_off_matrix()

        current_thread = threading.Thread(target=run_matrix)
        current_thread.start()

    except IndexError:
        print("Error: Invalid topic format for LED matrix.")
```

This code is also similar to the one seen for the LED strip. The function carefully manages threading to ensure that only one animation runs at a time, stopping any previous animation before starting a new one.
If the message is "NO WIND", the function turns off all the LEDs by setting them to (0, 0, 0) and displays this state immediately. Also after an event of stop, it is ensured that the LEDs are turned off by calling "turn_off_matrix".

In the final part of the code, the functions to turn off the matrix and to manage keyboard interruption with the "Ctrl+C" command are defined.
This part is the same one already seen in the previous section and there is no need to return to it.

Chapter 5

Simulation and experimentation

In this chapter, we delve into the practical aspects of the feedback system designed in the previous chapters. The focus is on validating the system's functionality through simulation and real-world experimentation. We will explore the methodologies employed to simulate various flight scenarios, the setup of the experiments, and the analysis of the outcomes.
The goal is to demonstrate how the feedback system could improve pilot awareness and safety by providing multisensory cues during critical flight operations.
This chapter outlines the experimental procedures, results, and observations about the limitations due to the flight simulator.

5.1 Experimental validation

To validate the system's effectiveness, a series of experiments were designed. These experiments were carried out to simulate real-world scenarios and evaluate the performance of the multisensory feedback system.
As a first step, a set of scenarios was predefined, ranging from routine flight conditions to more complex emergency situations like engine overheating and sudden weather changes, situations used respectively to activate the heating plates in the first case; the Peltier cells and the LED matrix which deals with the wind in the second.
Regarding the other conditions to be verified for communication to occur through the actuators, they have been reproduced by simulating a rash or careless action by the pilot. For example, voluntarily forgetting the extended landing gear at the end of a take-off operation, or not extracting it while approaching a landing; a dangerous maneuver that brings the aircraft to a stall condition; a turn with too high a bank angle.

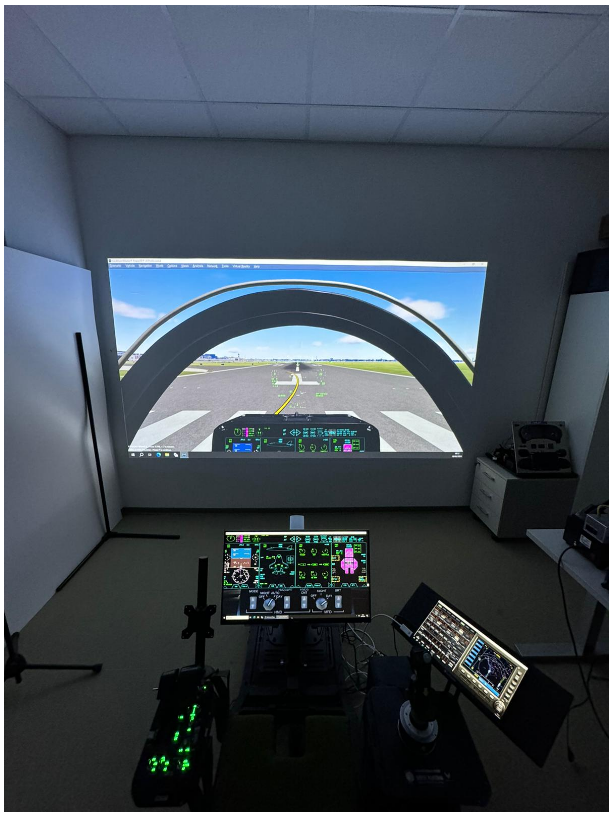


Figure 5.1: Flight simulator.

Once the scenario has been prepared and the pilot has worn the actuators, the connection of all clients to the MQTT network has been executed. The simulator began to publish its parameters which were constantly verified by the PC, while the Raspberry Pi remained waiting to receive messages to turn on the actuators when certain conditions occurred.

During each simulation, based on the data received, the actuators (haptic devices, LEDs, LED matrix, LED strips, thermal actuators) were triggered. The timing, accuracy, and appropriateness of the feedback were collected.
In a set of the experiments, pilot interaction with the system was observed to understand how the system impacted their situational awareness, decision-making and overall flight performance; the collected data were analyzed to evaluate the feedback system. The key factors evaluated included system responsiveness, feedback accuracy, and the opinion by the user too. It is worth specifying that the system has not been tested by real aircraft pilots but by users who simply tried the simulator and gave their opinion. However, in this chapter the term "pilot" was still used, also to avoid excessive repetition of the word "user" to indicate one of the subjects of the experiment.

5.2 Performance evaluation

The performance evaluation of the feedback system was a critical step in determining its effectiveness in flight simulation scenarios. The goal was to assess how well the system responded to the factors written in the previous section. A combination of objective data analysis and subjective user feedback was employed to conduct an overall evaluation of the system's performance.

5.2.1 Responce time and latency

One of the key metrics for evaluating the feedback system's performance was the response time between the moment a flight parameter exceeded its predefined threshold and the activation of the corresponding actuator.

- Haptic feedback response: The vibrations generated by the haptic actuators were tested for latency. The system showed a very good average response time. In fact, as soon as the aircraft exceeded the altitude used as a condition to communicate the message to re-enter or extract the landing gear, the actuators immediately turned on, making it difficult to even measure how much time passed. This result was more than sufficient for real-time alerts during critical flight phases, such the ones discussed above.

- Thermal feedback response: Thermal actuators, such as Peltier modules or adhesive heating plates, had a slightly longer response time due to the nature of heat generation, averaging 1 second before noticeable feedback was felt by the pilot.
  Although it was slower than other actuators, it was still within an acceptable range for use in specific alert conditions such as engine overheating or the presence of ice in the aircraft, considering that these situations are not immediate but take time to occur.

- Visual feedback response: In this type of actuators, there is a need to make a distinction. In fact, while small wearable LEDs demonstrated an almost immediate activation, the others do not.
  In fact, the LED matrix, observing the parameters of the lateral wind acting on the aircraft, showed a slight delay, but this too was in the order of 1 second, therefore still acceptable.
  Even for the LED strip there is sometimes a slight latency within 1 second. From the experiment it emerged that it happens during the "transition" from a situation of horizontal stability to a turning situation, this inconvenience depends on the fact

that, as can be seen in the block of code containing the "handle_led_strip_message" function in paragraph 4.2.5, when the absence of a roll has been communicated, a further command is given to turn off all the lights and then turn on only the central ones, to avoid that any LEDs remain lit in addition to those that must communicate the horizontal position. As for the other transitions from one rolling situation to another, there are no particular delays to highlight.

5.2.2 Accuracy of feedback

Another important performance criterion was the accuracy of the feedback provided by the system. The goal was to ensure that the actuators were triggered by the correct flight parameters and that the feedback delivered was relevant to the situation. Also in this subsection one actuator at a time will be analyzed.

The haptic actuators were highly accurate in providing feedback when the flight parameters breached their thresholds. Their vibration was definitely perceptible but at the same time not too strong to cause discomfort or annoyance. Furthermore, it is immediately distinguishable, managing to immediately attract the attention of the pilot who, knowing the reason for which it was activated, could take the appropriate countermeasure to get out of the dangerous situation.
As for thermal actuators, both manage to provide feedback with high precision. The heating plate never reaches too high temperatures and even the Peltier cell, despite some initial concerns, generates a noticeable but not exaggerated sensation of cold, and above all it does not risk overheating both faces (as written at the end of paragraph 3.3.2) despite realistic and prolonged use over time.

All the LEDs lit correctly, providing just the right lighting without risking blinding the pilot. The support actuators also have good accuracy, displayed the correct color patterns and symbols for various alerts, such as directional arrows and flowing lights.
The directional feedback, in particular, could be effective in guiding the pilot to make corrective maneuvers.
The operation of the LEDs under different conditions is shown in the following figures:

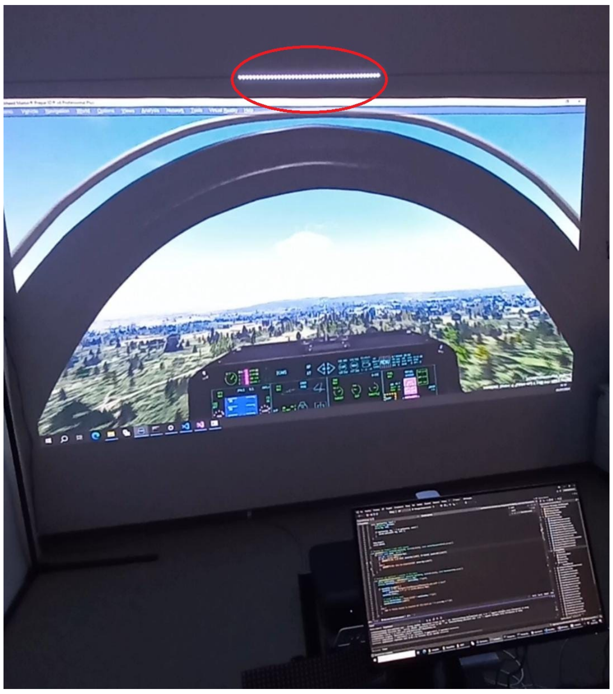


Figure 5.2: LED strip in no roll situation.

The image above shows the no roll situation where the LED strip remains lit just in the center.

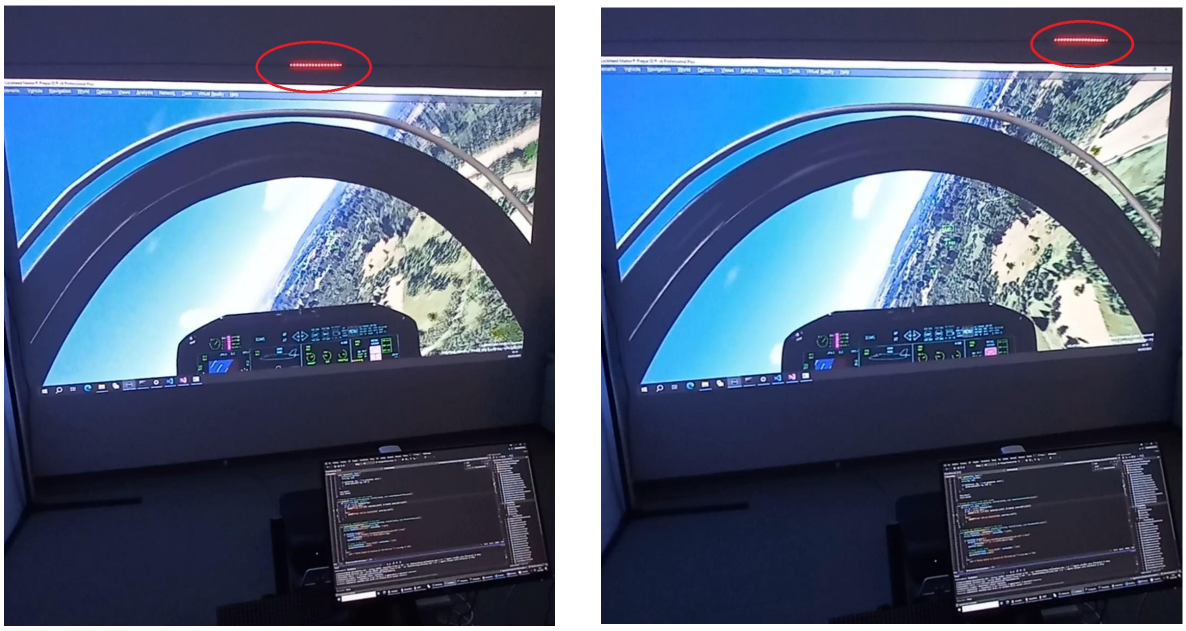


Figure 5.3: LED strip in turn sitation.

In these ones a right turn situation is shown. It is possible to note that the LEDs light up and slide along the strip.

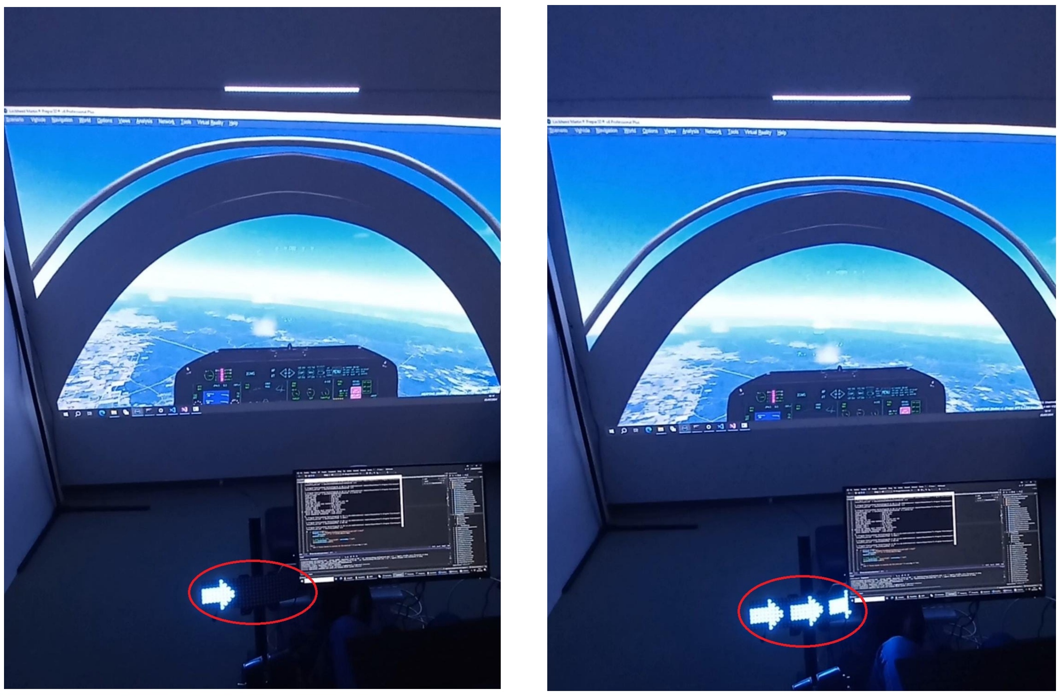


Figure 5.4: LED matrix in action.

The LED matrix, instead, comes into action only when there is a lateral wind component that exceeds a certain threshold. When this happens, three arrows light up and alternate in the direction the wind is blowing, as can be seen in Figure 5.4 in a right-wind situation. As written in the previous chapters, the color and speed with which the LEDs light up in different conditions are decided by the user using a configurable file.

5.2.3 User feedback

Pilot performance during the experimentation phase was a key factor in evaluating the system's overall effectiveness.
The system was found to significantly reduce the pilot's reaction time in responding to critical flight conditions. Users were able to interpret and react to alerts quickly, particularly when receiving multisensory feedback. They reported that the feedback system helped them make more confident and accurate decisions during flight too. In fact, just a simple preparation on all the feedback from the actuators and the conditions to which they are connected is enough to exploit the full potential of this system.
Furthermore, all feedback was never unpleasant or annoying, but on the contrary pilots reported that the feedback system could help them make more confident and accurate decisions during flight. The combination of tactile, visual, and thermal feedback allowed them to easily draw their attention to a potential dangerous situation.

5.3 System limitations

Despite the positive results, the performance evaluation also identified certain limitations and areas for potential improvement. These included some actuators imperfection, such as delays in the response or a better customization of feedback intensity; and limitations due to the flight simulator which cannot reproduce all the conditions in which the system can be exploited.
Understanding these limitations is crucial for future improvements and potential real-world applications.

5.3.1 Feedback delay

One of the most prominent limitations identified during testing was the delay in some actuators feedback response. For thermal actuators, this is not a big problem, because as already mentioned, the scenarios to which they are connected do not require rapid warning. In fact, just as the ice present in the structure of the aircraft takes time to form and is certainly not an immediate situation like the sudden stalling of the aircraft, then it makes sense that the Peltier cell also takes about 1 second to cool down.

As regards the LED matrix, as mentioned in the previous section, it also takes about 1 second to communicate what appears on the simulator parameters about the lateral wind, and this could be a problem. The reason could be due to the Raspberry Pi's limited computing power. In fact, although ideal for small projects, it is probably not able to manage the more complex ones in the best way, in which both the 16 actuators must be controlled via relays, and the intelligent ones such as matrix and strip which are composed of another 256 and 300 LEDs. Although the "dirty work" of analyzing and filtering the parameters from the simulator has been assigned to the PC, the Raspberry Pi still has difficulty controlling so many devices.
A solution could be to directly connect the LED matrix to the computer or even change the algorithm. In fact, the feedback system is "polling" for changes (constantly checking for updates), and this approach could introduce delays. On the other hand, using interrupts or event-driven programming, instead of the polling one, could make the system more responsive by reacting immediately to changes in parameters. However, this requires significant programming skills and was not adopted for this thesis project.
A similar reasoning can also be made for the LED strip, there is a slight delay in switching on, in particular when switching from/to the no roll situation and there is the command to turn off all the LEDs before turning on the central ones, as written in the previous section.

5.3.2 Limited feedback customization

During the experimentation phase, users expressed the need for greater customization of the feedback system, particularly in terms of intensity and type of thermal actuators feedback. In fact, the current system provides predefined feedback levels for actuators based on specific flight parameters.
The ideal solution would be for the thermal actuators to be proportional to the situation they want to communicate. To be more precise and clear, the heating plate should get hotter as the engine temperature gets hotter, proportionally, rather than only turning on when the engine reaches dangerous temperatures; the same way the Peltier cell could do with the presence of ice.
However, this solution is not so immediate and easy to obtain, in fact to obtain it you should use a temperature sensor that measures the heating plate and a controller in an automatic closed loop control system that keeps the actuator at a desired temperature, based on the information received from the temperature sensor connected to the engine.

5.3.3 System scalability

The current feedback system was developed and tested within the controlled environment of a flight simulator. While this setup successfully demonstrated the system's potential, scaling it up for use in actual aircraft presents several challenges. In the real world, flight environments are much more complex, with a greater number of variables, such as high vibrations and electromagnetic interference. The actuators may not perform optimally under these conditions without further testing and design adjustments.
Moreover, integrating this system with real aircraft systems, which are governed by strict aerospace safety standards, presents a major challenge. The system would need to pass rigorous certification processes, ensuring that it can operate reliably without interfering with other critical flight systems. Achieving this level of integration would require additional development and testing beyond the scope of this thesis project.
In conclusion, if the ultimate goal of this project is to design a system that can be used in the real world in a cockpit, there is still a lot of implementation work to be done to get to the goal.

5.3.4 Flight simulator limitations

One of the key limitations is the inability to fully reproduce certain flight conditions and scenarios that would be important for a comprehensive evaluation of the system's effectiveness in actual cockpit environments. Simulators, while sophisticated, cannot perfectly replicate all the dynamic factors encountered during real flights.

In fact, especially for the LED matrix and LED strip, the simulator did not allow them to be used for something more useful or crucial. While the indication of the wind direction could still be very important, the use of the strip to communicate the inclination of the aircraft could be avoidable, as there is already the presence of the artificial (or attitude) horizon for this purpose.

Some of the original ideas were to use the LED strip to indicate the correct direction to follow in case of deviation from the programmed route or even to indicate the direction to warn the pilot of the presence, in the flight area, of a drone, another unauthorized aircraft or even a bird to avoid birdstrike. However, these situations are impossible to replicate in the flight simulator available, for this reason we had to settle for using the bank angle.

Chapter 6

Conclusions

This chapter summarizes the main parts of the thesis, reflecting on the design, implementation and testing of the feedback system for flight simulators. Furthermore, an analysis of the results will be done.
The main objective of this work is to create a new HMI channel that can exploit wearable and intelligent devices that can send the pilot certain conditions and states in which the aircraft is located, which could bring a substantial improvement, in particular in safety, in controlling the aircraft and helping the pilot.
We will also explore future work and potential developments that could further improve the functionality and performance of the system, addressing both limitations encountered and opportunities for more advanced functionality.

6.1 Summary

This research and the development of a multisensory feedback system have highlighted the potential of combining haptic, visual, and thermal actuators to improve pilot situation during flight operations. The system was successfully designed to communicate critical flight data through a variety of sensory channels. The use of MQTT communication allowed real-time data exchanges between the flight simulator, the computer working as a filter and the Raspberry Pi controlling the devices integrated into the pilot's suit or distributed in the cockpit.
The project aimed to improve pilot awareness and safety by providing real-time feedback through a combination of multisensory actuators that will convey to the pilot particular conditions/states in which the aircraft is located.

The key components of the actuators system are:

- Haptic actuators: Vibrational motors were employed to provide tactile alerts, ensuring that the pilot receives immediate feedback on critical flight parameters;
- Visual feedback actuators: Wearable LEDs, a LED matrix and a LED strip were used to provide visual cues, adding an extra layer of redundancy to the visual information presented in the cockpit;
- Thermal actuators: The inclusion of these devices, such as Peltier modules and adhesive heating plates, offered an additional means of conveying information through temperature changes.

After choosing the actuators, a phase relating to their power supply and wired connection was carried out. The result was a system that was relatively easy to move, in fact the control unit, i.e. the Raspberry Pi, together with relays and a switch case, can be placed in a box that can be placed near the seat of the flight simulator, and outside from it only the actuator cables and those directed towards the power supply.

After that, a long programming phase followed, designing and implementing the feedback system with the simulator. The tests and results collected by the system have demonstrated its effectiveness in providing intuitive feedback to pilots, especially in situations where classic visual or auditory alerts might be less advantageous. Despite the positive results, some problems and imperfections have emerged, such as delays in the visual actuators (LED matrix and LED strip) and difficulties in replicating real-world scenarios in a flight simulator. However, these imperfections could be considered as opportunities to work on for future developments and improvements of the project.
However, despite this, the system still manages to satisfy its requirements and a good overall performance can be considered for the entire work thesis.

6.2 Future work and developments

While the project successfully achieved its primary objectives, there are numerous avenues for future work and system improvements.
One key area for improvement is trying to cope with and correct the small problems and imperfections seen in the previous chapters, for example addressing the delays observed in the LED matrix and LED strip feedback. Future studies and research could involve

switching to an event-driven programming approach, which would allow for more efficient and responsive communication between the flight simulator, Raspberry Pi, and actuators. Another challenge faced was the inability to reproduce certain real-world flight scenarios in the simulated environment. Future work could focus on expanding the range of scenarios that can be simulated, particularly those involving the presence of other flying objects, planes, drones or birds in the aircraft's airspace.
The other idea to improve the system is to also integrate a temperature sensor into a closed loop control system to better control the thermal actuators, in such a way as to provide more varied feedback and which can better communicate the condition it is in the aircraft or its engine temperature.

6.2.1 Wireless connectivity and wearability

Improving the wearability of the devices, perhaps by connecting them via wireless, can be a decisive step in improving the entire system. As written previously, the system uses soldered cables that connect the Raspberry PI with the actuators worn by the user or distributed in the simulation environment. While functional, this setup presents limitations in terms of mobility, comfort, and ease of use. By transitioning to wireless solutions, both the performance and ergonomics of the system can be significantly improved.
One of the main goals for future studies is to eliminate the need for physical wiring by integrating wireless communication protocols between the system components, such as Bluetooth and Wi-Fi. Furthermore, with studies and research on Wi-Fi protocols, in the future it will be possible to have faster connections and perhaps be able to reduce latency compared to the system tested for this project.
Another important aspect for the development and improvement of the system is to improve wearability, with the aim of making it more comfortable, especially considering prolonged use over time. The current setup, while effective, includes actuators already existing on the market, integrated into the pilot's suit, jacket or gloves. The challenge could lie in optimizing the design to make the system more comfortable, lighter, and less intrusive. Moreover, future designs could incorporate lightweight, high-capacity batteries to power the actuators. Another idea could be to embed the actuators directly into the fabric of the wearable suit to reduce bulk and improve durability.
Improving wireless connectivity and wearability will make the feedback system not only more practical for pilots but also more comfortable for prolonged use. With a wireless and lightweight system, pilots will be able to concentrate only on their task without being bothered by the bulk of the equipment.

6.2.2 Machine Learning for personalized feedback

Machine Learning is a field of study in computer engineering aimed at developing and studying algorithms which, based on collected data, can perform actions and carry out tasks autonomously. Incorporating these algorithms could allow the system to adapt its feedback based on the pilot's preferences and responses.
By analyzing the pilot's behavior during flight, the system could optimize when and how to provide feedback, ensuring that the information is delivered in the most effective and best way possible, also knowing the preference of the user who is using it at that moment. In this way, an improvement in performance could be achieved given by an intelligent and personalized feedback system, based on the pilot, capable of "predicting" particular situations and communicating them before they occur.

6.3 Final thoughts

The study, design and implementation of a multisensory feedback system represents a significant step forward in the sector of HMIs used in the aerospace sector. Through the integration of wearable haptic, visual and thermal actuators, combined with algorithms for their control, it was possible to create a communication channel between pilot and aircraft. This feedback mechanism has proven to be truly effective in improving awareness of the aircraft's status, reducing the pilot's workload by providing intuitive feedback and contributing to an overall improvement in avionics safety.

One of the key achievements of this work has been the successful integration of different actuator types into a system that communicates with a flight simulator in real time. The use of a Raspberry Pi to control the actuators, connected to the flight simulator via MQTT, ensured efficient data flow and system responsiveness. While the project has encountered some limitations, the overall functionality has shown promising and a good basis to work on for future projects and developments.
The use of actuators able to exploit different senses has opened up new possibilities regarding the way of communicating with the aircraft. This multi-sensory approach can help avoid information overload and an exaggerated dependence on traditional HMI devices, especially displays that could be difficult to interpret in potentially dangerous or high-stress situations.

Moving forward, the limitations of the system offer good ideas for future developments and improvements. Research on wearability, reduction of delays and latency and the integration of a system of wireless devices will be carried out in order to make everything more practical and suitable for real-world scenarios and applications.

In conclusion, this project has laid a solid foundation for the development of multisensory feedback systems in HMI used in aviation. The results obtained demonstrate the system's potential, and while there is still room for improvement, the work completed so far has provided a good framework for future research and innovation. By continuing research to improve and refine this technology, the aerospace industry will benefit from increased safety, improved pilot performance and a more intuitive interface between man and machine.

Bibliography

- Ricci, P. (2023). *L'evoluzione dell'interfaccia uomo-macchina nell'industria: ottimizzare l'efficienza e la produttività.* Innovation Post.

- Park, J., Lee, Y., Cho, S., et al. (2024). *Soft Sensors and Actuators for Wearable Human–Machine Interfaces.* School of Energy and Chemical Engineering, Ulsan National Institute of Science and Technology (UNIST).

- Wang, D., Zhao, S., Lou, Z., et al. (2020). *Wearable Sensors-Enabled Human–Machine Interaction Systems: From Design to Application.* Chinese Academy of Sciences.

- Kim, K., Suh, Y., Ko, S. (2020). *Smart Stretchable Electronics for Advanced Human–Machine Interface.* Institute of Advanced Machines and Design, Seoul National University.

- Francescotti, F.(2012). *Avionica: I sistemi elettronici dei velivoli.* Aviolibri.

- Yem, V., Kajimoto, H. (2017). *Wearable tactile device using mechanical and electrical stimulation for fingertip interaction with virtual world.* The University of Electro-Communications: The Department of Human Communication, Tokyo, Japan.

- Takahashi, N., Okazaki, R., Okabe, H., et al. (2011). *Sense-Roid: Emotional Haptic Communication with Yourself.* The University of Electro-Communications: The Department of Human Communication, Tokyo, Japan.

- Jamiul Haque, K. M., et al. (2024). *A Smart Jacket with Portable Thermoelectric Cooling System.* IOP Conference Series: MaterialsScience and Engineering.

- Lemmens, P., Crompvoets, F., Brokken, D., et al. (2009). *A body-conforming tactile jacket to enrich movie viewing.* WHC '09: Proceedings of the World Haptics 2009.

- Arafsha, F., El Saddik, A., Alam, K. M. (2012). *EmoJacket: Consumer centric wearable affective jacket to enhance emotional immersion.* Conference: Innovations in Information Technology (IIT).

- Van Baelen, D., et al. (2021). *Flying by Feeling: Communicating Flight Envelope Protection through Haptic Feedback.* International Journal of Human–Computer Interaction.
- Auer, S., Anthes, C., Reiterer, H., Jetter, H. (2023). *Aircraft Cockpit Interaction in Virtual Reality with Visual, Auditive, and Vibrotactile Feedback.* Proceedings of the ACM on Human-Computer Interaction.
- Chen, Y., Yang, Y., Li M., et al. (2021). *Wearable Actuators: An Overview.* Textiles.
- Safi, M., Chung, J., Pradhan, P. (2019). *Review of augmented reality in aerospace industry.* Aircraft Engineering and Aerospace Technology, Vol. 91 No. 9, pp. 1187-1194.

- Safi, M., Chung, J., Pradhan, P. (2019). *Review of augmented reality in aerospace industry.* Aircraft Engineering and Aerospace Technology, Vol. 91 No. 9, pp. 1187-1194.

- Fraden, J.(2010). *Handbook of Modern Sensors.* 4th edition.
- Dementyev, A., Olwal, A., Lyon, R. F. (2020). *Haptics with Input: Back-EMF in Linear Resonant Actuators to Enable Touch, Pressure and Environmental Awareness.* UIST '20: Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology.
- Terasaki, I. (2011). *Thermal Conductivity and Thermoelectric Power of Semiconductors.* Comprehensive Semiconductor Science and Technology, Vol. 1, pp. 326-358.