UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Calibratge, caracterització i aplicació d'un Sistema de mesura ambulatory de la temperatura de la pell i l'activitat física

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Nagore Castro Fornaguera

In partial fulfilment

of the requirements for the degree of

MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor:

Garcia Gonzalez, Miquel Angel

Ramos Castro, Juan Jose

Barcelona, May 2022

Title of the thesis: Calibratge, Caracterització i Aplicació d'un Sistema de mesura ambulatoria de la temperatura de la pell i l'activitat física

Author: Nagore Castro Fornaguera

Advisor: Garcia Gonzalez, Miquel Angel

Ramos Castro, Juan Jose

Abstract

The project focuses on the design and development of a portable peripheral temperature measurement device to study the sleep cycle that is closely linked. The distal skin temperature increases around the first hours of the night, reaching the maximum when the user is completely slept, where it remains constant until the awakening begins. This is the continuation of a previous project where improvements have been made. Changes have been made in the interconnection of the sensors and the user interface. Moreover, a calibration procedure to correct the error due to the ambient temperature drifts on the skin temperature has been made to obtain an accurate system that correctly acquires the temperature of the distal skin. Some statistics applied to the measurements allow to detect when a user has a good quality sleep. Physical activity, another function added in the project, has been seen to be related with changes in temperature obtained at night.

Acknowledgements

Different people have participated in the work carried out during this project, to whom I want to express my gratitude for the help and time invested. The project tutors, Miquel Ángel García González and Juan José Ramos Castro, for guiding me along the way and being available at any time to answer questions. To Alfonso Méndez Moya, laboratory technician, for helping me weld the newly designed prototype. Finally, to my family for being by my side supporting me during the project.

Revision history and approval record

Revision	Date	Purpose
0	01/04/2022	Document creation
1	28/04/2022	Document first revision
2	04/05/2022	Document second revision
3	09/04/2022	Document last revision

Written by:		Reviewed and approved by:	
Date	13/05/2022	Date	13/05/2022
Name	Nagore Castro Fornaguera	Name	Garcia Gonzalez, Miquel Angel Ramos Castro, Juan Jose
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1 Introduction

The project is a continuation of the design and development of a portable peripheral temperature measurement device to study the sleep/wake cycle which shows a close relationship with circadian changes in body temperature: core temperature decreases when an individual falls asleep, and the end of the sleep period coincides with the rising phase of the core temperature curve. However, recent evidence suggests that sleepiness, the variable to be studied at the project, may be more closely linked to increased peripheral skin temperature than to a core temperature drop, and that distal skin temperature seems to be correlated with core body temperature (CBT), suggesting that heat loss from the extremities may drive the circadian rhythm of CBT.

The previous project focused on the development of the hardware and software of the system, finding the components that meet the requirements, and position them in the most efficient way possible to get the system running smoothly. Regarding the software, a code was developed to obtain the desired temperature data, but the user testing or calibration was not carried out.

This project will allow to finish the implementation of the wearable device by improving some practical issues and to gain insight on the peripheral skin temperature dynamics during sleep cycles in field experiments in volunteers.

1.1 Objective

The aim of the project is to improve the design of the previously developed device to obtain the peripheral temperature of the wrist and add the function of measuring physical activity through an accelerometer to correlate the information obtained from the temperature sensors. Some of the required improvement include connection issues with the sensor attached to the skin, operability of the software for configuration of the device and data acquisition and a calibration to improve the accuracy of the measuring system against temperature changes.

On the other hand, the second main objective is to characterize the data obtained in measurements made to different users while looking for an indicator that correlates with the sleep quality of the subject as assessed by the level of activity experienced during a sleep cycle.

1.2 Requirements

The device is designed to be bracelet type, located on the wrist to record and store the peripheral and ambient temperatures through two temperature sensors. The sensors have

to present enough resolution to detect the variation of temperature over the measurement time. It will also be important to ensure a correct thermal attachment of the device to the wrist which guarantees a good contact with the skin sensor to get an accurate result.

In addition, related to the hardware, the fact of including the accelerometer it does not have to significantly increase the consumption of the battery since a low-power consumption it is required since the device have to work for a period of time longer than 24 hours to analyse an entire circadian cycle without the need of recharging. The technical requirements are listed in Table 1.

Body temperature measurement range	25ºC - 50ºC
Ambient temperature measurement range	-20ºC - 65ºC
Measurement resolution	Better than 0.05ºC
Battery life	> 24h
Data storage	> 24h

Table 1 Technical requirements of the project

Related to the software, it has to be user-friendly to be able to be used by any user. Besides, it is required to visualize the data on the serial monitor to observe if the device is obtaining the data correctly, together with the possibility to stop the acquisition without the necessity of restart the system which is a disadvantage observed in the previous project.

2 State of the art

The sleep/wake cycle shows a relationship with the circadian rhythm that affects the temperature changes. An individual normally falls asleep when the core temperature is decreasing, and the end of the sleep period coincides with the rising phase of the temperature circadian curve [1]. Nevertheless, the sleeping cycle has a tighter temporal relationship with the distal skin temperature than with the minimum core body temperature (CBT), suggesting that sleepiness may be closely linked to distal skin temperature.

Next, the physiological mechanism that regulate skin and core body temperature are explained and a short review on temperature sensors is included.

2.1 Circadian Rhythm

A circadian rhythm is a natural, internal process that regulates physiological functions such as the sleep-wake cycle, physical activity, alertness, hormone levels, body temperature, immune function, and digestive activity. The rhythms are driven by a circadian clock which is a biochemical oscillator that cycles with a stable phase and is synchronized with solar time. The circadian pacemaker is the suprachiasmatic nucleus (SCN) of the hypothalamus [2]. It coordinates the circadian rhythms across the entire body, influenced by a combination of internal and external cues. The intrinsic period of the cycle is approximately 24 hours [1], although it can vary slightly between individuals, so, to adjust to the exact 24-hour cycle, the SCN uses external signals, Zeitgebers, to synchronize. A Zeitgeber is any external or environmental cue that entrains or synchronizes an organism's biological rhythms inducing changes in the concentration of the molecular components of the clock to levels consistent with the appropriate stage in the 24-hour cycle. Light is the strongest synchronizing Zeitgeber for the circadian system and therefore keeps most biological and psychological rhythms internally synchronized, which is important for optimum function. Light resets the oscillations in the SCN through a mechanism involving melaniosin-containing retinal ganglion cells which project directly to the SCN via the retino-hypathalamic tract [2]. Through an indirect pathway, circadian information reaches the pineal gland where the hormone melatonin is produced. Together with neural information from retinal ganglion cells can also directly act on the sleep-wake system.

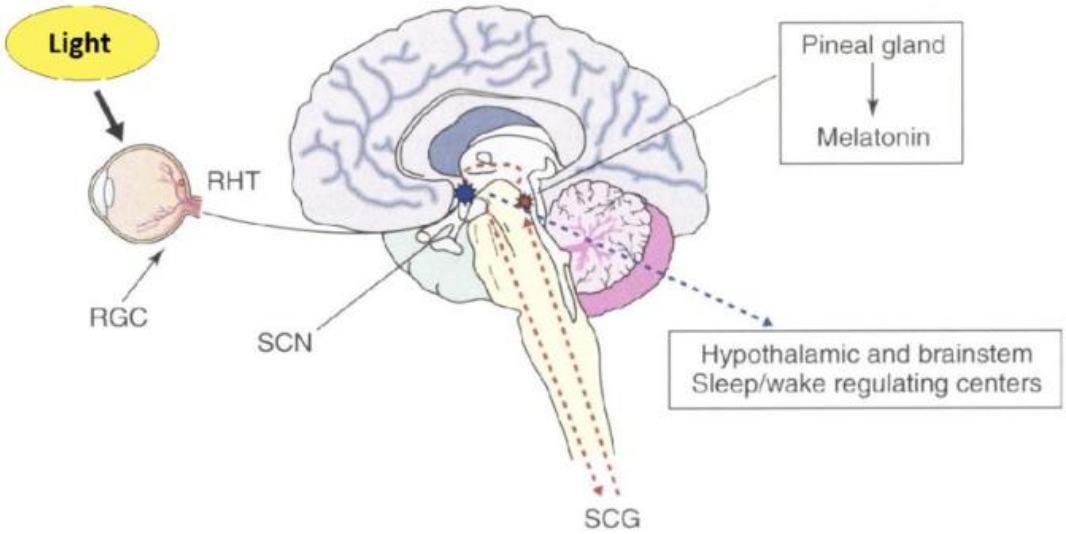

Fig. 1 Regulation of Circadian Rhythms

Some circadian rhythms are used as biological clocks, such as the work/rest cycle, salivary melatonin cortisone levels, heart rate, blood pressure, and core body temperature. Although its measurement is simple, a specialized laboratory is needed to have controlled conditions, which prevent monitoring of a real cycle. An example is the core body temperature (CBT) characterized by heat loss at night due to vasodilation in peripheral areas of the skin. The gold standard measurement used to estimate core body temperature is using a catheter. Since catheters are highly invasive, there are acceptable alternatives such as rectal measurements or ear thermometers. [1] Markers that can be measured on an outpatient basis, with non-invasive and comfortable devices for the subject, are gaining prominence as they allow long-term monitoring in real activity conditions. For that reason, different lines of research have been followed. Recent evidence suggests that sleepiness may be more closely linked to increased peripheral skin temperature than to a core temperature drop which seems to be correlated and phase-advanced with respect to CBT, suggesting that heat loss from the extremities may drive the circadian CBT rhythm. [1] The advantage is the possibility of monitoring with a non-invasive technique using thermal sensors located in peripheral areas of the skin. For that reason, the project follows the research line of monitoring the wrist temperature (WT) which is approximately 1 hour ahead of the CBT rhythm, being able to predict changes in core temperature from peripheral temperature data. Figure 2 shows how the CBT falls during the night hours and increases during daylight hours. Meanwhile, the distal skin temperature begins to increase around the early hours of the night, reaching the maximum when the user is fully asleep, where it remains constant until initiates to wake up, and going down to its minimum value during the day.

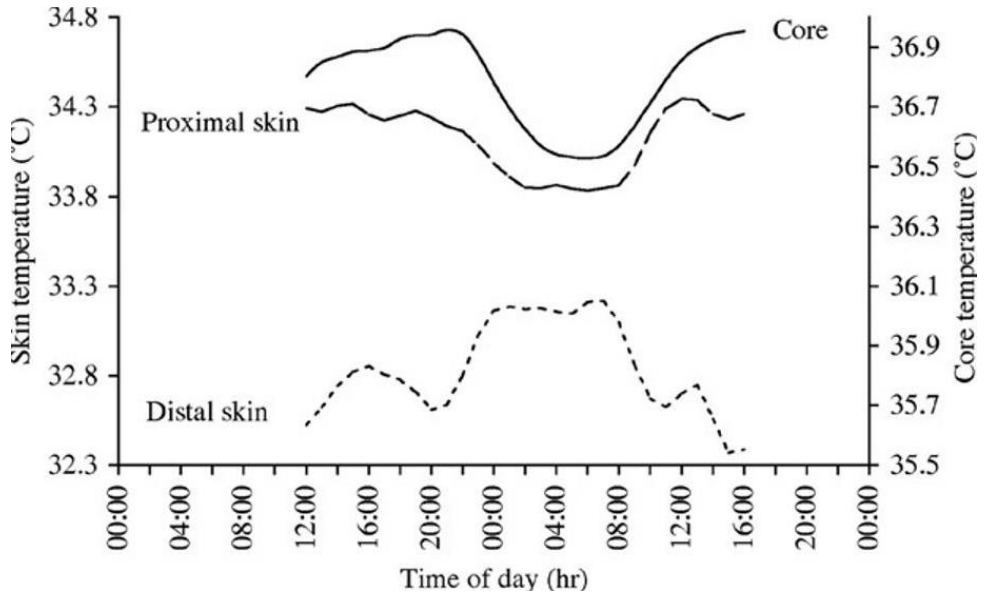

Fig. 2 Nocturnal variation of the core/distal skin temperature

However, the rhythm could be affected by exogenous factors such as ambient temperature or an irregular sleep cycle. Therefore, if the user presents an alteration of the sleep cycle, the temperature monitoring will depict peaks that will be related to that disturbance. These factors that influence both CBT and WT are physical activity, body position, light exposure, ambient temperature, etc.

2.2 Technology review

There are different commercial devices on the market for monitoring one or more vital signs such as body temperature, heart rate or blood oxygen saturation level. In this project, the different devices have been analysed, but first, a review of the types of existing temperature sensors has been carried out to understand why the selected sensor is adequate for the objective of this project.

A temperature sensor is a device, usually, a thermocouple or resistance temperature detector which provides temperature measurement in a readable form through an electrical signal. The most common types are the following:

- Resistance Temperature Detectors (RTD). It consists of a length of fine wire wrapped around a ceramic or glass core. RTD changes the resistance of the element by the temperature changes. They are used because offer a near linear response to temperature changes, are stable and accurate which provides accurate and repeatable responses and have a wide temperature range. However, they have disadvantages as they are slower, require an excitation current, and require signal conditioning.

- Thermocouples are the most commonly used type of temperature sensor. Thermocouples are self-powered, require no excitation, can operate over a wide temperature range, and have quick response times. They are made by joining two dissimilar metal wires together, hence based on the Seebeck Effect which is a phenomenon that a temperature difference of two dissimilar conductors produces a voltage difference between the two metals. The voltage difference is measured and used to calculate the temperature. The disadvantages of thermocouples include the fact that measuring temperature can be challenging because of their small output voltage, which requires precise amplification, susceptibility to external noise over long wires, and correction of the temperature of the cold junction.
- Thermistors are similar to RTDs in that temperature changes cause measurable resistance changes. They are normally made from a polymer or ceramic material. Thermistors are cheaper but less accurate than RTDs. The Negative Temperature Coefficient (NTC) are the most commonly used thermistor for temperature measurement application which its resistance decrease as the temperature increases.
- Semiconductor based ICs are classified in two different types: local temperature sensor, which measure their own die temperature by using the physical properties of a transistor and remote digital temperature sensor, which measures the temperature of an external transistor. Local temperature sensors can use either analog or digital outputs. Analog outputs can be either voltage or current while digital can be seen in several formats such as I2C, SMBus, 1-Wire, and Serial Peripheral Interface (SPI).
- Integrated Circuit (IC) sensors are mostly composed by two identical transistors which operate at different but constant collector current densities, and the difference in their base-emitter voltage is proportional to the absolute temperature of the transistor. The voltage difference is then converted to a single ended voltage or a current. The package is small with a low thermal mass and a fast response time. The most common temperature range is -55 to 150ºC, and the output can be analog or digital. The analog IC solid state sensor provides an output as a voltage or current that is proportional with temperature without additional circuitry. The digital IC sensor provides an output that has been processed through an integral A-D converter on an IC chip and is ready for input into digital control and monitoring systems. The IC sensor does not require linearization or other circuitry.

For the project, it was decided to use, for simplicity, a master-slave data bus (I2C) that allows controlling all the sensors with only two cables: SDA (Serial Data), a line where the

data is transferred, and SCL (Serial Clock), a line that helps all devices to be synchronized with a clock signal. Therefore, electronic sensors compatible with the standard were needed. Finally, considering all the specifications and for simplicity, an integrated circuit sensor with digital output and serial communication was chosen for the designed system.

Once the temperature sensors have been described, a search for commercial wearables devices that measure wrist temperature have been done. One commercial device is the E4 Wristband (Empatica) [4] that it has been also used for measuring in the project. The device is a medical-grade wearable device that offers real-time physiological data acquisition. The E4 is equipped with sensors to gather high-quality data and enable the measurement of blood volume pulse (BVP) from which heart rate variability can be derived, the measurement of the constantly fluctuating changes in certain electrical properties of the skin, captures motion-based activity, reads peripheral skin temperature and let the user the possibility to tag events during the measurement. The E4 uses as a temperature sensor an infrared thermopile which samples the temperature at 4Hz with an accuracy of 0.2ºC within 36ºC and 39ºC. It has a resolution of 0.02ºC. The price of the E4 wristband is around 1700$.

On the other hand, there are others devices which measures the peripheral skin temperature such as the Celsium [5] a clinically validated and CE marked wearable sensor which samples the body temperature every four seconds to get any small change, and VisiMobile (Sotera) [6], and ambulant vital signs monitor, created to replace punctual checks of vital signs and offer a continuous record of them, for a better monitoring of the patient's condition.

Although it is possible to find a variety of portable devices that continuously measure skin temperature, this project has been based on the design of a new device that allows the simultaneous measurement of temperature and physical activity to find out if there is a relationship between both variables, together with other advantages such as ambient temperature changes cancellation and long autonomy life, with a lower cost than the devices currently on the market.

3 Device Improvements

The section describes the methodology followed to make a usable prototype to accurately measure skin temperature. The development has been divided in three parallel parts, hardware, software and calibration parts. Each part presents a research line which their specific objectives.

3.1 Hardware

For the hardware part, it has been started with a previous design and tried to improve it. For the design, a study on the last project has been done to know the ergonomic requirements to develop an accurate wearable product. The aim of the design was to guarantee a precise contact sensor-skin with the user. The design consists of two main parts: the case, in which the associated electronics will be located, and the strap, which will secure the case to the user's wrist. The case is composed by the main body where the PCB with all the electronics, the ambient temperature sensor and the strap anchors are located; and the lid, to which a metallic conductive piece is added to transmit the heat from the user's skin to the peripheral temperature sensor.

Fig. 3 Final box design

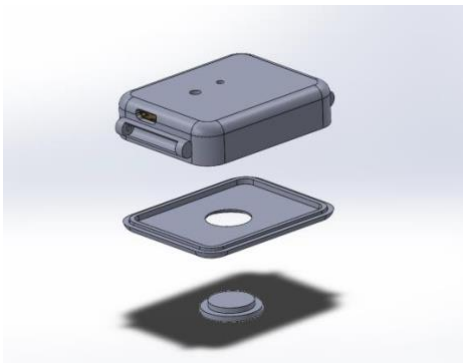

Fig. 4 Exploded view final box design

The final design, shown in Figure 3, consists of a case with dimension 36.8 mm x 47.82 mm x 13.33 mm which provides a comfortable design for the user. The box allows the PCB to be connected via micro-USB to the computer from a slot centred on one of the side faces. The USB connector is used to upload the code, download the recorded data and charge the internal battery. Besides, it presents two holes on the upper face, one to allow the passage of light from an LED indicating the status of the device, and another that allows the ambient temperature sensor to capture the outside temperature more precisely. In addition, internal structures were added to help fix the different electronic components in place, so that it would not move freely inside the box.

Fig. 5 Smartwatch strap

The design of the strap anchor is suitable for the commercial standard 20 mm wide watch straps that are fixed by a pin as shown in Figure 5. The box was printed in 3D using PLA. The lower cover has a projection that allows it to be attached to the box by pressure, and includes an aluminium conductive disk which facilitates the transfer of heat from the user's skin to the temperature sensor

The electronics used to achieve the goal of the project was the designed in a previous project as it is mentioned before. Here is a short summary of the electronic component's interplay. The block diagram of the hardware electronics is represented in Figure 6.

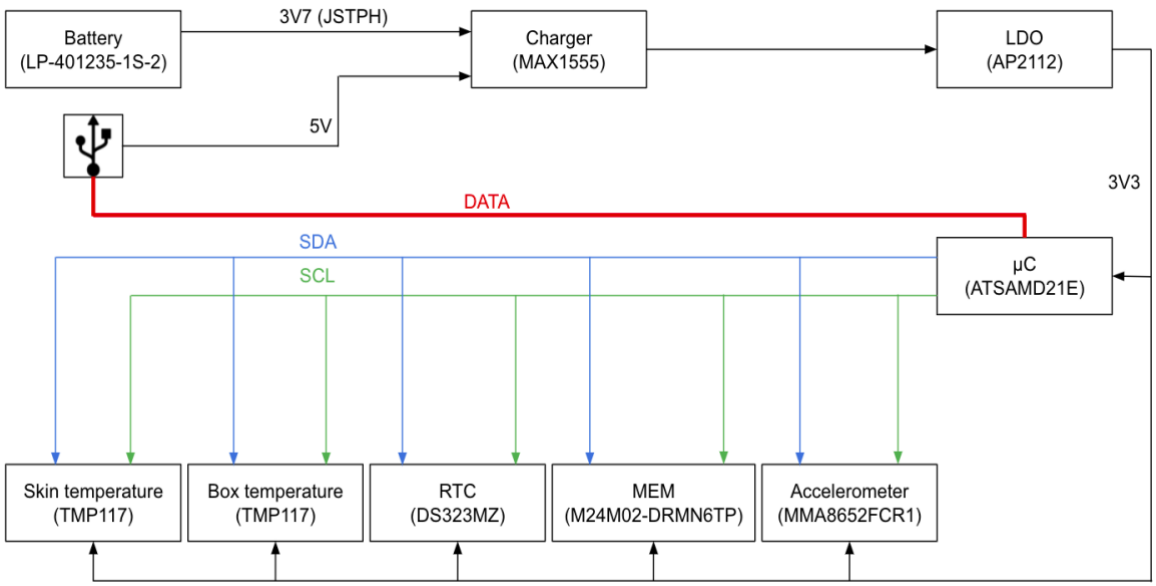

Fig. 6 Block diagram

The components needed to perform the measurement of the temperature are two high-precision digital temperature sensors (TMP117), to measure skin and ambient temperature, a real-time clock (DS323MZ) to measure the time when obtaining data, an accelerometer (MMA8652FCR1) to measure the user activity, an EEPROM (M24M02-DRMN6TP) to save the data obtained from the temperature sensors and accelerometer and a low-power microcontroller (ATSAMD21E) which controls each component to perform its function. On the other hand, it is needed a USB port for communication between the hardware, the computer and the power supply. As the system is needed to work autonomously, an external battery and charger are required. Finally, to guarantee a continuous voltage to the components a low-dropout linear regulator (AP2112) is included.

The selection of each component has been arranged using the datasheets offered by each manufacturer and the needs of the system which are explained below:

- TMP117 is a high-accuracy temperature sensor with low power consumption which minimizes the impact of self-heating. It provides a 16-bit temperature result with a resolution of 0.0078ºC and an accuracy up to +/-0.1ºC across the temperature range of -20ºC to 50ºC with no calibration. It operates from 1.7V to 5.5V and typically drains up to 3.5µA with I2C interface compatibility.

- MMA8652FCR1 is an intelligent, low power, three axes accelerometer with 12 bits of resolution. It has +/-2g, +/-4g and +/8g dynamically selectable full-scale ranges with output data rates (ODR) from 1-56Hz to 800Hz. It has a supply voltage from 1.95V to 3.6V and I2C interface supply of 1.62V to 3.6V.
- DS323MZ is a low power consumption I2C real-time clock (RTC) with 236 bytes of battery-backed SRAM. It provides an accuracy of +/-0.432 second/day from -40ºC to +85ºC. The device incorporates a battery input and maintains accurate timekeeping when the power supply is interrupted.
- M24M02-DRMN6TP is a 2Mbit I2C-compatible EEPROM (Electrically Erasable PROgrammable Memory) organized as 256K x 8 bits. It can operate with a supply voltage from 1.8V to 5.5V, over an ambient temperature range of -40ºC to +85ºC.

The previous schematic version of the electronics is shown in Figure 7. At the top are located the components related to the communication between the microcontroller, the computer and the power supply section. At the bottom right there are the components for data collection, the two temperature sensors, the accelerometer and RTC. At the bottom left there are the microcontroller and MEM.

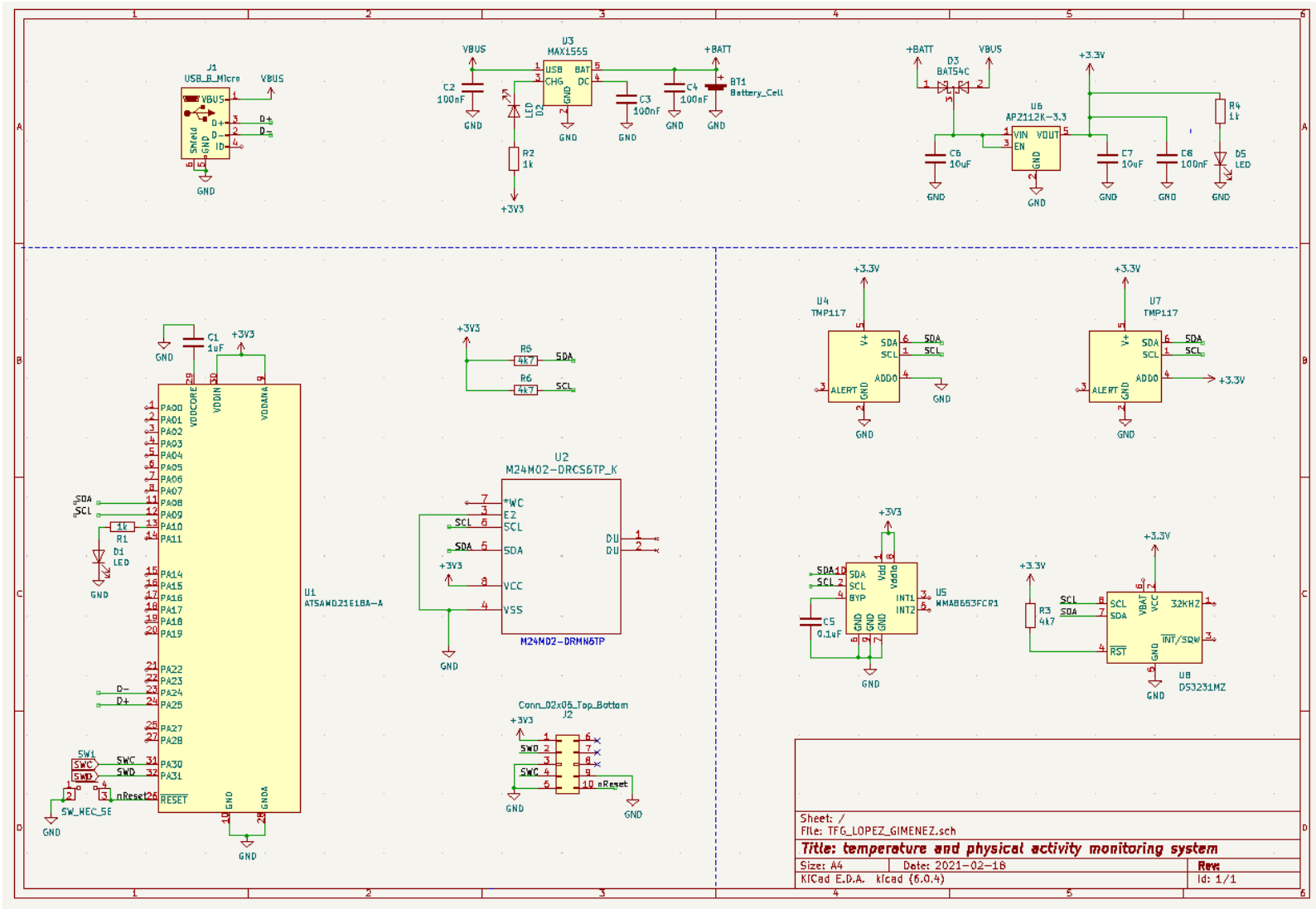

Fig. 7 Previous schematic version

The PCB, shown below, was designed to guarantee a correct connection between the components.

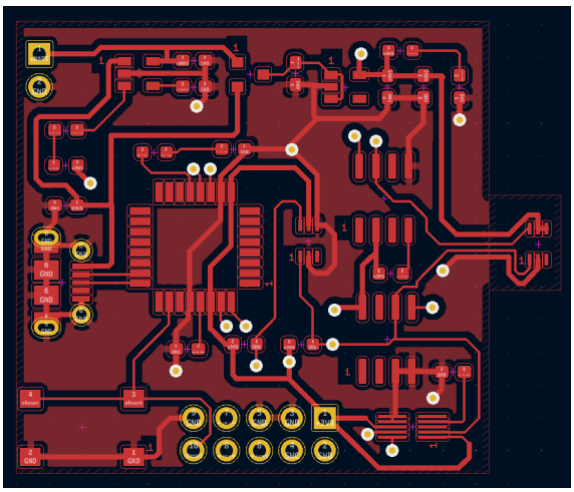

Fig. 8 Front side PCB previous version

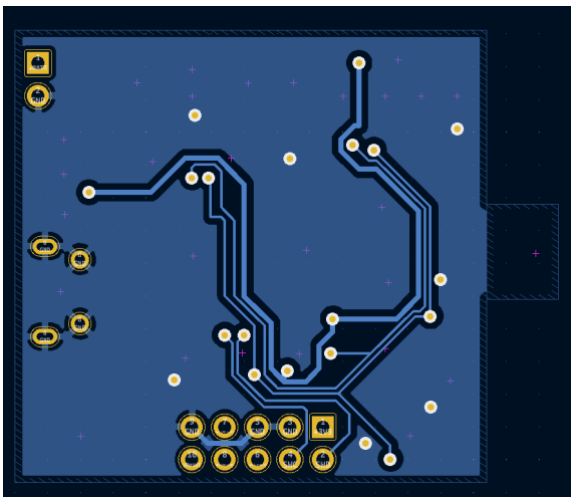

Fig. 9 Back side PCB previous version

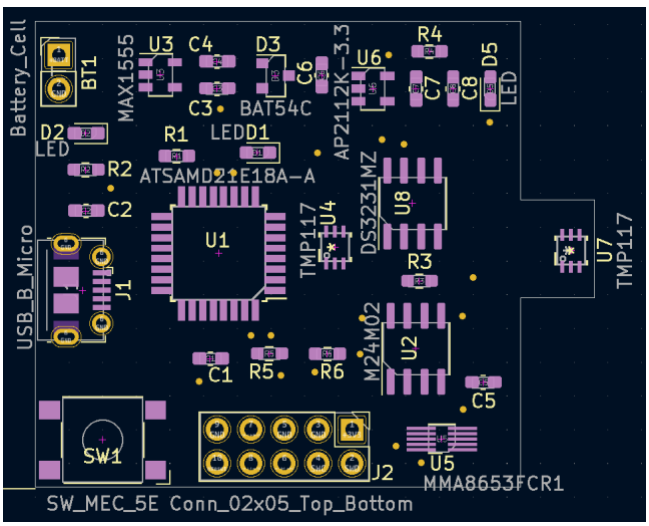

Fig. 10 PCBs components previous version

As can be seen in the Figure 10, the skin temperature sensor (U7) is located independently. This part of the PCB is cut to place the skin temperature sensor on the back of the case where a plate of aluminium, Figure 11, is placed to get a good thermal contact and therefore a good reading. Consequently, some wires must connect the temperature sensor with the microcontroller. In the previous project the wiring system used was a copper wire where each of the sensor terminals were soldered to the specific traces of the PCB.

Fig. 11 Back side

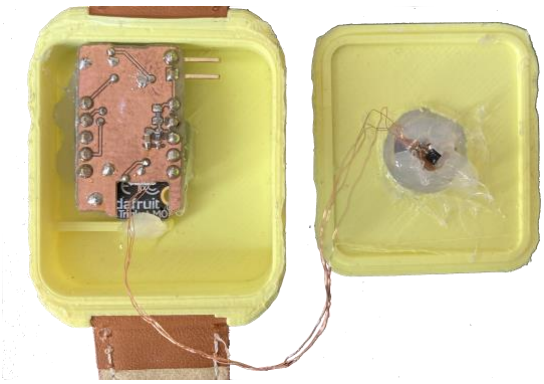

Fig. 12 Original connection system with skin TMP117

The previous wiring system had mechanical problems since the welds were weak and easily breakable due to their lack of flexibility. Therefore, in this project a solution is proposed to replace the wiring with a reliable connector. To solve the problem a premo-flex flat flexible (FFC) cable jumpers has been selected. It has characteristics such as being extra flexible, rated up to +105ºC and has a simple assembly process which makes it perfect for electrical connection between PCB's. The FFC needs specific connectors called FFCs and FPCs to make the connection between both ends. The two options found for FFC connectors were either a vertical mounting angle or a right angle. As the box had a specific height, the connector with a right angle was chosen since it offered more freedom when locating all the components inside the box.

The number of connection lines needed to create the communication between the temperature sensor and the microcontroller is four: the SDA, the SCL, power and ground. Therefore, a 4-position connector is required. Due to stock problems, the specific connector with four position was not available. The width for the 4-position connector selected was 5.20 mm following the rules on Figure 13. since the pitch was 1 mm. Therefore, a connector with a higher number of positions had to be found that would not exceed the width of the 4-position connector. A 6-position connector was found that had a pitch of 0.5 mm, so that, with the proposed formula at Figure 13, the width was 4.7 mm. By respecting the dimensions proposed from the beginning, a Molex connector with reference 538-503480-0600 was selected. It is shown in Figure 14.

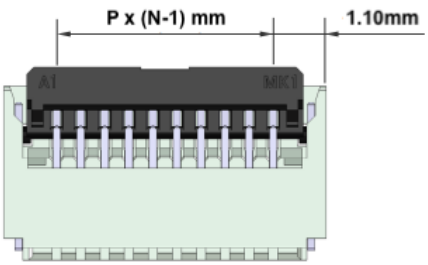

Fig. 13 Width equation

Fig. 14 Connector selected

As a 6-position connector with a 0.5mm pitch was finally selected, a cable with the same characteristics (reference 15020-0051) had to be chosen in order to make the connection correctly.

The change of the connection system caused a modification both in the schematic (see Figure 16) with the addition of the two connectors and in the PCB (see Figure 17) which can be seen in the two figures below.

If zoom is done in on the lower right part of Figure 15, it can be seen the addition of the two connectors: J4 connected directly to the temperature skin sensor (U7) and J3 connected to the corresponding connection lines.

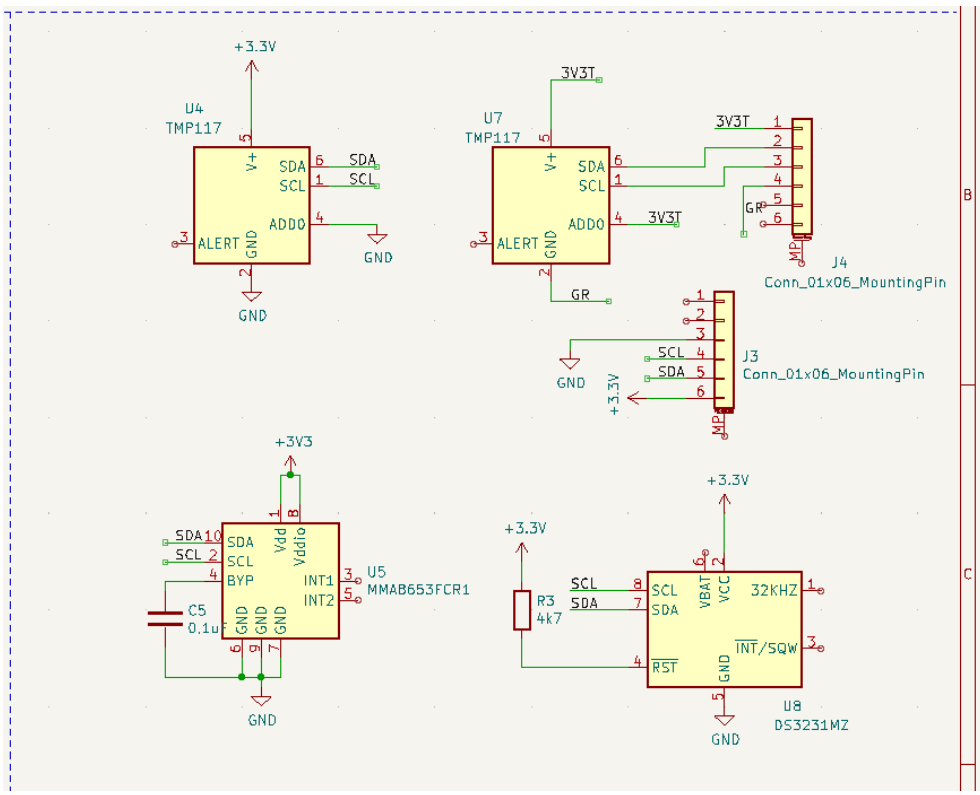

Fig. 15 Zoom of the lower right part of the new schematic version

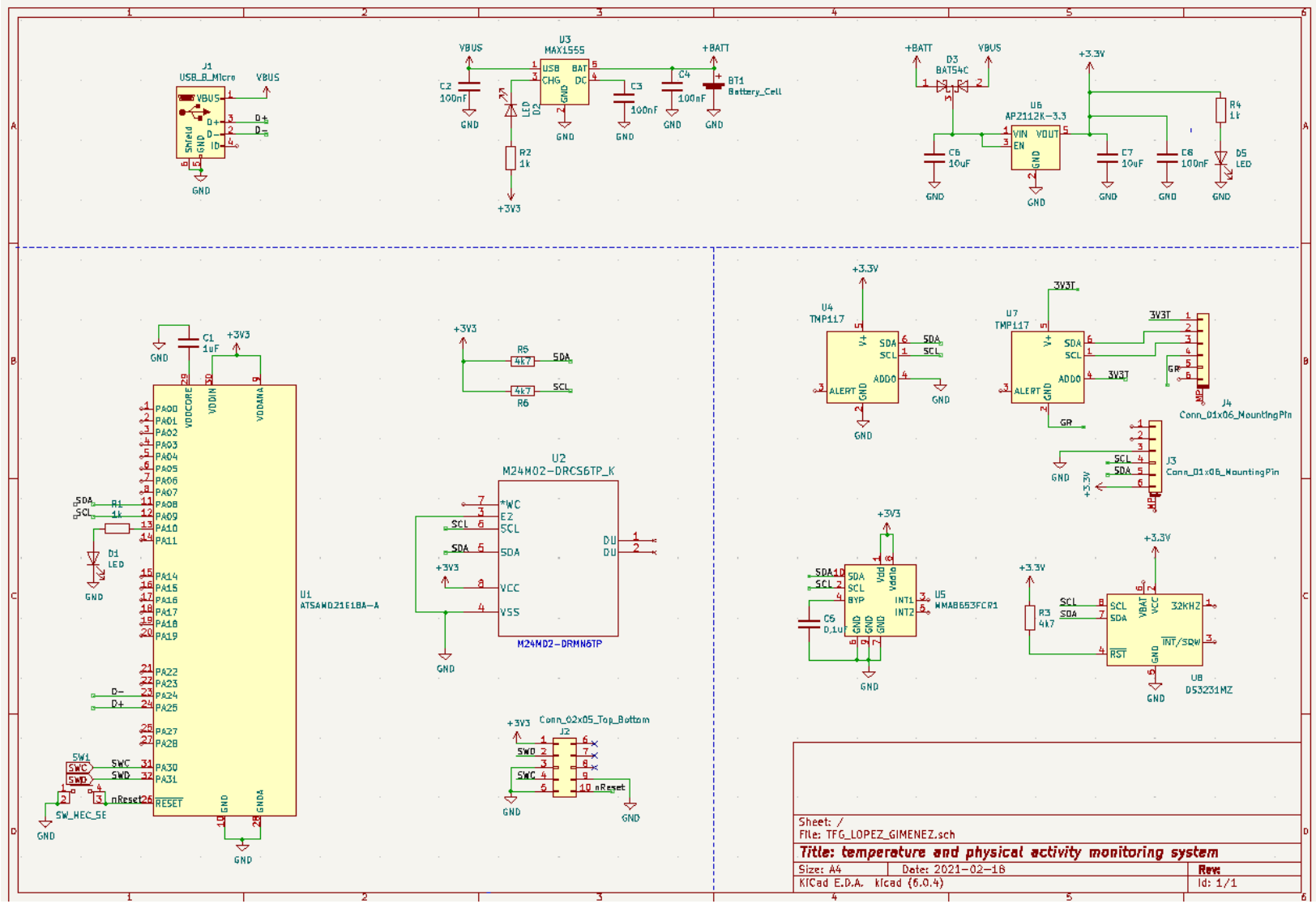

Fig. 16 New schematic version

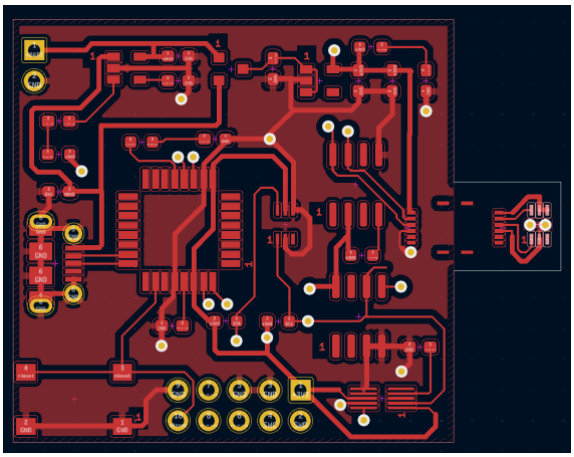

Fig. 17 New front side PCB version

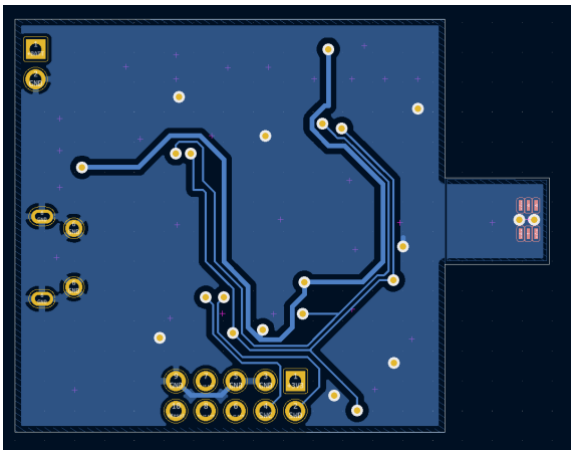

Fig. 18 New back side PCB version

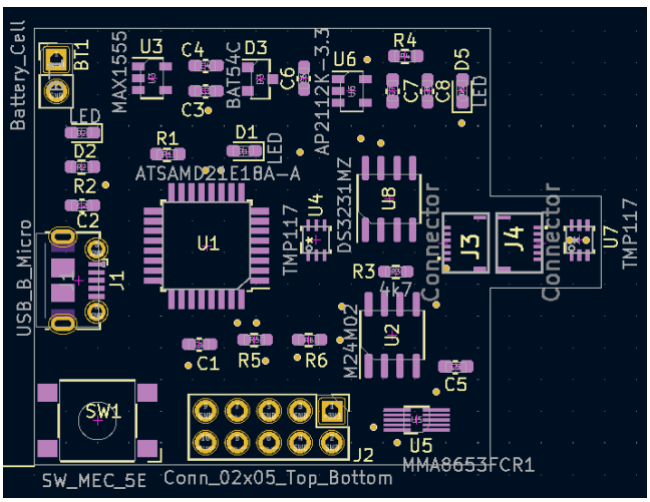

Fig. 19 New PCBs components version

Figure 19 shows at the right side the addition of the two connectors, one on the main PCB (J3) connected to the connection lines and the other (J4) at the secondary PCB connected to the skin temperature sensor (U7).

3.2 Software

The system aims to perform peripheral temperature measurements in volunteers for subsequent processing and analysis of the data obtained. In the same way, as hardware, the software section was not started from scratch but was based on the program made in a previous project. To better understand the program, the flowchart of the preceding project is attached below in Figure 20.

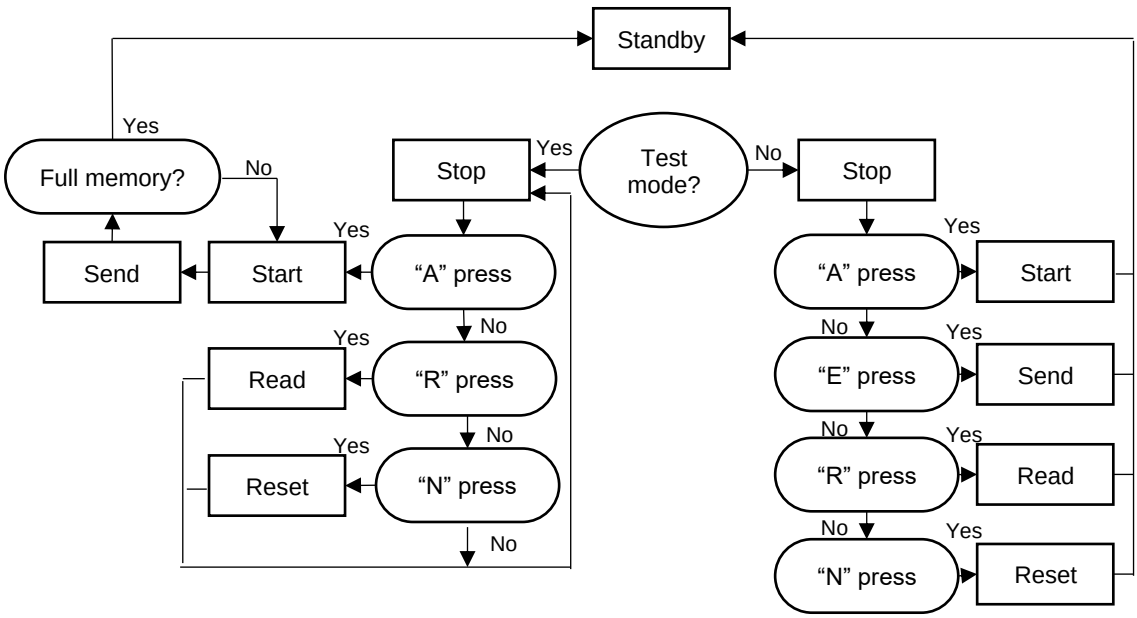

Fig. 20 Previous flowchart

The process begins when the user plugs the device into the computer, where the standby mode is activated. The monitor of the selected serial port shows the menu to choose the action it is wanted to be performed. The software allows three kinds of actions: acquire and store the data, read data and reset the device. In the improved version of this project, functions have been added offering advantages to the user, which will be explained later.

The start action allows the acquisition of the temperature as well as the acceleration every ten seconds, which are sent directly to the memory until it is full, where it returns to standby mode. The measurement can also be interrupted when the battery of the device runs out. In both cases, the device has to be restarted to read the data stored in the memory. After the acquisition of the data ends, the device has to be connected to the computer again to select the read action to obtain the data. The data was stored in pages of 255 bytes separated in blocks of 12 bytes the first and 8 bytes the remaining blocks. At the first block the byte 0 is assigned to the day, byte 1 to the month, bytes 2-3 to the year, byte 4 to the hour, byte 5 to the minute, byte 6 to the seconds, and the next five bytes to the acquisition of temperature and acceleration: Bytes 7-8 are reserved for ambient temperature, bytes 9-

10 for skin temperature and 11 for physical activity since internally the code performed a calculation based on the accelerometer axes and stored a 1 or 0 in memory depending on whether there was movement or not, respectively. The other blocks do not save the date of the acquisition to save space: the byte 0 corresponds to the hour, byte 1 to the minute, byte 2 to the seconds, bytes 3-4 for ambient temperature, bytes 5-6 for skin temperature and byte 7 it is for the activity. Figure 21 shows the distribution of the bytes explained previously.

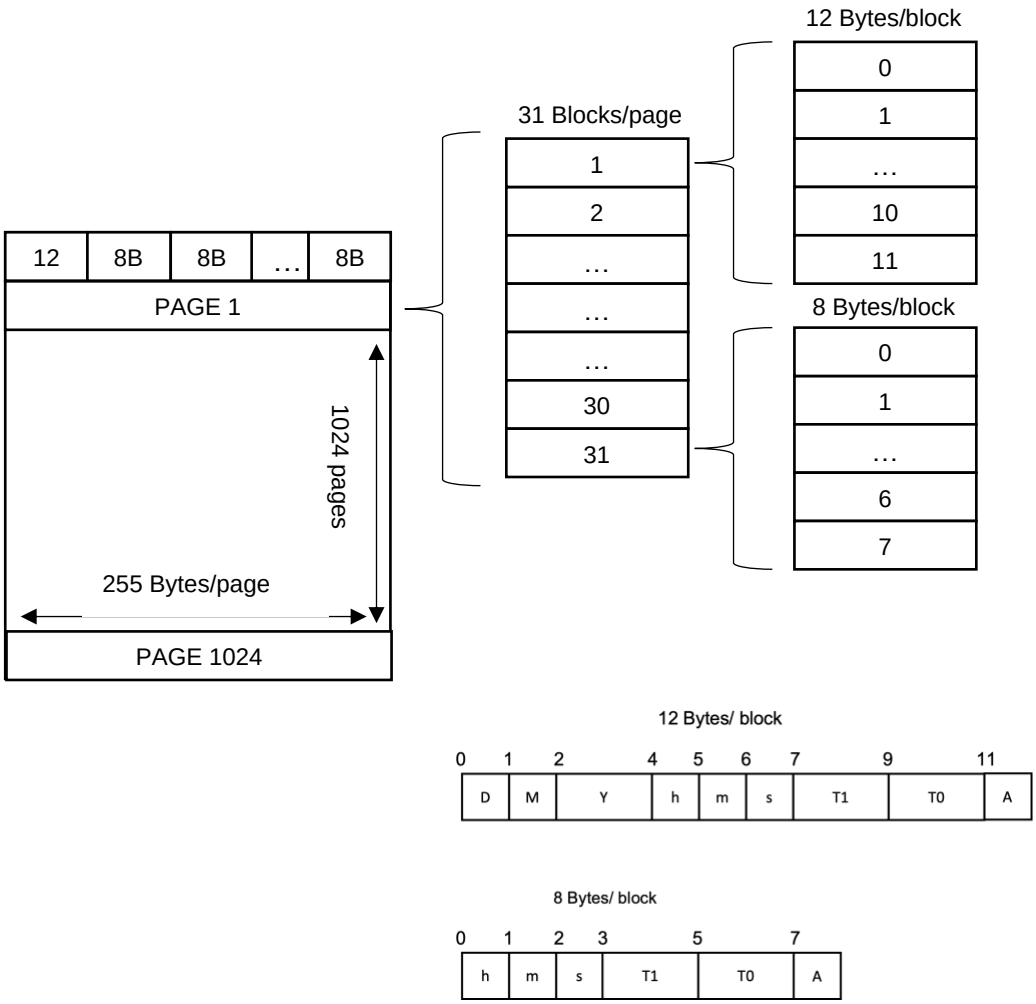

Fig. 21 Distribution of bytes on a page

The read process shows the information acquired from the serial monitor of Arduino. The data is printed in the same format as it is stored in the memory, shown in Figure 22, so it is not possible to directly read the real values of temperature or date and time of acquisition. Post-processing of the data with Matlab is necessary, which creates an excel file where the data is written correctly in units of time and temperature. Once the memory dump is complete, is the moment to reset the system to start a new measurement.

21 6 7 229 13 37 45 9 104 0 0 0 13 37 56 9 104 0 0 0 13 38 7 9 105 0 0 0 13 38 18 9 105 0 0 0 13 38 29 9 106 0 0 0 13 38 40 9 106 0 0 0 13 38 52 9 105 0 0 0 13 39 3 9 103 0 0 0 13 39 14 9 100 0 0 0 13 39 25 9 99 0 0 0 13 39 36 9 97 0 0 0 13 39 47 9 96 0 0 0 13 39 59 9 95 0 0 0 13 40 10 9 95 0 0 0 13 40 21 9 93 0 0 0 13 40 32 9 94 0 0 0 13 40 43 9 93 0 0 0 13 40 54 9 94 0 0 0 13 41 6 9 94 0 0 0 13 41 17 9 93 0 0 0 13 41 28 9 94 0 0 0 13 41 39 9 92 0 0 0 13 41 50 9 92 0 0 0 13 42 2 9 92 0 0 0 13 42 13 9 92 0 0 0 13 42 24 9 91 0 0 0 13 42 35 9 91 0 0 0 13 42 46 9 91 0 0 0 13 42 57 9 90 0 0 0 13 43 9 9 90 0 0 0 13 43 20 9 88 0 0 0

Fig. 22 Reading previous data format. In blue the first block of the page and in green the remaining blocks

The disadvantages found in the previous code were the need to restart the system to change the function, the impossibility of visualizing the data that is being acquired in real time, and the necessary post-processing for the correct reading of the data obtained. Apart from the resolution of these problems, more functions have been added to the system such as the reading of the current date or temperature value.

The process followed in the new version, can be seen in Appendix 1, is the same as in the previous one but improvements are added, that is, to begin with, it is necessary to connect the device to the computer to configure it before placing it on the user's wrist. However, now the number of actions offered by the program is seven: data acquisition and storage, data reading and system reset as before, the acquisition of the current time, the acquisition of the current temperature and acceleration and the possibility of setting the date and time of the system have been added in this project as it shown in Figure 23.

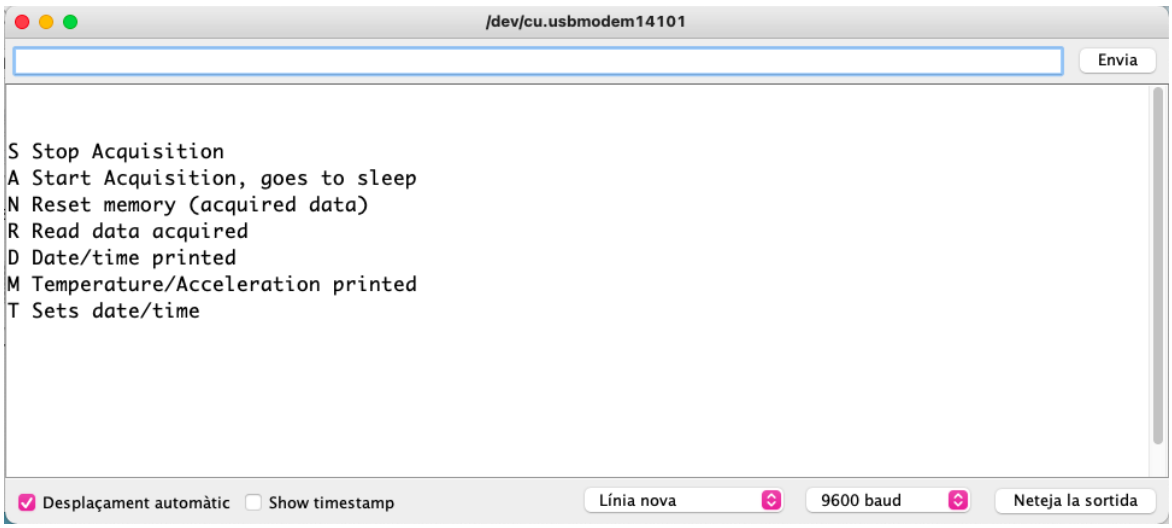

Fig. 23 Main menu of the system

The start state acts in the same way as the previous version, but it acquires the temperature and acceleration data every 10 seconds allowing a more exhaustive and quicker access to the data. Besides, it has been added that while the data is being acquired, if the watch is connected to the computer, the data acquired can be shown on the serial monitor at the current moment, offering the possibility of confirming that the device is acquiring the temperature values correctly. Once it is decided to finish the measurement, it is not necessary to restart the system as previously: there is the possibility to stop the measurement by pressing the 'S' button on the keyboard. As far as reading the data is concerned, it is recived on the serial monitor so that post-processing of the information is not required since the processing has been added to the Arduino code. Therefore, once the data is printed it can be copied to a notepad or excel directly. The new data reading format is shown in Figure 24: first is the date, followed by the acquisition time and then the acquired temperature and acceleration values. As can be seen, the new version offers the three axes of acceleration on the results since it has been considered more convenient for the calculation of the statistics.

The acquired samples of acceleration correspond to the instantaneous values of the axes every 10 seconds. In this way, if the data acquired in 10 seconds are equal to those previously acquired, it implies that the user has not moved, so, no physical activity is detected.

```
4/1/2022 16:54:29 24.25 22.28 248 192 252
4/1/2022 16:54:39 24.28 22.28 248 80 253
4/1/2022 16:54:49 24.30 22.26 247 224 252
4/1/2022 16:54:59 24.32 22.25 249 0 253
4/1/2022 16:55:9 24.34 22.25 249 64 253
4/1/2022 16:55:19 24.35 22.23 248 240 252
4/1/2022 16:55:29 24.37 22.22 248 32 252
4/1/2022 16:55:39 24.38 22.21 248 128 253
4/1/2022 16:55:49 24.39 22.21 248 240 252
4/1/2022 16:55:59 24.41 22.20 248 96 252
4/1/2022 16:56:9 24.43 22.19 247 224 252
4/1/2022 16:56:19 24.43 22.17 248 80 253
4/1/2022 16:56:29 24.45 22.16 248 176 251
4/1/2022 16:56:39 24.47 22.15 248 16 253
4/1/2022 16:56:49 24.47 22.15 247 176 252
4/1/2022 16:56:59 24.49 22.14 247 160 253
```

Fig. 24 Reading data format

The new flowchart with the three new states added is shown below in Figure 25.

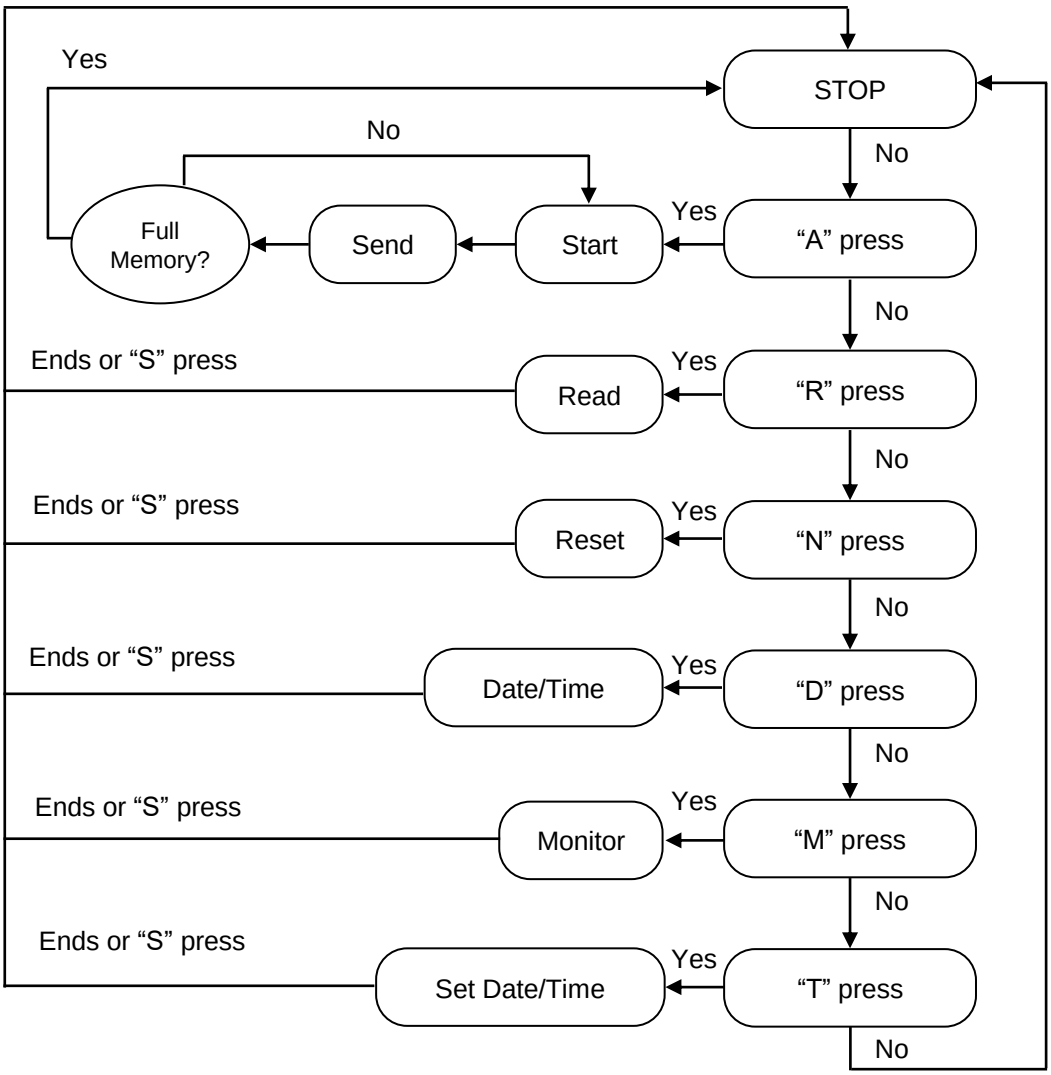

Fig. 25 Current flowchart

The ‘A’ key corresponds to the StartAcquisition function which allows the system to start acquiring the data. The ‘R’ key is used to Read the data acquired. The ’N’ key clears the memory of the system to delete the saved information. The keys added are: the ‘D’ key which corresponds to the Date/Time state that shows the current date and time of the system on the serial monitor as shown in Figure 26, indicating to the user if it is correctly configured. If it was not, there is the ’T’ key which allows setting both the date and time of the system, it is possible to observe the configuration format in Figure 27. Finally, the last key added is the Monitor state (‘M’) which let the system present the current values of temperature and acceleration as in Figure 28. The main difference between the old flowchart and the new version is the ease of ending an action by pressing the 'S' key without the need to reboot the system. The Arduino code is listed in Appendix 1.

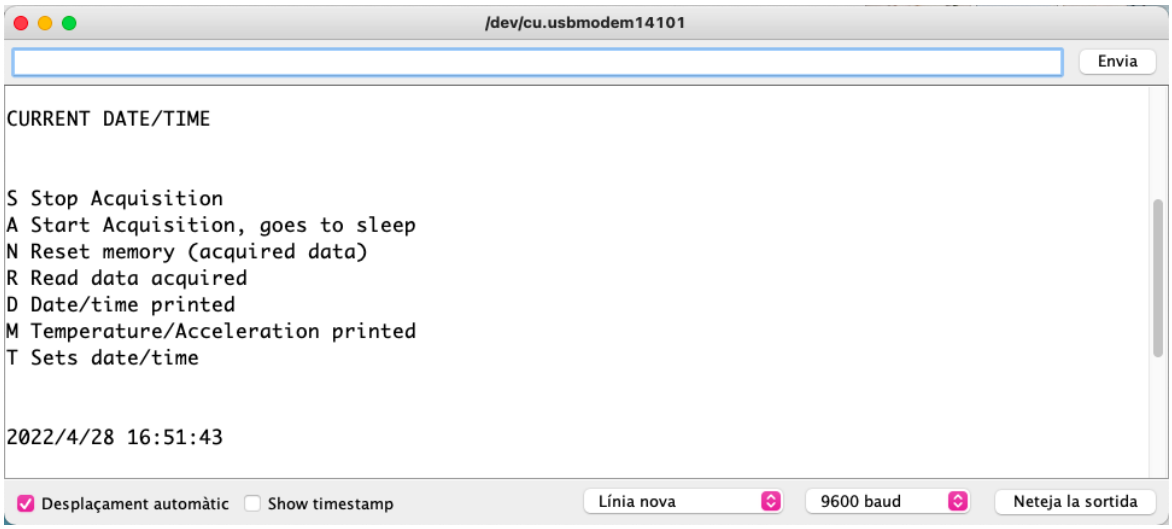

Fig. 26 Date/Time printed state

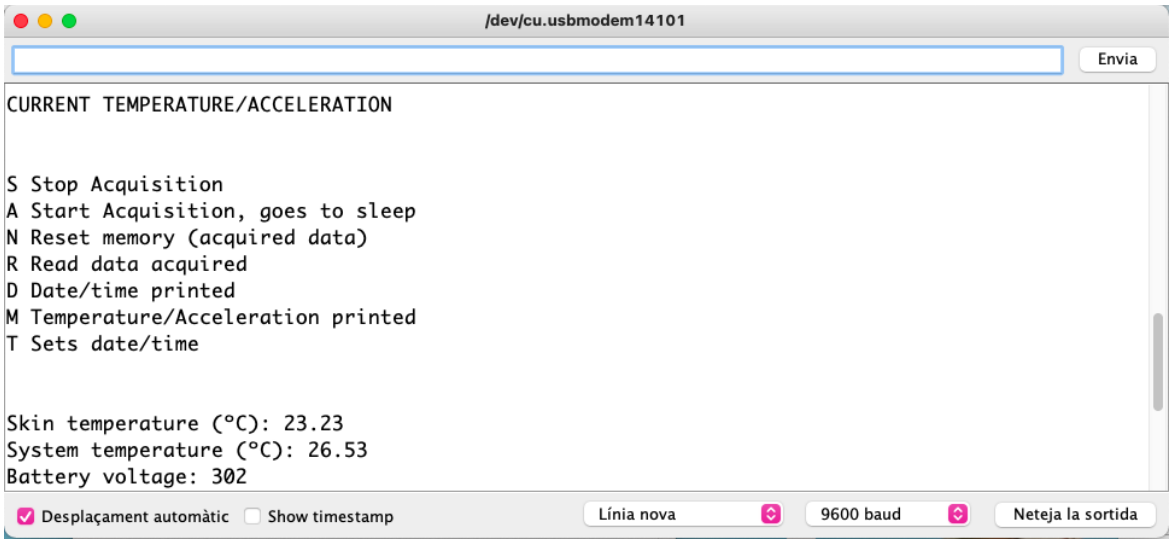

Fig. 27 Temperature printed state

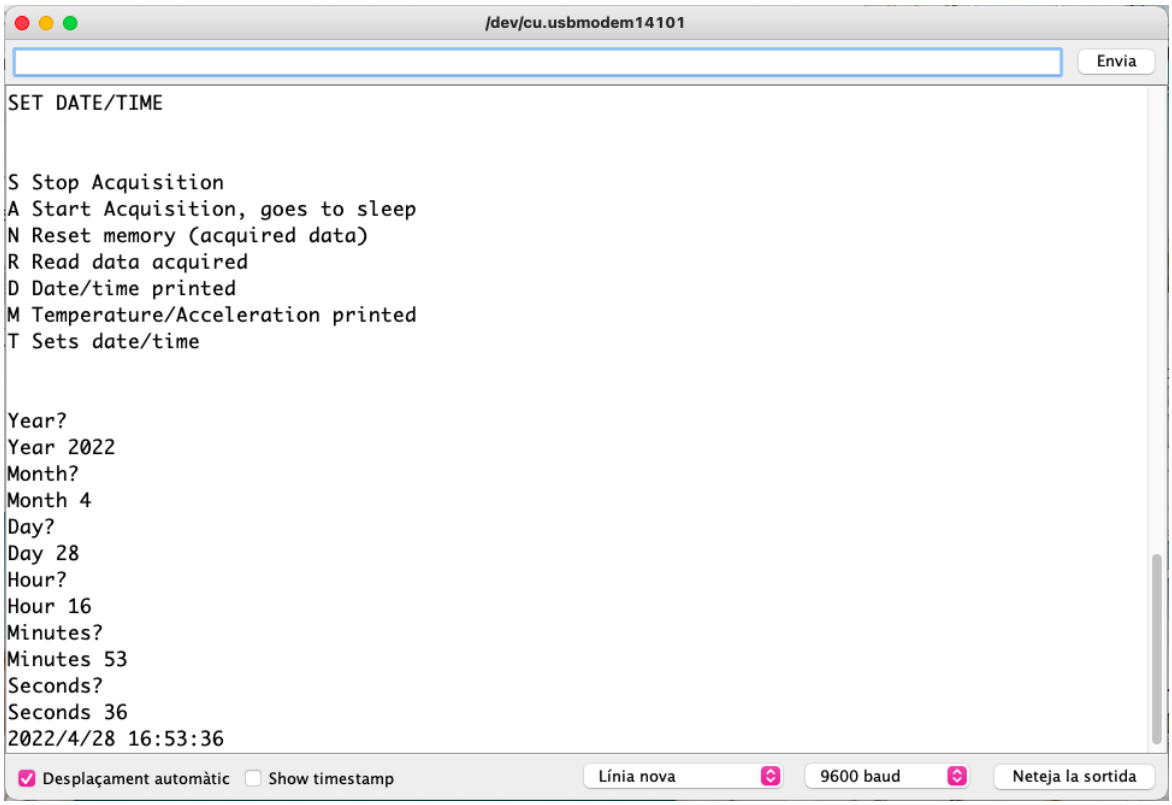

Fig. 28 Set Date/Time state

3.3 Calibration

The calibration system employed, Figure 29, use a controlled power supply which is configured to heat an aluminium plate at the desired temperature. In Figure 30 a plot of the arrangement of the sensors is shown.

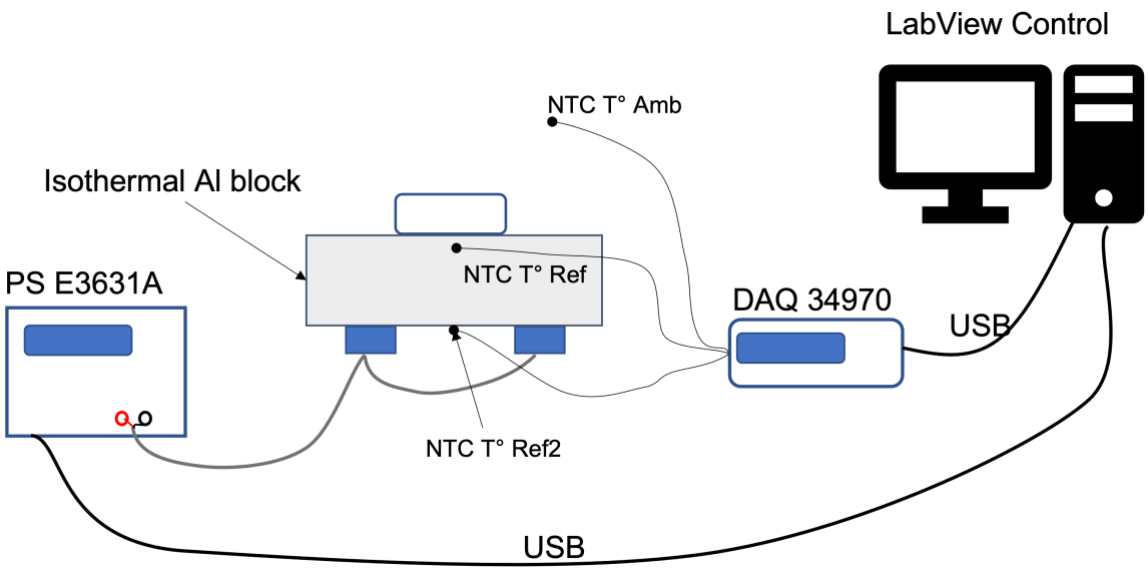

Fig. 29 Calibration system

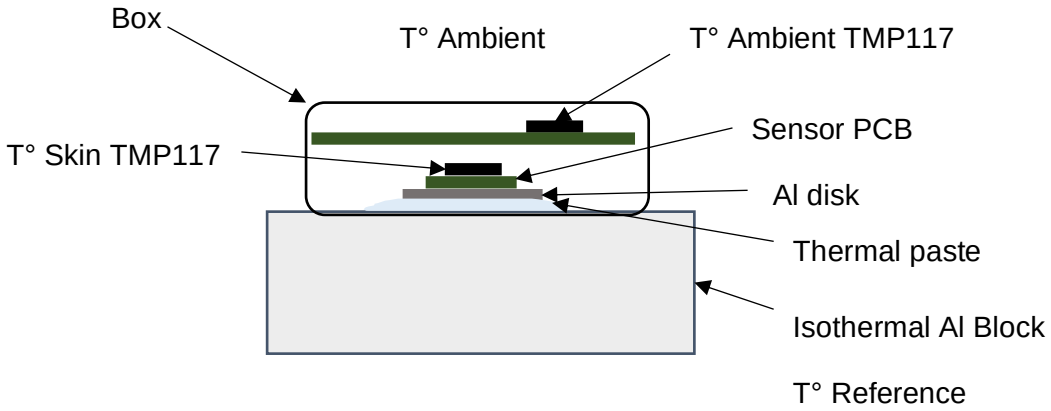

Fig. 30 Scheme of the arrangement of the sensors in the aluminium block

The control of the system has been carried out through a LabView program that allows the user to set the aluminium block at the desired temperature and once the measurement process has started, it accurately acquires the temperature of the aluminium plate and the ambient temperature in periods of 10 seconds using a sensor located outside the block. On the other hand, the designed watch acquires the temperature of the TSkinTMP117 and TAmbientTMP117 sensors with the developed Arduino code. During the measurement process it is possible to visualize how the temperature is evolving both in the LabView program, reference and ambient temperature, and in the Arduino Serial Monitor, TSkinTMP117 and TMPAmbientTMP117. Figure 31 shows the block diagram of the calibration system.

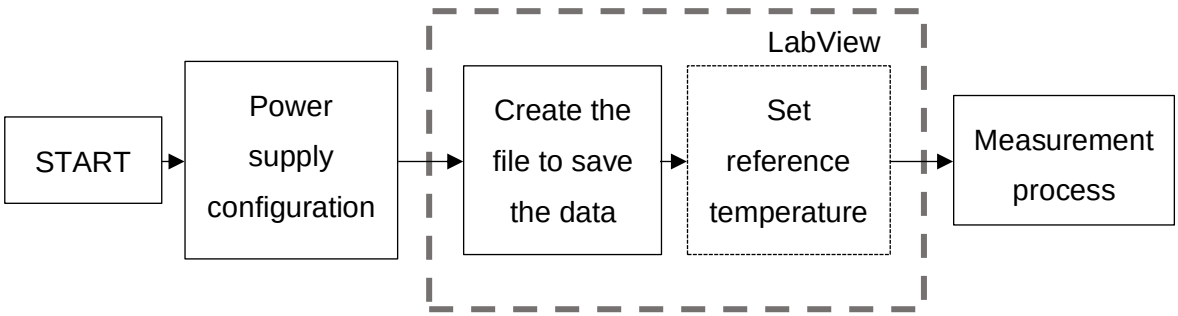

Fig. 31 Block diagram of calibration system

The model to make the calibration that is based on the arrangement of the sensors in the aluminium block is shown in Figure 32.

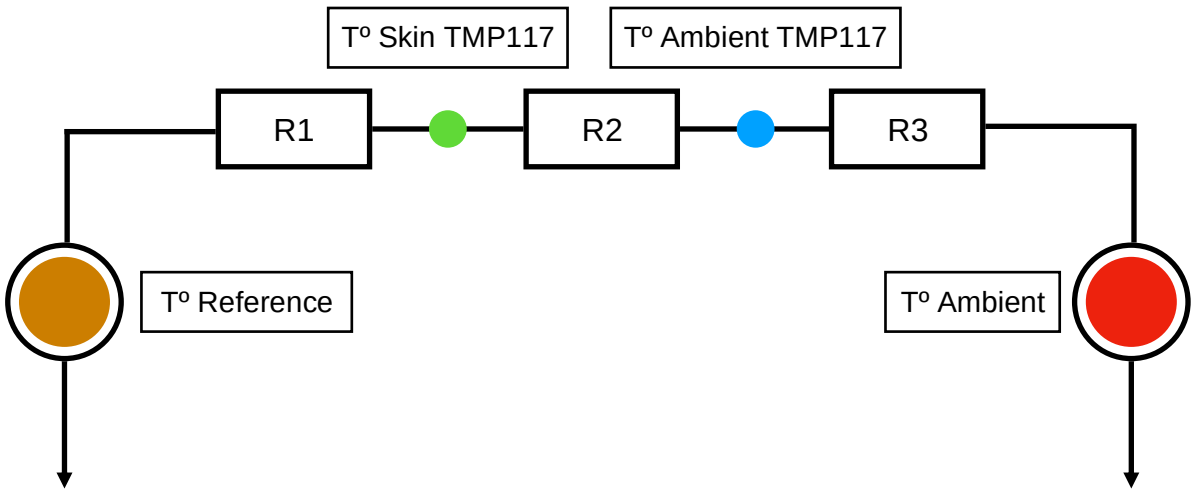

Fig. 32 Calibration system scheme

The diagram shows the thermal model that the calibration procedure intends to estimate. R1 is the thermal resistance between the aluminium plate and the sensor for the skin temperature using the TMP117. R3 is the thermal resistance between the ambient temperature and the ambient temperature sensor of the system designed for the project. Both R1 and R3 are intended to be low. Moreover, R2 is the thermal resistance between skin and ambient temperature sensors of the system designed. Ideally, it should be infinite but a change in ambient temperature could affect the skin temperature measurement through R2 which is measured by TAmbientTMP117.

$$T_{SkinTMP117} = T_{ref}\frac{R_2 + R_3}{R_1 + R_2 + R_3} + T_{ambient}\frac{R_1}{R_1 + R_2 + R_3}$$

$$T_{AmbientTMP117} = T_{ref}\frac{R_3}{R_1 + R_2 + R_3} + T_{ambient}\frac{R_1 + R_2}{R_1 + R_2 + R_3}$$

The calibration has been done with two measurements. In the first measurement, a constant reference temperature (Tref) of 37ºC was used to observe how changes in ambient temperature affected the SkinTMP and AmbientTMP sensors. Therefore, the slope of the graph Ambient Tº vs. TºAmbientTMP corresponded to $\frac{R_1+R_2}{R_1+R_2+R_3}$ and the slope of the TºAmbient vs. T°SkinTMP corresponded to $\frac{R_1}{R_1+R_2+R_3}$. For the second measurement, a constant ambient temperature of around 28ºC was used and it was studied how variations in the reference temperature affected the SkinTMP and AmbientTMP sensors. In this case, in periods of half an hour, the reference temperature was forced to 38ºC, 39ºC and 40ºC.

The slope of the graph TºReference vs. TºAmbientTMP corresponded to $\frac{R_3}{R_1+R_2+R_3}$ and the slope of TºReference vs. T°SkinTMP corresponded to $\frac{R_2+R_3}{R_1+R_2+R_3}$.

3.3.1 Measurement at constant reference temperature

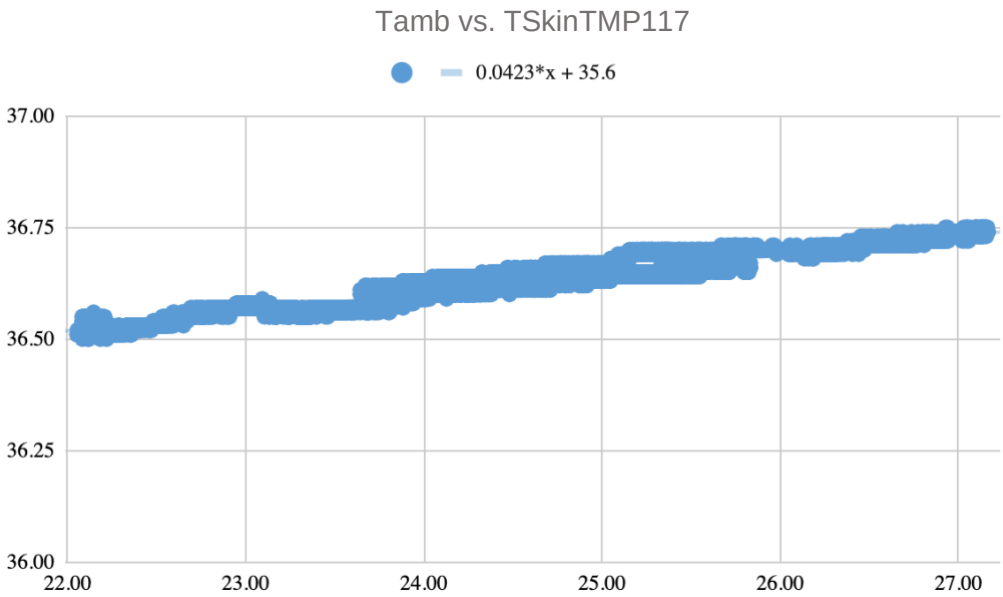

Fig. 33 Tamb vs. TSkinTMP117 at constant reference temperature

$$0.0423 = \frac{R_1}{R_1 + R_2 + R_3}$$

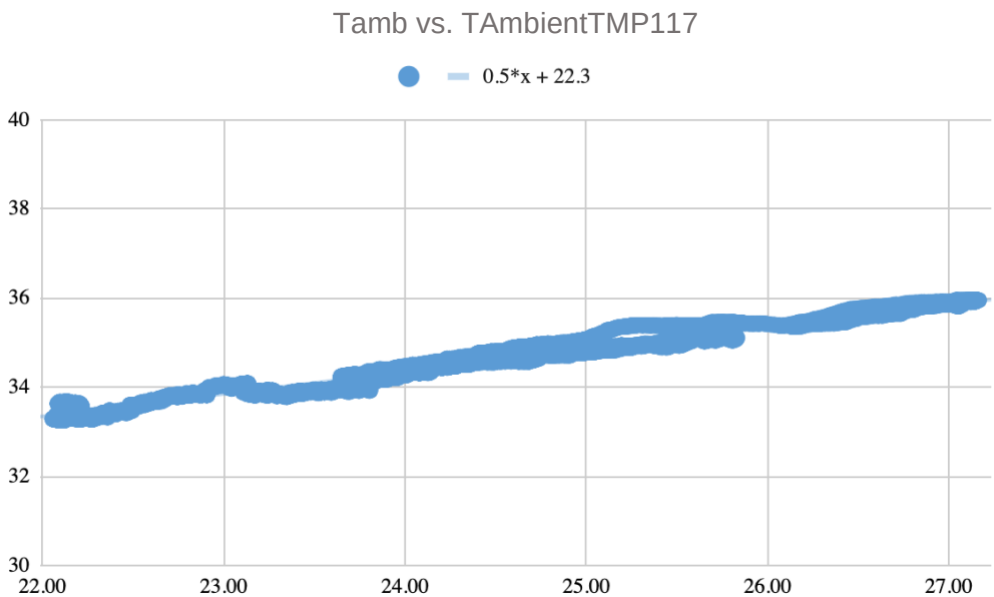

Fig. 34 Tamb vs. TAmbientTMP117 at constant reference temperature

$$0.500 = \frac{R_1 + R_2}{R_1 + R_2 + R_3}$$

3.3.2 Measurement at constant ambient temperature

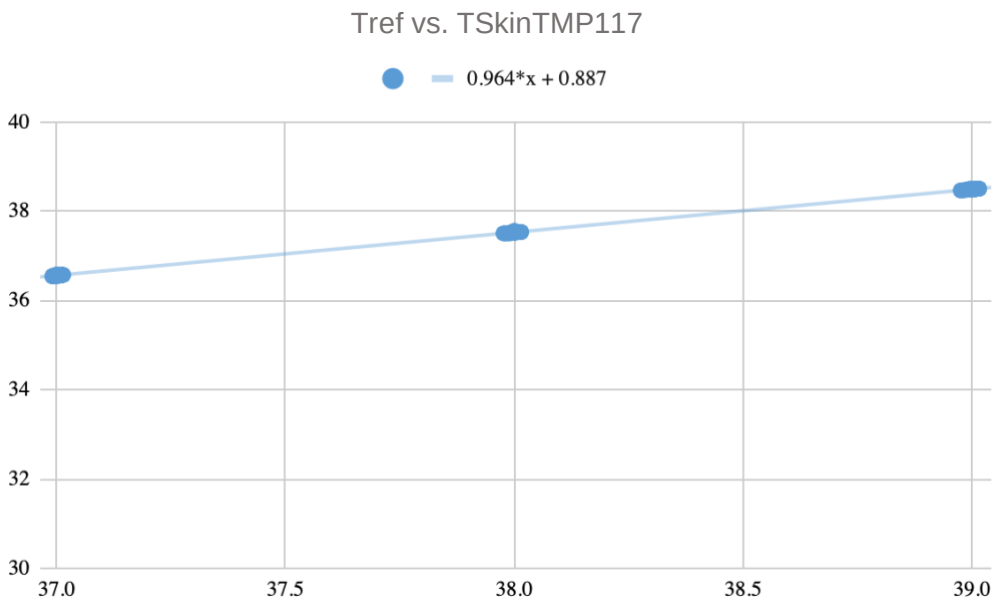

Fig. 35 Tref vs. TSkinTMP117 at constant ambient temperature

$$0.964 = \frac{R_2 + R_3}{R_1 + R_2 + R_3}$$

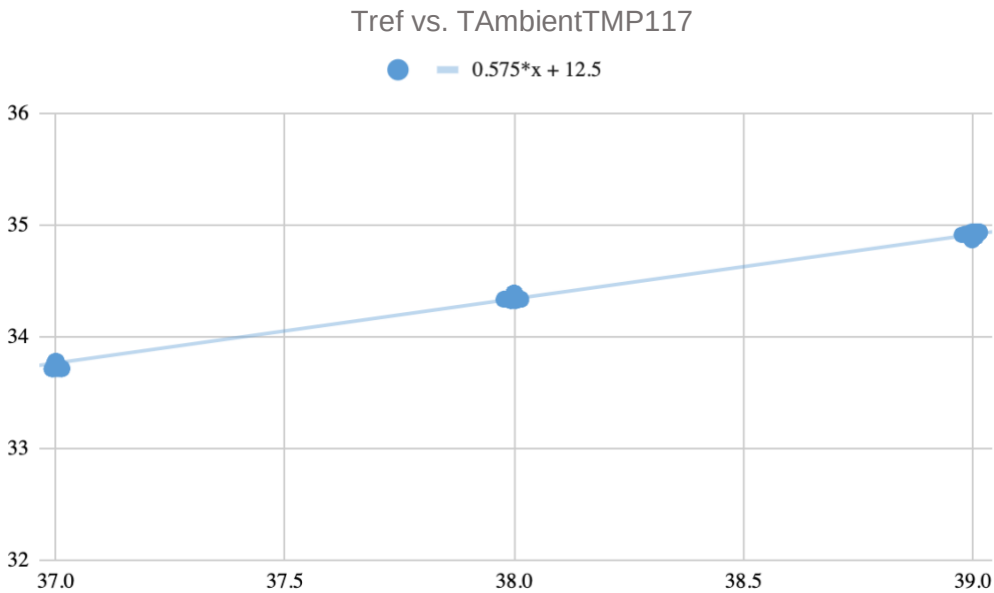

Fig. 36 Tref vs. TAmbientTMP117 at constant ambient temperature

$$0.575 = \frac{R_3}{R_1 + R_2 + R_3}$$

3.3.2.1 Calibration equations

Once all the slopes of the graphs are obtained, these values must be substituted in the main equations to obtain a final equation where, from SkinTMP and AmbientTMP, TReference can be calculated, a value that refers to the real skin temperature value (Tref).

$$T_{AmbientTMP117} = T_{ref}\frac{R_3}{R_1+R_2+R_3} + T_{ambient}\frac{R_1+R_2}{R_1+R_2+R_3}$$

$$T_{SkinTMP117} = T_{ref}\frac{R_2+R_3}{R_1+R_2+R_3} + T_{ambient}\frac{R_1}{R_1+R_2+R_3}$$

$$0.500 = \frac{R_1+R_2}{R_1+R_2+R_3} \quad 0.042 = \frac{R_1}{R_1+R_2+R_3}$$

$$0.575 = \frac{R_3}{R_1+R_2+R_3} \quad 0.964 = \frac{R_2+R_3}{R_1+R_2+R_3}$$

$$T_{SkinTMP117} = T_{ref}0.964 + T_{ambient}0.0423$$

$$T_{AmbientTMP117} = T_{ambient}0.500 + T_{ref}0.575$$

The equations are solved for Tref and TAmbient to provide the two calibration equations that depend on the reading of the TMP117 sensors.

$$T_{ref} = \frac{T_{SkinTMP117} - T_{AmbientTMP117}\frac{0.042}{0.500}}{(0.964 - \frac{0.575 * 0.042}{0.500})}$$

From this equation it is possible to obtain the value of the calibrated skin temperature taking into account the factors mentioned above. The same steps have been applied to get the equation to calculate the ambient temperature calibrated.

$$T_{ambient} = \frac{T_{AmbientTMP117}0.964 - T_{SkinTMP117}0.575}{0.500 * 0.964 - 0.042 * 0.575}$$

3.3.3 Implementation of the measurement system

Both equations developed above were applied to the data obtained with the TMP117 sensors during the temperature measurement generated by the power supply controlled at 37 ºC during a period where the ambient temperature presented variations. The following graphs show the actual temperature acquired by the controlled system in orange and the temperature acquired by the TMP117 sensors with the calibration applied in blue. The first graph corresponds to the skin temperature, which presents a small deviation of 0.25 ºC with respect to the real value due to delay in the time response of both sensors generated by a low pass filter. On the other hand, the second graph corresponds to the ambient

temperature, which presents a greater error since the sensor is inside the box where the components are, which are heated during the process.

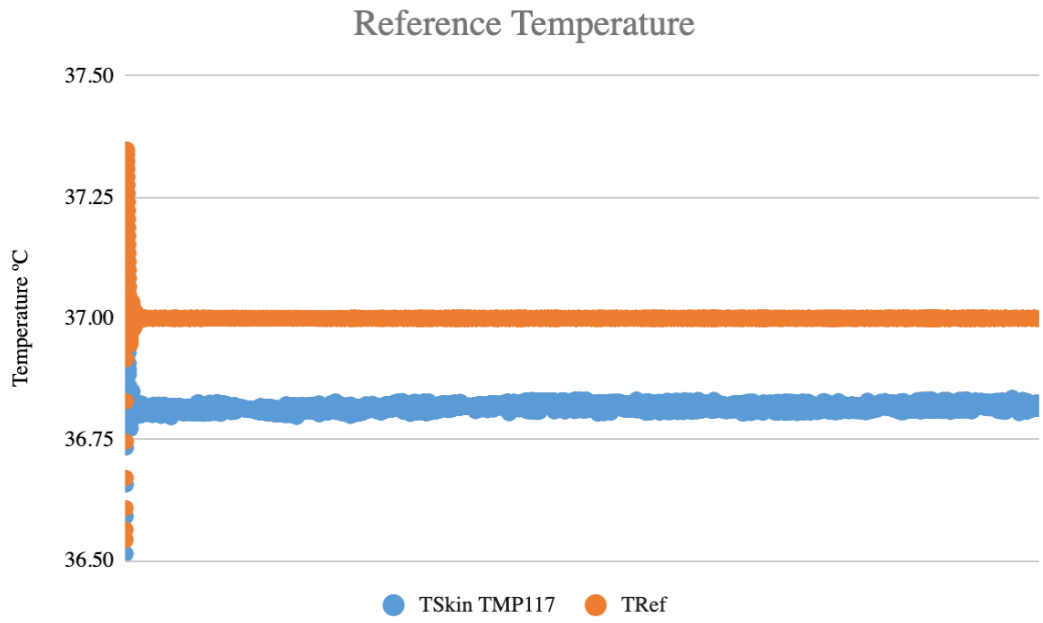

Fig. 37 TSkin obtained from the TMP117 with the calibration applied

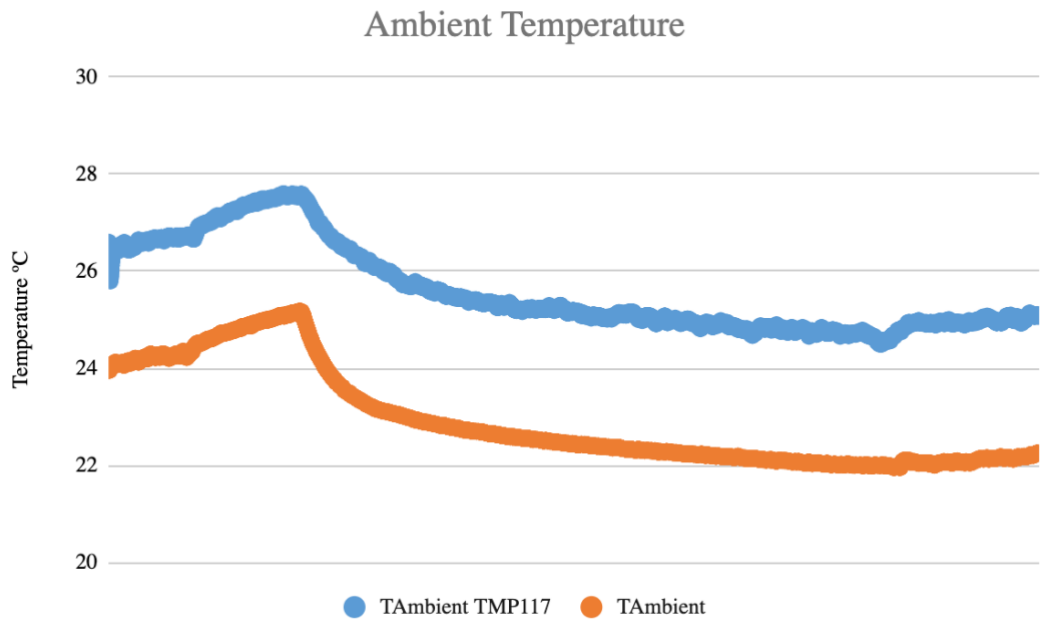

Fig. 38 TAmbient obtained from TMP117 with the calibration applied

The previous results confirm the correct functioning of the developed system. However, to confirm the correct operation of the hardware, software and calibration, a sleep cycle measurement has been carried out to observe the evolution of the temperature and how it is corrected with the calibration system.

The calibration equations have been included in the Arduino code explained above to get the results directly on the serial monitor and avoid the need for post-processing. Therefore, the value stored in the EEPROM is the already calibrated temperature value. But first, to observe the efficiency of the calibration system during a sleep cycle, it had been applied by post-processing. The results are explained in the next section.

4 Sleep cycle measurements and characterization

The first measurements have been made with the Empatica E4 bracelet while the project was being developed, in order to advance in the characterization of the temperature while some components were stalled due to their pandemic shortage. For the design of the system, a completely new prototype was developed where the connector mentioned in the previous section was added. Initially, the addition of the connectors increased the dimension of the PCB, and since the box from the previous project was wanted to be used, the connection lines around the connector were repositioned to avoid expanding the dimension of the new PCB prototype.

4.1 Sleep cycle results

The recording using the E4 wristband were made with 10 different subjects to confirm the behaviour of the temperature, explained in the State of Art section, during a sleep cycle. A questionnaire, added to the Annexes, was passed to all the participants to see if there was a relationship between how the subjects felt how they had slept and how it was shown in the temperature and acceleration graphs. The most relevant are related to the subject's ease of sleeping, whether they woke up during the night, and how they felt the next morning when they woke up.

Fig. 39 Sleep cycle measured with E4 wristband of Empatica

As seen in Figure 39, the temperature of the distal skin begins to increase around the first hours of the night, reaching a maximum when the user is fully asleep, where it remains constant until initiates to wake up, and falling to its minimum value during the day. The observable variations of temperature during the sleep cycle in Figure 35 have a value of less than 0.5ºC, so it can be said that the user of this measure has a high sleep quality

cycle. On the other hand, Figure 40 shows a sleep cycle of a user who has not slept properly and since the temperature has fluctuated up to 2ºC. The skin temperature begins to increase in the early hours of the night, reaching the maximum value when the user is completely asleep, but in this case the temperature does not remain constant. High temperature changes are related to exogenous factors such as waking up during the night, excessive movements, stress...

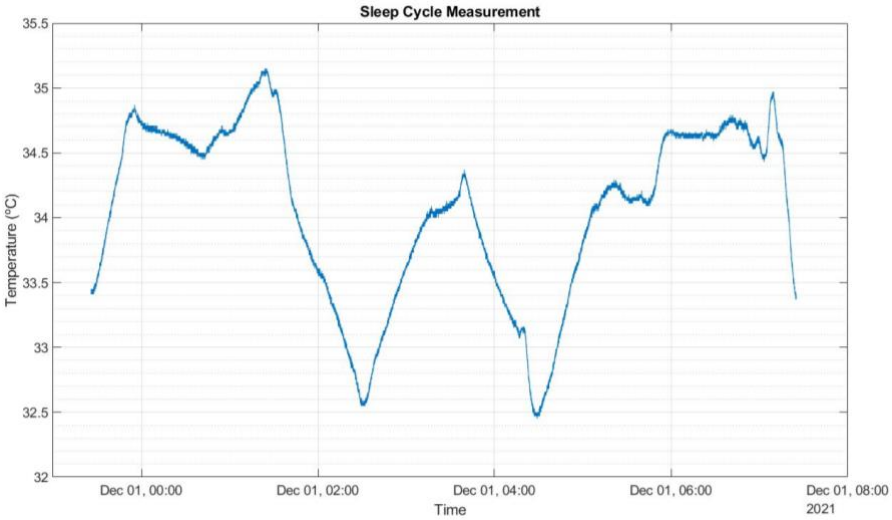

Fig. 40 Sleep Cycle measured by the E4 wristband of Empatica

Once the project system was designed and calibrated, the same measurements were made as with E4 wristband of Empatica, to confirm its correct operation. Figure 41 shows a sleep cycle measured with the designed system. It is observed the same temperature curve, the temperature starts to increase when night begins, reaching the maximum value when the user is completely asleep. The temperature decreases at the wake up of the user.

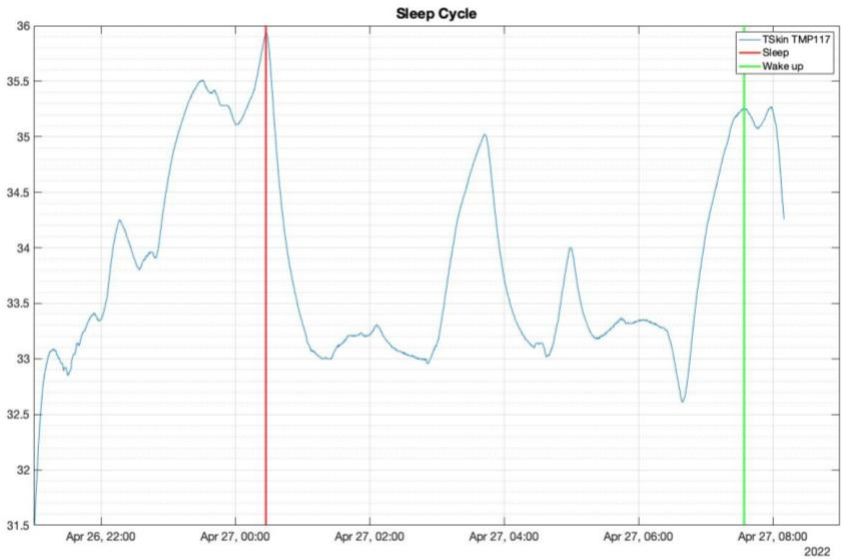

Fig. 41 Sleep Cycle acquired with the system designed

The measured sleep cycle presents temperature changes of around 2ºC. To rule out dependence on ambient temperature, the calibration performed has been applied to the measurement. The specific sleep cycle, from the user is fully asleep to wake up, has been selected from which a constant skin temperature is expected, and the calibration equations have been applied. However, the peaks observed during the sleep cycle are maintained after removing the dependence on ambient temperature. Therefore, temperature changes are due to external factors such as movement, user awakening, stress instead of changes in the temperature of the system. Figure 42 shows the commented variations due to external factors.

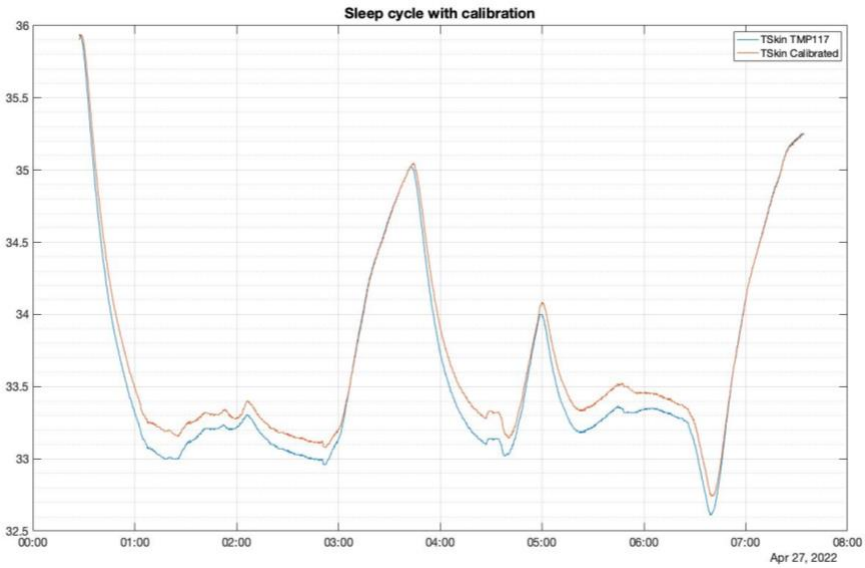

Fig. 42 Sleep Cycle with the calibration applied

As it was concluded that the temperature changes are due to movements, user awakening or stress, the acceleration analysis has been carried out to correlate the information. The acceleration offers information on whether the user moves during the night, which could imply an awakening of the subject. Therefore, if a movement is perceived in the acceleration data and the temperature changes at the same time, it could be concluded that the user woke up or was not sleeping soundly.

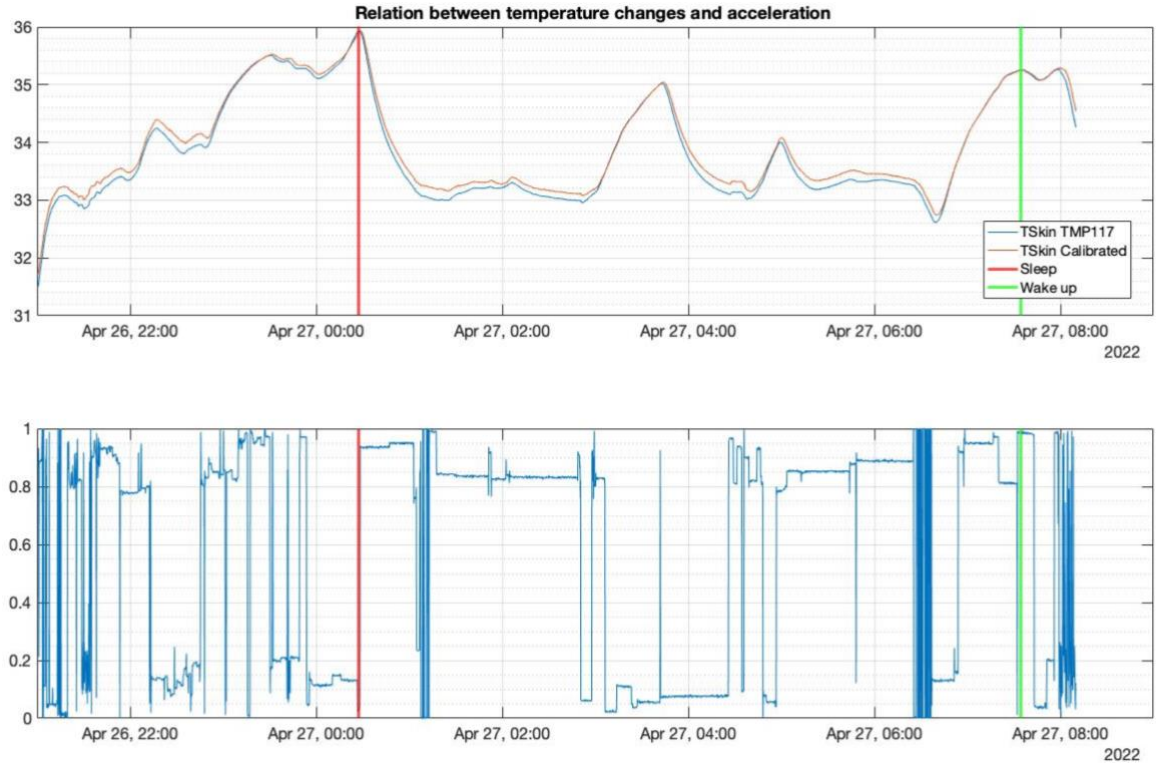

Fig. 43 Simultaneous display of temperature changes and acceleration

A constant value in the acceleration data means that the subject remains still, which results in a good sleep. If an oscillation is observed, it is when the subject begins to move and if a change in temperature is also observed it can be explained by the change in activity. At In figure 43 it can be seen how when a change in slope in temperature is perceived, there is a greater oscillation in acceleration. The displayed acceleration corresponds to the z coordinate. On the other hand, when the temperature remains constant, the acceleration also has a constant value. Therefore, it can be concluded that both variables present some relationship between them.

Although the system offers correct results, due to the project timing the characterization system has been made from the values obtained using E4 wristband of Empatica which is a medical-grade wearable device that offers the acquisition of physiological data in real-time.

4.2 Characterization of the sleep cycle

As it is mentioned on chapter 2: State of Art, if the user sleeps adequately, the curve of temperature must be symmetric since the temperature remains high and stable during sleep. However, if the rhythm is affected by exogenous factors and the sleep cycle presents disturbances, the curve temperature will be asymmetric since it is unstable.

Some statistics have been used to characterize the temperature variations during the sleep cycle. Here is a description of them:

4.2.1 Standard deviation

The standard deviation is a measure of the amount of variation or dispersion of a set of values. A high standard deviation indicates that the values are spread out over a wider range, while a low standard deviation indicates that the values tend to be close to the mean of the set.

Therefore, if the user is fully asleep, the temperature will be constant. As a result, the standard deviation will be low since the set of values will be near the mean. On the other hand, if the sleep cycle presents alteration, the values will be out over a wider range, consequently, the standard deviation will be high.

4.2.2 Skewness

Skewness is the measure of the asymmetry of the probability distribution of a real-valued random variable about its means. The skewness value can be positive, zero, negative, or undefined. A negative skew indicates that the longer tail is on the left side of the distribution shown in Figure 44; the mass of the distribution is concentrated on the right of the figure. While a positive skew indicates that the longer tail is on the right; the mass of the distribution is concentrated on the left of the figure.

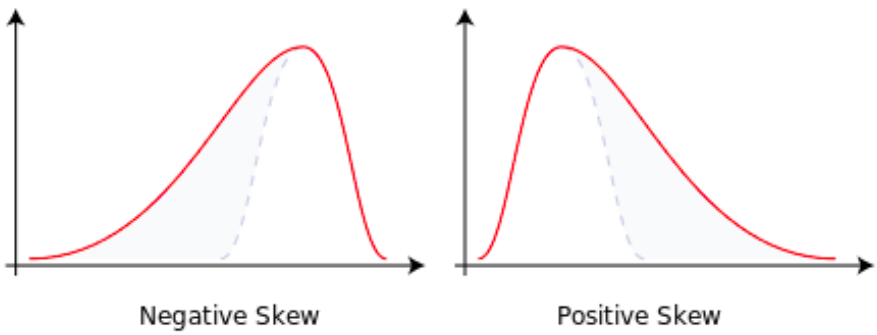

Fig. 44 Types of Skewness

A zero value means that the tails on both sides of the mean balance out overall: the case for symmetric distribution. But can also be correct for an asymmetric distribution where one tail is long and thin, and the other is short but fat.

As is mentioned above, if the user gets a good sleep the curve of temperature must be symmetric. As a result, the skewness value has to be near zero. However, if the sleep cycle is affected by alterations the curve could present peaks in some part of the sleep cycle. In consequence, the value of skewness will depart from zero.

4.2.3 Kurtosis

Kurtosis is a statistical measure used to describe the shape of probability distribution, which measures whether the data are heavy-tailed or light-tailed relative to a normal distribution. A large kurtosis implies a great concentration of values of the variable close to the mean of the distribution, peak, and far from it, tail, while there is a relatively lower frequency of intermediate values.

The kurtosis of any univariate normal distribution is 3, mesokurtic, common value to compare the kurtosis of a distribution. If the value is less than 3 it is platykurtic, the distribution produces fewer and less extreme outliers than does the normal distribution. The shape is more pointed and with thicker tails than normal. Distribution with kurtosis greater than 3 are leptokurtic, produces more outliers than the normal distribution. The shape is less pointed and with tails less thick than normal. The three types of kurtosis are shown in Figure 45.

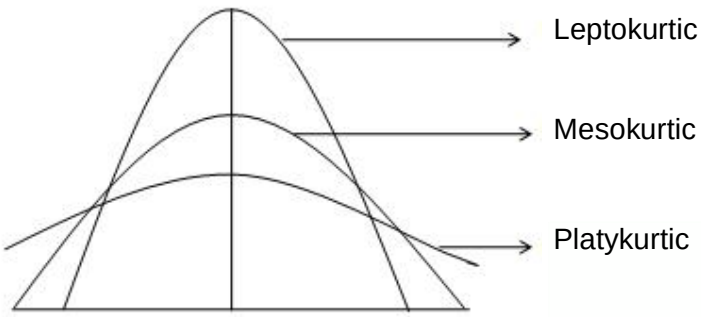

Fig. 45 Types of Kurtosis

The temperature curve presents a different shape when the user sleeps correctly compared to when has an unstable sleep, that is, disturbances can be observed during the sleep cycle. Therefore, it is expected that the value of kurtosis for a correct sleep cycle is different from that of a disturbed sleep cycle.

4.2.4 Characterization results

Below, Figure 46, it can be observed the histogram of the measurements. The histogram allows to analyse whether the temperature has been constant or shows variations. If the data is concentrated around a specific temperature, it means a constant temperature during sleep that reflects good sleep. On the other hand, if the data is spread over a large number of temperature values, it means large temperature variations resulting in disturbances in the sleep cycle.

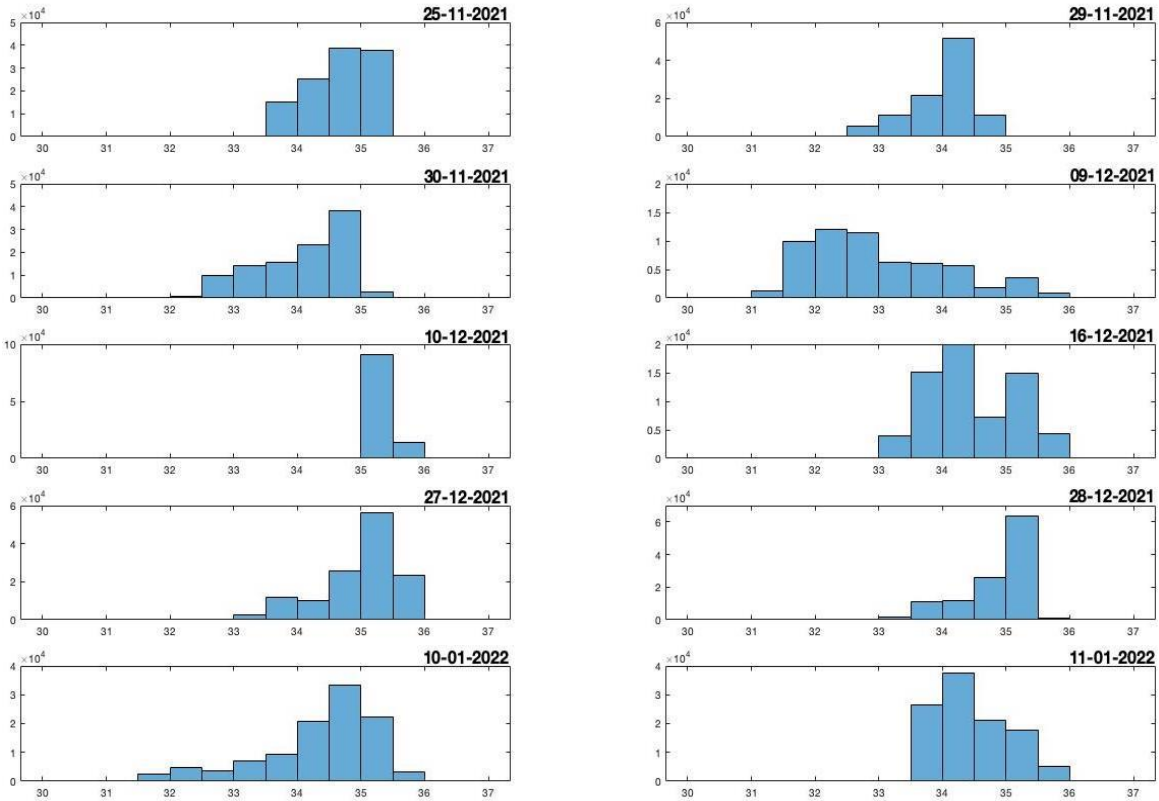

Fig. 46 Histograms of the original temperature curve

In each case, the value of the three statistics has been calculated. Previously to skewness and kurtosis computation, the temperature time series was smoothed using a moving average filter with a sliding window of 10 minutes. The temperature recording for the 10 subjects after smoothing are shown in Figure 47. This averaging aims to reduce the impact of the noise.

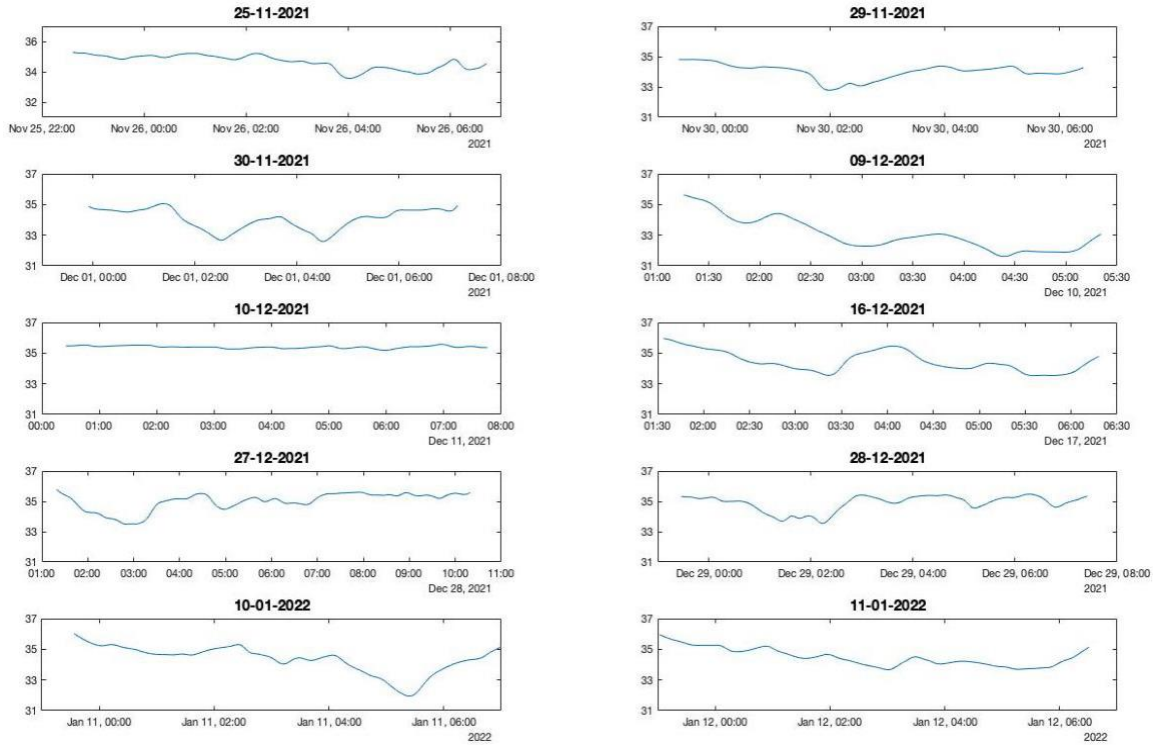

Fig. 47 Output signal after 10 min moving average applied to compute skewness and kurtosis

For the computation of standard deviation, because is in the short sleep interruptions, the temperature time series was detrended by subtracting to the original signal the output of the same signal after a moving average filter with 90 minutes. The subtracted trend for the 10 subjects is shown in Figure 48.

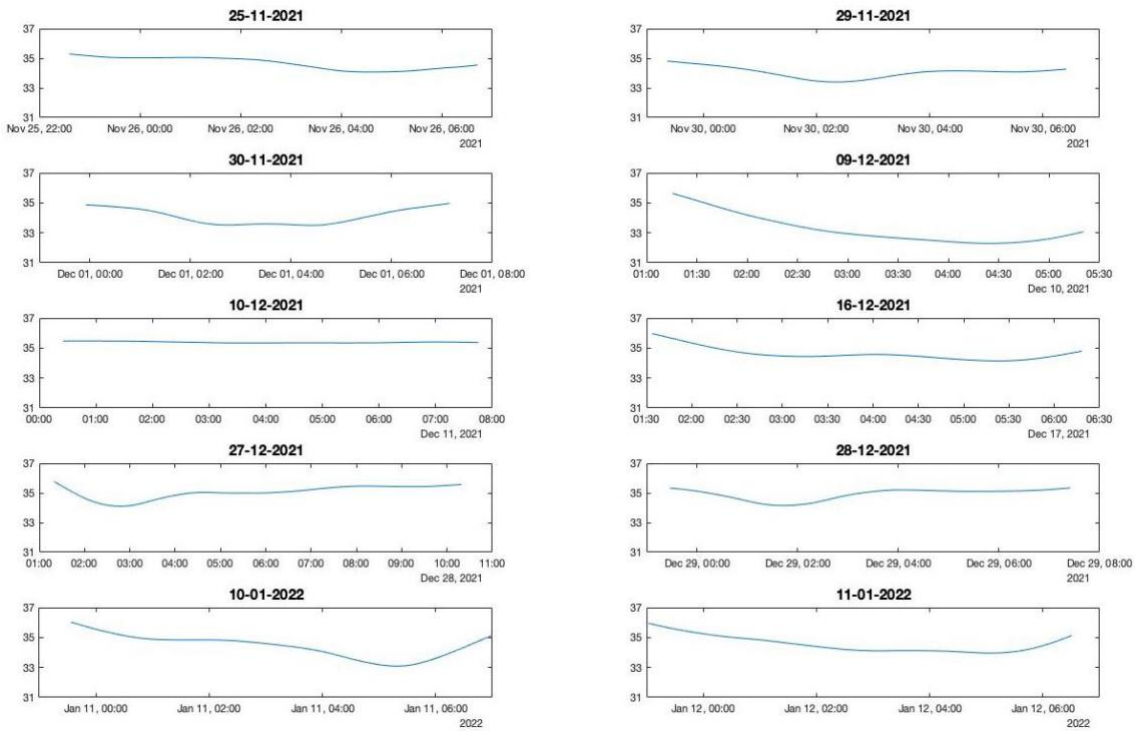

Fig. 48 . Output signal after 90 min moving average applied to compute standard deviation

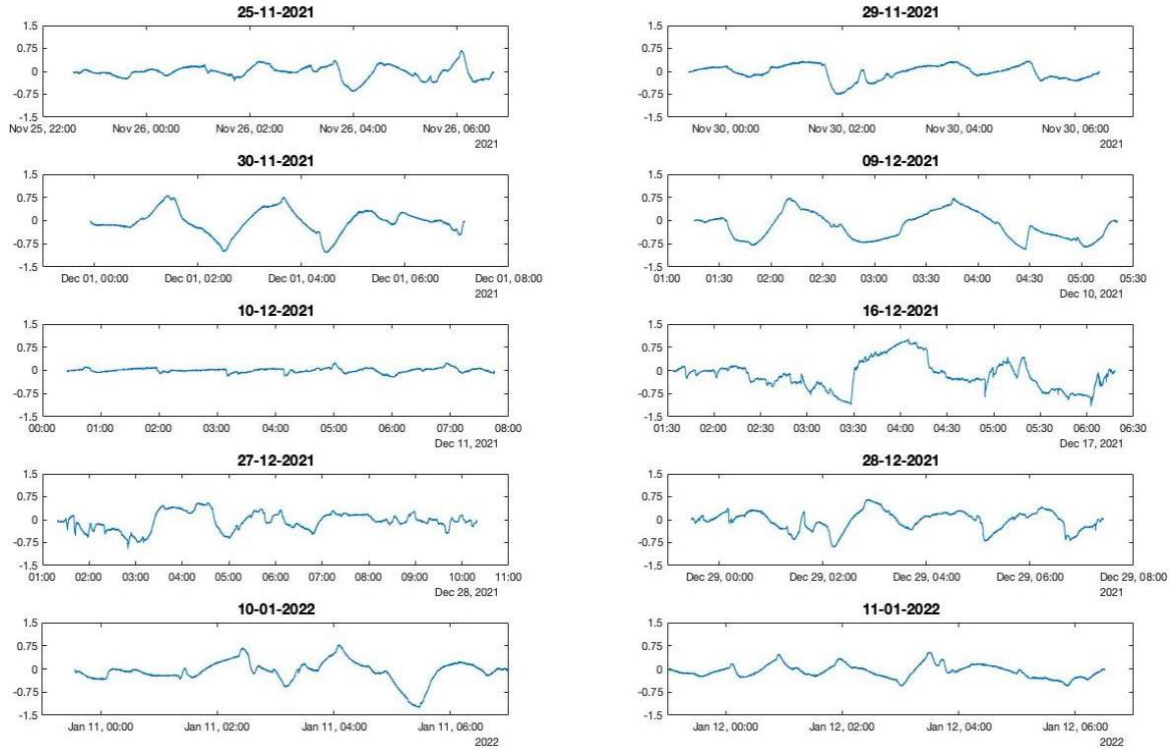

Fig. 49 Difference between original signal and processed signal with 90min moving average

In Figure 49 it can be seen the difference between the original signal and the processed signal with 90 minutes moving average. To implement the moving average filter the MATLAB function `output=filtfilt(ones(size((1:fs*60*L)))/(fs*60*L), 1, input )` has been employed, where L is the averaging time in minutes, fs is the sampling frequency of the temperature measuring system and 60 converts the minutes to seconds.

The expected values for a good sleep cycle are skewness close to zero, kurtosis less than three, and low standard deviation. If there is any disturbance (peak) in the sleep cycle, any of the three statistics could be altered.

Table 2 presents the values of statistics obtained. If the values of the standard deviation are compared to the histogram of each measurement, it can be observed that the measurement with a broad set of values presents a larger value of standard deviation. An example could be the recording ’09-12-2021’ where the temperatures values are spread from 31ºC to 36ºC which has a value of 0,37ºC. On the other hand, the recording ’10-12-2021’ where most values are around 36,5ºC presents a standard deviation of 0.07ºC. Another case with a great standard deviation is recording ’16-12-2021’ where the temperature presents a smaller range of values than in the recording '09-12-2021' but the amount of data in each temperature is higher, which translates into more marked peaks.

Date	Skewness	Kurtosis	SD	Acceleration
25/11/2021	0,72	2,22	0,23	0,15
29/11/2021	-0,81	3,10	0,24	0,14
30/11/2021	-0,56	2,08	0,39	0,18
09/12/2021	0,76	2,62	0,37	0,09
10/12/2021	-0,18	2,49	0,07	0,12
16/12/2021	0,48	2,13	0,47	0,16
27/12/2021	-1,20	3,44	0,29	0,16
28/12/2021	-1,06	2,84	0,31	0,19
10/01/2022	1,00	3,32	0,35	0,16
11/01/2022	0,54	2,30	0,20	0,19

Table 2 Statistics values of the measurements done

Moreover, the skewness and kurtosis value can be reflected on Figure 49 where the peaks and variations of temperature can be observed easily. An example could be the recording ’10-01-2022’ which presents a peak at the end of the measurement. The peak generates a tail on the left side as a result the skewness is positive, and the kurtosis as the peak provokes a high variation of temperature has a value higher than three. On the other hand, the recording ’28-12-2021’ as presents a peak at the beginning, the tail is on the right side so the skewness is negative, and as the peak generates a small variation of one degree the kurtosis is less than three.

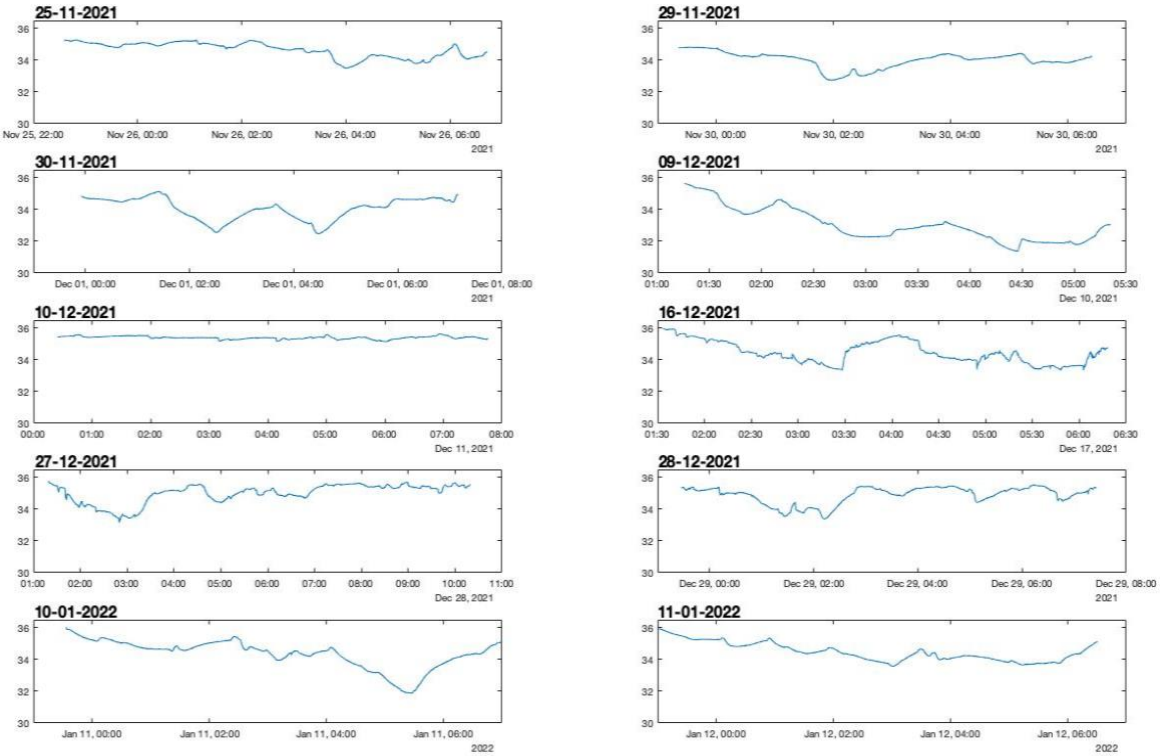

Fig. 50 Raw temperature reported by the E4

Figure 50 corresponds to the raw temperature provided by the E4 wristband. Comparing the histograms seen on Figure 46 with the original signals, those histograms that have fewer bars in them are those that have temperatures with smaller changes, therefore, more constant over time.

Another factor to analyse as an indicator of the sleep quality of the subject is acceleration. Acceleration implies movement that reflects user discomfort resulting in a disturbed sleep cycle. To calculate the acceleration, the three axes acceleration signals provided by the E4 wristband have been obtained during the sleep of the subject. Then, the square root of the quadratic sum of the three axes was calculated to obtain the magnitude of acceleration in m/s^2. After this, the average of the acceleration magnitude has been computed for each sleep cycle. The acceleration code to calculate the correlation with the temperature together with the characterization of temperature code can be found in Appendix 2 and 3.

The relationship between the standard deviation and the average acceleration is studied. It is expected that for a small standard deviation value, resulting from a good sleep cycle, the mean acceleration value is also small. Figure 51 shows the relationship between standard deviation and acceleration. The measures with correct sleep cycles are located at the bottom left, low standard deviation and low value of acceleration. As an example, the signal 10-12-2021 which shows a low standard deviation along with a low acceleration.

With the information of the other users, it is observed that as the standard deviation increases, the average acceleration also increases. There are small deviations, for example, between users 11-25-2021 and 11-29-2021, but the variation in values is not significant, which could be the result of a user turning during the sleep cycle that does not imply an alteration of sleep that is, it is not due to stress or discomfort.

However, there are users who do not have a good relationship between standard deviation and acceleration, such as 09-12-2021, 11-01-2021 and 16-12-2021. The disagreement observed can be caused by other not controlled causes such as variations in the temperature sensor not related to physiological events, such as sleeping during a period with the wrist exposed to room temperature or with the wrist covered by the blankets. For these cases, it would be necessary to analyse other factor, such as heart rate (HR) or electrodermal activity (EDA).

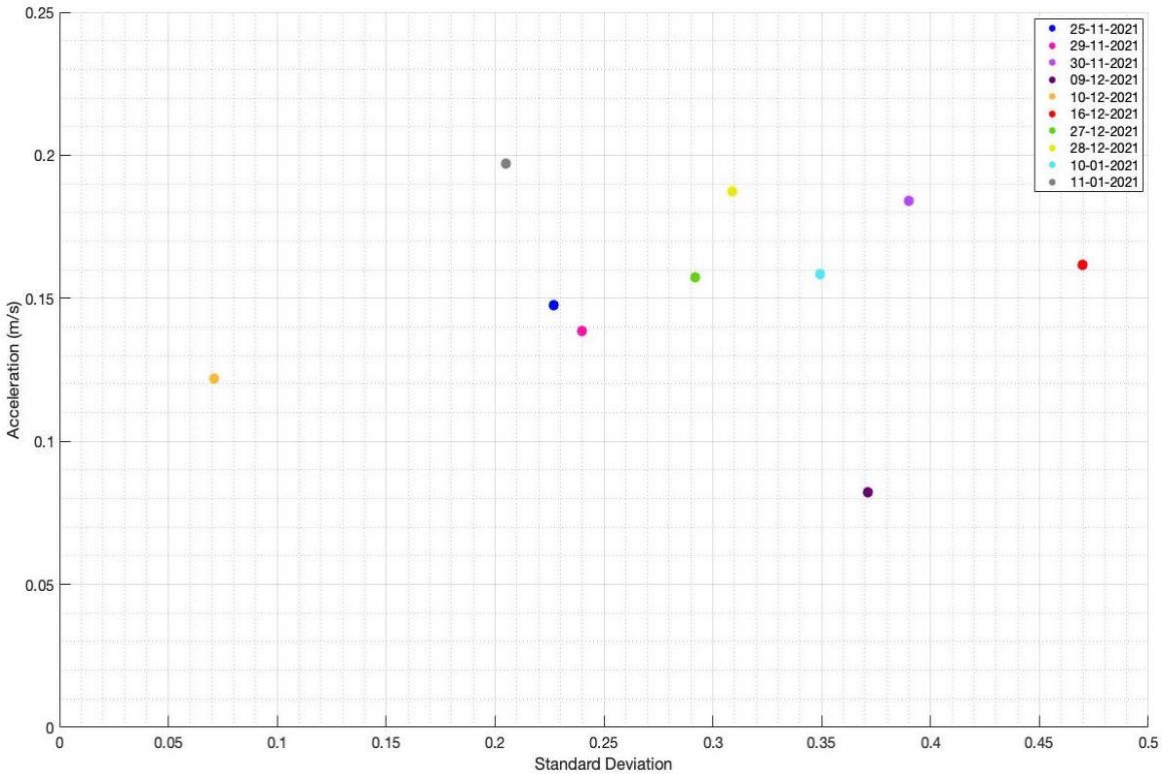

Fig. 51 Standard Deviation (ºC) vs. Mean of Acceleration (m/s²)

As it is mentioned early, a questionnaire was carried out to see if there was a relationship between how the subjects felt that they had slept and how it was shown in the temperature and acceleration graphs. The questionnaire can be found in Appendix 4. Table 3 shows the main questions where the study was focused on.

Question 1: I fell asleep last night...		
Easily	After a while	With difficulty
0	1	2
Question 2: I woke up during the night?		
No	Yes	
0	1	
Question 3: When I woke up, I felt...		
Fresh	A little tired	Tired
0	1	2

Table 3 Main questions of the questionnaire perform to the subjects

The square root of the quadratic sum of the three values obtained from the main questions has been performed to obtain a final value between 0 and 3, meaning a correct or bad sleep cycle, respectively. The value obtained has been compared with the standard deviation to analyse whether both values are correlated. The results of the main questions of the questionnaire and the final value are presented in Table 4 together with the standard deviation.

Date	Question 1	Question 2	Question 3	Result	SD
09/12/2021	1	1	2	2,45	0,37
28/12/2021	1	1	2	2,45	0,31
29/11/2021	1	1	1	1,73	0,24
27/12/2021	1	1	1	1,73	0,29
16/12/2021	1	0	1	1,41	0,47
11/01/2022	1	1	0	1,41	0,20
10/12/2021	0	1	0	1,00	0,07
10/01/2022	0	1	0	1,00	0,35
25/11/2021	0	0	0	0,00	0,23
30/11/2021	0	0	0	0,00	0,39

Table 4. Comparison of the values obtained from the questionnaire and the standard deviation of the detrended temperature

It has been seen, from the results, that the values obtained with the questionnaire are not related to the standard deviation. It can be concluded that the perception of the quality of sleep can differ greatly from how truly the user slept. Therefore, it is difficult to correlate the temperature and acceleration information obtained by the system through a questionnaire.

5 Budget

The development of the system requires the design of a prototype with the components commented in 'Section 3: Device Improvements'. The software used to design the PCB and program the microcontroller is open source with exception of Matlab which can access for free due to the student status.

Table 5 shows the components list as well as their unitary price. It is necessary to add the price of the prototype PCB which was 24.50€. The price of the used box has not been added since the one designed in the previous project has been reused as commented in the 'Sleep cycle measurements and characterization' section.

Component	Description	Commercial price (€)	Quantity	Subtotal (€)
TMP117	High-Precision Digital Temperature Sensor	4.64	2	9.28
DS323MZ	RTC	6.57	1	6.57
M24M02-DRMN6TP	EEPROM	2.88	1	2.88
MMA8652FCR1	MEMS Accelerometer	2.40	1	2.40
ATSAMD21E18A-A	Microcontroller	3.82	1	3.82
AP2112	LDO	0.34	1	0.34
MAX1555	Charger	2.25	1	2.25
BAT54C	Battery	0.16	1	0.16
503480-0600	Connector FFC/FPC	0.69	2	1.38
15020-0051	FFC/FPC Jumper cable	3.46	1	3.46
B3FS-1000	Tactile switches	0.61	1	0.61
105017-0001	Conector USB	0.84	1	0.84
YOBLP422339PACK	Rechargeable battery	18.41	1	18.41
Total				52.41

Table 5 Components summary

Regarding personnel, the project consists of a main operator and extra help from a laboratory technician. The main worker is listed as a junior engineer and the average salary for one is approximately €10/h. A laboratory technician is considered a Senior Engineer and has a salary of approximately €35/h. The hours used by the main operator are 20 weeks with a total of 20h/week, equivalent to 400h. As far as the laboratory technician is concerned, a total of 20 hours has been recorded.

Worker	Salary/hour (€/h)	Hours (h)	Salary (€) *without taxes*
Junior Engineer	10	400	4000
Laboratory Technician	35	20	700
Total			4700

Table 6 Personal salaries summary

Table 6 shows the salary for the people involved in the project. Considering the two totals, the project has had a total cost of 4776.91€.

6 Conclusions and future development

Distal skin temperature is an accurate biological signal modulated by the sleep-wake cycle. The system developed in the project allows to correctly measure the temperature changes experienced during the night hours as well as the physical activity of the user.

The improvements added to the design have created an efficient version of the system. The connector selected to make the connection from the general PCB board to the skin temperature sensor, TMP117, has made the system more robust and therefore, it is more difficult to disconnect it due to user movements during measurement. The improvement has allowed to have a correct sensor-skin contact which allows the precise acquisition of the skin temperature values. It has been observed in various measurements that the system is capable of detecting changes in temperature both in the skin and in the environment during the night, which allows a subsequent analysis of sleep. If the values obtained with the Empatica E4 wristband used to characterize the system are compared, the correct functioning of the developed system can be confirmed, where the increase in temperature is observed, reaching the maximum value when the user is completely asleep, where it remains constant until the user awakening.

On the other hand, once the correct operation has been achieved, the calibration performed on the system was applied to the final design. The aim of the calibration was to eliminate changes in skin temperature due to changes in ambient temperature.

Related to the software, the previous code did not facilitate the user the task of interacting with the system, since the results obtained required post-processing to be able to correctly display the temperature values. For this reason, it was also proposed as an objective to improve the wristband-pc connection. The code developed for the new version has added various functions to configure the system before the measurement starts such as set the date/time or visualize the current temperature. However, the main added function is the acquisition of the acceleration that it shows in periods of 10 seconds, as the temperature, the values of the three axes of the accelerometer. The function allows to correlate the information acquired of the temperature since it is related to user movement at night. As has been shown in the project, when the user is more physically active, meaning that has moved more during the night, the sleep quality can be considered worse than when the user remains still. The device acquires three axes of acceleration every 10 seconds as mentioned above, so consequently there are movements that are not acquired within 10 seconds between each sample. An improvement to be made in the next version of the system is to change the mode of operation of the accelerometer to acquire data from the

accelerometer every time a movement is detected. In this way, the accelerometer will always be active, and the data acquired will be more accurate.

The characterization of the system has made it possible to detect when the user has a standard sleep cycle, using statistics as features for changes in temperature. The three selected statistics to perform the characterization were: standard deviation that measures the amount of variation of a set of values; skewness, which measures the asymmetry of the histogram, and kurtosis, which measures whether the data has heavy or light tails respect to a normal distribution. The temperature of the distal skin during the night begins to increase in the first hours seeking the maximum value when the user is completely asleep and remains constant until waking up. For that, a standard sleep cycle has to present a constant temperature from when the user falls asleep until he/she wakes up. The values of statistic in function of the expected curve of temperature should be skewness close to zero, kurtosis less than three and low standard deviation. If there is any disturbance (peak) in the sleep cycle, any of the three statistics could be altered. To demonstrate that the statistics offered adequate results, they have been compared with the histogram of each of the measurements since it allows an easy visualization of when the temperature remains constant or extends in a large number of values, which would result in changes in temperature. If the values of the statistics correlate with the histogram, it can be said that they offer good results and are capable of detecting a bad sleep cycle.

The statistic that gives the best results is the standard deviation. Nevertheless, for the sake of reproducibility of results, an algorithm for automatically detect the start and the end of the sleep period could be a future improvement. These statistics are based on the distribution of the values, therefore it is necessary to obtain the complete cycle of the skin temperature from the time it starts to increase when the user goes to sleep until it decreases completely to the temperature value during the day. In this way, by having the complete cycle, the result will be more precise.

Finally, the last change to be performed on the next version is to substitute the wristband of the watch. The sensor used to detect skin temperature is a contact sensor which requires to be perfectly in physical contact with the skin to monitor correctly the changes in temperature. The strap used has the most common buckle closure, similar to a belt closure, which means that the user must ensure that it is closed properly to present good skin-sensor contact. An option for improvement would be a completely elastic rubber strap without closure that will automatically adapt to the user's wrist, thus always ensuring correct contact.

Bibliography

[1] J. A. Sarabia, M. A. Rol, P. Mendiola, and J. A. Madrid, “Circadian rhythm of wrist temperature in normal-living subjects. A candidate of new index of the circadian system,” Physiol. Behav., vol. 95, no. 4, pp. 570–580, 2008, doi: 10.1016/j.physbeh.2008.08.005.

[2] A. M. Vosko, C. S. Colwell, and A. Y. Avidan, “Jet lag syndrome: Circadian organization, pathophysiology, and management strategies,” Nat. Sci. Sleep, vol. 2, no. June 2014, pp. 187–198, 2010, doi: 10.2147/NSS.S6683.

[3] Texas Instruments, “TMP117 high-accuracy , low-power , digital temperature sensor with SMBus TM - and I2C-compatible interface,” no. June 2018, pp. 1–46, 2019.

[4] Empatica, “E4 wristband | Real-time physiological signals | Wearable PPG, EDA, Temperature, Motion sensors,” Empatica, 2020. https://www.empatica.com/en-eu/research/e4/ (accessed Apr. 26, 2022).

[5] “Celsium Body Temperature | Body Temperature Monitoring.” https://www.celsium.com/ (accessed Apr. 26, 2022).

[6] “Home - Sotera Wireless.” https://www.soterawireless.com/ (accessed Apr. 26, 2022).

Appendix

1 Arduino Code

```
//Useful Libraries
#include "ArduinoLowPower.h"
#include <Wire.h> //Library to use the I2C
#include "RTClib.h" //Library to use DS3231
#include <RTCZero.h> // Library to use internal RTC
#include <Adafruit_SleepyDog.h>//Library to use intern Watchdog timer
#include <SparkFun_TMP117.h>    // Used to send and receieve specific information from our sensor
#include <MMA8653.h>

//States of the program and commands to send
#define STOP 0 //S
#define START 1 //A
#define SEND 2 //E
#define RESET 3 //N
#define READ 4 //R
#define DATE 5 //D
#define MONITOR 6 //M
#define TIME 7 //T

//Microcontroller pin assigments
const int LED = 13; //Help you to know if the Arduino bootloader is correctly programmed

//Useful variables
byte State = 0; //Solve the actual state

//Temperature Sensor definitions
TMP117 sensor; // Initalize sensor
TMP117 sensor_SKIN; // Initalize sensor

//Internal RTC definitions
RTCZero intRTC;
RTC_DS3231 extRTC;
boolean TEST = false; //Variable for controling acquisition or send data (in acquisition mode goes to sleep
TEST=0)
// Set how often alarm goes off here
```

```
const byte alarmSeconds = 10;
const byte alarmMinutes = 0;
const byte alarmHours = 0;
volatile bool alarmFlag = true; // Start awake

const char daysOfTheWeek [7][12]={"Sunday","Monday","Tuesday","Wednesday", "Thursday", "Friday",
"Saturday"};

//Accelerometer definitions
MMA8653 accel = MMA8653();
int16_t x,y,z;
int Vbat = 0;

//EEPROM MEMORY
//EEPROM has I2C addresses 0x50, 0x51, 0x52 and 0x53 (0x50 | 0x00, 0x50 | 0x01,0x50 | 0x02 and 0x50 |
0x03)
#define EEPROM_DATA_ADD 0x50 // Address of the first 1024 page M24M02DRC EEPROM data buffer,
2048 bits (256 8-bit bytes) per page
#define EEPROM_IDPAGE_ADD 0x58 // Address of the single M24M02DRC lockable ID page of the
EEPROM
#define length_Fblock 14
#define length_Sblock 10
#define L_MSmax 256 //Maximum of pages per block and maximum data per page
#define maxBlocks 25 //Maximum number of blocks can be sended per page
#define Bmax 3
uint8_t Badd = 0, MSadd = 0, LSadd = 0; //Badd=Block address (0-3) ; MSadd=MSB(page 0-255) ;
LSadd=LSB(0-255)
bool EEPROM_empty = true;
int currentBlocks, sendedBlocks=31744; //(==31744 if you want read memory)

//Obtained data
struct data{ //Struct with all the variables that we will need
  byte day;
  byte month;
  int year;
  byte hour;
  byte min;
  byte sec;
  float int_temp;
```

```
  float pat_temp;
  short x;
  short y;
  short z;
};

typedef struct data Data;
Data ReadData; //data to print
Data ObtData; //Obtanined data to storage
uint8_t clearData[256]; //Variable to reset the EEPROM memory
uint8_t receiveData[L_MSmax]; //Variable to receive byte to byte from EEPROM memory

//Useful variables
byte prevHour;
byte prevMin;
int tt1=0, tt2=0;
int countdownMS;

void setup() {

  Wire.begin();
  while (!Serial) ;
  Serial.begin(115200); //opens serial port, sets data rate to 115200 bps
  Serial.println("Setup");
  pinMode(LED,OUTPUT);

  //External RTC initialize
  if (! extRTC.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }
  if (extRTC.lostPower()) {
    Serial.println("RTC lost power, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    extRTC.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
```

```
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }else{
    extRTC.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  //Internal RTC initialize if TEST=FALSE
  if(TEST==false){
    intRTC.begin(); // Set up clocks and such
    resetAlarm();  // Set alarm
    intRTC.attachInterrupt(alarmMatch); // Set up a handler for the alarm
    digitalWrite(LED,LOW);
    Serial.println("DORMIREMOS");
    delay(1000);
    LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, onWakeup, CHANGE);
    State=STOP;
  }else{
    digitalWrite(LED,HIGH);
  }

  // Configure TMP117
  Serial.println("Configure TMP117 Sensors");
  if (sensor.begin(0x48) == true) // Function to check if the sensor will correctly self-identify with the proper
Device ID/Address
  {
    Serial.println("Begin TMP117 Ambient");
  }
  else
  {
    Serial.println("Device failed to setup - Freezing code (ambient)");
    while (1); // Runs forever
  }

  if (sensor_SKIN.begin(0x49) == true) // Function to check if the sensor will correctly self-identify with the
proper Device ID/Address
  {
    Serial.println("Begin TMP117 Skin");
  }
  else
  {
```

```
    Serial.println("Device failed to setup- Freezing code (skin)");
    while (1); // Runs forever
  }

  //Configure accelerometer
  accel.init(MMA8653_2G_RANGE,MMA8653_10BIT_RES,MMA8653_ODR_50);
  accel.setMODS(MMA8653_MODS_HIGH_RES);
  //Put accelerometer in active mode
  accel.begin();
}

void loop() {
 byte command;
 switch (State){
  case STOP:{
    if(Serial.available()>0){

      command=Serial.read();
      switch (command){
        case 'S':{
          Serial.println("STOPPING");
          State= STOP;
          delay(100);
          break;
        }
        case 'A':{
          Serial.println("CHANGING TO ACQUISITION MODE");
          State=START;
          delay(100);
          break;
        }
        case 'N':{
          Serial.println("RESETTING THE MEMORY");
          State=RESET;
          break;
        }

        case 'R':{
          Serial.println("READING THE MEMORY");
```

```
        State=READ;
        break;
      }
      case 'D':{
        Serial.println("CURRENT DATE/TIME");
        State = DATE;
        break;
      }
      case 'M':{
        Serial.println("CURRENT TEMPERATURE/ACCELERATION");
        State = MONITOR;
        break;
      }
      case 'T':{
        Serial.println("SET DATE/TIME");
        State = TIME;
        break;
      }
    }
    // Plot menu
    Serial.println("  ");
    Serial.println("  ");
    Serial.println("S Stop Acquisition");
    Serial.println("A Start Acquisition, goes to sleep");
    Serial.println("N Reset memory (acquired data)");
    Serial.println("R Read data acquired");
    Serial.println("D Date/time printed");
    Serial.println("M Temperature/Acceleration printed");
    Serial.println("T Sets date/time");
    Serial.println("  ");
    Serial.println("  ");
  }
  break;
}//case:STOP

case START:{

if(Badd>=4){ //Exceed the number of blocks
  EEPROM_empty=false;
```

```
    Serial.println("Full memory, STOP acquisition");
    Serial.print("Blocks sended: ");
    Serial.println(sendedBlocks);
    State=STOP;
    break;
  }

  if (TEST==false){  //False implies that the watch is ready to read the sensors
    readSensors(); //Read temperature values
    State = SEND;//Send automatically the temperature values read
    Serial.print("LSadd: ");
    Serial.print(LSadd);
    delay(10000);
    command = Serial.read();
    if(command=='S'){
      State = STOP;
      break;
    }
  }else{
    readMemory();
    //readSensors();
    testReadSensors();
    State=SEND;
    delay(10000);
    command=Serial.read();
    if(command=='S'){
      State=STOP;
      break;
    }
  }
  break;
  }//case:START

  case SEND:{
    if(LSadd==0){
      M24M02DRCwriteBytes(EEPROM_DATA_ADD | Badd, MSadd, LSadd, true, ObtData);
      delay(10);// It takes a maximum of 10 ms for a read or write operation; the EEPROM won't respond until
the operation is done
      LSadd = LSadd + length_Fblock; //increasing the number of bytes written to the page
```

```
    currentBlocks++;
    sendedBlocks++;
    Serial.print("Current block: ");
    Serial.println(currentBlocks);
   }else{
    M24M02DRCwriteBytes(EEPROM_DATA_ADD | Badd, MSadd, LSadd, false, ObtData);
    delay(10);
    LSadd= LSadd+length_Sblock;
    currentBlocks++;
    sendedBlocks++;
    Serial.print("Current block: ");
    Serial.println(currentBlocks);
    if(currentBlocks == maxBlocks){
     LSadd=0;
     MSadd++;
     currentBlocks=0;
     if(MSadd == 0){
      Badd++;
      Serial.println(" BLOQUE ACTUALIZADO ");
      delay(500);
     }
    }
   }

   if(TEST==false){
    State=START;
    break;
   }else{
    State=STOP;
    break;
   }
  }//case:SEND

  case RESET:{
   //RESET EEPROM MEMORY
   for(uint8_t B=0; B < 4; B++){ //fixing block
    for(uint8_t M=0; M < (L_MSmax-1); M++){ //fixing MSB
     M24M02DRCclear(EEPROM_DATA_ADD | B, M, 0,clearData);
     delay(10);
```

```
    }
    M24M02DRCclear(EEPROM_DATA_ADD | B, 255, 0,clearData);
    delay(10);
  }
  Serial.println("Memory reset");
  EEPROM_empty=true;
  LSadd = 0;
  State=STOP;//once the memory is reset, the program goes to stop state
  break;
}//case:RESET

case READ:{ //read the bytes saved on the eeprom
int s=0;
uint8_t B=0;
while(B<=Bmax){
  for(uint8_t M = 0; M < (L_MSmax-1) ; M++){    //&& s<=sendedBlocks
    M24M02DRCreadBytes(EEPROM_DATA_ADD | B, M, 0);
    delay(10);
    s+=25;
    printData();
    }
 M24M02DRCreadBytes(EEPROM_DATA_ADD | B, 255, 0);
 delay(10);
 s+=25;
 printData();
 B++;
 }
 State=STOP;
 break;
}//case:READ

case TIME:{//set the time of the RTC
  serial_flush();
  int s_year = ask_for_number("Year");
  int s_month = ask_for_number("Month");
  int s_day = ask_for_number("Day");
  int s_hour = ask_for_number("Hour");
  int s_min = ask_for_number("Minutes");
  int s_sec = ask_for_number("Seconds");
```

```
    extRTC.adjust(DateTime(s_year,s_month,s_day,s_hour,s_min,s_sec));
    print_datetime();
    State = STOP;
    break;
  }//case:TIME

  case DATE:{
    print_datetime();
    State = STOP;
    break;
  } //case:DATE

  case MONITOR:{
    Serial.print("Skin temperature (ºC): ");
    Serial.println(sensor_SKIN.readTempC());
    Serial.print("System temperature (ºC): ");
    Serial.println(sensor.readTempC());
    Vbat = analogRead(A4);
    Serial.print("Battery voltage: ");
    Serial.println(Vbat);
    State = STOP;
    break;
  }//case: MONITOR
 }//switch
}//loop

////////////////////
//USEFUL FUNCTIONS//
////////////////////

//Set date/time
void print_datetime()
{
  DateTime now2 = extRTC.now();
  Serial.print(now2.year(),DEC);
  Serial.print('/');
  Serial.print(now2.month(),DEC);
  Serial.print('/');
  Serial.print(now2.day(),DEC);
```

```
  Serial.print(' ');
  Serial.print(now2.hour(),DEC);
  Serial.print(':');
  Serial.print(now2.minute(),DEC);
  Serial.print(':');
  Serial.print(now2.second(),DEC);
  Serial.println();
}

//Reset Internal RTC ALARM
void resetAlarm(void){
  intRTC.setTime(00,00,00);
  intRTC.setDate(1,1,1);

  //Set alarm time select previously
  intRTC.setAlarmTime(alarmHours, alarmMinutes, alarmSeconds);
  //intRTC.setAlarmTime(now.hour()+alarmHours, now.minute()+alarmMinutes,
now.second()+alarmSeconds);
  intRTC.enableAlarm(intRTC.MATCH_HHMMSS);
}

void alarmMatch(){ //REVISAR
  digitalWrite(LED, HIGH);
  alarmFlag = true; // Set flag
}

//Get current Time
DateTime current_Time(){
  return extRTC.now();
}

//Send bytes to EEPROM it's an adaptation of the function in github repository (it's thinking to have two
different devices in the same design)
void M24M02DRCwriteBytes(uint8_t device_address, uint8_t page_address, uint8_t data_address, bool first,
Data dest){

  int int_int =(int)(dest.int_temp*100);
  int int_skin=(int)(dest.pat_temp*100);
  //int int_int =(int)(dest.int_temp);
```

```
  //int int_skin=(int)(dest.pat_temp);

  if(first == true) {

    Wire.beginTransmission(device_address); // Initialize the Tx buffer
    Wire.write(page_address); // Put slave register address in Tx buffer
    Wire.write(data_address); // Put slave register address in Tx buffer
    Wire.write(dest.day);
    Wire.write(dest.month);
    Wire.write(dest.year >> 8);
    Wire.write(dest.year);
    Wire.write(dest.hour);
    Wire.write(dest.min);
    Wire.write(dest.sec);
    Wire.write(int_int >> 8);  //Sensor returns a float which we multiply*100 and we do a cast to int to be able to
send via I2C
    Wire.write(int_int);
    Wire.write(int_skin >> 8);
    Wire.write(int_skin);
    Wire.write(x);
    Wire.write(y);
    Wire.write(z);
    Wire.endTransmission(); // Send the Tx buffer

  } else{

    Wire.beginTransmission(device_address); // Initialize the Tx buffer
    Wire.write(page_address); // Put slave register address in Tx buffer
    Wire.write(data_address); // Put slave register address in Tx buffer
    Wire.write(dest.hour);
    Wire.write(dest.min);
    Wire.write(dest.sec);
    Wire.write(int_int >> 8);
    Wire.write(int_int);
    Wire.write(int_skin >> 8);
    Wire.write(int_skin);
    Wire.write(x);
    Wire.write(y);
    Wire.write(z);
```

```
    Wire.endTransmission(); // Send the Tx buffer
    //Serial.println(dest.hour);
  }

}

//Clear the EEPROM memory fully (own function, it's an adaptation of the writing function
void M24M02DRCclear(uint8_t device_address, uint8_t page_address, uint8_t data_address, uint8_t *dest){

    Wire.beginTransmission(device_address); // Initialize the Tx buffer
    Wire.write(page_address); // Put slave register address in Tx buffer
    Wire.write(data_address); // Put slave register address in Tx buffer

    for(uint16_t i=0; i < (L_MSmax-1) ; i++) {
      Wire.write(dest[i]); // Put data in Tx buffer
    }
    Wire.endTransmission(); // Send the Tx buffer
}

//Read bytes from EEPROM it's and adaptation of the function in github repository (it's thinking to have two
different devices in the same design)
void M24M02DRCreadBytes(uint8_t device_address, uint8_t page_address, uint8_t data_address){
  Wire.beginTransmission(device_address); // Initialize the Tx buffer
  Wire.write(page_address); // Put slave register address in Tx buffer
  Wire.write(data_address); // Put slave register address in Tx buffer
  Wire.endTransmission(false); // Send the Tx buffer, but send a restart to keep connection alive
  Wire.requestFrom(device_address,(L_MSmax-2)); // Read bytes from slave register address
  uint8_t i = 0;
  while(Wire.available()){
    receiveData[i] = Wire.read();
    i++;
  }
}

//Function to read all the Sensors
void readSensors(){

   //Read RTC (obtain the date/time)
    DateTime fecha = current_Time();
```

```
  ObtData.day=fecha.day();
  ObtData.month=fecha.month();
  ObtData.year=fecha.year();
  ObtData.hour=fecha.hour();
  ObtData.min=fecha.minute();
  ObtData.sec=fecha.second();

  //Read Temperature
  if (sensor_SKIN.dataReady() == true) //Function to make sure that there is data ready to be printed, only
prints temperature values when data is ready
  {
    float tempC_SKIN = sensor_SKIN.readTempC();
    ObtData.pat_temp = tempC_SKIN;
  }
  if (sensor.dataReady() == true)
  {
    float tempC = sensor.readTempC();
    ObtData.int_temp = tempC;
  }



  //Read Accelerometer
  accel.readSensor(&x,&y,&z);
  ObtData.x = x;
  ObtData.y = y;
  ObtData.z = z;

  Serial.print(ObtData.hour,DEC);
  Serial.print(':');
  Serial.print(ObtData.min,DEC);
  Serial.print(':');
  Serial.print(ObtData.sec,DEC);
  Serial.print(' ');
  Serial.print(ObtData.pat_temp);
  Serial.print(' ');
  Serial.print(ObtData.int_temp);
  Serial.print(' ');
```

```
    Serial.print(ObtData.x);
    Serial.print(' ');
    Serial.print(ObtData.y);
    Serial.print(' ');
    Serial.println(ObtData.z);
}

//Function to check the sensors
void testReadSensors(){

    //LEER TEMPERATURAS
    Serial.print(" Skin Temperature: ");
    Serial.print(ObtData.pat_temp);
    delay(100);
    Serial.print(" Ambient Temperature: ");
    Serial.println(ObtData.int_temp);
    delay(100);

    //Read Accelerometer
    accel.readSensor(&x,&y,&z);
    ObtData.x = x;
    ObtData.y = y;
    ObtData.z = z;

    //LEER RTC EXTERNO
    //Leer fecha
    Serial.print("Fecha: ");
    Serial.print(ObtData.day,DEC);
    Serial.print("/");
    Serial.print(ObtData.month,DEC);
    Serial.print("/");
    Serial.print(ObtData.year,DEC);
    Serial.print(" at ");
    Serial.print(ObtData.hour,DEC);
    Serial.print(":");
    Serial.print(ObtData.min,DEC);
    Serial.print(":");
    Serial.println(ObtData.sec,DEC);
    delay(100);
```

```
}

//Fucntion to printData in EEPROM memory
void printData(){
  uint8_t c = 0;
  uint8_t O = length_Fblock;
  while(c<(L_MSmax-2)){ //L_MSmax 256 so c < 255
    ReadData.day = receiveData[c];
    Serial.print(ReadData.day,DEC);
    Serial.print('/');
    ReadData.month = receiveData[c+1];
    Serial.print(ReadData.month,DEC);
    Serial.print('/');
    ReadData.year = (receiveData[c+2]*256.0)+receiveData[c+3];
    Serial.print(ReadData.year,DEC);
    Serial.print(' ');
    ReadData.hour = receiveData[c+4];
    Serial.print(ReadData.hour,DEC);
    Serial.print(':');
    ReadData.min = receiveData[c+5];
    Serial.print(ReadData.min,DEC);
    Serial.print(':');
    ReadData.sec = receiveData[c+6];
    Serial.print(ReadData.sec,DEC);
    Serial.print(' ');
    ReadData.int_temp = (receiveData[c+7]*256.0 + receiveData[c+8])/100.0;
    Serial.print(ReadData.int_temp);
    Serial.print(' ');
    ReadData.pat_temp = (receiveData[c+9]*256.0 + receiveData[c+10])/100.0;
    Serial.print(ReadData.pat_temp);
    Serial.print(' ');
    ReadData.x = receiveData[c+11];
    Serial.print(ReadData.x,DEC);
    Serial.print(' ');
    ReadData.y = receiveData[c+12];
    Serial.print(ReadData.y,DEC);
    Serial.print(' ');
    ReadData.z = receiveData[c+13];
    Serial.print(ReadData.z,DEC);
```

```
    Serial.println(' ');
    while (O<(L_MSmax-2)){
      Serial.print(ReadData.day,DEC);
      Serial.print('/');
      Serial.print(ReadData.month,DEC);
      Serial.print('/');
      Serial.print(ReadData.year,DEC);
      Serial.print(' ');
      ReadData.hour = receiveData[O];
      Serial.print(ReadData.hour,DEC);
      Serial.print(':');
      ReadData.min = receiveData[O+1];
      Serial.print(ReadData.min,DEC);
      Serial.print(':');
      ReadData.sec = receiveData[O+2];
      Serial.print(ReadData.sec,DEC);
      Serial.print(' ');
      ReadData.int_temp = (receiveData[O+3]*256.0 + receiveData[O+4])/100.0;
      Serial.print(ReadData.int_temp);
      Serial.print(' ');
      ReadData.pat_temp = (receiveData[O+5]*256.0 + receiveData[O+6])/100.0;
      Serial.print(ReadData.pat_temp);
      Serial.print(' ');
      ReadData.x = receiveData[O+7];
      Serial.print(ReadData.x,DEC);
      Serial.print(' ');
      ReadData.y = receiveData[O+8];
      Serial.print(ReadData.y,DEC);
      Serial.print(' ');
      ReadData.z = receiveData[O+9];
      Serial.print(ReadData.z,DEC);
      Serial.println(' ');
      O = O + length_Sblock;  //10
    }
    c = (L_MSmax-2);
  }
  Serial.println("");
}
```

```
void readMemory(){

  //Read RTC
   DateTime fecha = current_Time();
   ObtData.day=fecha.day();
   ObtData.month=fecha.month();
   ObtData.year=fecha.year();
   ObtData.hour=fecha.hour();
   ObtData.min=fecha.minute();
   ObtData.sec=fecha.second();

   if (sensor_SKIN.dataReady() == true) //Function to make sure that there is data ready to be printed, only
prints temperature values when data is ready
   {
     float tempC_SKIN = sensor_SKIN.readTempC();
     ObtData.pat_temp = tempC_SKIN;
   }
   if (sensor.dataReady() == true)
   {
     float tempC = sensor.readTempC();
     ObtData.int_temp = tempC;
     Serial.print(" Ambient Temperature: ");
     Serial.println(ObtData.int_temp);
   }
}

void serial_flush(void){
  while(Serial.available()) Serial.read();
}

int ask_for_number(String msg){
  Serial.print(msg);Serial.println('?');
  serial_flush();
  while (Serial.available() == 0) {}
  int number = Serial.parseInt();
  if (Serial.read() == '\n')
  {
    Serial.print(msg);
    Serial.print(' ');
```

```
    Serial.println(number);
  }
  return number;
}

void onWakeup() {
  digitalWrite(LED, HIGH);
}
```

2 Code to get proceed from E4

```
%% TEMPERATURE %%

load('TEMP.csv'); %Load temperature file
temperature = TEMP(3:end,:); %Delete the three firsts rows (information)
start_time = datetime(TEMP(1,1),'ConvertFrom','posixtime'); %Select initial time
(first row)
frequency = TEMP(2,1); %Select frequency (second row --> sample rate in Hz)

%Calculate vector of time/temperature
final_time =
datetime(1/frequency*length(temperature)+TEMP(1,1),'ConvertFrom','posixtime');
a_time_temp = TEMP(1,1):1/frequency:1/frequency*length(temperature)+TEMP(1,1);
b_time_temp = datetime(a_time_temp,'ConvertFrom','posixtime');
time = b_time_temp(1,1:length(temperature))';
values = 1:TEMP(2,1):length(temperature);

temperature_values = temperature(values);
time_values = time(values);
save('temp_25_11_2021.mat','temperature_values','time_values')

%% ACCELEROMETER %%

load('ACC.csv'); %Load accelerometer values
acc = ACC(3:end,:); %Delete the three firsts rows
sample_rate = ACC(2,1);%Calculate the sample rate with the second row
start_time = datetime(ACC(1,1),'ConvertFrom','posixtime');%Calculate the start
time with the first row

%calulate the time vector
final_time =
datetime(1/sample_rate*length(acc)+ACC(1,1),'ConvertFrom','posixtime');
a_time_ac = ACC(1,1):1/sample_rate:1/sample_rate*length(acc)+ACC(1,1);
b_time_ac = datetime(a_time_ac(1:length(acc)),'ConvertFrom','posixtime');


accx = acc(:,1)./64*9.81;
accy = acc(:,2)./64*9.81;
accz = acc(:,3)./64*9.81;
accelerometer = sqrt(accx.^2+accy.^2+accz.^2);
save('acceleration.mat',accelerometer)
```

3 Code to characterize the data

```
clear all;
clc;
%Load the measurements
load('temp_25_11_2021.mat');
load('temp_29_11_2021.mat');
load('temp_30_11_2021.mat');
load('temp_09_12_2021.mat');
load('temp_10_12_2021.mat');
load('temp_16_12_2021.mat');
load('temp_27_12_2021.mat');
load('temp_28_12_2021.mat');
load('temp_10_01_2022.mat');
load('temp_11_01_2022.mat');

%% TEMPERATURE %%
%Plot the raw temperature reported by the E4
figure(1)
subplot(5,2,1);plot(time_25_11,temperature_25_11);title('25-11-2021');
subplot(5,2,2);plot(time_29_11,temperature_29_11);title('29-11-2021');
subplot(5,2,3);plot(time_30_11,temperature_30_11);title('30-11-2021');
subplot(5,2,4);plot(time_09_12,temperature_09_12);title('09-12-2021');
subplot(5,2,5);plot(time_10_12,temperature_10_12);title('10-12-2021');
subplot(5,2,6);plot(time_16_12,temperature_16_12);title('16-12-2021');
subplot(5,2,7);plot(time_27_12,temperature_27_12);title('27-12-2021');
subplot(5,2,8);plot(time_28_12,temperature_28_12);title('28-12-2021');
subplot(5,2,9);plot(time_10_01,temperature_10_01);title('10-01-2022');
subplot(5,2,10);plot(time_11_01,temperature_11_01);title('11-01-2022');

%Plot the histogram of the temperature
figure(2)
subplot(5,2,1);histogram(temperature_25_11);title('25-11-2021');
subplot(5,2,2);histogram(temperature_29_11);title('29-11-2021');
subplot(5,2,3);histogram(temperature_30_11);title('30-11-2021');
subplot(5,2,4);histogram(temperature_09_12);title('09-12-2021');
subplot(5,2,5);histogram(temperature_10_12);title('10-12-2021');
subplot(5,2,6);histogram(temperature_16_12);title('16-12-2021');
subplot(5,2,7);histogram(temperature_27_12);title('27-12-2021');
subplot(5,2,8);histogram(temperature_28_12);title('28-12-2021');
subplot(5,2,9);histogram(temperature_10_01);title('10-01-2022');
subplot(5,2,10);histogram(temperature_11_01);title('11-01-2022');

%Smooth of the temperature signal using a moving average filter with a
%sliding window of 10min
tm_25_11 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_25_11);
tm_29_11 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_29_11);
tm_30_11 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_30_11);
tm_09_12 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_09_12);
tm_10_12 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_10_12);
tm_16_12 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_16_12);
tm_27_12 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_27_12);
tm_28_12 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_28_12);
tm_10_01 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_10_01);
tm_11_01 = filtfilt(ones(size((1:4*60*10)))/(4*60*10),1,temperature_11_01);

figure(3) %Plot output signal after 10min moving average
```

```
subplot(5,2,1);plot(time_25_11,tm_25_11);title('25-11-2021');
subplot(5,2,2);plot(time_29_11,tm_29_11);title('29-11-2021');
subplot(5,2,3);plot(time_30_11,tm_30_11);title('30-11-2021');
subplot(5,2,4);plot(time_09_12,tm_09_12);title('09-12-2021');
subplot(5,2,5);plot(time_10_12,tm_10_12);title('10-12-2021');
subplot(5,2,6);plot(time_16_12,tm_16_12);title('16-12-2021');
subplot(5,2,7);plot(time_27_12,tm_27_12);title('27-12-2021');
subplot(5,2,8);plot(time_28_12,tm_28_12);title('28-12-2021');
subplot(5,2,9);plot(time_10_01,tm_10_01);title('10-01-2022');
subplot(5,2,10);plot(time_11_01,tm_11_01);title('11-01-2022');

%Calculation of the value of Skewness and Kurtosis from the output signal
%after 10min moving average
skewness_25_11 = skewness(tm_25_11);
kurtosis_25_11 = kurtosis(tm_25_11);
skewness_29_11 = skewness(tm_29_11);
kurtosis_29_11 = kurtosis(tm_29_11);
skewness_30_11 = skewness(tm_30_11);
kurtosis_30_11 = kurtosis(tm_30_11);
skewness_09_12 = skewness(tm_09_12);
kurtosis_09_12 = kurtosis(tm_09_12);
skewness_10_12 = skewness(tm_10_12);
kurtosis_10_12 = kurtosis(tm_10_12);
skewness_16_12 = skewness(tm_16_12);
kurtosis_16_12 = kurtosis(tm_16_12);
skewness_27_12 = skewness(tm_27_12);
kurtosis_27_12 = kurtosis(tm_27_12);
skewness_28_12 = skewness(tm_28_12);
kurtosis_28_12 = kurtosis(tm_28_12);
skewness_10_01 = skewness(tm_10_01);
kurtosis_10_01 = kurtosis(tm_10_01);
skewness_11_01 = skewness(tm_11_01);
kurtosis_11_01 = kurtosis(tm_11_01);

%Smooth of the temperature signal using a moving average filter with a
%sliding window of 90min
tm_25_11_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_25_11);
tm_29_11_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_29_11);
tm_30_11_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_30_11);
tm_09_12_90=filtfilt(ones(size((1:4*60*80)))/(4*60*80),1,temperature_09_12);
tm_10_12_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_10_12);
tm_16_12_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_16_12);
tm_27_12_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_27_12);
tm_28_12_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_28_12);
tm_10_01_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_10_01);
tm_11_01_90=filtfilt(ones(size((1:4*60*90)))/(4*60*90),1,temperature_11_01);

%Plot the output signal after 90min moving average
figure(4)
subplot(5,2,1);plot(time_25_11,tm_25_11_90);title('25-11-2021');ylim([31,37]);
subplot(5,2,2);plot(time_29_11,tm_29_11_90);title('29-11-2021');ylim([31,37]);
subplot(5,2,3);plot(time_30_11,tm_30_11_90);title('30-11-2021');ylim([31,37]);
subplot(5,2,4);plot(time_09_12,tm_09_12_90);title('09-12-2021');ylim([31,37]);
subplot(5,2,5);plot(time_10_12,tm_10_12_90);title('10-12-2021');ylim([31,37]);
subplot(5,2,6);plot(time_16_12,tm_16_12_90);title('16-12-2021');ylim([31,37]);
subplot(5,2,7);plot(time_27_12,tm_27_12_90);title('27-12-2021');ylim([31,37]);
subplot(5,2,8);plot(time_28_12,tm_28_12_90);title('28-12-2021');ylim([31,37]);
subplot(5,2,9);plot(time_10_01,tm_10_01_90);title('10-01-2022');ylim([31,37]);
```

```
subplot(5,2,10);plot(time_11_01,tm_11_01_90);title('11-01-2022');ylim([31,37]);

%Perfom the difference between original signal and processed signal with
%90min moving average
temp_25_11 = temperature_25_11-tm_25_11_90;
temp_29_11 = temperature_29_11-tm_29_11_90;
temp_30_11 = temperature_30_11-tm_30_11_90;
temp_09_12 = temperature_09_12-tm_09_12_90;
temp_10_12 = temperature_10_12-tm_10_12_90;
temp_16_12 = temperature_16_12-tm_16_12_90;
temp_27_12 = temperature_27_12-tm_27_12_90;
temp_28_12 = temperature_28_12-tm_28_12_90;
temp_10_01 = temperature_10_01-tm_10_01_90;
temp_11_01 = temperature_11_01-tm_11_01_90;

%Output plot of the difference between original signal and processed signal with
90min moving average
figure(5)
subplot(5,2,1);plot(time_25_11,temp_25_11);title('25-11-2021');ylim([-1.5,1.5]);
subplot(5,2,2);plot(time_29_11,temp_29_11);title('29-11-2021');ylim([-1.5,1.5]);
subplot(5,2,3);plot(time_30_11,temp_30_11);title('30-11-2021');ylim([-1.5,1.5]);
subplot(5,2,4);plot(time_09_12,temp_09_12);title('09-12-2021');ylim([-1.5,1.5]);
subplot(5,2,5);plot(time_10_12,temp_10_12);title('10-12-2021');ylim([-1.5,1.5]);
subplot(5,2,6);plot(time_16_12,temp_16_12);title('16-12-2021');ylim([-1.5,1.5]);
subplot(5,2,7);plot(time_27_12,temp_27_12);title('27-12-2021');ylim([-1.5,1.5]);
subplot(5,2,8);plot(time_28_12,temp_28_12);title('28-12-2021');ylim([-1.5,1.5]);
subplot(5,2,9);plot(time_10_01,temp_10_01);title('10-01-2022');ylim([-1.5,1.5]);
subplot(5,2,10);plot(time_11_01,temp_11_01);title('11-01-2022');ylim([-
1.5,1.5]);

%Calculate standard deviation
sd_25_11 = std(temp_25_11);
sd_29_11 = std(temp_29_11);
sd_30_11 = std(temp_30_11);
sd_09_12 = std(temp_09_12);
sd_10_12 = std(temp_10_12);
sd_16_12 = std(temp_16_12);
sd_27_12 = std(temp_27_12);
sd_28_12 = std(temp_28_12);
sd_10_01 = std(temp_10_01);
sd_11_01 = std(temp_11_01);

%% ACCELERATION %%

%Load the measurements
load('acceleration_25_11_2021.mat');
load('acceleration_29_11_2021.mat');
load('acceleration_30_11_2021.mat');
load('acceleration_09_12_2021.mat');
load('acceleration_10_12_2021.mat');
load('acceleration_16_12_2021.mat');
load('acceleration_27_12_2021.mat');
load('acceleration_28_12_2021.mat');
load('acceleration_10_01_2022.mat');
load('acceleration_11_01_2022.mat');

%Calculate mean acceleartion
mean_25_11 = mean(accelerometer_25_11);
```

```
mean_29_11 = mean(accelerometer_29_11);
mean_30_11 = mean(accelerometer_30_11);
mean_09_12 = mean(accelerometer_09_12);
mean_10_12 = mean(accelerometer_10_12);
mean_16_12 = mean(accelerometer_16_12);
mean_27_12 = mean(accelerometer_27_12);
mean_28_12 = mean(accelerometer_28_12);
mean_10_01 = mean(accelerometer_10_01);
mean_11_01 = mean(accelerometer_11_01);

%Plot mean of acceleration vs. standard deviation
figure(6)
hold on
plot(sd_25_11,mean_25_11)
plot(sd_29_11,mean_29_11)
plot(sd_30_11,mean_30_11)
plot(sd_09_12,mean_09_12)
plot(sd_10_12,mean_10_12)
plot(sd_16_12,mean_16_12)
plot(sd_27_12,mean_27_12)
plot(sd_28_12,mean_28_12)
plot(sd_10_01,mean_10_01)
plot(sd_11_01,mean_11_01)
```

4 Code to analyse temperature changes vs. acceleration

```
load('measurement204.mat') %Load the measurement wanted to be
proced
table_temperature = t(30:end,:); %Delete the first data (time when
the user put the wristband)
date = datetime(table_temperature{:,1}); %Vector of date
hour = table_temperature{:,2}; %Vector of time
plot_time = datetime(strcat(string(date)," ",string(hour)));
%Vector of datetime

%% TEMPERATURE %%
TA = table_temperature{:,3}; %Vector of ambient temperature
TS = table_temperature{:,4}; %Vector of skin temperature

Tref = (TS-(TA*(0.042/0.5)))/(0.964-((0.575*0.042)/0.50)); %Skin
temperature calibrated

%Plot of skin temperature with/without calibration
figure(1)
plot(plot_time,TS)
hold on
plot(plot_time,Tref)

%Sleep cycle from the user fully asleep to awake
skin = TS(sleep:wakeup); %Skin temperature of the sleep cycle
ambient = TA(sleep:wakeup); %Ambient temperature of the sleep
cycle
sleeptime = plot_time(1240:3800); %Datetime of the sleep cycle
Tskin = (skin-(ambient*(0.042/0.5)))/(0.964-((0.575*0.042)/0.50));
%Skin temperature calibrated of the sleep cycle

%Plot of the skin temperature with/without calibration of the
sleep cycle
figure(2)
plot(sleeptime,skin)
hold on
plot(sleeptime,Tskin)

%% ACCELERATION %%

x = table_temperature{:,5}; %X coordinate vector
y = table_temperature{:,6}; %Y coordinate vector
z = table_temperature{:,7}; %Z coordinate vector
acceleration = sqrt(x.^2+y.^2+z.^2)/360; %Total acceleration

%Plot of the temperature changes and acceleration
```

```
subplot(2,1,1)
plot(plot_time, TS)
hold on
plot(plot_time,Tref)
xline(plot_time(sleep))
xline(plot_time(wakeup))
subplot(2,1,2)
plot(plot_time,acceleration)
xline(plot_time(sleep))
xline(plot_time(wakeup))
```

5 Questionnaire

AUTOREGISTRE DE SON

1. **Nom**

2. **Edat**

3. **Gènere**
   Marqueu només un oval

   - ◯ Home
   - ◯ Dona
   - ◯ Altres

4. **Data de resposta**

   Exemple: 26 de Novembre de 2021

5. **Dia de la setmana**
   Marqueu només un oval

   - ◯ Dilluns
   - ◯ Dimarts
   - ◯ Dimecres
   - ◯ Dijous
   - ◯ Divendres
   - ◯ Dissabte
   - ◯ Diumenge

6. **Horari de resposta***
   Marqueu només un oval

   - ◯ Mati (al llevar-se) *Ves a la pregunta 7*
   - ◯ Nit (abans d'anar a dormir) *Ves a la pregunta 6*

Aurorregistre Matí

7. **La passada nit vaig anar a dormir... (indicar hora d'anar al llit)**

   Exemple: 08:30am

8. **Aquest matí m'he llevat... (indicar hora)**

   Exemple: 08:30am

9. **Ahir a la nit em vaig adormir...**
   Marqueu només un oval
   - ◯ Fàcilment
   - ◯ Al cap d'una estona d'anar al llit
   - ◯ Amb dificultat

10. **Durant la nit t'has despertat?**
    Marqueu només un oval
    - ◯ Si
    - ◉ No *Ves a la pregunta 12*

11. **Quin és el temps total (en minuts) que has estat despert/a durant la nit?**

12. **Aquesta nit he dormit un total de (indicar hores:minuts, per exemple: 07:30)**

13. **La meva son s'ha vist destorbada per... (indica breument factors disruptors com per exemple: soroll, llum, mascotes, al·lèrgies, temperatura, malestar, estrès, etc.)**

14. **Al llevar-me aquest matí em sentia**
    Marqueu només un oval
    - ◯ Fresc/a (recuperat/da)
    - ◯ Una mica fresc/a (recuperat/da)
    - ◯ Fatigat/da

15. **Si us plau, indica, si escau, algun altra factor que consideris que ha pogut afectar al teu son (torns laborals, cicle menstrual en el cas de les dones, etc.)**

Autorregistre abans de dormir

16. **Durant el dia d'avui, has consumit begudes amb cafeïna o estimulants (cafè, té, refrescos amb cafeïna, taurina, etc.)**
Marqueu només un oval

- ◯ Sí
- ◯ No *Ves a la pregunta 18*

17. **Si us plau, indica les racions* consumides per a cadascuna de les franges horàries que s'indiquen**
**1 ració = 1 tassa de cafè/té o 1 got de refresc*
Marqueu només un oval per fila

	6-12h	12-16h	16-20h	20-6h
1 Ració				
2 Racions				
3 Racions				
Més de 3 Racions				

18. **Avui he practica, al menys 20 minuts d'exercici físic d'intensitat moderada-vigorosa... (selecciona la franja horària)**
Marqueu només un oval

- ◯ 6-12h
- ◯ 12-16h
- ◯ 16-20h
- ◯ 20-6h
- ◯ No n'he practicat

19. **Has pres alguna medicació?**
Marqueu només un oval

- ◯ Sí
- ◯ No *Ves a la pregunta 21*

20. **Quina medicació has pres?**

21. **Has fet migdiada?**
Marqueu només un oval

- ◯ Sí
- ◯ No *Ves a la pregunta 23*

22. Quant de temps has dormit durant la migdiada (indicar els minuts)?

23. Durant el dia, amb quina probabilitat creus que podries adormir-te mentre feies les activitats diàries?

Marqueu només un oval

- ◯ Cap probabilitat
- ◯ Probabilitat baixa
- ◯ Probabilitat moderada
- ◯ Alta probabilitat

24. Indica en la següent escala, en general, com ha estat el teu estat d'ànim durant el dia d'avui

Marqueu només un oval

	1	2	3	4	5	6	7	8	9	10	
Dolent/Pèssim	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Excel·lent

25. Aproximadament, en les 2-3 hores prèvies a anar-te al llit he consumit...

Seleccioneu totes les opcions que corresponguin

- Alcohol
- Àpats copiós
- Cafeïna o estimulants
- No he consumit cap dels anteriors

26. Una hora abans d'anar a dormir, la meva rutina pre-son ha consistit en... (indica breument activitats com per exemple: llegir, utilitzar dispositius electrònics (TV, ordinador, mòbil, tablet, etc.), dutxar-se o banyar-se, fer estiraments o exercicis de relaxació i/o meditació, etc.):