POLITECNICO DI TORINO

FACULTY OF ENGINEERING

Master's Degree course in Biomedical Engineering

Master's Degree Thesis

Temperature monitoring system for the assessment of thermal comfort in sports and work clothes

Candidate:

Francesco Ridolfi

Supervisors:

Prof. Alberto Vallan

Academic year 2019/2020

Abstract

The control of body temperature in humans is of vital importance because temperature must be maintained within a certain range, called neutrality, in order to guarantee the human thermal well-being and homeostasis. Human is a homoeothermic species, which means that it maintains its body temperature in a certain range of values, for example the oral cavity must remain between 36.5 and 37.5 °C.

However, if the external conditions are not favorable, adaptive thermogenesis may be necessary, by applying specific physiological methods, it heats or cools the body in order to maintain its temperature within a certain range, beyond which it could face frostbite, coma or death.

Clothing plays an important role here; several studies on human thermal well-being deal with two main themes: the first concerns the development and production of sports clothing, while the second concerns the development and production of work clothing. Sports clothing should not just allow adequate heat dissipation in the case of physical activity in a warm environment, but it should also thermally insulate the subject in the case of physical activity in a cold environment; these two goals should be achieved without limiting the athlete's movements. Work clothing is more critical because individuals can be subjected to extreme temperatures; they should protect the human body from chemical and biological agents and radiation, they should dissipate the sweat produced by the body and they should dissipate the heat related to temperatures to which they can be subjected.

In order to assess the effectiveness of work and sports clothing, it is therefore necessary to use a device that allows the measurement of human temperature to be carried out for extended time spans and using multiple temperature sensors, in order to better understand the dynamics of heat distribution in different parts of the body surface. In future, smart sports clothing which integrates these monitoring devices can provide a real-time feedback concerning the subject's thermal conditions.

The purpose of this thesis is to create a wearable temperature monitoring system equipped with miniaturized thermistors suitable to be fixed on the subject skin without interfere with

the subject activity. The developed system is able to manage six sensors whose measurements can be logged on the wearable device and sent via radio to a receiver connected to a Personal Computer. Moreover, the wearable system also embed a display useful to read the sensor measurements.

The wearable system has a rechargeable battery that guarantees measurement sessions up to 12 h. The PC collects the measurements sent by the wearable device and displays the results using a graphical interface. Data saving and loading capabilities have been embedded too. The hardware of the wearable system and of the receiver takes advantage of a micro controller platform produced by Adafruit. The microcontroller were programmed in C language and the program running on the PC was written in Python; in details, the latter has several useful features, including gathering new data, adding data, uploading data in order to visualize them, deleting data, visualizing just specific channels, monitoring battery voltage or changing data samples gathering time.

Preliminary tests were carried out monitoring the fingers temperature when the hands are protected with sport gloves. Then thanks to the small sensor dimension was possible to monitor the finger tip temperature without cause any discomfort to the subject.

Contents

Introduction

Monitoring the human skin temperature is important to quantitatively assess the human well-being and to prevent injuries such as heat strokes and hypothermia. This is particularly important when the subject performs physical activity or, in more serious circumstances, when it is exposed to extreme temperatures because of specific works.

This thesis is focused on the measurement of the human skin temperature during physical activity. Part of this work was devoted to the analysis of the human well-being range and its changes related to specific causes, such as metabolism, clothing and physical activity.

Temperature monitoring can be performed using several kind of temperature sensors, which are placed on the body of the subject in order to constantly observe the parameter on the skin surface. Sensing devices currently available on the market were analyzed and reviewed.

A wearable device useful to monitor skin temperature during physical activity was designed, realized and tested. The developed device is able to collect and send data to a receiver using a radio link. The receiver is connected to a personal computer where date can be displayed using a graphical interface. Moreover it stores all the measurements for a subsequent analysis.

The thesis is organized as follow:

- Chapter 1 describes the physiology of heat generation in the human body, analyzing the causes that change its temperature from the literature.

- Chapter 2 provides an overview of devices currently on the market that send wireless data to a receiving source used for sport activity.

- Chapter 3 describes the system created, first in a summary way and then dwelling more on the individual components used.

- Chapter 4 shows flow charts used for programming respectively the transmitter, receiver and graphic interface, explaining how the system in question was designed.

- Chapter 5 shows the results of the system and the tests carried out.

- Chapter 6 examines the conclusions and includes an analysis on the hypothetical future developments of this device.

Chapter 1

Factors influencing the change of the body temperature.

Human thermoregulation is one of the basic functions of life. Metabolic energy is converted by the body into mechanical and thermal energy with most of it (30 to 70%) [1] being converted into heat.

Although the system is considered inefficient, it allows our body to survive and generate enough heat to maintain homeothermia.

Homeothermic species, such as humans, maintain their body temperature (Tc) at a fixed value.

This regulation of internal temperature with the surrounding environment is a result of evolution, and, although it has a significant energy cost, it is of fundamental importance in order to withstand colder environments. The basal heat production would not be sufficient to keep the Tc constant.

The thermogenic mechanisms adopted by humans [2], generation and dissipation, are heat saving and vary depending on what is required for the body to adapt to sudden changes in external temperature.

The body should be able to produce significant quantities of heat. Otherwise, the ambient temperature range to which it can be subjected is limited.

Therefore, we should also distinguish the types of thermogenesis that we use to keep the body's core temperature constant. The first type is mandatory thermogenesis [2], which allows the body to stay in thermal equilibrium with the surrounding environment. If the external temperature falls within a defined range of thermoneutrality, no other methods are needed to produce heat.

Next comes adaptive thermogenesis [2]: it takes place when the external temperature leaves the range of thermoneutrality. In this case, the body needs additional heat to maintain its vital functions and it implements mechanisms such as vasoconstriction, piloerection, shivering or more complex metabolic mechanisms.

1.1 Analysis of the metabolic heat production.

Metabolism is a fundamental aspect of thermal comfort. It is analyzed as the trend of the metabolic rate and is influenced by the insulation of the clothing and by external thermal conditions to which the individual is subjected.

The equation used to analyze this phenomenon is second-order polynomial [3]; the result obtained is that the metabolism is lower in a neutral condition, and it increases in case the temperature gets warmer or colder.

PMV is an index among the models based on human body at balance calculation [3]. It is a function of the metabolic rate and insulation of clothing with regard to human physiological regulation. This includes: air temperature, relative humidity and air speed in regard to the surrounding environment.

M and I are related to human physiological regulation and behavioral regulation. The other parameters refer to the thermal environment.

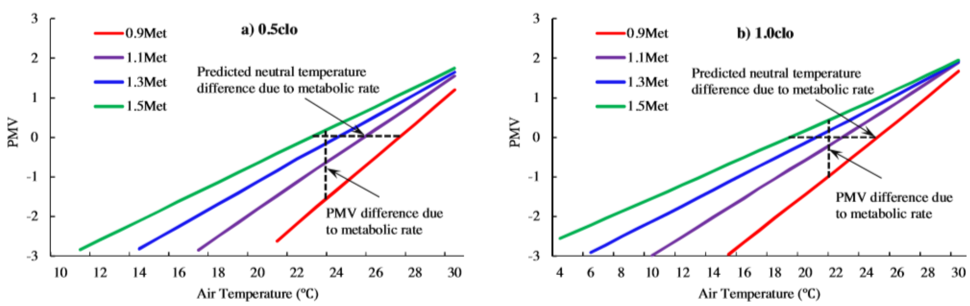


Figure 1.1: Effect of metabolic rate on PMV variance with insulation levels of 0.5 clo and 1.0 clo

In figure 1.1 an expected result is observed: the PMV increases linearly depending on the function of the thermal insulation of the clothing as the air temperature changes; in the case of 0.5 clo, it initially starts from a higher value, then progresses to the same value of 1.0 clo at around 30 °C.

MET is equal to 58.2 W/(m$\hat{2}$), and represents the metabolic rate [3].

Clo is a measure of thermal resistance, it includes the insulation provided by layers of trapped air between skin and clothing and insulation value of clothing itself [4]; higher is this value, higher is the insulation between body and external temperature.

In literature, considering the metabolic rate as a constant value is a common assumption. However, this is not true in most cases.

Levels for the determination of the metabolic rate.

Level	Method	Accuracy
1. Screening	1A: Classification according to occupation 1B: Classification according to activity	Rough information. Very great risk of error
2. Observation	2A: Group assessment tables 2B: Tables for specific activities	High error risk Accuracy: ±20%
3. Analysis	Heart rate measurement under defined conditions	Medium error risk Accuracy: ±10%
4. Expertise	4A: Measurement of oxygen consumption 4B: Doubly labelled water method 4C: Direct calorimetry	Errors within the limits of the accuracy of the measurement or of the time and motion study. Accuracy: ±5%

Table 1.1: Levels and methods for the determination of the metabolic rate.

There are 8 methods with 4 different levels of precision rates to evaluate the metabolic heat production in the human body(Table 1.1) [3], Refers to 4a method were studied metabolic rate, the heart rate, the blood flow and the skin temperature; in particular metabolic rate is described by equation 1.1.

$$M = \frac{21(0.23Rq + 0.77)Q02}{Ad} \tag{1.1}$$

- M is the metabolic rate (W / m2).

-Rq is the respiratory quotient, which is the molar ratio of Qo2(L / min) exhaled and Qco2(L / min) inhaled.

- Qo2 and Qco2 are respectively the volumetric rate of carbon dioxide production and oxygen consumption (ml / s, at conditions of 0 ° C, 101.3 kPa).

- Ad is the surface of Dubois (m$\hat{2}$). It can be determined by following the empirical equation 1.2[3]:

$$Ad = 0.202H^{0.725}W^{0.425} \tag{1.2}$$

where H is height (m) and W is weight (kg).

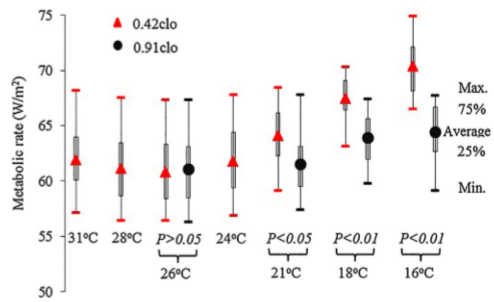


Figure 1.2: Metabolic rate in different thermal conditions

Fig. 1.2 illustrates the metabolic rate under different thermal conditions [3]. Despite the individual differences, the general metabolic rates increase when the thermal conditions move away from the neutral zone(around 32 °C). In fact at 16 °C, we can see that the metabolic rate increases by about 3 MET at 0.9 clo and about 8 MET at 0.42 clo; this is an expected result because with more clo the human body is more isolated thanks to clothing, and the metabolism doesn't have the necessity to create heat; if we measured the internal temperature with the device created, we would notice that with the same external temperature the body has a greater need to create heat if it is less dressed.

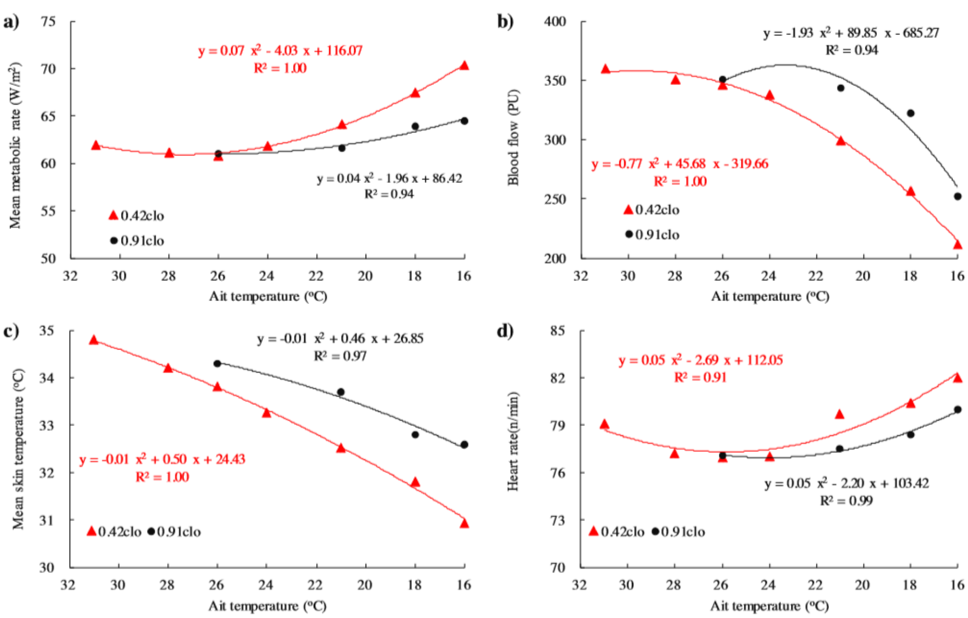


Figure 1.3: Physiological response in different temperatures and clothing

Fig. 1.3 further illustrates the overall changing trend of other parameters in different thermal conditions, such as the mean metabolic rate, blood flow, mean skin temperature and heart rate; as regards the mean metabolic rate it increases with decreasing temperature, blood flow decreases with decreasing temperature and the trend between 0.42 clo and 0.91 clo is different; about the mean skin temperatures, 0.42 clo and 0.91 clo have a more similar trend, and as expected, the temperature of hands decreases as the temperature decreases ; in fact with a greater thermal insulation it decreases less. The heartbeat have a minimum around 26 °C, however, moving from this temperature the heart beats faster with a drop or elevation in temperature.

This is a demonstration of how clothing affects different parameters internally in the human body: it is known that these parameters influence the subject's temperature in turn.

Consequently, it is essential to consider the quantity of clothing which the subject wears during measurements by the device, in order not to incur in unexpected results.

1.2 Pathologies related to the change in body temperature.

Body temperature can deviate from its physiological values due to some pathologies, which are grouped into hyperthermia and hypothermia.

There are three types of Hyperthermia:

- Heat collapse: it can be generally identified in elderly individuals, in whom thermoregulation is not particularly effective, in non-acclimated individuals and those who carry out particular physical activity, exposing themselves to high environmental temperatures. This pathology varies the body temperature between 38 and 39 °C.

- Malignant hyperthermia: this pathology is due to a malformation of the calcium channel in the cells of the skeletal muscle. The calcium channel releases excess ions for the hydrolysis of ATP in ADP, which are transferred to the sarcoplasmic reticulum and mitochondria.

- Heat stroke: it could happen because of an excessive exposure to heat. This type of stroke is fatal in 50% of cases.

  These individuals' body temperature could reach up to 41 °C. The subjects end up unconscious. It is essential to cool the body to lower the temperature.

Hypothermia is a temperature decreasing. The lower is the body temperature, the lower is the metabolic consumption of oxygen; if this process continues excessively over time, it could also lead to the risk of permanent brain injury or death.

1.3 Changing of the human temperature due to sport

Heat production increases sharply with exercise. It dynamically contracts the skeletal muscle and it further increases during the initial stages of training [1].

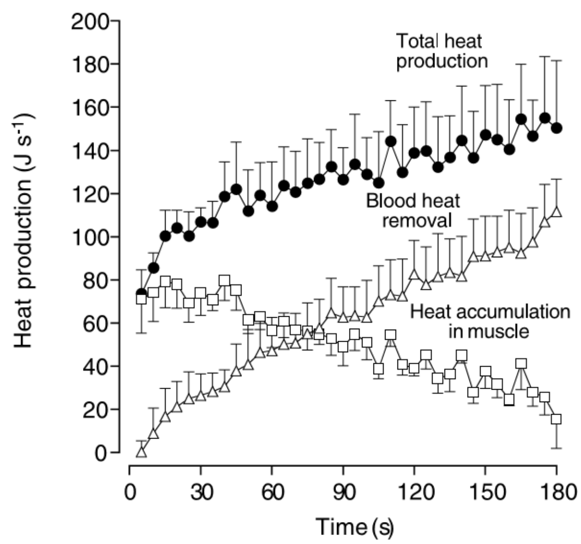


Figure 1.4: Heat production during physical exercise

The graph in figure 1.4 shows that the heat accumulated in the muscle decreases with the progress of physical exercise. At the beginning of the exercise the heat reaches its maximum (80 j / s). The blood heat removal starts from zero and then increases.

The total heat production, which increases overtime, is the sum of the blood heat removal and the heat accumulation in muscles.

This heat exchange is particularly significant in cold environments. It produces a high temperature gradient between the muscles, the subcutaneous tissue and the skin, requiring less thermogenesis mechanisms [2]. Whereas, when the external climate is hot, there is no significant temperature gradient.

There can be significant changes in the convective heat exchange between limbs and the bust, in case their temperature or blood flow is altered because of environmental or physical stress. When the exercise is carried out in ischemic conditions, the blood is accumulated in muscles because of some circulatory blockage in the blood [1]. In addition, the thermal and metabolic needs of our body depend on the hemodynamic response, which changes with respect to certain parameters, such as the ambient temperature, the physical exercise carried out, the duration and intensity of this[1].

The physical activity leads to a greater tachycardia; therefore it increases blood flow and heat exchanged in the body. We should also distinguish the type of physical activity, which can be small or large muscle exercise;

In the first case we have exercises on single parts of the body, which consequently require lower energy expenditure and a blood flow towards a specific part of the body: therefore, the temperature will be increased only in a certain limb.

Instead, in other more complex exercises that require the use of many joints, such as swimming or rowing, the heat is equally distributed throughout the body. The heart will then have to pump blood throughout the body, also depending on the duration of the exercise and external environmental conditions, which change metabolic and thermal needs of the local and systemic blood flow.

The combination of heat resistance and intensity of the exercise determines the amount of oxygen delivered to the brain, heart and muscles, blood pressure and regulation of the temperature. All these factors change the body temperature, so we can say that doing sports also increases significantly by 1-2 °C.

Another important feature concerns dehydration; the exercise in a warm environment determines a greater sweating and, therefore, dehydration [1].

From experimental tests can be observed that the heat request was apparently the same. Although in the case of hot external temperature the subject had lost the 4% of liquids mass, the cardiac output was not reduced. Therefore, we can say that blood flow and cardiac output are lower while practicing an exercise in a warm environment. Dehydration can explain the discrepancies of the cardiac output depending on the environment around the analyzed subject[1].

As regards the heat at the extremities of the limbs, studies reveal that the heat stress increases the blood flow to the arms and legs, whereas in the case of cold stress, limb perfusion is reduced[1].

By studying the limbs individually, it can be observed that during the heat stress process some elements increase. Those elements are: the blood flow in the leg tissues, the content of the femoral venous O2 and the muscle oxygenation[1].

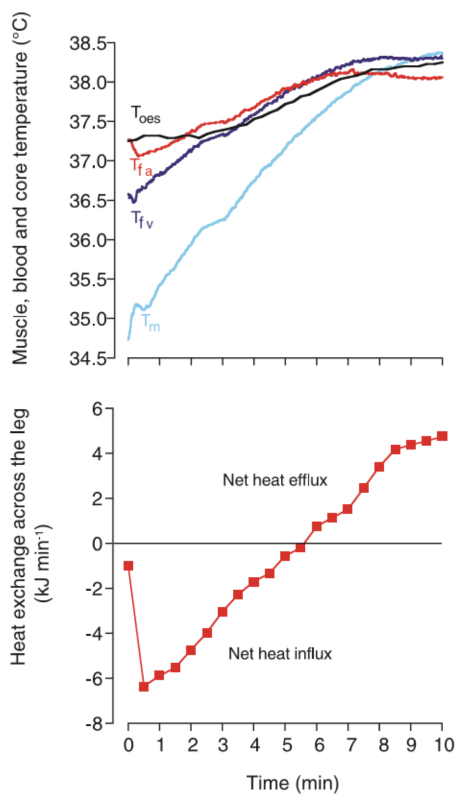


Figure 1.5: Example of temperature and heat changing in human legs during cycling exercise

In figure 1.5 a test, concerning a cycling exercise, was performed[1]. It can be observed that the core temperatures (Toes) and femoral arterial blood temperatures (Tfa) are higher than muscle temperatures (Tm) and femoral venous temperatures (Tfv); however these last two elements increase rapidly as the exercise proceeds, since the time required to warm the Toes is lower. Afterwards, it can be seen a net heat flow, in fact, after a certain period of time (here 5.5 minutes) the leg has fully heated.

The blood carries the heat inside the body in relation to the temperature of the blood and the blood flow. The transfer of heat in the main arteries and veins, which flows into the limbs, is bidirectional.

In normal resting conditions the temperature of muscles, limbs and blood is significantly lower than the internal arterial temperature. It is due to the fact that there is a rapid thermal balance between tissues and vessels.

Analyzing the negative gradient mayor supply vessels of the resulting arteriovenous temperature indicates that the heat is transferred from the upper nucleus to the extremities, rather than vice versa.

This net body-to-body heat transfer helps limbs to maintain their temperature when metabolic heat production is low. For example, according to figure 1.6, the leg VO2 is normally about 25 mlmin-1 in the resting state, corresponding to a total heat production of the leg of 0.5 kJmin-1. [1]

These simple estimates demonstrate that more heat is transferred to the resting leg. It implies that the temperature of the limb tissue will decrease if its circulation is stopped and the heat dissipation in the surrounding environment is kept constant.

By practicing exercise the body temperature mainly increases in the limbs, whereas it decreases when the physical exercise is finished.

If this process does not happen, it is because there are perfusion problems due to a previously described pathology; therefore it is fundamental to monitor the body temperature with a specific device to control the thermal well-being of the subject.

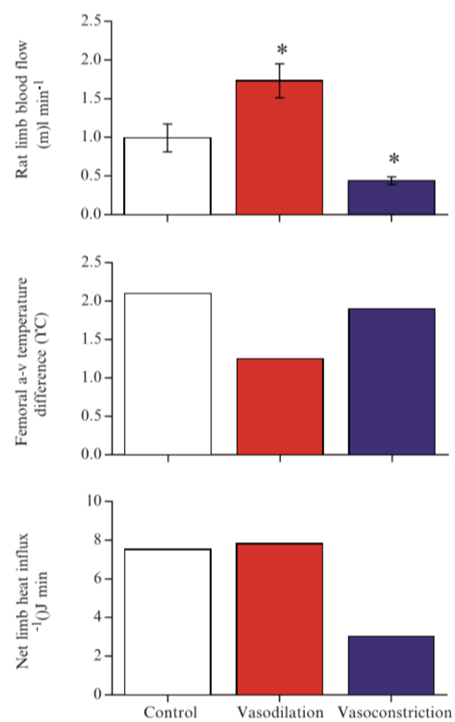


Figure 1.6: Test of blood temperature and heat exchange in a rat's hind limb

The net heat of the limbs was the same, as the increase in blood flow corresponds to a decrease in femoral arteriovenous temperature.

As for the femoral artery, however, when the blood flow was reduced, the heat flow in the hind limb accordingly decreased, so the temperature difference remained unchanged.

An experiment was conducted on rats [1] ; the results were that, in case the blood flow in the limbs increases, the amount of net heat, which is transferred from the trunk to the limbs, not necessarily increases. This fact happens due to the compensatory regulations of the fabric heat exchange - blood inside the leg tissues.

However, if the blood flow decreases, it is likely that there is a lower heat in the limbs; It was observed that the temperature of the human leg, during a resting phase, also decreases up to 0.5 degrees and the blood flow increases from 0.4 to 8 liters per minute for the infusion of ATP [1].

During the exercise the production of heat also increases, not just the convective heat exchange or the perfusion of the tissues.

Initially, in the practice of physical exercise, the temperatures of the contracting muscle and the femoral venous blood increase more than the temperatures of the femoral artery blood. In this phase a negative gradient of the femoral arteriovenous blood temperature prevails; therefore, in normal environmental conditions, a larger quantity of heat is transferred from the upper part of the body to the limbs. This process can be observed in figure 1.6 [1].

After a few minutes of exercise, the temperature of the muscular venous blood increases more than the arterial blood, and the internal temperature of the body consequently increases. The heat transferred from the limbs to the bust then becomes positive, and, in case the exercise is moderate, it increases until it reaches a plateau.

1.4 Other factors that affect the body temperature: age and microclimes

The practice of sports exercise radically changes the blood flow and the changing process of the body temperature. However, it is not the only factor to be taken into consideration: from the literature it is clear that some parameters change over the years, and these are basal metabolic rate (BMR), body weight (BW) and cardiac output (CO) [5].

It is known that thermal well-being is one of the main limits regarding the health of elderly people, who, compared to young people, have a reduced range of thermoneutrality.

This fact can put them at risk in case of extreme thermal stress; therefore, it is good to keep their physiological parameters monitored, in order to avoid hypothermia or lack of heat.

The IESD-vial model [5] analyzes the change in physiological parameters with respect to age. It mainly analyzed the basal metabolic rate, cardiac output and body weight. It also studied situations including vasoconstriction, vasodilation, sweating and shivering. The result was that basal metabolic rate is the most critical parameter for thermal comfort in the elderly. It was also shown an excessive exposure to a cold environment could lead to an excessive cooling in the body, due to the reduced metabolism.

Another consideration should be made with respect to the environment outside the subject: if, for example, the subject practices physical activity in a park, the thermal comfort will be different than practicing in an urban area, due to the vegetation present in green areas [6].

Breathable plants release water vapor into the surrounding environment, decreasing the air temperature and increasing relative humidity. Urban vegetation always plays an important role in the urban climate, producing microclimates, especially during the hot season.

The PCI (Park Cool Island) index defines the cooling of a park, and its value corresponds to the difference between the temperature of the urban area and the park under analysis[6]. However, it is not precise enough to state the specific temperature and humidity of the various areas of the park;

Meteorological station	Area name	Area characteristics	Total area size (ha)	Altitude (m)
1	Giardino Torrigiani	Private garden with trees	6.90	49
2	Giardino della Gherardesca	Private garden with trees	6.27	51
3	Giardino di Villa Agape	Private garden with trees	4.00	145
4	Giardino dei Semplici	Botanical garden	2.12	51
5	Giardino Convento del Carmine	Private garden with trees	0.07	46
6	Cortile alberato	Private courtyard with trees	0.02	51
7 (reference)	Osservatorio Ximeniano	Urban area (historic center)		75

Table 1.2: Park analysed.

Through analysis, characteristics of some green areas were found. They are shown in table 1.2.

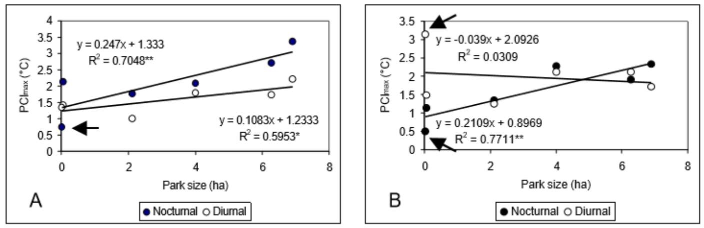


Figure 1.7: PCI max in green areas in summer

Park No. 3 was located on the slope of a small hill. In fact, its altitude is a little higher than the others [6].

It was seen that the PCI max of the green areas changed during the day according to the size of the area, between 1 and 2 °C (fig 1.7), whereas at night there were greater differences in the PCI max, that even reached up to 3.5 °C. Therefore, these factors are fundamental for monitoring the athlete's temperature, since a difference of 3.5 °C in external temperature cannot be overlooked [6].

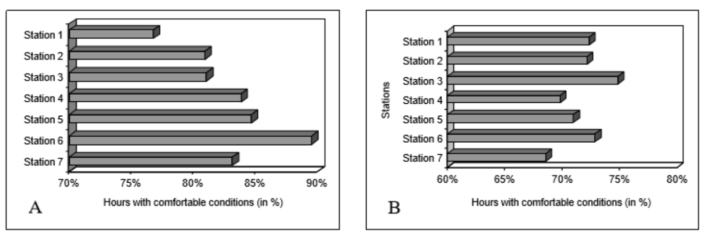


Figure 1.8: Percentage of hours with comfort conditions in winter and summer

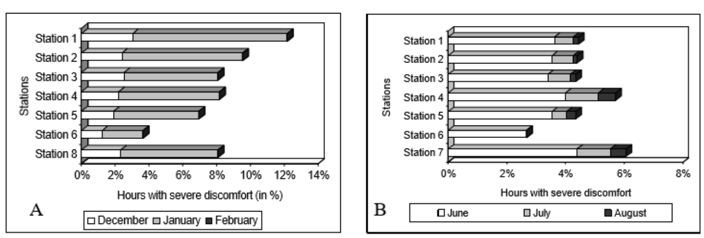


Figure 1.9: Percentage of hours with discomfort conditions in winter and summer

As for the biometereological indices, the comfortable conditions in figure 1.8 and the uncomfortable conditions related to the analyzed areas in figure 1.9 can be observed. As expected, in winter the area with the highest percentage of comfort is the courtyard, which can reach up to 90%, while in summer it approaches 73%. Looking at figure 1.9, it appears that the covered green area has less uncomfortable conditions: in winter its higher temperatures are given by the heating of the house, while in summer, as mentioned, from the shade of the walls.

The largest parks have a lower percentage of comfort and the largest percentage of uncomfortable hours, around 10%.

For these reasons,with regards to thermal comfort, it would be better to practice sports in smaller parks with regards to thermal well-being.

Chapter 2

Commercial acquisition systems for sports

There are several monitoring systems useful to measure skin temperature which embeds a wireless communication link. However, only few of them are also designed to be used during physical activity. This Chapter reports some relevant system and describes their technical characteristics.

2.1 MSR 147

Figure 2.1: Picture of acquisition system MSR 147

MSR147 is produced by MSR Electronics GmbH. It is an instrument equipped with Bluetooth and embeds a data memory that can record up to 1 million measurements [7].

It can measure the temperature of the skin and the level of humidity for long periods of time. It has 5 connectors for wired sensors: this element makes the device very versatile, being able to insert and remove sensors that are automatically recognized by the system. Moreover, it is possible to collect values via USB [7].

The system can be also managed using a an App for mobile phones. It has a led which, depending on the color, shows customizable alarms, the battery charge status or the record indicator [7].

Measured parameters	Working range	Accuracy	Storage rate
Temperature (ext. sensor)	-40...+125 °C	0.2°C (-10...+50 °C) ±1°C (-40...+125 °C)	1/s to every 12 h
Relative humidity with integrated temperature (ext. sensor)	0…100% rel. humidity -40...+125 °C	±1.8% rel. humidity (10…85%, 0…+40 °C) ±4% rel. humidity (85…95%, 0…+40 °C)	1/s to every 12 h
Air pressure absolute, with integrated temperature (int. sensor)	10…2000 mbar absolute, -20…+65 °C	±2 mbar (750…1100 mbar absolute, +25 °C)	1/s to every 12 h
3-axis-accele-ration (static) (int. sensor)	±15 g -20…+65 °C	±0.15 g (+25 °C)	1/s to every 12 h

Table 2.1: Features of MSR 147.

2.2 MSR 145

Figure 2.2: Picture of acquisition system MSR 145

This system is a different version form the same manufacturer and it is capable of taking up to 50 measurements per second [8]. It has a 900 mAh battery [8] and can make measurements for a period of even two years.

LEDs are present: the blue color shows that the device is collecting data, the red color indicates an alarm and the yellow color shows the battery status [8].

It has a USB connection, so the device can be easily connected to the computer. It also has two types of sensors, namely temperature and humidity [8];

Measured parameters	Working range	Accuracy	Measurement/ storage rate
Temperature	ext.: -10…+58°C	±0.1 °C (+5…+45°C) ±0.2 °C (-10…+58°C)	1/s to every 12 h
	ext.: -55...+125°C	±0.5 °C (-10…+65°C) ±2 °C (-55…+125°C)	
	int.: -20…+65°C	±0.5°C (-10…+65°C) ±2 °C (-55… +125°C)	
	ext.: 1 or 4 connectors for K-type thermocouples (excluding sensor) -250…+1200°C		
Relative humidity with integrated temperature	0…100% rel. humidity ext. -20…+85°C int. -20…+65°C	±2% rel. humidity (10…85%, 0…+40°C) ±4% rel. humidity (85…95%, 0…+40°C)	1/s to every 12 h
Air pressure absolute, with integrated temperature	0…2000 mbar absolute ext. -20…+85°C int. -20…+65°C	±2.5 mbar (750…1100 mbar absolute, +25°C)	10/s to every 12 h
	0…14 bar absolute -20…+65°C	±50 mbar (1…10 bar absolute, +25°C)	
Fluid pressure (external sensor) Material in contact with media: • Stainless Steel AISI 316L (DIN1.4404/1.4435) • O-Rings: Viton® 70° Shore	0…3000 mbar absolute -20…+85°C	±30 mbar	20/s to every 12 h
	0...30 bar absolute -20…+85°C	±300 mbar	
3-axis-acceleration opt. fast peak (1 kHz)	±15 g -20…+65 °C	±0.15 g (0...5 g, +25 °C) ±0.25 g (5...10 g, +25 °C) ±0.45 g (10...15 g, +25 °C)	50/s to every 12 h
Light	0…65‘000 lx	max. sensitivity at 500 nm	1/s to every 12 h

Table 2.2: Features of MSR 145.

2.3 MSR 160

Figure 2.3: Picture of acquisition system MSR 160

This further version is capable of 1000 samples per second [9].

Pluggable sensors of temperature, humidity and pressure can be added. An SD card can be added to save a billion measurements.

It includes a 900 mAh rechargeable battery [9]. The saved data can be brought to the PO or to the computer by a USB interface .

The weight is 80 grams and dimensions are 39x23x72 mm.

The temperature range is between -20 and +65 °C; the relative humidity is between 10% and 95% [9].

Measured parameters	Working range	Accuracy	Measurement/ storage rate
Temperature	int.: -20…+65 °C	±0.5 °C (-10…+65 °C)	1 /s to every 12 h
	ext.: -55…+125 °C	±0.5 °C (-10…+65 °C) ±2 °C (-55…+125 °C)	
Relative humidity with integrated temperature	0…100% rel. humidity -20…+65 °C	±2% rel. humidity (10…85%, 0…+40 °C) ±4% rel. humidity (85…95%, 0…+40 °C)	1 /s to every 12 h
Air pressure absolute, with integrated temperature	0…2000 mbar absolute -20…+65 °C	±2.5 mbar (750…1100 mbar absolute, +25 °C)	1 /s to every 12 h
	optional: 0…14 bar absolute -20…+65 °C	±50 mbar (1…10 bar absolute, +25 °C)	1 /s to every 12 h
3-axis-acceleration (Position)	±15 g -20…+65 °C	±0.15 g (+25 °C)	1 /s to every 12 h

Table 2.3: Features of MSR 160.

2.4 VitalPatch Biosensor

Figure 2.4: Picture of acquisition system VitalPatch Biosensor

Vital Patch Biosensor is a device created by Madgadget. It is a smart device and its dimensions are 4x12x2. The patch is disposable and it lasts up to 5 days: thanks to its short duration, it was possible designing a flexible and breathable device, so that it could suit the behavior of the body.

Its oval shape area permits the placement of two ECG / EKG leads; the patch can be stuck everywhere; however, the most appropriate part is above the bust (fig 2.5) [10].

To apply the patch it is enough to shave the selected section. Furthermore, after pressing a button a led will notify that the patch is active. Hereafter, once the location of the patient will be established, the patch will conduct 8 continuous measurements.

While the patch is worn, it should not be wetted: it is obviously important to avoid swimming and / or a direct exposure under the shower.

It is also possible to use a tablet as a point of connection between the device and the cloud; however, the internet connection is not strictly required because the device can collect up to ten hours of data [10].

The graphic interface of the tablet includes 7 frames which represent the measured data: the breathing and heart rate, ECG, body posture, pedometer, skin temperature and detection of drop. The eighth parameter is the variability of the heart rate, which is accessible only through the cloud [10].

The cloud is the control center of all patches.

The user can visualize on the cloud all described parameters, which are sent by all patches commissioned by a treatment center. All data can be downloaded in CSV format.

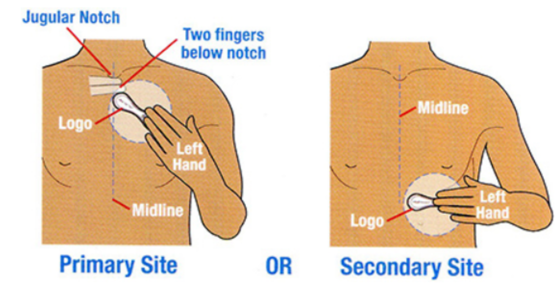


Figure 2.5: Example of application of VitalPatch Biosensor.

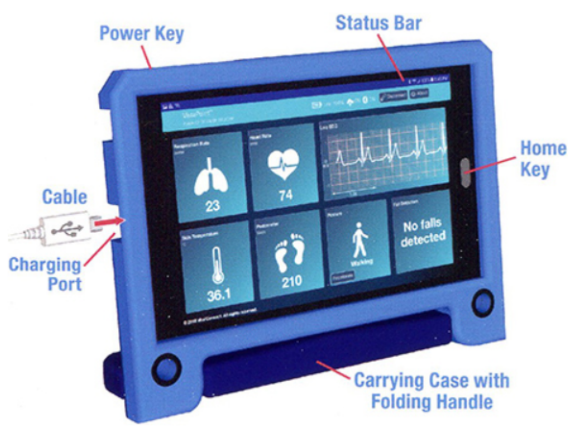


Figure 2.6: GUi on tablet of the device.

2.5 BT510

Figure 2.7: picture of BT510 Bluetooth 5 Long Range IP67 Multi-Sensor

BT510 is a sensor created by Laird it is an embadded sensor with an internal battery, is wearable and is used to transfer data via bluetooth, it has 1MB Flash memory and can last for years [11].

It has Integrated temperature sensor with proximity, accelerometer and magnetic reed switch sensors, it can be configured from its specific app, the battery consists of a replaceable CR2477 coin cell [10].

its characteristics are shown in the table 2.4 [11]

Accelerometer	3-axis MEMS – Programable motion and contact detection
Battery	CR2477 coin cell - replaceable
Certifications	FCC, IC, CE, MIC, RCM, Bluetooth SIG
Data Logging	7 days of temperature data at 15 minute read intervals
Dimensions (H x L x W)	19 mm x 80 mm x 51 mm
LED Status	1 LED, multi-state
Mounting Style	Screw, zip-tie, or industrial velcro
Open / Close Contact	Omni polar magnetoresistive sensor
Operating Temperature	-20° - 60°C (limited by coin cell type)
Software	Mobile App - Android & iOS - Configuration and/or remote sensor display, OTA Firmware Update
Software	IoT Gateway - Configuration and/or remote sensor display to cloud (PENDING)
Temperature Accuracy Range	+/-0.5°C (temperature sensor IC capability)
Temperature Range	-40° to +125°C (temperature sensor IC capability)

Table 2.4: Features of BT510.

Chapter 3

The hardware section of the developed system

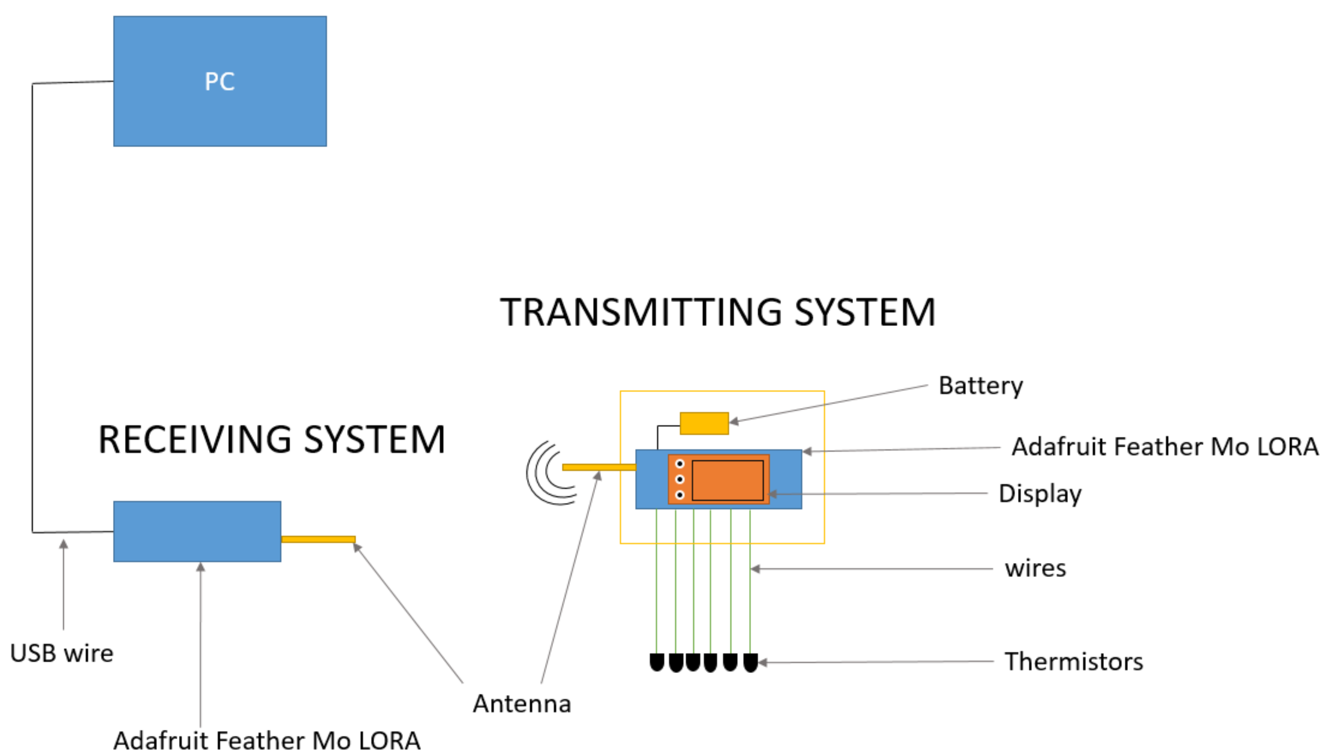


Figure 3.1: Architecture of the system

Figure 3.2: System on the subject during one of firsts characterization tests

The developed system is a wearable temperature datalogger with display and wireless connection based on microcontroller platform (Adafruit Feather M0 LORA). It also includes six temperature sensors (thermistors) and rechargeable battery. The circuit, via the LORA module, sends data to a second microcontroller of the same type which is attached via USB to a PC. Data are then collected, displayed and stored by means of a program written in Pyton language. The wearable system includes an elastic band that can be used to attach the device to the subject. The six thermistors are connected to the system using thin wires having a length of 1 m.

Characterization tests has been carried out testing the bare thermistors in a climatic chamber to verify the reliability of the sensor mathematical model provided by the manufacturer. In this way, defective sensors can be highlighted before being connected to the system.

Other tests were carried out with the full system. As an example the sensors were fixed on large metallic block that was previously heated at about 50 °C and then left to cool to the ambient temperature. In this way, the sensors experienced a slow temperature change. In this test each sensor has the same temperature because the block act as a large isothermal element. The setup is shown if Fig. 3.3. A temperature skew among all sensors of about 0.1 °C was recorder during the full temperature change.

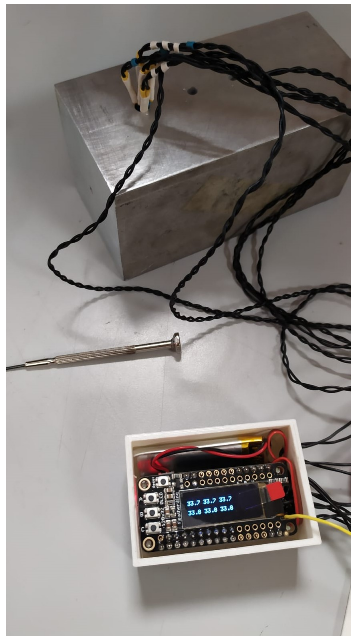


Figure 3.3: The system during the experimental assessment of the temperature skew among sensors.

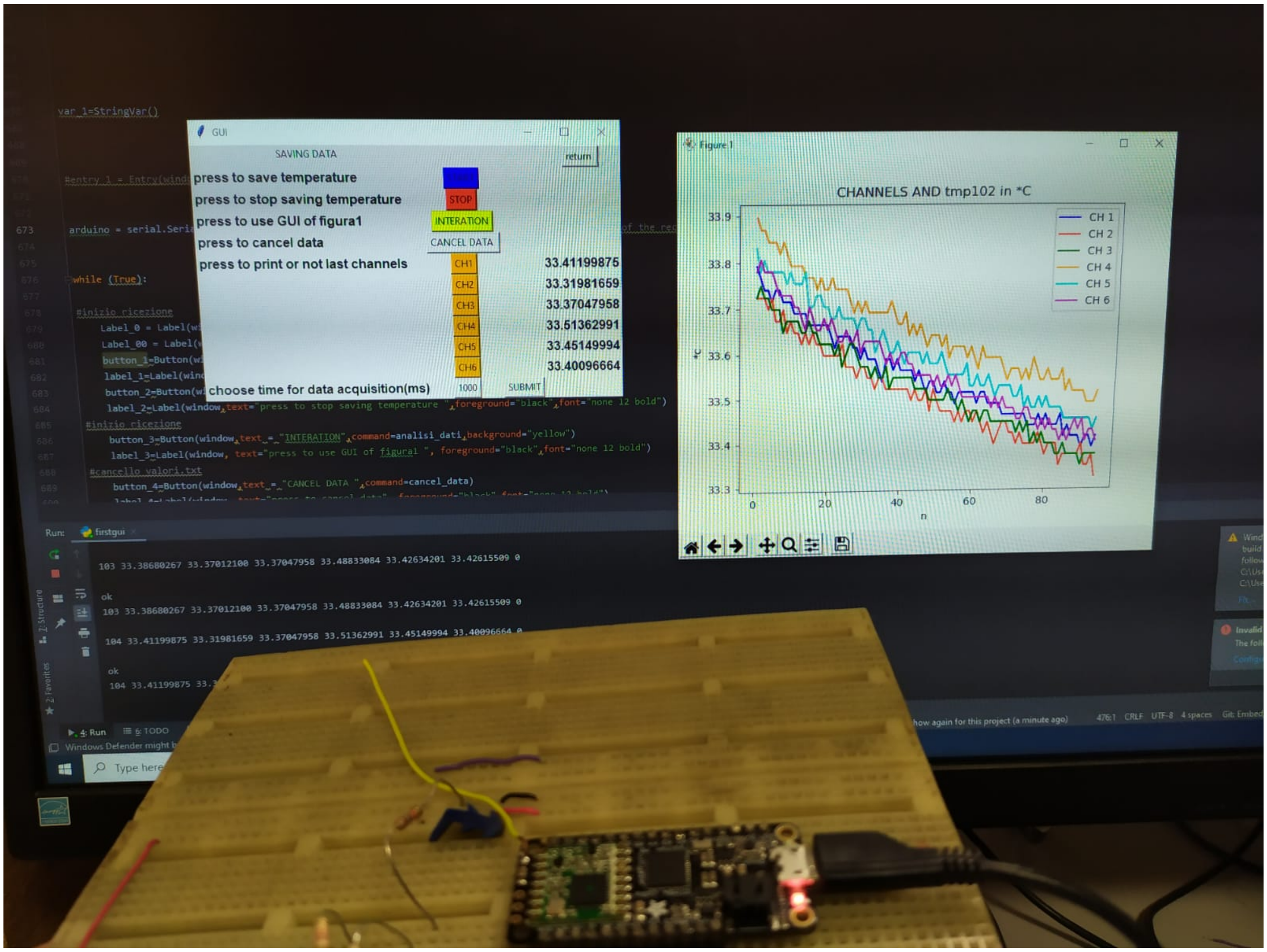


Figure 3.4: The receiver and Gui witten in Pyton; on the left side of the screen is the control menù, on the right side is the temperature evolution during the test shown in Fig. 3.3.

3.1 Adafruit Feather M0 LORA

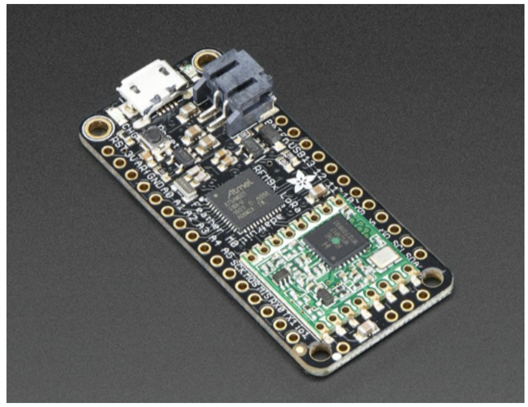


Figure 3.5: Adafruit Feather M0 LORA device

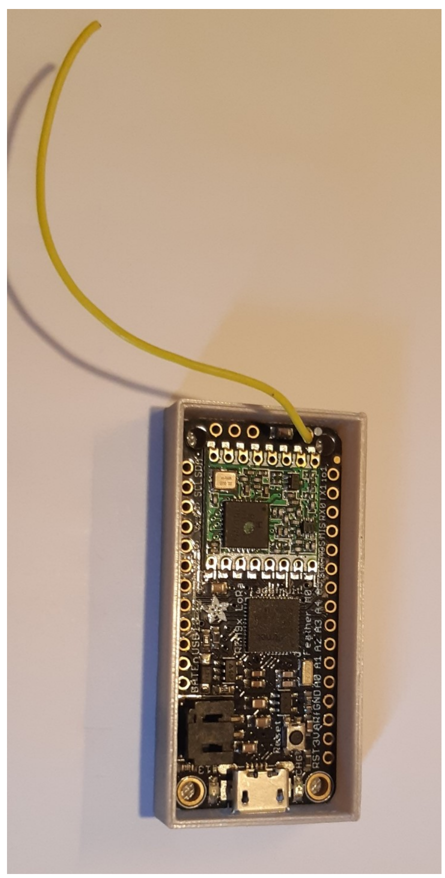


Figure 3.6: Image of the microcontroller in the system with antenna

The Feather M0 LORA is development board containing a microcontroller ATSAMD21G18 ARM Cortex M0 processor and LoRA module RFM69 Packet Radio (868 or 915 MHz) [12]. Two boards were used: one embedded in wearable part of the system and only working as a receiver.

This device supplies a voltage of 3.3 V with a peak of 500 mA of output current. As regards inputs, it has 10 analog pins: they all have been used to acquire the thermistor signals and the battery voltage, useful to highight a low battery. Another analog input pin was employed to set the Analog to Digital converter voltage reference voltage. Eventually, two pins were exclusively dedicated for the LORA device. All these features will be better explained below.

3.2 AT SAMD 21 G18

The microprocessor used is the ATSAMD21G18, in particular it has 256 K of flash memory and 32 k of ram; the details are described in table 3.1 [13].

	SAM D21G
Pins	48
General Purpose I/O-pins (GPIOs)	38
Flash	256/128/64/32KB
SRAM	32/16/8/4KB
Timer Counter (TC) instances	3
Waveform output channels per TC instance	2
Timer Counter for Control (TCC) instances	3
Waveform output channels per TCC	8/4/2
DMA channels	12
USB interface	1
Serial Communication Interface (SERCOM) instances	6
Inter-IC Sound (I^2S) interface	1
Analog-to-Digital Converter (ADC) channels	14
Analog Comparators (AC)	2
Digital-to-Analog Converter (DAC) channels	1
Real-Time Counter (RTC)	Yes
RTC alarms	1
RTC compare values	1 32-bit value or 2 16-bit values
External Interrupt lines	16
Peripheral Touch Controller (PTC) X and Y lines	12x10
Maximum CPU frequency	48MHz
Packages	QFN TQFP WLCSP

Table 3.1: Main features of the microcontroller ATSAMD21G18.

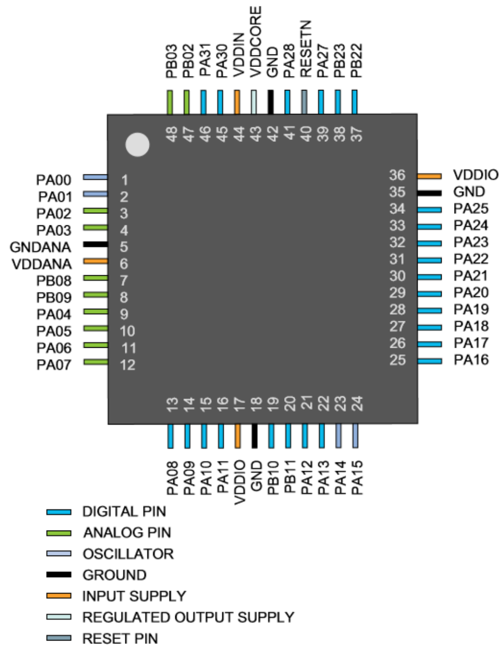


Figure 3.7: Pin description.

3.3 Pinout description

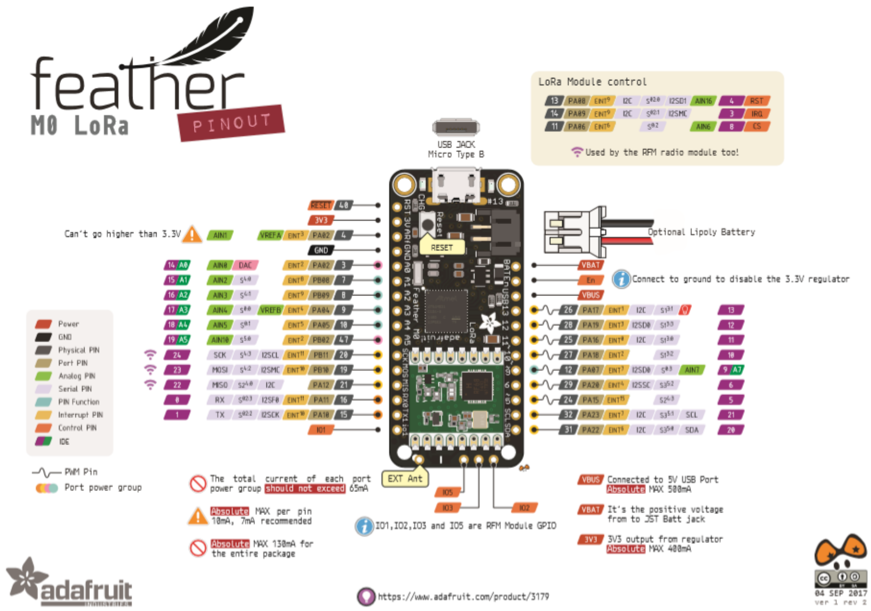


Figure 3.8: Adafruit Feather Mo LORA pinouts

As for the transmitter module the used pins are as follow:

- 3V, GND, SCL, SDA, 5, 6, 9 for the display.
- A0- A1- A2- A3- A4- A5 are attached to the conditioning circuit to take the voltage value provided by the 6 thermistors (fig 3.18), they are the analog inputs.
- Aref for the refefence tension.
- En, GND that are connected via a switch, which when activated disables the device and allows it to not consume power (except to a few micro amperes); in this way it's possible to increase its duration.

3.4 RFM96

Figure 3.9: The LORA module RFM95/96/97.

Sensor measurements are sent via radio with the module RFM 96 integrated inside the Adafruit board. The LORA system is able to send up to 300 kbytes per second [12].

The library used to program this device is <RH_RF95.h>, which can be employed to control all all the RFM9x devices.

The LORA characteristics are here summarized [14]:

- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs. V supply.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -148 dBm
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation
- Built-in bit synchronizer for clock recovery.

- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator.
- Module Size16x16mm

3.5 Battery

Figure 3.10: The employed LiPo battery, 3,7 V 500 mAh.

The battery used is a lithium-ion battery, having a voltage of 3,7 V and an electric capacity of 500 mAh.

The consumption is around 1 µA when the system is off, 30 mA in normal oprating conditions. It reaches a peak of 130 mA when the radio is on.

This battery run time is more than 12 hours. The Adafruit board embeds a battery charging system that charges the battery when the USB cable is connected to a PC or a power bank.

3.6 Oled Feather Wings display

A graphical display having a resolution of 128x32 pixels [12] has been added to the system in order to provide in real time the temperature measured by the sensors as well as other system information such as the battery voltage. An Oled display already mounted on a board compliant with the Feather board. The Oled display is produced by Adafruit and has exactly the same footprint of the microcontroller board so it can be directly soldered on it in order to save space.

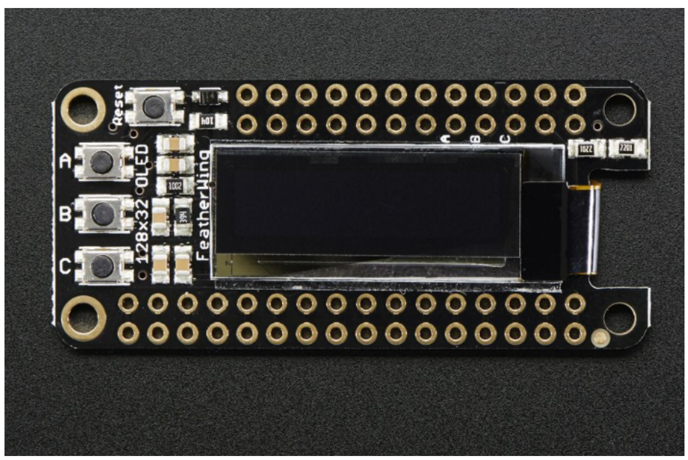
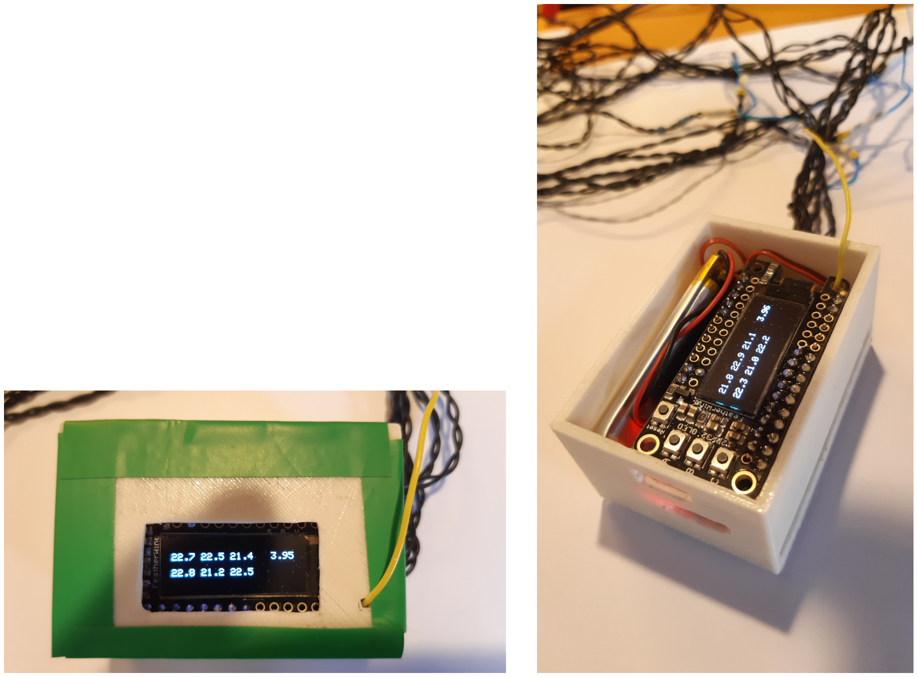


Figure 3.11: The Oled display Adafruit FeatherWings.



(a) *The wearable part of the system.* **(b)** *The system without cover.*

Figure 3.12: Images of developed system.

The microcontroller is programmed to show data according to seven modes:

- 6 temperatures , battery voltage and a marker that shows when the data has been sent and correctly received
- temperature on thermistor 1
- temperature on thermistor 2
- temperature on thermistor 3
- temperature on thermistor 4
- temperature on thermistor 5
- temperature on thermistor 6

The display also embeds three buttons and it has been programmed so that the modes can be changed by pressing button B; pressing button C instead changes the width of the moving average filter employed to reduce measurement noise (4 or 16 samples).

The microcontroller and display are connected by means of an I2C interface. This interface has a two-wire bus, composed of lines SDA and SCL [15]. In addition to these lines, a ground wire is required.

- SDA is the serial data line: It allows the subject to pass data information.
- SCL is the serial Clock line: It is used to time the passage of data.

The library used for programming the display is available online and it is <Adafruit_SSD1306.h>.

3.7 Sensors that can be used by the system

Different temperature sensors can be used to monitor skin temperature; currently the system has been designed with a conditioning circuit devised for thermistors because they have a very high sensitivity that simplify the design of the conditioning circuit. Moreover, some of-the-shelf thermistors have been produced in very small embodiment without impairing the robustness, thus allowing the usage in very demanding applications such as the one addressed in the work which concern the skin temperature during physical activity.

Nevertheless, other miniaturized sensors can be employed with the developed system provided that a suitable conditioning circuit is interposed between the sensor and the microncontroller. As an example, thermocouples are thin sensor known to be extremely versatile with accuracy that is compliant with this application. It can be shown that thermocouples can be a substitute of thermistors provided that an amplifier having a gain of about 150 is employed.

Pt100 can be also employed with minimal changes but minaiturized Pt100 sensors are not so easy to be found.

Below, the main sensor properties and characteristics are described.

3.7.1 Thermoresistance

Platinum sensors can be very accurate because platinum is not a metal subject to corrosion,it is stable over time. Furthermore, it is easily workable; the input-output relationship is showed in Eq 3.1 and 3.2 [16]

$$R = R0(1 + AT + BT^2) \tag{3.1}$$

between 0 and 850 °C.

$$R = R0(1 + AT + BT^2 + C(T - 100)T^3) \tag{3.2}$$

between -200 and 0 °C.

- R0 is the resistance of the Pt100 at room temperature, its value is 100 Ω.
- The coefficients A, B and C are standardized and are a function of the degree of aging and purity of the sensor.

Another fundamental parameter is the sensitivity [16].

$$S = \frac{dR}{dT} = R0(A + 2BT) \tag{3.3}$$

At room temperature, they have a sensitivity of about 0.4 Ω/°C. Using a linear model the resistance is:[16].

$$R = R0(1 + \alpha T) = R0 + ST \tag{3.4}$$

3.7.2 Thermocouples

Thermocouples have a different working principle being this sensor active sensor, that is, the produce a voltage in the presence of a temperature gradient. These sensors exploit the Seebeck effect [16]: the gradient in the metal wire alters the distribution of the charges and it creates a voltage; this voltage only depends on the material, it does not depend on the sensor dimensions. The effect is reversible; the Peltier effect is the complement of this[16].

The Seebeck coefficients for thermocouples made with different materials are shown in Fig. 3.13.

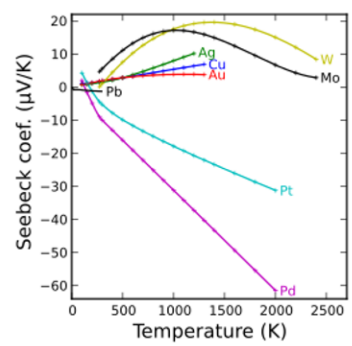


Figure 3.13: Relation between seebeck coefficient and the metal used.

The Figure shows the thermocouple sensitivity is very small. A type T thermocouple, typically employed at room temperature, has a sensitivity of about 40 μ V/°C.

3.7.3 Thermistors

Thermistors are made with non metallic material, which varies according to the sensor temperature. They can be NTC or PTC type: in the first case the sensitivity is negative (i.e. by increasing the temperature the resistance decreases); in the second case the sensitivity is positive [16].

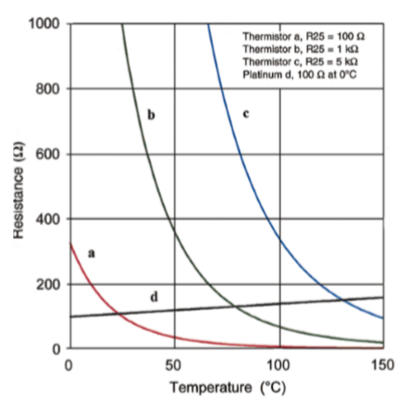


Figure 3.14: Characteristic of a typical NTC thermistor.

$$Rt = R0^{B(\frac{1}{T}-\frac{1}{T0})} \tag{3.5}$$

Temperatures are expressed in kelvin, and the typical error is 0.3 K in the range $(0 \div 50)$ K.

- R0 is the resistance at a room temperature of 298.15 K,R is obtained from Eq 3.9; the inverse characteristic is expressed in equation 3.10 [16]
- the coefficient B is called the characteristic temperature and is a value given by the manufacturer

$$T = \frac{B}{(ln(Rt) - ln(A))} \tag{3.6}$$

Where A is :

$$A = \frac{R0}{e^{\frac{B}{T0}}} \tag{3.7}$$

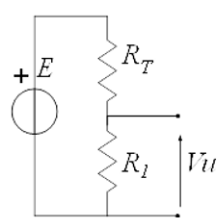


Figure 3.15: Example of conditioning circuit for thermistors.

The sensor sensitivity is not constant and depends on temperature. Moreover, the conditioning circuit has a non linear behaviour too, but its effect can compensate part of the sensor non linearity, provided that the resistances are carefully chosen.

3.8 The sensor employed in this work

Figure 3.16: Thermistors used for project

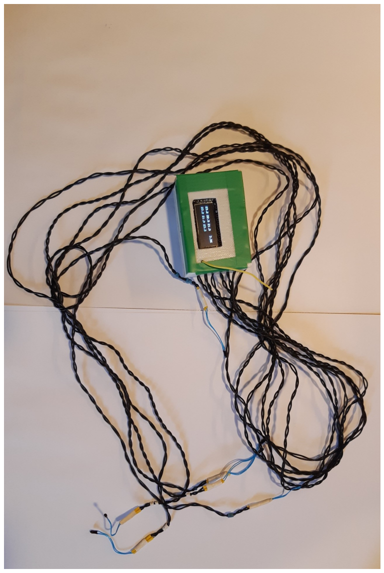
Thermistors used are thin, with dimensions of 6.5x2.4 mm and this allows to detect body temperature even in very close places; they have also been numbered using rings of heat shrink tubing, so that they can be easily recognized during applications.

They were welded to double wires of about 1 meter long, so that it is possible to make measurements further away from the point of application; a heat shrinkable tube was positioned on the upper part to cover the welded metal part.

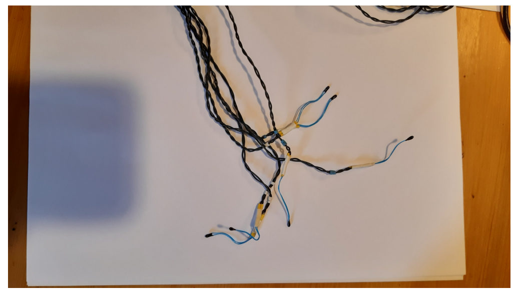
The thermistors fundamental parameters are [17]:

- R0=10 kΩ
- B=3988 K
- Tmin=-55 °C
- Tmax=150 °C
- Accuracy=1%

.



(a) *Thermistors with wires employed to measure skin temperature.*



(b) *Focus on thermistors,it's possible to distinguish them for the quantity of white and blue rings near thermistors (1, 2 or 3 white rings for thermistors 1,2 and 3; 1,2 or 3 blue rings for thermistors 4,5 and 6 .*

Figure 3.17: Images of sensors used.

3.9 conditioning circuit

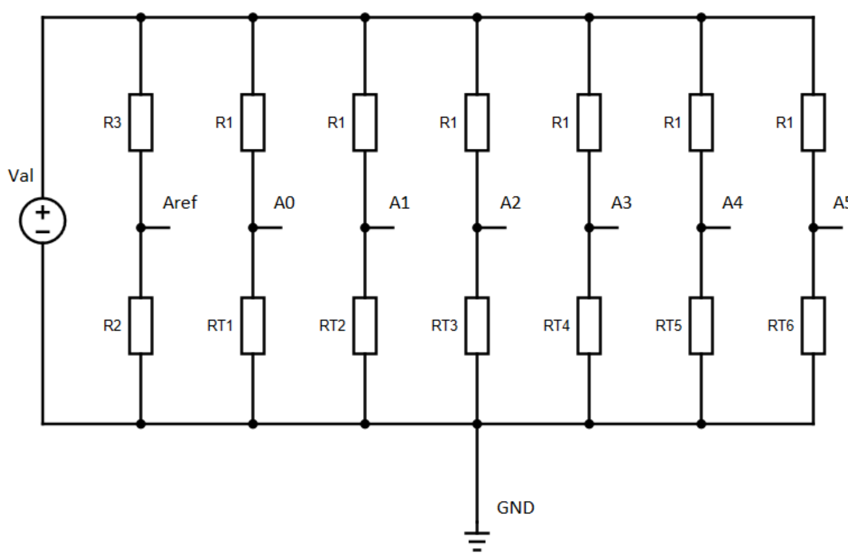


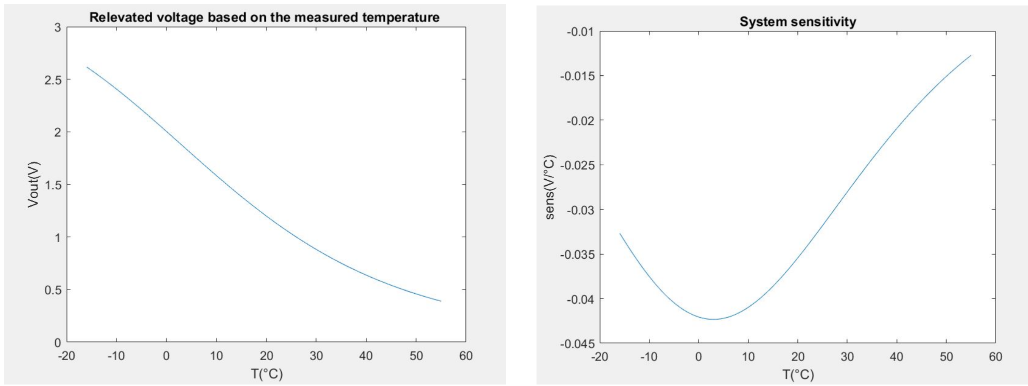
Figure 3.18: conditioning circuit for thermistors (output A0 to A5) and the voltage divider for the ADC voltage reference (output Aref).

- R1 = 22 kΩ
- R2 = 18 kΩ
- R3 = 4,7 kΩ
- Val= 3.3 V.
- RT is the resistance of thermistors

R2 and R3 were chosen in order to create a reference V of about 2.62 V, as specified by the manual. In fact, this reference V should be less than Val-0.6V [13].

To choose the R1 value, a simulation was performed on MATLAB of the conditioning circuit connected to the microcontroller, and various parameters were evaluated, such as minimum temperature, sensitivity and resistances on the market.

A compromise was found so that the temperature measured by the sensor can be below 0 °C without reduce the sensitivity at high temperatures. A value of 22 kΩ was found, which allows a minimum temperature of -16 °C to be measured.



(a) *Relevated voltage in function of external temperature .*

(b) *Sensitivity of the system .*

Figure 3.19: Graph on MATLAB of tension and sensitivity of the system between -16 and 55 °C.

Resistances were measured using an ohmmeter; their values are reported in Tab. 3.2.

R1 for th1	22175
R1 for th2	22167
R1 for th3	22143
R1 for th4	22222
R1 for th5	22067
R1 for th6	22138

Table 3.2: Actual values (in Ω) of the conditioning circuit resistances in series with the thermistors.

In order to verify thermistors expected values, tests have been conducted using a climate chamber. A Pt100 was used as a reference, were measured: the value of Pt100, then the values of the 6 thermistors and the value of Pt100 again in order to make sure that the temperature of the climate chamber maintained the same value. Values of resistances are shown in the table 3.3.

Values were initially measured at 20 °C, then at 35 °C and 50 °C. Hereafter, they were again measured at 20 °C, in order to test the reliability of the measurement procedure: results were those expected and variations were lower than 1%.

Expected values, calculated from equation 3.10, are:

- 12562.60 Ω for T = 20 °C.
- 6478.68 Ω for T = 35 °C.
- 3552.98 Ω for T = 50 °C.

	20*C	35*C	50*C	20*C
PT_100	107,8032	113,8825	119,94228	107,8704
Thermistor 1	12532	6371,6	3435,8	12441
Thermistor 2	12587	6400,2	3446,0	12498
Thermistor 3	12517	6361,0	3427,7	12430
Thermistor 4	12506	6359,5	3427,1	12406
Thermistor 5	12485	6349,0	3422,0	12393
Thermistor 6	12551	6381,3	3427,1	12457
PT_100	107,8024	113,8827	119,953	107,8637

Table 3.3: Test to verify that the thermistor values are in agreement with the expected values (resistance values are in Ω).

Chapter 4

The developed software

The designed software consists of 3 distinct parts:

- transmitter system, written in C
- receiver system, written in C
- GUI, written in Python

4.1 Transmitter

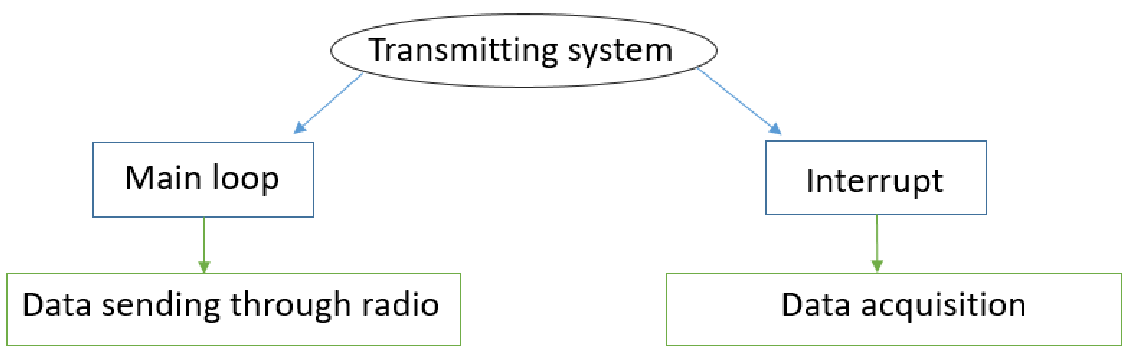


Figure 4.1: Block diagram of transmitting system

This code is composed of three main parts:

- A main loop, in which data are sent via radio to the receiver.
- An interrupt, in which data are taken in each selected sampling time. Through this process, the display is changed.
- Some subfunctions for an easier programming of the code.

The interrupt times two events: showing data on the display and the data sampling.

It conducts it cycle every 1 ms, counters were set so that the two events mentioned above take place at every set up time lapse. Data on the display are changed every 115 ms. By pressing button B it is possible to change the display mode, whereas by pressing button C the filtering mode can be modified.

The sampling of data, if not changed through the GUI, takes place every 1000 ms.

It renders the six data from the thermistors conditioning circuit and calculates the CRC: this process is conducted by sending data to a special algorithm that derives an 8-bit value.

The CRC algorithm basically provides for a polynomial division between data provided, which are converted into a polynomial, and a polynomial called the generator .

The generator polynomial used was `x^8+x^4+x^3`, it is obtained from the value 10001100, the value x replaces the units raised to their corresponding bit added together. In the same way, the polynomial is obtained by dividing data supplied (with respect to the temperature) converted into binary.

The value obtained is used to verify that data received by the receiver are the same as they are sent, since the CRC of data received is obtained with the same algorithm. If data received were different from those sent, the CRC would also be different; it can be said, with an error of 0,4%, that data was not received correctly.

The number of data, the CRC and the 6 temperature values are saved in a matrix which acts as a buffer. The position in which information will be saved, is defined by a pointer: it scrolls along the matrix and saves each new acquisition in the next free line.

In the main loop, data are sent, then the confirmation of correct reception is awaited.

Initially, 10 samples are acquired without the radio sending: in this way the buffer is filled at a minimum before starting to send.

Hereafter, acquisitions are sent one at a time, until they will be correctly received. The acquisition to be sent is decided by a second pointer, asynchronous with respect to the first.

The sending time is lower than the acquisition time, for this reason the second pointer should not exceed the first one, otherwise it would send empty lines. Once reached the end of the matrix, pointers start again from row zero.

sub-functions:

- MOD INTERRUPT: it changes the sample time.
- CRC_SEND: data preparation for the CRC_8 algorithm.
- CRC_8: executes the CRC algorithm.
- INIT_RADIO: it initializes the radio and sets the frequency to 915 MHz.
- INIT_INTERRUPT: initializes an interrupt every 1 ms.
- DISPLAY_FUNCTION: it shows the display modality and changes the modality of filtering, if is request.

4.2 Flow chart transmitter

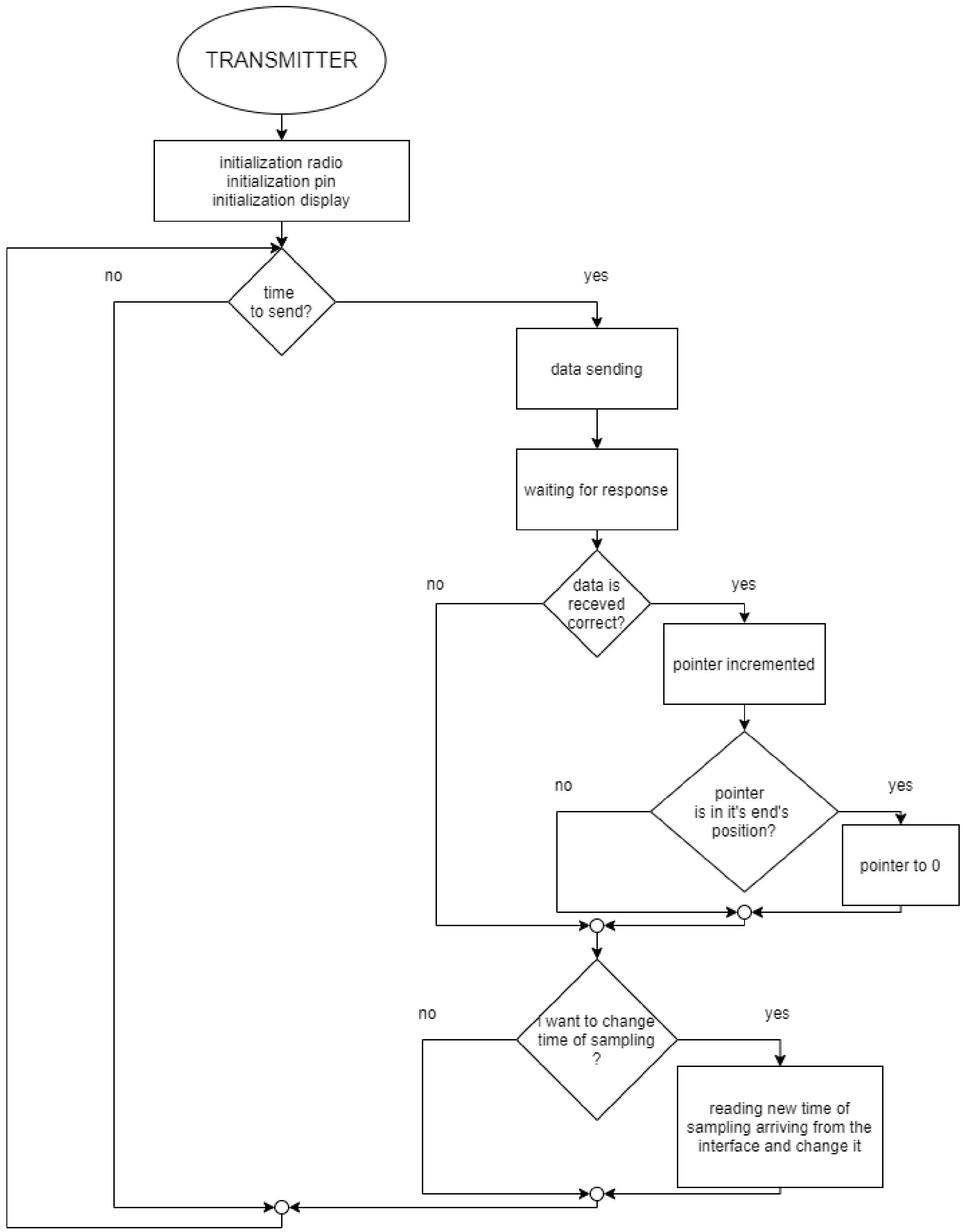


Figure 4.2: flow chart of main_loop's transmitter

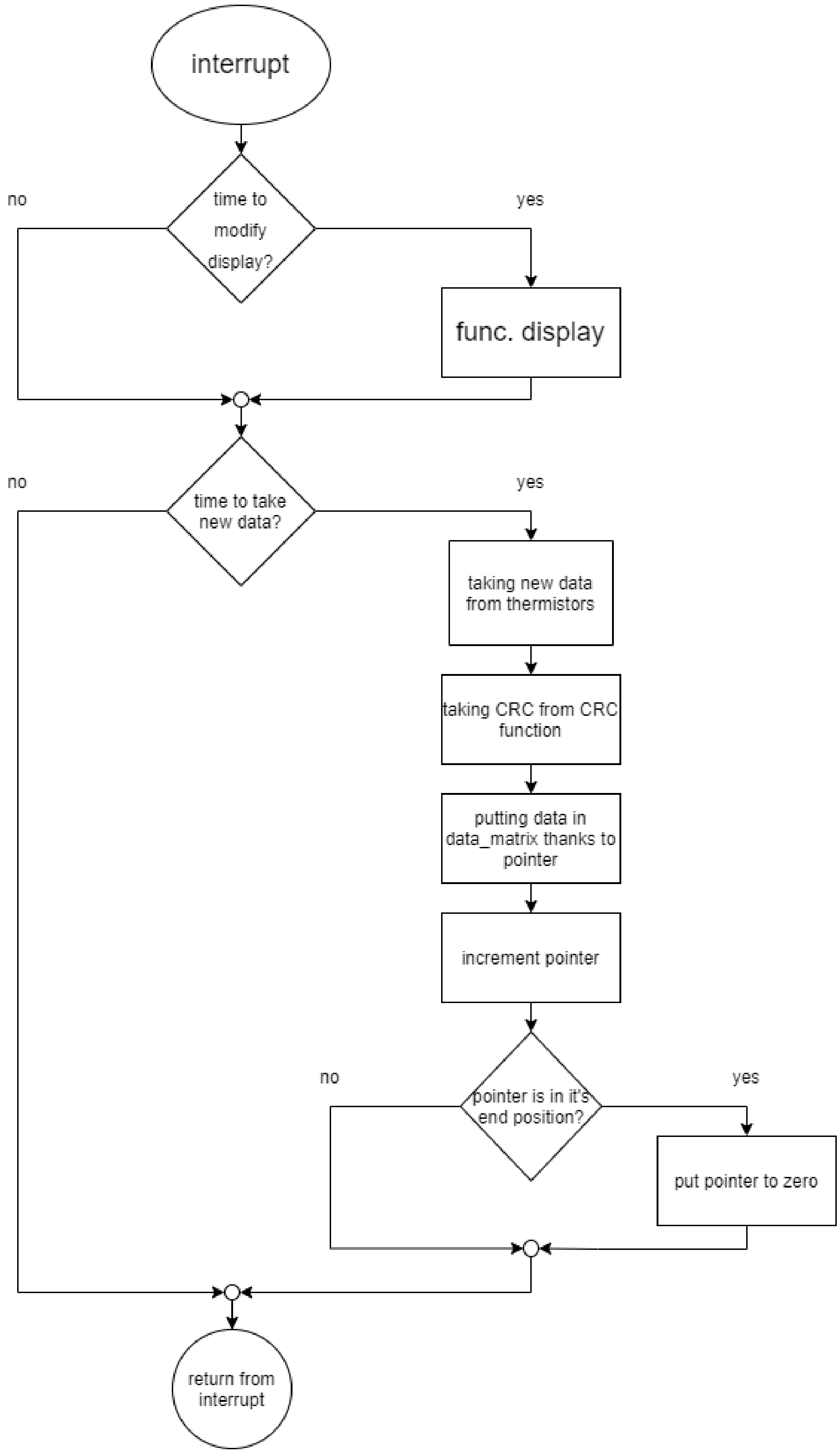


Figure 4.3: flow chart of interrupt's transmitter

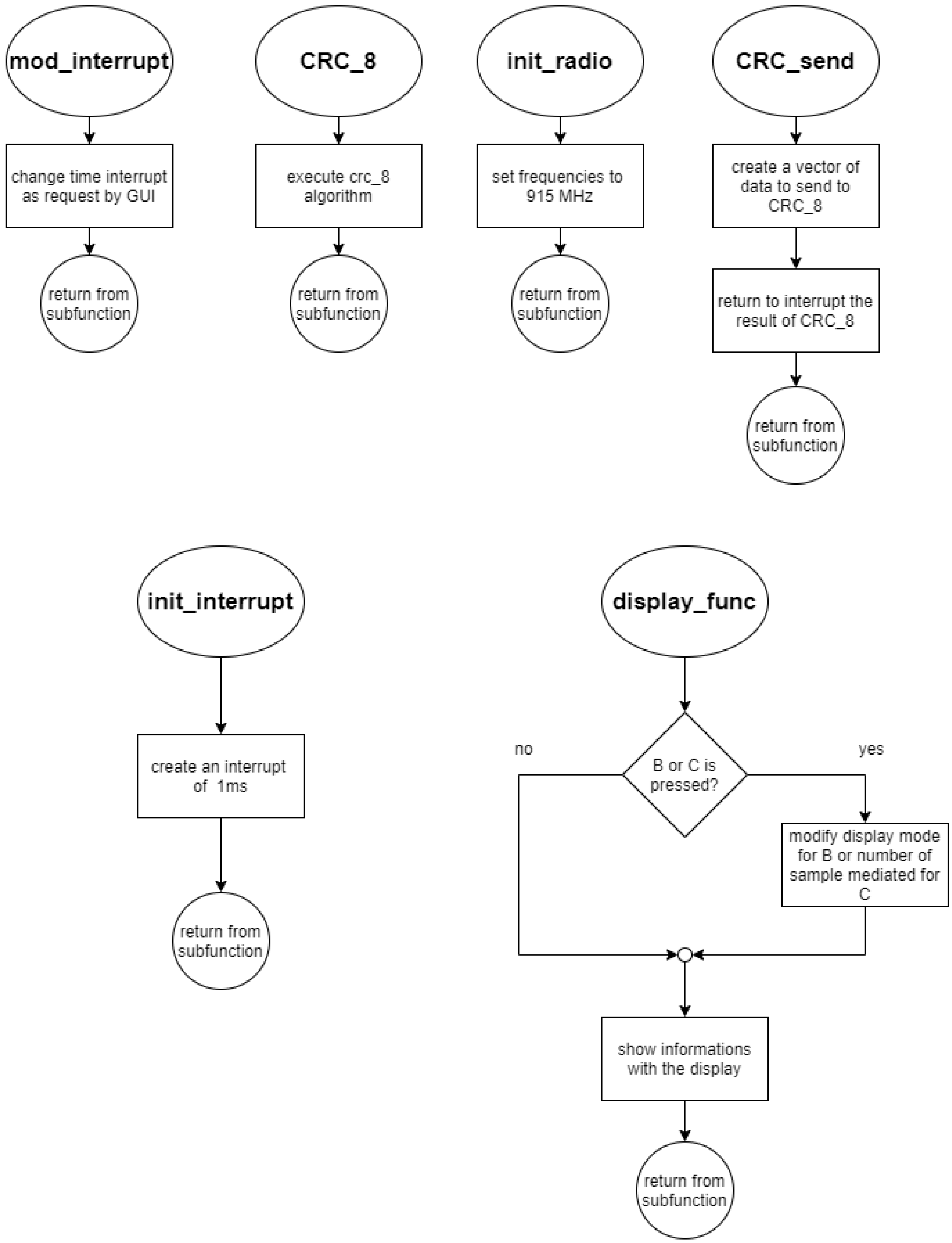


Figure 4.4: flow chart of subfunction's transmitter

4.3 Receiver

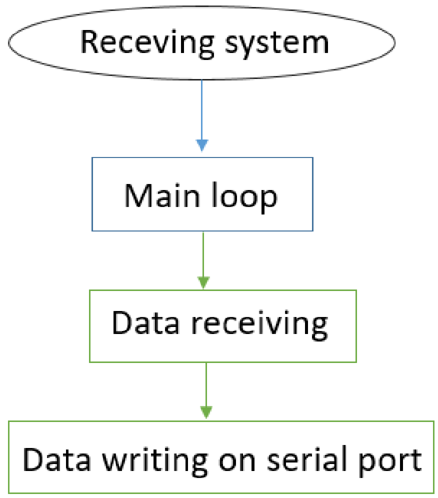


Figure 4.5: Block diagram of receving system

The code for the receiver consists only of a main loop and sub-functions: The main loop is waiting for data to be received via radio. Once data arrived, it breaks the string into the data number, CRC and temperature values. To verify that reception is correct, two checks are performed:

- 1)The first test is conducted to verify that data arrived correctly, the same algorithm of the CRC, described above, is performed. If the CRC arrived and the one obtained is different, it means that data are corrupted; therefore, the reception is not successful.
- 2)The second test verifies that the number of the received data is the expected one. Therefore, it is checked that the number of the data is the next compared to the one previously received.

It may also happen that the transmitter is reset: in this case, in order not to enter the loop after 3 errors on the data number, it is checked that the number of the expected data is not zero; in this way it is possible to resume the correct reception without resetting the receiver. Hereafter, if there are no reception errors, the data number, the battery voltage and temperature values are written in the serial port, so they can be read by the graphical interface. Lastly, the result of the correct reception is sent via radio, together with the new sampling time read in the serial port, if you want to change it from the graphical interface

Sub-functions:

- 1)CRC_SEND: data preparation for the CRC_8 algorithm.
- 2)CRC_8: it executes the CRC algorithm.
- 3)INIT_RADIO: it initializes the radio and sets the frequency to 915 MHz.

4.4 Flow chart receiver

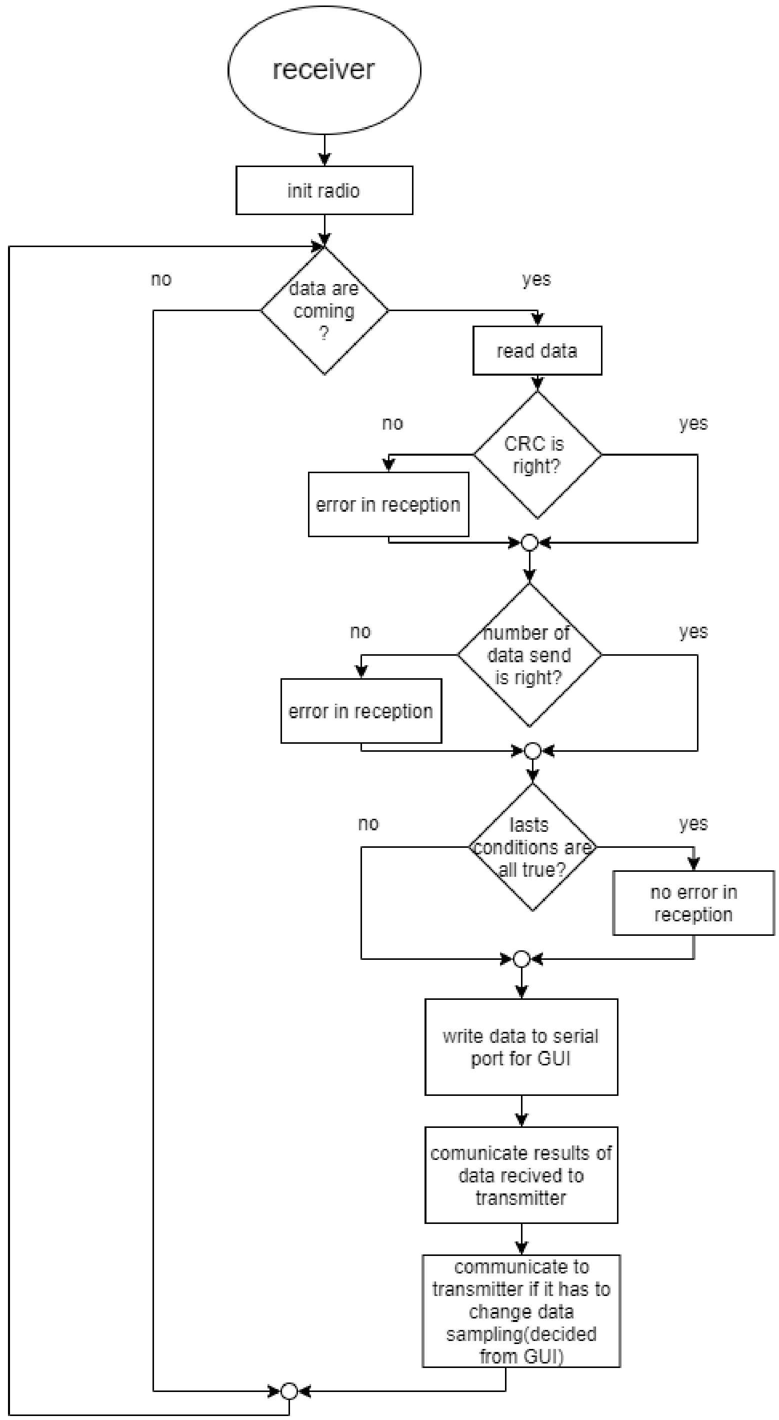


Figure 4.6: flow chart of main_loop's receiver

4.5 GUI

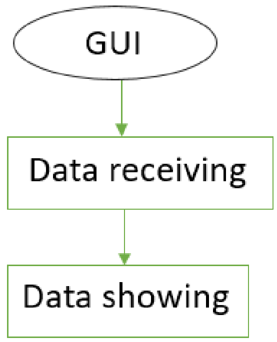


Figure 4.7: Block diagram of GUI

The graphical interface has a menu consisting of two buttons: NEW DATA and ADD / LOAD DATA.

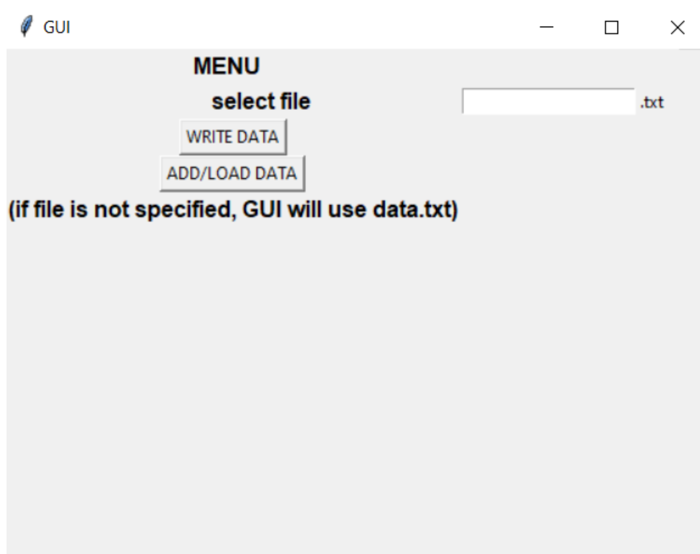


Figure 4.8: General menù of GUI

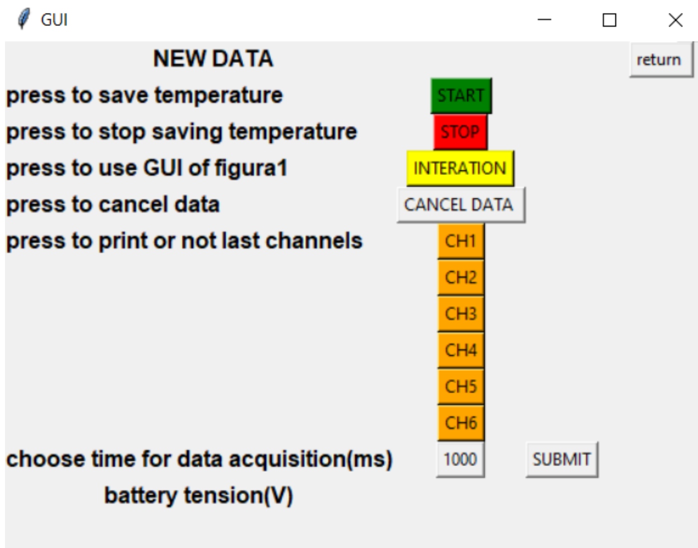


Figure 4.9: Menù new data of GUi

Pressing the first button it is possible to collect new data, create a new text file or overwrite a previous one. Using the function ADD / LOAD DATA it is possible to add data to a pre-existing file, or simply to load data in order to see its progress.

The buttons are:

- RETURN: in case you want to return to the main menu.
- START: press to start saving the data in the txt file and display them, once pressed the button turns blue, if you exit this mode it returns green.
- STOP: pressing this button, data are not saved but lost. This mode is activated in case there are problems of noise; if pressed the button turns blue, if deactivated it remains red.
- INTERATION: it allows the subject to interact with the figure. The button turns blue if this mode is entered; it returns yellow if the function is deactivated. When the button is active, no new data are displayed; if START is also pressed data are saved in txt file.
- CANCEL DATA: the data in that text file is deleted.
- CH: it shows values related to different channels and it will remain orange. If the function is deactivated the button turns blue and values are no longer displayed (they are still saved in the txt file).

- 1000: it allows the subject to choose the new time for the data sample.
- SUBMIT: it allows the subject to send the new chosen time of data sample to the receiver via serial port, which sends it to the transmitter via radio.

4.6 Flow chart GUI

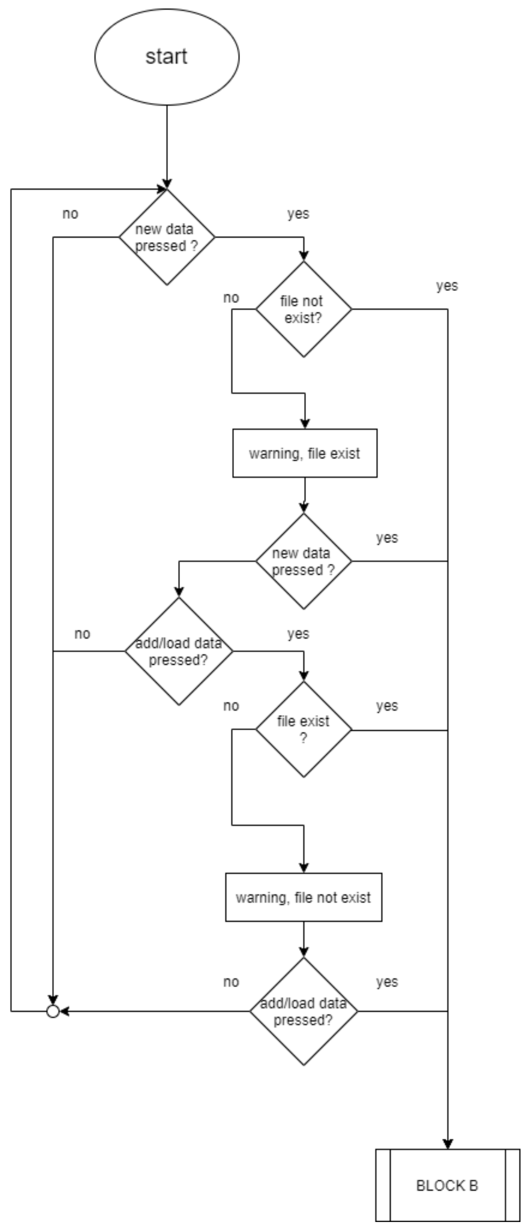


Figure 4.10: flow chart GUI BLOCK A

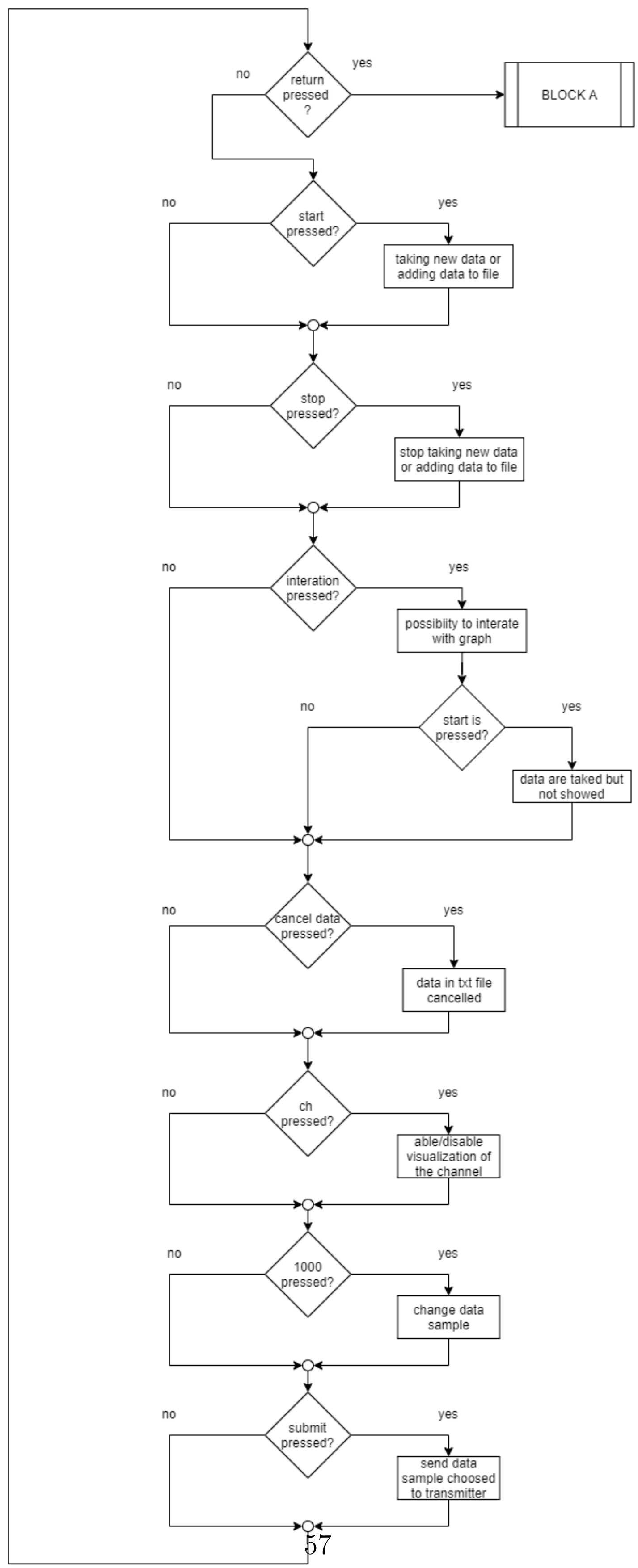


Figure 4.11: flow chart GUI BLOCK B

Chapter 5

Experimental results

5.1 Assessment of the glove thermal insulation

In these preliminary tests the thermal insulation of gloves was experimental assessed by monitoring the temperature on the finger surface.

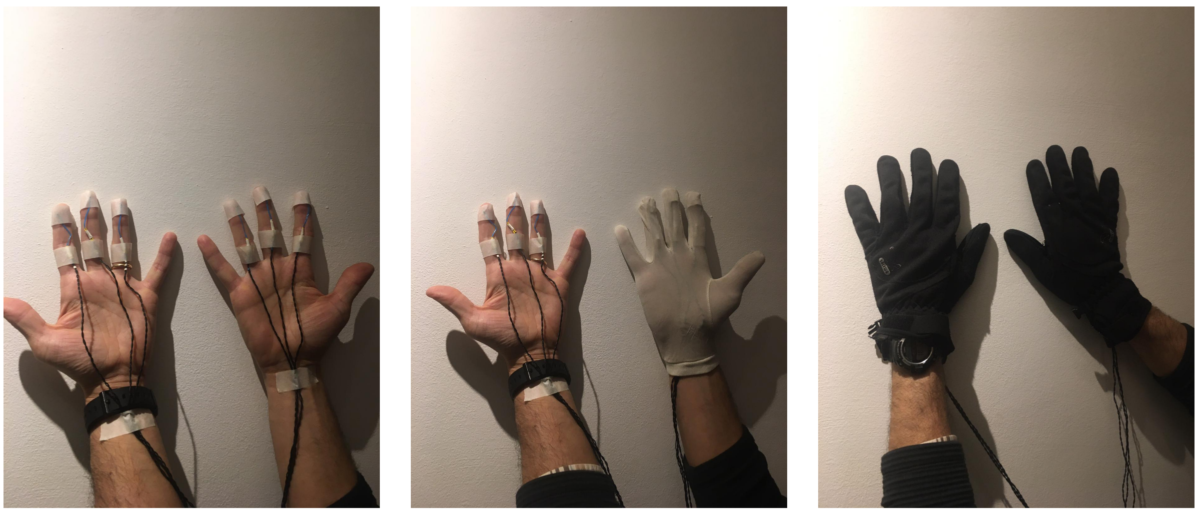
The monitoring system developed during this thesis work is equipped with miniaturized thermistors having a diameter of 2 mm. They are thus well suited to be employed inside gloves where the space is limited and the comfort of the athlete is of primary concern. The sensors were fixed with adhesive tape to the three middle fingers at a distance of about 1 cm from the finger tip. Three sensors were employed to monitor the right hand and the remaining three sensors were fixed on the left hand. Fig. 5.1a shows the sensors fixed on the fingers.

Aim of this test was the assessment of the further thermal insulation provided by a thin under glove made of silk. Fig. 5.1b shows the white under glove on the right hand and eventually Fig. 5.1.c shows both hands wearing a pair of gloves made by Salewa (Polarlite model).

For comparison purposes two test were carried out: in the first test the under glove was on the right hand while in the second test to under glove was on the left hand. In this way it is possible to highlight the temperature differences that arise hand perfusion. Both tests were performed outdoor by the same volunteer performed outdoor with an ambient temperature of about 9 °C.

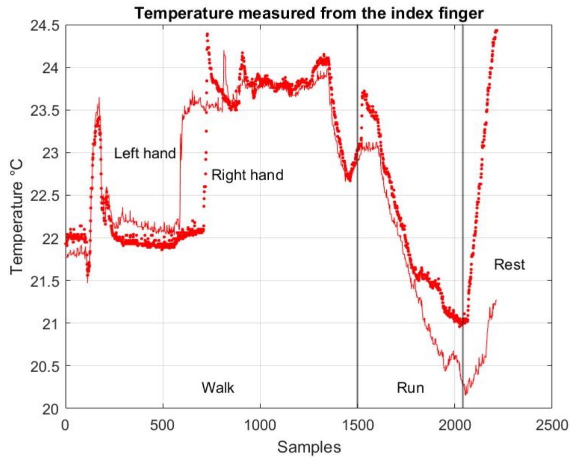
During the first test the under glove was on the right hand and the subject performed a moderate physical activity comprising walking and running phases according to the following protocol:

- 25 minutes of walking
- 10 minutes of running
- 6 minutes of resting

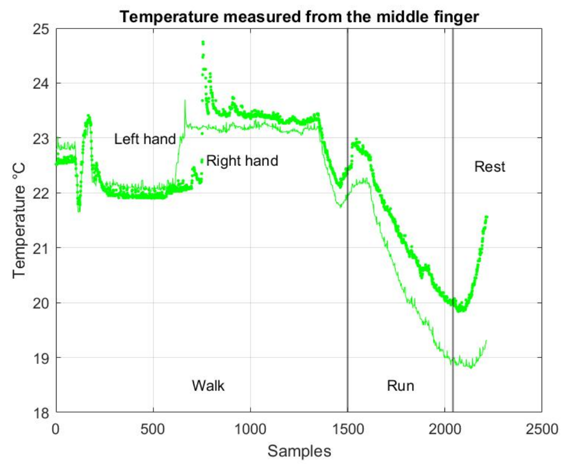


(a) *thermistors positioning.* **(b)** *putting underglove in right hand.* **(c)** *putting gloves in hands.*

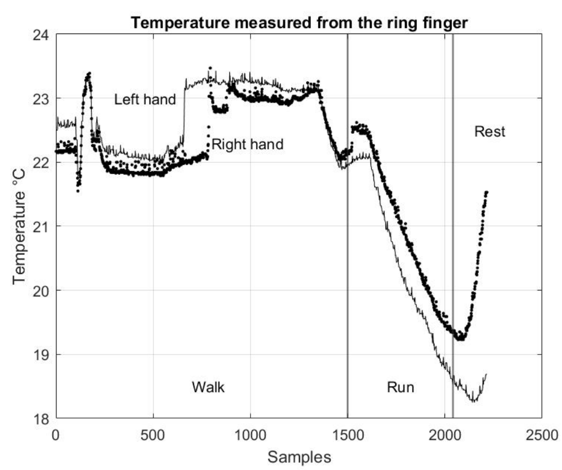
Figure 5.1: Cover used for hands.



(a) *Index finger.*



(b) *Middle finger.*



(c) *Ring finger.*

Figure 5.2: Data saved by system.

The system recorded the sensor measurements that were subsequently analyzed using MATLAB. Fig. 5.2 compares the temperatures of the same finger of the right and left hand. At the beginning of the walking phase the temperature of fingers remains around 22 °C. This value is low due to the fact that the subject was in a cold environment before the test. During walking, the temperature increased of about 1.5 °C and afterwards, during the running phase, the temperature dropped around 19 °C. This reduction can be explained because the peripheral vasoconstriction occurred and the cold ambient air entered inside the gloves because of the speed. Actually the gloves are made with a water repellent but breathable fabric.

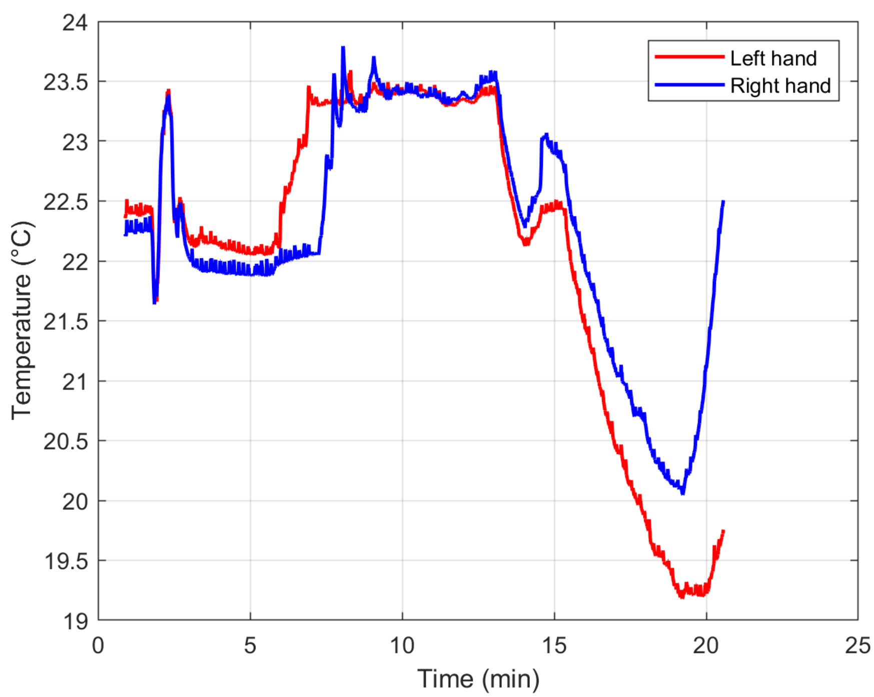


Figure 5.3: Average finger temperature when the right hand wears the under glove.

Figure 5.3 show the average finger temperature. During the running phase the right hand is warmer than the left hand thus proving the larger insulation capability of the silk underglove.

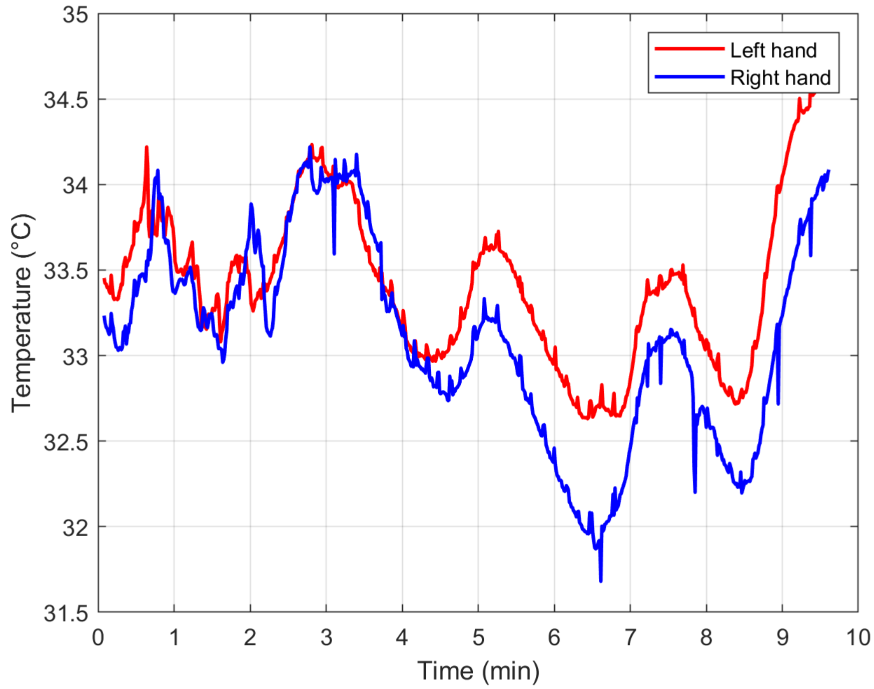


Figure 5.4: Average finger temperature when the left hand wears the under glove.

For comparison purposes, a similar test was carried when the under glove was on the left hand. In this test the volunteer followed a free protocol composed of short walking a running phases. The results are shown in Fig. 5.4. The finger temperature at the beginning of the test was higher that in the previous test because the volunteer was in a warm environment before the test. Again, it is possible to see that in the running phases the temperature decreases and increases during the walking phase. Again, the hand wearing the under-glove has a higher temperature.

In fact we can see in both tests that the effect of the under glove is evident; the hand that wears the glove is 1.5 °C warmer during activity, the system is able to detect the temperature of the fingers during sports activities even in different clothing conditions and outside temperature.

In the second test, however, we notice a bigger noise, probably the sensors had a slipped and it was not fixed as in the first test, to fix this in subsequent tests it is possible to increase the number of samples filtered by the display by pressing the button C if the number of filtered samples is 4.

Chapter 6

Conclusions and future developments

This thesis work was aimed at creating a system for monitoring human body temperature during sports or other physical activities. The system core is a wearable device that embeds six temperature sensors and a radio module which is employed to send measurements to a remote receiver interfaced to a Personal Computer. The wearable device is equipped with an internal rechargeable battery that provides autonomy of more than 12 hours. It powers an M0 LORA Adafruit Feather microcontroller, 6 thermistors and a display.

The display lets the athlete monitor temperatures, the battery status and the reliability of the wireless link. In case of the link is not working, the wearable device logs the measurements in a local memory.

The thermistors have their own conditioning circuit. They are connected to the wearable system via cables approximately 1 m long, so that they can be positioned in different parts of the subject's body.

The system acquires data and sends the measurements and other parameters to the receiver using a LORA module. The receiver is connected to a PC where a Python program display data on graphical interface. Data are also stored for a subsequent processing.

The sensors and the full system were tested and characterized. In isothermal conditions, a temperature skew of about 0.1 °C was found among the sensors. The noise of raw measurements is of about 0.3 °C, a value that is reduced to about 0.1 °C using a proper numerical filtering. The filer bandwidth can be also set by the user.

The system was employed to assess the thermal isolation of gloves. The employed thermistors are very small so they are well suited to be introduced inside the glove fingers without any discomfort for the athlete. Temperature changes due to the different training phases, as well as the presence of an underglove, were clearly detected.

A future development of the proposed system is the increase in measurement points. This can be obtained using several transmitters since they can support a wireless sensor network protocol. Moreover, the system can manage different temperature sensors, such as the thermocouples, provided that a suitable amplifier is integrated inside the wearable system. Digital sensors for different quantities can be also easily integrated. To this aim, would be also useful to take advantage of electrical connectors, so that the sensors can be removed when they are not employed.

Part I

Appendixes

Transmitter code

```
// Feather9x_TX
// -*- mode: C++ -*-

#include <RH_RF95.h>
// for feather m0
#define RFM95_CS 8
#define RFM95_RST 4
#define RFM95_INT 3
// Change to 434.0 or other frequency, must match RX's freq!
#define RF95_FREQ 915.0
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h> //for the display
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 32, &Wire);   //to comunicate
    with display using SDA and SCL

#define BUTTON_B  6
#define BUTTON_C  5

char tdataBuffer[100];//array to send
int i;

#define righe 600        //max=863
#define colonne 8
float data_matrix[righe][colonne];    //data are stored in that matrix
char buf_string[100];    // for  analize the array coming

int count=0;    //  number of data send
byte CRC;       //CRC for see if the data sended arrived correctly
```

```
int t_signal[12];   // for  analize the array coming
char *token;
char *ptr;

int LSB1 =0;      // LSB of data readed by thermistors(decided by internal ref)
int LSB2 =0;
int LSB3 =0;
int LSB4 =0;
int LSB5 =0;
int LSB6 =0;
float Rs1=0;     // resistence of thermistors
float Rs2=0;
float Rs3=0;
float Rs4=0;
float Rs5=0;
float Rs6=0;
float T1=0;      //temperature view from the thermistor
float T2=0;
float T3=0;
float T4=0;
float T5=0;
float T6=0;
                 //with CRC i compare also 4 decimal
float T7=0;
float T8=0;
float T9=0;
float T10=0;
float T11=0;
float T12=0;
float T13=0;
float T14=0;
float T15=0;
float T16=0;
float T17=0;
float T18=0;
float rounded=0.5;

float Vbattery=0;
char sending1=' ';
byte sending=0;
```

```
int x=0;          //this variable is used for change the data sampling


  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

byte crc = 0x00;        //for CRC8
byte extract = 0;
byte sum = 0;
byte tempI=0;

int filt=0;            //count for filtering
int N_filt=16;         //N samples that are mediated with 4 is 16 and viceversa, because in the first loop it changes


int Ctrad=0;         //pointer that indicates a free cell in data matrix in witch i can put new data
int Csend=0;         //pointer that indicates the cell to send

int countTC1=0;             //count for read new data
int interrupt1=1000;        //millisecond to read a new data
int countTC2=0;             //count for modificate display
int interrupt2=115;         //millisecond to modificate display
int  display_count=7;       //N of display to send

byte Cinit=0;  //the first time enter in the loop i take 10 sample, it will be 1 after this

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);//inizializza la radio

//variables for thermistors

int bits=12;
float A=0.01552230636;   //10000/(e^(3988/298.15))
int B= 3988;
float RRef=18208;
float R11=4705;
float RA0=22175;
float RA1=22167;
float RA2=22143;
float RA3=22222;
```

```
float RA4=22067;
float RA5=22138;
float K=0;

void setup()
{
    INIT_RADIO();
    init_interrupt();
    K=(RRef/(RRef+R11));
    Serial.begin(115200);
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Address 0x3C for 128x32
    digitalWrite(8, LOW);
    display.display();                                        // actually display all of the above
    delay(100);

    pinMode(A1,INPUT);
    pinMode(A2,INPUT);
    pinMode(A3,INPUT);
    pinMode(A7,INPUT);
    display.clearDisplay();                                   //clear the display
    display.display();
    Serial.println("N_filt_setup");
    Serial.println(N_filt);

    pinMode(BUTTON_B, INPUT_PULLUP);
    pinMode(BUTTON_C, INPUT_PULLUP);

    display.setTextSize(2);                                   //set dimention of textsize
    display.setTextColor(SSD1306_WHITE);                      //set colors of characters of display
    display.setCursor(0,0);
    display.setFont();
    //display.setFont(&FreeSerif9pt7b);
    display.print("  START  ");
    display.setFont();
    display.setCursor(0,0);                                   //initialize cursor
    display.display();
    display_count=7;     //in start button B and C are pressed, i want the starting conditions
```

```
    if(N_filt==4)
       {
        N_filt=16;
       }
    else
       {
        N_filt=4;
       }
}

void loop()
   {
    digitalWrite(13, HIGH);
                  //i take 10 sample without transmitting(only the first time, after this Cinit will be 1)

    while((Ctrad<10)&&(Cinit==0))
    {
    Serial.print(" ");
    }
    Cinit=1;

    if((Csend!=Ctrad))      //if Ctrad is highter than Csend i have to send data, Csend can't be highter than Ctrad
       {

          //data sended: count, CRC, T1,T2,T3,T4,T5,T6
        /*Serial.println("dato inviato");
        Serial.println(data_matrix[Csend][0]);
        Serial.println("Csend");
        Serial.println(Csend);*/

        sprintf(tdataBuffer,"%f,%f,%f,%f,%f,%f,%f,%f,%f" ,data_matrix[Csend][0],data_matrix[Csend][7],data_matrix[Csend][1],data_matrix[Csend][2],data_matrix[Csend][3],data_matrix[Csend][4],data_matrix[Csend][5],data_matrix[Csend][6],Vbattery);
        Serial.println("sending");
        Serial.println(tdataBuffer);

        rf95.send((uint8_t *)tdataBuffer, 100);//sending
        rf95.waitPacketSent();

      // Now wait for a reply
```

```
        buf[RH_RF95_MAX_MESSAGE_LEN]; //reception message from reciver
        len = sizeof(buf);

      //Serial.println("Waiting for reply...");

        if (rf95.waitAvailableTimeout(5000))
          {

        // Should be a reply message for us now
            if (rf95.recv(buf, &len))
              {

                Serial.print("Got reply: "); //here i have the result
                sprintf(buf_string,"%s",buf);
                Serial.println((char*)buf);
                token=strtok(buf_string,",");

                i=0;
                while(token!=NULL)
                  {
                   t_signal[i]= strtod   (token,&ptr);
                   i++;
                   token=strtok(NULL,",");
                  }

              }
            else
              {
                Serial.println("Receive failed");
              }
          }
      else                     //if i arrive here there is too delay, i write
         result=1(error) and t_signal [2] remain-49(is for not modify data
         sampling)
          {
            Serial.println("No reply, is there a listener around?");

            Serial.println("ritardo");
            t_signal[2]=-49;
            t_signal[0]=1;
          }

        Serial.println("interrupt1");
```

```
            Serial.println(interrupt1);

            if(t_signal[2]>-1)
                {
                 MOD_INTERRUPT();          //if line arrives here i want to modify data sampling
                }

            if(t_signal[0]==0)
                {
                 Csend++;
                 Serial.println("C_send");
                 Serial.println(Csend);
                 sending=1;
                }
            if(Csend==righe)
                    {
                    Csend=0;
                    }

          }

    }


//CRC-8 - CRC-8 formula-based algorithm of Dallas/Maxim
//code published under licence GNU GPL 3.0
byte CRC8(const byte *data, byte len)
{
     crc = 0x00;
     while (len--)
          {
            extract = *data++;
            for (tempI = 8; tempI; tempI--)
                {
                 sum = (crc ^ extract) & 0x01;
                 crc >>= 1;
                 if (sum)
                     {
                      crc ^= 0x8C;
                     }
                 extract >>= 1;
                }
```

```
        }
    return crc;
}


void INIT_RADIO(void)
    {

       //initialization of radio


    digitalWrite(8, HIGH);
    delay(1);
    digitalWrite(8, LOW);
                                                              //pin 8 low for
        activate radio, high for inactivate it

    pinMode(RFM95_RST, OUTPUT);
    digitalWrite(RFM95_RST, HIGH);


  //Serial.println("Feather LoRa TX Test!");

    digitalWrite(RFM95_RST, LOW);      // manual reset
    digitalWrite(RFM95_RST, HIGH);
    while (!rf95.init())
   {
    //Serial.println("LoRa radio init failed");
        while (1);
   }
  //Serial.println("LoRa radio init OK!");

  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
    if (!rf95.setFrequency(RF95_FREQ))
   {
    //Serial.println("setFrequency failed");
        while (1);
   }
 // Serial.print("Set Freq to: "); Serial.println(RF95_FREQ);

  // Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128
       chips/symbol, CRC on

```

```
  // The default transmitter power is 13dBm, using PA_BOOST.
  // If you are using RFM95/96/97/98 modules which uses the PA_BOOST transmitter pin, then
  // you can set transmitter powers from 5 to 23 dBm:
    rf95.setTxPower(23, false);

   }


void TC4_Handler()                                    // Interrupt Service Routine (ISR) for timer TC4
{

  // Check for overflow (OVF) interrupt
    if (TC4->COUNT8.INTFLAG.bit.OVF && TC4->COUNT8.INTENSET.bit.OVF)
      {
        REG_TC4_INTFLAG = TC_INTFLAG_OVF;          // Clear the OVF interrupt flag
      }

  // Check for match counter 0 (MC0) interrupt
    if (TC4->COUNT8.INTFLAG.bit.MC0 && TC4->COUNT8.INTENSET.bit.MC0)
     {
    // Put your counter compare 0 (CC0) code here:                                   //CC0
    // ...
         if(countTC2==interrupt2)
        {

         countTC2=0;                          //here i want to change display
         DISPLAY_FUNC();
        }
         if(countTC1==interrupt1)
            {
             countTC1=0;                          //here i want to take new data

```

```
        analogReadResolution(bits);
        analogReference(AR_EXTERNAL);                    //taking external reference
        LSB1 =0;                                         //initialize LSB
        LSB2 =0;
        LSB3 =0;
        LSB4 =0;
        LSB5 =0;
        LSB6 =0;

        for(  filt=1; filt <=N_filt; filt ++)     //filtering
          {
           LSB1 =analogRead(A0)+LSB1;
           LSB2 =analogRead(A1)+LSB2;
           LSB3 =analogRead(A2)+LSB3;
           LSB4 =analogRead(A3)+LSB4;
           LSB5 =analogRead(A4)+LSB5;
           LSB6 =analogRead(A5)+LSB6;
          }
        LSB1=LSB1/N_filt;
        LSB2=LSB2/N_filt;
        LSB3=LSB3/N_filt;
        LSB4=LSB4/N_filt;
        LSB5=LSB5/N_filt;
        LSB6=LSB6/N_filt;


        Serial.println(" ");

        Rs1=RA0/((pow(2,bits)/(LSB1*K))-1);          //taking resistence of thermistors
        T1=B/(log(Rs1)-log(A))-273.15;              //taking temperature from resistance

```

```
        Rs2=RA1/((pow(2,bits)/(LSB2*K))-1);
        T2=B/(log(Rs2)-log(A))-273.15;

        Rs3=RA2/((pow(2,bits)/(LSB3*K))-1);
        T3=B/(log(Rs3)-log(A))-273.15;

        Rs4=RA3/((pow(2,bits)/(LSB4*K))-1);
        T4=B/(log(Rs4)-log(A))-273.15;

        Rs5=RA4/((pow(2,bits)/(LSB5*K))-1);
        T5=B/(log(Rs5)-log(A))-273.15;

        Rs6=RA5/((pow(2,bits)/(LSB6*K))-1);
        T6=B/(log(Rs6)-log(A))-273.15;


        if(T1>0)                    //rounding results
        {
          rounded=0.5;
        }
        if(T1<0)
        {
          rounded=-0.5;
        }
        T1=(floor(T1*100+rounded))/100;

        if(T2>0)
        {
          rounded=0.5;
        }
        if(T2<0)
        {
          rounded=-0.5;
        }
        T2=(floor(T2*100+rounded))/100;
        if(T3>0)
        {
          rounded=0.5;
        }
        if(T3<0)
        {
          rounded=-0.5;
        }
```

```
        T3=(floor(T3*100+rounded))/100;
        if(T4>0)
        {
          rounded=0.5;
        }
        if(T4<0)
        {
          rounded=-0.5;
        }
        T4=(floor(T4*100+rounded))/100;
        if(T5>0)
        {
          rounded=0.5;
        }
        if(T5<0)
        {
          rounded=-0.5;
        }
        T5=(floor(T5*100+rounded))/100;
        if(T6>0)
        {
          rounded=0.5;
        }
        if(T6<0)
        {
          rounded=-0.5;
        }
        T6=(floor(T6*100+rounded))/100;

        CRC=CRC_SEND();

        data_matrix[Ctrad][0]=count;      //i put them in data_matrix
        data_matrix[Ctrad][1]=T1;
        data_matrix[Ctrad][2]=T2;
        data_matrix[Ctrad][3]=T3;
        data_matrix[Ctrad][4]=T4;
        data_matrix[Ctrad][5]=T5;
        data_matrix[Ctrad][6]=T6;
        data_matrix[Ctrad][7]=CRC;

        Ctrad++;
        if(Ctrad==righe)
          {
```

```
                        Ctrad=0;
                      }
                Serial.println("Ctrad");
                Serial.println(Ctrad);
                count ++;


              }
          countTC1=countTC1 +1;
          countTC2=countTC2 +1;

          REG_TC4_INTFLAG = TC_INTFLAG_MC0;             // Clear the MC0 interrupt flag
      }

            // Check for match counter 1 (MC1) interrupt
          if (TC4->COUNT8.INTFLAG.bit.MC1 && TC4->COUNT8.INTENSET.bit.MC1)
            {
    // Put your counter compare 1 (CC1) code here:
    // ...

                REG_TC4_INTFLAG = TC_INTFLAG_MC1;           // Clear the MC1 interrupt flag
            }
  }

void init_interrupt()
{
     // Set up the generic clock (GCLK4) used to clock timers
  REG_GCLK_GENDIV = GCLK_GENDIV_DIV(3) |            // Divide the 48MHz clock source by divisor 3: 48MHz/3=16MHz
                    GCLK_GENDIV_ID(4);              // Select Generic Clock (GCLK) 4
   while (GCLK->STATUS.bit.SYNCBUSY);               // Wait for synchronization

  REG_GCLK_GENCTRL = GCLK_GENCTRL_IDC |             // Set the duty cycle to 50/50 HIGH/LOW
                     GCLK_GENCTRL_GENEN |           // Enable GCLK4
```

```
                        GCLK_GENCTRL_SRC_DFLL48M |    // Set the 48MHz clock source
                        GCLK_GENCTRL_ID(4);           // Select GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                  // Wait for synchronization

  // Feed GCLK4 to TC4 and TC5
  REG_GCLK_CLKCTRL = GCLK_CLKCTRL_CLKEN |             // Enable GCLK4 to TC4 and TC5
                        GCLK_CLKCTRL_GEN_GCLK4 |      // Select GCLK4
                        GCLK_CLKCTRL_ID_TC4_TC5;      // Feed the GCLK4 to TC4 and TC5
  while (GCLK->STATUS.bit.SYNCBUSY);                  // Wait for synchronization

  REG_TC4_CTRLA |= TC_CTRLA_MODE_COUNT8;               // Set the counter to 8-bit mode
  while (TC4->COUNT8.STATUS.bit.SYNCBUSY);            // Wait for synchronization

  REG_TC4_COUNT8_CC0 = 0x55;                          // Set the TC4 CC0 register to some arbitary value
  while (TC4->COUNT8.STATUS.bit.SYNCBUSY);            // Wait for synchronization
  REG_TC4_COUNT8_CC1 = 0xAA;                          // Set the TC4 CC1 register to some arbitary value
  while (TC4->COUNT8.STATUS.bit.SYNCBUSY);            // Wait for synchronization
  REG_TC4_COUNT8_PER = 0xFF;                          // Set the PER (period) register to its maximum value
  while (TC4->COUNT8.STATUS.bit.SYNCBUSY);            // Wait for synchronization

  //NVIC_DisableIRQ(TC4_IRQn);
  //NVIC_ClearPendingIRQ(TC4_IRQn);
  NVIC_SetPriority(TC4_IRQn, 0);    // Set the Nested Vector Interrupt Controller (NVIC) priority for TC4 to 0 (highest)
  NVIC_EnableIRQ(TC4_IRQn);         // Connect TC4 to Nested Vector Interrupt Controller (NVIC)

  REG_TC4_INTFLAG |= TC_INTFLAG_MC1 | TC_INTFLAG_MC0 | TC_INTFLAG_OVF;        // Clear the interrupt flags
  REG_TC4_INTENSET = TC_INTENSET_MC1 | TC_INTENSET_MC0 | TC_INTENSET_OVF;     // Enable TC4 interrupts
  // REG_TC4_INTENCLR = TC_INTENCLR_MC1 | TC_INTENCLR_MC0 | TC_INTENCLR_OVF;  // Disable TC4 interrupts

  REG_TC4_CTRLA |= TC_CTRLA_PRESCALER_DIV64 |         // Set prescaler to 64, 16 MHz/64 = 256kHz
```

```
                         TC_CTRLA_ENABLE;                    // Enable TC4
  while (TC4->COUNT8.STATUS.bit.SYNCBUSY);                   // Wait for synchronization
}

int CRC_SEND()                                                             //CRC_SEND
{

                                          //analising CRC of data
     T7=((T1-int(T1))*100);
     T8=((T2-int(T2))*100);
     T9=((T3-int(T3))*100);


     T10=((T4-int(T4))*100);
     T11=((T5-int(T5))*100);
     T12=((T6-int(T6))*100);

     T13=((T7-int(T7))*100);
     T14=((T8-int(T8))*100);
     T15=((T9-int(T9))*100);


     T16=((T10-int(T10))*100);
     T17=((T11-int(T11))*100);
     T18=((T12-int(T12))*100);

     const byte data[]={T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17,T18};

//                                   fine oled
     CRC=CRC8(data,18);
     return CRC;
}

void DISPLAY_FUNC()                                                                      //DISPLAY_FUNC
    {
                                                              //display_count modify the display

     if(!digitalRead(BUTTON_B))                               //button A incremented display_count
```

```
        {
         display_count=display_count+1;
        }

    if(display_count>=8)
        {
         display_count=1;
        }
    display.clearDisplay();
    display.setCursor(0,0);
    Vbattery=analogRead(A7);
    Vbattery *= 2;                    // we divided by 2, so multiply back
    Vbattery *=3.3*RRef/(R11+RRef);   // Multiply by 3.3V, our reference voltage
    Vbattery /= pow(2,bits);          // convert to voltage

    if(sending==0)                    //if i send data i show it in the display
    {
      sending1=' ';
    }
    else
    {
      sending1='*';
      sending=0;
    }

    if (display_count==1)
        {
         display.setTextSize(2);
         display.println(" ");
         display.print("T1:");
         display.print(round(T1),1);
         display.print(" C");

        }

    if (display_count==2)
        {
         display.setTextSize(2);
         display.println(" ");
         display.print("T2:");
```

```
            display.print(T2,1);
            display.print(" C");


        }
    if (display_count==3)
        {
            display.setTextSize(2);
            display.println(" ");
            display.print("T3:");
            display.print(T3,1);
            display.print(" C");


        }
    if (display_count==4)
        {
            display.setTextSize(2);
            display.println(" ");
            display.print("T4:");
            display.print(T4,1);
            display.print(" C");


        }
    if (display_count==5)
        {
            display.setTextSize(2);
            display.println(" ");
            display.print("T5:");
            display.print(T5,1);
            display.print(" C");


        }

    if (display_count==6)
        {
            display.setTextSize(2);
            display.println(" ");
            display.print("T6:");
            display.print(T6,1);
            display.print(" C");
```

```
        }

    if (display_count==7)
        {
          display.setTextSize(1);
          display.println(" ");
          display.print(T1,1);

          display.print(" ");
          display.print(T2,1);

          display.print(" ");
          display.print(T3,1);
          display.print(" ");
          display.print(" ");
          display.print(" ");
          display.println(Vbattery);
          display.println(" ");
          display.print(T4,1);

          display.print(" ");
          display.print(T5,1);

          display.print(" ");
          display.print(T6,1);
          display.print(" ");
          display.print(" ");
          display.print(" ");
          display.print(" ");
          display.println(sending1);


        }

    if(!digitalRead(BUTTON_C))                  //button C modify N_filt(4 or 16)
        {

          display.clearDisplay();
          display.setCursor(0,0);
          if(N_filt==4)
        {
```

```
                N_filt=16;
          }
           else
          {
                N_filt=4;
          }
           display.setTextSize(1);
           delay(10);
           display.print("N campioni  mediati:");
           display.println(" ");
           display.println(" ");
           display.setTextSize(2);
           display.print(N_filt);
           countTC2=-1800;

          }

      display.display();
    }


void MOD_INTERRUPT()                    //this function modify data sampling
{
     x=int(t_signal[2]);
     switch(x)
    {

           case 0:
  interrupt1=600;
     break;
           case 1:
  interrupt1=800;
     break;
           case 2:
  interrupt1=1000;
     break;
           case 3:
  interrupt1=1200;
     break;
           case 4:
  interrupt1=1400;
     break;
           case 5:
```

```
   interrupt1=1600;
     break;
           case 6:
   interrupt1=1800;
     break;
           case 7:
   interrupt1=2000;
     break;
           case 8:
   interrupt1=2200;
     break;
           case 9:
   interrupt1=2500;
     break;
           case 10:
   interrupt1=3000;
     break;
           case 11:
   interrupt1=3500;
     break;
           case 12:
   interrupt1=4000;
     break;
           case 13:
   interrupt1=4500;
     break;
           case 14:
   interrupt1=5000;
     break;
           case 15:
   interrupt1=10000;
     break;
         }

    }
```

Receiver code

```

// Feather9x_RX
// -*- mode: C++ -*-

#include <SPI.h>
#include <RH_RF95.h>

// for feather m0
#define RFM95_CS 8
#define RFM95_RST 4
#define RFM95_INT 3

// Change to 434.0 or other frequency, must match RX's freq!
#define RF95_FREQ 915.0

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);                          //freq setted

// Blinky on receipt
#define LED 13

int i;
char buf_string[100];
char vettString_rec[100];// array for GUI

char *token;
 int result;              //if array recived has no problem is 0, 1 for different CRC, 2 for signal lost
int  count=0;              //count of Rx
int count_sended=0;       //count of Tx

```

```
float r_signal[12];
byte CRC;                    //CRC of Rx
char *ptr;

int error=0;
byte tempI=0;
int double_send=-1;

int CRC_sended=0;  //CRC of Tx

float b1=0;          //with CRC i compare also 4 decimal
float b2=0;
float b3=0;
float b4=0;
float b5=0;
float b6=0;
float b7=0;
float b8=0;
float b9=0;
float b10=0;
float b11=0;
float b12=0;
float b13=0;
float b14=0;
float b15=0;
float b16=0;
float b17=0;
float b18=0;

int first=0;

byte crc = 0x00;  //for subfunction in CRC8
byte extract = 0;
byte sum =0;

int GUI_signal=' ';
int GUI_signal2=' ';

        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
        uint8_t len = sizeof(buf);

void setup()
   {
```

```

  pinMode(LED, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  digitalWrite(RFM95_RST, HIGH);

  Serial.begin(115200);
  while (!Serial)
      {
       delay(1);
      }
  delay(100);

  Serial.println("Feather LoRa RX Test!");

// manual reset
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);
  digitalWrite(8, LOW);
  while (!rf95.init())
      {
       Serial.println("LoRa radio init failed");
       while (1);
      }
  Serial.println("LoRa radio init OK!");

// Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ))
      {
       Serial.println("setFrequency failed");
       while (1);
     }
  Serial.print("Set Freq to: "); Serial.println(RF95_FREQ);

// Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128 chips/symbol, CRC on

// The default transmitter power is 13dBm, using PA_BOOST.
// If you are using RFM95/96/97/98 modules which uses the PA_BOOST transmitter pin, then
// you can set transmitter powers from 5 to 23 dBm:
rf95.setTxPower(23, false);
```

```
}
void loop()
   {

     if (rf95.available())
        {

          buf[RH_RF95_MAX_MESSAGE_LEN];
          len = sizeof(buf);

          if (rf95.recv(buf, &len))//we are receving
             {
              sprintf(buf_string,"%s",buf);

              token=strtok(buf_string,",");    //this instruction is for  analize
                   the array coming

              i=0;
              while(token!=NULL)
                {
                  r_signal[i]= strtod   (token,&ptr);
                  i++;
                  token=strtok(NULL,",");
                }

              count_sended=r_signal[0]; //i rewrite values for a greater
                  readability
              CRC_sended=r_signal[1];

              b1=r_signal[2];
              b2=r_signal[3];
              b3=r_signal[4];
              b4=r_signal[5];
              b5=r_signal[6];
              b6=r_signal[7];

          /*Serial.println("count");
          Serial.println(count);
          Serial.println("count_sended");
          Serial.println(count_sended);*/

              CRC=CRC_SEND();

```

```
            if((CRC-CRC_sended)!=0)                //if CRC is different result=1
              {
               result=1;

              }
            else if(count_sended !=(count))        //if C_inf is different result=2
              {
               result=2;
               error=error+1;

             }
            else                                   //otherwise result=0
             {
               result=0;
               count=count+1;
               error=0;
               if((count_sended-double_send)!=0)
        //sending if data is correct and if there are no 2 same value
               {
               sprintf(vettString_rec,"%d %3.2f %3.2f %3.2f %3.2f %3.2f %3.2f %3.2f ",count_sended,r_signal[2],r_signal[3],r_signal[4],r_signal[5],r_signal[6],r_signal[7],r_signal[8]);
               Serial.println(vettString_rec);
               double_send=count_sended;
               }

             }

            if(error>1)
    //if for an error Cinf_sended>C_inf will be a loop, if i see for 3 times error in C_inf this stops the loop
             {
              count=count_sended ;
              error=0;
             }

            GUI_signal=Serial.read();
            GUI_signal2= int(GUI_signal)-'0';


            sprintf(buf_string,"%d,%d,%d",result,count_sended,GUI_signal2);
```

```
  //first the result, second what i have compared and third if i have to
      change data sample


              rf95.send((uint8_t *)buf_string, 500);

            }
        else
            {
              Serial.println("Receive failed");

            }


        }
    }

//CRC-8 - CRC-8 formula-based algorithm of Dallas/Maxim
//code published under licence GNU GPL 3.0
byte CRC8(const byte *data, byte len)
{
     crc = 0x00;
     while (len--)
         {
          extract = *data++;
          for (tempI = 8; tempI; tempI--)
              {
               sum = (crc ^ extract) & 0x01;
               crc >>= 1;
               if (sum)
                   {
                    crc ^= 0x8C;
                   }
               extract >>= 1;
              }
         }
     return crc;
}

int CRC_SEND()    //here i save also decimal ciphers and i will send them to
    CRC8
    {
     b7=((b1-int(b1))*100);
```

```
    b8=((b2-int(b2))*100);
    b9=((b3-int(b3))*100);


    b10=((b4-int(b4))*100);
    b11=((b5-int(b5))*100);
    b12=((b6-int(b6))*100);
    b13=((b7-int(b7))*100);
    b14=((b8-int(b8))*100);
    b15=((b9-int(b9))*100);
    b16=((b10-int(b10))*100);
    b17=((b11-int(b11))*100);
    b18=((b12-int(b12))*100);

    const byte data[]={b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,
        b17,b18};
    CRC=CRC8(data,18);
    return CRC;
  }
```

GUI code

```


from tkinter import*
import os.path
from matplotlib.pyplot import show,ion,draw
import matplotlib.pyplot as plt
import serial
import time
from matplotlib.collections import EventCollection
import numpy as np
import matplotlib.animation as animation
global countDataTime
countDataTime=2
global varDataTime
varDataTime=2
global xlabl
xlabl=1
global can1 #channels, they can be active or disactive
can1=1
global can2
can2=1
global can3
can3=1
global can4
can4=1
global can5
can5=1
global can6
can6=1
global button_1
global go # when is =1 we take data from serial port
```

```
global firstime # is for enter the first time in the main loop
firstime=1
global interazione #   =1 i can only interact with figure
interazione=0
global figure
figure=0
global first_click# is for return and new data
first_click = 0
global labell
labell=0
global loading
loading = 0
global newsho #newsho mi serve a mettere il warning su write data
newsho = 0
global batt
batt="4.10"
global label_bat2

go='s'

def animate(i):        #this subfunction is for read data from txt

    global text
    global changing
    global interazione
    global adding
    global xlabl

    if not interazione:
        global can1
        global can2
        global can3
        global can4
        global can5
        global can6
        #print(text)
        pullData = open(text,"r").read()
        dataArray = pullData.split('\n')
        X = []
        Y1 = []
        Y2 = []
        Y3 = []
        Y4 = []
```

```
        Y5 = []
        Y6 = []

        for eachLine in dataArray:      # is for reading all lines
            if len(eachLine)>1:
                x,xt,y1,y2,y3,y4,y5,y6,batt,c = eachLine.split(' ')  #in txt i read: position x in txt file , number of transmitter data , data from data 1 to 6 and /n
                X.append(float(x))
                Y1.append(float(y1))
                Y2.append(float(y2))
                Y3.append(float(y3))
                Y4.append(float(y4))
                Y5.append(float(y5))
                Y6.append(float(y6))

        ax1.clear()
                        #if i want to disactivate some channel it will disappear from the window
        if can1:
            ax1.plot(X,Y1)
            plt.plot(X,Y1,"b",label="CH 1")
        if can2:
            ax1.plot(X,Y2)
            plt.plot(X,Y2,"r",label="CH 2")
        if can3:
            ax1.plot(X,Y3)
            plt.plot(X,Y3,"g",label="CH 3")
        if can4:
            ax1.plot(X,Y4)
            plt.plot(X,Y4,"orange",label="CH 4")
        if can5:
            ax1.plot(X,Y5)
            plt.plot(X,Y5,"aqua",label="CH 5")
        if can6:
            ax1.plot(X,Y6)
            plt.plot(X,Y6,"m",label="CH 6")

        plt.xlabel("n")
        plt.ylabel("*c")
        plt.title("CHANNELS AND tmp102 in *C")
        plt.legend()

```

```

    command=ricezione()

def ricezione():
    global text
    global batt
    if(arduino.inWaiting()):                                                    #i do this if serial port is active
        TEMP = arduino.readline()
        string_to_display = TEMP.decode()
        global can1
        global can2
        global can3

        if go=='a':           #now i want to receve
            x, y1, y2, y3, y4, y5, y6,batt,c = string_to_display.split(' ')  # variables with data taked by serial port
            print(string_to_display)
                                                    #if result is 0 i want to read
            print("ok")
            print(string_to_display)
            global xlabl
            pullData = open(text, "r").read()      #xlabl has the value of last row in txt file
            dataArray = pullData.split('\n')
            xlabl=len(dataArray)
            global Label_7
            global Label_8
            global Label_9
            global Label_71
            global Label_81
            global Label_91
            strxlabl=str(xlabl)

            new_string_to_display = " ".join([strxlabl,x, y1,y2,y3, y4,y5,y6,batt,'\n'])     # in txt file are written: data number ;T1,T2,T3,T4,T5,T6
            file = open(text,"a")
            file.write(new_string_to_display)
            file.close()

            Label_7.grid_remove()
```

```
            Label_8.grid_remove()
            Label_9.grid_remove()
            Label_71.grid_remove()
            Label_81.grid_remove()
            Label_91.grid_remove()
            global label_bat2
            label_bat2.grid_forget()
            label_bat2 = Label(window, text=batt, foreground="black", font="
                none 12 bold")
            label_bat2.grid(row=13, column=1, sticky=N)


            #control if channels are active or disactive to print or not data
            if can1:
                Label_7=Label(window, text=y1, foreground="black", font="none
                    12 bold")
                Label_7.grid(row=6, column=3, sticky=N)
            else:
                Label_7.grid_remove()
            if can2:
                Label_8 =Label(window, text=y2, foreground="black", font="none
                     12 bold")
                Label_8.grid(row=7, column=3, sticky=N)
            else:
                Label_8.grid_remove()
            if can3:
                Label_9 =Label(window, text=y3, foreground="black", font="none
                     12 bold")
                Label_9.grid(row=8, column=3, sticky=N)
            else:
                Label_9.grid_remove()
            if can4:
                Label_71=Label(window, text=y4, foreground="black", font="none
                     12 bold")
                Label_71.grid(row=9, column=3, sticky=N)
            else:
                Label_71.grid_remove()
            if can5:
                Label_81 =Label(window, text=y5, foreground="black", font="
                    none 12 bold")
                Label_81.grid(row=10, column=3, sticky=N)
            else:
                Label_81.grid_remove()
```

```
            if can6:
                Label_91 =Label(window, text=y6, foreground="black", font="none 12 bold")
                Label_91.grid(row=11, column=3, sticky=N)
            else:
                Label_91.grid_remove()

def spegnimento_feather():          #this function permitts to not take other data
    global button_1
    global button_2
    global label_0
    button_1.grid_forget()                              #grid forget is for delete color of buttons for write another one
    button_2.grid_forget()
    button_2=Button(window,text = "STOP",command=spegnimento_feather, background="blue")      #blue when the state is active
    button_1=Button(window,text = "START",command=accensione_feather, background="green")      #green is inactive(only for start, every button has his own color for inactive state)
    global go
    go='s'                          #this don't permitt to enter in the routine
    print(go)
    Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
    Label_0=Label(window, text="    STOP SAVING DATA    ", foreground="black")
    button_2.grid(row=3, column=1,sticky=N)                                              #where to put buttons or label
    button_1.grid(row=2, column=1,sticky=N)
    Label_0.grid(row=0, column=0,sticky=N)

def accensione_feather():       #this function permitts to take other data
    global button_1
    global button_2
    global label_0
    button_1.grid_forget()
    button_2.grid_forget()
    button_1=Button(window,text = "START",command=accensione_feather, background="blue")
    button_2=Button(window,text = "STOP",command=spegnimento_feather, background="red")
    global go
    go='a'                      #when go is 'a' we take data
```

```
    global firstime
    Label_0=Label(window, text="         SAVING DATA         ", foreground="black")
    if firstime:
        firstime=0
        command = ricezione()
    button_1.grid(row=2, column=1,sticky=N)
    button_2.grid(row=3, column=1,sticky=N)
    Label_0.grid(row=0, column=0,sticky=N)


def analisi_dati():           #i make interaction to 1 for interact with figure
    global button_3
    button_3.grid_forget()
    global label_0

    global interazione                #with interaction to 1 we can only interact with figure
    if interazione:
        interazione=0
        button_3 = Button(window, text="INTERATION", command=analisi_dati, background="yellow")
        Label_0=Label(window, text="                                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="     GUI FIGURA1 STOP   ", foreground="black")
    else:
        interazione=1
        button_3 = Button(window, text="INTERATION", command=analisi_dati, background="blue")
        Label_0=Label(window, text="                                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="      USING GUI FIGURA1      ", foreground="black")
    button_3.grid(row=4, column=1,sticky=N)
    Label_0.grid(row=0, column=0,sticky=N)

def canale1():                               #to activate or not channels
    global can1
    global button_5
    button_5.grid_forget()
    if can1:
        can1=0
```

```
        button_5 = Button(window, text="CH1", command=canale1, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH1 DISABLED    ", foreground="black")
    else:
        can1=1
        button_5 = Button(window, text="CH1", command=canale1, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH1 ABLED       ", foreground="black")
    button_5.grid(row=6, column=1, sticky=N)
    Label_0.grid(row=0, column=0, sticky=N)


def canale2():
    global can2
    global button_6
    button_6.grid_forget()
    if can2:
        can2=0
        button_6 = Button(window, text="CH2", command=canale2, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH2 DISABLED    ", foreground="black")
    else:
        can2=1
        button_6 = Button(window, text="CH2", command=canale2, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH2 ABLED       ", foreground="black")
    Label_0.grid(row=0, column=0,sticky=N)
    button_6.grid(row=7, column=1, sticky=N)

def canale3():
    global can3
    global button_7
    button_7.grid_forget()
    if can3:
        can3=0
```

```
        button_7 = Button(window, text="CH3", command=canale3, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH3 DISABLED    ", foreground="black")
    else:
        can3=1
        button_7 = Button(window, text="CH3", command=canale3, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0 = Label(window, text="    CH3 ABLED        ", foreground="black")
    button_7.grid(row=8, column=1, sticky=N)
    Label_0.grid(row=0, column=0,sticky=N)

def canale4():
    global can4
    global button_51
    button_51.grid_forget()
    if can4:
        can4=0
        button_51 = Button(window, text="CH4", command=canale4, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH4 DISABLED    ", foreground="black")
    else:
        can4=1
        button_51 = Button(window, text="CH4", command=canale4, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH4 ABLED        ", foreground="black")
    button_51.grid(row=9, column=1, sticky=N)
    Label_0.grid(row=0, column=0, sticky=N)

def canale5():
    global can5
    global button_61
    button_61.grid_forget()
    if can5:
        can5=0
```

```
        button_61 = Button(window, text="CH5", command=canale5, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH5 DISABLED    ", foreground="black")
    else:
        can5=1
        button_61 = Button(window, text="CH5", command=canale5, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH5 ABLED       ", foreground="black")
    Label_0.grid(row=0, column=0,sticky=N)
    button_61.grid(row=10, column=1, sticky=N)

def canale6():
    global can6
    global button_71
    button_71.grid_forget()
    if can6:
        can6=0
        button_71 = Button(window, text="CH6", command=canale6, background="blue")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0=Label(window, text="    CH6 DISABLED    ", foreground="black")
    else:
        can6=1
        button_71 = Button(window, text="CH6", command=canale6, background="orange")
        Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
        Label_0 = Label(window, text="    CH6 ABLED       ", foreground="black")
    button_71.grid(row=11, column=1, sticky=N)
    Label_0.grid(row=0, column=0,sticky=N)

def battery():
    global batt
    global label_bat2
    label_bat2.grid_forget()
    label_bat2 = Label(window, text=batt, foreground="black", font="none 12 bold")
```

```
    label_bat2.grid(row=13, column=2, sticky=N)

def datatime():              #to change data sampling in transmitter

    global varDataTime
    global countDataTime
    global button_83
    button_83.grid_forget()
    countDataTime=countDataTime+1
    if countDataTime==16:
        countDataTime=0

    if countDataTime==0:                    #countDataTime can have 16 possibilities, every time i touch the button it changes
        varDataTime='0'
        button_83 = Button(window, text="600", command=datatime)
    if countDataTime==1:
        varDataTime='1'
        button_83 = Button(window, text="800", command=datatime)
    if countDataTime==2:
        varDataTime='2'
        button_83 = Button(window, text="1000", command=datatime)
    if countDataTime==3:
        varDataTime='3'
        button_83 = Button(window, text="1200", command=datatime)
    if countDataTime==4:
        varDataTime='4'
        button_83 = Button(window, text="1400", command=datatime)
    if countDataTime==5:
        varDataTime='5'
        button_83 = Button(window, text="1600", command=datatime)
    if countDataTime==6:
        varDataTime='6'
        button_83 = Button(window, text="1800", command=datatime)
    if countDataTime==7:
        varDataTime='7'
        button_83 = Button(window, text="2000", command=datatime)
    if countDataTime==8:
        varDataTime='8'
        button_83 = Button(window, text="2200", command=datatime)
    if countDataTime==9:
        varDataTime='9'
        button_83 = Button(window, text="2500", command=datatime)
```

```
    if countDataTime==10:
        varDataTime=':'
        button_83 = Button(window, text="3000", command=datatime)
    if countDataTime==11:
        varDataTime=';'
        button_83 = Button(window, text="3500", command=datatime)
    if countDataTime==12:
        varDataTime='<'
        button_83 = Button(window, text="4000", command=datatime)
    if countDataTime==13:
        varDataTime='='
        button_83 = Button(window, text="4500", command=datatime)
    if countDataTime==14:
        varDataTime='>'
        button_83 = Button(window, text="5000", command=datatime)
    if countDataTime==15:
        varDataTime='?'
        button_83 = Button(window, text="10000", command=datatime)


    print(countDataTime)
    button_83.grid(row=12, column=1, sticky=N)


def submit():                   #now i submit the new data sampling touching submit to transmitter
    global varDataTime
    sending=str(varDataTime)
    arduino.write(sending.encode('latin_1'))
    Label_0=Label(window, text="                                        ", foreground="black", font="none 12 bold")
    Label_0 =Label(window, text="    SENDING          ", foreground="black")
    Label_0.grid(row=0, column=0,sticky=N)
    print(sending)

def cancel_data():              #this clear the file txt
    file = open(text, "w+")
    file.close()
    Label_0 =Label(window, text="                                        ", foreground="black", font="none 12 bold")
    Label_0 =Label(window, text="    DATA CANCELLED         ", foreground="black")
    Label_0.grid(row=0, column=0,sticky=N)
```

```
def new_data():        #for taking new data, we overwrite the file if there is something
    global xlabl
    xlabl=0
    global interazione
    global text
    global newsho

    interazione=0
    filename =entry_1.get()      #we can change the name of txt file, if we doesn't use this command is data.txt
    print(filename)
    if(filename==""):
        filename="data"
    extenction=".txt"
    text = " ".join([filename,extenction])
    if os.path.isfile(text):
        print("exist")
        Label_17.grid(row=2, column=1)
        Label_18.grid(row=3, column=1)
        newsho+=1
    else:
        print("file doesn't exist")
        Label_17.grid_forget()
        Label_18.grid_forget()
        newsho=2
    if(newsho>1):
        file = open(text, "w+")
        file.close()
        Label_0 = Label(window, text="      NEW DATA ", font="none 12 bold")
        Label_0.grid(row=0, column=0)
        button_1.grid(row=2, column=1, sticky=N)
        label_1.grid(row=2, column=0, sticky=W)
        label_2.grid(row=3, column=0, sticky=W)
        button_2.grid(row=3, column=1, sticky=N)
        button_3.grid(row=4, column=1, sticky=N)
        label_3.grid(row=4, column=0, sticky=W)
        button_4.grid(row=5, column=1, sticky=N)
        label_4.grid(row=5, column=0, sticky=W)
        label_5.grid(row=6, column=0, sticky=W)
        button_5.grid(row=6, column=1, sticky=N)
        button_6.grid(row=7, column=1, sticky=N)
```

```
        button_7.grid(row=8, column=1, sticky=N)
        button_51.grid(row=9, column=1, sticky=N)
        button_61.grid(row=10, column=1, sticky=N)
        button_71.grid(row=11, column=1, sticky=N)
        Label_7.grid(row=6, column=3, sticky=N)
        Label_8.grid(row=7, column=3, sticky=N)
        Label_9.grid(row=8, column=3, sticky=N)
        Label_71.grid(row=9, column=3, sticky=N)
        Label_81.grid(row=10, column=3, sticky=N)
        Label_91.grid(row=11, column=3, sticky=N)
        button_82.grid(row=12, column=2, sticky=N)
        button_83.grid(row=12, column=1, sticky=N)
        Label_83.grid(row=12, column=0, sticky=N)
        button_15.grid(row=0, column=3, sticky=N)
        label_bat1.grid(row=13, column=0)
        label_bat2.grid(row=13, column=1)
        Label_14.grid_forget()
        button_13.grid_forget()
        button_16.grid_forget()
        Label_16.grid_forget()
        entry_1.grid_forget()
        Label_17.grid_forget()
        Label_18.grid_forget()
        Label_19.grid_forget()
        newsho=0
        plt.show()

def returnn():          #the button return permitts to go in the men , in
   witch there are new data or add data
    ax1.clear()
    global figure
    global go
    go='s'
    global interazione
    global button_1
    global button_2
    global button_5
    global button_6
    global button_7
    global label_bat2
    interazione=1
    Label_0 = Label(window, text="        MENU          ", font="none 12 bold"
       )
```

```
    Label_0.grid(row=0, column=0)
    button_1.grid_forget()
    label_1.grid_forget()
    label_2.grid_forget()
    button_2.grid_forget()
    button_3.grid_forget()
    label_3.grid_forget()
    button_4.grid_forget()
    label_4.grid_forget()
    label_5.grid_forget()
    button_5.grid_forget()
    button_6.grid_forget()
    button_7.grid_forget()
    Label_7.grid_forget()
    Label_8.grid_forget()
    Label_9.grid_forget()
    Label_71.grid_forget()
    Label_81.grid_forget()
    Label_91.grid_forget()
    button_82.grid_forget()
    button_83.grid_forget()
    Label_83.grid_forget()
    button_2.grid_forget()
    button_3.grid_forget()
    button_5.grid_forget()
    button_6.grid_forget()
    button_7.grid_forget()
    button_51.grid_forget()
    button_61.grid_forget()
    button_71.grid_forget()
    button_15.grid_forget()
    label_bat1.grid_forget()
    label_bat2.grid_forget()
    button_1=Button(window,text = "START",command=accensione_feather,
        background="green")
    button_2 = Button(window, text="STOP", command=spegnimento_feather,
        background="red")
    entry_1.grid(row=1, column=1)
    Label_14.grid(row=1,column=0)
    button_13.grid(row=2, column=0)
    button_16.grid(row=3, column=0)
    Label_16.grid(row=1, column=2)
    Label_19.grid(row=5, column=0)
```

```
    entry_1.grid(row=1, column=1)


def add():          #if we press add data we don't overwrite anithing, we add
    data to an existence file

    global text
    global interazione

    interazione = 1
    filename = entry_1.get()
    print(filename)
    if (filename == ""):
        filename = "data"
    extenction = ".txt"
    text = " ".join([filename, extenction])
    if os.path.isfile(text):
        print("exist")
        Label_15.grid_forget()
        interazione = 0
    else:
        print("file doesn't exist")
        Label_15.grid(row=3, column=1)
    if not(interazione):
        Label_15.grid_forget()
        Label_0 = Label(window, text="  ADD/LOAD DATA ", font="none 12 bold")
        Label_0.grid(row=0, column=0)
        button_1.grid(row=2, column=1, sticky=N)
        label_1.grid(row=2, column=0, sticky=W)
        label_2.grid(row=3, column=0, sticky=W)
        button_2.grid(row=3, column=1, sticky=N)
        button_3.grid(row=4, column=1, sticky=N)
        label_3.grid(row=4, column=0, sticky=W)
        button_4.grid(row=5, column=1, sticky=N)
        label_4.grid(row=5, column=0, sticky=W)
        label_5.grid(row=6, column=0, sticky=W)
        button_5.grid(row=6, column=1, sticky=N)
        button_6.grid(row=7, column=1, sticky=N)
        button_7.grid(row=8, column=1, sticky=N)
        button_51.grid(row=9, column=1, sticky=N)
        button_61.grid(row=10, column=1, sticky=N)
        button_71.grid(row=11, column=1, sticky=N)
        Label_7.grid(row=6, column=3, sticky=N)
```

```
        Label_8.grid(row=7, column=3, sticky=N)
        Label_9.grid(row=8, column=3, sticky=N)
        Label_71.grid(row=9, column=3, sticky=N)
        Label_81.grid(row=10, column=3, sticky=N)
        Label_91.grid(row=11, column=3, sticky=N)
        button_82.grid(row=12, column=2, sticky=N)
        button_83.grid(row=12, column=1, sticky=N)
        Label_83.grid(row=12, column=0, sticky=N)
        button_15.grid(row=0, column=3, sticky=N)
        label_bat1.grid(row=13, column=0)
        label_bat2.grid(row=13, column=1)
        Label_15.grid_forget()
        Label_14.grid_forget()
        button_13.grid_forget()
        button_16.grid_forget()
        Label_16.grid_forget()
        Label_19.grid_forget()
        entry_1.grid_forget()
        plt.show()


window = Tk()
window.title("GUI")

fig = plt.figure()      #plotting the figure

ax1 = fig.add_subplot(1, 1, 1)

ani = animation.FuncAnimation(fig, animate, interval=1)     #enter in the subroutine animation for seeing data


var_1=StringVar()

#entry_1 = Entry(window)
arduino = serial.Serial('COM8', 9600)       #we see data from this serial port that is the same of the reciver

while (True):
#inizio ricezione
    Label_0 = Label(window, text="  MENU ", font="none 12 bold")
    Label_00 = Label(window,text="                         MENU      ", font="none 12 bold")
```

```
    button_1=Button(window,text = "START",command=accensione_feather,background="green")
    label_1=Label(window,text="press to save temperature",foreground="black",font="none 12 bold")
    button_2=Button(window,text = "STOP",command=spegnimento_feather,background="red")
    label_2=Label(window,text="press to stop saving temperature ",foreground="black",font="none 12 bold")
#inizio ricezione
    button_3=Button(window,text = "INTERATION",command=analisi_dati,background="yellow")
    label_3=Label(window, text="press to use GUI of figura1 ", foreground="black",font="none 12 bold")
#cancello valori.txt
    button_4=Button(window,text = "CANCEL DATA ",command=cancel_data)
    label_4=Label(window, text="press to cancel data", foreground="black",font="none 12 bold")

#scrivo i canali
    label_5=Label(window, text="press to print or not last channels ", foreground="black",font="none 12 bold")

    button_5 = Button(window, text="CH1", command=canale1, background="orange")
    button_6 = Button(window, text="CH2", command=canale2, background="orange")
    button_7 = Button(window, text="CH3", command=canale3, background="orange")
    button_51 = Button(window, text="CH4", command=canale4, background="orange")
    button_61 = Button(window, text="CH5", command=canale5, background="orange")
    button_71 = Button(window, text="CH6", command=canale6, background="orange")
    button_82 = Button(window, text="SUBMIT", command=submit)
    button_83 = Button(window, text="1000", command=datatime)
    Label_83 = Label(window, text="choose time for data acquisition(ms)", foreground="black", font="none 12 bold")
    label_bat1 = Label(window, text="battery tension(V)", foreground = "black", font = "none 12 bold")
    label_bat2 = Label(window, text="", foreground="black", font="none 12 bold")

```

```
    Label_7 = Label(window, text="                              ", foreground="black", font="none 12 bold")
    Label_8 = Label(window, text="                              ", foreground="black", font="none 12 bold")
    Label_9 = Label(window, text="                              ", foreground="black", font="none 12 bold")
    Label_71 = Label(window, text="                              ", foreground="black", font="none 12 bold")
    Label_81 = Label(window, text="                              ", foreground="black", font="none 12 bold")
    Label_91 = Label(window, text="                              ", foreground="black", font="none 12 bold")


#choosing file
    entry_1 = Entry(window)
    entry_1.grid(row=1, column=1)
    button_13 = Button(window, text="WRITE DATA", command=new_data)
    button_16 = Button(window, text="ADD/LOAD DATA", command=add)
    Label_14=Label(window, text="            select file ", foreground="black", font="none 12 bold")
    Label_17 = Label(window, text="WARNING: file already exist", foreground="black", font="none 12 bold")
    Label_18 = Label(window, text="press WRITE DATA to overwrite",foreground="black", font="none 12 bold")
    Label_19 = Label(window, text="(if file is not specified, GUI will use data.txt)", foreground="black", font="none 12 bold")

    Label_16 = Label(window, text=".txt", foreground="black")
    Label_0.grid(row=0, column=0)
    Label_14.grid(row=1,column=0)
    Label_16.grid(row=1, column=2)
    button_13.grid(row=2, column=0)
    button_16.grid(row=3, column=0)
    button_15 = Button(window, text="return ", command=returnn)
    Label_19.grid(row=5, column=0)
    #button_15.grid(row=0, column=3)

#GREED
    Label_15=Label(window, text="this file doesn't exist ", foreground="black",font="none 12 bold")

    window.mainloop()
```

List of Figures

List of Tables

Bibliography

[1] José González-Alonso. “Human thermoregulation and the cardiovascular system”. In: *Experimental Physiology* 97.3 (2012), pp. 340–346.

[2] J. Enrique Silva. “Thermogenic Mechanisms and Their Hormonal Regulation”. In: *Physiological Reviews* 86.2 (2006), pp. 435–464.

[3] Maohui Luo et al. “Revisiting an overlooked parameter in thermal comfort studies, the metabolic rate”. In: *Energy and Buildings* 118 (2016), pp. 152–159.

[4] *DETERMINATION OF THERMAL COMFORT TEMPERATURE*. URL: `http://ergo.human.cornell.edu/studentdownloads/DEA3500notes/Thermal/thcomnotes1.html`.

[5] W. Larry Kenney and Thayne A. Munce. “Invited Review: Aging and human temperature regulation”. In: *Journal of Applied Physiology* 95.6 (2003), pp. 2598–2603.

[6] Laura Bacci et al. “Thermohygrometric conditions of some urban parks of Florence (Italy) and their effects on human well-being”. In: *The Fifth International Conference on Urban Climate* 2 (Jan. 2003).

[7] *Datasheet MSR147WD*. URL: `https://www.msr.ch/media/pdf/Datalogger_MSR147WD_Datasheet.pdf`.

[8] *Datasheet MSR145*. URL: `https://www.msr.ch/media/pdf/Data_logger_MSR145_Data_sheet.pdf`.

[9] *Datasheet MSR160*. URL: `https://www.msr.ch/media/pdf/Data_logger_MSR160_Datasheet.pdf`.

[10] *VitalPatch Biosensor*. URL: `https://www.medgadget.com/2018/12/vitalpatch-biosensor-and-vistatablet-monitor-a-medgadget-review.html`.

[11] *BT510 Bluetooth 5 Long Range IP67 Multi-Sensor*. URL: https://www.lairdconnect.com/iot-devices/iot-sensors/bt510-bluetooth-5-long-range-ip67-multi-sensor?adgroup=bt510&gclid=Cj0KCQjwx7zzBRCcARIsABPRscP520L2NaHmu2HHQhWXxfGoMpokWyAEcqe0dVi3ngMaAhm5EALw_wcB&matchtype=p&sncid=13&utm_campaign=platforms_general&utm_medium=cpc&utm_source=googlel.

[12] *Datasheet Adafruit Feather M0 Radio with LoRa Radio Module*. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-m0-radio-with-lora-radio-module.pdf?timestamp=1584253431.

[13] *Manual Atmel SAM D21E / SAM D21G / SAM D21J*. URL: https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Atmel-42181-SAM-D21_Datasheet.pdf.

[14] *Datasheet RFM95/96/97/98(W)*. URL: https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf.

[15] Samuele Crivellaro. *protocollo di comunicazione i2C*. 2017. URL: http://www.crivellaro.net/wp-content/uploads/2017/03/Protocollo-di-comunicazione-I2C.pdf.

[16] Alberto Vallan. *Slides Sensori di temperatura*. 2020.

[17] *B57861S103F40 Datasheet*. URL: https://www.mouser.it/datasheet/2/400/NTC_Mini_sensors_S861-1317170.pdf.

Acknowledgements

Per questo progetto di tesi ringrazio il mio professore Alberto Vallan, al quale devo tanto per questo risultato; in questi mesi in cui progettavo il dispositivo mi ha insegnato molto, è stato sempre disponibile ed ha fatto tutto in un clima estremamente positiva per me e per gli altri tesisti, questa resterà per me sempre un'esperienza profonda.

Ringrazio la mia famiglia,i miei genitori Sauro e Rita, senza i quali tutto questo non sarebbe potuto essere possibile,sia economicamente e sia per tutto il sostegno che mi hanno sempre dato fin da piccolo, non voglio nemmeno dilungarmi troppo, un sentimento del genere non si può scrivere, ma solo vivere di persona.

A seguire i miei zii Lucia Filippo e Giuliana che sono sempre stati maestri di vita, facendomi spesso ragionare fin da piccolo a problemi che ancora non mi ponevo e standomi sempre vicini in ogni situazione.

Appena fuori dall'ambiente familiare ci tengo a ringraziare particolarmente una persona che considero come un fratello, il mio amico Davide, che mi ha sempre supportato moralmente, c'è sempre stato e rimarrà per me sempre una persona su cui fare totale affidamento.

Il mio amico Edoardo, altra persona molto vicina alla mia famiglia con la quale ho passato bellissimi momenti, una persona estremamente forte ed un medico che come tanti altri senza chiedere nulla a nessuno è a fronteggiare il COVID-19;lui e tutti i medici ed infermieri hanno tutto il mio sostegno in questo momento.

Un ringraziamento speciale a Marta, una persona che da poco è entrata come un tornado nella mia vita ed alla quale devo tanti momenti bellissimi, tanto supporto e tanta allegria; senza di lei quest'ultimo periodo sarebbe stato totalmente diverso.

Ringrazio inoltre tutti gli amici che mi sono stati sempre vicini e mi hanno sempre sostenuto, in particolare le persone a cui devo i momenti condivisi in questo periodo sono stati coloro con cui ho studiato in questo ultimo periodo Manny, federica e Sara,i miei amici e compagni di tesi Edoardo ed Aurora che sono stati con me ogni giorno negli ultimi mesi e con i quali ho passato bellissimi momenti sebbene li abbia conosciuti da poco, i miei coinquilini Alex e Silvia e tutti i miei amici di sempre,che non posso nominare tutti o i ringraziamenti non finirebbero più, grazie mille a tutti voi ragazzi.

Per ultimo devo essere grato a Francesco ed Andrea,i quali durante la scelta riguardante l'università mi hanno fatto conoscere il Politecnico di Torino, sono entrato qui grazie a loro.