POLITECNICO
DI TORINO

Master's Degree in Electronic Engineering

Thesis

Low power operation of indoor sensors for human localization

Supervisors
Mihai Teodor Lazarescu
Luciano Lavagno

Candidate
Paolo Alloatti

Academic year 2018-2019

Acknowledgements

I'm extremely grateful to Professor Mihai Lazarescu and Professor Luciano Lavagno for their continuous support and guidance during this project.

I would also like to thank all the colleagues that worked with me and helped me in this project, especially Osama Bin Tariq, that gave me many valuable advices.

Finally, I cannot express my gratitude for my family and friends that supported me for all these years.

Contents

List of Tables

List of Figures

Abstract

Sensors for indoor human localization, are nowadays largely exploited for monitoring and automation systems. In the context of new smart homes, the localization sensors have become a very important component and exist many of them. The most commonly used are radio frequency, infrared, pressure, ultrasound, video and capacitive sensors. All of them have peculiar features, in terms of data processing, easiness of setup and utilization, privacy. A good localization system should be tagless, privacy aware and passive. The first and third needs arise from the fact that anyone can be a possible user of the localization system and he/she doesn't have to know how to use it. In fact, a possible implementation for this system is the monitoring of elderly. Therefore, the system has to be transparent to users and they don't have to wear or bring a tag to be monitored. Then, is also very important, in the current scenario of many data scandals, to assure the privacy.
Capacitive sensors has been used in this project for their advantages: they have a really good sensitivity compared to their size, they are tagless, so suited for any user, and they are also privacy aware. The project involves many aspects, from the front-ends optimizations, to the sensor fusion, to the use of neural networks for localization algorithms.
Within this project, I based my work on the power optimization of a particular front-end, based on the 555 circuit as astable oscillator, on the ATMega328P as microcontroller [1] (Capacitive sensor) and on a movement sensor for only the detection, based on Wilmsdorff et al. work [2].
The capacitive sensor based on 555 relies on the capacitive coupling between a metal plate (8x8 cm) and the human body. In fact, the human body couples with objects in the environment and the resulting capacitance depends on the distance. Then, this capacitance is used to make oscillate the 555 in astable mode with fixed resistances. Finally, the microcontroller measures the oscillation frequency.
The main idea behind my thesis was to interface the movement sensor and the capacitive one in order to use the former, that consumes very low power, to wake up the latter, that consumes more energy and is kept asleep for the unnecessary time.
The operating flow can be seen qualitatively in picture 1: the movement sensor is

always ON and the 555-based front end is in sleep mode, with the 555 OFF. When the movement sensor detects a person it sends an external interrupt to the microcontroller that wakes it up. Then, it monitors presence until the person leaves the sensing range of the sensor and goes back to sleep mode. During measurements the front end supplies the 555 for the time of a measurement and then stays in a soft sleep mode for the remaining time. The system is so duty-cycled, with periods of measurement and sleep.

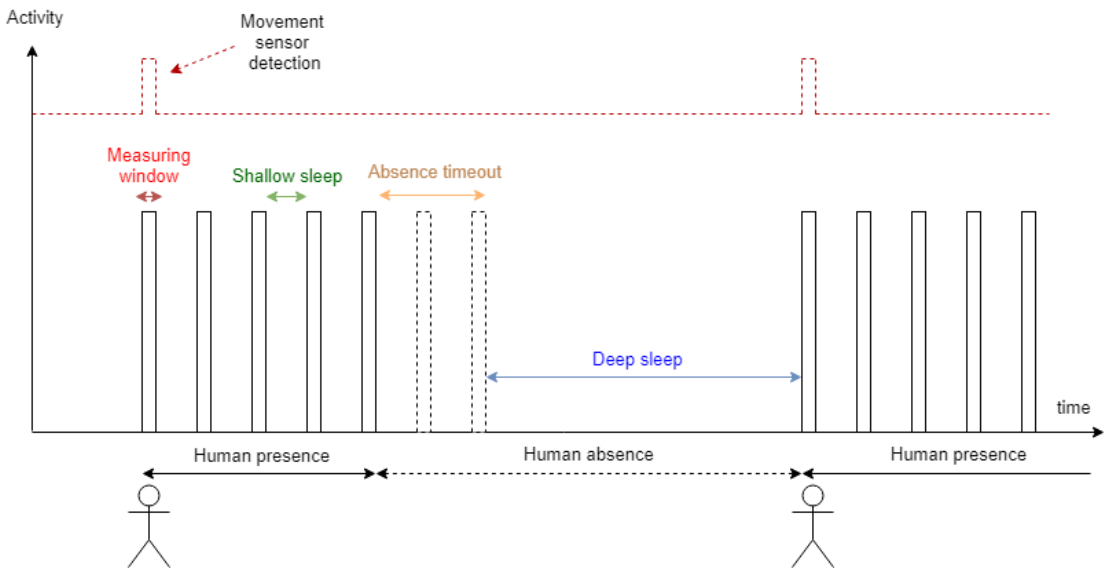


Figure 1. Operating flow of the system

The *movement sensor* exploits capacitive sensing techniques reported in [2] and changes voltage when there is a change in the electric fields in the environment. Hence, in general, it is sensible to electric fields. In the implementation, I used passive components and three very low power operational amplifiers, that have been used in this prototype as voltage follower, filter and comparator. The first stage is needed to provide high impedance for measurement to get the correct signal. Then it is denoised with a low pass filter with a cut-off frequency of $f_c = 2\,\text{Hz}$ to cut the 50 Hz network noise. Finally, the comparator translates the signal into a logical value to send it as an external interrupt to the microcontroller. The threshold of the comparator needs to be tuned in every environment, to signal to noise ratio. The *capacitive sensor* is composed of an astable oscillator using a 555 circuit and a microcontroller. To save power, I mainly used three different sleep modes: (from the deepest and less consuming) Power Down, Power save, Idle. The first is used

in sleep periods, while waiting the interrupt from the movement sensor, to start the measurement. The second one is used between consecutive measurements and exploits the watchdog timer (WDT) to wake it up. The last mode is used during measurements. In fact, the hybrid system of frequency measuring of the 555 is based on counting how many CPU clock cycles take a given number of 555 output signal periods using two timers, one clocked with a 16 MHz external oscillator and one with the 555 output. This system uses hardware counters to keep the microcontroller idle between overflows of the timers, saving power.
When supplying dynamically the 555, it's necessary to analyze its stabilization at startup. In fact, depending on the technology, there is a minimum time to wait after power on, to get the correct output frequency. For bipolar NE555 this time is very short, around 150 µs, while LMC 555, in CMOS technology, needs around 2 ms to stabilize.
In addition to this work on power optimization, I made an algorithm of absence, to put the capacitive sensor in sleep mode if no one is around the sensor. It needs a threshold to know the presence and it can be obtained by an initial setup. To evaluate this threshold and the possible frequency drift along the time I performed many tests. One of them, concerning 6 h of continuous measurement during the night showed that the frequency increases by about 10 Hz every hour in experiment conditions.
During this work, I also made experiments with another front-end for comparison. The other front-end works by modulating the phase of a carrier and demodulating it, resulting less sensitive to noise and drift. Regarding it, I made tests on the stability and sensitivity. In details, I have analyzed the drift on the output value along several hours, with some modification on the board to optimize stability.
Then, I have done many investigations about the stabilization time, voltage and current, in order to estimate a possible duty-cycling or any further power saving optimization.
Analyzing the results related to the 555-based front end, there is a significant energy save, after the optimization. Before this work the current consumption was 14 mA during the measurement time of 300 ms. Now the consumption is 650 µA for the whole system based on LMC555 (CMOS) and 1.1 mA for the one based on NE555 (bipolar). This is a significant power reduction, more than 90 %.
With the optimized system, measurements are done only when a movement is detected, so the actual consumed power in a realistic scenario can be even lower.
In details the consumption of the movement sensor is constant around 2.4 µA. This has a very low impact on total consumption. Then the 555 has a consumption of 130 µA - LMC555 and 3.28 mA - NE555, but it's OFF for much time. The microcontroller consumes different current in different modes but, considering the average current, it has a consumption of 200 nA in sleep mode and 630 µA during

a measurement period of 300 ms. The total consumption arises from the duty-cycling of the measurements: in a single measurement period the idle time, while the 555 is supplied is around 45 ms. So,the calculated duty cycle of operating mode is $DC = 15\%$.
I tried to estimate the possible battery duration to supply the system. It's possible to estimate the *maximum always ON time* and the *maximum sleep time* for a system supplied with a power-bank of 10 Ah. They are respectively more than 3 years and hundreds of year. Of course the battery would terminate before for chemical processes.
These results are significant because this capacitive sensor could work for more than one year without recharging the battery (neglecting the self-discharge).
In the future, possible improvements can be done on the system, also optimizing others front end for low power consumption.

Chapter 1

Introduction

1.1 Indoor human localization

Nowadays the houses are becoming "smart", where thanks to the Internet of things (IoT), the objects are connected and communicate among themselves, to create facilities to humans. In such context, the human localization is a very important topic because can provide even more advantages for house automation, like for example lighting and warming and for security purposes. But a possible problem could be the intrusiveness of these kind of sensors. In fact, having them on clothes or as accessories can be very annoying so it's important to develop a system of localization sensors totally tag-less [3].
This means that the sensor can localize a human without any paired device that the person should wear.
Then the system should respect the privacy of people. In these years, have been many instances about the data leakage and privacy of the user is now a strict requirement. Ideally the system has to be passive, so any user can be use it without specific interaction or knowledge about it.
Therefore, this kind of system for human indoor localization must be transparent to people as explained in [3] and it can be recommended to the observation of the safety of elderly.
Another concern, that is the focus of this thesis work is the consumed power of a sensor.
If it consumes very little power, the batteries should be recharged in a long period and it allows to let it even more transparent to the users [4].
There exist many kind of sensors like these but they rely on different technologies and have pros and cons.
Radio frequency: the human body adsorbs radio signals. So, sending a signal and receiving the returning intensity of the same signal can give indications about the

position of the people and their movements.
In [5] the authors use a method with radio frequency. It can be very precise but the problem is to handle an initial phase of setup for each environment/room where it want be used. This initialization is needed to map the signal response without any human. Then when using it actually this initial response is compared to the one currently measuring.
Then there are *pyroelectric sensors (PIR)* [3]. They are cheap but they require a strong computation effort. This kind of sensors can provide additional information, making them very suitable for extended applications, but they have a narrow detection field and many of them are needed. In addition they can be easily subjected to errors due to heat/light sources and they can also cause many privacy concerns [4].
There is also a large amount of *pressure* sensor cells [3]. They are good for identifying humans with their almost unique weight, but they have to be installed under the floor, so it's not so easy and affordable to handle. They are also transparent to users and are very spread [6].
Other methods comprehend *ultrasound* sensors, as described in [3]. They employ devices that, thanks to the known wavelength, can calculate the exact position of a human. They are very precise but quite expensive and one of them can cover just a small portion of space, also mentioning their possible harm for hearing and pets.
Moreover, various techniques with *cameras* and *Infrared thermal cameras*) require a very heavy computational effort and are not very well-suited neither for privacy nor for energy optimization. In addition also the high cost is a drawback.
Finally, techniques based on *RFID, Bluetooth and Wi-Fi* suffer from interference but, mostly, users must bring a tag, that they can forget making it unusable.
We focus on *capacitive sensors*, that are very suitable for mid-range detection and tracking, are tag-less, cheap, privacy aware and can be optimized for low power. The broader research project aims to optimize capacitive sensing in terms of noise rejection, power optimization, accuracy and range increase.
In the field of capacitive sensing a lot of work has been done by researchers, exploring the behaviour of indoor electric fields and their response to human movements. Some of these exploration led to the movement sensor Electric potential sensor (*EPS*) as reported mainly in [7] and other papers from Prance et al. This kind of sensors can operate at distances of several meters (up to 3 m so far), exploiting the slow variation of environmental electric fields. In [8] this kind of sensor has been also used to detect the breath of a human at a distance of 1.5 m.
This exploration is really interesting, although it is very prone to noise. It exploits the periodicity of the breath to better measure it, but unpredictable human movements are more difficult to detect in a single shot. In our team in Politecnico di

Torino it has been done a huge work during these years in developing and exploring the capacitive sensing and this thesis is part of this project.

1.2 Capacitive sensing

One of the techniques reported above is the *Capacitive sensing* that is widely used in most recent smartphones and computers to employ a touch screen interface. It is a relatively simple technique because it allows to sense the proximity or a touch as a capacitance, that is easy to measure. In this case the capacitance is converted into a frequency thanks to an astable vibrator with fixed resistances. The human body is a conductive element, that couples with the environment so these sensors can be used to track the position of a person compared to them.

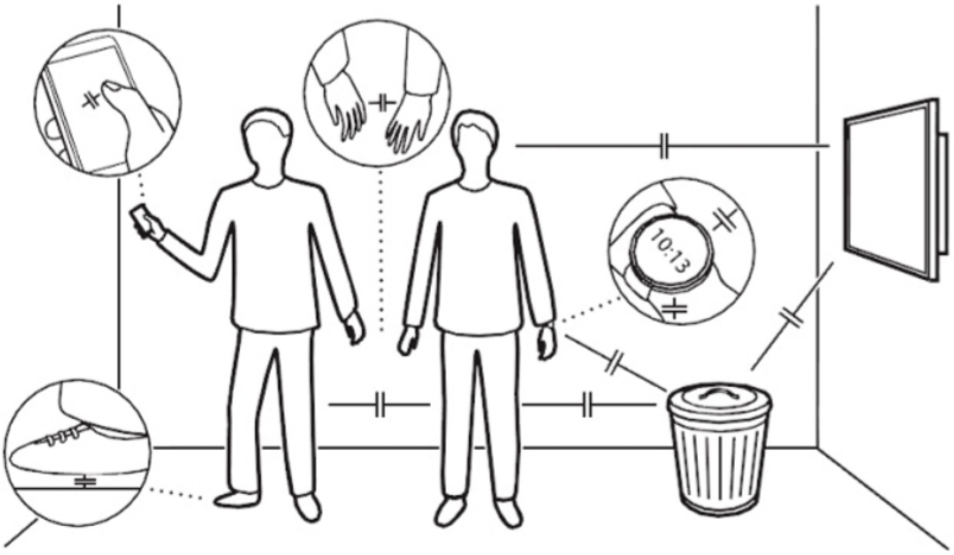


Figure 1.1. Capacitive coupling between human body and the environment from [9]

As shown in Figure 1.1 and referring to [9] the human body forms capacitances with each conductive object around him. In fact the human body acts like one conductive plate of the ideal parallel plane face capacitor, with air as dielectric. In this context is really important the role of the common ground. It is a potential shared for the two "plates" and can be the floor or building potential. In some

cases it may be the body itself.
So, when two different conductive elements are separated by a dielectric a capacitance exists. Then, if they are also at different potential an electric field exists.
From theory is known that the *self capacitance* **C** is the *charge* **Q** hold by the capacitor with a *voltage* **V**:

$$C = \frac{Q}{V} \tag{1.1}$$

As also mentioned in [1] the capacitance for a classical planar capacitor is depending on the area of the two faces A, on the distance d and on the dielectric constant ϵ_0 and k, considering $A >> d$.

$$C = \frac{\epsilon_0 k A}{d} \tag{1.2}$$

For the sensor this formula is not valid but there is still a dependency between the capacitance and the distance from the plate. An approximated formula for the sensor is:

$$C \sim \frac{k \cdot A}{d^{2-3}} \tag{1.3}$$

Referring to Figure 1.2 from [10], the whole capacitance is mainly due to these components:

1. C_{sb}, between human body and the sensor plate
2. C_{bg}, between human body and ground floor
3. C_{sg}, between the sensor plate and ground floor
4. C_{se}, between the sensor plate and other objects in the environment

Considering that also, environmental humidity and temperature affects the air dielectric and these capacitances, the C_{sb} is function of the distance between the sensor and the human and is the one of interest for measurements.

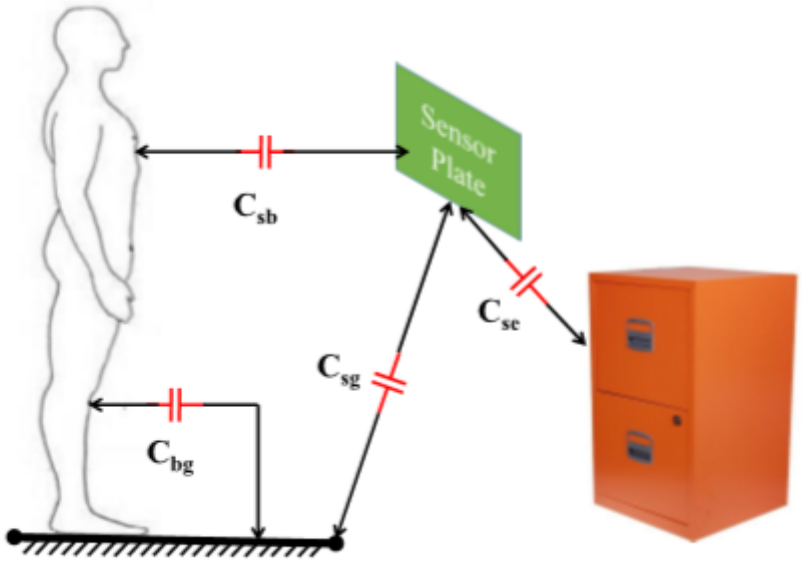


Figure 1.2. Capacitive coupling between sensor and objects(among them human body) from [10]

1.2.1 Working modes for capacitive sensing

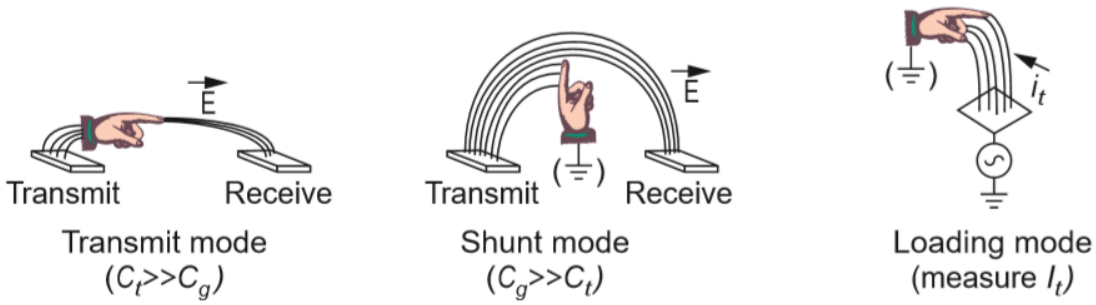


Figure 1.3. Different capacitive sensing modes from [11]

Considering the system shown in picture 1.3 and described in [11] there are mainly three working modes for the capacitive sensing. Two of them involve the use of two sensors, one for receiving and one for transmitting.

Transmit mode allows the human body to become part of the transmitter, so that the received signal is increased by a value that depends on the body proximity. This mode is possible for very near distance between the body and the transmitter, due to the prevalence of the capacitance between body and plate compared to the body ground capacitance.
Shunt mode happens when the body is not very near to the plates and the body ground capacitance prevails. In this case the received signal is decreased with the proximity of the body.
These two methods are very powerful for robustness and quantity of information. For example the transmit mode is very good at tracking movements when the user is touching the sensor. But referring to the discussion in the previous paragraph, this would not be an easy and tag-less method.
Loading mode is less complex because it needs just a single plate. It allows to measure the induced current in the plate. Both body and plate are referred to a ground, that is a shared potential.[11]

1.3 Previous work

This project involved many previous thesis works. All them investigate different aspect of the indoor human localization and develop methods and optimization for different front-ends. Some of them are [6] [4] [12] [13].
Their work has been continued, perfecting some aspects and investigating new.
The researchers of Sensor team in Politecnico di Torino performed many experiments with capacitive sensors. One of them, reported in thesis([6] [4] [12] [13]) and [1] article, was to monitor the movements of a human in a 3x3 m room with the following setup of front-end and base station as reported in Figure 1.4: *555* as astable oscillator, connected to fixed resistors and a 16x16 cm metal *plate* as capacitance. It generates the frequency in relation to the distance from the human. This frequency is measured by the microcontroller (MCU) *ATmega328P* on an *ARDUINO UNO*.
Finally the data is transmitted to the base station when is elaborated through an *XBEE* radio module. All this chain allows to conduct experiments easily but a drawback, due to the kind of front-end based on 555 is the poor noise rejection. So, the noise filtering and data clearing has to be made by software.
Looking in details at the 555 circuit setup it has been used in stable mode, as shown in Figure 1.5 with a metal plate that couples with human body, two fixed resistors and the output connected to an input pin of the microcontroller. The 555 provides a square wave depending on the resistances and the capacitance. The first are fixed so the frequency depends just on the capacitance, that depends in turn on the distance between the plate and the human.
The formula that gives the 555 frequency is

$$f_{555} = \frac{k}{(R_A + 2R_B) \cdot C} \tag{1.4}$$

with $k = 1.44$ for the two used 555 (NE555 and LMC555). So, from previous work, there is a working measurement system which I tried to optimize for low power consumption.
I didn't started from scratch with the capacitive sensor, but I used as reference for my optimization the code already written and tested for a long time. The frequency measurement system is based on [12] and [13]. Nothing of their system of *Hybrid frequency measurement* has been changed.

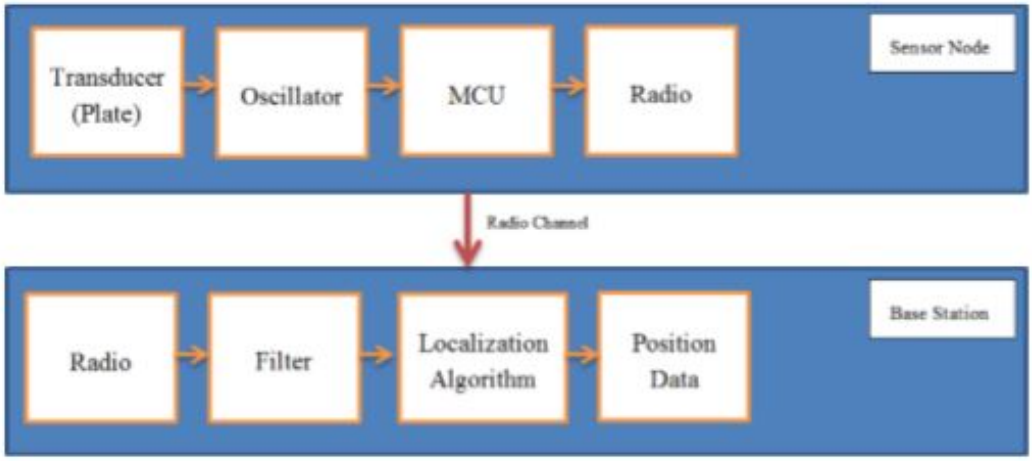


Figure 1.4. Measurement chain of 3x3 m room experiment from [1]

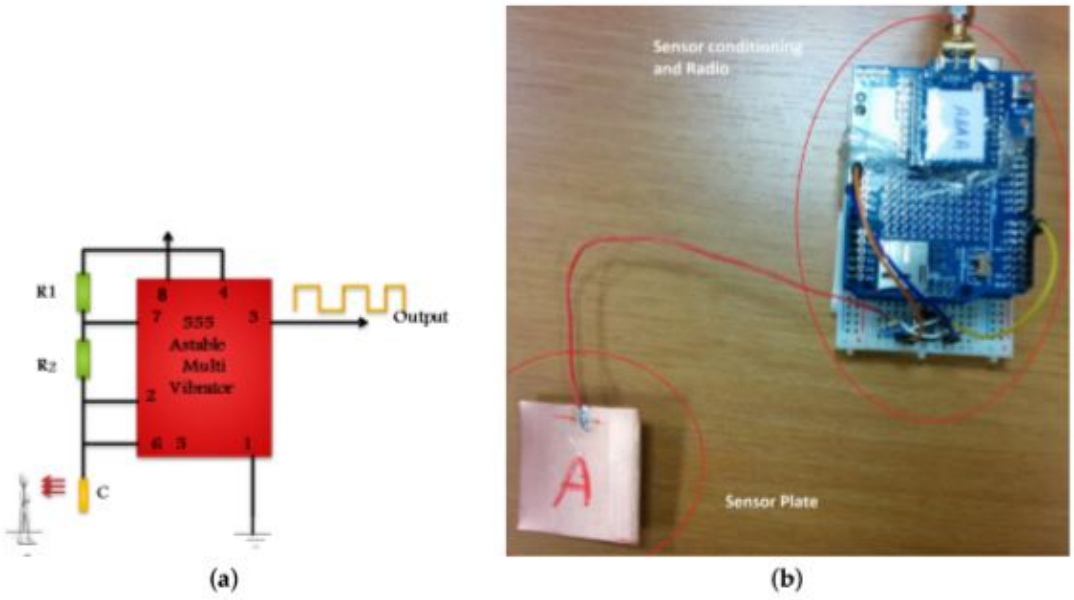


Figure 1.5. a. 555 schematic b. setup for experiment described in [1]

1.4 Project

The main goal of this thesis project is to apply some techniques and optimization for low power consumption to the previous work in this research project.
The starting point was the implementation of a movement sensor, according to [2] schemes and discussions. The purpose of this sensor is to wake up the capacitive sensor when it senses movements nearing. The capacitive sensor is the one described in [1] with the ATmega328p microcontroller and 555 circuit.
Then I performed the optimization of the capacitive sensor, using sleep modes and reducing, where possible, the consumption of the whole system. During all the process, particular attention has been dedicated to the functionality of the system, preserving the correct operation.
The final system works in this way: the movement sensor, that has a very low consumption, continuously checks the presence of a human movement and the capacitive sensor is in a deep sleep mode. When this happens, the capacitive sensor starts to measure the distance from the human and continues measuring until the human leaves the sensing range. Then it comes back in the low power mode and the movement sensor starts again to look for movements. In the figure 1.6 is represented the interaction between the two sensors.

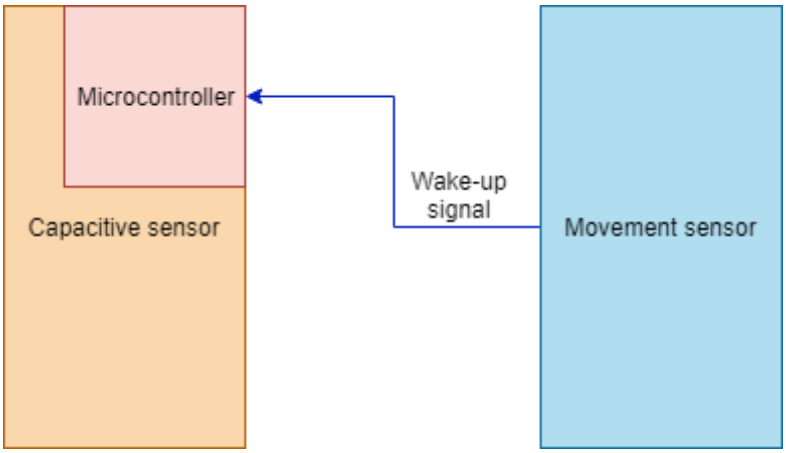


Figure 1.6. Block scheme of the two sensors and their interaction

Chapter 2

Movement sensor

2.1 Operating principle

Passive electric field sensing is a very interesting way to localize humans in a space, without any need of amplification stages. This mode is suitable for a very large range of applications, without creating its own electric field, but just sensing any change in the ambient electric fields. It also consumes less power than the capacitive sensor (based on 555 front-end) so it can be optimal for battery based systems that have to last longer.
As discussed in [2] the passive sensing relies on the capacitive coupling between the plate and a charged object that moves, as a human. In facts, objects can be charged, due to triboelectric and electrostatic effects. This charge, of course, interacts with the present electric fields, modifying them. The human body, as other charged objects, modifies the electric fields around him and is hence possible to detect its presence with capacitive sensors.
According to passive sensing theory and experiments, the range should be higher than an active sensing. But, as a drawback is possible to be shielded or disturbed easily. In fact, any interacting electric field can modify the induced current and so the measurement. This has to be considered while designing the further filter and comparator circuits.
Then, the sensor explained in [2] can only detect changes in electric fields: when a charged object interacts with it, a very small current is induced in the plate. This tiny current is translated into a significant voltage thanks to high impedance and can be read as a parameter to detect movements.
To modify the electric fields there are two possibilities: a charged object can move or a moving charge could be inside a still object.
The human body forms a capacitance with the plate and the quantity that is sensed is the voltage difference V created by the inducted current I. In fact from

the studies over electric fields we get the dependency between distance d and capacitance C. Only for planar capacitors that have $A >> d$ this relation is valid:

$$f = \frac{\epsilon_0 \epsilon_r A}{d} \tag{2.1}$$

$$C = \frac{Q}{U} \tag{2.2}$$

and deriving the charge over time we get current

$$I = \frac{\partial Q}{\partial t} \tag{2.3}$$

we obtain the dependency

$$I\partial t = \partial(U\epsilon_0\epsilon_r \frac{A}{d}) \tag{2.4}$$

making proportional the distance of two conductive object to a voltage difference.
For the capacitive sensors interacting with objects, that relation is not valid, but we know the dependency:

$$C \sim \frac{k \cdot A}{d^{2-3}} \tag{2.5}$$

with a plate significantly smaller than the distance.
A possible drawback is that the charge has to move, so if a human is still in front of the sensor it is not recognized. But in the real application, what is wanted is the movement of human, so this is not a problem in this thesis work.
Wilmsdorff et al. also show that there are a lot of potential applications with this technique as gesture recognition.
One of these is implementing the idea from [14] Cohn et al. with a circuit with the plate, a high resistance, comparable to the high impedance of the voltage follower stage. In [2] the output signal is sent to an ADC, but this is not this project purpose, because an acquisition system would consume too much power.
The idea behind this design is to get a similar effect to [14] where the sensor can be used to wake up another front-end. But, compared to Cohn et al. there isn't any amplification in the developed system.
So the final circuit in this project sensor will need a filter for the 50 Hz network noise and a comparator to obtain a digital signal every time a charged object moves near the plate. Then, this digital signal is sent to the microcontroller of the capacitive sensor, for the actual distance measurement.
The developed circuit is in fig 2.1. The main idea behind the movement sensor of this project is to employ it, due to its low power consumption, to look for

human movement, but not measuring anything else with it, leaving that duty to the capacitive sensor.

2.2 Implementation

In this section will be discussed the practical implementation of the circuit mentioned before. All the various optimizations have been made in order to achieve the lowest possible power consumption.

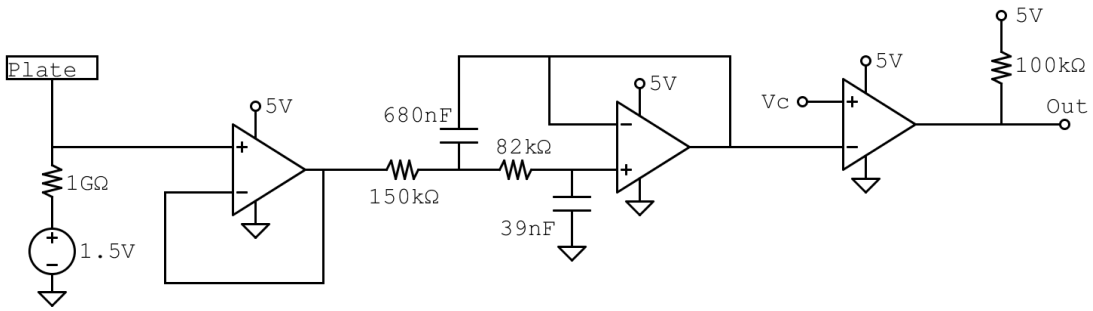


Figure 2.1. Schematic of the global movement sensor

2.2.1 Sensor

The first block implemented is the sensor itself. It is composed mainly by a RC, with a metallic plate, that is like one plate of the capacitance between it and the human and a big resistance, 1 GΩ.
The main motivation for this value of resistor is the desire of measuring a very little current coming from the plate, due to capacitive coupling with a charged object. So, to get a significant voltage change, a big resistance is needed. Then, this resistance is also in the GΩ range to be similar to the input impedance of the voltage follower OP-AMP that provides the signal to the filter.
There is also a bias of 1.5 V to give an offset to the signal and avoid any clipping to 0 V or V_{DD}. This voltage is provided with a low dropout voltage regulator.
The specific model used is the LP2951 from Texas Instruments. It is used in combination with two resistances of 56 kΩ and 270 kΩ to get the value of 1.5 V from the 5 V of the power supply. The schematic is reported in Figure 2.2

Then, in addition to that two capacitors are used as reported in the datasheet of the regulator [15], 1 µF and 2.2 µF.
The voltage follower is done thanks to the MCP6043 from Microchip [16] (schematic in fig. 2.3) and its power supply is a single supply between 0-5 V despite of what present in [2] (0-3 V). This supply is correct compared to the datasheet [16] and

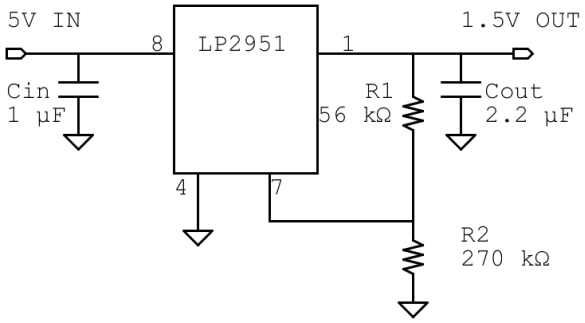


Figure 2.2. Schematic of the implemented voltage regulator

works well without high impact on the consumption.
This component is used for its low power consumption. The output signal from it needs to be filtered as can be seen in the next paragraph.

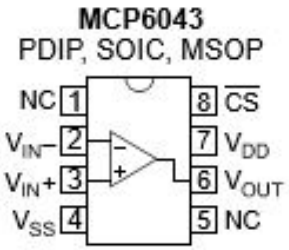


Figure 2.3. Pin schematic for MCP6043 from [16]

2.2.2 Filter

The filter has been implemented in order to remove the electrical noise in the range of 50 Hz, so it was chosen a Sallen-Key low-pass filter. This noise is due to the electrical network so has been filtered out.
The frequency of interest are related to the human movement, so the range is in the mHz, surely under 1 Hz. The cut frequency has been chosen with these considerations and the implemented passive components were: two resistances, 150 kΩ and 82 kΩ and two capacitors, 680 nF and 39 nF. The simulation on SPICE has been done and reported in Figure 2.4
The filter is made with the same low power amplifier used in the sensor itself, the

MCP6043. It provides the correct behavior with a very little current consumption and it has a single supply that is easier to manage.
The supply is 0-5 V and the signal received from the sensor is correctly filtered because it has a bias of 1.5 V, avoiding any possible clipping of the signal due to the dynamics. The filtered and unfiltered signal are visible in the images 2.5 and 2.6.

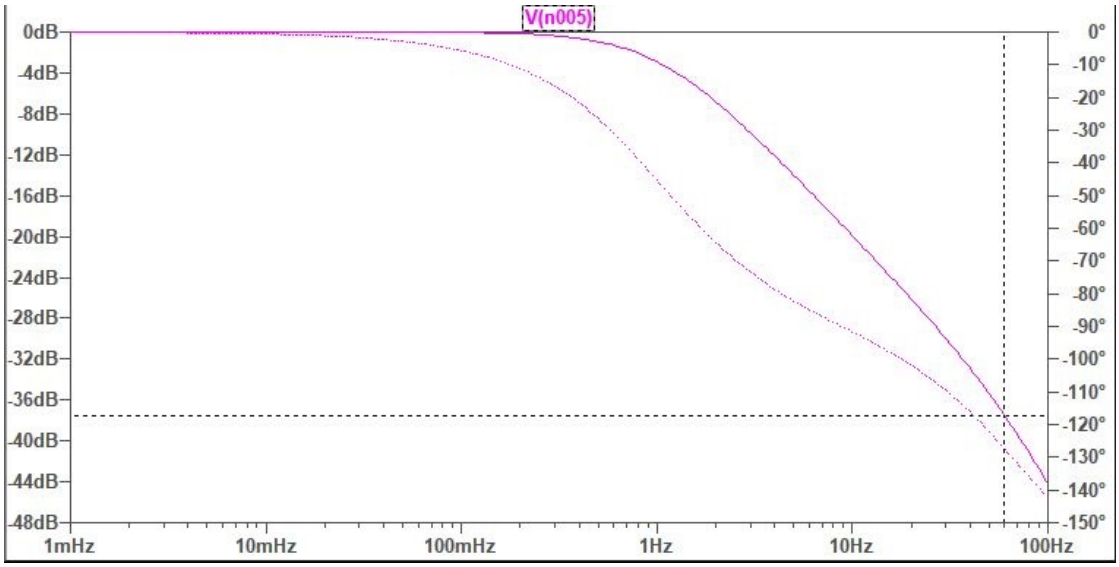


Figure 2.4. Frequency response of the low-pass filter. @ 50 Hz the attenuation is −37 dB

2.2.3 Comparator

At the end of the whole sensor, a threshold comparator is present, with the MCP6043, as operational amplifier out of linearity. Its aim is to provide a digital signal when the analog signal from the sensor (already filtered) is over a certain voltage.
In this way, when a movement is performed in front of the sensor, the output signal is led from 5 V to 0 V in order to wake up the micro-controller.
The value of the threshold V_c has been chosen as a fixed value, by experimentally trial. It was a voltage of 1.78 V, made with a resistive voltage divider.
We noticed that this reference value is strongly dependent on the surrounding environment, so the system has to be properly tuned, in order to get the correct behaviour of the comparator. In fact, the signal from the filter crosses this value just when a real movement is performed close to the sensor. To guarantee the

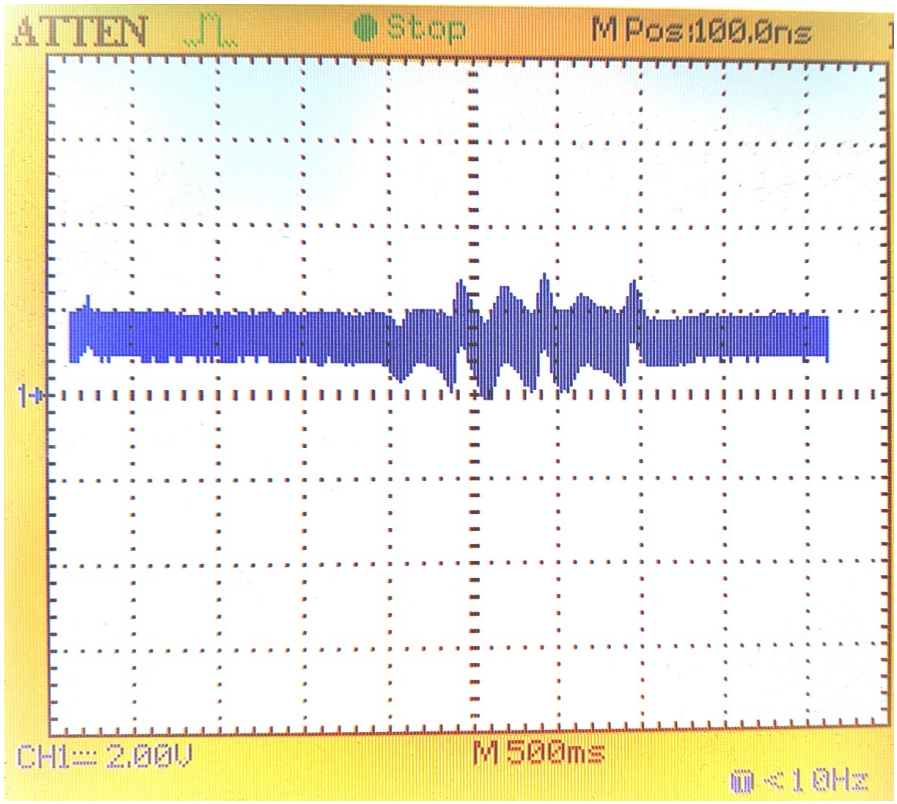


Figure 2.5. Unfiltered signal with movement. Affected by 50 Hz noise

expected operation, it has been extensively tested.
Finally there is a pull-up resistance, that keeps the signal to 5 V without any movement. The output signal from comparator, that becomes the external interrupt pin INT0 and the signal from filter are visible in the Figure 2.7.

2.2.4 Complete circuit

The whole circuit detects movements and provides a digital signal to the output. So, based on the theory from [2], the circuit has a correct behavior.
I prototyped it on a breadboard and I did several tests to guarantee a correct operation.
Tuning properly the threshold voltage of the comparator allows the sensor to recognize just real movements.
We also tested the condition when a person stands in front of it, not moving; as expected it doesn't detect any movement. So, moving on to implement the code

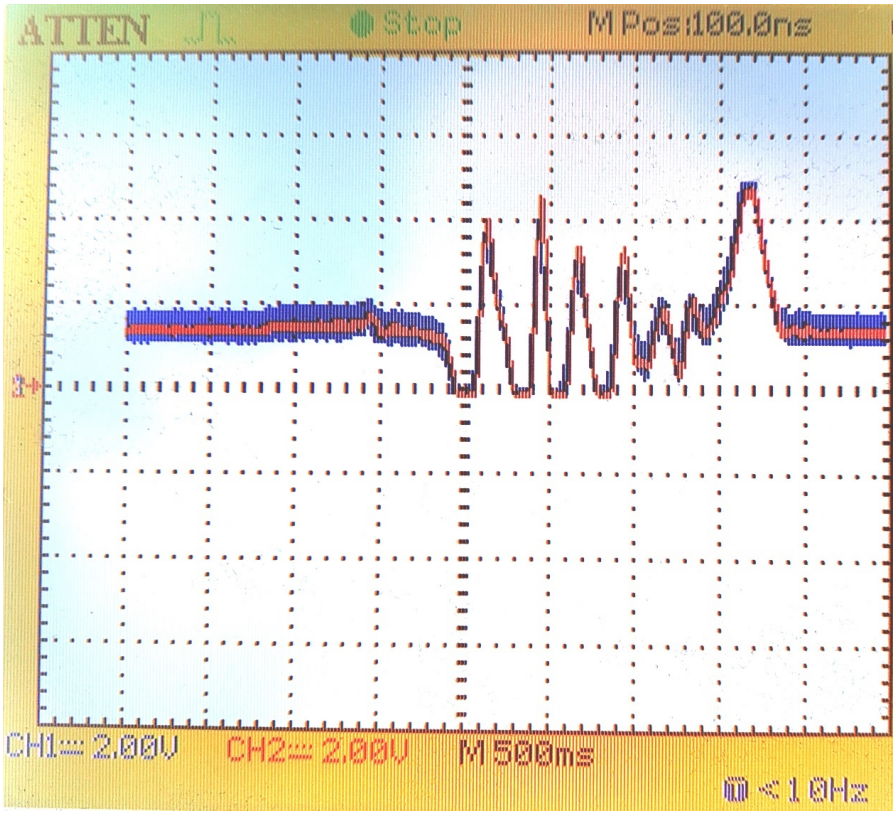


Figure 2.6. CH1: Unfiltered signal. CH2: Filtered signal. In the middle a movement is present

for the micro-controller, the aim was to use this sensor to wake up the capacitive sensor.

2.2.5 Noise

During the tests, we discovered that the environment can hugely influence the movement sensor behavior.
For instance, having it near and parallel to a wall, maybe with lots of electric wires in it, can cause a change in the noise levels and a different offset after the first op-amp.
In Figure 2.8 is shown the significant increase of the threshold level. In different experiment conditions it was 1.7 V, in others, like near to walls it increase up to 2.6 V.
In the case where, the wall was 3 m far from the sensor, the offset was 1.5 V as

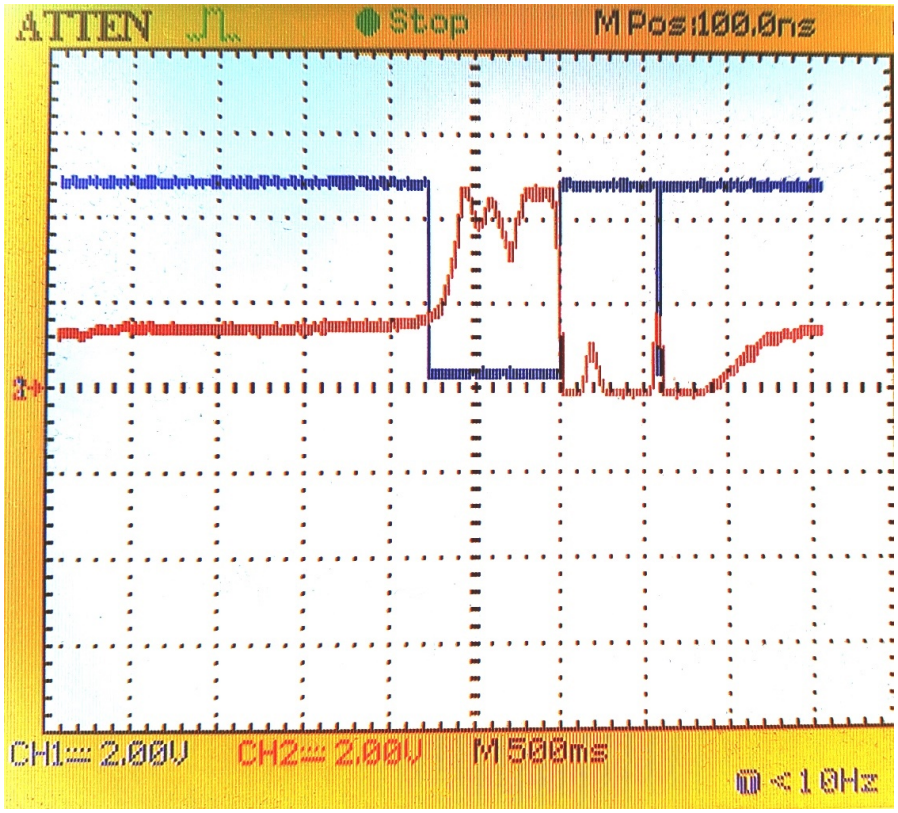


Figure 2.7. Filtered signal and comparator output in movement sensor. CH1: Comparator. CH2: Filtered signal

setup. Instead, with a wall 1 m behind it, an increase of 0.7 V of the offset has been measured. This is a huge variation, so the threshold of the comparator had to be changed accordingly.

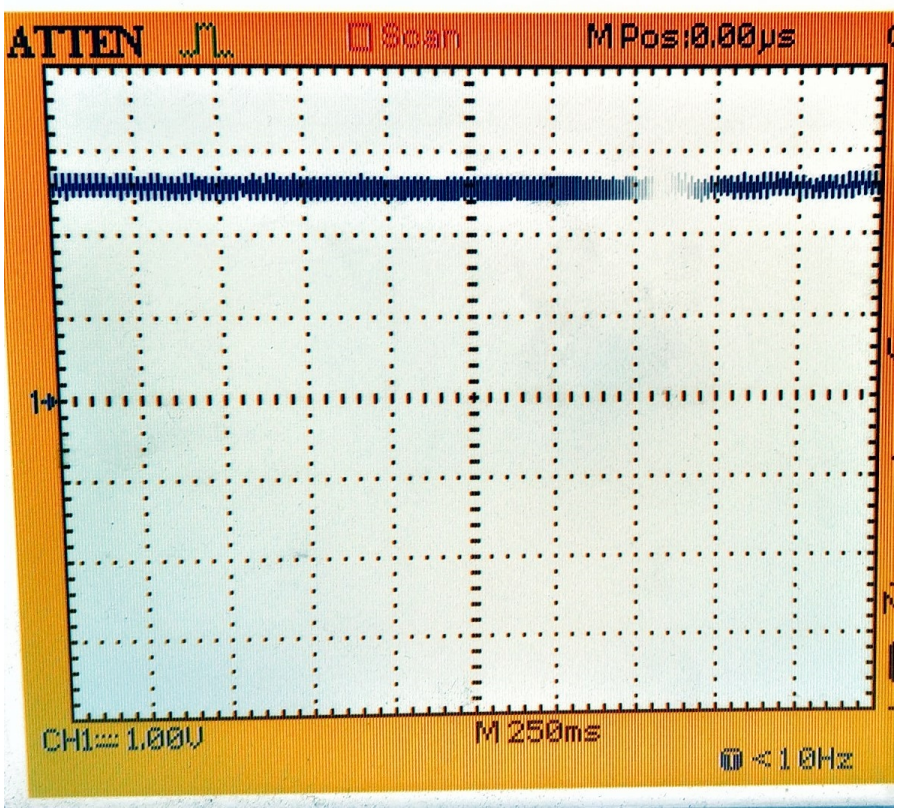


Figure 2.8. Offset of signal increased over 2 V, due to environment

2.3 Power saving considerations for movement sensor

The operational amplifiers used in the movement sensor circuit have one pin that is the chip select $\overline{CS}$ as reported in their datasheet [16]. So it could be useful to exploit it, in order to further reduce the current consumed along one measurement. In fact the movement sensor will be OFF since it detects a movement, until the capacitance of the capacitive sensor, based on the 555, can be considered exhausted. This can be a long time, while the movement sensor would consume uselessly. So, a possible implementation, could be to use a pin of the microcontroller to change the state of the op-amps dynamically.
At first, the stabilization time after the chip-select is enabled has to be considered. During this time the movement sensor would be blind and this would be certainly a drawback.

As it can be seen in the Figure 2.9, the settling is quite long for power saving considerations, around 30 ms. A possible solution to avoid any additional consumption could be to disable the external interrupts masking after two overflows of the watchdog timer (WDT) (16 ms of period) but the blindness is unavoidable.

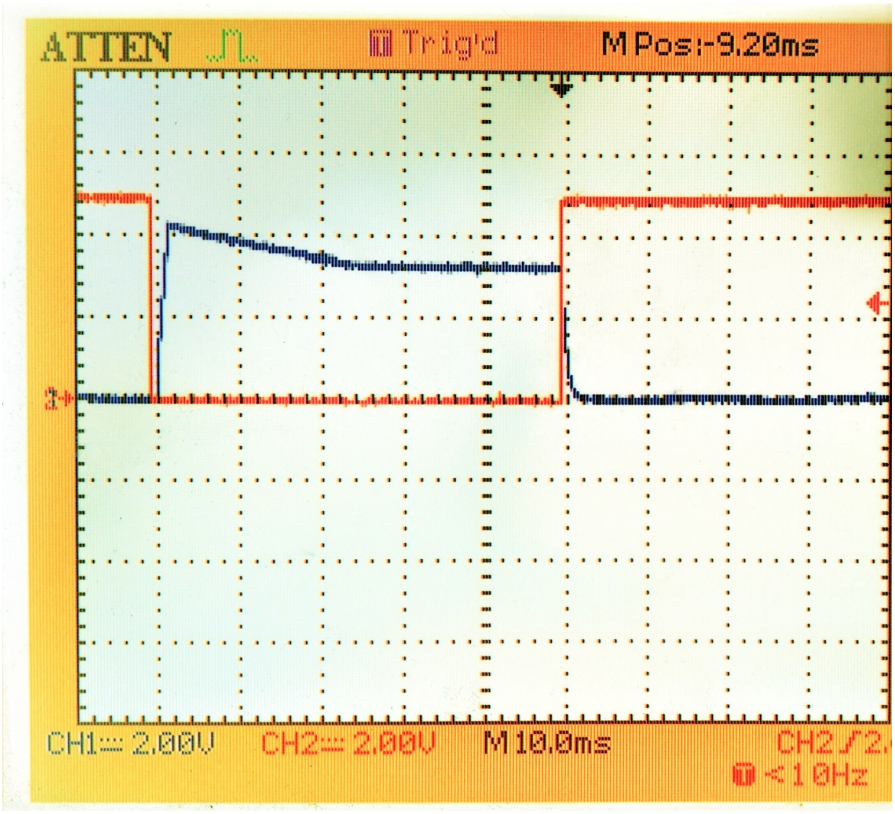


Figure 2.9. Settling time after chip-select enabling (CH2, falling edge) of the output signal from MCP(CH1), configured as a voltage follower with a fixed input voltage of 3.3 V

Then a second consideration must be done. Looking at the current consumption, is it convenient using this feature ?
Without the use of $\overline{CS}$ the useless current consumed during each measurement period of 300 ms is 2.4 µA.
Enabling the chip-select, the output will be zero and the current that flows would be around 200 nA (measured, datasheet [16] reports just 50 nA). So the difference, in terms of current is 2.2 µA for each measurement period.
This leads to a saved charge of 183 nAh, for each measurement. Comparing these results to the whole system consumption, as we will see in chapter 5, it appears

that the save is very poor, compared to the total consumption.
So, in a trade off between the blindness and the save of energy, the drawback is more affecting the behaviour and this solution should not be used. But the $\overline{CS}$ can be useful for others purposes, as avoiding repeated consecutive measurements, as reported in chapter 6.

2.4 Sensing range

The movement sensor has been tested, in order to get the maximum range in which it can detect the presence of a moving charge. This range strongly depends on the environment, due to capacitive coupling between the plate and the surrounding charged objects but a rough idea can be obtained.
Examples of different ranges are shown in Figure 2.10. Placing the sensor in the center of a room, with conductive objects farther than 2 m and using a metallic transducer of 8x8 cm, led to a capture range of 1.5 m in each direction. In fact, a positive fact is that even perpendicularly to the plate, it could still detect a person. Then, trying it in other contexts the results were different. Placing it in a metallic cupboard, led to a very tiny range, as expected, of 30 cm. This happens due to the high coupling between the plate and the metal. So in a final implementation, is obvious to avoid to place the sensor in a metallic structure that can shield it.
Then, positioning it near walls that contain electric cables, reduces the range at nearly 1 m. This is a possible problem, because is very probable that the sensor will be placed on a wall. But, looking carefully at the signal after the filter 2.1, is a problem caused by the increase of the level of it, as mentioned in 2.2.5. So an initial tuning, or better, a dynamic set of the threshold of the final comparator can provide better results in the maximum detection range.
During the tests, the 16x16 cm is also been used and the results were the same, the increased plate dimension does not increase the detection range of the sensor. As for each other passive capacitive sensing it is limited by environmental noise and at a distance farther than 10 times its size [10] the variation due to the presence of a human is lower than 0.01%. This fact clearly shows that at such a distance the presence can be confused easily with noise.

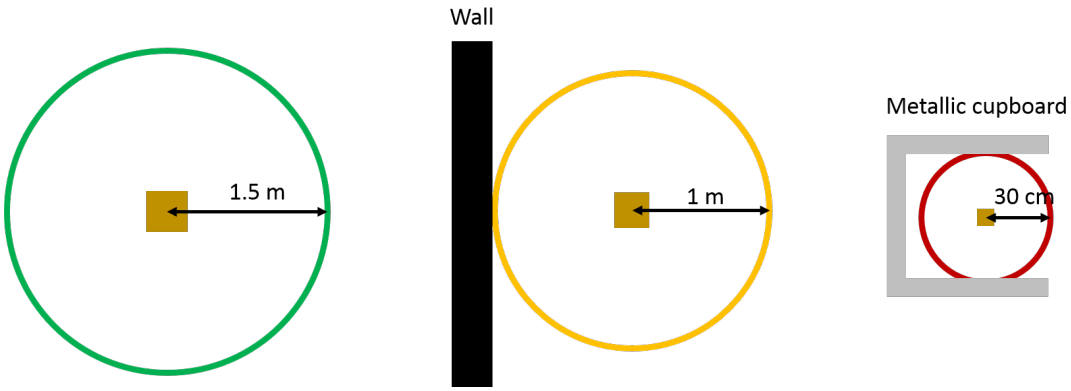


Figure 2.10. Range of the movement sensor, tested with 8x8 cm plate, threshold of comparator 2 V. From left: free air, 1 m near to wall, inside a metallic cupboard(shielded)

Chapter 3

Capacitive sensor front-end 555-based

3.1 Operating principle

The theory behind this sensor is explained in chapter 1 in the paragraph *Previous work*. In this section, the sleep modes and every possible optimization related to the microcontroller and the 555 circuit will be deeply investigated.
The main aim is to have the circuit working as before (no loose of frequency resolution) but having the lowest possible consumed current.
So, the principle is to use the sleep controls. But, in addition, it's necessary to provide the control and supply signals with right timing to analogical blocks, in order to have them working.
All this work is related to the 555 front-end but in a further development, the sensor could be implemented and used with other front-ends and microcontrollers as well.

3.1.1 ATmega328P power management

According to section 14.2 of [17] microcontroller datasheet these are the possible sleep modes and the wake up sources (Figure 3.1):

14.2 Sleep Modes

The following table shows the different sleep modes, BOD disable ability, and their wake-up sources.

Table 14-1. Active Clock Domains and Wake-Up Sources in the Different Sleep Modes

Sleep Mode	Active Clock Domains					Oscillators		Wake-Up Sources							
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT and PCINT	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			Yes	Yes	Yes	Yes	Yes(2)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
ADC Noise Reduction				Yes	Yes	Yes	Yes(2)	Yes(3)	Yes	Yes(2)	Yes	Yes	Yes		
Power-Down								Yes(3)	Yes				Yes		Yes
Power-Save					Yes		Yes(2)	Yes(3)	Yes	Yes			Yes		Yes
Standby(1)						Yes		Yes(3)	Yes				Yes		Yes
Extended Standby					Yes(2)	Yes	Yes(2)	Yes(3)	Yes	Yes			Yes		Yes

Figure 3.1. ATmega328P Sleep modes from [17]

The measured current of the microcontroller are:

- Power down, without WDT: 200 nA
- Power down, with WDT: 6 µA
- Power save, without WDT: 1.3 µA
- Idle, with Timer0 and Timer1: 4.5 µA

In this project measurements, the current is evaluated with an ammeter put in series to the ATmega328P's supply (VCC pin). The microcontroller has been put on a breadboard and connect with the minimum necessary pins to the Arduino UNO board.
The resolution of the ammeter is 100 nA.
The found results match with the datasheet graphs, as notable in figures 3.2 and 3.3, only the "Power Save" current is higher because this setup is different from the datasheet one, due to the presence of an external 16 MHz oscillator.

Power-down Supply Current

Figure 34-11. ATmega328P: Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

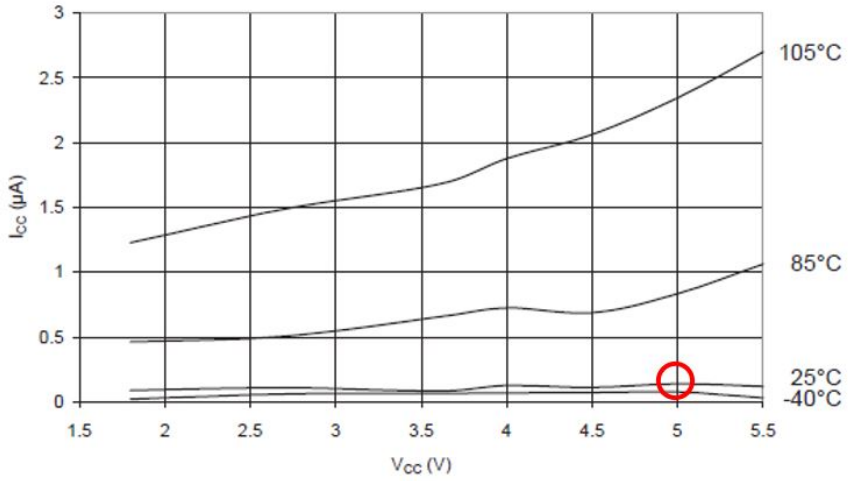


Figure 3.2. Graph of current vs. supply voltage on datasheet, no WDT from [17]

Figure 34-12. ATmega328P: Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)

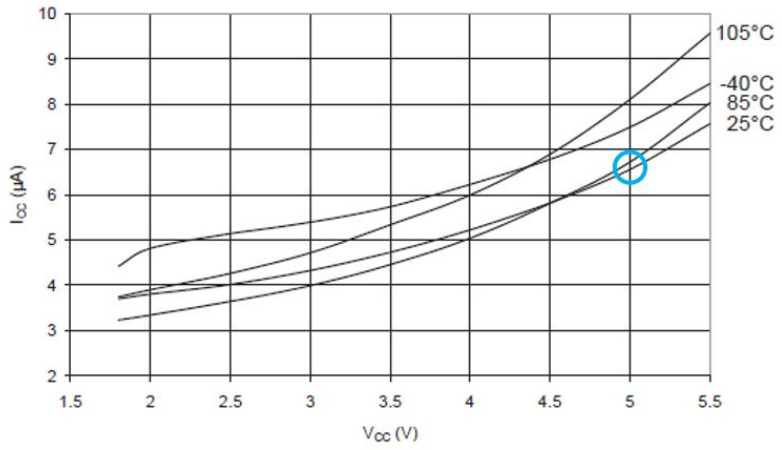


Figure 3.3. Graph of current vs. supply voltage on datasheet, with WDT from [17]

The capacitive sensor will be off until an external interrupt comes, as explained before, so two different operation modes can be distinguished:

- Deep sleep mode: The movement sensor is ON, waiting for a human presence, the microcontroller is in Power Down mode and the 555 is not supplied.
- Active mode: The capacitive sensor does measurements until the human is still present in the range, the global measurement is made of periods of 300 ms with a measuring windows of 43 ms (while the 555 is supplied). In the remaining part of the period the microcontroller is in sleep mode.

3.1.2 Power management settings

In the setup of various sleep modes, some functions have been instantiated in order to recall a certain mode easily. As described before three of them have been used in this project: Power Down, Power Save and Idle.
Each of these mode, when activated, allow the MCU to enter in a sleep mode when the code finds the command *sleep_cpu()*.
Then the execution of the code is suspended and only some features of the microcontroller are enabled as reported in 3.1. In that table are also present the possible wake-up sources. A duty of the designer is to assure that after each sleep_cpu() there is always an event with the correct timing that wakes up correctly the MCU.
To activate a specific sleep mode is necessary to run this routine:

1. Set the SM[2:0] bits of the Sleep Mode Control Register (SMCR) to get the chosen mode
2. Disable interrupts
3. Set the SE of SMCR register bit to logic one in order to enable the sleep mode
4. Re-enable interrupts

The SMCR register, with the bit SM[2:0] allows to select between the following modes 3.1 with these coding (reported from [17]):

The library $<$ *avr/sleep.h* $>$ helps the designer with two functions to do this: the first is *set_sleep_mode(MODE)*, that set the SM[2:0] with the name of the mode inside the parenthesis. The second is *sleep_enable()* that sets the SE bit.
A remarkable note must be done on deselecting the SE. In fact, to avoid that the MCU enters in a sleep mode in undesired conditions, the programmer should always clear the SE bit after every wake-up.
The handling of the supply of the 555 is made with an output pin, raised to logic

SM[2:0]	Sleep Mode
000	Idle
001	ADC Noise Reduction
010	Power-down
011	Power-save
100	Reserved
101	Reserved
110	Standby
111	Extended Standby

Table 3.1. Sleep mode bits of SMCR from [17]

one to switch ON the astable oscillator and pulled to logic zero to switch it OFF. In detail the pin is the PIN B1 of the microcontroller. The output pins can provide a stable voltage of 5 V, with a maximum current of 20 mA. These requirements are sufficient to supply the 555 circuit.
Reading the datasheet of the MCU, helps to find other optimization for the consumed current. The most important setting, for this purpose, is the Power Reduction Register (PRR) that allows to halt the clock to selected peripherals. It help to reduce the current during active and idle mode, because in the other sleep modes the clock is already stopped.
In this context the PRR (8 bits register) has been set to "0x87", with logic ones to disable. This means that the Timer/Counter 0,1,2 are enabled and the TWI, SPI0, USART0, ADC are disabled.
Then, also the ADC buffers have been switched OFF, setting to logic one the registers DIDR0 and DIDR1.
In addition, also the ADC voltage references ADMUX(REFS0,REFS1) have been deselected and the analog comparator ACSR(ACD).
Finally the Brown-Out-Detection (BOD) has been deselected with the BODFUSE. This feature would reset the MCU each time the supply voltage is under a certain threshold. It is an important instrument in some cases, but for this purpose it is not so important and can be removed to get the maximum save of power.
Considering all these optimizations the current in Active mode has been favorably reduced from 14 mA to 8.2 mA and in Idle mode from 8 mA to 4.5 mA.

3.1.3 Timer/counter setup

The Timer/Counter0 and Timer/Counter1 are used to measure the frequency of the signal from the 555, thanks to the hybrid frequency measurement system well described in [13] and [12].

It works using the overflows of these two counters, one clocked with the external oscillator 16 MHz and the other with the 555 output signal (TCCR0B− > [CS0x]=“111”). The external clock is in input on the pin D2 and activates the pin change interrupt (PCINT20). Timers are not prescaled and overflow interrupts are enabled.
After a fixed number of interrupts of the 16 MHz clocked counter the PCINT are masked and the frequency calculated on the remainder values of the two timers.

The Timer/Counter2 is used to keep the MCU in the settling mode, to guarantee that the 555 is stabilized after its supply. It has been setup to run for 2 ms. The prescaling used is 2024, the Output Compare (OCR2A) has been set for the value 15 that let it wake-up the MCU after this time. Then also the clear timer on compare has been activated, so that after 2 ms it is automatically reset for the next use.
In the code the function that starts the timer is *counter_2_func()* and after it the *sleep_cpu()* is executed. When the timer goes in the ISR of the compare wakes-up the MCU and the 555 can be considered surely settled.
The Watch Dog Timer WDT is used here in interrupt mode to wait a certain time between two consecutive measurements consuming the minimum possible current. To set the watchdog the WDE and WDCE bits of WDTCSR must be set to logical one with the fuse WDTON unprogrammed (logical one). Then to start it the prescaler and the mode have to be set within next four clock cycles.
To use the Interrupt mode, avoiding the Reset, the WDE and WDIE bit of the WDTCSR are written respectively to logical zero and one as reported in 3.4. The prescaler is set to 2048, obtaining 16 ms of period.
Setting a counter of ISR to 19 is possible to get a rough period of 300 ms. Please note that the WDT is not an accurate timer, so the final value of time elapsed can change of 15-20 %.

Table 15-1. Watchdog Timer Configuration

WDTON(1)	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt mode	Interrupt
1	1	0	System Reset mode	Reset
1	1	1	Interrupt and System Reset mode	Interrupt, then go to System Reset mode
0	x	x	System Reset mode	Reset

Note: 1. WDTON Fuse set to '0' means programmed and '1' means unprogrammed.

Figure 3.4. Watchdog timer modes from [17]

3.2 Implementation

For the capacitive sensor, that actually measures the distance of a person from it, the setup is based on a 555 used as astable oscillator and on the ATmega328P as microcontroller.
The microcontroller has the purpose to measure and send the data through the XBEE module to a base station.
From previous work described in [1] the code and interface between microcontroller and 555 circuit were already made. So the changes brought are about the sleep and wake up system and about the optimization of the current code for low power consumption.
The supply of the 555 circuit will be only switched ON when a measurement will be performed and kept OFF for all the remaining time. So, if an external interrupt won't arrive, the supply will be still OFF. Even during the "presence period", i.e. when a movement is recognized by the movement sensor and the system starts measuring, the supply will be switched ON and OFF accordingly to the duty cycle of measurements.
In this way it's possible to reduce the consumption of this circuit, as reported in the Table 5.2, in the results chapter 5.
So the most of the time the 555 circuit will be OFF, not consuming. A general idea of the behaviour of the system can be found in the abstract, with the Figure 1 that is reported also here for more clarity.

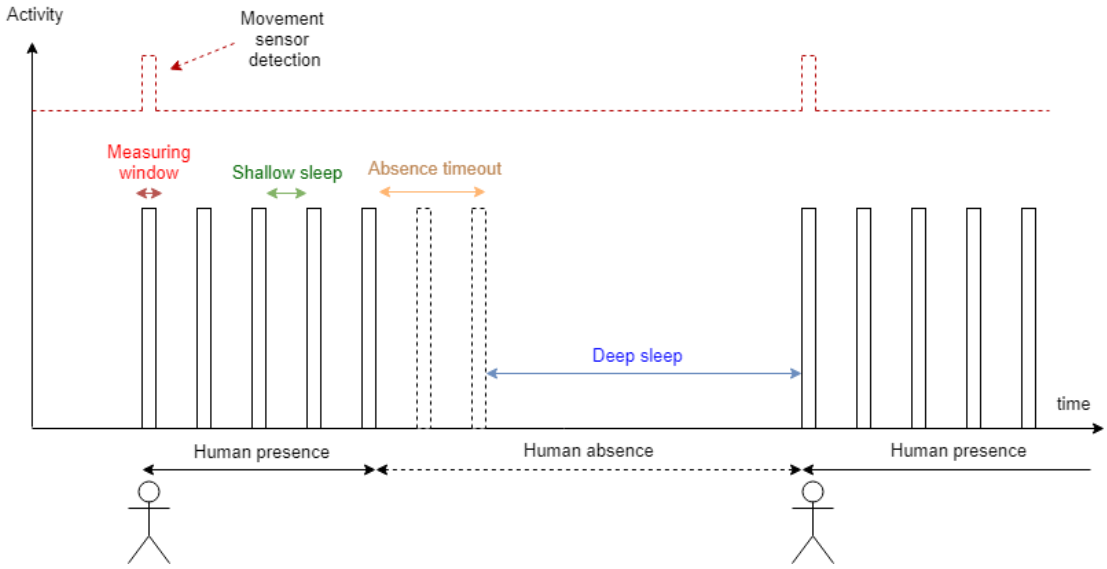


Figure 3.5. Operating flow of the system

3.2.1 555 output frequency settling time at power up

Every time that the 555 circuit is supplied, it takes some time to stabilize its oscillations and this is mostly due to the technology of the component. In order to have the microcontroller properly measuring, the behaviour of the 555 chip has been investigated, looking at its pins for stabilized values after the switch on. The two variants of 555 chip considered were:

- Bipolar NE555, by Texas Instruments [18]
- Low power CMOS LMC555, by Texas Instruments [19]

Looking at the schematic of the 555 used as astable oscillator 3.6 the analyzed pin had been: 3-output, 2/6 threshold/trigger, 7 discharge.
For each of them at first has been analyzed the dependency on the frequency.
So changing the resistances and putting different fixed capacitors in place of the plate, I obtained that both of them will become stable after an amount of time independent from the oscillating frequency (in our operating range) and from the values of passive components.
The two setup used had been:
1) $R_A = 200\,\text{k}\Omega$, $R_B = 560\,\text{k}\Omega$, $C = 15\,\text{pF}$, $f = 45.3\,\text{kHz}$
2) $R_A = 120\,\text{k}\Omega$, $R_B = 330\,\text{k}\Omega$, $C = 27\,\text{pF}$, $f = 50.5\,\text{kHz}$.

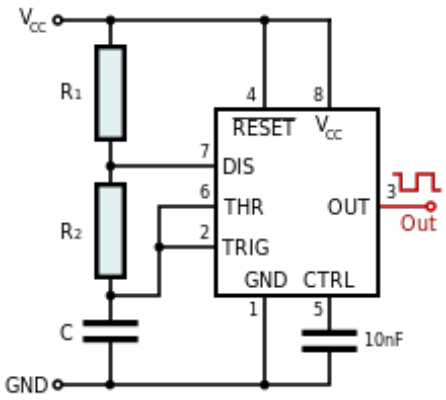


Figure 3.6. 555 schematic for astable mode from [20]

As it's possible to see in the Figure 3.7 and 3.8, this difference for the NE555 in the two different configurations. For the LMC555 the result is the same.

Subsequently I investigated the settling time for the pin 2-6 and 7 of both LMC555 and NE555. The aim is to use the CMOS LMC555, due to its low consumption compared to the bipolar NE555.

Looking at the NE555, the settling after the power up is very fast, approximately 150 µs, for both the threshold/trigger levels and discharge level, as shown in oscilloscope figures 3.9 and 3.10.
So, using this technology, the time spent waiting for stabilization is low and, of course, the consumed power by the microcontroller and 555 is little.
On the contrary, the LMC555 has a longer stabilization time, due to its technology and internal capacitive lines. For the correct measurement in this condition, the oscilloscope has been set to normal mode and infinite persistence, triggered by the supply rising edge. In this way the possible aliasing has been avoided and a correct estimation could be done.
The output signal starts oscillating 30 µs after the supply (fig. 3.11) but the levels are not settled, so the value is not plausible.
In fact the threshold/trigger can be considered stable after 1.3 ms, as in the Figure 3.12, and the discharge after 2 ms (fig. 3.13). So, taking the longest time period, wait time must be 2 ms after the switch ON, to actually start the measurement.

Considering the two technologies for the 555, we measured the values of time and current reported in Table 3.2, where is evident that, during the stabilization, in the NE version, the charge is 2.5 higher than the LMC one.
Then, assumed that, the number of measurement periods is high, this difference

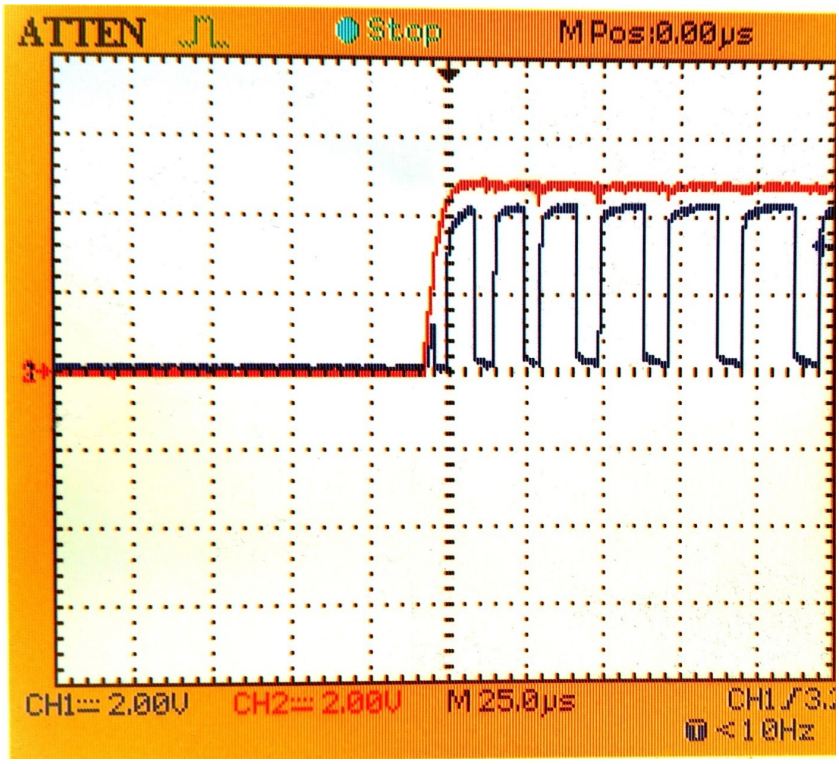


Figure 3.7. Output signal(CH1), pin 3 of NE555 with 15 pF as capacitor. Supply(CH2)

Technology	Current	Settling time
NE 555	3.28 mA	200 µs
LMC 555	130 µA	2 ms

Table 3.2. Comparison of settling time and current consumption for NE(bipolar) vs LMC(CMOS) 555

matters. So, just considering the stabilization, the LMC is preferable. But, analyzing this different consumption over the entire measuring window (that is always around 45 ms), it is even more important as shown in chapter 5, in the Table 5.3, because this time would be wasted waiting.

In the code, a busy wait was previously set, to wait for stabilization of 555 circuit, before measuring. In order to reduce the power during this time, where the microcontroller consumes doing anything, I put it in a sleep mode, where the main oscillator is active.
So, all the other sources of consumption are removed but, thanks to a timer, the

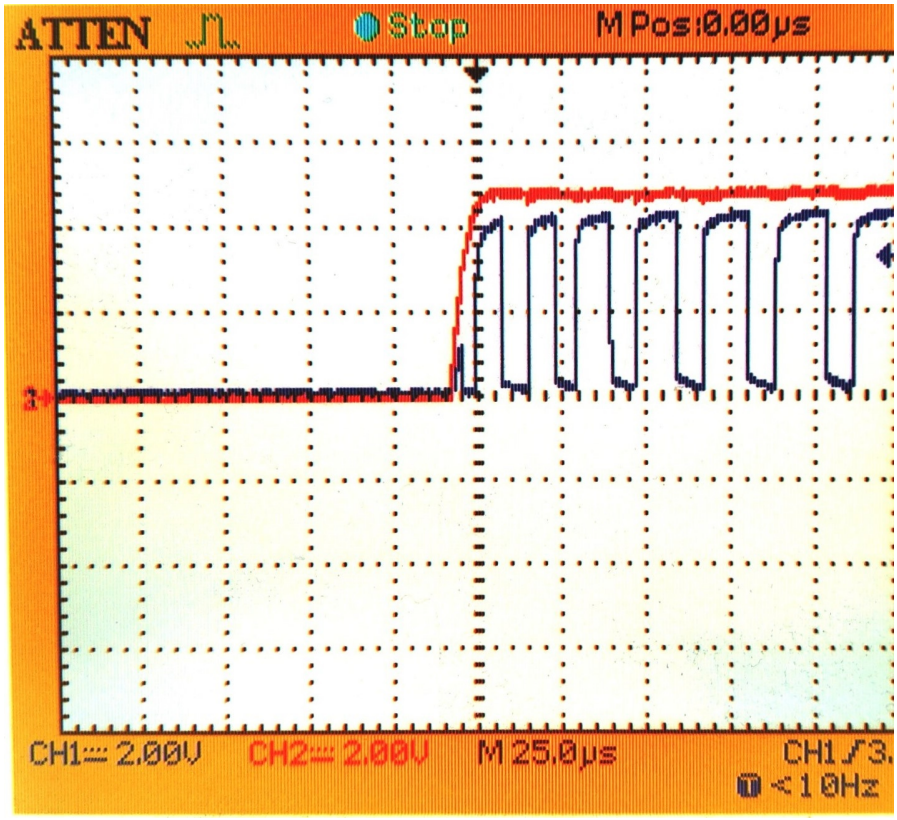


Figure 3.8. Output signal(CH1), pin 3 of NE555 with 27 pF as capacitor. Supply(CH2)

system is woken up after that time of waiting. In the details, the timer used was Timer/Counter2, with the Power Save mode as sleep mode and the wake-up is caused by the interrupt of the output compare ISR.
As parameter for OC, I choose 15 and PR=1024, according to the formula on the datasheet of ATmega328P [17],

$$OC = \left(\frac{f_{OSC}}{PR} \cdot t \right) - 1 \tag{3.1}$$

This counter works with the clk_{ASY} and allows to save significant power, instead of keeping the MCU active during this time.
A further reduction of power could be obtained by the use of the watchdog oscillator (128 kHz) instead of the main one. But, the latter has a very poor accuracy (20-30 %) and in this situation is important to be precise about the waking up, to get correct measurements.

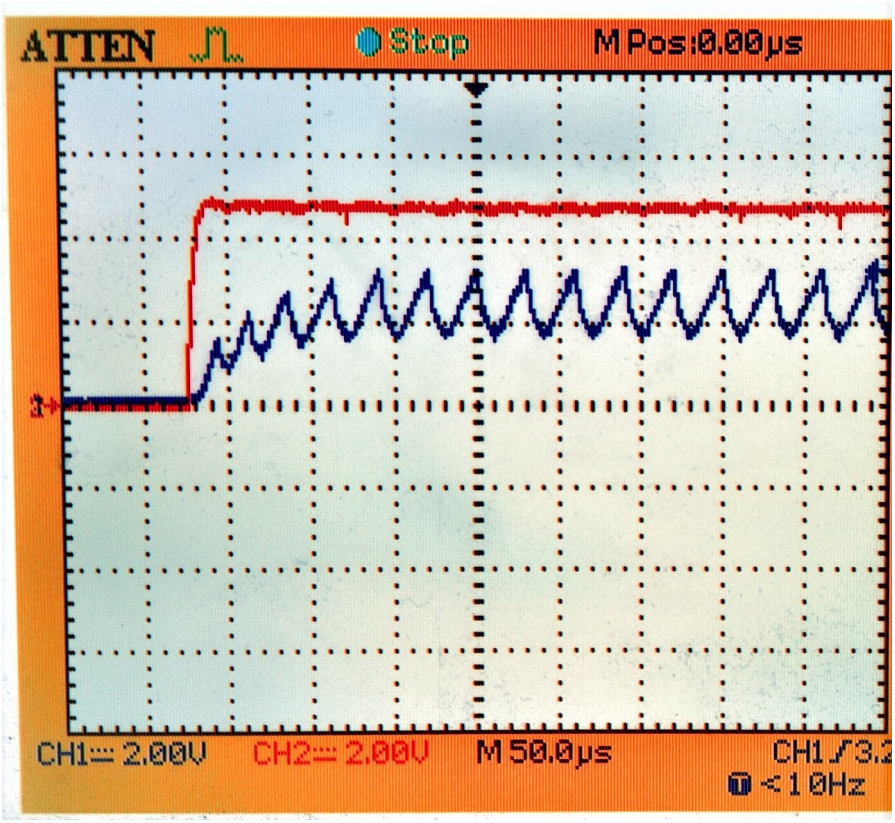


Figure 3.9. Trigger/threshold signal(CH1), pin 2-6 of NE555. Supply(CH2)

Although a late wake-up wouldn't cause many problems, an early one could break the measurement system, putting the MCU in a deadlock state, waiting for a 555 signal that won't be correct. In addition to that, the period is not so large and the differences in power consumed are not huge between the two, due to the duty-cycling of the measurement.
Another alternative, could also be, to use a 32 kHz low power oscillator. If it would have been present on the Arduino UNO board, that implements the ATmega328P, it would have been certainly a valid choice.
So, finally, a timer with main oscillator has been used to wake up the microcontroller after these 2 ms.

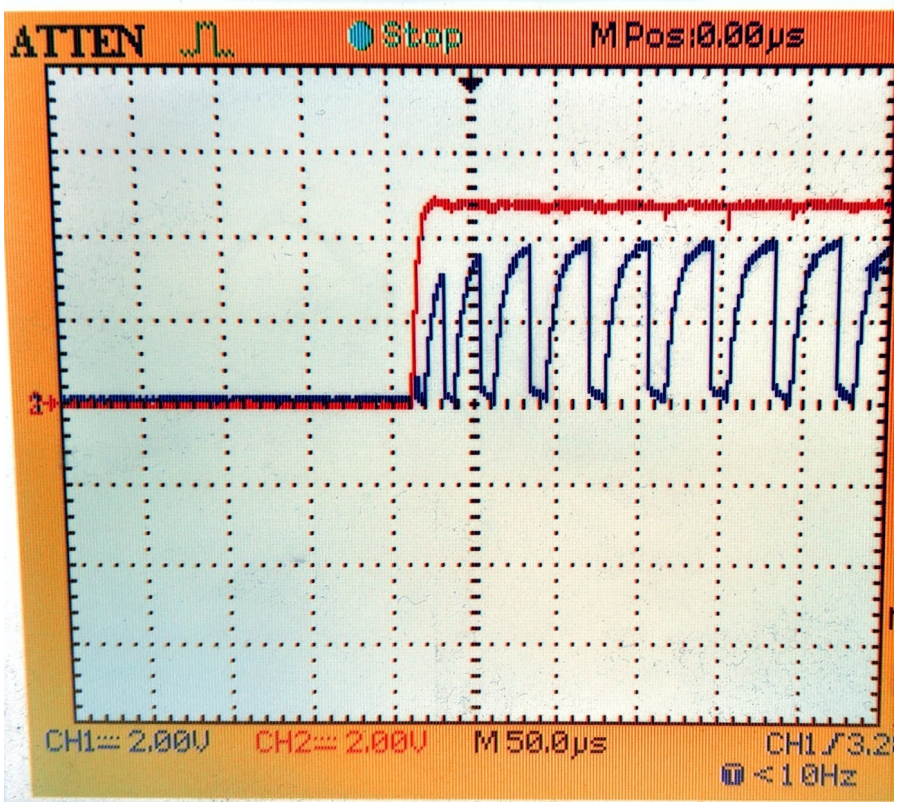


Figure 3.10. Discharge signal(CH1), pin 7 of NE555. Supply(CH2)

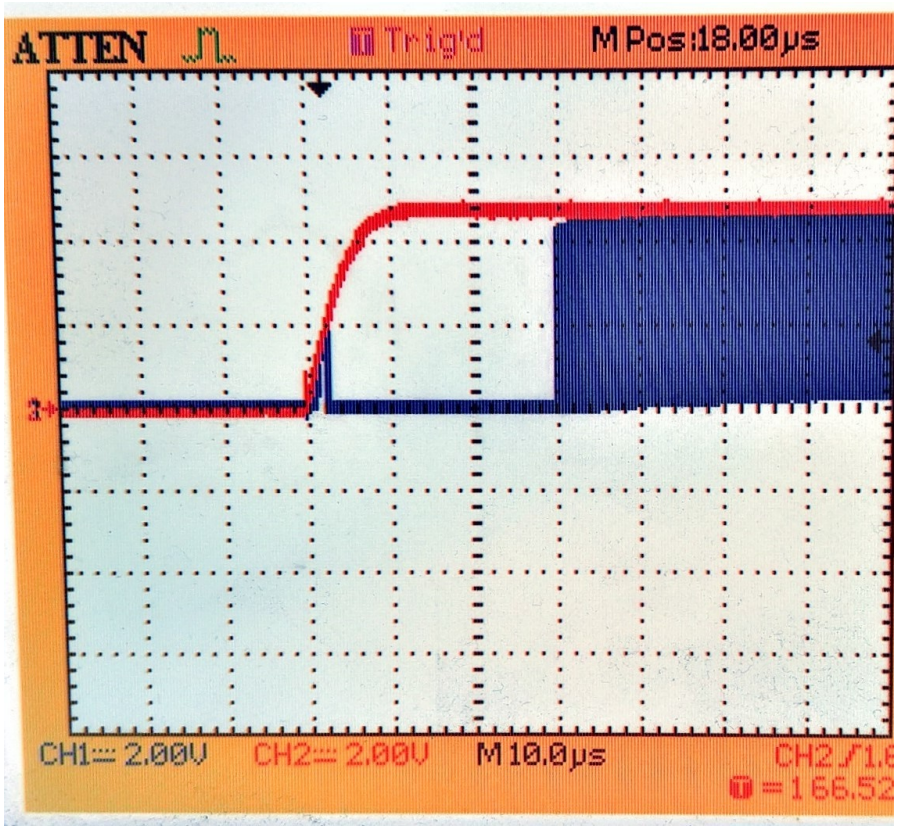


Figure 3.11. Output signal(CH1), pin 3 of LMC555. Supply(CH2)

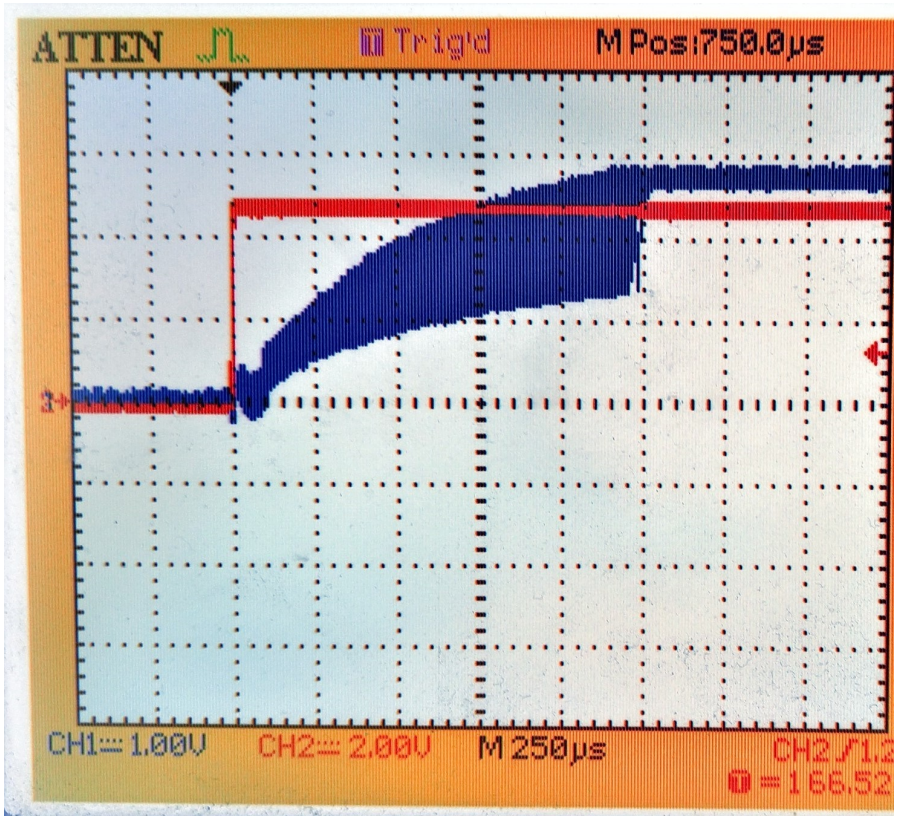


Figure 3.12. Trigger/threshold signal(CH1), pin 2-6 of LMC555. Supply(CH2)

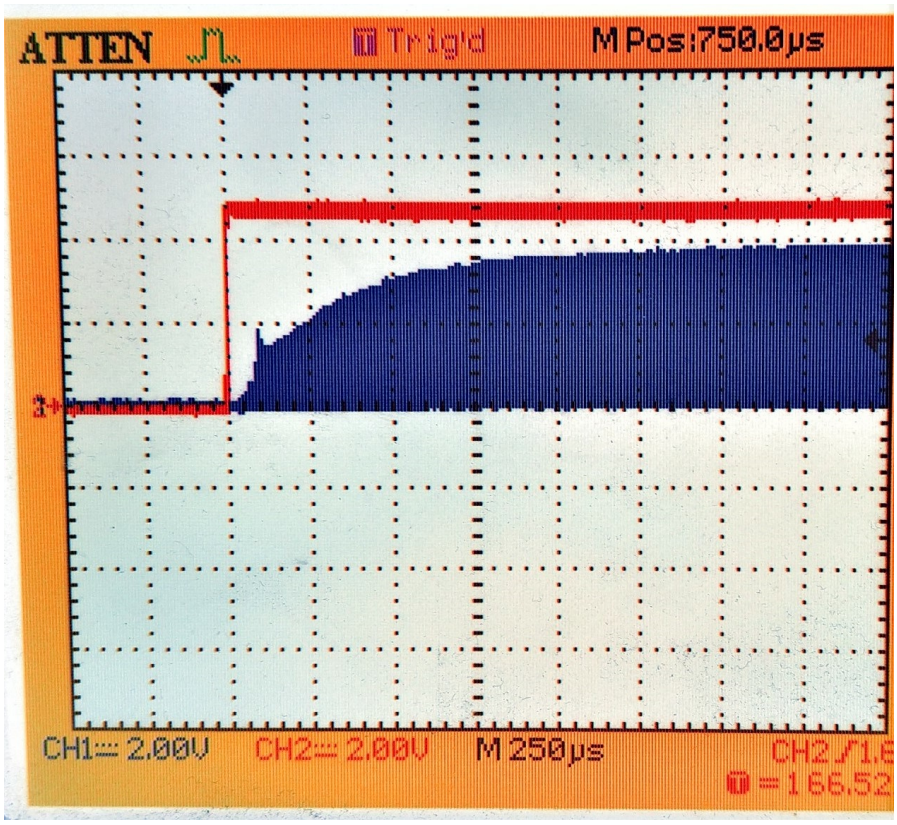


Figure 3.13. Discharge signal(CH1), pin 7 of LMC555. Supply(CH2)

3.3 Measurement window and duty cycle

Assuming that the external interrupts will come dependently on the environment and cannot be forecast, is very important to focus on the period of measurement. It is composed of the stabilization time (2 ms), of the measurement window (41 ms) and of the sleep-WDT time (257 ms). During this entire time, the external interrupts are masked to prevent any unexpected ISR. As reported in figure 3.14, the 555 supply is active for around 43 ms and OFF for the remaining part of the period.

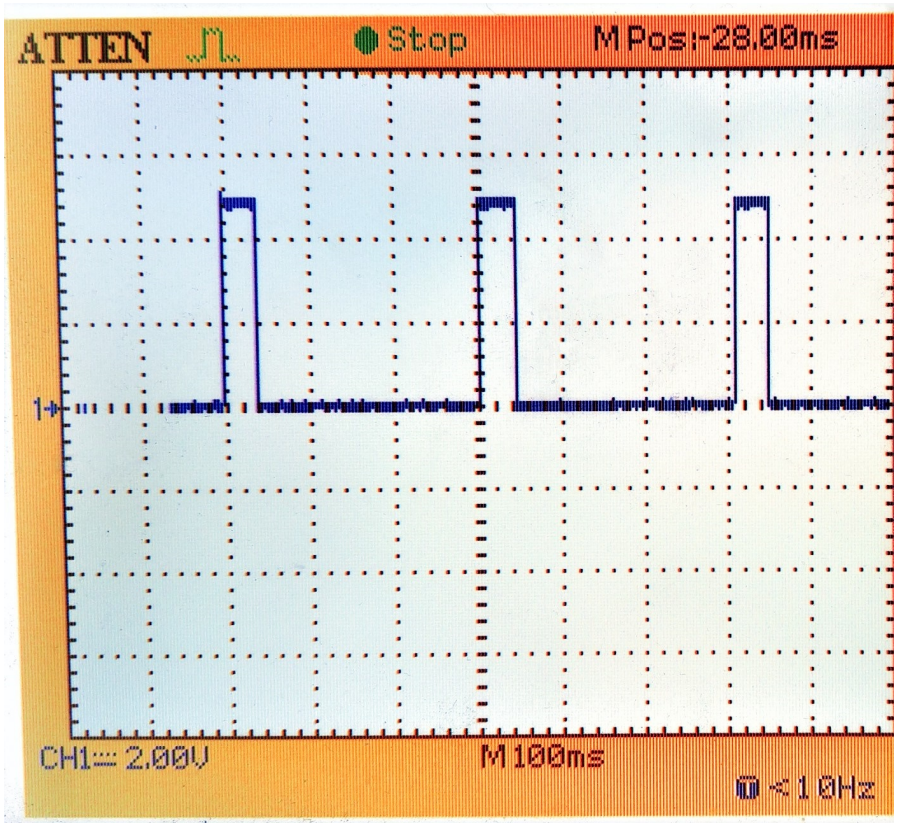


Figure 3.14. Measurement period of 300 ms. CH1 is the 555 supply, high for 43 ms and low for the remaining time

3.3.1 Measurement window - Idle

The stabilization has already been discussed, now the analysis goes on the measurement window. This time is not exactly fixed, because it depends on the number of overflows of the Timer/Counter0. It is an 8-bit counter, clocked with the external

oscillator at 16 MHz.
In the code has been chosen to put the limit to 10 overflows of that counter as time reference. In that time, thanks to the Timer/Counter1, that is clocked with the 555 square wave, are counted the rising edges of the 555 oscillations.
In this way the final frequency measurement can be obtained easily through these formulas, as studied and tested in [13], to get the frequency value with the minimum possible error:

$$N\#_{clk/meas} = ovf\_t1 \cdot 65536 + remaind\_clk \tag{3.2}$$

$$N\#_{555\_periods} = ovf\_t0 \cdot 256 + remaind\_inp\_per \tag{3.3}$$

$$f_{555} = \frac{N\#_{555\_periods}}{N\#_{clk/meas}} \cdot 16\,\text{MHz} \tag{3.4}$$

During a measuring window the 555 is supplied, so to save power, the microcontroller has been put in IDLE mode.
This sleep mode is the softest, because it allows more clock domains than the other modes and, by the way, lets the Timer0 and Timer1 work. During this window, just these two timers are needed to measure the frequency, so the Idle has to be preferred to the Active mode to save power.
As can be seen from the Table 3.1, the clk_{IO} is not halted. This clock is the one used for the mentioned timers. Therefore, the microcontroller can be awoken from the Idle by many events, among them, the timer overflows, that are the needed events.
Additionally, the ADC, Analog comparator and other unused modules have been disabled and the supply has been removed from them.
This helped to save 2 mA, going from the 13 mA of active current to 11 mA.
Then, the save obtained by the Idle mode, compared to Active mode, is around of 60%. In fact, the current has been decreased from 11 mA to 4.5 mA with Idle.
The impact of this change cannot seem too huge but, as can be seen in chapter 5, this allowed to reduce the average current per period of the whole system, due to the big contribution of microcontroller current to the total current.

3.3.2 Sleep mode with watch dog timer

Going through the "measurement period" there is the remaining time, where the microcontroller is in power down sleep mode, the 555 supply is removed and only the watchdog timer is enabled, in charge of waking up the ATmega328P after nearly 257 ms (300 ms - 43 ms).
The use of the watchdog timer in this case is very convenient, for two main facts:

the first, because it is supplied by a 128 kHz oscillator that is very low consuming. The second, because the WDT overflow, if properly enabled, can wake up the MCU from the deepest possible sleep mode, the Power Down mode.
According to [17], the WDT can be set in *interrupt mode* and is necessary to deselect the WDT_fuse, that would keep it in the *reset mode*. Then is possible to set a prescaler to get a certain period time, in this case the choice was 16 ms for a single period.
After 16 overflows of the WDT, the total elapsed time is around 257 ms. It is important to notice that the WDT is not very precise, due to its usual task, but for this purpose the precision is not needed.
The watchdog is just enabled to start measuring time, then after the last ISR, due to overflow it is disabled and the microcontroller wakes up. In this context of "waiting/doing anything" this implementation is very efficient to preserve useless power consumption, although other methods could be more time-accurate.
So, in conclusion, it's possible to define the duty cycle of the activity of the microcontroller (and the 555 supply accordingly):

$$DC = \frac{2\,\text{ms} + 41\,\text{ms}}{300\,\text{ms}} \cdot 100 = 15\% \tag{3.5}$$

The result is significant in the whole considerations, but more analysis must be done on the system's total current, during these periods.

3.4 Absence algorithm

In a first rough implementation, for each external movement detected by the movement sensor, ten consecutive measurements have been done, after which the microcontroller returns to sleep deeply and waits for another external interrupt by the movement sensor.
This is not suitable in each situation, because maybe a human would be still around the sensor and the movement sensor, that is only one in this system, sensible to movements, wouldn't activate the capacitive sensor.
Therefore, in all the successive time, the capacitive sensor would be asleep and would not measure anything. To avoid this kind of problem, it can be useful to implement an algorithm that states when the human is no longer near to the sensor and so the MCU can go to sleep until he returns in his range. This can be done observing the measured frequency, because the capacitance would decrease without any human and the frequency would increases.
So, generally speaking, is necessary to put a threshold in frequency, to state if someone is around the sensor or not. This is not enough, because a single erroneous measured frequency value could stop measurements. To avoid this kind of behaviour, it is possible to check the exceeding of the threshold for multiple consecutive measurements. In that case the system stops correctly.

```
1
2 #define init_meas 20
3 static void set_ref(){
4
5   //Time to get ready to initialization, please go away from sensor
6   _delay_ms(5000);
7   //LED white ON for initialization start
8   PORTB |= (1 << PORTB4);
9   //Supply the 555
10  PORTB |= (1 << PORTB1);
11  _delay_ms(2); //stabilization of supply (can use Timer2 reducing
      power)
12  init=0;
13  sum=0;
14  aver=0;
15
16  while(init < init_meas+1){//how many measures to average it
17
18    init++;
19
20    f_measurement_running = 0;      //Measurement will start with
      first 555 rising edge
21
22    PCMSK2 |= (1 << PCINT20);      //Turn ON PCINT20, start measuring
23
```

```
24      while (f_measurement_running == 0); // Wait for the actual START
        of the new measurement
25
26      while (f_measurement_running == 1); // Wait for the actual END of
        the new measurement
27
28      total_clocks_during_measurement = count_ovf_t1 * 65536 +
        remaineder_clocks;
29
30      total_input_periods = count_ovf_t0 * 256 +
        remaineder_input_periods;
31
32      interm = (total_input_periods * 16000);//avoid overflow
33
34      //gives 555 frequency (normalized in kHZ)
35      f555 = (interm / total_clocks_during_measurement);
36
37      sum= sum + f555;
38    }
39    PORTB &= ~(1 << PORTB1);
40    //set ref as the average of frequency over init_meas measurements
41    ref=sum/init;
42    PORTB &= ~(1 << PORTB4);  //LED white OFF, initialization is done
43  }
```

This code is executed after each measurement period of 43 ms and sets c_low to 1 when the system goes to deep sleep. The set_ref() function sets the value of ref, then consec_out is set in the main code. In the tested cases for this project, tried values were 2, 3, 5 and it worked perfectly.
For further development, it can be increased the number of consecutive absences or, better, using a dynamic averaging function. This would, as drawback, increase the power consumption of the microcontroller, so it's necessary an accurate analysis.
Now, the focus is on the method, to set a threshold on the frequency. It has to be set dynamically, because of the different coupling with the environmental electric fields. It's important to notice that the measured frequency without anyone near the sensor is not fixed, and by now will be called the *reference frequency*.
It changes, accordingly to many factors as explained. So it's impossible to set a unique value. Even after some time, the value of reference frequency will change due to changes in the environment, so a drift can be seen.
In the prototype used in this project, I created a *set_ref()* function. It is executed at each initialization of the sensor (i.e. at each power ON) and must be done without anyone around the sensor. It allows to see the reference frequency, due to the coupling with the room and set it as a threshold in order to know the human presence. This process works repeating a fixed number of measurements (tried 20 and more) and making an average on them.

So, a suitable threshold is now preset. Moving forward, the drift will be better analyzed, but it's possible to make the sensor repeat the set_res() function after a certain amount of time, due to the drift of reference frequency.

```
1
2  //Absence is detected in a measurement
3  if( f555 > ref) {
4
5    //It contains how many consecutive absences
6    flag_no_presence++;
7    //Absences for consec_out(set before) consecutive measurements
8    if( flag_no_presence == consec_out)
9      //Flag to exit from meas. period and go to deep sleep
10     c_low = 1;
11   }
12
13   else {
14     //Avoid that non-consecutive absences are detected
15     flag_no_presence = 0;
16   }
17 }
```

To better analyze the threshold level and the average of a position, I performed some test and checked the results on Matlab, as shown in Figure 3.15. What we noticed was that the sensor is highly subject to environmental noise and had a frequency drift along the time due to the possible capacitive couplings.
Setting the threshold for a distance higher than the range of the sensor is not optimal, because it ends in many false positive cases as tested in the laboratory.
So a possible useful method could be to set the threshold at a distance that allows the sensor plate to couple with the human body. In this case the frequency should be more precise to detect if the human is in this range of detection or not.

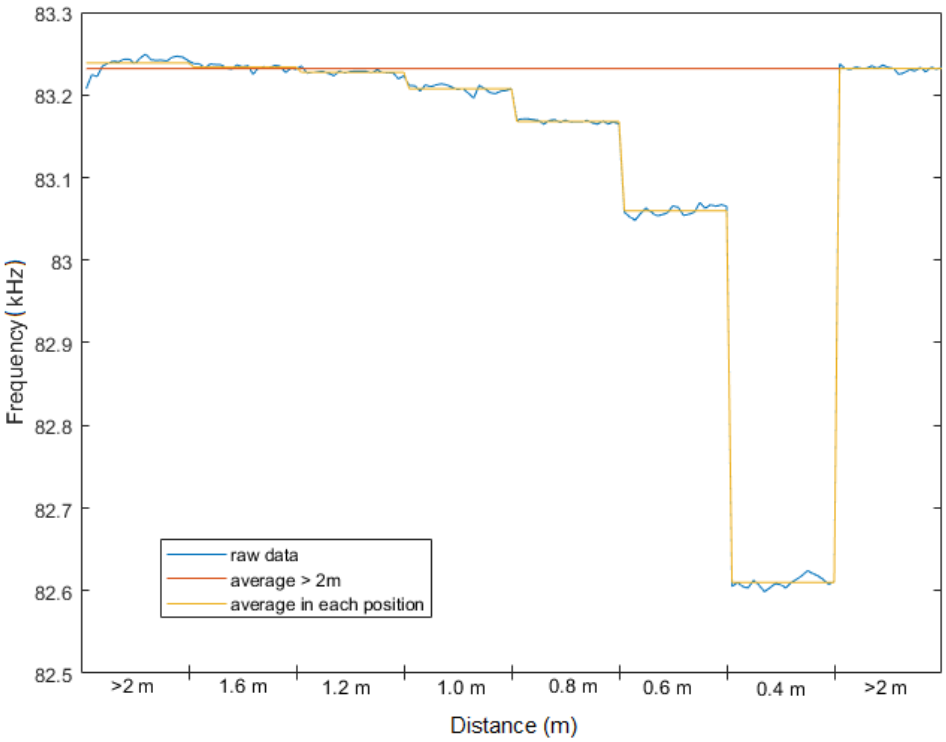


Figure 3.15. Sensitivity test, with average over each position and threshold

3.5 Measurement drift of 555-based front-end

As we saw, setting a threshold to detect the presence or not of a person is not so easy. So, to better analyze the behaviour of the front-end along the time some measurements have been done.
The setup for the analysis has been done with a 8x8 cm plate, with the circuits on a breadboard. The plate was far from any electromagnetic device for at least 1.5 m that is its capture range. It was fixed on a cardboard box, at the height of 1.5 m from ground. The room was for the most empty, but as we will see some movement of people have been detected.

As can be seen in the first plot 3.16, the measurement was set for 6 hours. During this time the sensor has been untouched. The negative peaks in frequency that are visible are due to a human near the sensor for a short time. The frequency drift can be seen easily, but in order to have a better "resolution", the data has been cleaned of the "movement" samples and then some averaging function has

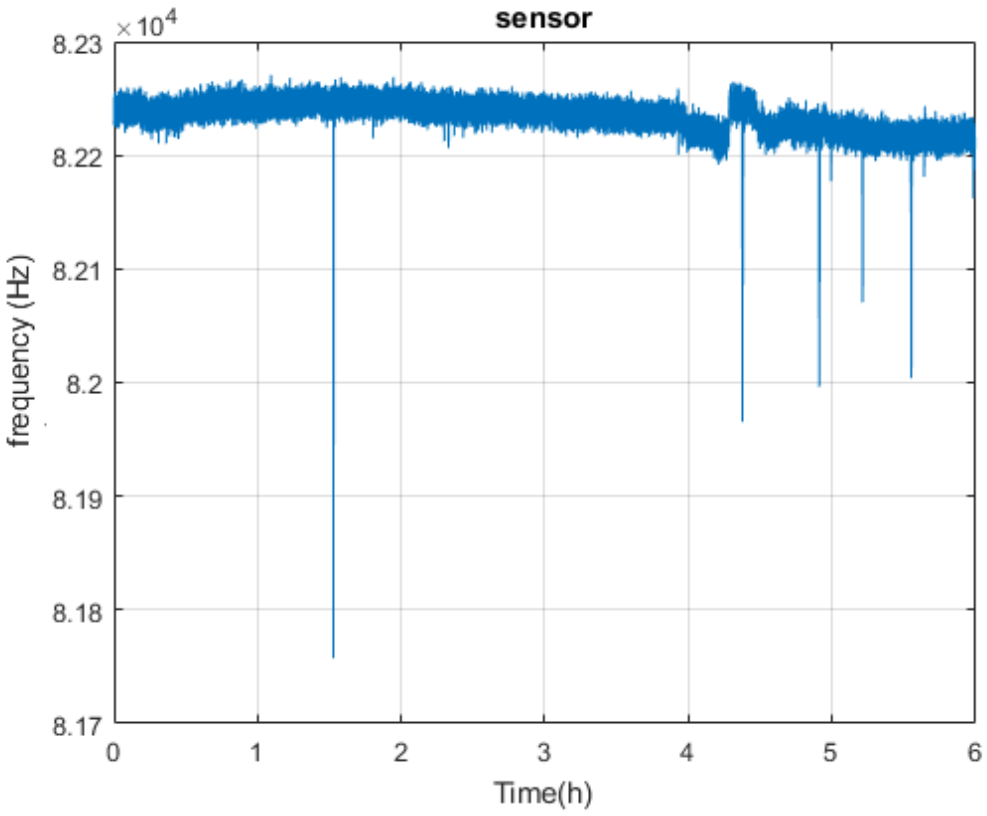


Figure 3.16. 6 hours measurement, with 555-based front-end. The negative peaks are human movements

been used.
In Figure 3.17 is possible to recognize the drift along the time. Then, using some windows of averaging, the slope of frequency variation has been emphasized. In fact, the frequency is more or less stable in a period of 15 minutes.
In addition I averaged the data obtained from the 6 hours experiment over different windows to better understand the drift. Two of them are reported, Figure 3.18, with an moving average window of 1000 samples and 3.19, with an moving average window of 5000 samples.
By a qualitative analysis, that is the only possible analysis, due to the problem itself, the drift is about 10 Hz per hour. But, after the presence of a human near to the sensor, the frequency change dramatically. This is due to the changed electric fields when someone moves and to the successive capacitive coupling with the environment that is different from the previous.
Therefore, using an adaptive set of the threshold could be optimal to avoid these

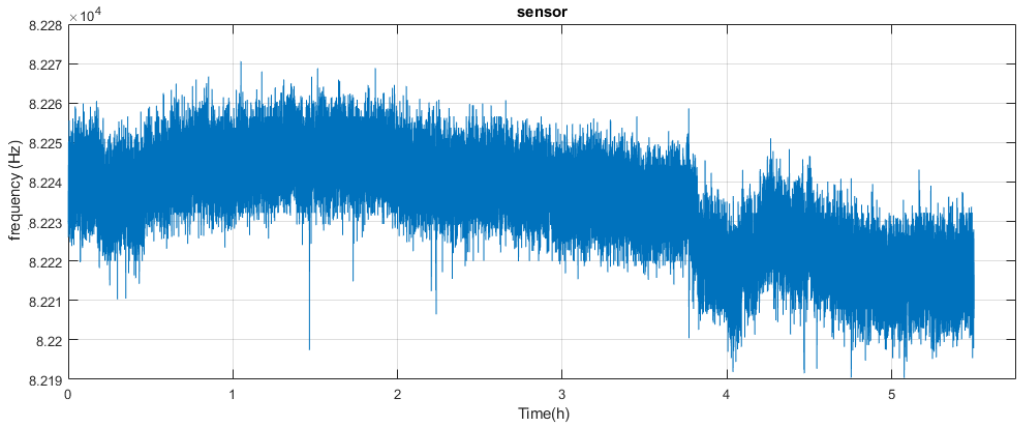


Figure 3.17. 6 hours measurement, with 555-based front-end. Removed movements

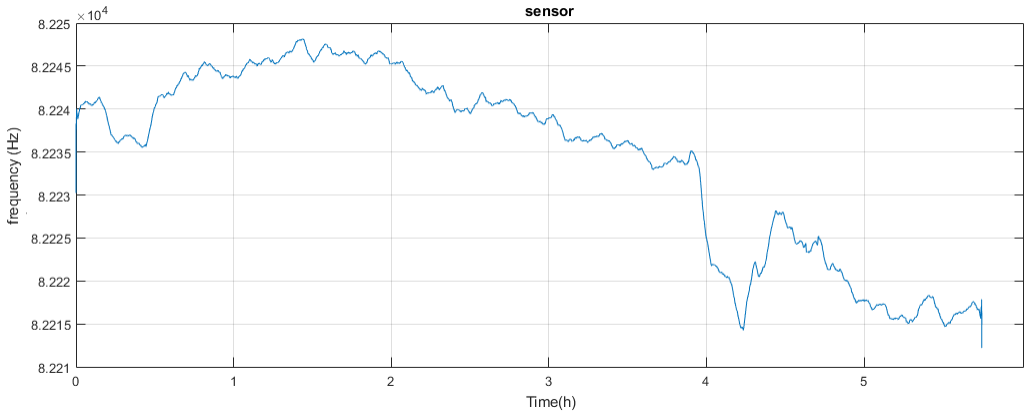


Figure 3.18. Measurements for 6 hours. An averaging function over 1000 samples has been used

problem related to noise and drift. Instead, using a dynamic averaging of last samples is not recommended because it would take into account also samples of low frequency, due to real presence near the sensor.

In the last two figures 3.20 and 3.21 there is the set of measurements done along six night hours. In this scenario, is easy to seen a drift of frequency, approximately 10 Hz every hour.
Then, in the last part, some troubles occurred on the sensor, maybe coupling with some other electronic device working on. But even to these conditions it should detect a person in the 1.5 m range. In fact, the impact of the drift is only on

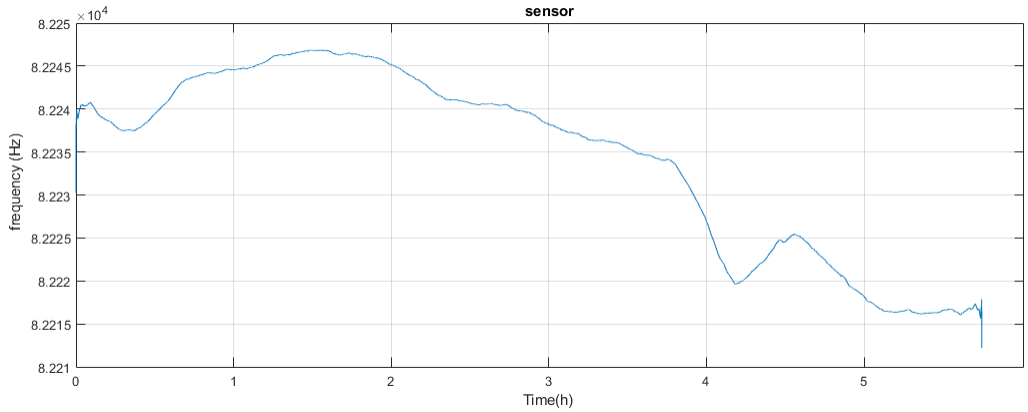


Figure 3.19. Measurements for 6 hours. An averaging function over 5000 samples has been used

the "empty room" frequency and doesn't affect the measurement of people in its range.
The encountered drift, so, is due to the environmental electric fields and their influence on the transducer of the front-end. This is a characteristic of this kind of sensors, so anything can be done in this way with this specific 555-based front-end. A possible solution is to implement a different front-end, based on carrier demodulation as described in [10] and [21], to reject most of environmental noise.

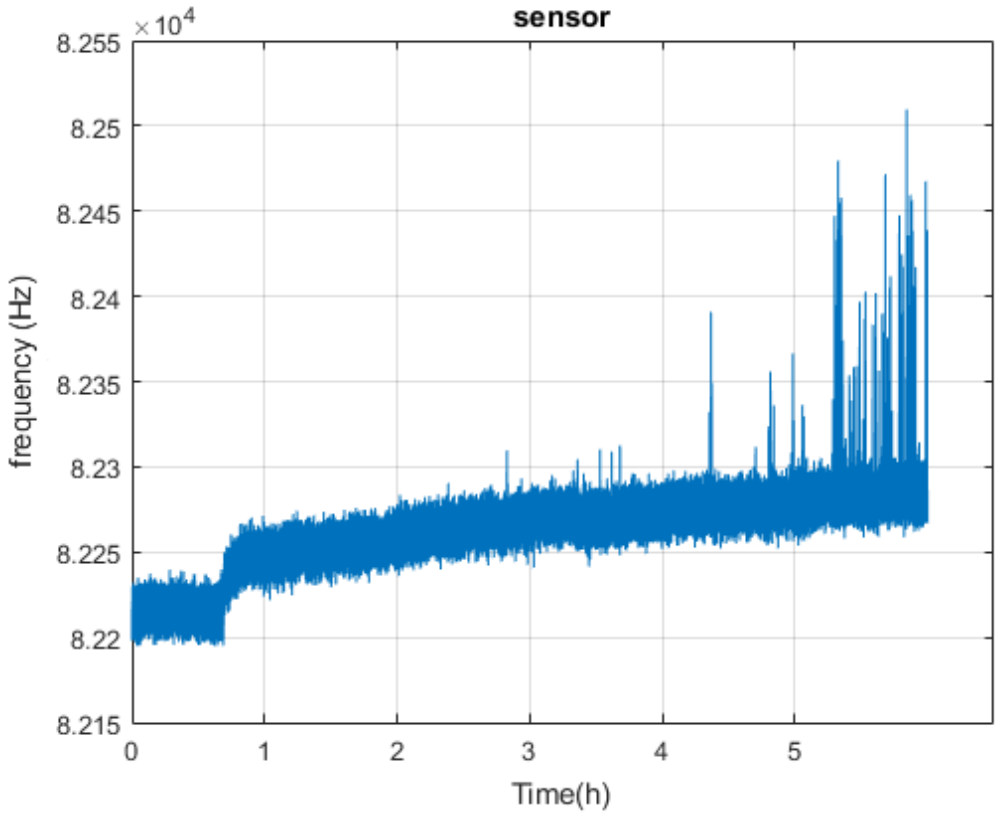


Figure 3.20. Measurements for 6 hours during night time.

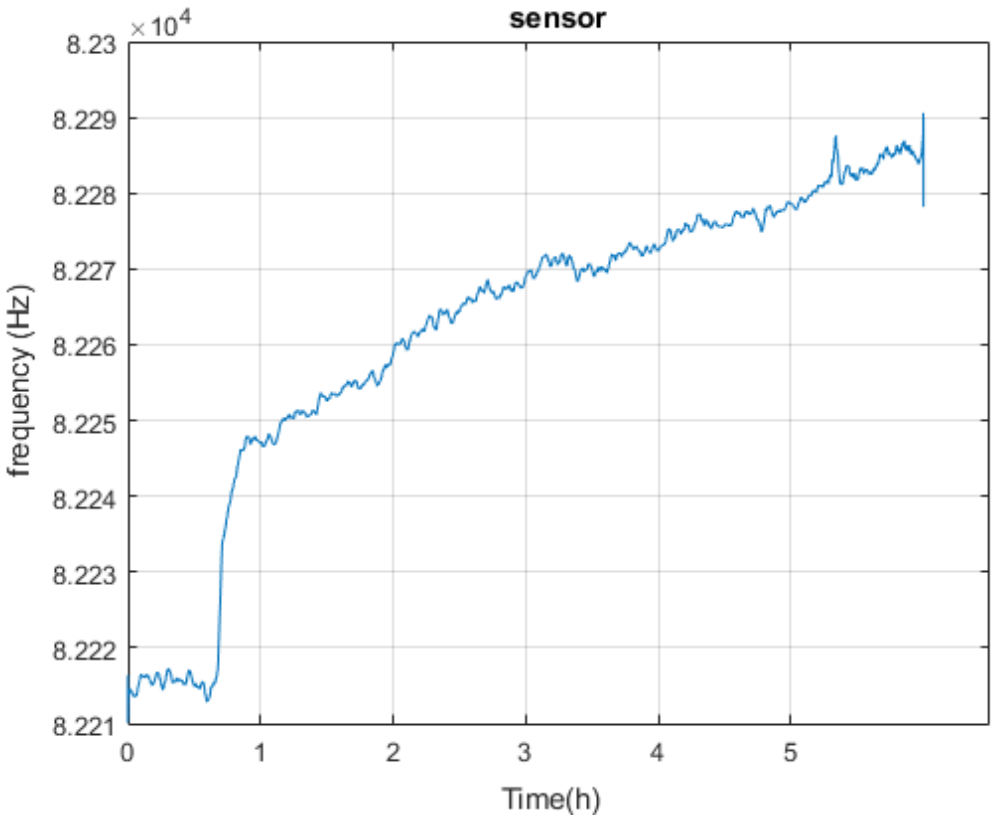


Figure 3.21. Measurements for 6 hours during night time. An averaging function over 1000 samples has been used

Chapter 4

Operating principle modulated carrier front-end

4.1 Front-end comparison

In the previous work, another front-end has been developed, for human localization with capacitive sensor, the one presented in [10] and its schematic is reported in 4.1. This sensor, has the aim of reducing the effects of the captured noise by the transducer, obtaining a long sensing range and guaranteeing a good stability and reliability.
It exploits the charge of a capacitance by the phase shift on a fixed and generated carrier frequency. The work I've done is a link between the 555 front-end and the testing work on this other front-end from [22].
In details, after [22] work on existing PCB, it had been analyzed by me the long term behaviour of the sensor as for the 555 one. This work also helped another thesis project in developing a new perfected PCB. By theory explained in [10] the demodulating carrier sensor should be less exposed to noise so this analysis should confirm the absence of a drift in the output value, that in this case is a voltage. The voltage would increase when a person is within its detection range.
The expectation from the ideal circuit is the absence of a drift in the output, but on the actual PCB the measurements demonstrated a different behaviour.

At first, we performed measurements with the plate, for 4 consecutive hours, without anyone nearby, in the morning, away from walls and furniture. The results showed a huge drift, as reported in figures 4.2 and 4.3.

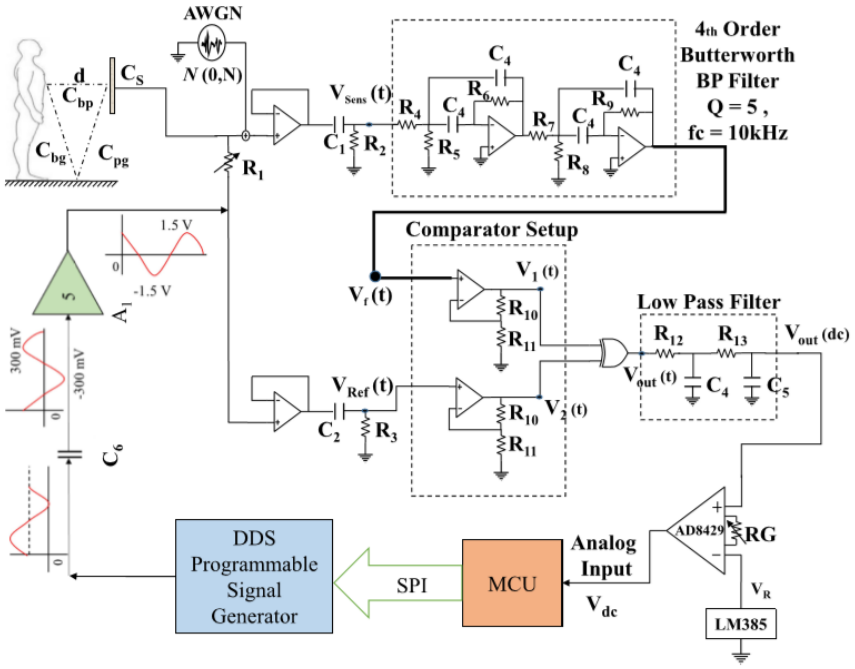


Figure 4.1. Full circuit of modulating carrier front-end from [10]

Then, obtaining these bad results, was necessary to better investigate the cause of this unexpected behaviour. To distinguish the external possible problems from internals, a fixed capacitance has been place in place of the plate. If the drift have been induced by the environmental electric fields (as for the 555 front-end), a fixed capacitor would have avoided the previous drift. The value chosen for the capacitor was 18 pF, to assure a measured value of voltage, similar to the one with the plate.
To find these value, we performed a test on the board with several capacitors in the range of tens of pF. Using the XY mode on the oscilloscope, we found many phase relations between the two branches of the system.
So the 18 pF capacitor has been chosen looking at the most similar phase relation compared to the situation with the plate and no one near to it. The perfect value would be in the range of 18-22 pF but we used capacitors with 20% of tolerance so it's enough to choose it.
What I found is that even in this configuration the system was drifting, excluding the external electric fields as the main source of noise. Figures, 4.4 and 4.5 show clearly the behaviour and the test condition compared to the previous were

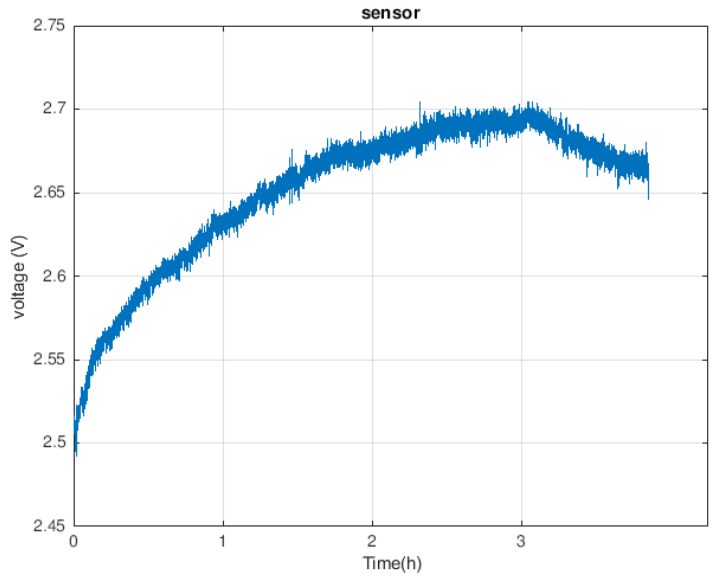


Figure 4.2. Output value from modulating carrier front-end, with no presence for 4 hours and plate, run 1

identical.

Therefore, the cause of these problems must be searched inside the board, that already showed some problems as reported and tested in [22]. A possible source of noise could be the thermal drift of the components, so we tried to keep temperature constant and seeing the results: the drift was still present.
I also performed some tests on the power supply: the first was trying if the XBEE supply could influence in any way the sine wave propagation in the circuit. It doesn't so, that part works as expected.
Subsequently, I tested the behaviour of the signals compared to the expected ones in three points: PWM entering in two branches, output of band-pass filters and XOR output.
In these points, we used a sine-wave generator to compare the 10 kHz wave to the real one. The modes utilized were, XY (to see the phase shift, and eventual frequency difference) and FFT to see any possible component.
During these tests, the supply to the board has been varied in the range 4-5.2 V, in order to see if it had some impact on the signals along the chain. What we obtained is that nothing was changing, with the supply variation. The power-bank we used was delivering 5.1 V so this kind of behaviour is valid for that supply too.
In addition, we observed that even after 10-15 minutes the levels and frequency of the waves were unchanged.

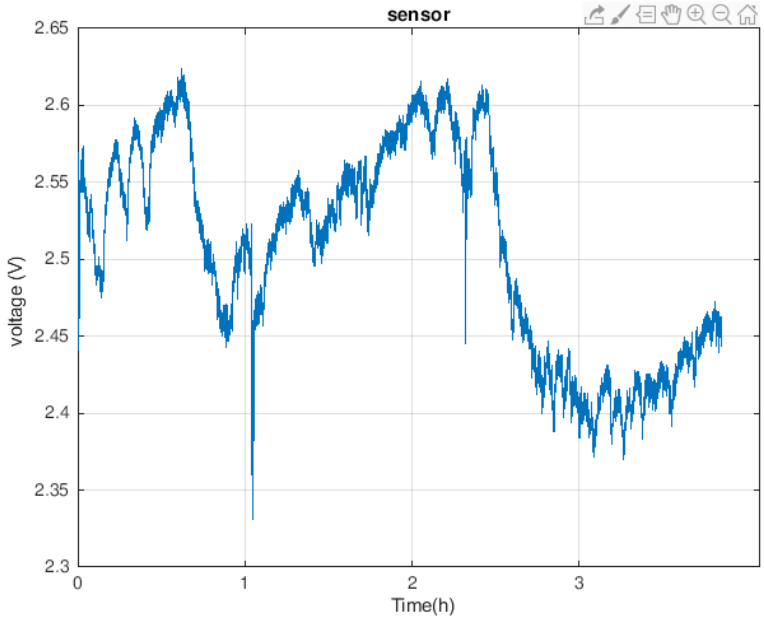


Figure 4.3. Output value from modulating carrier front-end, with no presence for 4 hours and plate, run 2

Thus, these tests showed that the supply systems (regulators) is not affecting the output values and that any drift can't be related to it.
In conclusion, the pass-band filter (the Butter-worth BP filter referring again to the schematic in Figure 4.1) could cause this drift, due to his development in frequency field and use in phase field. I verified its behaviour too, by bypassing it and trying the configurations with a fixed capacitor or the plate.

At first, I made tests with the fixed capacitor to know if there was still the drift. It wasn't, as reported in Figure 4.6 in a 1 hour short measurement. So the most of the noise was caused by it. Then, for a deeper analysis, the configuration was changed again, with the plate connected and the BP filter bypassed. In this condition, in a short-time measurement there wasn't any drift, but knowing the influence of external fields, a long term measurement has been done.
It lasted 17 consecutive hours from the late afternoon to the consecutive morning. It is reported in Figure 4.7. As notable, a little drift occurred, but the relative amplitude to the previous is very small, in fact here we got just a 25 mV change of voltage compared to the 250-300 mV of change encountered with the filter not bypassed.

In Figure 4.8, is reported a long run of 21 hours with the fixed capacitor of 18 pF and the band-pass filters bypassed. What emerges from these measurement

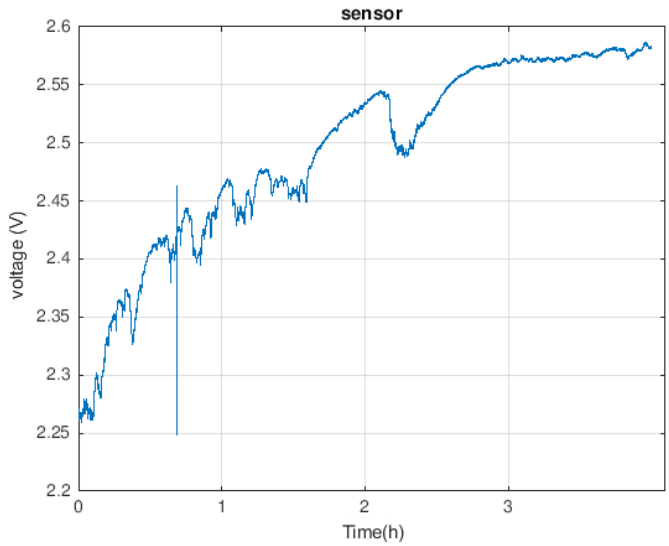


Figure 4.4. Output value from modulating carrier front-end, with a 18 pF fixed capacitor, run 1

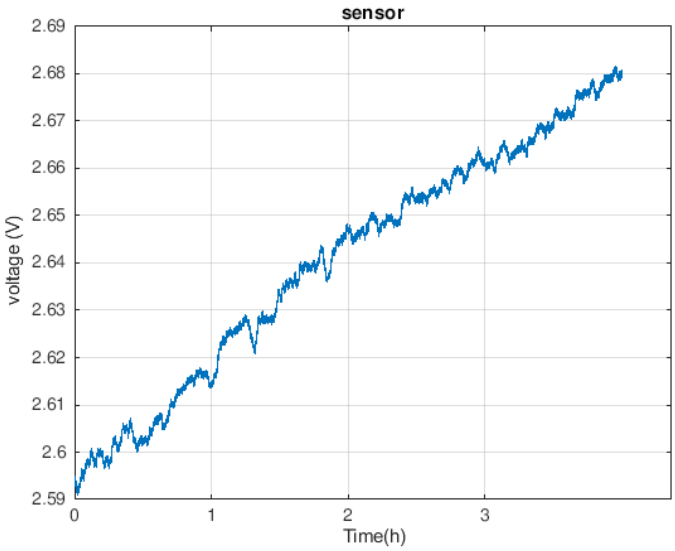


Figure 4.5. Output value from modulating carrier front-end, with a 18 pF fixed capacitor, run 2

is that in these conditions the drift is very limited, compared to the same situation with the plate.

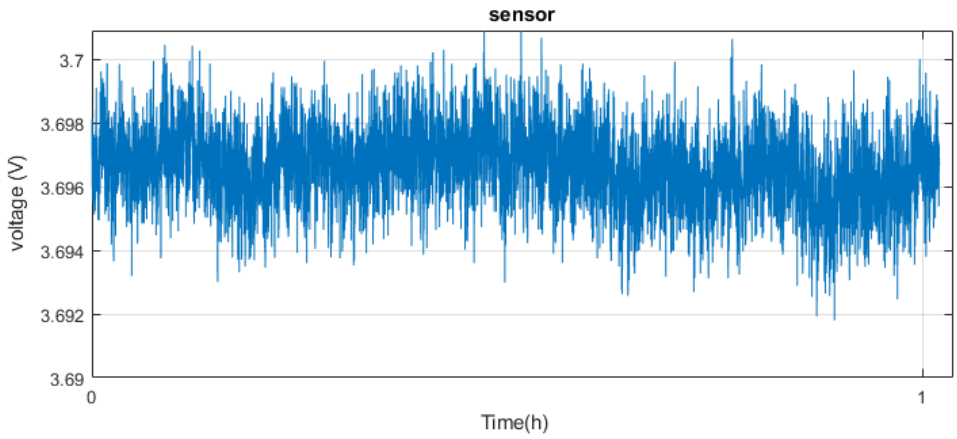


Figure 4.6. Short run of 1 hour, with fixed capacitor and bypassed filter

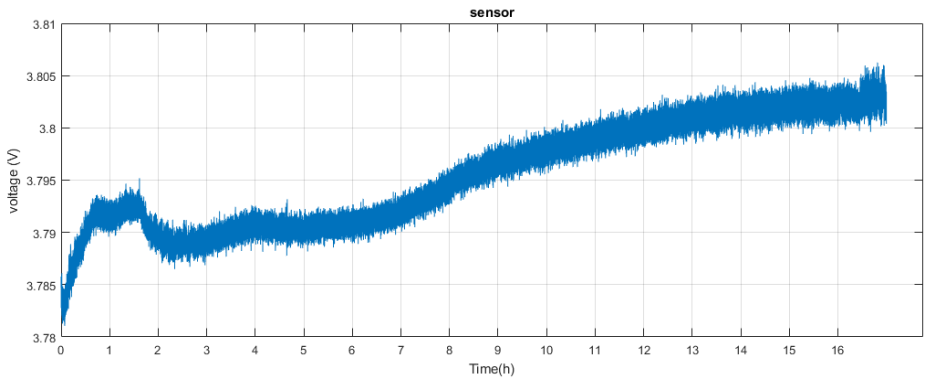


Figure 4.7. Long run of 17 hours, with plate and bypassed filter

In this run we measured a total drift of around 8-10 mV, in the previous one it was 25-30 mV. Hence, the plate is generating a 3 times higher drift. It is due to the high coupling with environmental fields and the sensitivity to humidity, temperature and human presence.
Looking at the Figure 4.8, around 8 AM there is an high peak: this is due to an open window for half an hour.
A further test has been done on the supply system and initial stabilization time. It has been observed the time between the switch on of the supply and the stabilization of the signal that enters in the ADC, so the minimum time to wait for the first measurement.In Figure 4.9, the time is reported and it is around 400 ms.

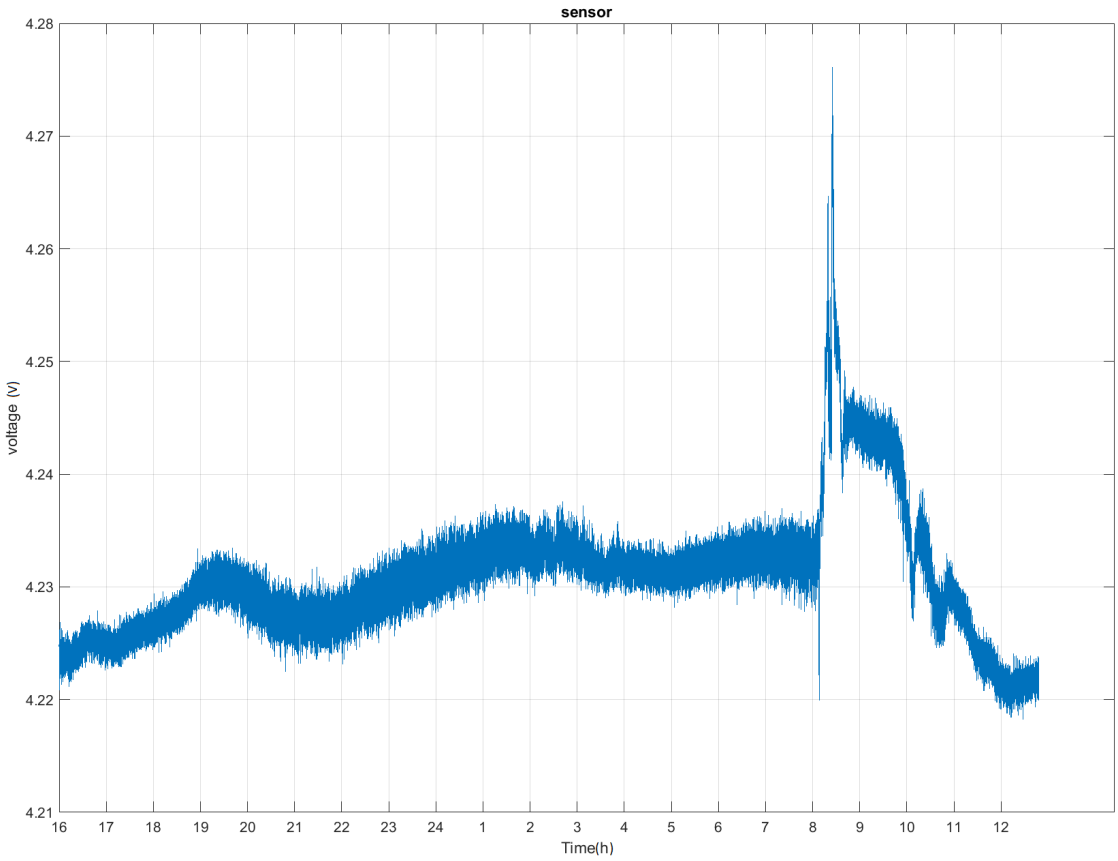


Figure 4.8. Long run of 21 hours, with fixed capacitor (18 pF) and bypassed filter

It is quite long, so a duty cycling, for lowering the power is not useful with this configuration. In this context of analysis for future power optimizations also the ADC sampling frequency has been analyzed and between two consecutive measurements it lasts 150 µs so the signal is sampled at $6.67\,\mathrm{kSa\,s^{-1}}$.

Finally, we made some tests on the sensitivity, with the band-pass filters bypassed. The RC with the plate has been tuned to get the best sensitivity, by looking at the sine-wave on the two branches, just before the comparator.
The best tune is when the cut-off frequency is exactly centered on 10 kHz, so that the phase is $-45°$. To know this, we refers to filter theory and that the amplitude of the filtered signal at the cutoff frequency, in a low-pass filter is $-3\,\mathrm{dB}$.
Looking at the reference value it has an amplitude of 2.56 V, so, to tune it properly, the potentiometer has been moved to get a filtered sine of 1.8 V. This happens for value of resistance around $650\,\mathrm{k\Omega}$. At $\frac{R}{10} = 67\,\mathrm{k\Omega}$ the attenuation is 1 with a voltage of 2.56 V.

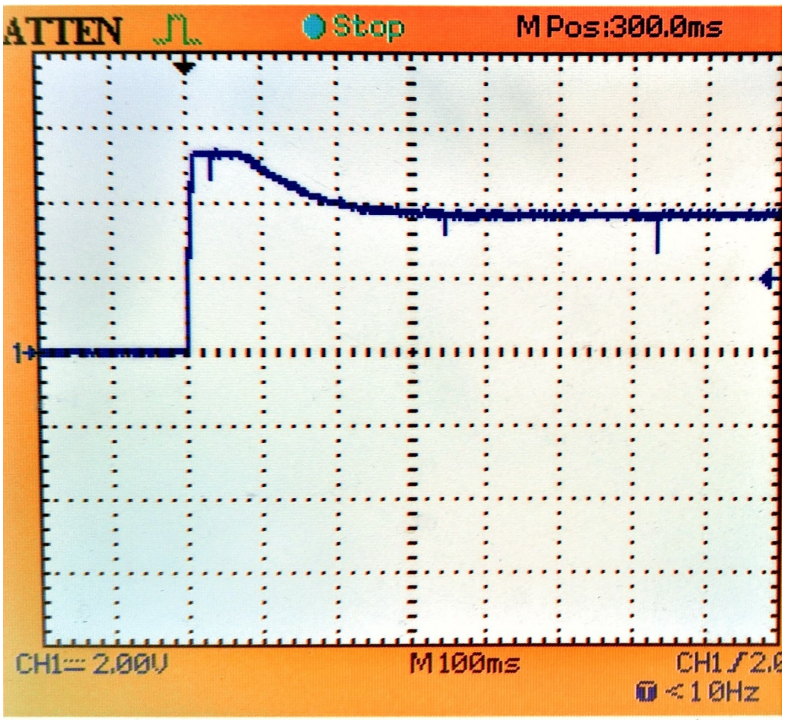


Figure 4.9. Stabilization of ADC input after supply, modulating carrier front-end

Instead at $R = 10 \cdot R = 6.7\,\mathrm{M\Omega}$ the attenuation is 0.13 with a voltage of 320 mV. Sensitivity tests have been repeated for these three configuration giving significant results.
The sensitivity, with the tuned RC is 25 cm as reported in Figure 4.10 and Figure 4.11. The former is a test within 0.5 cm and 2 m, the latter is a round trip test within the same distances. The maximum achieved range is 2 m.

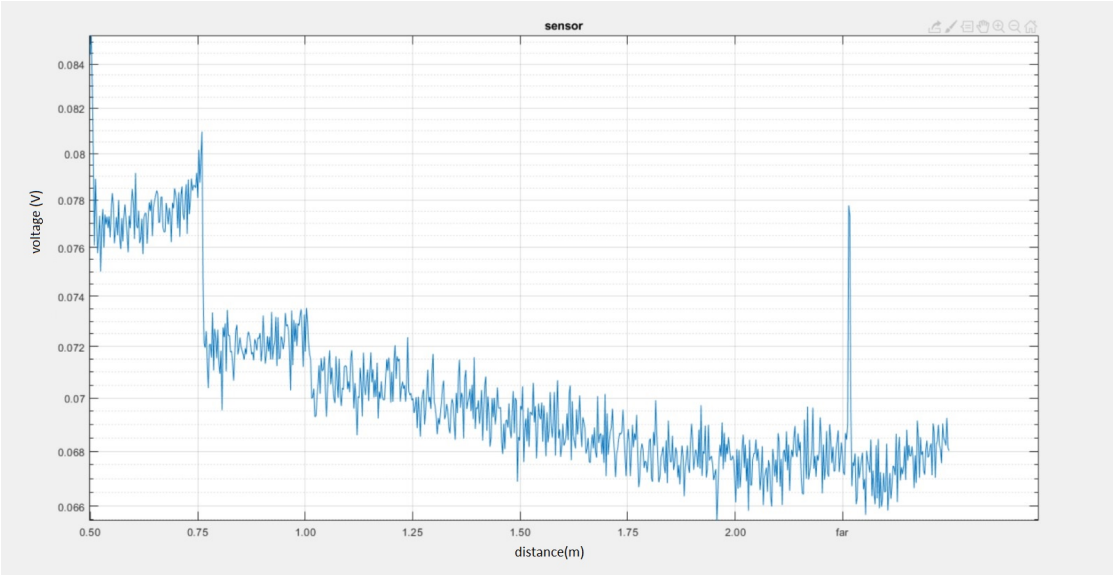


Figure 4.10. Sensitivity test for modulating carrier front-end

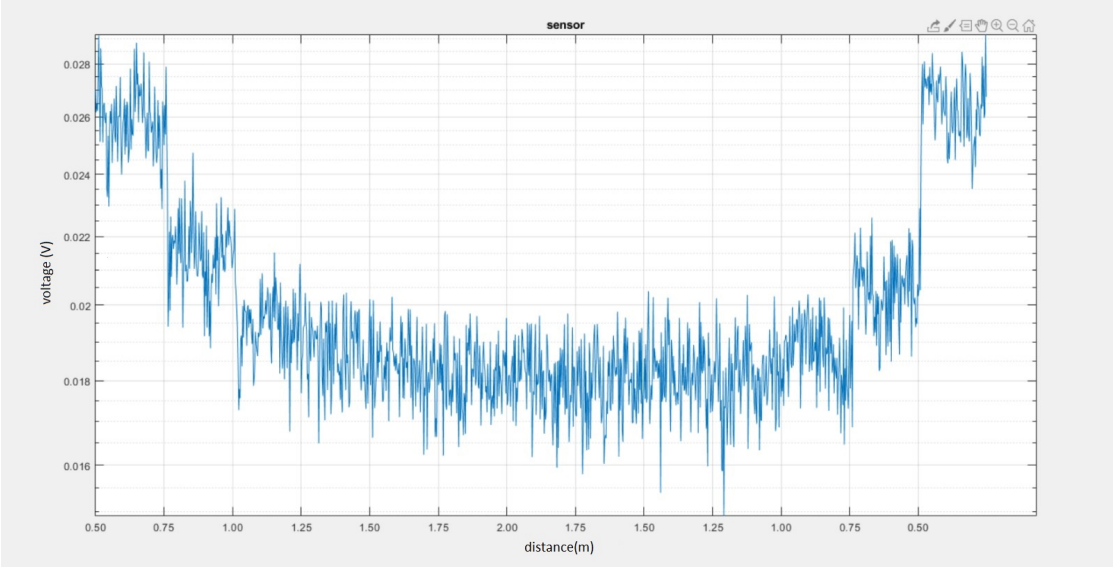


Figure 4.11. Sensitivity test for modulating carrier front-end

Chapter 5

Results

Measurement setup

All the current measurements have been done with an ammeter Wavetek Meterman XP5 Its resolution is 0.1 µA. During measurements the components were on some breadboards and the ammeter has been put in series to the supply. So, each contribution to the total current has been measured independently and they can be correctly summed.
The voltage of the supply has been measured several times to be sure about the value. It is provided by the Arduino UNO board, connected by USB to a computer. Using a power-bank instead, may vary the voltage, so the power calculated may change a little.

5.1 Power consumption of movement

The use of the movement has the aim to limit the power consumption of the whole project. In fact it can consume just some µA during the sleep time, that is the lapse of time without any movement.
The operational amplifiers used for each stage of the movement are the MCP6043 from Microchip [16] and their total current consumption is 2.4 µA with three of them.
All the current due to the passive components have been minimized. In the circuit are also present a voltage regulator and some resistances in order to provide the bias of the input signal and the threshold voltage of the comparator but they are neglected in the following tables, because they could be furtherly reduced with some low power components. The measured supply voltage is 4.86 V.
So, the total power consumption for the movement is 11.64 µW. The active time depends on how many movements it detects and the current during this time is

the same as during sleep mode. The charge consumed by it during a measuring period of 300 ms is 720 nAh.
Results are summarized in Table 5.1.

Current	Power	Supply
2.40 µA	11.64 µW	4.86 V

Table 5.1. Current and power consumption of the movement sensor

5.2 Power consumption of 555 circuit

The 555 circuit, that is composed by the 555, used in astable mode, and 2 resistances, 2 capacitors and the plate, will be just powered on during the measurement windows, so the most of the time it won't be supplied and won't consume any power. In the following table is summarized the consumption, with a measured supply, when present, of 4.86 V. Results are present in Table 5.2.

Mode	Current	Power	Supply
No supply(Sleep)	0 A	0 A	0 V
Active-LMC	130 µA	632 µW	4.86 V
Active-NE	3.28 mA	15.94 mW	4.86 V

Table 5.2. Current and power consumption of the 555 circuit

Then, considering the charge consumed during an entire measuring window, we have to take into account that the measurement window is always around 43 ms, so it's possible to get a difference of 24 times between the two technologies, as results shown in Table 5.3.
Hence, this causes the choice of the CMOS technology even if it takes longer to stabilize after supply.

	Current	Measurement window	Charge	Supply
LMC 555	130 µA	43 ms	1.5 nAh	4.86 V
NE 555	3.28 mA	43 ms	39.2 nAh	4.86 V

Table 5.3. Current, power and charge consumption of LMC(CMOS) vs NE(bipolar) 555

5.3 Power consumption of microcontroller

As already written, the measurements of current of the ATmega have been done in series with the supply it gets from the Arduino UNO. All unnecessary pins and even the JTAG Programmer have been unconnected to avoid any possible unwanted current drain. In the table below, 5.4 it is shown the measured consumption for each operating mode used in this project.
The details of each mode are present in chapter 3. The power refers to the 4.86 V supply voltage and the charge refers to the time period of measurement explained before.

	Current	Time	Charge	Power	Supply
Settling (Timer2)	1.26 mA	2 ms	0.7 nAh	6.12 mW	4.86 V
Measuring window(Idle)	4.5 mA	41 ms	52 nAh	21.9 mW	4.86 V
Sleeping (WDT)	6 µA	257 ms	0.4 nAh	29.2 µW	4.86 V
Total period	629 µA (av.)	300 ms	52.4 nAh	3 mW (av.)	4.86 V
Deep sleep	200 nA	—	—	972 nW	4.86 V

Table 5.4. Current, power and charge consumption of ATmega328P

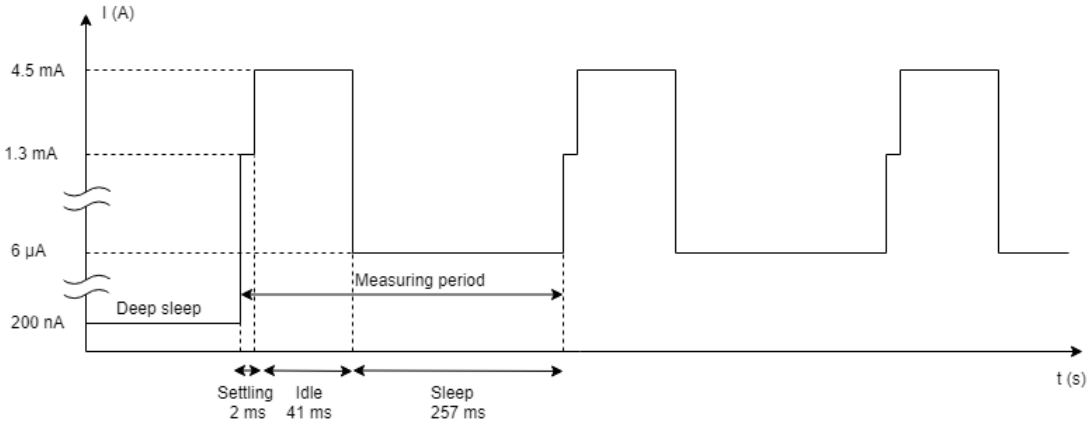


Figure 5.1. Qualitative graph of consumed current in each mode, along the measurement period, for ATmega328P

Figure 5.1 shows in a qualitative way (the axis are not proportional) the current consumption for each mode of use of the microcontroller.
It refers also to the Table 5.4 and explains more clearly the consumption along one measuring period of 300 ms. At the beginning it's represented the deep sleep

and a first external interrupt wake-up from the movement.
In Figure 5.2 is reported the percentage contribution of each working mode for the microcontroller. This consumption is considering the duty-cycle.

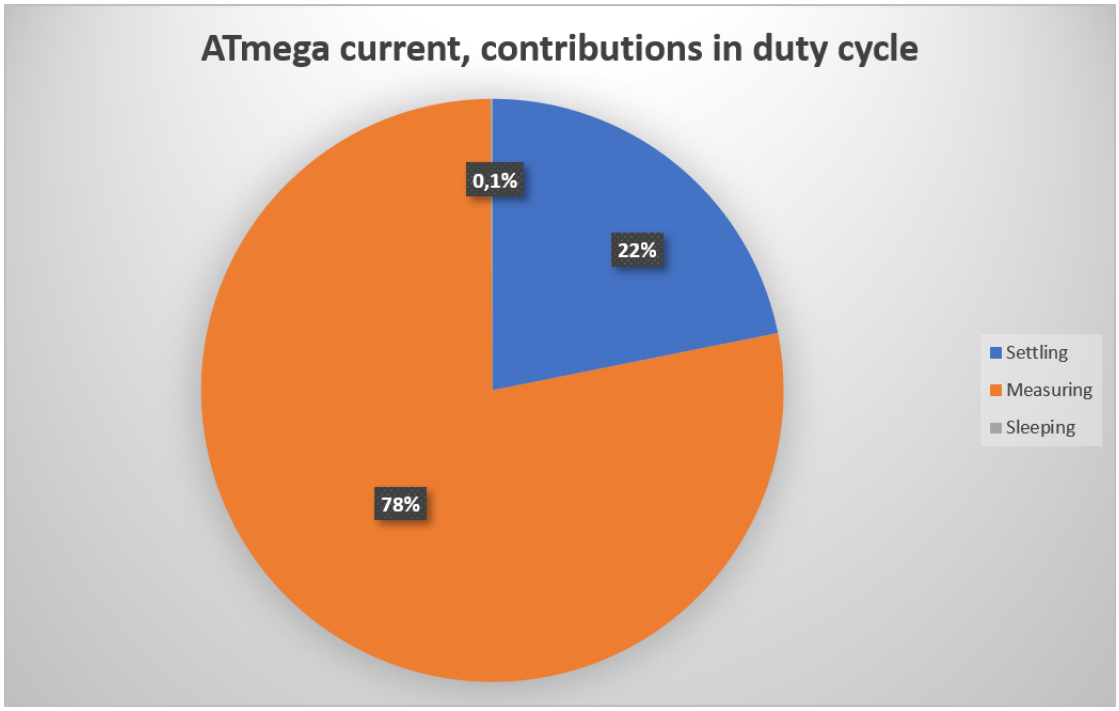


Figure 5.2. Current contribution of each mode along the 300 ms period for ATmega328P

5.4 System consumption and figures of merit

5.4.1 Total system consumption

By summing all the currents from the movement, the 555 and the microcontroller it's obtained the total consumption in each "mode of working" of the measurement period.
For the deep sleep the value is still 200 nA and its duration is only dependent on the kind of room where the sensor is present. For instance, a busy shop would probably keep the sensor always on, on the contrary an house room would wake it up less frequently and the deep sleep would last longer.
Hence, due to the repetition of the periods during the measurements, the most important value is the average current per period. It will let understand the actual consumption, when measurements are done.

Module	Average current	Charge
ATmega328P	629 µA	52.4 nAh
Movement sensor	2.40 µA	720 pAh
LMC555	18.6 µA	1.7 nAh
Total	650 µA	55 nAh

Table 5.5. Total average currents (duty cycled) consumed and charge over a period of 300 ms, for LMC 555

Module	Average current	Charge
ATmega328P	629 µA	52.4 nAh
Movement sensor	2.40 µA	720 pAh
NE555	470 µA	41.1 nAh
Total	1.1 mA	94 nAh

Table 5.6. Total average currents (duty cycled) consumed and charge over a period of 300 ms, for NE555

From the tables 5.5 and 5.6 two main facts emerge: the main contribution to consumption is from the microcontroller ATmega328P, especially during the measuring window (43 ms). Then, the second fact is that the NE555 (bipolar technology) has a bigger consumption than the CMOS LMC555.
This was expected but, looking at the total current and taking into account the relative variation of it, we get $\Delta I_{ave} = \frac{0.65mA}{1.1mA} \cdot 100 = 59\%$. This is a huge relative variation so, regarding the power consumption, also considering the considerations made before on the two 555 technologies, the CMOS one should be preferred.

As can seen from the graphs, 5.3 the movement has a very little contribution in the average total current. On the contrary, the 555 technology has a different contribution, according to the used technology.
Looking at the numbers, for the LMC implementation, the 555 is responsible only for the 3% of consumption, while, with the NE, it is the 43% on the total current consumption.
The NE consumes more current and it becomes comparable to the microcontroller, while the LMC has a less important role in the whole consumption. So reducing the current of the microcontroller, while using the LMC555, is fundamental and it will have more impact on the final consumption.

5.4.2 Figures of merit

Looking at some possible figures of merit for this system, mostly three are very interesting, considering some fixed battery capacities. The first one is the maximum number of measurements for one "battery charge" and is obtained considering the average current of the system along a measurement period of 300 ms.
Then, from the number of measurements, that is also the number of periods, it's possible to get the maximum time, while the sensor will be always ON. This scenario could be a place opened 24h/7d and always crowded. Finally the maximum time while the system is not activated and sleeping has been evaluated.

	LMC	NE
Max measurements	49 M	29 M
Always ON time	4000 h (165 days)	2400 h (100 days)
Maximum sleep time	$>$ 100 years	$>$ 100 years

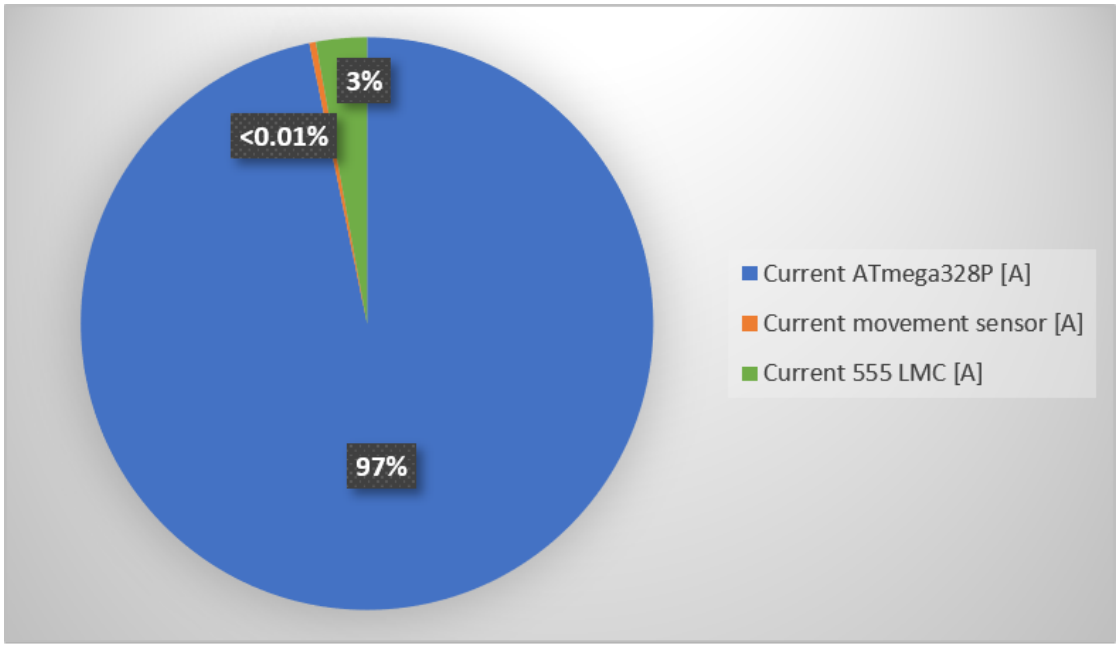
Table 5.7. Duration using an AA alkaline battery as supply, 2.7 Ah

	LMC	NE
Max periods	370 M	215 M
Always ON time	1250 days (3.5 years)	750 days (2 years)
Maximum sleep time	$>$ 700 years	$>$ 700 years

Table 5.8. Duration using a power-bank battery as supply, 10 Ah

Analyzing the results from tables 5.7 and 5.8 some considerations can be reported: with the system in sleep mode, without any activation, the time is very long and the result is very significant. This means that in a scenario with very few activations the battery would last until it will discharge by itself. This is an

upper bound for the system duration evaluation.
Therefore, considering the minimum duration, i.e. when it continues measuring without deep sleep, we got also remarkable results. In fact, for an AA battery with 1.5 V, that usually is around 2.4 Ah the battery should be charged/changed roughly every four/five months for the CMOS technology of 555 and three months for bipolar technology of 555.
But, with an even bigger battery, as a power-bank of 10 Ah of capacity it would continuously working for more than one year. So we got also the lower bound.
The other consideration is about the technology: the bipolar technology for the 555 will consume more than the CMOS as already seen and, in terms of time, we get as expected the same increase of 1.29 as for the average consumed current.
The graph 5.4 represents graphically the comparison of always ON time between technologies and batteries' capacity.
Obviously, the analysis on the AA is purely informational, because an alkaline AA has a nominal voltage of 1.5 V. This analysis has been done to estimate the figures of merit for a generic battery. A possible implementation can be with 3 AA batteries in series, to provide a sufficient voltage. But, in a future development, using a very efficient DC-DC boost converter, can allow to obtain the 5 V.



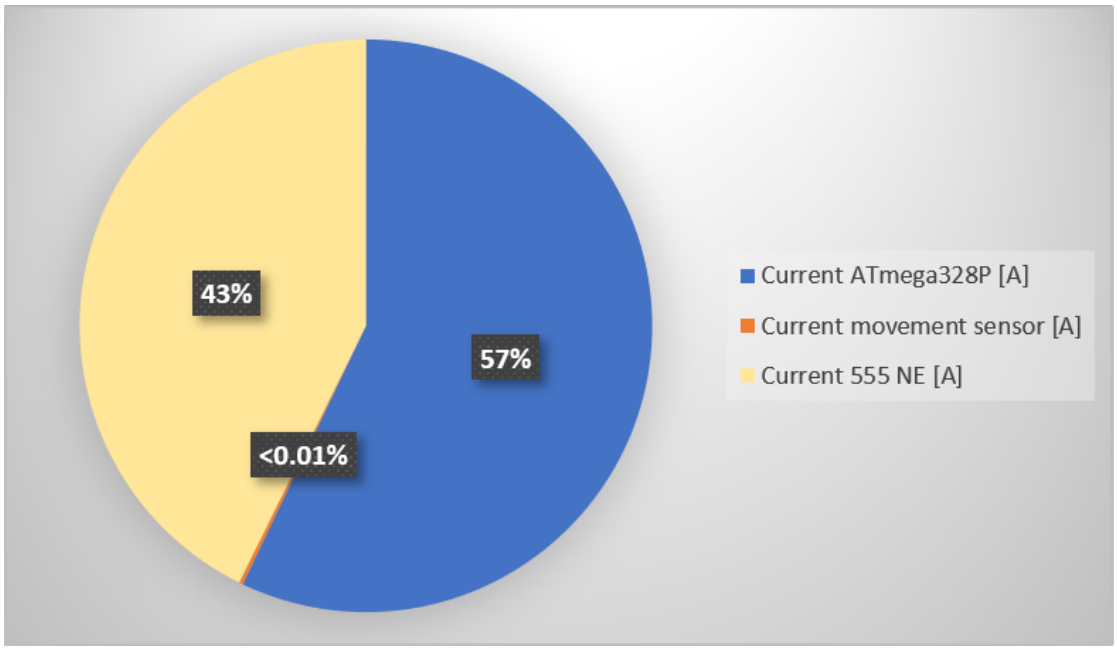


Figure 5.3. Contribution of each module to total average current (values are already duty cycled). The graph above considers CMOS LMC555, the one below bipolar NE555

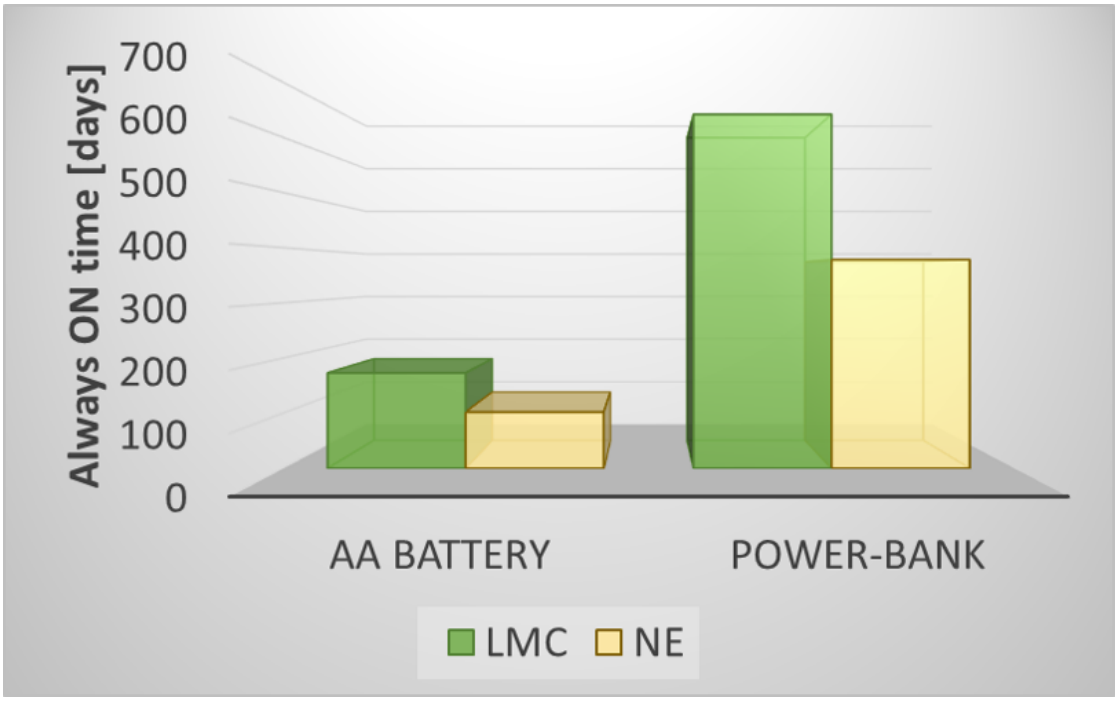


Figure 5.4. Always ON time for the two technologies with the two considered batteries

Chapter 6

Conclusions and further work

6.1 Movement sensor

6.1.1 Comparator threshold tuning

As reported in chapter 2, the position of the sensor in a room can vary the offset of the signal and so the threshold of the comparator should be changed accordingly. In this project, for sake of simplicity the threshold has been selected on the level of the signal and made with a fixed voltage divider. This can't certainly be a possible final retail solution.
So the threshold should be tuned manually in a certain environment, but it would be necessary a difficult setup for possible users. Therefore, a possible future improvement of the system should be an automatic threshold tuning system.
On one hand that would cost some extra consumption, but if done as a first and unique setup it wouldn't be so impacting. A possible idea could be letting the microcontroller control the threshold and set it nearly 300 mV above the signal, when no one is moving. This arbitrarily value is set by looking at the signal whit and without movements.
In fact, a possible noise won't exceed of this quantity, on the contrary it would cross that level when someone will move. A lot of considerations on it could be done, mainly to keep the power consumption low, but it could definitively be an interesting improvement.

6.1.2 Movement sensor integration

In a real application, would also be necessary to avoid the movement sensor detection after a certain number of measurements. Imagine that only the first set of measurements matters and after them, tracking is not important to detect anything for some time.

It can be possible to make the movement sensor blind, through the use of chip-select. In a further development, it could be possible to select externally, in the setup phase, the possible blind time of the sensor after a successful measurement. In conclusion, using a movement sensor like this to detect the presence is very efficient looking at the consumption, but can have some problems due to its sensitivity to noise. Some problems can be solved as explained, but others, as the influence of external electric fields and the shielding are not avoidable, so the position in the room must be chosen carefully.

6.2 Capacitive sensor

6.2.1 Initialization of capacitive sensor (555 front-end)

As described in chapter 3, it's important to provide the first setup, in order to set the reference frequency and use it as a threshold to deduce if a human is present or not.
In fact, a case when a person moves near the sensor and then sits will be detected in this way: the movement sensor sees the first movement in his range so the presence one starts to measure with 3 measurements per second. Then it continues until the person stands up and goes away. Without this reference, for instance, when the human is sit, the capacitive sensor would stop measuring and the movement one wouldn't detect anything because it's only sensible to movements.
If the user is not interested in such mode of working it will be possible to deselect this mode and set a fixed consecutive measurements. There are, in fact, many possible little changes to apply to the sensor, in order to make it suitable to many different purposes and situations.

6.3 Modulated carrier front-end

6.3.1 Optimizations

With this other front-end, that exploits the phase shift over a carrier frequency, many tests have been done in order to optimize it, in its structure and for low power purposes.
About the stability, the pass band filters in one branch are cause of a drift, not so dramatic, but they could be optimized, in order to reduce it, without loosing the sensitivity as with bypass of them.
Also the dimension and speed of the Op-amps could be optimized by changing them. For example using a TL034xx Operational Amplifier could be a well suited choice. They are appropriate for low power, as already seen, and they can also be

suited for this purpose because of their frequency operating range. It has a gain bandwidth product of 500 kHz, so it could well fit the applications of this sensor. Then, the typical supply current is around 270 µA, appropriate for this kind of circuit. In addition to this, it is a quadruple operational amplifier and can fit the four needed (excluding the two pass-band filters).
It would reduce the area of the board and it consumption. Reducing the consumption, that now is nearly 140 mA, for the entire board, could also be a good solution for DC-DC converters and voltage regulators. In fact, during the tests, happened to have to substitute them. The probably were damaged by the maximum flowing current through them.

Appendix A

Code

```
1
2 #include <avr/io.h>
3 #include <stdlib.h>
4 #include <avr/interrupt.h>
5 #include <util/delay.h>
6 #include <avr/sleep.h>
7 #include <avr/wdt.h>
8
9 // Constants
10
11
12 #define F_CPU   16000000UL
13 #define BAUDRATE 9600
14 #define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)
15
16 // Local variables
17
18 uint32_t total_clocks_during_measurement = 0;
19 uint32_t total_input_periods = 0;
20 uint32_t pkt_counter_temp = 0;
21
22
23 volatile unsigned char f_measurement_running = 0;
24 volatile unsigned char count_ovf_t0 = 0;
25 volatile unsigned char count_ovf_t1 = 0;
26 volatile unsigned char count_ovf_t2 = 0;
27 uint8_t measurement_window_expected_duration = 0;
28
29 volatile uint8_t remaineder_input_periods = 0;
30 volatile uint16_t remaineder_clocks = 0;
31
32 volatile int c_low = 0;
33 volatile int flag_no_presence = 0;
```

```
34 volatile unsigned char ovf_wdt = 0;
35 volatile unsigned char awake = 0;
36
37 //volatile float f555 = 0;
38 //volatile float interm;
39
40
41 // Local function prototypes
42
43 static void sleep_mode_func_pwrdwn();
44 static void sleep_mode_func_idle();
45 static void sleep_mode_func_pwrsave();
46 static void watchdog_set_func();
47 static void counter_2_func();
48
49
50 static void reverse(char *str, int len);
51 static long intToStr(long x, char str[], int d);
52 static void ftoa(float n, char *res, int afterpoint);
53 // ISR
54 static void set_ref();
55
56 //when WDT overflow
57 //for our purpose 250 ms of counting
58 ISR(WDT_vect)
59 {
60   ovf_wdt++;
61   if(ovf_wdt==14)
62   {
63     awake=1;         //Wake up and exit from sleep while(1)
64     ovf_wdt=0;
65
66   }
67
68 }
69
70
71 //if external interrupt comes
72 ISR (INT0_vect)
73 {
74 }
75
76
77 // if Timer/Counter0 overflow flag
78 ISR(TIMER0_OVF_vect)
79 {
80   count_ovf_t0++;              // count number of Counter0 overflows
     the external signal
81 }
```

```
82
83
84 // if Timer/Counter0 overflow flag
85 ISR(TIMER1_OVF_vect)
86 {
87     count_ovf_t1++;
88
89     if (count_ovf_t1 >= measurement_window_expected_duration)
90         PCMSK2 |= (1 << PCINT20); // Look for next input signal
       rising edge (end measurement).
91 }
92
93 //if Timer/Counter2 output compare
94 ISR (TIMER2_COMPA_vect)
95 {
96   //stop counter and avoid OC overflows
97   TCCR2B = TCCR2B & ~7;
98   TIMSK2 &= ~(1 << OCIE2A);
99
100 }
101
102 /*
103  * This interrupt is enabled ONLY when we are looking for input
104  * signal edges, either:
105  * - to start a new measurement
106  * - to stop an ongoing measurement.
107  *
108  * NOTE: during a measurement, this interrupt is ALWAYS disabled.
109  */
110 ISR(PCINT2_vect)                        // Port d, PCINt20
111
112   {  char statepin = 0;
113
114     // No raising edge of input signal
115     statepin = PIND;
116     if ((statepin & 0x10) != 0x10)
117         return;
118
119     // Raising edge of input signal.
120     //
121
122     // Start a new measurement.
123     if (f_measurement_running == 0) {
124
125
126
127
128         // Disable PCint until the end of measurement window.
129         PCMSK2 &= ~(1 << PCINT20);
```

```
130
131         // Reset counters and their overflow counters.
132         TCNT1 = 0;
133         TCNT0 = 0;
134         count_ovf_t0 = 0;
135         count_ovf_t1 = 0;
136
137         // Start measuring the time.
138         TCCR1B |= (0 << CS12) | (0 << CS11) | (1 << CS10); //no
    prescaling internal clock source
139
140         // Connect input signal as Timer0 clock (rising edge of
    external signal, no prescaling)
141         TCCR0B |= (1 << CS02) | (1 << CS01) | (1 << CS00);
142
143         // Enable counter overFlow interrupts.
144         TIMSK1 |= (1 << TOIE1);
145         TIMSK0 |= (1 << TOIE0);
146
147         f_measurement_running = 1;
148     }
149     // End of the measurement.
150     else {
151
152
153     // Disable pin change interrupts.
154     PCMSK2 &= ~(1 << PCINT20);
155
156     // Stop counters.
157     TCCR1B = TCCR1B & ~7;
158     TCCR0B = TCCR0B & ~7;
159
160     // Disable counter overflow interrupts.
161     TIMSK1 &= ~(1 << TOIE1);
162     TIMSK0 &= ~(1 << TOIE0);
163
164         // Get remainders from counters.
165         remaineder_clocks = TCNT1;
166         remaineder_input_periods = TCNT0;
167
168     f_measurement_running = 0;
169
170     SMCR &= ~(1 << SE);          //avoid IDLE sleep after a
    measurement window
171     }
172
173 }
174
175 int
```

```
176 main(void)
177 {
178   cli();
179 
180    // Measurement window in terms of 16 bit counter overflows (
    TIMER1)
181    measurement_window_expected_duration = 15;
182 
183    TCCR0A = 0;                          // reset timer/counter0 control
     register A
184    TCCR0B = 0;                          // reset timer/counter0 control
     register A
185    TCNT0 = 0;                           // counter value = 0
186 
187    // timer1 setup / is used for frequency measurement gate time
    generation with 16MHZ/no prescaling
188    TCCR1A = 0;
189    TCCR1B = 0;
190 
191    f_measurement_running = 0;
192 
193    //pinchange interrupt
194    DDRD &= ~(1 << DDD4);                // Set pd4/pcint20 as input
195    PCICR |= 0b00000100;                 // turn on port d interrupts (
    PCINT[23:16])
196 
197 
198 
199  //sleep controls
200  DDRD &= ~(1 << DDD2);          //PD2 is input
201  PORTD |= (1 << PORTD2);          //pullup for PD2
202 
203  DDRB |= (1 << DDB5);           //PB5 is output
204  PORTB &= ~(1 << PORTB5);         //PB5 set to 0, led off
205 
206  DDRB |= (1 << DDB1);           //Set PB1 as output, this is the 555
     supply
207  PORTB &= ~(1 << PORTB1);         //Switch OFF 555 supply
208 
209  DDRB |= (1 << DDB4);           //PB4 is output, connected to a LED
     to signal the set_ref
210  PORTB &= ~(1 << PORTB4);
211 
212  //Select the edge for External interrupt INT0
213  //00 low level  01 any change 10  falling edge  11 rising edge
214  EICRA &= ~(1 << ISC00);
215  EICRA |= (1 << ISC01);
216  //Is sensible only to falling edges
217 
```

```
218   EIMSK |= (1 << INT0);          // Turns on INT0
219
220   sleep_mode_func_pwrdwn();       //Power_down mode as sleep (waiting
        extint INT0)
221
222   //Switch off ADC, analog comparator and stuff not need
223   DIDR0 = 0xFF;               //ADC buffers OFF
224   DIDR1 = 0xFF;
225   ADCSRA &= ~(1 << ADEN);          //ADC OFF
226   ADMUX &= ~(1 << REFS0);          //Voltage reference ADC OFF
227   ADMUX &= ~(1 << REFS0);
228   ACSR |= (1 << ACD);            //Analog comparator OFF
229   PRR = 0x87;                 //Remove supply to unused modules
230
231   sei();
232
233   set_ref();
234
235     while (1) {
236
237     sleep_cpu();              //Go to sleep Power_down
238
239     EIMSK &= ~(1 << INT0);          //Avoid INT0 during measurements
240
241     PCMSK2 &= ~(1 << PCINT20);       //Avoid PCINT20 before effective
      measurement start
242
243     c_low = 0;                //The capacitance is not low
244     flag_no_presence = 0;        //This flag becomes 1 when one
      measurement states no presence. Two repeated meas without presence
       will exit(c_low=1)
245     PORTB |= (1 << PORTB5);         //LED ON
246     //START OF MEASUREMENT WINDOW
247     while( c_low == 0){
248
249     PORTB |= (1 << PORTB1);         //Switch ON 555 supply
250
251
252     sleep_mode_func_pwrsave();       //Power_save mode as sleep (for
      555 stabilization , 2 ms)
253
254     //SETTLING 555 2ms
255     counter_2_func();          //Run Timer2 (Asyn) that wakes up
      microcontroller after 2 ms
256     sleep_cpu();              //Go to sleep Power_save
257
258
259     //MEASUREMENT 44ms
```

```
260     sleep_mode_func_idle();         //Idle mode as sleep (for
    measurement window)
261         f_measurement_running = 0;       //Measurement will start with
     first 555 rising edge
262
263         PCMSK2 |= (1 << PCINT20);       //Turn ON PCINT20, start
    measuring
264
265         while (f_measurement_running == 0) sleep_cpu(); // Wait for
    the actual START of the new measurement in IDLE to save power
266
267     while (f_measurement_running == 1) sleep_cpu(); // Wait for the
    actual END of the new measurement in IDLE to save power
268
269         // Measurement completed.
270
271         // Calculate input signal frequency.
272         //input_signal_frequency = (float)total_input_periods / ((
    float)F_CPU * (float)total_clocks_during_measurement);
273
274     // Total time for all input signal periods.
275     total_clocks_during_measurement = count_ovf_t1 * 65536 +
    remaineder_clocks;
276
277     // Full periods of input signal.
278     total_input_periods = count_ovf_t0 * 256 +
    remaineder_input_periods;
279
280     //Calculate frequency, interm is for avoiding overflow
281
282     //interm = (total_input_periods * 16000);
283     //f555 = (interm / total_clocks_during_measurement);
284
285     //Send data via serial or Xbee...can send remaineder_clocks and
    remaineder_input_periods and calc f555 on basestation
286
287     remaineder_clocks = 0;
288     remaineder_input_periods = 0;
289
290     PORTB &= ~(1 << PORTB1);        //Switch OFF 555 supply
291
292     //Exhausted capacitance algorithm or 20 measurements
293
294     //temp++;
295     //if(temp==20){ c_low=1; temp=0;} //Now, we are doing 10
    measurements per extint, allows to extit from measurement windows
296
297     if( f555 < 1000) {          //No presence is detected in a
    measurement
```

```
298
299       if( flag_no_presence == 1){    //No presence for 2 consecutive
    measurements
300         c_low = 1;           //Flag to exit from meas. period and go
    to deep sleep
301       }
302       flag_no_presence = 1;      //Set flag to signal first no
    presence
303     }
304
305     else {
306       flag_no_presence = 0;      //Avoid that non-consecutive no
    presence are detcted
307     }
308
309     //SOFT SLEEP + WDT 256ms
310
311     sleep_mode_func_pwrdwn();      //Power_down mode as sleep (waiting
    WDT with 14 ovf)
312     watchdog_set_func();         //Set the Watchdog timer to start
    counting time
313     awake=0;                 //Flag to keep the micro sleep until N WDT
    ovfs
314     while(awake==0) sleep_cpu();     //Go to sleep Power_down
315     wdt_disable();              //Disable WDT after waking-up
316
317     }
318
319     PORTB &= ~(1 << PORTB5);       //LED OFF
320     EIMSK |= (1 << INT0);         //Wait INT0, extint
321    }
322
323
324 }
325
326 //function to set reference for exhausted algorithm, possible to
    recall it to avoid drift effect
327 //not needed if using N consecutive measurements
328 static void set_ref(){
329
330   float aver=0;
331   float sum=0;
332   float ref=0;
333   int init=0;
334
335   _delay_ms(5000);             //Initial wait to make person ready at
    1.5 m
336   PORTB |= (1 << PORTB1);
337   _delay_ms(2);
```

```
338
339   PORTB |= (1 << PORTB4);           //LED to signal initialization
        phase
340   while(init <21){
341
342     init++;
343     f_measurement_running = 0;       //Measurement will start with
        first 555 rising edge
344
345     PCMSK2 |= (1 << PCINT20);       //Turn ON PCINT20, start measuring
346
347     while (f_measurement_running == 0); // Wait for the actual START
        of the new measurement in IDLE to save power
348
349     while (f_measurement_running == 1); // Wait for the actual END of
         the new measurement in IDLE to save power
350
351     total_clocks_during_measurement = count_ovf_t1 * 65536 +
        remaineder_clocks;
352
353     total_input_periods = count_ovf_t0 * 256 +
        remaineder_input_periods;
354
355     interm = (total_input_periods * 16000);
356
357     f555 = (interm / total_clocks_during_measurement);
358
359     sum= sum + f555;
360   }
361   PORTB &= ~(1 << PORTB1);
362   PORTB &= ~(1 << PORTB4);
363   aver=sum/init;
364
365   ref = aver;
366
367 }
368
369
370 static void sleep_mode_func_pwrdwn()
371 {
372   set_sleep_mode(SLEEP_MODE_PWR_DOWN);   //select sleep mode POWER
        DOWN
373   cli();
374   sleep_enable();              //set SE bit in register
375   sei();
376 }
377
378 static void sleep_mode_func_pwrsave()
379 {
```

```
380   set_sleep_mode(SLEEP_MODE_PWR_SAVE);  //select sleep mode POWER
       SAVE
381   cli();
382   sleep_enable();              //set SE bit in register
383   sei();
384 }
385
386 static void sleep_mode_func_idle()
387 {
388   set_sleep_mode(SLEEP_MODE_IDLE);     //select sleep mode IDLE
389   cli();
390   sleep_enable();              //set SE bit in register
391   sei();
392 }
393
394 static void counter_2_func()
395 {
396   cli();
397   OCR2A = 15;                 //Output compare value
398   TCNT2 = 0;
399   TCCR2A |= (1 << WGM21);          // Set to CTC Mode Clear timer on
       compare
400   TIMSK2 |= (1 << OCIE2A);         //Set interrupt on compare match
401   TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20); //1024
       prescaling
402   sei();
403
404 }
405
406 static void watchdog_set_func()
407 {
408   cli();
409   wdt_reset();                //WDT reset
410   WDTCSR |= (1 << WDCE) | (1 << WDE);   //Ready to set prescaling
411   WDTCSR = 0x40;               //Prescaling to 16ms, for 300ms 18,75
       cycles
412   sei();
413 }
414
415 static void
416 reverse(char *str, int len)
417 {
418   int i = 0, j = len - 1, temp;
419   while (i < j) {
420     temp = str[i];
421     str[i] = str[j];
422     str[j] = temp;
423     i++;
424     j--;
```

```
425   }
426 }
427 // Converts a given integer x to string str[].  d is the number
428 // of digits required in output. If d is more than the number
429 // of digits in x, then 0s are added at the beginning.
430 static long
431 intToStr(long x, char str[], int d)
432 {
433   int i = 0;
434   while (x) {
435     str[i++] = (x % 10) + '0';
436     x = x / 10;
437   }
438
439   // If number of digits required is more, then
440   // add 0s at the beginning
441   while (i < d)
442   str[i++] = '0';
443
444   reverse(str, i);
445   str[i] = '\0';
446   return i;
447 }
448
449 // Converts a floating point number to string.
450 static void
451 ftoa(float n, char *res, int afterpoint)
452 {
453   // Extract integer part
454   long ipart = (long)n;
455
456   // Extract floating part
457   float fpart = n - (float)ipart;
458
459   // convert integer part to string
460   long i = intToStr(ipart, res, 0);
461
462   // check for display option after point
463   if (afterpoint != 0) {
464     res[i] = '.';           // add dot
465
466     // Get the value of fraction part upto given no.
467     // of points after dot. The third parameter is needed
468     // to handle cases like 233.007
469     fpart = fpart * pow(10, afterpoint);
470     intToStr((long)fpart, res + i + 1, afterpoint);
471   }
472 }
```

Bibliography

[1] A. Ramezani Akhmareh, M. T. Lazarescu, O. Bin Tariq, and L. Lavagno, "A tagless indoor localization system based on capacitive sensing technology," *Sensors*, vol. 16, no. 9, 2016.

[2] J. v. Wilmsdorff, F. Kirchbuchner, B. Fu, A. Braun, and A. Kuijper, "An experimental overview on electric field sensing," *Journal of Ambient Intelligence and Humanized Computing*, Jun 2018.

[3] T. Kivimäki, T. Vuorela, P. Peltola, and J. Vanhala, "A review on device-free passive indoor positioning methods," *International Journal of Smart Home*, vol. 8, no. 1, pp. 71–94, 2014.

[4] A. Bellinaso, "Machine learning algorithms for human localization using long distance capacitive and infrared sensors / aurora bellinaso ; supervisors luciano lavagno, mihai teodor lazarescu," 2017.

[5] M. Youssef, M. Mah, and A. Agrawala, "Challenges: device-free passive localization for wireless environments," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pp. 222–229, ACM, 2007.

[6] R. Gambotto, "Analog front-ends for long range capacitive sensing for iot indoor human localization / roberto gambotto ; supervisors luciano lavagno, mihai teodor lazarescu," 2017.

[7] H. Prance, P. Watson, R. J. Prance, and S. T. Beardsmore-Rust, "Position and movement sensing at metre standoff distances using ambient electric field," *Measurement Science and Technology*, vol. 23, p. 115101, oct 2012.

[8] X. Tang, J. Liang, Y. Wang, and S. Mandal, "Indoor occupancy awareness and localization using passive electric field sensing," in *2018 IEEE SENSORS*, pp. 1–4, Oct 2018.

[9] T. Grosse-Puppendahl, C. Holz, G. Cohn, R. Wimmer, O. Bechtold, S. Hodges, M. S. Reynolds, and J. R. Smith, "Finding common ground: A survey of capacitive sensing in human-computer interaction," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, (New York, NY, USA), pp. 3293–3315, ACM, 2017.

[10] J. Iqbal, M. T. Lazarescu, A. Arif, and L. Lavagno, "High sensitivity, low noise

front-end for long range capacitive sensors for tagless indoor human localization," in *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, pp. 1–6, Sep. 2017.

[11] J. Smith, T. White, C. Dodge, J. Paradiso, N. Gershenfeld, and D. Allport, "Electric field sensing for graphical interfaces," *IEEE Computer Graphics and Applications*, vol. 18, pp. 54–60, May 1998.

[12] P. Poolad, "Optimization of capacitive sensor front-end for indoor human localization and identication / pooya poolad ; supervisors luciano lavagno, mihai teodor lazarescu," 2017.

[13] kuku Tena Nigatu, "Tagless long/distance capacitive sensors for indoor human localizations / kuku tena nigatu ; supervisors luciano lavagno, mihai teodor lazarescu," 2018.

[14] G. Cohn, S. Gupta, T.-J. Lee, D. Morris, J. R. Smith, M. S. Reynolds, D. S. Tan, and S. N. Patel, "An ultra-low-power human body motion sensor using static electric field sensing," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, (New York, NY, USA), pp. 99–102, ACM, 2012.

[15] "Texas instruments - lp2951 datasheet." Available at http://www.ti.com/lit/ds/symlink/lp2951-n.pdf [Accessed 05.01.2019].

[16] "Microchip - mcp6043 datasheet." Available at http://ww1.microchip.com/downloads/en/DeviceDoc/21669D.pdf [Accessed 05.01.2019].

[17] "Microchip - atmega328p datasheet." Available at http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf [Accessed 15.10.2018].

[18] "Texas instruments - ne555 datasheet." Available at http://www.ti.com/lit/ds/symlink/ne555.pdf [Accessed 08.03.2019].

[19] "Texas instruments - lmc555 datasheet." Available at http://www.ti.com/lit/ds/symlink/lmc555.pdf [Accessed 08.03.2019].

[20] jjbeard on Wikipedia, the free encyclopedia, "Circuit diagram of a standard 555 astable circuit." [Online; accessed January 22, 2019], available https://commons.wikimedia.org/wiki/File:555_Astable_Diagram.svg, 2006.

[21] J. Iqbal, M. T. Lazarescu, O. B. Tariq, and L. Lavagno, "Long range, high sensitivity, low noise capacitive sensor for tagless indoor human localization," in *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pp. 189–194, June 2017.

[22] F. H. C. Ulrich, "Front-end for long range capacitive sensor / fotso hondjie christian ulrich ; supervisors luciano lavagno, mihai teodor lazarescu," 2019.