ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

SCUOLA DI INGEGNERIA ED ARCHITETTURA
Corso di Laurea Magistrale in
Ingegneria Elettronica e Telecomunicazioni per l'Energia

CHARACTERIZATION AND INTERFACING OF A CAPACITIVE RAIN SENSOR

Elaborato in
Elettronica per l'elaborazione analogica dei segnali LM

Tesi di Laurea di:
ROBERTO NERI

Relatore:
Prof. Ing.
SERGIO CALLEGARI

SESSIONE II
ANNO ACCADEMICO 2023–2024

KEYWORDS

Capacitive Sensing

Rain Monitoring

Time Encoding

Smart Sensors

Characterization

Contents

Abstract

Environmental sensing is a crucial part of recent home automation systems. This project exploits a thick-film humidity capacitive sensor manufactured by AUREL S.p.A. and used by its customers mainly as a rain sensor. Currently, AUREL s.p.a. sells this sensor as a standalone product to its customers, who design their own interfaces. The work presented in this thesis regards the characterization of the sensor and the design of interfaces that could appeal the company customers and englarge its client portfolio. The input was to design and compare different solutions, also characterizing the sensor+interface combination and its response to changes in the environmental conditions. Five different solutions have been designed, each with its own advantages for various kinds of applications. Some of the proposed solutions are fully analog, enabling maximum flexibility in terms of application. Others let the device operate as a smart sensor, with a digital interface, prioritizing the simplicity in the application. These designs employ a microcontroller unit (MCU). For what concerns the transduction mechanism, time encoding is employed, with variations, in three of the proposed solutions. Generally speaking, the project involved the study of multiple transduction mechanisms, prototyping a pre-industrialization, with the design of both schematics and the printed circuit boards (PCBs) as well as the testing and characterizing each solution. Because the operating conditions, in terms of temperature, supply voltages, and quality of interconnection (mainly expressed by the distance between the sensor and the circuit) can influence the data gathered from the sensor, each of these effect had to be carefully characterized to understand the magnitude of its influence on the measurements. Ultimately, testing data enabled a classification of the proposed solutions via their pros and cons, which may in future provide guidance to the company's customers to identify the interfacing solution best suited for their needs.

Introduction

In recent years, the proliferation of home automation systems has transformed residences into increasingly intelligent environments. Central to this evolution are sensors, which play a crucial role in smart homes by performing diverse functions such as security monitoring, environmental sensing, and health management. These sensors collect raw data that must be processed and transformed into formats that can be interpreted by MCU or human operators.

This thesis describes the characterization and interfacing of one such sensor, specifically a rain sensor. In the context of smart homes, the device might be employed in a garden irrigation system or to automatize the closure of windows, shade sails, patio screens, etc. Additionally, it may find application in weather stations, agriculture, environmental monitoring, vehicles, the automatic adjustment of speed limits in smart cities, and more. All these tasks involve making the sensor data available in a format suitable for computer or micro-controller system consumption. This work is specifically related to this readout process as well as the design, prototyping, pre-industrialization and validation of the electronic boards relevant to the task.

The presented material is the result of some months of practical experience and internship at Aurel S.p.A, the company producing the sensor. The proposed results are thus based not just on studies conducted at a feasibility level but rather on a hands-on experience on real-world devices, conducted side-to-side with those very same groups in charge of defining the sensor's physical and mechanical properties, its manufacturing steps, and its marketing related aspects, including the price point. Notably, the sensor on which the development activities have been based is a product available on the market. During the development work, the sensor itself was upgraded into a new model so that multiple sensor prototypes at different stages of refinement have been considered. This made the *characterization* activities particularly important. At the same time, merit factors that can often receive limited attention when working at a low technology readiness levels (TRLs), had to be given prominence. These included cost, simplicity, robustness, ease of application when handed to customers, ability to cope with harsh environments, stability over environmental variations, flexibility, and applicability to existing designs with minimal disruption, to mention a few.

Multiple sensing approaches had to be investigated and prototyped to this aim. This is not just to collect the information needed to select an optimal solution but also to offer a wide enough solution portfolio. Specifically, at least two different designs were the objective of the work: one characterized by a *raw* (fundamentally analog) output and offering maximum flexibility; another one presenting the sensing device as a *smart sensor*, delivering digital data through a standardized bus and a relatively high-level application interface.

As it will be better clarified further on, an aspect deserving significant attention is that to satisfy the requirements of a commercial product, the rain sensor used in this work is a *multi-sensor*. In other words, it hosts not just the sensing elements directly involved in rain detection but also additional ones. This is a frequent situation in many sensor-related applications where the probes need not just to deliver the quantities of direct interest but also to assure additional functionalities, which may involve items such as the assessment of the conditions for reliable operation, the measurement of additional environmental quantities, initialization or reset abilities, and so on. Particularly, the sensor at hand can detect rain by providing information on whether it is wet or not. This means that to detect the *arrival of rain* it must start from a *dry state*, so that the variation in the state of its surface *from dry to wet* can be read. Similarly, to detect the *end of rain*, one needs to observe the variation of its state *from wet to dry*. The problem here is an inherent asymmetry between the wetting and the drying. Because the rain directly affects wetting, this is a rapid transition. The other way round, if left to natural means, the drying would be too slow and too influenced by environmental conditions to offer a useful response. Therefore, the unit must be equipped with a device capable of artificially assuring rapid drying and periodic restoration of its dry state. In the adopted sensor, the physical approach employed for the drying involves evaporation powered by a heating resistor. Because the sensor is small, heat-induced drying can be practiced on an acceptable energy budget. As a result of this choice, the rain sensor also hosts a thermal actuator. In turn, this makes some form of thermal monitoring necessary so that the rain sensor also needs to host a thermal probe in the form of a negative temperature coefficient (NTC).

The characterization activities and the interface design described in this thesis thus involve all three elements present in the multi-sensor and consider their interactions. For instance, in addition to the *wetness-to-output* relationship, items such as the heater operation, drying profiles, and influence of temperature over wetness measurements need to be dealt with.

In the near future, some of the results presented in this thesis might become part of the commercial offer of the company where the internship took place, either as interfacing boards

for the sensor to be put on the market or, more likely, as a series of application notes for the customers of their sensor offer.

Anticipating some aspects of the sensor interfacing, it is worth noticing that the process of obtaining valuable data from raw variations in the properties of some passive sensors involves several critical steps, including the provision of some excitation and various forms of signal conditioning. Frequently, a key aspect is the adoption of some form of modulation, which can both simplify the readout of the quantity of interest and enhance robustness against noise during transmission. On the one hand, modulations allow forms of information representation where substantially analog data is encoded in the time-relationship between ”events” that characterize the signal evolution rather than in the signal amplitude. This is known as *time encoding* and can often simplify the acquisition of useful information at the digital level, saving *data converters* as counters can be adopted to measure time-related information such as *delays*, *frequencies*, *periods*, etc. On the other hand, time-encoding decouples the information content from the signal amplitude, reducing the impact of attenuation, fading, noise and interference on the correct propagation of the information content. This is particularly important in scenarios where the sensor and the MCU are physically separated, as in many practical applications.

Given the variety of sensors available, multiple transduction techniques exist, each with its own set of advantages and drawbacks. A skilled designer must carefully evaluate these techniques to select the most suitable combination for the intended application. Factors such as cost, production time, component availability, and overall efficiency are pivotal in this decision-making process. Additionally, considerations related to noise immunity and the physical dimensions of the sensor also play a significant role in the design.

The primary objective of the activities described in this thesis was to develop a sensor interface hosted on a minimally sized PCB and based on commercial components, capable of operating reliably with minimal maintenance and featuring a maximal connection distance of approximately ten centimeters from the sensing device. The design must ensure resilience on par with that of the sensor device that has already been optimized over several years to endure challenging environmental conditions.

The structure of this thesis is organized to provide a comprehensive understanding of the sensor’s development and evaluation process.

Chapter 1 introduces the sensor by offering an overview of its functionality and the underlying transduction mechanisms. This chapter places particular emphasis on time encoding, detailing

why this method was chosen and how it enhances the sensor's performance for the given application.

Moving forward, Chapter 2 delves into the various design approaches considered during the development process. It explains the rationale behind each approach and provides insights into the decision-making criteria that guided the design choices.

Chapter 3 shifts focus to the hardware aspects of the sensor's circuit. It presents the design choices made when moving from the theoretical design to the PCB one.

In Chapter 4, the thesis discusses the different elements that could disrupt the sensor's reading. It explains what those phenomena are, how they could affect such reading, and what tests were done in order to check how much of an influence those parameters effectively have on the reading itself.

Finally, Chapter 5 encompasses the characterization that has been done, specifically the coverage and the temperature characterization, reporting all the results. In the end, a little focus on an experiment to try and replicate the effects of the morning dew on top of the sensor will be explored.

Together, these chapters provide a thorough exploration of the sensor's design, implementation, and performance evaluation, culminating in a well-rounded conclusion on its overall efficacy.

1 Primitive sensor, transduction mechanisms, time encoding

This chapter introduces the sensor used in this study, providing a comprehensive overview of its key features. It also explores various transduction mechanisms emphasizing the rationale for selecting time encoding as the primary measurement method. By examining these aspects, we lay the foundation for understanding the sensor's operation and its applications in environmental monitoring.

1.1 Sensor features

As already mentioned in the introduction, what is indicated as *the sensor* is, in fact, *a small collection of subsystems*, entailing the rain sensor proper, a heater, and a thermal sensor.

The actual rain sensor is based on a *capacitive approach*. As shown in the diagram in Figure 1.1, two conductive regions are patterned over an insulating piece of supporting material and covered by a thin, protective insulating coating. When the sensor is dry, the capacitance between these regions is low because the facing surface between them is modest, the distance is relatively large, and the dielectric is given by a mixture of the coating and air, both having a low dielectric constant. When the sensor is wet, water plays the role of an additional conductor placed over the two *patterned regions*. Consequently, the capacitance in between the regions is now given by the series of two capacitors: one involving the first region and the wet surface and the other one involving the wet surface and the second region. Both the involved capacitance values end up as relatively large because the facing surface is now large, and the distance between the patterned regions and the wet spot is modest (being substantially determined by the thickness of the coating). The resulting capacitance between the patterned regions is thus larger than in the dry state.

The *heating element* is basically a resistor. It is obtained as a layer of conducting material

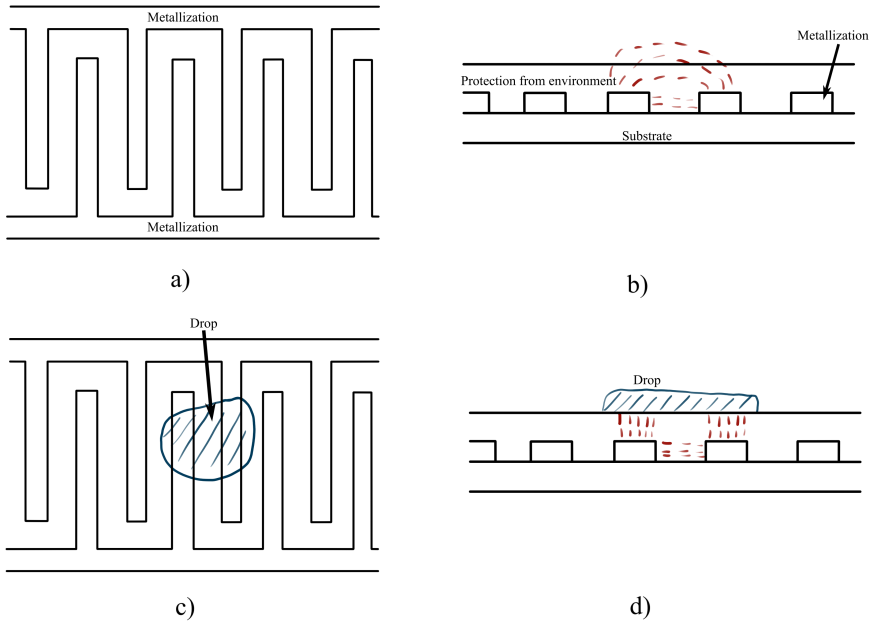


Figure 1.1: Operating principle of the rain sensor. In (a), top view of the capacitive sensor, with two conductive regions patterned on an insulating substrate in an interleaved way. In (b) a section view of the sensor, showing the substrate, the conductive (metalized) regions, and an insulating, protective coating. In this picture a sketch of the electric field lines is also proposed, limited to a portion of the two conductive regions. Most of the field lines close via long paths through air. In (c), the same top view as in (a), but now with a drop over the sensor. In (d), the same section view as in (b), again with the drop. Assuming that the drop is conductive, now the electric field lines can close through the drop, assuring a much higher capacitance. The figure is purely illustrative and does not reflect the geometries of the real sensor.

patterned on the same piece of supporting material, on the opposite side than the capacitive sensor, and again protected by a *suitable coating*. To provide uniform heating, the resistor needs to span a similar surface to that of the capacitive sensor.

Finally, the *thermal probe* is again a resistor whose resistance varies according to temperature. A commercial NTC is employed, mounted over the supporting material.

Figure 1.2 show a photograph of the real sensor that features six pins, each serving a specific function related to the three functionalities outlined above:

- pins one and six are connected to the capacitive element;
- pins two and five are connected to the resistive element used as a heater;
- pins three and four are connected to the NTC that requires external soldering.

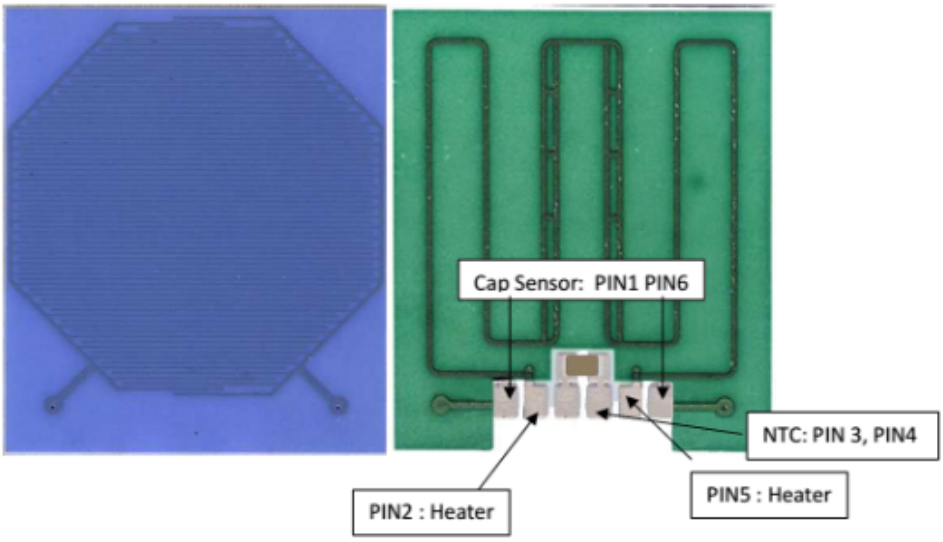


Figure 1.2: Photograph of the first (MKI) revision of the sensor employed in this work. The image on the left shows the side (blue) on which the capacitor used as a sensing element is visible. The image on the right side shows the other side (green), on which the heating element is present. The NTC component is visible right above the corresponding pins.

As anticipated in the Introduction, multiple prototype sensors have been employed during the development work, corresponding to different revisions of the hardware design. In this document, they are identified as MKI (mark 1), MKII (mark 2), etc. The illustration refers to the first revision but is, in fact, sufficiently representative of all of them.

For *optimal performance*, the blue side of the sensor must face upward during operation. This orientation is critical for achieving accurate and consistent readings, as it ensures that the sensitive surface is exposed to the environment where it can interact effectively with moisture. The heating element remains consequently on the bottom. When powered by a 12 V supply, this resistor quickly raises the surface temperature, facilitating the drying process. Figure 1.3 illustrates the time required to heat the sensor from an ambient temperature of 25 °C when powered at 12 V.

The sensor must be mounted at an angle (with 30° being optimal) to ensure *accurate measurements*. This installation prevents water from pooling on the surface, which could impede evaporation and lead to inaccurate readings.

An essential component of the sensor system is the NTC resistor. This *temperature-sensitive element*, which requires separate mounting, is sourced from TDK and has a resistance of 100 kΩ at 25 °C [7]. The NTC's resistance decreases as temperature increases, allowing it to provide *crucial temperature readings*. These readings are vital for ensuring that the sensor's

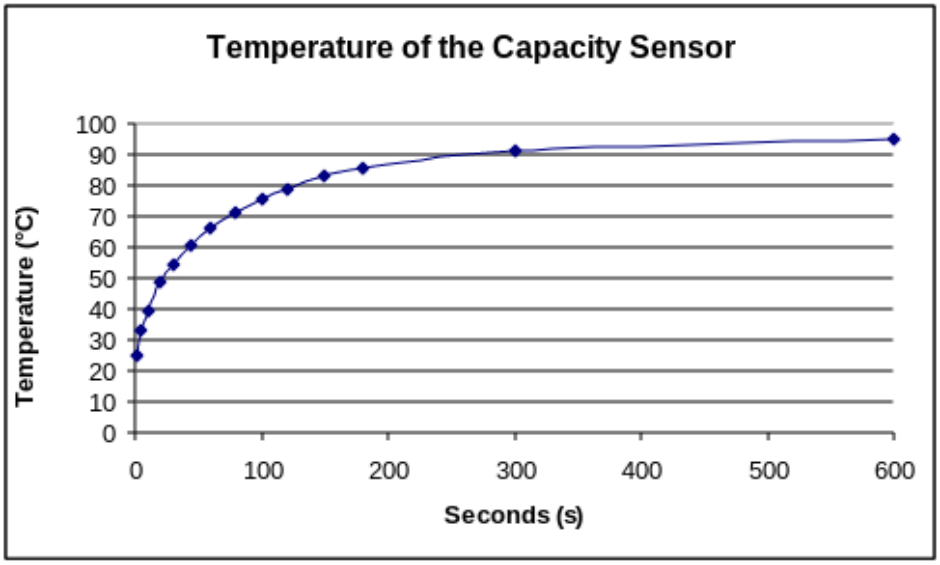


Figure 1.3: Temperature increase profile achieved by the sensor heating element. The plot refers to a sensor located in an environment at 25 °C. After a few hundreds seconds, the temperature stabilizes close to 100 °C (actual final value depending on the ambient temperature and amount of humidity in the air), sufficient to rapidly achieving evaporation of the droplets on the sensor without damaging the sensor itself.

surface is properly dried before taking measurements. However, the NTC response is non-linear, necessitating the use of *interpolation methods* (such as the *Steinhart-Hart equation* [5]) to accurately interpret the temperature data. All specifications and performance characteristics detailed in this section are supported by datasheets and reference material provided by AUREL s.p.a.

For what concerns the main capacitive sensor, the major difference among the sensor revisions used through the development work is in the range of capacitive values. The batch of 10 MKIII sensors used in the final portion of the development work exhibited capacitive values ranging from 81 pF when completely dry to 488 pF when fully covered in (distilled) water.

1.2 Transduction Mechanisms

Transduction mechanisms play a crucial role in converting physical phenomena, such as changes in capacitance, into *measurable electrical signals*. These signals can then be processed by electronic systems such as MCUs, and central processing units (CPUs) to perform various tasks. Capacitive sensing [8, 4, 10], in particular, is a widely used technology that detects changes

in capacitance to measure parameters such as proximity, moisture, and touch. This section explores different transduction techniques, each with its unique advantages and applications.

1.2.1 Frequency Measurement

Frequency measurement is a sophisticated method of capacitance detection that integrates the capacitive element of a sensor into an oscillator circuit. The circuit oscillates at a frequency determined by the values of its components, including the capacitance. As the capacitance changes, the oscillation frequency alters, providing an indirect means of measurement. For example, in an *LC* (inductor-capacitor) circuit, the *resonant frequency* is determined by the inductance (L) and capacitance (C) according to the formula:

$$f = \frac{1}{2\pi\sqrt{L \cdot C}} \tag{1.1}$$

This relationship allows for precise measurement of capacitance changes by monitoring the circuit's frequency. Alternatively, in an *RC* (resistor-capacitor) circuit, the frequency of oscillation is related to the *time constant* τ of the circuit, which is the product of resistance (R) and capacitance (C):

$$\tau = R \cdot C \tag{1.2}$$

In an *RC* oscillator, as capacitance changes, the time constant τ changes as well, leading to a shift in the oscillation frequency. Frequency measurement offers several advantages, particularly its high sensitivity and stability with respect to external influences. Oscillator circuits are typically characterized by good short- and long-term stability and can operate accurately in environments with high *electrical noise*, making this method ideal for applications where precision is critical. However, implementing frequency detection still requires careful design of the oscillator circuit and consideration of factors such as temperature stability and component tolerances. These complexities may present challenges in certain applications, particularly where simplicity and cost-effectiveness are priorities.

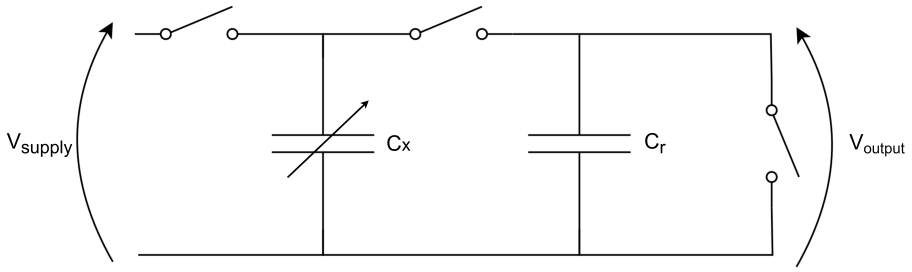


Figure 1.4: Conceptual architecture of a charge transfer circuit. Practical implementations might require additional elements to reset the charge on the feedback capacitor.

1.2.2 Charge Transfer

Charge transfer is a powerful technique used to measure capacitance, relying on the principle of transferring charge between capacitors. This method deploys a similar operating principle as a *switched capacitor* circuit, where the focus is on the charge transfer process itself rather than equivalent resistance. In a typical charge transfer circuit, a reference capacitor C_r is used in conjunction with the target capacitor C_x, which is the capacitor being measured. The process begins when both capacitors are fully discharged. The target capacitor is then charged to a known voltage, after which it discharges into the reference capacitor, transferring its charge.

A key aspect of this method is that the reference capacitor must typically be significantly larger than the target capacitor. This size difference ensures that the reference capacitor can accumulate charge over multiple cycles of charging and discharging the target capacitor. The process is repeated until the reference capacitor is fully charged, with the number of cycles required being directly proportional to the value of the target capacitor. For example, using the circuit in Figure 1.4, the value of C_x can be determined using the formula:

$$C_x = \frac{k}{N} \tag{1.3}$$

where k is a constant that depends on the reference capacitor and the circuit design, and N is the number of *charge-discharge cycles* required to charge the reference capacitor to a predetermined threshold [6]. Charge transfer circuits are relatively simple to implement and do not require complex components. However, the accuracy of the measurement depends on the stability of the reference capacitor, particularly with respect to temperature changes. Any variation in the reference capacitor's value due to temperature fluctuations can introduce errors in the measurement, making it important to select components with high stability and low temperature coefficients, which can be costly.

1.2.3 Charge Amplifier

A charge amplifier operates by directly measuring the *charge accumulated* on the capacitor and converting this charge into a voltage signal. The fundamental relationship governing this process is given by:

$$Q = C \cdot V \tag{1.4}$$

where Q is the charge on the capacitor, C is the capacitance, and V is the voltage across the capacitor.

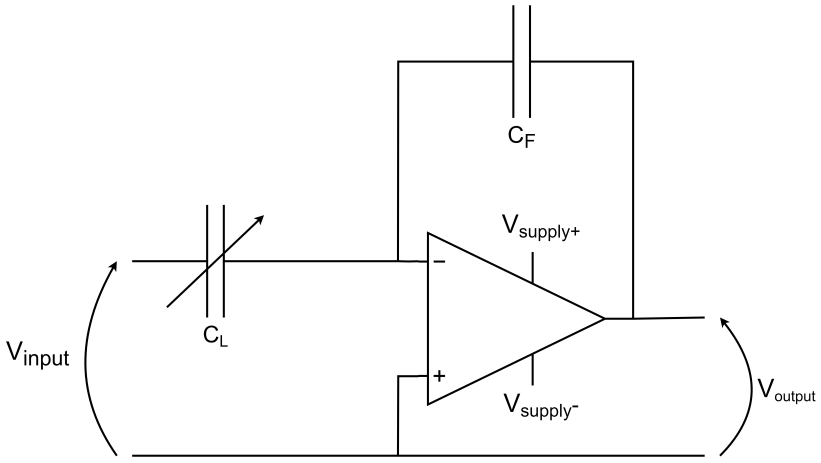


Figure 1.5: Schematic of a charge amplifier circuit

The typical configuration of a charge amplifier (Figure 1.5) includes a feedback capacitor C_F connected between the output and the inverting input of an operational amplifier (op-amp). The inverting input is also connected to the input capacitance C_L, which varies in response to the sensed object or condition. When there is a change in the input voltage, the output voltage V_{out} of the charge amplifier also changes by an amount as large as:

$$\Delta V_{\text{out}} = -\frac{\Delta Q}{C_F} \tag{1.5}$$

Because the charge ΔQ is proportional to the input capacitance C_L and the input voltage variation ΔV_{in}, the effect on the output voltage can also be expressed as:

$$\Delta V_{\text{out}} = -\frac{C_L}{C_F}\Delta V_{\text{in}} \tag{1.6}$$

Assuming that the operation starts with a reset so that the final value of V_{out} is ΔV_{out}, any change in the input capacitance C_F will result in a proportional change in the output voltage V_{out}, which can then be measured and processed. Charge amplifiers are particularly useful in applications requiring *high sensitivity and precision*. Additionally, they are less susceptible to noise compared to other measurement techniques, as the charge signal is amplified directly before any significant noise can affect it. Still, the design and implementation of a charge amplifier require careful consideration of factors such as the feedback capacitance, the operational amplifier (OpAmp)'s characteristics, and the overall circuit stability. The choice of OpAmp is particularly critical to ensure that the amplifier operates within the desired frequency range and with minimal noise.

1.2.4 RC Time Constant

The RC time constant method is a straightforward yet effective technique that leverages the charging and discharging behavior of a capacitor in an RC circuit, as shown in Figure 1.6, where R is the resistance and C is the capacitance.

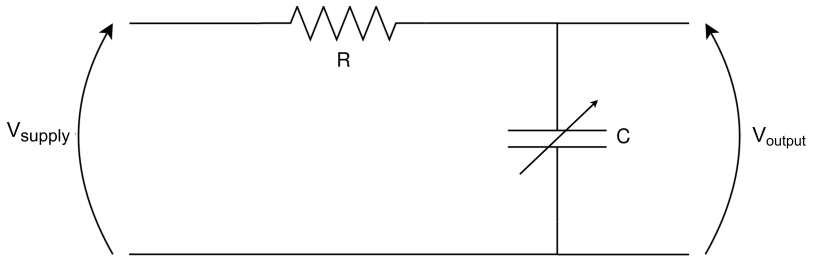


Figure 1.6: Schematic of a basic RC filter circuit

The time constant τ of an RC circuit is defined as:

$$\tau = R \cdot C \tag{1.7}$$

This time constant indicates how quickly the capacitor charges or discharges through the resistor. In a typical application, the capacitor in the RC circuit is charged to a specific voltage and then allowed to discharge through the resistor. For instance, consider a simple RC circuit where a capacitor is charged through a resistor until it reaches a certain voltage V_{th}. The time required for this process is directly proportional to the capacitance and to the value of τ. If the value at which the capacitor is charged is around two-thirds of the nominal value, then $t \approx \tau$ so the capacitance can be determined with the formula:

$$C = \frac{t}{R} \tag{1.8}$$

Where t in this equation represents the measured time. This method is particularly advantageous in situations where *simplicity* and *cost-effectiveness* are important. The circuit components required for the *RC* time constant method are minimal, typically consisting of a resistor, a capacitor, and a voltage source. Moreover, the *RC* time constant method is robust against electrical noise, as the measurement is based on time intervals rather than voltage or frequency, which can be more easily affected by noise. However, the accuracy of the measurement depends on the precision of the resistor and the stability of the capacitor. Any variations in these components, such as temperature-induced changes in resistance or capacitance, can affect the measurement.

1.2.5 Time Encoding

Time encoding is a highly effective technique for transducing capacitance changes into measurable signals by converting the variations in capacitance into time intervals, which can be measured using digital circuits. In low bandwidth applications where capacitance changes are often slow, there is no need for high-speed processing or complex electronics. Time encoding excels in these scenarios because it does not require fast components, making it a cost-effective solution. Time encoding focuses on time intervals that are less susceptible to such interference; transduction methods based on frequency variations, such as the ones presented in Section 1.2.1, could also be considered a type of time encoding. This characteristic makes this technique particularly valuable in remote sensing applications. Time encoding also offers significant advantages in terms of signal modulation. By converting capacitance changes into a time-based signal, it allows for transmission over long distances without degradation. The time-encoded signal remains intact even in less-than-ideal conditions, ensuring that the data is accurately received and processed. Since the time intervals are measured digitally, they are less affected by changes in temperature than analog signals, which can drift or vary with temperature. This stability makes time encoding a reliable choice for applications that require consistent performance across a range of environmental conditions. In summary, time encoding's precision, simplicity, and reliability make it an optimal choice for a wide range of applications, from *environmental monitoring* to *industrial sensing*, where accuracy and durability are essential. Its ability to maintain signal integrity over long distances and in varying environmental conditions makes it particularly well-suited for *remote sensing applications*.

The selection of time encoding as the primary measurement method for this study is based on its numerous advantages, including:

- cost-effectiveness in low bandwidth applications;
- robustness against electrical noise and environmental interference;
- ability to transmit signals over long distances without degradation;
- stability across varying temperature conditions;
- simplicity of implementation.

These factors combined make time encoding an ideal choice for the sensor system described in this chapter. It provides a reliable and efficient means of measuring capacitance changes in environmental monitoring applications.

2 Comparison of Different Measurement Approaches

This chapter introduces and describes the five techniques chosen for the sensing applications. It explains the design choices at hand, discussing their advantages, disadvantages, working principles and selection rationale.

2.1 Oscillator based on a timer integrated circuit (IC)

The first circuit under consideration is an oscillator using the *TS555* IC, manufactured by STMicroelectronics [12]. The 555 is a flexible *timer* that can be employed for a variety of purposes, including application as a proper timer, as a delay block, and as an oscillator. It is one of most popular timing ICs, thanks to its convenience and low price. Even if has been originally introduced in the '70s it is still in wide use, thanks to the fact that it has historically gone through several revisions and improvements (including the upgrade from the original bipolar techology to more mordern ones) and that it is proposed with substantial compatibility by various IC manufacturers.

When employed as an *astable multivibrator* (i.e., a block that remains in constant oscillation, never reaching a stable equilibrium point), the oscillation is based on *first-order transients* whose features depend on the value of an external capacitor. As a consequence, the oscillation frequency ends up depending on the capacitance value and can be considered a proxy for its determination.

Figure 2.1 illustrates the TS555's block diagram. The IC includes a couple of comparators, using some fractions of the supply voltage V_{CC} as their reference value. This design ensures that the output is independent of the supply level, allowing consistent performance across various V_{CC} values.

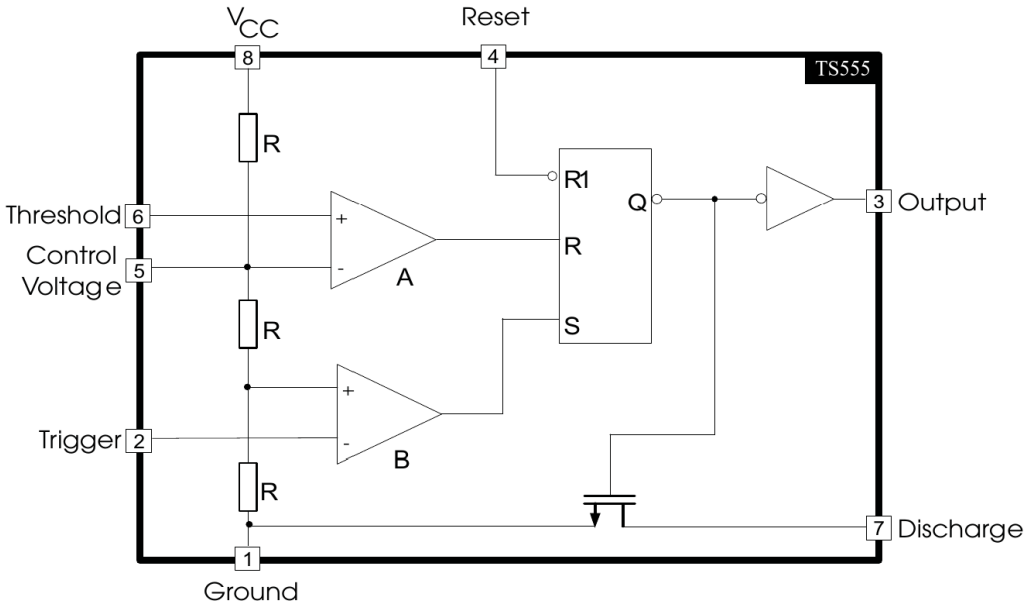


Figure 2.1: Block diagram of the TS555 (image from the datasheet[12]).

In the present work, the TS555 was chosen for several compelling reasons. First of all, it was readily available in AUREL's inventory, which ensured ease of procurement. Secondly, it can cope with a wide range of supply voltages, from 2 V to 16 V, providing flexibility in the power supply design. Finally, and possibly most importantly, its maximum output frequency of 2.7 MHz encompasses with good clearance the bandwidth required for the application, making it an ideal choice for our needs.The astable setup, shown in Figure 2.2, was developed into its final design with the aid of the datasheet[12] and an electrical handbook [3].

The following equation determines the output frequency:

$$f = \frac{1.44}{(R_A + 2R_B)C} \tag{2.1}$$

where R_A and R_B are external resistors, and C is the capacitor value.

2.2 Inverter-based oscillator designs

Multiple techniques exist for creating oscillators using only *digital inverters*. Three approaches will be discussed: the ring oscillator, the single-inverter oscillator, and the dual-inverter oscillator.

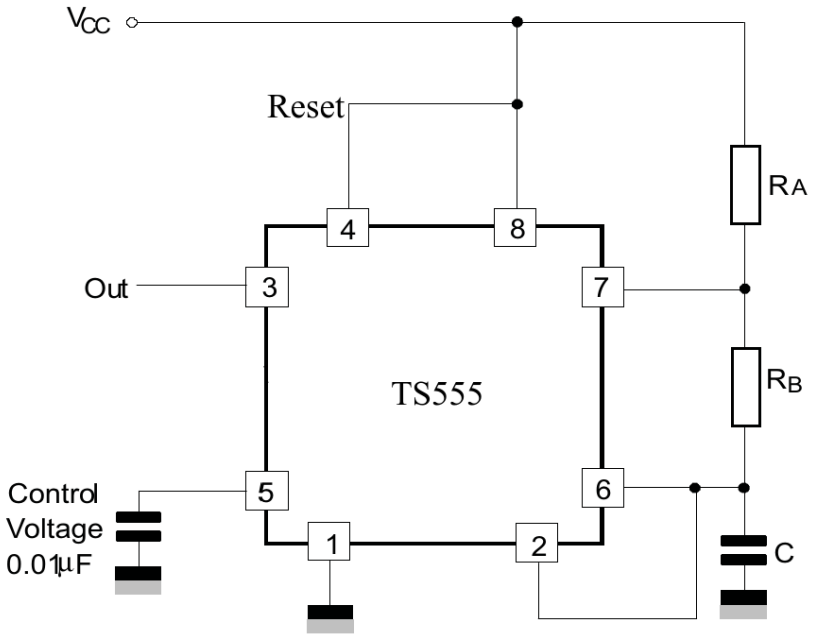


Figure 2.2: Astable TS555 application configuration.

2.2.1 Ring Oscillator

The ring oscillator is the simplest and most commonly studied in electrical engineering curricula. It requires an odd number of inverters (minimum three) connected in a loop. The principle relies on the fact that an odd number of inversions causes oscillation, as the input and output of the circuit can never settle at the same value. The parasitic capacitances of the MOSFETs act as the charging elements. However, this solution was discarded due to its lack of external components for frequency control.

2.2.2 Single Inverter Oscillator

The single inverter oscillator utilizes, as the name suggests, a single inverter, in fact, not a regular one, but an inverter characterized by a *hysteretic behavior* (i.e., a Schmitt trigger). This component is made to operate with a feedback resistor and an input capacitor to obtain a sequence of first order transients that result in oscillations. The hysteresis is mandatory since the circuit (shown in Figure 2.3) could not oscillate without it. The output frequency is linked to the value of the capacitor as reported in the following equation:

$$f \propto \frac{1}{RC} \tag{2.2}$$

The proportionality of the frequency shown in Equation 2.2 is linked to the value of the hysteresis thresholds.

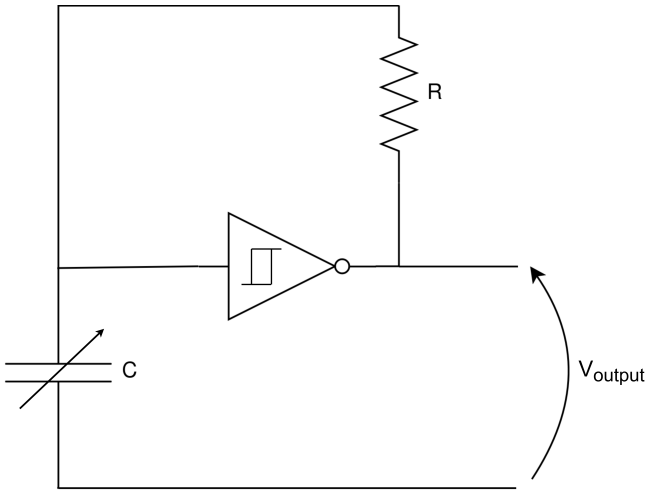


Figure 2.3: Oscillator with single inverter (the inverter has hysteresis).

Despite its simplicity, this solution was not chosen due to the relative scarcity and higher cost of single-inverter components compared with the two-inverter ones. Additionally, simulations revealed noisier waveforms compared to the dual-inverter solution.

2.2.3 Dual inverter oscillator

The dual inverter oscillator uses two inverters connected in a loop and two extra components (a capacitor and a resistor) in order to create an oscillation. The oscillation is generated thanks to the capacitor that needs to charge and discharge introducing a delay to the overall output and the feedback connection between the center connection of the two inverters and the two ”external” ones. The dual inverter oscillator was ultimately selected as the best solution for this application, offering a balance of cost-effectiveness, availability, and lower noise.

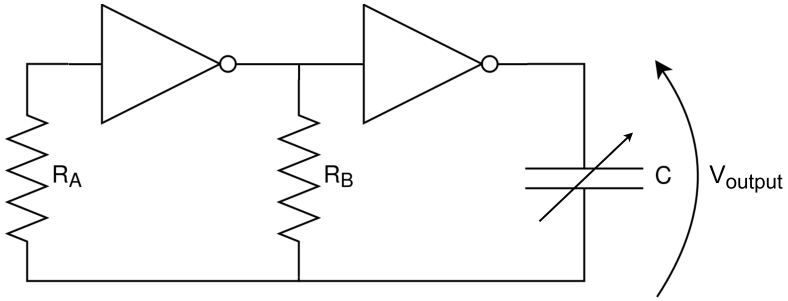


Figure 2.4: Oscillator with two inverters.

Figure 2.4 shows the complete circuit. R_A does not influence the value of the output frequency but ensures protection on the input of the first inverter from high voltage oscillation. The oscillation frequency only depends on R_B and C.

2.3 Op Amp Oscillator (Integrator)

OpAmps are one of the most fundamental circuits used in electronics. They cover multiple applications based on the different components and connections around them.OpAmps offer several advantages for capacitive sensing and noise-prone circuits. They exhibit robust performance against stray capacitance with strategically placed components and connections. While Chapter 1 introduced the charge amplifier as a capacitive sensing solution, it converts capacitance variations into voltage level variations.

This application required time encoding due to the sensor's distance from the central electronics. Consequently, a solution using an OpAmp in *integrator configuration* (Figure 2.5) followed by a *Schmitt trigger* was optimal. Figure 2.6 shows the complete design.

The fundamental concept is as follows: noise initiates a *chain reaction*, causing the integrator to begin diverging towards the voltage limits (determined by the supply voltage) in either the positive or negative direction. As the integrator's output connects to a Schmitt trigger's input, the output switches to the opposite state when the integrator's output reaches a certain level due to the integrator's inverting configuration.

This setup creates an oscillator since the trigger's output loops to the integrator's input. Consequently, each time the trigger's output switches between digital values, the integrator's input also switches, thus perpetuating the cycle.

The capacitor value modifies the integrator's gain. By varying the capacitance, the time it takes for the integrator to transition between digital values is altered. To ensure that the op-amp based circuit can effectively operate as an integrator, reference was given to the handbook [3], which states that the frequency given by Equation 2.3 represents the minimum value for integrating behavior:

$$f_c = \frac{1}{2\pi R_p C} \tag{2.3}$$

To be certain that the OpAmp circuit could perform correctly as an integrator, $f_c = 1\,\text{kHz}$ was

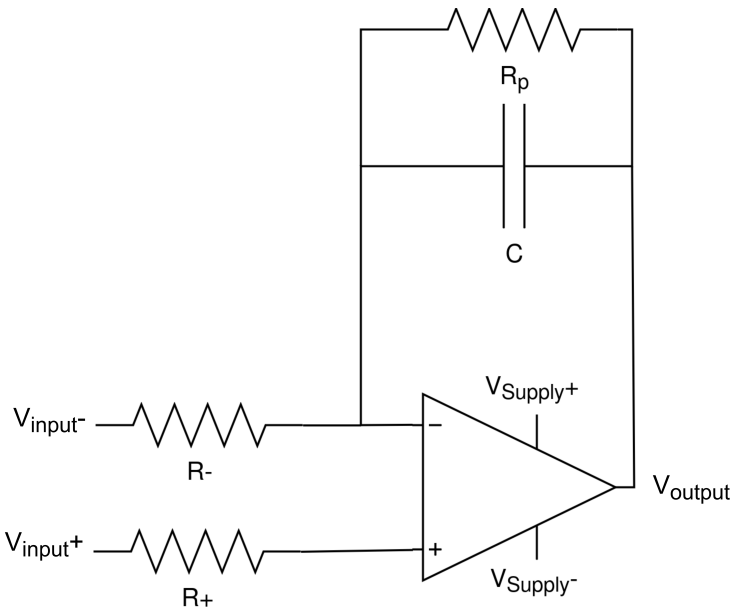


Figure 2.5: OpAmp in integrator configuration.

chosen. With this value, along with a minimum capacitance of 80 pF (worst-case scenario) plus an estimated 30 pF stray capacitance from the cable, the resulting value for R_p is 1.4 MΩ. Even if this may appear large, modern OpAmps can handle it thanks to their high input impedance.

The integrator's gain is related to R^- according to Equation 2.4:

$$G = -\frac{1}{R^- C} \tag{2.4}$$

Initially, the aim was to achieve a significant *gain variation* in a wide frequency range. However, this proved challenging as R^- needs to be higher for a lower gain. After numerous simulations, the choice was to have a very high gain, selecting a 10 kΩ value for R^-. Simulations revealed evident changes in gain even with these high values, resulting in responsive and precise frequency output variations for the complete oscillator.

The second part of this circuit is a Schmitt trigger, which enables oscillation by forming a closed loop with the integrator.

Simulations showed that changing the capacitor value affected the steepness of the integrator's output curve. The basic principle is that the integrator's output reaches the trigger's threshold at different times as the capacitance varies. The resistor determines the hysteresis trigger value in the positive feedback loop, as this application requires a *non-inverting trigger*.

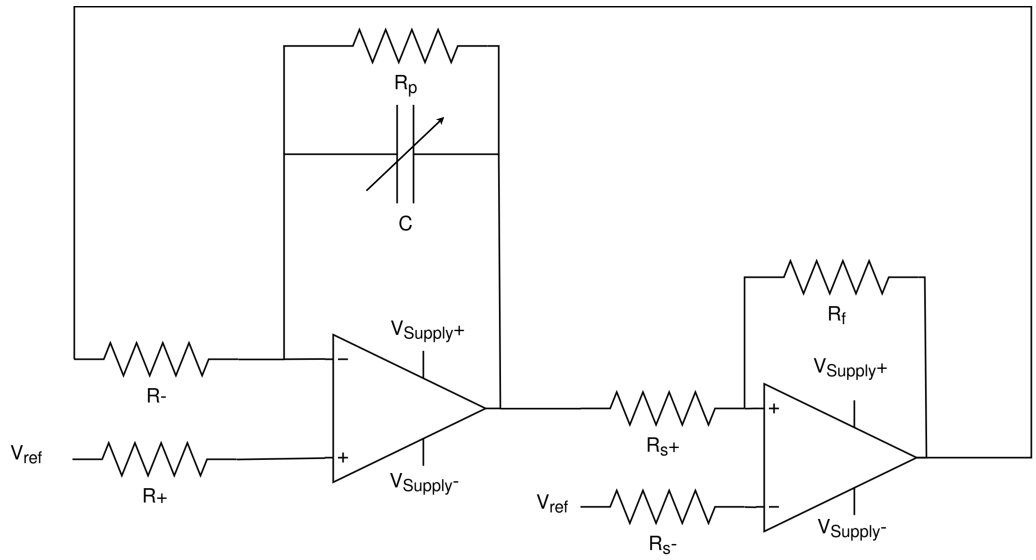


Figure 2.6: Integrator followed by a Schmitt trigger.

The equations governing the Schmitt trigger behavior are:

$$V_{\mathrm{t_H}} = V_{\mathrm{sat}} \cdot \frac{R_{s^+}}{R_{s^+} + R_f} \tag{2.5}$$

$$V_{\mathrm{t_L}} = -V_{\mathrm{sat}} \cdot \frac{R_{s^+}}{R_{s^+} + R_f} \tag{2.6}$$

$$V_H = V_{\mathrm{t_H}} - V_{\mathrm{t_L}} = 2 \cdot V_{\mathrm{sat}} \cdot \frac{R_{s^+}}{R_{s^+} + R_f} \tag{2.7}$$

The resistor values based on the required trigger levels were calculated using Equations 2.5 to 2.7. Note that $V_{\mathrm{t_L}}$ and $V_{\mathrm{t_H}}$ are symmetrical concerning the reference voltage. The threshold was then set to $0.45\,\mathrm{V} \cdot V_{\mathrm{supply}}$ because simulations showed that the integrator's output for larger capacitor values was minimal but still needed to reach the threshold.

2.4 Smart sensor sensing techniques

The following techniques consider the use of a MCU, the STM32L011F4 model from STMicroelectronics [11]. This device is a *low power Cortex* with Arm architecture, it has a low number of ports (20) and low memory (up to 16 kB of Flash and 2 kB of RAM) but its low

power architecture ensures low power consumption. Its low cost and little dimension made this model an excellent choice for the application. The goal was to design a *smart sensor* that could handle some basic signal processing in order to output a more refined signal that could be easier to understand by a *central hub* mounted in another place.

Because the sensor at hand is actually a multisensor, also including a NTC and a heater, the *microcontroller* comes in handy in view of managing these other elements.

2.4.1 Direct RC Sensing

This approach is perhaps the simplest in terms of mechanisms in regards to smart sensors. This technique was found in a book about *sensor-to-microcontroller interfaces* [9] and subsequently adapted to the case at hand. It is based on the principle discussed in Section 1.2.4, utilizing the τ formula (Equation 1.7) to derive the sensor's value from time measurements.

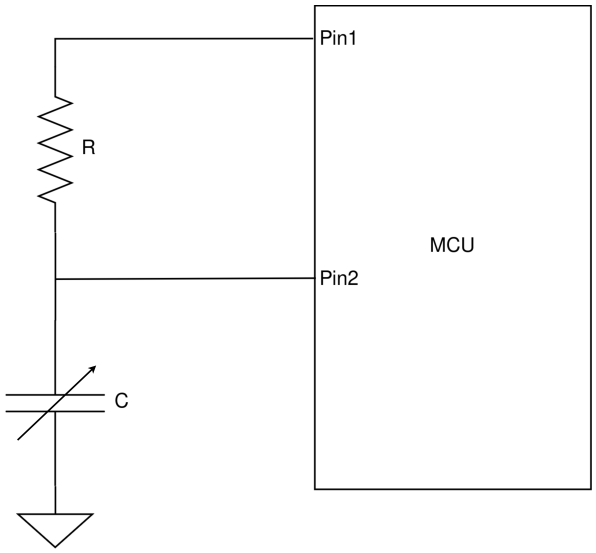


Figure 2.7: Direct MCU connection.

As shown in Figure 2.7, both ends of the resistor are connected to a MCU, while the other end of the sensor is grounded. This configuration allows for free charging and discharging of the sensor, enabling measurements on demand. The default configuration establishes a foundation for accurate measurements. Initially, Pin 2 is set to digital level 0 and programmed as an output, which ensures that the capacitor remains consistently discharged and primed for measurement. When a measurement is required, a sequence of events unfolds.

First, Pin 1 transitions from digital level 0 to 1, acting as an output. Simultaneously, Pin 2 switches from output to input. Pin 2's function changes incorporate a comparator with a custom, pre-selected reference voltage V_{ref}. As these pin transitions occur, a timer is activated and begins its count. The measurement process continues until a critical point is reached. Once the voltage on Pin 2 attains the threshold level, determined by the mentioned above V_{ref}, two actions are triggered quickly. The timer has been counting since the measurement's initiation and immediately halts. Following this, both Pin 1 and Pin 2 are reset to LOW digital outputs, concluding the measurement cycle and readying the system for the next round of data collection. This configuration and process ensure a consistent, repeatable method for capturing precise measurements while maintaining the system's readiness between cycles. It returns the number of milliseconds elapsed between the start of the measuring process and when the voltage on Pin 2 reaches the threshold. This method's error margin is approximately ± 1 ms.

2.4.2 Charge Transfer Method

This design was found in a scientific publication [6] and included to investigate whether the advantages described could be obtained without precise knowledge of the reference capacitor, as required in the original publication.

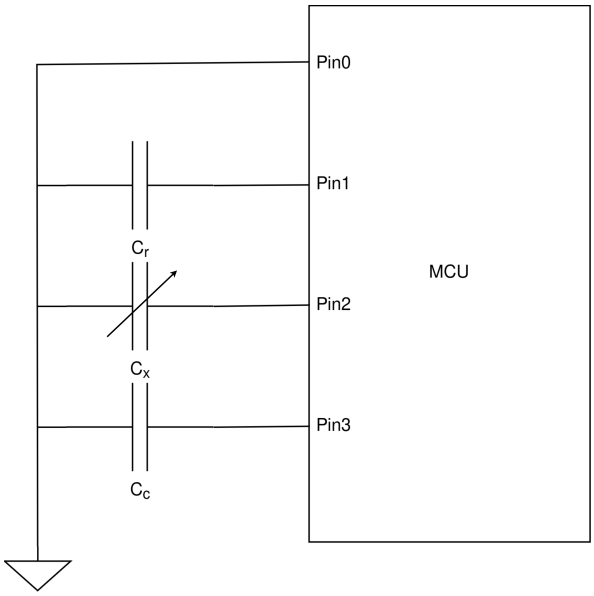


Figure 2.8: MCU connection for charge transfer method.

The charge transfer method has been previously explained in Section 1.2.2 and in the scientific

publication from which the circuit has been adapted [6]. Figure 2.8 illustrates the block diagram of the final connection. It involves three capacitors:

- C_r: reference capacitor
- C_c: calibration capacitor
- C_x: sensor capacitor

As detailed in the original study [6], C_c is used to measure C_x independently of any stray capacitance on the circuit board. However, this effect can only be achieved if both C_r and C_c are precisely measured before mounting. They must also be temperature-stable; otherwise, the added precision would be compromised. The algorithm exploits the charge transfer method, using C_c to charge C_r for calibration. Since both values are known precisely before mounting, this allows for exact proportions between the output and capacitor values. Note that C_r is not charged to total capacity but rather to a predetermined value. The original study used the *intrinsic value* of the port when set to input as the threshold. Due to the study's age, the MCU didn't have a programmable comparator input. For comparison purposes, the characterization algorithm used the intrinsic port threshold instead of the comparator (unlike the direct *RC* solution). Fortunately, the values were consistent.

3 PCBs design

This chapter will explain the design rules used during the PCB realization and the considerations made during the circuit design using commercially available components. PCBs are a medium used to connect components in a circuit. Figure 3.1 shows the final PCB design.

The PCBs have been made following some rules; for instance, all circuits are *dual-layered*, and the bottom one is the ground plane. The circuits are made of *vetronite*, which is 1.6 mm thick. One of the advantages of the high thickness of the vetronite is that it is possible to ignore some *stray capacitance* from the copper on the two layers. Most components have been chosen based on the availability of AUREL's storage, and the sensor is AUREL's proprietary technology. To get the best possible component positioning in the circuit, the placement has been based on *application notes* and the help of employers.

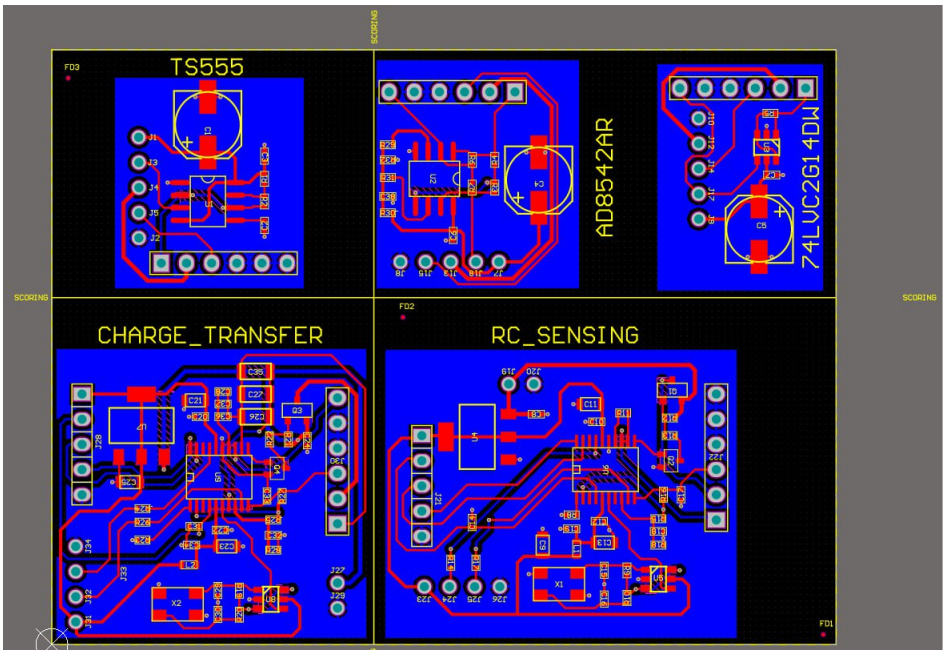


Figure 3.1: The complete PCB.

3.1 Altium designer and design rules

Altium Designer is the software that was available during my internship at AUREL. It is considered *state-of-the-art*, and most companies use it. Altium Designer is a computer-aided design (CAD) software that PCB designers use so that the process that goes from the end of the PCB design to the printing of the latter is easy and compatible.

3.1.1 Design rules

Design rules are a group of rules that the CAD software is required to follow. This is especially useful when the company that will manufacture the PCB is known since the machines will have some physical constraints, such as the minimum width that a track could be or the minimum clearance between two different tracks. Here, a list will be reported with all the rules that needed a setup different from the default values:

- clearance is 0.15 mm in general, while copper is 0.4 mm;
- minimum routing width is 0.254 mm, and the maximum is 5 mm;
- vias connections are set to direct.

Value	Diameter	Hole
min	0.3 mm	0.2 mm
max	1.27 mm	0.5 mm
typ	0.5 mm	0.3 mm

Table 3.1: Vias hole sizes and diameter constraints used as design rules for the PCB.

The diameters and hole sizes design rules for vias are reported in Table 3.1.

During the PCB design, all power lines were placed as 0.5 mm thick lines, all signal lines were 0.3 mm, and all the other lines were 0.4 mm.

3.2 TS555 based interface

The design targets a maximum frequency of 20 kHz to minimize *client-side requirements* for sensor output reading. The worst-case scenario occurs at the minimum capacitor value.

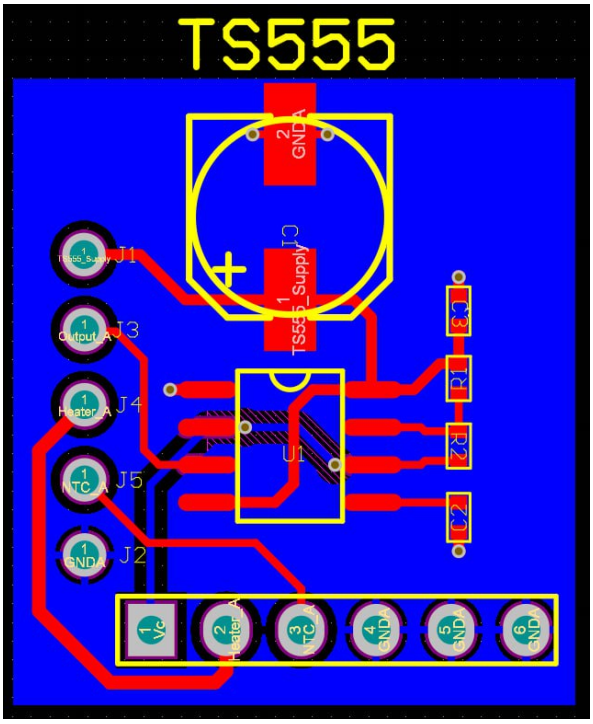


Figure 3.2: TS555 circuit.

Accounting for stray capacitance in a typical 10 cm cable application (estimated at 20 pF to 30 pF), a total minimum capacitance of 110 pF has been calculated. Using Equation 2.1 with $f = 20\,\text{kHz}$ and $C = 110\,\text{pF}$, it has been derived $R_A + 2R_B = 654\,545\,\Omega$. As this exact value is impractical, it has been approximated using two 200 kΩ resistors. This approximation affects the *duty cycle* according to:

$$\delta = \frac{T_1}{T} = \frac{R_A + R_B}{R_A + 2R_B} \tag{3.1}$$

It results in a duty cycle of approximately 67% ($\frac{2}{3}$). While a 50% duty cycle is achievable (e.g., $R_A = 50\,\text{k}\Omega$, $R_B = 300\,\text{k}\Omega$), it wasn't crucial for this application given the low frequency and ease of time interval measurement also using two resistors with the same value could sometimes end up in a lower overall cost for the components.

3.3 Dual inverter oscillator based interface

Referencing Figure 2.4, the frequency of this circuit is given by:

$$f = \frac{1}{2.2R_BC} \tag{3.2}$$

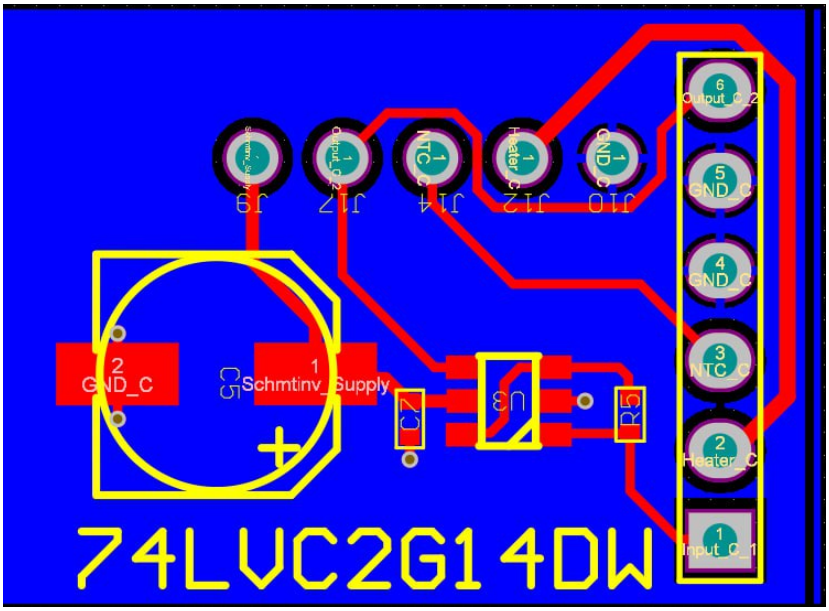


Figure 3.3: Dual inverter oscillator circuit.

Since R_A does not influence the frequency, and the characteristics of the chosen component allow to remove it without reducing the lifetime of the IC itself, it was removed in the final design. The factor $\frac{1}{2.2}$ arises from the switching threshold occurring at half the supply voltage, which, after calculations, introduces a factor of $\ln 3$.

Using Equation 3.2, a value of $R_B = 206\,611\ \Omega$ was calculated using 110 pF as the value of C to achieve an output frequency below 20 kHz. Moreover, this solution inherently provides a 50% duty cycle. 74LVC2G14DW from Diodes Incorporated has been chosen, which incorporates Schmitt triggers with hysteresis that enhance noise immunity.

3.4 Op Amp oscillator based interface

During the design phase, it was assumed the supply would vary between −3.3 V and 3.3 V. The circuit was tested with that supply voltage and worked well. However, the PCB had an error, and the negative voltage had to be delivered manually. Since another PCB could not be printed on time, testing was done with that supply.

After the testing, it was suggested that obtaining negative voltage in a passive sensor could be difficult, so the circuit was redesigned with a supply varying from 0 V to 3.3 V, hence the reason for the circuit in Figure 3.4.

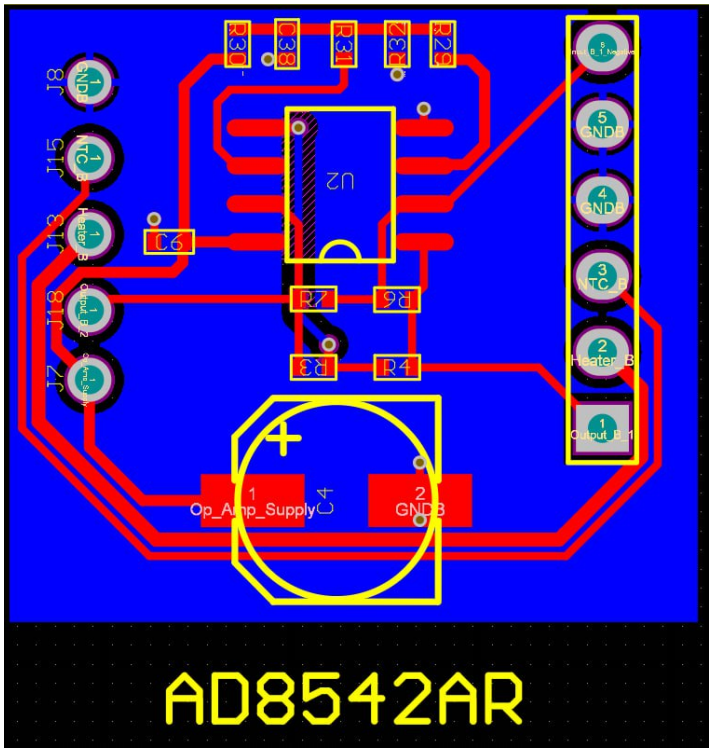


Figure 3.4: Op Amp oscillator circuit.

3.5 Smart sensors

The PCB design of the two smart sensors happened at the same moment. The use of the same MCU has been a good choice, considering the reduction of the design times. There are many application notes and guides on how to properly create the circuit around the MCU.

The most notable aspects are the positioning and choices for the *bypass capacitors*, *the quartz and external clock* choices, and the *LC* filter used to separate the analog from the digital part of the circuit. Bypass capacitors are critical and delicate when a MCU is involved; it is necessary to put them as close to the MCU as possible, and they should be directly connected with the right supply pins. The clock has been designed to be external, using a quartz oscillating at 8 MHz. The external quartz has been deemed necessary due to the low tolerance of the *serial communication*; it can, in fact, tolerate only a 2% mismatch, and the internal clock error at high temperatures was above that. The external quartz generates the oscillation, and then the MCU can raise the value using phased locked loop (PLL). Finally, the *LC* filter was designed to isolate the analog and digital supply voltages.

This design is crucial for high-end applications, but in this case, the analog part didn't need precision since the only signal was digital, and the rest was the supply directly connected to

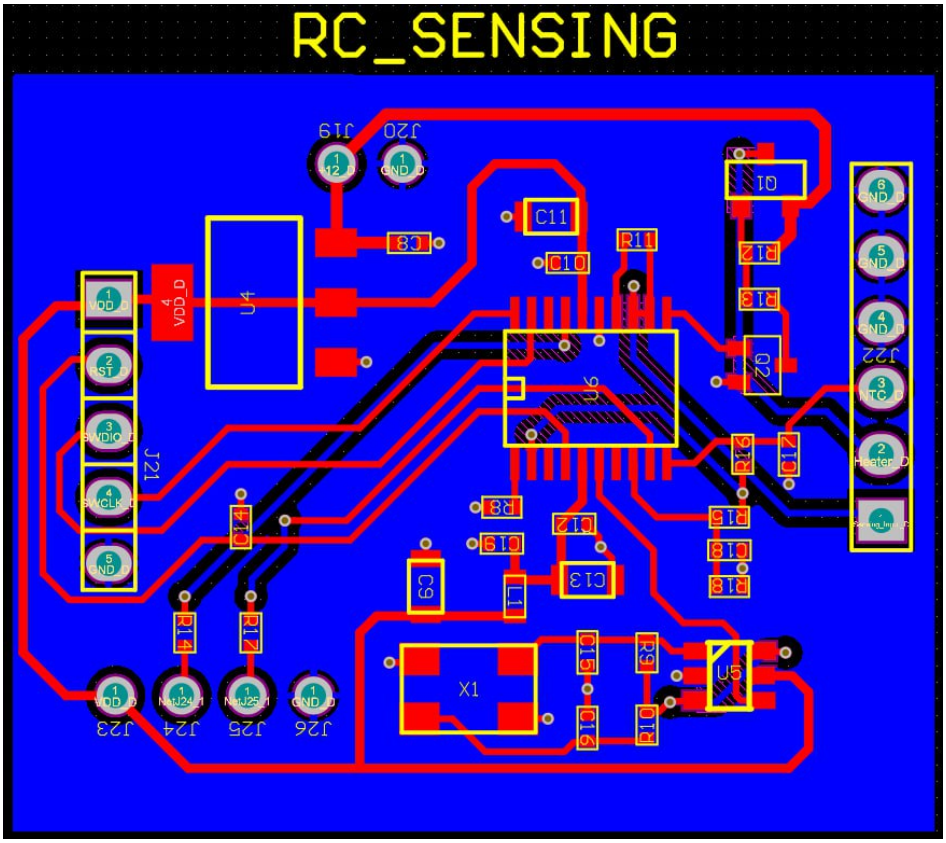


Figure 3.5: RC smart sensor circuit.

the heater. So even if the noise were to go on the analog section, there would be no real issue in the matter.

The discussion in this section is valid both for the RC smart sensor and the charge transfer one since the only difference between the two is the technique used for sensing, which has already been explained in Chapter 2.

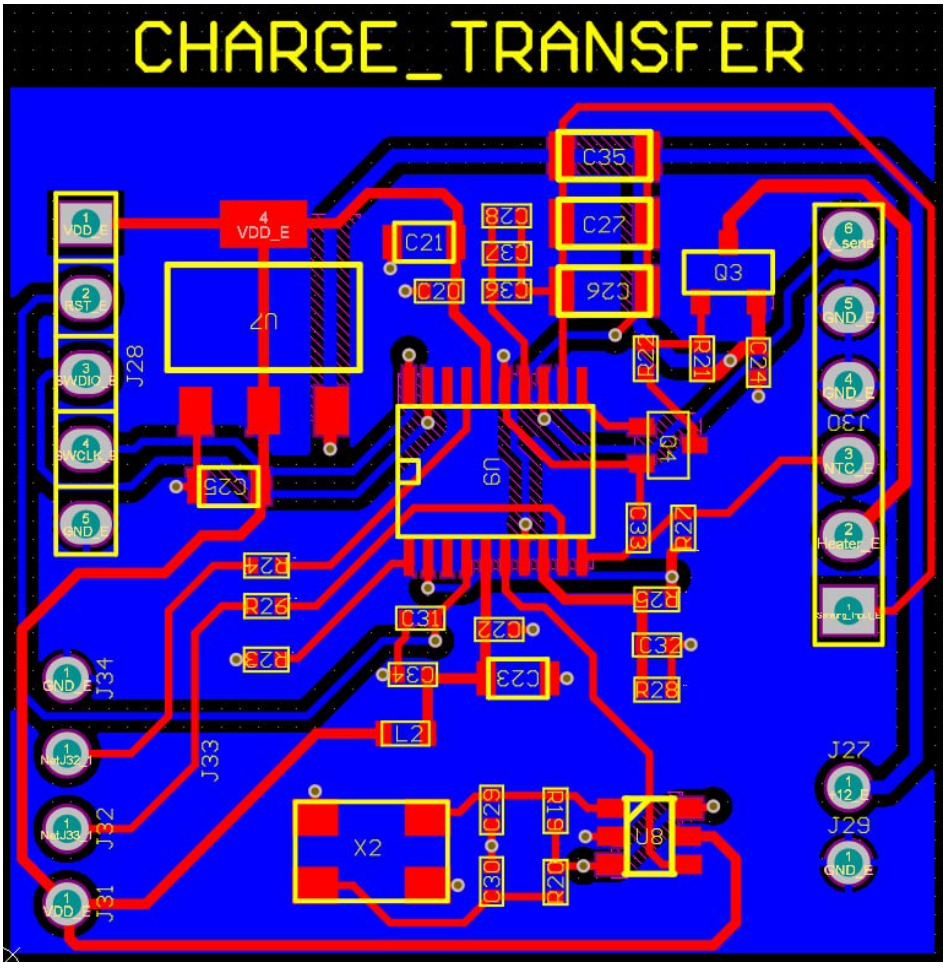


Figure 3.6: Charge transfer smart sensor circuit.

4 Testing

This chapter will explain the main *physical* conditions that could affect the measurement once the sensor is in place, why these conditions matter, and how to measure them. Moreover, this chapter will tell what interfaces are needed for MCUs to interface themselves with the circuits. Finally, a section will discuss the algorithm used during testing and the creation of the test benches.

The results herein presented have been obtained using various *testbenchs*, some built around the versatile *prototyping platform* Arduino [2].

4.1 Physical parameters

This section will briefly explain the physical parameters affecting the measurement, the reasoning behind it, and how they could be measured to better characterize the solutions.

The first parameter to consider is the *moisture level* on the sensor's upward-facing surface. As this level changes, the capacitance value changes, and it is crucial to understand how this is happening. To do that, a *testing jig* was designed and built, and the testing proceeded as follows: with a plastic dropper, a single drop was placed on the sensor's surface, and then the frequency was measured. This process is then repeated until nine drops are on the surface. After nine drops, since there was usually no room left for other drops, the next measure was with the sensor fully covered in water. The sensor was connected to an Arduino and an oscilloscope to measure the frequency output so that the values could be constantly confronted.

Temperature is also a significant factor. The sensor itself showed to be quite resistant to temperature variations. While measuring it using a *precision capacitance meter*, from −20 °C to 90 °C, the reading only varied by a couple of pF. The other components in those circuits are not as temperature stable as the sensor, nor is the cable used for testing the circuits. During the temperature characterization of the smart sensors, the MCU needed to be heated together

with the sensor; the datasheet states that the MCU should perform well in the range considered interesting for this application. The tests were done using a climate chamber so that the temperature at which the measures would be taken could be controlled precisely. Also, a test was conducted to determine the reaction the sensor would have to dew using the same climate chamber.

The length of the cable connecting the sensor to the circuits can affect the measurement by introducing noise and stray capacitances. The longer the cable, the higher the stray capacitances and the higher the chance of external noises. During the tests, the cable length from the circuits to the hub (a laptop) was ignored. Unfortunately, testing for external noise sources during characterization was impossible, so the cable was only issued as a stray capacitance. Two different methods were used to test the cable sensitivity to temperature variations: a cable 65 cm long and a directly soldered 200 pF capacitor in place of the ”short cable”. The reasoning behind using a soldered capacitor instead of the sensors itself was that soldering and desoldering the sensors could damage them and the supply was limited, while the same could not be said for the capacitors. Also, 200 pF was near the middle value for the MKIII model of the sensors.

The cable’s proximity to metal surfaces could introduce noise into the reading, especially if it is long enough. To test this, the cable has been placed on top of a conductive metal plate (the cable is isolated, but some radiation could pass over). None of the tests done were positive, so it is safe to say that the proximity to a conductive metal plate does not introduce enough noise to be able to change the measurement in an observable way.

4.1.1 Testing Jig

The sensor is usually mounted with a 10 cm cable. All the samples had to be tested and characterized using all the circuits, so soldering and desoldering every time could lead to damaged sensors or circuits. Three versions of a testing jig have been developed and tested. The first model was made out of plastic, and while it was good enough to test the sensor at environmental temperature, it couldn’t sustain the high temperature when heated. A second version was designed using vetronite as the primary material, and this is the version that has been used during the whole testing and characterization. A third version was developed since the second tended to leak once the sensor’s surface was covered entirely in water, so the goal was to create a *water-tight* zone that could keep the liquid from leaking. Unfortunately, this

version risked damaging the sensor due to the pressure needed to achieve that water-tightness, so it was ultimately discarded in favor of the second one.

4.2 Microcontrollers and algorithms

This section will detail the *interfaces* needed to utilize the sensor and the circuits designed. It will also explain the *algorithms* used to take the measures during the characterization process. It will then discuss the firmware design in general and present a simple example of a communication protocol at the end.

4.2.1 Interfaces

Interfaces are *devices* that a MCU needs to communicate with other circuits and ICs; they are also used to receive information and compute operations without internal resources. There are many interfaces, each with its use case, and their availability strictly depends on the MCU model. The interfaces used in this project will be reported here, briefly explaining how they work.

The *comparator* is a simple interface that compares the input signal with a reference value. The reference value could be the MCU's standard value, usually based on the supply voltage level and the manufacturer, or it could be chosen digitally and changed while the code runs in newer MCUs. The output of the comparator is a digital high signal when the input is greater than the reference (used as a threshold) and a digital low when the input is lower than the reference. It is possible to have some comparators using a hysteresis, but they were not used in this project.

Timers are used to count the passing of time under certain conditions. They are crucial when measuring how long a signal stays at a certain level. Timers are based on clock times, and their resolution depends on the clock speed. The faster the clock, the higher the resolution of timers; during this project, an 8 MHz crystal was used to power the intelligent sensors. It was then *upscaled* to 32 MHz using PLL techniques. However, the internal Arduino clock had to suffice for the circuits without the MCU.

A core interface in every modern MCU is the analog to digital converter (ADC). ADCs can transform an analog signal into a digital one, with its precision varying based on the *number of bits* it has. The number of bits varies based on the MCU model and manufacturer, and, in

some cases, it also depends on the supply voltage since some MCUs have different interfaces available with different supply voltages. The least significant bit (LSB) corresponds to the minimum voltage level the ADC can discriminate, and it depends on the reference voltages following equation 4.1 if the ADC is not differential and 4.2 otherwise. In both equations, N is the number of bits.

$$LSB = \frac{V_{\text{ref}}}{2^N} \tag{4.1}$$

$$LSB = \frac{V_{\text{ref}^+} - V_{\text{ref}^-}}{2^N} \tag{4.2}$$

Pulse width modulation (PWM) is a particular kind of *modulation* primarily used in *power* applications to have reasonable control over the mean value of a current. It is usually used in electrical motors since their rotating speed is directly linked to the current level, and the mechanical inertia works as a mathematical mean for currents. This technique is mainly used because it acts as a *voltage divider* without the losses that a passive voltage divider using resistors would bring and because of the versatility given by the fact that the value could be changed via software. One of the critical parameters of PWM is the duty cycle. The duty cycle is the ratio in which the output signals stay at a high level over time, and it is used to determine the average power delivered to the load. This solution has been used to drive the heater because it is *energy efficient* and can be controlled with high precision. The heater must heat up to a specific temperature and for a particular time to make the water on top of the sensor evaporate. With PWM, the heater could also heat the sensor if the temperature is too low to take a measure.

The serial interface is the last interface that will be discussed here. It is the only communication between Arduino or the smart sensors and the laptop used to read the measures. Arduino has a serial interface that exploits the USB to connect it to the laptop, so the same cable powers up the Arduino and enables serial communication from the computer to it. Using the STM32 MCU, the intelligent sensor needed a high-frequency quartz with minimal temperature variation. Because the intelligent sensor's circuit could be placed near the sensor itself, it needed to be temperature stable, unlike the Arduino, which has been placed at a distance from the circuits.

4.2.2 Arduino

Arduino is an *open-source* electronic platform based on easy-to-use hardware and software[2]. Depending on the model (Uno, Due, Mega, etc.) Arduino mounts a different MCUs. It is a board that enables the user to use most of the interfaces of the MCU within minutes and in a very easy way; the designers created *wiring*, a *C++* library that gives the user easy-to-use instructions. Arduino Uno [1] (based on the ATmega328p) was used to test the circuits without the STM32 mounted. Since the ATmega328p is very powerful, resources were never scarce, and the vast amount of libraries allowed the writing of the code in a rapid and *easy-to-debug* way.

Although Arduino could be coded using C++, wiring allowed for a simpler overall code, and since performance was never an issue for this project, the loss in performance linked to the choice of language was not so important. Another critical factor in selecting Arduino is that most of its interfaces are already biased, so most of the time, connecting sensors to Arduino is just as easy as plugging it in the pin holder.

Measuring frequency

The frequencies in this application are expected to be low; in fact, the highest frequency measured is under 50 kHz. Having such low frequencies means it is reasonably easy to make the MCU count how much time the signal is high and how much time the signal is low and then sum it up and calculate the frequency from the total time. `PulseIn` is a function of Arduino that works just like this: `PulseIn(Pin, VALUE)` starts a timer when the signal in the input pin selected has the value specified in the function. It returns how much time the signal stays at that digital level. Using that function two times in a row, one with a high value and the other with a low will return two values. Adding those values returns the period of the signal. Once the period has been measured, the equation:

$$f = \frac{1}{T} \tag{4.3}$$

is used to extract the frequency from the period T. There are a couple of issues with this method: first, the value of the sensor could change while the timer is counting, so the period is not very accurate. Then, the function does not start immediately; it waits one entire cycle from when it has been launched to be sure to get a front before starting. Those two problems combined could result in a bad reading. To have a good comparison with the oscilloscope,

the code and the oscilloscope were connected and instructed to use a *256-sample average*. Averaging 256 samples helps with wrong readings, and the results between Arduino and the oscilloscope varied about 3% to 5%, so the algorithm could be considered pretty accurate.

Steinhart-Hart equation

The Steinhart-Hart equation[5] is an interpolation method used to interpolate the curves of the NTC resistors. This equation utilizes three points in the NTC characteristics: one from negative temperatures, one at ambient temperature, and the last at high temperatures. All temperatures used are in °C. First, a logarithm of the resistance value at the three temperature points is calculated using the following equations:

$$\begin{aligned} L_1 &= \ln R_1 \\ L_2 &= \ln R_2 \\ L_3 &= \ln R_3 \end{aligned} \tag{4.4}$$

Then, the inverse of the temperature at which the resistor values had been selected is stored as three terms according to:

$$\begin{aligned} Y_1 &= \frac{1}{T_1} \\ Y_2 &= \frac{1}{T_2} \\ Y_3 &= \frac{1}{T_3} \end{aligned} \tag{4.5}$$

In Equations 4.4 and 4.5, the resistor value and the corresponding temperature value are used to achieve the L and Y parameters (e.g., R_1 is the NTC value at temperature T_1 and are used to calculate L_1 and Y_1). Once Y and L values are all calculated, those parameters are used to obtain γ_2 and γ_3 using the equations:

$$\begin{aligned} \gamma_2 &= \frac{Y_2 - Y_1}{L_2 - L_1} \\ \gamma_3 &= \frac{Y_3 - Y_1}{L_3 - L_1} \end{aligned} \tag{4.6}$$

All parameters previously computed are then combined and used following the equations:

$$\begin{aligned} C &= \left(\frac{\gamma_3 - \gamma_2}{L_3 - L_2}\right)(L_1 + L_2 + L_3)^{-1} \\ B &= \gamma_2 - C\left(L_1^2 + L_1 L_2 + L_2^2\right) \\ A &= Y_1 - (B + L_1^2 C)L_1 \end{aligned} \tag{4.7}$$

Since A needs B and C to be calculated and B needs C, this last parameter should be the first to be computed. Once A,B, and C are obtained, all that is needed to interpolate the reading from the NTC is achieved.

$$\frac{1}{T} = A + B \ln R + C(\ln R)^3 \tag{4.8}$$

Following Equation 4.8 and combining the result with Equation 4.3, the frequency will be found.

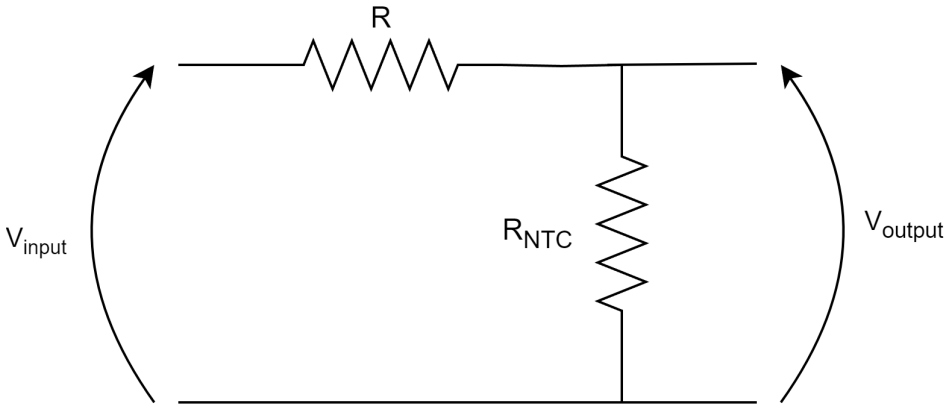


Figure 4.1: NTC connection to Arduino.

The circuit in Figure 4.1 has been used to connect the NTC to the Arduino, R represents a resistor whose value is the same as the NTC one at 25 °C.

$$R_{\text{NTC}} = \frac{V_{\text{output}} \cdot R}{V_{\text{input}} - V_{\text{iutput}}} \tag{4.9}$$

Using Equation 4.9 and its result as the R factored in Equation 4.8 makes obtaining the temperature value trivial. Although this process is complex and occupies memory, it is used to obtain greater precision than other more conventional methods. While Arduino did calculate all the values every time the code restarted, it is possible to calculate A,B, and C n advance and then report only the three values into the code. This will ease the memory usage for less powerful systems.

Heater

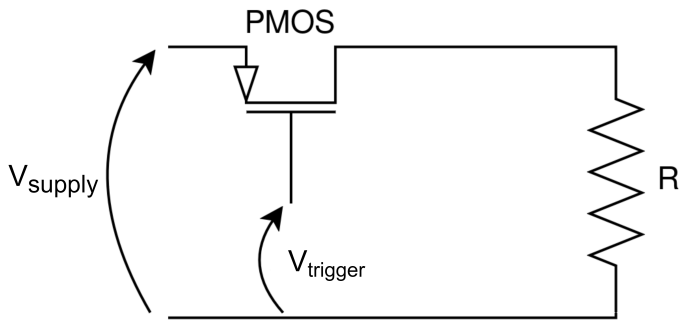


Figure 4.2: Heater connection driven with Arduino.

A PMOS has been used to drive the heater as shown in Figure 4.2. The heater is a resistance with a low resistance value (45 Ω) that heats up when a current high enough (200 mA to 300 mA when using 13 V, the difference in current is due to the use of the PWM) passes through it. When the signal at the gate of the PMOS reaches a specific value, the PMOS opens, and the voltage drop on the heater becomes the one from the supply. During testing, it was noted that, although the datasheet said that 12 V was the optimal choice, a 14 V to 16 V voltage was better suited to evaporate water when the sensing surface was parallel to the ground. Once the sensor was mounted with an inclination so that water drops slid on it (it is usually mounted with a 30° angle), 13 V was more than enough to heat it to the point where there was no more water on it. The inclination is necessary not only because the water could slide on it but also because if the sensor was parallel to the ground, the time the water would take to evaporate once the surface is completely covered was two to three minutes, which is too long. This circuit was used only during testing, and the time needed to heat the surface was similar to the ones reported on the datasheet; a thermal camera was used to test the reached temperatures.

4.2.3 STM32

The STM32L011F4P6 is a low-power Cortex MCU from STMicroelectronics. It's small and low-cost, so it was an excellent choice for the project. The low number of pins led to specific design decisions (like using the same pin to power both the NTC and the sensor), even if it could sometimes be ignored since the applications didn't need many pins. This particular model was readily available in AUREL's storage. Two softwares are mandatory to code it: CubeMX and CubeIDE, both available for free from the manufacturer's site. The first software helps simplify the process of enabling hardware abstraction layers (HALs) or low layers (LLs)

to be used when programming. This process would be complex and prone to error-making, so this software is fundamental. It is also used to program the devices and their interfaces embedded into each model and to preset pins and ports for when the software first starts so that no pin value is unknown after a reset of the MCU. CubeIDE is an integrated development environment (IDE) based on Eclipse and customized so that it can automatically detect when the STLink (the device needed to program every STMicroelectronics MCU) is connected to the PC and if it is connected to a compatible and functioning device. CubeIDE also has customizations that allow it to check inside the memory of the MCU while the code is running, a very useful tool during debugging sessions.

HALs and LL

HALs and LL are free libraries created by STMicroelectronics that determine the type of function that will be usable during programming, while HALs are usually at a higher level of programming, they also occupy way more space in the memory of the device. Unfortunately, since the device used has very little memory backed into it, this choice had to be discarded. As a side note, it is important to say that the compiler used during the project was the GNU one. It is important because it is not as well optimized as some paid ones, so using one of the latter, there is a chance that even though the memory of the MCU used is very little, it could fit. Since using the GNU compiler didn't let the HAL software fit into the MCU chosen, the use of LL was mandatory. LL are more difficult to use because they provide functions at a lower level (as the name suggests), allowing operations even on single bits. This opens up various shortcuts during programming since some high-level constructs could be more complex and pack more functions into them, and using the bit-a-bit logic helps save memory space.

Circular buffers

The circular buffer is a fairly common technique used when there is the need to temporarily store information until it has been utilized. Circular buffers are regular arrays that, once filled, reset their index to the start, and the new data is stored in place of the older one. This kind of buffer was beneficial, especially during the transmission and reception of data during the project. The buffer needs to be bigger than the longer data it will contain to avoid losing information. In C language, the implementation of the circular buffer is relatively easy. All it takes is to create a control function whenever new data needs to be stored; if after the storing happened, the incremented index would be equal to the length of the buffer (because in C,

array indexes start at 0), then the index would be reset to 0. To do this more efficiently, a bit-a-bit operation was used.

```
indexWrite=(indexWrite+1)&(bufferLength-1);
```

In that line of code, `indexWrite` is the index in which the data will be written next time and `bufferLength` is the total length of the buffer.

Data averaging

To read the NTC, an ADC needed to be used. To get better readings, averages have been used so that interferences such as thermal noise wouldn't interfere with the readings. The ADC library uses a circular buffer to do averages efficiently and constantly; it updates the value every time a new one is added to the average. This way, the ADC output is the average only of the last N values, where N is the length of the circular buffer. The algorithm works as follows: first, the older value in the buffer gets subtracted from a variable that stores the sum of all the values in the buffer, and then the new value is saved where the now subtracted one was stored. Finally, the new value is added to the total. After that, the sum of all the values in the buffer is divided by the total number of slots, and then the index is updated in the same way as the circular buffer one.

Data exchange

Transmission is divided into two different processes: transmission and reception. Interrupts have been used for transmission and reception without halting the rest of the code. With interrupts, the code doesn't need to check through polling if there is data ready to be sent or received; the code runs as usual until there is data to send or to receive; at that point, the code interrupts and starts executing routines associated with that particular interrupt. This way, no data is ever lost, the code runs as much as needed, and it starts to send or receive only when required.

The transmission process is simple: a circular buffer stores the data that needs to be transmitted and has two indexes. One indicates the position of the last stored data, while the other contains the position of the data sent last. Every time those two indexes differ, the interrupt occurs. During the interrupt, the MCU is instructed to send out a single data and update the corresponding index; the interrupts continue to be triggered every cycle until all the data has been sent and the two indexes are the same value again. If the data is generated fast enough,

this could lead to overflow problems when the two indexes overlap due to the reset caused by the circular buffer structure. It is, however, quite rare, and it never happened during testing because the dimension chosen for the buffer was enough larger than the data that needed to be sent.

Reception also worked using interrupts, so the routine used for it was only executed when necessary without stopping the code for polling. In a sense, reception works similarly to transmission since it can receive only one character at a time and stores all the received data in a circular buffer. It is notable that, before starting the reception process, the code checks if there were any errors using bit-a-bit operations so that only data without any reception error was received, and the other data were discarded from the buffer but not processed after. Once the reception process has ended, a custom function checks if there is received data to be read (so data with no errors) and, if there is, checks what data it is. The decoding of the instruction received had to be implemented using a circular buffer to prevent crashing errors in case some strings were too long, or the code received too many.

Example protocol

A simple communication protocol using ASCII words has been implemented to test the system's efficacy as a whole. This protocol was a test to see what the system could handle regarding memory and to give a customer a teaching example that works if they don't want to code anything on the sensor.

The code implemented generates new data every cycle. When a command has been received, it transmits the output of that specific command or modifies the data stored according to it. If used as implemented during the project, this protocol is a peak detector; since the sensor is mounted at a 30° angle, the water drops tend to slide off the surface, so the MCU continuously checks for the value of the capacitor to find new peaks, it stores the highest value only and, when checked, the current peak is reset to zero.

This way, the intelligent part is left to the system's central hub, and the sensor only gives the information to the hub when asked to. The peak detector was helpful during testing, so the hub considered it an anomaly when the peak was only one in an extended period. Still, when there were multiple peaks in a short time, the hub started to control the shutters as it was raining. The decision to leave the information processing to the central hub has been made primarily because of memory constraints, so only one sensor could be mounted for a whole location instead of a single sensor for every shutter. This is reinforced by the fact that serial

communication is one-on-one communication and that, this way, it's possible to buy just one sensor for a whole area.

The protocol has been thought to be modular, so a C language struct has been used to represent the sensor and have all the data linked to it; the data related to the sensor in the struct is more than what has been used during testing, this is so that the protocol could be modified with ease.

5 Characterization

The testbench used during the characterization process has been the same throughout all prototypes: the sensor is placed in the custom-made testing jig in AUREL s.p.a. The jig is then connected to a circuit, and the circuit either contains a MCU or is connected to an Arduino. The circuit with the MCU or the Arduino is connected to a PC in order to make it simpler for the user to read the outcomes of the experiments.

Coverage characterization was practiced at ambient temperature (25 °C) by dropping drops of water on the sensor's sensing surface. Initially the first measure was taken with the sensor completely dry, then by adding one drop per measure until nine, and finally with the sensor's surface fully covered in water. With the instrument at disposition, it was impossible to be sure that every drop was identical to all the others and that the water dropped in the same way each time. Still, after different tests, the readings with a similar number of drops were close. For this test, the sensor needed to be mounted parallel to the table; otherwise, the drops of water would slide. Coverage characterization has not been done on circuits that contain an MCU. This is because the output of such circuits is already processed, while the one from the circuits without it is not, so the comparison would not be equal.

Temperature characterization was done using a climate chamber, an enclosed space where it was possible to monitor both temperature and humidity. The climate chamber that could monitor and control humidity changed it unevenly to achieve the correct temperature, thus ruining the measurement. An older climate chamber that only monitors temperature has been used since this one would not change the humidity artificially. The climate chamber was a bit old, and the integrated temperature sensor was placed under some metal plates, so a thermocouple was used to monitor the temperature inside the chamber more accurately.

A third test has been discussed to monitor if the proximity of the cable to a conductive material (like metal) could introduce some noises in the signal. After numerous tests, no variation that could not be attributed to the measuring instrument's uncertainty could be found, so this characterization was ultimately discarded.

5.1 Coverage characterization

This section will be divided into three subsections. The first two will explain and report the results of the characterization of a single circuit, while the last one will detail a brief and simple experiment made to see what the effects of morning dew would be on the sensor.

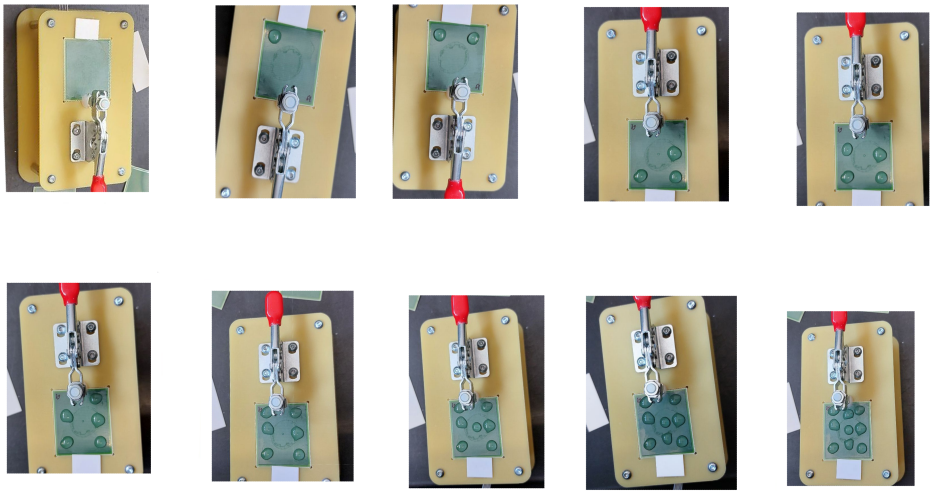


Figure 5.1: Visual illustration of how the sensors have been ”wetted” to various degrees during the characterization. From left to right, top to bottom: dry sensor, and sensor wetted with an increasing number of drops (1 to 9).

5.1.1 NE555

The main strength of this approach is that the output frequency doesn't depend on the supply voltage, so both 3.3 V and 5 V have the same output if all the conditions are equal. This solution has an excellent output, both precise and true, so overall, it is very accurate. The only downside is that the full-scale span is smaller than any solution without the MCU. The frequency output lowers exponentially (as Figure 5.2 shows) as the capacitor gets more covered. The values reported in tabular format in Appendix B as Table 1 show that values are rarely equal but don't vary much from one sensor to the other.

5.1.2 Inverter oscillator

At first glance, the oscillations obtained using this circuit were not precise. The frequency had significant variations (5% to 10%) at higher values while, at lower ones, it lowered to

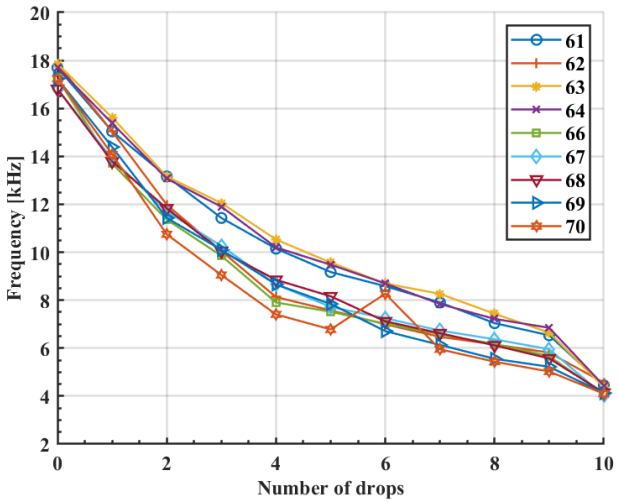


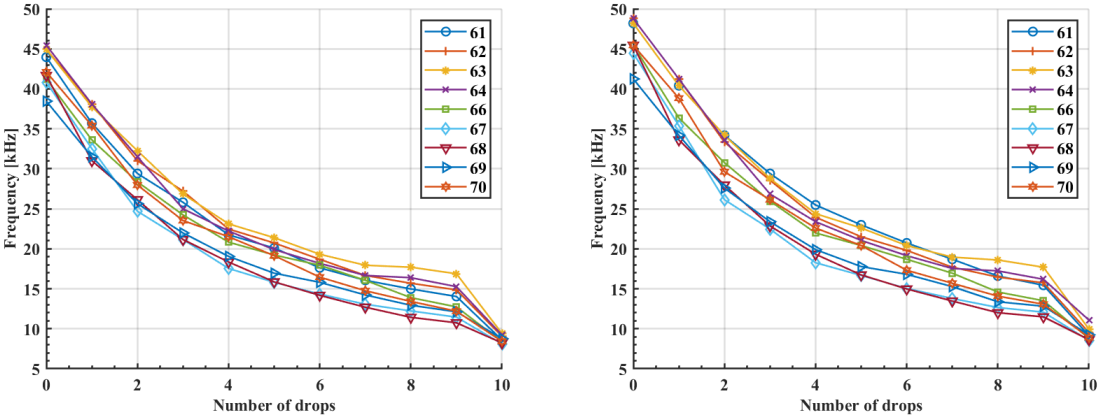
Figure 5.2: Frequency, in kHz, measured during coverage characterization of NE555. Each number represents a different sensor; the numbers were assigned during the characterization processes.

0.1%. This variations are not linked to the code used in Arduino since the oscilloscope had them too. Arduino and the oscilloscope have been programmed to do a 256-sample average to improve the frequency output that is shown in Figure 5.3. Using averages, the output frequency becomes very steady and precise.

The Tables 2 and 3, situated in Appendix B, show that supply voltages play a role in the output frequency. To prevent this, the leading suggestion would be adding an low drop-out (LDO) to the board, increasing the final product's cost. This circuit was the cheapest, so adding components would lower its appeal. Using averages could impact the memory of the MCU in case it's a low-memory one like the one used in this project, but it is otherwise a good option.

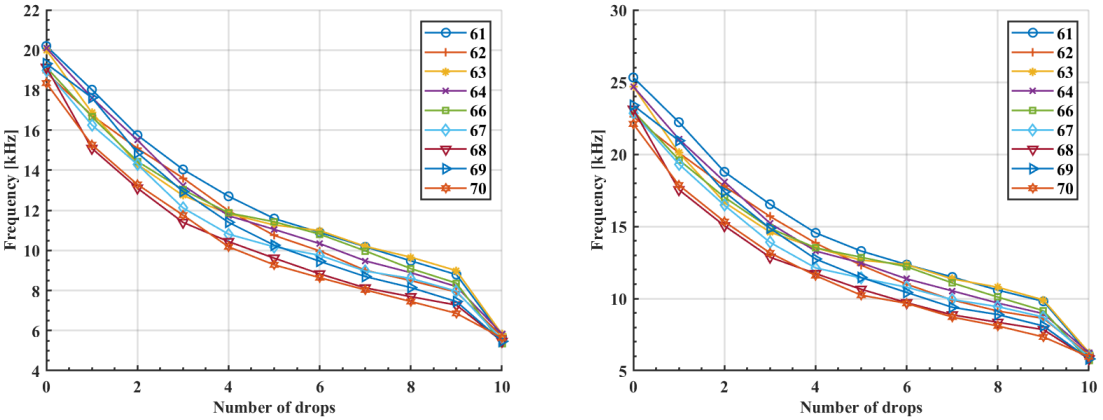
5.1.3 Integrator

The integrators' output is a good tradeoff between the first two solutions presented. It is very precise as it is both true and accurate; even the readings without the averages were stable enough that they could be used as samples. It is on par with the output of the TS555 circuit, even if it is not stable regarding the supply voltage, because the design didn't account for that. Also, the measures reported in tabular format in Appendix B as Tables 4 and 5 and in Figure 5.4 are from a circuit that uses the negative supply voltage, while the final design would have only the positive one.



(a) (b)

Figure 5.3: Frequency, in kHz, measured during coverage characterization of inverter oscillator. Each number represents a different sensor; the numbers were assigned during the characterization processes. Data obtained with the oscillator powered at (a) 3.3 V and (b) 5 V.



(a) (b)

Figure 5.4: Frequency, in kHz, measured during coverage characterization of integrator circuit. Each number represents a different sensor; the numbers were assigned during the characterization processes. Data obtained with the oscillator powered at (a) 3.3 V and (b) 5 V.

5.2 Temperature characterization

This section will be divided into five subsections, each detailing the behavior of one circuit when the temperature varies in a controlled space. At the end of this section, there will be a little insight into one test that has been done to see how the *RC* circuit with the MCU reacts in case of sudden condensation on top.

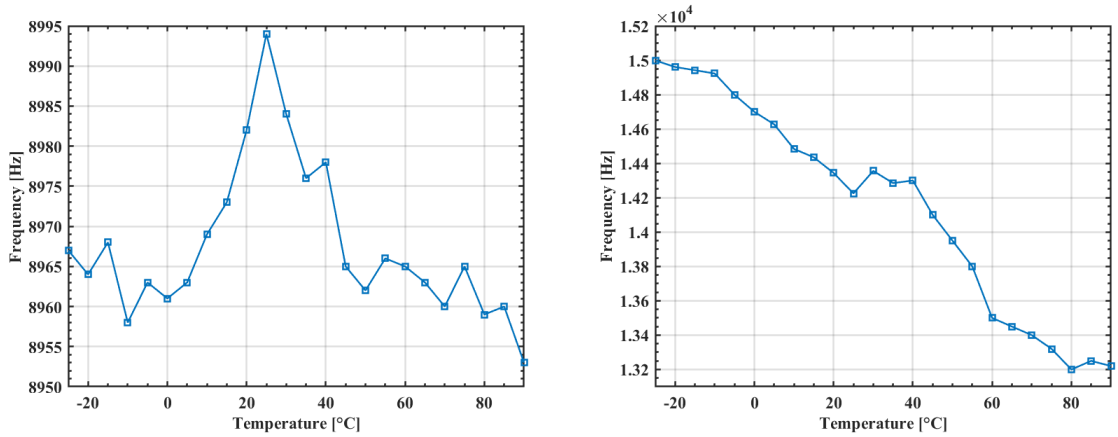
Temperature characterization is divided into two types: long cable tests and short cable ones. When testing with the short cable, interfaces have been operated with a 200 pF directly soldered on the circuits in place of the sensor. Not only did this ensure the shortest possible cable length, but it also prevented damage to sensors due to repeated soldering and desoldering. Also, the sensors were previously tested in the climate chamber to ensure they were stable in temperature. Long cable is about 60 cm long. The test cell worked like a big oven, with hot or cold air that was blown into the chamber. This caused the NTC readings to be off since the sensor was placed directly in front of the fan. So the NTCs were disconnected because, at this stage, all the measures and tests with them had already been done. In this test, the temperature readings were obtained from the thermocouple present in the test equipment and shielded from the direct action of the fan in order to get the most accurate and repeatable measurements.

In this section, the discussion of temperature stability will be based on the observation of variations expressed as percentages. The variations have been calculated using 25 °C as a baseline, so for example, a 5% variation at 90 °C is to be considered a 5% variation regarding 25 °C, and the same for a 2% variation at −10 °C.

5.2.1 TS555 based interface

From −20 °C to 90 °C, the TS555 was really stable. The maximum variations in temperature are lower than 0.5% with the short cable and are around 7% when the cable is long. This trend will be the same for every circuit since long cables introduce more noise due to heating. The higher values with the long cable could be explained by the fact that the dry value of the sensor added to the stray capacitance of the cable were lower than the 200 pF capacitor used for short cable testing.

The data in Figure 5.5 and available in tabular format in Appendix B as Table 6 provides the measured quantities.



(a) (b)

Figure 5.5: Frequency, in Hz, measured during temperature characterization with the TS555 based interface. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

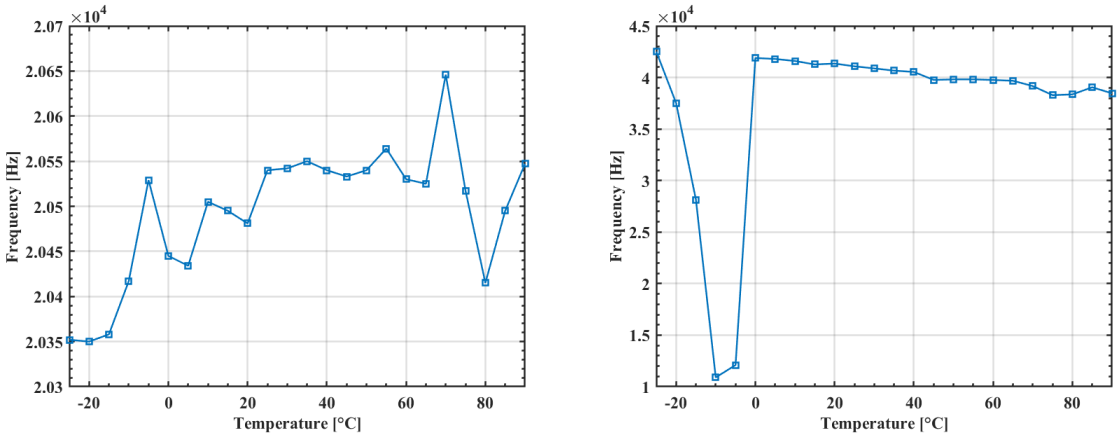
5.2.2 Inverter-based oscillator

The inverter-based oscillator performed very well when operated with the short cable; the variations were below 1%, while in long cable, they were around 6%. It must be noted that, when doing long cable measurements, the interval between −10 °C and 0 °C had a strange behavior, with values that changed very fast and a lot. The lowest reading was around 9 kHz, while the highest was about 40 kHz. Once it reached 0 °C, the endeavor returned to normal, and stability was acquired again. This issue could be attributed to the fact that in that interval, the climatic chamber blew hot air on top of a cold surface, but those same variations never happened in any other circuit tested.

The data in Figure 5.6 and Table 7 situated in Appendix B provides the measured quantities.

5.2.3 Integrator oscillator

The integrator oscillator performed well on both the long and short cables. Variation was around 1% with the short cable and about 2.5% with the long one. This is the one circuit without the MCU that performed better with the long cable. This is probably due to how the cables were attached to the OpAmp since it could reduce noise thanks to its high input impedance. The hysteresis introduced by Schmitt's trigger helped too.



(a) (b)

Figure 5.6: Frequency, in Hz, measured during temperature characterization of the interface using the inverter-based oscillator circuit. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

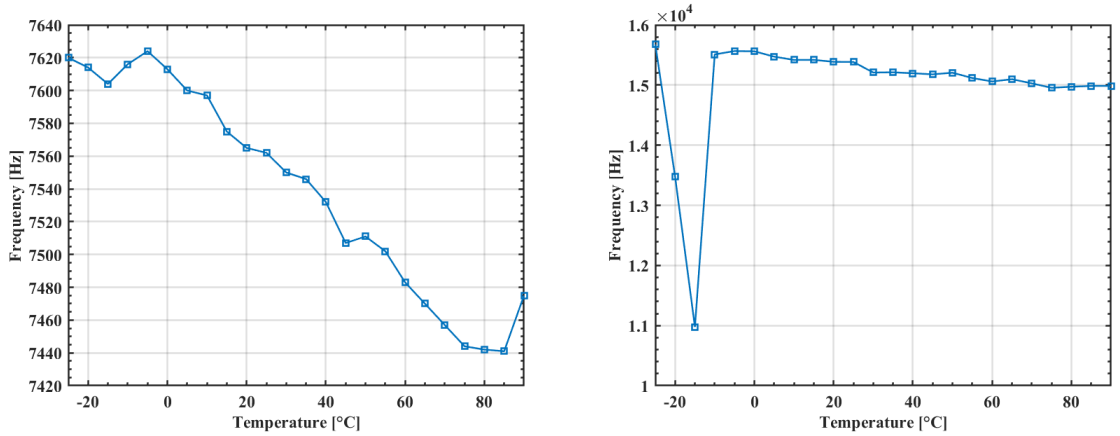
The data in Figure 5.7 and Table 8 situated in Appendix B provides the measured quantities.

5.2.4 Smart RC sensor

The data in Figure 5.8 and available in tabular format in Appendix B as Table 9 represents the temperature characterization of this design, do not report frequency. Every 30 ms the MCU returns the number of ticks that the timer TIM21 counted from when the 30ms countdown started up until the moment the voltage on the sensor reached the threshold selected digitally. The duration of one tick is linked to the internal clock speed, which was 32 MHz in this case, and so every tick is approximately 31 ns. In this circuit, when measuring using long cables, something similar to the inverter oscillator happened between −20 °C and 0 °C, with the number of ticks way higher than the rest of the measures, with a peak of seven thousand ticks. Ignoring that interval, the maximum variation is 5%. Once the temperature was no more in that critical interval, the circuit stabilized again. With the short cable the variations kept under 0.5%. This is the best result, albeit not too different from the TS555 circuit.

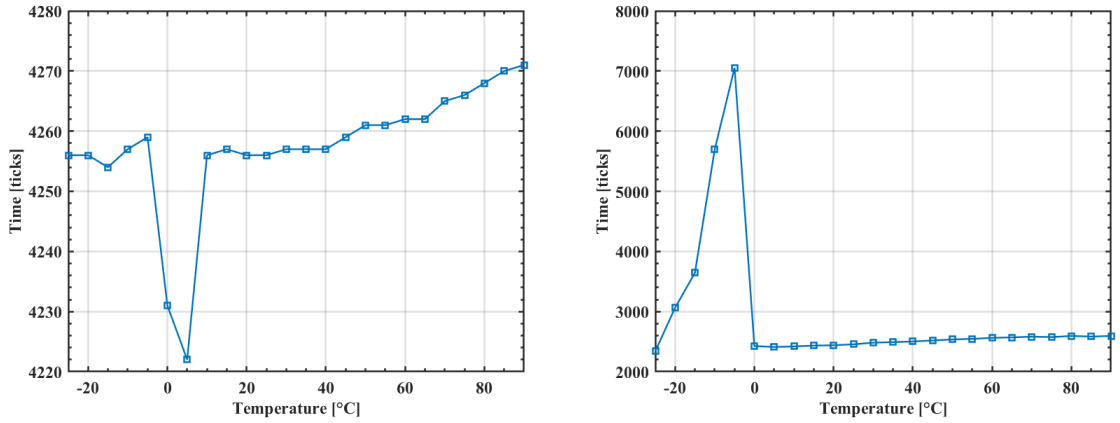
5.2.5 Smart charge transfer sensor

The tables here represents the number of times the sensor needs to be charged and discharged to charge the reference capacitor to a determined value. In this case, the value was the one of



(a) (b)

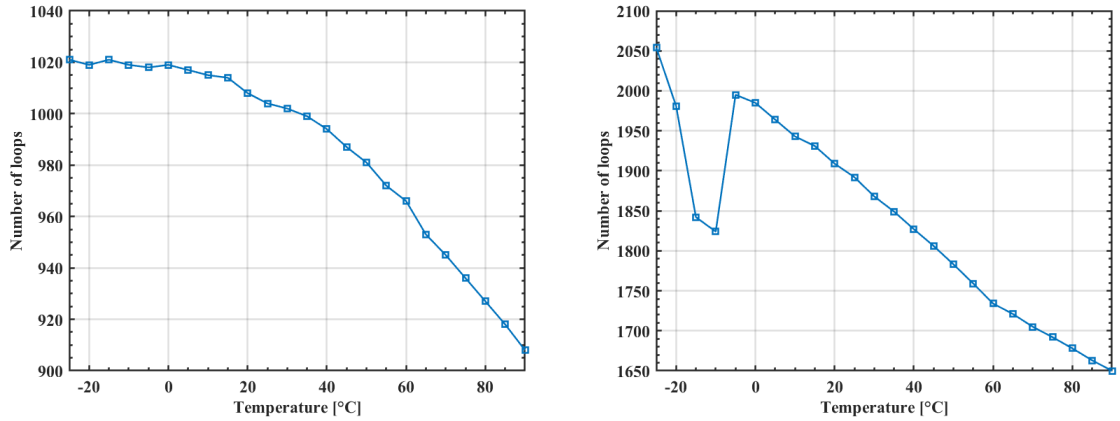
Figure 5.7: Frequency, in Hz, measured during temperature characterization of the interface using the integrator circuit. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.



(a) (b)

Figure 5.8: Time [ticks] measured during temperature characterization of RC smart sensor. Ticks are the number of times the timer TIM21 counted from the start of the measure up until the voltage on the sensor reached the threshold digitally selected. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

the internal switching logic of the MCU because that was how it was done in the document [6] that inspired this solution to make the comparison more accurate. The internal switching logic was two-thirds of the supply voltage, which could also be compared with the *RC* circuit. The data in Figure 5.9 and available in tabular format in Appendix B as Table 10 show, in both long and short cable characterization, the value peaked at −25 °C, and it steadily decreased until 90 °C, where it had its lowest point. The variations are significant, especially in long cables, where they sometimes reach 12%, but in short cables, they are also non-negligible, where they set around 6%. All in all, this solution had the most significant percentage variation.



(a) (b)

Figure 5.9: Iterations measured during temperature characterization of charge transfer smart sensor. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

5.3 Dew experiment using the climate chamber

The last test done with the circuits is a simple one using the climate chamber to simulate an effect that resembles morning dew. Since the sensor will be mounted outside, this situation could occur every day if the environment is humid enough. In order to simulate this effect, the sensor has been placed inside the climate chamber, and the temperature has been set to −30 °C. Once the inside of the climate chamber has reached the target temperature, the chamber has been opened without waiting so that the temperature difference (which was drastic because it was made during summer) will cause condensation on top of the sensing surface. Water started condensing on the sensor surface the moment the climate chamber's door opened. The results were exactly as expected: the frequency started to lower as soon as the door opened,

and the change was gradual. All the circuits could handle the gradual change well; in every case, the software would have had enough time to process what was happening.

6 Conclusions

In this work we have successfully designed and tested some interface circuits for a capacitive rain sensor. The characterization of the sensor and of the various interfaces has also been practiced, both to assess the sensitivity of the system to the quantity of interest (the rain-induced wetting) and to observe the influence that other external factors could have on the measurement (temperature, cabling, etc.).

The ultimate goal was to devise different designs for the sensor interfaces in order to accommodate the different requirements that could be expected from different customer groups and different applications.

In this regard, two main approaches have been followed: first, designs that use time encoding to transfer data from the sensors were found; then, smart sensors were designed based on a MCU so that the sensor's circuit itself could process part of the information and communicate higher level data to a central hub. Once those designs were functional, two main characterizations were made: one regarding the time encoding circuits only, which characterize the different values of the frequency the circuit outputs based on coverage, and the other regarding temperature characterization, which involved using a climate chamber to model the temperature variations of the circuits while the sensor's sensing area was dry.

With the gathered data, it is possible to draw some conclusions regarding the performances and various trade-offs needed to create those circuits.

First, the circuit created using the TS555 was the best time encoding ones. It was the more stable both in terms of output frequency and supply voltages. Moreover, the total cost was low because it only needed three additional passive components (ignoring the sensor). Even if the performance of the other circuits was on par with this one, the low cost and the output stability regarding the supply voltage leave this circuit as the best overall for the time encoding ones.

The circuit using inverters was the cheapest among the time-encoding ones because it was just one integrated circuit with two extra components (apart from the sensor): a resistor and a

bypass capacitor. However, the performance is worse than that of the other solutions exploiting time encoding. The output must be averaged to get a good reading because it is not stable regarding the supply voltage, and it has some issues linked to the output impedance when used with the Arduino. Thanks to the low-cost design, this solution is best suited for customers prioritizing budgeting.

The design that uses OpAmps as an integrator and Schmitt's trigger has good performance because the output is stable enough that measures could be taken accurately without using the average. It has no impedance adaptation problems thanks to the high output impedance of the OpAmps. This design is the most expensive of the time-encoding ones. However, thanks to the low variations obtained during long cable temperature characterization, this circuit is best suited for applications that need the sensor to be distant from the circuit, even if its output is unstable regarding the supply voltage.

The RC solution is precise since it uses interrupts to count, and using an external clock helps with temperature stability. It also has an LDO mounted on it so that it can be supplied with a variety of supply voltages. The model used during this project has very low memory, but using a different MCU could be a good solution since the code isn't dependent on the specific architecture of the model but rather on the architecture of the series STM32L0. The simple protocol coded has been intended as an example of what the drivers could handle, rather than a complete application.

The charge transfer circuit, albeit promising, was not better than the RC one. The capacitors needed to be temperature stable (for example, NP0 type) and, as such, very expensive. Because of the technology involved during manufacturing, NP0 capacitors can only be found in small quantities. This could be troublesome because, in order to use this technique properly, the reference capacitor needs to be much higher than the sensor one. Because the value of the reference and the calibration capacitors needs to be known precisely to get the advantages reported in the document that first presented this design [6], it is unfeasible with mass production in mind. However, it could be a good choice if the customers only needs a small number of very precise pieces because all of the capacitors need to be measured precisely by hand.

In conclusion, all of the designs presented in this thesis worked well during tests and characterizations. The choice of one over the other could be based on the characteristics previously mentioned.

References

[1] *Arduino UNO R3. Product Reference Manual SKU: A000066*. Data sheet. Arduino. 13th Sept. 2024. URL: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf (visited on 17/09/2024).

[2] Massimo Banzi and Michael Shiloh. *Getting Started With Arduino: The Open Source Electronics Prototyping Platform*. 4th ed. Make Community, Mar. 2022. ISBN: 978-1680456936.

[3] Giuseppe Biondo and Enrico Sacchi. *Manuale di elettronica e telecomunicazioni*. Italian. 5th ed. Hoepli, Apr. 2005. ISBN: 9788820334901.

[4] *Capacitive sensing*. Wikipedia. URL: http://en.wikipedia.org/w/index.php?title=Capacitive%5C%20sensing&oldid=1234147061 (visited on 14/09/2024).

[5] Yu Cong et al. "Study on NTC thermistor characteristic curve fitting methods". In: *Proceedings of 2011 International Conference on Computer Science and Network Technology* (Harbin, China, 24th–26th Dec. 2011). Vol. 4. IEEE, 2011, pp. 2209–2213. DOI: 10.1109/ICCSNT.2011.6182415.

[6] Jorge E. Gaitán-Pietre, Manel Gaulla and Ramon Pallás-Areny. "Direct interface for capacitive sensors based on the charge transfer method". In: *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007* (Warsaw, Poland, 1st–3rd May 2007). IEEE, 2007. DOI: 10.1109/IMTC.2007.379227.

[7] *NTC thermistors for temperature measurement. SMD NTC thermistors*. Data sheet. Version 3. TDK. Mar. 2024. URL: https://product.tdk.com/system/files/dam/doc/product/sensor/ntc/chip-ntc-thermistor/data_sheet/50/db/ntc/ntc_smd_standard_series_0402-0803.pdf (visited on 14/09/2024).

[8] Ramon Pallàs-Areny and John G. Webster. *Sensors and Signal Conditioning*. 2nd ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[9] Ferran Reverter and Ramon Pallás-Areny. *Direct Sensor To Microcontroller Interface Circuits: Design And Characterisation*. Marcombo Ediciones Técnicas, Oct. 2005. ISBN: 978-8426713803.

[10] Ryan Seguine. *Capacitive Sensing Techniques and Consideration*. White paper. Cypress Semiconductor Corp., 2007. URL: https://www.infineon.com/dgdl/Infineon-TA1235-Whitepaper-v01_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0f8e3f3a7827&da=t (visited on 14/09/2024).

[11] *STM32L011x3 STM32L011x4. Access line ultra-low-power 32-bit MCU Arm-based Cortex-M0+, up to 16KB Flash, 2KB SRAM, 512B EEPROM, ADC*. Data sheet. DocID027973 Rev 5. STMicroelectronics. Sept. 2017. URL: https://www.st.com/resource/en/datasheet/stm32l011f4.pdf (visited on 17/09/2024).

[12] *TS555 Low-power single CMOS timer*. Data sheet. Version 3. DocID 4077 Rev 4. ST Microelectronics. June 2015. URL: https://www.st.com/resource/en/datasheet/ts555.pdf (visited on 14/09/2024).

Appendix A: Schematic figures

In this appendix, the images of the schematics designed using Altium Designer are reported for enhanced readability.

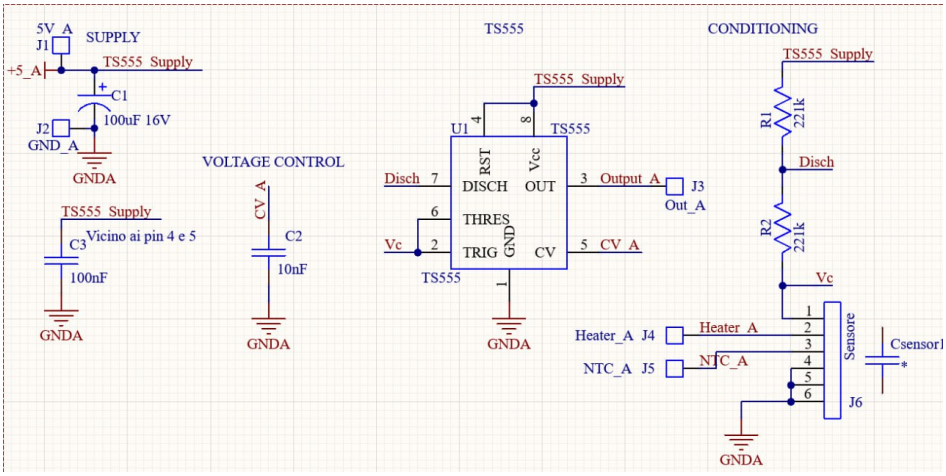


Figure 1: TS555 solution.

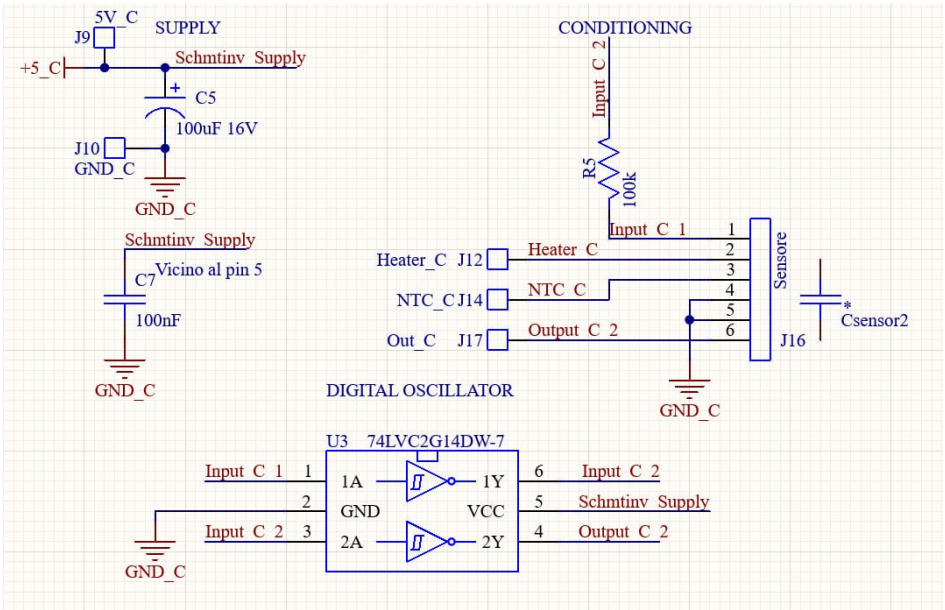


Figure 2: Dual inverter solution.

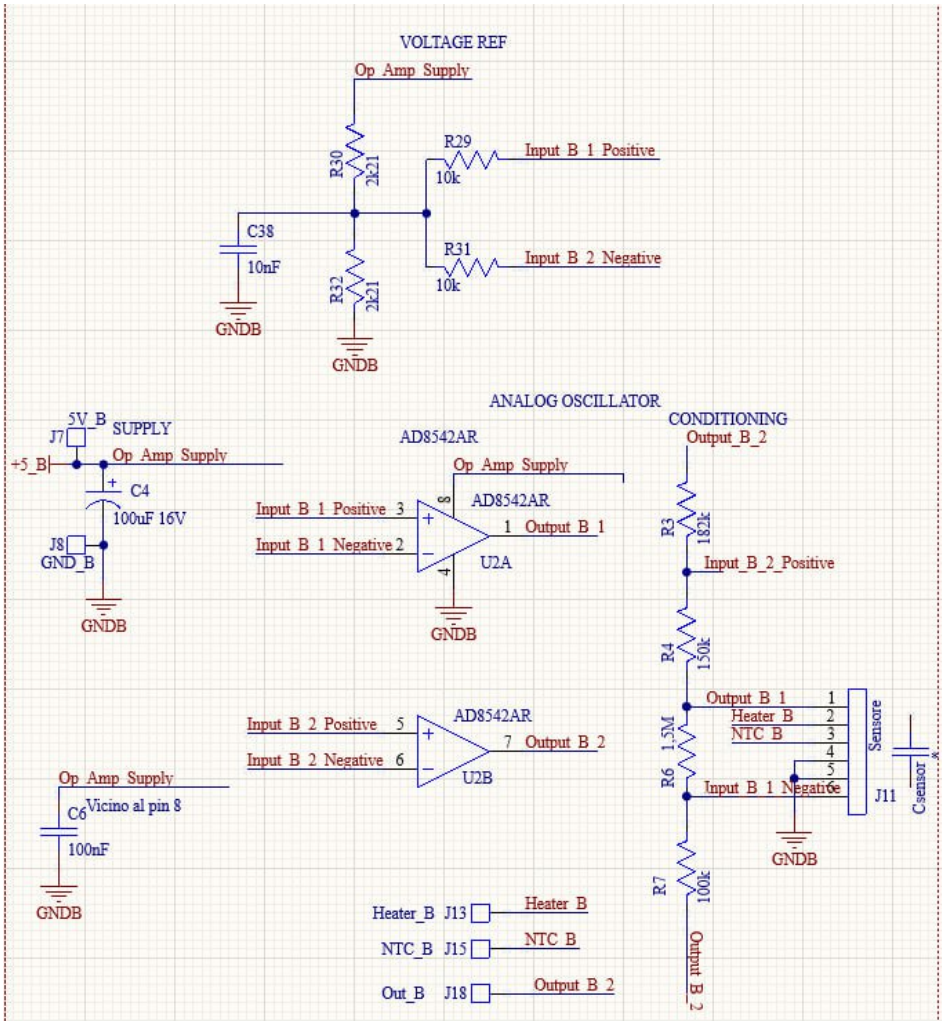


Figure 3: Integrator and trigger solution.

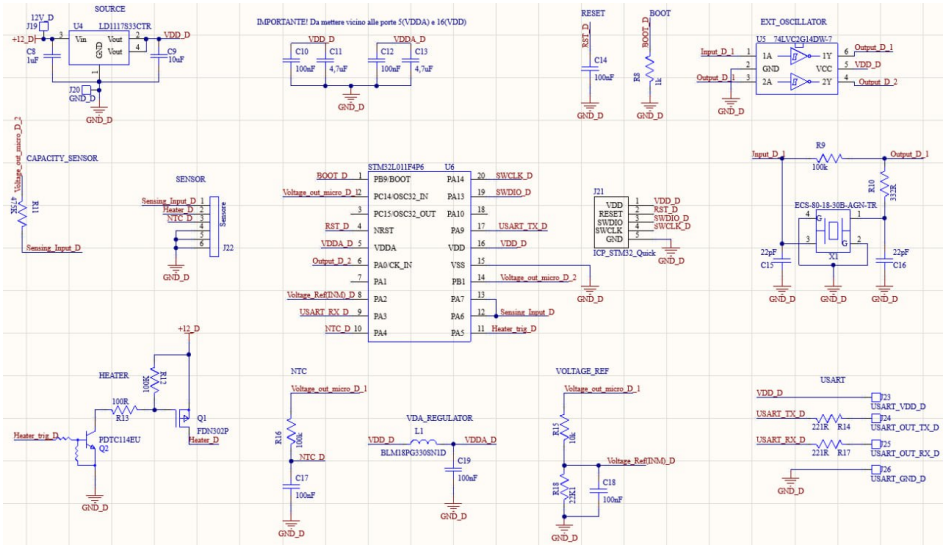


Figure 4: RC smart sensor solution.

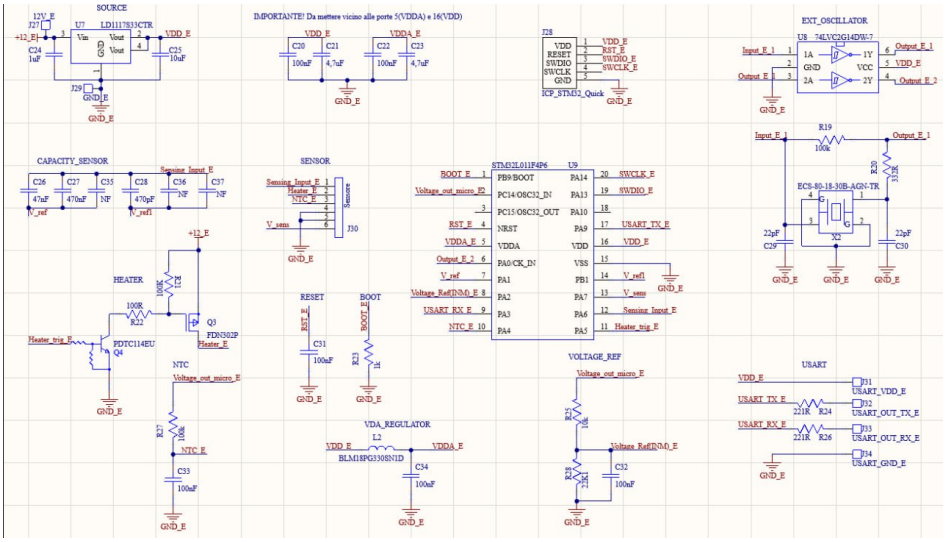


Figure 5: CT smart sensor solution.

Appendix B: Characterization tables

This appendix will first report all the tables with the data regarding the coverage and temperature characterization explained in Chapter 5.

	Wetness degree (# of drops or full)										
Id	0	1	2	3	4	5	6	7	8	9	Full
61	17.70	15.04	13.16	11.43	10.15	9.174	8.584	7.905	7.042	6.536	4.444
62	17.86	15.04	11.98	10.05	8.130	7.576	6.993	6.472	6.154	5.814	4.556
63	17.86	15.62	13.16	12.05	10.53	9.569	8.696	8.264	7.435	6.645	4.376
64	17.70	15.38	13.07	11.90	10.20	9.479	8.696	7.843	7.220	6.849	4.435
66	17.24	13.70	11.36	9.852	7.905	7.519	7.018	6.579	6.173	5.666	4.073
67	16.81	13.89	11.70	10.26	8.658	7.722	7.246	6.734	6.369	5.952	4.032
68	16.81	13.79	11.83	10.05	8.850	8.163	7.117	6.623	6.116	5.571	4.141
69	17.24	14.39	11.43	10.10	8.658	7.843	6.711	6.135	5.556	5.222	4.124
70	17.24	14.08	10.75	9.050	7.407	6.780	8.270	5.952	5.435	5.025	4.098

Table 1: Frequency, in kHz, measured during coverage characterization using the TS555 interface circuit. The ”Id” is a tag assigned to the measurement during the characterization process.

	Wetness degree (# of drops or full)										
Id	0	1	2	3	4	5	6	7	8	9	Full
61	43.96	35.71	29.41	25.81	21.74	20.10	17.62	16.06	14.98	14.04	8.602
62	44.94	38.10	31.01	27.21	22.47	20.73	18.69	16.67	15.69	14.87	9.05
63	44.94	37.74	32.26	26.85	23.12	21.39	19.32	17.94	17.70	16.88	9.412
64	45.45	38.10	31.50	25.00	22.22	19.90	18.18	16.67	16.39	15.27	9.217
66	41.24	33.61	28.37	24.24	20.83	19.23	17.94	16.06	13.89	12.74	8.386
67	40.82	32.52	24.69	21.16	17.54	15.81	14.34	13.07	12.23	11.46	8.114
68	41.67	31.01	26.14	21.16	18.35	15.87	14.18	12.70	11.43	10.75	8.230
69	38.46	31.50	25.64	21.98	19.05	16.95	15.81	14.23	12.94	12.12	8.772
70	42.11	35.40	27.97	23.53	21.51	19.14	16.46	14.76	13.42	12.20	8.421

Table 2: Frequency, in kHz, measured during coverage characterization of inverter oscillator circuit. The first column contains the number that was assigned to the sensors during the characterization processes, and the first row represents the number of drops dropped on top of the sensing area as shown in Figure 5.1. Data obtained with the oscillator powered at 3.3 V.

	Wetness degree (# of drops or full)										
Id	0	1	2	3	4	5	6	7	8	9	Full
61	48.19	40.40	34.19	29.41	25.48	22.99	20.73	18.69	16.60	15.44	9.132
62	48.78	41.24	33.33	28.57	23.95	21.51	19.80	17.70	16.46	15.75	9.524
63	48.19	40.40	34.19	28.78	24.39	22.60	20.41	18.96	18.60	17.70	9.950
64	48.78	41.24	33.61	26.85	23.39	21.05	19.14	17.54	17.24	16.19	11.08
66	45.45	36.36	30.77	25.97	21.98	20.41	18.69	16.95	14.60	13.51	8.791
67	44.44	35.40	26.14	22.47	18.26	16.67	15.04	13.79	12.66	12.08	8.529
68	45.45	33.61	27.97	22.86	19.32	16.74	14.98	13.47	12.01	11.49	8.639
69	41.24	34.19	27.59	23.39	19.90	17.78	16.81	15.27	13.38	12.82	9.238
70	45.45	38.83	29.63	26.14	22.60	20.41	17.32	15.69	14.08	13.07	8.909

Table 3: Frequency, in kHz, measured during coverage characterization of inverter oscillator circuit. The first column contains the number that was assigned to the sensors during the characterization processes, and the first row represents the number of drops dropped on top of the sensing area as shown in Figure 5.1. Data obtained with the oscillator powered at 5 V.

	Wetness degree (# of drops or full)										
Id	0	1	2	3	4	5	6	7	8	9	Full
61	20.20	18.02	15.75	14.04	12.70	11.59	10.90	10.18	9.479	8.791	5.714
62	18.87	16.74	15.09	13.61	11.98	10.75	9.975	9.009	8.493	7.921	5.714
63	20.00	16.88	14.29	12.74	11.83	11.27	10.99	10.20	9.662	8.989	5.822
64	20.10	17.62	15.50	13.25	11.73	11.05	10.34	9.479	8.889	8.197	5.839
66	19.14	16.67	14.44	13.03	11.87	11.43	10.81	9.975	9.091	8.368	5.340
67	18.96	16.26	14.29	12.12	10.81	10.18	9.756	8.969	8.602	7.968	5.548
68	19.14	15.09	13.11	11.40	10.44	9.615	8.830	8.130	7.692	7.273	5.464
69	19.32	17.62	14.87	12.94	11.40	10.26	9.456	8.696	8.147	7.449	5.464
70	18.35	15.27	13.29	11.76	10.18	9.281	8.639	8.032	7.449	6.861	5.634

Table 4: Frequency, in kHz, measured during coverage characterization of integrator circuit. The first column contains the number that was assigned to the sensors during the characterization processes, and the first row represents the number of drops dropped on top of the sensing area as shown in Figure 5.1. Data obtained with the oscillator powered at 3.3 V.

	Wetness degree (# of drops or full)										
Id	0	1	2	3	4	5	6	7	8	9	Full
61	25.32	22.22	18.78	16.53	14.55	13.29	12.35	11.49	10.58	9.804	6.126
62	22.60	20.10	17.78	15.69	13.84	12.31	10.99	9.901	9.132	8.621	6.107
63	24.69	20.10	16.67	14.60	13.51	12.66	12.35	11.36	10.78	9.901	6.260
64	24.69	21.05	18.10	15.21	13.29	12.46	11.36	10.53	9.685	8.949	6.250
66	22.99	19.61	17.02	14.87	13.51	12.86	12.20	11.08	10.10	9.153	5.690
67	22.86	19.32	16.46	13.89	12.12	11.43	10.78	9.926	9.434	8.715	5.952
68	23.12	17.54	15.04	12.86	11.73	10.64	9.709	8.869	8.351	7.843	5.814
69	23.39	20.94	17.39	14.87	12.74	11.46	10.42	9.390	8.889	8.097	5.814
70	22.10	17.86	15.33	13.16	11.59	10.23	9.662	8.715	8.097	7.339	5.961

Table 5: Frequency, in kHz, measured during coverage characterization of integrator circuit. The first column contains the number that was assigned to the sensors during the characterization processes, and the first row represents the number of drops dropped on top of the sensing area as shown in Figure 5.1. Data obtained with the oscillator powered at 5 V.

Frequency [Hz]	°C
8967	-25
8964	-20
8968	-15
8958	-10
8963	-5
8961	0
8963	5
8969	10
8973	15
8982	20
8994	25
8984	30
8976	35
8978	40
8965	45
8962	50
8966	55
8965	60
8963	65
8960	70
8965	75
8959	80
8960	85
8953	90

(a)

Frequency [Hz]	°C
14999	-25
14961	-20
14942	-15
14923	-10
14799	-5
14700	0
14627	5
14484	10
14435	15
14344	20
14225	25
14356	30
14285	35
14300	40
14100	45
13950	50
13800	55
13500	60
13450	65
13400	70
13320	75
13200	80
13250	85
13220	90

(b)

Table 6: Frequency, in Hz, measured during temperature characterization with the TS555 based interface. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

Frequency [Hz]	°C
20352	-25
20350	-20
20358	-15
20417	-10
20529	-5
20445	0
20434	5
20505	10
20495	15
20481	20
20540	25
20542	30
20550	35
20540	40
20533	45
20540	50
20564	55
20530	60
20525	65
20646	70
20517	75
20415	80
20495	85
20547	90

(a)

Frequency [Hz]	°C
42508	-25
37514	-20
28138	-15
10895	-10
12096	-5
41889	0
41794	5
41595	10
41263	15
41347	20
41081	25
40876	30
40662	35
40536	40
39759	45
39811	50
39815	55
39755	60
39680	65
39166	70
38285	75
38356	80
39045	85
38462	90

(b)

Table 7: Frequency, in Hz, measured during temperature characterization with the inverter-oscillator based interface. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

Frequency [Hz]	°C
7620	-25
7614	-20
7604	-15
7616	-10
7624	-5
7613	0
7600	5
7597	10
7575	15
7565	20
7562	25
7550	30
7546	35
7532	40
7507	45
7511	50
7502	55
7483	60
7470	65
7457	70
7444	75
7442	80
7441	85
7475	90

(a)

Frequency [Hz]	°C
15676	-25
13473	-20
10977	-15
15510	-10
15565	-5
15561	0
15472	5
15418	10
15419	15
15386	20
15384	25
15208	30
15212	35
15196	40
15178	45
15205	50
15117	55
15059	60
15097	65
15028	70
14954	75
14972	80
14985	85
14986	90

(b)

Table 8: Frequency, in Hz, measured during temperature characterization with the integrator based interface. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

Time [ticks]	°C
4256	-25
4256	-20
4254	-15
4257	-10
4259	-5
4231	0
4222	5
4256	10
4257	15
4256	20
4256	25
4257	30
4257	35
4257	40
4259	45
4261	50
4261	55
4262	60
4262	65
4265	70
4266	75
4268	80
4270	85
4271	90

(a)

Time [ticks]	°C
2340	-25
3064	-20
3644	-15
5697	-10
7053	-5
2426	0
2411	5
2420	10
2434	15
2439	20
2455	25
2482	30
2492	35
2502	40
2519	45
2536	50
2545	55
2563	60
2570	65
2582	70
2575	75
2594	80
2584	85
2594	90

(b)

Table 9: Time [ticks] measured during temperature characterization of RC smart sensor. Ticks are the number of times the timer TIM21 counted from the start of the measure up until the voltage on the sensor reached the threshold digitally selected. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

Iterations	°C
1021	-25
1019	-20
1021	-15
1019	-10
1018	-5
1019	0
1017	5
1015	10
1014	15
1008	20
1004	25
1002	30
999	35
994	40
987	45
981	50
972	55
966	60
953	65
945	70
936	75
927	80
918	85
908	90

(a)

Iterations	°C
2340	-25
3064	-20
3644	-15
5697	-10
7053	-5
2426	0
2411	5
2420	10
2434	15
2439	20
2455	25
2482	30
2492	35
2502	40
2519	45
2536	50
2545	55
2563	60
2570	65
2582	70
2575	75
2594	80
2584	85
2594	90

(b)

Table 10: Iterations measured during temperature characterization of charge transfer smart sensor. In (a), data obtained using the short cable and a soldered capacitor in place of the actual sensor; in (b), data obtained with the long cable and the sensor completely dry.

List of Figures

List of Acronyms

ADC analog to digital converter

CAD computer-aided design

CPU central processing unit

HAL hardware abstraction layer

IC integrated circuit

IDE integrated development environment

LDO low drop-out

LL low layer

LSB least significant bit

MCU microcontroller unit

NTC negative temperature coefficient

OpAmp operational amplifier

PCB printed circuit board

PLL phased locked loop

PWM pulse width modulation

TRL technology readiness level