Selected Experiments with an Arduino Board to Teach Analog-to-Digital Conversion

ISSN 1870-9095

João E. M. Perea Martins
Computer Science Department, School of Sciences (FC), São Paulo State University (UNESP), 17033-360, Bauru-SP, Brazil.

E-mail: joao.perea@unesp.br

(Received 02 March 2020, accepted 28 February 2021)

Abstract
This work presents a sequence of selected topics to teach the fundamentals of analog-to-digital conversion, including its operational principles and the analog sensors interfacing. The pedagogical strategy is based on a direct interaction between theory and practices based on simple electronic experiments, and the article expectation is that teachers can use it to provide students with a concrete base to allow them to understand and use this technology in future advanced studies or projects in the experimental physics area.

Keywords: Analog-to-digital conversion, Analog sensors, Data acquisition.

Resumen
Este trabajo presenta una secuencia de temas seleccionados para enseñar los fundamentos de la conversión de analógico a digital, incluidos sus principios operativos y la interfaz de los sensores analógicos. La estrategia pedagógica se basa en una interacción directa entre la teoría y las prácticas basada en experimentos electrónicos simples, y la expectativa del artículo es que los docentes puedan usarlo para brindar a los estudiantes una base concreta que les permita comprender y utilizar esta tecnología en futuros estudios avanzados o proyectos en el área de la física experimental.

Palabras clave: Conversión de analógico a digital, Sensores analógicos, Adquisición de datos.

I. INTRODUCTION

Analog-to-digital converters (ADC) allow the interfacing between analog sensors and processors, which can be applied in different areas such as industry, agriculture, medicine, control, and computing. In addition, they have a high relevance degree in the experimental physics area, where the measurement of physical phenomena is a decisive factor.

There are several articles, sites, and books that address this subject, but there is a lack of works developed with educational concernments focused on beginner physics students. This context motivated the present article that proposes the integration between theory and practices, but avoids extreme cases of excessive formalisms or only superficial practices. Besides, it presents the subject through a logical sequence that can be used as a class guide that may be adapted to different educational realities.

The ADC in-depth study is a complex task because of the large number of technical parameters involved, such as electrical characteristics, error types, noises, architecture, and grounding [1, 2, 3]. Therefore, this work selected fundamental topics that can compose a concrete students' base, which can allow future advanced studies in this area or even motivate students to design their own application systems.

The article starts with an explanation about general sensor concepts, and next it presents selected concepts of the Analog-to-digital conversion, including several details that are shown sequentially. It also proposes practical experiments as a way to reinforce the theoretical learning, and all the hardware and software aspects required by the experiments are detailed in their respective sections.

The presented theory doesn't require previous knowledge in the subject, and therefore even beginning students may follow a class based on the proposed sequence and level. The experiments require only instruments and devices common in electronic laboratories and can be mounted even by students with few practices in electronic.

II. SENSORS

ADCs interface electrical sensors to processors, and therefore the initial concepts to be learned in the classroom are about sensors that represent the first stage in a measurement system.

Sensors are devices that have an internal parameter, such as resistance or capacitance, whose magnitude varies according to an external phenomenon variation, such as temperature or humidity. Sensors have several characteristics and their detailed classification is a complex task because it can include different aspects in fields as physics, chemistry, mathematics and engineering [4], and therefore it composes a scenario that requires a special attention to avoid misunderstandings. This work focuses only on some essential sensor characteristics, and it suggests the International Vocabulary of Metrology (VIM) usage, which can be an interesting source for future students' doubts clarification, and besides it can also be a source for teachers to prepare future classes focused on this area.

The VIM [5] is a trustworthy source of information created by the International Bureau of Weights and Measures (BIPM) and supported by other notorious international institutions as the International Organization for Standardization (ISO) and the International Union for Pure and Applied Physics (IUPAP) [5, 6], and besides it presents the definitions in an objective and summarized way, which is interesting for an initial study that may be later complemented with other references.

The definitions of *sensors* and *transducers* are fundamental, and the VIM defines them as:

1) Sensor: "Element of a measuring system that is directly affected by a phenomenon, body, or substance carrying a quantity to be measured".
2) Measuring Transducer: "Device, used in measurement that provides an output quantity having a specified relation to the input quantity".

There are several types of sensors, such as the mechanicals mercury thermometer and the playful hair tension hygrometers. However, this work focuses only on electrical sensors, which provides an electrical signal as output, and present advantages as the direct output signal processing, storing, and transmission through communication systems.

The effective use of the electrical sensor advantages usually depends on processing systems as computers and microcontrollers that receive the signals. However, processors work only with numbers and can't directly receive the sensor voltage signal. Therefore, the interfacing of an analog sensor to a processor requires an analog-to-digital converter (ADC) that is an electronic device that receives the sensor output voltage and generates a number proportional to it. Figure 1 exemplifies an electrical measurement system, where the sensor and the ADC are the initial parts.

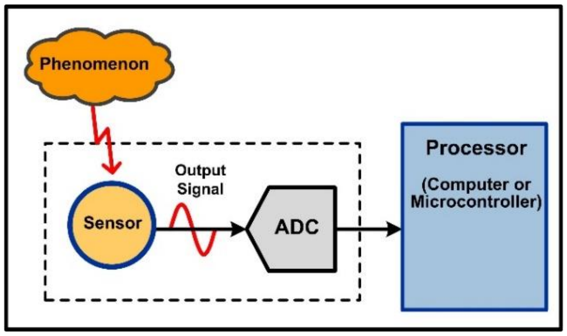


FIGURE 1. A basic sensing system structure.

III. ACTIVE AND PASSIVE SENSORS

The powering requirement classifies sensors as either *active* or *passive*, where passive sensors naturally generate an electrical output signal according to the measured phenomenon magnitude, without an external power source. For example, thermocouples are sensors that convert heat into electricity and naturally generate a small output voltage signal according to the ambient temperature. In contrast, the active sensors don't generate directly output electrical signals, and therefore require an external power source [4]. It is important to emphasize in the classroom that there isn't a final consensus about these terms, and there are authors that use them with exactly opposite meanings [7].

These terms are associated with the terms *transducer* and *measuring-transducer* that have different meaning [5] and should not be confused. Transducer is a device that converts energy from one form to another, and therefore "energy conversion" is its key idea. Measurement-transducer is a specific type of device designed exclusively for sensing and measurement applications, and therefore "sensing" is its key idea.

Figure 2 shows that active and passive sensors can compose a measuring-transducer design. However, only a passive sensor converts energy, and therefore it classifies this sensor type as both, transducer and measuring-transducer. On the contrary, an active sensor only modulates the power from an external source without energy conversion, and therefore, it is a measuring-transducer, but it isn't a transducer.

Note that the VIM definition of the measuring-transducer doesn't use the term "conversion" and only emphasizes the relation between input and output quantities.

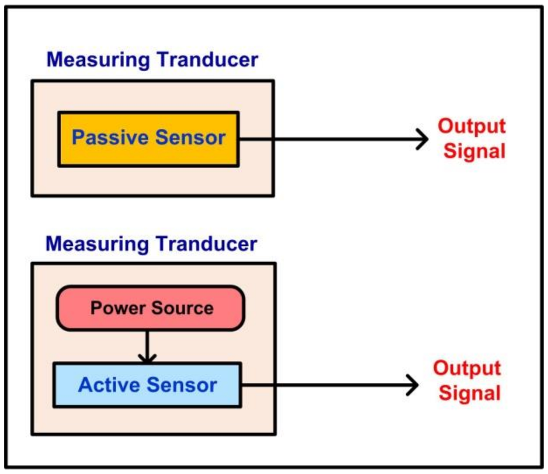


FIGURE 2. Measuring Transducers with active and passive sensors.

This context may seem confusing for beginner students, but figure 3 summarizes it as a visual alternative for an easier explanation in the classroom.

The measuring-transducer with an active sensor in the figure 3 shows that the output energy is in the same form of the power source energy, but its intensity varies according to the external measured phenomenon.

It is also interesting to show in the classroom that the use of transducers is very common in everyday life, and perhaps many people haven't perceived it. For example, toasters, immersion heaters, and coffee machine are home appliances whose operation is based on resistive elements that convert electrical energy into heat, according to the heating Joule's law, blenders and hand mixers have motors that convert electrical energy in mechanical power, and solar panels (photovoltaic cells) convert *s*unlight into electricity.

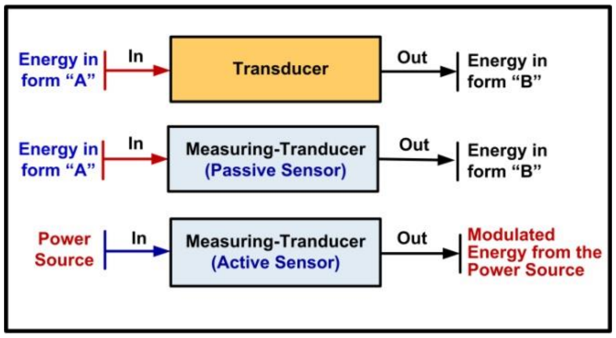

Energy in form "A"
In
Transducer
Out
Energy in form "B"
Energy in form "A"
In
Measuring-Tranducer (Passive Sensor)
Out
Energy in form "B"
Power Source
In
Measuring-Tranducer (Active Sensor)
Out
Modulated Energy from the Power Source

FIGURE 3. Measuring-transducers and energy conversion.

The VIM is an important guide and the BIPM a notorious organization, but there are other organizations that also propose definitions and standards. It means that may occur small differences between the terms from different organizations, which usually occur because of the focus and specificities of each area. There great efforts to make definitive official definitions, but the measurement is one of the experimental physics bases, which includes several sub areas, and therefore it creates a very dynamic context that yet has been evaluating, as proved in different works [8, 9, 10].

The measurement terms may become more critical when people use definitions based mainly on their previous experiences [11] or based on practical approaches rather than theoretical principles [12], which can lead readers to misinterpretations. It justifies the use of an official guide as the VIM, which is based on solid studies and previous discussions. This article emphasizes that the discussion about these technical definitions in the classroom is extremely relevant to provide the students with a critical sense that may allow them to understand and discern the content of different technical articles and books.

IV. ANALOG SENSORS

Figure 4 shows that sensors are classified according to their electrical output signal shape as *analog* or *digital*, which influences the electronic sensing system design.

This classification becomes more consistent when associated with the signal domain in the amplitude and time. Analog signals are continuous in the amplitude because they can assume any infinite real value inside a range. An analog signal also exists at any time, and therefore is also continuous in the time domain. On the contrary, discrete signals can assume only some specific values inside a range, and therefore they are discrete in the amplitude domain. Figure 4 shows the analog and digital signals, where the amplitude domain ranges are [A, B] and [C, D], respectively. This digital signal is referred as binary because it assumes only two values (high or low) in the time domain.

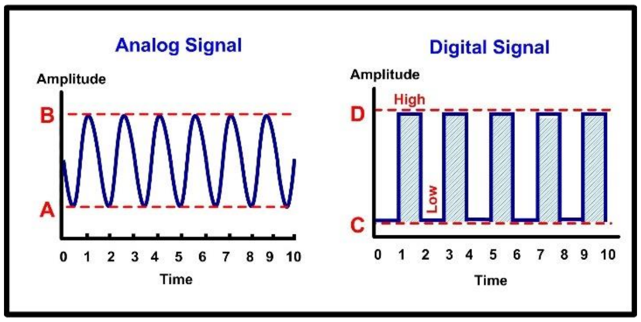


FIGURE 4. Analog and Digital Signals.

Figure 5 shows a simple experiment to demonstrate an analog sensor in the classroom, using only the analog temperature sensor LM35 and a voltmeter. It is an inexpensive sensor that requires only a power source with any value between 4 V and 30 V to works, and its *sensitivity* that represents the relation between the measured phenomenon (temperature) and the sensor output signal (voltage) is 10 mV/°C. For example, for an ambient temperature at 26.5 °C, the voltmeter will show a measured voltage of 265 mV.

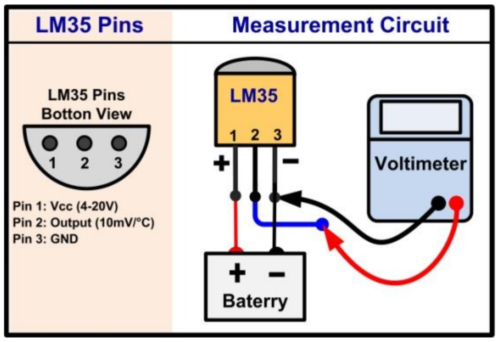


FIGURE 5. Experiment for the LM35 analog temperature sensor demonstration with a voltmeter.

The LM35 is an analog sensor and therefore provides output signals continuously at any instant. Its sensitivity is 10 mV/°C, and it measures temperatures in a range from 0 to 150 °C. Therefore, its output voltage signal (x) can assume infinite real values inside a range from 0 to 1.5 V, which is expressed as $\{x \in \mathrm{R} \mid 0 \leq x \leq 1.5\}$.

V. THE ANALOG-TO-DIGITAL CONVERSION PRINCIPLE

Figure 6 shows the operational idea of an analog-to-digital converter (ADC), where it samples an input voltage signal to measure its amplitude (voltage value), and generates an output integer number proportional to it. Next, if necessary, a processor can handle this integer number [1, 4, 13, 14].

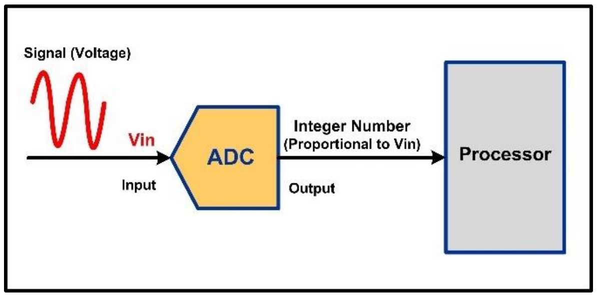


FIGURE 6. The ADC idea of conversion, which associate a voltage value to an integer number.

The first fundamental ADC parameter is the relation between the input voltage (*Vin*) and the generated output number (*Nout*) expressed as:

$$\text{Nout} = integer\left(\frac{Vin}{CW}\right) \quad (1)$$

CW is the called *Code Width*, which is also referred as *ADC resolution* or LSB *(Least Significant Bit)* and defines the smallest voltage input variation that the ADC can detect, computed as:

$$CW = \frac{Vref}{2^N}. \quad (2)$$

Where: *Vref* is the ADC *reference voltage*. *N* is the *ADC number of resolution bits.*

Vref is fixed by the user and acts as a reference for calculating *CW*, and besides it also defines the maximum allowed Vin value, where *Vin* ≤ *Vref*.

N is a constant fixed by the ADC manufacturer and defines how many output integer numbers it can generate. The *N* unit is *bit*, which is a unit also used in other different contexts of computing.

For example, for an ADC with *N* of 3 bits, and *Vref* fixed at 1.0 V, *Nout* could assume up to 2^3 or 8 output integer values in a range from 0 to 7, and *CW* would be computed as:

$$\text{CW} = {}^{1.0V}/_{2^3} = 0.125\,V\,.$$

The terms "number of resolution bits" and "ADC resolution" have different meanings, but frequently the term "resolution" appears alone with no complements, and therefore these cases require a special attention about the context where they are presented.

Figure 7 shows the relation between *Vin* and *Nout* for an symbolic ADC with *N* at 3 bits. The *Nout* variations occur only when *Vin* varies at steps larger than *CW* that is 0.125 V in this example. For example, for any *Vin* value between 0.375 V and 0.50 V the Nout value is fixed at 3, and *Nout* will assume another value only when Vin becomes less than 0.375 V or greater than 0.50 V.

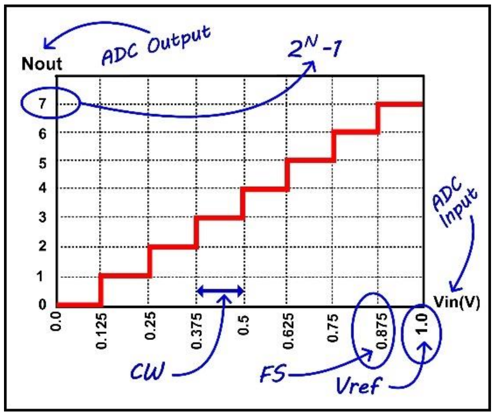


FIGURE 7. ADC relation between input and output.

Figure 7 also shows the maxim allowed *Vin* equals *Vref*, but the maximum detectable *Vin* value is called *Full Scale* (FS) and computed as:

$$FS = Vref - CW. \quad (3)$$

Figure 8 exemplifies a simple experiment with an Arduino board to verify the relation between *Vin* and *Nout*. The 1 KΩ and linear potentiometer is powered with the Arduino source output (5V) and the potentiometer output (central pin) provides an output voltage that can be adjusted between 0 and 5.0 V, and it is connected to the Arduino analog input (A0) to acts as *Vin*.

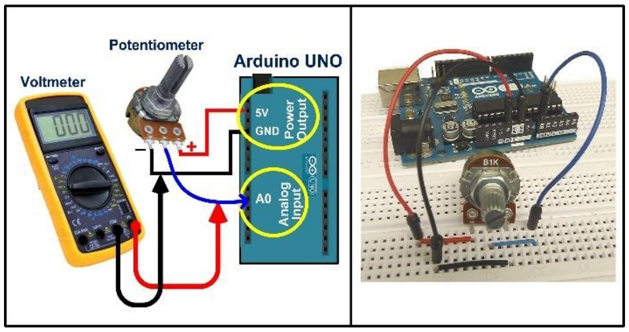
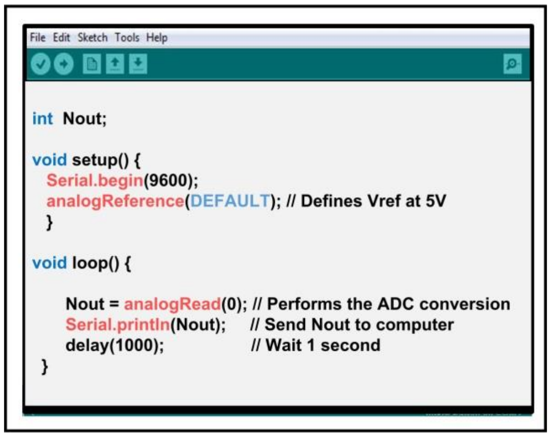


FIGURE 8. Experiment to verify the Arduino ADC operation.

The Arduino Uno has an internal ADC with 10 bits of resolution and an internal *Vref* source fixed at 5.0 V. Therefore, its resolution is computed as:

$$cw = {}^{5V}\!/_{2^{10}} = 0.00488\,mV \cong 5\,mV.$$

Figure 9 shows the Arduino program used in this experiment, which verifies *Vin* and send the correspondent *Nout* to a computer. More specifically, it performs the pin *A0* input signal conversion every second and sends the result (*Nout*) to a computer that shows it in the window called "serial monitor", which is part of the Arduino software (IDE).



```
int  Nout;

void setup() {
   Serial.begin(9600);
   analogReference(DEFAULT); // Defines Vref at 5V
   }

void loop() {

       Nout = analogRead(0); // Performs the ADC conversion
       Serial.println(Nout);    // Send Nout to computer
       delay(1000);             // Wait 1 second
  }
```

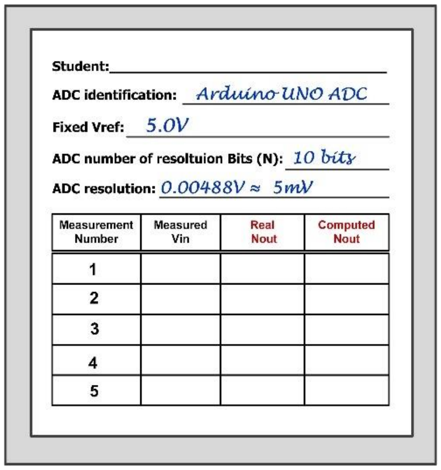
FIGURE 9. Arduino program for the figure 8 experiment.

For the figure 8 experiment, the students must:

1. Turn the potentiometer to any position.
2. Measure the potentiometer output voltage with a voltmeter, which acts as the *Vin* connected to the Arduino analog pin *A0*.
3. Calculate de theoretical *Nout* value according to the equations 1 and 2.
4. Compare the theoretical computed value *Nout* with the real value shown on the computer Arduino window.

Figure 10 shows an example of a laboratory report to be filled by students, where they can note theoretical and real *Nout* values for later analysis.

Theoretical and experimental *Nout* values may present small differences in this experiment, which is a usual problem cause by factors such as noise, electrical grounding, *Vref* inaccuracy, *Vin* source oscillation, and mathematical rounding. In fact, this problem can be attenuated or even eliminated with the use special advanced techniques that can include hardware and software aspects.



Student:____________________

ADC identification: Arduino UNO ADC

Fixed Vref: 5.0V

ADC number of resoltuion Bits (N): 10 bits

ADC resolution: 0.00488V ≈ 5mV

Measurement Number	Measured Vin	Real Nout	Computed Nout
1			
2			
3			
4			
5			

FIGURE 10. Laboratory report for the figure 8 experiment.

V. THE ANGULAR DISPLACEMENT MEASUREMENT

Another experiment similar to the figure 8 hardware structure is for the *angular displacement* measurement, which represents the angle which a point or line rotated with respect to a fixed reference axis. In the present case, the experiment verifies the potentiometer axis angular displacement.

Figure 11 exemplifies the internal potentiometer structure, where the angular movement of the axis causes a proportional internal slider movement on a resistive element. The slider is connected to the central potentiometer pin (pin 2), and therefore the resistance value between the pins 1 and 2, and between the pins 2 and 3, vary according to potentiometer axis movement.

Assuming the pin 1 is connects the ground and the pin 3 connects the Vcc (5V). When the axis turns completely to the left it connects pins 2 and 1, then the pin 2 output voltage will be 0V. As the axis rotates from left to right, pin 2 gets closer to pin 3 and its output voltage increases. This means that by increasing the axis angle, the output voltage also increases proportionally for a linear potentiometer.

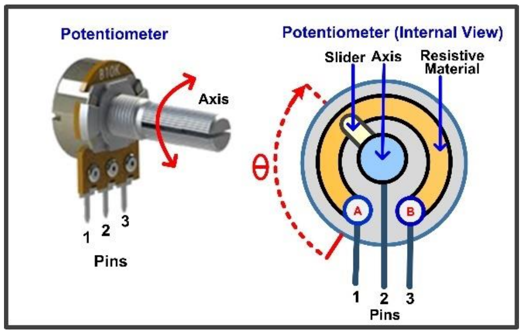


FIGURE 11. A potentiometer and its internal structure.

Figure 12 shows the experiment mounting for the potentiometer axis angular displacement measurement with a 1 KΩ linear potentiometer, where its axis is fixed through a hole in the center of a 360° protractor. A knob with an indication line covers the potentiometer axis and allows the angle verification on the protractor surface.

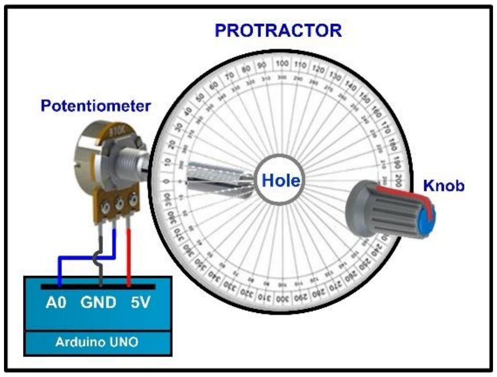


FIGURE 12. The angular displacement experiment mounting.

For conventional potentiometers, the maximum axis rotation angle (*Amax*) is smaller than 360° and it can be verified in the potentiometer data sheet or experimentally with the protractor. The sensitivity (*Sp*), that defines the between the displacement angle (*Ad*) and the potentiometer output voltage at its pin 2 (*Vp*) is:

$$Sp = {}^{Vcc}/_{Amax}. \tag{4}$$

In this work *Vcc* was fixed at 5V and the linear potentiometer used had an Amax of 290°, and therefore the *Sp* was 0.0172V/°, or 17.2 mV per degree.

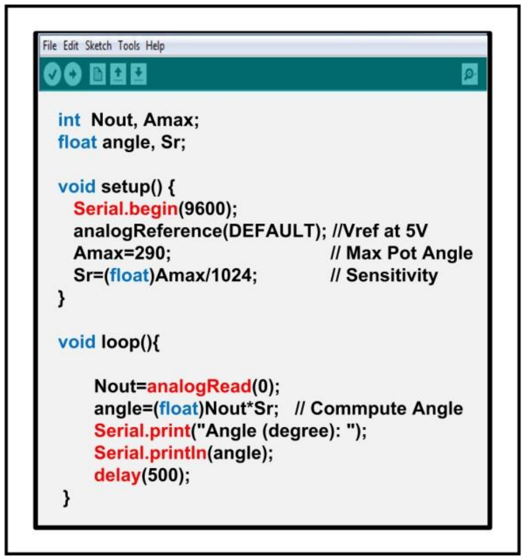
The Arduino ADC resolution at 5mV defines the measurement resolution (*Sr*) that represents the smallest detectable angle variation computed as:

$$Sr = {}^{5mV}/_{Sp}. \tag{5}$$

In this example, *Sr* would be 0.25°, and it means that for each axis rotation of 0.29° the ADC output value (*Nout*) varies by one unit.

Figure 13 shows the Arduino program that computes the potentiometer axis angular displacement. In the sixth program line there is a statement "*Amax*=290" that defines the maximum angle of the used potentiometer. The number 290 must the changed according to the user's potentiometer angle.

Besides the possible theoretical and experimental *Nout* differences explicated early, the potentiometer may also influence this relation because its internal characteristic, such as rotational noise or small linearity imperfections along the resistive element. In fact, there are accurate and precise potentiometer models that operate as rotary displacement sensors and allow the design of high-performance measurement system. However, they have a higher cost and aren't usual in education laboratories where a simple and common linear potentiometer is usually satisfactory for educational demonstrations.



```
int  Nout, Amax;
float angle, Sr;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT); //Vref at 5V
  Amax=290;                        // Max Pot Angle
  Sr=(float)Amax/1024;             // Sensitivity
}

void loop(){

    Nout=analogRead(0);
    angle=(float)Nout*Sr;   // Commpute Angle
    Serial.print("Angle (degree): ");
    Serial.println(angle);
    delay(500);
 }
```

FIGURE 13. Program to verify the potentiometer axis angular displacement.

VI. THE TEMPERATURE MEASUREMENT

Figure 14 shows the hardware of an experiment that is also similar to the figure 8 hardware, but now the potentiometer is replaced by the analog temperature sensor model LM35.

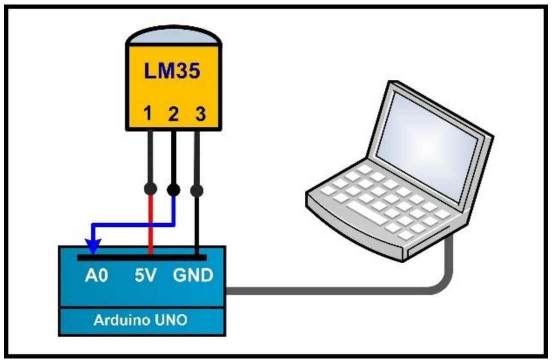
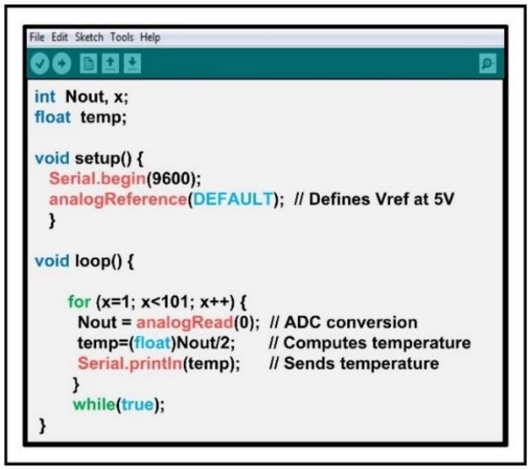


FIGURE 14. The temperature measurement with a LM35 sensor and an Arduino board.

Figure 15 shows the Arduino program for this experiment, which defines the *Vref* at 5.0 V, performs 100 temperature measurements and sends them to a computer. Later, the students must copy the data from the computer screen to design the graph with a spreadsheet.

Figure 16 shows the measurement results for an ambient temperature at 24.5 ºC. The arithmetic measurement mean was 25.0 ºC, with a standard deviation of 0.44 ºC, coefficient of variation of 1.8%, minimum value of 23.5 ºC, and maximum value of 26.4 ºC.



```
int Nout, x;
float temp;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT); // Defines Vref at 5V
  }

void loop() {

    for (x=1; x<101; x++) {
     Nout = analogRead(0);  // ADC conversion
     temp=(float)Nout/2;    // Computes temperature
     Serial.println(temp);  // Sends temperature
    }
    while(true);
 }
```

FIGURE 15. Arduino program for the figure 14 experiment.

This experiment used the sensor version LM35DZ whose *measurement uncertainty* (θ), which defines the maximum allowed result variation, is ±1.5 °C [15]. The figure 16 values are inside this limit and it proves the experiment coherence. This degree of variation may be acceptable or not, which will depend on the requirements of each real application.

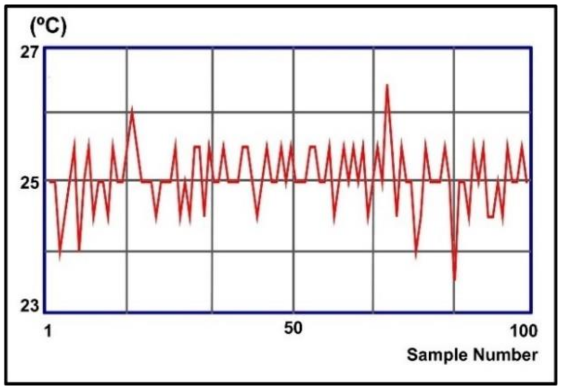


FIGURE 16. Temperature measurement with a LM35 sensor.

Figure 16 is also a way to introduce in the classroom the concepts of *measurement accuracy* and *measurement precision*, which are different and shouldn't be confused.

The measurement accuracy is a qualitative concept associated mathematically with the uncertainty or *difference*. A good accuracy level indicates a small uncertainty value, which represents only a small difference between the true value and measurement value [16, 17].

The measurement precision is associated with the idea of repeatability and indicates the results repeatability degree for several measurements of the same phenomenon magnitude at the same measurement conditions. In this case, a good precision level indicates that the measurement results are repetitive. It is also a qualitative parameter that mathematically is usually expressed according to the data standard deviation or coefficient of variation [16, 17].

Figure 17 exemplifies the ideas of accuracy and precision through four hypothetical temperature sensors with an uncertainty of ±1.5 °C, measuring a true temperature of 25 °C. Note that, philosophically, there aren't perfect (completely correct) measurements even with high performance measurement devices. Therefore, any measurement will always have a margin of error, even if it is insignificant for most applications. It means that the concept of "true value" based on a previous measurement becomes relative, and it could be conceptually better referred as "value assumed as true".

The sensor "*A*" results (navy) don't vary and it represents an excellent precision level, besides they are very close to the true value, which means an excellent accuracy. Sensor "*B*" (cyan) has excellent precision, but its results are far from the true value, and therefore its precision is poor. Sensor "*C*" (green) presents a middle level of accuracy and precision, while the sensor "*D*" (purple) presents a poor degree for both parameters. Therefore, the sensor "*A*" is the best. However, the sensor "*B*" presents a constant difference in relation to the true value and its measurement results can be mathematically later corrected to achieve a good accuracy.

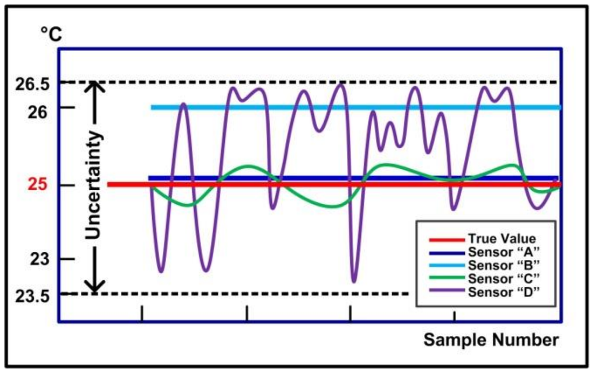
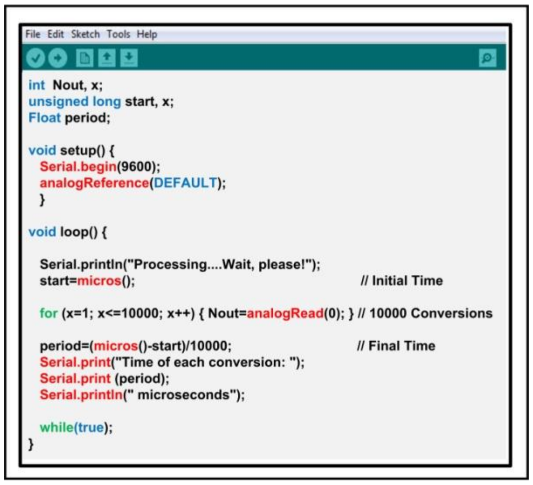


FIGURE 17. Analysis of the accuracy and precision concepts.

VII. THE ADC CONVERSION TIME

Figure 18 shows an Arduino program to verify its ADC *conversion time* (*Tc*), which is the time required to sample the input signal and generate the output number. *Tc* represents the period between two successive ADC conversions, and it may be a problem because all the new input signal variation during it will be lost, and therefore it should be as small as possible. Suppose that during this period a rare event occurs, then it will not be detected by the system as the ADC will not sample the sensor signal relative to the event.



```
int  Nout, x;
unsigned long start, x;
Float period;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);
  }

void loop() {

  Serial.println("Processing....Wait, please!");
  start=micros();                                   // Initial Time

  for (x=1; x<=10000; x++) { Nout=analogRead(0); } // 10000 Conversions

  period=(micros()-start)/10000;                    // Final Time
  Serial.print("Time of each conversion: ");
  Serial.print (period);
  Serial.println(" microseconds");

  while(true);
}
```

FIGURE 18. Program to verify the Arduino ADC conversion time.

The figure 18 program performs consecutively 10000 analog-to-digital conversions of the input signal at the pin *A0*. After the conversions, the program prints the arithmetic mean time required for each ADC conversion, which was 112 μs in this work with an Arduino Uno board.

The conversion time question is also an opportunity for the physics teacher to introduce the concepts of *sampling* and *quantization*, which are exemplified in figure 19.

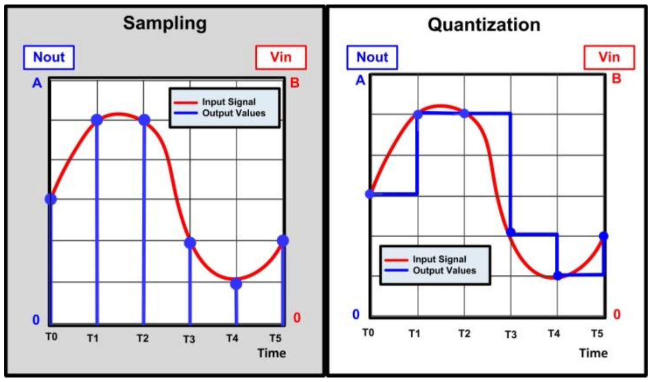


FIGURE 19. Sampling and quantification processes.

The *sampling* is mathematically interpreted as the transformation of a function continuous in the time domain in a function discrete in the time domain. Physically, it represents the process to verify the amplitude (voltage value) of an analog input signal at regular intervals of time, called *sampling interval* (*Ts*).

The sampling interval includes the conversion time (*Tc*) and other periods that may be required for the information processing or even periods defined by the user according to its technical interests, and therefore *Ts* ≥ *Tc*. It also defines the maximum number of signal conversions per period, which is called *sampling rate* or *sampling frequency* (*Fq*), and computed as:

$$Fq = {}^{1}/_{Ts}. \tag{6}$$

The *quantization* associates an input signal subrange to a fixed discrete (integer) output value. For example, in the figure 7, any input signal value between 0.625 V and 0.75 V is associated with the integer number 5. Therefore, all the infinite real numbers in the range [0.625 V, 0.750 V] are quantified as the integer number 5. In this process, the quantified integer number remains as the result until the next signal conversion.

VIII. SIGNAL ALIASING

Aliasing is an effect that occurs when the conversion results are used to represent the input signal shape, but it occurs without the fidelity necessary to allow the correct representation.

According to the Nyquist theorem, the maximum input signal frequency (*Fin*) allowed for an efficient sampling depends on the sampling frequency (*Fq*) and their relation is expressed as:

$$Fin \leq {}^{Fq}/_{2}. \tag{7}$$

Figure 20 exemplifies the aliasing problem, where the red line represents the true input signal, the black dots represents the ADC output values in intervals *Ts*, and the blue line represents the line drawn with the ADC results (black dots). Note that the final result (blue line) differs completely from the true red line.

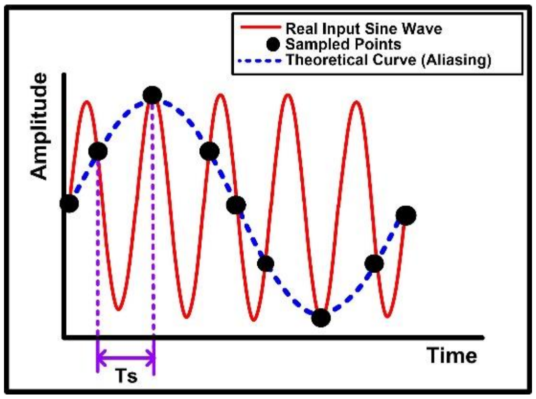


FIGURE 20. Aliasing exemplification.

This work proposes an experiment to verify the aliasing effect in the classroom, which requires a signal generator and an Arduino board, as shown in figure 21. If the signal generator is reliable and has a display that shows exactly the output signal frequency, then the oscilloscope usage would not be mandatory.

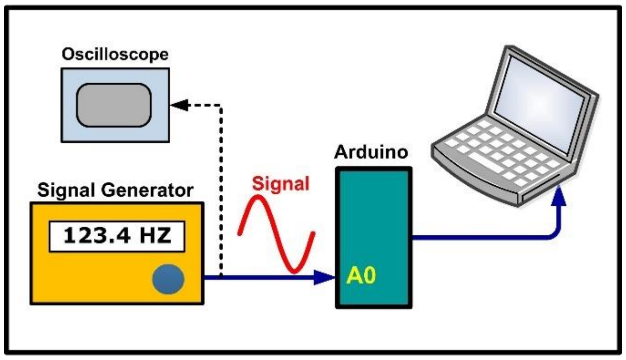


FIGURE 21. Experiment to verify the real aliasing effect.

The figure 21 experiment fellows the steps:

1. Adjust the signal generator output as a sine wave with frequency and amplitude defined by the students.
2. Use the figure 22 program to perform the input signal sampling with the Arduino ADC.
3. Plot the input signal (sine wave) graph with the original input frequency and amplitude.
4. Plot the graph with the ADC data.
5. Overlaps both graphs.

Figure 22 shows the Arduino program that performs the data acquisition. For each sampling, the program performs the ADC conversion, the respective voltage value computation and this value sending to a computer. These three operations time represents the sampling time (*Ts*), which is also shown after the data acquisition process.

In this experiment with an Arduino Uno the sampling time (*Ts*) was about 5.8ms, which represents a *Fq* of 172 Hz. Therefore, the maximum allowed input frequency (*Fin*) is 86 Hz. This program performs 10000 signal conversions, which represents a data acquisition for a period at about one minute, but it can be changed in the “for” line, according to the user interests.

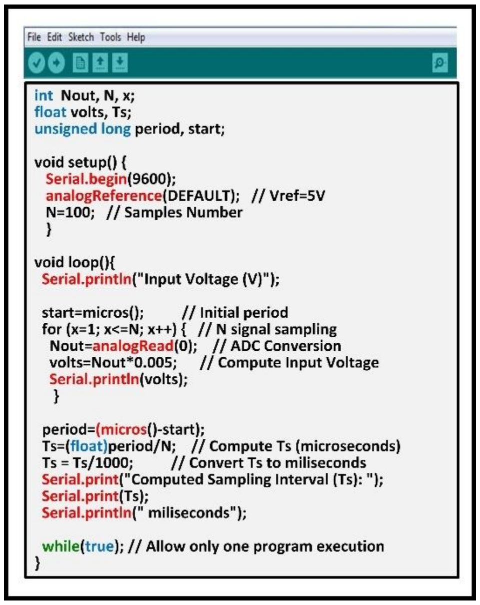
After the program execution, the students must note the *Ts* value and highlight the data on the computer screen with the mouse and copy them to the computer processing software. This software can be a spreadsheet such as the Excel, but the use of mathematical software as the Matlab or the free of charge Octave may be a motivational factor in the classroom.

The student must plot the sine wave and the ADC data graphs with Cartesian coordinates (*x*, *y*) and next overlaps them. These operations are explained in the next paragraphs.

The first graph represents the signal generator sine wave (*ys*) that is expressed as:

$$ys = A\sin\big(fr * (xs - B)\big) + C. \tag{8}$$

Where: *fr* is the signal frequency, *xs* is the horizontal axis range in radians. *A* is the signal amplitude. *B* is the horizontal shift. *C* is the vertical shift.



```
int  Nout, N, x;
float volts, Ts;
unsigned long period, start;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);   // Vref=5V
  N=100;   // Samples Number
  }

void loop(){
 Serial.println("Input Voltage (V)");

 start=micros();        // Initial period
 for (x=1; x<=N; x++) {   // N signal sampling
  Nout=analogRead(0);    // ADC Conversion
  volts=Nout*0.005;     // Compute Input Voltage
  Serial.println(volts);
   }

 period=(micros()-start);
 Ts=(float)period/N;    // Compute Ts (microseconds)
 Ts = Ts/1000;         // Convert Ts to miliseconds
 Serial.print("Computed Sampling Interval (Ts): ");
 Serial.print(Ts);
 Serial.println(" miliseconds");

 while(true); // Allow only one program execution
}
```

FIGURE 22. Arduino program for the figure 21 experiment.

The *A* parameter is adjusted according to the sine wave voltage. The parameters *B* and *C* move the graph on the horizontal and vertical coordinates, which allow the later overlapping with the ADC values graph.

The second graph represents the ADC data set (*yd*), whose content is the ADC results that were sent from the Arduino to the computer, and later copied by the students from the computer screen.

The *yd* data set was originally defined as a function of the time *Ts*, and therefore it must be redefined as radians to allow the overlapping with the sin wave graph. Remember that 1s = 2πrd, and therefore a simple rule of three gives *Ts* in radians (*Tr*) as:

$$Tr = Ts\ 2\pi\,. \tag{9}$$

The horizontal axis to plot the *yd* data is called ‘*xd*’ in this work, and it is defined as:

$$xd\ =\ \{0, Tr, 2Tr, 3Tr, \dots, (N-1)\,Tr\}. \tag{10}$$

Where *N* is the number of ADC measurements, which is 10000 in the figure 22 program.

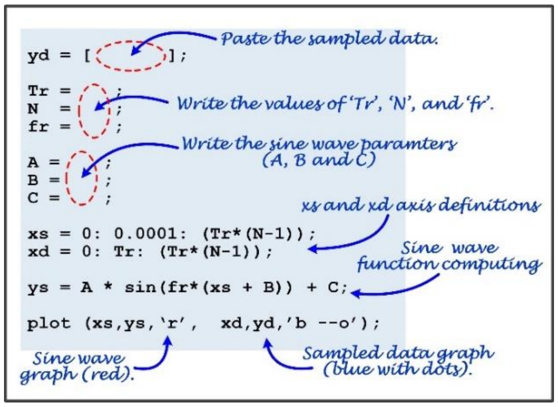
The sine wave graph based on *ys* must be plotted on a horizontal axis (*xs*) that follows the *xd* size, and therefore it is defined as:

$$xs\ =\ \{0, \Delta\text{t}, 2\Delta\text{t}, 3\Delta\text{t}, \dots, (N-1)\,\Delta\text{t}\}. \tag{11}$$

Where Δt is a value fixed by the user and must as small as possible to ensure a good sine wave graph resolution.

Assuming that the sine wave graph coordinates (*xs*, *ys*) and ADC data graph coordinates (*xd*, *yd*) were defined above, the students can use these definitions to plot and overlap both graphs. Figure 23 shows the commands sequence to perform all these operations, step-by-step, in the Matlab and Octave software.

The figure 23 commands represent a simple sequence of command lines, written directly by the students in the Octave command window. Therefore, it doesn't require programming experience and the students must only copy these commands, filling the respective data spaces. After the command "plot" the Octave will plot both graphs overlapped, similarly to the figures 24 and 25.



```
yd = [        ];

Tr =    ;
N  =    ;
fr =    ;

A =   ;
B =   ;
C =   ;

xs = 0: 0.0001: (Tr*(N-1));
xd = 0: Tr: (Tr*(N-1));

ys = A * sin(fr*(xs + B)) + C;

plot (xs,ys,'r',  xd,yd,'b --o');
```

Paste the sampled data.
Write the values of 'Tr', 'N', and 'fr'.
Write the sine wave paramters (A, B and C)
xs and xd axis definitions
Sine wave function computing
Sine wave graph (red).
Sampled data graph (blue with dots).

FIGURE 23. Commands of Matlab or Octave for the data processing of the figure 21 experiment.

After the overlapped graph plotting, the students must analyze the graph to verify the *A*, *B*, and *C* values that will allow a correct fitting. Usually, more than one attempt is required. A start may be with *A* equals 1, and *B* and *C* equals 0, which represents the equation 8 simplification.

The aliasing analysis requires a special attention because the signal generator, the oscilloscope and the Arduino are devices that have an accuracy degree that can interfere with the process.

This work verified the aliasing using two input sine wave signals at 13 Hz and 350 Hz, which are smaller and higher than the *Fq* of 172 Hz.

Figure 24 shows the 13 Hz overlapping, where the sampled data (blue line) overlaps perfectly the real input signal (red line).

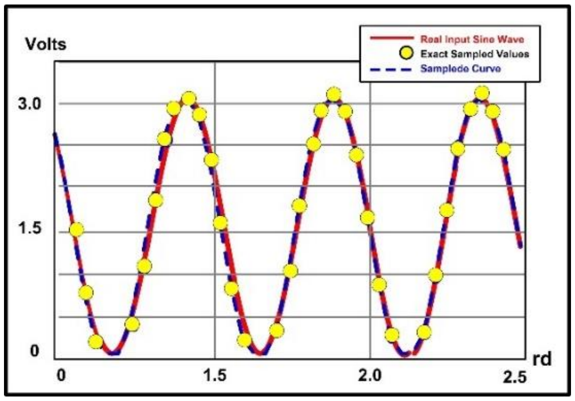


FIGURE 24. The curves overlapping for *Fin* at 13 Hz.

Figure 25 shows the overlapping with the input sine wave (*Fin*) at 350 Hz, which is greater than *Fq*/2 and therefore will present the overlapping problem. The figure 25 sampled data graph (blue line) generated a signal completely different from the real signal (red line).

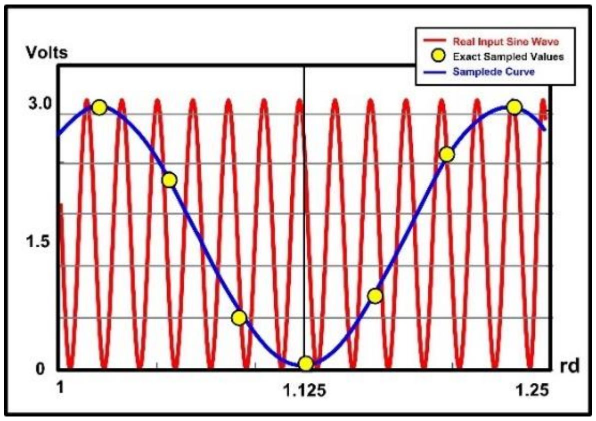


FIGURE 25. The curves overlapping for *Fin* at 350 Hz.

IX. THE ADC REFERENCE VOLTAGE

Equations 1 and 2 show the *Vref* influence on the ADC resolution, and therefore *Vref* has a high relevance in the conversion process. The Arduino Uno has an internal ADC that accepts three different options for *Vref*, which can be defined by software through the commands:

1. *analogueReference(DEFAULT)*: It fixes *Vref* at 5V, using an internal Arduino source to provide this value.
2. *analogueReference(INTERNAL)*: It fixed *Vref* at 1.1V, using an internal Arduino source to provide this value.
3. *analogueReference(EXTERNAL)*: It fixes *Vref* at any value up to 5 V, which is derived from an external source connected to the Arduino pin called AREF.

For a *Vref* at 5.0 V, the maximum *Vin* is 5.0 V and the Arduino ADC resolution is 4.9 mV, but for a *Vref* at 1.1 V the ADC resolution is 1.07 mV. The smaller ADC resolution has the advantage of detecting smaller input signal variations, which means the detection of smaller phenomenon variations.

The internal *Vref* advantage is that it doesn't require any additional hardware, but usually it is less accurate than the *Vref* provided by external chips called *reference voltage*, such as the MCP1541 or the LM4040 that can provide stable and accurate voltages at very specific values as 2.048 V or 4.096 V. For an Arduino, the ADC resolution with *Vref* at 4.096 V is 4.096V/1024, or 4 mV, which also avoids mathematical rounding.

This work proposes an experiment to verify the *Vref* influence on the measurement using the Arduino ADC with two *Vrefs*. Figure 26 shows the experiment that heats slowly the environment around a LM35 sensor with a small air heater. The measurement of an increasing temperature allows an easier comparison between both results, and in the absence of the air heater an incandescent bulb can provide the heat.

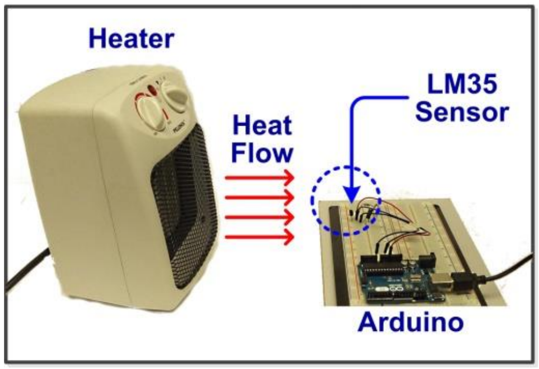
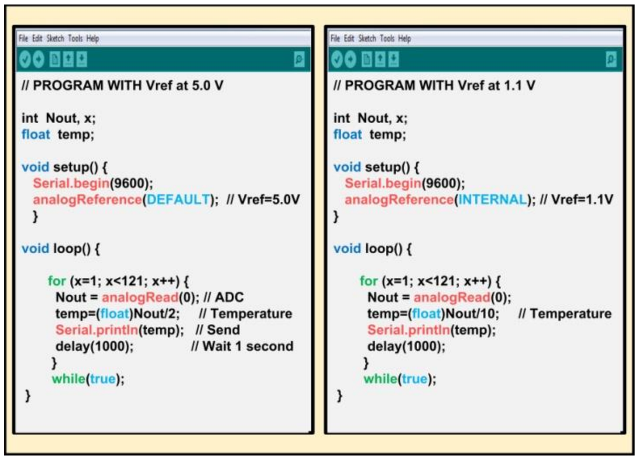


FIGURE 26. Experiment to verify the VREF influence on the temperature measurement of an air heating process.

This experiment uses the figure 26 program that measures the temperature with intervals of one second, during two minutes. However, the students must perform the experiment twice, one time with *Vref* at 5.0 V and another with *Vref* at 1.1 V.

The figure 27 program shows two programs associated with this experiment. The fist program uses the ADC *Vref* at 5.0 V, and the second program uses 1.1 V. The *Vref* variation requires an alteration in the program line that computes the temperature.



```
// PROGRAM WITH Vref at 5.0 V

int  Nout, x;
float  temp;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);  // Vref=5.0V
  }

void loop() {

     for (x=1; x<121; x++) {
      Nout = analogRead(0); // ADC
      temp=(float)Nout/2;     // Temperature
      Serial.println(temp);   // Send
      delay(1000);            // Wait 1 second
     }
     while(true);
}
```

```
// PROGRAM WITH Vref at 1.1 V

int  Nout, x;
float  temp;

void setup() {
  Serial.begin(9600);
  analogReference(INTERNAL); // Vref=1.1V
}

void loop() {

     for (x=1; x<121; x++) {
      Nout = analogRead(0);
      temp=(float)Nout/10;     // Temperature
      Serial.println(temp);
      delay(1000);
     }
     while(true);
}
```

FIGURE 27. Programs for the figure 26 experiment.

Figure 28 shows the measured temperatures in the experiment using the ADC with *Vref* at 5 V and 1.1 V. Both measurements show the temperature variation from 26 to 35 ºC, but the measurement with *Vref* at 1.1 V presented a small level of oscillations.

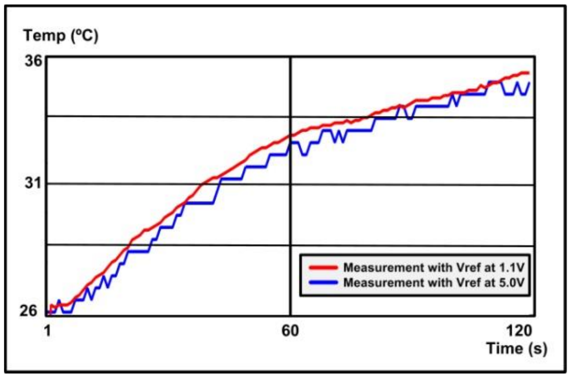


FIGURE 28. Temperature measurement using two ADC *Vref.*

X. THE SYSTEM RESOLUTION

A small ADC resolution allows the detection of small input voltage variations, which represent the detection of small variations in the measured phenomenon. However, the sensor and the ADC operate in association to compose a measurement system that has its own specific resolution, called *measurement system resolution*.

Sensors have a parameter called *sensor resolution*, which represents the smallest phenomenon amount that a sensor can detect. The LM35 can detect small temperatures but the Arduino ADC can't recognize signals smaller than 5 mV, which are proportional to 0.5 ºC. Therefore, part of the LM35 sensor efficiency to detect small temperature variations is lost due to the ADC resolution. For example, in this case, supposing an initial temperature at 20 ºC that is increasing. As long as it is higher than 20 ºC and smaller than 20.5 ºC, the ADC with a resolution of 5 mV will not recognize these variations and will keep recording the initials 20 ºC until the temperature reaches 20.5 ºC.

Table I exemplifies the system resolution for different ADC number of resolution bits (*N*), using a temperature sensor with sensitivity of 10 mV/ºC and an ADC *Vref* fixed at 5.0 V.

TABLE I. System resolution exemplification with different ADC.

ADC - N	ADC Resolution	System Resolution
8 bits	20 mV	2.0 ºC
10 bits	5 mV	0.5 ºC
12 bits	1.2 mV	0.12 ºC
16 bits	76 μV	0.008 ºC

The ideal system resolution is the smallest possible to allow the detection of small measured phenomenon variations. However, small electrical signals may be more susceptible to the noise influence, and therefore its treating may require electronic devices more complexes and accurate, which usually present a higher cost.

XI. THE SIGNAL CONDITIONING

There are cases where the measurement system may be improved with some additional electronic devices between the sensor output and the ADC input to become the sensor output signal more suitable. It is a process called *signal conditioning* and can include different operations with the signal, such as signal amplification, filtering, coupling, or noise suppression.

For example, the system resolution shown in table I can be improved with the sensor output signal amplification, which is an example of signal conditioning. In this case, the Arduino ADC resolution is 5 mV that corresponds to 0.5 °C. To improve the system resolution to 0.2ºC, the sensor output voltage must be amplified 2.5 times (0.5°C/0.2°C) and it means that after the amplification stage, the sensitivity will be 25 mV/ºC and the ADC resolution of 5 mV will correspond to 0.2 °C.

Figure 29 shows an operational amplifier (OP) configured as a non-inverting amplifier circuit to perform the sensor signal (Vs) amplification.

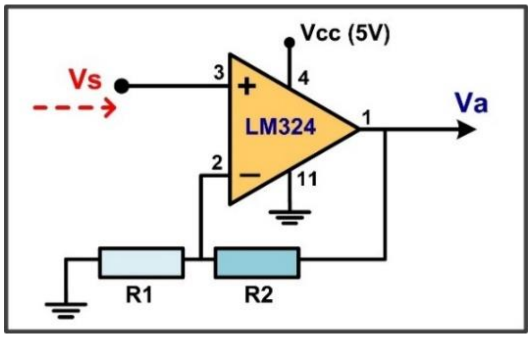


FIGURE 29. The signal amplifier schematic.

The relation between the signal from the sensor (*Vs*) and amplified output voltage (*Va*) is expressed as:

$$Va = \left(1 + \frac{R2}{R1}\right) Vs. \qquad (12)$$

For the amplification (gain) of 2.5 times, the relation between *Vs* and *Va* is *Va*=2.5Vs, and therefore:

$$2.5 = \left(1 + \frac{R2}{R1}\right) 1,$$

$$1.5 = \ R2 \,/\, R1.$$

The relation *R*2/*R*1 is 1.5. For example, for *R*1 fixed at 1 KΩ, so *R*2 would be 1.5 KΩ. In fact, *R*2 should be replaced by a precision potentiometer (trimpot) of 2 KΩ to allow a calibration, where the user must use a fixed *Vs* and turn the trimpot until *Va* reaches the value of 2.5*Vs*.

The amplification process is also susceptible to noise influence, and it can become more critical as the signal amplitude decreases. Therefore, a special amplifier circuit called *instrumentation amplifier* is recommended for low voltage signals or applications that require a high level of accuracy. Currently, there are commercial modules designed with specific instrumentation amplifier chips, as the INA129 and the INA333, but the figure 29 circuit can work well for simple demonstrations.

Another example of signal conditioning is an RC dumper circuit associated with the LM35 sensor, which can improve the sensor output signal stability [15]. Figure 30 shows this circuit composed by only a resistor and a capacitor. It is a simple circuit that becomes interesting for educational exemplifications. However, it is important to emphasize in the classroom that signal conditioning can require complex and expensive circuits in many cases.

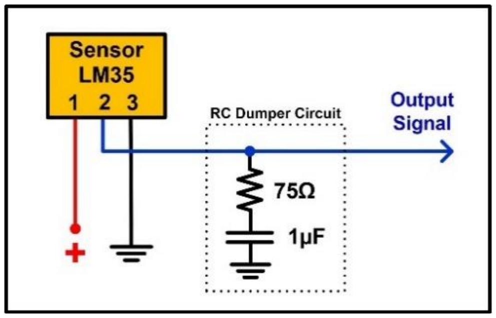


FIGURE 30. The LM35 with the RC damper circuit.

Figure 31 shows the temperature measurement with the same LM35DZ piece, but using different signal treatment techniques when the ambient temperature was 27.5 °C.

The dotted black curve in the figure 31 shows the measurement connecting directly the sensor and the Arduino with no improvement technique. The red curve shows the temperatures defined as the arithmetic mean of ten consecutive measurements, which is a software technique to improve the result quality and whose Arduino program is shown in figure 32. The blue curve is the measurement with the RC dumper circuit, which is a hardware technique for measurement improvement. These results prove the signal conditioning by hardware and the signal processing by software are techniques with large potential to improve significantly the measurement result quality.

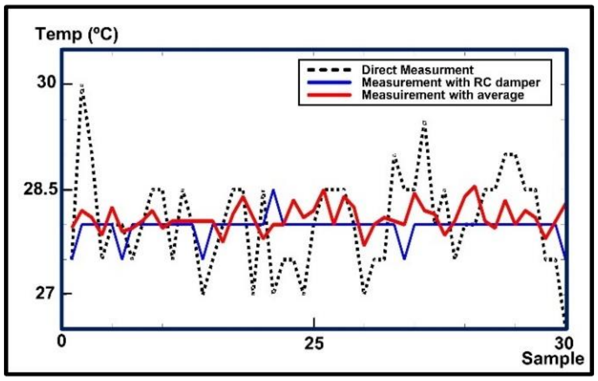
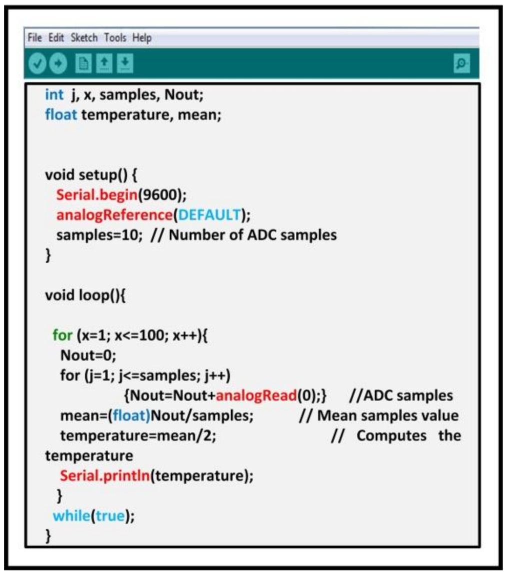


FIGURE 31. Measurements with hardware and software techniques for results improvement.



```
int  j, x, samples, Nout;
float temperature, mean;


void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);
  samples=10;  // Number of ADC samples
}

void loop(){

 for (x=1; x<=100; x++){
  Nout=0;
  for (j=1; j<=samples; j++)
             {Nout=Nout+analogRead(0);}     //ADC samples
  mean=(float)Nout/samples;         // Mean samples value
  temperature=mean/2;                   //  Computes  the
temperature
  Serial.println(temperature);
  }
 while(true);
}
```

FIGURE 32. Program that measures several temperatures consecutively and presents the arithmetic mean as the result.

XII. CONCLUSIONS

This article presented selected analog-to-digital conversion concepts and proved that they can be taught through simple experiments with an Arduino board, without previous students' knowledge. Despite simplicity, the presented experiments have quantitative results that allow objective analysis in the classroom.

REFERENCES

[1] Le, B. *et al*. *Analog-to-digital converters*, IEEE Signal Processing Magazine **22**, 69-77 (2005).
[2] Manganaro, G., *Emerging data converter architectures and techniques*, 2018 IEEE Custom Integrated Circuits Conference (CICC), San Diego, CA, 1-8 (2018).
[3] Jonsson, B. E., *A survey of A/D-Converter performance evolution*, 17th IEEE International Conference on Electronics, Circuits and Systems, Athens, 766-769 (2010).
[4] Fraden, J., *Handbook of Modern Sensors Physics, Designs, and Applications*, Fourth Edition Springer, New York (2010).
[5] JGCM (Joint Committee for Guides in Metrology), *2012 International Vocabulary of Metrology – Basic and General Concepts and Associated Terms (VIM)*, International Bureau of Weights and Measures Ed., Sevres, France (2012).
[6] De Bièvre, P., *The 2012 International Vocabulary of Metrology: VIM*, Accreditation and Quality Assurance **17**, 231-232 (2012).
[7] Zook, J. D., Schroeder, N., *Sensors as Information Transducers*, in Encyclopedia of Sensors, edited by Grimes, C. A. *et al*. **9**, 329-359, Stevenson Ranch CA (2005).
[8] Mari, L. A., *Quest for the Definition of Measurement*, Measurement **46**, 2889-2895 (2013).
[9] Mills, I., Marquardt, R., *The New SI: The International System of Units is Getting a Makeover*, Chemistry International **41**, 32-35 (2019)
[10] Foster, M. P., *The Next 50 Years of the SI: A Review of the opportunities for the E-Science Age*, Metrologia **47**, R41 (2010).
[11] Eshach, H., Kukliansky, I., *University Physics and Engineering Students' Use of Intuitive Rules, Experience, and Experimental Errors and Uncertainties*, International Journal of Science and Mathematics Education **16**, 817–834 (2018).
[12] Suter, G. W., Cormier, S. M., *Pragmatism: A Practical Philosophy for Environmental Scientists*, Integrated Environmental Assessment and Management **9**, 181-184 (2013).
[13] Derenzo, S. E., *Practical Interfacing in the Laboratory: Using a PC for Instrumentation, Data Analysis and Control*, 1st Edition, (Cambridge University Press, New York, 2003).
[14] Gray, Nicholas, *ABCs of ADCs Analog-to-Digital Converter Basics*, (National Semiconductor Corporation Ed., USA, 2006).
[15] Texas Instruments, *LM35 Precision Centigrade Temperature Sensors Datasheet* (Rev. H), (Literature number SNIS159H, USA, 2017).
[16] Perea Martins, J. E. M. *Introducing the concepts of measurement accuracy and precision in the classroom*, Physics Education **54**, 055029 (2019).
[17] Menditto A., Patriarca M., Magnusson B., *Understanding the Meaning of Accuracy, Trueness and Precision*, Accred and Quality Assurance **12**, 45-47 (2007).