No. 36 | 2026
ISSN Online 2007-9621
http://doi.org/10.15174/au.2026.4298
Acta Universitaria
Multidisciplinary Scientific Journal

Educational implementation of lookup tables for special functions on microcontroller PIC

Implementación educativa de tablas de consulta para funciones especiales en microcontroladores PIC

Mario A. Sandoval-Hernandez[1], Hector Vazquez-Leal[2]*, Hugo Jimenez-Islas[3], Roberto S. Murphy-Arteaga[4], Gerardo C. Velez-Lopez[4], Uriel A. Filobello-Nino[2], Griselda J. Morales-Alarcon[5], Victor M. Jimenez-Fernandez[2]

[1] Centro de Bachillerato Tecnológico Industrial y de servicios No. 190, Boca del Río, Veracruz, México, CP. 94297.
ORCID: 0000-0002-5518-3858
[2] Facultad de Instrumentación Electrónica, Universidad Veracruzana, Xalapa, Veracruz, México, CP. 91000. Tel. 2288422700 ext. 14801 ext. 14801,
ORCID: 0000-0002-7785-5272, hvazquez@uv.mx
ORCID: 0000-0002-3543-834X
ORCID: 0000-0003-1811-1238
[3] Ingeniería Bioquímica, Tecnológico Nacional de México en Celaya, Celaya, Guanajuato, México, CP. 38010.
ORCID: 0000-0002-1084-5520
[4] Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonanzintla, Puebla, México, CP. 72840.
ORCID: 0000-0001-9664-9207
ORCID: 0000-0002-0726-1153
[5] Instituto de Psicología y Educación, Universidad Veracruzana, Xalapa, Veracruz, México, CP. 91010,.
ORCID: 0000-0003-0142-0923
*Autor de correspondencia

Abstract

Keywords: tables; functions; microcontrollers; education; systems.

The main objective of this article is to detail the teaching of the implementation of lookup tables that address specific mathematical functions -such as the error function, among others-, using economical hardware resources. At the same time, a didactic application of these constructs is presented. This implies that the development of this application is centered around PIC and LCD microcontrollers, adapted to the pedagogical and economic needs of students in technical high schools and engineering. This set of pragmatic educational tools facilitates understanding of these mathematical functions and their implementation. Additionally, the use of low-cost components allows students from economically disadvantaged regions, particularly those from low-income countries, to participate in the implementation of this microcontrolled system.

Resumen

Palabras clave: Tablas; funciones; microcontrolador; educación; sistemas.

El objetivo de este artículo es detallar la enseñanza de la implementación de tablas de búsqueda que atienden funciones matemáticas específicas -como la función de error, entre otras-, aprovechando recursos económicos de *hardware*. Al mismo tiempo, se presenta una aplicación didáctica de estos constructos. Esto implica que el desarrollo de esta aplicación esté centrado en microcontroladores PIC y LCD, adaptado a las necesidades pedagógicas y económicas de estudiantes de bachillerato tecnológico y de ingeniería. Este conjunto de herramientas educativas pragmáticas facilita la comprensión de estas funciones matemáticas y su implementación. Además, la utilización de componentes de bajo costo permite a los estudiantes de regiones económicamente desfavorecidas, particularmente aquellos de países de bajos ingresos, participar en la implementación de este sistema microcontrolador.

Recibido: 02 de julio de 2024

Aceptado: 22 de julio de 2025

Publicado: 18 de febrero de 2026

Cómo citar: Sandoval-Hernandez, M. A.; Vazquez-Leal, H.; Jimenez-Islas, H.; Murphy-Arteaga, R. S.; Velez-Lopez, G. C.; Filobello-Nino, U. A.; Morales-Alarcon, G. J.; & Jimenez-Fernandez, V. M. (2026). Educational implementation of lookup tables for special functions on microcontroller PIC. *Acta Universitaria, 36*, e4298. doi. https://doi.org/10.15174/au.2026.4298

Introduction

Humanity has been using data tables for approximately 4500 years. In ancient Sumer, for example, multiplication tables were used to facilitate calculations; moreover, wages, livestock, agricultural production, and other matters were recorded in "cases" or boxes with numerical signs and ideograms, although without strict classification (Campbell-Kelly *et al.*, 2003). It was not until the 19th century BC, with some two hundred tabular lists in the temple of Ninurta in Nippur, that they began to be used continuously.

The modern row-column format appeared in the tables of logarithms of Napier and Briggs (1614, 1617) and was refined by G. Prony from 1792 when he was commissioned by Napoleon (Peaucelle, 2012; Vazquez-Leal *et al.*, 2020). In 1871, the British Association's Committee on Mathematical Tables was established (with figures such as Cayley, Stokes, and Thomson), and its first official volume was published in 1931, with printing attributed to L. J. Comrie and with a precision standard for future editions (Campbell-Kelly *et al.*, 2003).

In 1906, the association for tabulating Bessel functions was reactivated, and in 1941, tables of probability functions were published, which were also useful for calculating the error function (Campbell-Kelly *et al.*, 2003; National Bureau of Standards, 1953, 1954). Today, there is an extensive bibliography (Abramowitz & Stegun, 1960; Korn & Korn, 1968; Oldham *et al.*, 2009; Spiegel, 1988) that is being used in secondary education for transcendental functions (Caballero-Caballero & Bernández, 1999).

Lookup tables (LUT) are applied in numerous areas, from astronomical anniversaries (Campbell-Kelly *et al.*, 2003) to agricultural models (Nguyen *et al.*, 2021), image classification (Aiyer & Gray, 1999), round-robin algorithms (Dong & He, 2020), railway multibody simulations (Escalona & Aceituno, 2019), fast division (Hung *et al.*, 1999), and random number generation (Mitchell & Stone, 1977).

The present didactic work has two objectives. The first is to provide students with a fundamental understanding of the implementation of lookup tables (LUT), a topic that often lacks detailed explanation in introductory courses on numerical analysis and microcontroller programming. The second is to create an application program on a PIC microcontroller-based system for special functions. The system developed in this article features a rudimentary graphical user interface (GUI) on a graphic liquid crystal display (GLCD) using a PIC18F4620 microcontroller. This consideration is crucial in developing countries, where students may not have the resources to purchase expensive development systems for studying and practicing microcontrollers, allowing them to acquire the essential components. Likewise, this article presents a didactic sequence based on Gagne's theory (Barraza-Macías, 2020) for the implementation of LUT in the classroom.

Literature review

The theoretical background and literature review presented here cover the framework of the mathematical discourse of speech (MDS), the special functions implemented in this work, and some basics of lookup tables and fixed-point arithmetic. Additionally, the literature review includes a description of the PIC18F4620 microcontroller.

Mathematical discourse

When a text, story, article or book is written, the authors seek to impose opinions or behaviors in their practical discourses, using information and arguments to persuade the reader as well as to make known the concepts they wish to express (Elejalde, 1998). In general, speeches are utilitarian and manipulative in nature, with a structure based on the logic of demonstration and the rhetoric of argumentation. Literary discourses, however, can also be academic if they apply to the real world. The academic discourse is within the framework of academic activities oriented towards the transmission and production of knowledge in schools. This requires a review of previous knowledge (Arceo *et al.*, 2010) and the consideration of bibliographical references. Thus, the mathematical discourse of speech (MDS) is the language used in the classroom, where teachers and students share their ideas and propose solutions to problems (Cantoral *et al.*, 2015; Soto *et al.*, 2012; Soto & Cantoral, 2014).

Implementation of Special Functions

Special functions arise from the solution of differential equations of various physics problems, such as optical, acoustic, and thermal phenomena (Korn & Korn, 1968; Lakshminarayanan & Varadharajan, 2015; Oldham *et al.*, 2009). These functions are often expressed in terms of integrals that cannot be solved by conventional analytical methods (Leithold, 2014; Stewart, 2018). Additionally, these functions exhibit different behaviors with respect to their properties and evaluations (Arfken & Weber, 1999). The mathematical approximations for special functions presented in this section were published for educational purposes in Sandoval-Hernandez *et al.* (2019, 2021), intended for high school or undergraduate students.

Elliptic integrals of the first and second kind

Elliptic integrals have been studied for more than three and a half centuries and have been widely used in various fields of science and technology. They have applications in electrodynamics (Sandoval-Hernandez *et al.*, 2021) and the nonlinear pendulum (Segovia-Chaves & Dussán-Penagos, 2016), among others. The complete elliptic integral of the first kind $K(k)$ can be defined as (equation 1):

$$K(k) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}}, \quad -1 < x < 1 \tag{1}$$

while the complete elliptic integral of the second kind $E(k)$ is given by equation 2:

$$E(k) = \int_0^{\frac{\pi}{2}} \sqrt{1-k^2\sin^2\theta}, \quad -1 \leq x \leq 1 \tag{2}$$

Figure 1 shows a graph of the complete elliptic integrals $K(k)$ and $E(k)$, respectively, both of which are even.

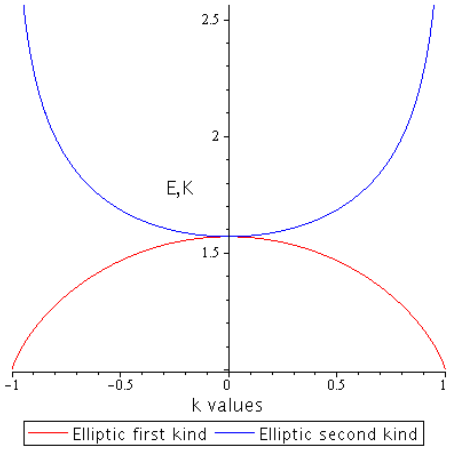


Figure 1. Complete elliptic integral of $K(k)$ and $E(k)$.
Source: Authors' own elaboration.

The Fresnel integrals

Fresnel integrals are two special functions, the sine S(t) and cosine C(t), expressed as integrals. Both functions have many important applications in various fields of physics, mathematics, and engineering (*e.g.*, optics, electromagnetism, antenna design, and signal processing). For example, in optics, the Fresnel functions are applied in the Fresnel diffraction solution for a rectangular aperture (Abedin & Rahman, 2012; Sandoval-Hernandez *et al.*, 2018). Figure 2 shows the Fresnel integrals, which are odd.

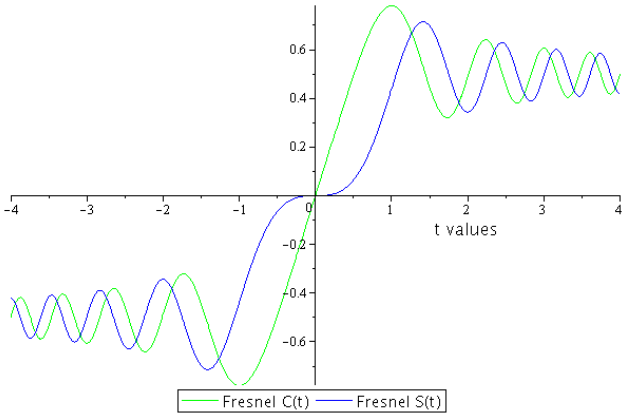


Figure 2. Graph of Fresnel integrals of $C(x)$ and $S(x)$.
Source: Authors' own elaboration.

The Euler spiral, also known as Cornu spiral or Clothoid, is the curve generated by a parametric plot of $S(t)$ against $C(t)$ (Sandoval-Hernandez *et al.*, 2018). The Fresnel sine and cosine integrals $S(t)$ and $C(t)$ are given by equation 3 and 4:

$$S(t) = \int_0^t \sin\left(\frac{\pi x^2}{2}\right)dx, \quad -\infty \leq x \leq \infty \tag{3}$$

$$C(t) = \int_0^t \cos\left(\frac{\pi x^2}{2}\right)dx, \quad -\infty \leq x \leq \infty \tag{4}$$

Error function integral and cumulative distribution function

The error function, has various applications in fields such as Newtonian fluid analysis (Bird, 2002), analog signal processing (Papoulis, 1977), and noise analysis in communications systems, including protocols such as phase-shift keying (PSK) modulation (Proakis, 2001), among others. The error function is given by equation 5:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt, \quad -\infty \leq x \leq \infty \tag{5}$$

It is commonly used in statistics, probability theory, and numerical analysis to represent the cumulative distribution function of the standard normal distribution. It also has applications in several fields such as signal processing, image processing, and control systems.

The cumulative distribution function (CDF) is also a Gaussian integral and is considered one of the most widely applied functions in various disciplines of science and engineering within statistical analysis (Devore, 2011; Eisberg & Resnick, 1978). This function allows modeling various phenomena such as those found in biology (Lange, 2003; Wayne, 1987), social sciences (Berg, 2004; Preacher *et al.*, 2006), psychology (Howitt & Cramer, 2005; Wilkinson, 1999), and finance (Anderson *et al.*, 2014; McClave, 2012), among others. The cumulative distribution function is given by equation 6:

$$\operatorname{CDF}(x) = \frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{x} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) dt, \quad -\infty \leq x \leq \infty \tag{6}$$

Figure 3 shows error function and cumulative distribution function.

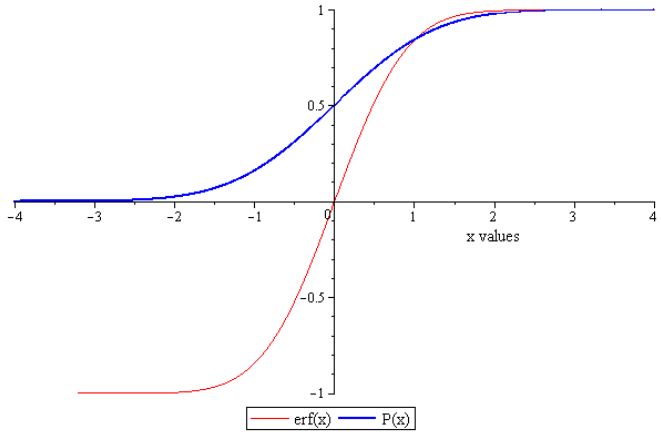


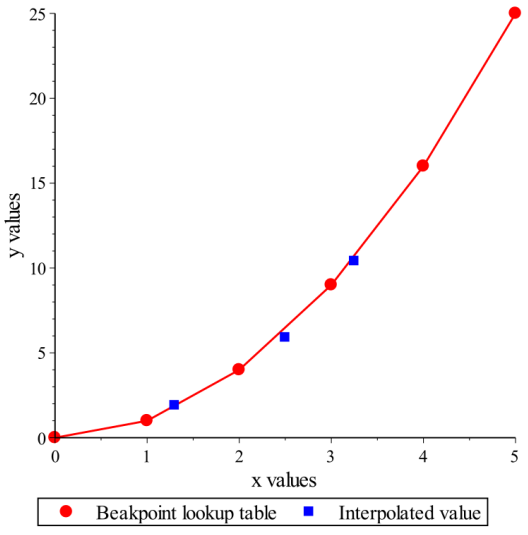
Figure 3. Graph of error and cumulative distribution, erf(x) and CDF(x).
Source: Authors' own elaboration.

Lookup tables

With the advent of electronic devices and computer equipment, the use of lookup tables (LUT) has become a popular method for improving processing time and efficiency. A LUT is an array that replaces run-time computation with a simpler array indexing operation. This can greatly reduce processing time, as it involves retrieving a value from read-only memory (ROM), which is typically faster than performing a calculation. LUT are precomputed and then stored in the ROM of a device, or they can be calculated as part of a program's initialization phase (Kwok *et al.*, 1995).

The process of employing LUT is conceptually simple: a predefined array of input indices facilitates constant-time retrieval of the associated output values stored in the LUT. When an exact input is absent, interpolation -often linear or higher-order- is applied to approximate the result efficiently. This strategy enables rapid approximation of mathematical functions, offering significant advantages, especially when the analytical form entails intensive computation or when it is unavailable (Mitchell & Stone, 1977). LUT are extensively utilized across fields such as signal processing, image processing, control systems, and some others based on numerical computations.

Figure 4 shows an example of a one-dimensional LUT that approximates the quadratic function $y = x^2$ with a domain of [0,5]. The LUT defines its output data $y(x)$ discretely over the range of [0,25]. For example, for a breakpoint of 3, the LUT retrieves the corresponding value of 9. If $x = 2.5$, however, there is no value in the LUT for the corresponding value of y.



x	0	1	1.3	2	2.5	3	3.3	4	5
y	0	1	1.9	4	5.5	9	11.1	16	25

Figure 4. Table and graph that illustrate the breakpoints, input/output relationship.
Source: Authors' own elaboration.

In this case, a linear or quadratic interpolation process must be carried out (Burden *et al.*, 2015; Chapra & Canale, 2016; Teukolsky *et al.*, 1992). The equation for linear interpolation is given by equation 7.

$$y = y_0\left(\frac{x_1 - x}{x_1 - x_0}\right) + y_1\left(\frac{x - x_0}{x_1 - x_0}\right) \tag{7}$$

where $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$ are known. Quadratic interpolation can be calculated by equation 8:

$$y = y_0 + (x - x_0)\left(\frac{y_1 - y_0}{x1 - x_0}\right) + (x - x_0)(x - x_1)\left(\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}\right)/(x_2 - x_0) \quad (8)$$

Note that in equation 8 the interpolation is carried out with known $P_0(x_0, y_0), P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. For example, in reference to Figure 4, linear interpolation was used to determine y for $x = 2.5$ with $P_1(2,4)$ and $P_2(3,9)$. Using equation 7 we get $y = 5.5$.

There are two fundamental limitations in building LUT. The first one is the available memory of the program. The second one is the time required to calculate the values. It is important to mention, however, that there are established query-table design algorithms. For example, in Tang (1991) an algorithm is presented to design LUT that can be implemented in hardware. The design method consists of three steps.

The first one is to establish a set of breakpoints. In the second step, the approximation is calculated using a polynomial within the interval of interest. With the aid of Remez algorithm, the coefficients of the polynomial approximation can be determined (Pachón & Trefethen, 2008; Tasissa, 2019). The final step is to reconstruct the function by means of some specific relations that depend on the functions to be implemented, then the values are tabulated[1].

Fixed-point arithmetic

Fixed-point arithmetic is a numerical representation technique to perform mathematical operations on numbers where the position of the radix point (*i.e.*, the decimal point) is fixed. Instead of storing numbers as floating-point values, fixed-point numbers are stored as integers, and the radix point is assigned to a specific location by the programmer. This allows for more efficient computation, as operations are performed on integers rather than floating-point values. However, it also requires the programmer to carefully manage the radix point during the arithmetic operations to ensure the correct results are obtained (Oberstar, 2007; Pyeatt & Ughetta, 2019; Yates, 2009). In fixed-point operations, the following criteria must be considered:

1. Whether the number is signed or unsigned.
2. The position of the radix point in relation to the sign bit for signed numbers and the most significant bit for unsigned numbers.
3. The number of bits allocated for the fractional part of the number.

The Q notation is used to represent fixed point numbers as Qm, n with m bits for the integer part and n bits for the fractional part. The total number of bits is $N = m + n + 1$ for signed numbers; for example, if $N = 16$ bit number, the $Q2,13$ format indicates it has 2 bits for the integer part, 13 bits for the fractional part, and 1 bit for the sign. Similarly, in Figure 5, an example of an $N = 8$ bit number with $Q3,4$ format is shown; it has 3 bits for the integer part, 4 bits for the fractional part, and 1 bit for the sign. To obtain the value of N-bit number in $Qm.n$ format, it can be calculated by equation 9 (Sandoval-Hernandez *et al.*, 2023):

[1] A more detailed explanation can be found in Tang (1991).

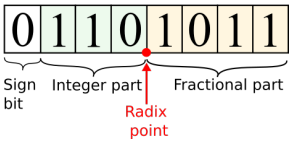


Figure 5. Fixed-point representation.
Source: Authors' own elaboration.

$$\text{Value} = -b_{N-1}2^{m} + \sum_{i=0}^{N-2} b_i 2^{i-n} \tag{9}$$

For example, expanding equation 9 and substituting values for N, m, n with $b_7 = 0$, $b_6 = 1$, $b_5 = 1$, $b_4 = 0$, $b_3 = 1$, $b2 = 0$, $b_1 = 1$, and $b_0 = 1$, we get equation 10:

$$\text{value} = 4b_6 + 2b_5 + 0.5b_3 + 0.125b_1 + 0.0625b_0 \tag{10}$$

The decimal value obtained is 6.875.

When performing arithmetic operations, it is important to consider the integer and fractional parts of the numbers involved; i_1 and f_i act as the integer and fractional parts of the first number and i_2 and f_2 the integer part and fractional parts of the second one. In the case of addition and subtraction, the numbers must be aligned with the same number of bits in the integer and fractional parts.

In multiplication, the result will have a total number of bits equal to the sum of the number of bits in the two numbers being multiplied, one with n bits and another with m bits, and the integer and fractional parts of the result will be the sum of the integer and fractional parts of the original numbers, *i.e.*, $i_1 + i_2$ and $f_1 + f_2$, respectively.

In division, the quotient can be reduced to several cases, depending on whether the numbers being divided are signed or unsigned. Care must be taken to avoid negative fractional parts, which can be achieved by shifting the dividend to the left if the divisor has more fractional bits[2]. In the case of division, the results can be reduced to following cases:

1. The quotient of dividing two unsigned fixed-point numbers is given by a number with an integer part $i_1 + f_2$ and a fractional part $f_1 - f_2$.
2. The quotient of dividing a signed number and an unsigned number results in a number with an integer part $i_1 + f_2$ and a fractional part $f_1 - f_2$.
3. The quotient of dividing two signed numbers gives a number with an integer part $i_1 + f_2 + 1$ and a fractional part $f_1 - f_2$.

[2] For more detailed information, please refer to Oberstar (2007), Pyeatt & Ughetta (2019), and Yates (2009).

It is important to note that care must be taken when $f_2 > f_1$, as this leads to a negative fractional part. To avoid this, the dividend can be shifted to the left, ensuring that at least as many fractional bits are available as the divisor. This leads to the following rule: if $f_2 > f_1$, then the divisor should be converted to i_1, x, where $x \leq f_2$ (Oberstar, 2007; Pyeatt & Ughetta, 2019; Yates, 2009). To convert a floating-point number to a fixed-point representation, we follow these steps:

- Multiply the floating-point number by 2^n, where n is the number of desired fractional bits.
- Round the number to the nearest whole number.
- If the number is negative, take the two's complement of the value obtained in step 2. Store the rounded value in an integer container.

In practice, fixed-point arithmetic is often implemented in programming languages such as C, C++, Python, and others, typically using macros. Examples of such macros for basic operations on fixed-point numbers can be found in the literature, such as in ARM Developer (1996) and Schlösser (2013). For example, in ARM Developer (1996), the following macros for basic operations on two fixed-point numbers a and b of q format, returning the answer in q format, are presented:

```
/* This code was published in Schlösser (2013) The basic arithmetic operations*/

#define FADD(a,b) ((a)+(b))

#define FSUB(a,b) ((a)-(b))

#define FMUL(a,b,q) (((a)*(b)) >>(q))

#define FMUL(a,b,q) (((a)<<(q))/(b))
```

where >> is a right shift and << a left shift (Ceballos, 2019).

Also, there are open source code for C++ for fixed-point such as fixedptc and libfixedmath (Schlösser, 2013). There is also open source code for C++ for fixed-point arithmetic, such as fixedptc and libfixedmath. This notation has been around for a long time (Schlösser, 2013).

The microcontroller PIC18F4620

PIC microcontrollers, or peripheral interface controllers, have gained widespread acceptance and development in recent years due to their exceptional characteristics such as low cost, low power consumption, reliability, and readily available technical support. These programmable integrated circuits possess the necessary architecture to perform a variety of tasks. PIC can be classified according to their capacity into 8-bit, 16-bit, and 32-bit categories. These, in turn, are divided into different families. In the case of 8-bit PIC microcontrollers, the family is divided into the following ranges: Low range (PIC10, some PIC12, and PIC16), Mid range (some PIC12, and PIC16), and High range (PIC18).

The PIC18F microcontroller family delivers robust performance while maintaining cost-efficiency, featuring support for C-language development and a wide range of standard communication interfaces, including CAN and USB. These devices offer between 8 KB and 128 KB of flash program memory and can operate at frequencies up to 64 MHz, making them suitable for a diverse array of embedded applications (Microchip Technology Inc., 2008).

The PIC18F4620 microcontroller, part of the PIC18 family, was developed to enhance performance in energy-sensitive applications using NanoWatt technology (Microchip Technology Inc., 2008). It is available in both standard ("F") and low-voltage ("LF") versions, operating across voltage ranges of 4.2 V to 5.5 V and 2.0 V to 5.5 V, respectively. Notable features include 10 oscillator options: four crystal modes compatible with crystals or ceramic resonators, an internal oscillator block offering up to eight user-selectable clock frequencies from 125 kHz to 4 MHz, and a phase lock loop (PLL) that enables clock speeds up to 40 MHz, with combinations that can yield frequencies from 31 kHz to 32 MHz (Microchip Technology Inc., 2008).

Other characteristics of the PIC18F4620 are summarized in Table 1. It can be noted that the PIC18F4620 can reach a speed of 40 MHz and is available in different packages such as Dual In-line Package (DIP) 40. Ports A, B, and C have 8 bits, and ports D and E have 4 bits (Microchip Technology Inc., 2008). In Figure 6 we can see the pins of the microcontroller as well as their different functions, such as the channels of the digital analog converter. For specific details please refer to Microchip Technology Inc. (2008).

Table 1. Some features of Pic18F4620.

Features	Pic18F4620
Operating frequency	DC-40MHz
Program Memory (Bytes)	65536
Program Memory (Instructions)	32768
Data Memory (Bytes)	3968
Data EEPROM Memory (Bytes)	1024
Interrupt Sources	20
I/O Ports	A, B, C, D, E
Timers	4
-Bit Analog-to-Digital Module	13 Input Channels
Instruction Set	75 Instructions 83 with Extended Instruction Set Enabled
Packages	40-Pin PDIP 44-Pin QFN 44-Pin TQFP

Source: Authors' own elaboration.

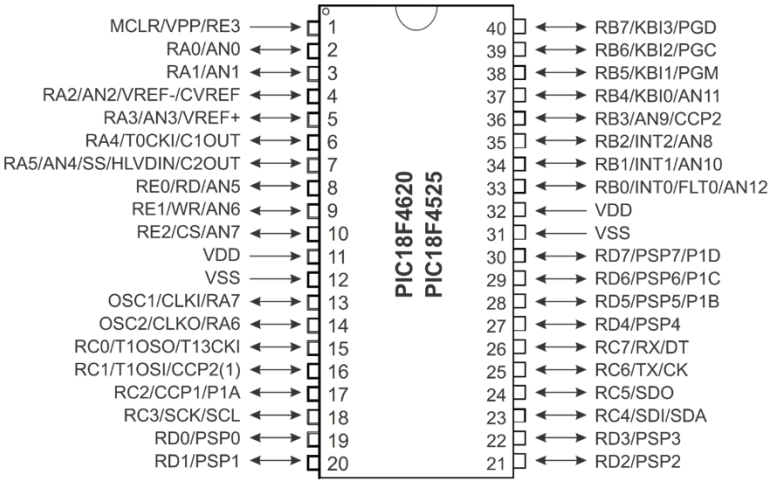


Figure 6. Pin diagram for PIC18F4620.
Source: Taken from Microchip Technology Inc. (2008).

Materials and methodology

Development and description of the implementation

In this work, a computer with dual boot and the operating systems Windows 7 and Ubuntu 18.04.05 was used, equipped with an NVIDIA GeForce GTX 1050 Ti and an Intel i7-7700 @ 3600GHz x 4 processor. The programs used for the design of the lookup tables (LUT) were Maple 2017 and GCC 7.5.0 (GNU Compiler Collection) for Linux.

For the implementation of the LUT on the system, the MikroC 7.6.0, Visual GLCD 2.7.1.0 for GUI design and MikroE EasyPIC V7 Connectivity Development Board from MikroElectronika were used. Additionally, the PIC18F4620 Microcontroller was selected for the project.

LUT implementation

To begin the implementation, the design of the lookup table breakpoints was created using the seq command in Maple 2017. The seq command is used to construct a sequence of values within a function command. This allows for the avoidance of implementing an approximate polynomial.

For example, in the implementation of the LUT for the error function, 104 breakpoints were used (for this case study), which correspond to the number of elements. In this case, the erf function of Maple 15 (command erf) was evaluated in the interval [0.3.1] with a step size of $3x10^{-2}$, with the help of the seq command. For example, the evaluation is given by:

```
/*Use seq command to implement LUT*/

CDF_erf := [evalf(seq(subs(x = i, erf(x)), i = 0. .. 3.1, 0.03))];
```

where seq and erf are the built-in functions.

The values obtained in the LUT have floating-point values with 10 significant digits. To convert a floating point number to a fixed-point representation, we will multiply by 2^n because this vector will be implemented in MikroC, and this language considers a float or double variable with 4 bytes (MikroElektronika, 2009). For the function erf(x) we use $n = 24$.

Table 2 indicates the number of breakpoints used, steps, and evaluation intervals in the design of LUT for the functions that are implemented. The factor 2^n for conversion to fixed-point is also specified. In the case of elliptic functions, two LUT have been implemented for each one in order to make the evaluation in the upper interval more accurate.

Table 2. Implementation of LUT for special functions.

Function (x)	Break-Points	Interval of evaluation	Step	Factor 2^n
Erf	104	[0,3.1]	0.03	2^{24}
CDF	127	[0,3.8]	0.03	2^{24}
FresnelC	131	[0,5.2]	0.04	2^{20}
FresnelS	131	[0,5.2]	0.04	2^{20}
EllipticE	92, 92	[0,0.91], [0.89,1.00]	0.01, 0.01	2^{20}
EllitpticK	111, 97	[0,0.94], [0.9180,0.9999]	0.0085, 0.0085	2^{20}

Source: Authors' own elaboration.

To obtain the fixed-point representation of the LUT for the Fresnel functions, each vector was multiplied by 2^n with $n = 20$. For the floating-point conversion for the LUT erf(x) and CDF(x), $n = 24$ was used.

The LUT for each of the functions has been obtained in fixed-point representation using Maple. The implementation resides in the microcontroller's flash memory. To determine values that are not given by the breakpoints, quadratic interpolation, as described by equation 8, was used.

For the EllipticK(x) function, two LUT have been implemented in the intervals [0,0.91] and [0.89,1.00] in order to get more significant digits near the upper limit, as shown in Figure 1. The asymptotic behavior of this function requires the implementation of more breakpoints, 117 for the first interval and 97 in the second one. Similarly, the elliptic function EllipticE(x) also has two LUT in its implementation in order to make the evaluation at the upper bound more accurate.

Algorithm 1 presents the pseudocode for implementing each one of the procedures programmed for each special function in Table 2. Line 1 has the procedure header, which has the argument $xsigno$, it receives a numerical value, positive or negative. Line 2 defines the variables that will be used in the procedure, which are of type double, and the variable index is of type integer. Line 4 checks the sign of the numeric value. If the value is negative, then the procedure assigns the negative sign to the value it returns (see line 23). This is implemented for odd functions such as the Fresnel, Elliptical, and erf. For the implementation of the CDF, replace 1 - value in line 23. Lines 12 to 18 perform quadratic interpolation.

Algorithm 1: Pseudocode for special functions

```
1:  procedure FSpecialxsigno  /* Function */
2:    Define variables
3:    double mifunction[104] = {0, 567761, ...}
4:    x = abs(xsigno)
5:    Xscale = round(2^n * x)  /* n = 20 or 24 */
6:    for (index = 0; step; upper limit) do
7:      if (Xscale <= index * 2^n) then break
8:      end if
9:      Ac = Ac + 1
10:   end for
11:   /* Quadratic interpolation */
12:   x0 = index
13:   x1 = index + step
14:   x2 = index + 2 * step
15:   y0 = (1 / 2^n) * mifunction[Ac]
16:   y1 = (1 / 2^n) * mifunction[Ac + 1]
17:   y2 = (1 / 2^n) * mifunction[Ac + 2]
18:   value = ...  /* Equation in text (8) */
19:
20:   if (xsigno > 0) then
21:     return value
22:   else
23:     return -(value)
24:   end if
25: end procedure
```

Library implementation

The packaging of all the functions was carried out with MikroC's package manager. The procedure is as follows:

1. Create a C file containing all the special functions. In this work, the file is named Special_Functions.c. Additionally, create an h file containing only the function headers; in this work, the file is named Special_Functions.h. The C file should include the line #include "Special_Functions.h" to access the function headers.
2. In the MikroC Package Manager, choose the option *create new package*. Assign the compiler and a name for the package. Select the microcontroller family, in this case, PIC18. In the dependencies, enable the MikroC libraries to be used, such as *C Math* and *CStdlib*. In the Main Library file section, open the C file, in this case, Special_Functions.c. To protect the contents of the library, choose the MikroC Compiled mcl file instead of the C file, in this case, Special_Functions.mcl. In the Other Library Files section, add the h file, in this case, Special_Functions.h.
3. In the MikroC Package Manager, go to the "File" menu and choose "Save As" to save the package as an mpkg file, in this case, Special_Functions.mpkg.

4. In menu File of Package Manager choose *Install Package*. In the MikroC Library Manager, choose Rebuilt Libraries. The desired library will be placed in the Library Manager Third Party Functions, in our case, Special_Functions.

Description of the application

The graphical user interface (GUI) for this study was developed using Visual GLCD (MikroElektronika, 2012). Two separate GUI were created: one for inputting and returning data, and the other one for selecting the desired function.

The programming of this project was implemented using the event-driven paradigm, in which the structure and execution of routines are determined by objects with properties and methods, as well as user-defined events that can occur within the system. The routines for calculating specific functions were programmed into each of the command buttons. Each command button has the property of *Press color* enabled and responds to the *Onclick* event. Figure 7 illustrates the GUI where the programmed functions are located.

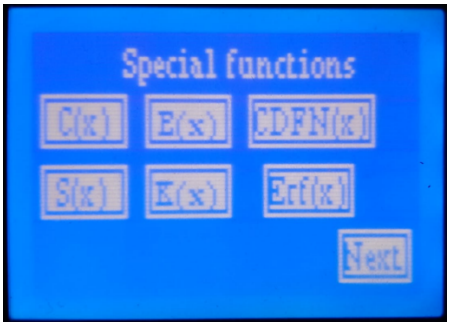


Figure 7. Graphical user interface 2 for special functions menu.
Source: Authors' own elaboration.

Figure 8 shows the GUI featuring command buttons labeled with the names of the implemented functions. Figure 8 presents the flowchart of the program. The calibration touch symbol represents the calibration of the touch screen through the digital-analog converter of the PIC18F4620 microcontroller and the configuration of the GLCD 127x64 pixels on the EasyPicV7 Card. Additionally, the PLL is configured with the internal clock to operate at a frequency of 32 MHz. GUI 1 receives input data and stores it in a text box. GUI 2 displays six command buttons, each containing its corresponding routine. The result is then returned to the text box in GUI 1.

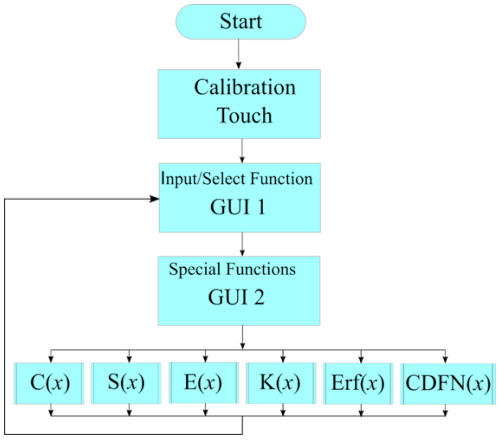


Figure 8. Flowchart of program.
Source: Authors' own elaboration.

Results and discussion

The MDS is often criticized for lacking context and frames of reference that would make learning more meaningful. This is a common issue with textbooks from other areas of knowledge used in undergraduate education as well. Many authors assume that readers have a solid background and possess the necessary prerequisites to understand the text, but they fail to provide adequate pedagogical support. This lack of attention to pedagogy makes autonomous learning difficult for university students. Furthermore, the academic discourse presented by various authors in their textbooks can make it challenging for students to understand and apply the material.

We define academic discourse of science and engineering (ADSE) as textbooks that present content in a rigid and formal manner, without providing examples that foster meaningful learning. An example of books that exhibit ADSE are mathematics textbooks that present only theorems and proofs without providing examples, or else providing simple examples that do not aid the understanding. Similarly, some computer science books present algorithms in a general manner without giving examples of their practical implementation. Additionally, this issue also arises in some research papers in the fields of physics and mathematics where the content is highly abstract and does not provide clear examples for readers to replicate. As a result, these texts become inaccessible to students who are new to certain areas of knowledge.

In this project, we overcome the ADSE and MDS by presenting details about the implementation of the special-function approximations and their LUT, and by discussing how this implementation is analyzed throughout the article. Many details are ignored because the authors consider the reader to be a well-educated professional who fully understands his/her work.

Figure 9 shows the GUI when the Click event has occurred on the command button for CDFN(x) using the touch screen on the EasyPic V7 connectivity card. The format used to show the results returned by each of the functions on the display is $Q1{,}6$. The command button with the property caption=Input activates the input of the data on the same screen. We have for $x = 0.960$ a value equal to 0.831473 (*i.e.*, $Q1{,}6$ format), showing five decimal digits. The command button with the property caption=Next activates the GUI 1 (Figure7) in which we choose one of the six command buttons with the respective implemented function. Choosing one of them returns the result again in GUI 1.

Figure 9. Graphical user interface 1 for cumulative distribution function.
Source: Authors' own elaboration.

Figure 10 shows the significant digits for CDFN(x) of accuracy that can be obtained by using the LUT described in Table 1. Note that at least five significant digits of accuracy are available. Near $x = 1$, more significant digits are obtained; however, when the value obtained is displayed on the GLCD, 6 significant digits are shown. In Figure 9 five significant digits of accuracy are shown; this value is in good agreement with the significant digits shown in Figure 10.

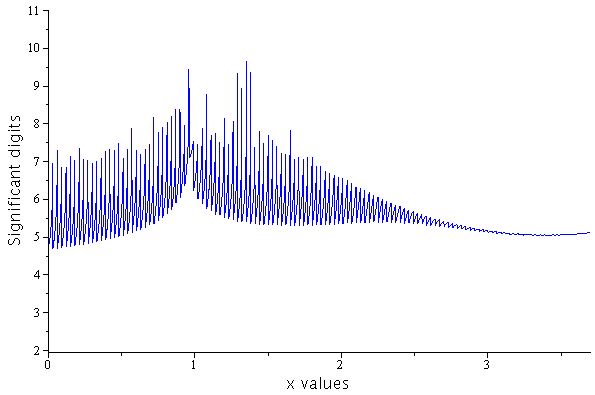


Figure 10. Significant digits for cumulative distribution function.
Source: Authors' own elaboration.

The significant digits were calculated (Barry, 1995a; Barry, 1995b) using equation 11:

$$\mathrm{SD} = -\log_{10}\left|\frac{f(x)-\tilde{f}(x)}{f(x)}\right| \qquad (11)$$

where

1. $f(x)$ is the exact function. In this case it is calculated from the built-in functions of Maple.
2. $\tilde{f}(x)$ is the approximate function for which we want to know the number of significant digits. In this case, it is determined from the LUT that have been programmed.

Figure 11 shows the significant digits for the lookup table (LUT) for the erf(x). For values close to 0, at least three significant digits are obtained. The accuracy improves significantly as we approach $x = 3$. Note that it is possible to obtain six significant digits within the interval shown; this is made possible using breakpoints. For certain values, it is possible to obtain at least four significant digits due to the quadratic interpolation used in the algorithm. The same is observed in the graph of significant digits for the cumulative distribution function in Figure 10.

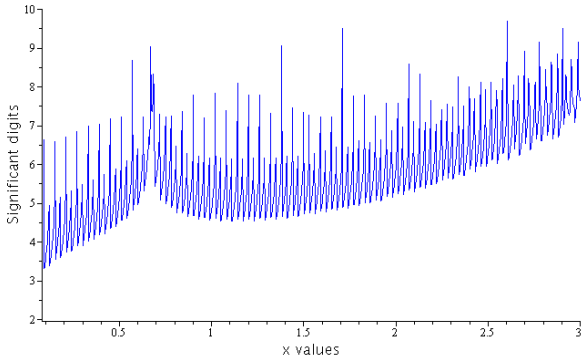


Figure 11. Significant digits for function error erf(x).
Source: Authors' own elaboration.

Figure 12 shows the evaluation of the complete elliptic K(x) integral of the first kind on the GLCD using LUT. In this case, we set $x = 0.500$ and obtain a value equal to 1.685750. It is important to note that six significant digits of accuracy have been shown and displayed on the GLCD.

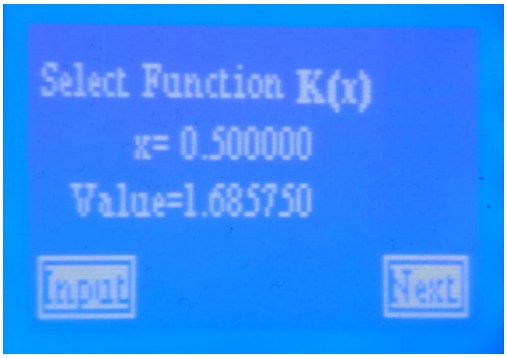


Figure 12. Graphical user interface 1 for the complete elliptic integral $K(x)$.
Source: Authors' own elaboration.

In Figure 13, it can be observed that for values of $x < 0.6$, at least five significant digits are obtained. However, for higher values, such as $x = 0.867$, six significant digits can be achieved, with a result of 2.159552. It is important to note that for values of $x > 0.90$, the number of significant digits increases due to the use of the second implemented LUT, as shown in Table 2. The second LUT allows for increased accuracy near $x = 1$, where the elliptic function starts to exhibit an asymptotic behavior.

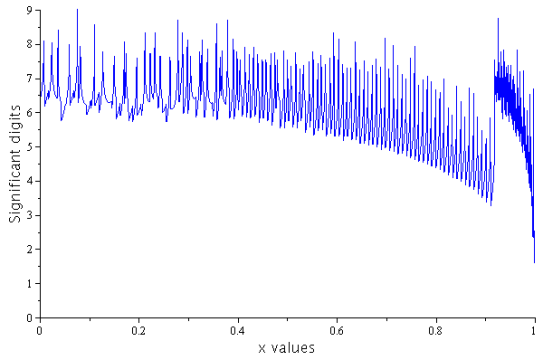


Figure 13. Significant digits for the complete elliptic integral $K(x)$.
Source: Authors' own elaboration.

Figure 14 shows the significant digits for the complete elliptic integral E(x) of the second kind. Note that there are at least five significant digits for $x < 0$. This is also true for $x > 0.89$; the number of significant digits increases due to the presence of the second implemented LUT. Around $x = 1$, four significant digits are displayed.

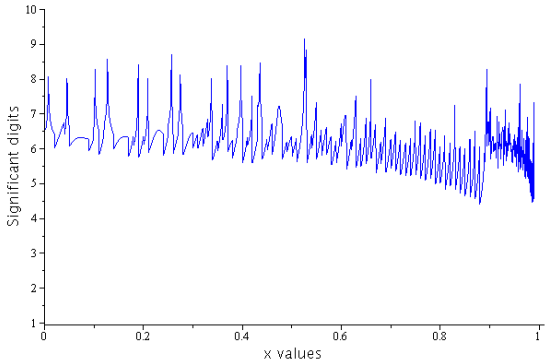


Figure 14. Significant digits for the complete elliptic integral $E(x)$.
Source: Authors' own elaboration.

Figure 15 shows the evaluation of the Fresnel integral sine function for $x = 1.395$, resulting in a value of 0.713150 with five significant digits of accuracy in the graphical user interface (GUI).

Figure 15. Graphical user interface 1 for Fresnel sine integral $S(x)$.
Source: Authors' own elaboration.

Figure 16 shows the significant digits for the implemented LUT for S(x). The significant digits in Figures 16 and 17 are less than in the previously shown figures because the period of the Fresnel functions decreases as x increases, resulting in a decrease in the number of breakpoints in each cycle. For the interval shown in the figures, at least two significant digits are obtained.

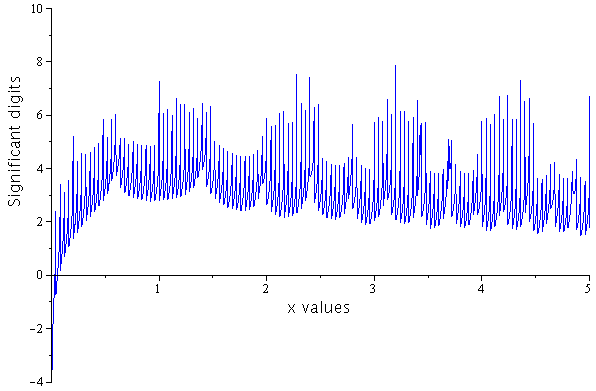


Figure 16. Significant digits for Fresnel sine function $S(x)$.
Source: Authors' own elaboration.

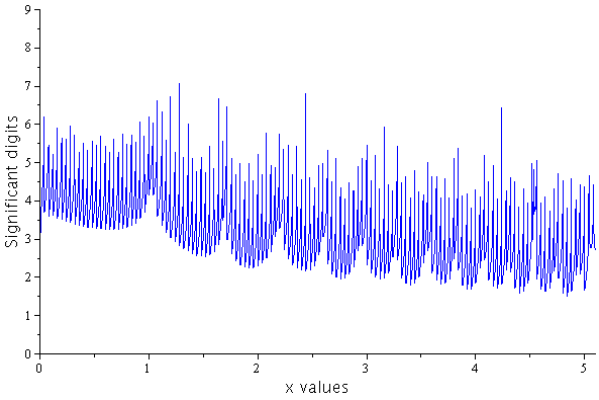


Figure 17. Significant digits for Fresnel cosine function $C(x)$.
Source: Authors' own elaboration.

It is imperative to emphasize that for the enhancement of precision in numerical representation, particularly concerning the error function (erf), cumulative distribution functions of the normal distribution (CDFN), and elliptical functions, a procedural algorithm can be devised to integrate a functionality within the graphical LCD (GLCD) apparatus. Such an algorithmic construct could be activated via a dedicated toggle mechanism integrated into the display interface, or alternatively, through the utilization of a supplementary input field designated for this specific purpose. Concurrently, the incorporation of secondary input modality, such as a keyboard interface, remains a viable option for the input of requisite data. The implementation of the routines is left as an exercise to the students.

Table 3 presents a set of specifications for the lookup table (LUT) based routines of the special functions that have been implemented using MikroC. For example, FE_Erf(x) is the notation used in the library for the error function. It also displays the evaluation interval for each function, as well as the size in Kb of each LUT stored in the flash memory of the PIC microcontroller. The elliptical functions have larger memory storage requirements due to the use of two LUT. Additionally, the execution times of each function on the microcontroller at a speed of 16 MHz are presented. The time measurement is based on arbitrary values of x, such as $x = 0.96$, $x = 0.96$, $x = 2.5$, $x = 2.5$ for the first four functions, respectively, and $x = 0.54$ and $x = 0.92$ for the elliptical functions. To obtain a reliable measurement of time, the measurements were determined by the average time of 100 000 iterations for each point. For all the cases shown, the time spent by the microcontroller was less than 8 ms.

Table 3. Features for routines implemented for special functions.

Function (x)	Evaluation for x	Size Kb	Execution time (ms)
FE_Erf	$\|x\| < 3.250$	5.96	2
FE_CDFN	$\|x\| < 3.720$	6.34	2
FE_FresnelC	$\|x\| \leq 5.120$	6.38	3.1
FE_FresnelS	$\|x\| \leq 5.120$	6.38	2.9
FE_EllipticE	$\|x\| < 0.9960$	7.91	2.60,1.20
FE_EllipticK	$\|x\| < 0.9978$	8.85	3.00,4.40

Source: Authors' own elaboration.

The measurement of time was carried out on the microcontroller by programming an interruption generated by timer 0 every 100 μs. Each time an interruption is generated, a count is performed. Finally, at the end of the cycle, we multiply the total count by 100 μs and divide it by 100 000 -which is the total number of evaluations- to obtain the average time spent by the microcontroller in executing the routine. To obtain the time after 100 μs, the registers TMR0H = 0xFE, TMR0L = 0x70, T0CON = 0x88 must be loaded, assuming a clock speed of 16 MHz. If the clock speed is different, the values in TMR0 and T0CON will be different. The routine used to measure the execution time on the microcontroller follows.

Algorithm 2: Pseudocode for time measurement

```
1:    /* Configurations, Variables, ...*/
2:     void InitTimer0()
3:     {
4:       //setting TMR0 for 100us,T0CON
5:      TMR0H=0xFE; TMR0L=0x70 ; T0CON=0x88;
6:      GIE_bit= 1;
7:      TMR0IE_bit= 1;
8:     }
9:
10:   /*   interruption      */
11:    void Interrupt()
12:    {
13:     if (TMR0IF_bit)
14:      {
15:       /* Clear flag of TMR0 Load TMR0 */
16:       tiempo=tiempo+1;
17:      }
18:    }
19:
20:    void main()
21:    {
22:     /* Initialize GLCD . . .
23:       . . .
24:      While(1)
25:       {
26:        Lcd_Out(1,1,"Times measured");
27:        X=2.5; /* Value for test */
28:        for(count=1;count<100000;count+1)
29:        {
30:         /* put your s. function to measure */
31:           zz=FE_FresnelS(x);  /* Example */
32:        }
33:
34:        GIE_bit = 0;
35:        Lcd_Cmd(_LCD_CLEAR);
36:       Sprintf(txt,"TMR,x=%f", time);
37:       Lcd_out(1,1,txt);
38:       Delay_ms(500);
        }
37:
38:    }
```

The computing time (CT) that an algorithm spends evaluating a function is an important aspect in numerical analysis and system design. This is also the case for the functions in Table 3, where measuring CT is crucial. The CT was measured in C using the clock_t command, with gcc 5.4.0 and optimization level 3 for the compiler. To avoid operating system instabilities, the time measurement was determined by the average time for 100 000 evaluations for each point. The results are represented in Figures 18 and 19. From these figures, it can be observed that the CT is less than 0.4 μs. As the evaluation of each point in all routines approaches the upper limit, the CT increases. This analysis allows us to know how the implemented routines behave. In other words, the CT dedicated to the moment of evaluating the routines increases since, as we approach the upper limit, the entire lengths of the LUT are evaluated. It is important to notice, however, the difference in times; the CT for the microcontroller is measured in milliseconds.

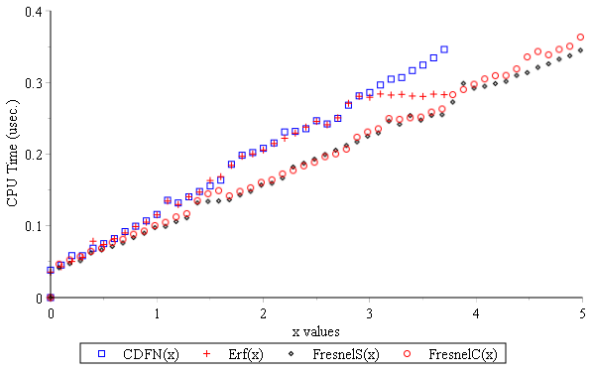


Figure 18. Computation time to evaluate routines for CDF, erf and Fresnel functions.
Source: Authors' own elaboration.

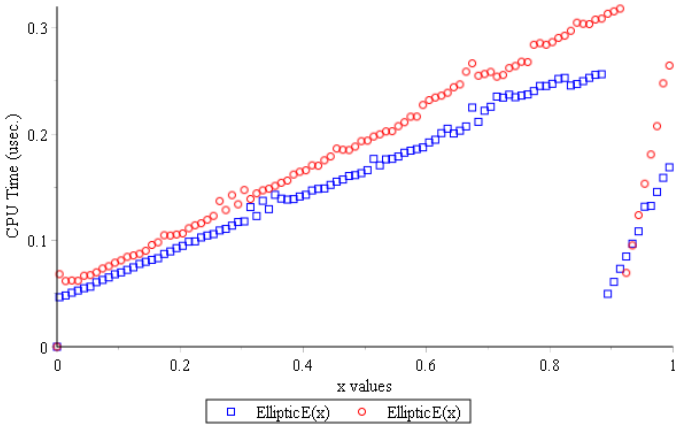


Figure 19. Computation time to evaluate routines for elliptic functions.
Source: Authors' own elaboration.

Didactic sequence

The purpose of this didactic sequence is to enable students to develop the skills required to generate, verify, and implement lookup tables (LUT) for special functions. To achieve this, a routine developed in Maple will be used to create LUT, with the normal cumulative function as a case study. Students will evaluate, plot, and validate the function within a defined interval and then generate a vector of sample points. They will then apply fixed-point scaling and rounding to produce the final LUT in fixed-point format. Finally, they will translate this data into a C program that can be deployed on the PIC18F4620 microcontroller.

Inside the microcontroller, this function will accept an input value for x, convert it to fixed-point format, and search for its position within the LUT. If the input lies between two stored values, linear interpolation will be performed to approximate the result. A separate function will be created for each LUT, depending on the number of special functions covered in the course. This practice helps students gain a complete understanding of the processing workflow, from mathematical analysis and numerical validation to embedded programming optimized for systems with limited resources. Table 4 below shows the teaching sequence for eight sessions of approximately 1 hour, each LUT will be stored as a fixed-point vector within a dedicated subroutine.

Table 4. Didactic sequence.

Session	Gagné Phase	Session Objective	Main Activities	Resources	Evidence
1	1) Gain Attention 2) Inform Objectives 3) Stimulate Recall	Contextualize the importance of LUTs for special functions and the use of Maple to generate them.	Motivational talk about LUTs and Maple. Guiding questions. Summary of the base article.	Printed article, PPT slides, whiteboard.	Participation in discussion and notes on prior ideas.
2	4) Present Content	Understand the structure and principles of look-up tables.	Analyze LUT examples for microcontrollers. Review Fresnel and Bessel tables from the article.	Article, diagrams, Proteus/MPLAB simulator.	Concept map showing LUT structure.
3	5) Provide Learning Guidance	Explore the Maple routine to generate special function tables.	Review each section of the Maple script. Identify graphing, scaling, and rounding steps.	Printed Maple code, computer with Maple.	Notes and step-by-step routine outline.
4	6) Elicit Performance/Practice	Run the Maple routine to generate a normal cumulative distribution table.	Execute the script. Verify graphs and accuracy. Analyze relative errors.	Computer with Maple, routine file, practice guide.	Screenshots of graphs and generated table.
5	6) Elicit Performance/Practice	Export the Maple table and convert it to fixed-point format for C language.	Export table data. Create the corresponding routine in C. Configure the MikroC project.	Maple, C editor, MikroC compiler, PIC18F4620 datasheet.	Exported table file and C code fragment.
6	7) Provide Feedback	Compare results from the Maple routine and the C implementation in MikroC.	Simulate table queries in Proteus/MPLAB. Discuss differences and accuracy optimizations.	Proteus, MikroC, verification rubric.	Simulation log and verification report.
7	7) Provide Feedback	Review and optimize full integration of the LUT in the PIC microcontroller.	Peer review session. Instructor-guided corrections. Final functional test.	Computers, simulator, evaluation rubric.	Checklist confirming functionality.
8	8) Assess Performance and Enhance Retention	Reflect on practical applications of LUTs generated with Maple and their transfer to embedded systems.	Group presentation of results. Discussion on advantages and improvements. Closing reflection.	Projector, evaluation forms, group feedback.	Presentation rubric and written reflection.

Source: Authors' own elaboration.

Limitations and future work

The implementation of LUT requires the use of enough FLASH memory to store them. It remains to optimize the implementation of the query tables. The microcontroller has an 8-bit data bus which limits the number of significant digits in each evaluation. The MikroC compiler treats float, double, and long double data types as bytes of the same size, 4. Consequently, some memory locations are not used to represent the data. Also, LUT must be implemented on microcontrollers with at least 16 Kb of memory to store the intended task program.

Future work encompasses the use of functions that approximate certain intervals of the domain of the special functions using polynomials generated with the Remez or Chebyshev algorithms. The evaluations that could be done with these polynomials must be implemented with fixed-point arithmetic. It is also possible to implement the Airy, Dawson, Gamma and Lambert W functions as well as some probability distributions such as Poisson and t-student, among others using LUT, considering that they can be performed within a limited range when implemented on an 8-bit PIC microcontroller.

Conclusions

The goal of this project was to contribute to the improvement of the academic discourse presented in books and articles, particularly in the field of engineering. Many authors tend to omit pedagogical details in their writing, making the learning process difficult for those who do not have a solid academic background.

In this paper, we have expounded on the integration of lookup tables (LUT) in an instructional manner, facilitating the assimilation of their application within electronics laboratories for students.

Furthermore, the present exposition encompasses the realization of a didactic framework wherein a graphical liquid crystal display (GLCD) touchscreen interfaces with a PIC microcontroller, effectuating the computation of distinct specialized functions through the utilization of LUT. The resulting execution times, not exceeding the duration of 8 ms, mark a notable achievement. The derivation and construction of the LUT were meticulously undertaken using the Maple computational tool, yielding precision reaching magnitudes as low as 6 significant figures.

The libraries that have been implemented could also be programmed on 16 –bit microcontrollers, such as dsPIC (digital signal controllers), which have the same architecture but with additional blocks for digital signal processing.

In conclusion, the developed libraries could have multiple applications within the field of science and engineering and have been designed to be implemented within a simple programming environment.

Acknowledgements

Authors would like to thank Roberto Ruiz Gomez for his contribution to this article.

Conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

Abedin, K. M., & Rahman, S. M. M. (2012). The iterative Fresnel integrals method for Fresnel diffraction from tilted rectangular apertures: theory and simulations. *Optics & Laser Technology*, *44*(4), 939–947. https://doi.org/10.1016/j.optlastec.2011.11.004

Abramowitz, M., & Stegun, L. A. (1960). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Bureau of National.

Aiyer, A., & Gray, R. M. (1999). A fast, table-lookup algorithm for classifying document images. In *Proceedings of the 1999 International Conference on Image Processing* (Vol. 1) (pp. 590-594). IEEE. https://doi.org/10.1109/ICIP.1999.821699

Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., & Cochran, J. J. (2014). *Statistics for business & economics* (12th ed). Cengage Learning.

Arceo, F. D. B., Hernández, G. R., & González, E. L. G. (2010). *Estrategias docentes para un aprendizaje significativo: una interpretación constructivista*. McGraw-Hill Interamericana.

Arfken, G. B., & Weber, H. J. (1999). *Mathematical methods for physicists* (4th ed). Academic Press.

ARM Developer. (1996). *Application Note 33: Fixed point arithmetic on the ARM (ARM DAI 0033A)*. Advanced RISC Machines Ltd. https://developer.arm.com/documentation/dai0033/a/

Barraza-Macías, A., Valles-Terrones, A., Piñón-Torres, G. J., Soto-Aguilar, P. M., Segovia, V. M., Bustillos-García, S., Heredia-Corral, L. Y., Torrecillas-Herrera, N., Uribe-Salazar, G., García-Vázquez, I. J., Castañeda-García, A., Martínez-Arcineaga, H., Vallejo, J. L., Ortiz-Martínez, E., Reyes-Marín, M., Ortega, S., & Valenzuela-Parra, S. J. (2020). *Modelos de secuencias didácticas* (1st ed.). Universidad Pedagógica de Durango.

Barry, D. A., Barry, S. J., & Culligan-Hensley, P. J. (1995a). Algorithm 743: WAPR–A Fortran routine for calculating real values of the W-function. *ACM Transactions on Mathematical Software (TOMS)*, *21*(2), 172–181. https://dl.acm.org/doi/10.1145/203082.203088

Barry, D. A., Culligan-Hensley, P. J., & Barry, S. J. (1995b). Real values of the W-function. *ACM Transactions on Mathematical Software (TOMS)*, *21*(2), 161–171. https://dl.acm.org/doi/10.1145/203082.203084

Berg, B. L. (2004). *Methods for the social sciences* (5th ed). Pearson Education.

Bird, R. B. (2002). *Transport phenomena*. John Wiley & Sons.

Burden, R. L., Faires, J. D., & Burden, A. M. (2015). *Numerical analysis*. Cengage Learning.

Caballero-Caballero, A., & Bernández, J. (1999). *Tablas matemáticas*. Esfinge.

Campbell-Kelly, M., Croarken, M., Flood, R., & Robson, E. (2003). *The history of mathematical tables: from Sumerian to spreadsheet*. Oxford University Press.

Cantoral, R., Montiel, G., & Reyes-Gasperini, D. (2015). Análisis del discurso matemático escolar en los libros de texto, una mirada desde la teoría socioepistemológica. *Avances de Investigación en Educación Matemática*, (8), 9–28. https://doi.org/10.35763/aiem.v1i8.123

Ceballos, F. J. (2019). *Curso de Programación C/C++* (5th ed.). AlfaOmegaRAMA.

Chapra, S. C., & Canale, R. P. (2016). *Métodos numéricos para ingenieros* (7th ed.). McGraw-Hill.

Devore, J. L. (2011). *Probability and statistics for engineering and the sciences* (7th ed.). Cengage Learning.

Dong, X., & He, Y. (2020). CRC algorithm for embedded system based on table lookup method. *Microprocessors and Microsystems*, *74*, 103049. https://doi.org/10.1016/j.micpro.2020.103049

Eisberg, R. M., & Resnick, R. (1978). *Física cuántica*. Limusa.

Elejalde, A. (1998). *Discurso literario y discurso académico*. Apuntes Cultura Hispánica.

Escalona, J. L., & Aceituno, J. F. (2019). Multibody simulation of railway vehicles with contact lookup tables. *International Journal of Mechanical Sciences*, *155*, 571–582. https://doi.org/10.1016/j.ijmecsci.2018.01.020

Howitt, D., & Cramer, D. (2005). *Introduction to statistics in psychology*. Prentice Hall.

Hung, P., Fahmy, H., Mencer, O., & Flynn, M. J. (1999). Fast division algorithm with a small lookup table. In *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers* (Vol. 2) (pp. 1465–1468). IEEE. https://doi.org/10.1109/ACSSC.1999.831992

Korn, G. A., & Korn, T. M. (1968). *Mathematical handbook for scientists and engineers*. McGraw-Hill.

Kwok, W., Haghighi, K., & Kang, E. (1995). An efficient data structure for the advancing-front triangular mesh generation technique. *Communications in Numerical Methods in Engineering*, *11*(5), 465–473. https://doi.org/10.1002/cnm.1640110511

Lakshminarayanan, V., & Varadharajan, L. S. (2015). *Special functions for optical science and engineering*. SPIE. https://doi.org/10.1117/3.2207310

Lange, K. (2003). *Mathematical and statistical methods for genetic analysis*. Springer Science & Business Media.

Leithold, L. (2014). *El cálculo EC7*. Oxford University Press.

McClave, J. T. (2012). *Statistics for business and economics: student value edition*. Prentice Hall.

Microchip Technology Inc. (2008). *Microchip PIC18F2525/2620/4525/4620 Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology*. Microchip Technology Inc. https://ww1.microchip.com/downloads/en/devicedoc/39626e.pdf

MikroElektronika. (2009). *MikroC PRO for PIC User Manual. MikroE Software and Hardware Solutions for Embedded World*. MikroElektronika. https://download.mikroe.com/documents/compilers/mikroc/pic/mikroc-pic-manual-v101.pdf

MikroElektronika. (2012). *Visual GLCD: GLCD GUI design software. MikroE Software and Hardware Solutions for Embedded World*. MikroElektronika. https://www.mikroe.com/visual-glcd

Mitchell, R. L., & Stone, C. R. (1977). Table-lookup methods for generating arbitrary random numbers. *IEEE Transactions on Computers*, *26*(10), 1006–1008. https://doi.org/10.1109/TC.1977.1674735

National Bureau of Standards. (1953). *Tables of normal probability functions*. U.S. Government Printing Office.

National Bureau of Standards. (1954). *Tables of the error function and its derivative* (Vol. 41). U.S. Government Printing Office.

Nguyen, N. Y., Kha, D. D., & Ichikawa, Y. (2021). Developing a multivariable lookup table function for estimating flood damages of rice crop in Vietnam using a secondary research approach. *International Journal of Disaster Risk Reduction*, *58*, 102208. https://doi.org/10.1016/j.ijdrr.2021.102208

Oberstar, E. L. (2007). *Fixed-point representation & fractional math*. https://doi.org/10.13140/RG.2.1.3602.8242

Oldham, K. B., Myland, J., & Spanier, J. (2009). *An atlas of functions: with equator, the atlas function calculator*. Springer. https://doi.org/10.1007/978-0-387-48807-3

Pachón, R., & Trefethen, N. (2008). Barycentric-Remez algorithms for best polynomial approximation in the chebfun system. *BIT Numerical Mathematics*, *49*, 721-741. https://doi.org/10.1007/s10543-009-0240-1

Papoulis, A. (1977). *Signal analysis*. McGraw-Hill.

Peaucelle, J.-L. (2012). A white elephant during the French Revolution: Prony's logarithm tables. *EKSA*, *107*(1), 74-86.

Preacher, K. J., Curran, P. J., & Bauer, D. J. (2006). Computational tools for probing interactions in multiple linear regression, multilevel modeling, and latent curve analysis. *Journal of Educational and Behavioral Statistics*, *31*(4), 437-448. https://doi.org/10.3102/10769986031004437

Proakis, J. G. (2001). *Digital communications*. McGraw-Hill.

Pyeatt, L., & Ughetta, W. (2019). *ARM 64-Bit assembly language*. Newnes.

Sandoval-Hernández, M. A., Hernández-Méndez, S., Torreblanca-Bouchan, S. E., & Díaz-Arango, G. U. (2021). Actualización de contenidos en el campo disciplinar de matemáticas del componente propedéutico del bachillerato tecnológico: el caso de las funciones especiales. *Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, *12*(23), e278. https://doi.org/10.23913/ride.v12i23.1044

Sandoval-Hernández, M. A., Vazquez-Leal, H., Filobello-Nino, U., & Hernandez-Martinez, L. (2019). New handy and accurate approximation for the Gaussian integrals with applications to science and engineering. *Open Mathematics*, *17*(1), 1774–1793. https://doi.org/10.1515/math-2019-0131

Sandoval-Hernandez, M., Vazquez-Leal, H., Hernandez-Martinez, L., Filobello-Nino, U. A., Jimenez-Fernandez, V. M., Herrera-May, A. L., Castaneda-Sheissa, R., Ambrosio-Lazaro, R. C., & Diaz-Arango, G. (2018). Approximation of Fresnel integrals with applications to diffraction problems. *Mathematical Problems in Engineering*, *2018*, (1), 1-13. https://doi.org/10.1155/2018/4031793

Sandoval-Hernández, M. A., Velez-López, G. C., Vázquez-Leal, H., Filobello-Nino, U. A., Morales-Alarcón, G. J., De-Leo-Baquero, E., Bielma-Pérez, A. C., Sampieri-González, C. E., Pérez-Jácome-Friscione, J. E., Contreras-Hernández, A. D., Álvarez-Gasca, O., Sanchez-Orea, J., & Cuellar-Hernández, L. (2023). Basic implementation of fixed-point arithmetic in numerical analysis. *International Journal of Engineering Research & Technology*, *12*(01), 313–318. https://www.ijert.org/basic-implementation-of-fixed-point-arithmetic-in-numerical-analysis

Schlösser, O. (2013). *Implementing a C++ Fixed-Point class for embedded systems*. Fachhochschule NordwestschweizFHNW, IME. http://hdl.handle.net/11654/17723

Segovia-Chaves, F. A., & Dussán-Penagos, A. (2016). Un estudio de la dinámica del péndulo no lineal. *Revista Científica*, *24*(1), 63–72. https://doi.org/10.14483/udistrital.jour.RC.2016.24.a6

Soto, D., & Cantoral, R. (2014). Discurso matemático escolar y exclusión. Una visión socioepistemológica. *Bolema: Boletim de Educação Matemática*, *28*(50), 1525–1544. https://doi.org/10.1590/1980-4415v28n50a25

Soto, D., Gómez, K., Silva, H., & Cordero, F. (2012). Exclusión, cotidiano e identidad: una problemática fundamental del aprendizaje de la matemática. *Comité Latinoamericano de Matemática Educativa*, *25*(1), 1041-1048. https://funes.uniandes.edu.co/wp-content/uploads/tainacan-items/32454/1211081/SotoExclusionALME2012.pdf

Spiegel, M. R. (1988). *Manual de fórmulas y tablas matemáticas: 2400 fórmulas y 60 tablas* (Schaum's Outline Series). McGraw-Hill.

Stewart, J. (2018). *Single variable calculus: Concepts and contexts*. Cengage Learning.

Tang, P. T. P. (1991). Table lookup algorithms for elementary functions and their error analysis. In *Proceedings of the 10th IEEE Symposium on Computer Arithmetic* (pp. 232–236). IEEE. https://doi.org/10.1109/ARITH.1991.145565

Tasissa, A. (2019). Function approximation and the Remez algorithm. https://sites.tufts.edu/atasissa/files/2019/09/remez.pdf

Teukolsky, S. A., Flannery, B. P., Press, W. H., & Vetterling, W. T. (1992). *Numerical recipes in C: the art of scientific computing* (2nd ed.). Cambridge University Press.

Vazquez-Leal, H., Sandoval-Hernandez, M. A., & Filobello-Nino, U. (2020). The novel family of transcendental Leal-functions with applications to science and engineering. *Heliyon*, *6*(11), e05418. https://doi.org/10.1016/j.heliyon.2020.e05418

Wayne, W. D. (1987). *Biostatistics: a foundation for analysis in the health sciences*. John Wiley & Sons.

Wilkinson, L. (1999). Statistical methods in psychology journals: guidelines and explanations. *American Psychologist*, *54*(8), 594–604. https://psycnet.apa.org/doi/10.1037/0003-066X.54.8.594

Yates, R. (2009). *Fixed-point arithmetic: an introduction*. Digital Signal Labs, technical Reference. http://www.digitalsignallabs.com/fp.pdf