SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V12I7P113

Volume 12 Issue 7, 184-199, July 2025

Original Article

Simulation Applied to Phase Detection in a Coriolis Effect Based Flowmeter

Luis Fernando Gutierrez Belizario[1], Javier Pablo Montesinos Quispe[1], Carlos Enrique Villanueva Portal[1], German Alberto Echaiz Espinoza[2*], Daniel Domingo Yanyachi Aco Cardenas[2], Carmelo Mayta Ojeda[3], Fernando Enrique Echaiz Espinoza[4]

[1]*Professional School of Electronic Engineering, Universidad Nacional de San Agustin, Arequipa, Peru.*
[2]*Academic Department of Electronic Engineering, Universidad Nacional de San Agustin, Arequipa, Peru.*
[3]*Academic Department of Physics, Universidad Nacional de San Agustin, Arequipa, Peru*
[4]*Institute of Mathematics, Federal University of Alagoas, Alagoas, Brazil.*

[*]*Corresponding Author : gechaiz@unsa.edu.pe*

Received: 09 May 2025 Revised: 11 June 2025 Accepted: 10 July 2025 Published: 31 July 2025

Abstract - *This paper presents a proposal for the electronic simulation of a Coriolis sensor based on the resonance frequency of a straight tube made of stainless steel, with emphasis on the design of the simulation for the measurement of the angular phase shift of the signals generated by the MPUs in the sensor. The system is based on an ESP32 microcontroller and two MPU-6050 sensors, whose data are displayed on a 4-digit, 7-segment display (TM1637). The code used in the simulation employs a C algorithm that estimates the phase difference between the two signals; a digital PLL was implemented to obtain the phase of each signal, and a DFT was used for the precise measurement of the phase difference, which is directly proportional to the mass flow. Disturbances due to flow pulsations and their impact on system accuracy are also considered, incorporating Gaussian noise in the simulation. Validation of the algorithm was performed by comparing the results simulated in Wokwi with those obtained in the Octave software. The errors obtained were less than ±0.2°, which supports the feasibility of the algorithm for measuring phase shift in a low-cost Coriolis sensor, applicable in industries such as chemical processing and fluid transport monitoring.*

Keywords - *Coriolis sensor, Signal phase shift, Resonant frequency, Gaussian noise.*

1. Introduction

1.1. Statement of the Problem and Justification of the Research

This work proposes a replicable model and a system with reasonably priced components (<US$200), thus addressing two important shortcomings: the lack of detailed protocols for phase detection in straight-tube Coriolis sensors and the reliance on specialized hardware in current solutions. Particularly valuable in sectors such as chemical processing, fluid dosing, and quality control systems, Coriolis-based flow meters constitute a basic technology for the direct and accurate measurement of mass flow rate. The devices are quite reliable under different conditions because, unlike other techniques, they determine the fluid's mass without relying on its thermal or pressure characteristics. However, a clear research gap exists surrounding the electronic implementation of straight-tube Coriolis sensors using accessible technologies. Technical difficulties in accurately identifying angular displacement among signals produced by tube-mounted inertial sensors define the electronic implementation of Coriolis sensors, particularly those with straight tubes [1]. The problems are compounded by noise, flow pulsations, and mechanical disturbances, which compromise the system's accuracy. These challenges, underexplored in low-cost, open-source solutions, limit the technology's real-world applicability in contexts outside of specialized laboratories or large industries. Currently, the technical literature is limited in terms of studies that address these aspects with sufficient depth and replicability, since most current studies address the problem superficially, which restricts their practical relevance and leaves an academic gap in terms of comprehensive and repeatable methods. This article presents the simulation and validation of a stainless-steel embedded system for a straight-tube Coriolis sensor. The proposal is based on common, low-cost components, such as an ESP32 microcontroller, two MPU-6050 inertial sensors, and a seven-segment display with driver (TM1637), all readily available on the market. Through this intentional selection of low-cost hardware, Coriolis technology is democratized and adopted in industrial, academic, and resource-constrained SMEs, where commercial solutions with specialized hardware are not always feasible. The main method combines a phase-locked loop (digital PLL)

for real-time tracking and a Discrete Fourier transform (DFT) for angular displacement accuracy, thus estimating the displacement between the two signals. The impact of flow pulsations on measurement accuracy is investigated, and Gaussian noise is included in the simulation to assess the system's robustness to disturbances. By achieving an error of less than ±0.2° and comparing the simulation results obtained in Wokwi with those of GNU Octave, the validation supports the feasibility of the proposal.This contributes to bridging a significant gap between academic research and real-world industrial needs by offering a replicable, accessible, and scientifically validated solution. This lays the groundwork for future advances in demanding environments where component availability and cost-effectiveness are paramount, combining sophisticated digital signal processing, rigorous simulation, and integrated electronics at reasonable prices.

1.2. Fundamental Concept of Coriolis Flowmeters

Currently, flow meters are more common and are distinguished by their high accuracy.

A Coriolis flow meter can come in different shapes and sizes, but is primarily composed of one or two tubes vibrating at their natural frequency. Its operating principle is the Coriolis force, FC, which arises when a mass, m, has a velocity, v, in a rotating reference frame. The rotating reference frame has an angular velocity. The relationship is described by: [1]

$$F_C = -2m\omega \times v \quad (1)$$

In a Coriolis flowmeter, the moving mass is the fluid, and the rotating reference system is the moving tube or tubes. Figure 1 shows two cases in which a straight tube vibrates: in the upper case, when it moves upward, and in the lower case, when it moves downward. Ma C. explains that the Coriolis force arises when the fluid collides with the walls of the tube, which is slightly bent due to vibration. In the upper part of Figure 1, the fluid on the left side collides with the bottom wall of the tube, creating the downward Coriolis force. On the right side, the opposite occurs, creating the upward Coriolis force. [1].

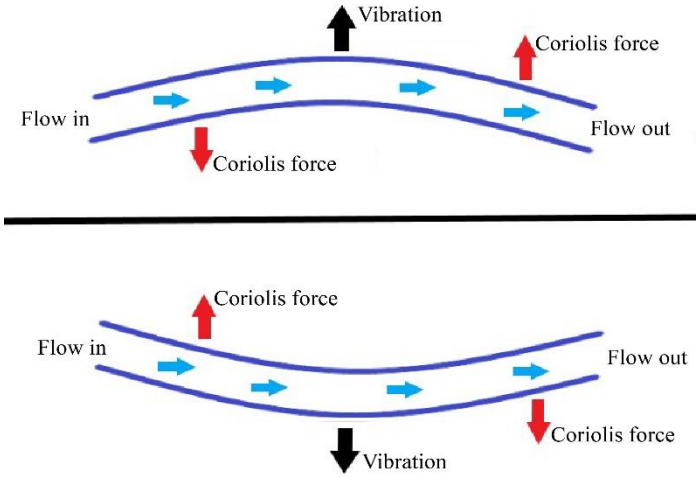

Fig. 1 Illustration of coriolis forces acting on a straight tube put into vibration; the force arises when there is a flow through the tube. The image depicts the tube at extremes of vibration [1].

2. State of the Art

2.1. Validation of the Use of Coriolis Mass Flow Meters as Reference Standards in Custody Transfer Applications

2.1.1. Objective

To metrologically confirm the Coriolis-type mass flow meters used as reference standards in the calibration of on-site custody transfer flow meters, evaluate their performance through comparisons between measurements made with the National Standard for Liquid Flow and a bidirectional tester under real operating conditions.

2.1.2. Methodology

Initial calibration: Coriolis Mass Flow meters (CMF) were calibrated with the National Standard for Liquid Flow by the gravimetric method using water as the working fluid.

On-site validation: Subsequently, the meters were validated in the field using a bidirectional tester with petroleum products as the working fluid.

Techniques employed:

- Calibration using reference standards.
- Comparisons between gravimetric and volumetric methods.
- Evaluation of correction factors (density, temperature, pressure).
- Calculation of error, Meter Factor (MF) and K-factor.

2.1.3. Results and Methodology

It was concluded that the laboratory and on-site calibration results are equivalent within the specified

uncertainty. The systematic errors between the gravimetric and volumetric methods were acceptably low. Validation was obtained for the use of Coriolis-type meters as reference standards for custody transfer applications in the petroleum industry.

2.1.4. Limitations

No leakage between the tester and the meter under calibration was considered. Equal temperatures in the fluid and the bodies of the gauges and testers were assumed, which may not be perfectly met in the field.

Thin-walled cylinder theory was applied, which introduces an approximation that may affect accuracy at extreme pressure conditions. No rapid dynamic variations of flow rate or transient operating conditions were addressed; only stationary conditions.

2.2. A New Phase Difference Measurement Method for the Coriolis Mass Flowmeter based on Correlation Theory [2]

2.2.1. Objective

To develop a new method for measuring phase difference in Coriolis-type mass flow meters, based on correlation theory, to improve accuracy, reduce bias in signals of unknown frequency and solve the problem of non-integer period sampling signals.

2.2.2. Methodology

Frequency estimation: An Adaptive Notch Filter (ANF) is used to estimate the frequency of the signals and filter out noise. Data extension: The length of the sample data is adjusted to match an integer number of periods, avoiding edge effects in the Hilbert Transform. The Hilbert Transform is applied to extended signals to generate analytical signals. Calculation of cross-correlation functions between analytical and real signals. Determination of the phase difference using a formula based on trigonometric functions on the results of correlations.

2.2.3. Results and Methodology

The proposed method shows better accuracy than traditional methods, such as:

- Standard correlation,
- Direct Hilbert transform,
- Sliding Goertzel Algorithm (SGA),
- Discrete Time Fourier Transform (DTFT).
- Maintained high accuracy even with low SNR (Signal-to-Noise Ratio) signals and non-integer period samples.
- Better dynamic performance with lower computational complexity compared to DTFT and SGA.
- Real experiments:
- Applied to a RHEONIK Coriolis meter (RHE08 sensor).
- The phase difference estimated with the proposed method was closer to the theoretical value than the SGA and DTFT methods.
- Relative errors of the new method were less than 0.15 %, improving the accuracy of the flow measurement.
- Proposed an innovative method that eliminates the bias caused by non-integer sampling periods.
- Improved noise resistance and accuracy under dynamic conditions.
- Reduced computational complexity over existing high-accuracy methods such as DTFT.
- Extended the applications of Coriolis mass flow signal processing to scenarios where the frequency is unknown.
- Proposed a correlation-based scheme combined with Hilbert that is different from traditional approaches.

2.2.4. Limitations

Algorithm complexity: Although it improves accuracy, it may be more complex than necessary for applications where only phase measurement is needed, without frequency estimation.

Limited applicability: Currently the method was tested in controlled flow environments; it remains to be tested in more diverse energy fields.

Processing: Although the computational burden is lower than in DTFT, it is still higher than purely correlation or zero-crossing detection methods on simple signals.

2.3. A Simple Parametric Design Model for Straight-Tube Coriolis Flowmeters [3]

2.3.1. Objective

Develop a simple parametric model to predict the sensitivity and natural frequency of straight-tube Coriolis flowmeters while minimizing reliance on expensive numerical simulations.

2.3.2. Methodology

A one-dimensional (1D) numerical simulation, based on the finite difference method, is used to derive a parametric model characterized by three dimensionless parameters: bending stiffness (Σ), proximity to the buckling limit (R) and inter-sensor separation distance (χ). The model is experimentally validated using 11 data sets.

2.3.3. Results and Methodology

The model predicts sensitivity with a margin of error of 2 to 5% and allows estimation of natural frequency, providing designers with a quick and intuitive tool to optimize sensor performance.

2.3.4. Limitations

The one-dimensional (1D) approach may not fully capture complex three-dimensional dynamics, and validation was performed under a limited range of conditions and materials.

2.4. Study on Resonant Frequency Control in Coriolis Meters Using PLL [4]

2.4.1. Objective

To evaluate the performance of a Phase-Locked Loop (PLL) control system to maintain the resonant frequency in a straight-tube Coriolis meter, ensuring accurate fluid density and mass flow measurements under different operating conditions.

2.4.2. Methodology

DC component analysis for frequency correction, phase comparison between signals to maintain resonance (90° phase shift), electromagnetic excitation and vibration detection using piezoelectric accelerometers, real-time control with a PLL implemented in LabVIEW.

2.4.3. Results and Contributions

Settling time: ~2 s (to reach 90% of the target frequency), steady-state error: ±0.003 Hz (high stability), proven adaptability to density changes (e.g., water-to-air transition).

2.4.4. Limitations

Simplified linear model (does not consider real nonlinearities), single accelerometer configuration (does not fully evaluate Coriolis flow effects). Tests are only under steady-state conditions (not rapid dynamics) and extreme conditions (pressure, transient flows).

2.5. Comparison of Methods

The proposed hybrid approach (PLL + DFT) outperforms traditional correlation-based methods [2] in three critical aspects for low-cost Coriolis flowmeters: Accuracy, with a phase error of <±0.2° (vs. ~0.5°–1° for correlation methods), crucial for mass flow accuracy; Computational efficiency, as the PLL operates in O(n) per sample and the DFT in O(N log N), avoiding the O(N^2) cost of cross-correlation; and adaptability to noise and non-stationary frequencies, where the PLL dynamically tracks phase shifts while correlation requires additional adaptive filters (e.g., ANF). This combination enables real-time performance on resource-constrained hardware (e.g., ESP32), addressing a key gap in affordable industrial solutions.

2.6. Analysis

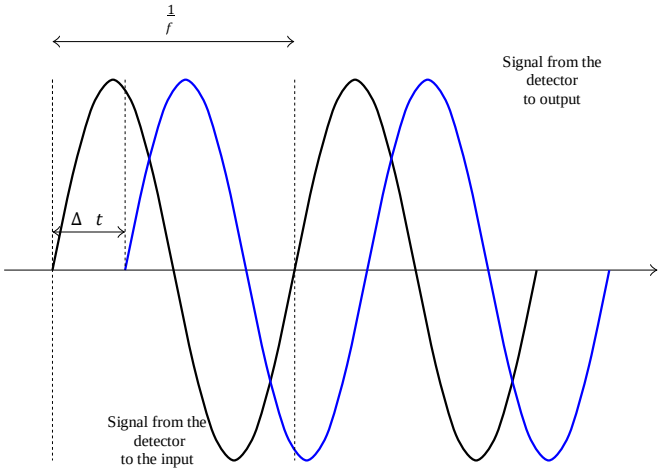

Fig. 2 As the liquid flows through the oscillating sensor tubes, the coriolis force is produced, the transmitter of the measuring system processes the sinusoidal signal from the detectors and determines the phase shift, Δt (μs), of the signal between the sensor input and output

3. Resources and Methods

3.1. Computational Resources

In this study, a computational simulation of defase of a stainless-steel straight tube Coriolis sensor was performed. The resources used were the following:

3.1.1. Software

- Wokwi is a tool that includes an Arduino project simulator and supports STM32 and ESP32 microcontrollers. In addition to running code and demonstrating its functionality, it can visually simulate moving objects and connecting boards, wires, and other components [5].
- The GNU Octave: Programming environment is used to solve the differential equations describing the tube vibration and apply the Discrete Fourier Transform (DFT).
- Custom algorithms: Octave scripts were created to calculate the phase shift, frequency, and amplitude.

3.1.2. Hardware

ESP32: The ESP32 DevKit v1 is a development board based on the ESP32-WROOM-32 module, which integrates a 32-bit, 240 MHz Xtensa LX6 dual-core Xtensa LX6 microcontroller with Wi-Fi and Bluetooth connectivity. It has 520 KB of internal SRAM memory and typically 4 MB of external flash memory. This board provides a wide range of

peripheral interfaces, including digital GPIOs, 12-bit ADCs, I2C, SPI and UART buses, ideal for embedded and IoT applications. Figure 2 shows the simulated module [6].

MPU: The InvenSense MPU-6050 integrated circuit contains a MEMS accelerometer and gyroscope in a single package. 16-bit analog-to-digital converter on all axes. The gyroscope has four user-programmable scales: ±250, ±500, ±1000 and ±2000 °/sec (dps). The accelerometer scale is user programmable with values of ±2g, ±4g, ±8g and ±16g. Integrated temperature sensor. Programmable low-pass filter [7].

4-Digit display Module TM1637: It is a kind of special Light-Emitting Diode (LED) display drive control circuit with keypad scanning interface. It is internally integrated with the MCU digital interface, latch data, LED high-pressure unit and keypad scanning. This product is in a DIP20/SOP20 package, type with excellent performance and high quality, which is mainly applicable to the display drive of induction stoves, microwave ovens and small household appliances [8].

3.2. Methodology

This paper is based on the simulation of the phase angle detection of a Coriolis flowmeter using a digital pll and a DFT. Full details of the simulation methodology are described in the Simulation Methodology section. The use of widely available commercial components, such as the MPU6050 inertial sensor, the ESP32 microcontroller, a TM1637 7-segment display, and a 3.5V rechargeable battery, enables the development of a functional, compact, and low-cost phase measurement system. These components, commonly used in electronics and prototyping projects, offer an affordable alternative to commercial Coriolis measurement systems, which can cost over US$2,000. The ESP32, with Wi-Fi/Bluetooth connectivity and greater processing power than traditional microcontrollers such as the Arduino Uno, enables local data storage and processing, such as natural frequency calculation and phase estimation. The MPU6050 sensor, which combines an accelerometer and a gyroscope, communicates via I²C, facilitating the connection of multiple sensors with only two pins (SDA and SCL), reducing circuit complexity.

Table 1. Advantages and disadvantages of the prototype

Criteria	Proposed System	Coriolis Sensor
Total Cost	Low (< US$200)	High (US$2000 to US$5000)
Component Availability	High (commercial, online, maker-friendly)	Limited (only authorized distributors)
Technical Accessibility	High (open hardware, easy to program)	Low (closed systems, restricted documentation)
Measurement Accuracy	Medium (±0.2° phase error)	High (certified accuracy and factory calibration)
Size and Portability	High (compact, integrated rechargeable battery)	Medium to Low (requires industrial power and fixed installation)
User Interface	Basic (TM1637 display, serial, optional Wi-Fi/Bluetooth)	Advanced (graphical displays, industrial communication: HART, Modbus, etc.)
Ease of Maintenance	High (modular, replaceable components)	Low (requires specialized technical service)
Physical Robustness	Medium (assembly-dependent)	High (sealed housing, resistant to vibrations and chemicals)
Laboratory Reproducibility	High (ideal for prototypes and academic validation)	Low (difficult to modify or integrate into open experiments)
Application in Critical Environments	Limited (not certified for demanding industrial settings)	High (complies with international standards and certifications)

4. Mathematical Formulation of Digital PPL and Phase Detection from DFT

4.1. Signal Generation

Ideal signals are modeled as:

$$x[n] = A\cos(2\pi f_0 t[n] + \theta_x) \quad (2)$$

$$y[n] = A\cos(2\pi f_0 t[n] + \theta_y) \quad (3)$$

$$x_{noisy}[n] = x[n] + \sigma_\omega N(0,1) \quad (4)$$

Where σ_ω is the standard deviation of the noise

$$\boldsymbol{SNR = 10\,log_{10}(\frac{A^2/2}{\sigma_\omega{}^2})} \quad (5)$$

Phase detection

$$e[n] = x_{noisy}[n].vco[n] \quad (6)$$

$$vco[n] = sin(2\pi\hat{f}[n]t_n + \widehat{\emptyset}[n]) \quad (7)$$

$$\hat{f}[n+1] = \hat{f}[n] + \alpha[n]e[n] + \beta \sum_{k=0}^{n} e[k] \quad (8)$$

Where $\hat{f}[n]$ is the estimated frequency y $\widehat{\emptyset}[n]$ is the estimated phase

$$\alpha[n] = \alpha_0 e^{-\gamma n} \quad (9)$$

DFT

$$\boldsymbol{\omega_H[n] = 0.5\left(1 - cos\left(\frac{2\pi n}{N-1}\right)\right)} \quad (10)$$

$$X[k] = \sum_{n=0}^{N-1} x[n].\omega_H[n].e^{-j2\pi kn/N} \quad (11)$$

Phase estimation

$$\theta_x = tan^{-1}\left(\frac{Im\{X[k]\}}{Re\{X[k]\}}\right) \quad (12)$$

5. Simulation Methodology

5.1. Electronic Diagram

Two MPU6050 sensors, an ESP32, a 7-segment and 4-digit display, and a rechargeable battery with a 3.5V output are used to assemble the electronic components. The ESP32 offers better data storage capacity for natural frequency calculation with the MPU6050 sensor using I2C communication for programming. The sensors are powered with 3 to 5v which is connected to the VCC and GND terminals of the ESP32, the SCL pin which is the line of the clock pulses that synchronizes the system goes to pin 21 of the ESP32 on the two sensors and the SDA pin which is the line where the data is transferred between devices goes to pin 22.

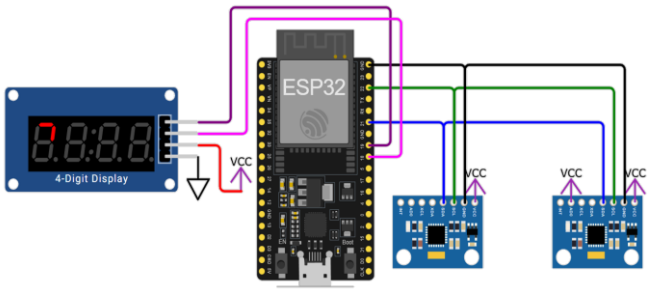

Fig. 3 Diagram for Wokwi simulation

Table 2. Details of the circuit components

Quantity	Description	Cost
1	Esp32	S/ 34.00
2	MPU6050	S/ 30.00
1	TM1637	S/ 10.00
1	Drive Coil	S/ 85.31
1	4000mah Powerbank	S/ 39.00
1	on/off switch	S/ 2.50
10	Jumpers	S/ 5.90
Total		S/ 206.71

Table 3 Summarizes the parameters used in the phasing simulation; these values were selected based on the resonance frequency of a straight stainless steel tube.

Table 3. Parameters used in the simulation

Symbol	Description	Value	Unit
A	Signal amplitude	1	V
f_0	Fundamental frequency	$115 \approx 723rad$	Hz
θ_x, θ_y	Initial phase	0, $\pi/6$	rad
$\emptyset$	Phase difference	30º	rad
t	Continuous time	2	ms
n	Discrete index	2000	dimensionless
T_s	Sampling period	2	ms
f_s	Sampling frequency	2000	Hz
SNR	Signal-to-noise ratio	20	dB

5.2. Phase Change and Mass Flow as a Function of Fluid Velocity

The phase shift (Δφ) and mass flow rate ($\dot{m}$) were calculated as a function of fluid velocity (V) using established mathematical models. The results show that the phase shift decreases as the fluid velocity increases, while the mass flow rate follows an increasing linear trend.

5.2.1. Mathematical Model Used

The time lag is calculated using the following Equation [9]:

$$\Delta\phi = c\left(\frac{2f_{21}g_1M_fV}{L\theta_2(M_f+M_t)(g_1^2-g_2^2)}\right)A_1 \quad (13)$$

Where:

- ($\Delta\phi$): Phase difference.
- ($\dot{m}$): Mass flow.
- (c): Modified amplitude.
- (f_{21}): Factor related to vibration modes.
- (g_1, g_2): Sensor-related frequencies.
- (M_f): Fluid mass.
- (V): Fluid velocity.
- (L): Tube length.
- (θ_2): Factor related to the second vibration mode.
- (A_1): Amplitude ratio.

The mass flow is calculated as: [9]

$$\dot{m} = \frac{\Delta\phi\, L\, \theta_2\,(M_f+M_t)\,(g_1^2-g_2^2)}{2A_1g_1cf_{21}}. \quad (14)$$

5.2.2. Derivation of the Second Expression

To derive the second expression, we start from the first equation and isolate the mass flow rate. ($\dot{m}$). The procedure is detailed below:

- Starting with the phase difference in Equation (13)
- Isolating the term $(M_f V)$:

$$M_f V = \frac{\Delta\phi\, L\, \theta_2 (M_f + M_t)(g_1^2 - g_2^2)}{2A_1 g_1 c f_{21}} \quad (15)$$

- Rate $(M_f V)$ for mass flow $(\dot{m})$, since

$$\dot{m} = M_f V \quad (16)$$

- Finally, the mass flow is obtained.

$$\dot{m} = \frac{\Delta\phi\, L\, \theta_2 (M_f + M_t)(g_1^2 - g_2^2)}{2A_1 g_1 c f_{21}} \quad (17)$$

5.3. Description of the Program Used

5.3.1. Program 1: Phase Difference Estimation

It first shows a general algorithm for measuring the phase difference between two sinusoidal signals with noise.

Algorithm 1: Phase Difference Estimation

Imputs
$f_s = 2000\ Hz.$
$N = 2000.$
$f_0 = 115$ Hz.
$\emptyset = 30°.$
Outputs
$f_{est}.$
$\emptyset_{est}$
Initialize parameters
Generate signals $x(t)$ and $y(t)$
Add noise
Estimate frequency using PLL
Calculate phases using DFT
Compute phase difference
Display results

5.3.2. Program 2: Generation of Sinusoidal Signals

It generates two sinusoidal signals with a specific phase difference. The sampling frequency f_s and the number of samples N are used to generate a discrete time vector t first. Then, the desired phase difference $(\Delta\phi)$ is converted from degrees to radians $((\theta_2)$. In the end, two signals are obtained, one of which is phase-shifted.

Algorithm 2: Sinusoidal Signal Generation

$t \leftarrow [0: N-1]/f_s$
$\theta_2 \leftarrow \Delta\phi_{real} \times \pi/180$
$x \leftarrow \cos(2\pi f_{real} t)$
$y \leftarrow \cos(2\pi f_{real} t + \theta_2)$

5.3.3. Program 3: Gaussian Noise

To replicate realistic circumstances, Gaussian noise is added to the clean sinusoidal signals x(t) and y(t). To get the desired Signal-to-Noise Ratio (SNR), it first determines the power of the original signal (P_x) and the noise power (P_n). The noisy versions x_{noisy} and y_{noisy} are then produced by adding Gaussian noise with zero mean and standard deviation $\sqrt{P_n}$ to both signals. This procedure makes it possible to assess the system's performance in interference-prone environments.

Algorithm 3: Add Gaussian Noise

Calculate signal power $P_x \leftarrow var(x)$
Calculate signal power $P_n \leftarrow P_x / 10^{\frac{SNR}{10}}$
Generate noisy $n \leftarrow N(0, \sqrt{P_n})$
$x_{noisy} \leftarrow$ x + n
$y_{noisy} \leftarrow$ y + n

5.3.4. Program 4: PLL

Uses a Phase-Locked Loop (PLL) to estimate a noisy signal's phase and frequency. The parameters are initialized (α = 0.003, β=0.00003), and for every sample, a reference vco signal is generated, the error is calculated by multiplying the input signal by the vco, the frequency and phase are adjusted using a PI (Proportional-Integral) filter, and the estimates are updated iteratively. The resulting signals, estimated frequency (f_{est}) and estimated phase (φ_{est}), gradually approach the input signal's true values.

Algorithm 4: Phase-Locked Loop

Inputs: Signal, f_s, f_{init}, t
Outputs: $f_{est}, \emptyset_{est}$
Initialize:
$\alpha \leftarrow 0.003, \quad \beta \leftarrow 0.00003$
$f_{est}[0] \leftarrow f_{init}, \emptyset_{est}[0] \leftarrow 0$
$e_{init} \leftarrow 0$
for k ← 1 to length(signal) do
vco ← $\sin(2\pi f_{est}[k-1] t[k] + \emptyset_{est}[k-1]$
$e \leftarrow$ signal[k]. vco
$e_{int} \leftarrow e_{int} + \beta.$ e
$f_{est}[k] \leftarrow f_{est}[k-1] + \alpha.$ e $+ e_{int}$
$\emptyset_{est}[k] \leftarrow \emptyset_{est}[k-1] + 2\pi f_{est}[k]/f_s$

5.3.5. Program 5: DFT

It uses a ventaneated DFT to determine a signal's phase. To lessen spectral leakage, it first creates and applies a Hann window.

After that, it computes the windowed signal's FFT, finds the frequency bin that is closest to the earlier estimate (f_{est}), and uses the atan2 function on the real and imaginary components of that bin to determine the phase.

Because of the fenestration that reduces frequency-domain artifacts, this technique offers a precise phase estimation even in noisy signals. The estimated phase (φ_{est}) in the bin of interest is the outcome.

Algorithm 5: DFT

1: Inputs: signal, f_s, f_{est}, N
2: Outputs: $\emptyset_x$
3: Generate Hann window:
$\omega \leftarrow 0.5(1 - \cos(2\pi[0: N-1]/N)$
4: Apply window
$x_{win} \leftarrow signal.\omega$
5: Calculate DFT:
$X \leftarrow FFT(x_{win})$
6: Find a good reference.
$f_{axis} \leftarrow [0: N-1].f_s/N$
$k_{est} \leftarrow \arg\min_k |f_{axis} - f_{est}|$
7: Calculate phase
$\emptyset_x \leftarrow \text{atan2}(\text{Im}(\text{X}[k_{est}]), \text{Re}(\text{X}[k_{est}]))$

5.3.6. Program 6: Phase Difference

Calculates and corrects the phase difference between two signals. It first obtains the raw difference ($\Delta\emptyset = \emptyset_y - \emptyset_x$), then uses the atan2 function to correct the result to ensure that it is in the range [-π, π] radians. It then converts this difference to degrees ($\Delta\emptyset_{deg}$) and adjusts the value to the range [0°,360°] by adding 360° if it is negative.

Algorithm 6: Phase difference

1: Calculate gross difference
$\Delta\emptyset = \emptyset_y - \emptyset_x$
2: Correct envelope:
$\Delta\emptyset_{corr} \leftarrow \text{atan2}(\sin(\Delta\emptyset), \cos(\Delta\emptyset))$
3: Convert to degrees
$\Delta\emptyset_{deg} \leftarrow \Delta\emptyset_{corr}.180/\pi$
4: Adjust to range [0º: 360º]:
if $\Delta\emptyset_{deg} < 0$ then
$\Delta\emptyset_{deg} \leftarrow \Delta\emptyset_{deg} + 360$
5: end if

6. Results and Discussion

The phase estimation algorithm is reliable (errors < ±0.2°), except near 360° (-0.179° error at 359°). The estimated frequency seems to be insensitive to phase changes, which indicates that it could be fixed or calculated with another criterion (e.g. constant average). An error of ±0.2° is equivalent to a margin of ±0.5% in mass flow rate for fluids such as water, acceptable in chemical dosing (ISO 10790 standard).

Table 4 Summarizes the frequency phase differences estimated for a real phase, the results were given according to the code and the parameters of Table 2, knowing that the real frequency is 115 Hz. The following image shows a time domain signal plot.

Two sinusoidal signals labeled “Signal x” (in blue) and “Signal y” (in red), both affected by noise, are presented. The “y” signal is out of phase with respect to the “x” signal, evidenced by the horizontal offset between their peaks. The horizontal axis represents time (in seconds) and the vertical axis represents amplitude. This graph is useful to visualize the phase difference between the two signals, which is fundamental in the estimation of the mass flow in Coriolis-type sensors.

Tabla 4. Simulation table at different grades

Phase difference Δ∅ (deg)	Estimated phase difference $\Delta\emptyset_{est}$(deg)	Estimated frequency f_{est}(Hz)
30	30.024	114.803
60	59.836	114.843
90	89.812	114.833
120	119.922	114.823
150	150.268	114.905
180	180.093	114.911
320	320.062	114.877
359	358.821	114.893

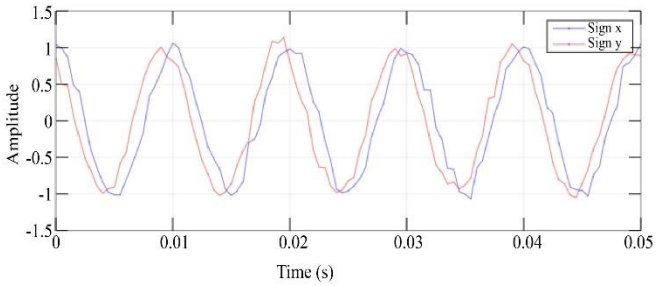

Fig. 4 Amplitude (v) vs. Time (s) relationship at a 30º displacement with Gaussian noise

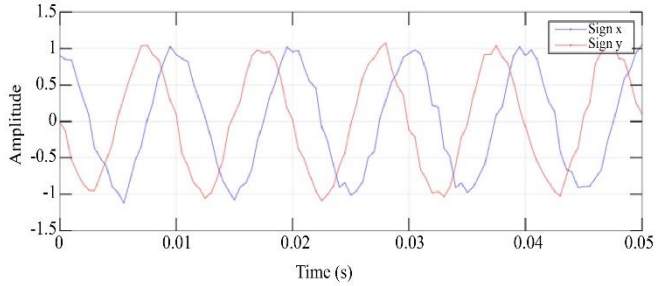

Fig. 5 Amplitude (v) vs. Time (s) relationship at a 90º displacement with Gaussian noise

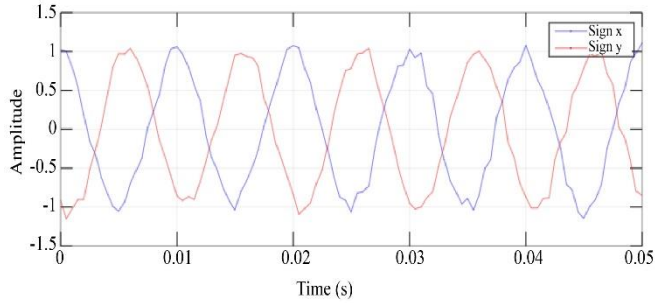

Fig. 6 Amplitude (v) vs. Time (s) relationship at a 150º displacement with Gaussian noise

Table 5. Simulation table in different degrees

$\Delta\emptyset$ (deg)	$\Delta\emptyset_{est}$(deg)	Absolute error (°)	Relative error (%)
30	30.024	0.024	0.080
60	59.836	0.164	0.273
90	89.812	0.188	0.209
120	119.922	0.078	0.065
150	150.268	0.268	0.179
180	180.093	0.093	0.052
320	320.062	0.062	0.019
359	358.821	0.179	0.050

Mean:

$$\bar{e} = \frac{1}{8}\sum_{i=1}^{8}|\Delta\emptyset - \Delta\emptyset_{est}| \quad (18)$$

$$\bar{e} = 0.132° \quad (19)$$

Standard deviation:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i - \bar{e})^2} \quad (20)$$

$$\sigma = 0.077° \quad (21)$$

The value of ±0.2° covers more than 87.5% of the measured data and falls about one standard deviation above the mean. Therefore, it is a statistically representative and conservative margin.

Table 6. Phase difference at different frequencies

$\Delta\emptyset$ real (°) a 100 Hz	$\Delta\emptyset$ est (°) a 100Hz	$\Delta\emptyset$ real (°) a 100 Hz	$\Delta\emptyset$ est (°) a 100Hz
30	30.027	30	30.098
60	59.831	60	60.204
90	89.808	90	90.184
120	119.082	120	119.052
150	150.081	150	150.24
180	180.093	180	179.03
320	320.212	320	320.012
359	358.941	359	358.98

7. Conclusion

To estimate mass flow rate by examining the angular phase shift caused by the Coriolis effect, this work demonstrated the feasibility of implementing an embedded system for a low-cost straight-tube Coriolis sensor using an ESP32 microcontroller and MPU-6050 inertial sensors. The phase difference between the gyroscope signals could be accurately estimated by translating it into an optimized computational algorithm in C. The use of digital signal processing methods, such as the Discrete Fourier Transform (DFT), in combination with a Phase-Locked Loop (PLL), allowed the vibration frequency and relative phase angle, two crucial parameters for determining mass flow rate, to be estimated reliably and efficiently. Using time windowing (Hann) and proportional-integral control filtering strategies proved effective even in the presence of noise and disturbances typical of industrial environments, such as flow pulsations. The suggested approach offers a lower error rate in both frequency estimation and phase difference, according to system validation, based on a comparison with the results from Octave software.

However, to strengthen the conclusions, it is recommended to complement these results with case studies in real-life industrial environments, evaluating factors such as electromagnetic interference, temperature variations, or multiphase flows. This would more convincingly demonstrate the potential of these sensors for practical applications, especially when scalability, cost, and ease of use are priorities.

7.1. Future Works

Based on the design simulated and validated in this study, the physical implementation of the straight-tube Coriolis sensor prototype is suggested as a next step. The main objective will be to build and test the entire system-which includes the ESP32 microcontroller, the MPU-6050 sensors, and the stainless-steel-tube-under realistic mass flow conditions. During this phase, several important factors will be taken into account:

- Experimental calibration: To create an accurate quantitative relationship between the measured deviation and the actual mass flow rate, the system must be carefully calibrated with a commercial reference flowmeter and validated in at least three representative industrial scenarios (e.g., laminar, turbulent, and multiphase flow).
- Design and construction of the mechanical support: To reduce external vibrations, ensure proper sensor alignment, and allow for installation in realistic fluid conduits, a robust structure will be created. Its performance will be evaluated under typical low mechanical vibrations (e.g., 10–100 Hz ranges) to ensure its robustness. Due to its ability to produce precise and regulated displacements at high frequencies, the moving coil linear motor model GVCM-016-010-01 [10] was demonstrated for the actuator part. Thanks to its fast response, high acceleration and superior dynamic control without the need for an intermediate mechanical transmission, this type of actuator is especially suitable for stimulating the system at its resonant frequency. Nevertheless, future work will have to quantify its energy efficiency and lifetime in continuous operation, since these aspects are critical for its industrial adoption.

Water and solutions with different densities and viscosities will be used to test sensor performance and examine how they affect system sensitivity and linearity. Additionally, it is recommended to include non-Newtonian fluids and gas-liquid mixtures to cover a broader range of applications.

Stability and repeatability analysis: Extensive experiments will be conducted to evaluate measurement repeatability in the presence of disturbances, with a minimum of 100 test cycles per condition.

The stability of the detected phase shift will also be examined under varying pressure or temperature conditions ranging from 0–100°C and 1–10 bar, parameters common in industrial environments.

The creation of this prototype will pave the way for the incorporation of low-cost Coriolis sensors into industrial flow monitoring and control systems. To ensure adoption, experimental results must be published against ISO or ASTM standards and validated with at least one industrial partner. Furthermore, the data obtained will allow fine-tuning the mathematical model and optimizing the algorithm for complex scenarios.

Acknowledgment

The authors would like to thank the National University of San Agustin (UNSA) and Dr. Fernando from the Federal University of Alagoas (UFAL) for their support during the development of this research. Special thanks to colleagues who provided valuable comments and suggestions during the preparation of this manuscript.

References

[1] Evelina Ekström, "*Evaluation and Optimatization of PolyCor - A Single- Use Coriolis Flowmeter*," Master's Thesis, Umea University, pp. 1-45, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Tao Wang, and Roger Baker, "Coriolis Flowmeters: A Review of Developments Over the Past 20 Years, and An Assessment of the State of the Art and Likely Future Directions," *Flow Measurement and Instrumentation*, vol. 40, pp. 99-123, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jože Kutin, and Ivan Bajsić, "PLL Control of the Coriolis Meter Resonance Frequency," *Conference: Proceedings 16th IMEKO World Congress*, Vienna, Austria, vol. 6, pp. 25-28, 2000. [Google Scholar]

[4] C.L. Ford, "A Simple Parametric Design Model for Straight-Tube Coriolis Flow Meters," *Flow Measurement and Instrumentation*, vol. 79, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Wokwi: A Simulator for Arduino, ESP32, and STM32 Projects, Microsiervos, 2022. [Online]. Available: https://www.microsiervos.com/archivo/hackers/wokwi-simulador-proyectos-arduino-esp32-stm32.html

[6] ESP32 DevKit V1 NodeMCU 32 30 PIN ESP32 WIFI Micro USB, Naylamp Mechatronics, 2023. [Online]. Available: https://naylampmechatronics.com/espressif-esp/384-esp32-devkit-v1-nodemcu-32-30-pin-esp32-wifi-micro-usb.html

[7] Interface MPU6050 Accelerometer and Gyroscope Tutorial with Arduino, Last Minute Engineers, 2025. [Online]. Available: https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/

[8] Titan Micro Electronics, LED Drive Control Special Circuit TM1637, 2023. [Online]. Available: https://uelectronics.com/wp-content/uploads/2018/01/AR0217-Modulo-4-Digitos-7-Segmentos-TM1637-Datasheet.pdf?srsltid=AfmBOopnRItM8ew-ZoWtYC9Q9YN8VC0Tj_ERRn7y5oNwpWoKn_zrMTgs

[9] R. Cheesewright, and C. Clark, "The Effect of Flow Pulsations on Coriolis Mass Flow Meters," *Journal of Fluids and Structures*, vol. 12, no. 8, pp. 1025-1039, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[10] Linear Voice Coil Motors with Shaft and Internal Bearing, Moticont. [Online]. Available: https://www.moticont.com/GVCM-016-010-01.htm

[11] G. Bobovnik, J. Kutin, and I. Bajsic, "The Effect of Flow Conditions on the Sensitivity of the Coriolis Flowmeter," *Flow Measurement and Instrumentation*, vol. 15, no. 2, pp. 69-76, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[12] Evgeniia Shavrina et al., "Fluid-Solid Interaction Simulation Methodology for Coriolis Flowmeter Operation Analysis," *Sensors*, vol. 21, no. 23, pp. 1-20, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Steven C. Chapra, and Raymond P. Canale, Numerical Methods for Engineers, 3rd ed., Mexico, McGraw-Hill, 2000. [Google Scholar] [Publisher Link]

[14] Dennis G. Zill, and Warren S. Wright, *Advanced Engineering Mathematics*, 4th ed., Jones and Bartlett Publishers, USA, 2011.

[15] C. R. Wylie, *Advanced Engineering Mathematics*, 3rd ed., McGraw-Hill, USA, 1966.

[16] Lisandro Massera, Mauro Podoreska, and Mónica Romero, "Coriolis Mass Audible Meter Design and Implementation," Computational Mechanics, vol. XXVI, no. 35, pp. 3019-3042, 2007. [Google Scholar] [Publisher Link]

Appendix 1: Link to the Mendeley database

Additional resources are available at the Mendeley link below. They include an explanatory video of the model and its implementation, and the open-source software files and code used in the simulations. [Publisher Link]

Appendix 2: Simulation in Wokwi

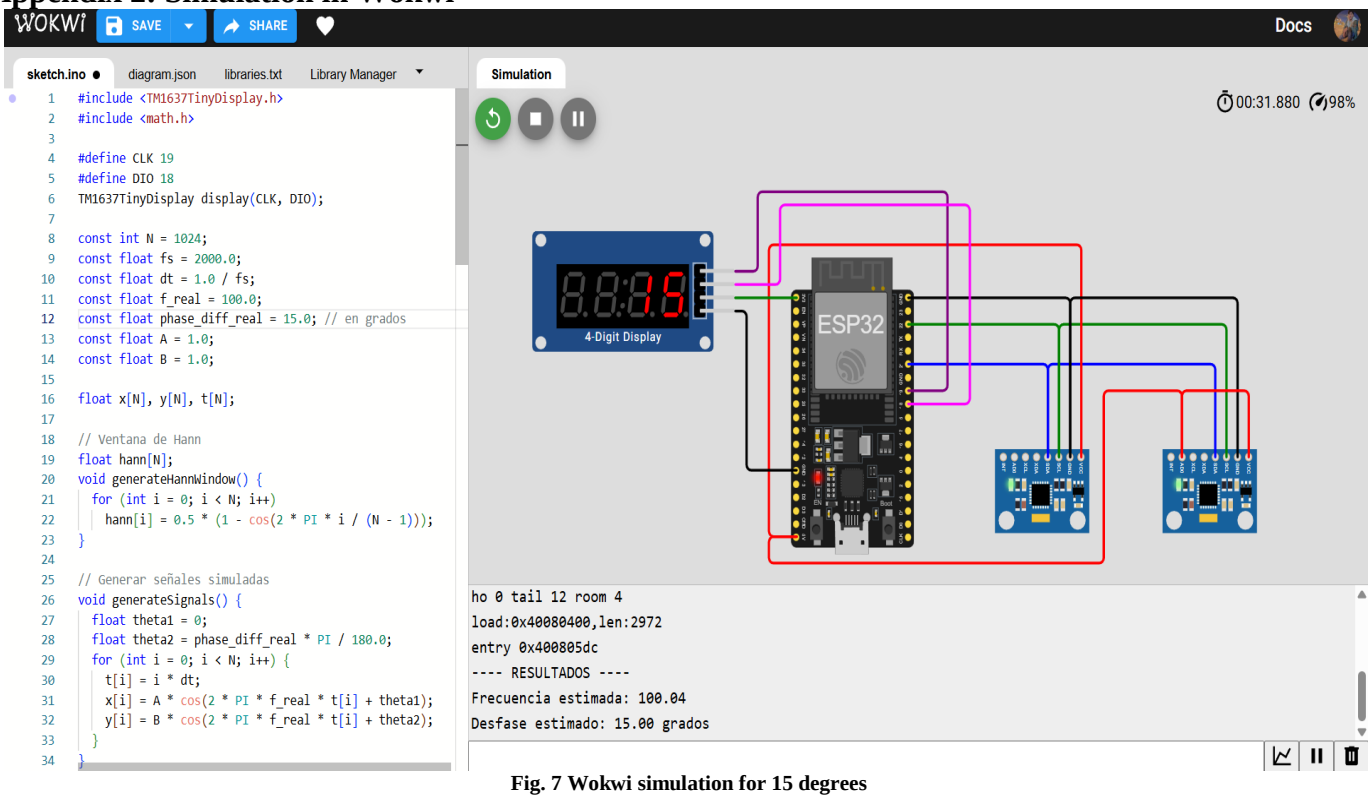

Fig. 7 Wokwi simulation for 15 degrees

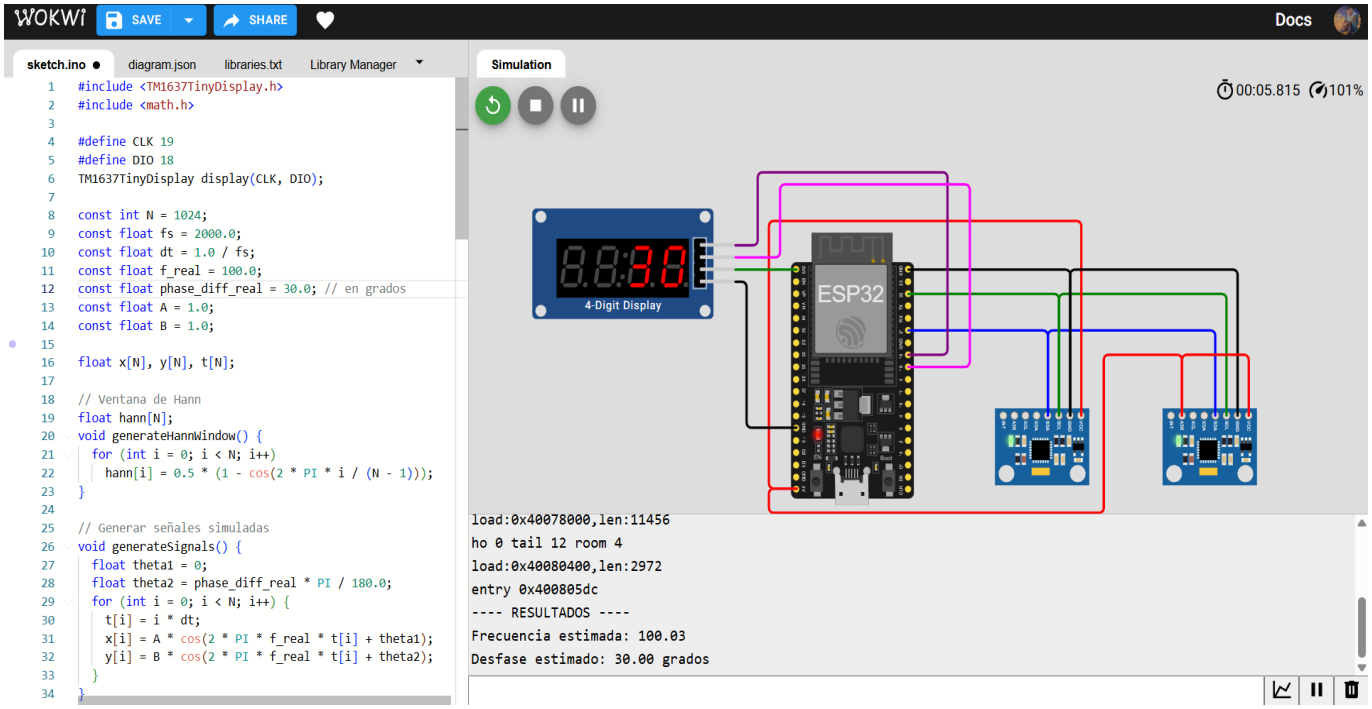

Fig. 8 Wokwi simulation for 30 degrees

Appendix 3: Flowchart

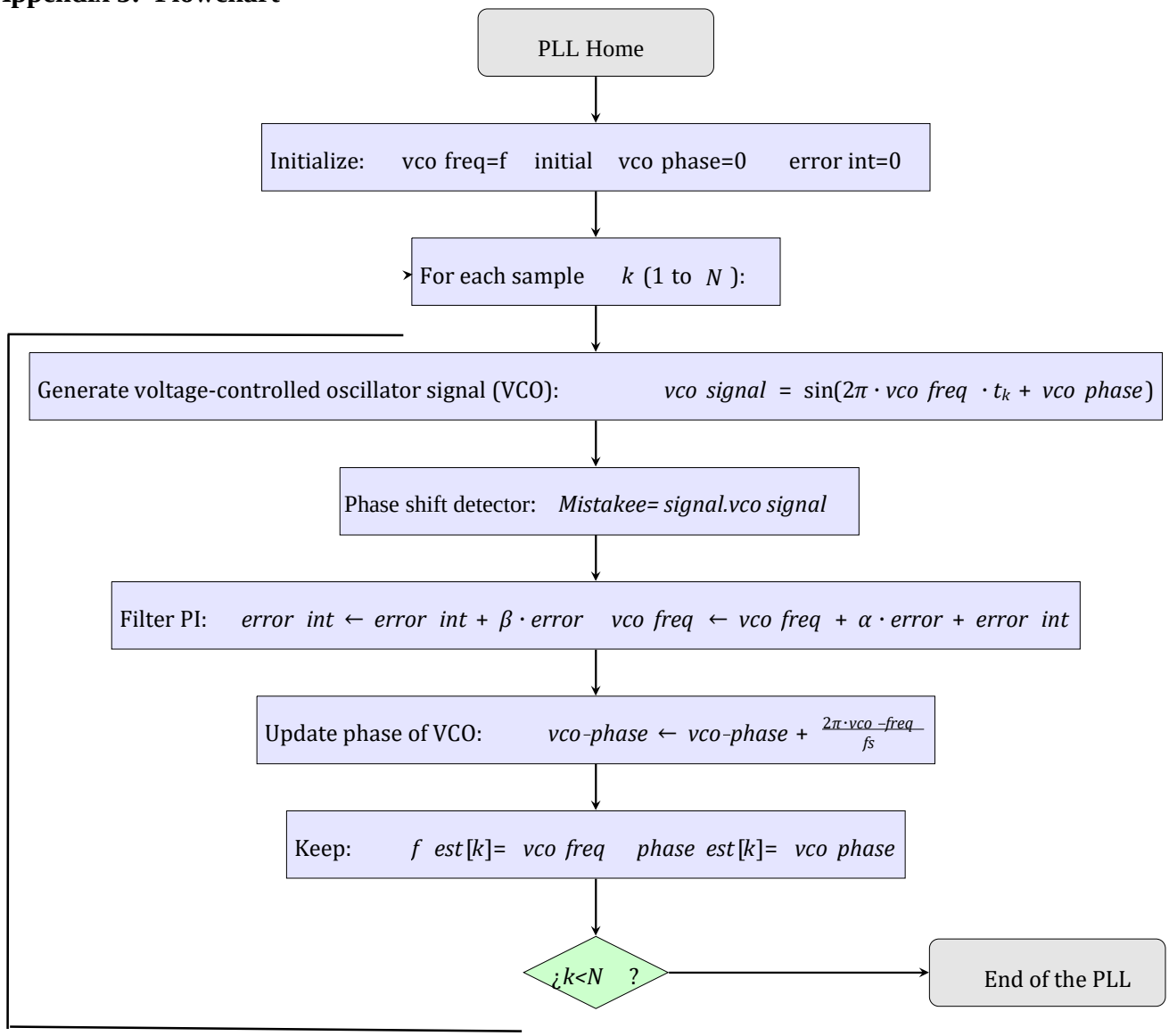

Fig. 9 Flowchart 1: PLL algorithm

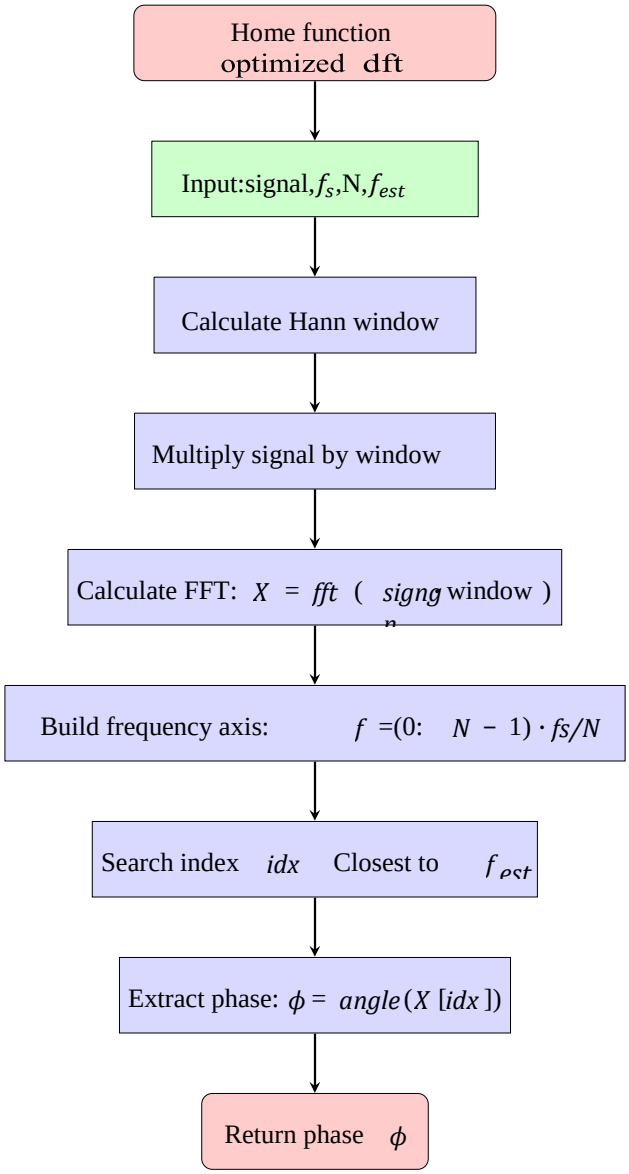

Fig. 10 Flowchart 2: DFT algorithm

Appendix 4: Code for Simulation

Code 1: Code in Octave without Hardware

```
%% Initial setup
clear all; close all; clc;
pkg load signal;

%% Parameters optimized for speed and accuracy
fs = 2000;          % Sampling rate
N = 2048;           % Number of points (speed-accuracy balance)
t = (0:N-1)/fs;     % Time vector
f_real = 115;       % Actual frequency
phase_diff_real = 30; % Actual phase difference (degrees)
A = 1; B = 1;       % Amplitudes
SNR = 20;           % Signal-to-noise ratio

%% Signal generation
theta1 = 0;                         % theta1
```

```
theta2 = deg2rad(phase_diff_real);  % Initial phase signal 2

x = A*cos(2*pi*f_real*t + theta1);
y = B*cos(2*pi*f_real*t + theta2);
x_noisy = awgn(x, SNR, 'measured');
y_noisy = awgn(y, SNR, 'measured');

%% PLL
function [f_est, phase_est] = optimized_pll(signal, fs, f_initial, t)
    alpha = 0.003;   % Optimized proportional gain
    beta = 0.00003;  % Optimized integral gain

    n = length(signal);
    f_est = zeros(1,n);
    phase_est = zeros(1,n);
    error_int = 0;
    vco_phase = 0;
    vco_freq = f_initial;

    for k = 1:n
        % VCO signal
        vco_signal = sin(2*pi*vco_freq*t(k) + vco_phase);

        % Phase detector
        error = signal(k) * vco_signal;

        % PI filter
        error_int = error_int + beta*error;
        vco_freq = vco_freq + alpha*error + error_int;
        vco_phase = vco_phase + 2*pi*vco_freq/fs;

        % Save estimates
        f_est(k) = vco_freq;
        phase_est(k) = vco_phase;
    end
end

%% Frequency estimation with PLL
[f_est_x, ~] = optimized_pll(x_noisy, fs, 110, t);
[f_est_y, ~] = optimized_pll(y_noisy, fs, 120, t);
f_est = mean([f_est_x(end-50:end), f_est_y(end-50:end)]);

%% Optimized DFT (corrected version))
function phase = optimized_dft(signal, fs, N, f_est)
    % Apply Hann window
    window = hann(N)';
    X = fft(signal .* window, N);

    % Find nearest bin
    f_axis = (0:N-1)*fs/N;
    [~, idx] = min(abs(f_axis - f_est));

    % Calculate phase
    phase = angle(X(idx));
end

%%
phase_x = optimized_dft(x_noisy, fs, N, f_est);
phase_y = optimized_dft(y_noisy, fs, N, f_est);

% (sintaxis fix)
phase_diff = phase_y - phase_x;
phase_diff_est = rad2deg(angle(exp(1i*phase_diff)));

% 0-360°
if phase_diff_est < 0
    phase_diff_est = phase_diff_est + 360;
end

%% Show
```

```
figure('Position',[100,100,800,600]);

% Signs in time
subplot(2,1,1);
plot(t, x_noisy, 'b', t, y_noisy, 'r');
title('Signals with noise');
xlabel('Time (s)'); ylabel(' Amplitude ');
legend('sign x', 'Sign y');
xlim([0 0.05]); grid on;

% PLL Convergence
subplot(2,1,2);
plot(t, f_est_x, 'b', t, f_est_y, 'r');
hold on; line([0 t(end)], [f_real f_real], 'Color', 'k', 'LineStyle', '--');
title('Frequency estimation frecuencia (PLL)');
xlabel('Time (s)'); ylabel('Frequncy (Hz)');
legend(' Frequency x', ' Frequency y', 'Real value');
ylim([f_real-10 f_real+10]); grid on;

%% Numerical results
fprintf('\n----------------------------------------\n');
fprintf(' RESULTADOS FINALES\n');
fprintf('----------------------------------------\n');
fprintf(' Frecuencia real: %.2f Hz\n', f_real);
fprintf(' Frecuencia estimada: %.3f Hz (Error: %.3f%%)\n',...
        f_est, 100*abs(f_est-f_real)/f_real);
fprintf(' Diferencia de fase real: %.2f°\n', phase_diff_real);
fprintf(' Diferencia estimada: %.3f° (Error: %.3f%%)\n',...
        phase_diff_est, abs(phase_diff_est-phase_diff_real)/phase_diff_real*100);
fprintf('----------------------------------------\n');
```

Code 2: Code in Wokwi with Hardware

```
#include <TM1637TinyDisplay.h>
#include <math.h>

#define CLK 19
#define DIO 18
TM1637TinyDisplay display(CLK, DIO);

const int N = 2000;
const float fs = 2000.0;
const float dt = 1.0 / fs;
const float f_real = 100.0;
const float phase_diff_real = 50.0; //
const float A = 1.0;
const float B = 1.0;

float x[N], y[N], t[N];

// Ventana de Hann
float hann[N];
void generateHannWindow() {
  for (int i = 0; i < N; i++)
    hann[i] = 0.5 * (1 - cos(2 * PI * i / (N - 1)));
}

// Generar señales simuladas
void generateSignals() {
  float theta1 = 0;
  float theta2 = phase_diff_real * PI / 180.0;
  for (int i = 0; i < N; i++) {
    t[i] = i * dt;
    x[i] = A * cos(2 * PI * f_real * t[i] + theta1);
    y[i] = B * cos(2 * PI * f_real * t[i] + theta2);
  }
}

```

```
// Estimate frequency using PLL
float pll_freq_est(float* s, float f_init) {
  float alpha = 0.003, beta = 0.00003;
  float vco_freq = f_init, vco_phase = 0, err_int = 0;
  float f_sum = 0;
  for (int i = 0; i < N; i++) {
    float vco = sin(2 * PI * vco_freq * t[i] + vco_phase);
    float error = s[i] * vco;
    err_int += beta * error;
    vco_freq += alpha * error + err_int;
    vco_phase += 2 * PI * vco_freq / fs;
    if (i >= N - 50) f_sum += vco_freq;
  }
  return f_sum / 50.0;
}

// Estimar fase con DFT y ventana
float estimate_phase(float* s, float freq_est) {
  float re = 0, im = 0;
  for (int i = 0; i < N; i++) {
    float angle = 2 * PI * freq_est * t[i];
    re += s[i] * hann[i] * cos(angle);
    im -= s[i] * hann[i] * sin(angle);
  }
  return atan2(im, re);
}

// Mostrar en display
void showPhase(float phase_deg) {
  int value = (int)phase_deg;
  display.showNumberDec(value, true);
}

void setup() {
  Serial.begin(115200);
  display.setBrightness(7);
  display.showString("INIT");
  delay(1000);

  generateHannWindow();
  generateSignals();

  float f_est_x = pll_freq_est(x, f_real - 5);
  float f_est_y = pll_freq_est(y, f_real + 5);
  float f_est = (f_est_x + f_est_y) / 2.0;

  float phase_x = estimate_phase(x, f_est);
  float phase_y = estimate_phase(y, f_est);

  float dphi = phase_y - phase_x;
  float deg = dphi * 180.0 / PI;
  if (deg < 0) deg += 360.0;

  Serial.println("----RESULTS ----");
  Serial.print("Estimated frequency:"); Serial.println(f_est);
  Serial.print("Estimated offset"); Serial.print(deg); Serial.println("degrees ");

  showPhase(deg);
}

void loop() {
  // show
  delay(1000);
}
```