Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELÉCTRICA

ÁREA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, DE COMPUTADORES Y SISTEMAS

TRABAJO FIN DE GRADO Nº 18010097

DESARROLLO DE UN SMART METER PARA MEDIDA DE PARÁMETROS ELÉCTRICOS

D. GONZÁLEZ ORTEGA, Sergio
TUTOR: D. BLANCO CHARRO, Cristian

FECHA: OCTUBRE DE 2019

Table of contents.

1. Introduction.

1.1. Aim.

The aim of this project is to design a Smart Meter for measuring the electrical parameters of a three phase, single phase or DC installation.

This project is an extension of the one developed during an Erasmus stay in the Technical University of Cluj-Napoca in Romania, and for which the electrical installation for an off-grid camper vehicle based on solar energy was designed.

The designed Smart Meter would be of great convenience in order to measure and control the production and consumption of electrical energy in said installation.

1.2. Objectives.

In the theoretical aspect, a solid understanding of the methods for obtaining precise measurements of electrical parameters is sought, and an evaluation of different approaches is carried out. This includes synchronisation methods for three-phase and single-phase systems, as well as the necessary transfer functions and operators.

During the simulation stage, obtaining precise results with sufficient rapidity will be prioritised. Familiarisation with the used software is an important part of this project.

Finally, an experimental verification of the device's correct functioning is proposed.

1.3. Background and state of the art.

The *Real Decreto 110/2007* established the requirement for distributors to replace the current electric meters for smart devices which are able to calculate energy consumption in a much more efficient manner than their precursors.

These devices can be connected via the Internet in order for the information to be easily accessible from multiple platforms, as well as for statistical and billing purposes.

The main purpose for the existence of smart meters is energy efficiency. By being able to track energy consumption in a continuous way, a consumer might be able to identify low-efficiency equipment.

Most of the devices that are currently available on the market fall in one of these categories:

- First generation Smart Meters: these devices were the first to be introduced to the general public. They usually include an LCD, or more recently, LED screen on which a few electrical parameters, mainly instantaneous power consumption and energy consumption over a period of time, are showcased (Fig. 1.1).
- Second generation Smart Meters: these devices are coupled to a Wi-Fi or mobile data network, and send the collected data to an external server. These data can be processed and visualised from multiple devices, a common choice being to send them to an app installed on a smartphone. For this reason, the device itself usually does not include a screen (Fig. 1.2).

Fig. 1.1. First generation Smart Meter. Source: Meterus.

Fig. 1.2. Second generation Smart Meter, with a clamp sensor and Wi-Fi connectivity. Source: Leroy Merlin.

An existing commercial system manufactured by Circutor S.A. and named Wi-Beee, functions as a consumption analyser that is able to display instantaneous and historical data via its app or through a built-in internet service [1].

This system is installed by simply locating the module embracing the existing cables in the installation's control box, and measures the flow of current using non-invasive current sensors (Fig. 1.3).

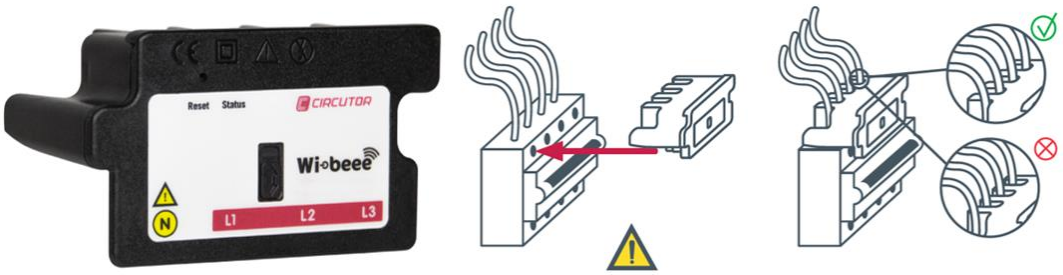

Fig. 1.3. View of the Wi-beee module and installation method.

Source: http://wibeee.circutor.com/index_en.html

A system of this kind reduces the risks associated with modifying existing installations, making them suitable for their use by customers without an extensive knowledge of electrical circuits.

With the emergence of open source microcontroller kits such as Arduino and Raspberry, multiple projects consisting on different approaches to building a smart meter have already been developed.

Most of these projects solely focus on the analysis of the current flow in a single-phase installation, and usually showcase their results on an LCD screen or as graphics on a PC via USB connection.

This, while being useful for analysing isolated appliances or even the total energy consumption of a home, turns out to be insufficient for more complex installations, as well as imprecise due to ignoring voltage fluctuations and harmonic content.

The smart meter developed in this project will perform more detailed calculations based on both current and voltage measurements, and the data produced by it will be available for further processing and representation.

The algorithm that obtains and processes the data will be open-code, in order to propitiate future development of the project.

1.4. Installation from the previously designed project.

During the second semester of an Erasmus+ stay in the Technical University of Cluj-Napoca, Romania, the electrical installation for the conversion of a commercial van model to an off-grid camper vehicle was designed as part of a Final Project course.

This installation included the photovoltaic array mounted on the top of the vehicle, a charge controller, a battery bank for energy storage, an inverter with the option of connection to the external grid, and the DC and AC appliances (Fig. 1.4).

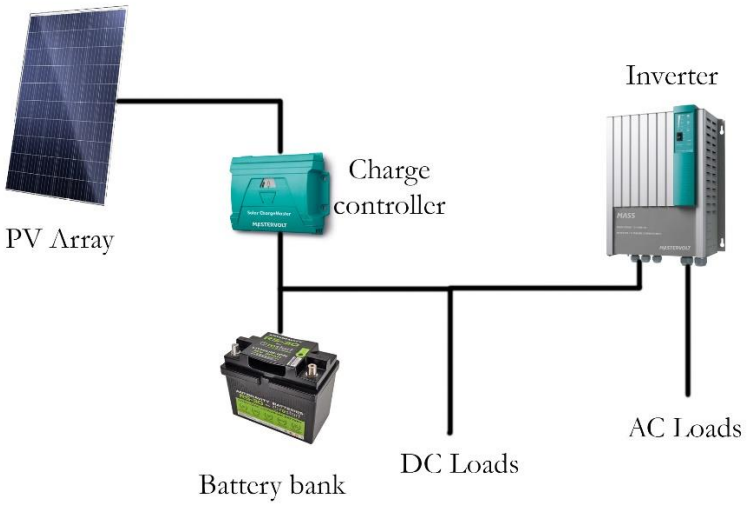

Fig. 1.4. Topology of the electrical installation on the off-grid vehicle.

A total of 600 W_p of peak power can be supplied using four Enerdrive SP-EN150W panels. The array is connected to a Mastervolt SMC40 charge controller, from where the produced energy is automatically distributed to the batteries, the DC loads and an inverter.

The battery bank is composed by two equal Li-ion batteries connected in parallel, with a combined storage capability of 180 Ah. This amount was calculated to be sufficient as to provide the system with an autonomy of two and a half days in the situation of total malfunction of the photovoltaic array.

The total combined power of the DC loads connected directly to the charge controller is 155 W. These DC loads include appliances such as LED lighting, a ceiling fan, the water pumps, USB power outlets and a rear vision camera. Even though it can be assumed that these loads will not work with a total simultaneity factor, the system has been sized for such an event.

A 1600 W inverter is connected between the charge controller, which manages the DC part of the circuit, and the AC installation. This inverter supplies power to the AC appliances, which consist on a refrigerator, a microwave oven and two power outlets.

The total power needed to be supplied was calculated considering that highly demanding loads such as a vacuum cleaner might be plugged in one or both of the power outlets.

A complete diagram of the installation is shown in Fig. 1.5.

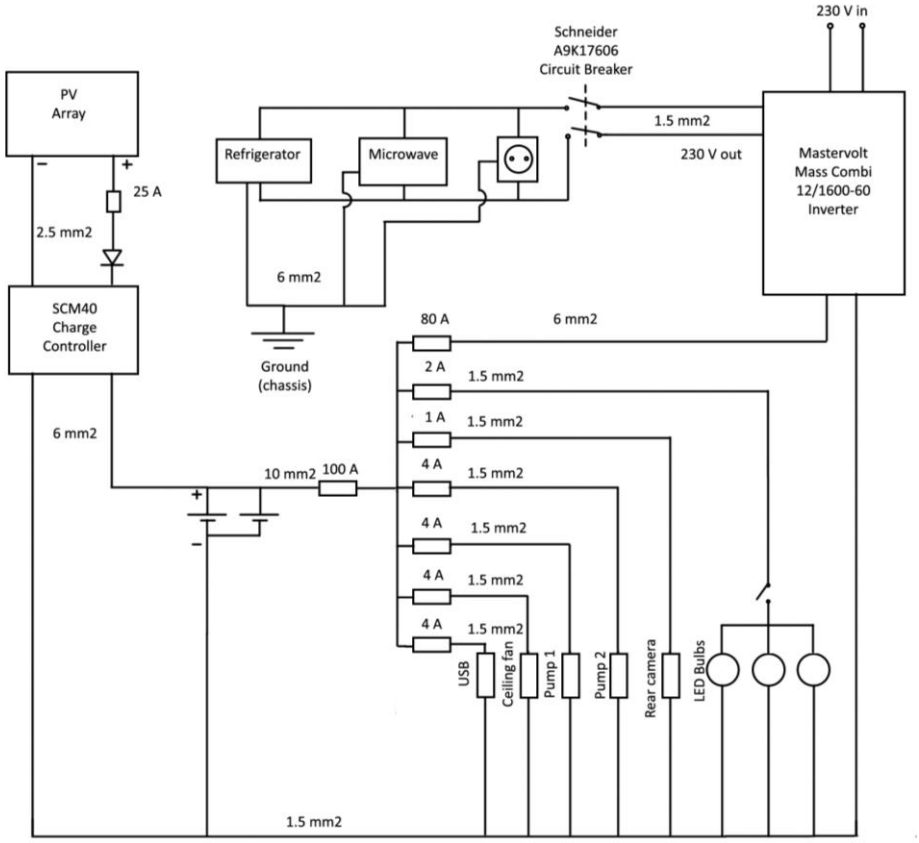

Fig. 1.5. Schematic of the electrical installation.

This project aims to be a complement to provide a hypothetical user with all the data regarding the electrical parameters of the system. Even though some of the installed components are able to showcase consumption data, the system lacks the ability to perform more complete calculations and to export the obtained results.

These results, regarding consumption of energy, production in the PV array, or the quality of the AC voltage, could be visualised in real time by the user, making it easier for them to identify possible necessities or upgrade the installation.

1.5. Proposed solution.

The proposed solution for the implementation of this Smart Meter is to use a Raspberry Pi 3 Model B as the platform on which a measuring algorithm is deployed.

The calculation of the parameters regarding voltage, current and power, is performed by means of a synchronous reference frame phase lock loop (SRF-PLL), a

synchronisation method commonly used for the interconnection of distributed energy sources and the grid.

The development of this SRF-PLL was carried out using Simulink, a graphic programming tool included in the MATLAB environment. The final algorithm was deployed on the Raspberry Pi via the same software package.

Voltage and current sensors acquire the data from the measuring point, and an adaptation stage transforms the read signals to fit a range suitable for the Raspberry Pi.

During this project, the current measuring circuit was designed. Measuring the voltage implies breaking the continuity in the existent installation, and is proposed for further development.

The code developed for measuring parameters in a single-phase installation was deployed to the Raspberry Pi directly from Simulink. However, a proper lecture of the input data was not achieved. For this reason, the code was translated to Python, and it was tested on the Raspberry Pi using data from various simulations.

Further development of the project would include building a test board with the proposed sensors and testing the ability of the programme to perform the desired calculations in different scenarios.

Annex I and Annex II contain the developed code and the datasheets for the used components, respectively.

2. Theoretical framework.

2.1. Complex vector notation and reference frame.

A three-phase sinusoidal voltage system U_{abc} (Fig. 2.1) can be represented in an $\alpha\beta$ stationary reference frame by means of the Clarke transformation, also known as alpha-beta transformation (Fig. 2.2) [2].

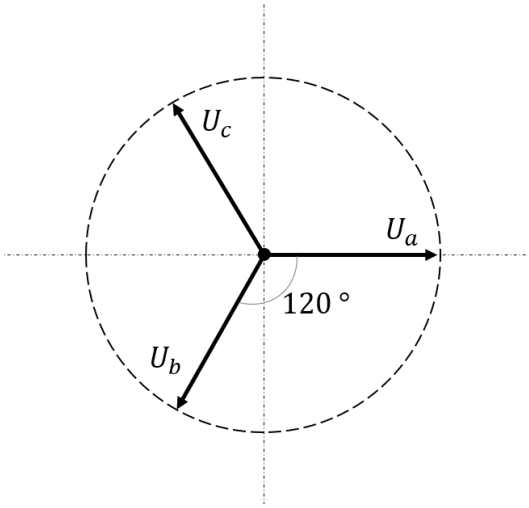

Fig. 2.1. Three-phase voltage system, in direct sequence.

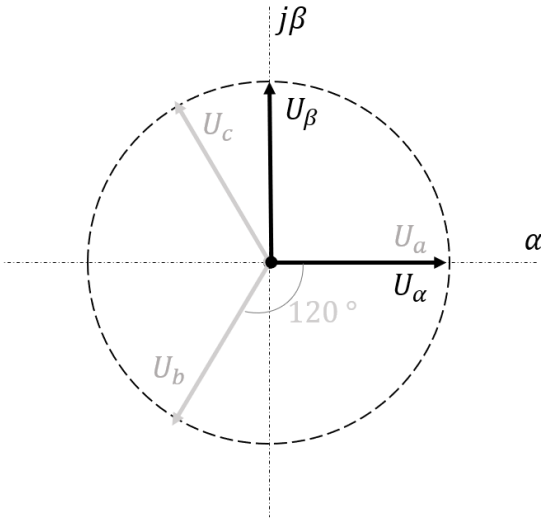

Fig. 2.2. Clarke transformation is applied, obtaining U_α and U_β.

Alpha is aligned with the original phase (a) and beta is shifted 90 degrees.

This transformation can be expressed as [3]:

$$U_\alpha = \frac{2 \cdot U_a}{3} - \frac{U_b}{3} - \frac{U_c}{3}$$

$$U_\beta = \frac{U_b}{\sqrt{3}} - \frac{U_c}{\sqrt{3}}$$

(2.1)

This set of variables $\alpha\beta$ can be transformed into a rotating dq reference frame via the Park transformation [4], as shown in Fig. 2.3.

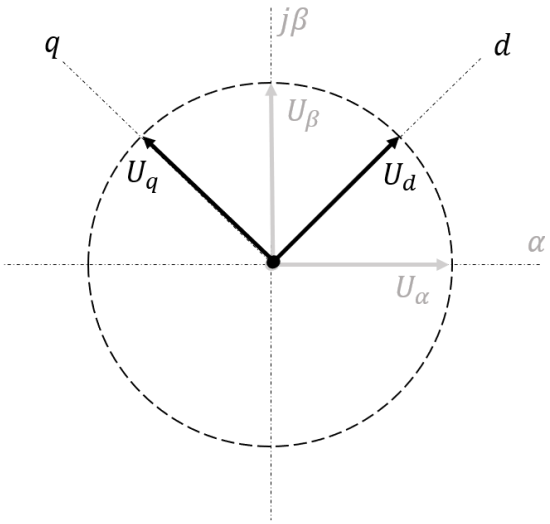

Fig. 2.3. Park transformation is applied, obtaining U_d and U_q.

This transformation is expressed as:

$$U_d = \cos(\theta) \cdot U_\alpha + \sin(\theta) \cdot U_\beta$$

$$U_q = -\sin(\theta) \cdot U_\alpha + \cos(\theta) \cdot U_\beta$$

(2.2)

2.2. Synchronisation methods.

Synchronisation methods, typically used to guarantee the stability in the connection of distributed energy resources and a grid, obtain the frequency, phase and the magnitude of an input signal.

They play a tremendously important role in protecting the components connected at the coupling point between the energy source and the grid, as differences in voltage, frequency, phase, or a combination of them could induce overcurrents, which compromise the safe and stable operation of the system.

Synchronisation methods can be open-loop or closed-loop. Open-loop methods base their operation on detecting the voltage zero crossing after applying a filtering stage to the input voltage, while closed-loop methods lock one characteristic of the input signal, such as the frequency. [1]

Open-loop systems offer advantages such as their fast dynamic response and unconditional stability, as they do not have a feedback loop. A commonly studied open-loop method is based on zero-crossing detection. A zero-crossing is the point where the sign of a mathematical function changes.

This method has been proven to be robust for systems with small amounts of distortion [5]. However, when the input signal is degraded, e.g. when harmonics are present or there occurs a power sag, the detection becomes insufficient, and so, the accuracy measured results decreases.

Open-loop methods require additional hardware to compensate for these effects, and for this reason they are not extensively applied.

Phase-locked loop (PLL) methods are closed-loop methods which have been used for synchronisation between distributed energy resources and the grid. They consist on a control system whose output signal has a phase related to that of the input signal.

By keeping the input and output phase locked, the input and output frequencies are kept equal to each other. This way, the input frequency can be tracked.

The basic elements that take part in a PLL are a variable frequency oscillator and a phase detector in a feedback loop. This loop continuously compares the input and output values, and ideally reaches a value that converges when said input and output are equal.

2.3. Usual components in PLLs.

A series or components that are commonly found in most PLL topologies will be introduced in this section. These include low-pass filters, notch filters and PI controllers.

2.3.1. Low-pass filter.

A low-pass filter is a filter which passes signals with a frequency lower than a selected cutoff frequency while attenuating those with a higher frequency (Fig. 2.4).

During this project, second order digital filters were used.

The continuous-time expression for a second order low-pass filter is:

$$LP(s) = \frac{\omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2} \tag{2.3}$$

Where:

ω_n is the cutoff frequency.

ζ is the damping ratio.

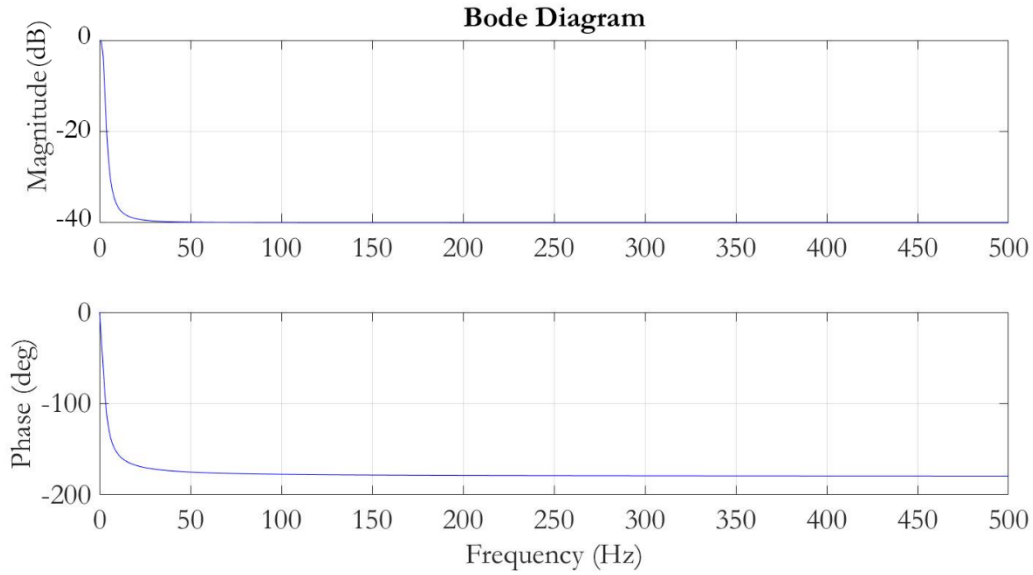

Fig. 2.4. Response of a second order low-pass filter with a cutoff frequency of 3 Hz.

2.3.2. Notch filter.

A band-stop filter passes most frequencies unaltered. However, those in a specific range are attenuated to very low levels. A notch-filter is a band-stop filter with a narrow stopband [6].

The continuous-time expression for the notch filters used in this project is:

$$NF(s) = \frac{s^2 + \omega_n^2}{s^2 + \zeta \cdot s + \omega_n^2} \tag{2.4}$$

Where:

ω_n is the notch frequency.

ζ is the damping ratio.

The Bode diagram of a notch filter is shown in Fig. 2.5:

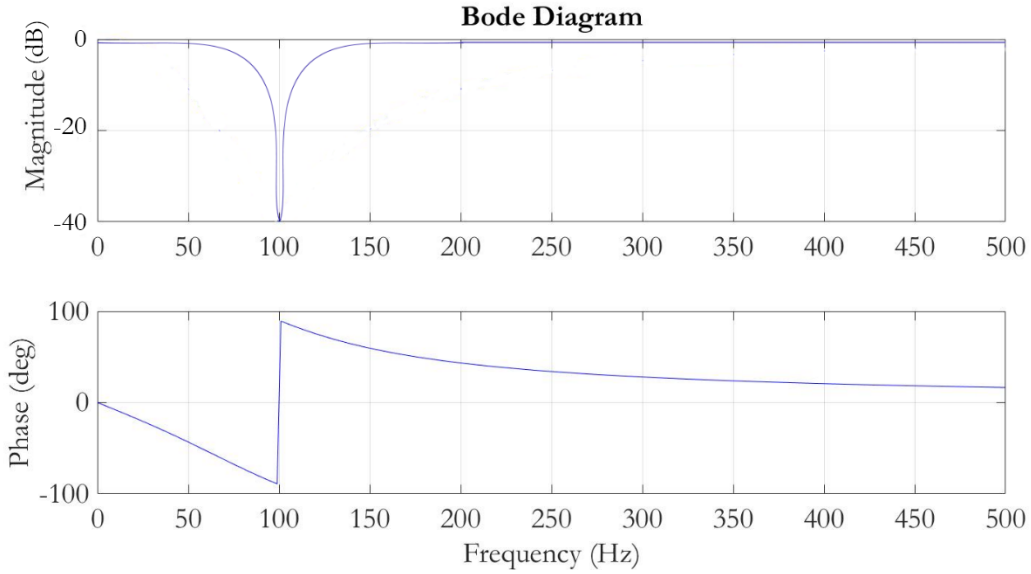

Fig. 2.5. Response of a notch filter with a notch frequency of 100 Hz.

2.3.3. PI controller.

A PI controller is a control loop feedback mechanism used when continuously modulated control is required.

This type of controller calculates an error value $e(t)$ as the difference between a desired setpoint and a measured process variable, and applies a correction based on proportional and integral terms.

PI controllers are part of the larger PID controller's family, which also perform corrections based on a derivative term.

The proportional term acts in relation of the error. This actuation is not sufficient for different situations, as a greater error in measurement does not necessarily mean the need for a greater response. For this reason, integral and/or derivative terms are added.

The integral term increases response in relation not only to the error but also its duration. A pure integral controller would bring the error to zero, but in a slow and rough manner, as it would keep increasing the action even with very low errors.

The expression for a PI controller is:

$$PI(s) = k_P \cdot \left(1 + \frac{k_i}{s}\right) \tag{2.5}$$

2.4. Different PLL topologies.

2.4.1. Synchronous Reference Frame PLL.

This type of PLL, also known as dq-PLL, is one of the most commonly used three-phase synchronisation methods.

This PLL obtains an estimation of the input signal's magnitude, frequency and phase in the manner shown in Fig. 2.6.

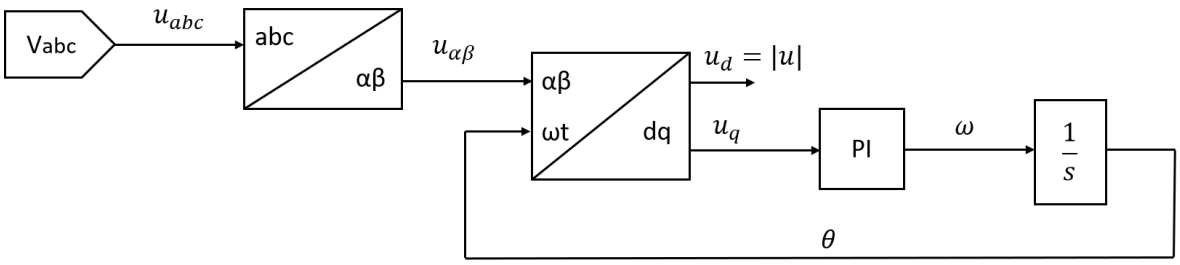

Fig. 2.6. Three-phase SRF-PLL.

The instantaneous phase angle is detected by synchronising the reference frame to the voltage vector. A PI controller sets the axis refence voltage V_d or V_q to zero. Then, the reference is locked to the voltage vector's phase angle. The frequency and magnitude are obtained as well, and their estimations depend on that of the phase angle.

The response from this type of PLL is fast and precise in the absence of harmonic distortions or unbalances, for a high bandwidth. In situations where these requirements are not met, the bandwidth on which the PLL operates optimally is reduced [2].

This can be mitigated by a reduction in the PI controller's bandwidth, or by filtering the induced harmonics at a pre-filter or filter on the loop stages.

The results obtained from an SRF-PLL are average information, not based on a single-phase. Thus, this method cannot be directly applied to a single-phase situation.

2.4.2. Complex power PLL.

This PQ-PLL, noted in this manner because of the P and Q components of the complex power, offers a solution for the problems observed during the start of operation under adverse conditions. They settle the oscillations caused by the presence of subharmonics and provide with a stable point of operation, maintaining synchronisation [8].

The block diagram of a PQ-PLL is shown in Fig. 2.7.

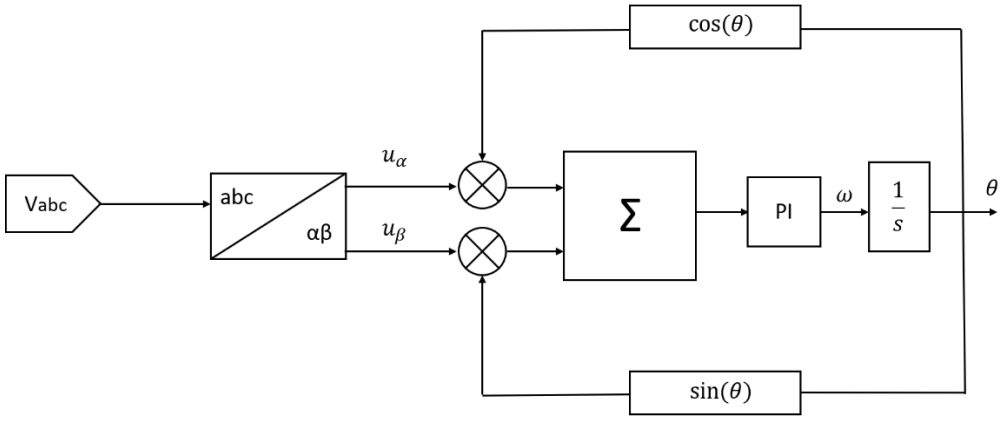

Fig. 2.7. Three-phase PQ-PLL.

2.4.3. Double synchronous frame PLL.

A double synchronous frame (DSF) PLL transforms both the positive and negative sequence components of the input voltage into a double synchronous frame (Fig. 2.8). The detection errors in a conventional SRF-PLL can be eliminated using this method.

As bandwidth reduction is no longer necessary, these PLL are suitable for operating in environments with large unbalances or heavy presence of harmonics.

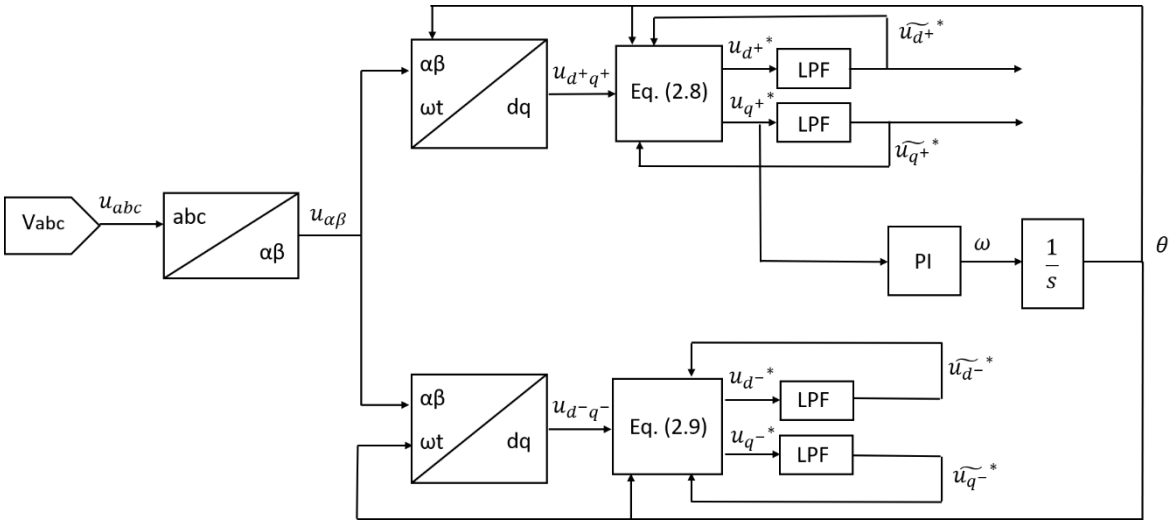

Fig. 2.8. Three-phase DSF-PLL.

Where:

$$\begin{bmatrix} u_{d^+} \\ u_{q^+} \end{bmatrix} = \begin{bmatrix} u_{d^+}{}^* \\ u_{q^+}{}^* \end{bmatrix} + \begin{bmatrix} \widetilde{u_{d^+}}{}^* \\ \widetilde{u_{q^+}}{}^* \end{bmatrix}$$

$$\begin{bmatrix} \widetilde{u_{d^+}}{}^* \\ \widetilde{u_{q^+}}{}^* \end{bmatrix} = \begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ -\sin(2\theta) & \cos(2\theta) \end{bmatrix} \begin{bmatrix} \overline{u_{d^-}} \\ \overline{u_{q^-}} \end{bmatrix} \tag{2.6}$$

and

$$\begin{bmatrix} u_{d^-} \\ u_{q^-} \end{bmatrix} = \begin{bmatrix} u_{d^-}{}^* \\ u_{q^-}{}^* \end{bmatrix} + \begin{bmatrix} \widetilde{u_{d^-}}{}^* \\ \widetilde{u_{q^-}}{}^* \end{bmatrix}$$

$$\begin{bmatrix} \widetilde{u_{d^-}}{}^* \\ \widetilde{u_{q^-}}{}^* \end{bmatrix} = \begin{bmatrix} \cos(-2\theta) & \sin(-2\theta) \\ -\sin(-2\theta) & \cos(-2\theta) \end{bmatrix} \begin{bmatrix} \overline{u_{d^+}} \\ \overline{u_{q^+}} \end{bmatrix} \tag{2.7}$$

2.4.4. Dual second order generalised integrator PLL.

A dual second order generalised integrator PLL (SOGI-PLL) filters the input signal and obtains a quadrature component for $V_{\alpha\beta}$ (Fig. 2.9).

This method is particularly useful for measuring the magnitude, frequency and phase of a single-phase signal, using an SRF-PLL.

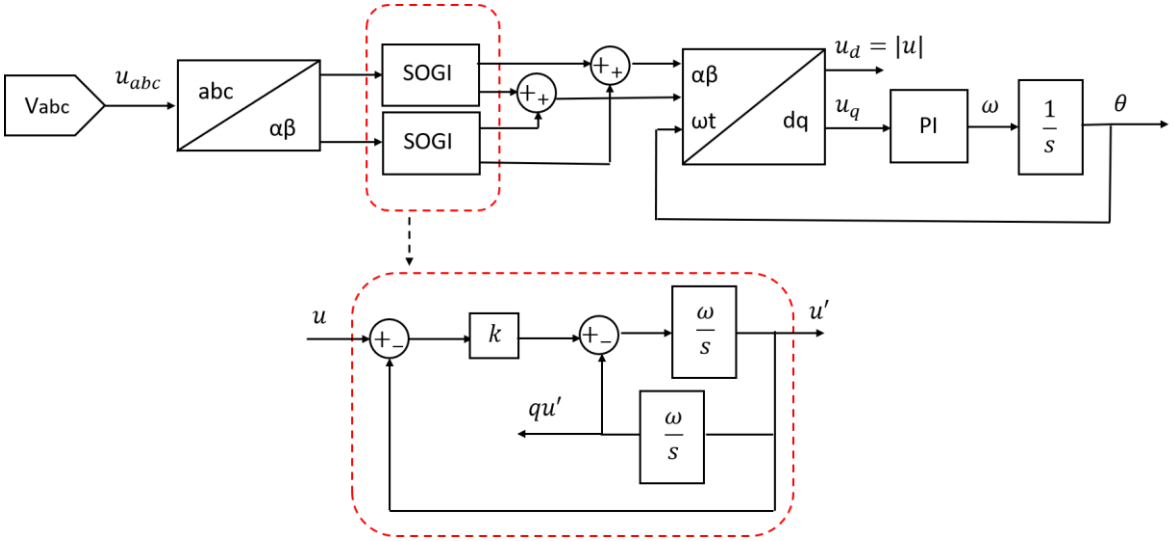

Fig. 2.9 Block diagram of a dual second order generalised integrator PLL.

3. Simulation and code design.

3.1. Spectral analysis of the input signal.

In order to decide the best approach for the measuring method that will be used in the Smart Meter, the input signal consisting on a three-phase voltage V_{abc} is simulated, its components being V_a, V_b and V_c. This voltage creates a set of currents I_{abc}, composed by I_a, I_b and I_c. All these parameters will be the inputs of the prototype after being appropriately adapted (Fig. 3.1).

For this purpose, a three-phase installation will be modelled in Simulink, a graphical programming environment, developed by MathWorks, for modelling, simulating and analysing multidomain dynamical systems. The main interface consists on a graphical block diagramming tool.

Apart from the simplicity of the block-based design, this programme offers tight integration with the rest of the MATLAB environment, and is able to generate code readable by the Raspberry Pi by using support packages available in its library.

A system can be modelled using the blocks already included in Simulink's libraries, or blocks created by the user. This last type of blocks is referred to as 'MATLAB Function' blocks, and usually contain one or various MATLAB functions.

The process followed during this project will be to transition from a model composed entirely by Simulink default bocks, to two final MATLAB Function blocks, one of them containing a three-phase PLL, and the other one, a single-phase PLL.

With the intention of comparing the results obtained in the simulation with the theoretical ones, an installation consisting of a three-phase 400 V (RMS value), 50 Hz source, a three-phase load (RL, 50 Ω and 1 mH per phase), and a single-phase load (50 Ω and 1 H) is modelled as shown in Fig. 3.1.

[V_{abc}] and [I_{abc}] are tags for the line voltages and line currents, respectively. These will be the inputs of the three-phase measuring subsystem.

V and *I* are the voltage between phase A and phase B and the current flowing through a load connected to those phases. *V* and *I* are the inputs to the single-phase subsystem.

Both the three-phase and the single-phase loads are modelled as R-L loads, due to them being the usual in most industrial and domestic situations. The values for both loads can be modified at any moment during the simulation stage, in order to verify that the results are consistent regardless of the analysed load.

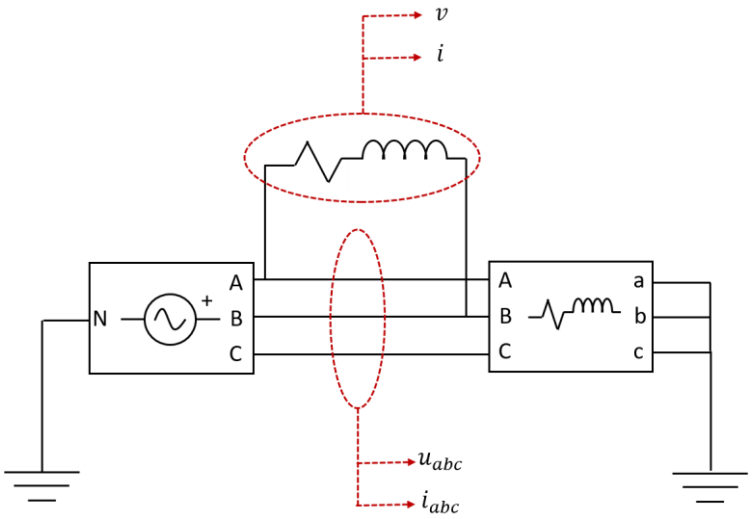

Fig. 3.1. Obtained measurements in the three-phase and single-phase installation.

The spectral analysis of V_{abc} will be carried out by creating a MATLAB function which will perform the Fourier Fast Transform using as an input said voltage V_{abc}. The code used in this function can be found in Annex I.

The results of this analysis indicate that in a three-phase installation, the most significant harmonic components are those located at -5· ωn and at 7· ωn, ωn being equal to 2π times the fundamental frequency of 50 Hz (Fig. 3.2).

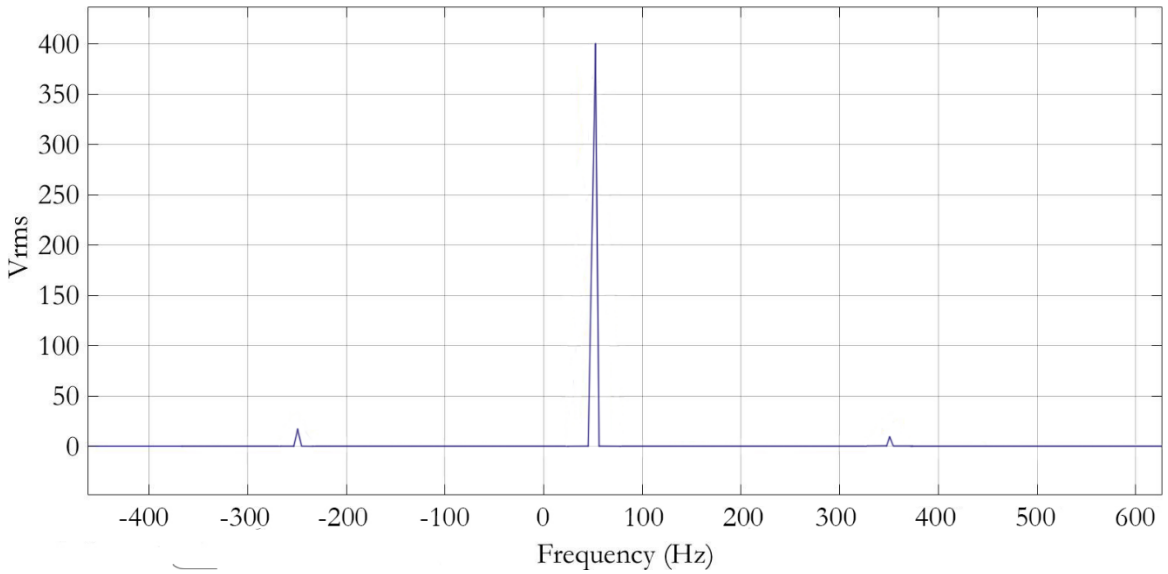

Fig. 3.2. Spectral analysis of the voltage input.

3.2. Simulation process in Simulink.

During this stage, the synchronous frame PLL was modelled using different approaches in Simulink. The goal was to obtain a final function which would perform the necessary calculations to obtain the magnitude, frequency, harmonic components, and power, starting from a basic design and successively improving it.

A series of parameters were used during this entire stage, and are listed in the tables below:

Peak three-phase voltage	Fifth harmonic component	Seventh harmonic component	Three-phase load	Single-phase load	Frequency
$400 \cdot \sqrt{2}$ V	5%	2.5%	50 Ω+1 mH	50 Ω+1 H	50 Hz

Table 3.1. Parameters of the voltage source.

Sample time	K_p	K_i	Damping ratio (low-pass filter)	Damping ratio (notch filter)	Bandwidth (notch filter)
10^{-4} s	10	600	$\sqrt{2}/2$	0.85	3 Hz

Table 3.2. Parameters for the function blocks.

3.2.1. Three-phase PLL using Simulink default blocks.

The first approach during the modelling stage was to implement a three-phase PLL using exclusively Simulink default blocks, in order to better understand its performance as a system, and to study the subsequent iterations performed on the model (Fig. 3.3).

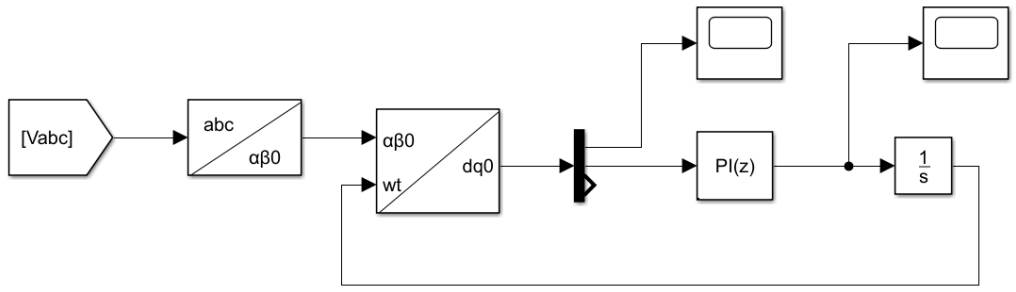

Fig. 3.3. Implementation of the three-phase PLL exclusively using Simulink blocks.

The results for the estimated voltage and frequency are shown in Fig. 3.4 and Fig. 3.5, respectively.

These results are coherent with the previously stated conditions in the three-phase voltage source (a peak voltage of $400 \cdot \sqrt{2}$ V, a frequency of 50 Hz, and without harmonic distortion), as the voltage measurement is 565.7 V, which equals $400 \cdot \sqrt{2}$ V, and the frequency is around 314 rad·s^{-1}, which is equal to the fundamental frequency of 50 Hz times 2π.

However, both measurements exhibit an important harmonic content. The goal during the successive implementations of the PLL will be to reduce such distortion, with the intention of obtaining a measurement resembling the input values as closely as possible.

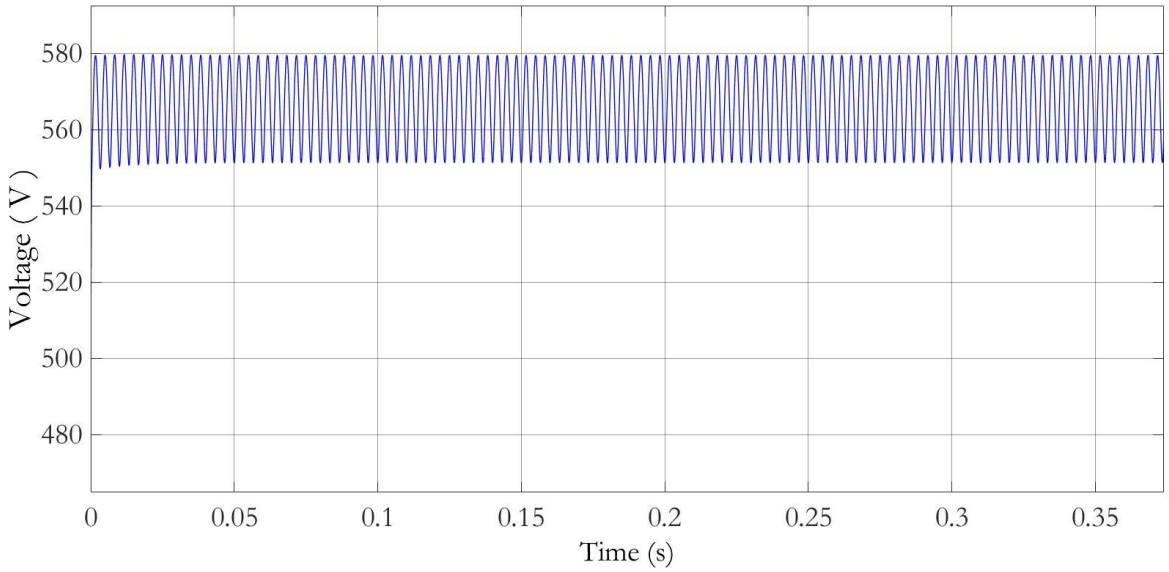

Fig. 3.4. Peak magnitude of the estimated voltage. PLL using Simulink default blocks.

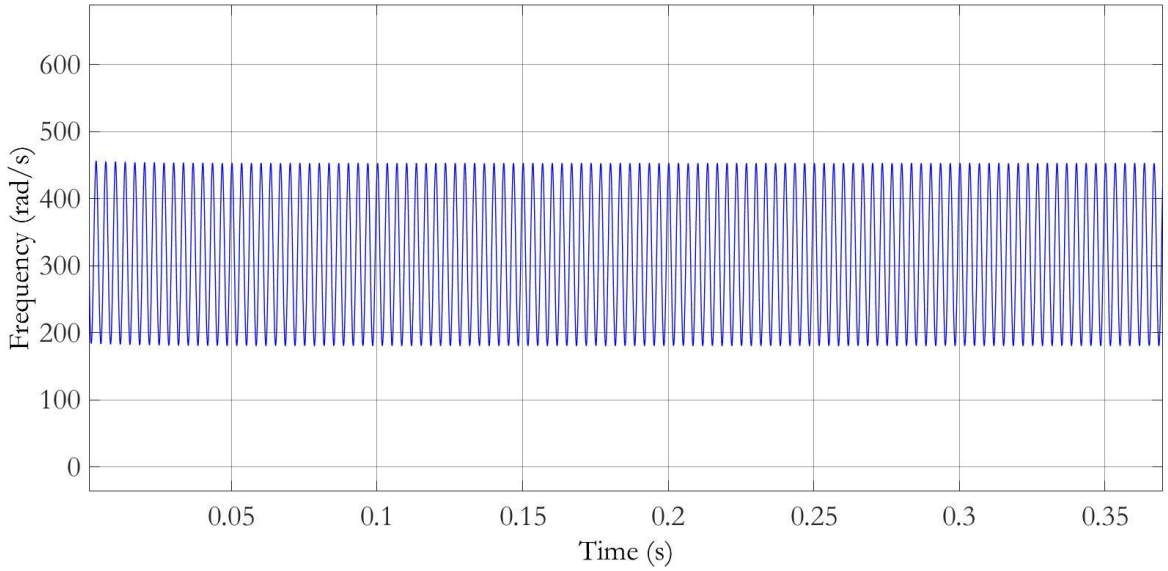

Fig. 3.5. Estimated frequency. PLL using Simulink default blocks.

At this stage, the system is operating in continuous time, as the default blocks included in Simulink operate in such conditions.

3.2.2. Three-phase PLL modelled with function blocks.

The first logical advance to the model built with Simulink blocks, was to develop a model composed by MATLAB Function blocks.

Each one of these blocks contains a MATLAB function that replicates the behaviour of its equivalent Simulink block.

At this step, external parameters for some functions, such as the time step T_S (in this case it is the same as the prototype's sample rate, 100 μs) for the PI regulator and the integrator were not included inside the blocks. Instead, those values were fed into the Function blocks as external constants.

For the functions which depend on their own previous value, the delay block z^{-1} was used. This was coded inside each function block thereafter.

The diagram for this PLL is shown in Fig. 3.6, and the results for the voltage's magnitude and frequency, in Fig. 3.7 and Fig. 3.8 respectively.

It can be noticed that the voltage's magnitude signal exhibits a significant harmonic content. This distortion does not allow for a precise estimation of the voltage, so a Low-Pass filter was implemented on the loop later on, in order to discard the harmonics during the rotation of the input signal.

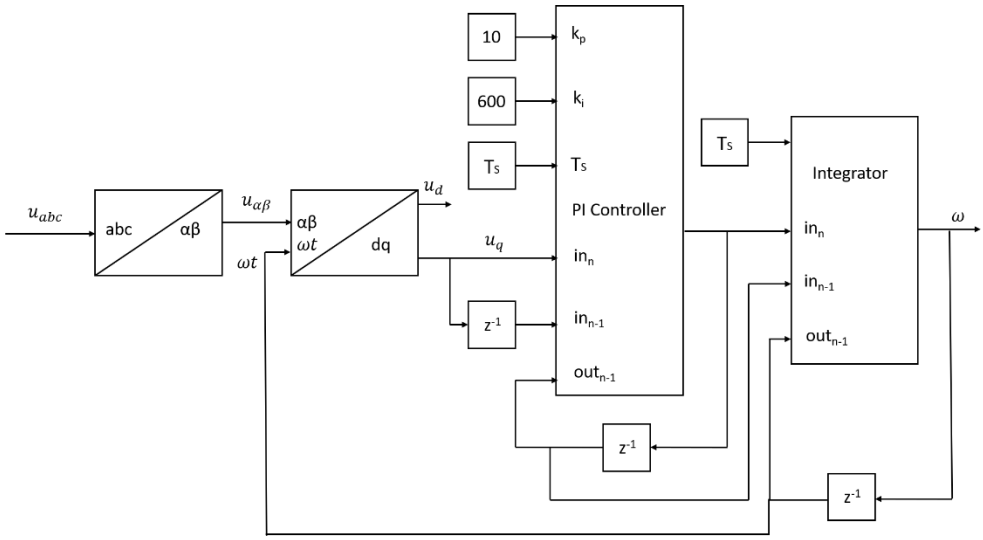

Fig. 3.6. Diagram for the three-phase PLL modelled with independent blocks.

At this stage, the model is already operating in discrete time. This transformation from the previous model, which operates in continuous time, is achieved by applying the bilinear transform, also known as Tustin's Method, explained in section 3.2.2.3.

The particular blocks on which Tustin's Method was applied, namely the PI controller and the integrator, and later on the various filters, use a sample rate of $T_s = 10^{-4}\ s$.

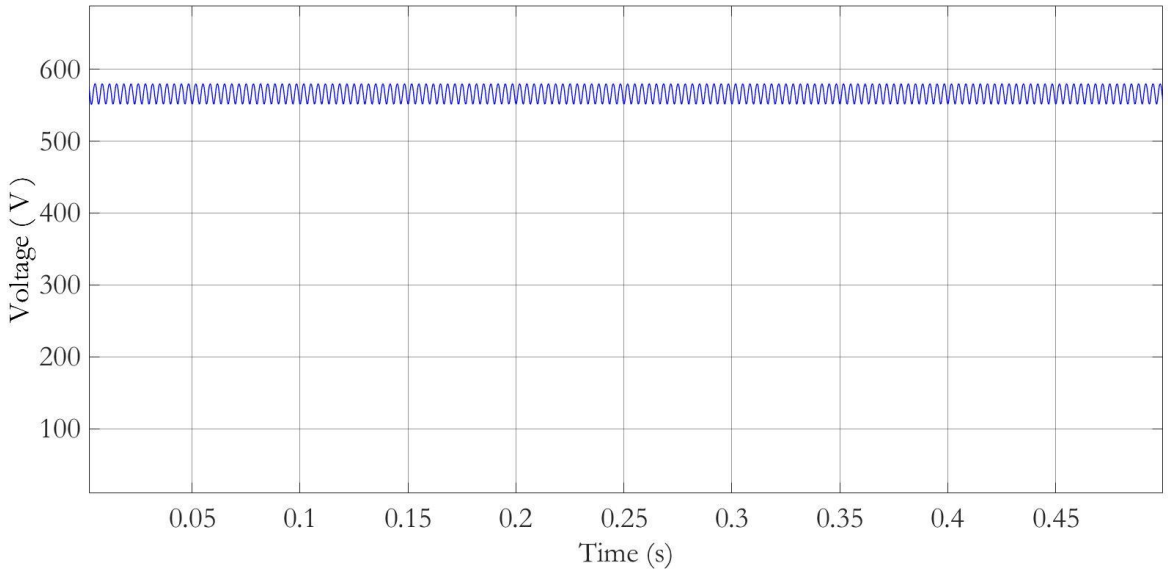

Fig. 3.7. Magnitude of the estimated voltage. PLL using Simulink default blocks.

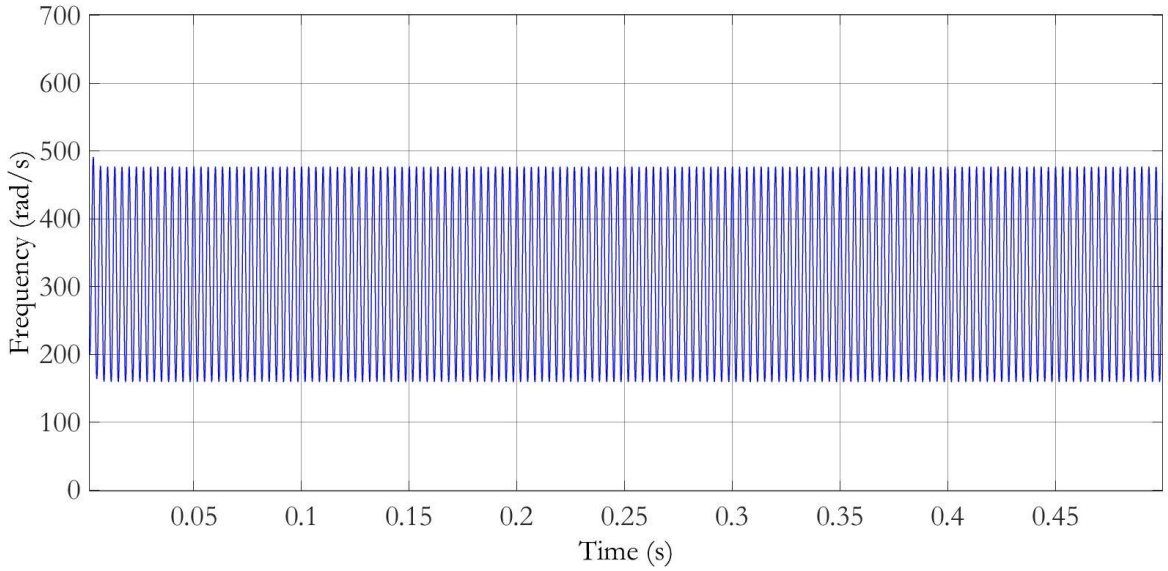

Fig. 3.8. Estimated frequency. PLL modelled with function blocks.

These figures showed a considerable distortion from the expected values, so a low-pass filter was added before the scope. However, this was only a temporary solution, as the harmonics causing such distortion were filtered in later stages.

The filtered results are shown in Fig. 3.9 and Fig. 3.10.

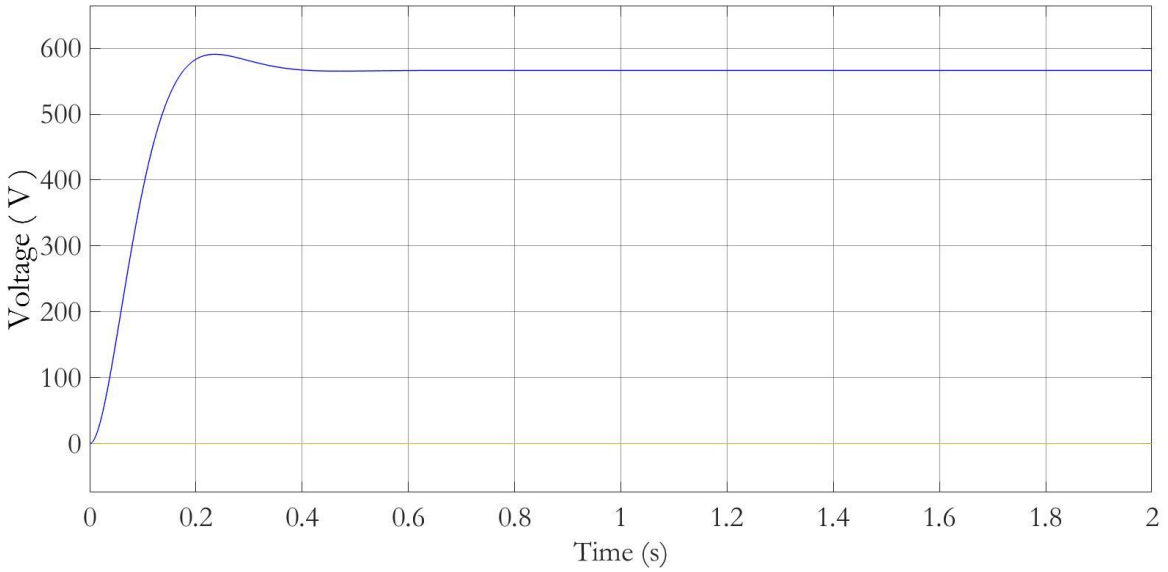

Fig. 3.9. Filtered estimated peak voltage magnitude.

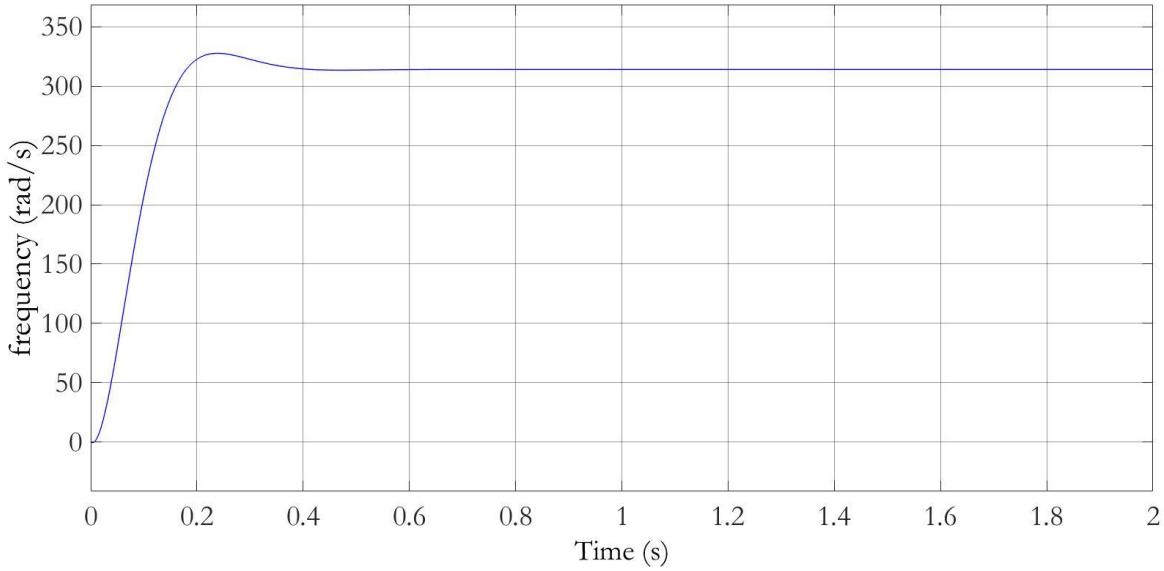

Fig. 3.10. Filtered estimated peak frequency magnitude.

3.2.2.1. Function block a-b-c to alpha-beta.

This function block contains the code that performs the Clarke transformation of the input variables, which are the components V_a, V_b and V_c. Its outputs are the $\alpha\beta$ components of the input voltage, using equation (2.1).

3.2.2.2. Function block alpha-beta-phase to d-q.

This function block performs the Park transformation and generates the direct component d and its quadrature q from the previously obtained *aβ* components and the estimated phase, using equation (2.2).

3.2.2.3. Bilinear transform. Tustin's method.

While Simulink's default blocks operate as continuous-time systems, the code developed on Simulink and deployed onto the Raspberry Pi board at a later stage works in the discrete-time domain.

To transform the functions from one domain to another, the bilinear transform, also known as Tustin's method, is used [7].

For a discrete-time transfer function *H=H(z),* its Tustin transform at frequency ω_0>0 is defined by:

$$G(s) = Tu_{\omega_0}[H](s) = H\left(\frac{\omega_0 + s}{\omega_0 - s}\right) \tag{3.1}$$

The inverse of Tu_{ω_0} takes a continuous-time system and produces the discrete-time system as:

$$H(z) = {Tu_{\omega_0}}^{-1}[G](s) = G\left(\omega_0 \frac{z-1}{z+1}\right) \tag{3.2}$$

For example, applying the Tustin transform at a frequency ω_0=1 to $H(z) = 1/z$ yields $G(s) = \frac{1-s}{1+s}$.

3.2.2.4. PI controller.

This block is the one on which Tustin's method was first applied during this project.

The continuous-time expression for a PI controller is:

$$PI(s) = k_p \cdot \left(1 + \frac{k_i}{s}\right) \tag{3.3}$$

Where:

k_p is the proportional constant, in this case, 10 (Table 3.2).

k_i is the integral constant, in this case, 600 (Table 3.2).

These values for k_p and k_i are a compromise for precision and response speed.

Applying Tustin's method, the discrete-time expression for a PI controller is obtained:

$$out_n = \frac{\left(2 \cdot k_p + k_p \cdot T_S \cdot k_i\right) \cdot in_n + \left(k_p \cdot T_S \cdot k_i - 2 \cdot k_p\right) \cdot in_{n-1} + 2 \cdot out_{n-1}}{2} \tag{3.4}$$

Where:

k_p is the proportional constant.

k_i is the integral constant.

T_S is the time step constant.

out_n is the output at a stage n.

out_{n-1} is the output at a stage $n-1$.

in_n is the input at a stage n.

in_{n-1} is the input at a stage $n-1$.

3.2.2.5. Integrator.

For the implementation of this function, Tustin's method was also needed.

The continuous-time expression for an integrator is:

$$H(s) = \frac{1}{s} \tag{3.5}$$

Applying Tustin's method, the discrete-time expression is:

$$out_n = \frac{T_S}{s} \cdot (in_n + in_{n-1}) + out_{n-1} \tag{3.6}$$

Where:

T_S is the time step constant.

out_n is the output at a stage n.

out_{n-1} is the output at a stage $n-1$.

in_n is the input at a stage n.

in_{n-1} is the input at a stage $n-1$.

3.2.3. Implementation of the harmonics' estimation.

According to the frequency analysis performed on the voltage from the model source, the main harmonic components that might cause distortion in the obtained measurements are the ones located at -5· ωn and at 7· ωn, ωn being equal to 2π times the fundamental frequency of 50 Hz.

Thus, a loop for each one of these harmonic components must be implemented. The first approach was to directly apply the Park transformation to the *aβ* components obtained from the Clarke transformation, using a Gain block with a value of -5 and another with a value of 7 before each 'alpha-beta-omega to d-q' block.

The outputs u_d and u_q were the real and the imaginary part of the harmonic signals, and from them, their module could be calculated.

The results at this stage showed an excessive harmonic content in the measurements for the fifth and seventh harmonic, so it was decided to use low-pass filters to reduce the distortion at the output of the PI controller and at outputs u_d and u_q (Fig. 3.11).

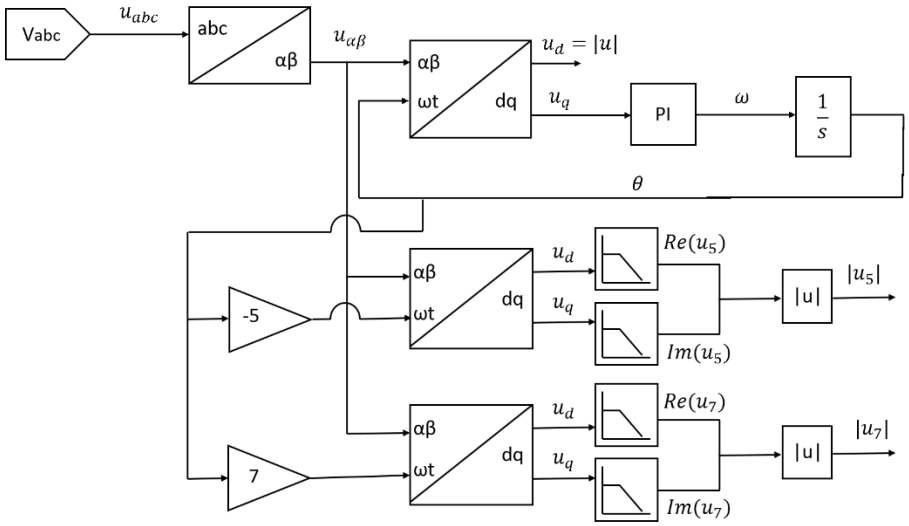

Fig. 3.11. Diagram for the three-phase PLL with harmonic loops and low-pass filters.

The continuous-time expression for the low-pass filter is that of a second order one:

$$H(s) = \frac{\omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2} \tag{3.7}$$

Where:

ω_n is the natural frequency.

ζ is the damping ratio (Table 3.2).

Applying Tustin's method, the following expression is obtained for the discrete-time domain:

$$out_n = \frac{\omega_n^2 \cdot T_S^2 \cdot (in_n + 2 \cdot in_{n-1} + in_{n-2}) - (2 \cdot \omega_n^2 \cdot T_S^2 - 8) \cdot out_{n-1} - (4 - 4 \cdot \zeta \cdot \omega_n \cdot T_S + \omega_n^2 \cdot T_S^2) \cdot out_{n-2}}{4 + 4 \cdot \zeta \cdot \omega_n \cdot T_S + \omega_n^2 \cdot T_S^2} \quad (3.8)$$

Where:

ω_n is the natural frequency.

T_S is the time step constant.

out_n is the output at a stage n.

out_{n-1} is the output at a stage $n-1$.

out_{n-2} is the output at a stage $n-2$.

in_n is the input at a stage n.

in_{n-1} is the input at a stage $n-1$.

in_{n-2} is the input at a stage $n-2$.

The original loop used for estimating the fundamental magnitude and frequency was left unaltered, and a new Park transformation stage was modelled for the fundamental component, on which the low-pass filters were also used.

The results for the magnitudes of the fifth and seventh harmonics obtained with the filtered PLL are shown in Fig. 3.12 and Fig. 3.13 respectively.

The obtained results are coherent with the values selected for the fifth and seventh harmonics in the simulated voltage source (5% and 2.5% of the fundamental voltage, respectively, equivalent to 28.28 V and 14.14 V).

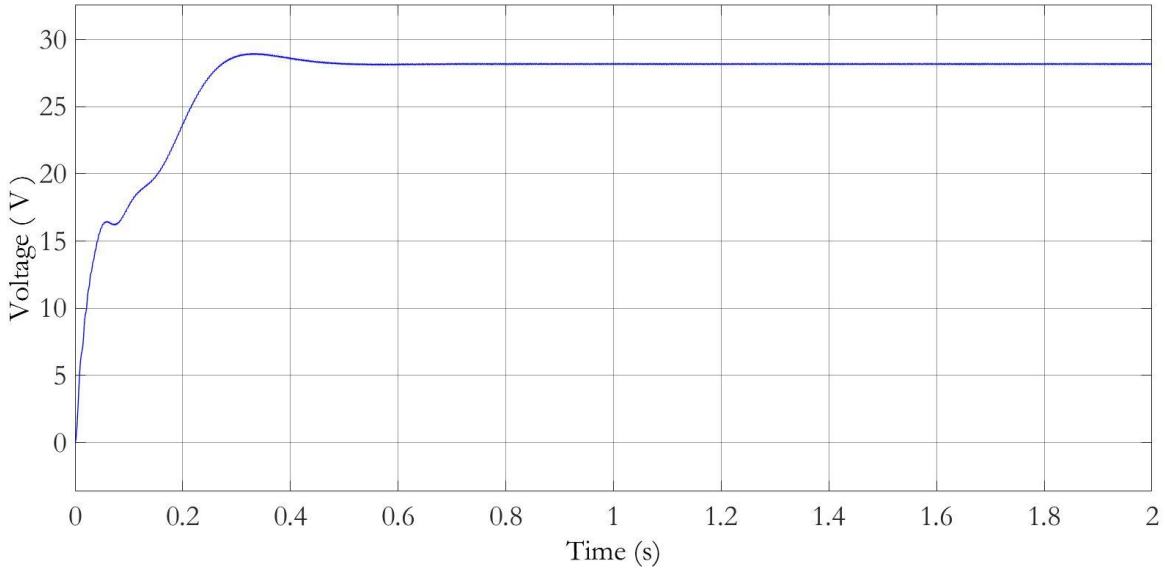

Fig. 3.12. Magnitude estimation for the fifth harmonic of the input voltage.

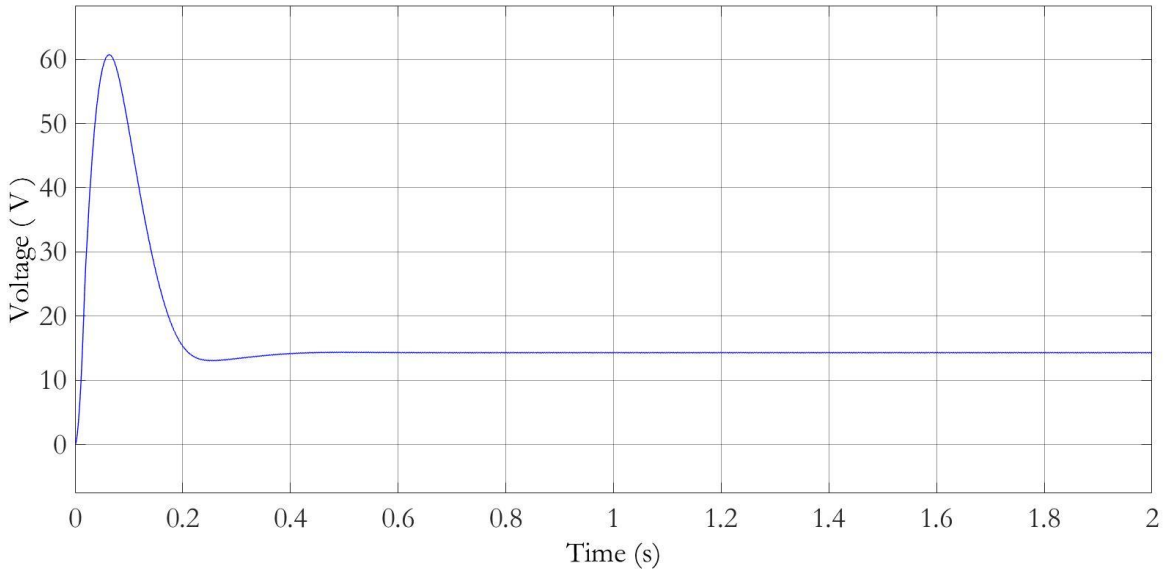

Fig. 3.13. Magnitude estimation for the seventh harmonic of the input voltage.

3.2.4. Three-phase PLL modelled on a single block.

At this stage, the PLL was modelled on a single block containing a function with multiple inputs and outputs. This allowed for the possibility of including the necessary constants as part of the code.

For the functions whose output value depend not only on their input value at an instant, but also on their input and output values at previous instants, a type of variable known as persistent variables is needed.

These variables are declared at the beginning of the code with an initial value. For increased response speed, a value in the proximity of the expected measurement might be selected. This value was chosen to be zero for all the persistent variables in this project, in order for it to adapt to multiple applications.

After each cycle is completed, the value for a persistent variable gets updated with the value of that variable at the previous instant, e.g. out_{n-1} would take the value of out_n, and out_n would take the output value at that instant.

The results obtained for the fundamental, the fifth and the seventh harmonic components of the voltage signal are given in Fig. 3.14, Fig. 3.15 and Fig. 3.16.

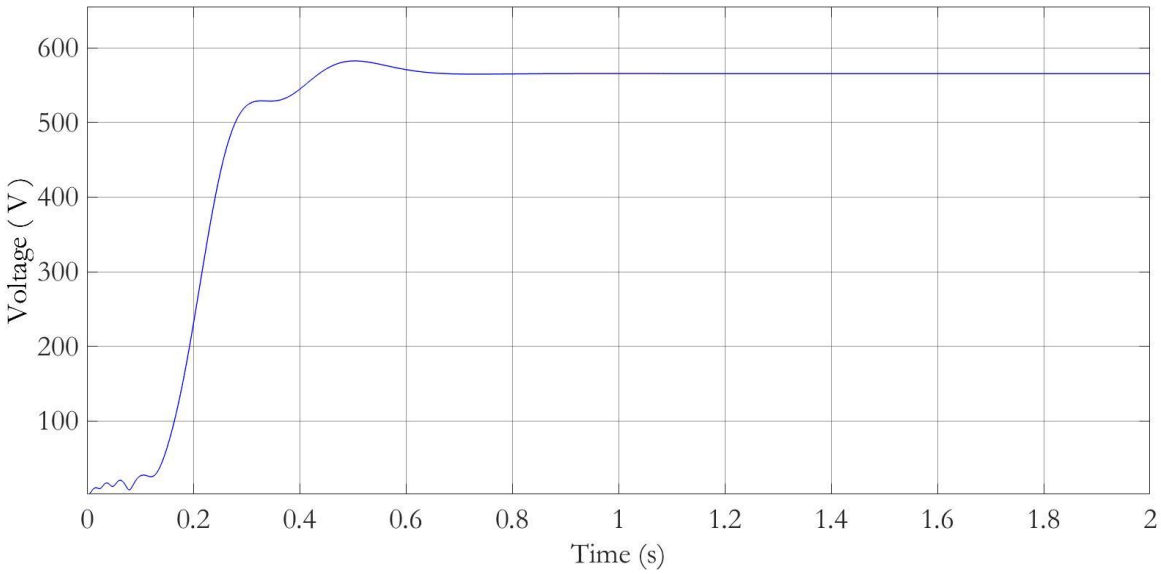

Fig. 3.14. Magnitude estimation for the fundamental component of the voltage.

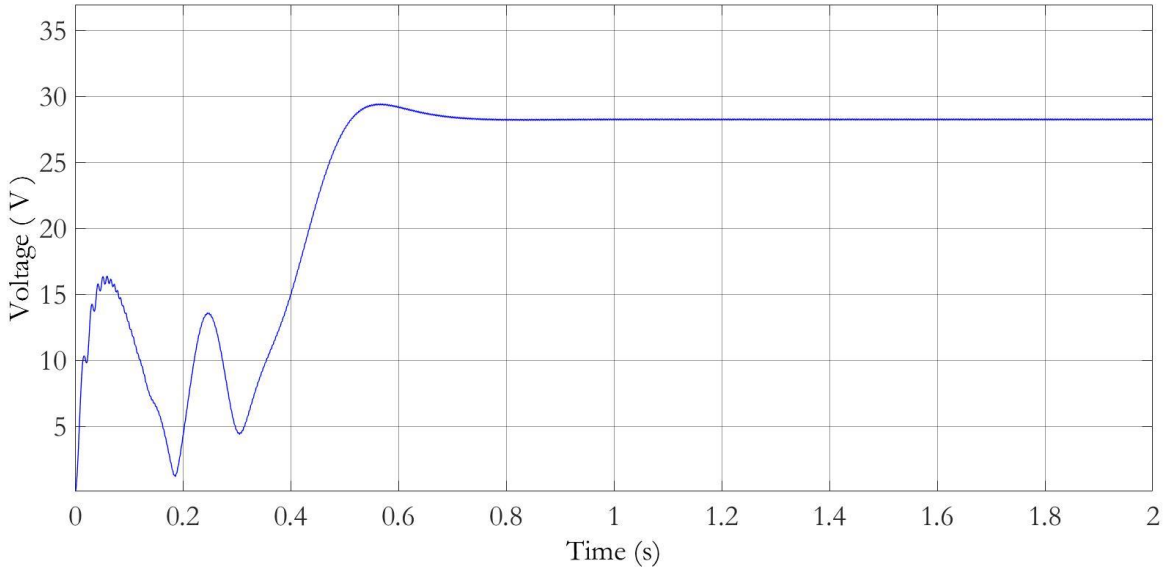

Fig. 3.15. Magnitude estimation for the fifth harmonic.

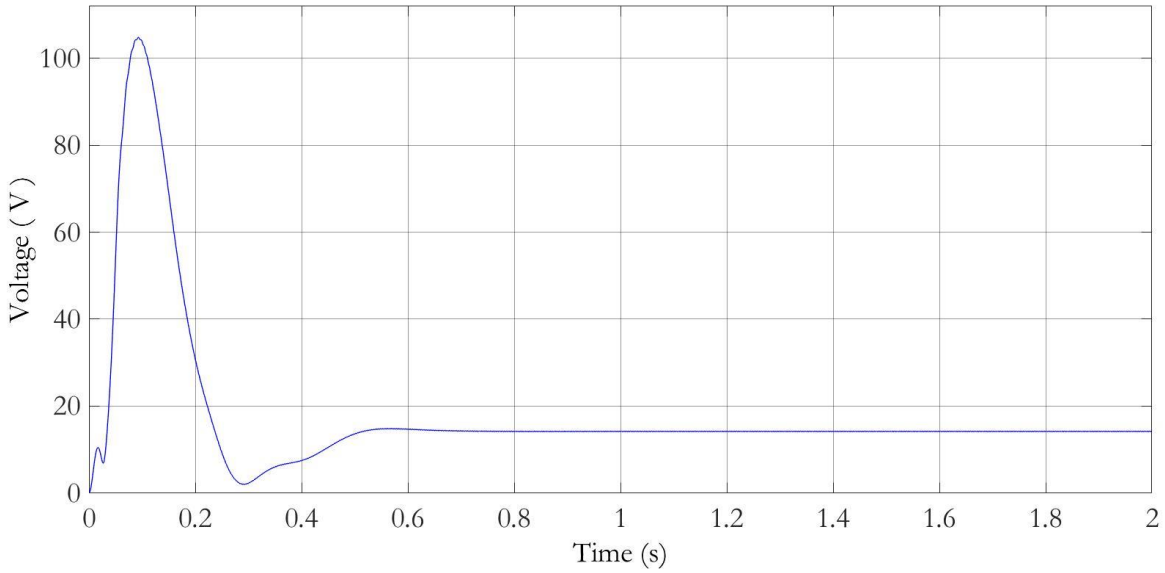

Fig.3.16. Magnitude estimation for the seventh harmonic.

As shown in Fig. 3.17, the estimated frequency, although centred in 314 rad/s, which would be the precise value, still contained a great harmonic content, which caused significant distortion in the measured results.

For this reason, a series of filters were added in the next version of the PLL.

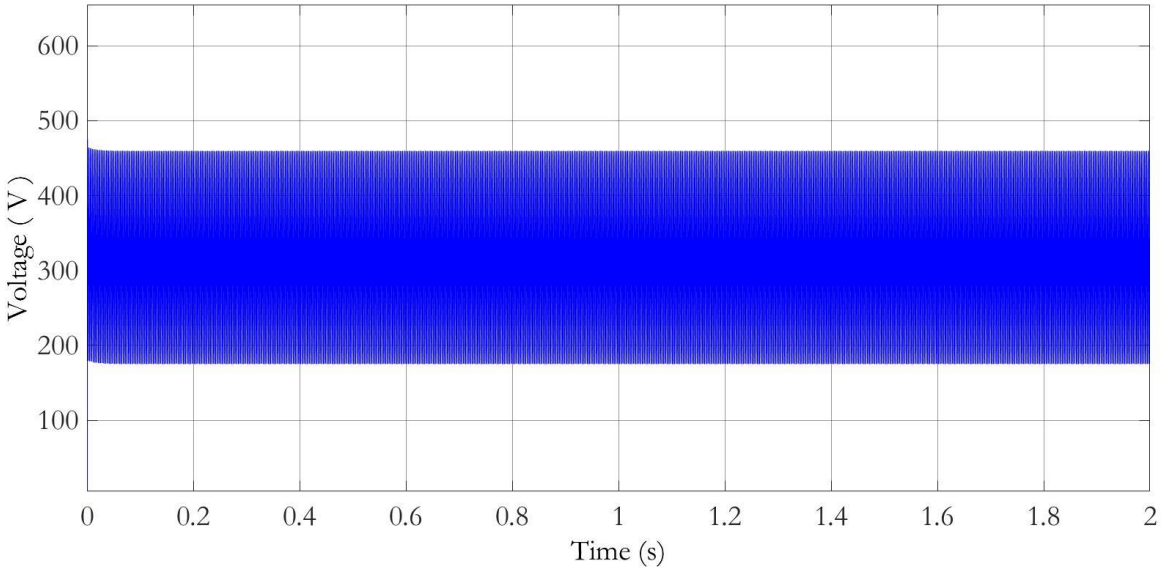

Fig. 3.17. Estimated frequency of the input voltage.

3.2.5. Implementation of notch filters.

During this stage, notch filters were added in the sections of the code that turned out to be the source of the distortion.

The most accurate results were obtained when the notch filter was applied to the output q of the alpha-beta-phase to d-q section. The filtered output was then used as the input of the PI controller, as done previously (Fig. 3.18).

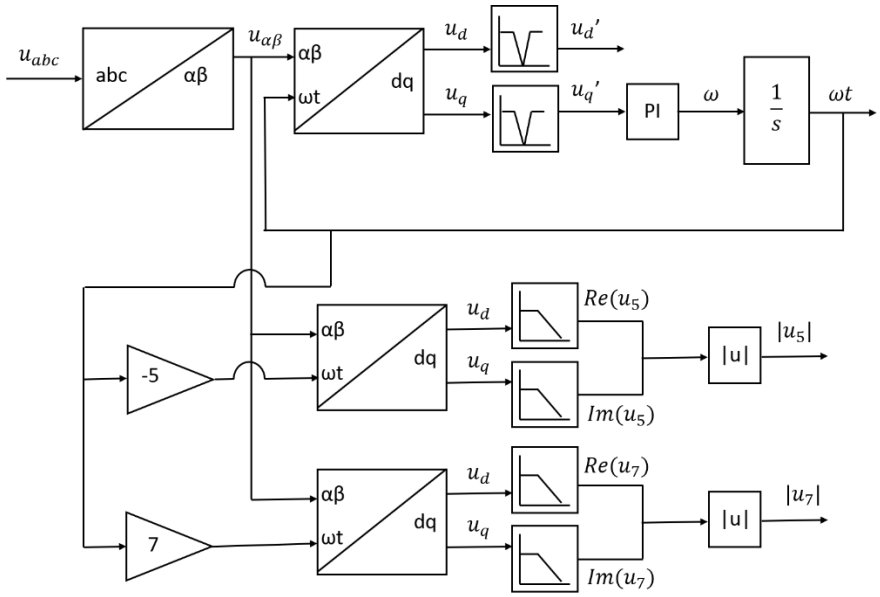

Fig. 3.18. Three-phase PLL with harmonic loops and notch filters.

A frequency analysis at this point prior to the implementation of the notch filters showed that the sixth and twelfth harmonics were present (Fig. 3.19). Thus, the notch frequencies for the filters were established as six times the fundamental, and twelve times the fundamental, respectively.

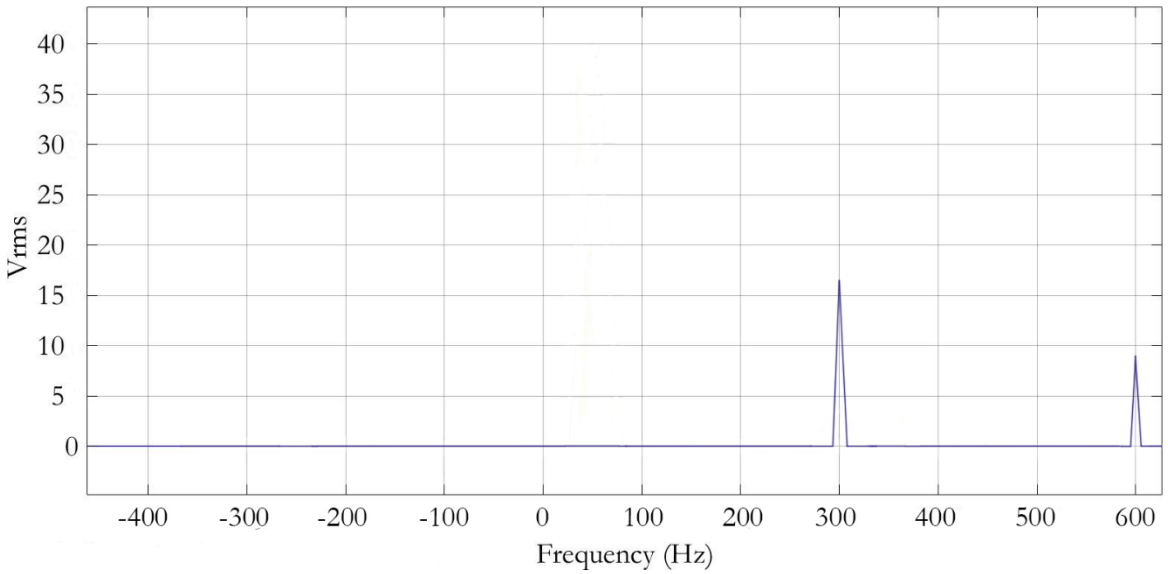

Fig. 3.19. Spectral analysis of u_q before the implementation of the notch filters.

The continuous-time expression for the notch filters used in this project is:

$$NF(s) = \frac{s^2 + \omega_n^2}{s^2 + \zeta \cdot s + \omega_n^2} \tag{3.9}$$

Where:

ω_n is the notch frequency.

ζ is the damping ratio.

The following expression is obtained for the discrete-time domain:

$$NF_{\omega_n}(z) = b \cdot \frac{1 - 2 \cdot \cos(\omega_n) \cdot z^{-1} + z^{-2}}{1 - 2 \cdot \cos(\omega_n) \cdot z^{-1} + z^{-2} + (2 \cdot b - 1) \cdot z^{-2}} \tag{3.10}$$

Where:

$$b = \frac{1}{1 + \beta} \tag{3.11}$$

$$\beta = \frac{\sqrt{1 - A_{-3\,dB}^2}}{A_{-3\,dB}} \cdot \tan\left(\frac{BW}{2}\right) \tag{3.12}$$

$A_{-3\,dB}$ is the notch filter magnitude at -3 dB, equal to $10^{-3/20} = 2 \cdot \sqrt{2}$.

$$BW = \frac{\Delta\omega}{f_s/2} \tag{3.13}$$

BW is the notch filter's bandwidth.

$\Delta\omega$ is the notch filter's bandwidth, in rad/s.

Thus, the expression that was implemented in the code was:

$$out_n = b \cdot (in_n + in_{n-2}) + 2 \cdot b \cdot \cos(\omega_n) \cdot (out_{n-1} - in_{n-1}) - (2 \cdot b - 1) \cdot out_{n-2} \tag{3.14}$$

Where:

ω_n is the notch frequency.

out_n is the output at a stage n.

out_{n-1} is the output at a stage $n-1$.

out_{n-2} is the output at a stage $n-2$.

in_n is the input at a stage n.

in_{n-1} is the input at a stage $n-1$.

in_{n-2} is the input at a stage $n-2$.

b is a coefficient related to the filter's quality factor.

By this method, the harmonics that appeared during the vector rotation where removed, and more precise results were obtained.

The results obtained for the fundamental, the fifth and the seventh harmonic components of the voltage signal are given in Fig. 3.20, Fig. 3.21 and Fig. 3.22.

No significant difference was found between these results, and the ones obtained in the previous iteration, the PLL built as one block without notch filters. This similarity occurs due to the fact that the PI controller (with $k_p = 10$ and $k_i = 600$) acts as a filter itself. The measurements most directly influenced by it, in this case the fundamental, the fifth and the seventh harmonic components of the voltage, are already filtered.

However, the measurement for the estimated frequency proved to be greatly improved from the previous stage, as shown in Fig. 3.23, where it can be noticed that, after the initial tenth of a second, the estimation reaches the expected value.

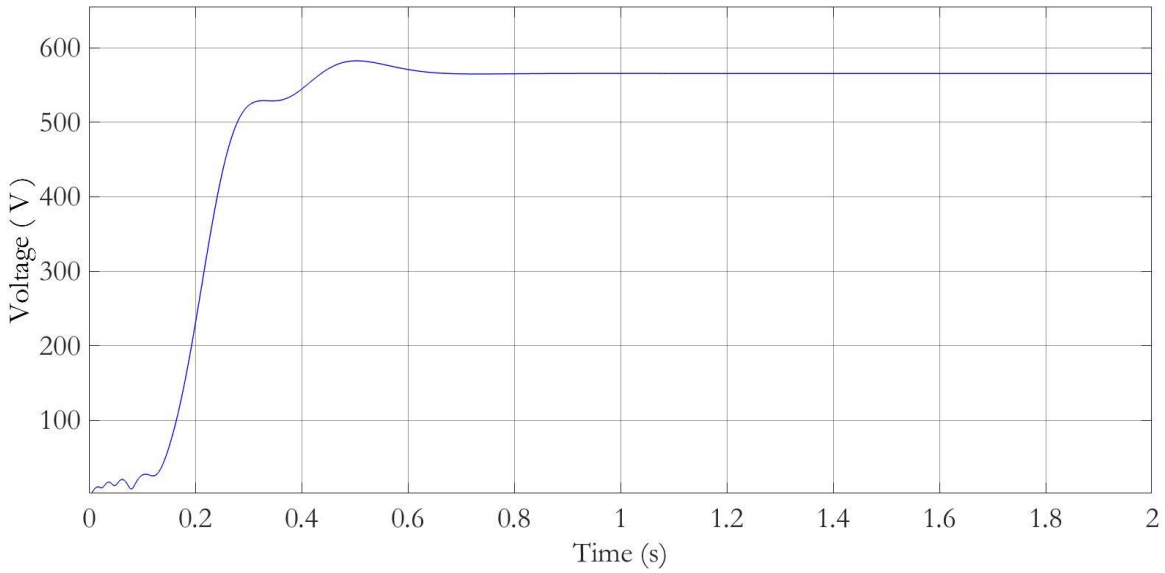

Fig. 3.20. Magnitude estimation for the fundamental component of the voltage.

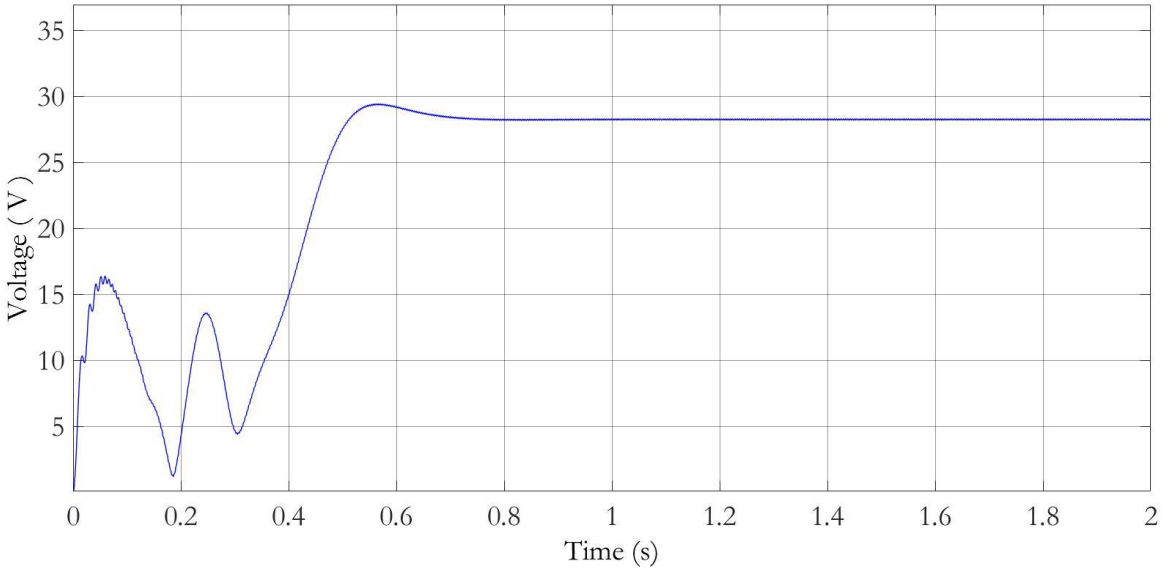

Fig. 3.21. Magnitude estimation for the fifth harmonic.

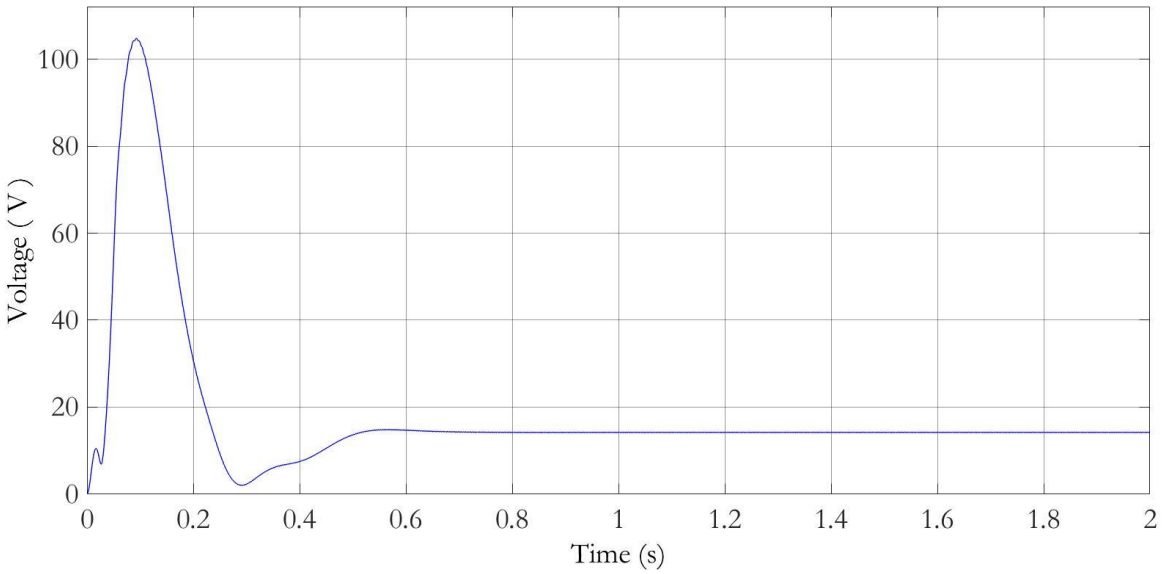

Fig. 3.22. Magnitude estimation for the seventh harmonic.

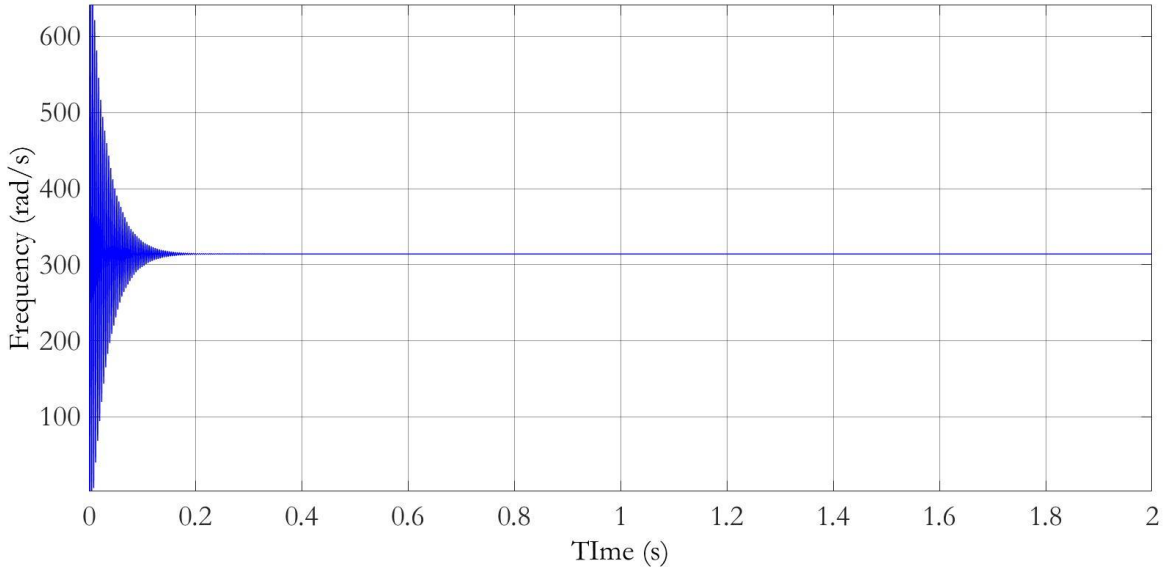

Fig. 3.23. Estimated frequency of the input voltage.

3.2.6. Implementation of the current loop.

This loop measures the magnitude of the current's fundamental component, as well as the magnitudes of its fifth and seventh harmonic components.

As the method applied for measuring the different variables of the input voltage was proven to be solid, the same approach was followed for the current.

Variables with memory were created for the I_a, I_b and I_c components of the input current, and for the variables used in the current loop.

The same steps that were previously applied for the V_a, V_b and V_c components were followed for I_a, I_b and I_c. In summary, a PLL was created for the current loop, operating in parallel to the one for the voltage loop.

The results obtained for the fundamental, the fifth and the seventh harmonic components of the voltage signal are given in Fig. 3.24, Fig. 3.25 and Fig. 3.26.

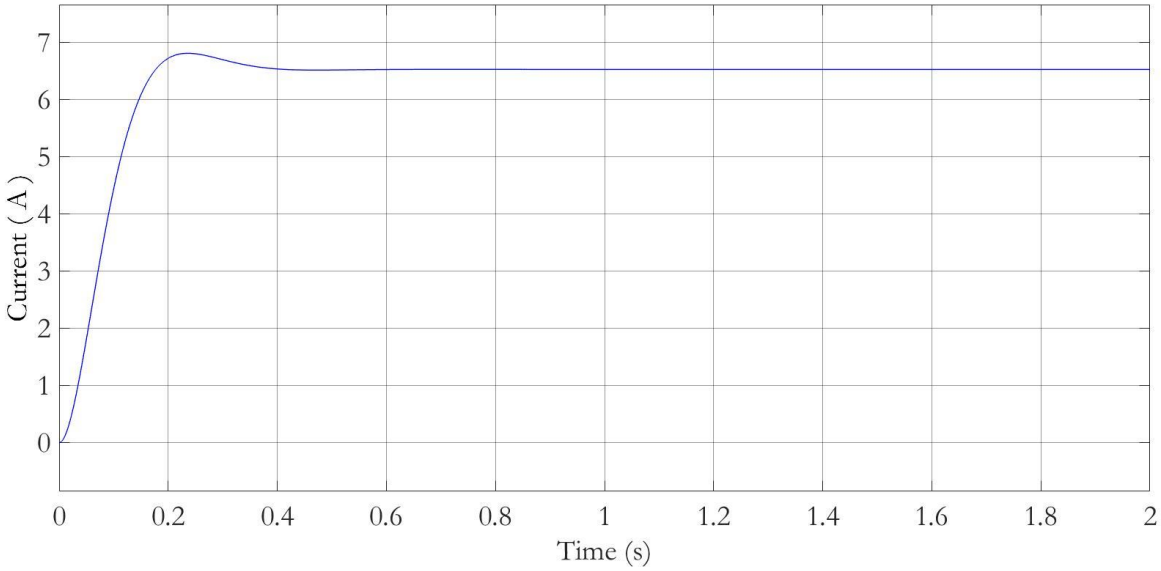

Fig. 3.24. Estimated magnitude of the current's fundamental component.

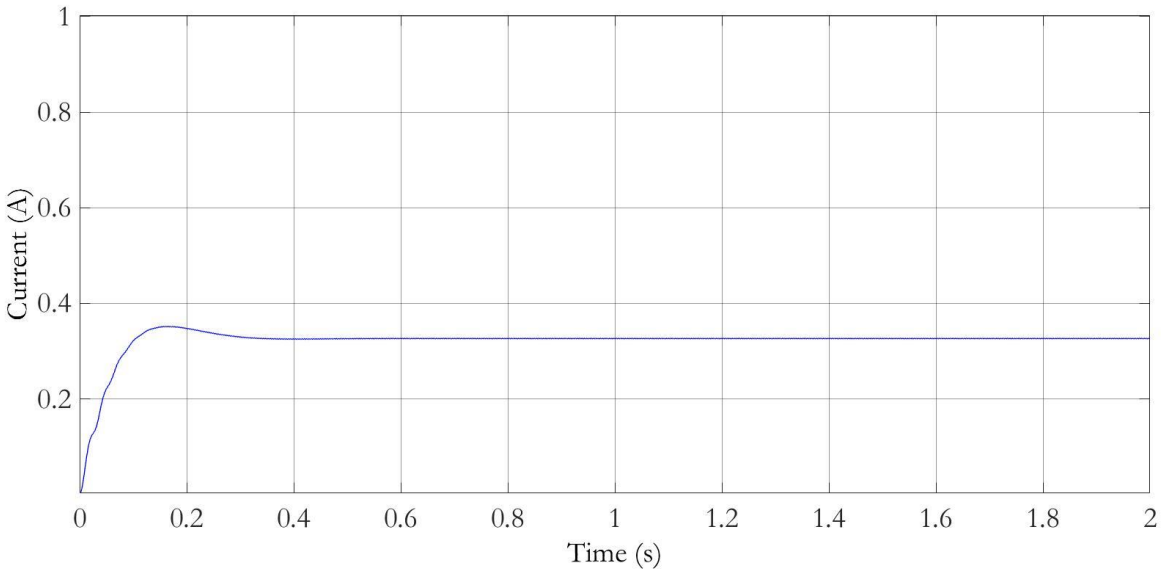

Fig. 3.25. Estimated magnitude of the current's fifth harmonic component.

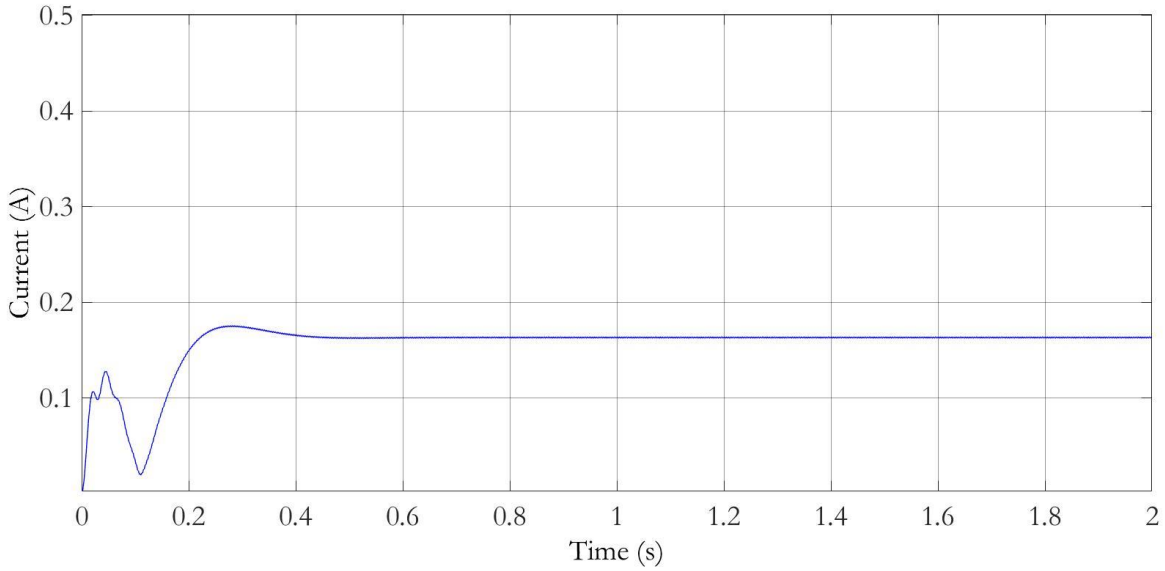

Fig. 3.26. Estimated magnitude of the current's seventh harmonic component.

3.2.7. Final three-phase PLL model.

The final stage for the three-phase PLL was to join the voltage loop and the current loop, in order to obtain both sets of measurements, as well as the consumed or produced active and reactive power.

The code from the function blocks containing the voltage and the current loops was joined, and runs simultaneously. Then, the phase difference between the voltage and the current was calculated, and from this, the active and reactive power were obtained.

In a three-phase equilibrated system, the active and reactive power follow the expressions:

$$P = \sqrt{3} \cdot V_L \cdot I_L \cdot \cos(\varphi)$$
$$Q = \sqrt{3} \cdot V_L \cdot I_L \cdot sin(\varphi) \quad (3.15)$$

Where:

P is the active power.

Q is the reactive power.

V_L is the line voltage.

I_L is the line current.

φ is the phase difference between V_L and I_L.

The results obtained for the sets of variables for both voltage and current were the same as in the previous stage, as those portions of the code were left unmodified.

The results for the active and reactive power are shown in Fig. 3.27 and Fig. 3.28, respectively.

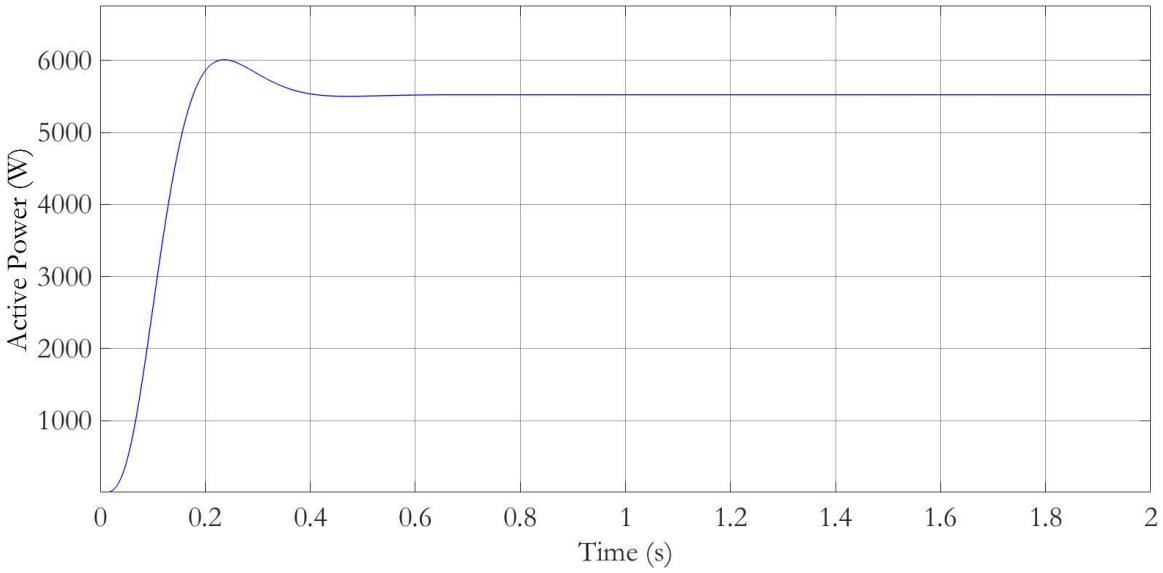

Fig. 3.27. Estimation of the consumed active power.

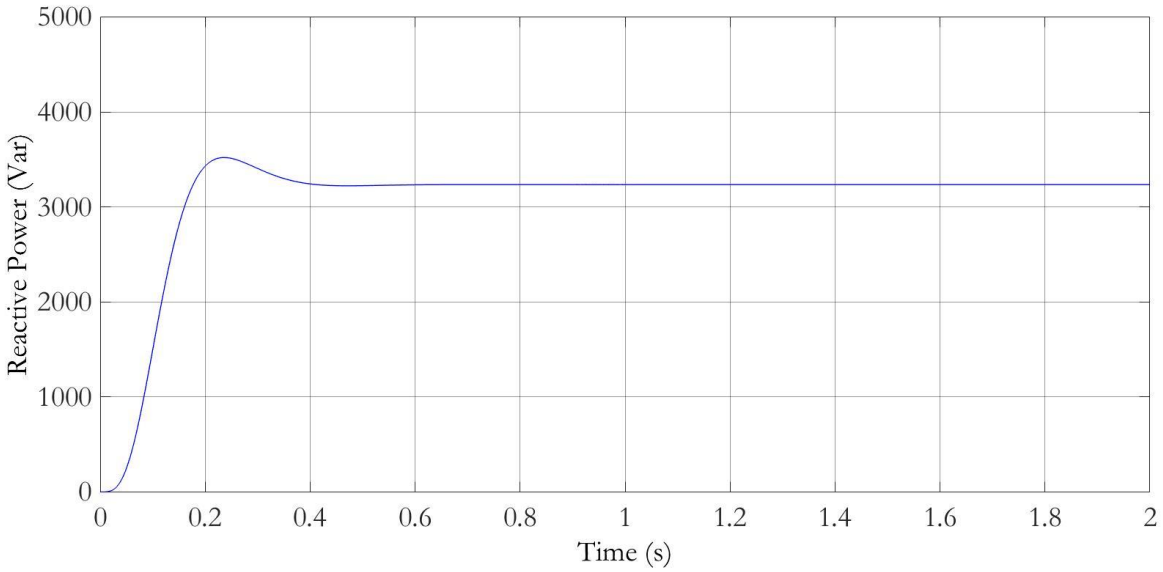

Fig. 3.28. Estimation of the consumed reactive power.

These results are coherent with the selected values for the three-phase voltage, the RL load and the resulting current, and do not show any significant distortion that would require an additional filtering stage.

Thus, this model was selected to be the final three-phase PLL.

3.2.8. Single-phase PLL model.

Single-phase installations appear in most domestic situations, as well as in small to medium businesses and public spaces. In these installations, a single phase is in charge of supplying power to the connected components, and the return of current happens along the neutral wire.

This situation was simulated by connecting a single-phase RL load (a 50 Ω resistor in series with a 1 H inductor) between the terminals A and B of the simulated source (Fig. 3.29). The sensors would then go on to provide a value V for the measured voltage, and I for the current.

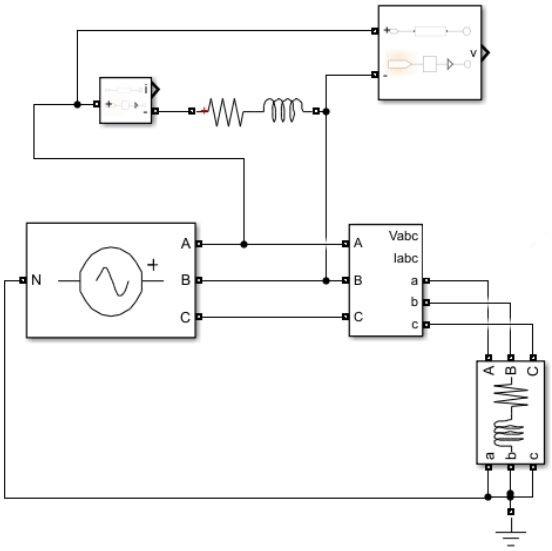

Fig. 3.29. Three-phase source with R-L load connected between phases A and B.

The procedure to retrieve all the electrical parameters was similar to that with a three-phase PLL, explained previously. However, as the input signals are not a set of three-components, an intermediate step needed to be carried out.

First, a signal shifted 90 degrees with respect to the input signal had to be created, both for the voltage and the current. This was achieved by means of a second order generalized integrator-quadrature signal generator (SOGI-QSG). Said shifted signal is the quadrature component, then fed to the alpha-beta-phase to d-q stage (Fig. 3.30).

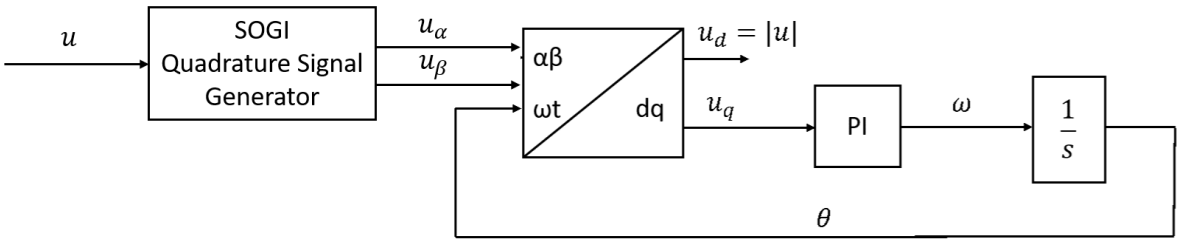

Fig. 3.30. Single-phase PLL using a SOGI-QSG.

The continuous time expression for the outputs of the SOGI-QSG are:

$$D(s) = \frac{u'}{u} = \frac{k \cdot \omega' \cdot s}{s^2 + k \cdot \omega' \cdot s + \omega'^2}$$
$$Q(s) = \frac{qu'}{u} = \frac{k \cdot \omega'^2}{s^2 + k \cdot \omega' \cdot s + \omega'^2} \quad (3.16)$$

Where:

$D(s)$ is the direct component.

$Q(s)$ is the quadrature component, shifted 90 degrees with respect to $D(s)$.

k is a gain coefficient. A critically-damped response is achieved with $k = \sqrt{2}$.

ω' is the estimated input frequency.

It can be observed that the frequency is maintained after the quadrature component is generated.

Applying Tustin's method, the discrete-time expression is obtained:

$$out_n = \frac{2k\omega T_S(in_n - in_{n-2}) - (2\omega^2 T_S^2 - 8)out_{n-1} - (\omega^2 T_S^2 - 2k\omega T_S + 4)out_{n-2}}{\omega^2 \cdot T_S^2 + 2 \cdot k \cdot \omega \cdot T_S + 4}$$
$$outq_n = \frac{k\omega^2 T_S^2(in_n + 2in_{n-1} + in_{n-2}) - (2\omega^2 T_S^2 - 8)outq_{n-1} - (\omega^2 T_S^2 - 2k\omega T_S + 4)outq_{n-2}}{\omega^2 T_S^2 + 2k\omega T_S + 4} \quad (3.17)$$

Where:

out_n is the output at a stage n.

$outq_n$ is the quadrature output at a stage n.

out_{n-1} is the output at a stage $n-1$.

out_{n-2} is the output at a stage $n-2$.

$outq_{n-1}$ is the quadrature output at a stage $n-1$.

$outq_{n-2}$ is the quadrature output at a stage $n-2$.

in_n is the input at a stage n.

in_{n-1} is the input at a stage $n-1$.

in_{n-2} is the input at a stage $n-2$.

k is a gain coefficient equal to $\sqrt{2}$.

ω is the natural frequency.

T_S is the time step constant.

This operation can be tested by using the single-phase voltage signal V as an input for a block containing the SOGI-QSG function.

As it is shown in Fig. 3.31, the direct output out_n and its quadrature component $outq_n$ have a unitary magnitude at the fundamental frequency (50 Hz), and have a phase shift of 90º respect to each other.

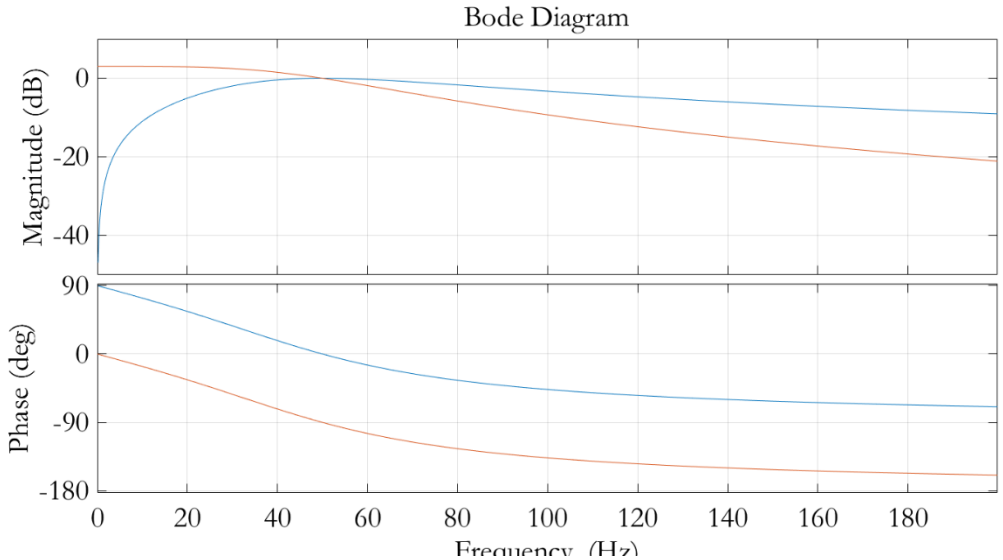

Fig. 3.31. Bode diagram of out_n (blue) and $outq_n$ (red), using V as the input.

For the segment of the code that calculates the frequency and phase shift for the voltage and the current, the input signals V and I were filtered using two notch filters in cascade in order to remove the fifth and seventh harmonics before entering each SOGI-QSG (Fig. 3.32).

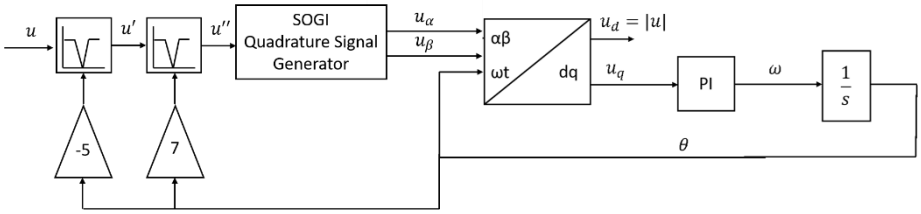

Fig. 3.32. Single-phase PLL, use of notch filters at the SOGI-QSG's input.

This was done to achieve a precise measurement for the phase shift, as the SOGI-QSG might generate unwanted harmonics, and a small change in the angle φ might cause large distortion in the results for the active and reactive power.

Then, the outputs from the SOGI-QSG were processed in the same way as the dq components in the three-phase PLL, during the calculation of the frequency and phase shift.

If these filtered signals were used in the calculation of the magnitudes of the fundamental, fifth and seventh components of the input voltage and current, the results obtained would not be precise, as they would lack the information about the harmonics that were previously removed.

For this reason, during that stage of the code, the inputs for another SOGI-QSG are the unfiltered input signals V and I.

The obtained results for the fundamental components of the input voltage and current are shown in Fig. 3.33 and Fig. 3.34, respectively.

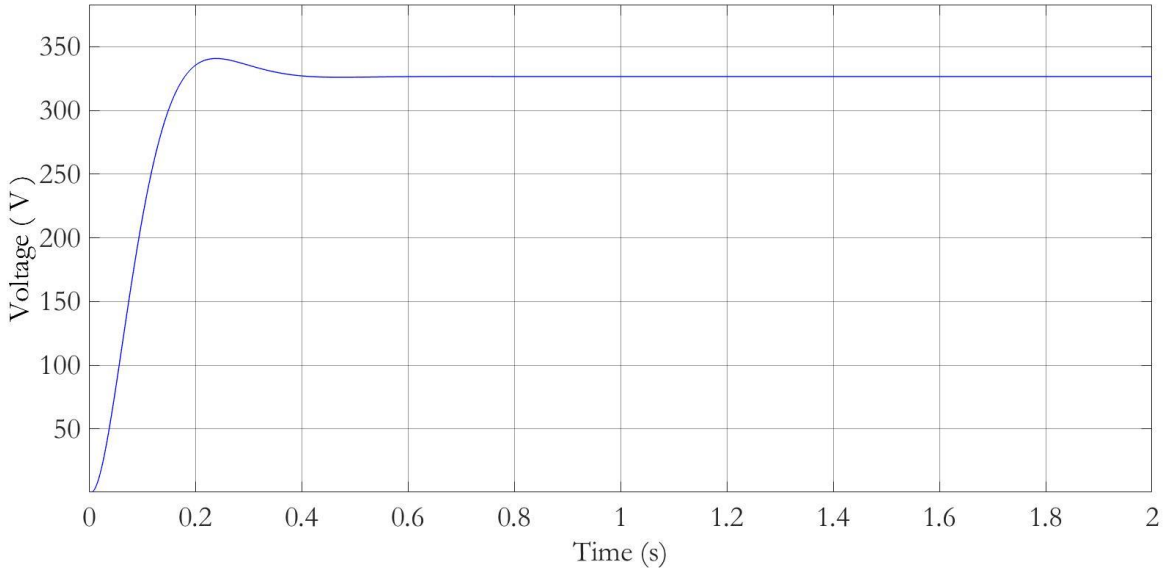

Fig. 3.33. Estimated peak magnitude of the voltage's fundamental component.

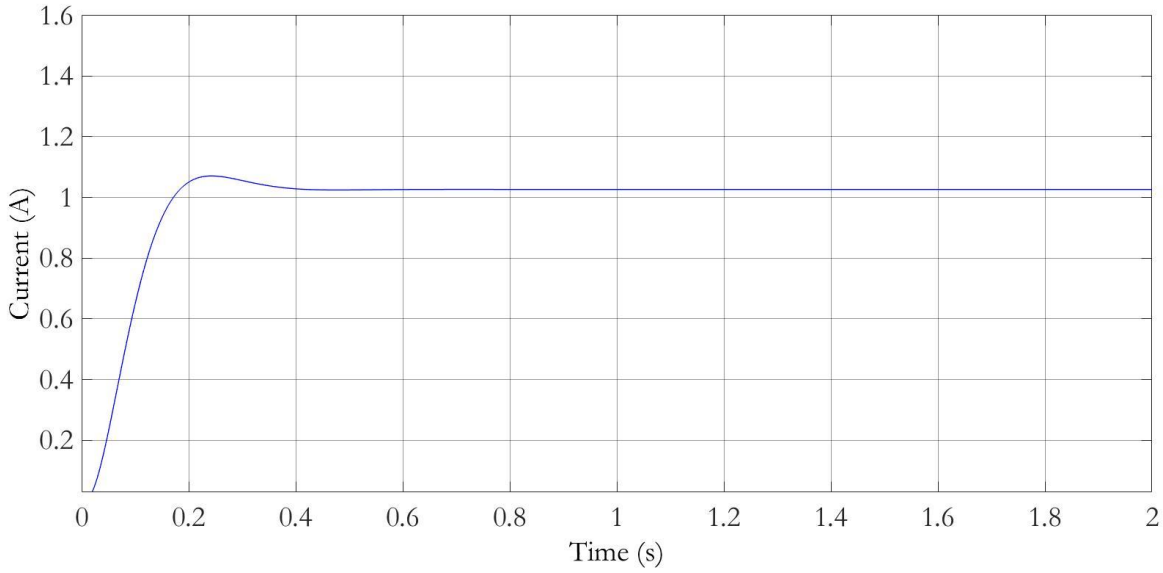

Fig. 3.34. Estimated peak magnitude of the current's fundamental component.

The estimated frequency is shown in Fig. 3.35.

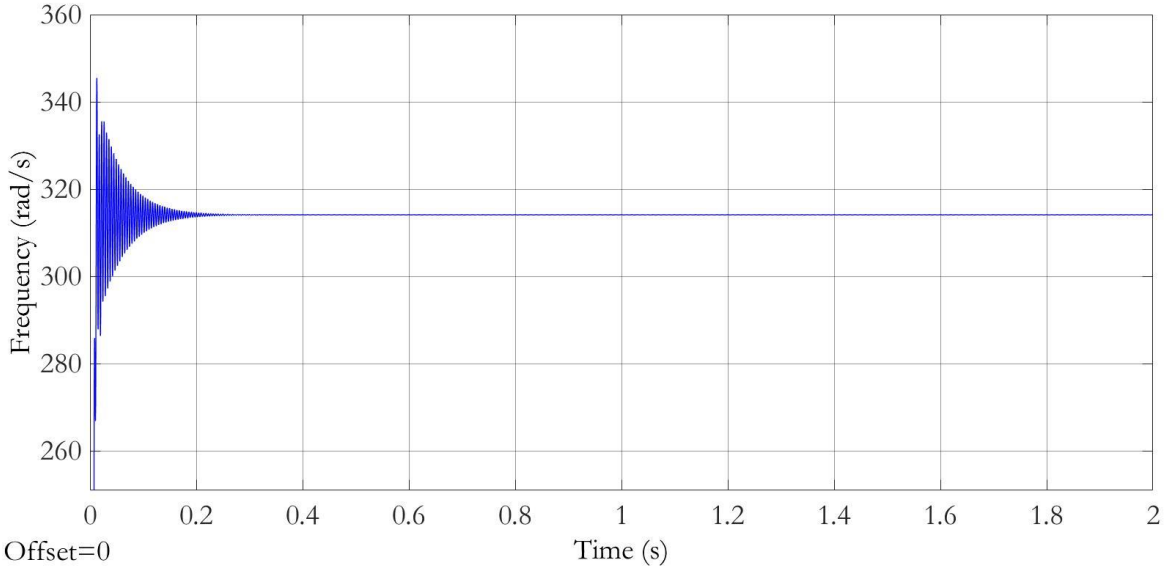

Fig. 3.35. Estimated frequency for the single-phase signal.

The results for the consumed active and reactive power are presented in Fig. 3.36 and Fig. 3.37, respectively.

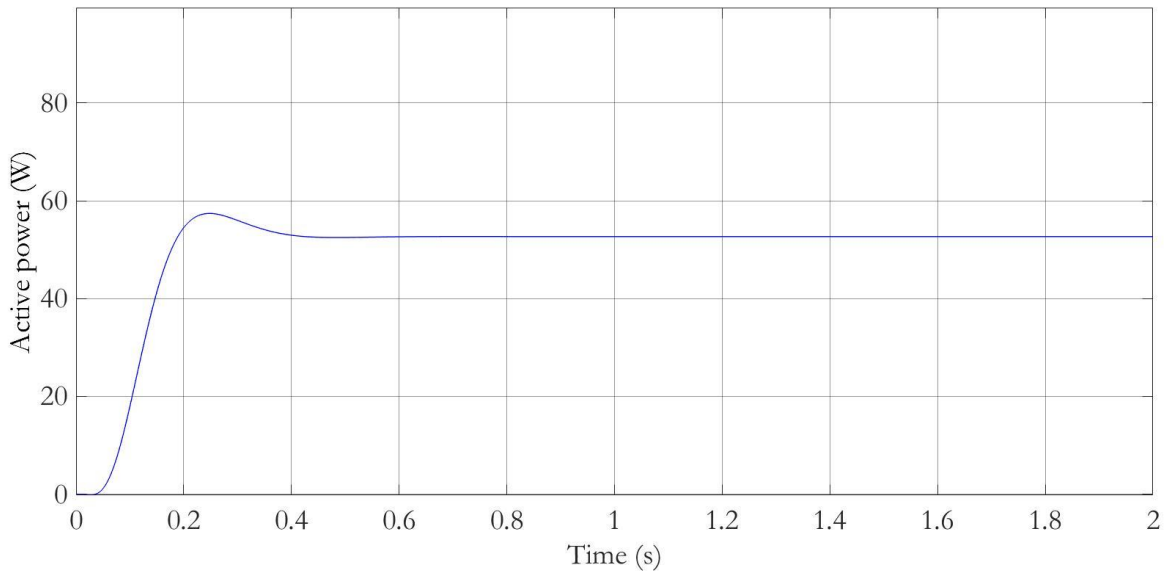

Fig. 3.36. Estimation for the consumed active power.

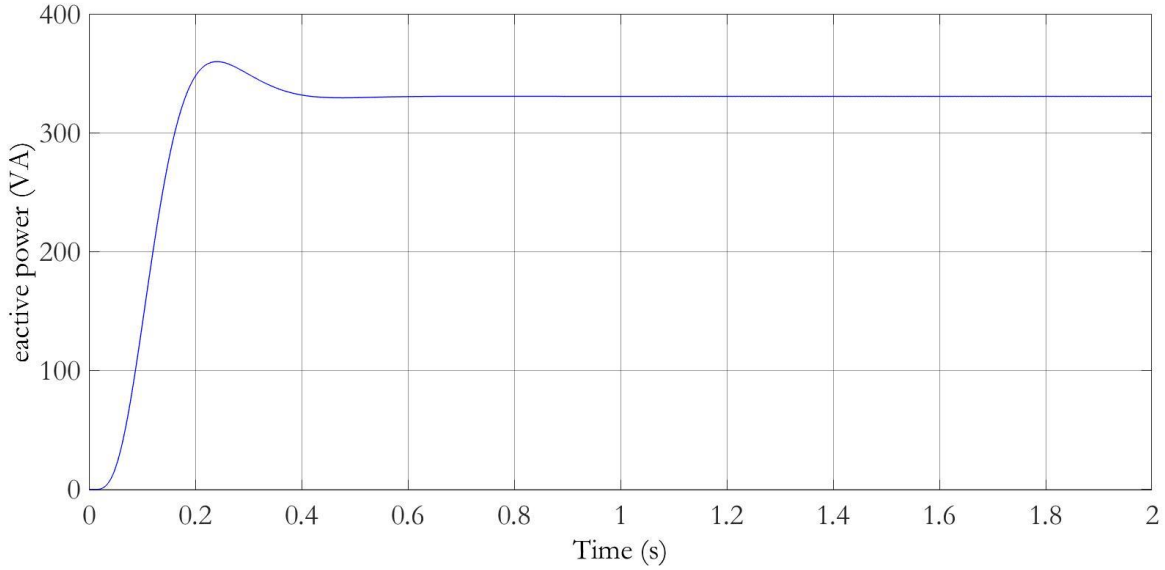

Fig. 3.37. Estimation for the consumed reactive power.

These results correspond to those expected from highly reactive loads.

The magnitudes for the fifth and seventh harmonic components of the voltage signal are shown in Fig. 3.38 and Fig. 3.39, respectively.

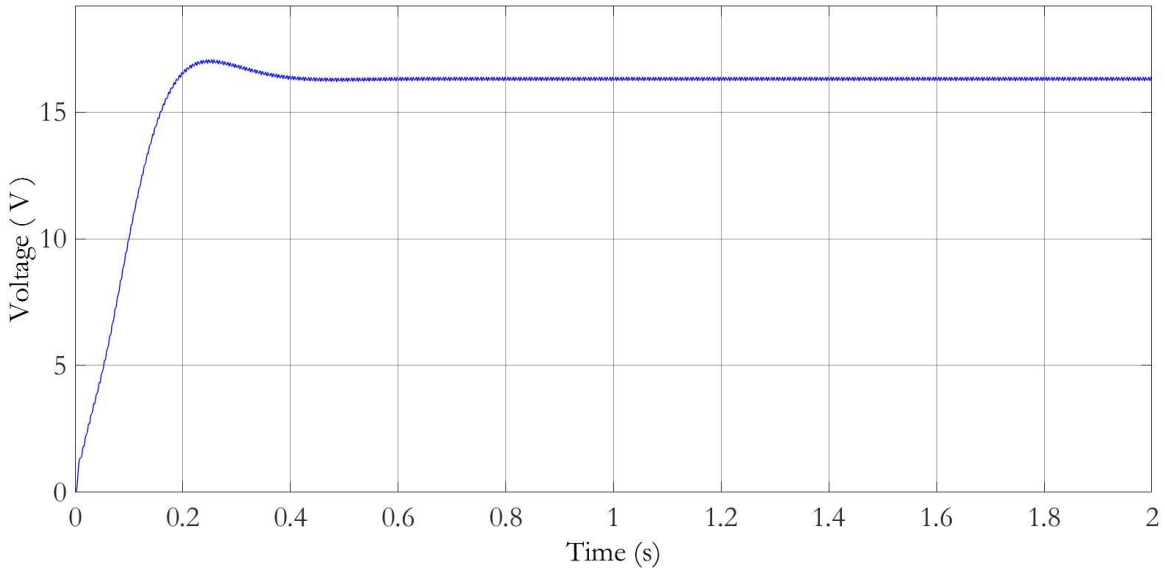

Fig. 3.38. Estimated magnitude of the voltage's fifth harmonic component.

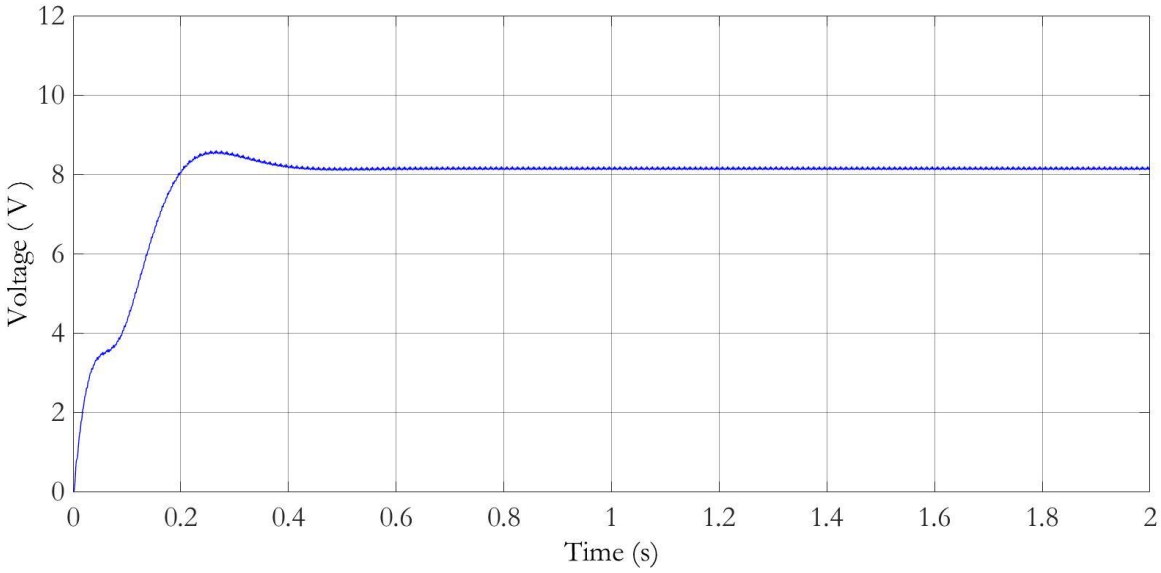

Fig. 3.39. Estimated magnitude of the voltage's seventh harmonic component.

4. Experimental verification.

4.1. Simulation using a real voltage source's equivalent.

The previous tests performed during the design process had all been carried out using ideal three-phase sources.

As the aim of this project was to create a device suitable for real domestic and industrial situations, a simulation with a variable source must be carried out in order to verify that the measurements are able to withstand changes in the input variables.

For this purpose, a variable voltage source was applied at the V_{abc} input of the three-phase PLL.

The events were scheduled as follows:

- *Magnitude:* from t = 0 s to t = 1.5 s, the voltage is set as $\sqrt{2} \cdot 230$ V. In t = 1.5 s, the magnitude rises to 1.5 times its previous value, and returns to its initial value at t = 3 s.
- *Frequency:* from t = 0 s to t = 1 s, the frequency is set as 50 Hz. In t = 1 s, it is changed to 51 Hz, and returns to its initial value at t = 2.5 s.

The fifth and seventh harmonics are present from t = 2 s onwards, and both of their magnitudes equal 5% that of the fundamental component.

White noise is present during the entire cycle.

This source simulates the behaviour of a 2210 TC-ACS-50-480-400 programmable source from the company Regatron, used for the article *Predictive Frequency-Based Sequence Estimator for Control of Grid-Tied Converters Under Highly Distorted Conditions* [9].

For the I_{abc} input of the PLL, the simulated installation that was used during the development of the algorithm was modified. The three-phase ideal voltage source was substituted by three controlled voltage sources, with their control inputs connected to V_a, V_b and V_c from the variable source.

The load connected to the installation was established as a RL load with a resistance of 1 Ω and an inductance of 1 H, in order to obtain visible results for both active and reactive power.

The results obtained during this stage for the estimation of the fundamental, fifth and seventh harmonic components of the voltage are shown in Fig. 4.1, Fig. 4.2 and Fig. 4.3, respectively.

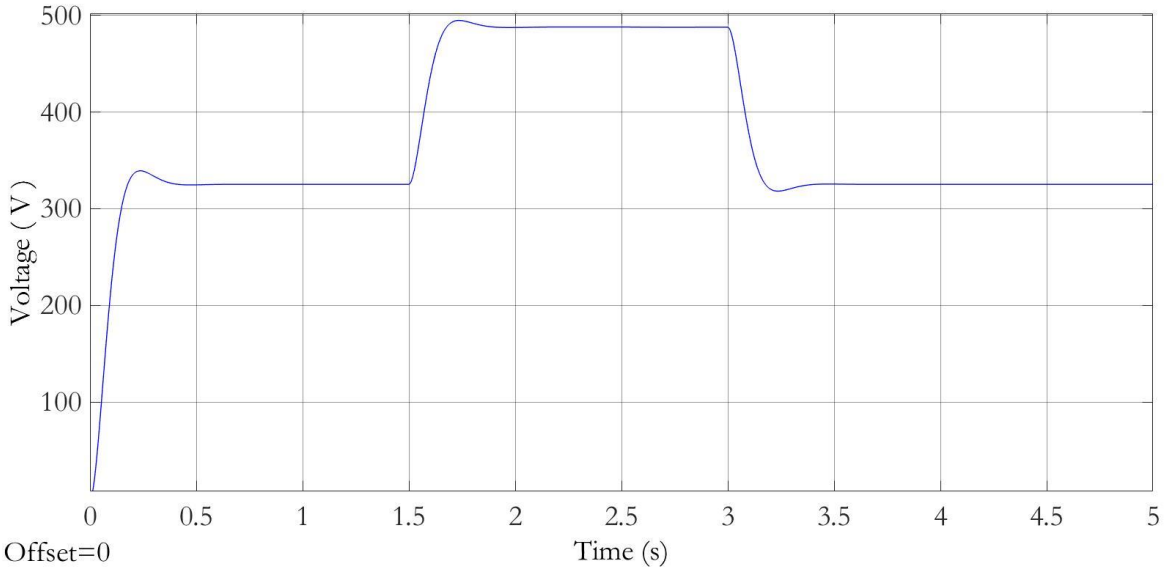

Fig. 4.1. Measurement of the fundamental component of the input voltage.

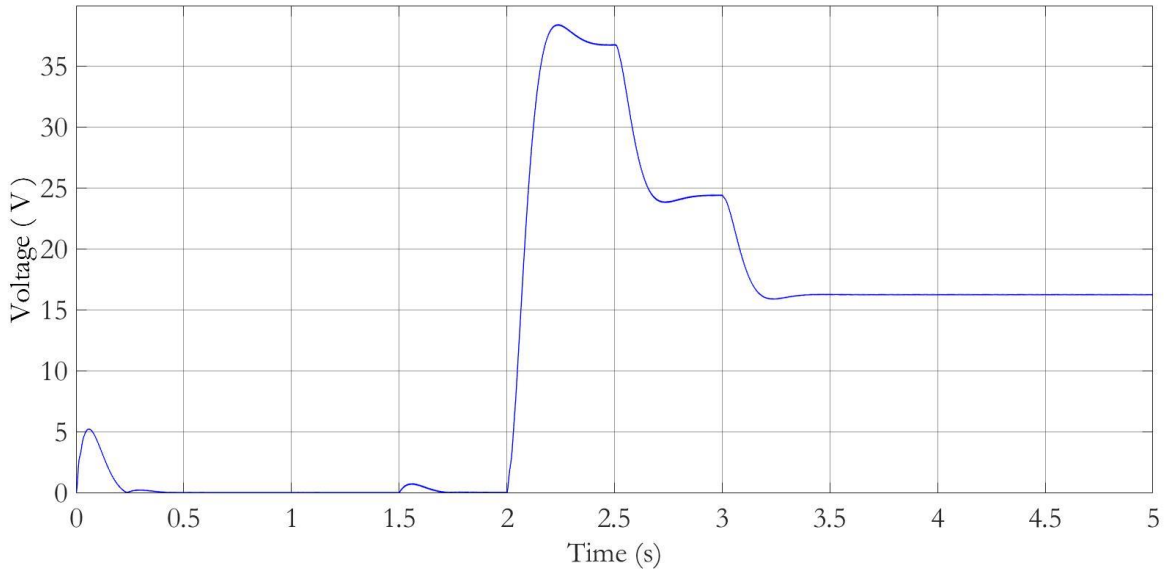

Fig. 4.2. Measurement of the fifth harmonic component of the input voltage.

It was observed that the obtained values rapidly adapt when the magnitude of the fundamental component changes, an also when a change in its frequency takes place.

The transient time for these measurements is in the order of 0.2 seconds, which is considered sufficiently short for most domestic or industrial applications.

The PLL also functions correctly in terms of obtaining precise results, regardless of the existence of white noise during the entire simulation period.

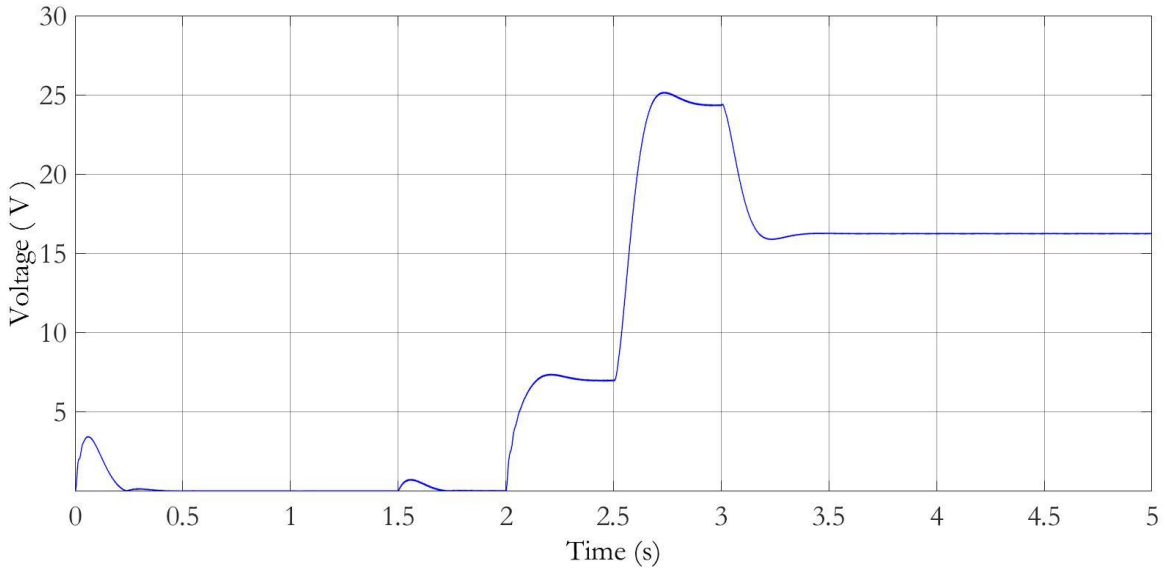

Fig. 4.3. Measurement of the seventh harmonic component of the input voltage.

The consumed current is affected by the amplitude of the input voltage and its frequency, as the RL load's value is dependent on the latter.

The measurements of the fundamental, fifth and seventh harmonic components of the consumed current are shown in Fig. 4.4, Fig. 4.5 and Fig. 4.6, respectively.

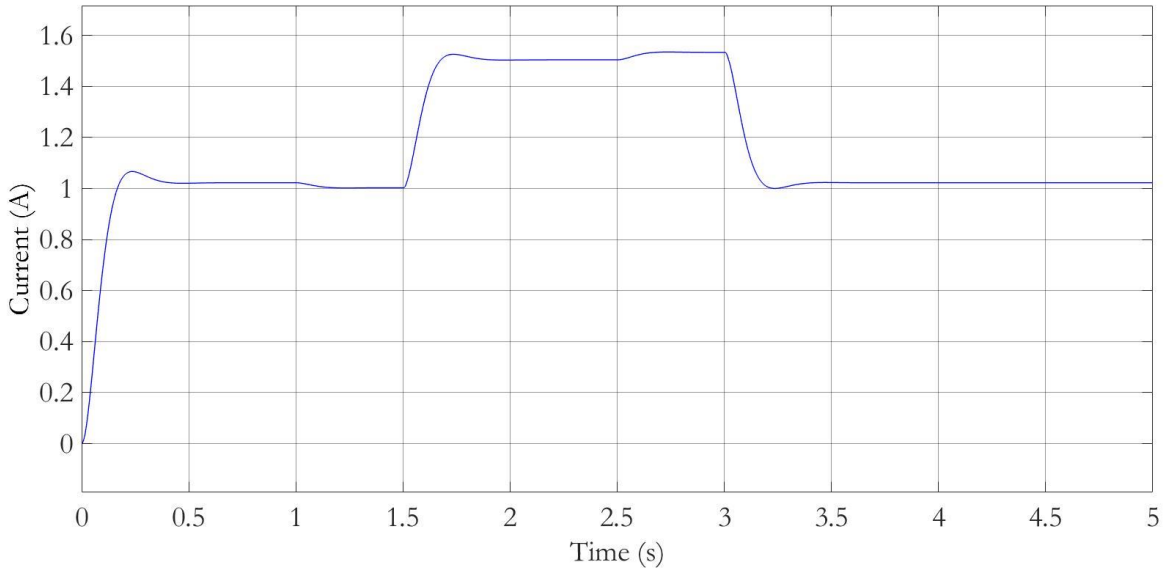

Fig. 4.4. Measurement of the fundamental component of the consumed current.

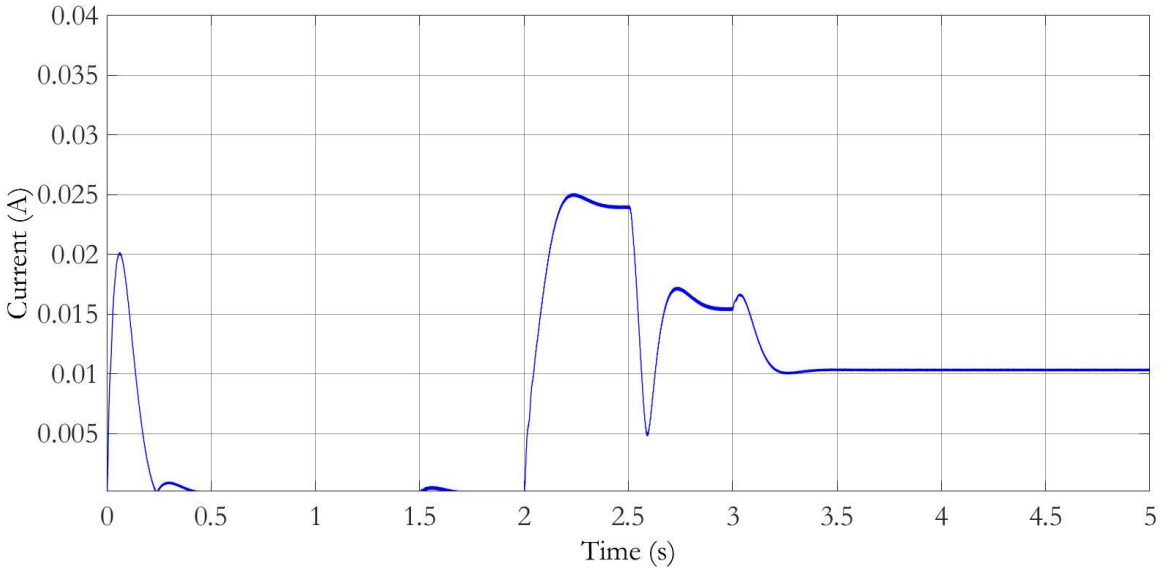

Fig. 4.5. Measurement of the fifth harmonic component of the consumed current.

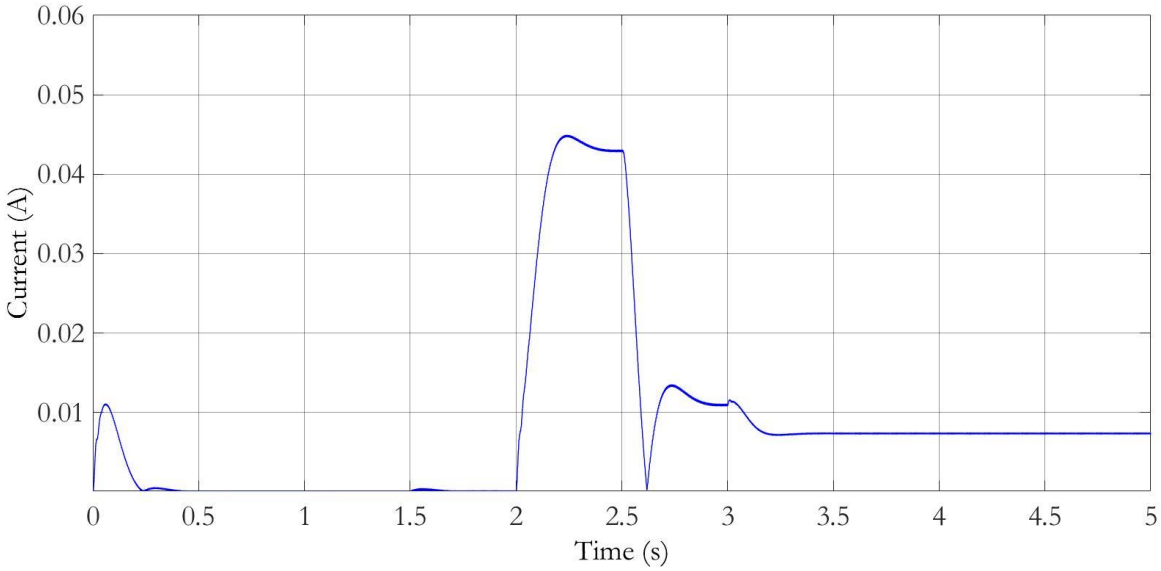

Fig. 4.6. Measurement of the seventh harmonic component of the consumed current.

The measurement of the voltage's fundamental frequency is presented in Fig. 4.7.

It can be noted that a small distortion appears in the instants where a change in the voltage's magnitude or its frequency happens, and also when the fifth and seventh harmonic components are introduced.

This behaviour is normal, and a precise measurement of the frequency is obtained after a transient period with a shorter duration to that of the voltage's transient, except for the instant in which harmonics are introduced, when the distortion lasts for a longer period.

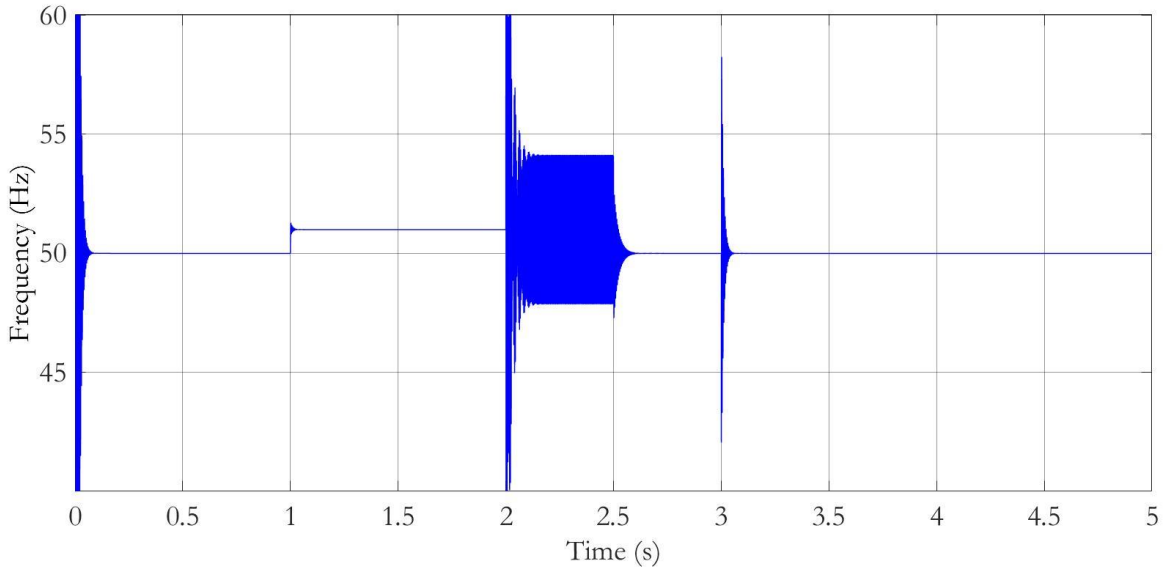

Fig. 4.7. Estimated fundamental frequency of the input voltage.

In Fig. 4.8 and Fig. 4.9, the results for the consumed active and reactive power are presented.

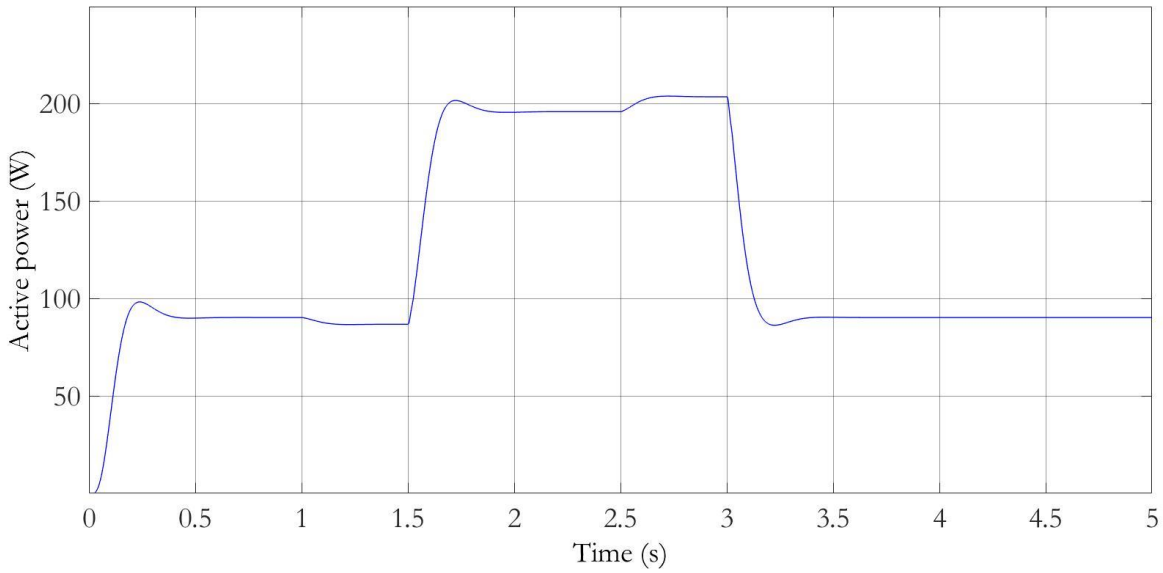

Fig. 4.8. Estimation for the consumed active power.

These estimations are consistent with the increments on the voltage's magnitude, and the changes in the frequency, which subsequently modify the impedance of the RL load, affecting the P/Q ratio.

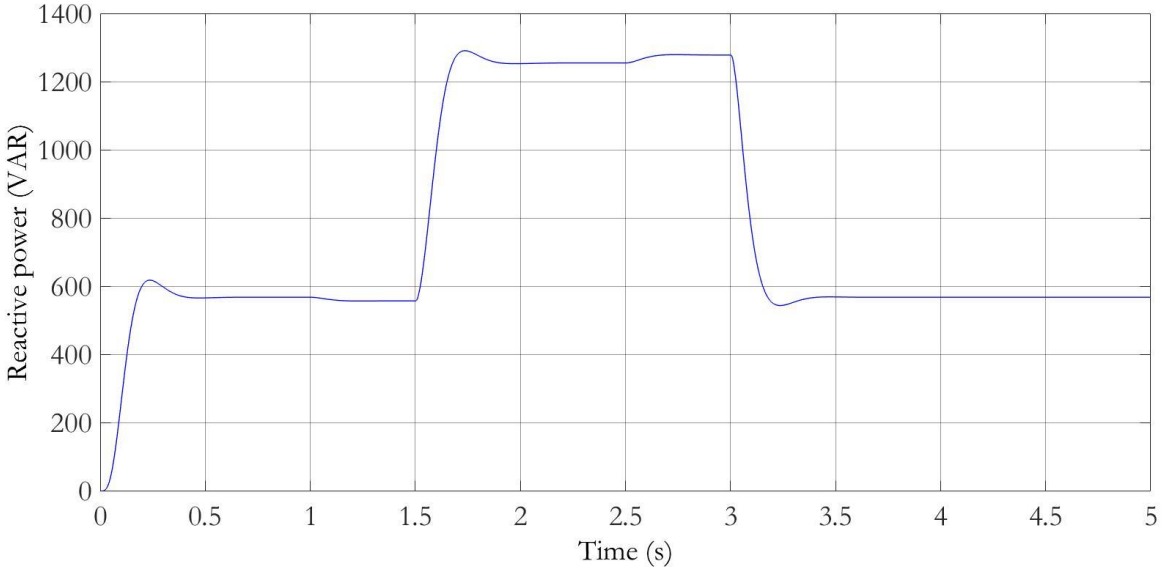

Fig. 4.9. Estimation for the consumed reactive power.

After the evaluation of these results, it was observed that the PLL was able to obtain precise results for all the required parameters, in a sufficiently short period of time. Thus, at the next stage, the designed algorithm was adapted to be deployed on a physical device.

4.2. Choice of the microcontroller board.

The most commonly used microcontroller boards for projects with comparable characteristics are the ones produced by Arduino and Raspberry Pi.

Both architectures share similarities in terms of processing power, size, capability for connections and cost. However, Arduino is considered to be a microcontroller, while Raspberry Pi functions as a credit card-sized computer.

When it comes to clock speed, Raspberry Pi is faster than Arduino. This offers the possibility to receive, analyse and showcase more data, thus allowing for a higher precision in calculations.

For this reason, the chosen board was one from the Raspberry Pi catalogue.

The initial choice for the board was the Raspberry Pi Zero model. However, the board that was finally used was a Raspberry Pi Model 3 B+ (Fig. 4.10), due to connectivity and compatibility issues found during the development stage

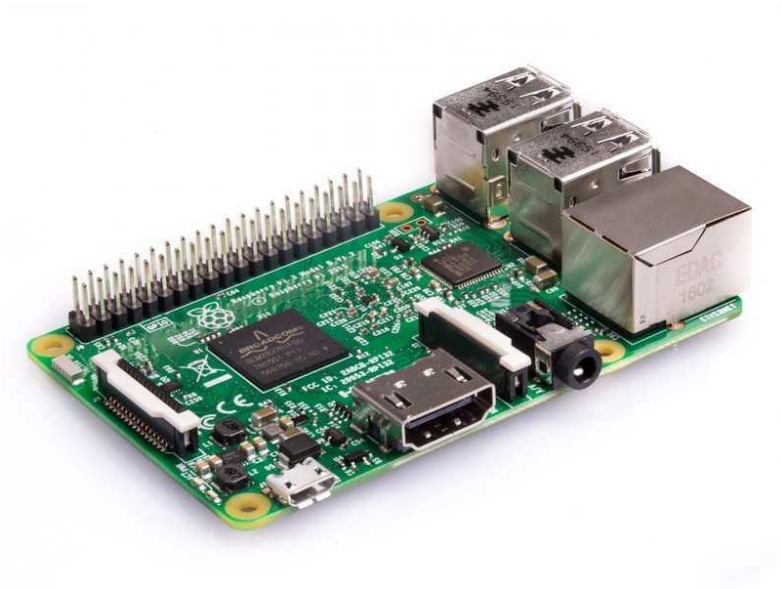

Fig. 4.10. View of the Raspberry Pi 3 Model B. Source: www.raspberrypi.org

This board has the following technical specifications:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2 ports.
- 4 Pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi camera.
- DSI display port for connecting a Raspberry Pi touchscreen display.
- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source up to 2.5A.

4.3. Installation of Raspbian Operative System.

The operative system on which the board would typically run in most designs is called Raspbian, and consists of a version of the open-source OS Linux especially designed for the Raspberry Pi series.

This was the first choice in this project, and was installed using NOOBS (New Out Of Box Software), which is the basic OS installation manager developed by Raspberry Pi.

This tool is available for download at the link:

https://www.raspberrypi.org/downloads/noobs/

In order to install the OS on an SD card, these steps are to be followed:

- An SD with a capacity of 8 Gb or higher must be formatted as FAT.
- The .zip file obtained from the previous link must be extracted, and the resulting files will be copied on the SD card's root.
- After inserting the SD card on the Raspberry Pi board, it can be connected to a screen using an HDMI cable, and using a keyboard or a mouse, the installation steps can be followed on screen.
- The desired OS is selected during this process. As previously mentioned, we will choose Raspbian.

The entire process is fairly simple, and at the end of it the Raspberry Pi board is ready to use. At this moment, the default username (pi), and the password (raspberry), can be modified, and the Wi-Fi connection can be configured.

4.4. Implementation of the single-phase PLL using Python.

The algorithm for the single-phase PLL designed using Simulink can be implemented using different programming languages. In this case, Python was selected for its simplicity and the existence of many libraries that allow for the interaction of the Raspberry Pi with different sensors. The Python language also offers a selection of methods for exporting the obtained results, for example, direct visualization in the terminal or the creation of Excel files.

The code developed follows a similar structure to that produced in Simulink. The initial values for the variables are set, then the functions are declared and used to perform calculations, the output values are updated in each iteration, and finally these results are shown or exported.

The complete code for the single-phase PLL can be found in Annex I-Code.

A series of libraries need to be called at the beginning of the code:

- `import spidev` – Allows for the SPI ports to be used.
- `import sys` – Imports the system configuration.
- `import time` – Imports the clock process.
- `import threading` – Allows for multiple processes to occur at the same time.
- `import math` – Imports mathematical formulae and constants.
- `import numpy` – Imports the complex numbers' library.
- `import os` – Imports the system configuration.
- `import csv` – Allows for CSV files (comma separated values) to be used.

For the preliminary verification using Python, the simulation points obtained from the single-phase installation without harmonics in Simulink were saved in a CSV file.

This file contained 20000 values (the simulation lasted 2 seconds, with a time step of 0.1 ms) for the time, phase voltage and current. These data were the same that served as inputs for the Simulink function.

In order to read from said CSV file, the following function was used. It was necessary to convert the values from string to float type before using them.

```
with
open('dataclean.csv','r') as
csvfile:

readCSV=csv.reader(csvfile,d
elimiter=';')
    for row in readCSV:
        Vcsv=row[1]
        Icsv=row[2]

v=float(Vcsv)
        i=float(Icsv)
```

The obtained results for the simulations both with and without harmonic components were saved in different CSV files that can be visualised using Microsoft Excel, for example, using the following function:

```
f=open('resultswoharmonics.c
sv','a',newline='')
        cssv=csv.writer(f)

        data=[magv1]
        cssv.writerow(data)
```

In the PI controller function, new values were selected for its constants:

- $k_p = 251.3274$
- $k_i = 62.8319$

The resulting estimated frequency is shown in Fig. 4.11.

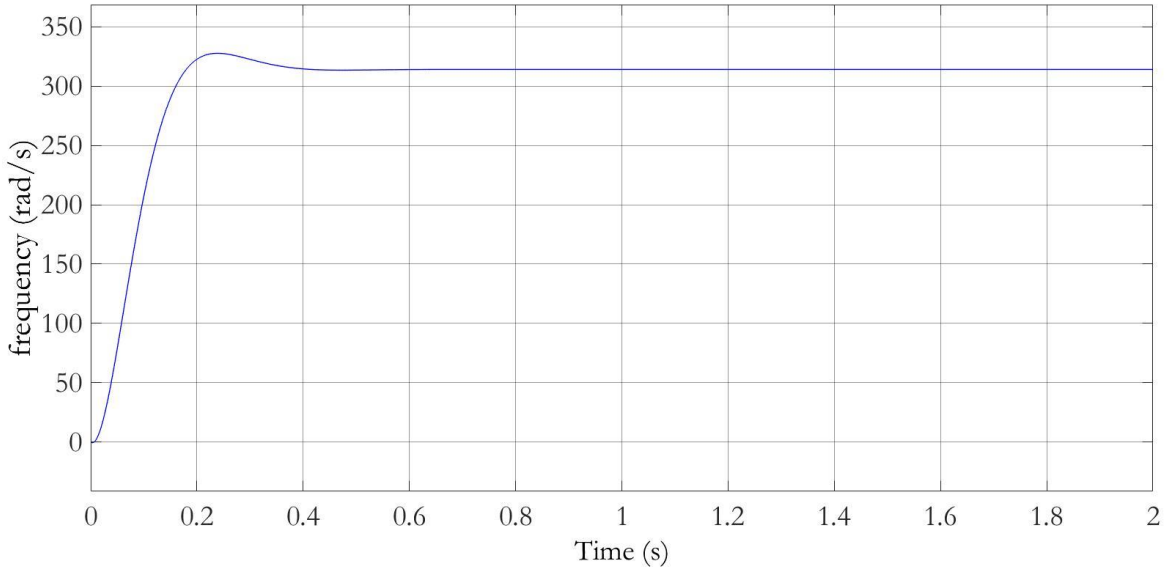

Fig. 4.11. Estimated frequency for the single-phase signal without harmonics.

It is worth mentioning that the correct frequency of 314.16 rad/s is estimated at a much faster rate using the Raspberry Pi, compared to using Simulink. Thus, the values for the variables estimated using the frequency as an input are expected to converge faster.

The results for the estimation of the peak voltage are shown in Fig. 4.12. This value is expected to be in the proximity of 325.27 V.

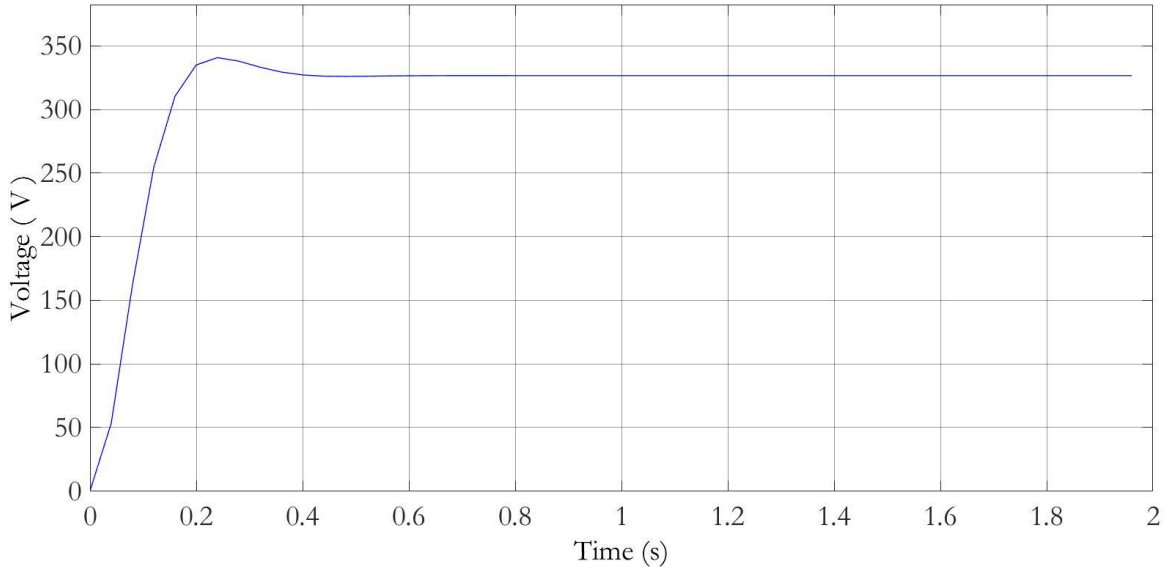

Fig. 4.12. Estimated peak voltage for the single-phase signal without harmonics.

Fig. 4.13 shows the estimation of the peak current.

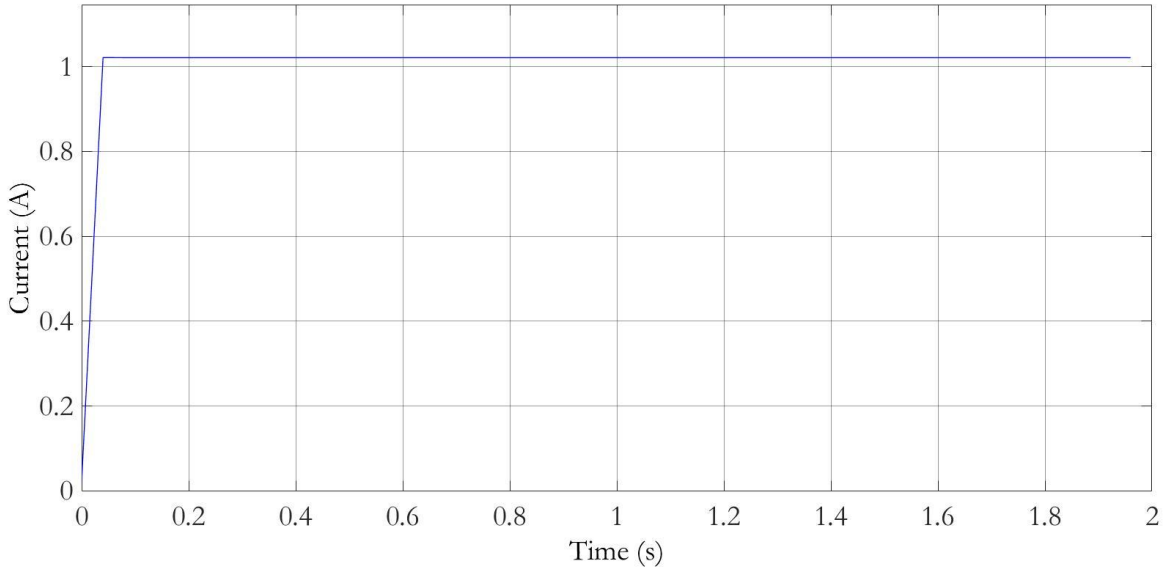

Fig. 4.13. Estimated peak current for the single-phase signal without harmonics.

These estimations follow a similar path to those previously obtained using Simulink.

The rate of convergence has increased compared to those previous estimations due to the improved constants for the PI regulator.

The results for the consumed active and reactive power are showcased in Fig. 4.14 and Fig. 4.15, respectively.

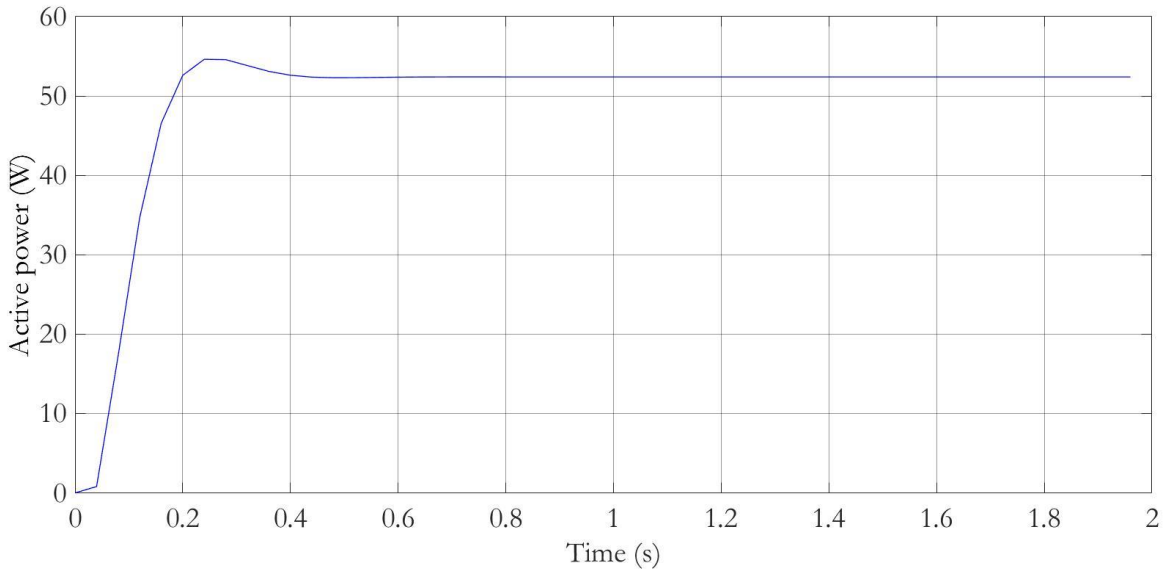

Fig. 4.14. Estimated consumed active power.

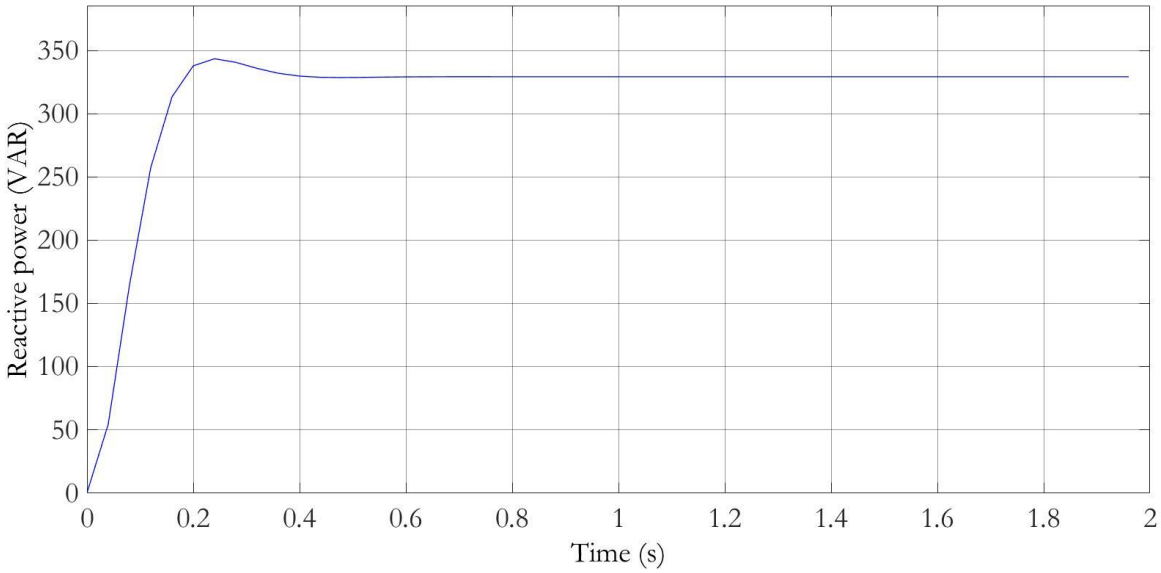

Fig. 4.15. Estimated consumed reactive power.

These results are coherent with those to be expected from a highly inductive load such as the one modelled (a 50 Ω resistor in series with a 1 H inductor).

These loads are uncommon in real applications, but the software was tested considering these values in order to test the sturdiness of the algorithm in extreme situations.

After having verified the correct functioning of the code while processing a signal without harmonics, a new CSV file containing a signal with harmonic components was tested. This signal contained a fifth component with a magnitude equal to 5% of the fundamental's magnitude and a seventh component with a magnitude equal to 2.5% of the fundamental's magnitude. The resulting estimated frequency is shown in Fig. 4.16.

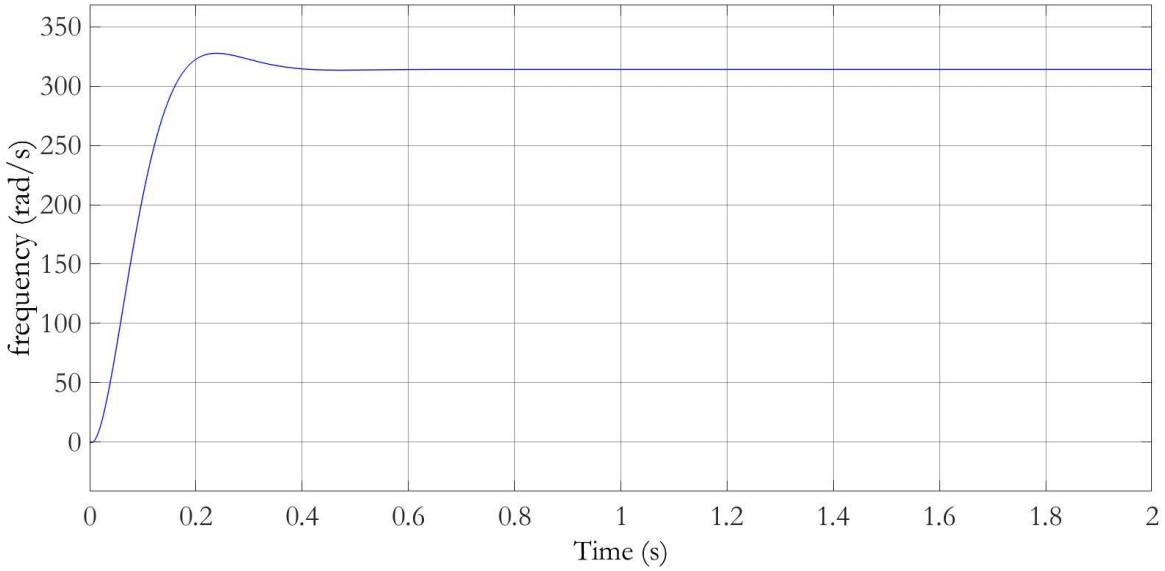

Fig. 4.16. Estimated frequency for the single-phase signal with harmonics.

The results for the estimation of the magnitude of the voltage's fundamental component are shown in Fig. 4.17.

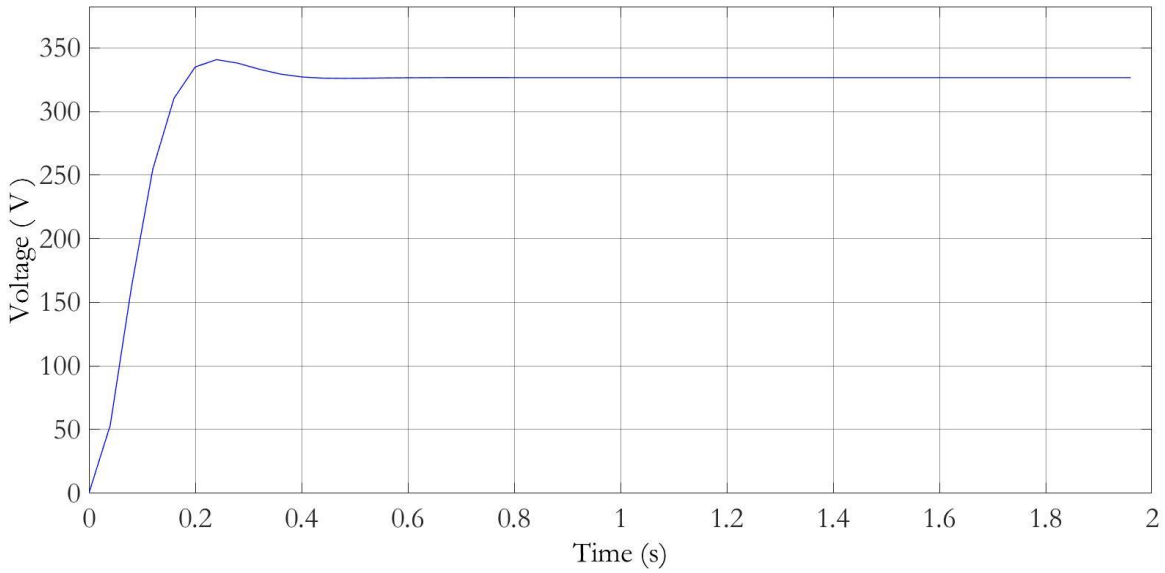

Fig. 4.17. Estimated peak voltage for the single-phase signal with harmonics.

The results for the estimations of the fifth and seventh harmonic components of the voltage signal are shown in Fig. 4.18 and Fig. 4.19.

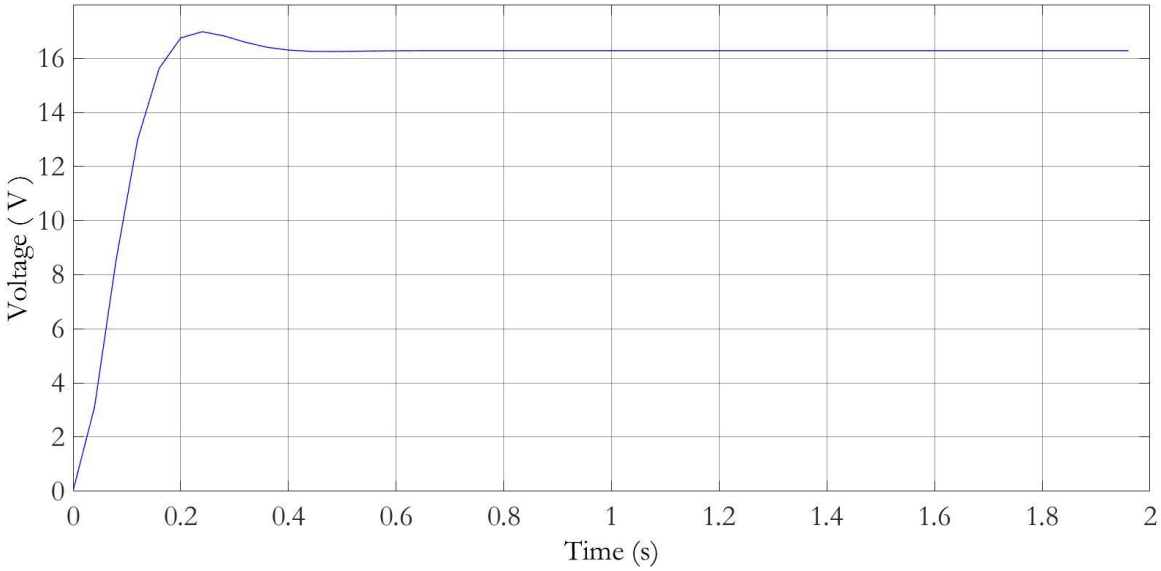

Fig. 4.18. Peak magnitude of the voltage's fifth harmonic component.

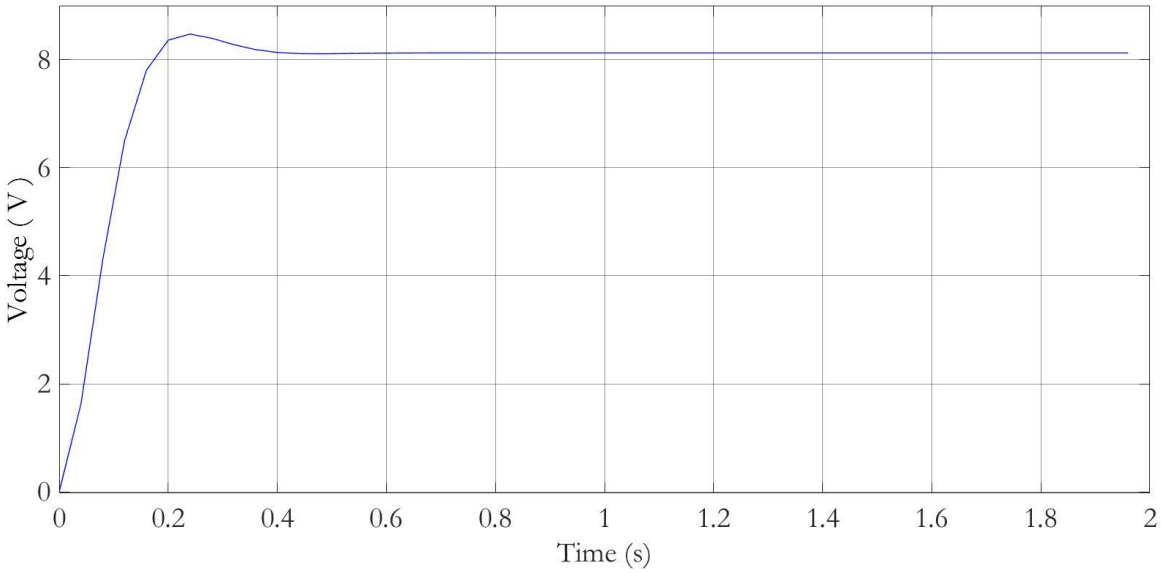

Fig. 4.19. Peak magnitude of the voltage's seventh harmonic component.

These results are, as expected, estimated at magnitudes equal to 5% and 2.5% of that of the fundamental component, respectively.

The results for the estimation of the current's fundamental component are shown in Fig. 4.20. The results for its fifth and seventh harmonic components are not shown, as their magnitudes are negligible.

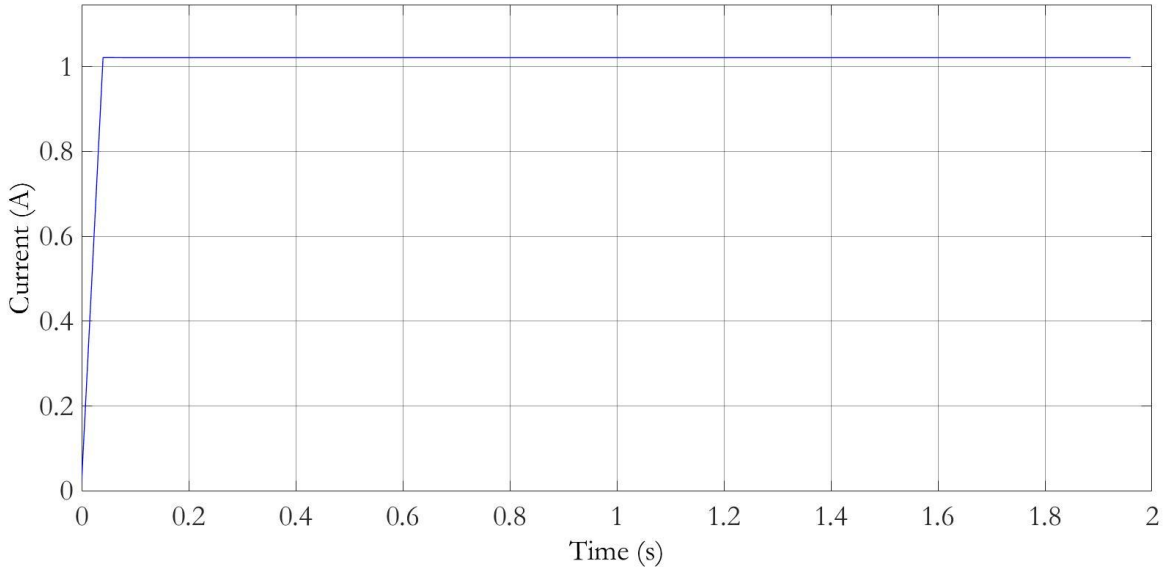

Fig. 4.20. Peak magnitude of the current's fundamental component.

The active and reactive power consumed by the load are shown in Fig. 4.21 and Fig. 4.22 respectively.

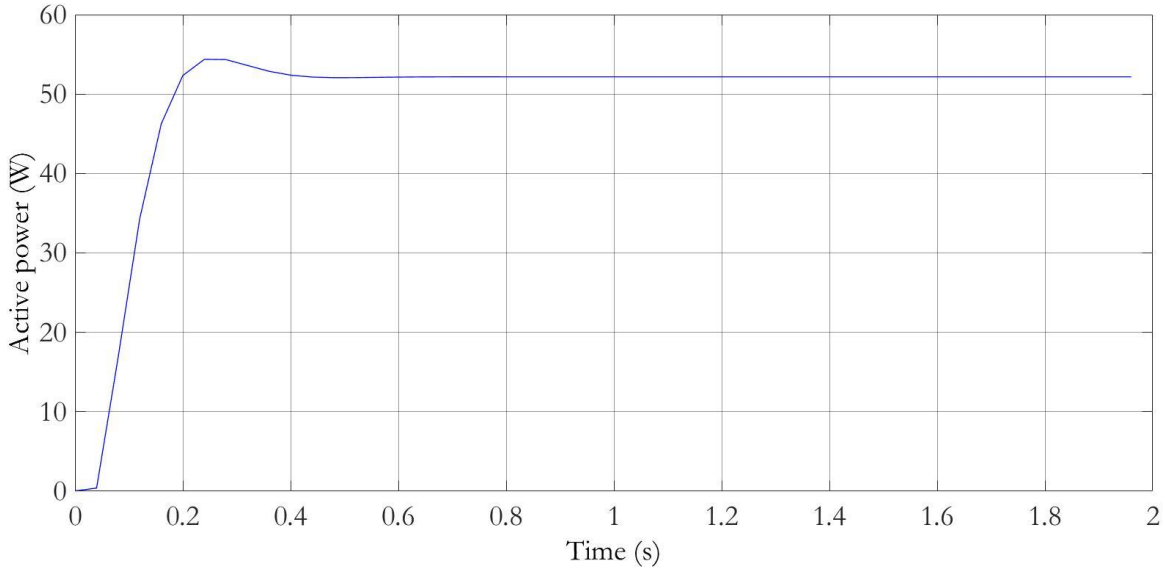

Fig. 4.21. Estimated consumed active power.

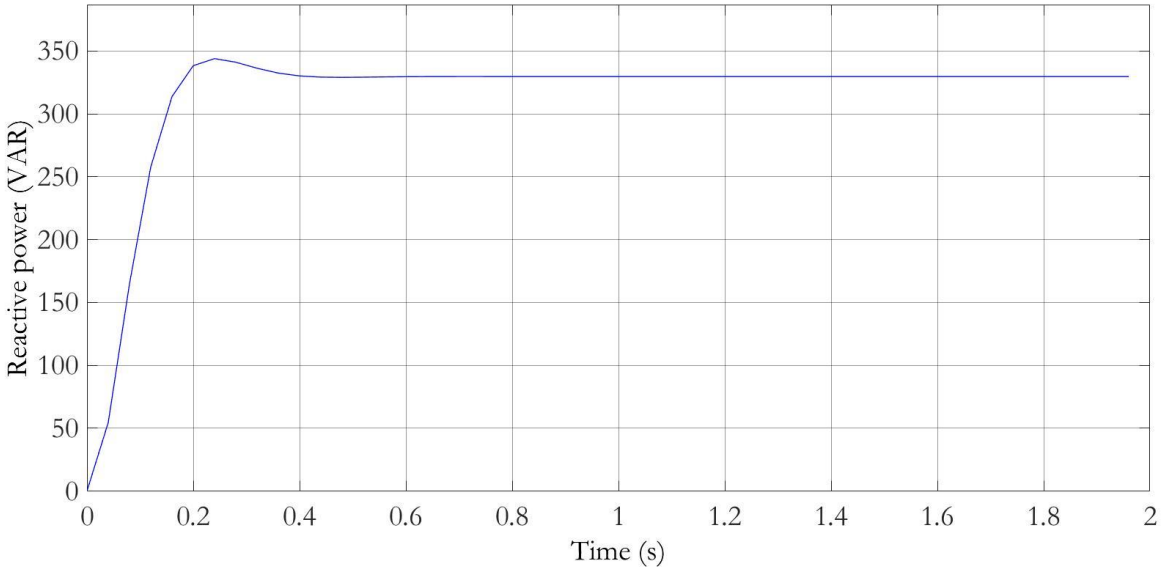

Fig. 4.22. Estimated consumed reactive power.

4.5. Choice of the current sensors.

The actual data regarding current and voltage are to be retrieved using different sensors.

For the Smart Meter to be as easy to install as possible, a non—invasive current sensor was chosen. One of the most commonly used sensors fitting into this category is the SCT013 (Fig. 4.23).

Fig. 4.23. SCT013 non-invasive current sensor.

This sensor contains a current transformer with a transformation ratio of 1/2000. The primary coil is a solid piece of ferrite, divided in half and inserted in a clamp. The secondary coil has 2000 turns. When clamped around a single cable, the current flowing through said cable magnetises the primary coil, which in turn induces a current on the secondary coil 2000 times smaller than the measured one (Fig. 4.24).

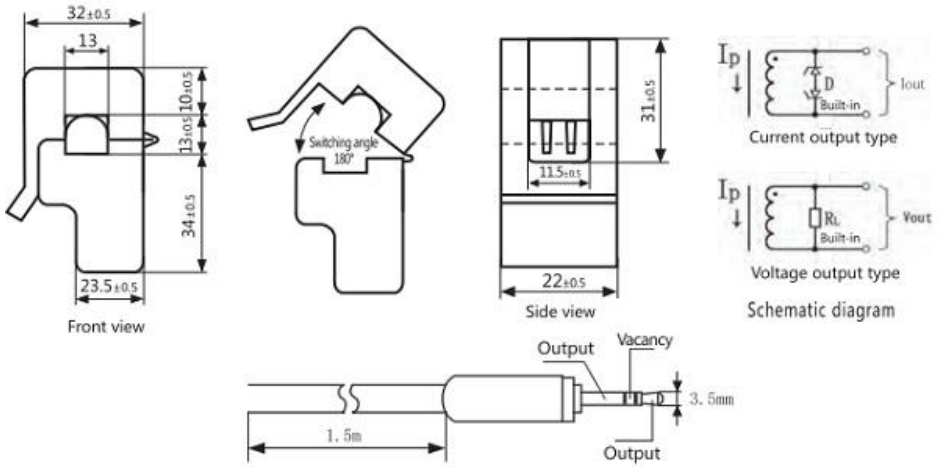

Fig. 4.24. Dimensions and schematic diagram of the SCT013 current sensor.

The sensor's output is a 3.5 mm audio jack, through which the secondary current flows.

The selected version was the SCT013-000, which allows for a maximum primary current of 100 A, thus inducing a maximum secondary current of 0.05 A.

This model does not include a built-in resistor, so an adaptation stage built with electronic components is necessary for it to be connected to the board.

4.6. Choice of the analogue to digital converter.

The output from the current sensor is analogic. As the Raspberry Pi board does not include analogic inputs, an analogue to digital converter (ADC) is needed.

The simulations on which the code was developed were executed using a sample time of 0.1 milliseconds. As the normalised voltage in Europe has a frequency of 50 Hz, equivalent to a period of 20 ms, a sample time of 1 ms is considered to be sufficient for the ADC.

The maximum input current, 100 A, should match the maximum voltage supplied by the ADC. As the ADC will be connected to a 3.3 V GPIO, this is the maximum voltage it will be able to provide.

In order to obtain a precision of 0.1 A in the measured current, the following method is applied to calculate the ADC's required resolution:

$$0.1 \cdot \frac{A}{Digital\ Value} = \frac{100\ A}{3.3\ V} \cdot X \frac{V}{Digital\ Value}$$

$$X = 0.0033 \cdot \frac{V}{Digital\ Value}$$

$$Resolution = 0.0033 \cdot \frac{V}{Digital\ Value} = 3.3\ V \cdot \frac{1}{Different\ values} \quad (4.1)$$

This results in a required amount of different values provided by the ADC of 1000.

With a converter able to provide an output of 10 bits, $2^{10} = 1024$ different output values are achievable.

The MCP3008 (Fig. 4.25) is a commonly used ADC that complies to these specifications, in terms of resolution and sample time.

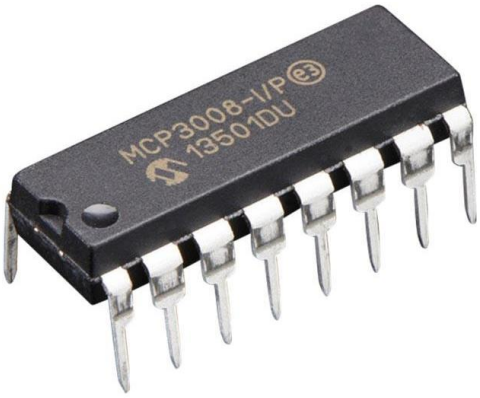

Fig. 4.25. MCP3008 analogue to digital converter.

This converter can be supplied with a voltage in the range of 2.7 to 5.5 V, where the output voltage of 3.3 V provided by the Raspberry Pi is included.

It has eight analogue inputs, numbered CH0 to CH7, and a Serial Peripheral Interface (SPI). These output ports will be the ones used to connect the ADC to the board.

For a single-phase installation, only one MCP3008 is needed, while for three-phase installations, two MCP3008 would be necessary, as each one of them includes two output channels that can be connected to the Raspberry Pi.

4.7. Adaptation stage for the current sensors.

The input pins on the Raspberry Pi board only support positive digital signals. When clamped to an AC device, the current sensor provides a sinusoidal signal with an amplitude 2000 times smaller than the measured current.

This signal has an offset of 0 V and cannot be directly used as the input for the MCP3008, as its output during the semi-period when the current is negative would not be properly interpreted by the board.

For this reason, an offset must be added to the input voltage of the MCP3008. As the ADC will be supplied 3.3 V from the board, the simplest method to achieve this purpose is to add 1.65 V DC to the input voltage of the MCP3008.

This can be done using a circuit similar to the one shown in Fig. 4.26.

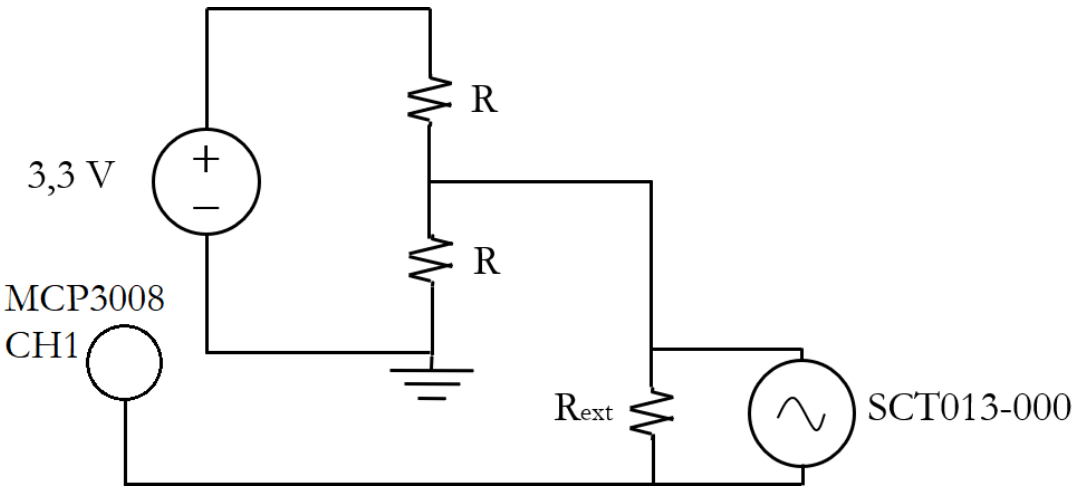

Fig. 4.26. Circuit for the input current's adaptation stage.

In the circuit shown, the two resistors R have the same value. Their purpose is to divide the 3.3 voltage supplied by the Raspberry Pi, so that on each one of them, 1.65 V drop. The point in between, thus, has a potential of 1.65 V, which is the offset added to the voltage generated in the current sensor.

The other resistor, R_{ext}, is the one where the voltage drop caused by the secondary current from the sensor occurs. This resistor should be scaled according to the estimated total power of the target installation.

Considering the maximum current supported by the current sensor, 100 A, in order to obtain a voltage drop of 1.65 V, which would mean the maximum input voltage of 3.3 V to the MCP3008 when added to the 1.65 V offset, $R_{ext} \approx 60.61\,\Omega$. These values should be checked, and modified if necessary, during the experimental stage.

4.8. Simulation of the electronic adaptation stage.

As the Raspberry Pi cannot measure negative voltages in its pins, the input signal must be in the range of 0 V and 3.3 V (the voltage selected as the reference voltage V_{Ref}).

An input signal connected to the input channel CH0 of the MCP3008 was simulated using the PicoScope 2204A, an USB oscilloscope with an integrated Arbitrary Waveform Generator (AWG).

This device is able to supply a peak-to-peak voltage of 4 V, and has a bandwidth of up to 10 MHz. It has two input channels A and B, and an output channel for the AWG, and is powered and controlled via USB using a PC and the PicoScope 6 software.

First, a sinusoidal signal with an offset of 1 V, and an amplitude of 1 V was configured as the output AWG. Then the probe connected to the output AWG, and the one connected to the input A were joined (Fig. 4.27).

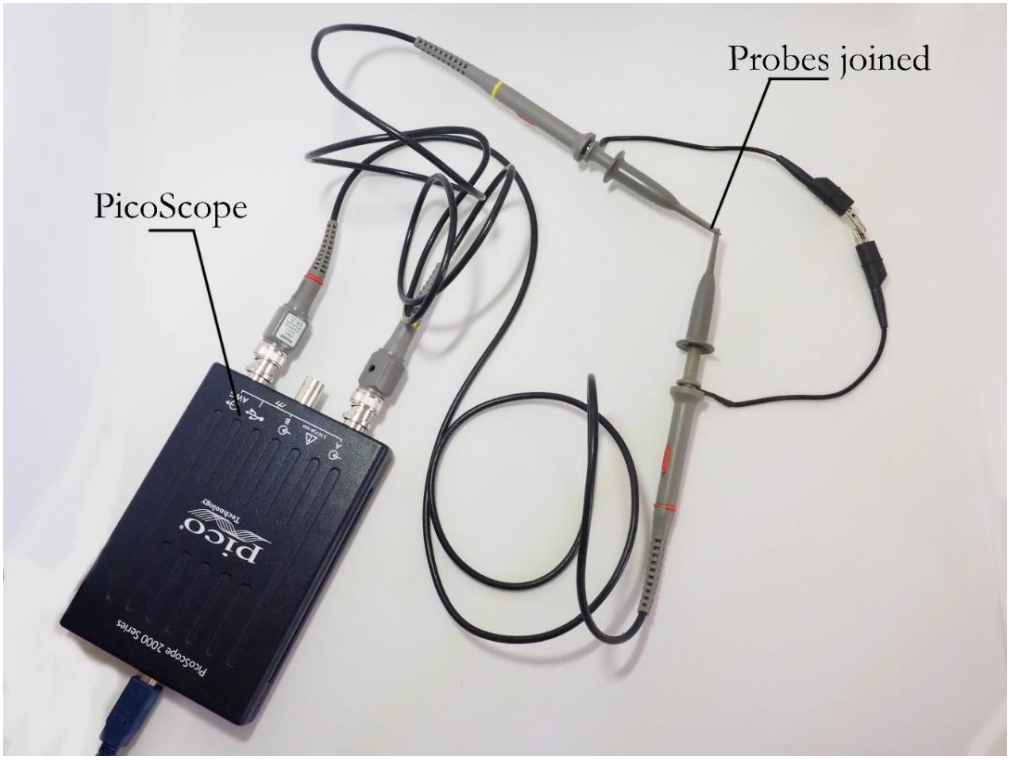

Fig. 4.27. Input and output of the PicoScope 2204A connected together.

This was merely done to ensure that the AWG generated the correct signal, and that such signal could be measured by the input A (Fig. 4.28).

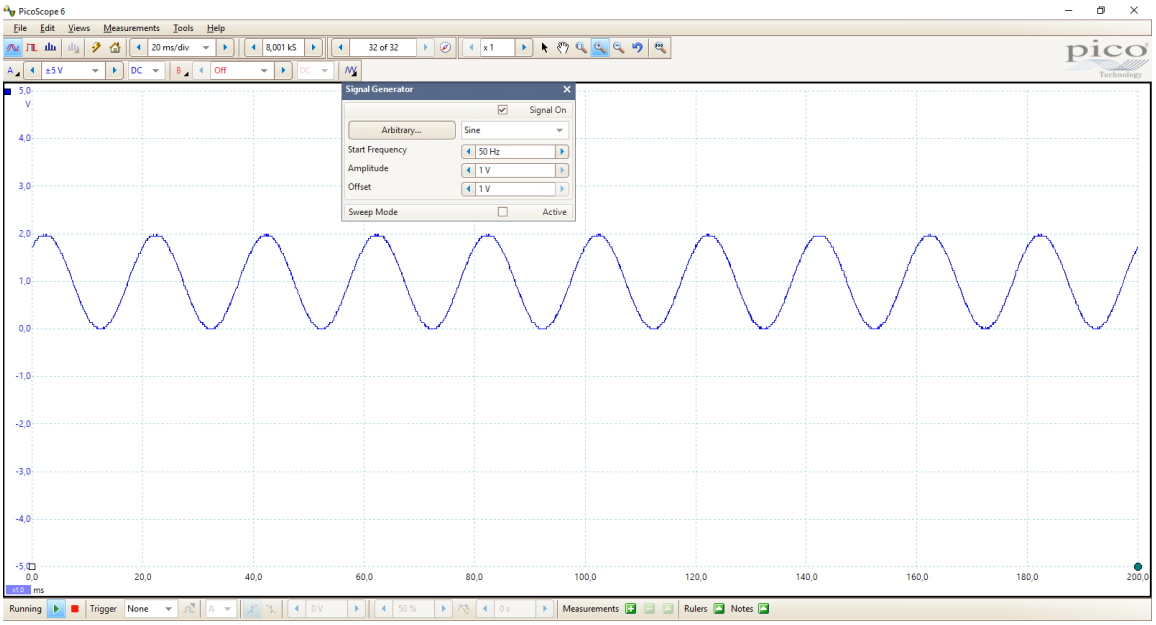

Fig. 4.28. Measurement from input A, coherent with the signal generated.

This test was repeated with various signal types, amplitudes and offsets, and alternating both channels as inputs. The results of these tests showed that the signal generated by the AWG can be measured by both input channels, meaning that all the device's ports worked properly.

The connection between Raspberry board and the MCP3008 was established as follows:

MCP3008	Raspberry Pi 3 Model B
V_{DD}	3.3 V (Pin 1)
V_{Ref}	3.3 V (Pin 17)
AGND	GND (Pin 9)
CLK	SPI0_CLK (Pin 23) GPIO 11
D_{OUT}	SPI0_MISO (Pin 21) GPIO 9
D_{IN}	SPI0_MOSI (Pin 19) GPIO 10
CS/SHDN	SPI0_CE0 (Pin 24) GPIO 8
DGND	GND (Pin 39)

Table 5.1. Connection between the MCP3008 and the Raspberry Pi board.

A schematic representation of these connections is shown in Fig. 4.29.

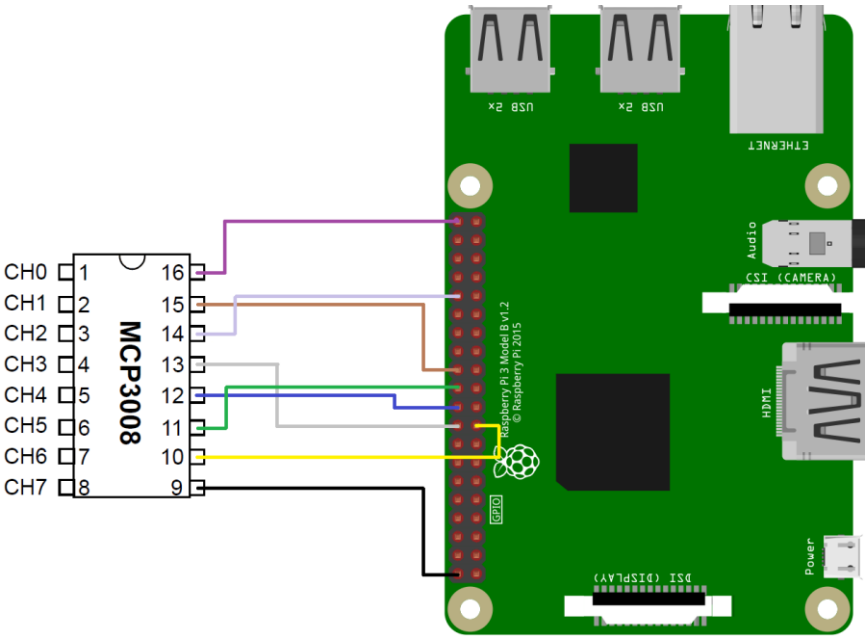

Fig. 4.29. Scheme of the connection between the MCP3008 and the Raspberry.

The positive terminal of the probe connected to the AWG of the PicoScope 2204A was connected to terminal CH0 of the MCP3008 ADC, and the probe's ground was connected to Pin 39 of the Raspberry Pi (Fig. 4.30).

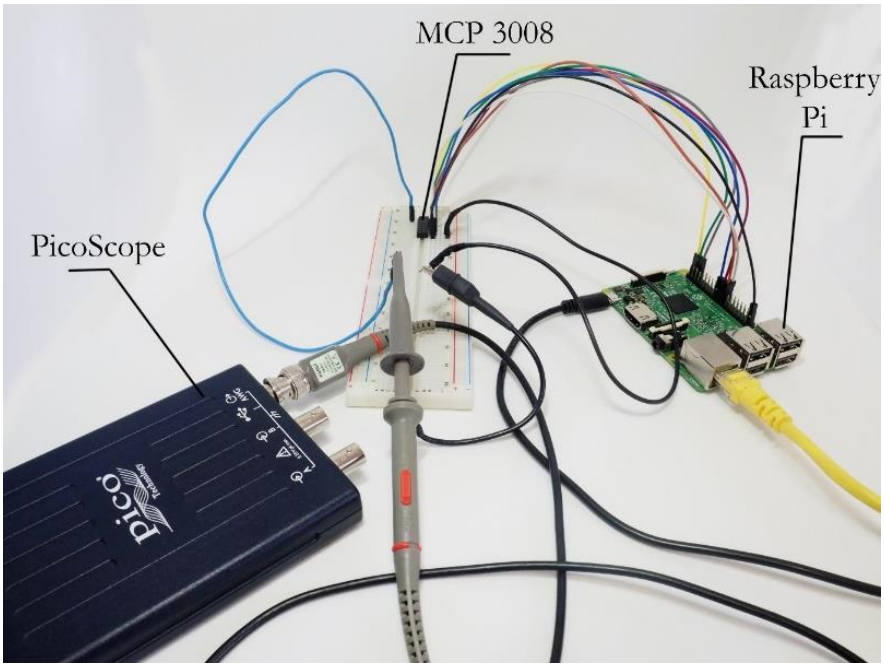

Fig. 4.30. AWG connected to MCP3008 ADC.

A sinusoidal signal was generated using the AWG. This signal would be equivalent to the one present after the adaptation stage.

In order to read the values from the MCP3008, the following code was added to the main programme.

```
from gpiozero import MCP3008
import time

pot=MCP3008(0)
while 1:
      i=pot.value
```

This code gets the value from Channel 0 of the MCP3008 using a built-in function from the 'gpiozero' library, and returns its value, saving it as variable 'i'.

This variable is later processed, serving as the current input for the SOGI-PLL.

After extensive testing, it was found that the Raspberry Pi Python compiler cannot read the values from the MCP3008 in real time. Instead, each value was obtained with a slightly different delay in each iteration.

5. Cost analysis.

The budget for this prototype can be divided in regard of the costs of the components, and the costs of the software and measuring tools used, and the total work hours.

Component	Unitary cost	Units	Total cost
Raspberry Pi Model 3 B+	36.30 €	1	36.30 €
MCP3008 ADC	2.06 €	2	4.12 €
SCT-013 current sensor	3.34 €	3	10 €
16 Gb SD card	4.49 €	1	4.49 €
Miscellaneous components	5 €	-	5€

Table 6.1. Costs of the components.

The total cost for components was 59.91 €.

Sotware or measuring tool	Total cost
MATLAB and Simulink	0 € (student´s license)
PicoScope 2204A	114.95 €

Table 6.2. Costs of the software and measuring tools.

Work hours	Hourly salary	Total salary
300	11.50 €/h	3450 €

Table 6.3. Costs derived from salaries.

The data about hourly salaries in Spain for a professional whose tasks resemble those developed in this project were obtained from Glassdoor's website.

6. Conclusions.

Regarding the theoretical basis of the project, it has been proved that the SOGI-PLL method developed and simulated is robust and achieves precise and quick results while analysing not only steady-state signals with or without harmonic components, but also signals where one or more of the variables vary during the analysis.

Different PLL architectures could be studied for situations requiring more precision or calculation speed, thus, increasing the software´s complexity.

The software used to develop and perform the preliminary tests on the algorithm, MATLAB Simulink, has been confirmed as a satisfying option for designing and evaluating transient functions, as well as for creating new algorithms which conform to different specifications.

However, MATLAB's dedicated operating system for Raspberry Pi, Mathworks, resulted inefficient for this application, as it was not possible to access the data collected by the MCP3008 analogue to digital converter from the Simulink environment. Hence, unless this issue is solved by the software's developer, the functionalities offered by Mathworks OS would be relegated to other applications, such as computer vision or GPS positioning, both of which have the support of dedicated Simulink modules.

The operating system that was finally installed on the Raspberry Pi, Raspbian OS, has served as the main frame for developing the algorithm on Python language. This language has a long history of synergy with the Raspberry environment, and so, includes a wide variety of libraries and functions for the interaction of both.

The main problem observed while testing the code built with Python in conjunction with the selected sensors and analogue to digital converter, was that the reading of the signals was not performed in real time. Future development of the project should focus on this last stage, as a real time OS or the translation of the Python code to other programming language could solve this issue.

The budget for the prototype, considering the board, SD cards, sensors and the ADC, falls short of 60 €. This does not take into consideration other tools used during the project, such as keyboards, the mouse or the screen, as they are commonly found items that were not specifically purchased for this project.

Increased capabilities could be added to the system, such as the ability to export the processed data to online repositories, where they would be available for their remote access from a PC or smartphone, for example. These advanced functionalities would of course increase the cost of the project.

7. References

[1] http://wibeee.circutor.com/index_en.html

[2] Blanco Charro, C. (2015). *Synchronization, islanding detection and power quality improvement in distributed power generation systems.* Gijón.

[3] Duesterhoeft, W., Schulz, M. W., & Clarke, E. (1951). Determination of Instantaneous Currents and Voltages by Means of Alpha, Beta, and Zero Components. In *Transactions of the American Institute of Electrical Engineers* (pp. 1248–1255).

[4] Park, R. (1929). Two Reaction Theory of Synchronous Machines. In *Transactions of the American Institute of Electrical Engineers* (pp. 716-730).

[5] N. Jaalam, N.A. Rahim, A.H.A. Bakar, ChiaKwang Tan, Ahmed M.A. Haidar (2016). A comprehensive review of synchronization methods for grid-connected converters of renewable energy source. *Renewable and Sustainable Energy Reviews 59 (2016)* (pp. 1471-1481).

[6] Şuşcă, M., & Dobra, P. (2016). Analog and Digital Notch Filter Implementation. *Technical University of Cluj-Napoca. Department of Automation.*

[7] Megretski, A. (2004). Multivariable control systems. *Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science.*

[8] Xiao-Qiang GUO, Wei-Yang WU & He-Rong GU (2011). Phase locked loop and synchronization methods for grid-interfaced converters: a review. *Yanshan University Electrical Review.*

[9] Blanco Charro, C., García Fernández, P., Navarro Rodríguez, Á., & Sumner, M. (2018). Predictive Frequency-Based Sequence Estimator for Control of Grid-Tied Converters Under Highly Distorted Conditions. *IEEE Transactions on Industry Applications, Vol. 54, No. 5*, 5306-517.