École Nationale d'Ingénieurs de Brest
DE CUYO

PPE 2019

# Implementation of an algorithm for monitoring electrical signals on a STM32 microcontroller

*Students:*

TORRES LÓPEZ, Mía

PÉREZ SERPA, Jerónimo

*Associate professors:*

V. CHOQUEUSE

JM. BOURGEOT

# Contents


**1 Introduction** **4**

**2 General board layout** **5**

**3 Signal conditioning** **5**

3.1 Voltage measurement . . . . . 5

3.1.1 Input side . . . . . 5

3.1.2 Output side . . . . . 6

3.1.3 Transfer equation . . . . . 9

3.2 Current measurement . . . . . 9

**4 Power supply** **10**

4.1 Dimensioning . . . . . 10

4.2 Battery voltage measurement . . . . . 11

**5 Board manufacturing** **11**

5.1 Board testing . . . . . 15

5.1.1 Voltage side . . . . . 15

5.1.2 Current side . . . . . 18

5.2 Current side redesign . . . . . 19

5.2.1 Transfer equation . . . . . 19

5.2.2 Testing . . . . . 19

**6 Signal sampling** **22**

**7 Estimation of the parameters of the signal** **24**

7.1 Context . . . . . 24

7.2 Objective . . . . . 25

7.3 Maximum Likelihood Estimator . . . . . 25

7.3.1 Estimation of the frequency . . . . . 25

7.3.2 Estimation of the amplitude and the phase . . . . . 27

7.4 Methodology . . . . . 27

7.4.1 Coarse search . . . . . 27

7.4.2 Refined search . . . 30
7.5 Implementation . . . 32

**8 Transmission of data towards the Cloud** **34**
8.1 Context . . . 34
8.1.1 Why AWS? . . . 34
8.1.2 Used services . . . 36
8.2 Objective . . . 37
8.3 Methodology . . . 37

**9 Conclusion** **42**
9.1 Future work . . . 43

**Appendices** **45**

**A Simulation parameters** **45**

**B Recurrence** **45**

# 1 Introduction

In high power electrical systems, the electrical signal is most often transmitted in three-phase form. In the ideal case, the three-phase current and/or voltage signals are described by three sinusoids of frequency $f_0 = 50Hz$ perfectly balanced with respect to each other (same amplitude and phasing of $2\pi/3$).

In practice, real three-phase signals can deviate significantly from this model:

- For three-phase electrical machines, the presence of an amplitude or frequency modulation of the stator current and/or an imbalance between phases may reveal the presence of an electrical or mechanical defect.
- For the electrical networks, the presence of a deviation from the fundamental frequency can reveal the presence of an imbalance between the production and the consumption of energy in one point of the network.

Therefore, the objective of this study is to develop a high power triphasic measurement unit oriented to smart grid applications, generators and motors diagnosis with the capability of wireless data transfer to the Amazon cloud web services.

We start with the hardware development for the signal reading and conditioning then we focus on the estimation of the parameters of the received signal using the microcontroller to perform signal processing algorithms to estimate the fundamental frequency as well as the value of the signal's amplitude and phase. We finish by sending the estimated values to a database hosted in the Amazon cloud service.

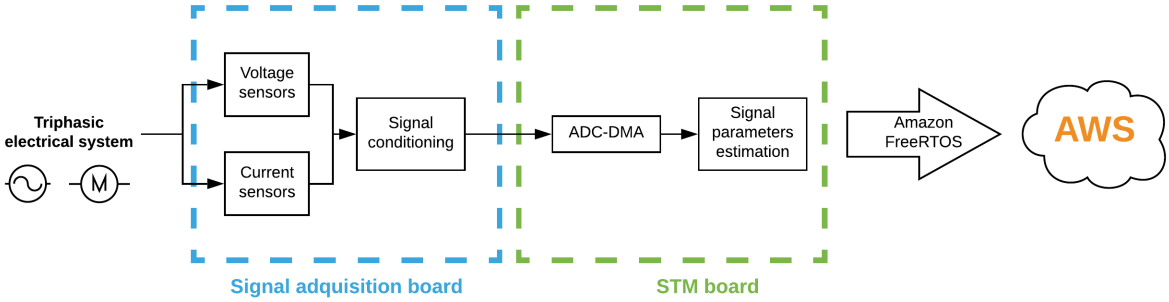



Figure 1: System scheme

This project is realized at the École Nationale d'Ingénieurs de Brest (ENIB) which provided the microcontroller used during this work, the SMT32L4 embedded in the B-L475E-IOT01A Discovery Kit IoT Node,

which integrates a WiFi module, is able to run the Amazon FreeRTOS OS and satisfies all the hardware requirements for this project.

Although the general objective is to develop a triphasic measurement unit, the testing and implementation of most of this work is treated for a single signal due to time limitations.

# 2 General board layout

As we have mentioned in the introduction (1), we need a circuit board to measure triphasic voltage and current,so we need to condition six signals (three for voltage between phases and three for current in each phase) with galvanic isolation for safety reasons, and it has to be able to communicate with the chosen microprocessor. Given the features needed we think that the best way to proceed is a shield card with Arduino™ Uno V3 connectivity support[1].

# 3 Signal conditioning

As we are treating with high voltage signals, a conditioning circuit is mandatory to archive the low voltage levels used in a microcontroller.

## 3.1 Voltage measurement

### 3.1.1 Input side

The development of the sensor board is the same for both the voltage and the current sensor so we are explaining it in depth for the $LV-25P$ voltage transducer.

We can see in the datasheet of the component that the input nominal current is $10mA$ with a maximum of $14mA$ so we need to dimension the input resistor in a way that the peak voltage $380V \times \sqrt{2} = 537.4V$ measured between phases produces around 10mA considering also the internal resistance, which is about $220\Omega$.

By Ohm law:

$$V_{in} = I_{in}(R_{internal} + R1) \tag{1}$$

$$R_{in} = \frac{V_{in}}{I_{in}} - R_{internal} = \frac{537,4V}{10mA} - 220\Omega = 53,5k\Omega \tag{2}$$

[1]This is one of the connector types supported by our development card

We have to use commercially available resistors so we use a $56k\Omega$ with 1% tolerance .With this resistor, the maximum input voltage that we could have (staying under de 14 mA max current ) would be:

$$V_{in_{max}} = I_{max} \times (R_{internal} + R_{in}) = 787V \tag{3}$$

The nominal input power dissipation is:

$$P = I_{in}^2 \times (R_{internal} + R1) = 5.6W \tag{4}$$

In the datasheet we can see that the current consumption is given by:

$$10(@15V)mA + I_s mA \tag{5}$$

**Electrical data**

| | | | | | |
|---|---|---|---|---|---|
| $I_{PN}$ | Primary nominal rms current | | 10 | | mA |
| $I_{PM}$ | Primary current, measuring range | | 0 .. ± 14 | | mA |
| $R_M$ | Measuring resistance | | $R_{M\,min}$ | $R_{M\,max}$ | |
| | with ± 12 V | @ ± 10 mA $_{max}$ | 30 | 190 | Ω |
| | | @ ± 14 mA $_{max}$ | 30 | 100 | Ω |
| | with ± 15 V | @ ± 10 mA $_{max}$ | 100 | 350 | Ω |
| | | @ ± 14 mA $_{max}$ | 100 | 190 | Ω |
| $I_{SN}$ | Secondary nominal rms current | | 25 | | mA |
| $K_N$ | Conversion ratio | | 2500 : 1000 | | |
| $U_C$ | Supply voltage (± 5 %) | | ± 12 .. 15 | | V |
| $I_C$ | Current consumption | | 10 (@ ± 15 V) + $I_S$ | | mA |

Figure 2: LV25-P Datasheet

$I_s$ is the nominal output current given a conversion ratio of 2500 : 1000 ($k_N$) so the current consumption will be arround:

$$10mA + 25mA = 35mA \tag{6}$$

### 3.1.2 Output side

As we are using a sensor that outputs current, we need a resistor to transform the current to voltage that is what the microprocessor measures. The measuring resistor $R_m$ is limited to a range of values that we can find in the datasheet and depends of the supply voltage of the sensor.

We know that the 12-bits ADC of the STM32 can measure voltages between 0 and 3.3V but we have alternating voltage in the input and current in the output so we need to adapt the amplitude and level of the signal. Half of the input range of the ADC is used for the positive part of the signal and the other half for the negative so the 25mA is transformed into 1.6 V.

For the signal conditioning we chose a combination of a voltage follower and an Op Amp attenuator with level shifting

$$R_m = 200\Omega \implies V_m = \pm 25mA \times 200\Omega = \pm 5V \tag{7}$$

And we transform this alternating signal into a positive value with a suitable range with the following circuit:

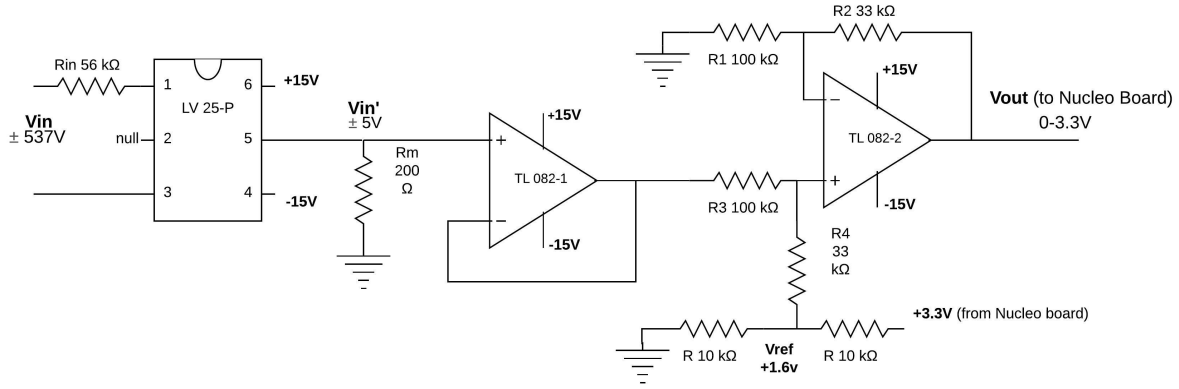



Figure 3: Signal conditioning circuit

To dimension the components of the attenuator circuit, we can start with the basic principles of an Op Amp:

$$V_- = V_+ \ \ \& \ \ I_+ = I_- = 0 \tag{8}$$

The input voltage of the attenuator is the output of the voltage follower and theoretically the same value at the input of it so we call it $V_{in}^{'}$.

Analyzing the non inverter line:

$$\frac{V_{in}^{'} - V_{ref}}{R_3 + R_4} = \frac{V_+ - V_{ref}}{R_4} \tag{9}$$

$$\implies V_+ = (V_{in}^{'} - V_{ref}) \times \frac{R_4}{R_3 + R_4} + V_{ref} \tag{10}$$

And for the inverter line:

$$\frac{V_{out}}{R_1 + R_2} = \frac{V_-}{R_1} \tag{11}$$

$$\implies V_{out} = V_- \times \frac{R_1 + R_2}{R_1} \tag{12}$$

Knowing that $V_+ = V_-$ we can find the relationship between $V_{in}^{'}$ and $V_{out}$:

$$V_{out} = \left[ (V_{in}^{'} - V_{ref}) \times \frac{R4}{R_3 + R_4} + V_{ref} \right] \times \frac{R_1 + R_2}{R_1} \tag{13}$$

And analyzing the equation for the two extreme cases:

If $V_{ref} = 0$:

$$\frac{V_{out}}{V_{in}^{'}} = \frac{R_4}{R_1} \times \frac{R_1 + R_2}{R_3 + R_4} = A \tag{14}$$

Being $A$ the gain of the signal.

If $V_{in} = 0 \rightarrow V_{in}^{'} = 0$:

$$\frac{V_{out}}{V_{ref}} = \frac{R_3}{R_1} \times \frac{R_1 + R_2}{R_3 + R_4} = A_{Vref} \tag{15}$$

Being $A_{Vref}$ the gain of the offset voltage.

We can see here that the relationship $\frac{R_1+R_2}{R_3+R_4}$ is affecting both gains in the same way, and if we choose the resistors values in a way that $\frac{R_1+R_2}{R_3+R_4} = 1$ we can simplify the attenuator transfer equation.

$$V_{out} = V_{in}^{'} \times \frac{R4}{R1} + V_{ref} \times \frac{R_3}{R_1} \tag{16}$$

As we know that the input range is $10V$ and the output must be $0 - 3.3V$, the gain should be:

$$A_v = \frac{33}{100} = \frac{R_4}{R_1} \tag{17}$$

So we will use:

$$R_1 = 100k\Omega, R_4 = 33k\Omega, R_2 = 33k\Omega, R_3 = 100k\Omega$$

With this setup $A_{Vref} = \frac{R_3}{R_1} = 1$ but we don't have a source of +1.6V so we will use a voltage divider and the +3.3V output of the Nucleo board.

### 3.1.3 Transfer equation

Knowing each part of the circuit and the relationship between the outputs and inputs we can get together the complete transfer equation:

$$V_{out} = A_v \times R_m \times K_n \times \frac{V_{in}}{R_{in}} + V_{ref} \tag{18}$$

$$= \frac{R4}{R1} \times R_m \times K_n \times \frac{V_{in}}{R_{in}} + V_{ref} \tag{19}$$

$$= \frac{33 \times 10^3}{100 \times 10^3} \times 200 \times \frac{2500}{1000} \times \frac{V_{in}}{56 \times 10^3 \Omega} + 1.6V \tag{20}$$

$$= \frac{33}{11200} \times V_{in} + 1.6V \tag{21}$$

## 3.2 Current measurement

The LA-25P transducer also works as a current source so the only difference in this circuit is the input range of the sensor,being the input nominal current $\pm 25A$.

With a gain $K_N = \frac{1}{1000}$, the nominal output current will be $25mA$ and using the same circuit equations already explained in section 3.1.1 we can obtain the transfer equation.

**Electrical data**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $I_{PN}$ | Primary nominal rms current | | 25 | | | | A |
| $I_{PM}$ | Primary current, measuring range | | 0 .. ± 55 | | | | A |
| $R_M$ | Measuring resistance | | $T_A$ = 70 °C | | $T_A$ = 85 °C | | |
| | | | $R_{M\,min}$ | $R_{M\,max}$ | $R_{M\,min}$ | $R_{M\,max}$ | |
| | with ± 12 V | @ ± 25 $A_{max}$ | 10 | 280 | 60 | 275 | Ω |
| | | @ ± 55 $A_{max}$ | 10 | 80 | 60 | 75 | Ω |
| | with ± 15 V | @ ± 25 $A_{max}$ | 50 | 400 | 135 | 395 | Ω |
| | | @ ± 55 $A_{max}$ | 50 | 140 | 135 | 135 | Ω |
| $I_{SN}$ | Secondary nominal rms current | | 25 | | | | mA |
| $K_N$ | Conversion ratio | | 1 : 1000 | | | | |
| $U_C$ | Supply voltage (± 5 %) | | ± 12 .. 15 | | | | V |
| $I_C$ | Current consumption | | 10 (@ ± 15 V) + $I_S$ | | | | mA |

Figure 4: LA25-P Datasheet

Following the same procedure as in section 3.1.3, we obtain the following equation:

$$V_{out} = A_I \times R_m \times K_n \times I_{in} + V_{ref} \tag{22}$$

$$= \frac{R4}{R1} \times R_m \times K_n \times I_{in} + V_{ref} \tag{23}$$

$$= \frac{33 \times 10^3}{100 \times 10^3} \times 200 \times \frac{1}{1000} \times I_{in} + 1.6 \tag{24}$$

$$= \frac{33}{500} \times I_{in} + 1.6 \tag{25}$$

# 4 Power supply

## 4.1 Dimensioning

The hole system energy supply will be a 12V battery from which we will generate the different voltage levels needed. As this is a stand-alone system which could be used in a remote location, a battery level monitoring circuit will be included and explained later.

Table 1: Power supply distribution

| Power supply distribution | | | |
|---|---|---|---|
| Voltage (V) | Component | Quantity | Current consumption(mA) |
| ±15 | LA 25-P | 3 | 35 |
| | LV 25-P | 3 | 35 |
| | TL082 | 6 | 3.6 |
| +15 | TL082 | 1 | 3.6 |
| +5 | STM32 board[11] | 1 | $< 500$ |
| +3.3 | Vref[1] | 1 | $\approx 0$ |

[1] Obtained from the STM32 board.

We can see that the most power requiring supply is the one of $\pm 15V$ with almost 7 watts needed so we have chosen a Tracopower TEN 8-1223 8W DC-DC converter, and for the 5V supply a TEC 2WI. Both converters are directly connected to the battery. For the 3.3V supply we will use the STM32 voltage output because the power consumption is negligible.

### 4.2 Battery voltage measurement

If the system is installed in a remote location, a low battery alert sent to the cloud is mandatory to assure continuous monitoring of the network or machinery. For this purpose we are including a simple circuit to measure the battery voltage with an analog input of the microprocessor.

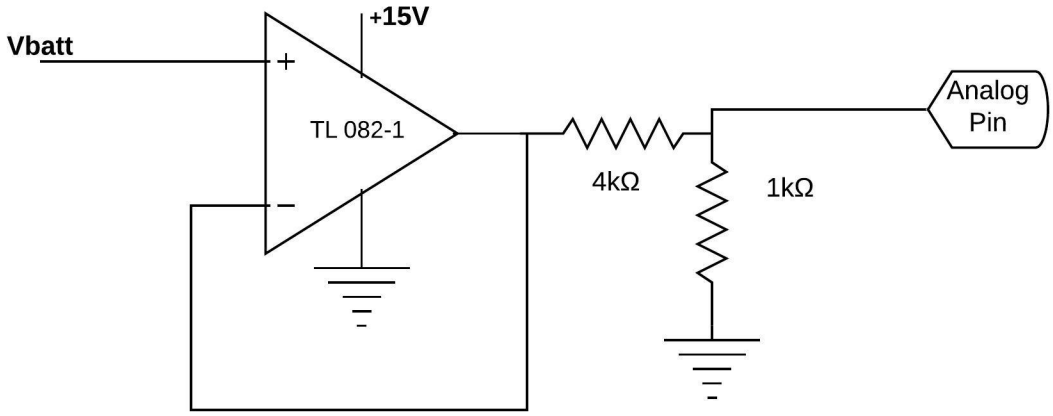



Figure 5: Battery measurement circuit

## 5 Board manufacturing

For the prototyping process we will be using Eagle PCB design software and UV lithography. The first version of the board had some troubles so we had to make minor changes and try again.

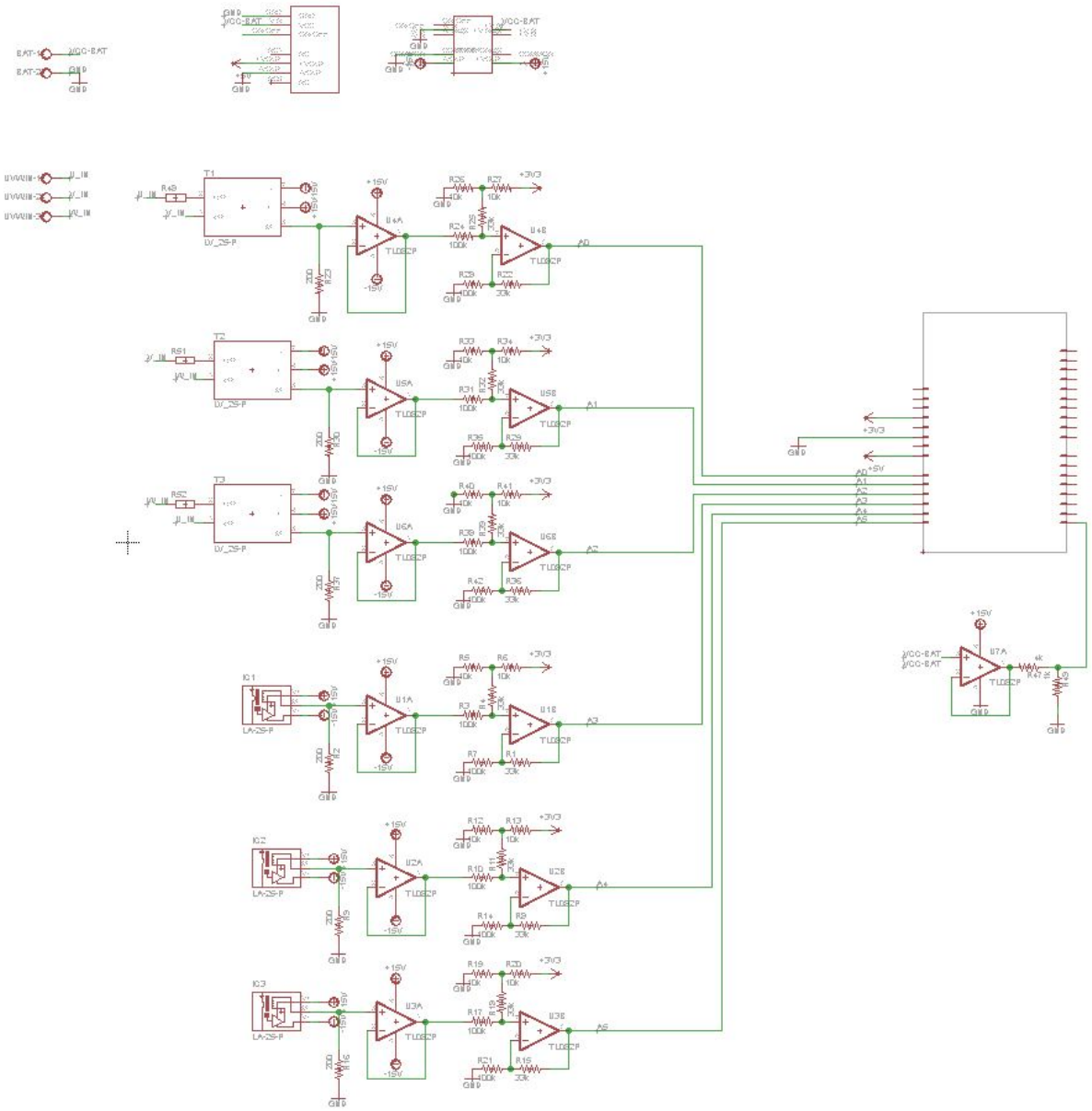
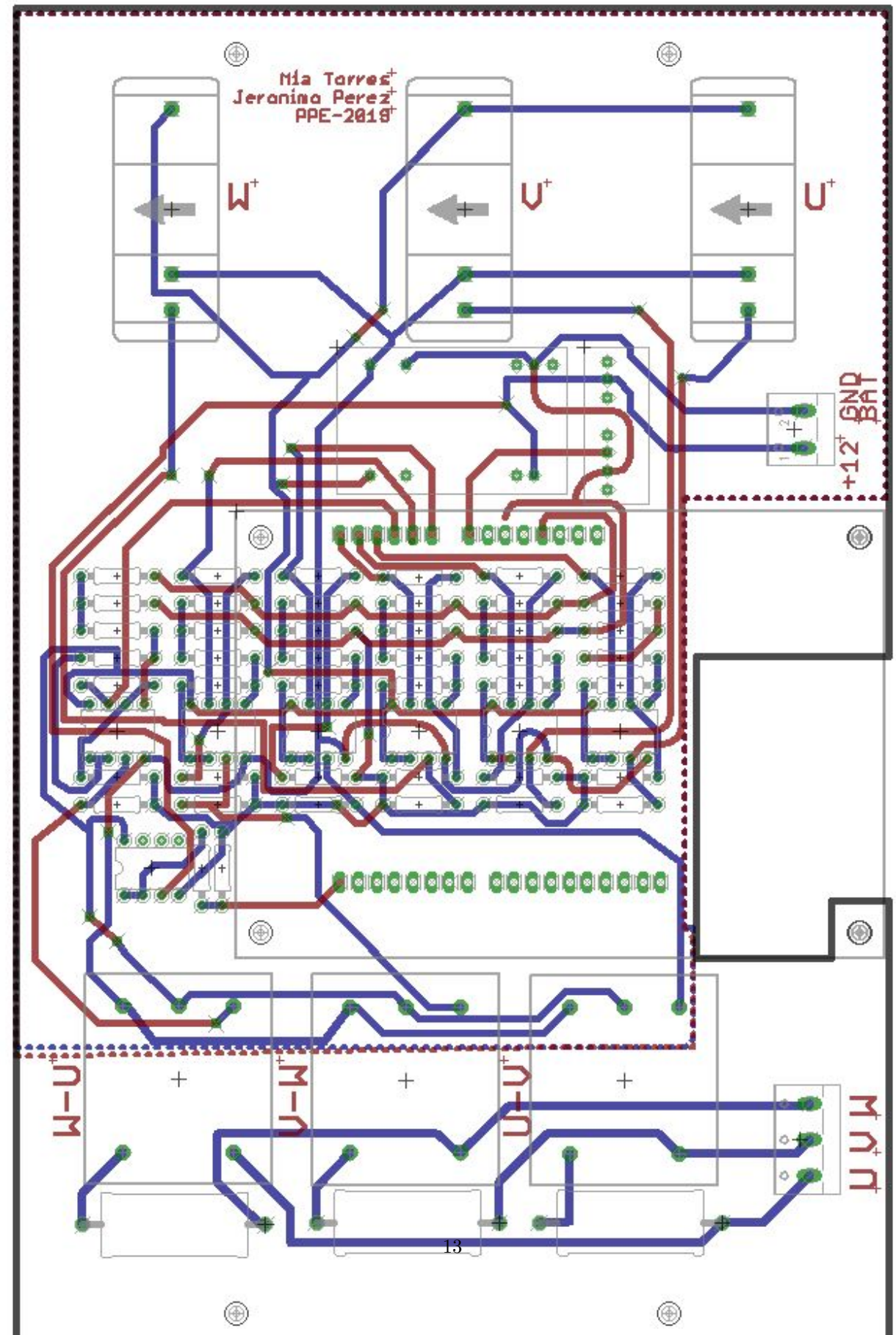
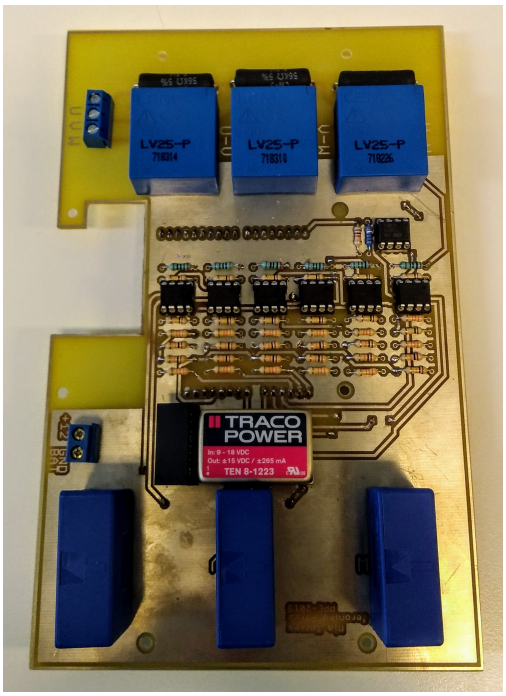
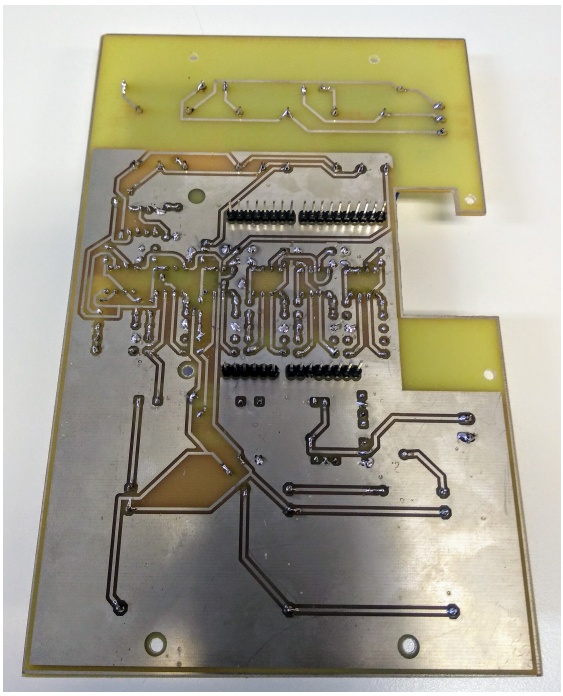


Figure 6: Schematic


Mia Torres
Jeranima Perez
PPE-2018
W
V
U
+12
GND
BAT
W-U
V-W
U-V
W V U




(a) Front side



(b) Back side

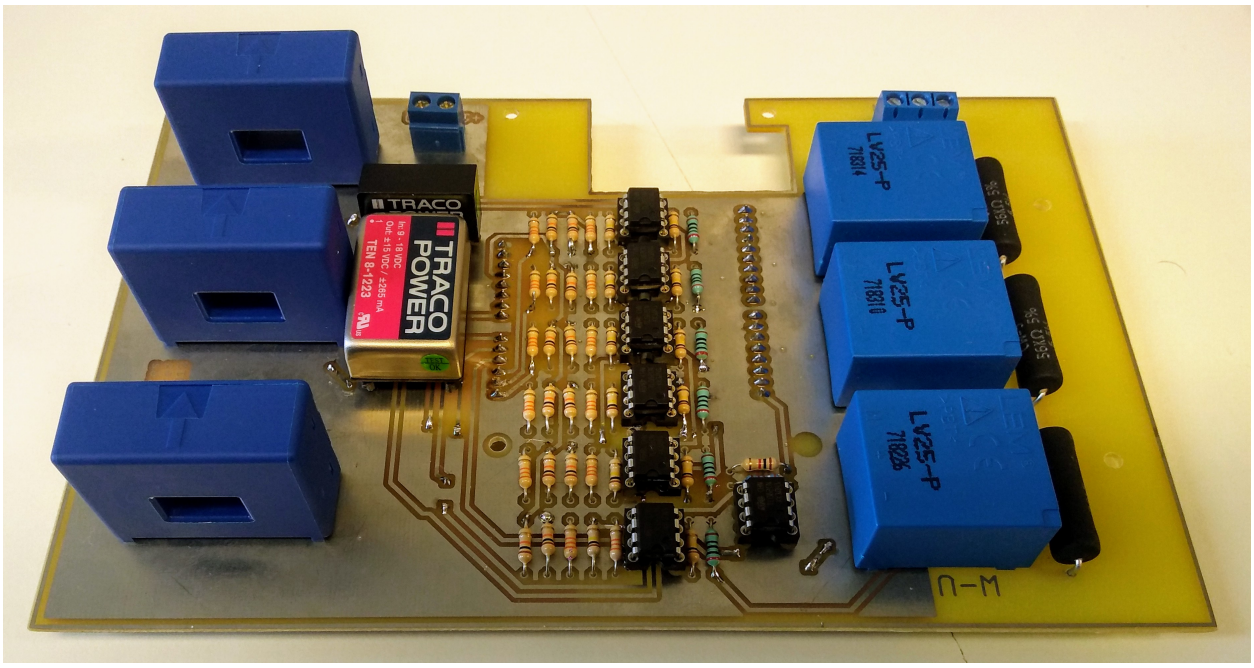



Figure 8: Front side horizontal

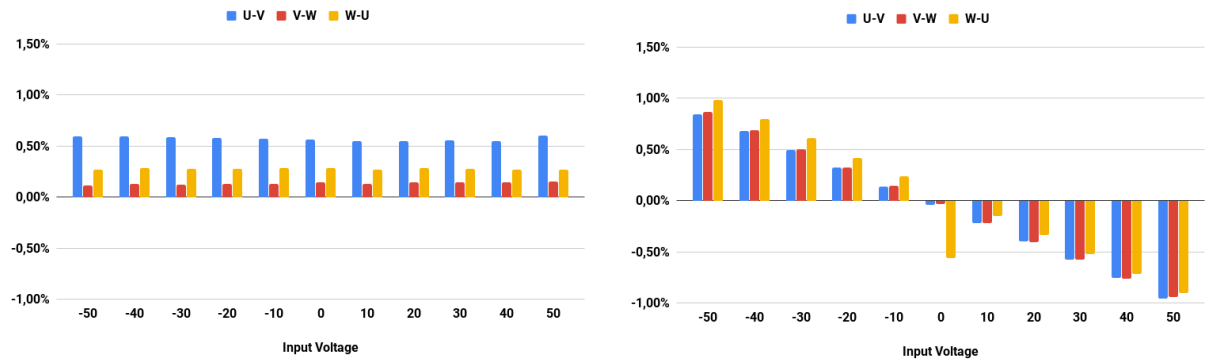
## 5.1 Board testing

All the tests are made with $V_{ref} = 0$, because this tension is supplied by the STM board which is not connected in the tests for safety reasons[2]. We connect $V_{ref}$ directly to ground.

### 5.1.1 Voltage side

To test the voltage measurement circuit, we use a bench power supply with adjustable voltage and current output. Even when this is not the ideal test scenario given the low voltages used, it's good enough to have an idea of the general performance.

We test each phase with continuous voltage in the range of $\pm 50V$ in 10V steps measuring the sensor output $Vin'$, the voltage follower output $Vfollower$ and the attenuator circuit output $Vout$ to calculate the error percentage (relative to the maximum possible value) between the theoretical and the measured values in each part of the signal conditioning process.




(a) $V_{in}^{'}$ sensor circuit isolated

(b) Signal attenuator circuit isolated.

Figure 10: Error percentage relative to the measurement range in the two main parts of the circuit.

[2]We have to be sure that there are no voltages higher than 3.3V in the board inputs.

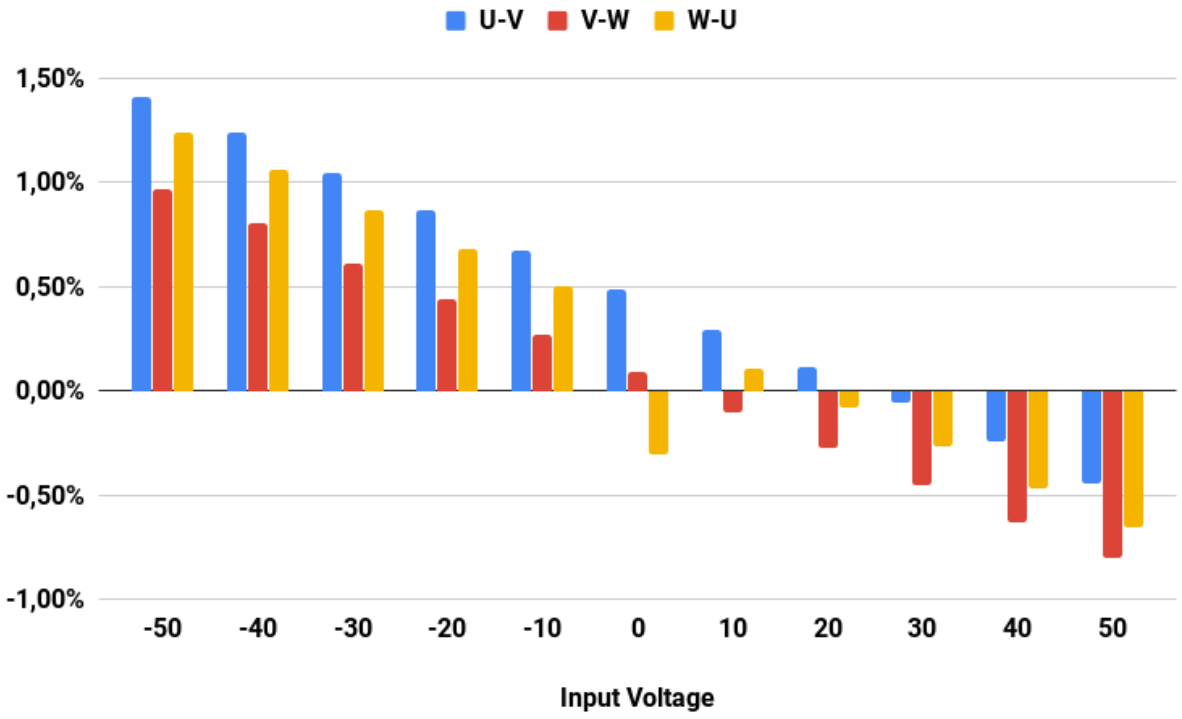



Figure 11: $V_{out}$ error percentage relative to the measurement range

When the two main parts of the circuit are isolated, we can see clearly the error contribution of each one. When added this two graphs (see figure 10a and 10b) and depreciating the voltage follower circuit error which is minimal (see figure 13) we obtain approximately the $V_{out}$ error percentage chart (see figure 11). Here we can see that in some cases error is compensated between each part and in others it is increased, this explains the asymmetry in the $V_{out}$ error percentage chart. This kind of errors are probably caused by differences between resistors values, given that all the resistors used in the board have a 5% tolerance. To test this hypotheses the maximum theoretical error is calculated by taking the worst case scenario of the resistors values variation, calculating with the transfer equation the correspondent theoretical output values and plotting them together with the $V_{out}$ values measured in each phase.

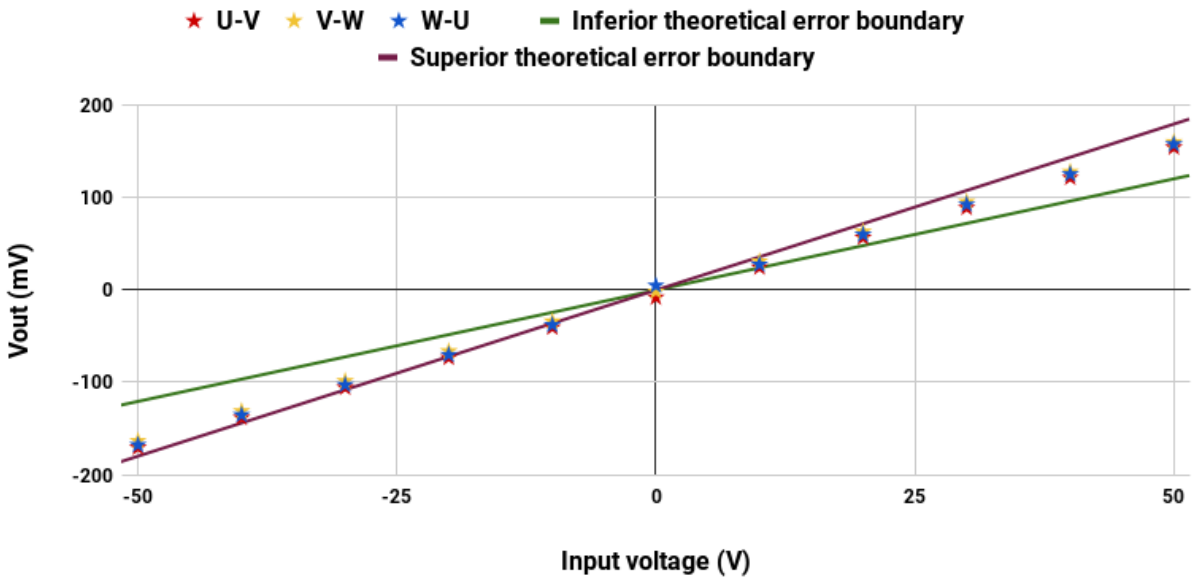



Figure 12: $V_{out}$ Theoretical error boundaries.

We can see that the error is inside the expected values so this part of the circuit behaves as expected and no changes are needed. In future iterations, it will be necessary to evaluate if error values are low enough for signal analysis to work properly.

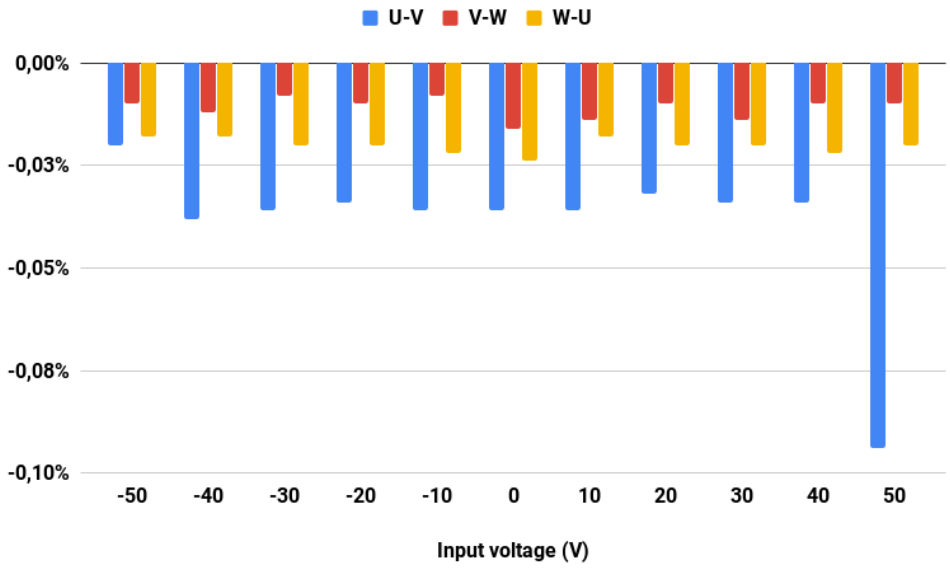



Figure 13: Voltage follower error.

### 5.1.2 Current side

In this test we use the current control feature of the bench power supply to feed each phase with a continuous current in a range of 0-2.5A with 0.5A steps.

newline

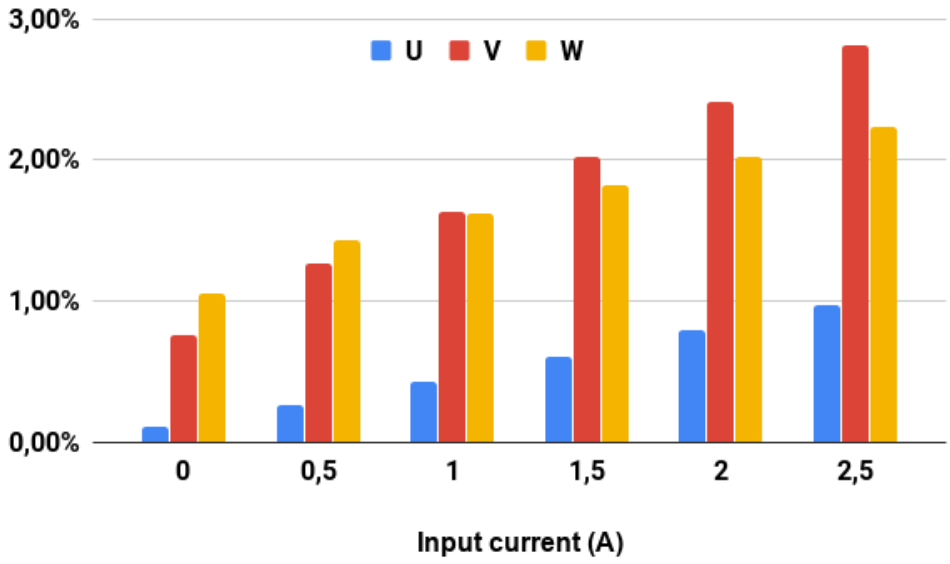
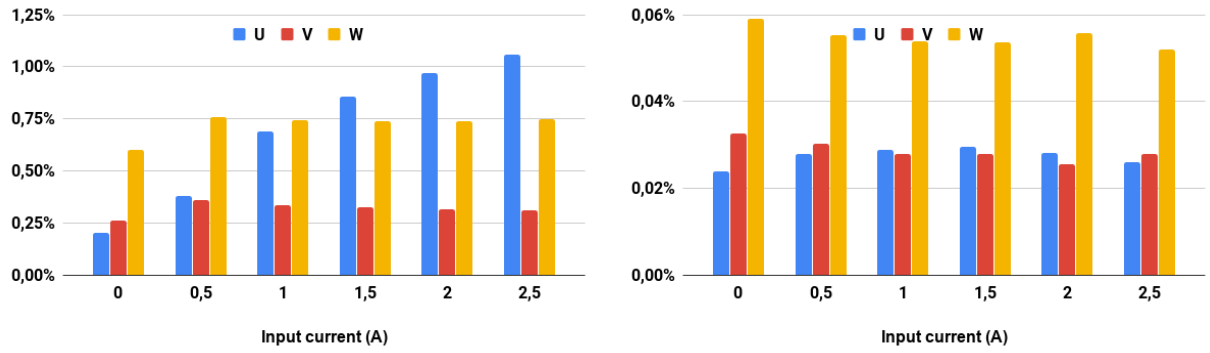



Figure 14: $V_{out}$ error percentage relative to the measurement range




(a) (b)

Figure 15

With an initial analysis of the results showed in the charts (see figures 14, 15a and 15b we see that the error in the output of the sensor and in the output of the signal conditioning circuit is already much higher

than the one in the voltage measurement part, and after revising the circuit design we have come to the conclusion that the range chosen is too big for the available test equipment, so we proceed to make a redesign of this part.

## 5.2 Current side redesign

After a reevaluation of the circuit, we figured that the $\pm 25A$ measuring range was too high so we change it to a $\pm 5A$ range and redesign the signal conditioning circuit to work with higher voltages, making it less susceptible to noise and errors in the resistors values.

### 5.2.1 Transfer equation

The equations developed in the section 3.2 are still valid but we are changing some resistors values. As our maximum output current of the sensor is $\pm 5mA$, the measuring resistor $R_m$ is $2k\Omega$. With this configuration the maximum $V_{in}^{'}$ voltage is $\pm 10V$ and as this value is doubled, the gain of the attenuator circuit is halved so we double the R1 and R3 values, being the new transfer equation:

$$V_{out} = A_I \times R_m \times K_n \times I_{in} + V_{ref} \tag{26}$$

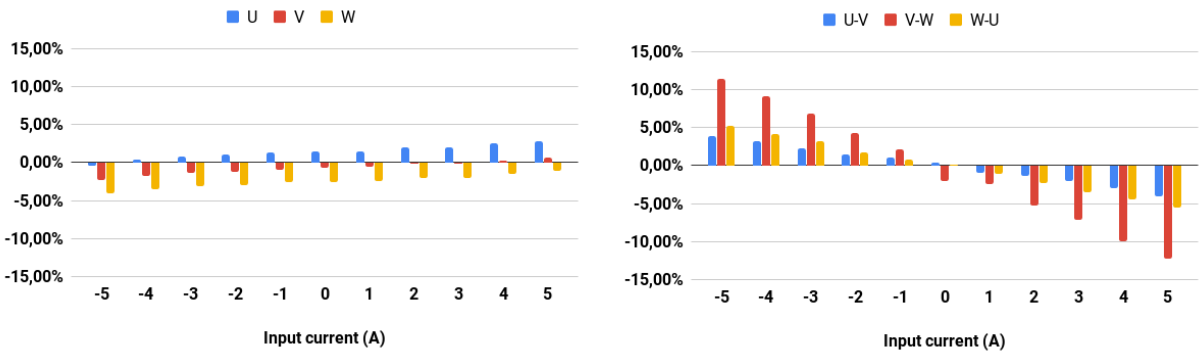
$$= \frac{R4}{R1} \times R_m \times K_n \times I_{in} + V_{ref} \tag{27}$$

$$= \frac{33 \times 10^3}{200 \times 10^3} \times 2000 \times \frac{1}{1000} \times I_{in} + 1.6 \tag{28}$$

$$= \frac{33}{200} \times I_{in} + 1.6 \tag{29}$$

### 5.2.2 Testing

We test each phase with continuous current in the range of $\pm 5A$ with 1A steps measuring the voltage in each section of the board, as in section 5.1.1 and calculate the error percentage (relative to the maximum possible value) between the theoretical and the measured values.




(a) $V_{in}^{'}$ sensor circuit isolated

(b) Signal attenuator circuit isolated.

Figure 16: Error percentage relative to the measurement range in the two main parts of the circuit.

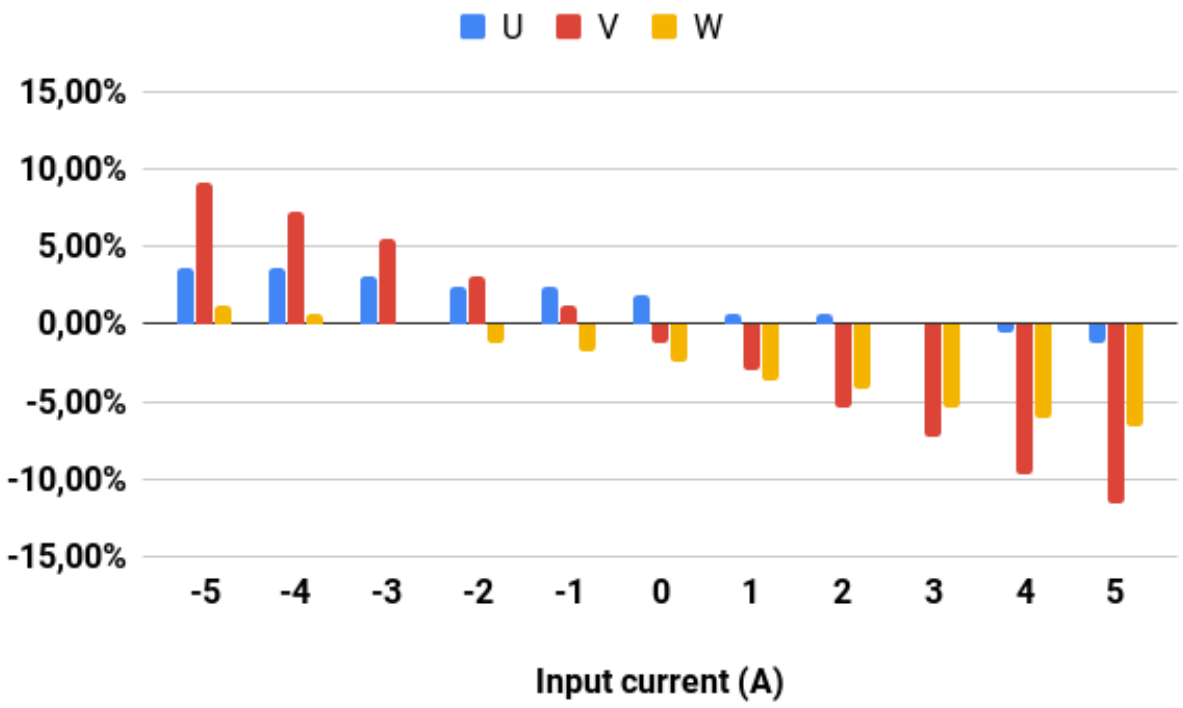



Figure 17: $V_{out}$ error percentage relative to the measurement range

As we saw in section 5.1.1 adding up this two graphs (see figure 16a and 16b) and depreciating the voltage follower circuit error which is even lower than in the first version due to the higher voltages used (see figure 19) we obtain approximately the $V_{out}$ error percentage chart (see figure 17). Following the same procedure

that in section 5.1.1 the values corresponding to the maximum theoretical error are calculated and plotted together with the total $V_{out}$ values for each phase.

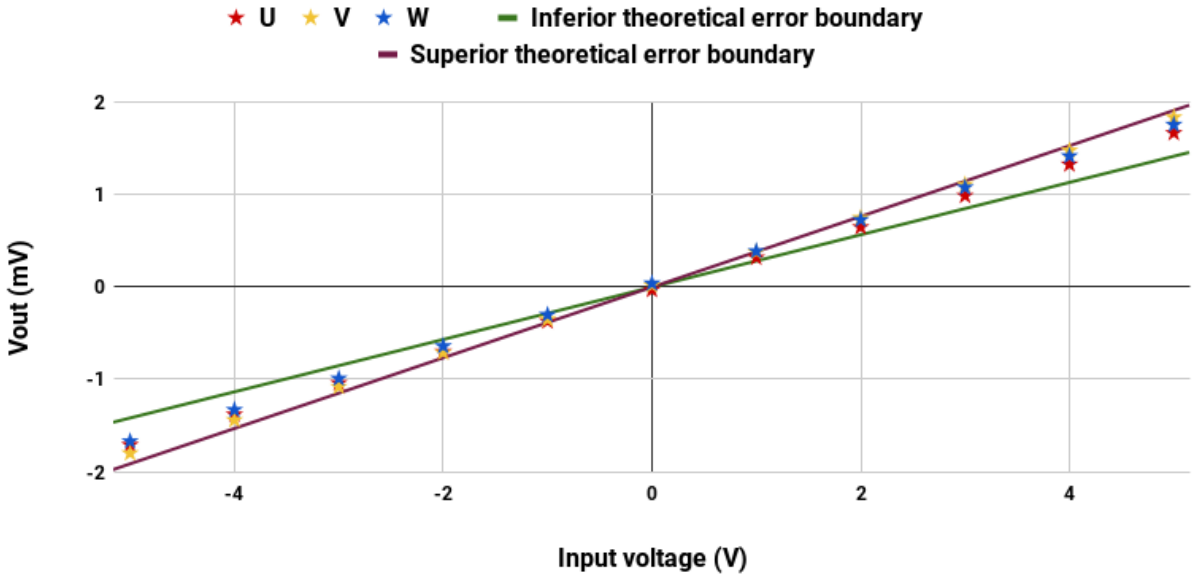



Figure 18: $V_{out}$ Theoretical error boundaries.

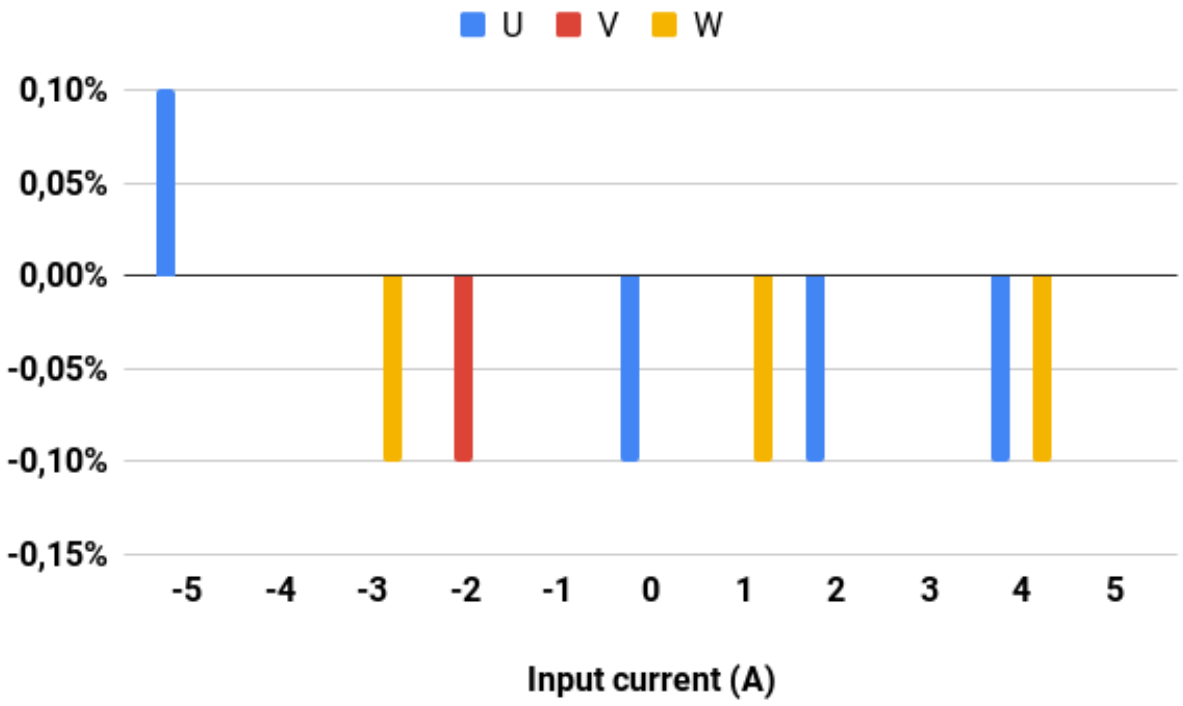



Figure 19: Voltage follower error.

We can see that the error is inside the expected values (see figure 18 ) so this part of the circuit behaves as expected. Even when the error values seem to be higher, this is the error relative to the maximum value measurable so for example, a 10% error over 5A it's equal to 0.5A, while with the previous design the highest error value was around 3% over 25A which means 0.75A so the circuit behaviour has been indeed improved.

# 6 Signal sampling

In the STM32L4 microprocessor we have two ADC units with 12 bits resolution and a maximum sampling rate of 2 Msps in DMA mode. As our signal has a low frequency, the sampling rate covers our needs. We will be using the direct memory access mode "DMA" [12] of the ADC to leave the microprocessor free to execute the signal processing algorithm. To choose the sample frequency there are two main requirements to take in count. The first one is to respect the Nyquist-Shannon theorem, being $F_s$ the sampling frequency and $F_{max}$ the highest frequency component of the analog signal to read:

$$F_s > 2 \times F_{max} \tag{30}$$

With a 50Hz frequency in the grid (60Hz in some countries) our minimum sampling rate is 100Hz, that is a easily achievable frequency so we can focus in the other limitation that is the number of samples to be analyzed. With a high number of samples, the performance of the frequency estimator algorithm could increase but also the time needed for it's execution [13]. We will stay with a low number of samples for the development but further testing will be needed later to determine which is the optimal number.

The ADC configuration has four parameters that determines the sampling rate:

- Clock source
- Clock prescaler
- Conversion time[3]
- Resolution

Being the sampling frequency determined by the following equation:

$$F_s = \frac{ADC_{CLK}}{ADC_{Prescaler} \times (T_{Conversion} + ADC_{Resolution})} \tag{31}$$

The minimum configurable $ADC_{CLK}$ frequency is 8MHz with an $ADC_{Prescaler} = 32$, the default $T_{Conversion}$ is 2.5 clock cycles but configurable up to 640.5 cycles. With the 12 bits resolution, the lowest sampling frequency will be 383.14Hz , this is more than enough to meet the Nyquist-Shannon condition and will give us a reasonable number of samples to deal with. For example, with $F_s = 383.14Hz$ being N the number of samples and a 10 periods sample time:

$$N = \frac{1}{60Hz} \times 10 \times 383.14 \tag{32}$$

$$= 95.785 \implies 95^{4} \tag{33}$$

We will see in depth the time needed for the signal analysis in the section 7.5.

[3] In the configuration tab of CubeMX, *ADC configuration* → *Parameter Settings* → *Rank* → *Sampling time*.

[4] We round down the value because the number of complete samples is the one that we find in the memory. If the conversion is not completed, the value is not saved

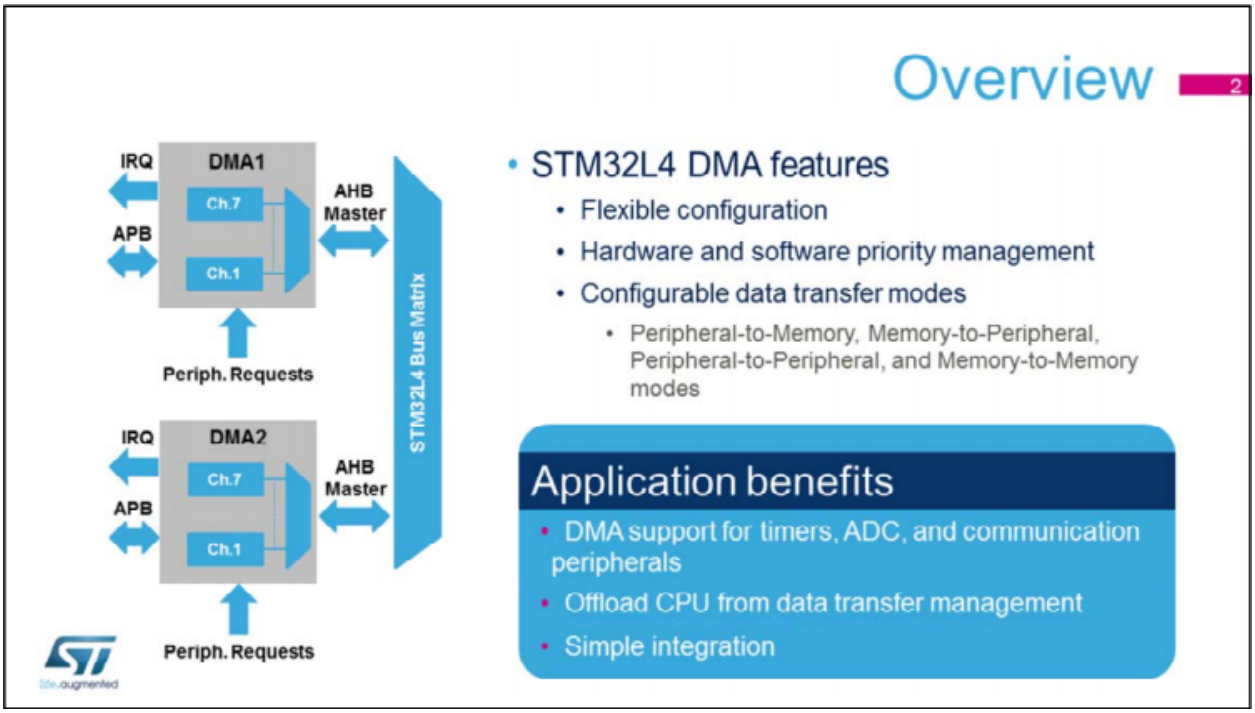



Figure 20: DMA

# 7 Estimation of the parameters of the signal

## 7.1 Context

For this study, we assume that the signal can be described by a sinusoidal wave corrupted by an additive noise. Mathematically, the signal can be expressed as

$$x[n] = a.\cos(2\pi f \times \frac{n}{Fs} + \phi) + b[n] \tag{34}$$

where $f$ is the signal frequency, $a$ and $\phi$ are the amplitude and phase of the signal, $Fs$ is the sampling rate (number of samples by second) and $b[n]$ is the additive noise component.

The additive noise component $b[n]$ is modelled as a zero-mean Gaussian noise with variance $\sigma^2$ i.e. $b[n] \sim \mathcal{N}(0, \sigma^2)$ (Here we use the notation $\sim$ to indicate that $b[n]$ "is distributed according to" $\mathcal{N}(0, \sigma^2)$). The probability density function (PDF) is then described as

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}x^2}. \tag{35}$$

## 7.2 Objective

In this section, we show how to estimate the parameters $a$, $f$ and $\phi$ from N samples of x[n]. The development and testing of this section we use a signal simulated via software.

Throughout this section it's important to keep in mind that in signal processing applications there's always a trade-off between:

- statistical performance
- computational complexity
- robustness (ability of the algorithm to estimate a parameter accurately under a wide range of scenarios)

## 7.3 Maximum Likelihood Estimator

To estimate these parameters, a natural solution is to minimize the Least Squares (LS) cost function, given by

$$\{\widehat{f}, \widehat{a}, \widehat{\phi}\} = \arg\min_{f,a,\phi} \sum_{n=0}^{N-1} \left| x[n] - a\cos(2\pi f \times \frac{n}{Fs} + \phi) \right|^2 . \tag{36}$$

This estimator, called the LS estimator, corresponds to the Maximum Likelihood (ML) estimator when $b[n]$ is modelled as a zero-mean Gaussian noise with variance $\sigma^2$ [9].

In this initial form, the estimation of $f$, $a$ and $\phi$ requires the minimisation of a cost function with respect to three parameters. In practice, this minimisation can be challenging. Nevertheless, it has been demonstrated in [10] that the multidimensional minimisation can be decomposed into 2 sequential steps:

- Estimation of the frequency $f$
- Estimation of the amplitude $a$ and the phase $\phi$

### 7.3.1 Estimation of the frequency

The frequency $f$ is estimated as:

$$\widehat{f} = \arg\max_{f} \mathcal{C}(f) \tag{37}$$

where the cost function $\mathcal{C}(f)$ is

$$\mathcal{C}(f) = \beta(f)\gamma(f). \tag{38}$$

The definitions of $\beta(f)$ and $\gamma(f)$ are available in [7] for three-phase systems and in [6] for a single phase, where the equations are developed for the angular frequency $\omega$, which is related to the frequency $f$ by

$$\omega = 2\pi \times \frac{f}{Fs}. \tag{39}$$

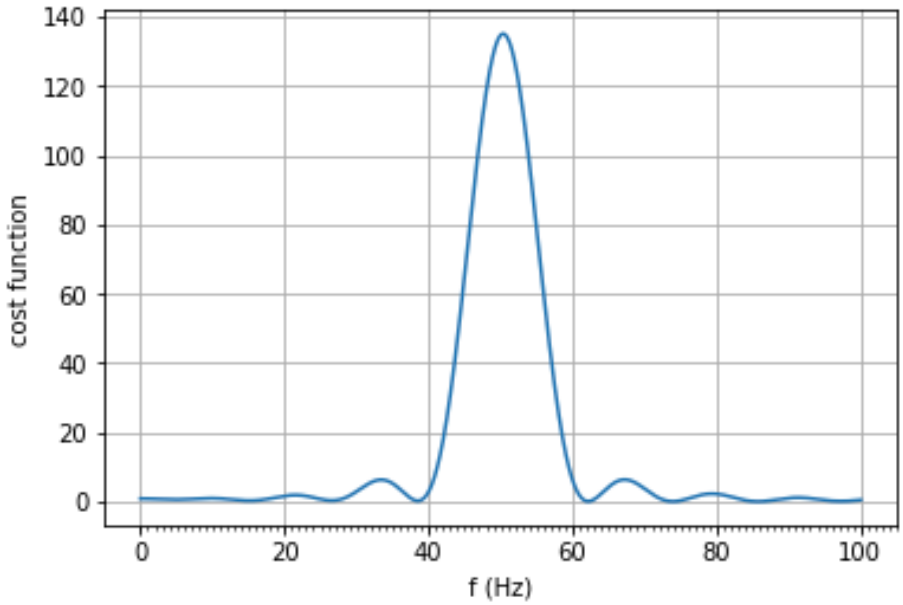



Figure 21: Cost function

The figure 21 represents the general form of the cost function $\mathcal{C}(f)$. The parameters for this simulations and all the other in this section can be found at the appendix A.

The optimization of the cost function can be really difficult in practice. For that reason, in the following sections we implement a low complexity two steps approach.

First, a coarse estimate of the frequency is obtained by using a Grid Search optimisation algorithm [5] in order to get the global maximum zone of the cost function and avoid local maximum zones. Since Grid-searching can be computationally expensive, we use it with an approximated cost function $\mathcal{C}_a(f)$ that doesn't have the best statistical performance but can reduce the computational complexity.

To get the $\mathcal{C}_a(f)$, we consider N tends to infinite (it's only an approximation of $\mathcal{C}(f)$ because we work with a limited number of samples N) then:

[5]Grid Search algorithm is an exhaustive searching through a finite set of values for each hyperparameter

$$\mathcal{C}_a(f) = \frac{1}{N}\left|X(f)\right|^2 \tag{40}$$

where $X(f)$ [6] is the Discrete Fourier Transformation (DFT) of $x[n]$:

$$X(f) = \sum_{n=0}^{N-1} x[n] \times e^{-j2\pi f \times n/Fs}. \tag{41}$$

The second steps starts from the frequency estimate obtained from the coarse search, performing a refined search using the Newton-Raphson (NR) optimisation algorithm with the ML estimator with the real cost function $\mathcal{C}(f)$, as defined in equation (38). This cost function provides a better statistical performance without increasing the computational complexity too much.

#### 7.3.2 Estimation of the amplitude and the phase

$$\widehat{a} = |\widehat{c}| \tag{42}$$

$$\widehat{\phi} = \arg[\widehat{c}] \tag{43}$$

where the complex phasor $\widehat{c}$ can be adapted from [7] to correspond to a single phase:

$$\widehat{c} = 2\beta(\widehat{f})(NX(\widehat{f}) - q^*(\widehat{f})X^*(\widehat{f})) \tag{44}$$

We find in [7] the definition of $q(w)$.

### 7.4 Methodology

#### 7.4.1 Coarse search

To perform the Coarse search, we use the approximate estimator $\mathcal{C}_a(f)$ in equation (40) with the Grid Search algorithm.

Therefore, we have:

$$\widehat{f} = \arg\max_{f\in\theta}\left|X(f)\right|^2. \tag{45}$$

where $\theta$ is the set of frequencies and $X(f)$ from equation (41) can be expressed as:

[6] Paper [7] uses $Y$ instead of $X$

$$X(f) = \sum_{n=0}^{N-1} x[n] \times [cos(2\pi f \times \frac{n}{Fs}) + jsin(2\pi f \times \frac{n}{Fs})]. \tag{46}$$

To perform the Grid Search algorithm, we need to define the set $\theta$, which is a vector of frequencies from a minimum frequency $f_{min}$ to a maximum frequency $f_{max}$ and can be expressed as

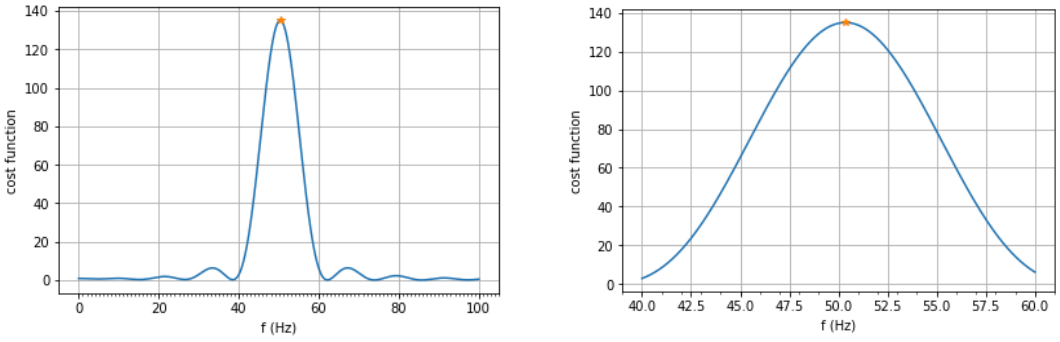
$$\theta = \{f_{min}, f_{min} + step, f_{min} + 2 * step, ..., f_{max} - 2 * step, f_{max} - step, f_{max}\}\text{Hz}. \tag{47}$$

so the cardinal of the set $\theta$ is expressed as

$$|\theta| = \frac{f_{max} - f_{min}}{step} \tag{48}$$

**Selection of values for set $\theta$**

Assuming the case in which we're measuring the electrical network, we know that the theoretical frequency of the electrical system of France is 50Hz, so we start by defining $f_{min}$<50 and $f_{max}$>50.




(a) $f_{min}$=0, $f_{max}$=100

(b) $f_{min}$=40, $f_{max}$=60

Figure 22: Cost function

The figure 22 represents the value of this cost function when we change $f_{min}$ and $f_{max}$ as the yellow point shows the value found by the Coarse Search.

We can also deduce that if the *step* is too small, the Grid Search will perform better but the computational complexity increases but if it's too big, we could miss the global maximum of the function.

We can conclude that a good selection of the set $\theta$ can help us find the maximum global minimizing the amount of necessary calculations, since it's evident that the bigger the set $\theta$ is (and higher it's cardinal), the evaluation of the cost function has to be performed more times.

**Pseudo code**

We perform the DFT estimator by calculating the absolute value of $X(f_m)$ for each $f_m$, in order to find the frequency of the signal, which is going to be the one corresponding to the maximum value found of $\left|X(f_m)\right|$, referred to as $\mathcal{C}_{MAX}$.

The pseudo code describing this steps is:

**Algorithm 1:**

**Data:** $x[n], \theta$

initialization

$f_{\bar{m}} = f_{min}$

$C_{MAX} = 0$

**for** $f$ *in* $\theta$ **do**

  Compute $X(f_{\bar{m}})$ with eq.(46)

  $X_{abs} = \left|X(f_{\bar{m}})\right|$

  **if** $X_{abs} > \mathcal{C}_{MAX}$ **then**

    $\mathcal{C}_{MAX} = X_{abs}$ ;

    $f_{est} = f_{\bar{m}}$

  **end**

**end**

**Result:** $f_{est}$ is the frequency of the signal.

**Why we need a refined search**

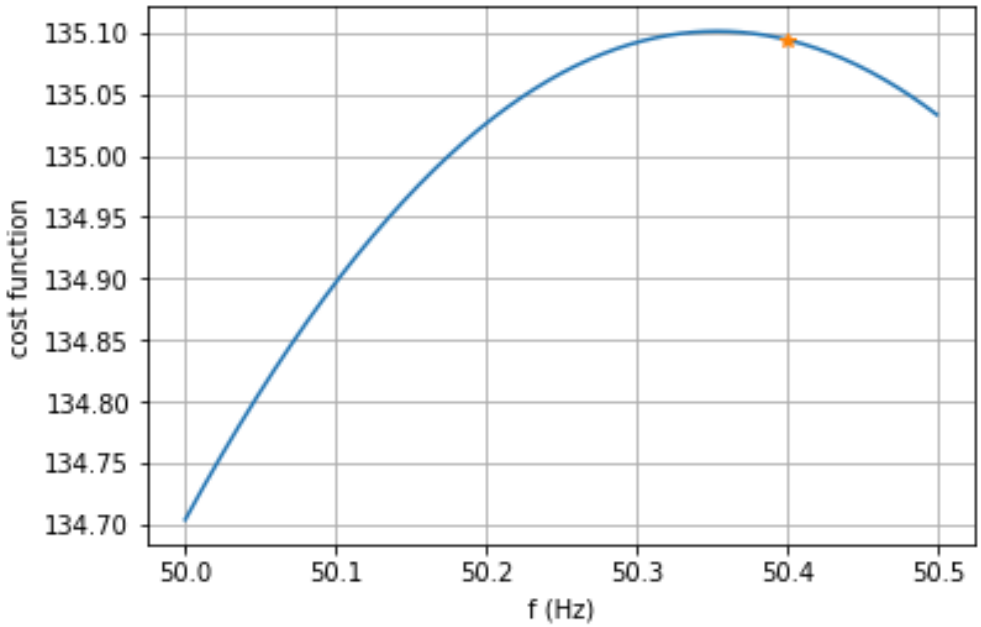



Figure 23: Coarse Search result

We see in figure 23 that the Coarse Search finds a maximum global zone but a better approximation would require a smaller step. A better way to get a more precise frequency value is to use the frequency obtained by the Coarse Search and use it as the initial value to a Refined Search.

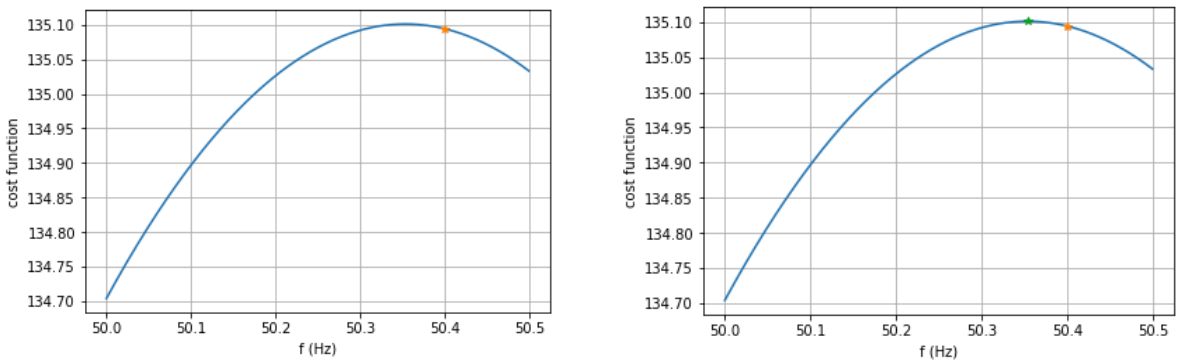
### 7.4.2 Refined search

We part from the estimated frequency obtained from the Coarse Search. In this section, we are looking to obtain a refined estimate of the frequency implementing the ML estimator with a fixed number of NR's algorithm iterations as follows

$$\omega_{k+1} = \omega_k - \frac{C'(\omega)}{C''(\omega)} \tag{49}$$

where $\omega_k$ is the estimate of the angular frequency at the $k^{(th)}$ iteration, $C'(\omega)$ and $C''(\omega)$ are the first and second derivatives of $C(\omega)$ (equation (38)) with respect to $\omega$.

The definitions of the derivatives $C'(\omega)$ and $C''(\omega)$ are available in [6].




(a) First step for the Refined Search (yellow)

(b) Result of the Refined Search with N iterations (green)

Figure 24: Refined Search

In figure 24 we observe that the Refined Search (green) finds a better approximation to the global of the cost function than the Coarse Search does (orange) and we can obtain then a better approximation of the estimated frequency.

The NR algorithm runs N iterations and the bigger N, it's expected to give better results. Keeping in mind that there's always a compromise between the statistic performance and the reduction of the computational complexity, we ran some tests on the computer, we verified that this algorithm approximately needs two iterations to reach an accurate estimation of the frequency.

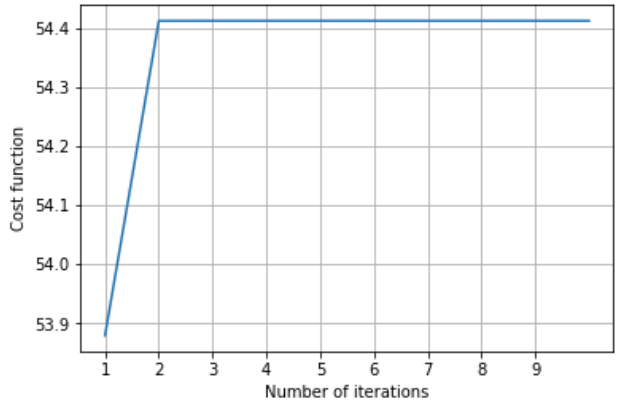



Figure 25: cost function - Number of NR iterations

As show in figure 25, more iterations don't make a significant refinement.

## 7.5 Implementation

Parting from two already existing algorithms for estimating frequency, amplitude and phase, one based on the DFT estimator and the Grid Search algorithm [5] and the other baser on the ML estimator and the NR algorithm, both implemented in Python by Associate Professor V. Choqueuse [8], we implemented this same code in C language in order to run it on a STM32L4 Discovery kit IoT node MCU.

First, knowing that the fundamental frequency is approximately equal to 50 Hz, we put into practice the Coarse Search algorithm for the frequency in the range from 40Hz to 60Hz with a step equal to 0,1Hz, so our set $\theta$ is

$$\theta = \{40.0, 40.1, 40.2, ..., 49.8, 49.9, 50.0\} \tag{50}$$

We obtained the same results as the Python code, meaning the translation of the code was correct.

However, testing the developed C code on the STM32L4 Discovery kit IoT node showed that the time required by the MCU to finish all the calculations can be improved, as we explain in this section.

We know there are two kinds of float number in C:

- Single precision floats (SPF)
- Double precision floats (DPF)

The STM32L4 (M4 Core) is not able to work natively with Double Precision floats, it will use a software emulation if the code has this type of float. We tested the algorithm with a test signal and we obtained the following results:

| | Frequency result | Time consumption |
|---|---|---|
| Python code | 50,2941107103675 | Can't be tested on the MCU |
| C code using DPF | 50,294110710297574 | 1051 ms |
| C code using SPF | 50,2941132 | 987 ms |

Table 2

The table 2 shows that using SPF, the result obtained is equal to the Python result up to 5 decimals after the comma, which can be considered enough for this study. It also reduces the time consumption, although we expected a bigger difference since the FPU in Cortex-M4 supports single precision FP operations, but not double precision, and if double precision calculations are present, the C compiler will use run time library functions to handle the calculation in software.

Now we focus on each algorithm

| Algorithm | Time consumption |
|---|---|
| Coarse search | 800 ms |
| Refined Search | 187 ms |

Table 3

As shown by the algorithm 1, the evaluation of the cost function requires the computation of equation (46) and that implies calculating cosine and sine functions multiple times. This motivated the use of the mathematical concept of Recurrence Relation [7].

- 1rst elements ($n = 0$)

$$cos(2\pi \times f \times \frac{n}{Fs}) = 1 \tag{51}$$

$$sin(2\pi \times f \times \frac{n}{Fs}) = 0. \tag{52}$$

- n >1 elements

Computed before looping:

$$cos_{term} = cos(2\pi \times \frac{f}{Fs}) \tag{53}$$

$$sin_{term} = sin(2\pi \times \frac{f}{Fs}) \tag{54}$$

Computed in loop:

$$cos(2\pi \times f \times \frac{n}{Fs}) = cos(2\pi \times f \times \frac{n-1}{Fs}) \times cos_{term} - sin(2\pi \times f \times \frac{n-1}{Fs}) \times sin_{term} \tag{55}$$

$$sin(2\pi \times f \times \frac{n}{Fs}) = cos(2\pi \times f \times \frac{n-1}{Fs}) \times sin_{term} + sin(2\pi \times f \times \frac{n-1}{Fs}) \times cos_{term}. \tag{56}$$

The obtaining of equations (55) and (56) are explained at appendix (B). We observe that we can calculate the firsts elements and the following elements are defined by the preceding, allowing us to reduce the amount of calculations realized inside a loop.

We applied this on the loop that calculates the DFT for each frequency of the vector of frequencies and obtained the following results:

[7] A recurrence relation is an equation that recursively defines a sequence or multidimensional array of values, once one or more initial terms are given: each further term of the sequence or array is defined as a function of the preceding terms.

| DFT loop | Time consumption |
|---|---|
| No recurrence | 800 ms |
| Recurrence | 26 ms |

Table 4

This table shows that the use of the Recurrence Relation is a very effective way to reduce the time consumption.

After having applied all of this considerations to our code, we observed a considerable improvement in the time performance of the code, as summed up the table 5:

| Coarse search + Refined Search | Time consumption |
|---|---|
| First version | 1051 ms |
| Last version | 58 ms |

Table 5

# 8 Transmission of data towards the Cloud

## 8.1 Context

Working on the cloud provides scalability and flexibility by not being limited to physical constraints of on-premises servers, reliability of multiple data centers with multiple redundancies, customization by configuring servers to your preferences, responsive load balancing that can easily respond to changing demands, as well as security of the stored information.

As mention in the section 1, the ENIB provided us with the STM32 Discovery Kit, which integrates a WiFi module that allow us to transmit data via WiFi. This motivated our interest in the possibility of saving the estimated parameters in a database hosted in a Cloud service.

A cloud service provider is a third-party company that offers cloud-based platform, infrastructure, application, or storage services. There are many Cloud services providers and the best known are Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform.

Since the ENIB has recently joined the AWS Educate partnership, that allows the students and professors to use certain services of the AWS for free for a year, we decided to investigate more about this service.

### 8.1.1 Why AWS?

- Availability

It operates 64 Availability Zones within 21 geographic regions around the world, with announced plans for 12 more Availability Zones and four more Regions in Bahrain, Cape Town, Jakarta, and Milan. Therefore it has a very strong presence in the public cloud market.

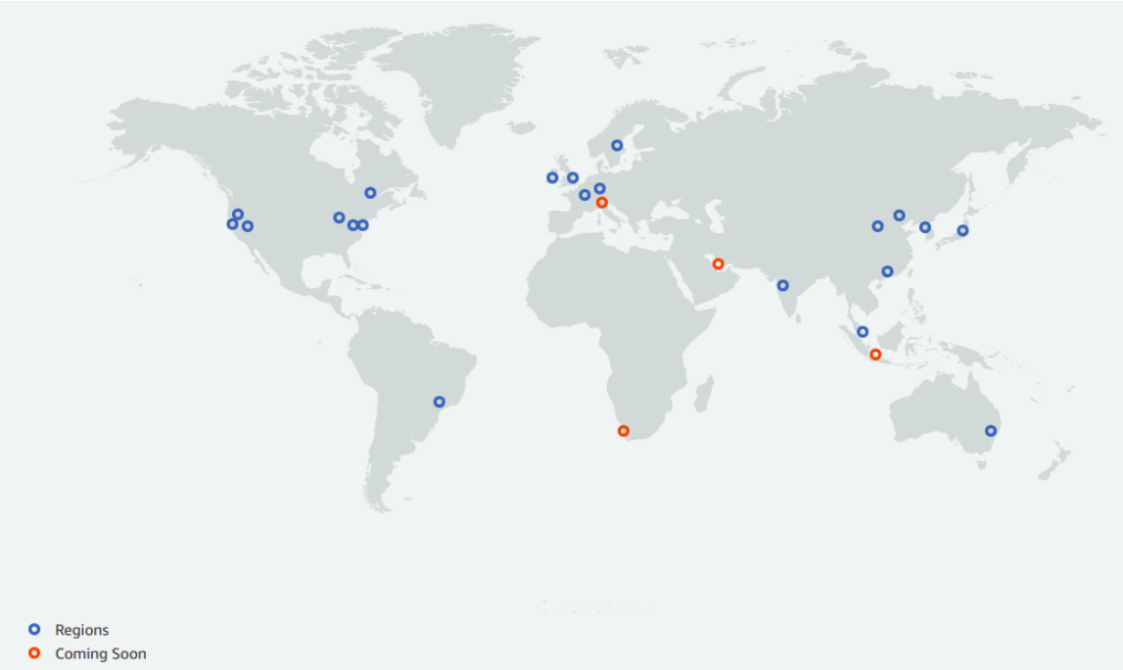



Figure 26: AWS regions. Source [4]

- Pay-As-You-Go Pricing

It allows the customer to manage the amount of compute power, storage, and other resources they use, stopping and starting instances as needed, paying for what they use, with no long-term contracts or up-front commitments.

- Services

The AWS offers a number of services for server rental (EC2), database server rental (Amazon dB, RDS), static file hosting (S3). In recent years, Amazon has begun to expand its offering to the Internet of Things (IoT), which offers solutions to connect a wide range of devices.

- Flexible

AWS enables the customer to select the operating system, programming language, web application platform, database, and other services. The customer receives a virtual environment that lets them load the software

and services their application requires.

- Security

AWS utilizes an end-to-end approach to secure and harden our infrastructure, including physical, operational, and software measures.

Also, the access to the AWS resources can be restricted using the IAM(Identity and Access Management) and defining privileges for user actions, this offers Data Privacy.

### 8.1.2 Used services

For this study, among the various different products AWS offers, we're interested on the Internet of Things (IoT), which provides solutions to connect different devices and to collect, store and analyze device data. The AWS IoT services that are going to be used for this work are:

- IoT Core: It's a managed cloud service that lets connected devices easily and securely interact with cloud applications and the other devices. AWS IoT Core supports HTTP, WebSockets, and MQTT, a lightweight communication protocol specifically designed to tolerate intermittent connections, minimize the code footprint on devices, and reduce network bandwidth requirements.

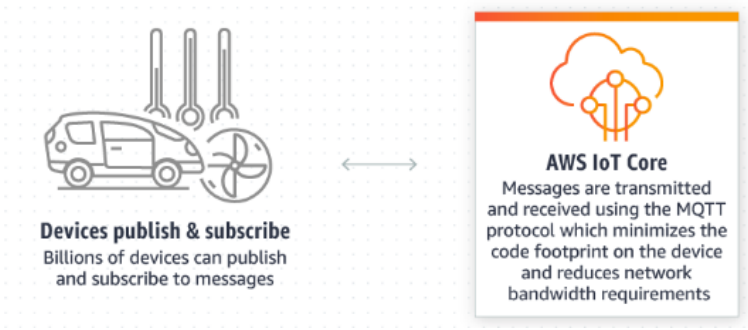



Figure 27: Source [1]

- Amazon FreeRTOS: It's a real time operating system for microcontrollers that makes small, low-power edge devices easy to program, deploy, secure, connect, and manage. Amazon FreeRTOS extends the FreeRTOS kernel, a popular open source operating system for microcontrollers, with software libraries that make it easy to securely connect a small, low-power devices to AWS cloud services like AWS IoT Core.

## 8.2 Objective

The STM32 Discovery Kit card has a basic WiFi module for transmitting data via WiFi, so our objective is to transmit data in MQTT procotol to AWS. This allow us to remotely monitor data, so we can be notified when the frequency changes or is different to the expected values, and this can be the first step to perform Preventive Maintenance.

This will require certain steps:

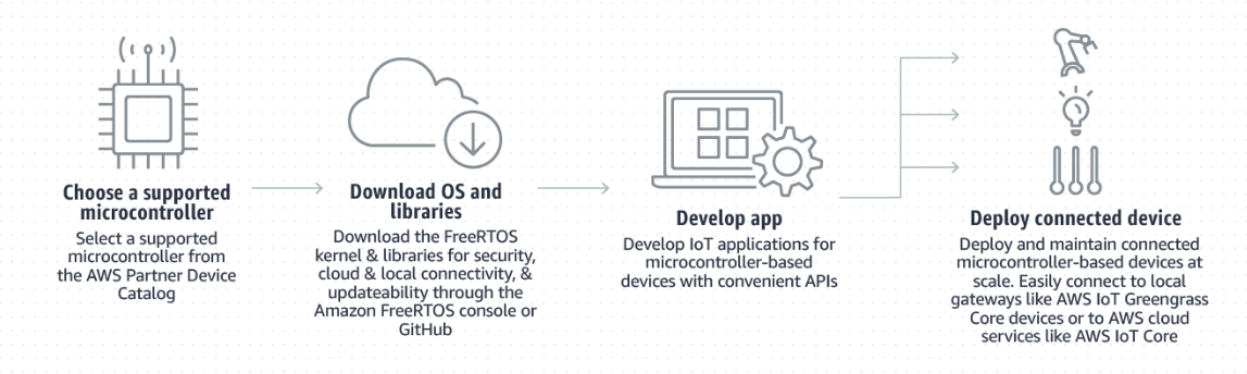



Figure 28: Source [3]

## 8.3 Methodology

**Choose a supported microcontroler**

We're using for this work the STM32L4 Discovery kit IoT node (as mentioned in 1). We can check that our selected hardware supports Amazon FreeRTOS features and capabilities from the AWS Partner Device Catalog, as shown in figure 29.

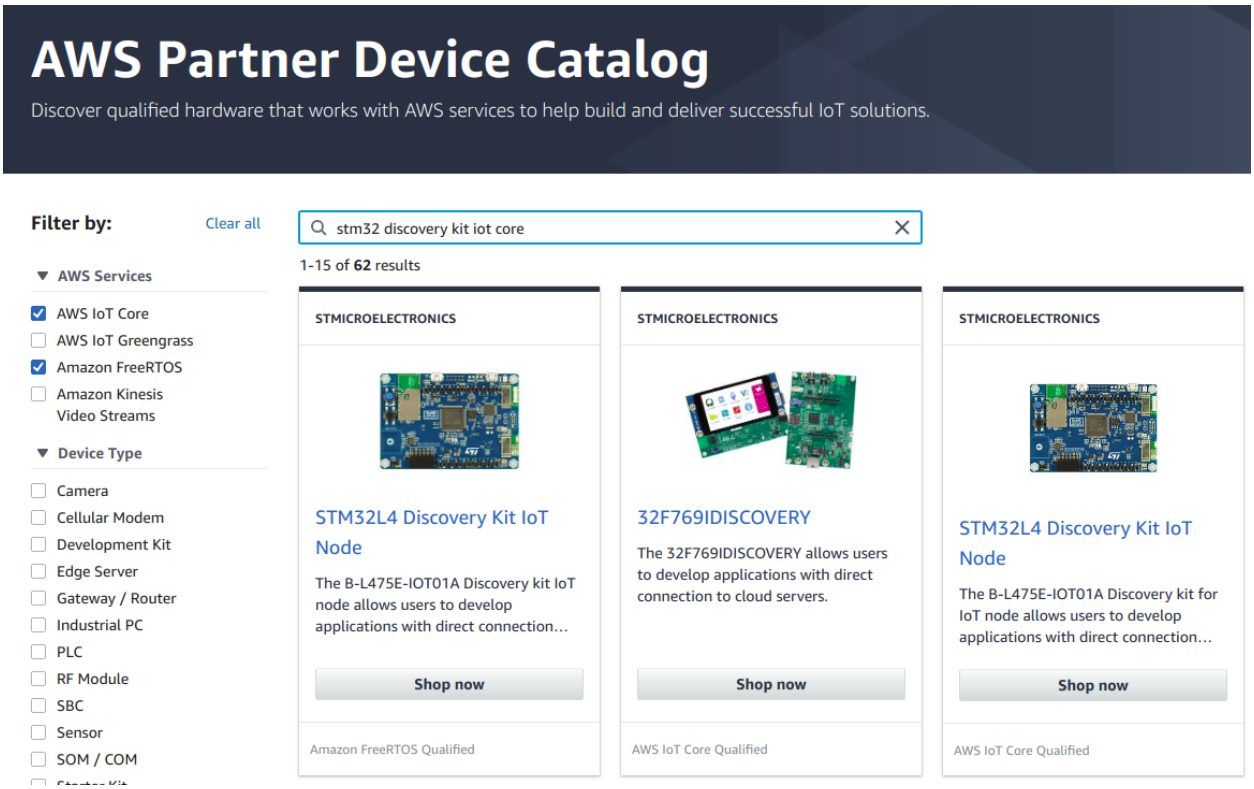



Figure 29: AWS Partner Device Catalog. Source [2]

### Download OS and libraries

We go to the Amazon FreeRTOS console figure (30) to get a custom download of the operating system by choosing the libraries relevant to our case (figure 31).

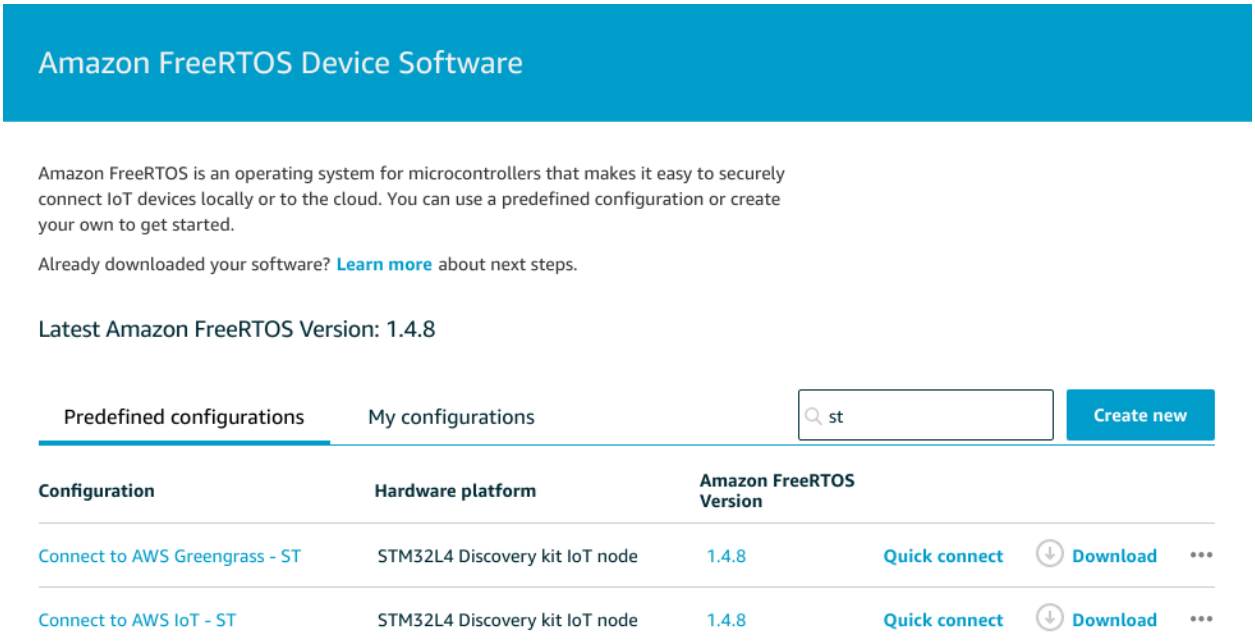



Figure 30: AWS Console.

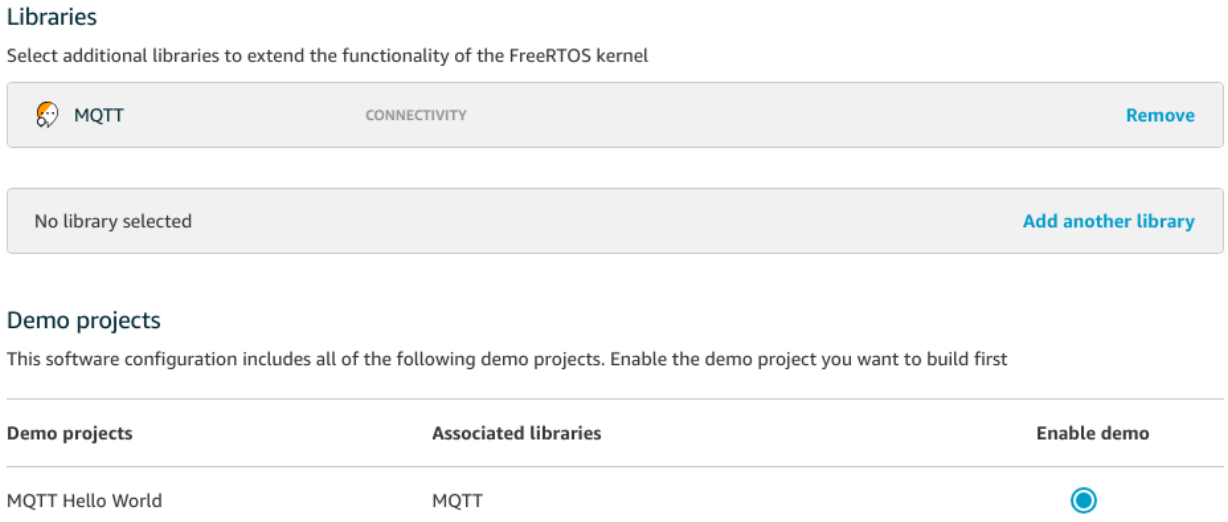



Figure 31: Selection of Libraries and demo projects.

Then, we follow the console instructions.


Figure 32: Quick Connect Instructions

Once we downloaded our Amazon FreeRTOS software, we register our device, this means creating an AWS IoT thing. A thing is a record in the cloud that represents the physical device. A physical device needs a thing in order to work with AWS IoT.

The third step is to download the credentials.

The following AWS IoT resources are created:

- An AWS IoT policy: it grants the device permissions to access AWS IoT resources, in this case, to send and receive messages.
- A private key and a certificate: it allows the device to authenticate with AWS IoT.

After this, we have to configure Amazon FreeRTOS on our device, this means:

Set Up your Development Environment To set up your board for development, follow the steps in Setting Up your Environment in your device's Getting Started guide .

- Configure Wi-Fi Credentials: define the network's SSID, the network's password, and the network's security type.
- Configure AWS Credentials: get the credentials in the Amazon FreeRTOS folder, downloaded in the first step.

**Develop app**

Once we our device is connected to the AWS Cloud, we proceed to modify the code in order to be able to send the estimated frequency to the AWS IoT Core, so we need that our application runs the developed code to estimate the frequency, as explained in section 7.4.

- initializes the WiFi module
- writes certificate
- writes device private key
- connects to the network defined in the WiFi Credentials
- creates MQTT taks
- connects to the AWS Cloud
- subscribes to the topic we send the messages to
- receives the signal via DMA
- estimates the frequency of the received signal
- publishes the frequency estimation

This last three steps can be done as many times as desired. The objective is only to publish the frequency when it has changed so it can alerts us about the condition of the received signal.

**Deploy connected device**

The following figure shows the final result at the AWS IoT console:

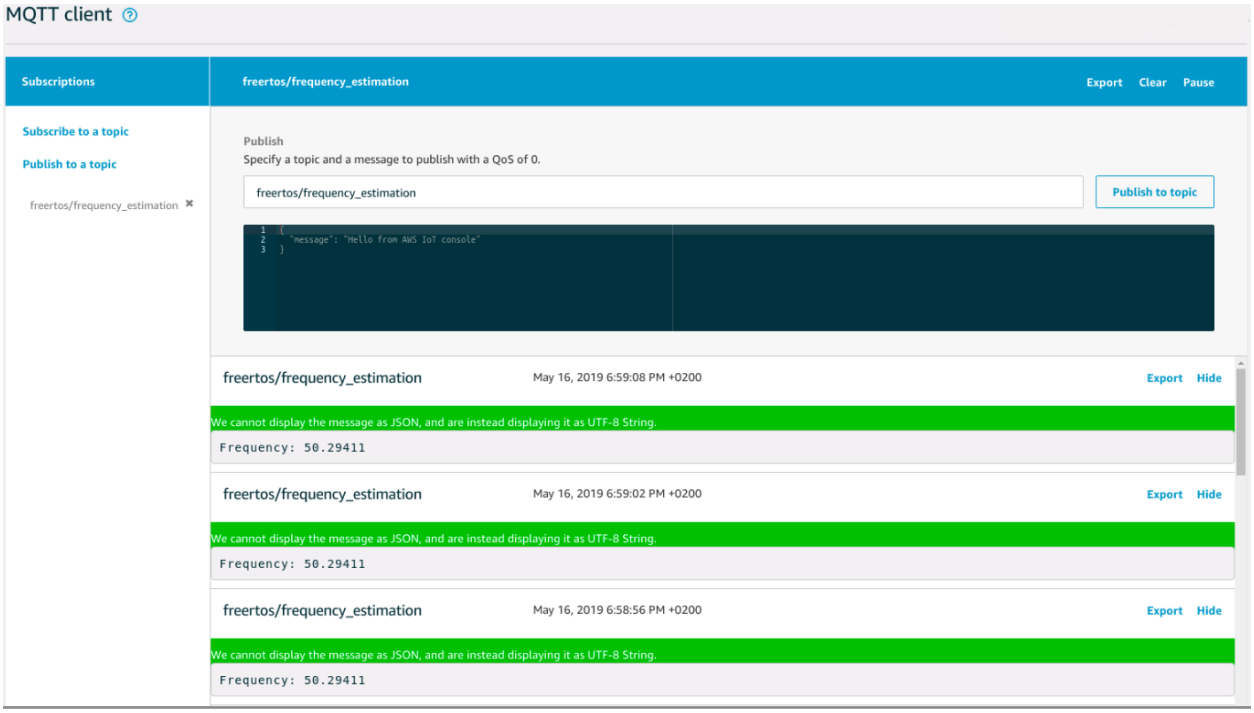



Figure 33: Messages published at AWS IoT

# 9 Conclusion

In this work we have developed a fully functional triphasic measurement system, which allow us to safely sense high voltage signals and current signals that are transformed to low voltage DC values and level shifted to satisfy the input requirements of a microcontroller. We designed the signal treatment circuit around the main components which are the LV-25P and LA-25p sensors. The transfer equations of the system were found by developing the electrical equations and all the required components were dimensioned. Two board prototypes were made, being the latest one a fully functional version that was tested and improved until it satisfied our requirements.

To apply the estimation algorithms, we needed to sample the analog signal that outputs our board and for that purpose we used the ADC of the STM board in DMA mode, which made possible to do the sampling

with zero cost to the microcontroller leaving it free to the algorithms execution. A first estimation of the sample frequency needed was made and the process to set it was explained, but further testing will be needed once the whole system is integrated to assure the ideal setting.

We implemented estimation algorithms to estimate the frequency, amplitude and phase of the signal, separating the multidimensional problem so we could estimate the frequency first. The frequency estimation required two steps, first we used the Grid Search optimization algorithm with a Discrete Fourier Transform based estimator and from the result this gave us, we used the Newton-Raphson algorithm with a Maximum Likelihood estimator. Since we are using a microcontroller to perform the calculus, it was important to measure the computational complexity through the time the algorithms took. This allowed us to improve our code and this reflected clearly in the time performance.

Since we worked with a STM32 board with a WiFi module and the ENIB has a partnership with AWS Educate, we connected our board to AWS Cloud in order to remotely monitor the frequency estimation of the tested signal by sending messages to the cloud using the MQTT protocol and using the IoT Core and Amazon FreeRTOS services. We explained step by step how we worked with the AWS Cloud so future students can continue to work with this.

The integration of the software and hardware was not tested because of time limitations so each part was tested individually.

## 9.1 Future work

Due to time limitations, only the developed hardware was tested and implemented for three-phase signals. After testing the board performance we saw that there is place to future improvements, like using lower tolerance resistors and developing a board with SMD components making the connections shorter and less susceptible to noise, and integrating an over tension-current protection system. Further testing of the board is recommendable to make a complete analysis of the performance before making another iteration and using a simulation software may also be recommendable to make this testing process easier.

We would like to adapt our code for a single-phase signal so it can estimate the parameters of triphasic systems, using the paper [7], where the algorithms for three-phase signals are explained, so we can implement them in language C to work with the microcontroller.

We are sending only the frequency estimation to the AWS Cloud, we would like to also send the amplitude and phase estimations. This would allow us to remotely monitor all three parameters and implement ways to be notified (for example, via SMS) when any of them changes.

# References


[1] AWS. *AWS IoT Core Overview*. URL: https://aws.amazon.com/iot-core/?nc2=h_m1. (accessed: 02.06.2019).

[2] AWS. *AWS Partner Device Catalog*. URL: https://devices.amazonaws.com/search?sv=freertos. (accessed: 02.06.2019).

[3] AWS. *Getting started with Amazon FreeRTOS*. URL: https://aws.amazon.com/freertos/getting-started/. (accessed: 02.06.2019).

[4] AWS. *Global Infrastructure*. URL: https://aws.amazon.com/about-aws/global-infrastructure/. (accessed: 29.05.2019).

[5] V. Choqueuse. *Coarse Search - Python Code*. URL: https://git.enib.fr/choqueuse/ppe_2019_stm32/blob/master/src/estimator1.py. (accessed: 29.05.2019).

[6] V. Choqueuse. *Expression of the MV estimator for Single Phase Frequency Estimation*. URL: https://git.enib.fr/choqueuse/ppe_2019_stm32/raw/master/pdf/estimator_MV_newton/estimator.pdf?inline=false. (accessed: 20.05.2019).

[7] V. Choqueuse. *Frequency and Phasor Estimations in Three-Phase Systems: Maximum Likelihood Algorithms and Theoretical Performance*. URL: https://git.enib.fr/choqueuse/ppe_2019_stm32/blob/master/pdf/TSG-00963-2017.pdf. (accessed: 20.05.2019).

[8] V. Choqueuse. *Refined Search - Python Code*. URL: https://git.enib.fr/choqueuse/ppe_2019_stm32/blob/master/src/algorithm_MV.py. (accessed: 29.05.2019).

[9] The School of Computer Science and Engineering at The Hebrew University of Jerusalem. *Maximum Likelihood Estimator for the Linear Model with White Gaussian Noise*. URL: http://www.cs.huji.ac.il/~csip/tirgul4.pdf. (accessed: 29.05.2019).

[10] G.H. Golub and V. Pereyra. *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*. URL: https://www.jstor.org/stable/2156365?seq=1#page_scan_tab_contents. (accessed: 11.06.2019).

[11] ST. *Discovery kit for IoT node, multi-channel communication with STM32L4*. URL: https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/b1/b8/7a/f2/f7/8d/4b/6b/DM00347848/files/DM00347848.pdf/jcr:content/translations/en.DM00347848.pdf. (accessed: 03.06.2019).

[12] ST. *STM32L4 Direct memory access controller*. URL: https://www.st.com/content/ccc/resource/training/technical/product_training/8c/89/2c/0e/ca/48/41/2f/STM32L4_System_DMA.pdf/files/STM32L4_System_DMA.pdf/jcr:content/translations/en.STM32L4_System_DMA.pdf. (accessed: 03.06.2019).

[13] F. Auger V. Choqueuse A. Belouchrani and M.Benbouzid. *Frequency and Phasor Estimations in Three-PhaseSystems: Maximum Likelihood Algorithms and Theoretical Performance*. URL: https://www.researchgate.net/publication/324224124_Frequency_and_Phasor_Estimations_in_Three-Phase_Systems_Maximum_Likelihood_Algorithms_and_Theoretical_Performance. (accessed: 03.06.2019).

# Appendices

## A Simulation parameters

Unless indicated otherwise, the parameters for the simulations in section 7 are:
$a = 2.31466$
$\phi = 0.0409$
$f = 50,29411$
$N = 100$
$\theta = \{f_{min}, f_{min} + step, f_{min} + 2 * step, ..., f_{max} - 2 * step, f_{max} - step, f_{max}\}$ where $f_{min} = 40.0$, $f_{max} = 60.0Hz$ and $step = 0.01Hz$

## B Recurrence

$$cos(2\pi \times f \times \frac{n}{Fs}) = cos(2\pi \times f \times (\frac{n-1}{Fs} + \frac{1}{Fs})) \tag{57}$$
$$= cos(2\pi \times f \times \frac{n-1}{Fs} + 2\pi \times f \times \frac{1}{Fs}) \tag{58}$$
$$= cos(2\pi \times f \times \frac{n-1}{Fs}) \times cos(2\pi \times \frac{f}{Fs})) - sin(2\pi \times f \times \frac{n-1}{Fs}) \times sin(2\pi \times \frac{f}{Fs})) \tag{59}$$

$$sin(2\pi \times f \times \frac{n}{Fs}) = sin(2\pi \times f \times (\frac{n-1}{Fs} + \frac{1}{Fs})) \tag{60}$$
$$= sin(2\pi \times f \times \frac{n-1}{Fs} + 2\pi \times f \times \frac{1}{Fs}) \tag{61}$$
$$= cos(2\pi \times f \times \frac{n-1}{Fs}) \times sin(2\pi \times \frac{f}{Fs})) + cos(2\pi \times f \times \frac{n-1}{Fs}) \times sin(2\pi \times \frac{f}{Fs})) \tag{62}$$