Robust low-cost energy meter

Abel Facal Malvar

July 7, 2023

Abstract—
This paper consists of the development of a robust low-cost energy meter. The paper aims to use an alternative method to the one existing for reading currents with high slopes, this existing method consists on capturing the high-frequency components. Focusing in the lower frequencies provides a robust yet accurate reading, as most of the power is contained in the fundamental frequency. This new approach provides robustness agains electromagnetic interferences and allows to use lower-cost components. An ESP32 microcontroller programmed in C is used to get the readings from a low-cost current transducer (SEN0211) and a differential voltage probe. Some experiments with traditional linear loads and modern non-linear loads are made to validate the system. The deviations obtained stay under 1% in linear loads and range between 3.24% to 74% for the most complex non-linear load setups, probably due to the low sampling frequency used. The paper discusses the project's design, measurements and results, giving explanations and guidelines for better understanding and future improvements in these measurements.

I. Introduction

Daily measuring of the power consumption in homes, offices, or other facilities is necessary for the electricity providers to charge the users for the services provided. A modern problem arises in these routinary measurements with developing new technologies resulting in over or under-billing clients [1] [2] [3].

Multiple studies, some made at the University of Twente, have shown that the static energy meters commonly used in any location where electricity is used can present errors.

These meters have been mainly designed to measure the power consumption of linear loads. However, lately, with the use of new technologies based on semiconductors, these loads have become non-linear and have presented harmonic distortions. These harmonics distortions, produced by currents with high slopes, induce errors in the measurements of the power meters. Some of these experimental deviations have been up to +2675% when using the standard supply of a power grid. In contrast, when using an ideal power supply with standardized impedance, these deviations are lower (up to +483%) [4].

Some studies have checked the direct effects of these non-linear loads by measuring the errors in the static meters. To better understand these errors, dimmers were used to study the importance of the phase firing angle of these loads and also give an overview of how different types of current sensors behave in these measurements [5].

For this reason, it is essential to check whether the current energy meters installed in households are trustworthy and if they fall under the quality measuring accuracy set by the European Union standards [6].

The most recent approach to this problem is to capture all the high-frequency harmonic components from the current measurements to obtain an appropriate measurement. This approach comes with expensive parts that can work at high frequencies, as these are challenging to capture.

It has been theoretically proven that a different approach can get good results that fall within the error acceptance stated by the European Convention. This approach is based on the orthogonality of the non-fundamental power components, which explains that most of the power is contained in the low-frequency harmonics [7].

The objective is to obtain a robust measurement using cheap and widely available components, using the least amount of hardware possible. This meter aims to monitor pre-installed energy meters, so the system's non-intrusiveness will also be considered.

The paper will present a complete hardware and software analysis of the project's design. As this study's importance relies on the quality of the current and voltage measurements, especially for non-linear loads, alternative hardware and setups will be studied, analyzing the advantages and disadvantages of each one of them and presenting alternative configurations where each one can be used.

Once the hardware is chosen, a couple of experiments will be made, using a high-quality power analyzer as a reference for the power measurements for three different loads with multiple setups: no load, linear loads (electric heater) and non-linear loads (water pump).

The results will be displayed and analyzed for the different experiments performed. Furthermore, the individual measurements of current and voltage captured from the developed hardware will be plotted and compared to the similar measurements obtained from an oscilloscope, as in these measurements are the most critical aspects of this study.

A small discussion of possible future steps for this research will be shown, giving some insights into what could have been done differently if the knowledge and discoveries presented had been known at the beginning of this study.

Lastly, a conclusion on the overview of the results and methodology used will be presented, evaluating the results to measure the project's success and analyzing the potential of this study.

II. Hardware design

In this section, the design and analysis of the hardware will be shown. The system can be seen in Fig. 1.

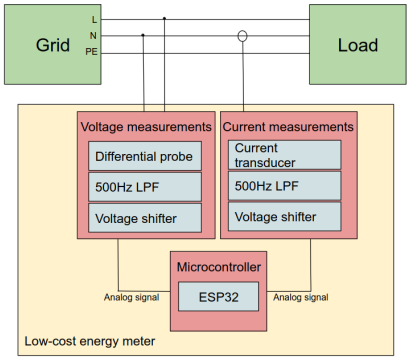

Fig. 1: Schematics of the system

It consist on one microcontroller, which will be an ESP32. This microcontroller will get the current measurements from a low-cost current transducer (CT) that after passing by a circuit with a low-pass filter (LPF) and a voltage divider it will have the appropiate values for the microcontroller to read. Similarly to how the currents are measured, the voltages measurements also consist on an analog signal, that comes out of a differential voltage probe, which is read after transforming the signal with a LPF and a voltage shifter.

A. *Microcontroller*

Some different microcontrollers were considered for this application. The considerations were Arduino, ESP32 and STM32F4 (a family of STM32 microcontrollers).

The microcontroller was chosen taking into account technical specifications on top of cost and availability.

The ESP32 has 2 ADCs (STM32F4 has up to 3 depending on the model), which Arduino does not have available in the low-cost models considered (UNO and NANO).

Having 2 ADCs allows to read current and voltage with the minimum phase delay possible. Even though Arduino consists of only 1 ADC, it can also read multiple channels, but the time needed to set the channels for each reading would add a more significant delay between readings. On top of this, the Arduino ADC has 10 bits resolution, which gives, taking into an account that the limits of it's ADC is 0-5 V [8], 4.88mV of resolution, while for the ESP32 it's ADCs has 12 bits resolution, and in its maximum configuration, as it can be seen in table I, the resolution would be 0.32 mV.

TABLE I: Voltage ranges ADC ESP32 [9]

Attenuation(dB)	Voltage input range(mV)
0	100-950
2.5	100-1250
16	150-1750
11	150-2450

The ESP32 was the microcontroller decided to use as it is low-cost and widely available on top of meeting with the technical specifications.

To prove that this microcontroller matched the expectations, a couple of experiments needed to be conducted to check that the system could work under the desired constraints before the final setup.

The system should be able to:

- Sample at 1 kHz
- Sample current and voltage with negligible phase delay

Sampling at 1 kHz is chosen because it allows to capture signals up to 500 Hz. The 500 Hz was the cutoff frequency decided to use in the LPF as filtering at similar frequencies has been proven to work for these measurements [10].

To check for both these things, the two ADC pins were shortcircuited and fed a sinusoidal signal of $V_{PP} = 2$ V with a 1.25 V offset to get a fully positive voltage wave with 50 Hz of frequency using a function generator.

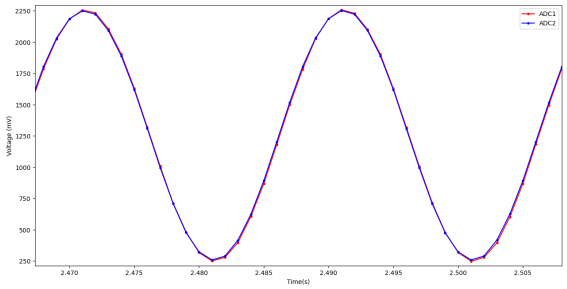

Fig. 2: Reading of the same wave from the 2 ADCs

The result can be seen in Fig. 2. The figure shows that the wave nearly overlaps completely, being able to neglect the phase difference. The time difference between ADC1 reading and ADC2 reading is 60 μs which is 1.08º, this could still have an impact in pulsed currents, specially in cases where a pulse appears when the voltage of the grid is close to 0 V.

Fig. 2 shows that each full 50 Hz wave is completed with 20 points, indicating that the sampling is done every 1 ms (1 kHz), this was also tested with an internal clock that would print the time taken for every measurement.

It is essential to mention that the baud rate of the communication can directly impact the sampling, as the ESP32 needs to be able to finish transmitting the data before starting the next measurement. This is extremely important as the time the microcontroller uses to transmit data is generally much higher than other processes of the microcontroller, like reading the ADC.

Once this worked, the next step was to start with real current and voltage measurements.

B. *Current measurements*

As this paper aims to show a robust low-cost energy meter, a low-cost (CT) was bought (SEN0211). This CT can read AC currents from 0 to 20 A, giving a linear AC output from 0 to 1 V.

As shown in Fig. 3 where the transfer function of the current probe was taken using the extension "FRA for picoscope" and the picoscope 4824, the real cutoff frequency of the probe is closer to 10 kHz. This is different that what is said in the datasheet, where the CT is said to have a frequency bandwidth from 50 Hz to 1 kHz [11].

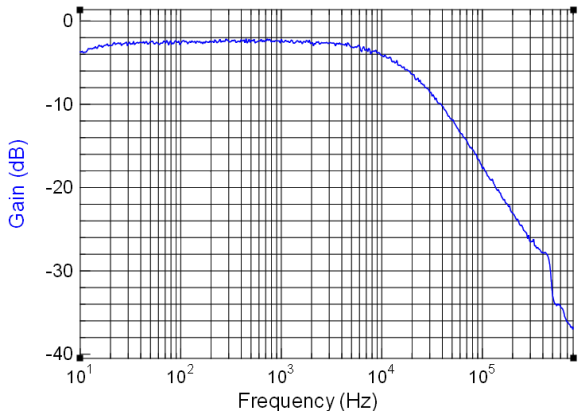

Fig. 3: Bode plot for the magnitude of the current probe

This CT came with a module to read the current from the microcontroller that will not be used because it reads current as a multimeter, giving as an output the I_{rms}, so the raw data can not be obtained. These rms values are not the desired ones as the approach of power orthogonality uses the instant value of the current and voltage, and not the rms values.

C. Voltage measurements

For the voltage readings, a differential probe will be used. Even though this is not a low-cost component to measure the voltage, the objective for this proof of concept relies in the current measurements. Other more affordable alternatives should be implemented in future versions.

The differential probe used will be the TA042 (price $>$ 500€), using the mode with a ratio 1:100, which means the output for the grid measurements of this probe will be an AC reading between -3.53 V to 3.53 V.

D. Voltage shifter to read AC waves

As the ESP32 cannot read negative voltages (the readings have to stay between the range of 0.125 V to 2.45 V [9]), some extra hardware is needed to capture the signals of current and voltage. For this, the circuit with resistors from Fig. 4 will be used. The output of this circuit shifts the voltage to possitive and attenuates the signal, as it is described by the equation 1.

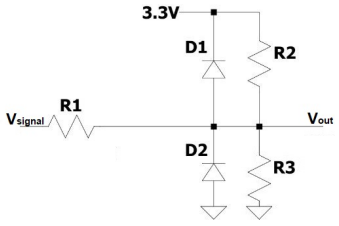

Fig. 4: Positive voltage shifter

The equation that defines the system is:

$$V_{out} = (\frac{V_{Signal}}{R_1} + \frac{V_{DD}}{R_2}) \frac{R_1 R_2 R_3}{R_1 R_2 + R_3 R_2 + R_1 R_3} \quad (1)$$

Where V_{DD} is the 3.3V terminal in Fig. 4.

The diodes in the circuit of Fig. 4 are clamp diodes used for protection to ensure no voltages out of the expected range harm the primary circuit. In the circuit, they can just be considered short circuits that get activated when the voltage exceeds VDD or goes below the GND value.

This circuit will be used for both measurements, the ones with current and the ones with voltage. The AC waves are different for both, so the circuits will vary their resistors values.

1) CT measurements: The values for the case of the currents, where the AC signal from the probe lies in the range -1 to 1 V, the circuit will have the values in the table II.

TABLE II: Values for the CT voltage divider

Component	Value
R1	331Ω
R2	594Ω
R3	$9.24k\Omega$
D1,D2	SR240

This values will transform equation 1 in (Taken into an account $V_{DD} = 3.3$ V):

$$V_{out} = 0.628 \cdot V_{signal} + 1.15(\text{V}) \quad (2)$$

Where, for the max possible value of the CT,the signal will be the one seen in Fig. 5.

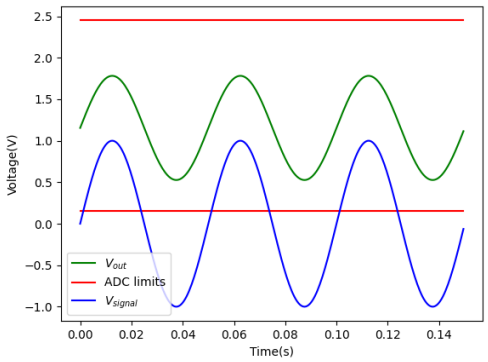

Fig. 5: Plot for eq. 2 to read the CT

It should be noted that the water pump can reach peaks higher than 20 A. According to the datasheet the maximum current is 20 A [11] so at this point the CT would get saturated. Also for the waterpump such high current pulses are very high frequency, so in case they could be read these would be filtered by the LPF.

2) Differential voltage probe measurements: As the output of the differential voltage probe is also a sinewave with positive and negative components, the topology of the circuit from Fig. 4 will be used again. The values used can be seen in table III.

TABLE III: Values for the differential probe voltage divider

Component	Value
R1	$15k\Omega$
R2	$9.49k\Omega$
R3	$9.38k\Omega$
D1,D2	SR240

Using the values from table III combined with the equation 1 will give the next equation:

$$V_{out} = 0.239 \cdot V_{signal} + 1.25(\text{V}) \quad (3)$$

Taking into account that the differential probe works with the ratio 1:100 probe (this means for 100 V input, the output of the probe is 1 V), so for the grid, which had a maximum voltage $V_{grid_P} = 353$ V, the output is $V_{signal_P} = 3.53$ V. This V_{signal_P} is the input of the circuit in Fig. 4 which output can be seen in Fig. 6.

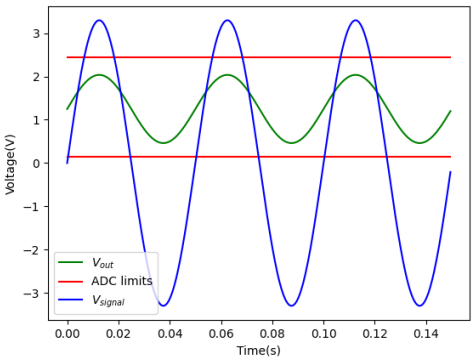

Fig. 6: Plot for eq. 3 to read the differential voltage probe

E. 500 Hz Low-pass filter

Although the current gets filtered by the CT, a LPF will still be used to get a lower cutoff frequency.

This filter will have a cutoff frequency of around 500 Hz, as a filter with a close value has been proven to work for similar power measurements in the paper [10], yet to be published, where the objective was also to filter the higher frequency components to provide reliable measurements. Also this value is due to the fact that for a sampling frequency was decided to be 1 kHz, the maximum signal sampled can be 500 Hz ($f_{sampling} \geq 2 \cdot f_{signal}$) [12], so anything above that will be filtered.

To make this filter, a capacitor is added to the circuit in Fig. 4.

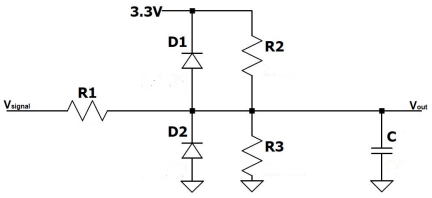

Fig. 7: Low-pass filter topology

A simple RC LPF could not be used directly, as resistors were already being used to influence the signal. Therefore, once the capacitor was added, the transfer function of the signal would be defined by the next equation:

$$H(S) = \frac{\frac{R_2R_3}{R_1R_2+R_2R_3+R_1R_3}}{1 + S \cdot C \cdot \frac{R_1R_2R_3}{R_1R_2+R_2R_3+R_1R_3}} \tag{4}$$

Where the cutoff frequency will be given by:

$$W_{cutoff} = \frac{R_1R_2 + R_2R_3 + R_1R_3}{CR_1R_2R_3} \tag{5}$$

To filter at 500 Hz, the desired value of the capacitor will be 1.53 μF; therefore, a 1.5 μF capacitor will be used in the circuit, which has a cutoff frequency of 510 Hz.

Taking the Bode plot from the simulation made in LTSpice for frequencies from 0Hz to 10 kHz, as it can be seen in Fig. 8, it can be seen that the cutoff frequency is indeed at around 510 Hz.

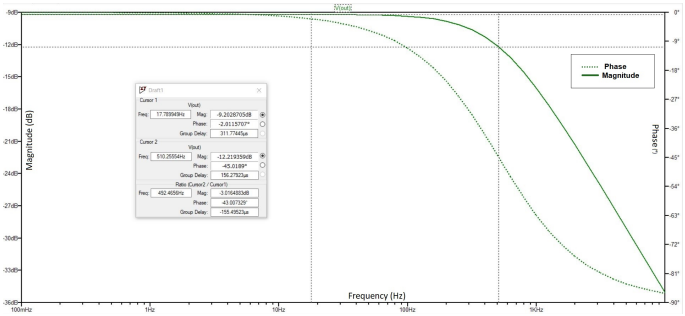

Fig. 8: Simulated Bode plot for the voltage shiftter

This is confirmed by the theoretical plot obtained, as shown in Fig. 9, where the simulation was also made for frequencies from 0 Hz to 1 kHz.

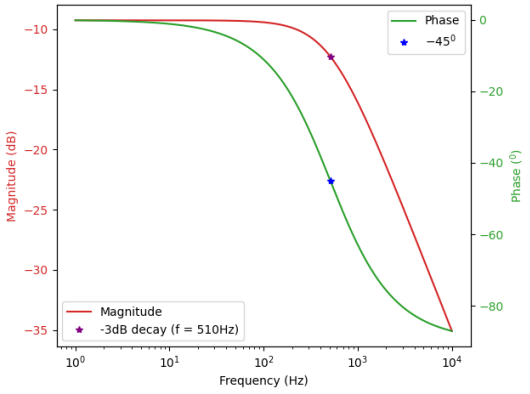

Fig. 9: Theoretical Bode plot for the voltage shiftter

As a phase delay should be compensated to avoid inducing errors, software compensation should be applied. To avoid this, as it would add complexity to the code, a filter of the same order and cutoff frequency was also applied to the voltage signal. This will add the same phase delay to both measurements.

As it can be hard to get the same phase delay due to the avaliable values of capacitors, the objective will be to obtain a phase delay as close as possible at 50 Hz.

As it can be seen from the phase in the Bode diagram in Fig. 9, the phase delay at 50 Hz is 5.70^0.

For this, following the same steps as before, a filter was designed with a cutoff frequency of 510 Hz. The desired capacitor value would be 88 nF. In the real circuit a capacitor with the value of 82 nF is used, which gives a phase delay of 5.38^0 for 50 Hz.

F. Alternative design for AC readings

An alternative design could also have been used to measure the current and grid voltage.

This design would use a full bridge rectifier. This flips the negative voltage into positive so the microcontroller can read the signal. To differenciate the negative from the positive parts of the signal, zero cross-detection can be used.

Although this method to read the AC voltages could seem more straightforward, the reality is that the voltages the

ESP32 ADCs can capture does not go as low as 0V, as it can be seen from the table I.

This minimum voltage can be a problem, as it can be deduced from Fig. 11, a simulation from the circuit in Fig. 10. It can be seen that the signal read by the microcontroller misses some information as it clips when trying to reach 0V.

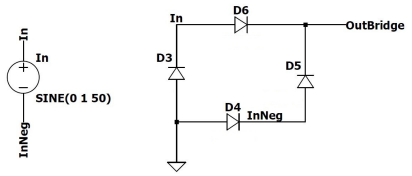

Fig. 10: Full bridge rectifier circuit

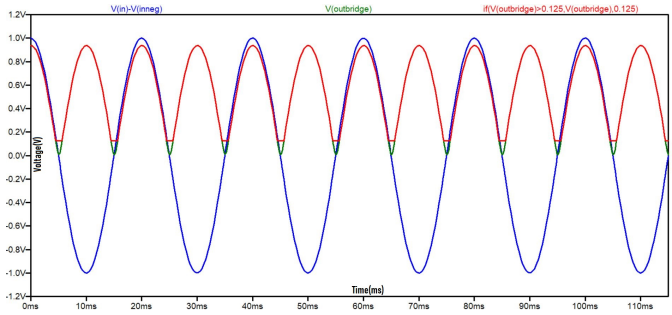

Fig. 11: Simulation with the input sinewave (blue), the output of the full bridge rectifier(green) and the signal the ESP32 would read(red)

This method could be used with an STM32F4, where the ADCs of this family of microcontrollers can read from 0 V.

III. Software Design

In this section a design and analysis of the software will be given. A flowchart with the main functionality of the code can be seen in Fig. 12.

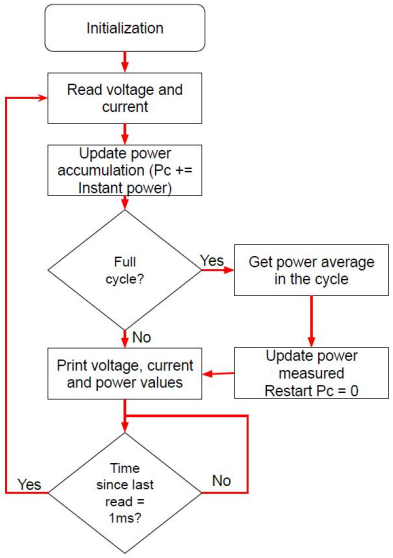

Fig. 12: Flowchart of the code's functionality

A. Code functionality

The microcontroller will be programmed in C language using the Espressif extension in Visual Studio Code. This is the extension from the official coding IDE from the manufacturer of the product.

The microcontroller works as follows:

It uses the 2 ADCs and a timer interrupt to sample constantly at 1 kHz. On top of this, the UART baud rate was configured at 460800 bps to send the data through a USB port to the computer fast enough so it does not interfere with the sampling. The code has an option to use multisampling for the ADC readings to compensate for the accuracy in readings, that option is not used to avoid introducing a phase delay.

The code starts by configuring the system's hardware to work as intended, configuring the UART0, the ADC1, ADC2 and Timer0. In addition, the watchDog, which restarts the code if it is not correctly handled, is disabled to simplify the code.

The code gets to an infinite loop where it waits and runs when the interrupt of the Timer gets activated.

In the interrupt, the ESP32 gets the readings of the 2 ADCs. It calculates the real value of the signal from the attenuated measurements. After every 20 readings, which is one complete cycle, it updates the value of power(using the formula $P = \sum_{n=1}^{20} V_n \cdot I_n$, where n is each reading) and sends this value along with the readings of current and voltage.

B. Data gathering

In this project, it is essential to save the data so it can be analyzed and validated. For this, a code with Python was made where the serial port of the computer is read and the data saved in a .csv file.

This code also has the functionality to process the gathered data and plot it so it is easier to analyze and validate it.

IV. Measurements

To validate the measurements taken, parallel to the setup from Fig. 1, the precision power analyzer Yokogawa WT5000 was used as a reference. The measurement setup can be seen in Fig. 13.

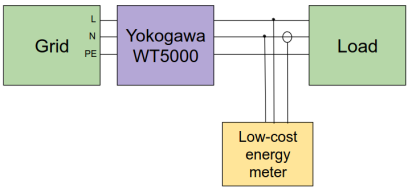

Fig. 13: Measurement and validation setup

This power analyzer monitors current and voltage to measure power precisely, with 0.3% precision for AC readings [13]. The measurements for this are taken in a period of time using Matlab. The output of this script gives the average power consumption in the time taken. This data will be compared to the data obtained from the measurements with the microcontroller.

As loads for the experiments, three types will be used for validation: no load, linear and non-linear loads.

The no load setup will be used in order to check for major deviations present in the system.

As a linear load, a heater with multiple configurations will be used. This heater has a knob that can be turned to get different consumption levels.

As a non-linear load, a water pump will be used. This water pump has up to 10 configuration levels that can be changed using a remote controller.

The measurements were taken for current and voltage, and used to calculate the power in the ESP32. As previously explained, the power would be calculated per cycle; this means in 20 measurements taken per cycle (as $n = f_{sampling}/f_{signal}$), the power would be

$$P = \frac{\sum_{n=1}^{20} V(n) \cdot I(n)}{20} \tag{6}$$

The information on the three parameters, voltage, current and power, is then sent to the computer. At the same time, the Yokogawa WT5000 was used as a reference to get an average of the power consumed (these measurement are not fully simultaneous as it was not possible to syncronize the readings).

These measurements are post-processed (PP), where the DC offset of the voltage signal is eliminated using Python to check for the validity of the data.

The decision to neglect the DC voltage is based on its nature. A differential probe is unable to measure DC components because its output, given by $V_{out} = V_+ - V_-$, cancels out any DC component. Therefore, any DC offset does not originate from this probe.

Also, for the current probe, being a CT makes it only able to read AC currents; as for the DC currents, it would have a 0 output, making it also impossible to have a DC offset that is not induced by the shifter circuit.

Despite from the previously explained, the appearance of this DC offset makes sense, as the voltage shifter circuit of Fig. 4 induces a voltage offset to be able to read the entire wave. The offsets in the components values make it very hard to calculate the DC offset exactly. It should also be taken into an account that the temperature of the components can change their values.

Due to this, the post-processing data will be the one analyzed for validation. Later, it will be explained how to get rid of the DC offsets in future measurements so post-processing is not needed to obtain the same measurements.

The measurements will be presented then as non-processed data, which will be the ones marked ESP32 measurements, PP data, marked as PP and the Yokogawa WT5000, marked as Y measurements. This will allow to make a good comparison and show the importance of taking out the DC offset from the measurements.

The experiments made will be presented in the following order; first, no load measurements; after this, measurements for linear loads, before and after adding a LPF, and finally, non-linear loads.

For this, the testing setup is the one that can be found in Fig. 13. Fig. 14 shows a picture of the actual lab setup.

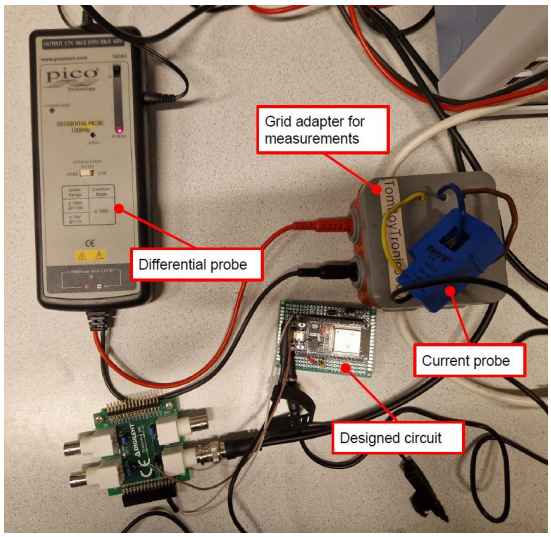

Fig. 14: Lab setup

A. *Measurements for no load*

To validate the system, no load measurements were taken. This can ensure that there are no significant deviations that could result in future mistakes.

Fig. 15 shows one of the measurements presented for the no load configuration.

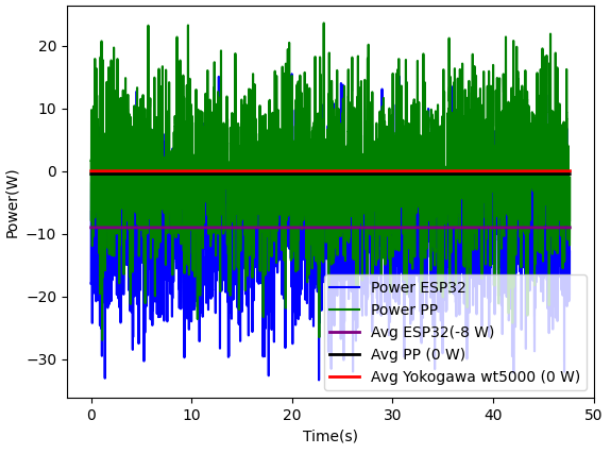

Fig. 15: No load measurement

Table IV shows the power measured in the different experiments done with no load.

TABLE IV: Power in the no load configuration

Time (s)	P_{ESP32} (W)	P_{PP} (W)
9.5	-8.91	-0.64
12	-8.99	-0.60
47.5	-9	-0.48
64	-25.8	-0.61

As shown in table IV, there is a pretty constant deviation when measuring the no load setup of around -0.6 W for the PP data. In the table it can also be seen that the measurements from the ESP32 have a much more considerable deviation. This finds its explanation on the DC offset component from both measurements.

The DC offset would not be a problem, but due to the non ideality of the components, the DC offset was not a

straightforward calculation. The resistors value can vary with changes in the temperature of the circuit and a small value of them can have a big impact on the DC offset when re-scaling. Also the parasitic resitance from the capacitor was not taken into an account.

It can be seen from Fig. 15 that the readings have a lot of noise, as the power measured oscillates between values around ± 20 W having an average over time close to 0 W. These high peaks come from the low accuracy of the ADC.

This can still be considered an acceptable value as it is very low (especially when comparing this offset to the total consumption a household could have in the long term), and it seems by the experiments done that it is very constantly close to 0 over time.

Although the value seems very stable in the different measurements, it cannot be proven to be just a coincidence, and therefore adding 0.6 W to compensate for this deviation could end up being a new source of errors.

B. Measurements for linear loads

This section shows the measurements for the linear loads before and after adding the LPF. The obtained results can be seen in table V.

TABLE V: Errors measured in linear loads

Load type	LPF	Load size (W)	Error ESP32 (%)	Error PP (%)
Heater (Linear)	No	1800	0.57	0.32
		800	3.79	0.92
		310	5.67	-0.28
	Yes	1800	0.94	0.50
		800	4.66	0.69
		310	6.00	-0.82

1) Measurements with no LPF: Proving that the energy meter works for linear loads before adding the filtering is necessary to validate the final system. For this, a couple of experiments using an electric heater were made.

The three experiments done were for the heater with a 1800 W configuration, 800 W configuration and 310 W configuration. The results can be found in the table V.

The measurement for 1800 W configuration can be seen in Fig. 16. The rest of the measurements can be found in the appendix A.

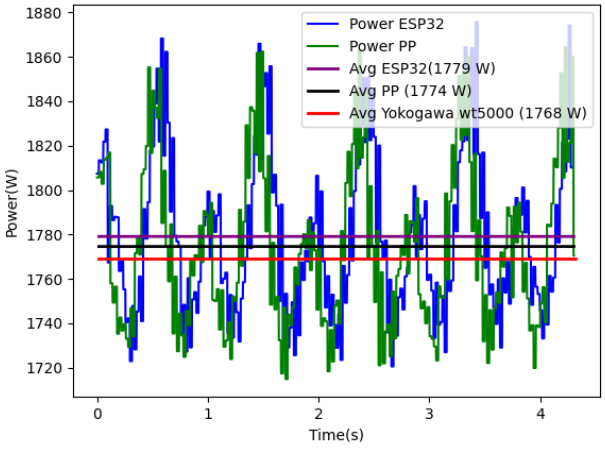

Fig. 16: Measurements for heater in 1800 W configuration

Taking a look at the table V it can be seen that both measurements are close to the reference measurement given by the Yokogawa, being the PP measurement slightly better.

As the next results are checked, it can be seen that the PP measurements are always under 1% of deviation. In contrast, as explained in the previous sections, the deviation in power calculated by the ESP32 grows the smaller the power consumption is (3.76% for 800 W configuration and 5.67% for 310 W configuration).

This makes sense as the DC deviation will have more weight in smaller measurements.

2) Measurements with LPF: Once the filter was added, a new set of measurements were taken.

First, a measurement of current and voltage was done to ensure that the filters used provided the same phase shift or a neglectable difference. This can have a big impact when dealing with non-linear loads, as the peaks of current are dependent in the phase of the voltage to give the correct power measurement. This result can be seen in Fig. 17.

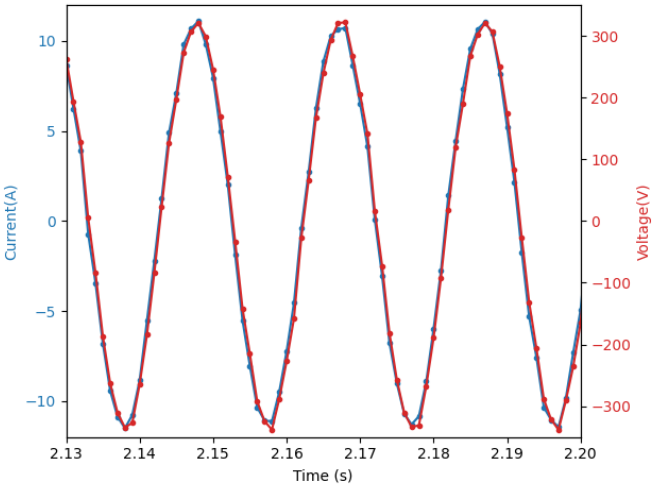

Fig. 17: Current and voltage after filtering

As seen in Fig. 17, the phase difference between them is negligible. For this reason, as expected, there was no need to compensate digitally for the phase difference.

After this, the same experiments as in the previous setup were taken. The objective of this is repeating the experiments to double-check if the system needs again calibration.

For 1800 W, the results can be seen in Fig.18.

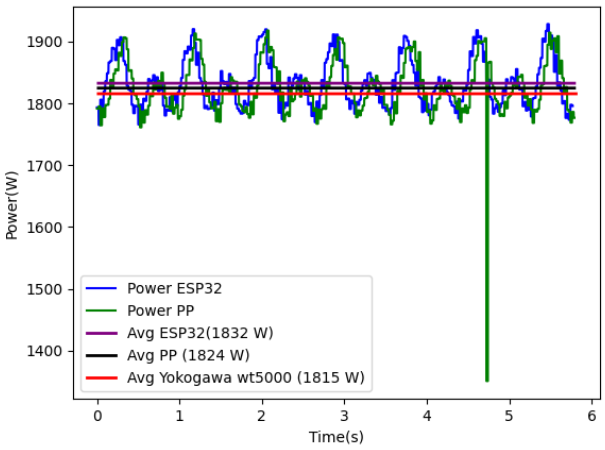

Fig. 18: Measurements for heater in 1800 W configuration with filter

As seen in table V, the results are within the expectations, being very similar to the ones obtained for the measurements with no filter.

In Fig. 18 it can be seen a dip of consumption in the PP data, which cannot be seen in the measurements of the ESP32, most likely this could be due to an error in the communication between the microcontroller and the computer, as the baud rate is half the maximum baud rate that the microcontroller supports and the faster the baud rate the easier it is to have transmission errors (like bit errors).

This statement can be supported by Fig. 19 where it can be seen that the voltage does not follow (only in one point) the expected value for the grid, and this transmission error would explain the difference of the dip in the PP power and the power calculated by the ESP32.

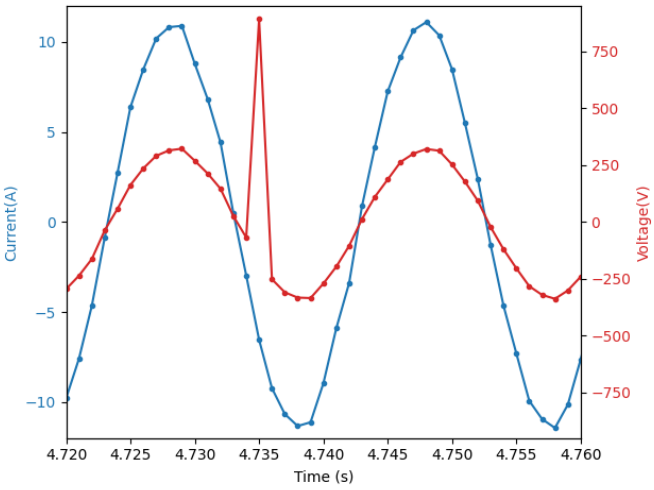

Fig. 19: Error in current measurement from Fig. 18

As it can be seen from these experiments, all the reading have been more than satisfactory having a maximum deviation of less than 1%.

C. Measurements for non-linear loads

Lastly, the measurements for non-linear loads were made. For this, a water pump with different consumption levels was used. The experiments were done for three out of the ten possible consumption levels, 10, 5 and 1.

These are the most important measurements of the project, as they are the ones causing problems in the measurements of the static meters and the aim of this project is to prove how to measure them.

To begin a comparison between the current and voltage readings will be displayed. For this the picoscope was used to capture similar measurements as a reference for comparison.

Later the power measurements will be shown taking again the Yokogawa WT5000 as reference for the measurements.

1) Currents and voltage in non-linear loads: As the importance of the non-linear loads rests on the current measurements, some current and voltage measurements from the power measurements displayed to be studied.

Firstly the current in the level 10 configuration can be seen in Fig. 20

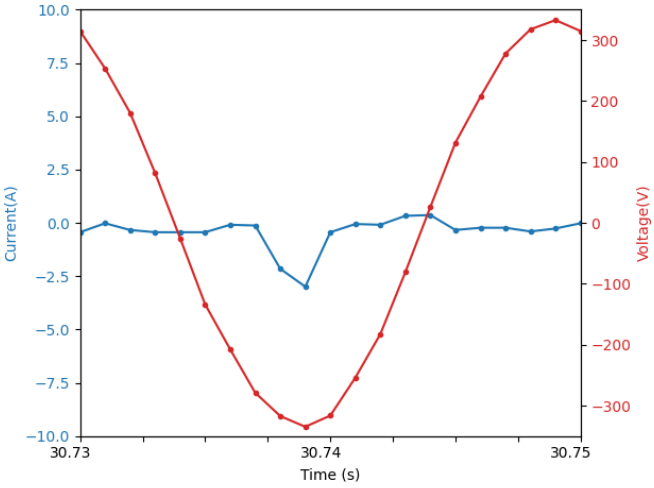

Fig. 20: Current and voltage measurement from power (level 10) in Fig. 39

For comparison, current and voltage measurements were also taken using the picoscope (not simultaneously). The result can be seen in Fig. 21.

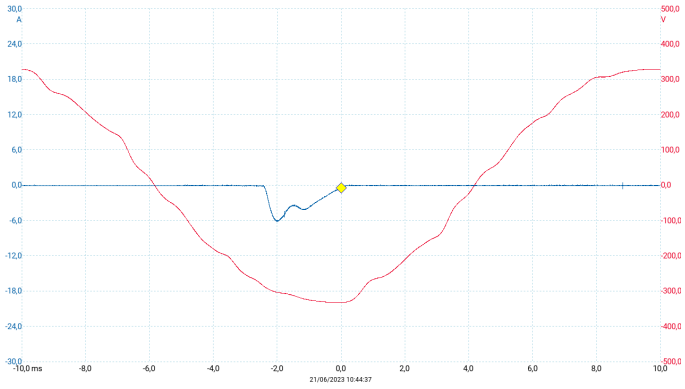

Fig. 21: Current and voltage measurement for water pump in level 10

Checking Fig. 20 it can be seen a spike made by 2 points, which can be considered similar to the one in Fig. 21 captured by the picoscope, although it seems some information is still missing. They are not the same amplitude, but this could be because they are not synchronized, and the consumption for the water pump is also not a constant value.

It should be noted that the water pump changes the phase of the peaks and the amplitude to get different instant power, so at different times, for the same level, we can still get different readings with the same tools.

It can also be seen that the zero-current measured by the microcontroller has some significant noise. This values of this noise in the current should cancel out in extended periods, averaging 0, as it happens for measurements with no load.

It is also essential to check the signals measured for the lower levels of functionality of the water pump. These levels have been the ones to give the most problems in other research where the deviations came up to be, as said in the introduction, up to 2675% [4].

Therefore the same procedures were made for the water pump in levels 5 and 1. The figures for level 5 can be found in the appendix A.

For level 1, the measured current and voltage can be seen in Fig. 22 and the readings from the picoscope can be seen in Fig. 23.

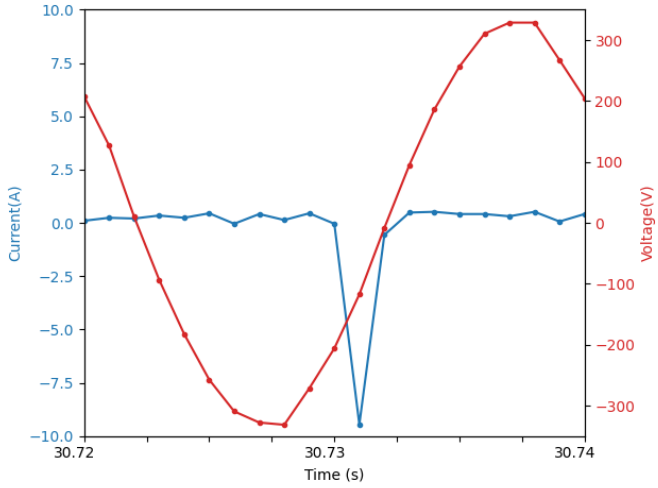

Fig. 22: Current and voltage measurement from power (level 1) in Fig. 27

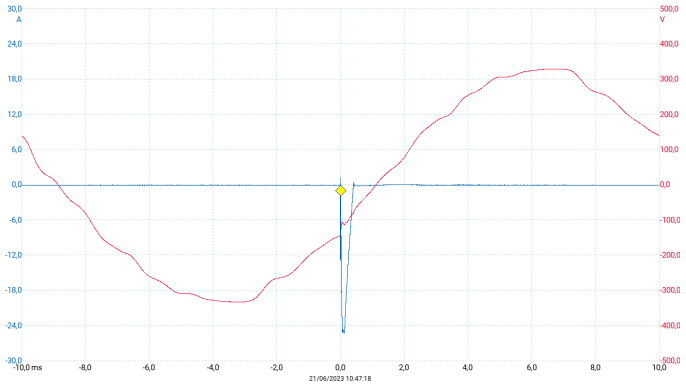

Fig. 23: Current and voltage measurement for water pump in level 1

In this case, again, the plot from Fig. 22 shows a very similar result to the one found in Fig. 23. The current peaks are much bigger than for level 10, but also higher frequency, as they get captured only with one measurement. Also, these peaks are found in a phase where the grid voltage is much lower than for the level 10 configuration.

It is also good to check that the current captured by the picoscope was up to 24 A at some points, while the maximum read by the microcontroller was never above 15 A. Althought different probes were used, for the picoscope and the ESP32, and as said in the hardware analysis the CT used cannot read more than 20 A, this has its origin in the filter, as the point where it measures 24 A (where the CT should measure 20 A) is very short in time, so it gets filtered.

Another difference can be seen not in the current measurement but in the voltage, as it can be seen the voltage has a dip in the readings gotten from the picoscope in Fig. 23, which is not captured in the measurements done with the microcontroller, this is expected as the dip is high frequency and the sampling only allows it to be represented by one point, the voltage there can truly have a dip, but it will not be visible. For these voltage measurements the same differential probe wa used, so the differences are more visible in these measurements.

In some measurements some change in the DC offset of the current can be seen. Fig. 24 is an example where this DC offset is seen.

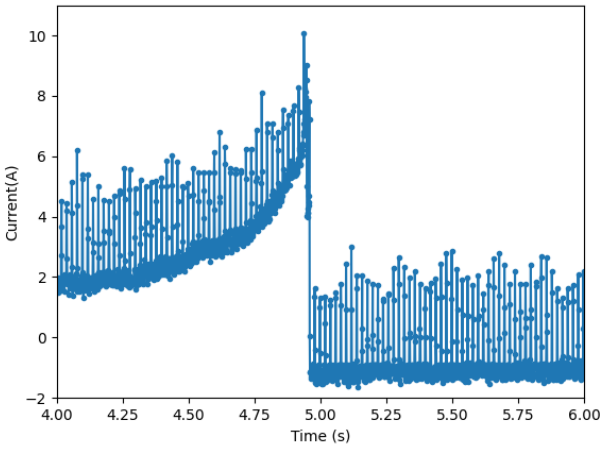

Fig. 24: Two seconds of current readings from Fig. 35

This did not seem to cause a negative effect on the readings, as this still gave similar power readings as before because there was a positive and negative peak in power that got cancelled, as it can be seen in Fig. 35 in the appendixA, so it can be clearly said that these high frequency changes in DC offset get cancelled in the PP measurements, not so much in the ESP32 measurements where it can be seen a DC component in the power as well.

2) Power measurements: The power calculated for these experiments can be seen in table VI

TABLE VI: Errors measured in non-linear loads

Water pump level (Non-linear)	Time (s)	P_Y (W)	P_{ESP32} (W)	E_{ESP32} (%)	P_{PP} (W)	E_{PP} (%)
10	15	99.8	53	-46.5	96.6	-3.24
	15	102.5	57.7	-43.7	97.4	-4.95
	20	116.1	101.3	-12.8	109.1	-6.05
	40	135.7	95.0	-30	106.2	-21.7
	40	132.1	86.5	-35.2	108.3	-18.0
	40	116.5	94.7	-18.7	106.1	-8.93
5	40	67	52.9	-21.0	63.5	-5.25
	40	68.1	53.2	-21.8	63.8	-6.28
1	40	26.9	19.4	-34.3	31.6	6.61
	40	26.9	22.9	-22.5	32.3	9.09
	40	18.4	22.0	19.5	32.0	74

.

The errors calculated do not fall into the expectations.The minimum error for PP measurements is 3.24% and up until -74%. Although many of the measurements are below 10% of deviation this seems to be more arbitraty than a patern. At the same time, the power calculated by the ESP32 has a minimum error of -12.8%, being the highest -46.5%.

Firstly the consumption for level 10 was measured. Six different measurements were taken, two for 15 seconds, one for 20 seconds, and three for 40 seconds, to avoid deviations due to the Yokogawa not being fully syncronized with the ESP32. The relevant data can be seen in the table VI, and one of the measurements for 40 seconds can be found in Fig. 25. The rest of the measurements can be found in the appendix A.

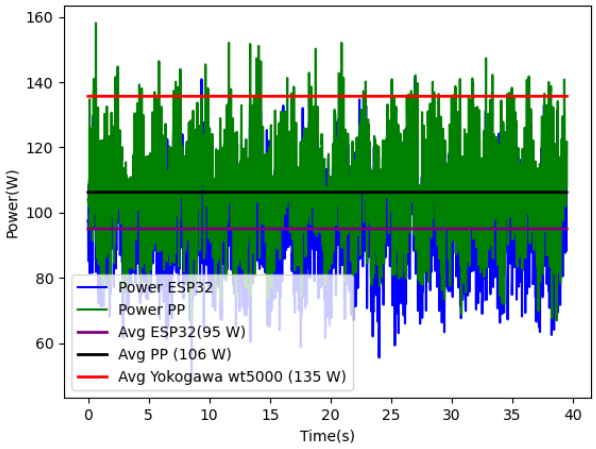

Fig. 25: Measurement water pump level 10 ($P_Y = 135.7W$)

The measurements are displayed in chronological order. First the measurements for 15 and 20 seconds were taken. These measurements had an acceptable value, having their highest deviation at 6.05% in PP data. But after the measurements for lower configurations it was seen that the non-syncronization between the ESP32 and the Yokogawa WT5000 was causing big discrepances. Therefore the measurements were also taken for longer time periods.

The new measurements for 40 seconds showed a bigger devation than expected taking into an account that the previous had been much better.

After this, two measurements for the water pump in level 5 were taken. These measurements were done for 40 seconds. The results can also be found in table VI, and one of the measurements can be seen in Fig. 26.

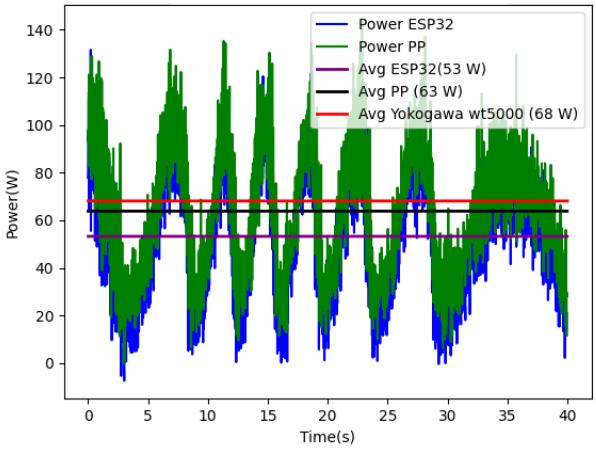

Fig. 26: Measurement water pump level 5 ($P_Y = 68.1W$)

The last set-up for measurements was done using level 1. Three measurements were done for 40 seconds each.

The results can be found in the table VI, and one of the measurements can be seen in Fig. 27.

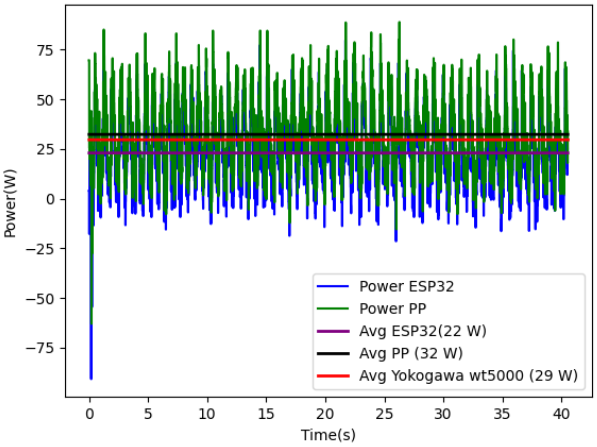

Fig. 27: Measurement water pump level 1 ($P_Y = 29.6W$)

These measurements also show a similar performance than before, having one of the experiments for level 1, which is the most challenging measurement, up to -74% of deviation.

From the table VI, it can be seen that the values of the measurements are very repetitive. This could find an explanation in the peaks being captured at the exact moment of the voltage phase, having similar power consumption per cycle. This makes the most probable problem that the microcontroller cannot capture important parts of the signals when sampling at this frequency, such as the voltage dip of the grid or some parts of the current signal.

V. Next steps

This section discusses what could have been done differently, knowing what is known after this research.

As seen from the measurements, the DC offset introduced by the voltage shifter was a big problem throughout the process and could only be solved with post-processing.

This problem could be solved by saving significant amounts of data and then processing them similarly to

how it is done in Python. This would take time where the ESP32 stops reading unless the double core is partitioned to handle both tasks simultaneously, but this could still impact the performance other tasks. Therefore, a more suitable direction is using the alternative measurement technique shown in the analysis, the full bridge rectifier.

For this, a change of microcontroller should also be done, as it was shown in the analysis part that the ADCs of the ESP32 cannot read up to 0 V value, which would be necessary for this application.

The STM32F4 family of microcontrollers is a suitable option, they are still a widely available microcontroller, and many models in this family of STM32 microcontrollers have multiple ADCs that can read 0 V signals.

As discussed, the sampling frequency should be increased to capture more information about the signal. This should be easily doable in both ESP32 and STM32F4, but it needs to have some constrains taken into an account. During the project, much time was spent coding the communication between the ESP32 and the Python script to save and process the data. There were problems with the class of the data sent by the microcontroller, which directly impacted the transmission time and, as a result, the system's capability to sample at the correct frequency. Therefore this should be taken into an account when applying these changes.

Multisampling is also an option that should help filter the high-power spikes produced by the inaccuracy of the ADC and give a more reliable power measurement in shorter periods. This could be applied by measuring both channels alternatively on each sample so it does not need compensation for the phase shift, instead of how it is now designed in the code, which, if used, still needs compensation for the phase difference.

VI. Conclusion

This research aimed to create a non-intrusive robust low-cost energy meter focusing in the low-frequency components of the current measurements. For this, the least amount of extra hardware had to be used and still be able to read correctly, especially the non-linear loads, which have been a source of severe mistakes in energy measurements.

For this, an ESP32 was combined with a low-cost CT, a differential voltage probe and some extra hardware based on resistors and capacitors to design a LPF. Although the differential probe will not be used in the future to read voltages, as it is pretty expensive, the main objective of this paper is to focus more on the current measurements.

The measurements show that linear loads have very satisfying results after post-processing (always less than 1% deviation). Still, for the non-linear loads, the results could be more satisfactory (never less than 3.24% deviation).

The problems with non-linear loads, as discussed in the paper, are most likely due to the low sampling that prevents the microcontroller from capturing all the information needed from the signal. Considering that the filtering is done at 500 Hz, the sampling rate was set to the minimum allowed value to capture the signal's information, which is 1 kHz. A higher sampling frequency (while maintaining the filtering) would allow to capture more points of the signal.

As discussed, the next step this project should follow is changing the microcontroller to one that can read from 0 V. This will allow the use of a full-bridge rectifier for the measurements avoiding the need to post-process the data in the computer to get rid of the DC offsets that appear in the current and voltage measurements. An STM32F4 with at least 2 ADCs fits the project well.

Using a higher sampling frequency should also be one of the next steps, but some constraints with the transmission of data to the computer should be considered as it might interfere with the sampling frequency.

Although the power measurements of the non-linear loads were not as precise as expected when starting the project, it can be seen from the plots of current and voltage that the individual measurements show a lot of potential.

Therefore the objective of this project, of proving that a robust low-cost energy meter can be done, still needs some more time and testing to be entirely answerable. Nevertheless, it shows a lot of promise with the proposed changes in hardware and software.

Glossary

CT Current transducer. 2, 3, 6, 9, 11

LPF Low-pass filter. 2–4, 6, 7, 11

PP Post-processed. 6, 7

References

[1] T. H. Ortmeyer, K. R. Chakravarthi, and A. A. Mahmoud, "The effects of power system harmonics on power system equipment and loads," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104,, no. 9, pp. 255–2563, Sep. 1985.

[2] R. B. Timens, F. J. Buesink, V. Cuk, J. F. Cobben, W. L. Kling, and F. B. Leferink, "High harmonic distortion in a new building due to a multitude of electronic equipment," in *IEEE International Symposium on Electromagnetic Compatibility*, 2011, pp. 393–398.

[3] L. Cividino, "Power factor, harmonic distortion; causes, effects and considerations," in *[Proceedings] Fourteenth International Telecommunications Energy Conference - INTELEC '92*, 1992, pp. 506–513.

[4] B. Ten Have, T. Hartman, N. Moonen, C. Keyer, and F. Leferink, "Faulty readings of static energy meters caused by conducted electromagnetic interference from a water pump," Jul. 2019, 17th International Conference on Renewable Energies and Power Quality 2019, ICREPQ 2019 ; Conference date: 10-04-2019 Through 12-04-2019.

[5] Z. Marais, H. van den Brom, G. Rietveld, R. van Leeuwen, D. Hoogenboom, and J. Rens, "Sensitivity of static energy meter reading errors to changes in non-sinusoidal load conditions," in *2019 International Symposium on Electromagnetic Compatibility - EMC EUROPE*, 2019, pp. 202–207.

[6] European Committee for Electrotechnical Standardization, "Electricity metering equipment (a.c.) - part 3: Particular requirements - static meters for active energy (class indexes a, b and c)," Standard EN 50470-3:2006/A1:2018, 2018.

[7] T. Hartman, R. Grootjans, N. Moonen, and F. Leferink, "Electromagnetic compatible energy measurements using the orthogonality of nonfundamental power components," *IEEE Transactions on Electromagnetic Compatibility*, vol. 63, no. 2, pp. 598–605, 2021.

[8] Microchip Technology Inc., *ATmega328P Datasheet*, Microchip Technology Inc.

[9] "Esp32 datasheet: https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html."

[10] B. N. Tom Hartman, Martin van Mast and Frank Leferink, "Improving static energy meter robustness against conducted emi via a basic low-pass filter," notes: Yet to be published.

[11] Digi-Key Electronics, "SEN0211 Datasheet," Digi-Key Electronics Website. [Online]. Available: https://www.digikey.jp/htmldatasheets/production/2061125/0/0/1/sen0211.html

[12] Chris Zeinstra, "Digital signal processing class slides. university of twente."

[13] Yokogawa, "BUWT5000-01EN: Power Analyzer - WT5000," Yokogawa Website, YEAR. [Online]. Available: https://cdn.tmi.yokogawa.com/BUWT5000-01EN.pdf

APPENDIX

A. Additional measurements

In this section all the measurements can be found.

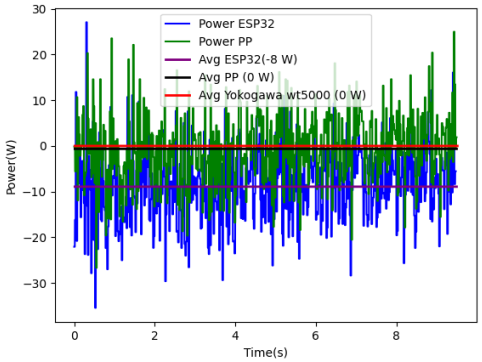

Fig. 28: First no load measurement

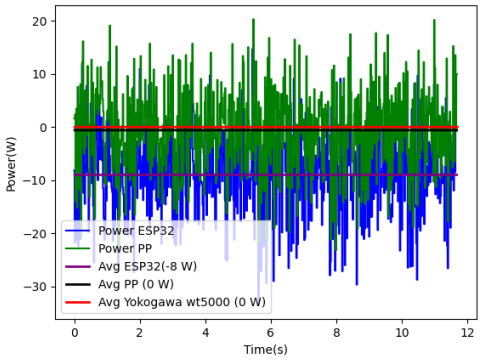

Fig. 29: Second no load measurement

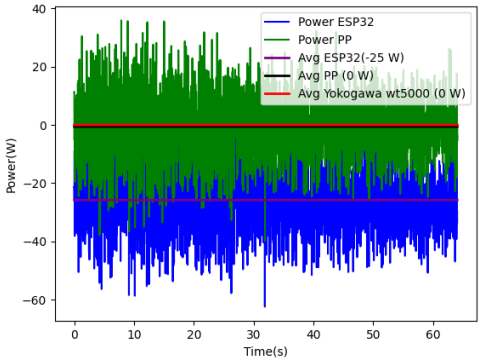

Fig. 30: Fourth no load measurement

1) Measurements for no load:

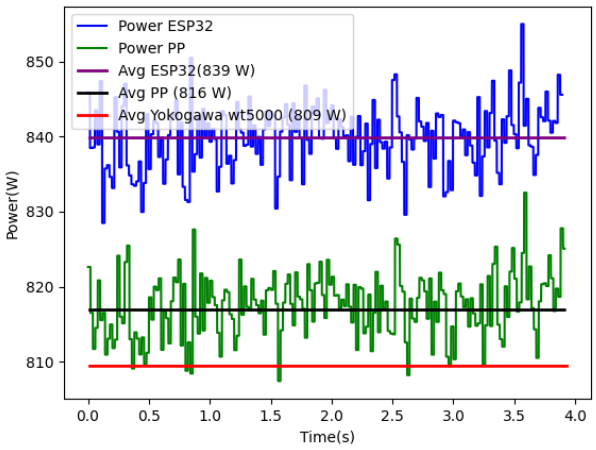

Fig. 31: Measurements for heater in 800W configuration

2) Measurements for linear loads: 32

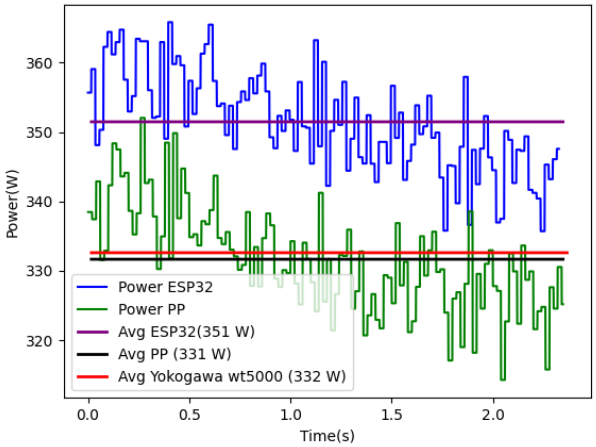

Fig. 32: Measurements for heater in 310W configuration

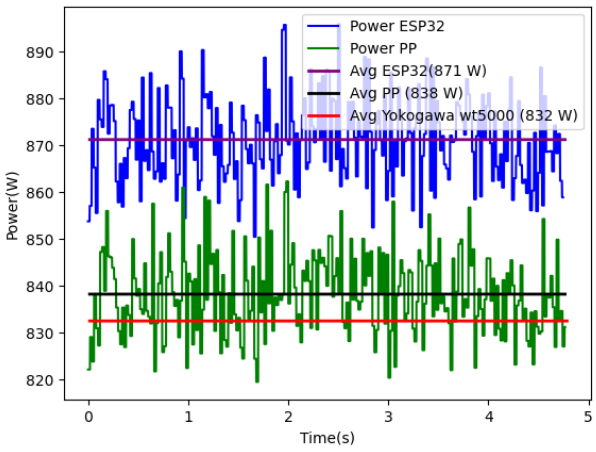

Fig. 33: Measurements for heater in 800W configuration with filter

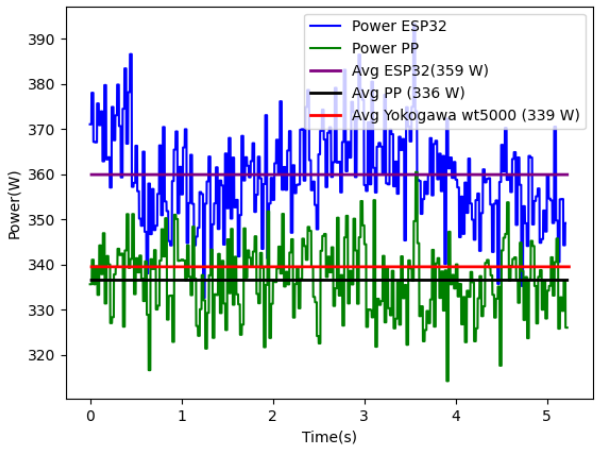

Fig. 34: Measurements for heater in 310W configuration with filter

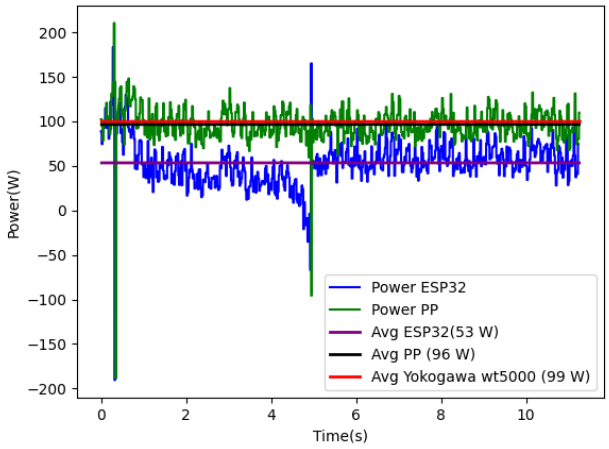

Fig. 35: Measurement water pump level 10 ($P_Y = 99.8W$)

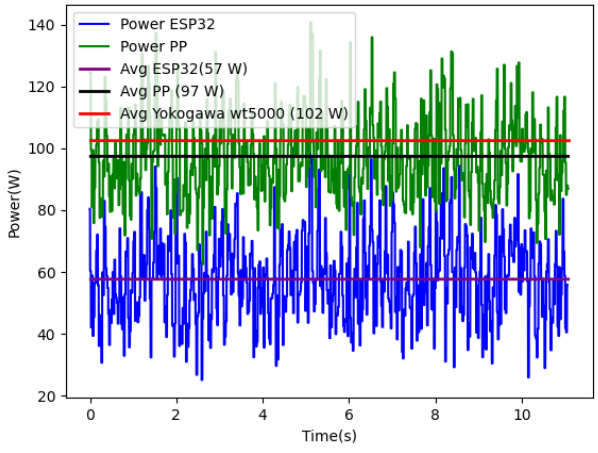

Fig. 36: Measurement water pump level 10 ($P_Y = 102.5W$)

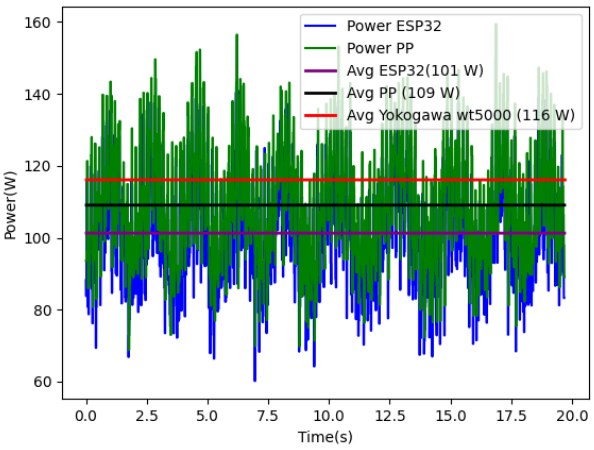

Fig. 37: Measurement water pump level 10 ($P_Y = 116.1W$)

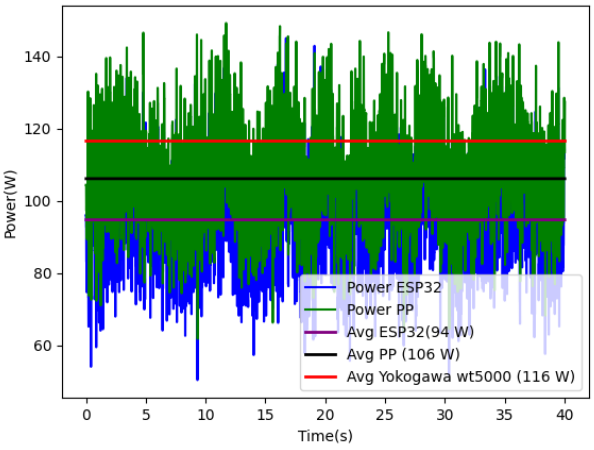

Fig. 38: Measurement water pump level 10 ($P_Y = 116.5W$)

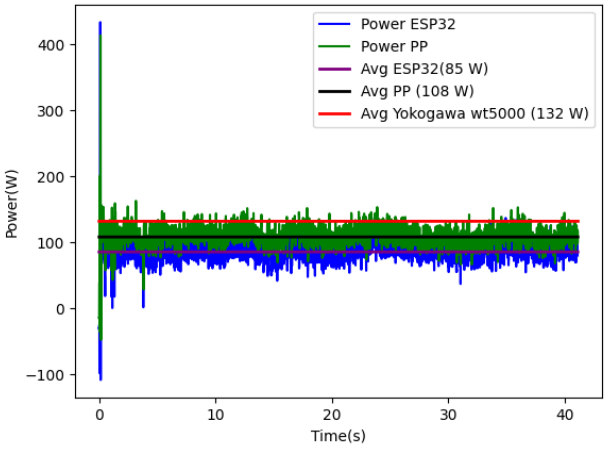

Fig. 39: Measurement water pump level 10 ($P_Y = 132.1W$)

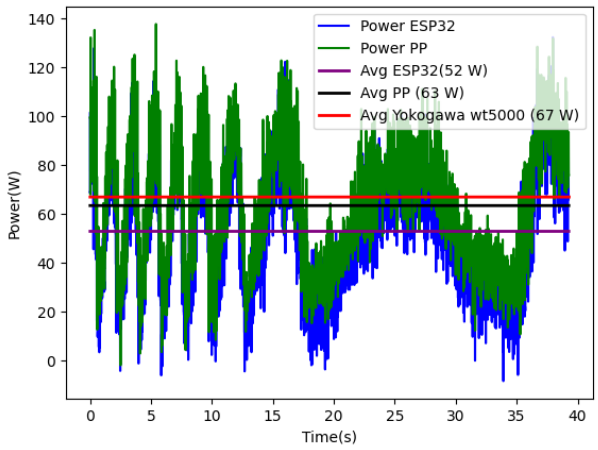

Fig. 40: Measurement water pump level 5 ($P_Y = 67.0W$)

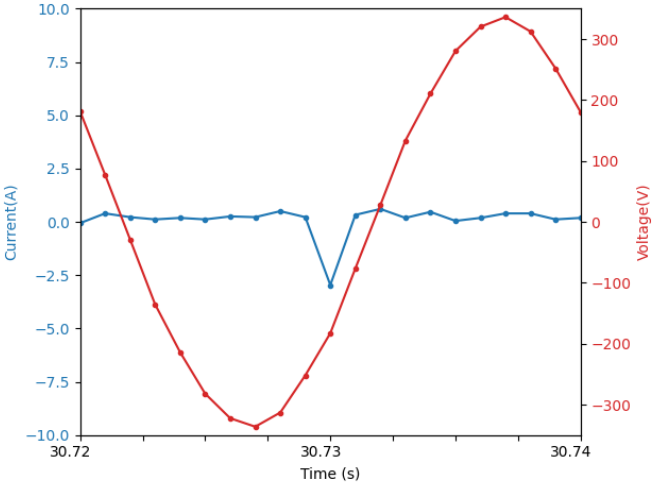

Fig. 43: Current and voltage measurement from power (level 5) in Figure 26

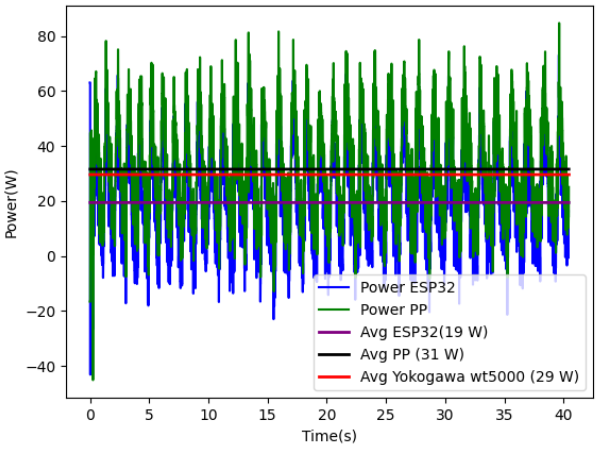

Fig. 41: Measurement water pump level 1 ($P_Y = 29.6W$)

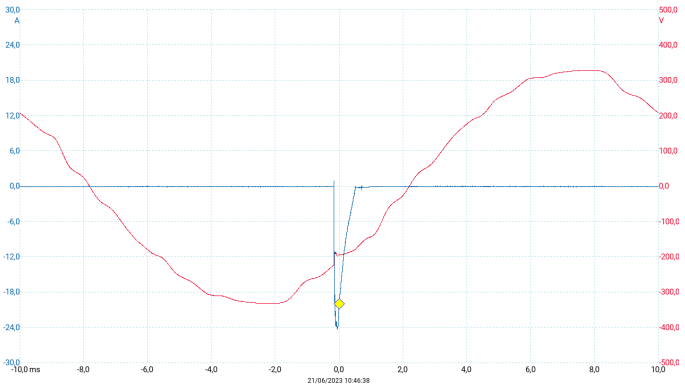

Fig. 44: Current and voltage measurement for water pump in level 5

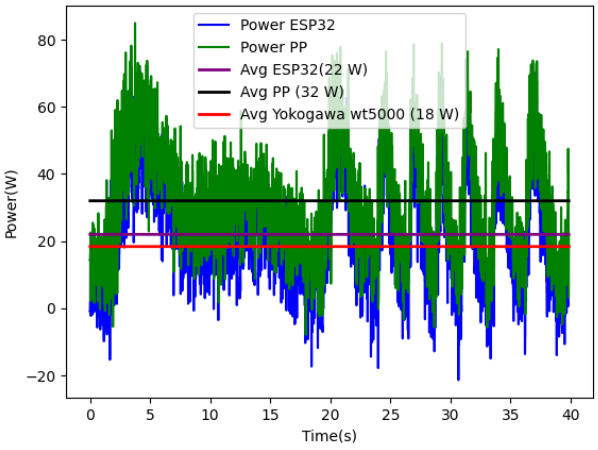

Fig. 42: Measurement water pump level 1 ($P_Y = 18.4W$)

3) Measurements for non-linear loads:

4) Currents and voltage in non-linear loads: 43

B. *Code for the ESP32*

The code developed for the microcontroller is the next

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "soc/soc_caps.h"
#include "esp_log.h"
#include "esp_adc/adc_oneshot.h"
#include "esp_adc/adc_cali.h"
#include "esp_task_wdt.h"
#include "esp_adc/adc_cali_scheme.h"
#include "esp_timer.h"
#include "esp_sleep.h"
#include "esp_system.h"
#include "sdkconfig.h"
#include "driver/gpio.h"
#include "driver/uart.h"


//WatchDog initialization (set to 1 to initialize -> if it's initialized the function to refresh ... needs to be called)
#define WATCHDOG_STATE  0

#define PRINT_STATE     0 //put to 1 to print a line that indicates that the configurations are done

//ADC Channels and attenuation chosen
#define ADC1_CHAN          ADC_CHANNEL_4  //pin D32 of the board
#define ADC2_CHAN          ADC_CHANNEL_0  //pin D4 of the board
#define ADC1_ATTEN           ADC_ATTEN_DB_11
#define ADC2_ATTEN           ADC_ATTEN_DB_11

//Timer and sampling
#define N_MULTISAMPLING  1  //number of readings for each sample (more samples -> bigger phase shift)
#define SAMPLING_FREQUENCY  1000            //select the sampling frequency (in Hz), may have ... problems if set > 2kHz
#define GRID_FREQ       50      //frequency of the grid, needed for some calculations
#define TIME_US_TIMER   1000000/(SAMPLING_FREQUENCY)    //value passed to the timer interruption
#define SAMPLES_PER_CYCLE   SAMPLING_FREQUENCY/GRID_FREQ //number of samples per cycle for power ... cycle calculation

//Attenuation of the resistors attached to transistor and voltage probe
#define TRANSDUCER_BOARD_ATTENUATION   0.5488
#define DIFF_PROBE_BOARD_ATTENUATION   0.245629798
#define VOLTAGE_OFFSET_TRANSDUCER   1400  //mV offset given by the resistor board attached to the ... current probe
#define VOLTAGE_OFFSET_DIFF_VP      1280 //mV offset given by the resistor board attached to the ... differential voltage probe
#define CURRENT_PROBE_SCALING  20        //current probe scaling (current 1:20)
#define DIFFERENTIAL_PROBE_SCALING 100   //diferential probe scaling (current 1:100)


//Baudrate of UART0 communication (if it's too small it might not be able to sample fast,
    // if the sampling frequency needs to be bigger than 2kHz, or there is a lot of information
    // to be sent there is the possibility to boost the baudrate to 922190, according to datasheet)
#define BAUDRATE    460800 //bps


//initialization of functions
static bool adc_calibration_init(adc_unit_t unit, adc_atten_t atten, adc_cali_handle_t *out_handle);
int make_ADC_read(adc_oneshot_unit_handle_t handle_ADC, adc_cali_handle_t handle_CAL,adc_channel_t ... chan, int N_SAMPLES);
static void sampling_timer_interrupt(void* arg);
void timer_INIT();
void ADCs_INIT();
void system_INIT();
void UART_init();
void power_calculation (int current_read, int voltage_read);


//global handles for the use of ADCs
adc_oneshot_unit_handle_t adc1_handle;
adc_oneshot_unit_handle_t adc2_handle;
adc_cali_handle_t adc1_cali_handle = NULL;
adc_cali_handle_t adc2_cali_handle = NULL;



void app_main(void)
```

```
74  {
75     system_INIT(); //Hardware initialization of the microcontroller
76
77
78      while (1); //Code runs in the interruption
79
80
81  }
82
83  //..............USER FUNCTIONS.............................//
84
85  void power_calculation (int current_read, int voltage_read)
86  {
87      float voltage_grid = 0, current_grid = 0;
88      float instant_power;
89      static uint8_t cycle_counter = 0;
90      static float total_power_cycle = 0, power = 0;
91
92      //transform the voltage readings into the proper values of current and voltage of the grid
93      current_grid = CURRENT_PROBE_SCALING * (current_read - ...
            VOLTAGE_OFFSET_TRANSDUCER)/TRANSDUCER_BOARD_ATTENUATION; //scale_probe * (reading - ...
            offset) / transfer_factor
94      voltage_grid = DIFFERENTIAL_PROBE_SCALING * (voltage_read - ...
            VOLTAGE_OFFSET_DIFF_VP)/DIFF_PROBE_BOARD_ATTENUATION;
95
96      instant_power = current_grid * voltage_grid; //calculate the instant power of the circuit
97
98      total_power_cycle += instant_power; //add the instant power to the total power in each cycle
99
100     cycle_counter++;   //each interaction update cycle counter
101
102     if(cycle_counter ≥ SAMPLES_PER_CYCLE)  //When a cycle is completed we update the power value ...
            for that cycle
103     {
104        // printf("%f\n\r",total_power_cycle);
105         power = total_power_cycle / cycle_counter;
106        // otherPower = total_power_cycle/cycle_counter;
107         cycle_counter = 0;
108         total_power_cycle = 0;
109       //  printf("%f,%f,%f,%f\n\r", voltage_grid,current_grid,power,otherPower); //send ...
              information through the serial port
110       //  while(1);
111
112     }
113
114
115    printf("%f,%f,%f\n\r", voltage_grid,current_grid,power); //send information through the serial port
116
117 //    printf("%ld,%ld\n\r", voltage_grid,current_grid);
118 }
119
120
121 static void sampling_timer_interrupt(void* arg) //timer interruption, main code runs here
122 {
123     int ADC1_v, ADC2_v;
124
125     /*
126     uint64_t time, time2 = 0;
127     time = esp_timer_get_time(); //time readings can be used to debug the ammount of time it takes ...
            to execute the code
128     printf("%lld\n\r",time);
129     */
130
131     //read the voltage in the 2 channels
132     ADC2_v = make_ADC_read(adc2_handle,adc2_cali_handle,ADC2_CHAN, N_MULTISAMPLING);
133     ADC1_v = make_ADC_read(adc1_handle,adc1_cali_handle,ADC1_CHAN, N_MULTISAMPLING);
134
135     power_calculation(ADC2_v,ADC1_v);
136
137     /*
138    //Debugging time?? uncomment these lines and line 97 to check the time taken to execute the code
139     time2 = esp_timer_get_time();
140
141     printf("%lld\n\r", time2-time);
142     */
143
144 }
145
146
147 //Function to give back the voltage reading (USER CALL TO READ CHANNEL)
148 int make_ADC_read(adc_oneshot_unit_handle_t handle_ADC, adc_cali_handle_t handle_CAL,adc_channel_t ...
        chan, int N_SAMPLES)
149 {
```

```
    int register_value = 0, voltage_read = 0, raw;

    //for loop to use multisampling (loop doesn't execute when N_SAMPLES = 1)
    for(int i = 0; i < N_SAMPLES; i++)
    {
        adc_oneshot_read(handle_ADC, chan, &raw);
        register_value = register_value + raw;

    }


    //get the average of multiple readings (Up to this point we have a register value)
    register_value = register_value / N_MULTISAMPLING;

    //Transform register into voltage
    adc_cali_raw_to_voltage(handle_CAL, register_value, &voltage_read);

    return voltage_read;
}



//.....................INITIALIZATION FUNCTIONS................................//
/*
 * The functions below are mostly taken from the datasheet, they can be changed mostly from the
 * definitions at the beginning of the code
 */


//Function to give back the voltage after the ADC reading (USED IN FUNCTION make_ADC_read)
static bool adc_calibration_init(adc_unit_t unit, adc_atten_t atten, adc_cali_handle_t *out_handle)
{
    adc_cali_handle_t handle = NULL;
    esp_err_t ret = ESP_FAIL;
    bool calibrated = false;

    if (!calibrated) {

        adc_cali_line_fitting_config_t cali_config = {
            .unit_id = unit,
            .atten = atten,
            .bitwidth = ADC_BITWIDTH_DEFAULT,
        };
        ret = adc_cali_create_scheme_line_fitting(&cali_config, &handle);
        if (ret == ESP_OK) {
            calibrated = true;
        }
    }

    *out_handle = handle;

    return calibrated;
}



//Configuration of the ADC, both used in ONE_SHOT mode
void ADCs_INIT(){

    //ADC1
    adc_oneshot_unit_init_cfg_t init_config1 = {
        .unit_id = ADC_UNIT_1,

    };
    ESP_ERROR_CHECK(adc_oneshot_new_unit(&init_config1, &adc1_handle));

    //ADC2
    adc_oneshot_unit_init_cfg_t init_config2 = {
        .unit_id = ADC_UNIT_2,
        .ulp_mode = ADC_ULP_MODE_DISABLE,
    };
    ESP_ERROR_CHECK(adc_oneshot_new_unit(&init_config2, &adc2_handle));


    //ADCs Calibration Init
    adc_calibration_init(ADC_UNIT_1, ADC1_ATTEN, &adc1_cali_handle);
    adc_calibration_init(ADC_UNIT_2, ADC2_ATTEN, &adc2_cali_handle);

     //ADCs load Config
    adc_oneshot_chan_cfg_t config = {
        .bitwidth = ADC_BITWIDTH_DEFAULT,
        .atten = ADC1_ATTEN,
    };
```

```
233
234
235     ESP_ERROR_CHECK(adc_oneshot_config_channel(adc1_handle, ADC1_CHAN, &config));
236     ESP_ERROR_CHECK(adc_oneshot_config_channel(adc2_handle, ADC2_CHAN, &config));
237
238     if(PRINT_STATE)
239         printf("ADCs initialized succesfully\n\r");
240 }
241
242 //Timer initialization, esily configurable time from the defines at the beginning of the code
243 void timer_INIT()
244 {
245     //Timer to handle sampling
246      const esp_timer_create_args_t sampling_timer_args = {
247             .callback = &sampling_timer_interrupt,
248             .name = "sampling"
249     };
250
251     esp_timer_handle_t sampling_timer;
252     ESP_ERROR_CHECK(esp_timer_create(&sampling_timer_args, &sampling_timer));
253
254     //Start running the timer
255     ESP_ERROR_CHECK(esp_timer_start_periodic(sampling_timer, TIME_US_TIMER));
256
257     if(PRINT_STATE)
258         printf("Timer initialized succesfully\n\r");
259 }
260
261 //UART0 configuration, to change the baudrate do it from the defines at the beginning of the code
262 void UART_init() {
263      if (uart_driver_install(UART_NUM_0, 2*1024, 0, 0, NULL, 0) != ESP_OK) {
264       //  ESP_LOGE(TAG, "Driver installation failed");
265         vTaskDelete(NULL);
266     }
267
268     uart_config_t uart_config = {
269         .baud_rate = BAUDRATE,
270         .data_bits = UART_DATA_8_BITS,
271         .parity    = UART_PARITY_DISABLE,
272         .stop_bits = UART_STOP_BITS_1,
273         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
274         .source_clk = UART_SCLK_DEFAULT,
275     };
276
277     uart_param_config(UART_NUM_0, &uart_config);
278
279     if(PRINT_STATE)
280         printf("UART0 initialized succesfully\n\r");
281 }
282
283 //call the initialization functions
284 void system_INIT()
285 {
286     //WatchDog de-initialization?
287     if(!WATCHDOG_STATE)
288         esp_task_wdt_deinit();
289     ADCs_INIT();
290     timer_INIT();
291     UART_init();
292
293     if(PRINT_STATE)
294         printf("System initialized succesfully\n\r");
295 }
```