# POWER FREQUENCY WEBSITE

Supervised by: Prof. Dr. Waffenschmidt[1]

Dervis Mujagic[1]

[1]Cologne University of Applied Sciences
F07, Institut für Elektrische Energietechnik, Betzdorferstr. 2, 50679 Köln, Germany

The aim of the project was to measure the line frequency in the UCTE network with an own measuring device and to publish it on a website.

## 1. INTORDUCTION

The power frequency has a very high importance in AC grids. The deviation from their nominal amount, provides information about the produced and the consumed electric instantaneous power.
Thus, the frequency remains stable, the balance between production and consumption of electric power must always be given.
If the balance is not given, the frequency decreases or increases, depending on the eclectic instantaneous power in the grid.
When more power is produced than is being consumed, the frequency rises. When less is produced than needed, the frequency decreases.
If the frequency deviation in the network is too large, electrical equipment and generators can also be damaged.

Because the power frequency is not publicly available for free, it was decided to measure the power frequency and publish it on a website.
The frequency is measured in the European UCTE network, whose nominal frequency is 50 Hz.

The following illustration shows how the frequency measurement project is realized.

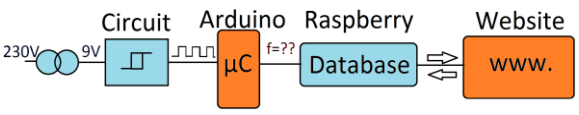

Figure 1: The project layout

- First of all the Voltage is transformed with a transformer. The Voltage is transformed form 230V down to 9V. This voltage is required for the circuit.
- Then a square wave voltage is generated by the circuit. The circuit is connected to the Arduino.
- Arduino is measuring the frequency and after that, the measured frequency is transmitted to the Raspberry Pi via USB.
- Raspberry Pi is the database, where the frequency is directly combined with a timestamp and saved in a table.
- Last part of this project is a website. It is located on a server at the Cologne University of Applied Sciences. The website presents the frequency graphically and as an additional feature the user can download frequency values from the past.

This paper will focus on the Arduino and give a brief information about the used charts on the website. More information about the other parts of the project can be found in the papers [1], [2] and [3].

## 2. FREQUENCY MEASUREMENT

As mentioned above the power frequency is measured with the circuit and the Arduino. The output of the circuit is connected to a digital pin of the Arduino. Arduino captures the square wave signal and calculates the frequency. The whole process is explained in the following chapters.

### 2.1 Arduino Leonardo

The used microcontroller-board type is the Arduino Leonardo. Arduino Leonardo is a modern microcontroller-board that is based on Atmels Conrtoller ATmega32u4. It has 20 digital I/O Pins, a 16-MHz-Quarz, a Micro-USB-port, an ISP-Header

and several additional features. Furthermore Arduino has an own Integrated Developing Environment (IDE). Arduinos IDE is very clear and simple to use. The programming language is C++ and a big advantage is that the distribution of the Arduino is open source. The Arduino community is up to date, so it is very easy to get in contact with other users and get support.

## 2.2 Principle of the frequency measurement

The conception of the frequency measurement is kept simple. The following figure and the formula explain the coherence mathematically.

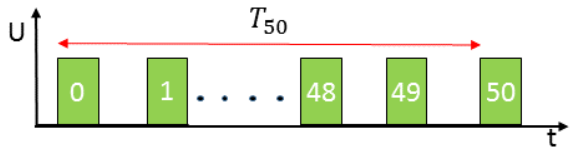


Figure 2: Square wave signal

Equation 1: frequency $$f = \frac{1}{T_1} = \frac{50}{T_{50}}$$

In a frequency measurement always 50 periods are counted, but to measure one period its required to measure the time between two rising edges. That's because the start of the following period is the end of previous period. So in this case 50 periods correspond to 51 rising edges. Therefore in the picture the rising edges are numbered from zero to fifty. Additionally, the time is measured from the first to the fiftieth period.
To get the periodic time of one period it is necessary to divide the time by 50. Finally the periodic time is inversed, to get the frequency.[4]

## 2.3 The Algorithm

The algorithm for frequency measurement is independent of the programming language and is also illustrated with the flow chart. In the flow chart it can be seen that, immediately after the start, the Process so called "Setup for measurement" will be enforced. At this point some variables are declared, defined and initialized. In the flow chart, only the variable "periods" in which the number of periods are stored, is shown.
Periods is initialized in setup for measurement with zero. The remaining variables are not listed here, because they are not necessary for understanding.
A loop follows directly after the Setup for measurement. Normally, it would be an endless loop since the measurement is to be carried out continuously.
In the endless loop, numbers in different colors are attached to the branches. These are used in the flow chart for better orientation and a simpler understanding of the algorithm.
In the following chapters the combination of the colored numbers will be used to explain the algorithm.

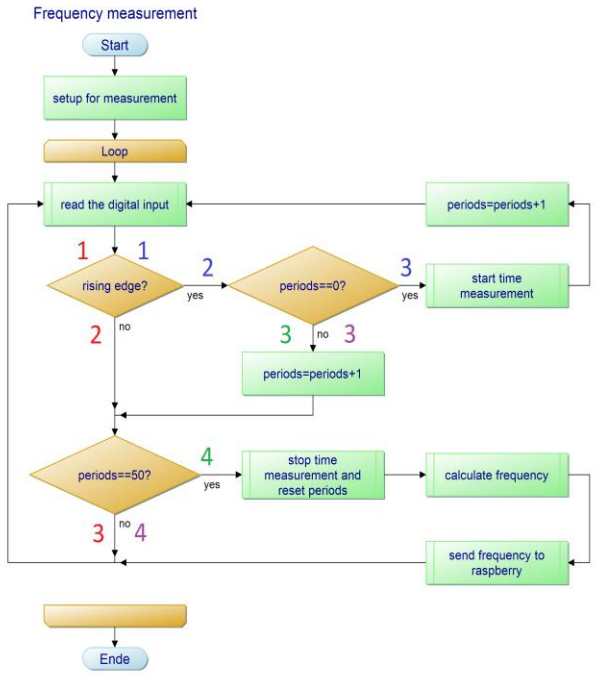


Figure 3: Flow chart

### 2.3.1 Combination: 1, 2, 3

At the beginning of the loop in "read the digital input" the digital pin of the Arduino is read out. This is followed by a branch. The branch (1) "rising edge?" checks whether a rising edge at the input is received. If a rising edge is detected, so the query results in (2)"yes", another branch "periods==0?" checks if a measurement is currently running. Because after each measurement, the variable "periods" is reset, the value of the variable can be used to detect a running measurement. At the start of the algorithm and if the last measurement is completed successfully this branch will always result in (3) "yes". If it results in (3) "yes" the time measurement starts and the variable "periods" will be incremented. After that the algorithm returns to the beginning of the loop.

### 2.3.2 Combination: 1, 2, 3

Now the Arduino is again reading the digital input and checks in (1) "rising edge?", if a rising edge is detected. If it results in (2) "no" another branch checks in "periods==50?" whether the variable "periods" has the value 50. If the query results in (3) "no" the algorithm again returns to the top of the loop.

2.3.3 Combination: 1, 2, 3, 4

In this Combination the algorithm starts again with the “reading the digital input”. If in (1) “rising edge?” a rising edge is detected the query results in (2) “yes”. Now the branch “periods==0?” checks whether a measurement is currently running. Only if a measurement is running the query results in (3) “no” and the variable “periods” is incremented because at the input of the Arduino a rising edge has been received. The following branch “periods==50?” checks again if the measurement has to stop because 50 periods have been counted. If this results in (4) “no” the loop starts again at the beginning.

2.3.4 Combination: 1, 2, 3, 4

This combination shows what happens if one measurement is completed successfully. The first three steps are the same like the combination before. The only difference is that in this case the variable “periods” is equal 50, thus the branch “periods==50?” results in (4) “yes”. Now the time measurement stops and the variable “periods” is set back to zero. The power frequency is calculated with the simple formula defined in 2.2.
Afterwards the frequency is send to the Raspberry Pi via USB. Finally the program returns to the beginning of the loop and the whole process starts again.

2.3.5 The algorithm in C++

As mentioned in Chapter 2.1 the programming language for the Arduino program is C++. In this chapter the main parts of the code are described.
The setup will not be discussed in detail because in the setup only all variables are declared and initialized with zero or LOW. In the following illustration the Loop of the algorithm can be seen.

```
void loop()
{
  newState=digitalRead(inputPin);
  if(oldState==LOW&&newState==HIGH)
  {
    if(risingEdge==0)
      oldTime=micros();
        risingEdge++;
  }
  if(risingEdge==51)
  {
    newTime=micros();
    freguency=50.000E6/(newTime-oldTime);
    Serial.println(frequency,3);
    risingEdge=0;
  }
  oldState=newState;
}
```

Figure 4: Arduino loop sourcecode

At first the rising edge detection had to be realized.
When a rising edge is at the digital input of the Arduino the Signal changes always from low (oldState) to high (newState).
This fact is used and implemented in C++. With the command “newState=digitalRead(inputPin)” the Arduino reads the digital input and saves the result in the variable “newState”. Now the if statement “if(oldState==LOW&&newState==High)” checks if the Signal changes from low to high. Then another if statement checks whether the “risingEdge” are equal to zero. The variable rising edges is similar to the variable “periods” in the flowchart. If it results in yes the time measurement starts with “newTime=micros()”. This statement saves the actual timestamp in the variable “newTime” and after that the “risingEdges” are incremented.
Now another if statement checks in ”risingEdge==51” whether 51 rising edges have been detected. If this statement is true, again a timestamp is created by “newTime=micros()”. After that the frequency is calculated with “frequency= 50.000E6/(newTime-oldTime). In the calculation can be seen that the “newTime” is subtracted by the “oldTime”. This is necessary because the function “micros()” provides a timestamp in microseconds since program start. As can be seen the number of periods is written as “50.000E6”. Again the reason for this is the fact that the time is measured in microseconds and to get the frequency in Hz it’s necessary to multiply the number of periods with E6, which means a multiplication with one million.
After that the “frequency” is sent via the function “Serial.println(frequency,3)” to the Raspberry via USB and the variable “risingEdge” is set back to zero.
One last very important statement that has to be mentioned, which is a part of the rising edge detection is the “oldState=newState”. It’s required to save the value of the “newState” in the variable “oldState”, because after that the Leonardo reads again the digital input and the variable “newState” gets a new value. Finally the whole process starts again.

2.4 Measurement accuracy

To determine the measurement accuracy, the first idea was to measure a frequent normal and to determine the deviation. Because it was impossible to find the frequent normal, so the measured values are compared with the values of other two websites that measure the power frequency. The comparison results in a mean measurement accuracy of 2 mHz.

3. POWER FREQUENCY WEBSITE CHARTS

The power frequency website has to visualize the measured frequency and give users the opportunity to

download values from the past. To get the measured values, the website communicates with the Raspberry Pi. The communication is realized by using the programming languages PHP and SQL.

PHP is a widely-used open source general-purpose server-side scripting language that is especially suited for web development and can be embedded into HTML. All of the PHP-scripts are located on the server where the website is located. That's because the PHP doesn't get executed on client side, but on the server the client requested the page from. The results are then handed over to the client, and displayed in the browser of the client. [5]

SQL is a standard language for accessing and manipulating databases. The SQL code is embedded in to the PHP-scripts. SQL is necessary because the database which is installed on the Raspberry is a MySQL database. [6]
The following chapter will not explain all types of charts. Because all programed charts have the same structure, only the dynamic chart will be explained. In comparison with the other charts, the dynamic chart has the highest complexity, because of the additional feature of secondly synchronization.

4.1 Dynamic Chart

The charts are created by using Highcharts. Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to a website or web application.
Highcharts library provides everything that is required for a dynamic chart. The library is copied on the webserver and included in a HTML file. All of the functions can now be used to generate a dynamic chart. [7]
The special feature of the dynamic chart is that the current frequency is retrieved form the Raspberry every second. To realize this data transmission, also a web technology named asynchronous JavaScript and XML (AJAX) had to be used.

With AJAX it's possible to update the chart without reloading the whole page. Thereby, the amount of data to be transferred is kept as small as possible. The following figure shows the defined function "requestData()" which uses AJAX to update the chart.[8]

```
function requestData() {
    $.ajax({
        url: "read-last-value.php",
        success: function(point) {
            var series = chart.series[0],
                shift = series.data.length > 60;

            // add the point
            chart.series[0].addPoint(eval(point), true, shift);

            // call it again after 1000ms
            setTimeout(requestData, 1000);
        },
        cache: false
    });
}
```

Figure 5: AJAX function

The important AJAX attributes in the function are:

- url
- success

The first one "url" gets the path of a script that has to be executed. In this case it's the "read-last-value.php".
Read-last-value.php reads the current value of the frequency from the Raspberry and converts it into the JavaScript format. If this is successful (attribute "success") a function passes the value to the chart and the function "setTimeout(requestData,1000)" will call the same function "requestData" in 1000 milliseconds. This is done endlessly because the function calls itself.

5. Conclusion

In summary the project is completed and the website offers many feature for free. All interfaces were successfully connected to each other. One big advantage is that the Arduino is directly connected to the database, so it even saves the data if the connection to the internet interrupts. In the near future the measuring device can be used for other purposes. For example it could be modified and used to control an inverter in dependence of the frequency. In addition the user can check the frequency if a Blackout happens in the UCTE like in Italy 2003.

7. QUELLEN


[1] Ibrahim Nassar: Power frequency website, University of Applied Sciences Cologne, 2015
[2] Julius Brügelmann: Power frequency website, University of Applied Sciences Cologne, 2015
[3] Cihan Mansuroglu: Power frequency website, University of Applied Sciences Cologne, 2015
[4] http://groups.uni-paderborn.de/physik/studieninfos/praktika/versuche_anleitungen/messmethoden/mtb_kapc.pdf accessed: 15.05.2014
[5] https://php.net/manual/de/index.php accessed: 20.05.2014
[6] http://downloads.mysql.com/docs/refman-5.1-de.a4.pdf accessed: 01.07.2014
[7] http://www.highcharts.com/ accessed: 01.01.2015
[8] http://www.w3schools.com/ajax/ accessed: 20.12.2014