Power frequency website

Julius Brügelmann
j.bruegelmann@live.de
Matr.: 11098432

Cologne University of Applied Sciences
Prof. Dr. Eberhard Waffenschmidt

Abstract - The task of the project is to develop an instrument able to measure the frequency in the UCTE power grid. The measurement results are then published on a website to make the data freely available to the public.

Frequency is an important variable because it gives information about the stability of an AC voltage grid. Power plants must always deliver the required amount of power to the network. If more power is used than the power plants can supply, the frequency decreases. At excessively high feed or low power, the frequency increases. To control the balance between consumed and produced power, three main control stages are used. They are the primary, secondary, and tertiary control stages. Without these mechanisms, the network would collapse. [1][2]
The Union for the Coordination of Transmission of Electricity (UCTE) is an association of European transmission system operators for electricity. This group creates the European interconnected power system with a long term target frequency of 50 Hz. [3]

Figure 1 shows the project outline.

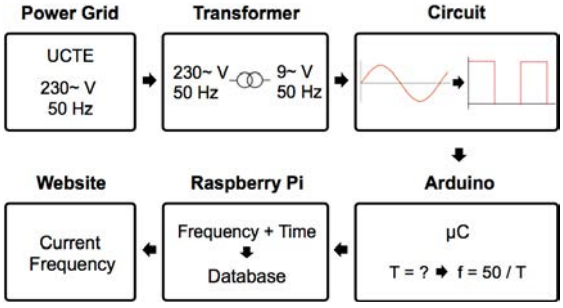

Figure 1

First, the voltage is transformed from 230V AC to 9V AC. Then a circuit converts the analog signal to a digital. A microcontroller, in this case an Arduino, measures the time T of 50 periods and calculates the frequency using f=50/T. The calculated frequency is sent via USB to a single board Raspberry Pi computer. The Raspberry Pi combines the frequency with a timestamp and stores it in an installed MySQL database.

Finally, the website is given access to the database, making it possible to see and download current and past frequency information.
In this paper only a part of the project is treated. It deals with the installation and configuration of the mini computer Raspberry Pi and the script that runs on it. The script allows communication between the Arduino and the MySQL database. This paper also discusses how the complete hardware is built and housed. To learn more about the remaining parts of this project, please refer to the additional papers, which go into detailed explanations. [4]

I. Raspberry PI

A Raspberry Pi is a single Board Computer (SBC) with the size of a credit card. The Raspberry Pi was developed in the United Kingdom by the Raspberry Pi Foundation. The product first appeared on the market in early 2012.
This SBC can be used for multiple purposes, and different operating systems, such as Linux, Android and RISC OS, can be installed. For the various operating systems, there exist various accessories and software packages. Since 2012, the Raspberry Pi Foundation has developed and sold several models. [5]
In this project the Raspberry Pi B+ is used.
The following table lists the specifications of the Raspberry Pi B+. [6]

Raspberry Pi B+

Target price	26 Euro
Size (mm)	93 x 63,5 x 20
Weight	45 g
System on Chip	Broadcom BCM2835

CPU	700 Mhz single Core
Memory (SDRAM)	512 MB
Video / Audio outputs	HDMI, 3.5 mm phone jack
USB 2.0 ports	4
On-Board storage	MicroSD slot
On-board network	10/100 MBit-Ethernet
GPIO-Pins	26 (3,3 V)
Power ratings	600 mA, 3W
Power source	5 V, 2 A

Table 1 Specifications

There are two ways to connect the Arduino to the Raspberry Pi. In the first approach, GPIO (General-purpose input/output) interface pins are used. However to use the GPIO pins, an additional level converter is needed because the Arduino and the Raspberry Pi work with different voltages. Therefore in this project the Arduino is connected to the Raspberry Pi via USB. [7]
The Raspberry Pi is equipped with a 16GB MicroSD card, providing enough space for the required software and the measured data. After installation of the complete software, there is still 10Gb left for the database. This space would be sufficient to store frequency values for 8 years.

The first step is to install the operating system on the MicroSD card of the Raspberry Pi. The operating system used is Raspbian, which is a custom version of Debian GNU / Linux for mini computers. After installation, the Raspberry Pi can be operated via a terminal or via a graphical user interface. [8]
Since the website must be able to access the database, the computer is set up as a web server.
For this, the web server lighttpd is installed. This is a stripped down version in contrast to other available web servers and therefore uses less computing power. However, the stripped-down version is perfectly adequate for this project. After configuring the webserver and connecting the Raspberry Pi to a network cable with a fixed IP address, it is possible to access the Raspberry Pi over the World Wide Web. [9]
Next, the MySQL database is installed to enable storage of measurement data. At first, the idea was to use the SQLite database, which has fewer functions but is resource-saving. However the database of SQLite can only be accessed over an internal network and not over the Internet. After installing and configuring MySQL database, phpMyAdmin is installed, which is a web-based management software for the MySQL database. This allows the database to be managed via a graphical interface. [10]

Lastly, the watchdog timer is enabled and configured. A watchdog timer is a special kind of timer that is used to detect if and when the software is hung up on some task. This is just a precaution. The system must operate 24 hours each day and should not crash. But if a crash happens, the system will reboot and only a few measured values are lost. The watchdog only makes sense if it runs independently of the actual system, therefore the Raspberry Pi comes with a hardware-based watchdog timer. [11]

II. Python script

After the Web server and the database are successfully installed and configured, the next step is to write a script read and store the measured values. The script is written in the Python programming language to be run permanently on the Raspberry Pi. For this reason the development environment of Python must be installed on the Raspberry Pi. The reason to use the Python programming language is because it is the official language of the Raspberry Pi (Pi stands for Python interpreter). This means that there are many tutorials to learn the language.

In Figure 2, the source code is shown.

```
01  import  MySQLdb
02  import  time
03  import  serial
04
05  connection  = MySQLdb.connect
06                      (host="localhost",user="xxx",
07                       passwd="xxx",db="Measurements")
08  cursor = connection.cursor()
09  cursor.execute("CREATE TABLE IF NOT EXISTS
10                      Measurements
11                      (ID INT NOT NULL AUTO_INCREMENT,
12                      PRIMARY KEY(ID),
13                      Date_Time DATETIME,
14                      Milliseconds INT,
15                      Frequency FLOAT);")
16
17  def hertz():
18        ser = serial.Serial('/dev/ttyACM0', 9600)
19        hertz = ser.readline().strip()
20        return hertz
21
22
23  def milliseconds():
24        now = time.time()
25        milliseconds = int((now - int(now)) * 1000)
26        return milliseconds
27
28
29  def insert_into_database():
30        cursor.execute("INSERT INTO Measurements
31                      (ID, Date_Time, Milliseconds, Frequency)
32                      VALUES (NULL,Now(),'%s','%s')" %
33                      (milliseconds(),hertz()))
34        connection.commit()
35
36
37  def main():
38        while True:
39              insert_into_database()
40
41
42  if __name__ == '__main__':
43        main()
```

Figure 2

First, the libraries are involved with the import-statement (Figure 3).

```
01  import  MySQLdb
02  import  time
03  import  serial
```

Figure 3

The library MySQLdb must be integrated so that MySQL commands can be executed in Python. With the library serial, it is possible to access the serial ports such as the USB on Python. So this module encapsulates the access for the serial port. To work with date and time within Python, the library time is required. The libraries MySQLdb and serial must be installed separately and are non-standard libraries. [14] [15] [16]
Then the connection is made to the database and the table Measurements is created (Figure 4).

```
04  connection  = MySQLdb.connect
05                      (host="localhost",user="xxx",
06                       passwd="xxx",db="Measurements")
07  cursor = connection.cursor()
08  cursor.execute("CREATE TABLE IF NOT EXISTS
09                      Measurements
10                      (ID INT NOT NULL AUTO_INCREMENT,
11                      PRIMARY KEY(ID),
12                      Date_Time DATETIME,
13                      Milliseconds INT,
14                      Frequency FLOAT);")
```

Figure 4

To establish the connection to the database, the method connect() is called from the MySQLdb module. In addition, the host, the user, the password and the name of the database must be entered. In the next row on line 07, a cursor is created, which is to resort to the MySQL database. Then the table Measurements with the columns ID, Date_Time, Milliseconds and Frequency is created. For this, the command CREATE TABLE IF NOT EXISTS is used. With the function execute() the SQL command is sent to the database. [13]
Next, the custom created functions are explained, starting with the function hertz() (Figure 5).

```
15  def hertz():
16      ser = serial.Serial('/dev/ttyACM0', 9600)
17      hertz = ser.readline().strip()
18      return hertz
```

Figure 5

This function allows reading the USB port, to which the Arduino is connected. In line 16, the port is defined. The name of the port is ttyACM0 and the transmission speed is 9600 Bd. With the method readline(), the specified port is read. [15]
The second function milliseconds() creates the needed milliseconds for the time (Figure 6).

```
19  def milliseconds():
20      now = time.time()
21      milliseconds = int((now - int(now)) * 1000)
22      return milliseconds
```

Figure 6

Line 20 shows the return of the time in seconds since the epoch as a floating-point number. The epoch is the point where the time starts. For Unix, the epoch is 1970. However, only the milliseconds are required after the decimal point, so the time since the epoch has to be converted. For this, first the time is converted in an integer and subtracted from the time as a floating-point number. As a result, only the value after the decimal point is present and can be multiplied by 1000. [16]
The function insert_into_database() writes the consecutive ID, the date, the time, the milliseconds and the frequency in the table Measurements (Figure 7).

```
23  def insert_into_database():
24      cursor.execute("INSERT INTO Measurements
25                      (ID, Date_Time, Milliseconds, Frequency)
26                      VALUES (NULL,Now(),'%s','%s')" %
27                      (milliseconds(),hertz()))
28      connection.commit()
```

Figure 7

For that the command INSERT INTO is used, which is sent with the function execute() to the database. The two existing functions primary key() and now() are utilised for creating the ID, the time and date. The developed functions hertz() and milliseconds() are used to determine the measured values and the milliseconds. With the method commit() the current transaction is completed. [13]
Every second a measured value should be stored in the database, so an endless loop with “while True" is realized in the function main() (Figure 8).

```
29  def main():
30      while True:
31          insert_into_database()
32
33
34  if __name__ == '__main__':
35      main()
```

Figure 8

The code in line 34 specifies which function is executed first when the program starts. In this case, the function main() starts. [12]

Figure 9 below shows the result of the script in phpMyAdmin.

ID	Date_Time	Milliseconds	Frequency
31	2015-03-15 11:06:45	471	49.984
32	2015-03-15 11:06:46	471	49.985
33	2015-03-15 11:06:47	471	49.987
34	2015-03-15 11:06:48	479	49.986

Figure 9

Finally, the finished script is added to the autostart of the Raspberry Pi. When the system starts or restarts, the script is executed automatically.

III. Housing of the measurement instrument

The measurement instrument consists of the circuit, the Arduino, and the Raspberryi Pi. The three parts are mounted inside of a plastic housing with dimensions 150 mm x 100 mm x 60 mm.
Figure 10 shows where and how the parts are attached. The Arduino is fixed to the cover, the circuit and the Raspberry Pi are fixed to the bottom of the housing. The componets do not require an extra fan because they are sufficiently cooled by the air flow through the housing holes.
Figure 11 presents the connections in the box. There is a powerswitch, an ethernet interface, four USB ports and two sockets. The first connector provides the Raspberry Pi with 5V DC and the second supplies the circuit with 9V AC. Also the two sockets are different in size, so that the plugs can not be confused.
The opening above the USB ports is used by the USB cable from the Raspberry Pi to connect to the Arudino. In addition, the box has one HDMI port and one MircroSD card slot.

Figure 10

Figure 11

IV. Summary

The task of the project is to measure the line frequency of the UCTE power grid and to present the measured values on a website. Two parts of the project are discussed in this paper, the creation of the database with a Python script on the Raspberry Pi and the preparation of the housing for the measuring instrument. During the installation and configuration of the Raspberry Pi, many well-thought-out decisions have to be made, for example choosing which programs has to be used. There are many programs with similar functions, so the program most compatible with the project is selected. The programming language Python has to be learned for this part of the project. Finally, it turns out that Python syntax is very clean, with an emphasis on readability and usage of standard English keywords.
The result of the previously presented parts of project is that the measured values can be saved in the database and be used prospectively for different analyses.

V. References

[1] Amprion GmbH: Netzfrequenz, URL: http://www.amprion.net/netzfrequenz (14.3.2015)
[2] Leuschner, Udo: Die Netzfrequenz darf nur minimal schwanken, URL: http://www.udo-leuschner.de/basiswissen/SB124-04.htm (14.3.2015)
[3] Entsoe: Union for the Coordination of Transmission of Electricity (UCTE), URL: https://www.entsoe.eu/news-events/former-associations/ucte/Pages/default.aspx (14.3.2015)
[4] Mujagic, D., Nassar, I., Mansuroglu, C: Paper, Power frequency website
[5] Raspberry Pi Foundation: What is a Raspberry Pi?, URL: http://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (14.3.2015)
[6] Hammerschmidt, Anton: Raspberry Pi Guide, URL: http://raspberrypiguide.de/ (14.3.2015)
[7] Simon, Ronny: Raspberry Pi und Arduino: Serielle Verbindungen, URL: http://blog.simtronyx.de/raspberry-pi-und-arduino-serielle-verbindungen/ (14.3.2015)
[8] Raspberry Pi Foundation: Installing operating system images using windows, URL: http://www.raspberrypi.org/documentation/installation/installing-images/windows.md (14.3.2015)
[9] Watkiss, Stewart: Running a lightweight webserver on the Raspberry Pi, URL: http://www.penguintutor.com/linux/light-webserver (14.3.2015)
[10] Kriwanek, Andreas: Raspberry Pi Webserver mit Apache2, PHP5 und MySQL, URL: http://www.kriwanek.de/raspberry-pi/raspberry-webserver/287-raspberry-pi-als-webserver-mit-php-und-mysql.html (14.3.2015)
[11] Dietrich, Manuel: Raspberry Pi – Stabiler 24/7 Dauerbetrieb, URL: http://www.datenreise.de/raspberry-pi-stabiler-24-7-dauerbetrieb/ (14.3.2015)
[12] Klein, Bernd: Python Tutorial, URL: http://www.python-kurs.eu/kurs.php (14.3.2015)
[13] Behrens, Fionn: MySQL mit Python abfragen, URL: http://www.linux-magazin.de/Ausgaben/2002/06/Kaninchen-oder-Schlange (14.3.2015)
[14] Hecht, Georg: Sourcecodes - MySQL-Zugriff mit Python, URL: http://www.online-tutorials.net/mysql/mysql-zugriff-mit-python/sourcecodes-t-130-316.html (14.3.2015)
[15] Liang Oscar: Connect Raspberry Pi and Arduino with serial USB Cable, URL: http://blog.oscarliang.net/connect-raspberry-pi-and-arduino-usb-cable/ (14.3.2015)
[16] Python Software Foundation: Time - Time access and conversions, URL: https://docs.python.org/2/library/time.html (14.3.2015)