Sensors and Materials, Vol. 29, No. 6 (2017) 741–756
MYU Tokyo

S & M 1364

A Power Frequency Sensing Device Using an Arduino Device and Zero-Crossing Algorithm and Its Implementation on Android App

Hui-Yu Shen,[1,2] Yeong-Chin Chen,[1*] and Chao-Hsing Hsu[2]

[1]Department of Computer Science and Information Engineering, Asia University,
500 Liufeng Rd., Wufeng, Taichung 41354, Taiwan
[2]Department of Information and Network Communications, Chienkuo Technology University,
No. 1, Jieshou N. Rd., Changhua, Changhua 50094, Taiwan

(Received November 11, 2016; accepted December 26, 2016)

Keywords: frequency sensing device, Arduino, zero crossing, Android app

In this study, an Arduino controller is used to build a low-cost data acquisition (DAQ) sensing device with a sampling frequency of up to 50 kHz. The utility power signal captured through the Arduino controller is converted into digital data, which are then transmitted through a WiFi module to an Android mobile device so that mobile devices can display the signal waveform and power quality information. Using a discrete Fourier series (DFS) filtering algorithm, noise and high harmonics are filtered out, and then a zero-crossing algorithm is applied to accurately calculate the fundamental frequency of the power signal. Since the power signal is an AC signal with high voltage, we designed a voltage step-down circuit for signal preprocessing to adjust the signal level to the standard level accepted by the Arduino controller. Research confirms that we can design a portable power signal acquisition device to view signal waveforms using an Android app and calculate the frequency variation of power signals. Additionally, the power signal data can also be conveniently transmitted to a cloud server to record and analyze unusual power events.

1. Introduction

In recent years, the use of nonlinear power system loads has been increasing, so the harmonic pollution of power line quality has been growing. When an imbalance between power supply and demand occurs between power systems, it results in system frequency variation,[(1)] and these changes are likely to cause accidents or damage to the power line system. Owing to current technological advances, conveniences in life, and increases in the amount of livelihood equipment, power consumption has been increasing so that frequency changes are quite common. The degree of change varies depending on the load characteristics and design of the power supply system.[(2)] In power line system operation, when the frequency is lower than some nominal value, the system is under the conditions of overload. However, if the frequency is higher than the nominal value, it shows an oversupply to the power lines. The frequency represents whether the power line reaches an equilibrium state between supply and demand, so frequency is a very important indicator of the reliability, safety, and economy of a power line system. In this study, an Arduino sensing device is

*Corresponding author: e-mail: ycchenster@gmail.com
http://dx.doi.org/10.18494/SAM.2017.1497

ISSN 0914-4935

used to build a low-cost data capture card combined with mobile devices to build a portable power line frequency detection device. By observing changes in the frequency, we can immediately detect the status of the power line system operation, and interferences in the power line quality can be monitored at the same time. We also present a full analysis, including data resolution and sampling frequency, to describe the feasibility and simplicity of our Arduino sensing device.

2. Research Methods

In this study, first, a power signal sensing module, composed of a signal conditional circuit, namely, Arduino, and a data transmission device, was developed; the system architecture is shown schematically in Fig. 1. In this sensing module, the power signal is pretreated through voltage step-down and voltage level shift processing. After that, the signal data processed by the Arduino analog-to-digital converter (ADC) module[3] is digitalized, packaged, and then transmitted to a mobile device. The signal collected by the Arduino is then monitored at any time in the Android app. A cloud server works to complete the function monitoring power line quality.[4,5] A discrete Fourier series (DFS) is used to filter out high harmonics and noise to collect the fundamental sine wave signal. The data matrix of the fundamental sine frequency is stored in the data array of the mobile device, and then the zero-crossing algorithm is used to estimate the actual frequency of sinusoidal components. The DFS algorithm and zero-crossing algorithm can be reused in monitoring the high frequency and amplitude of the original signal.

3. Power Signal Sensing Device

The power signal sensing device is composed of three modules: a signal conditional circuit, an Arduino ADC, and a data transmission module. Initially, the amplitude of the power signal first uses voltage level shifting to connect the utility power signal to the Arduino analog input. Additionally, we also analyze the frequency issues and data resolution with the ADC function of the Arduino. After that, the relationship between the sampling frequency and the delay time is thoroughly discussed. Finally, environmental information including the temperature, humidity,

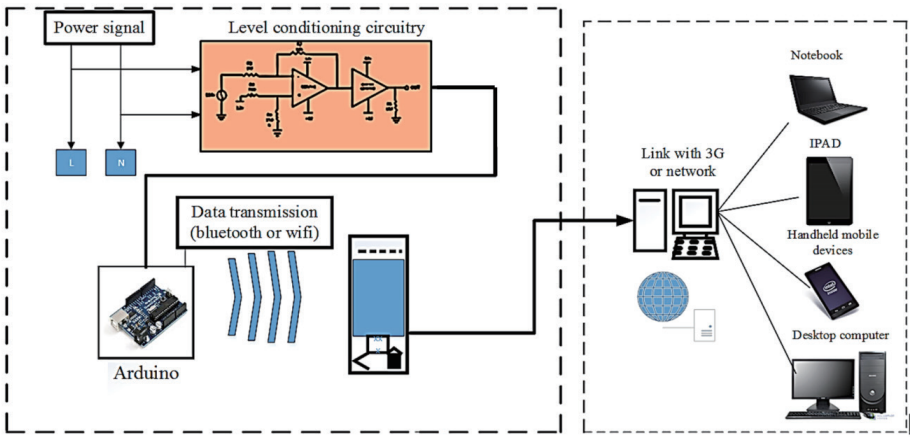


Fig. 1. (Color online) Diagram of mobile device for power signal monitoring.

geolocation, and site identification (ID) is collected and sent to a remote Internet server. The overall functional block diagram for the power signal sensing Arduino device is shown in Fig. 2.

3.1 Power signal level shifting circuit

Since the power signal input is a 60 Hz AC signal and 110 V root mean square (RMS), the utility power voltage peak is about 155.5 V. However, the signal level sent into the Arduino ADC module is only 0 to 5 V; the power signal level shifting circuit is shown in Fig. 3. The amplitude level of the power signal is adjusted using the following formula:

$$\begin{aligned} V_{out} &= V_{out1} + V_{out2} \\ &= V_{AC}\left(-\frac{R_2}{R_1}\right) + V_{DC} \times \frac{R_4}{R_3 + R_4}, \end{aligned} \tag{1}$$

where V_{out1} sets the $\frac{R_2}{R_1}$ ratio to $\frac{10\ \text{k}\Omega}{1\ \text{M}\Omega} = 0.01$, making the AC voltage decrease from ±155.5 to ±1.55 V, and V_{out2} is added with a 2 V DC signal to accept the conversion from an analog to a digital signal.

The voltage level is adjusted to 0.5–3.6 V to match the input signal of the Arduino ADC module. Figure 4 shows the multisim simulation results, where ch1 is the power signal input, and ch2 (triangle mark on the waveform) is the simulation signal after the voltage level adjustment. The simulation showed the desired consistency with the circuit design. In addition, the actual measurement of the output power signal in the GwINSTEK oscilloscope is shown in Fig. 5, where the measured signal is consistent with the simulation results.

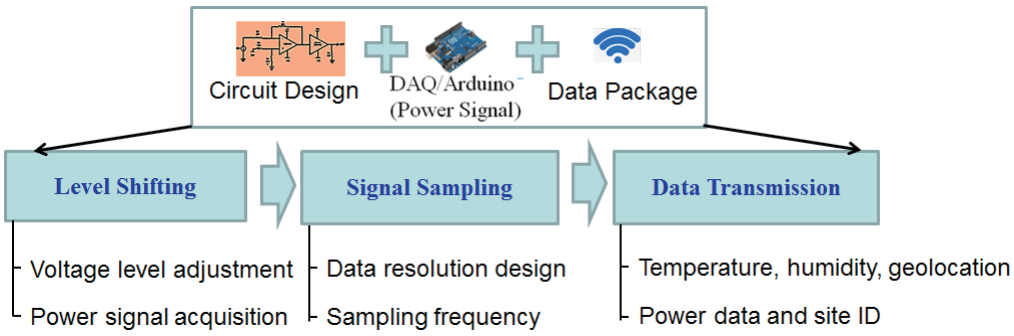


Fig. 2. (Color online) Power frequency sensing device.

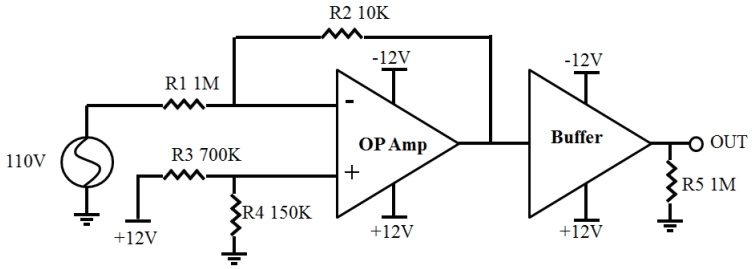


Fig. 3. Circuit design for power signal level shifting.

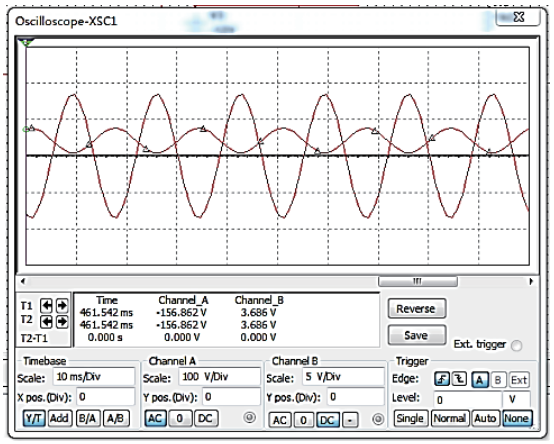


Fig. 4. (Color online) Simulation results of power signal level shifting circuit.

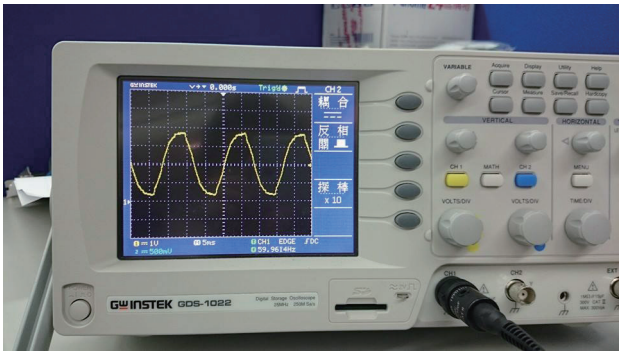


Fig. 5. (Color online) Measured results in GwINSTEK oscilloscope after power signal level shifting.

3.2 Power signal acquisition in the Arduino ADC module[6,7]

The Arduino data acquisition (DAQ) approach is to use analogRead (0) instructions to convert analog information into digital information (ADC module). The chip used in the Arduino is the ATMEG chip; its oscillation frequency is 16 MHz and its internal ADC module prescaler setting is shown in Table 1.

Generally, the default value of the ATMEG ADC[6] prescaler is 128, and the implementation of an ADC function requires 13 cycles. Therefore, the ADC processing speed is 16 MHz divided by (128/13) and the ADC sampling rate is 9600 Hz. If the prescaler is set to 16, the sampling rate in the Arduino ADC module can be theoretically increased to about 77 kHz (16M /16/13). If sampling accuracy is taken into account, the sampling frequency needs to maintain more than ten times the measured signal. Thus, we can use the Arduino ADC module to measure a periodical signal of 7.7 kHz (10 sampling points per period). Since the fundamental frequency of power signals is 60 Hz, and 25 times the harmonic wave is only 1.5 kHz, the Arduino ADC module is sufficient to meet the requirement of our study.

3.3 Data resolution design in Arduino

The Arduino ADC has 10-bit resolution, thus the captured dynamic signal amplitude can be resolved into 1024 (2^10) deciles. The internal reference voltage of the dynamic signals is 3.3 or 5 V, but the voltage also varies with the method of the Arduino power supply, as shown in Table 2.

For example, when the reference voltage is 5 V, the voltage upon receiving an analogRead(0) is converted to (1023 * analogRead(0) / 5). When the input voltage is 3 V, its value displayed by the analogRead(0) function is about 614. When the dynamic signal variable becomes small (e.g., 0.1–1 V), the reference voltage signal will become 1 V to improve the resolution. We can connect the AREF pin of the Arduino to 1 V, and software analogReference (EXTERNAL) is used to set the reference voltage to 1 V to improve resolution. To verify the actual sampling rate of the Arduino ADC, this

Table 1
Arduino ADC prescaler setting.

ADPS2	ADPS1	ADPS0	Division factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Table 2
Arduino reference voltage.

Set internal reference voltage	3.3 V pin	5 V pin
Power supply	3.3 V pin	5 V pin
USB power only	3.41 V	4.78 V
9 V battery	3.42 V	5.02 V
9 V power supply	3.4 V	5.02 V

study used a 1 kHz signal generator to generate a sine wave signal of 1 to 2 V in amplitude; the AREF pin reference voltage is set to 3.3 V; the prescaler is set to 16, and 200 sets of data are read. The data read by analogRead(0) is (1023 * analogRead(0)/3.3). The Arduino code for the read loop of the read data is shown as

```
for (i=0 ; i<size; i++){
  time[i]=Micros();
value[i]=analogRead(0);}
```

The instruction Micros() executes simultaneously to record the read time of each data. The partial data from 100 to 106 recorded in Table 3 are read and the program execution times are recorded. The captured ADC data is drawn by Excel as shown by the red line in Fig. 6.

It is worth noting that the sampling interval for the data obtained in Table 3 is 24 μs, and the sampling frequency is approximately 41.6 kHz. For the 200 data points shown by the red line in Fig. 6, there are 41 points for each cycle, so the actual sampling frequency by program is (1 kHz) × 41 = 41 kHz, which is quite consistent with 41.6 kHz.

3.4 Sampling frequency adjustment in the Arduino

To maintain the estimation accuracy for the following zero-crossing algorithm, we added a delay instruction code to let the Arduino ADC module have power signal data over a fixed duration and modulated the sampling rate of the Arduino ADC module.

The delay times for our experiment processes were set to 5, 10, 15, and 20 μs. The corresponding sampling points per period obtained by the zero-crossing algorithm were 39.449, 32.700, 28.124, and 24.374 points, respectively. Since we provide a frequency of 1 kHz for the

Table 3
Arduino ADC module data.

No. of points	Time (μs)	Amplitude (0–1023)
100	100	135
101	124	153
102	148	177
103	172	206
104	196	240
105	220	277

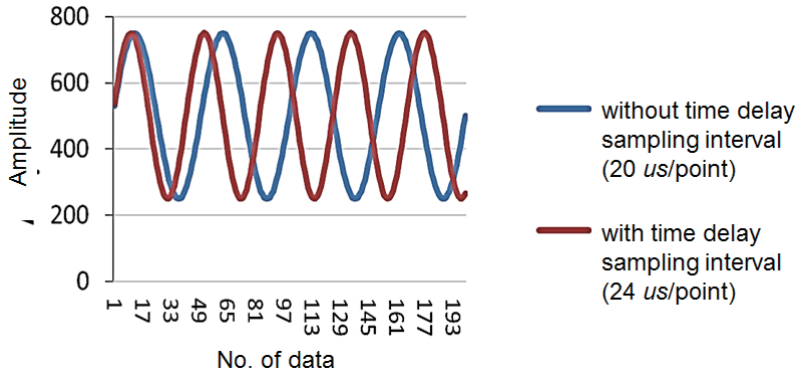


Fig. 6. (Color online) (1 kHz) sine waveform signals acquired by ADC.

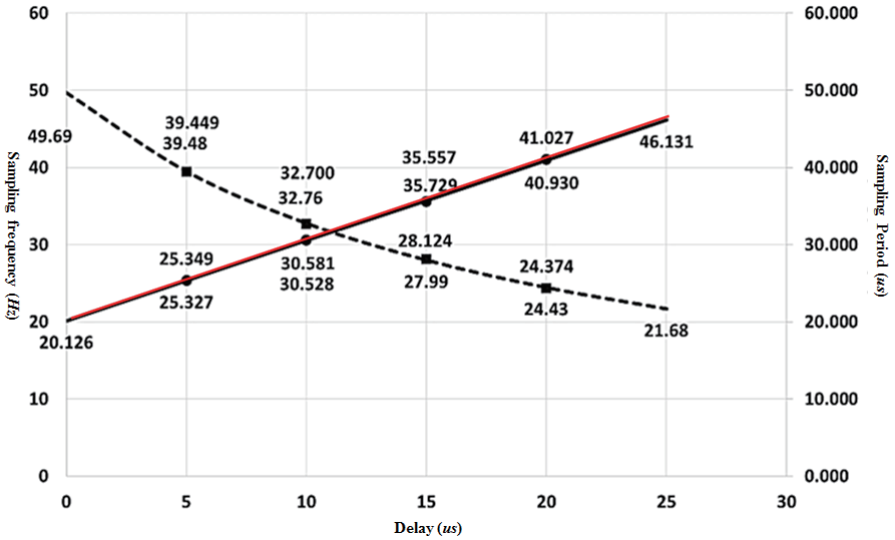
Arduino ADC module, the sampling frequencies are 39.449, 32.700, 28.124, and 24.374 kHz, respectively. If we take the reciprocal of the sampling frequency, the sampling period can be obtained and plotted as shown by the red lines in Fig. 7. To predict the delay time influence on the sampling frequency, the least-squares method was used[8] as shown in Fig. 8 to predict the relationship of program latency time (X) and sampling period (Y). The linear regression equation is

$$Y = c + aX, \tag{2}$$

where $c = 20.12601$ and $a = 1.040209$.

4. Power Signal Analysis

A high-accuracy and simple zero-crossing algorithm accompanied by the DFS has been provided to calculate the power signal quality effectively. The zero-crossing algorithm uses a very simple concept and provides fast computing performance. However, the zero-crossing algorithm function is apt to be limited by large-amplitude harmonics, non-integral harmonics, or even noise. Therefore, we first use a DFS approach to filter out the high-order harmonics and noise. Then, the zero-crossing algorithm is applied to accurately obtain the baseband amplitude and frequency.



Delay (μs)	0	5	10	15	20	25
● Sampling interval (μs)		25.349	30.581	35.557	41.027	
—Minimum regression line (μs)	20.126	25.327	30.528	35.729	40.930	46.131
■ Sampling frequency (kHz)		39.449	32.700	28.124	24.374	
---Reciprocal of the regression line (frequency)	49.69	39.48	32.76	27.99	24.43	21.68

Fig. 7. (Color online) Relationship between delay time (X) and sampling period (Y) in the Arduino ADC module.

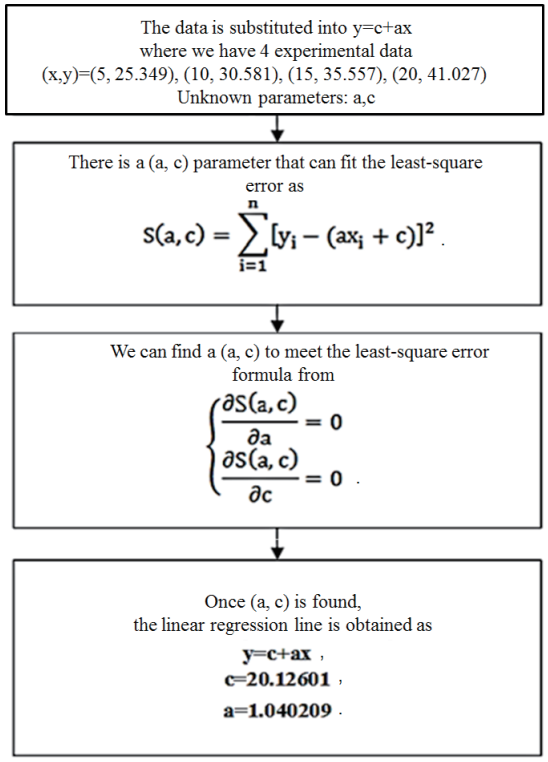


Fig. 8. Flow chart to predict the relationship between sampling frequency and delay time.

4.1 Discrete Fourier sequence algorithms for extracting fundamental waves[9]

The voltage signal to be measured can be expressed as

$$v(t) = C\cos(\omega t+\varphi) + R(t), \tag{3}$$

where C is the fundamental wave amplitude, $\omega = 2\pi f$ is the fundamental frequency, φ is the phase of the fundamental wave, and $R(t)$ is the high harmonics and signals of zero mean noise.[10–12]

If the exact value of the fundamental frequency is unknown, we assume that the angular frequency of the fundamental frequency is ω_a, V_a (DFS parameter) is the estimated amplitude of fundamental signal, m is the number of samples sampled by the fundamental frequency wave period, and v_n is the n-th sample value of signals. Since $f_a = \frac{1}{T_a}$, $T_a = \frac{2\pi}{\omega_a}$ is obtained, where $\psi = \frac{\omega_a T_a}{m} = \frac{2\pi}{m}$. With the fetched sample value v_n, the DFS can be expressed as

$$V(i) \cong \frac{2}{m}\left[\sum_{n=i}^{m+i-1} v_n \cos\left(\frac{\omega_a T_a}{m}n\right) - j\sum_{n=1}^{m+i-1} v_n \sin\left(\frac{\omega_a T_a}{m}n\right)\right] = A(i) + jB(i). \tag{4}$$

For the i-th data window, its sine and cosine components, $A(i)$ and $B(i)$, can be calculated as follows.

$$A(i) = \frac{2}{m}\sum_{n=i}^{m+i-1} v_n \cos(\psi n) \tag{5}$$

$$B(i) = \frac{2}{m}\sum_{n=i}^{m+i-1} v_n \sin(\psi n) \tag{6}$$

The term $C(i)_a = \sqrt{A(i)^2 + B(i)^2}$ is the estimated amplitude of the fundamental frequency of the i-th data window, $\phi_f = \frac{B}{A}$, and $C(i)$ is accurate only when $\omega_a = \omega$. The signal sampling frequency is $f_s = m/T_a = 1/T_s$, where T_s is the sampling period. If the data window is swept by the measured signals, Eqs. (5) and (6) can provide $A(t)$ and $B(t)$ with the corresponding values of a cycle where the sine and cosine components, A and B, are periodic functions of time. If the frequency of the fundamental harmonic signal is equal to the basic frequency of the Fourier series ($T_s \bullet m = T_a = T = \frac{1}{f}$), $A(t)$ and $B(t)$ in frequency are orthogonal to each other; $A(t)$ is the pure cosine wave, and $B(t)$ is a pure sine wave; when $T_a \neq T$, $A(t)$ and $B(t)$ are not pure sine and cosine waves, but their fundamental frequency is still f.

Figure 9 shows a 60 Hz fundamental frequency and its harmonic signal, and the contents are assigned as 100% of the 1st harmonic, 40% of the 3rd harmonic, and 30% of the 5th harmonic. In practice, the cosine and sine signals in Eqs. (5) and (6) can be represented by the vector matrix of length m as

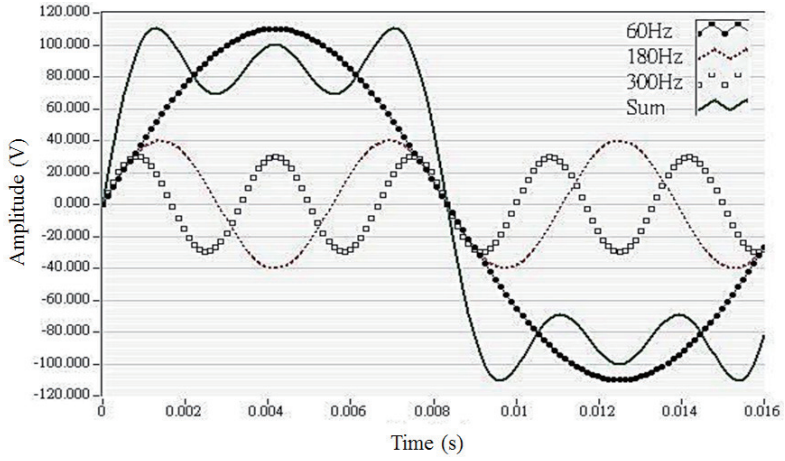


Fig. 9. Mixed signal of fundamental and 3rd and 5th harmonic signals.

$$\boldsymbol{COS} = \frac{2}{m}\left[\cos(\psi), \cos(2\psi), ..., \cos((m-1)\psi), 1\right]^{\mathrm{T}} \tag{7}$$

$$\boldsymbol{SIN} = \frac{2}{m}\left[\sin(\psi), \sin(2\psi), ..., \sin((m-1)\psi), 1\right]^{\mathrm{T}} \tag{8}$$

The vector of signal samples is

$$\boldsymbol{SAMs} = [v_1\ v_2\ v_3\ v_4\ v_5\ ...\ v_n]^{\mathrm{T}}. \tag{9}$$

With the ***COS*** and ***SIN*** auxiliary vector matrix, the computation of Eqs. (5) and (6) requires only multiplications and additions, no trigonometric calculations, and the calculation is very simple. After each sampling, the sampling points should be rearranged as

$$v_1 = v_2,\ v_2 = v_3,\ ...,\ v_n = v_{new}. \tag{10}$$

Thus, the sliding data window and the signal processing of the signal samples are considered scalar. For each data window, according to the relationship in Eqs. (5) and (6), one can calculate its corresponding cosine $A(t)$ and $B(t)$ sine components. For example, assuming five cycles with 128 points per cycle, there will be $128 \times (5 - 1) + 2 = 513$ data windows (i). The cosine function can be calculated using Eq. (6), where $i = 1, 2, 3, \ldots, 128 \times (5 - 1) + 1$.

4.2 Zero-crossing algorithm for computing power signal fundamental frequency

The zero-crossing algorithm is the most direct and simple approach for frequency calculation, and it has fast execution without the use of a complex mathematical formula. For one cycle signal, the period between two zero-crossing points is half a signal period. When using the signal before and after the zero-crossing position, by numerical computation, converting into units of time, and after seeking the reciprocal, the actual frequency of the power signal can be obtained. The following is the zero-crossing algorithm, and the solution process is described.

Consider a power signal waveform to be applied by a zero-crossing algorithm. Such a sine wave can be represented as

$$y(t) = A\sin(\omega t), \tag{11}$$

where A is the amplitude, $\omega = 2\pi f$ is angular frequency, and f is the signal frequency.

When the $t = kT_s$ signal represented in a discrete manner is substituted, Eq. (11) may be expressed as

$$y(k) = A\sin(\omega k T_s). \tag{12}$$

In the process of signal sampling, the angle difference between two continuous points is

$$\Delta\theta = \omega T_s = 2\pi f T_s, \tag{13}$$

where T_s is the sampling interval. Assuming that the sampling points of a full cycle is m, the signal cycle is equal to the sampling interval multiplied by the number of points. Therefore, the signal frequency can be calculated as

$$f = \frac{1}{mT_s}. \tag{14}$$

In practice, because the sampling frequency is not an integer multiple of the actual signal frequency, the signal period is not an integer multiple of the sampling period. Therefore, an accurately calculated m value is not usually an integer. When we observe the T_i cycles, Fig. 10 shows zero crossing points (Y_{i-1}, Y_i) from negative to positive, and secondly, zero crossing points (Y_{i+m}, Y_{i+m+1}) from negative to positive, and the exact period of the sine wave can be expressed as

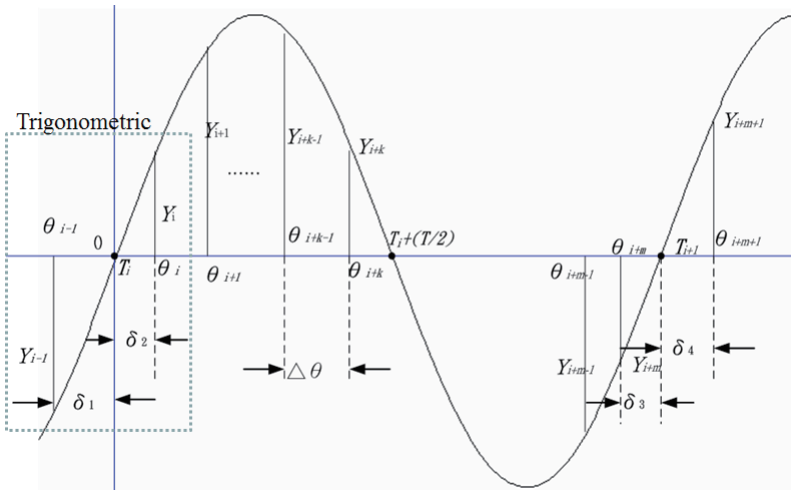


Fig. 10. Sine waveform for zero-crossing algorithm calculation.

$$T = mT_s + \delta_2 + \delta_3 = m_A T_s = m_A \Delta\theta, \tag{15}$$

where $T_s = \Delta\theta$.

Because the sampling interval is not an integer multiple of the signal cycles, and m_A is not an integer, according to Fig. 10, m_A is obtained as

$$m_A = m + \frac{\delta_2}{\Delta\theta} + \frac{\delta_3}{\Delta\theta} = m + \frac{\delta_2}{\delta_1 + \delta_2} + \frac{\delta_3}{\delta_3 + \delta_4}, \tag{16}$$

where δ_2 is the time difference between the i-th sampling point and the zero point (T_i). In Fig. 10, using a definition similar to the trigonometric, Eq. (17) is expressed as

$$m_A = m + \frac{|y_i|}{|y_i| + |y_{i-1}|} + \frac{|y_m|}{|y_m| + |y_{m+1}|}. \tag{17}$$

The true frequency of test signals can be estimated as

$$f = \frac{1}{m_A T_s} = f_a \frac{m}{m_A}. \tag{18}$$

Even if y_i or y_m is separately located under the conditions of T_i and T_{i+1} (zero amplitude value), Eqs. (17) and (18) can still be used for frequency detection. In terms of electric power measurement algorithms, the frequency estimation is based on zero-crossing algorithms, so the algorithm is simple and fast.

As a matter of fact, several factors still exist that affect the computation of the zero-crossing algorithm such as greater harmonic amplitude, non-integer harmonics, and even noise. That is the reason why the modified Fourier series approach is proposed and applied in this research. In other words, by adding Fourier-series-based filtering, the traditional zero-crossing algorithm can considerably improve the calculation of the fundamental frequency.

5. Measurement and Verification[13]

In our research, we try to focus on a convenient and low-cost mobile device for power signal measurement. It is worth noting that we have provided here not only a simple and inexpensive frequency measurement device but also an acceptable and comparable accuracy with an error rate smaller that 0.1% by using a simple computing Arduino device and an Android app. That is, we do not need a high-performance processor and memory space to process the frequency measurement when using the Fourier algorithm and zero-crossing technique.

In Ref. 13, the traditional Fourier algorithm provided an error range about 0.1940–0.0110%, and the improved Fourier algorithm[13] yielded better results, a 0.0638–−0.0001% error range, which is calculated based on the empirical model and used the least error squares to fit the error compensation curve to correct the result. Table 4 shows the comparative results for M1, changing the amplitude of harmonic components; M2, changing the phase of harmonic components; and M3, changing both the phase and the amplitude of harmonic components. Obviously, the error rate is closely dependent on the amplitude of the measured signal.

5.1 Measurement verification of Android program[14–16]

In this study, using the app Inventor2, we wrote the Android program, used LabVIEW simulation to mix three different frequencies and amplitudes of sine waves, and adopted NI-myDAQ to output signals while simultaneously receiving the waves in LabVIEW as shown in Fig. 11. Signals were captured by the Arduino and then transmitted to a web site via WiFi Shied; the Android app captures the web site data, which was then drawn by Google Chart API as shown in Fig. 12. Compared with a portable device drawing data received directly from NI-myDAQ, the measured error rate is less than 0.1%, which confirmed the Arduino hardware's reliability and practicality.

Table 4
Comparison of frequency measurement error rate with improved algorithm (IA) and traditional algorithm (TA).[13]

Errors (%) \ f (Hz)		49.5	49.7	49.9	50.1	50.3	50.5
M1	TA	0.1771	0.0783	0.0158	−0.0052	0.0144	0.0678
	IA	0.0468	0.0200	0.0034	−0.0016	0.0042	0.0190
M2	TA	0.1458	0.0623	0.0122	−0.0042	0.0077	0.0377
	IA	0.0156	0.0039	−0.0001	−0.0006	−0.0025	−0.0111
M3	TA	0.1940	0.0816	0.0158	−0.0053	0.0110	0.0519
	IA	0.0638	0.0233	0.0035	−0.0017	0.0007	0.0031

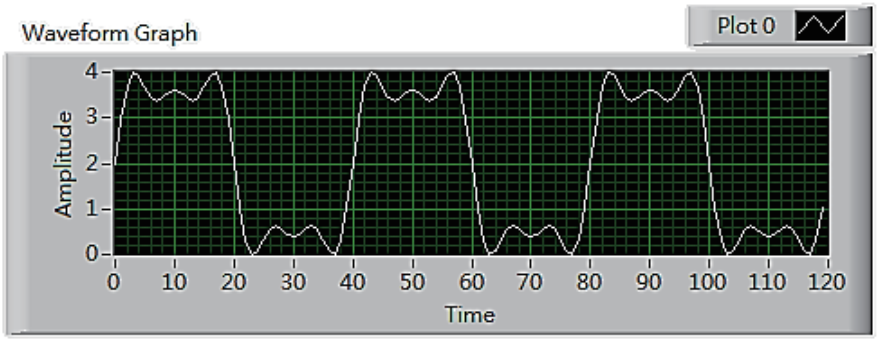


Fig. 11. (Color online) Mixed signal simulated with fundamental and 3rd and 5th harmonic signals for different amplitudes of 1, 0.4, and 0.2 V.

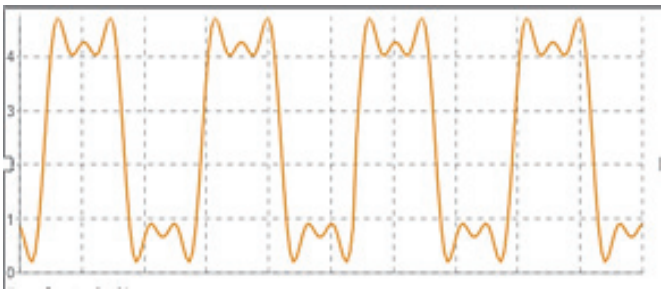


Fig. 12. (Color online) Mixed signal drawing received on a mobile device.

5.2 Power quality algorithm verification

The purpose of this study was to measure the signal quality of utility power. After stepping down voltage using the circuits shown in Fig. 13, the signals captured by the Arduino are processed by ADC via WiFi Shield network transmission, so the Android app can plot and monitor the signals after receiving them. After the power signal acquisition, Fourier filter calculation is based on Eq. (6) as in the flowchart shown in Fig. 12, and then Eq. (17) is used to calculate the zero-crossing algorithm. The cycle time and frequency of power fundamental signals are calculated as shown in Fig. 14. Figure 15 shows the step-down devices for utility power and the photos received by the mobile devices. Figure 16 shows the stepped-down utility power signals received by the Android app and drawn by Google Chart API plotting. Thus, the noise of this wave is filtered out by Fourier calculation to make it a sine wave, as shown in Fig. 17.

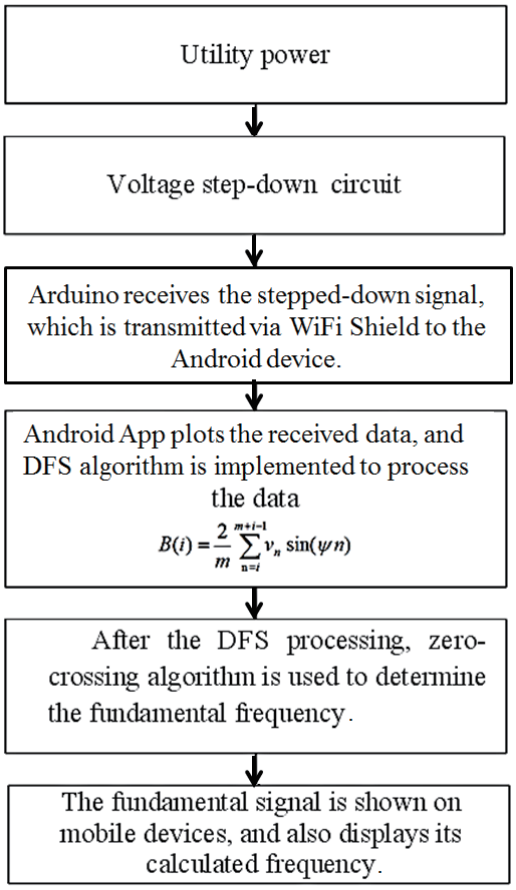


Fig. 13. Flowchart for DFS algorithm.

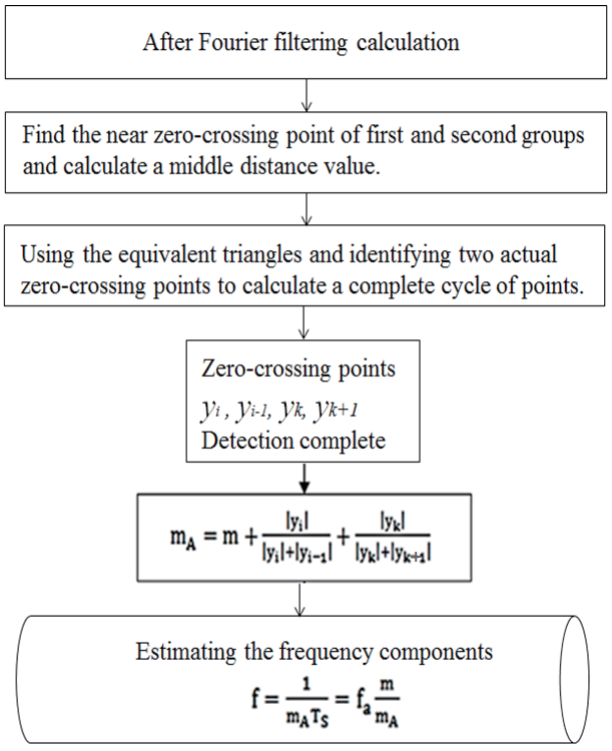


Fig. 14. Zero-crossing flow diagram.

Fig. 15. (Color online) Setup for the remote power signal measurement implementation.

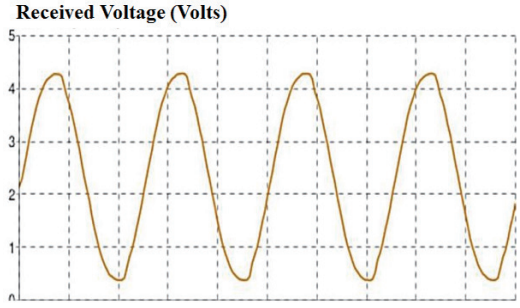


Fig. 16. (Color online) Receiving signal after voltage stepping-down treatment.

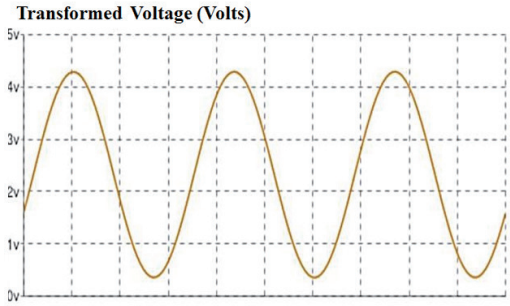


Fig. 17. (Color online) Signal plot after DFS.

5.3 Client–server architecture

In this study, in addition to using an Android app to perform an algorithm at the mobile device end, power signal data are also transmitted to the MySQL database for storage. Then, the server side carries Fourier and zero-crossing algorithms to overcome the problem of inadequate CPU computing at the mobile device end. After the mobile devices capture the data transmitted by the Arduino via WiFi Shield, by PHP web server design, the data is uploaded to the MySQL database. Then, other mobile devices open a web connected to the server by the corresponding app. After the website function through PHP syntax reads out the data in the database, Fourier filtering and zero-crossing algorithm computing are carried out. Finally, the JavaScript graph function is used to present data and Fourier filtering graphics data on the web. The mobile device directly loads the web page to show the final results illustrated in Fig. 18.

Given that a large amount of data or multiple data sets may be uploaded simultaneously and to keep the uploading and algorithm results efficient, in the future, MongoDB will be used to implement a cloud database to cope with loading and operating on large amounts of data.

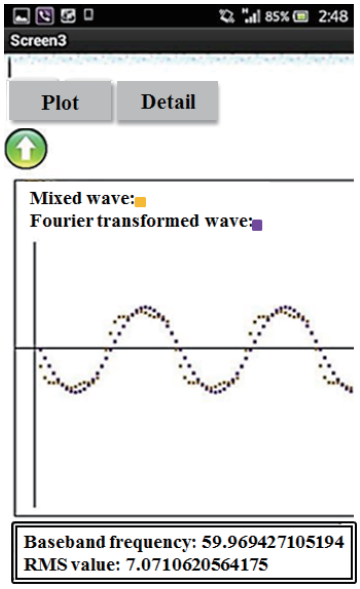


Fig. 18. (Color online) Signal plot of power signal acquired and shown on mobile device.

6. Conclusions

In this paper, we have shown that the Fourier algorithm can be used as digital filter to extract cosine and sine waveforms of the fundamental frequency component, and the zero-crossing method can be applied to obtain the baseband amplitude and frequency. Additionally, we had shown that the Fourier algorithm accompanied with zero-crossing approach is robust and effective. Using the DAQ card NI myDAQ, the Arduino DAQ accuracy shows less than a 0.1% error rate, which is better than the traditional Fourier algorithm. Additionally, our research results show the suitability of a low-cost mobile device.

It is planned that power signals captured by a DAQ card device will be sent to a cloud server by Android mobile devices to analyze the operation of captured signals. Finally, mobile devices are used to monitor the measured results. Finally, we have completed in this paper research on the DAQ technology, development of the power line quality algorithm , development of the WiFi mobile phone application, and implementation of the client–server system for power signal analysis.

References

1 T. Q. Tran, L. E. Conrad, and B. K. Stallman: IEEE Trans. Power Delivery **11** (1996) 1041.
2 R. C. Dugan, M. F. McGranaghan, and H. W. Beaty: Electrical Power Systems Quality (McGraw-Hill International Editions, 2000).
3 ATMEL application note: AVR120: Characterization and Calibration of the ADC on an AVR, http://www.atmel.com/dyn/resources/prod_documents/DOC2559.pdf (accessed November 2016).
4 A. I. Abu-El-Haija: Proc. 2000 IMTC/2000 (2000) p. 941.
5 C. H. Hong: Master's thesis, Institute of Electrical Engineering, National Taiwan University (2001).
6 Arduino: The Arduino DAQ Chronicles, https://sites.google.com/site/measuringstuff/the-arduino (accessed November 2016).

7 M. Margolis: Arduino Cookbook (O'Reilly, Sebastopol, 2011).
8 P. L. Meyer: Introductory Probability and Statistical Applications (Addison-Wesley, Reading, 1970).
9 M. B. Đurić and Ž. R. Đurišić: Renewable Energy Power Quality J. **1** (2005) 330.
10 M. B. Đurić and Ž. R. Đurišić: Electronics **7** (2003) 11.
11 I. Kamwa and R. Grondin: IEEE Trans. Power Delivery **7** (1992) 789.
12 P. J. Moore, R. D. Carranza, and A. T. Johns: Proc. Inst. Elect. Eng. (1994) p. 529.
13 J. Zhang and X. Cai: 2012 Int. Conf. Future Electrical Power and Energy Systems (Energy, procedia) p. 1697.
14 J. H. Zeng, Y. T. Cai, K. Q. Huang, W. M. Lai, W. Y. Lu, and L. W. Shi: Android Phone Super Simple Program!! - Introduction Volume (Fu Lin Press).
15 D. W. Zhang: Master's thesis, Department of Electrical Engineering, Shou University (2006).
16 Z. W. Huang, T. Gao, H. Y. Zhang, X. Han, J. W. Cao, Z. H. Hu, S. J. Yao, and Z. G. Zhu: China Int. Conf. Electricity Distribution (2014) p. 23.