Sine Inverter Controller with 8 Bit Microcontroller

Radim KUNCICKY, Lacezar LICEV, Michal KRUMNIKL, Karolina FEBEROVA, Jakub HENDRYCH

Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava, Czech Republic

radim.kuncicky@vsb.cz, lacezar.licev@vsb.cz, michal.krumnikl@vsb.cz, karolina.feberova@vsb.cz, jakub.hendrych@vsb.cz

DOI: 10.15598/aeee.v14i3.1715

Abstract. *This article describes the design of a sine wave inverter control unit. We will define the basic principles and requirements to control and maintain a good quality of inverter's output. The solution that can achieve full 8 bit output resolution with 10 bit measurement and regulation time shorter than mains period is proposed. Furthermore, the possibilities and ways of implementing the control unit using the 8 bit microcontrollers are described, so it is possible to gain a complete understanding of this issue.*

Keywords

Inverter, microcontroller, PID, pulse width modulation, regulator, sine wave inverter.

1. Introduction

In the recent years, there has been a large expansion of the solar plants, mainly thanks to the huge subsidy provided by the Czech Republic government. It was followed by a rapid price fall of the solar panels, which started to appear on the roofs all over the country. Solar panels can serve as an independent autonomous power system. The power from the cells is stored in batteries and can be used later on, when needed. However, the batteries cannot be connected directly to the appliances, since they do not provide correct voltage. Therefore, it must be transformed via an inverter providing an output voltage of 230 V and frequency of 50 Hz.

Inverters are very well known since the time of first switching semiconductors. Due to the versatility, efficiency, size and costs, they became one of the best choices in situations when there was a need to change the voltage, waveform or current.

The simplest constructions use a low frequency Pulse Width Modulation (PWM) to approximate a line sinusoidal waveform with a rectangular waveform. This type of converter is not very suitable to power many commonly used appliances and can also be dangerous. For example, let's mention devices based on a single phase asynchronous motor. The rectangular waveform is not able to create rotating magnetic field and the motor will not work correctly. These motors can be found in building's heating systems as a circulation pump and their failure may result in considerable damages. Another example is energy-saving lamps, which can be even destroyed by the sharp rising edge. Therefore, more advanced inverters must be used.

A review of common multilevel inverter topologies and control schemes has been covered in [1]. Multilevel inverters are typically used in medium and high power applications due to their lower dissipation, harmonic distortions and electromagnetic interference. However, the topic of this article is focused on smaller inverters, typical for small solar power plants. These inverters use Synchronous Pulse Width Modulation (SPWM) or simple PWM for driving output circuits. A comparative study of multiple inverters approaches, including SPWM and pure sine wave techniques, providing their main characteristics and advantages was publisher in [5]. The performance analysis of square wave and modified square wave inverters [6] showed significant performance drawbacks and higher harmonics. We can say that the only disadvantage of the sine wave inverters is the higher complexity compared to the simple PWM units. As the sine inverters need more complex drivers, several options were proposed ranging from general purpose microcontrollers, microcontrollers with DSP

units, FPGA and single purpose (dedicated) integrated circuits. The inverter can be based even on multivibrator IC, but this is rather an awkward construction [10].

General purpose microcontrollers and FPGAs are popular choices for driving sine wave inverters. A single phase full bridge inverter using Field Programmable Logic Array (FPGA) was proposed in [2] and [4]. The designs based on microcontrollers differ mainly in the control software and microcontroller architecture, the hardware design remains almost the same, e.g. [7], [8] and [9]. Probably the most similar approach to ours is the work presented in [3]. However, proposed sine inverter based on 8 bit PIC microcontroller 16F84 provides only basic functions without voltage regulation.

The sine inverter we describe approximates the AC waveform using a high frequency PWM. Such inverter has the output waveform (after the filtering) much more similar to the true sine wave. Moreover, we have added additional features, such as frequency shifting, battery monitoring and computer control interface.

The operation of the proposed inverter is controlled by the control unit. The control unit is responsible for the correct pulse timing, pulse duration, measurement and maintaining the operating parameters within the required limits.

The remaining part of the article discusses the implementation of the converter control unit based on a modern 8 bit microcontroller. Special attention is given to the computational performance and optimization techniques.

2. Pulse Inverters

The selection of the suitable power source type is heavily affected by the requirements of the application. For the linear power sources speaks the simple design, silent operation and relatively low price. As the biggest disadvantage, we can consider low efficiency that ranges between $30-60$ %. Another problem is required power design factor, where every component needs to be designed according to the limits of possible operating load. These edge conditions are rarely met in everyday operation. It is important to mention a significant constraint of the linear power supplies - they are not able to operate in a step-up mode. The pulse inverters provide much higher efficiency, up to $68-90$ %, but they require more complex structural design. A substantial part of the structural complexity lies in the necessary control electronics. The following Section describes the basic principles of an increasing DC to AC inverter.

Understanding the basic principle is not hard. A major difference from the linear regulator is that the linear sources continuously regulate the amount of current to reach the constant output voltage. The switching power supplies regulate the flow of the current by chopping the same input voltage and maintaining the mean value of the output voltage by changing the duty cycle. If it is necessary to supply greater amount of the current to the load, the width of pulse is increased to compensate the change.

The DC input voltage can be converted to the AC by the semiconductor circuit operating in the switching mode. The AC current is transformed and the output voltage is, therefore, dependent on the conversion ratio. The operating frequency of such transducers is in the range above the audible range, at frequencies higher than 20 kHz. The transformer does not require a heavy iron core, but due to the higher operating frequency, it can use a lightweight ferrite core.

The transistors Q_A and Q_B in Fig. 1. are driven by the control unit. They supply the current to the different parts of the primary wiring. On the other hand, the current is rectified and smoothed by the output filter. The output voltage and current are sensed by the probes of the control unit. The control unit evaluates the deviation between the output voltage and the reference and uses the controlling algorithm to determine the width of the pulse width modulation. The unit calculates the duration of switching Q_A and Q_B transistors. The method is typical for the low-frequency modulation. For the high frequency PWM, the switching time of the transistor is based on the regulation algorithm and the current instantaneous voltage value.

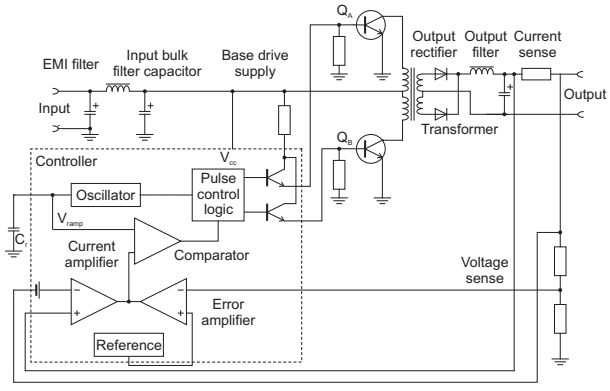

Fig. 1: A walk through a representative switching regulator circuit [12].

The control unit must provide the following functions:

- voltage measurement,
- current measurement,
- calculation of the regulation deviation,
- evaluation of the regulation algorithm,

- generation of the PWM carrier frequency,
- PWM coding.

The control unit may also include other parts like circuits for synchronization, soft start or remote control and monitoring.

For the testing purposes, we have created a simple push pull construction made of discrete components. The power of the prototype was limited to 100 W as it was not designed for deployment in the real-world conditions.

3. Control Complexity and Microcontrollers Performance

In this Section will be discussed the performance requirements that are necessary to be considered for a suitable microcontroller core selection.

We have chosen Atmel RISC architecture (called AVR), with performance up to 20 MIPS at 20 MHz clock timing. The core is capable of processing single instruction per clock cycle and provides two cycles 8 bit hardware multiplier. An important advantage is that it is supported by the highly optimized C compiler. The developed control unit was based on the ATMega 644PA model.

The performance requirements for the inverter controller can be easily demonstrated as follows. In the previous Section, we have discussed that the switching frequency should be higher than 20 kHz. When using 16 MHz clock frequency and 8 bit counter in the Fast PWM mode, the switching frequency is given by the Eq. (1) [11]:

$$f_{\mathrm{PWM}} = \frac{f_{osc}}{\mathrm{prescaler} \cdot \mathrm{counter}_{\max}} = \\ = \frac{16}{1 \cdot 256} = 62.5 \text{ kHz}. \tag{1}$$

When using the phase-correct mode, the modulation frequency will be 31.3 kHz. From Eq. (1), it is obvious that the core of the microcontroller will be forced to handle the interrupt from counter every 255 machine cycles.

Handling the AVR architecture interrupt and the return jump require about 10 cycles on average. Therefore, only 245 cycles remain (until the arrival of the next interrupt) for calculating the regulation algorithm and processing any additional features.

Mathematical operations and especially multiplication can quickly become a bottleneck of the software performance. However, they are necessary for computing the pulse width or measurement of the operating parameters. The Tab. 1 gives an idea of an AVR core performance, such as the machine cycles counts for different kinds of software multiplications with the optimized versions of the algorithm.

Tab. 1: AVR core performance [13].

	Machine cycles	**Code size**
8*8=16 bit unsigned	34	34
16*16=32 bit unsigned	105	105
8/8=8+8 bit unsigned	66	58
16/16=16+16 bit unsigned	196	173

In the worst case, the multiplication of two double data precision float types can take almost up to 1800 cycles. This is totally unacceptable for this application.

The preceding paragraphs assume that calculations are made by the software and processed by the binary adder. Most of the modern MCUs include a hardware multiplication unit. We provide the Tab. 2 for a quick comparison. The table shows performance for several families of microcontrollers. We have included different families of 8 to 32 bit microcontrollers. We have highlighted the number of cycles needed to execute unsigned 16×16 multiplications. It should be noted that not all architectures necessarily complete one instruction per one machine cycle. The instruction completion may consume more cycles.

Tab. 2: Machine cycles count to perform multiplication on different MCU architectures.

PIC 16F	**PIC 18F**	**AVR 8**	**dsPIC 33F**	**Atmel xMega**	**Cortex M0**
140	28	19	1	1	1

As you can see, the 32 bit MCUs are equipped with hardware multiplier of sufficient width. On the other hand, the 8 bit MCU multipliers are limited with the bus width, which is usually only 8 bit.

It was necessary to carefully consider the desired precision of each variable. For the development purposes, it was decided to keep the accuracy of calculations between 8 and 16 bits.

One of the basic techniques for limiting calculations is the usage of a look-up table. This technique can be demonstrated on the calculation of the sine function. If we know that the input data will always be in the range from 0 to 255 - i.e. with an accuracy of 8 bits and the desired result will be also in the same range, the simplest solution is to allocate an array of 256 elements to which the individual results will be pre-calculated as shown in following code example. The calculation functions can be then limited to a simple memory access, which is much faster.

```
void initSinusAprox() {
  for( uint8_t i=0;i<255;i++) {
    SinusAprox[i]
    = 255 * sin( i *(M_PI / 256));
  };
};
```

We can consider some loss of the legibility of the code as the biggest disadvantage. When we call the function `sin(value)`, it is immediately apparent that the unit of `value` is radian. When accessing the `SinusAprox[value]` array the parameter is losing its dimension and obviously its accuracy as well.

If the situation requires the use of more tables for the calculation, it could be necessary to convert the dimensionless variable to some unit. This is the case of the creation of, so called, magic numbers. Their origin or purpose is not obvious at first glance. The limiting factor for this approach is the amount of available memory.

Another significant limitation factor is the very simple interruption unit, which does not allow the advanced techniques, such as the masking. Interrupts are processed according to the priorities that are set by the address of the interrupt - the lower address, the higher priority. It seems logical to divide software into individual layers according to the priority given to their functionality. The masking is a necessary precondition for this approach. The PWM signal generation is the most crucial function while for the communication, we can assign lower priority. Simple interrupt unit is a hardware limitation that is not possible to overcome by the software. The only option is to use the nested interrupts (but with the risk of stack overflow).

4. Generating PWM

The pulse width modulation is discrete in its value and the modulation variable is voltage or current. The value is transmitted and encoded by the impulse width during the constant period. Width can take a range of $0 - 100$ %. We need to know the range of modulation voltage in advance, because when the maximum value is exceeded, the information is lost.

The PWM has to fulfill the condition of the Shannon Nyquist theorem. The carrier frequency must be twice greater than the maximum frequency of the transferred signal.

The carrier and the modulation signal are compared in the comparator. When the value of the carrier signal is smaller than the modulation, the output of the comparator is set to logical one.

The microcontrollers offer the ability to generate the PWM with the usage of the integrated counters. The counter operates with the constant frequency and the maximum is usually set to the maximum value of the counter. Usually, two interrupts are needed. When reaching the maximum value, the output is set to log 0. The current value of the modulation signal is stored in the comparator register. If the counter register value is equal to the comparator register, the output is flipped to log 1. Generating is implemented in the hardware and it does not require any additional cooperation with the core. The principle of the operation and usage of the registers is shown in Fig. 2.

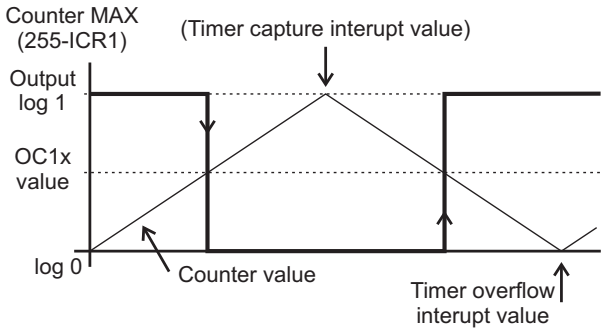

Fig. 2: Registers usage for PWM generating.

5. Measurement and Regulation

For the regulation purposes, it is necessary to know the regulation deviation, which is derived from the desired value and the real value of measured output. For the measurement we used an integrated 8-channel approximation converter with the 10 bit resolution.

The timing of the conversion was derived from the master clock of the microcontroller via the frequency divider. The suitable timing was determined to be in the range from 50 to 200 kHz. The conversion is finished in 13 cycles. Firmware was designed to measure three variables. From the above-mentioned parameters, we can derive the maximum sampling rate as in Eq. (2):

$$f_{\mathrm{SAMPLE}} = \\ = \frac{f_{CPU}}{\mathrm{Prescaler}_{\mathrm{ADC}} \cdot \mathrm{ConversionCycles} \cdot \mathrm{Channels}} \\ = \frac{16}{128 \cdot 13 \cdot 3} = 3.2 \text{ kHz}. \tag{2}$$

The battery voltage, which is not galvanically separated from the control unit, is fed to the pins of the microcontroller via the voltage divider. The protection against the battery undervoltage has been implemented in the firmware to avoid the battery damage. The measuring range is $0 - 15$ V with an accuracy of 8 bits.

In the prototype, it was necessary to isolate the voltage and the current of the output. The output current is measured on the resistor using the SHF-615 optocoupler. When using the optocoupler, it is necessary to compensate its non-linear characteristics. So the conversion characteristics of the current and the ADC value had to be determined. Subsequently, the linearization using three lines with the least squares method was performed. The transfer function is stored in a look-up table. The measured current values are used in the overload protection.

The output voltage is measured with the voltage measurement transformer that reduces the amplitude for the ADC converter. The input has a floating center, which is equivalent to 511 after the conversion. With the 10 bit, it is possible to achieve a range of ±350 V with a resolution of ±512. That means that the accuracy of measurement is approximately 1 V.

The evaluation of the true Root Mean Square (RMS) of output voltage is performed according to the measured values. Each calculation is made from 256 samples according to the Eq. (3):

$$U_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\sum x_n^2}. \tag{3}$$

The calculation of the true RMS is the most difficult operation, which is implemented in the control unit.

One of the considered options was to use the IC with the function of the true RMS to the DC conversion. The generally available ICs for this purpose are, e.g., AD 7360 or LTC 1966. However, the full integration was prioritized.

The converter, the output sensing and the control unit together constitute the control loop. The evaluation and the maintaining of a constant output are the main tasks of the controller.

The PWM width is changed based on sensing the output. The goal is to achieve a state where the actual value corresponds to the desired output value. The stability of the output can be disrupted by the interference sources, such as changing the load.

The control unit determines the control difference according to the Eq. (4):

$$e = w - x, \tag{4}$$

where e is regulation error, w is desired value, and x is measured value.

The deviation value itself is not enough to run the regulation. It is necessary to determine the y set value of the output. The procedure used for the determination of the value is called the regulation algorithm. In general, we can use two types of controllers. The adjustment algorithm calculates the adjustment function and the velocity algorithm calculates the additions to the y value. In our case we use the discrete (digital) PI velocity algorithm.

5.1. PID Controller

The setting signal consists of a weighted sum of the three components. P part, the integral of P component (I part) and the derivation of P component (D part). The part P is proportional to the control difference. The P controller creates an immediate permanent lasting change of the adjustment signal, when the first control difference occurs. I part creates an active part of the adjustment signal even at small regulatory differences, which can remove the lasting P difference at the achievable time. D member is applied while changing the regulatory differences. It would predictable overshoot the adjustment of the signal and speed up the compensation of the difference.

The PID is difficult to setup. For this reason, P or PI controllers are used. The suitable controller is selected according to the regulatory dependences and the behavior of the object.

The Discrete adjustment algorithm is given by the following Eq. (5):

$$\begin{aligned} &\text{P part: } y_{Pn} = K_p \cdot e_n, \\ &\text{I part: } y_{In} = K_p \cdot \frac{T_A}{T_I} \sum_{i=0}^{n} e_i, \\ &\text{D part: } y_{Dn} = K_p \cdot \frac{T_V}{T_A} \cdot (e_n - e_{n-1}), \end{aligned} \tag{5}$$

where K_p is proportionality constant, T_I integral time constant, T_V is derivative time constant and T_A is sampling period time.

By using the modified Eq. (5), the Eq. (6) for the speed algorithm can be obtained. From this Eq. (6) we can obtain the equation of PI controller by omitting the derivative part.

$$\begin{aligned} \Delta y_n = K_P \Big[& e_n - e_{n-1} + \frac{T_A}{T_I} \cdot e_n + \dots \\ & \dots + \frac{T_V}{T_A}(e_n - 2e_{n-1} + e_{n-2}) \Big]. \end{aligned} \tag{6}$$

It is obvious that the calculation is performed from control deviations and few constants that are the object of the optimization.

The controller is carefully set, so it quickly reaches the desired value without the unwanted overshooting. The value of regulatory response t_0 describes the time of reaching the set point.

The simplest method for achieving this is the Ziegler-Nichols method [14] of the critical gain. During the

setting up, at first the integral and then the derivative part are eliminated. The constant of proportionality is incremented from zero until reaching KU, where the regulation loop starts to undamped oscillation with the constant amplitude and period t_p. Then the parameters of proportional and derivative component can be calculated according to the Tab. 3.

Tab. 3: Setting of controllers constants by Ziegler-Nichols method.

Type	K_P	K_I	K_D
P	$0.5K_P$	-	-
PI	$0.45K_U$	$\frac{1.2K_P}{T_P}$	-
PD	$0.8K_U$	-	$\frac{K_P T_P}{8}$
PID	$0.6K_U$	$\frac{2K_P}{T_P}$	$\frac{K_P T_P}{8}$

This optimization implementation is suitable for development and educational purposes, some advanced tuning methods based on fuzzy logic, artificial neural networks or genetically inspired algorithms achieve much more precise results. It is also possible to implement several advanced and modified PID algorithm versions for example predictive PID variant as supposed in [15].

6. Communication

One of the design requirements was the integration of the communication and remote control of the unit.

The RS 232 interface has been selected due to its simplicity and widespread in personal computers, as well as in industrial applications.

Atmel AVR microcontrollers contain integrated, full-duplex, hardware-operated universal synchronous/asynchronous serial ports.

The clock generator is derived from the MCU clock. The port provides high precision and allows a large spectrum of modulation speeds. There are three interrupt vectors available for the communication control, receiving a byte, empty the output queue and complete a transmission. For the controlling purpose, a simple text-based protocol was implemented in the firmware. Characters are encoded in ASCII. The protocol operates in the request/response mode. The unit acts as the slave controlled by the master.

The beginning of the message is indicated with the Start of Text character (0×02). STX is followed by one character called the function code, which can be followed by a numeric parameter. The message ends with the End of Transmission (0×04). In the firmware, we have implemented messages related to the operational status, monitoring of the input voltage, the output voltage and the current. It is possible to setup the control constants and the reference values as well. For example, the message for setting the reference output value in volts should look like this: STX, E230, EOT. If the completion of setup message is successful, the answer of the firmware is OK.

The only byte transfers from USART registers are operated within the interrupts. The actual processing is carried out in the main loop. The time used in the interrupt is minimized and the idle time in the main loop is used as much as possible.

7. Functionality Testing

The development of both hardware and software parts coincided. The various implementation variants were considered in order to find the optimal solution.

In the first part, the basic principles of generating the sine wave using the PWM were tested. The output of this stage is shown in Fig. 3, showing the output sine wave filtered by the RC elements.

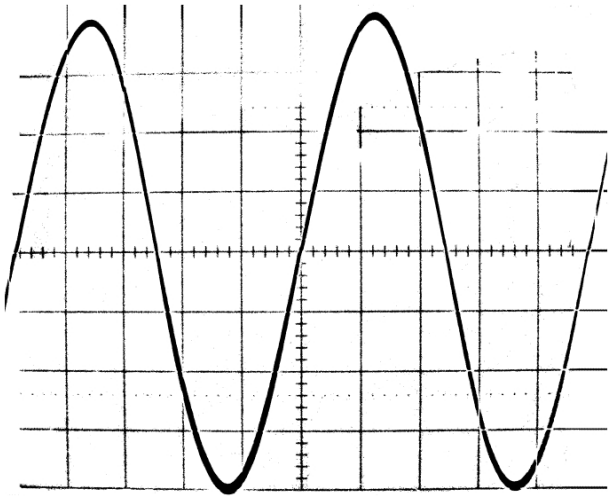

Fig. 3: Sine wave generated by MCU (H: 4 ms per div, V: 80 V per div).

The testing and the verification of the measurements followed. If the firmware is able to effectively measure the output values, the regulatory deviation can be calculated and the control algorithm checked. The P algorithm itself was not sufficient, because of its permanent deviation. The PI algorithm was sufficient enough. The output regulation worked relatively slowly due to the low computational power of the microcontroller. The convergence time t_0 was about 1.5 seconds. The

integration constant was adjusted high enough to compensate this phenomenon. However, as the result, this causes a small overshoot of the desired value.

8. Conclusion

The article describes the implementation of the controlling algorithm of the sine wave inverter for the appliances sensitive to the correct supply voltage waveform. The design used an 8 bit microcontroller. We have demonstrated the ability of the simple monolithic microcontrollers to handle this critical application. The price of needed optimization is a large number of man hours needed for coding.

The simple MCU did not offer priority interrupts; thereby it was not possible to divide the firmware into appropriate priority areas and implement all functionalities.

Today, the price of the ARM family is falling very fast and the developers do not have to be limited to 8 bit cores. More suitable architecture would be Atmel's XMEGA or Microchip's dsPIC [16].

This work provided a comprehensive understanding of the possibilities of the inverter control unit firmware development.

Acknowledgment

This work was partially supported by Grant of SGS No. SP2015/142, VSB–Technical University of Ostrava.

References

[1] COLAK, I., E. KABALCI and R. BAYINDIR. Review of multilevel voltage source inverter topologies and control schemes. *Energy Conversion and Management*. 2011, vol. 52, iss. 2, pp. 1114–1128. ISSN 0196-8904. DOI: 10.1016/j.enconman.2010.09.006.

[2] AFARULRAZI, A. B., M. ZARAFI, W. M. UTOMO and A. ZAR. FPGA implementation of Unipolar SPWM for single phase inverter. In: *International Conference on Computer Applications and Industrial Electronics*. Kuala Lumpur: IEEE, 2010, pp. 671–676. ISBN 978-1-4244-9054-7. DOI: 10.1109/ICCAIE.2010.5735019.

[3] MAMUM, A. A., M. F. ELAHI, M. QUAMRUZZAMAN and M. U. ZOMAL. Design and Implementation of Single Phase Inverter. *International Journal of Science and Research (IJSR)*. 2013, vol. 2, iss. 2, pp. 163–167. ISSN 2319-7064.

[4] NGALAMOU, L. and L. MYERS. Digital SPWM synthesis for the design of single phase inverters. *International Journal of Electronics*. 2008, vol. 95, iss. 5, pp. 489–503. ISSN 0020-7217. DOI: 10.1080/00207210801976420.

[5] CHEEMA, M. B., S. A. HASMAIN, M. M. AHSAN, M. UMER and G. AHMAD. Comparative analysis of SPWM and square wave output filtration based pure sine wave inverters. In: *15th IEEE International Conference on Environment and Electrical Engineering (EEEIC)*. Rome: IEEE, 2015, pp. 38–42. ISBN 978-1-4799-7992-9. DOI: 10.1109/EEEIC.2015.7165289.

[6] KUMAR, N., D. JOSHI and S. SINGHAL. Design, implementation and performance analysis of a single phase PWM Inverter. In: *6th IEEE India International Conference on Power Electronics (IICPE)*. Kurukshetra: IEEE, 2014, pp. 1–6. ISBN 978-1-4799-6045-3. DOI: 10.1109/IICPE.2014.7115742.

[7] QAZALBASH, A. A., A. AMIN, A. MANAN and M. KHALID. Design and implementation of microcontroller based PWM technique for sine wave inverter. In: *International Conference on Power Engineering, Energy and Electrical Drives*. Lisbon: IEEE, 2009, pp. 163–167. ISBN 978-1-4244-4611-7. DOI: 10.1109/POWERENG.2009.4915171.

[8] HAIDER, R., R. ALAM, N. B. YOUSUF and K. M. SALIM. Design and construction of single phase pure sine wave inverter for photovoltaic application. In: *International Conference on Informatics, Electronics & Vision (ICIEV)*. Dhaka: IEEE, 2012, pp. 190–194. ISBN 978-1-4673-1153-3. DOI: 10.1109/ICIEV.2012.6317332.

[9] CHOWDHURY, A. S. K., M. S. SHEHAB, M. A. AWAL and M. A. RAZZAK. Design and implementation of a highly efficient pure sine-wave inverter for photovoltaic applications. In: *International Conference on Informatics, Electronics & Vision (ICIEV)*. Dhaka: IEEE, 2013, pp. 1–6. ISBN 978-1-4799-0397-9. DOI: 10.1109/ICIEV.2013.6572634.

[10] HASAN, M., M. Q. BAIG, J. MAQSOOD, S. M. A. S. BUKHARI and S. AHMED. Design & Implementation of Single Phase Pure Sine Wave Inverter Using Multivibrator IC. In: *17th UKSIM-AMSS International Conference on Modelling and Simulation*. Cambridge: IEEE Computer Society, 2015, pp. 451–455. ISBN 978-1-4799-8713-9.

[11] AVR131: Using the AVR's High-speed PWM: AVR 8-bit Microcontrollers. In: *Atmel* [online]. 2016. Available at: http://www.atmel.com/Images/Atmel-2542-Using-the-AVR-High-speed-PWM_ApplicationNote_AVR131.pdf.

[12] BROWN, M. *Practical switching power supply design*. San Diego: Academic Press, 1990. ISBN 0121370305.

[13] AVR200: Multiply and Divide Routines: 8-bit Microcontrollers. In: *Atmel* [online]. 2009. Available at: http://www.atmel.com/Images/doc0936.pdf.

[14] ZIEGLER, J. G. and N. B. NICHOLS. Optimum settings for automatic controllers. *Journal of dynamic systems measurement and control*. 1993, vol. 115, iss. 2B, pp. 220–222. ISSN 0022-0434. DOI: 10.1115/1.2899060.

[15] AYLOR, J. H., R. L. RAMEY and G. COOK. Design and Application of a Microprocessor PID Predictor Controller. *IEEE Transactions on Industrial Electronics and Control Instrumentation*. 1980, vol. 27, no. 3, pp. 133–137. ISSN 0018-9421. DOI: 10.1109/TIECI.1980.351665.

[16] Microchip's Digital Pure Sine Wave Uninterruptible Power Supply (UPS) Reference Design. In: *Microchip Technology Inc.* [online]. 2016. Available at: http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=Digital-Pure-Sine-Wave-UPS.

About Authors

Radim KUNCICKY was born in Karvina, Czech Republic. He received his master's degree in informatics from VSB–Technical University of Ostrava, in 2014. He is Ph.D. student in informatics. His research interests include embedded systems development and the usage of unconventional computational methods.

Lacezar LICEV was born in Bulgaria in 1953. He is associate professor at VSB–Technical University of Ostrava. His research interests include computer architectures and digital image processing in biomedical applications.

Michal KRUMNIKL received the master's degree and the Ph.D. degree in computer science from VSB–Technical University of Ostrava in 2006 and 2014, respectively. From 2007 he works as an assistant professor at the Department of Computer Science and from 2014 as a junior researcher at IT4Innovations. His research interests include embedded systems and digital image processing - especially the 3D scene reconstruction.

Karolina FEBEROVA was born in Cesky Tesin, Czech Republic. In 2014 she received her Master's degree in biomedical engineering at VSB–Technical University of Ostrava. She is Ph.D. student in informatics. Her research is focused on analysis and processing of biomedical images.

Jakub HENDRYCH was born in Cesky Tesin, Czech Republic. He received his Master's degree in informatics from VSB–Technical University of Ostrava in 2014. Now he is a Ph.D. student in department of computer science. His research interests include image processing, embedded system development and the implementation of unconventional methods.