Design and Implementation of Single-Phase Inverter Based on Lightweight TCP/IP Protocol

Botao Zhang[1, a] and Hongwei Xiao[1, b]

[1]Department of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, 430065, China

[a]zhangbt2000@163.com, [b]850664316@qq.com

Keywords: Lightweight TCP/IP protocol stack; Linux; Embedded; CAN; Single-phase inverter

Abstract. As for resource-limited embedded devices, lightweight TCP/IP protocol (LwIP) is used for communication in order to meet real-time requirements of embedded devices. This paper offered and implemented a single-phase inverter solution based on lightweight TCP/IP protocol. The designed hardware circuit of the master control board is equipped with the AM3352 as its core while the hardware circuit of slave control board uses TMS320F28335 as its core. The master and slave control boards are equipped with CAN communication module, digital and analog gathering module, extended Ethernet, PWM module, RS232/485 and other peripheral interfaces. The master control software completes tasks in the embedded Linux OS including setting the various system parameters and processing and converting data. The slave control software receives data from master control software, controls the single-phase inverter by using the PWM module, gathers the voltage and current changes in the inverter. Finally, the gathered data are fed back to the master control software. Test results show that the system enjoys the advantages of easy manipulation, outstanding real-time performance and stable operation, which can satisfy practical application.

Introduction

With the extensive application and development of embedded technology and network technology in the field of industrial control, more and more embedded devices are required to realize the function of intercommunication with Internet[1]. Concerning the realization of the network interconnection, the key step is to apply TCP/IP protocol stack to the embedded devices. However, as resources are limited in embedded devices, the open source TCP/IP protocol stack is thus applied in order to reduce the memory usage. As for the open source TCP/IP protocol, it is derived from BSD TCP/IP [2], mainly including LwIP, uIP, uC/IP, TinyTcp, etc [3-4]. Lightweight TCP/IP protocol stack (LwIP)[5] is an open source TCP/IP stack for embedded systems and can be transplanted into a variety of embedded OS such as Linux and μCOS. It can also be run in an OS-free bare computer [6].

[Reference 7] proposed the application of LwIP protocol in embedded remote monitoring terminals. That is the transplanting of LwIP protocol into the real-time operating system μCOS-II to realize the communication between the server-side and the gathering and controlling modules. [Reference 8] proposed the application of LwIP protocol in the embedded power monitoring system, namely, transplanting LwIP protocol into embedded OS μCLinux to realize the communication between the power supply system's remote monitoring software and its control module. This paper proposed a solution for controlling single-phase inverter based on LwIP protocol. Firstly, the ARM processor was used as the core and the LwIP protocol was transplanted into the Linux system. Then, the voltage and current data of single-phase inverter collected from the control module was sent, through DSP processor and CAN module, to the ARM processor. Finally, the data were sent by the LwIP protocol to the monitoring client.

Copyright © 2017, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

System Framework Design

The system is mainly composed of the master control module, the slave control module, the single-phase inverter module and the client computer. The system block diagram is shown in Fig. 1.

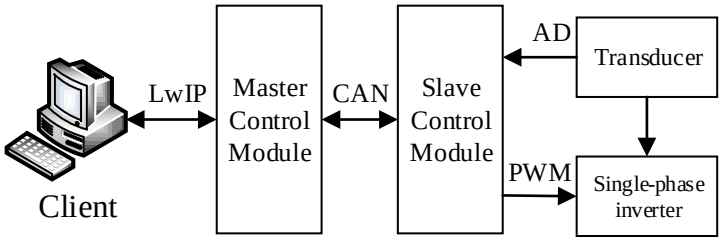

Figure 1. System Block Diagram

The master controller and the slave controller serve as the core parts of the system. Firstly, the controller outputs square waves through PWM module to control the single-phase inverter. Then, detection data from the sensor are received and sent out through the CAN bus. The master control board equipped with Linux system firstly runs the master control software to rece ive CAN data packets from the slave controller. Then, the client receives the data of the main controller through LwIP protocol to fulfill real-time data display and real-time setting of master control board's parameters.

System Hardware Design

The chip of the system's main controller is TI's Cortex-A8 32-bit processor AM3352 with an operating frequency up to 800MHz. It supports Linux OS and is outfitted with plenty peripheral interfaces, featuring high reliability and low power consumption. Master controller's hardware circuit mainly includes such modules as the clock circuit, reset circuit, power supply, RTC module, Ethernet module, CAN module, AD acquisition module, RS485 interface and 128MB DDR2 SDRAM. Its block diagram is shown in Fig. 2.

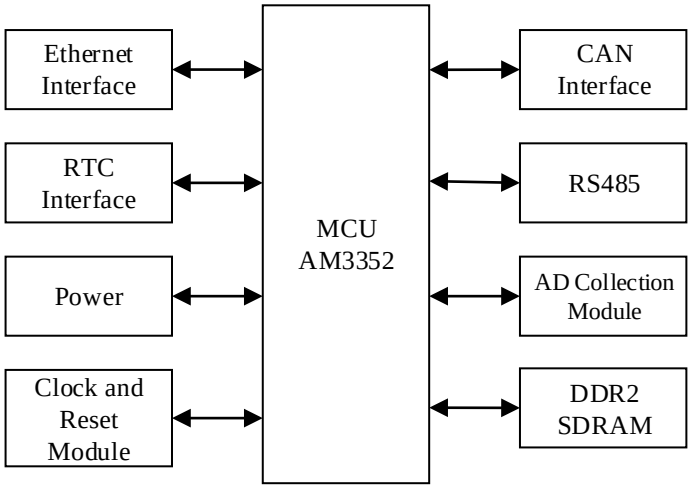

Figure 2. Block Diagram of the Master Controller's Hardware

The slave controller's chip is TI's 32-bit floating-point DSP TMS320F28335 of C2000 series with a frequency up to 150MHz, mainly used in areas of motor control and power equipment control. Slave controller's hardware circuits consist mainly of the clock circuit, reset circuit, CAN module, AD module, power supply, PWM module, and RS232 interface. Its block diagram is shown in Fig. 3.

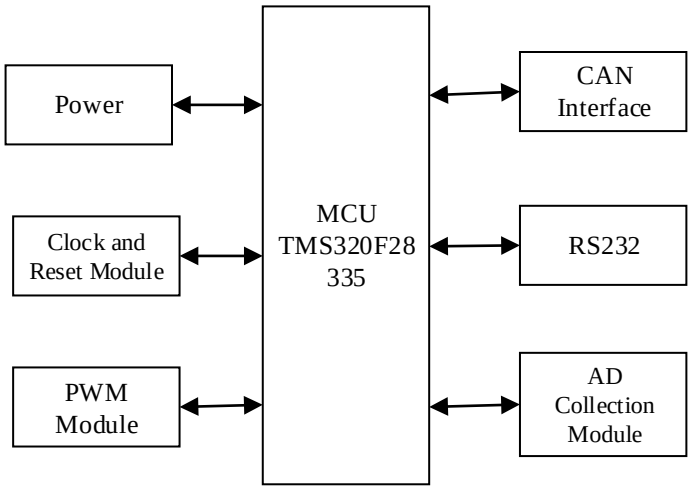

Figure 3. Block Diagram of the Slave Controller's Hardware

Ethernet Module. Ethernet module uses MICREL's Gigabit Ethernet PHY chip KSZ9031RNI. The chip supports 10M/100M/1000Mbps Ethernet transceiver and provides RGMII interfaces with 3.3V and 1.2V power supply. Its network socket is the built-in network transformer HY911130AE. Ethernet module's principle is shown in Fig. 4.

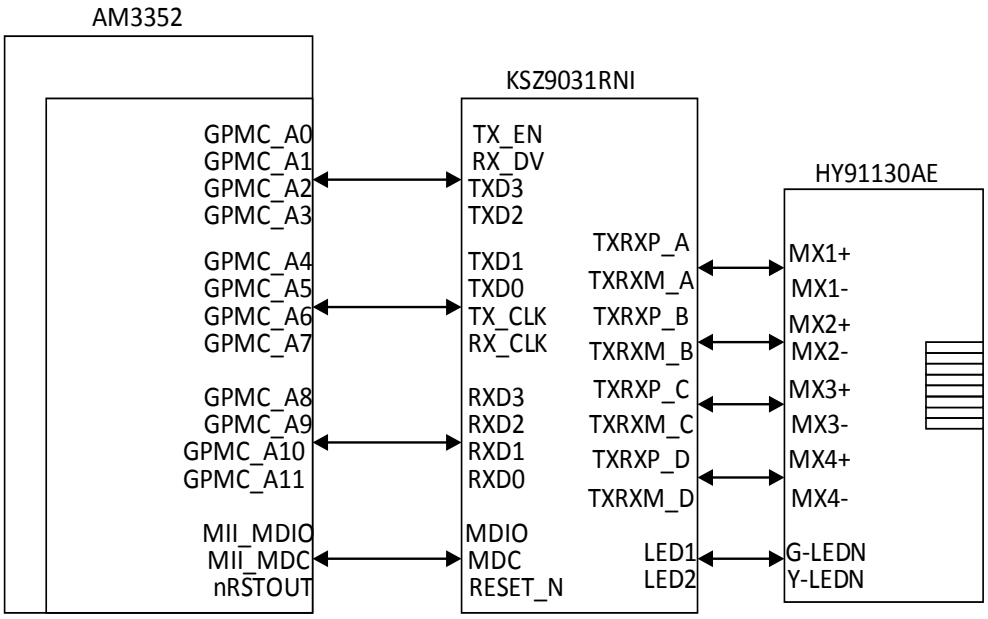

Figure 4. Diagram of Ethernet's Circuit Schematics

System Software Design

As shown in Fig 5, the process is specifically divided into three steps :(1) Transplanting of the LwIP protocol into Linux OS. (2)Development of KSZ9031RNI's device driver. (3)Development of the server-side program of the network's upper port.

Software Platform Transplanting. Before writing the system simulation layer of LwIP, a header file cc.h shall be added to the source code. The cc.h mainly defines environment variables and data type declaration files related to hardware platforms and compilers. Data types used by LwIP are defined as u8_t, s8_t, u16_t, s16_t, u32_t, s32_t and mem_ptr_t. sys_arch.h and sys_arch.c are added to the source code to implement the LwIP simulation layer. In both files, the main fulfillments are the semaphore functions, message queue functions, timeout processing functions and new-created thread functions.

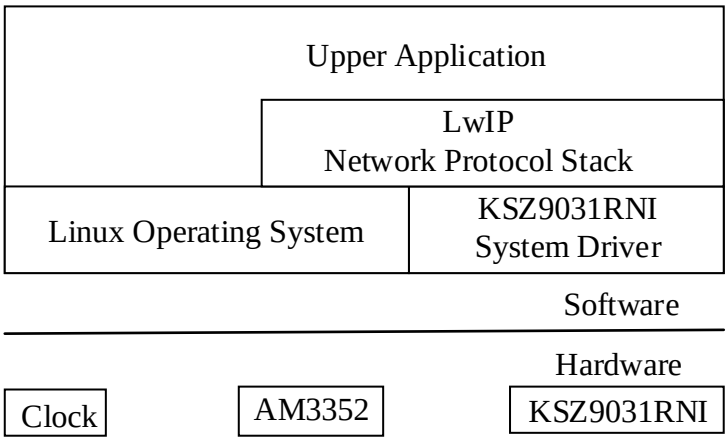

Figure 5 System Software's Architecture

Implementation of Semaphore Manipulation Functions. The Semaphore Manipulation Functions needed to be implemented mainly include sys_sem_new () establish and echo a semaphor -e; sys_sem_signal () send semaphore; sys_sem_free () release semaphore;sys_arch_sem_wait() wait for the specified semaphore and block the thread.

In Linux, semaphore manipulation functions are also provided. Semaphore manipulation functions of Linux are declared in sys/seem. Semaphore manipulation functions of Linux mainly include:(1) the creation function of semaphore set: semget;(2) the initialization and deletion function of semaphore set: semctl ;(3) the release and application function of semaphore set: semop.

The only effort needed is to package Linux semaphore manipulation functions into the corresponding LwIP semaphore manipulation functions.

Mailbox Manipulation Functions. Mailbox is used to deliver messages. The user can implement it as a queue, allowing both multiple messages and single message to be delivered to the mailbox. Both ways can be run properly by LwIP. Message queue manipulation function of Linux can also be used to achieve the mailbox manipulation function of LwIP. The message queue manipulation function of Linux is declared in the sys/msg.h.

Mailbox manipulation functions of LwIP needed to be implemented mainly include:sys_mbox_ new () create an empty mailbox; sys_mbox_free() release a mailbox; sys_mbox_post() post messages to the specified mailbox;sys_arch_mbox_fetch() receive messages in the specified mailbox, this function will block the thread until it times out or the mailbox receives at least one message.

Message queue manipulation functions of Linux are mainly:(1)Message queue creation function: msgget;(2)Message queue deletion function: msgctl;(3)Message delivery function: msgsnd ;(4) Message receiving function: msgrcv.

The same effort is also needed to package the message queue manipulation functions of Linux into the corresponding mailbox manipulation functions of LwIP.

Thread Manipulation Functions. Linux provides a powerful thread manipulation function, declared in pthread.h.

The thread mani pulation function's main task is to create a new thread for sys_thread_new().

sys_thread_new() calls the pthread_create () and the pthread_attr_setschedparam () to create a new thread and to assign their priorities.

Implementation of the Timeout Function. LwIP provides a timeout attribute for each thread connected with the network. After the specified time, the thread will call the sys_arch_timeouts() function which is mainly used to obtain the pointer of the sys_timeouts structure used by the current thread. The sys_arch_timeouts() first determines the priority of the thread and then judges whether this priority is within a reasonable range. If it is, the pointer of the sys_timeouts structure used by the current thread is returned.

Implementation of the KSZ9031RNI Network Interface Card Drivers. The implementation of underlying drivers depends on the network interface card (NIC) chips used by the hardware. Chip

manufacturers offer rich driver functions. For transplantation, you only need to encapsulate these interface functions and encapsulate received data packets into data structures that LwIP protocol stack is familiar with. This paper uses the KSZ9031RNI NIC chip. According to the driver file reference template ethernetif.c given by the LwIP kernel file, the main job of driver transplantation is to implement the ethernetif.c function.In LwIP, each network interface attribute corresponds to one struct netif in which netif includes the attribute, receiving and sending functions, and interruption handling function of that network interface. LwIP calls the netif methods netif->input() and netif->output() to receive and send Ethernet packets.

The drivers work at the network interface layer of the IP protocol model. It provides interface functions such as ethernetif_init(), ethernetif_input(), and ethernetif_output() for the IP layer to complete initializing, receiving, sending, and interruption handling tasks.Fig. 6 shows the driver interface data flow of LwIP.

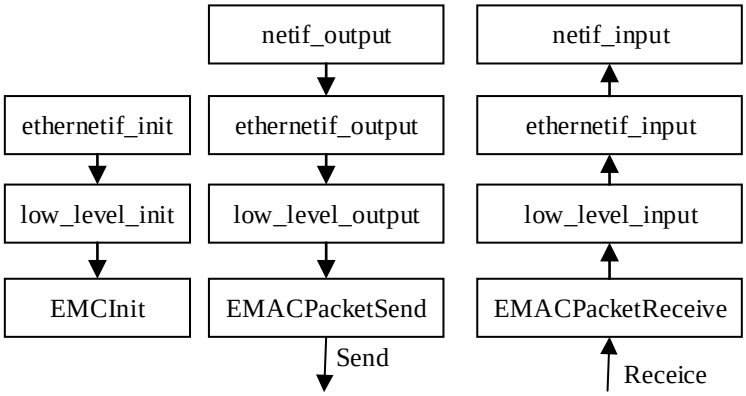

Figure 6. Driver interface data flow of LwIP

Ethernetif_init() is called when the network interface is established. It initializes the underlying interface, sets the function pointer, calls low_level_init(). It also sets the MAC address to the netif-> hwaddr array, sets the MTU value, calls the EMACInit() function, and completes the KSZ9031RNI chip initialization.When a network chip interruption occurs, the system enters into the interruption handler function, sends a semaphore through sys_sem_signal() through the receiving handler thread which, by using the low_level_input() function, calls pbuf_alloc to apply for a buffer of pbuf and calls EMACPacketReceive() to copy the packet data from KSZ9031RNI's receiving buffer to the buffer. Finally, it returns the pointer of pbuf to the ethernetif_input () function which then checks whether the data packet type is ARP or IP. If it is IP, the ARP table is updated and the netif->input () of the driver interface transfers data to the transmission layer. The transmission layer then sends data to the network layer through the netif->output() function.

Development of Upper Server Program. Network upper-layer server program allows a client to be connected with the server through the LwIP protocol stack and receive data sent by the server. The server-side task uses the tcp_new() function to establish a connection, the tcp_bind() function to bind the IP address and server port number, and the tcp_listen() function to monitor and wait for the connection. The tcp_accept() function waits for the semaphore of the connection. Once the connection comes, the function will return the tcppcb structure. The tcp_recv () function obtains the data pointer of the connection and uses tcppcb_data () to point data at the location of the data so that the network data can be processed. After use, the tcppcb_delete () function deletes this data structure and finally the tcp_close() closes this connection.

System Operational Results

The experiment tests the output current I_o, reference voltage U_r, and output voltage U_o at full load and half load.

At full load, the reference voltage and the output voltage collected by the voltage sensor are

listed in Table 1. The output current collected by the current sensor is listed in Table 2. In the experiment, there are 32768 reference values for the reference voltage, the output voltage, and the output current collected by the voltage and current sensors. The tables give the typical data values.

Table 1 Reference values of the reference voltage and actual voltage

	Reference values of voltage				
	1	2	3	4	5
U_r	3900	4000	1000	-2000	-4000
U_o	3855	4080	1090	-2088	-4100

Table 1 lists the collected reference voltage and output voltage data at full load.The voltage waveforms generated by MATLAB are shown in Fig. 7.

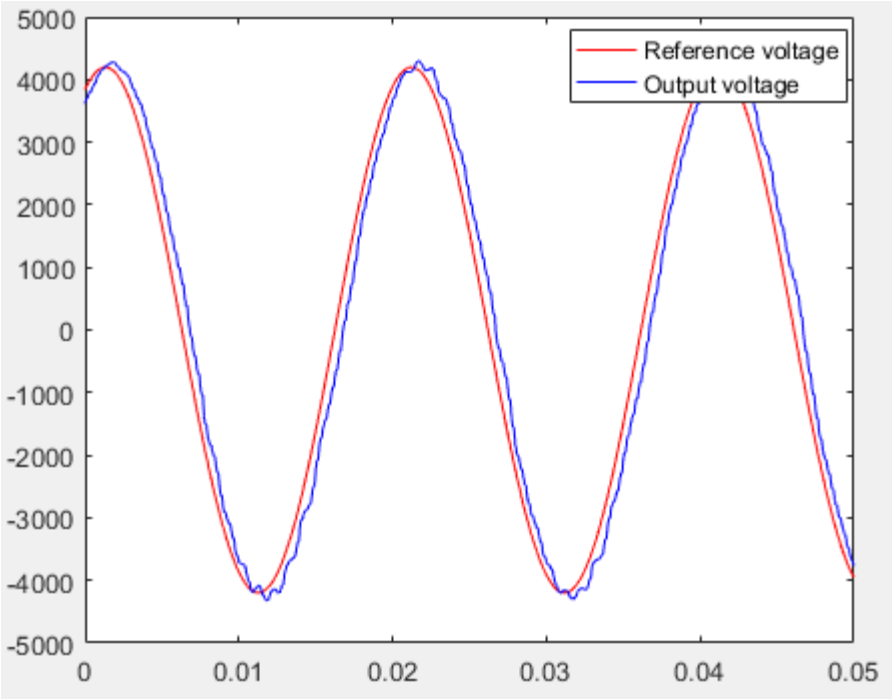

Figure 7. Waveforms of reference voltage and output voltage

Table 2 Actual current reference value

	Reference value of current				
	1	2	3	4	5
I_o	3982	4000	2680	-2980	-4000

Table 2 lists the output current data at full load. The current waveform generated by MATLAB is shown in Fig. 8.

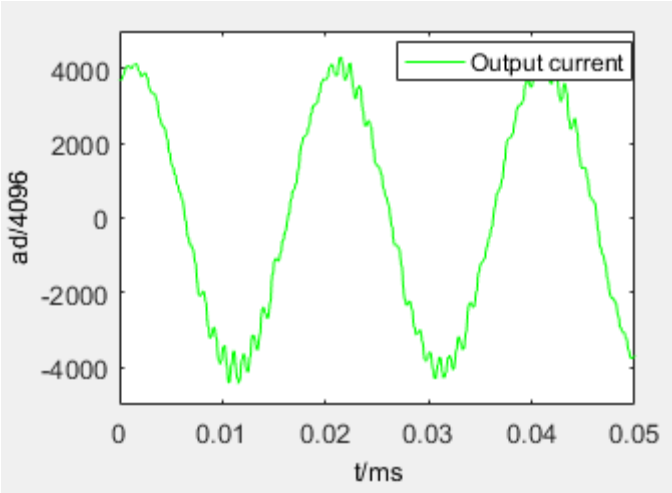

Figure 8. Waveform of output current

At half load, the reference voltage and output voltage collected by the voltage sensor are listed in Table 3 and the output current collected by the current sensor is listed in Table 4.

Table 3 Reference values of the reference voltage and actual voltage

	Reference values of voltage				
	1	2	3	4	5
U_r	-500	-4000	1000	4000	2000
U_o	0	-2988	-4000	1000	4000

Table 3 lists the collected reference voltage and output voltage data at half load. The voltage waveforms generated by MATLAB are shown in Fig. 9.

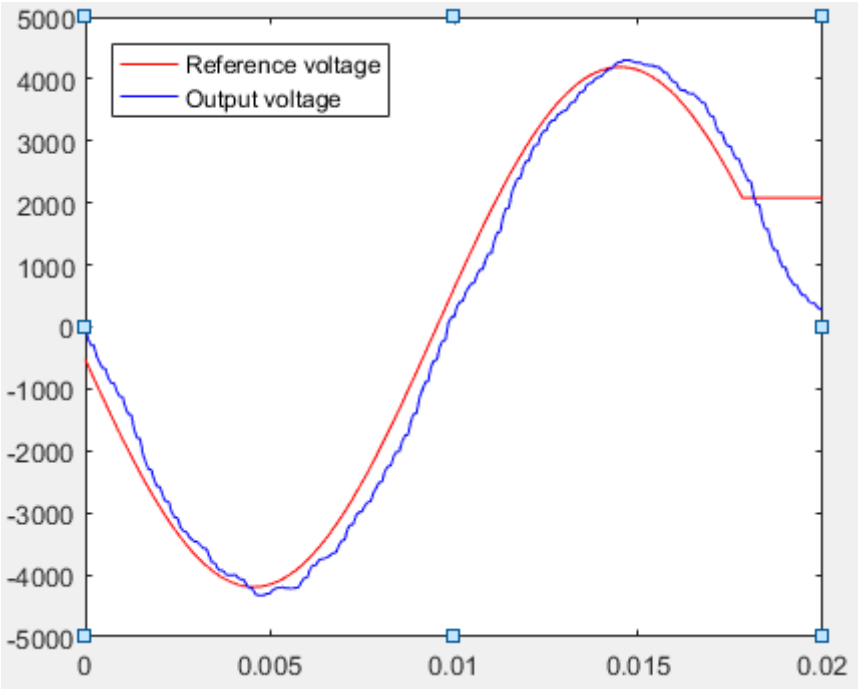

Figure 9. Waveforms of reference voltage and output voltage

Table 4 Actual current reference value

	Reference value of current				
	1	2	3	4	5
I_o	-800	-2000	1800	2000	500

Table 4 lists the output current data at half load. The current waveform generated by MATLAB is

shown in Fig. 10.

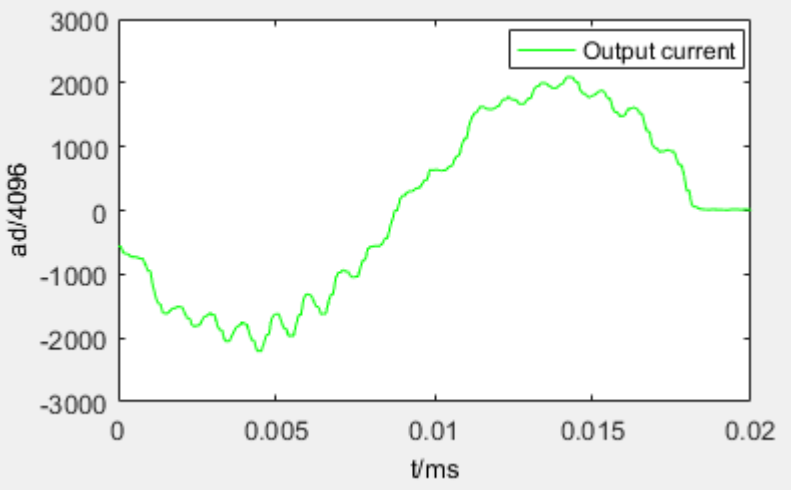

Figure 10. Waveform of output current

Summary

This paper implements control of single-phase inverter using the LwIP protocol stack. The whole system shows good stability in communication and display. However, in terms of data packet transmission, packet loss problem will arise which needs to be further and deeply studied in follow-up work.

References

[1] Jie Zhang, Ming Fu, Transplanting and Implementation of LwIP Protocol Stack in Embedded Linux [J]. Microcomputer Information(2011)94-96.

[2] Chuanxiong Guo, Shaoren Zheng. Analysis of Queuing at IP Layer of TCP/IP Network Protocol in Linux Operating System[J].Chinese Journal of Computers(2001)860-865.

[3] Jianping Wang, Chenfei Zhou, Chenghui Zhu et al. A ZigBee-TCP/IP Seamless Gateway Model[J].Journal of Hefei University of Technology(Natural Science)(2013)1058-1062.

[4] Huabing Chen. Study of Extension and Task Scheduling Algorithms Based on μC/OS-II Module [D] .Wuhan University of Technology(2007).

[5] Deqiang Han, Qishan Yang, Zongxia Wang et al.. Transplanting and implementation of LwIP protocol stack based on μC/OS-III[J]. Electronic Technology Application(2013)18-21.

[6] Zhongshu Yuan, Yang Lu. Analysis and Optimization of Lightweight TCP/IP Protocol Stack Mechanism[J]. Computer Engineering(2015)317-321.

[7] Luoqing Gao, Yuanchang Zhuang. Research and Implementation of embedded Remote Monitoring Terminals Based on LwIP Protocol[J]. Computer Technology and Application (2015)49-51.

[8] Lixia Wang, Design of Embedded Power Source Monitoring Systems based on Ethernet [J].Chinese Journal of Power Sources(2014)973-974.

[9] Guoping Liu, Jiao Sun, Yunbo Zhao. Design Analysis and Real-time Implementation of Networked Predictive Control Systems[J]. Acta Automatica Sinica(2013)1769-1777.

[10] Yun Li, Xiaojuan Zhao, Bo Zhang, Improvement of TCP Performance in Long Term Evolution [J]. Computer Application(2012)3474-3477.