issue №4, December 2016

брой №4, декември 2016г.

CAx technologies journal

CAx технологии списание

ISSN 1314-9628

CAx технологии

CAx technologies

Computer Aided Technologies

- PLM
- CAD/CAM/CAE
- Intelligent Systems (incl. Web Intelligence)
- Modeling and Simulation
- E-Engineering with Internet Applications
- E-Learning
- Digital Humanities
- Business Management Systems
- CAx in Medicine
- Other

CAx Technologies

Publisher: Technical University of Sofia, ISSN 1314-9628
Publisher Address: Bulgaria, Sofia 1000, Kliment Ohridski blvd. №8, Technical University of Sofia

CAx Technologies, issue №4, December 2016, ISSN 1314-9628

CONTENTS

Using Arduino for prototyping an alarm system

Aleksandar Hristov
Technical University of Sofia, Sofia, Bulgaria
E-mail: ahristov@tu-sofia.bg

Veronika Vakadinova
Technical University of Sofia, Sofia, Bulgaria
E-mail: vvakadinova@tu-sofia.bg

Trifon Trifonov
Technical University of Sofia, Sofia, Bulgaria
E-mail: trifonov_UKTC@yahoo.com

Abstract. *Arduino is open source hardware, which is its biggest advantage. It is possible to use the Arduino for designing a variety of devices, ranging from simple thermostat to aircrafts.*
The purpose of this paper is to present the possibilities of using Arduino for prototyping an alarm system for cottages or cars. In the present paper some of Arduino boards are analyzed. Based on this analysis is determined the specific Arduino board which is used for designing and prototyping an alarm system for cottage. Also, in this paper are proposed microcontroller program in C language and structure of prototype of alarm system.

Keywords: *Arduino, Alarm System, C Programming*

1. INTRODUCTION

Arduino is open source hardware [1],[6], which is its biggest advantage. On the official website you can download Integrated Development Environment- IDE for Windows, Linux, or Mac OS [2]. There are a text editor, compiler program, samples of model programs and a set of libraries for solving common tasks. When you run IDE, it opens window, in which you can type your program (into C code), compile it, download this program (into BIN code) to Arduino microcontroller, and launch the program for execution. Although, Arduino has been divided in two independent companies- Arduino LLC and Arduino SRL, they continue to manufacture products under one brand name, which creates confusion. The IDE software is updated regularly. In addition to IDE provided by official developers, there are many other development software, e.g. [2],[12],[13]. One of the advantages of Arduino is a wide set of additional cards, called shields. Using a lot of already made libraries, these shields allow the Arduino board to connect to computer networks via Ethernet and Wi-Fi, even to transmit data trough GSM cellular network, as well as to work with sound, powerful drives for robots, etc. Naturally, it is possible to use the Arduino for designing a variety of devices [7],[8],[9],[10],[11], ranging from simple thermostat to aircrafts. Even Russian aerospace company Lin Industrial [4] uses block for registration of flight parameters (based on Arduino) in design of its experimental rockets. Scientists from Institute of science education and research in Trivandrum (India) used Arduino in the educational system [5].

According to Wikipedia, an alarm device or system of alarm devices gives an audible, visual or other form of alarm signal about a problem or condition [3], e.g. burglar or fire. With any kind of alarm, it is necessary to balance between, on the one hand, the danger of false alarms (called "false positives") — the signal going off in the absence of a problem — and, on the other hand, failing to signal an actual problem (called a "false negative"). Alarm devices are often outfitted with a siren. Some burglar alarms are silent alarms, thus the police or guards are warned without any indications to the burglar, which increases the chances of catching him or her. However, in not so responsible cases, such burglar alarms have to be used with extreme caution, due to possibilities of false alarms which can waste resources and/or money (fines).

There are requirements for the alarm systems of public buildings to be designed by certified designers and companies. For home alarm systems there are no such requirements and they can be designed with components that can be bought e.g. Paradox components. For cheap defend of properties and havings can be used handmade alarm systems (naturally, the price of the alarm system should not be higher than the value of the belongings).

The purpose of this paper is to present the possibilities of using Arduino for prototyping an alarm system for cottages or cars.

2. Development of alarm system based on Arduino

The Arduino UNO board is based on ATmega328P microcontroller which has 2 Kb of RAM and 32 Kb program memory. This microcontroller works on 16 Mhz clock frequency stabilized by Quartz resonator. Most versions of the Arduino boards are connected to computer through a micro-USB connector. Connected to the computer, Arduino board is powered from the USB port. In order to work without a computer, 7-12 V external power source is needed. There is a special connector on the board with a built-in surge protector. Arduino UNO does not impose special requirements on the power supply voltage. So, almost any compact power supply, which output voltage is in the required interval can be used, even 9V battery, e. g. 6F22. For communication with external devices, such as data from sensors, there are 14 digital input-output lines, called D0-D13. On six of these lines microcontroller can output PWM pulses. There are six analogue input lines A0-A5, which are connected to microcontroller ADC converter. Three LEDs are installed: power indicator LED (ON) on the Arduino UNO board, which is permanently connected to the digital I/O line (L), and two LEDs, which indicate the exchange of information with an external device via the serial port (TX and RX) as well as a microcontroller reset button. The Arduino UNO board is well suited for developments, building designs, and debugging programs. If the parameters/characteristics of Arduino UNO are not enough for building application, usage of Arduino Mega is needed. It has more memory and number of input-output lines. For other practical applications the parameters and/or the size of Arduino UNO board may be too large for installation in the finished product, so it can be replaced with Arduino Nano or Arduino Mini. Both of them have almost identical parameters to Arduino UNO, but have a simplified design, smaller dimensions and they are cheaper than Arduino UNO. This is the reason why to choose the small, compact and battery supplied Arduino Mini board, used for designing and prototyping alarm systems.

During Arduino installation process the folder with IDE software (including text editor, compiler program, and a set of libraries for solving common tasks) is placed in the root folder of hard drive. Connecting for first time the Arduino board with the USB port of the computer, the operating system will detect a new device and require driver installation. After installing it and restarting computer, as a result of successful installation in the Windows Device Manager will be displayed an additional COM port. As mentioned above when IDE runs it opens window, in which our program is typed (fig. 1).

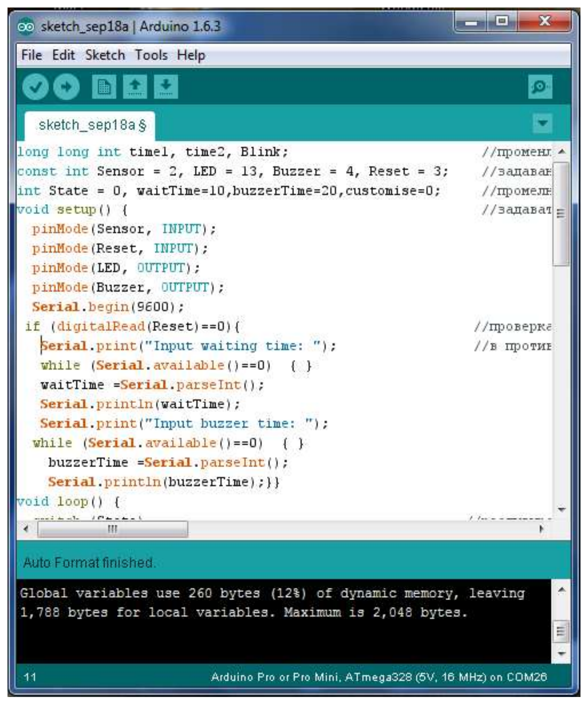

Fig. 1 ***Screenshot of Arduino IDE***

A program written in a low-level language (assembler) would take less space on the memory and would work very fast.

Using high-level language simplifies and greatly reduces skill requirements of the programmer, but on the other hand, the resulting program will not be optimal for memory usage and speed. When developing and prototyping simple devices as an alarm system, this disadvantage would be neglected.

As one can see in the appendix at the end of this paper the main program (created in C) is a finite state machine- FSM (fig. 2).

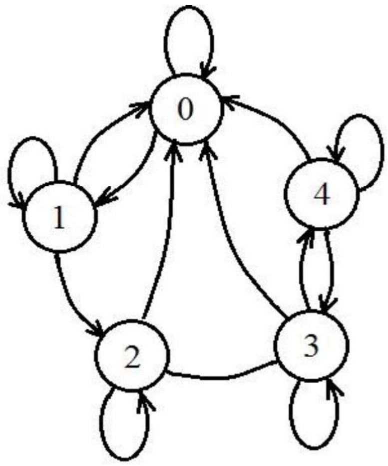

Fig. 2 ***Graph of states of the alarm system***

After the board has been powered, FSM goes to state 0, in which FSM stays for specific time. The purpose of this is to give the owner enough time to leave the cottage. After this delay if there is a signal form the sensor, FSM goes to state 1. In this state the LED switches ON to indicate that the alarm system has been activated. FSM stays in state 1 for a specific time, in which the owner would be able to reset the alarm system if he or she enters the cottage. When the waiting time passes the FSM goes to state 2. In this state the buzzer starts signalizing that there is a burglar. After this the FSM goes to state 3, in which the buzzer stops but the LED indication is still turned ON. In state 4 the LED indication turns OFF. Then the FSM switches between states 3 and 4 which is an indication that a burglar has entered the cottage. In all states FSM could be sent to state 0 by turning the switch ON, i.e. the alarm system could be restarted by the owner.

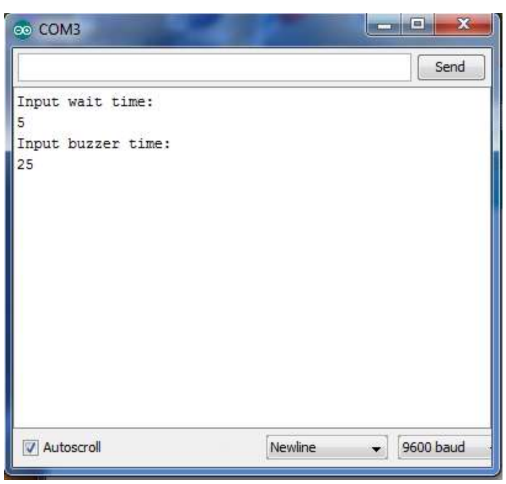

Fig. 3 ***Customization of the alarm system***

There is a special program, called boot loader, installed on Arduino board, which helps you to record the compiled (in PC or laptop) user program in the program memory of the microcontroller. Boot loader itself occupies some part of this memory from one to four kilobytes, depending on the version. Thanks to this organization of interaction with computer (PC or laptop) the user can customize the time before alarm system starts signalizing (by default 10 seconds) and specific time for signalizing (by default 20 seconds). Screenshot from the Serial Monitor (part of IDE) for the customization is shown on figure 3.

Below there is an explanation of the code shown in the Appendix. At the first line three variables are declared:

- time1 - counts the time before the activation of the alarm;
- time2 - the time before starting signalization (beeping);
- blink – the flashing period of the LED indication.

At the second line the pins of Arduino board are declared for the sensor, the LED and the buzzer.

On next line one can see variables for the condition of the finite state machine, the waiting time, the time for which the buzzer is on and if the settings would be customized.

At lines 5 to 8 it is assigned the operating modes of the microcontroller pins, where the sensor, the reset switch and the buzzer are connected.

The function pinMode()configures the specified pin to behave either as an input or an output.

At line 9 the data rate is set in bits per second for serial data transmission.

At line 10 there is a check if the reset switch is ON while starting the security system and if so, the waiting time and the time for which the buzzer is on are configured (lines 11 - 18), through the serial interface of the microcontroller (fig. 3). Otherwise, the alarm system will be working with default values.

On several lines Serial.print() and Serial.println() are used to print data to the serial port as human-readable ASCII text.

At line 13, parseInt() function is used, which returns the first valid (all characters that are not digits or a minus signs are skipped)integer number from the serial port.

Between line 22 and the last one of the program one can see switch…case construction, which corresponds to FSM. There are 5 cases (0-4) and in every one there is different code for execution.

In the first case (case 0, or lines 23-31) the alarm system is waiting until the activation of the sensor (there is a burglar) and we move to case 1.
The function delay() pauses the program for amount of time (in milliseconds).
At line 27 there is a check if the sensor has high value and if it does so, the variable time1 is set with value in milliseconds since the begging of the current program (it counts the time before the activation of the alarm).
After that we go to state 1.
Case 1 (lines 32 - 40) turns the LED indicator on by digitalWrite() function and wait awhile the security system to be turned off (when the intruder is the owner, for example). If reset switch is on, FSM goes back to state 0. Else there is a check if there is more time to wait and if it does not have, FSM goes to state 2 and time2 variable is set with returned value of the function millis().
In case 2, the same check as above is done. If the system was not turned off, the alarm is activated (case 2, lines 42 - 47) for the definite time. Otherwise, if the signalizing time has already passed, FSM goes to state 3.
In case 3, first the buzzer stops. After that again the reset switch is checked. If the switch is on, FSM returns to state 0. Apart from that, the LED is on and after the predefined flashing period (in Blink variable), FSM goes to state 4.
In case 4 the LED is turned off for the flashing period, then FSM goes to state 3 i.e. LED repeatedly turns the on and off.

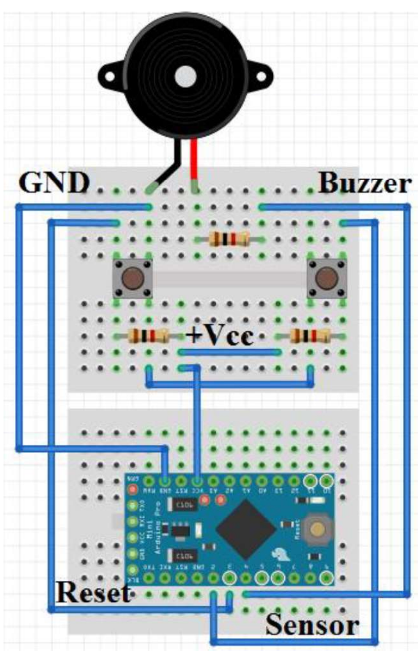

Fig. 4 ***Prototype of alarm system***

On figure 4 it is shown the prototype of alarm system using Arduino breadboard. Note, that sensor is emulated by button on the right side of the breadboard and reset switch by the left one. Each button has four pins as each vertical pair of pins is connected (conducted). When a button is pressed its left and right pins will be connected, i.e. there will be logical 0 on corresponding input port. If a button is not pressed there will be logical 1 from Vcc through corresponding (left or right) resistor. In order to reduce loudness of buzzer, between it and the microcontroller there is a serial connected 100 ohm resistor.

3. CONCLUSION

In present paper some Arduino boards are analyzed. Based on this analysis is determined the specific Arduino board which is used for designing and prototyping an alarm system for cottage. Also, microcontroller program in C language and structure of the alarm system prototyped on breadboard are proposed.
As future work, the alarm system will be placed in cottage, situated for example in Simeonovo (near Sofia), and some improvements could be made, such as warnings about burglar on a specific GSM phone number.

REFERENCES:

1. Arduino Site - URL: https://www.arduino.cc (23.09.16).
2. Arduino Software - URL: http://www. arduino.org/software (23.09.16).
3. https://en.wikipedia.org/ (23.09.16).
4. Lin Industrial official site - URL: http://spacelin.ru
5. Mandal P. et al. Everyday Radio Telescope - URL: http://arxiv.org/abs/1601. 02982 (23.09.16).
6. Arduino Single-Sided Serial Board (version 3) - URL: https://www.arduino.cc/en/Main/ArduinoBoardSerialSingleSided3 (23.09.16).
7. Ivanov A., Ivanov I., and V. Hristov, Street light control system with a microcontroller , Proc. of "Electronics'2004", 22-24 September, Sozopol, Bulgaria, book 4, pp. 103-107
8. Nikolova, I, Zapryanov, G., Manoilov, P., Kucidimova, E., „FPGA-based Architecture for Digital Image Visualization", In Proceedings of the 5th International Conference Computer Science'2009, November 5-6, Sofia, Bulgaria
9. Nikolova, I., Nikolov, H., Manoilov, P., Zaprynov, G. (2009), "FPGA-based Architectures for Digital Image Processing and Visualization", Journal on "Computer & Communication Engineering", Year III, №1/ 2009, ISSN 1313-2717.
10. Oxer J., Blemings H, Practical Arduino: Cool Projects for Open Source Hardware, Apress, 2009, ISBN: 978-1-4302-2477-8.
11. Taneva L., V. Hristov, Development a system for treatment and control of posttraumatic stress, based on MCU MSP430F149, Bulgarian Journal for Engineering Design, No 16,October 2012, pp.31-35
12. Site of "Robocraft" - URL: http:// robocraft.ru (23.09.16).
13. Глушенко С. FLProg - система визуального программирования для Arduino - Радио, 2015, № 10, с. 27-30.

APPENDIX

```
long long int time1, time2, Blink;
const int Sensor = 2, LED = 13, Buzzer = 4, Reset = 3;
int State = 0, waitTime=10,buzzerTime=20,customise=0;
void setup() {
    pinMode(Sensor, INPUT);
    pinMode(Reset, INPUT);
    pinMode(LED, OUTPUT);
    pinMode(Buzzer, OUTPUT);
    Serial.begin(9600);
    if (digitalRead(Reset)==0){
      Serial.print("Input waiting time: ");
      while (Serial.available()==0)  { }
        waitTime =Serial.parseInt();
        Serial.println(waitTime);
        Serial.print("Input buzzer time: ");
      while (Serial.available()==0)  { }
      buzzerTime =Serial.parseInt();
      Serial.println(buzzerTime);
    }
  }
  void loop() {
    switch (State){
    case 0:
      delay(10000);
        digitalWrite(LED,0);
      digitalWrite(Buzzer,0);
      if (digitalRead(Sensor)) {
          time1 = millis();
          State = 1;
      }
      break;
    case 1:
      digitalWrite(LED, 1);
      if (digitalRead(Reset) == 0)
          State = 0;
      if (millis() - time1 > waitTime*1000){
        State = 2;
          time2 = millis();
      }
      break;
      case 2:
      digitalWrite(Buzzer, 1);
      if (digitalRead(Reset) == 0)
         State = 0;
      if (millis() - time2 > buzzerTime*1000)
         State = 3;
      break;
    case 3:
      digitalWrite(Buzzer, 0);
      if (digitalRead(Reset) == 0)
        State = 0;
      if (millis() - Blink > 500){
          digitalWrite(LED, 1);
            State = 4;
          Blink = millis();
      }
      break;
    case 4:
      if (digitalRead(Reset) == 0)
          State = 0;
      if (millis() - Blink > 500){

          digitalWrite(LED, 0);
          State = 3;
          Blink = millis();
      }
    }
  }
```