Journal of Communication and Computer 13 (2016) 11-18
doi:10.17265/1548-7709/2016.01.002

An Introductory Embedded Systems Teaching Using Open-Source Hardware and Software Platforms

Peter Yakimov

Department of Electronics, Faculty of Electronic Engineering and Technologies, Technical University of Sofia, Sofia 1000, Bulgaria

Abstract: Embedded systems are used everywhere in human lives. They are in alarm clocks, automobiles, mobile phones, personal digital assistants, home appliances and etc. The Faculty of Electronic Engineering and Technologies at Technical University of Sofia decided after many iterations and discussions with the partners companies and employers organizations to accept a new curriculum for Bachelor degree in Electronics. In order to prepare the students for the challenges of their future job in the new curriculum it was emphasized on the practice and especially on obtaining skills in embedded systems programming. The laboratory classes in "Practice on open source platforms programming" are intended to give the students basic knowledge and skills in this field. The open source hardware and software platforms—Arduino development board and the software IDE, which are globally recognized as tools for introductory education, are chosen for this purpose. In this paper it is proposed a practical approach for teaching the basics of embedded systems hardware and software. The stress is on the program control of inputs and outputs to convince the students in the flexibility and universality of the programmable devices. Also there are introduced the principles of the sensors and their application. Some examples are presented.

Key words: Embedded systems, open-source platforms, Arduino, education.

1. Introduction

Embedded systems are used everywhere in humans lives. In recent years, they are becoming increasingly more important due to their widespread utilization in every aspect of people's lives [1]. Learning to design and program embedded systems is a critical skill that is necessary for many industry and scientific jobs [2]. At the level of today's technology consumer, there appears to be an increasing desire to interface the technological power-machines to the real physical world. This desire to connect may have been always present, but there appears to be more of a push towards closing the gaps between human and technology, by leveraging technology in a more personal, private and autonomous manner, under control of the user [3]. The Faculty of Electronic Engineering and Technologies at Technical University of Sofia accepted a new curriculum for Bachelor degree in Electronics in order to prepare the students for the challenges of their future job. The curriculum was created after many iterations and discussions with the partners companies and employers organisations. It was intended to give the students theoretical knowledge about basic electronic circuits and devices and practical skills for their program control. After investigating a lot of curricula from many leading universities it was accepted that the practical training has to begin in the first semester in order to prepare the students for the specialized courses in the next years. For this purpose a brand new course entitled "Practice on open source platforms programming" with two hours laboratory work per week was included in the first semester. As a development environment was chosen the Arduino platform which is open-source hardware, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware platform is a simple open source design for the Arduino board with an Atmel AVR processor and on-board I/O support.

Corresponding author: Peter Yakimov, Dr. Eng., assoc. prof., research fields: embedded systems in measurement and control.

The software consists of a standard programming language and the boot loader that runs on the board. Arduino hardware is programmed using a Wiring-based language (syntax + libraries), similar to C++ with some simplifications and modifications, and a Processing-based IDE [4]. Another advantages of Arduino are that it can run on Windows, Macintosh and Linux, and the active community of users who permanently contribute with new code examples for the enlargement of the educational possibilities [5]. The open source hardware and software platforms — Arduino development board and the software IDE are globally recognized as effective tools for introductory education in embedded systems.

The course has been performed for two academic years and some considerations about its advantages can be shared.

2. New Course Development

According to the philosophy of the new curriculum a major part of the course is based on the usage of Arduino. The development board OLIMEXINO-328 based on the microcontroller ATmega328P is the hardware [4]. The aim of the course is to help the students in understanding the relationship between computer devices and the surrounding world with appropriate examples. The experimental work includes an application software design and debugging, and measuring the response of controlled peripheral circuits. Thus, students acquire practical skills and obtain knowledge about basic electronic circuits and devices, and the possibilities for their program control. During the laboratory work experiments on development boards are carried out. Initially the given task is analysed and the algorithm is drawn. Then a program is written and run. Thus the students individually find possible errors and after analysis the results conclusions are made and mistakes are corrected. The topics are directly related to the field of the next courses.

Except the development board the necessary equipment is traditional for related to embedded systems courses and includes a personal computer with an installed IDE (integrated development environment), a breadboard, some simple components as resistors and LEDs, and USB cable.

The on-board user buttons and LEDs are used too as it is shown in Figs. 1 and 2 [4]. They are enough for introductory studying of the basic operations for control of digital inputs and outputs. To use them, the students have to know preliminary that the digital input reads logic “0” from the pressed button, and to light the LED a high level must be set from the digital output.

To study the operation of the analog inputs and outputs some additional components are needed. To set the analog input voltage a potentiometer has to be connected to the chosen analog input from Analog 0 to Analog 5 (pins from 23 to 28). Digital outputs with numbers 3, 5, 6, 9, 10 and 11 can be used as analog

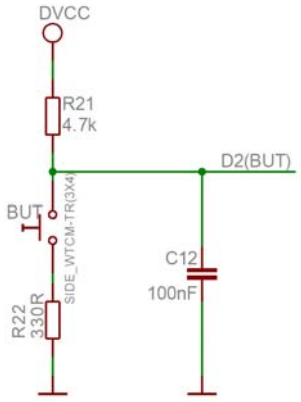

Fig. 1 User button with name BUT connected to ATmega328P pin 32 (digital signal D2).

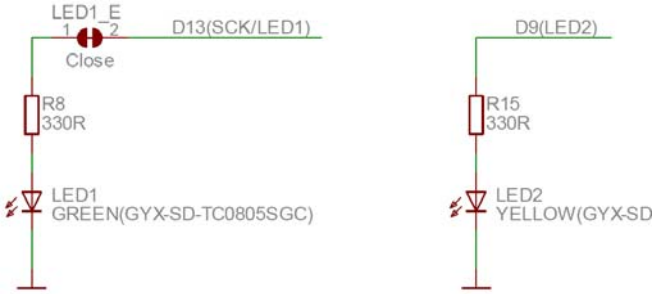

Fig. 2 Light emitting diodes connected to ATmega328P pins 17 (LED1—digital signal D13) and 13 (LED2 —digital signal D9).

outputs because of their ability to set a pseudo analog voltage using PWM (pulse width modulation). To observe the response of the analog outputs a digital multimeter is necessary. Also off-board LEDs are used for this purpose. This gives the students an additional knowledge—to understand the analog operation of the LED and the principle of colours mixing.

The laboratory set-up is shown in Fig. 3. The analog inputs A0, A1 and A2 are connected to the wipers of three potentiometers with 10k value. The voltages that are derived are set by the reference voltage of the built-in analog-to-digital converter in order to be stable and also that the maximal code will correspond to the maximal value of the input voltage. Outputs D3, D5 and D6 are connected to the cathodes of the RGB-LED with common anode organisation. This laboratory set-up is used for studying the basics of the analog-to-digital and the digital-to-analog conversion. Also using the RGB-LED the students will understand the difference between the digital and the analog control of the brightness. They will realise the principle of colours mixing too.

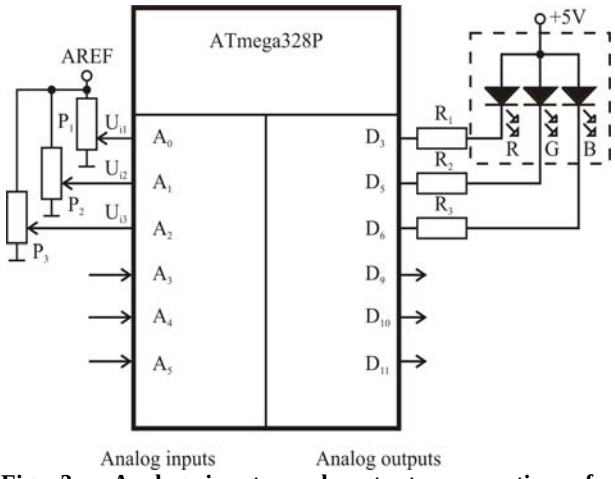

Fig. 3 Analog inputs and outputs connections for RGB-LED control.

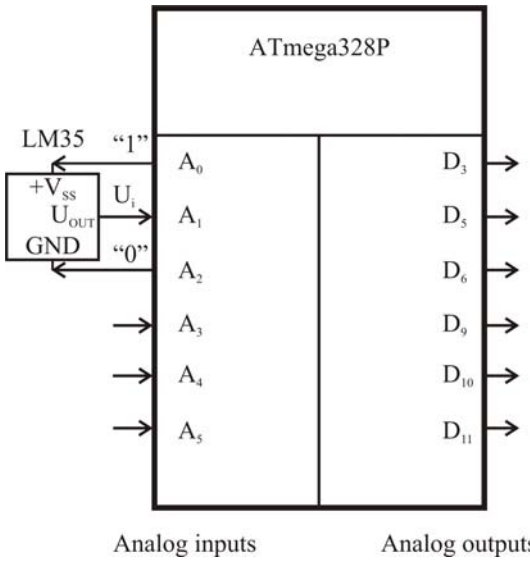

Fig. 4 Analog inputs connections for attachment of a temperature sensor LM35.

A very important part of the electronics is the control of sensors. They convert the physical quantities into electric signals—voltage, current and etc. Sensors circuits mastering gives the students useful skills for their future job.

The laboratory set-up is depicted in Fig. 4. It is intended for temperature measurement using the sensor LM35. Its transfer characteristic is linear with a slope of 10 mV/°C.

To simplify the laboratory set-up is used the specific feature of the sensor—i.e., its very little current drain—less than 60 μA. This allows the sensor to be supplied from two pins (A0 and A2) determined as outputs and set logic 1 from A0 and logic 0 from A2. The output pin of the sensor is connected to the analog input A1.

3. Experiments

Here will be given some examples from the laboratory work that represent the practical approach

for introductory teaching in embedded systems programming. Arduino development board and a software IDE are used. The goal is that using simple tasks the students will master the basic statements of C language which is the most often used language in the field of embedded systems. Also the students have to learn the structure of the program and the possible approaches in the software design. The exercises start with very easy projects where the result is obvious and the complexity slightly increases with the progress of the course. Arduino IDE gives a lot of examples of

code that can be used for preparation and self study. Learning the examples from the preloaded in the IDE the students study the basic statements of the programming language for control of digital inputs and outputs. After understanding the given programmes the students have to modify the code in order to master the knowledge. The next code examples which will be presented here are designed by the teaching team and they can be used in the process of learning.

3.1 Digital Inputs and Outputs Control

The first task is to control the alternative blinking of the two LEDs on the development boards shown in Fig. 2. It is intended to study the statements for digital outputs control. The code is as follows:

```
int GreenLed = 13;//green LED is connected to D13
int YellowLed = 9;//yellow LED is connected to D9
void setup()
{
pinMode(13, OUTPUT); // D13 is set as output
pinMode(9, OUTPUT); // D9 is set as output
}
void loop()
{
digitalWrite(GreenLed, HIGH); // LED1 is on
digitalWrite(YellowLed, LOW); // LED2 is off
delay(1000); // delay one second
digitalWrite(GreenLed, LOW); // LED1 is off
digitalWrite(YellowLed, HIGH); // LED2 is on
delay(1000); // delay one second
}
```

The next step in programming is to study the statements for reading the state of a digital input and making decisions according to it using the statement *if…else*. The following example of code illustrates this:

```
int Button = 2; // button is connected to pin D2
int GreenLed = 13;
int YellowLed = 9;
int State = 0; // variable for the read value
void setup() {
pinMode(GreenLed, OUTPUT);
pinMode(YellowLed, OUTPUT);
pinMode(Button, INPUT); // D2 is set as input
}
void loop() {
// read the state of the pushbutton value:
State = digitalRead(Button);
// check if the pushbutton is pressed
// if it is, the State is LOW:
if (State == LOW) {
// turn LED1 on and LED2 off:
digitalWrite(GreenLed, HIGH);
digitalWrite(YellowLed, LOW);
}
else {
// turn LED1 off and LED2 on:
digitalWrite(GreenLed, LOW);
digitalWrite(YellowLed, HIGH);
}
}
```

3.2 Analog Inputs and Outputs Control

The built-in analog-to-digital converter has 10 bits resolution. So the code returned after the conversion is in the range from 0 to 1023. The digital-to-analog conversion equals effectively to 8 bits resolution. Then to match their ranges the input value has to be divided by 4. To control analog inputs and outputs the students have to learn the proper statements. They are similar to those used for the digital ones. The following example contains these statements. The program reads the value of an analog input A0 and using this value controls the brightness of a LED connected to D9 which can be used as an analog output too.

```
int YellowLed = 9;
int AnalogInput = 0; // potentiometer attached to A0
int Value; // Analog value
void setup () {} // there is no need of setup
void loop ()
```

```
{
// reading the analog value
Value = analogRead (AnalogInput);
// correspondence between input and output ranges
Value /= 4;
//setting analog output value
analogWrite (YellowLed, Value);
}
```

The above program can be made slightly harder if the task is to control alternatively the brightness of two LEDs according to the analog input value set by a potentiometer. This will be useful for the students to master the skills for analog inputs and outputs control. The code example is:

```
int LED1 = 3; // LED1 connected to A3
int LED2 = 5; // LED2 connected to A5
int AnalogInput = 0; // potentiometer attached to A0
int Value; // Analog value
void setup () {} // there is no need of setup
void loop ()
{
Value = analogRead (AnalogInput);
Value = Value / 4;
analogWrite (LED1, Value); // LED1 control
analogWrite (LED2, 255 - Value); // LED2 control
}
```

3.3 RGB LED Control

To attract the attention of the students and to give them an understanding about the modern information technologies it is useful to show them the basic principles of colours mixing. For this purpose a RGB LED can be used as it is shown in Fig. 3. The organisation of the LED is with common anode. Thus the needed level to turn on the chosen LED is low. The three built-in LEDs can be controlled using digital or analog outputs. The students will recognise the difference and will understand the two level digital control and the multilevel analog control. The following code example illustrates the colours mixing using digital control.

```
int RedLed = 3;
int GreenLed = 5;
int BlueLed = 6;
int Button = 2;
int buttonState = 0;
int state = 0;
void setup()
{
pinMode(RedLed, OUTPUT);
pinMode(GreenLed, OUTPUT);
pinMode(BlueLed, OUTPUT);
pinMode(Button, INPUT);
//initially black colour
digitalWrite(RedLed, HIGH);
digitalWrite(GreenLed, HIGH);
digitalWrite(BlueLed, HIGH);
}
void loop()
{
buttonState = digitalRead(Button);
if (buttonState == LOW)
{
if (state == 0)
{
digitalWrite(RedLed, LOW); //red colour
digitalWrite(GreenLed, HIGH);
digitalWrite(BlueLed, HIGH);
state = 1;
}
else if (state == 1)
{
digitalWrite(RedLed, HIGH);
digitalWrite(GreenLed, LOW); //green colour
digitalWrite(BlueLed, HIGH);
state = 2;
}
else if (state == 2)
{
digitalWrite(RedLed, HIGH);
digitalWrite(GreenLed, HIGH);
digitalWrite(BlueLed, LOW); //blue colour
```

```
state = 3;
}
else if (state == 3)
{
digitalWrite(RedLed, LOW); //yellow colour
digitalWrite(GreenLed, LOW);
digitalWrite(BlueLed, HIGH);
state = 4;
}
else if (state == 4)
{
digitalWrite(RedLed, LOW); //magenta colour
digitalWrite(GreenLed, HIGH);
digitalWrite(BlueLed, LOW);
state = 5;
}
else if (state == 5)
{
digitalWrite(RedLed, HIGH); //cyan colour
digitalWrite(GreenLed, LOW);
digitalWrite(BlueLed, LOW);
state = 6;
}
else if (state == 6)
{
digitalWrite(RedLed, LOW); //white colour
digitalWrite(GreenLed, LOW);
digitalWrite(BlueLed, LOW);
state = 7;
}
else if (state == 7)
{
digitalWrite(RedLed, HIGH); //black colour
digitalWrite(GreenLed, HIGH);
digitalWrite(BlueLed, HIGH);
state = 0;
}
delay(1000);
}
}
```

The possible colours are red, green, blue, yellow, magenta, cyan, white and black. They are changed after pressing the button connected to D2. The mixing of the colours is shown in Fig. 5.

To apply analog control for the colours mixing every LED from the RGB must be controlled with output voltage that can be set within the range 0-255 levels as it is shown in Fig. 6.

The following code example illustrates the control. The laboratory set-up is shown in Fig. 3. This program also shows the possibilities of the serial monitor and the command *Serial.print*. It is used to visualize on the display strings and values. In this case are visualized the values for the red, green and blue colours and they can be compared with those from the RGB colour wheel which is depicted in Fig. 6.

```
int RedLed = 3;
int GreenLed = 5;
int BlueLed = 6;
```

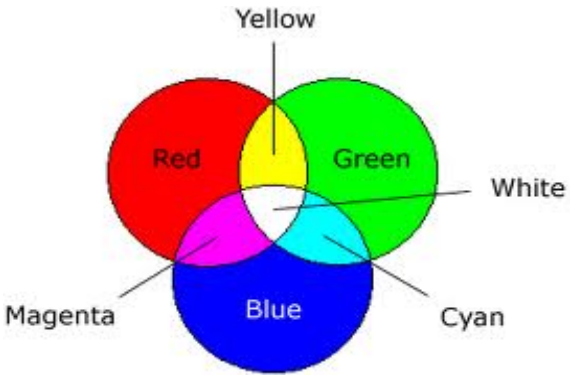

Fig. 5 Colours mixing using RGB-LED with digital control.

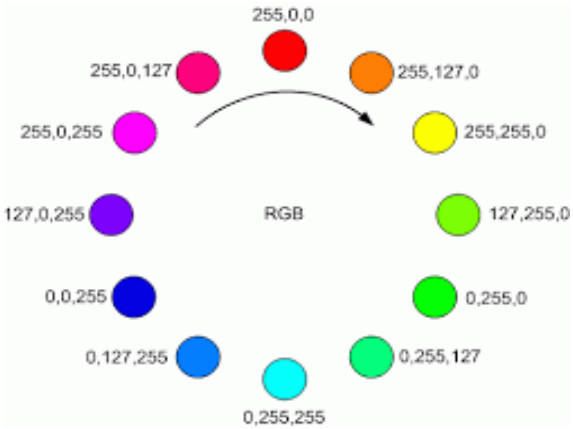

Fig. 6 Colours mixing using RGB-LED with analog control.

```
int AnalogInput1 = 0;//potentiometer attached to A0
int AnalogInput2 = 1;//potentiometer attached to A1
int AnalogInput3 = 2;//potentiometer attached to A2
int Value1; // Analog value1 for Red colour
int Value2; // Analog value2 for Green colour
int Value3; // Analog value3 for Blue colour
void setup () {
// initialize serial communications at 9600 bps:
Serial.begin(9600);
}
void loop ()
{
Value1 = analogRead (AnalogInput1);
Value1 = Value1 / 4;
Value2 = analogRead (AnalogInput2);
Value2 = Value2 / 4;
Value3 = analogRead (AnalogInput3);
Value3 = Value3 / 4;
analogWrite (RedLed, 255 - Value1);
analogWrite (GreenLed, 255 - Value2);
analogWrite (BlueLed, 255 - Value3);
Serial.print("R = " );
Serial.print(Value1);
Serial.print("\t G = ");
Serial.print(Value2);
Serial.print("\t B = ");
Serial.println(Value3);
delay(1000);
}
```

3.4 Temperature Measurement

Temperature measurement is one of the most common performed operations in the applied electronics. The obtained result is used not only for visualization but as an input for various control and signalization systems too. The next code example shows the sequence of operations in temperature measurement as follows: reading the code of the sensor's voltage from the analog-to-digital converter; determining the sensor's voltage in milivolts; determining the temperature in Celsius degrees; determining the temperature in Fahrenheit degrees. In addition it represents the relation between Celsius and Fahrenheit scales. The results are visualized using the serial monitor and the information is refreshed every second. From there can be observed the sensor's voltage in milivolts, the temperature in Celsius degrees and the temperature in Fahrenheit degrees.

```
const int analogInPin = A1;
// Analog input pin that the sensor is attached to
long analogIn = 0; // digital code
float sensorValue = 0;
// sensor analog output voltage value
float temperatureValue_C = 0;
// temperature value Celsius
float temperatureValue_F = 0;
// temperature value Fahrenheit
void setup() {
// sensor power supply on:
pinMode(A0, OUTPUT);
pinMode(A2, OUTPUT);
digitalWrite(A0, HIGH); // sensor (+5V) on
digitalWrite(A2, LOW);    // sensor (GND) on
// initialize serial communications at 9600 bps:
Serial.begin(9600);
}
void loop() {
// reading the code of the sensor's voltage:
analogIn = analogRead(analogInPin);
// calculates the sensor output voltage in milivolts:
sensorValue = analogIn * 5000 / 1024;
// calculates the temperature:
temperatureValue_C = sensorValue / 10;
temperatureValue_F=temperatureValue_C*9/5+32;
// print the results to the serial monitor:
Serial.print("sensor, mV = " );
Serial.print(sensorValue);
Serial.print("\t temperature, C = ");
Serial.print(temperatureValue_C);
Serial.print("\t temperature, F = ");
Serial.println(temperatureValue_F);
delay(1000);
```

```
}
```

4. Results

After completing the laboratory work dedicated to programming using Arduino board the students obtain experience in working with programmable devices. Basic skills in C programming language are mastered. The students realise the philosophy of the embedded systems and their application in all fields of the human activities and especially in the technical area. Also they are given knowledge about the most popular indicator and sensor elements, and basic skills to work with them. The initial explanation of the principles of data conversion and colours mixing introduce the students in the world of the modern information technologies. After the first approbation it was considered that most of the students solved the problems successfully and the tasks were made harder—sensors control and the related with it approaches were added.

5. Conclusion

In this paper a part of the course program in "Practice on open source platforms programming" from the new curriculum for Bachelor degree in Electronics at the Technical university of Sofia has been presented. The course gives the students theoretical knowledge about embedded systems programming and practical skills in this field. The practical approach is useful for the beginners. With emphasizing on the practice rather than on the academic theory they accept the material easier. The open source hardware and software—Arduino development board and the software IDE are very useful for the introductory education. They offer friendly environment for beginners and give them experience for the further courses.

References

[1] Wong, S., and Cotofana, S. "On Teaching Embedded Systems Design to Electrical Engineering Students." Available:http://www.researchgate.net/publication/228408026_.On_Teaching_Embedded_Systems_Design_to_Electrical Engineering_Students.

[2] http://cse.unl.edu/~carrick/courses/2012/236/236_2012_spring_embedded_systems.pdf.

[3] Lamers, M. H., Verbeek, F. J., and Putten, P. W. H. V. 2013. "Tinkering in Scientific Education." In *Proceedings of 10th International Conference on Advances in Computer Entertainment Technology (ACE 2013)*, LNCS 8253, 568-71.

[4] OLIMEX Ltd. 2011. "OLIMEXINO-328 development board. Users Manual."

[5] Banzi, M. 2011. *Getting Started with Arduino*. O'Reilly Media, Inc., ISBN: 978-1-449-309879.