Teaching Basic Skills in Embedded Systems Using Open-source Platforms

Peter Yakimov

Abstract – **In this paper a practical approach for teaching the basics of embedded systems hardware and software is proposed. It is emphasizing on the programme control of inputs and outputs to convince the students in the flexibility and universality of the programmable devices. Some examples are presented.**

Keywords – **Embedded systems, Open-source platforms, Arduino.**

I. INTRODUCTION

Embedded systems are everywhere. 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]. In order to prepare the students for the challenges of their future job the Faculty of Electronic Engineering and Technologies at Technical University of Sofia accepted a new curriculum for Bachelor degree in Electronics which was created after many iterations and discussions with the partners companies and employers organisations. The goal was to give the students theoretical knowledge about basic electronic circuits and devices and practical skills for their programme control. Also it was accepted that the practical training has to begin in the first semester in order to prepare the students for the specialised courses in the next years. 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 consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board I/O support. 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 [5].

Peter Yakimov is with the Faculty of Electronic Engineering and Technologies at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mail: pij@tu-sofia.bg.

II. NEW COURSE OBJECTIVES

According to the philosophy of the new curriculum a brand new course entitled "Practice on open source platforms programming" with two hours laboratory work per week was included. One major part of it 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 with appropriate examples the students to understand the relationship between computer devices and the surrounding world. The experimental work includes an application software writing 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 programme 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 programme 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.

III. LABORATORY SET-UP

Except the development board the necessary equipment includes a personal computer with an installed integrated development environment (IDE), a breadboard, some simple components as resistors and LEDs, and USB cable.

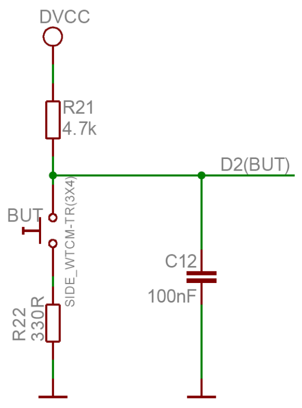

Fig. 1. User button with name BUT connected to ATmega328P pin 32 (digital signal D2)

The on-board user buttons and LEDs are used too as it is shown on 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.

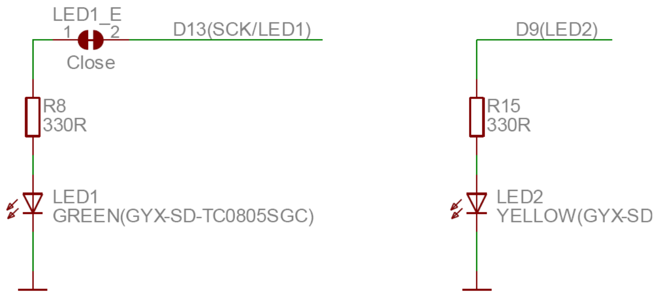

Fig. 2. Light emitting diodes connected to ATmega328P pins 17 (LED1 - digital signal D13) and 17 (LED2 - digital signal D9)

To study the operation of the analog inputs and outputs some additional tools 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 outputs because of their ability to set a pseudo analog voltage using pulse width modulation (PWM). 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.

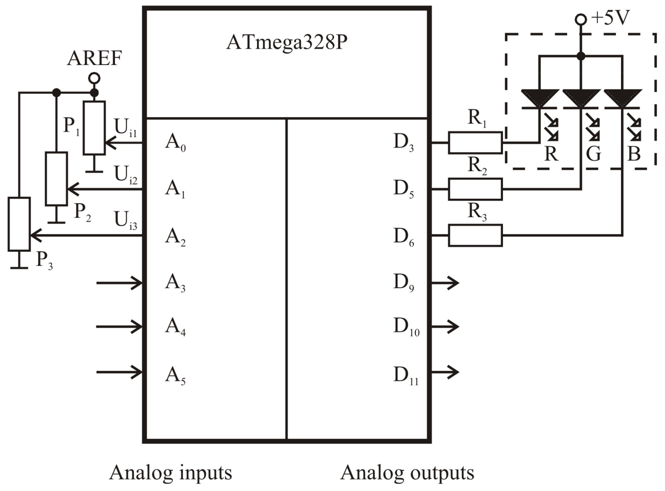

Fig. 3. Analog inputs and outputs connections for RGB-LED control

The laboratory set-up is shown on 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.

IV. 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 programme 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.

The first task is to control the alternative blinking of the two LEDs on the development boards shown on 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 pin 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. 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);
  }
}
```

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. 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 programme 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 connected 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 programme 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 connected 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
}
```

To attract the attention of the students and give them an understanding about the modern information technologies is useful to show them the basic principles of colours mixing. For this purpose a RGB LED can be used as it is shown on 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 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 state = 0;
void setup()
{
pinMode(RedLed, OUTPUT);
pinMode(GreenLed, OUTPUT);
pinMode(BlueLed, OUTPUT);
pinMode(Button, INPUT);
}
void loop()
{
if (digitalRead(Button))
{
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);
}
}
```

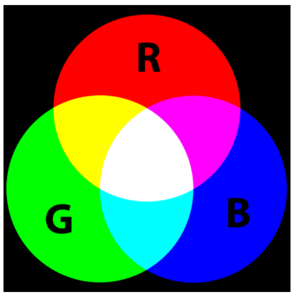

Fig. 4. Colours mixing using RGB-LED with digital control

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 on Fig. 4.

To apply analog control for the colours mixing every LED from the RGB must be controlled with output voltage that can be set with up to 255 levels as it is shown on Fig. 5.

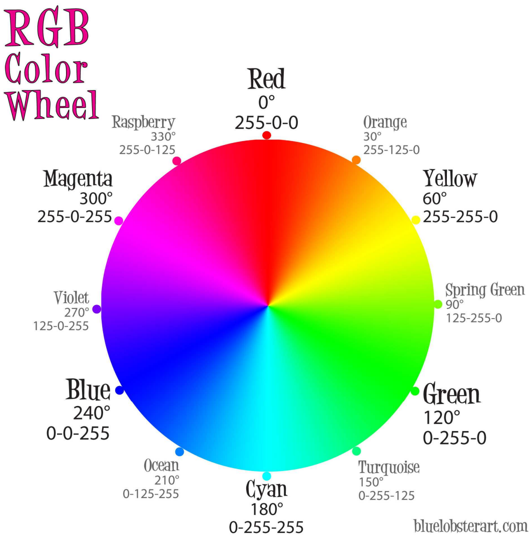

Fig. 5. Colours mixing using RGB-LED with analog control

The following code example illustrates the control. The laboratory set-up is shown on Fig. 3.

```
int RedLed = 3;
int GreenLed = 5;
int BlueLed = 6;
int AnalogInput1 = 0; // potentiometer connected to A0
int AnalogInput2 = 0; // potentiometer connected to A1
int AnalogInput3 = 0; // potentiometer connected to A2
int Value1; // Analog value1
int Value2; // Analog value1
int Value3; // Analog value1
void setup () {} //there is no need of setup
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);
}
```

V. 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 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.

VI. Conclusion

In this paper a part of the course programme 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.

Acknowledgement

The paper preparation is supported by Project BG051PO001-3.1.07-0048 “Updating the curricula of specialties in FEET, FTC and MTF of TU-Sofia and create a new joint Master's degree in accordance with the needs of the labor market”.

References

[1] S. Wong, and S. Cotofana, “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] M. H. Lamers, F. J. Verbeek, and P. W.H. van der Putten, “Tinkering in scientific education”, Proceedings of 10th International Conference on Advances in Computer Entertainment Technology (ACE 2013), LNCS 8253, pp. 568–571, 2013.

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

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