The 16th International Scientific Conference
eLearning and Software for Education
Bucharest, April 23-24, 2020
10.12753/2066-026X-20-082

NEW TEACHING METHODS BY USING MICROCONTROLLERS IN TEACHING PROGRAMMING

József UDVAROS, PhD., Ladislav VÉGH, PhD.
Faculty of Economics and Informatics, J. Selye University, Komárno, Slovakia
udvarosj@ujs.sk, veghl@ujs.sk

***Abstract**: Everyone knows that IT and IT education are important today. Within that, programming has a high priority. From the point of view of education, it does not matter what method we teach. In the educational process, it is very important to attract students' attention and interest in programming.*
Programming methodology is one of the oldest areas of computer science education, so several methods are used to teach it today. Some of these can be used effectively in primary and secondary education, while others can be used in higher education. Important teaching methods have now emerged in certain areas of programming education. Most teachers do not use a single method, but a mixture of methods in which one of them dominates.
Nowadays almost everything is controlled by electronics. We see various electronics gadgets almost everywhere around us. More and more microcontrollers are being used.
In any school, it would be good to teach IT to mean more than basic computer management. Information technology has a lot of possibilities and it is very easy to do amazing things with it! Technology, electronics, and information technology are increasingly part of our world. By programming microcontrollers, we can make the teaching of programming more interesting. With the help of lights and sounds, students' attention can be better captured.
This can be used to make programming learning interesting. Nowadays, a lot of primary and secondary schools are using this method and students are visibly motivated and more and more students are choosing electrotechnical and IT in their further studies. In this article, we introduce the use of this method.

***Keywords:** Microcontrollers; Programming; Control structures.*

INTRODUCTION

Students should have a minimum knowledge of electronics, which can be easily learned in physics and computer science lessons (Digital and Analog Signals, Electricity, Conductors, Semiconductors - Led Diode, ...), but the problem is that usually only a few teaching lessons are determined by a national curriculum [3]. Therefore we suggested developing short and intensive training courses, where students may get basic knowledge in electronics and microcontrollers. Authors of the articles below, point to the results of studies, which prove the power of teaching programming using microcontrollers.

Robert B. Reese and Bryan A. Jones in their article [8] declared, that the increased power and functionality of modern microcontrollers provides both an opportunity and a challenge to educators. Increased complexity, the necessity of integrating a hands-on lab experience, and downward pressure on total curriculum hours demand a significant investment of time to modernize microcontroller education, which few educators can afford. However, a successful microcontroller's course provides a unique opportunity to equip students with the necessary tools to produce large, complex systems in their capstone design course. The ever-growing complexity of microcontrollers challenges the

educator while also providing unique abilities to equip students with the ability to create large, complex systems. Effective teaching of microcontroller programming and hardware interfacing is challenging within EE and CPE programs for several reasons. First, a modern microcontroller is a complex system, with datasheet, programming reference, and family reference materials totalling over one thousand pages. Becoming familiar with a microcontroller family in the detail necessary for course integration is a time-consuming task for an educator. Second, microcontroller education still requires a significant hands-on laboratory experience despite advances in system simulation. Developing this laboratory experience requires an even greater investment of time by the educator.

Samuel N Cubero in the his article [6] describes goals of a new microcontroller course 'Mechatronic Project 234', and how students can be guided to become their own best teachers, able to test their newly acquired skills with only minimum or no direct assistance from an instructor or lecturer. 'Closed-loop' student-centred learning and problem-based learning approaches are described, involving weekly lectures and hands-on lab activities that keep students highly curious, motivated and engaged in self-regulated learning. Students are required to design and test their own original circuits and software code by modifying, extending or expanding the sample circuits and example code described in the lecture notes, in order to complete and demonstrate specific objectives or requirements for each weekly lab session. These 'closed-loop' student-centred learning labs ensure that all teams of students achieve a common or minimum acceptable level of practical skills, which adequately prepares them for a 'design-and-build' competition. This style of learning also helps develop generic life-long learning skills such as investigating and identifying problems (problem definition and analysis), independent research and experimentation, decision making, communication and teamwork. Even without any prior hands-on experience with electronic circuit design, programming and microcontrollers, each team of students was able to apply and demonstrate new knowledge and skills, devise and test original designs for circuits and software without any supervision, solve and fix complex problems successfully and confidently, and build an operational remote-controlled electric vehicle or mobile robot for a final competition. Some went even further and built sensor-guided fully autonomous mobile robots.

Che Fai Yeong, Hisyam bin Abdul Rahman and Eileen SU Lee Ming present a study [7] of student preference and performance while taking a microcontroller class with a 2-day curriculum that emphasized a hands-on approach. The curriculum uses the PIC16F877A microcontroller and participants learned to develop basic circuits and several other applications. Programming was completed on the MPLAB platform. Results show that participants had better understanding in this subject after attending the hands-on course. The practical sessions were fun for the students as they could explore their ideas through programming, and each participant came up with a variety of solutions for the same task. Students learnt from each other and improved their performance. A 2-day hands-on course would not be able to replace theoretical lectures, but could serve as a helpful addition to enhance student's learning.

In his article [9] C. E. Nunnally provides a brief look at the process used to teach microcontrollers to Junior/Seniors in EE and CpE. Each pair of students is issued a development hardware kit and related software. All software development is perfected on the student's required personal computer. The style presented allows for the teaching of several microcontrollers i.e. Motorola 68HC11, Intel 8051 and Microchip PIC family. The feedback obtained from participants of the short course have been favourable and encouraging. Many claimed that they have a better understanding of the microcontroller after the two day, hands-on course. From the survey, it was found that pairing the students had a positive impact on learning. Most participants preferred to work in pairs and from observation during the course, participants who were allocated individual kits also chose to discuss with friends and work together to complete their tasks. This scenario applied mostly to novice participants, who had no experience. Expert participants enjoyed working individually as they were able to work faster, and could attempt more complicated tasks based on their interest, with help from the instructor. To make the course effective for all participants, the instructor has to be aware of participants' skills and prepare additional tasks with a higher difficulty level for skilled participants, who complete their tasks early.

I. VISUALIZATION OF CONTROL STRUCTURES USING MICROCONTROLLERS

In education, it is the easiest to program an ATMega 32u4 microcontroller on the A-Star panel. After gaining experience, we can easily switch to other major controllers such as Arduino Uno, Arduino Mega, Raspberry PI, ...

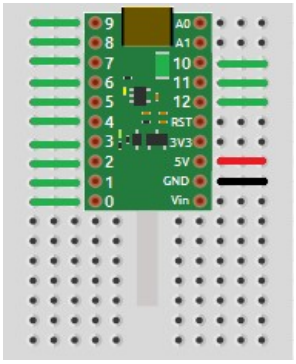

Figure no. 1. ATMega 32u4 microcontroller on the A-Star panel

You can program the points marked with 0-12. These can be used for output (e.g. led, beep) or input (button and all kinds of sensors). Sometimes we will use sensors that need continuous electricity, this is found at the 5V output. According to its name, a voltage of 5V is present here, which is relative to the GND point.

In addition to the hardware, we also need software support to program the microcontroller. This program is the Arduino IDE, which is free to download. Before you start programming, you need to make some adjustments (select the type of microcontroller, determine the COM input,...). The advantage of the Arduino IDE is that it has a library that contains pre-written programs with descriptions [1].

Let's look at a simple program:

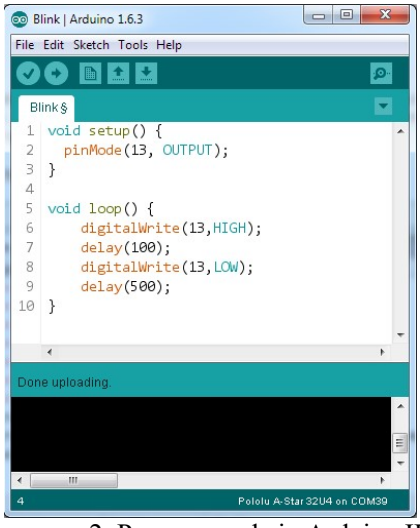

Figure no. 2. Program code in Arduino IDE

The *setup()* function contains the declaration of variables, constants, initial settings. It is executed only once during program execution. Here we set the microcontroller pin 13 to output with *pinMode ()*. Within the parentheses, the first number or variable denotes the digital pin between 0 and 12, the second a constant that can be either "OUTPUT" or "INPUT" depending on usage. Initialization is always an assignment, a setting.

The *loop()* function is the body of the program and it is executed until the program stops. *Void* is empty, the function has no return value. The body of the function (the instructions to be executed) is always enclosed in brackets. The *digitalWrite()* statement sets the output, the first value is always to identify the output, the second constant can take two values, “HIGH” on, i.e. 5V output, or “LOW” output off, 0V. The *delay()* statement can cause a report to be delayed, so it delays the program to its specified millisecond (500 milliseconds = 0.5 seconds). The *loop()* function is executed continuously until you stop it[4].

1.1. Instructions cycle

For instruction

Using the *for* statement, the instructions within the brackets are executed repeatedly using a numerical aid. In most cases, the cycle counter increases and exits the cycle when it reaches a limit. The *for* statement has three parameters, separated by a semicolon:

for(initialization; condition; expression) {instructions;}

In the section labeled initialization, we define a local variable which value varies per cycle run, taking into account the condition. If the expression is true, then the loop kernel is executed and the check begins again. When the condition becomes false, the cycle ends.

In the following example, the output 12 is powered 20 times for 100 milliseconds with 500 millisecond breaks.

```
for (int i=0; i<20; i)
{
digitalWrite(12, HIGH);
delay(100);
digitalWrite(12, LOW);
delay(500);
}
```

While instruction

The *while* loop is executed continuously until the expression in parentheses becomes false. The test characteristic must change during the run of the loop, otherwise the program will never move on. This can be either a rising variable or an external condition, such as a sensor signal.

while (someVariable ?? value) {instruction;}

In the following example, the instructions in the loop core are executed and it is repeated until the value of the variable 'varButton' is true.

```
While (varButton == True)
{
digitalWrite(12, HIGH);
delay(100);
digitalWrite(12, LOW);
delay(500);
}
```

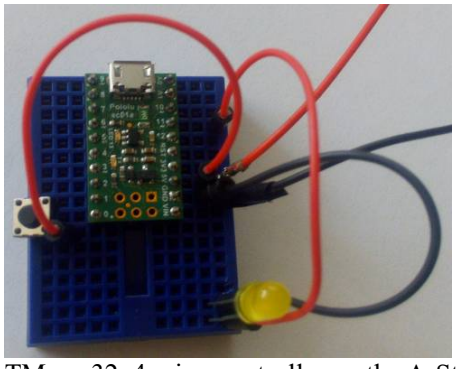

Figure no. 3. ATMega 32u4 microcontroller on the A-Star panel in use

Do...While instruction

The *do…while* loop is similar to the while type described above. The difference between the two solutions is that the loop ends with a condition, so in this case the condition is after the loop, which means that the loop runs at least once!

```
do {
instruction;
} while (someVariable ?? value);
```

In the following example, with *digitalRead()*, we examine the state of input 1 every 100 milliseconds, and write the value returned to the value variable. We send voltage to the output 12. The cycle is repeated until the value is HIGH. No voltage is applied to output 12 after the cycle is complete. In fact, output 12 will have voltage until input 1 is HIGH.

```
do
{
value = digitalRead(1);
digitalWrite(12, HIGH);
delay(100);
} while (value == HIGH);
digitalWrite(12, LOW);
```

1.2. Branch If…Else

The *if...else* statement represents a conditional branch. The part after else is executed if the condition is not met [2,5]. For example, if we examine the state of an input, something happens in the high state, but at the same time it can be given a command in the low state. In its spelling, it looks like this:

```
if (inputPin == HIGH)
{
instruction1;
} else
{
instruction2;
}
```

The else part can be used to examine multiple conditions in a single instruction line.

The following example shows that voltage is applied to its 11 or 12 outputs, depending on the state of input 1.

```
value = digitalRead(1);
if (value == HIGH)
{
digitalWrite(12, HIGH);
digitalWrite(11, LOW);
} else
{
digitalWrite(12, LOW);
digitalWrite(11, HIGH);
}
```

II. EXPERIENCES

A lot of things have happened in the past hundred years, nowadays everything has a chip, program and everything is getting smarter. Sadly fewer and fewer people understand these things now. This teaching method is about one really complex informatics course, not about the “boring” school exercises which the luckiest, who had programming lessons, solve with 20 year old technology. This

method prepares students exactly for the technology and challenges of the real world. Which students' attention wouldn't be fully captured if they could vocalize a tweeter or could manipulate LED lights?

The efficiency of the teaching methods was proved in a high school (gymnasium). Within a pilot program it was possible to open a study group, where 15 students were chosen. During the first lesson they learnt about the microcontroller and the programming surface, where they could write the solo program. We connected the microcontroller with the computer and then we wrote the first application, which lit the LED. After planning the first printed circuits and the programming of the microcontrollers, the students with their own creative ideas tried to modify the written program (a light show with the help of LED diode). At the beginning of the first lesson the students seemed distant, because they hadn't heard about the microcontroller and electronic parts before. At the end of the first lesson all the students seemed interested.

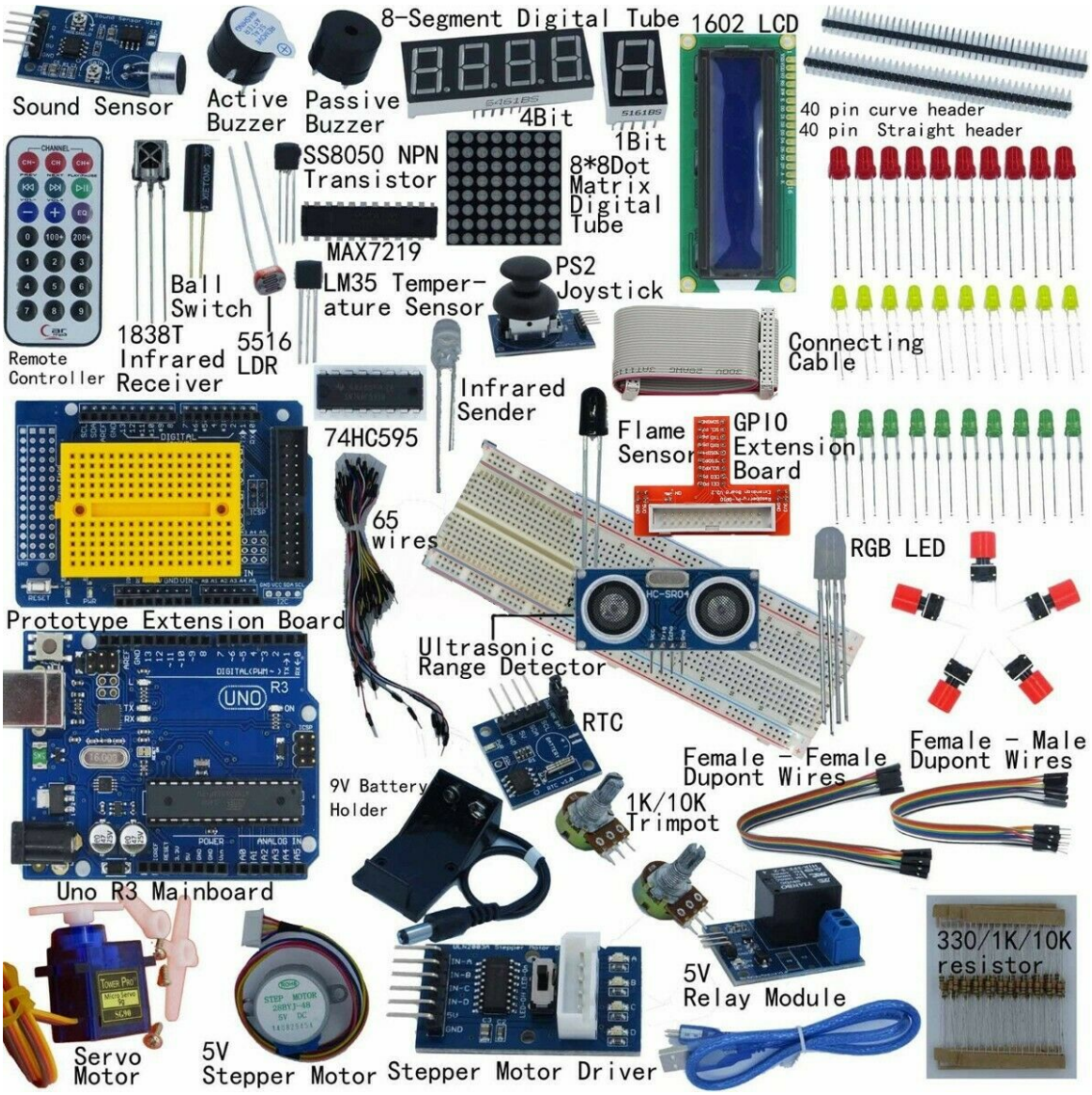

Figure no. 4. Electronic parts of the Arduino Starter kit

In the lessons that followed the students learnt about many other electronic parts, which could be used by connecting to the microcontroller. After the mastery of basics of some Arduino controllers, we showed them more possibilities. We suggest procurementing an Arduino starter Kit (Figure no. 4), which gives many more opportunities for electronic parts for an Arduino project. We can find many

descriptions for this on the internet, even a guide for mounting on YouTube. Among the students the Smart Robot Car Kit for Arduino is commonly known because they can get deeper into motor controlling. Also it is a very good opportunity to study and practice the line following, labyrinth entrance, usage of the remote control and practice control by Smart Phone (Figure no. 5).

At the end of study group we received very positive feedback from the students and parents. We have found that we can raise students` interest in electronics and programming. The students who joined this study group, want to go to universities in the future to study electrotechnics, to gain new skills in this field.

Figure no. 5. Opportunity of the Smart Robot Car Kit for Arduino

III. CONCLUSIONS

We created programs in C programming language in the Arduino development environment. We introduced the simplest control structures and the initial steps of programming an ATMega 32u4 microcontroller on the A-Star panel. All kinds of sensors, LEDs, and beeps can be connected to the microcontroller on a test panel. During the demonstration we connected an LED and a button.

While programming microcontrollers, electronics students can learn:

- how to use the test panel

• how to use a lot of interesting parts, such as Infrared sensor, NeoPixel led, Touch sensor, ...
• the uses of the microcontroller.

And in the meantime, they are also unobtrusively learning software development including:

• how to use the Arduino IDE
• the parts of the C programming language (constant, variable, instruction block, condition, cycles, etc.)
• a couple of built-in Arduino lib commands (digitalRead, digitalWrite, delay, etc.)

Acknowledgements

This work has been supported by the project KEGA 012TTU-4/2018 “Interactive animation and simulation models in education”.

Reference Text and Citations

[1] Arduino Tutorials (tronixstuff): http://tronixstuff.wordpress.com
[2] Stoffa, V. (2004). Modelling and simulation as a recognising method in the education. Educational Media International, 41(1), 51–58
[3] Udvaros, J. (2016). The investigation of OOP helper application effects in Slovakian secondary schools. Journal of Logistic – informatics - management 2016, volume 2016/1, ISSN 2498-9037.
[4] Udvaros, J. (2019). Visualization of Control Structures Using Microcontrollers. XXXII. Didmattech 2019. Trnava, Slovakia: Trnava University in Trnava Faculty of Education, p. 20, 6 p.
[5] Végh, L. (2011). Animations in Teaching Algorithms and Programming (Animácie vo vyučovaní algoritmov a programovania). Paper presented at the Nové technologie ve vzdělávání, Olomouc, CZ
[6] Cubero, N. S.(2015). Fun and effective self-learning approach to teaching microcontrollers and mobile robotics. International Journal of Electrical Engineering Education 2015, Vol. 52(4) 298–319
[7] Yeong, Ch. F., Rahman, H. A., Ming, E. S. L. (2013). A hands-on approach to teaching microcontroller. Systemics, Cybernetics And Informatics. Vol. 11(1). ISSN: 1690-4524.
[8] Reese, R. B., Jones, B. A. (2015). Improving the Effectiveness of Microcontroller Education. DOI: 10.1109/SECON.2010.5453894 · Source: IEEE Xplore
[9] Nunnally, C. E. (1996). "Teaching microcontrollers ", Proc. of Frontiers in Education Conference, 1996. FIE '96. 26th Annual Conference, vol.1, 6-9 Nov 1996, pp.434-436.