MICROCONTROLLER BASED AMBINENT LED LIGHTING

Vladislav Bidikov
Faculty of Computer Science and Engineering
Skopje, Macedonia

Nevena Ackovska
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

Efficient room lighting was always a real challenge especially in these modern times where energy efficient solutions are always more than welcome. On the other hand LED technology was always on the most economical and efficient way of providing lighting. The main problem we always face with LED is that in order to properly generate the right ambient colour we need the right combination on the R/G/B based LED system. Usually, given such a task, a preferable choice is a microcontroller. However, programming such a system is always a challenge. In this paper we will show how a modern Atmel microcontroller combines with the opensource Arduino platform. We will present how using little supporting electronics a PWM based RGB LED light system can be built.

KEYWORDS:
Microprocessor, Arduino, Atmel, PWM, LED, Ambient light.

I. INTRODUCTION

When we started our analysis on the task at hand we had to define some basic concepts in order to make the project more sustainable. Generally these concepts where based in two different categories: hardware and software.

On the hardware side we would like the platform to be minimalist, mainly based on the least needed number of components. Additionally, the idea was to build the system from components which are easily available and known to students in order to be possible to further develop the system.

On the software side we decided to try to use most common used programming languages and also if possible have some code reusage. Most microcontroller platforms are base on C or alike languages, so this was not a project stopper.

When looking into the different options one platform made it really easy to adopt the above principles both on the hardware and software side. It's called Arduino project and basically it's an Atmel AtMEGA microprocessor put on a minimalist development board which can easily be programmed via USB with C programming language.

This allowed for the project to be easy to develop and expand and also allowed future student cooperation on extending the project or using its knowledge for more complicated projects. One of the biggest advantages was the fact that the development board allowed almost no additional electronic for driving the RGB LED strip, since it is already PWM ready.

II. THE ARDUINO PLATFORM

1. HARDWARE: An Arduino board consists of an 8-bit Atmel AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, now superseded by the Duemilanove, for example, provides 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs. These pins are on the top of the board, via female 0.1 inch headers. Several plug-in application shields are also commercially available.

The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards provide male header pins on the underside of the board to be plugged into solderless breadboards.

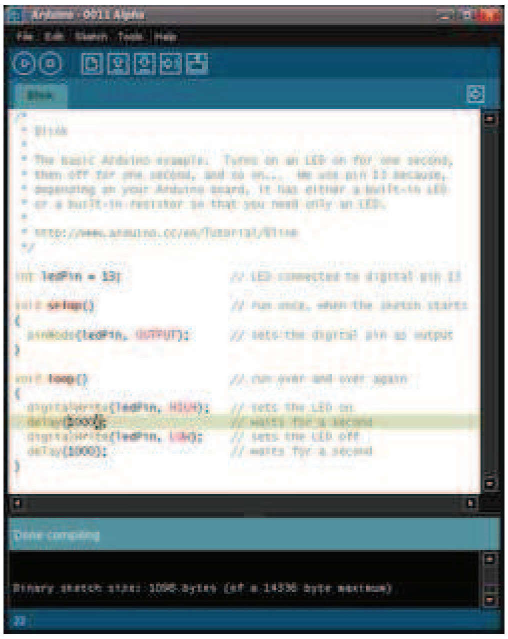
2. SOFTWARE: The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ino. The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name),

which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:

* setup() – a function run once at the start of a program that can initialize settings

* loop() – a function called repeatedly until the board powers off

An example of the Arduino IDE is shown on Fig.1



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                 // LED connected to digital pin 13

void setup()                     // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);       // sets the digital pin as output
}

void loop()                      // run over and over again
{
  digitalWrite(ledPin, HIGH);    // sets the LED on
  delay(1000);                   // waits for a second
  digitalWrite(ledPin, LOW);     // sets the LED off
  delay(1000);                   // waits for a second
}
```

```
Done compiling

Binary sketch size: 1098 bytes (of a 14336 byte maximum)
```

Figure 1: The Arduino IDE

For this project out of the many different versions we decided to use the Arduino MEGA1280 version which has the required number of PWM outputs as well as the required I/O lanes for the other elements.

III. DESIGN OF THE SYSTEM

The system was designed to be very easily extendable and also with an idea to be an example for students which would like to know about its design and functioning concept. For that purpose all elements where modular. The system is made from these elements:

1. Arduino MEGA Board – this is the heart of the system. The board will use the IR receiver to receive commands from the remote control and then transform these commands into PWM signals which will drive the RGB LED strip or multiple RGB LED strip elements. (shown on Fig. 2)
2. IR receiver – a 38 KHZ IR receiver module which can receive commands from generic IR remote controllers commonly found in all modern appliances (TV, DVD, Surround system etc.) (shown on Fig. 3)
3. RGB LED driver - since most of the electronics are already on the Arduino board the RGB LED driver will actually be a ULN2003 chip which is a Darlington transistor monolithic kit. (shown on Fig. 4)
4. RGB LED strip – this is the active element of the system which will generate the planned ambient lighting. (shown on Fig. 5)

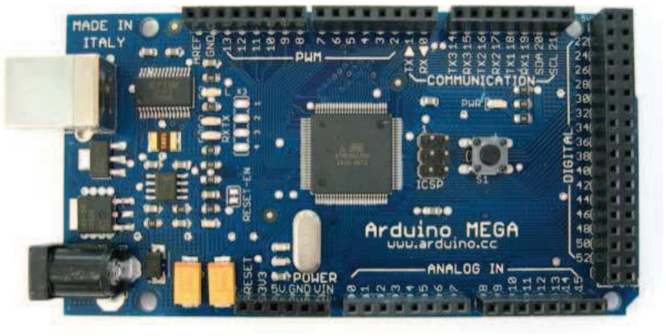


Figure 2: Arduino MEGA board

Figure 3: IR Receiver

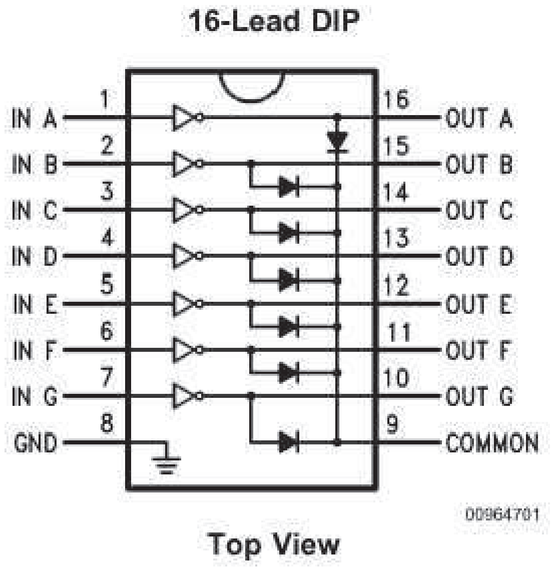


Figure 4: ULN 2003 IC

Figure 5: RGB LED strip element

The design on the system was based on these functional concepts:

1. The system will start do a simple self-test and then make the LED strip white with intensity of 1/6 of the output power of the LED strip.
2. After this stage, the system will listen for commands while maintain the predefined values for the color of the LED as well as their intensity.
3. When a command is received the system will react accordingly.
4. In order to see a more natural light the system will to these operations every 200ms which combined with the PWM mode on the LED strip it will make the whole effect natural.

When designing the command reference we decided to have a few different ways of giving commands especially since we plan so use a transmitter which will have a substantial number of keys. Some of these keys we will program them with predefined colors, while other will allow for change of color and/or output power with a defined command sequence.

A very small subset of keys we will program with special macros which will allow using the system under special circumstances like going to sleep, waking up or emergency situations.

Also we plan the system to remember the last state it was in before we press the power off button. Since we expect such a system to have a power on function when going in power off mode that system will only shutdown the RGB LED elements (which are the biggest consumer) and remain active in a power saving mode awaiting for the power-on command to arrive. While in this mode all other commands will be ignored. In case of a system error the system will use the onboard green LED to signal and give additional feedback thru a LCD panel which can be connected for diagnosing purposes.

Special attention in the code will be made for the possibility to remap the keypads on the IR transmitter since we expect that the end user will have a slightly different IR transmitter. However, such a remap operation will be easy since the code is in C language and it is easy to do the remap process once we connect a LCD for debugging.

The initial set of commands is given in Table 1.

Table 1: Commands reference of programmed functions…..

Button	Function
1	Color 1 (Red)
2	Color 2 (Green)
3	Color 3 (Blue)
4	Color 4 (Yellow)
5	Color 5 (Magenta)
6	Color 6 (Dark Blue)
7	Color 7 (Brown)
8	Color 8 (Orange)
9	Color 9 (Dark Green)
0	Color White
V+	Increase output 5%
V-	Decrease output 5%
A	Output 25%
B	Output 50%
C	Output 75%
D	Output 100%
MUTE	Sleep macro
M	Emergency 100% macro
OK	Enter RGB color value macro (followed by 3 digit combinations for R/G/B values)

IV. BUILDING THE DEVELOPMENT VERSION

Building the development version of the hardware was basically soldering some short wires to all the modules and then use the Arduino pads to connect the corresponding modules. After this short soldering and connection process we started developing the software. Since the Arduino platform has many excellent examples the process of the writing of code was very fast and after the initial writing we had all the

needed functions in order to start the process of mapping the buttons with their specified commands. For better future development all commands where done using sub functions which will even further make the project modular and easy to be additionally developed.
The only problem we encountered was a small bug in the IR transmitter which was sending the same code twice in a row. After a short debug session it was brought to our attention that the IR transmitter was faulty so we just replaced it with a working one but there was a possibility in the code to compensate for such occurrences if needed.
After we build the specified command set we decided to take the system for a field test in order to test the commands as well as see how the system will react under normal usage from people which are not part of the developer team.
The results where quite interesting, since all of the test subjects managed to use the system without any noticeable error. The only time the system failed actually was where there was a short power spike which made the microcontroller power cycle.

V. CONCLUSION

The experience we learned from this project is indeed a very interesting one. We saw how easily we can develop a complex microcontroller circuit with the minimal set of additional electronics to support it as well as the possibility to develop software for it in a very friendly way. The Arduino IDE gave us almost the same environment as a modern C programming language editor as well as a full set of additional microcontroller based diagnostic tools like the serial console. Also the IDE provided a easy way of uploading the completed code to the microcontroller by using a USB interface. On the hardware side we can see how this project could be further developed in means of adding more complex features while the possibility to add additional add-on cards allow to further think into using other technologies like Ethernet or even wireless 802.11 communication. These possibilities will allow making such light controller network aware and even be used with a possibility to have their own AI. All these tasks can be easily research in the curriculum on the university since the Arduino platform has the versatility to perform all these tasks.

REFERENCES

[1] The arduino Project – http://www.arduino.cc

[2] Jonathan Oxer, Hugh Blemings: Practical Arduino: Cool Projects for Open Source Hardware – ISBN-13: 978-1430224778, 2009