**Arduino and Morse**

**Rus Bogdan Alexandru**

E-mail: [rusbogdanalexandru@yahoo.com](mailto:rusbogdanalexandru@yahoo.com)

**Bogdan Sever Cristian**

E-mail: [severica2312@gmail.com](mailto:severica2312@gmail.com)

**Abstract.** This documentation contains relevant information about the Arduino and Morse project realized by Bogdan Sever Cristian and Rus Bogdan Alexandru. In this regard, it includes the steps into making the connections on the breadboard and with the Arduino Uno board. Furthermore, it contains the code and relevant explanations of it, which will help the reader to understand the process behind it. In the end, we obtained a nicely done and easy to understand project, that not only replicates the old Morse transmission machines, but also one that will display the message decoded on an LCD.

1. **Introduction**

Arduino is an open-source hardware and software company, that manufactures microcontrollers and microcontrollers kits for building digital devices. So, for our project, to implement the usage of the Morse code, we used this type of microcontroller, that lets us to modify it in any way, through a code.

In the next sections and subsections, we will try to take you through every step in the implementation of our Arduino and Morse project.

1. **The scope**

Our intention was to create a simple and usable Morse machine, using the Arduino kit. Also, our scope is to show how simple it can be to make such projects, only using a few knowledge from the iT domain and a little bit from the electrical domain.

1. **Components and supplies**

For this project, the main component is the Arduino board, respectively the Arduino UNO. This board will encrypt the code through the microcontroller and will permit us to work with the electrical impulses.

Next, there was used a LED1, which will allow us to see the Morse code we introduced, as dots and dashes. The input of the Morse code will be realized through a button connected to the Arduino board. Moreover, there was used an LCD screen to display the letters and/or the numbers translated from the Morse code and a 10K Potentiometer that allows us to change the LCD’s screen luminosity. Not least, there were needed 2 x 220K Resistances, one for the LED and one for the LCD screen, and wires to connect the entire circuit.

1. **Apps and online services**

The only app needed for the whole project is the Arduino IDE2, which can be easily downloaded and installed.

1. **Schematics**

In this section will be presented the electrical schematics of the circuit and it will have 4 subsections: LED, Button, LCD and the whole circuit.

*5.1 LED schematic*

|  |
| --- |
|  |
| **Figure 1.** The LED should be connected to 13th digital pin on the Arduino board and should have connected between the pin and the ground a 220K resistance, as the scheme shows. |

*5.2 Button schematic*

|  |
| --- |
|  |
| **Figure 2.** The Button should be connected to 8th digital pin on the Arduino board, as the scheme shows. |

*5.3 LCD schematic*

|  |
| --- |
|  |
| **Figure 3.** The LCD has 12 connections. For a better reading, the connections will be simplified as: “LCD pin” – “Arduino pin”. VSS – GND; VDD – 5V pin; V0 – 2nd head of potentiometer; RS – 12th pin; R/W – GND;  E – 9th pin; DB0 – no pin; DB1 – no pin; DB2 – no pin; DB3 – no pin; DB4 – 5th pin; DB5 – 4th pin;  DB6 – 6th pin; DB7 – 2nd pin; LED+(anode) – 5V pin (through a resistance R2); LED-(cathode) – GND.  The Potentiometer has 2 other connections: 1st head to 5V pin and the 3rd head to the GND. |

*5.4 The circuit schematics*

|  |
| --- |
|  |
| **Figure 4.** After following the schematics above, the whole circuit should have the connections as the figure above shows. |

1. **Code**

The next part of the documentation will give an insight into how the actual code helps us to implement the Morse Code using an Arduino Board.

Firstly, there exists 2 main functions: setup() and loop(). The setup() function will be used by the microcontroller only once, because this function initializes the pins as inputs/outputs and the LCD. The loop() function will do the actual work, as it will go in a loop as long as the Arduino Board receives any current.

Secondly, to use the LCD, there must be included the “LiquidCrystal.h” library and be initialized the pins of the LCD as the schematic above shows (see Figure 3).

Moreover, if you want to use the code, you can have it at this link:

<https://codeshare.io/morsearduino>

Now, in the next part, the code will be explained on pieces, so that the reader has a better understanding of it.

*6.1 The definition and initialization of the constants and variables*

|  |
| --- |
| Text  Description automatically generated |
| **Figure 5.** In the first part, the button pin, the led pin, and the LCD’s pins are defined and initialized as constants, as the connections were made on the board. Next, there are the definitions and the initializations of the global variables that help us in determining the period of the pressing of the button, so that the transformation in dots and dashes can be realized. Also, there is the definition and initialization of the error message and the reset function. |

*6.2 setup() function*

|  |
| --- |
| Text, letter  Description automatically generated |
| **Figure 6.** The setup() function will setup the LCD screen as a 16x2 matrix of characters and will set the cursor at the first position. Next, it will set the pin for button as an input and the pin for LED as an output. |

*6.3 LCDDisplay() function*

|  |
| --- |
|  |
| **Figure 7.** This function allows the error text being scrolled from left to right, one character at a time. Firstly, it will display the first 16 characters, then it will scroll to right, so the user can see all the text. It will have been displayed for 21 seconds. |

*6.4 MakeString() function*

|  |
| --- |
|  |
| **Figure 8.** This function determines if the input is a dot or a dash. To determine the result, the input signal is measured through the variable pres\_len. In the case the input signal is less than 0.75 seconds, then it is a dot. In the other case, it is a dash. |

*6.5 light() function*

|  |
| --- |
|  |
| **Figure 9.** This function activates the LED, depending on the dots or dashes that forms the letter or the number. For a dot signal, it will be on for 0.75 seconds and for a dash signal it will be on for 1.75 seconds. Between the signals there it will be 0.5 seconds of delay. |

*6.6 Morse\_decod()*

*6.6.1 First part*

|  |
| --- |
| Chart, scatter chart  Description automatically generated |
| **Figure 10.1.** The first part of the function stores the codes of the letters and of the numbers in 2 separate arrays for a better understanding of the code. All the codes are in alphabetic/numeric order (i.e., A, B, C... and 0, 1, 2…). |

*6.6.2 Second part*

|  |
| --- |
|  |
| **Figure 10.2.** The second part of this function verifies if the code that the user introduced is among the codes that corresponds to the letters in the array (see Figure 10.1.). If the code is found, then on the LCD screen will appear the letter and will scroll the text one bit to the left. Also, the LED will turn on and off corresponding to the Morse code of that letter (see Figure 9.). |

*6.6.3 Third part*

|  |
| --- |
|  |
| **Figure 10.3.** The third part of this function verifies if the code that the user introduced is among the codes that corresponds to the numbers in the array (see Figure 10.1.). If the code is found, then on the LCD screen will appear the number and will scroll the text one bit to the left. Also, the LED will turn on and off corresponding to the Morse code of that number (see Figure 9.). |

*6.6.4 Fourth part*

|  |
| --- |
|  |
| **Figure 10.3.** The fourth part of the function will only be active if the user entered a wrong input. In that case, an error message will appear on the LCD screen for a predetermined period and then the code will be reset for the next iteration. |

*6.7 loop() function*

*6.7.1 First part*

|  |
| --- |
| *Graphical user interface, text, application, email  Description automatically generated* |
| **Figure 11.1.** The first part of the function measures the time that the user has pressed the button and memorizes it in the pres\_time variable. Next, if the user did not press the button for 3 seconds, on the LCD screen will be shown a space. Then, it will measure again the time in which the button is not pressed and will save it in the variable rel\_time. |

*6.7.2 Second part*

|  |
| --- |
|  |
| **Figure 11.2.** The second part of the function calculates the time in which the button had been pressed and then, if it is longer then a minimum delay of 10 milliseconds, the input will form a “code” through the MakeString() function (see Figure 8). Next, if the input time between the pressings of the button is smaller than 0.75 seconds, it will go back to recognize another input. If not, the Morse\_decode function (see Figure 10.) will form the letter, the number or an error code, that will be shown on the LCD screen. |

*6.8 Tips for a better understanding of the code*

For a better understanding of the code presented above, every important sequence has been commented. Also, for other pieces of information regarding the pre-defined functions (i.e. millis(), digitalRead(), lcd.print() etc.) you can visit the reference page of Arduino at the link: <https://www.arduino.cc/reference/en/> .

**7. Extensions**

If the user of this project may feel that he/she can handle it, there can be added a buzzer that makes a sound every time a letter or a number is shown on the LCD screen. Moreover, there can be added a speaker that would say the words that the user has written and the Arduino board has displayed on the LCD screen.

**8. Conclusion**

To put it in a nutshell, by pressing a button on the breadboard, you can easily introduce the Morse code of letters and numbers which will be displayed on the LCD screen. Also, when the button is pressed a LED will be on, which will be indicating that the circuit it is working and will show the Morse code as dots and dashes.

**9. Acknowledgements**

We want to thank our professor for the opportunity to work on a practical project that involves both team-working and self-research and also for all the tips and tricks during the making of the project!