CLS PROJECT REPORT

Converting Morse Code to String and Vice versa

Group Members:

Ghazal Shenavar 97101897

Arshia Akhavan 97110422

Sara Khosravi 97101586

Professor : Doctor Asadi

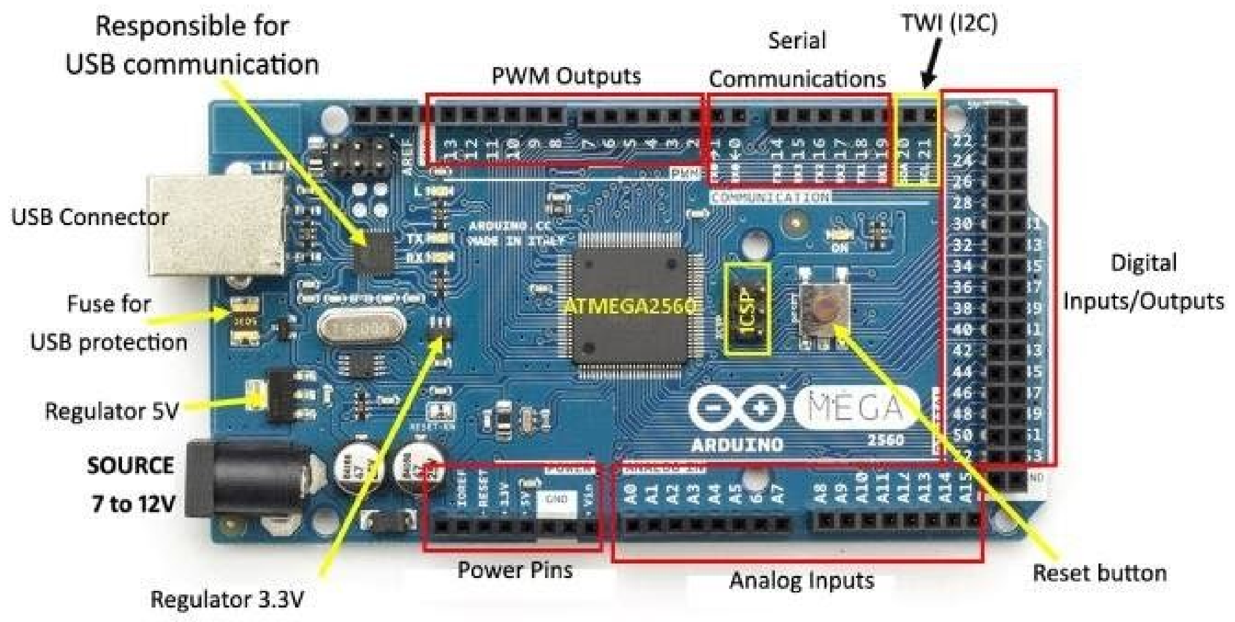
Sharif University of Technology
Computer Engineering Department

Project Introduction:

This project uses a few modules(introduced below) and a C program to do two things: 1. Get a string of characters as input via a keypad, turn them to morse code and display them using an LED 2. Get a morse code as input via a button and display their translation into characters on an LCD.

Used Modules:

Arduino mega 2560:



The ATmega 2560 processor has a 256kb memory and 86 general purpose pins.

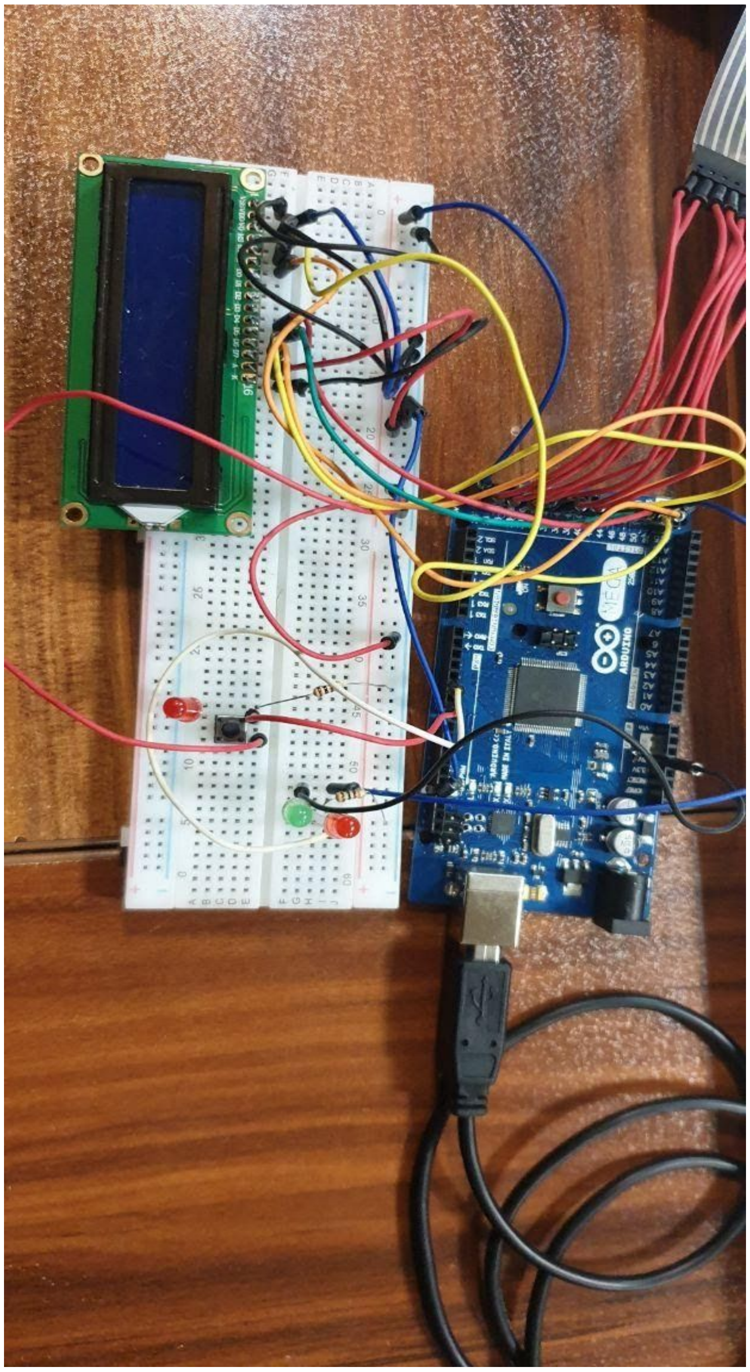
Keypad 4x4:

This module has 8 pins; the first 4 are used for rows and the other 4 are used for columns. When a row and a column are selected it means that the button they share is selected on the keyboard.

LCD 16x2:

The first 2 pins are power supplies of LCD and the 3rd pin controls the contrast. The 4th pin, called register select, decides which one of the 2 registers of LCD should be used; inputting 0 saves the sent instruction in the instruction register and inputting 1 saves data in the data register. The 5th pin, read/write, recognizes 0 as write mode. The 6th pin is the enable pin. The 7th to 14th pins are used for data transmission; we only used the last 4 in this project. The 15th pin is backlight anode and the 16th pin is the backlight cathode.

The Circuit:

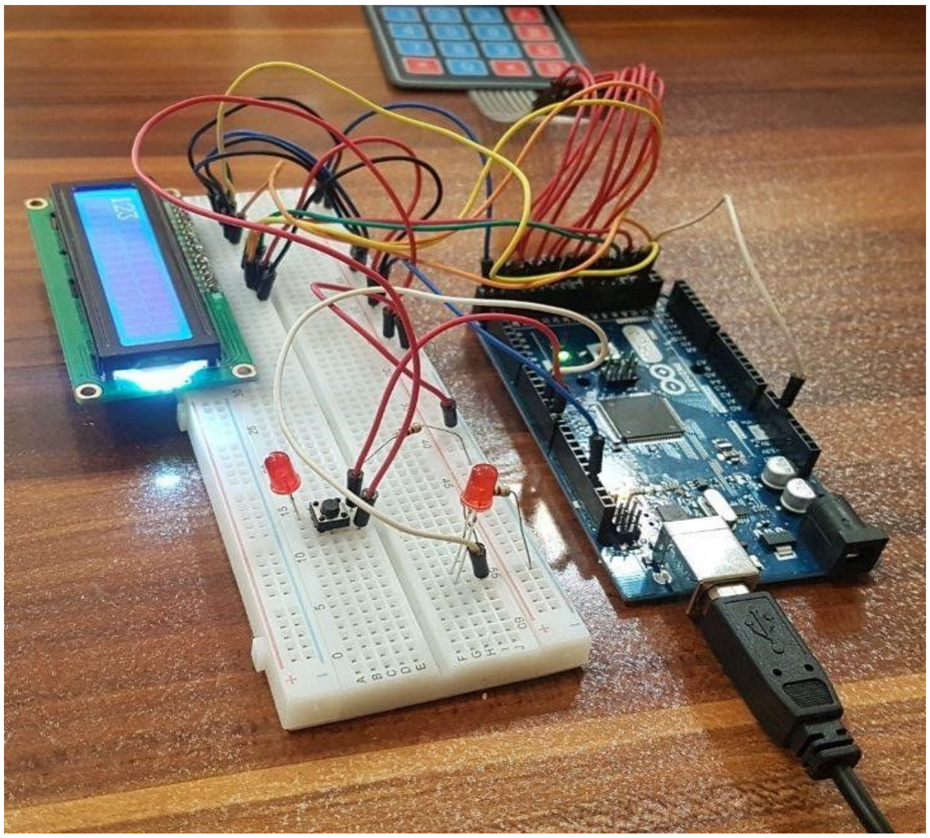


We connected the arduino board to the laptop via the USB connector; that was also used as our board's power supply. The rest of the modules were connected to the board via digital input/output or pwm outputs.

Because of the lack of a 2 mode switch the white wire on the right of the circuit chooses the mode; and the green LED displays the current mode. If the white wire is connected to Gnd the green LED is off and the input is taken from the keypad (then converted to morse code then shown via the red LED that is connected to the 5th pin). If the white wire is connected to Vcc the green LED is on and the input in morse code is taken from the button connected to the arduino's 7th pin (then the translation is shown on LCD).

The keypad module was connected to the arduino via digital pins 22, 24, 26, 28, 30, 32, 34, 36.

For connecting the LCD we used this link. The 1st pin is connected to Gnd and the 2nd pin is connected to Vcc. As we didn't have a potentiometer we connected the 3rd pin directly to Gnd. The 4th pin is connected to digital pin 48, the 5th pin to Gnd, the 6th pin to digital pin 50 and data lines d4 to d7 to pins 46, 44, 42, 40. Anode is connected to Vcc and cathode to the Gnd.



Code:

All the codes are available in Git.

Conversion of string to morse code:

Morse codes had at most 5 lines/dots so we created codes, starting from the least significant placing, putting 0 instead of dot and 1 instead of line and continued each code smaller than 5 digits with 2(look at morse.txt in morse_data folder). Then we converted the resulting numbers from base 3 to base 10 and placed them in the morseCodes array(in morse.ino file). We used the MultitapKeypad.h library for the keypad. Whenever a key is pressed the array cell connected to that character is retracted and the value of that array cell is given to a function. The said function then converts the number to its base 3 equivalent step by step; in any step if 2 is seen the function terminates, if 0 is seen LED is turned on for 0.1 of a second and if 1 is seen LED is turned on for 0.5 of a second(these functionalities are done by handleKey function in Keypad.ino, morseCode and genMorse functions in morse.ino). The star key deletes the last character from the buffer and the hashtag key prints the buffer on LCD then clears the buffer.

Conversion of morse code to string:

The morse code input is taken from a button. The input is considered a dot if the button is pressed for less than 0.3 of a second and is considered a line if the button is pressed for longer than that. The end of a character is declared by not pushing the button for 0.5 of a second. If the equivalent of the character is in morseDecodes array the character is considered valid and is printed on the LCD(morseDecode function in morse.ino). For each invalid morse code a space is printed to inform the user of a mistake(handle button function in button.ino).

Anytime the LCD reaches the end of its screen it starts printing from the start of the screen again.

All the durations are changeable variables.