University *Polytechnica* of Bucharest

2023

Applied Informatics Project

Morse code player

Coordinator : Alexander Guzu

Students :

Chivu Rares-Ioan
Angheluta Andrei-Alexandru
Tuturman Madalina-Silvia
412G

Contents:

I. Introduction:

This practical work aims to display certain characters entered on the serial port in Morse code on the screen, simultaneously with the emission of sounds played through a buzzer.

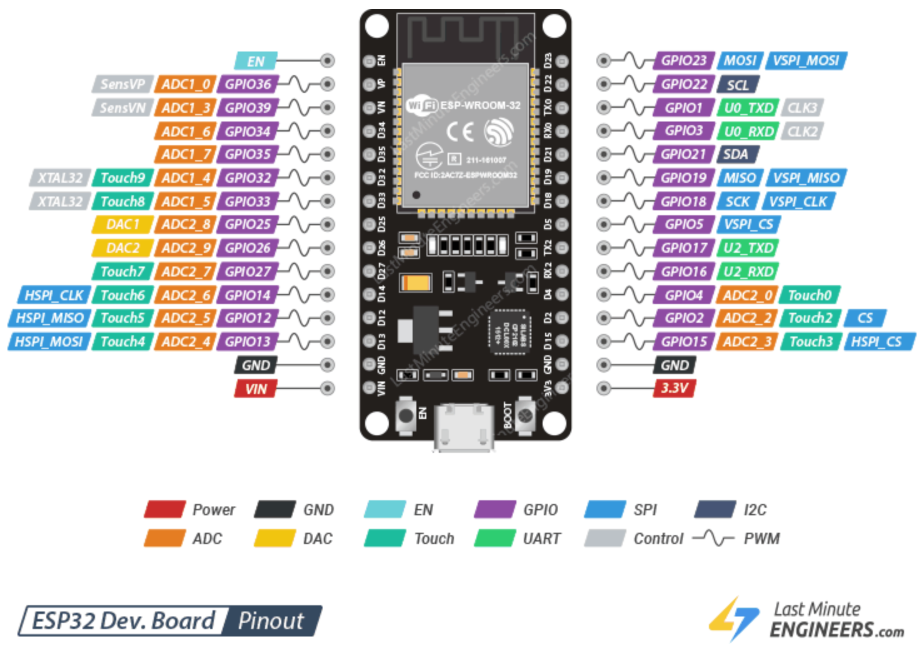
Purpose of a such work is communication in Morse code using the Arduino device.

II. Resources:

To complete the project, we used the ESP32 development board provided by the faculty and additional components purchased separately .

The ESP32 is a popular microcontroller and Wi-Fi/Bluetooth module developed by Espressif Systems. It features a dual-core processor, integrated Wi-Fi and Bluetooth connectivity, versatile hardware peripherals, and a wide range of functionality.

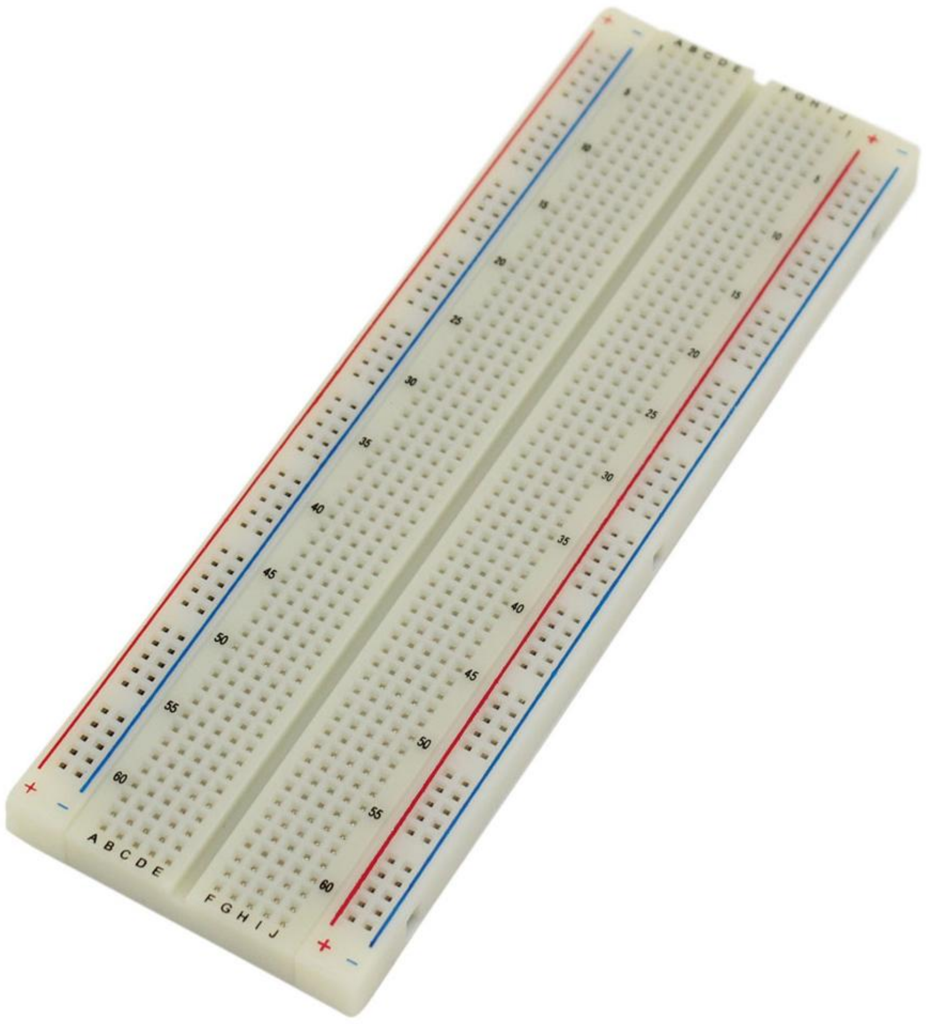
It is widely used in various Internet of Things applications, robots, and other embedded systems projects.



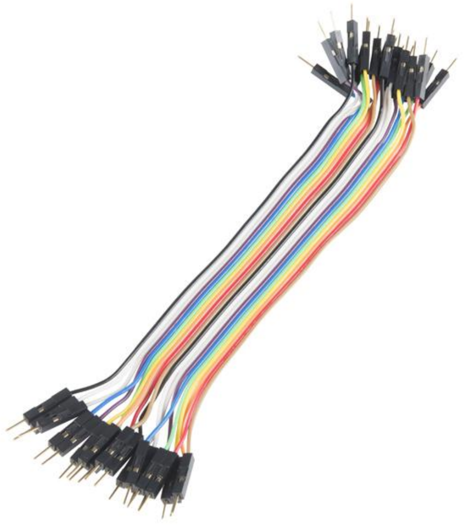
A breadboard is a tool used in testing electronic circuits. It is a support in which electronic components can be connected and interconnected without the need for soldering or welding.

A breadboard consists of a plastic board with a grid of small holes and metal columns below each hole. These columns are electrically connected in groups. Each hole in the grid is connected horizontally to the holes in the same column, and the vertical ones are connected in smaller groups. This allows for easy connection of electronic components such as resistors, capacitors, transistors, LEDs, and others using connecting wires or pins.

By using a breadboard, prototypes and experiments can be made quickly and easily without the need to make a permanent circuit by soldering or welding.



To connect components between they were used father-father type wires, and for connecting the microcontroller with the computer we used a type cable microUSB.



Next was used a resistor. Resistors are composed of a material with electric resistance electric materials, such as carbon or metal that is formed into a long and thin shape. Electrical resistance of a resistor is measured in ohms (Ω) and determines how much the resistor opposes electric current.

An active buzzer was used as the output device for the Morse code in sound mode. The buzzer is a very easy way to play sounds with frequencies up to 2KHz. It has an oscillating source inside that produces sounds when current flows through it.

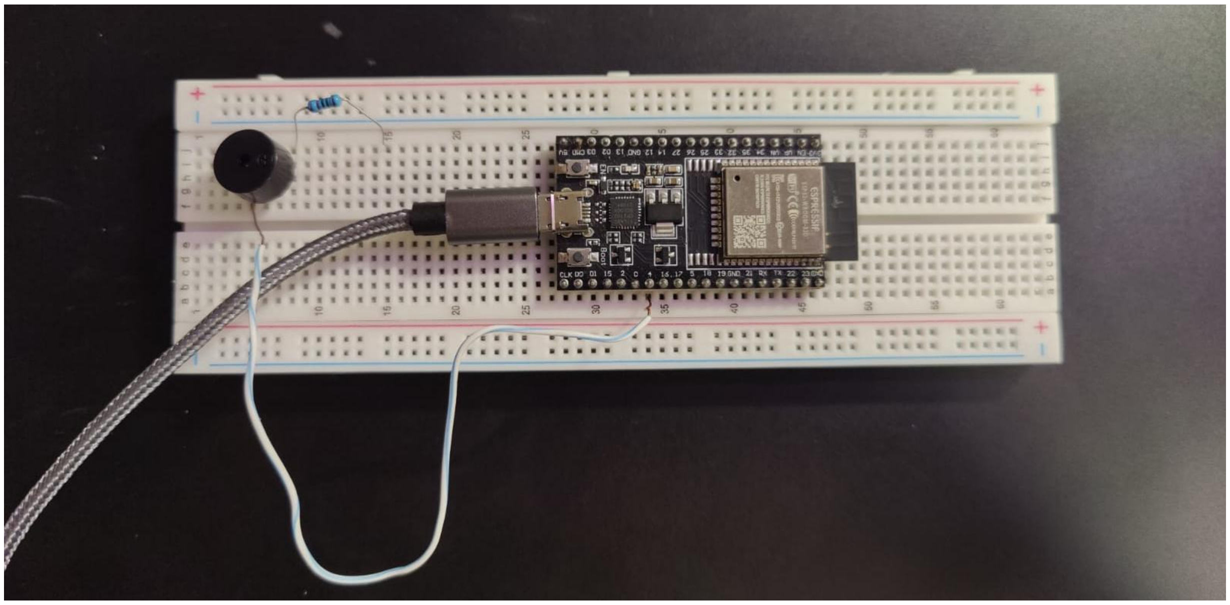
III. <u>Resources:</u>

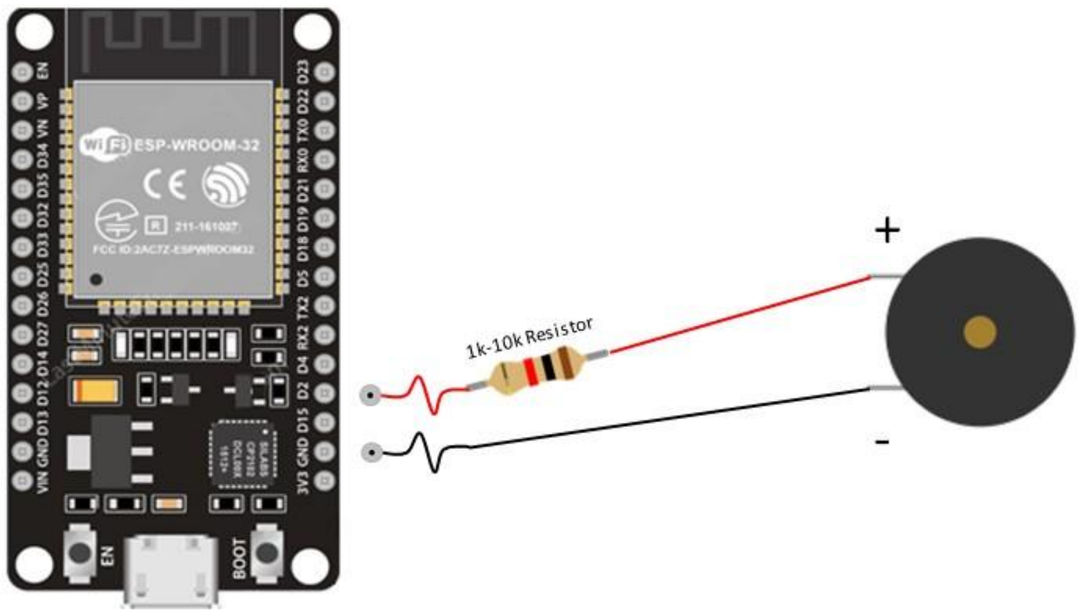
The software component is what ensures the functionality of each hardware component; it is a set of programs. Arduino is an open-source hardware and software development platform used for prototyping and building various electronic projects.

The Arduino platform consists of a development board with a microcontroller and a programming environment It provides a simple and friendly programming environment, based on the C/C++ programming language, which allows us to write code to control and interact with components connected to the Arduino board.

IV. <u>Hardware implementation:</u>

The photo below illustrates the actual circuit accomplished for the project :





V. Implementation software:

The following diagrams represent the logic diagrams of the code used:

```
char stringToMorseCode[] = "";

int audio8 = 4;
int note = 500;
int dotLen = 100;
int dashLen = dotLen * 3;
void setup() {
  Serial.begin(9600);
}
void loop(){
  char inChar = 0;
  char inData[100] = "";
  String variable = "";
  int index1 = 0;
  if ( Serial.available() > 0 ) {
    while (Serial.available() > 0 && index1 < 100)
    {      delay(100);
      inChar = Serial.read();
      inData[index1] = inChar;
      index1++;
      inData[index1] = '\0'
    }
    variable.toUpperCase();
    for (byte  i = 0 ; i < 100 ; i++) {
        variable.concat(String(inData[i]));
      }
      delay(20);
    }
    String  stringToMorseCode = String(variable);
    for (int i = 0; i < sizeof(stringToMorseCode) - 1; i++)
    {  char tmpChar = stringToMorseCode[i];
tmpChar = toLowerCase(tmpChar);
  GetChar(tmpChar);
}

void MorseDot()
{  tone(audio8, note, dotLen);
  delay(dotLen);
}
{  tone(audio8, note, dashLen);
  delay(dashLen);
}

void GetChar(char tmpChar)
{  switch (tmpChar) {
     case 'a':
     MorseDot();
     delay(100);
     MorseDash();
     delay(100);
     break;
     case 'b':

     MorseDash();
     delay(100);
     MorseDot();
     delay(100);
     MorseDot();
     delay(100);
     MorseDot();
     delay(100);
     break;
     case 'c':
     MorseDash();
     delay(100);
     MorseDot();
     delay(100);
     MorseDash();
     delay(100);
     MorseDot();
     delay(100);
     break;
```

```
case 'd':
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
case 'e':
MorseDot();
delay(100);
break;
case 'f':
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
break;
case 'g':

MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
break;
case 'h':
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
case 'i':
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
case 'j':
```

```
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
break;
case 'k':
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
break;
case 'l':
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
case 'm':
MorseDash();
delay(100);
MorseDash();
delay(100);
break;
case 'n':
MorseDash();
delay(100);
MorseDot();
delay(100);
break;
case 'o':
MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
break;
case 'p':
```

```
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
break;
case 'q':
MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
break;
case 'r':
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
break;
case 's':
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
case 't':
MorseDash();
delay(100);
break;
case 'u':
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
break;
case 'v':
```

```
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
break;
case 'w':
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
break;
case 'x':
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
break;
case 'y':
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDash();
delay(100);
MorseDash();
delay(100);
break;
case 'z':
MorseDash();
delay(100);
MorseDash();
delay(100);
MorseDot();
delay(100);
MorseDot();
delay(100);
break;
default:
```

```
         break;
   }

}
```

VI. Bibliography:

https://www.arduino.cc/en/software
https://www.hwlibre.com/ro/siren%C4%83/