Translator din codul Morse

Autor: Remus Mihai Neatu

Introducere

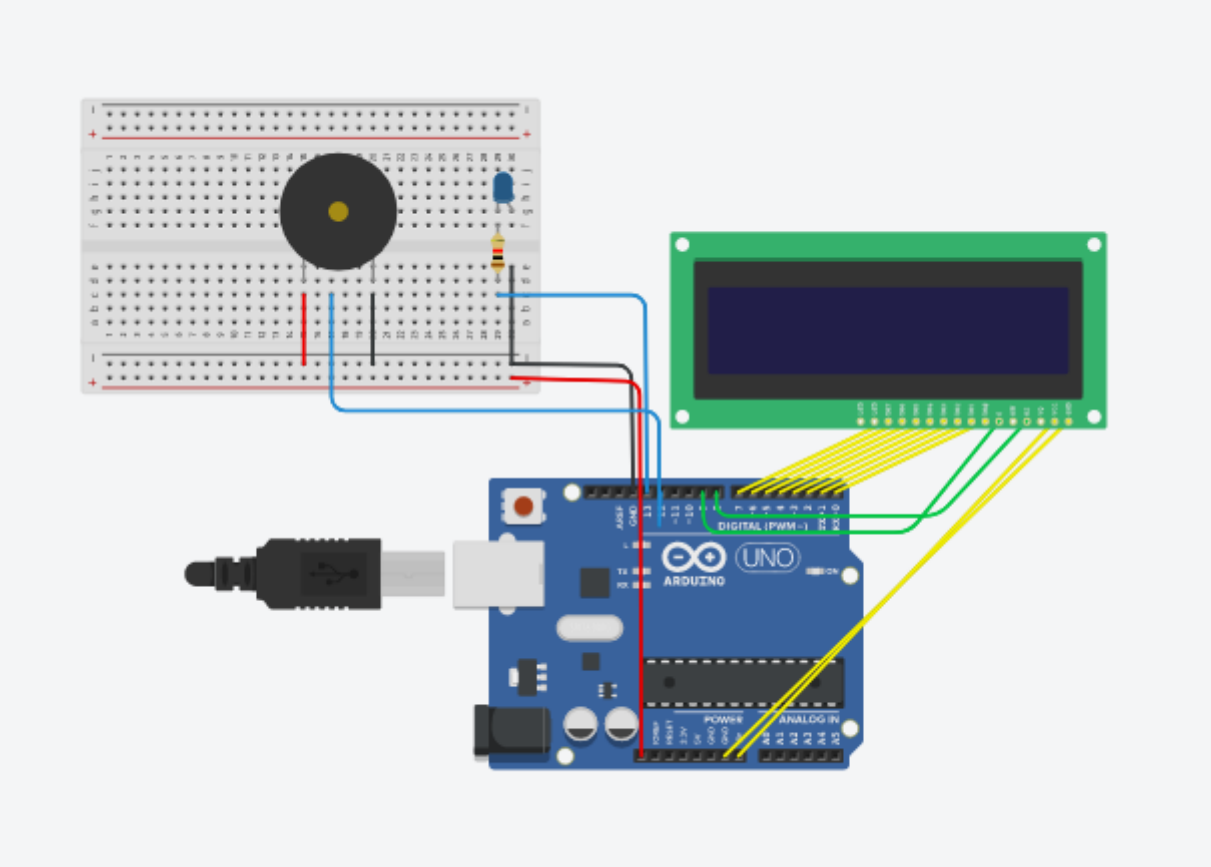
Ideea de baza a proiectului este comunicarea prin diferite cai, folosing codul Morse. Proiectul este creat pentru un grup divers de persoane, intrucat prezinta o inclusivitate foarte mare catre oamenii din categorii defavorizate. Pentru oamenii cu deficiente de vedere, comunicarea se face pe cale auditiva, folosind un buzzer, iar pentru cei cu deficiente de auz, comunicarea se realizeaza pe cale vizuala, folosind un LED. Buzzer-ul si LED-ul functioneaza concomitent.

Hardware Design

Piese:

- Placa Arduino UNO
- Breadboard
- Fire mama-tata
- Fire mama-mama
- Ecran LCD
- Buzzer
- Butoane
- LED

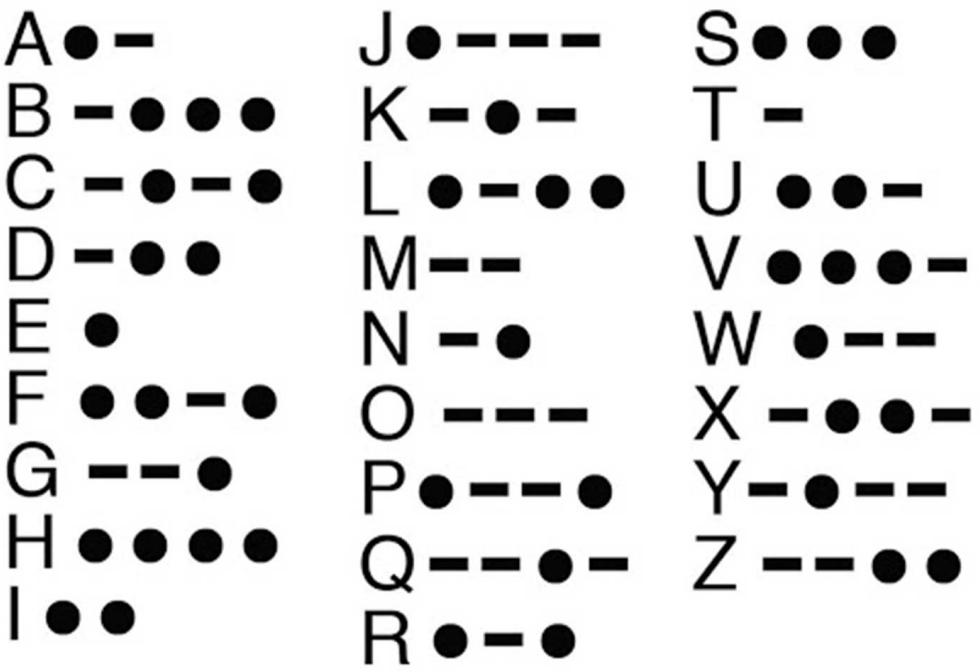
Schema electrica:



Descriere generală

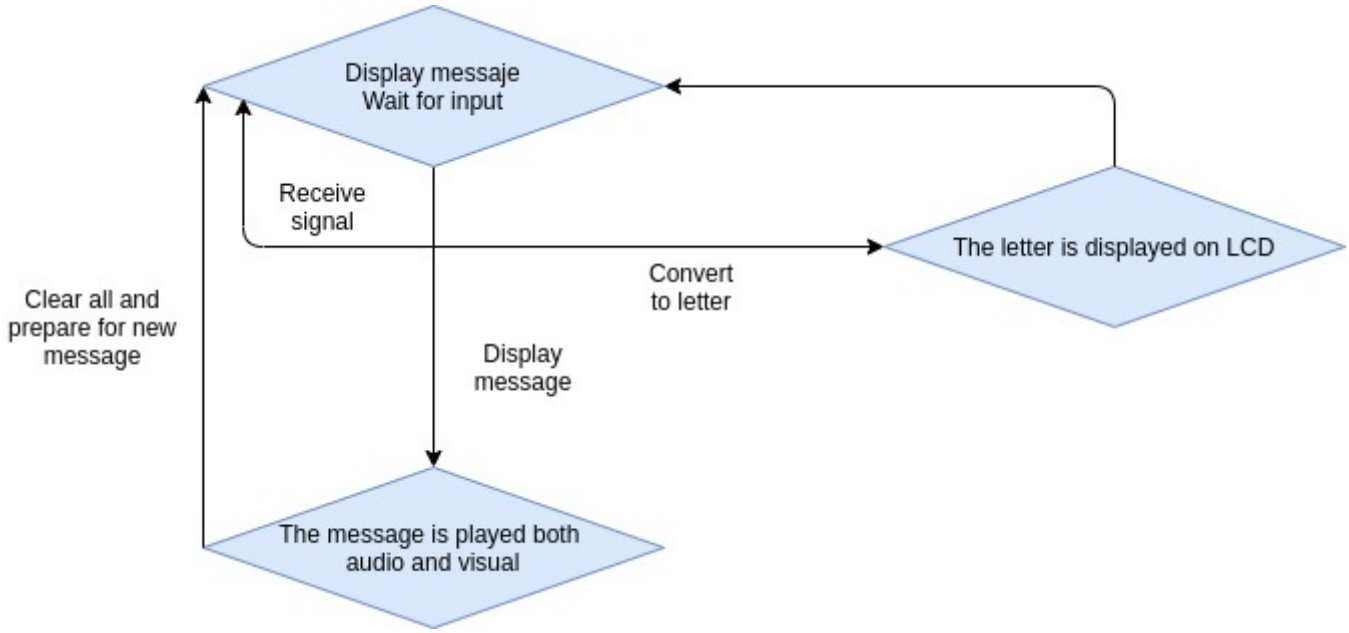
Circuitul capreaza semnalele morse prin ajutorul butoanelor de pe LCD shield. O apasare scurta de buton se traduce intr-un semnal scurt (.), iar o apasare cu mentinere a butonului se treaduce intr-un semnal lung (-).

Semnalele se transmit folosind butonul RIGHT de pe tastatura. Imediat, semnalul Morse este afisat pe ecranul LCD. Atunci cand se apasa pe butonul LEFT, caracterele Morse sunt transformate in litera (inclusiv in afisarea pe LCD). Se pot introduce codari pentru litere si pentru caracterul space (—-). Atunci cand este apasat butonul SELECT, tot mesajul ce a fost construit pana in acel moment este redat in Morse, folosing buzzer-ul si LED-ul. De asemenea, mesajul integral tradus in Morse, va aparea pe ecran, miscandu-se spre stanga, astfel incat primul caracter reprezinta mereu semnalul ce este redat la momentul respectiv.



Software Design

Schema bloc:



Cod:

```
#include <LiquidCrystal.h>

// select the pins used on the LCD panel
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

// define some values used by the panel and buttons
int lcd_key     = 0;
```

```
int adc_key_in  = 0;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   36
#define btnSELECT 4
#define btnNONE   5

#define BUZZER_PIN 12
#define LED_PIN 13

char currentChar[16], bigWord[16], buzzerMorse[100];
int charIndex = 0, wordIndex = 0, buzzerIndex = 0;
int isReady = 0, isSelect = 0;

// read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);      // read the value from the sensor
 // my buttons when read are centered at these valies: 0, 144, 329, 504,
741
 // we add approx 50 to those values and check to see if we are close
 if (adc_key_in > 1500) return btnNONE; // We make this the 1st option for
speed reasons since it will be the most likely result
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 195)  return btnUP;
 if (adc_key_in < 380)  return btnDOWN;
 if (adc_key_in < 500)  return btnLEFT;
 if (adc_key_in < 700)  return btnSELECT;
 return btnNONE;  // when all others fail, return this...
}

void silence() {
  digitalWrite(BUZZER_PIN, HIGH);
}

void buzz() {
  digitalWrite(BUZZER_PIN, LOW);
}

void led_on() {
  digitalWrite(LED_PIN, HIGH);
}

void led_of() {
  digitalWrite(LED_PIN, LOW);
}

void print_word() {
  lcd.setCursor(0,1);            // move to the begining of the second line
```

```
  for (int i = 0; i < wordIndex; ++i) {
    lcd.print(bigWord[i]);
  }
}

void print_char() {
  lcd.setCursor(wordIndex,1);            // move to the end of the word

  for (int i = 0; i < charIndex; ++i) {
    lcd.print(currentChar[i]);
  }
}



void print_morse(int start) {
  lcd.setCursor(0,1);            // move to the begining of the second line

  for (int i = start; i < buzzerIndex; ++i) {
    lcd.print(buzzerMorse[i]);
  }
}

int get_signal_duration() {
  unsigned long start = millis();

  while (lcd_key == btnRIGHT) {
    lcd_key = read_LCD_buttons();
  }

  unsigned long finish = millis();
  return finish - start;
}

char get_char_from_signal() {
  unsigned long duration = get_signal_duration();

  if (duration > 500) {
    return '-';
  }

  return '.';
}

char morse1() {
  if (currentChar[0] == '.') {
    return 'E';
  }

  return 'T';
}
```

```
char morse2() {
  if (currentChar[0] == '.' && currentChar[1] == '.') {
    return 'I';
  }
  if (currentChar[0] == '.' && currentChar[1] == '-') {
    return 'A';
  }
  if (currentChar[0] == '-' && currentChar[1] == '.') {
    return 'N';
  }
  if (currentChar[0] == '-' && currentChar[1] == '-') {
    return 'M';
  }
}

char morse3() {
  if (currentChar[0] == '.' && currentChar[1] == '.' && currentChar[2] ==
'.') {
    return 'S';
  } else if (currentChar[0] == '.' && currentChar[1] == '.' && currentChar[
2] == '-') {
    return 'U';
  } else if (currentChar[0] == '.' && currentChar[1] == '-' && currentChar[
2] == '.') {
    return 'R';
  } else if (currentChar[0] == '.' && currentChar[1] == '-' && currentChar[
2] == '-') {
    return 'W';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '.') {
    return 'D';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '-') {
    return 'K';
  } else if (currentChar[0] == '-' && currentChar[1] == '-' && currentChar[
2] == '.') {
    return 'G';
  } else if (currentChar[0] == '-' && currentChar[1] == '-' && currentChar[
2] == '-') {
    return 'O';
  }

}

char morse4() {
  if (currentChar[0] == '.' && currentChar[1] == '.' && currentChar[2] ==
'.' && currentChar[3] == '.') {
    return 'H';
  } else if (currentChar[0] == '.' && currentChar[1] == '.' && currentChar[
2] == '.' && currentChar[3] == '-') {
    return 'V';
```

```
  } else if (currentChar[0] == '.' && currentChar[1] == '.' && currentChar[
2] == '-' && currentChar[3] == '.') {
    return 'F';
  } else if (currentChar[0] == '.' && currentChar[1] == '-' && currentChar[
2] == '.' && currentChar[3] == '.') {
    return 'L';
  } else if (currentChar[0] == '.' && currentChar[1] == '-' && currentChar[
2] == '-' && currentChar[3] == '.') {
    return 'P';
  } else if (currentChar[0] == '.' && currentChar[1] == '-' && currentChar[
2] == '-' && currentChar[3] == '-') {
    return 'J';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '.' && currentChar[3] == '.') {
    return 'B';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '.' && currentChar[3] == '-') {
    return 'X';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '-' && currentChar[3] == '.') {
    return 'C';
  } else if (currentChar[0] == '-' && currentChar[1] == '.' && currentChar[
2] == '-' && currentChar[3] == '-') {
    return 'Y';
  } else if (currentChar[0] == '-' && currentChar[1] == '-' && currentChar[
2] == '.' && currentChar[3] == '.') {
    return 'Z';
  } else if (currentChar[0] == '-' && currentChar[1] == '-' && currentChar[
2] == '.' && currentChar[3] == '-') {
    return 'Q';
  } else if (currentChar[0] == '-' && currentChar[1] == '-' && currentChar[
2] == '-' && currentChar[3] == '-') {
    return ' ';
  }
}

char morse_to_char() {
  switch (charIndex) {
    case 1: {
        return morse1();
    }
    case 2: {
        char c = morse2();
        return c;
    }
    case 3: {
        return morse3() ;
    }
    case 4: {
        return morse4();
    }
```

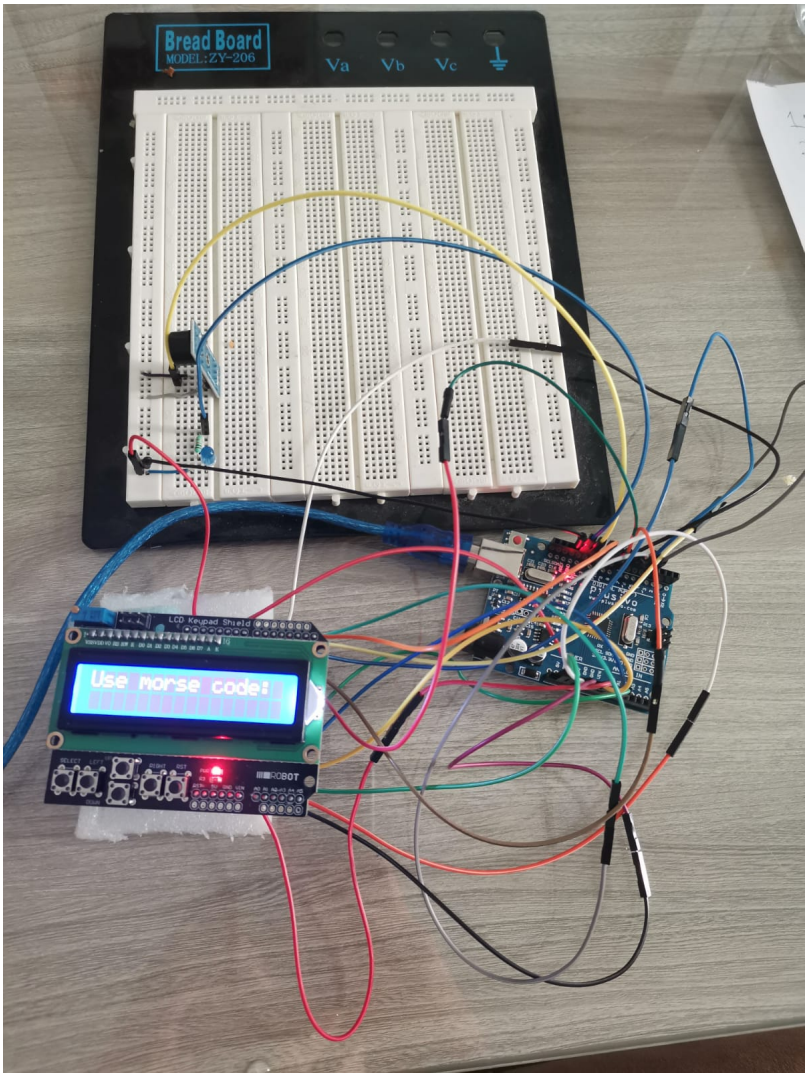
```
    default: {
      return ' ';
    }
  }
}

char get_letter_from_morse() {
  char sol = morse_to_char();

  charIndex = 0;
  lcd.setCursor(wordIndex,1);
  lcd.print("                ");
  return sol;
}

void prepare() {
  lcd.setCursor(0,1);
  lcd.print("                ");
  lcd.setCursor(0,0);
  lcd.print("Track the buzzer!");
}

void morse_to_buzzer() {
  delay(1500);


  for (int i = 0; i < buzzerIndex; ++i) {
    prepare();
    print_morse(i);

    if (buzzerMorse[i] == '.') {
      buzz();
      led_on();
      delay(300);
      silence();
      led_of();
    } else if (buzzerMorse[i] == '-') {
      buzz();
      led_on();
      delay(900);
      silence();
      led_of();
    } else {
      delay(400);
    }

    delay (200);
  }
}

void cleanup() {
```

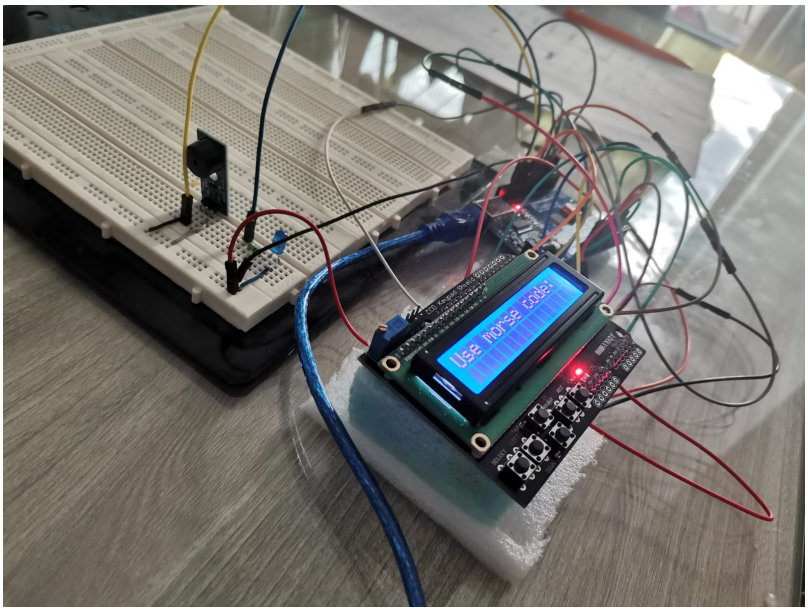
```
  lcd.setCursor(0,0);
  lcd.print("Use morse code:   ");
  lcd.setCursor(0,1);
  lcd.print("                ");
  charIndex = 0;
  wordIndex = 0;
  buzzerIndex = 0;
}

void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  lcd.begin(16, 2);              // start the library
  lcd.setCursor(0,0);
  lcd.print("Use morse code:");
}

void loop()
{
  silence();
  lcd_key = read_LCD_buttons();  // read the buttons

  print_word();
  print_char();

  switch (lcd_key)               // depending on which button was pushed,
we perform an action
  {
   case btnRIGHT:
      {
        isReady = 0;
        isSelect = 0;
        char sig = get_char_from_signal();
        currentChar[charIndex++] = sig;
        buzzerMorse[buzzerIndex++] = sig;
        break;
      }
   case btnLEFT:
      {
        isSelect = 0;
        if (!isReady) {
          isReady = 1;
          buzzerMorse[buzzerIndex++] = ' ';
          bigWord[wordIndex++] = get_letter_from_morse();
        }

        break;
      }

   case btnSELECT:
```

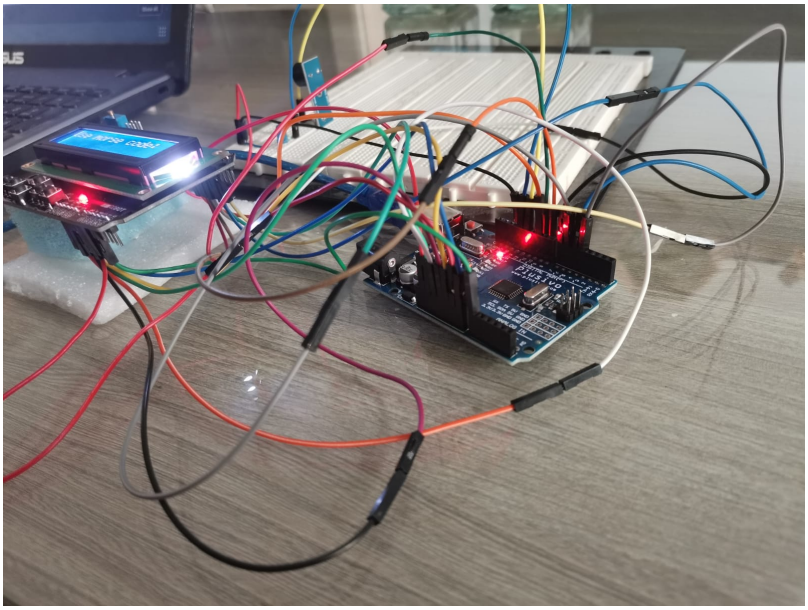
```
      {
         if (!isSelect) {
           isSelect = 1;

           prepare();
           morse_to_buzzer();
           cleanup();
         }
        break;
      }
    case btnNONE:
      {
        break;
      }
  }
}
```

Rezultate si Experiment

Youtube Link


Bread Board
MODEL: ZY-206
Va
Vb
Vc
LCD Keypad Shield
Use morse code!
ROBOT


Use morse code!



Download

331ca_neaturemusmihai_proiectpm.zip

Bibliografie/Resurse

Link [1]

Link [2]

Link [3]

Link [4]

Link [5]

Link [6]

Export to PDF