**Модул „Въведение в Операционни Системи**

**и Вградени Системи“**

**Разработка на Проект:**

**Декодер на Морзов Код**

**Изготвили: Калоян Гуцев и Микаел Ерамян**

**Съдържание:**

1. **Какво представлява и как работи декодера**
2. **Блокова схема на декодера**
3. **Електрическа схема на декодера**
4. **Списък на съставните части**
5. **Сорс Код и описание**
6. **Заключение и функционалност**
7. **Линк към проекта в Tinkercad**

**Какво представлява и как работи декодера**

Морзовият декодер представлява вградена система, която служи за превръщане на морзови код във букви и тяхното извеждане на дисплей. Тази система се състои от хардуерна и софтуерна част, като хардуерната част служи за въвеждането на морзовия код и изписването на преведения текст, а софтуерната част превежда кода.

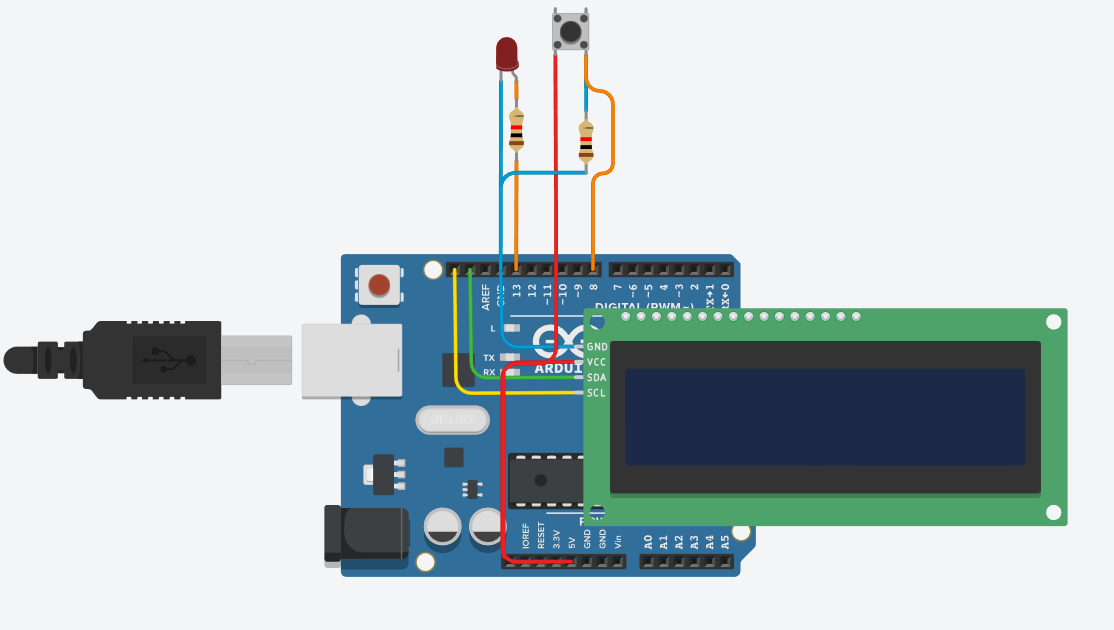
Хардуерната част се състои от компонентите бутон, светодиод (LED) и LCD дисплей. Софтуерната част се състои от 182 линии код на C++. Когато бутонът се натисне, програмата измерва продължителността на натискането. В зависимост от тази продължителност, се разпознава дали е била изпратена точка или тире в морзовия код. Ако продължителността е под 300 милисекунди, това се смята за точка, а ако е между 300 и 1000 милисекунди - за тире.

Ако продължителността на натискането е между 1000 и 2000 милисекунди, програмата търси буквата, която съответства на декодирания морзов код, и я извежда на LCD дисплея. В случай, че продължителността на натискането е по-голяма от 2000 милисекунди, на дисплея се извежда интервал.

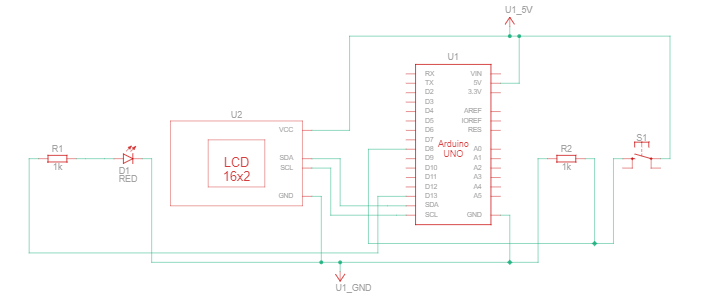
Този процес се повтаря непрекъснато, което позволява на програмата да декодира поредица от морзови символи и да ги извежда на дисплея.

Програмата е създадена в Tinkercad.

**Блокова схема на декодера**



**Електрическа схема на декодера**



**Списък на съставните части**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Име | Брой | Компонент | |  |
| U1 | 1 | Arduino Uno R3 | |  |
| U2 | 1 | MCP23008-based, 32 LCD 16 x 2 (I2C) | | |
| D1 | 1 | Red LED |  |  |
| R1, R2 | 2 | 1 kΩ Resistor | |  |
| S1 | 1 | Pushbutton | |  |

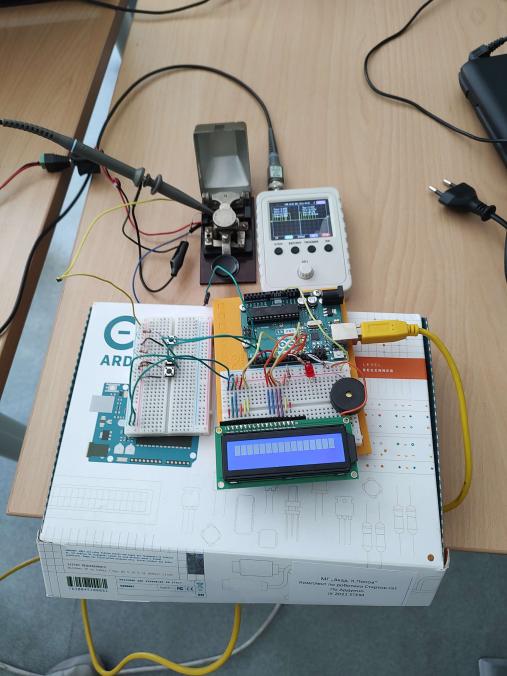
**U1** – Ардуино

**U2**- LCD дисплей

**D1** – червен LED светодиод

**R1, R2** – резистори със съпротивление 1 kΩ

**S1** – бутон



**Сорс Код**

|  |
| --- |
| #include <Adafruit\_LiquidCrystal.h> // Include the Adafruit LCD library  using namespace std;  int led1 = 13; // Pin number for LED  int button1 = 8; // Pin number for button  int buttonLast = LOW; // Previous button state  Adafruit\_LiquidCrystal lcd\_1(0); // Create an instance of the LCD display  char letter[1000]; // Array to store Morse code letter  void setup()  {  Serial.begin(9600); // Initialize serial communication for debugging    lcd\_1.begin(16, 2); // Initialize the LCD display with 16 columns and 2 rows  pinMode(led1, OUTPUT); // Set LED pin as output  pinMode(button1, INPUT); // Set button pin as input  digitalWrite(led1, LOW); // Turn off the LED initially  }  void buttonPressed()  {  int duration = 0; // Variable to store the duration of the button press    digitalWrite(led1, LOW); // Turn off the LED    // While the button is pressed, turn on the LED, increment the duration, and delay for 1 millisecond  while (digitalRead(button1) == HIGH)  {  digitalWrite(led1, HIGH);  duration++;  delay(1);  }    if (duration != 0) {  if (duration < 300) {  // Dot  Serial.print(".");  strcat(letter, ".");  Serial.print(duration);  }  else if (duration >= 300 && duration < 1000)  {  // Dash  Serial.print("-");  strcat(letter, "-");  }  else if (duration >= 1000 && duration < 2000)  {  searchLetters();  Serial.print("/");  Serial.print(duration);  }  else  {  lcd\_1.print(' ');  Serial.print(" ");  Serial.print(duration);  }  }    duration = 0; // Reset the duration after processing  }  void searchLetters()  {  // Decode the letter and print it to the LCD  if (strcmp(letter, ".-") == 0)  {  lcd\_1.print('a');  }  else if (strcmp(letter, "-...") == 0)  {  lcd\_1.print('b');  }  else if (strcmp(letter, "-.-.") == 0)  {  lcd\_1.print('c');  }  else if (strcmp(letter, "-..") == 0)  {  lcd\_1.print('d');  }  else if (strcmp(letter, ".") == 0)  {  lcd\_1.print('e');  }  else if (strcmp(letter, "..-.") == 0)  {  lcd\_1.print('f');  }  else if (strcmp(letter, "--.") == 0)  {  lcd\_1.print('g');  }  else if (strcmp(letter, "....") == 0)  {  lcd\_1.print('h');  }  else if (strcmp(letter, "..") == 0)  {  lcd\_1.print('i');  }  else if (strcmp(letter, ".---") == 0)  {  lcd\_1.print('j');  }  else if (strcmp(letter, "-.-") == 0)  {  lcd\_1.print('k');  }  else if (strcmp(letter, ".-..") == 0)  {  lcd\_1.print('l');  }  else if (strcmp(letter, "--") == 0)  {  lcd\_1.print('m');  }  else if (strcmp(letter, "-.") == 0)  {  lcd\_1.print('n');  }  else if (strcmp(letter, "---") == 0)  {  lcd\_1.print('o');  }  else if (strcmp(letter, ".--.") == 0)  {  lcd\_1.print('p');  }  else if (strcmp(letter, "--.-") == 0)  {  lcd\_1.print('q');  }  else if (strcmp(letter, ".-.") == 0)  {  lcd\_1.print('r');  }  else if (strcmp(letter, "...") == 0)  {  lcd\_1.print('s');  }  else if (strcmp(letter, "-") == 0)  {  lcd\_1.print('t');  }  else if (strcmp(letter, "..-") == 0)  {  lcd\_1.print('u');  }  else if (strcmp(letter, "...-") == 0)  {  lcd\_1.print('v');  }  else if (strcmp(letter, ".--") == 0)  {  lcd\_1.print('w');  }  else if (strcmp(letter, "-..-") == 0)  {  lcd\_1.print('x');  }  else if (strcmp(letter, "-.--") == 0)  {  lcd\_1.print('y');  }  else if (strcmp(letter, "--..") == 0)  {  lcd\_1.print('z');  }    letter[0] = '\0'; // Reset the letter after searching  }  void loop()  {  buttonPressed(); // Call the buttonPressed function repeatedly  } |

Кода използва библиотеката **Adafruit\_LiquidCrystal.h** за контролиране на LCD дисплея. Декларират се глобалните променливи: позициите на бутона, светодиода, размера на екрана и масив за запазване на въведените по-късно точки и тирета. В метода setup. Функцията **pinMode** в този код се използва за конфигуриране на определен пин като входен (**INPUT**) или изходен (**OUTPUT**).

Когато определен пин е конфигуриран като входен (**INPUT**), програмата може да чете сигнали от външни източници, като например състоянието на бутона. Когато пинът е конфигуриран като изходен (**OUTPUT**), програмата може да управлява свързаното с него устройство, като например светодиода. Това означава, че pinMode позволява на програмата да зададе режима на работа на пиновете в съответствие с нуждите на хардуерните компоненти, с които взаимодейства.

Функцията **digitalWrite** в този код се използва за задаване на състоянието на определен пин. Когато се използва **digitalWrite** с пин, конфигуриран като изходен (**OUTPUT**), може да се зададе стойност **HIGH** (включен) или **LOW** (изключен) на пина. Когато пинът е зададен като **HIGH**, това означава, че пинът се извежда на високо ниво със зададеното напрежение (5V в случая на Arduino Uno). Това може да управлява свързаните устройства, като например светодиода, който ще светне. Когато пинът е зададен като LOW, това означава, че пинът се извежда на ниско ниво, или се изключва. В случая на светодиода, той ще бъде изключен и спрял да свети.

Така digitalWrite позволява на програмата да управлява състоянието на пиновете, контролирайки свързаните хардуерни компоненти и изпълнявайки нужните действия върху тях.

Функцията **buttonPressed()** в този код се използва за обработка на натискането на бутона и измерване на продължителността на натискането. При извикване на **buttonPressed(),** програмата проверява дали бутона е натиснат. Ако бутона е натиснат (**HIGH** състояние), програмата стартира измерването на продължителността на натискането и включва светодиода. Докато бутона е натиснат, програмата продължава да увеличава продължителността и забавя за 1 милисекунда (с **delay(1)**). Това се използва за определяне на дължината на сигнала (кратък или дълъг), базирано на времето на натискане на бутона. След като бутона бъде отпуснат (**LOW** състояние), измерването на продължителността приключва. Ако продължителността е различна от нула, програмата разпознава символа в зависимост от продължителността: като точка (.), тире(-), разстояние или край на буквата.

След това програмата нулира продължителността и се завърта, готова да приеме следващо натискане на бутона и да продължи с декодирането на морзовия код.

Функцията **searchLetters()** в този код се използва за декодиране на получената последователност от символи в морзов код и извеждане на съответния символ на LCD дисплея.

Когато се извика **searchLetters(),** програмата проверява последователността от символи, съхранена в масива **letter**, и сравнява я с възможните символи в морзовия азбучен речник. Използвайки оператора **strcmp(),** програмата сравнява последователността от символи в **letter** с всяка възможна буква или символ в речника. Когато открие съвпадение, тя извежда съответния символ на LCD дисплея с помощта на **lcd\_1.print().** След това програмата нулира масива **letter**, за да е готова да декодира следваща поредица от символи.

По този начин програмата декодира входния морзов код и извежда съответните символи на LCD дисплея, което прави процеса на комуникация чрез морзов код разбираем и лесно възприемлив.

Функцията **loop()** в този код представлява основната безкрайна циклична част на програмата. В този случай, **loop()** извиква функцията **buttonPressed(),** която обработва натискането на бутона и измерва продължителността на натискането. След приключване на **buttonPressed(),** контролът се връща обратно към **loop().** Тъй като **loop()** е безкрайна циклична функция, тя се повтаря постоянно, позволявайки програмата да продължи да обработва натисканията на бутона и декодира морзовия код, докато Arduino е включена.

Това позволява на програмата да работи в постоянен режим, откривайки и обработвайки нови натискания на бутона и декодирайки последователности от морзови символи в реално време.

**Заключение и функционалност**

Тази програма предлага няколко функционалности за ползване от потребителите:

1. Декодиране на морзовия код: Потребителите могат да въведат морзов код, като използват бутона на свързаното устройство. Програмата ще декодира последователността от точки и тирета и ще изведе съответния символ на LCD дисплея.
2. Извеждане на символи на LCD дисплея: Декодираните символи се показват на LCD дисплея, който е свързан с Arduino. Това позволява на потребителите да виждат прочетените символи от морзовия код на удобен и четим начин.
3. Интерактивност с устройството: Потребителите могат да взаимодействат с програмата, като натискат бутона и след това наблюдават декодираните символи на LCD дисплея. Това осигурява интерактивен и забавен начин за комуникация чрез морзов код.
4. Гъвкавост на програмата: Речникът с морзови символи може да бъде разширен или променен според нуждите на потребителя. Ако потребителите желаят да добавят или променят символи, те могат да го направят, като редактират функцията searchLetters() и добавят съответните символи и техните морзови представяния.

Тези функционалности правят програмата полезна за хора, които се интересуват от морзов код и искат да превърнат входен морзов код в разбираем текст. Освен това, програмата може да бъде адаптирана и разширена според конкретните нужди и предпочитания на потребителите.

**Линк към проекта в TinkerCad**