Politechnika Rzeszowska
Wydział Elektrotechniki i Informatyki

Podstawy Elektroniki

Stacja nadająca sygnał w kodzie Morse’a drogą dźwiękową oraz świetlną oparta o moduł Arduino

Autor: Mariusz Bigos

1. Wstęp

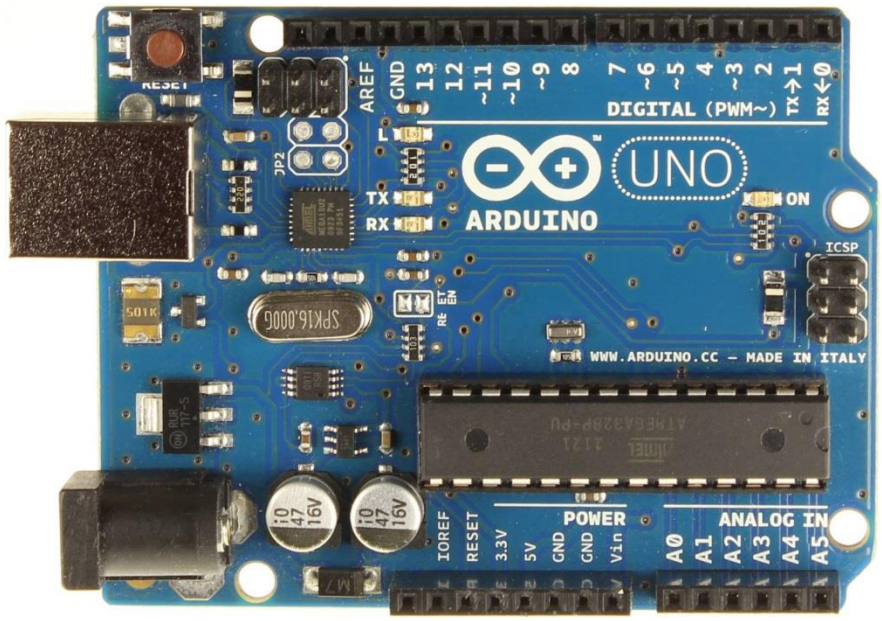
Celem ćwiczenia jest budowa stacji nadającej sygnał w kodzie Morse'a. Stacja powinna realizować nadawanie 2 drogami. Świetlną poprzez zastosowaną diodę LED oraz dźwiękową poprzez buzzer, który również może być zastąpiony innym głośnikiem. Priorytetem przy budowie urządzenia była funkcjonalność oraz łatwość w użyciu nawet dla osób nie znających się na cyfrowych układach elektronicznych. Jako podstawa została użyta płytka Arduino Uno, która dzięki swej niskiej cenie oraz wystarczającej funkcjonalności dla tego projektu daje możliwość zbudowania takiego urządzenia małym kosztem każdemu chętnemu.

Praktycznym zastosowaniem urządzenia mogła by być nauka kodu Morse'a, a jeszcze ciekawszym wykorzystanie nadajnika aby skonstruować odbiornik takiego sygnału aby 2 urządzenia mogły przenosić informacje na odległość bez konieczności udziału człowieka w procesie tłumaczenia.

Największy problem przy tworzeniu stacji było zaprogramowanie klawiatury działającej na wzór tych z klasycznych telefonów komórkowych. Klawiatura ta umożliwia wpisanie 1 z 3 różnych liter za pomocą 1 przycisku używając go odpowiednią ilość razy.

2. Elementy użyte do stworzenia układu

Arduino Uno



Jest to płytka prototypowa służąca do budowy obwodów elektronicznych. Posiada programowalne złącza zarówno analogowe w liczbie 6 jak i 14 złącz cyfrowych. Użyta w projekcie wersja jest oparta na procesorze ATmega328 taktowanym 16MHz. Zastosowane pamięci to FLASH 32kB, SRAM 2KB oraz EEPROM 1kB. Nie są to największe pojemności i nie wystarczają przy większych projektach z bardziej rozbudowanym kodem jednak do prostych naukowych czy prototypowych rozwiązań sprawdzają się bardzo dobrze. Układ posiada również kilka diod LED min. do sygnalizowania o stanie komunikacji z komputerem czy 1 programowalną diodę dostępną dla użytkownika. Zasilanie może odbywać się poprzez port USB typu B bądź wejście na zasilacz DC 7-12V.

Aby zaprogramować moduł Arduino potrzebna jest podstawowa wiedza na temat języka używanego do programowania tego mikrokontrolera. Dzięki temu, że jest on oparty na popularnych językach C/C++ to nauka przebiega bardzo szybko. To tworzenia i edytowania projektów, a

następnie kompilowania kodu i przesyłania na urządzenie służy Arduino IDE, które jest dostępne na stronie producenta.
https://www.arduino.cc/en/main/software

Buzzer

Jest to sygnalizator akustyczny sterowany elektrycznie wydający charakterystyczne brzęczenie przez co można się spotkać z nazwą brzęczyk. W projekcie zastosowano buzzer aktywny czyli posiadający generator.

Dioda LED

Dioda półprzewodnikowa emitująca promieniowanie w zakresie światła widzialnego dla ludzkiego oka. Prąd płynący od anody do katody przepływa przez chip półprzewodnikowy gdzie emitowane jest światło widzialne.
Symbol diody LED został zaprezentowany poniżej.

Rezystor (220 Ω)

Jest to element bierny w układach elektronicznych. Jego dołączenie do układu powoduje spadek napięcia dzięki czemu możemy ograniczyć prąd aby np. nie spalić diody LED czy innego elementu pracującego w określonym zakresie prądu. Przy przepływie prądu przez opornik wytwarza się ciepło.

Przewody połączeniowe

W układzie zostały użyte przewody połączeniowe męsko-męskie oraz męsko-żeńskie. Dzięki nim połączenie pomiędzy urządzeniami a płytką

stykową oraz płytką stykową a modułem Arduino jest prostsze i szybsze w realizacji.

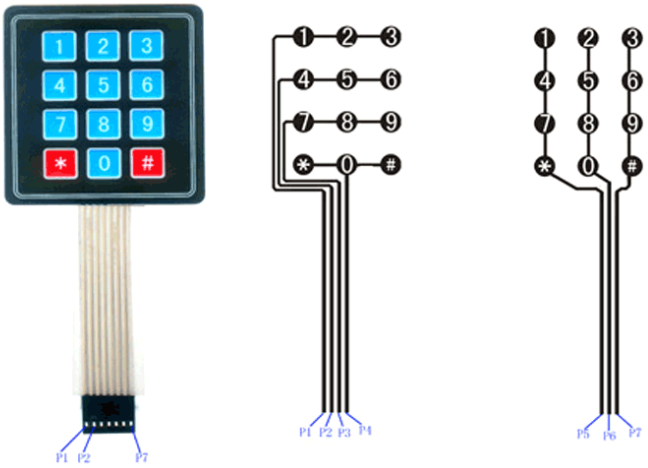
Płytka stykowa

Płytka stykowa posiada wejścia połączone pionowo oraz 4 linie wejść połączonych ze sobą poziomo. Ułatwia ona tworzenie prototypów urządzeń bez konieczności lutowania oraz jest niezastąpiona przy tworzeniu projektów naukowym z komponentów, które znajdą jeszcze inne zastosowanie w przyszłości.

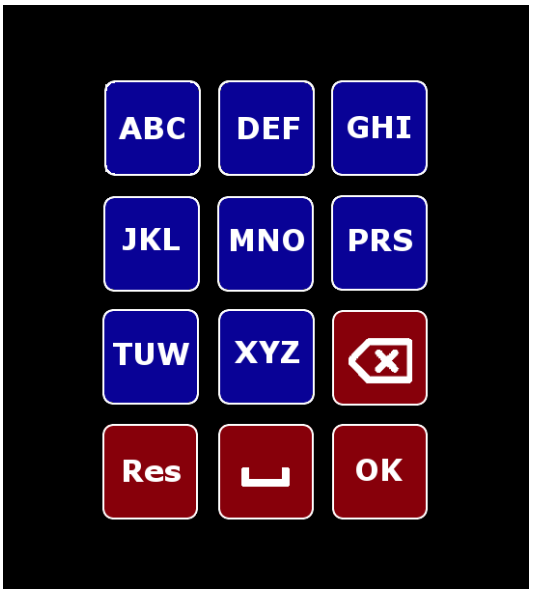
Wyświetlacz LCD z konwerterem I2C

Użyty wyświetlacz ciekłokrystaliczny czyli LCD charakteryzuje się wielkością 16x2 co pozwala na raz zapisać 32 znaki. Użyty konwerter I2C daje możliwość oszczędzenia pinów na module Arduino przez to, że wykorzystuje tylko 4 z nich. Dzięki temu można zastosować więcej przydatnych peryferii.

Klawiatura membranowa 4x3 z nakładką graficzną



Klawiatura membranowa jest funkcjonalnym jak i bardzo prostym w budowie urządzeniem. Posiada połączone przewodami kolumny oraz wiersze. Przy naciśnięciu przycisku zostają zwarte blaszki i wysyłany jest sygnał, w której kolumnie i, w którym wierszu został naciśnięty przycisk. Połączenie tych 2 danych pozwala nam na identyfikację konkretnego przycisku. Z barku dostępnych na rynku rozwiązań odpowiadających moim wymaganiom sam zaprojektowałem nakładkę graficzną oraz zaprogramowałem klawiaturę tak aby działa na wzór znanej klawiatury z telefonów klasycznych. Przy pojedynczym naciśnięciu klawisza pojawia się litera. Przy ponownym naciśnięciu tego samego klawisza litera zostaje zastąpiona kolejną z tego przycisku. Jeżeli upłynęły 2 sekundy od czasu ostatniego naciśnięcia przycisku to naciśnięcie ponownie tego samego pola nie spowoduje zastąpienie obecnej litery, a pojawienie się nowej. Poniżej widoczna jest nakładka graficzna.



Kabel USB A męski – B męski

Kabel USB jest obecnie najczęściej wykorzystywanym przewodem do komunikacji z komputerem. Również twórcy modułu Arduino zdecydowali się na takie rozwiązanie dzięki czemu nie ma problemów z kompatybilnością niezależnie jaki komputer posiadamy, ponieważ każdy dzisiejszy PC posiada porty USB.

Źródło zasilania

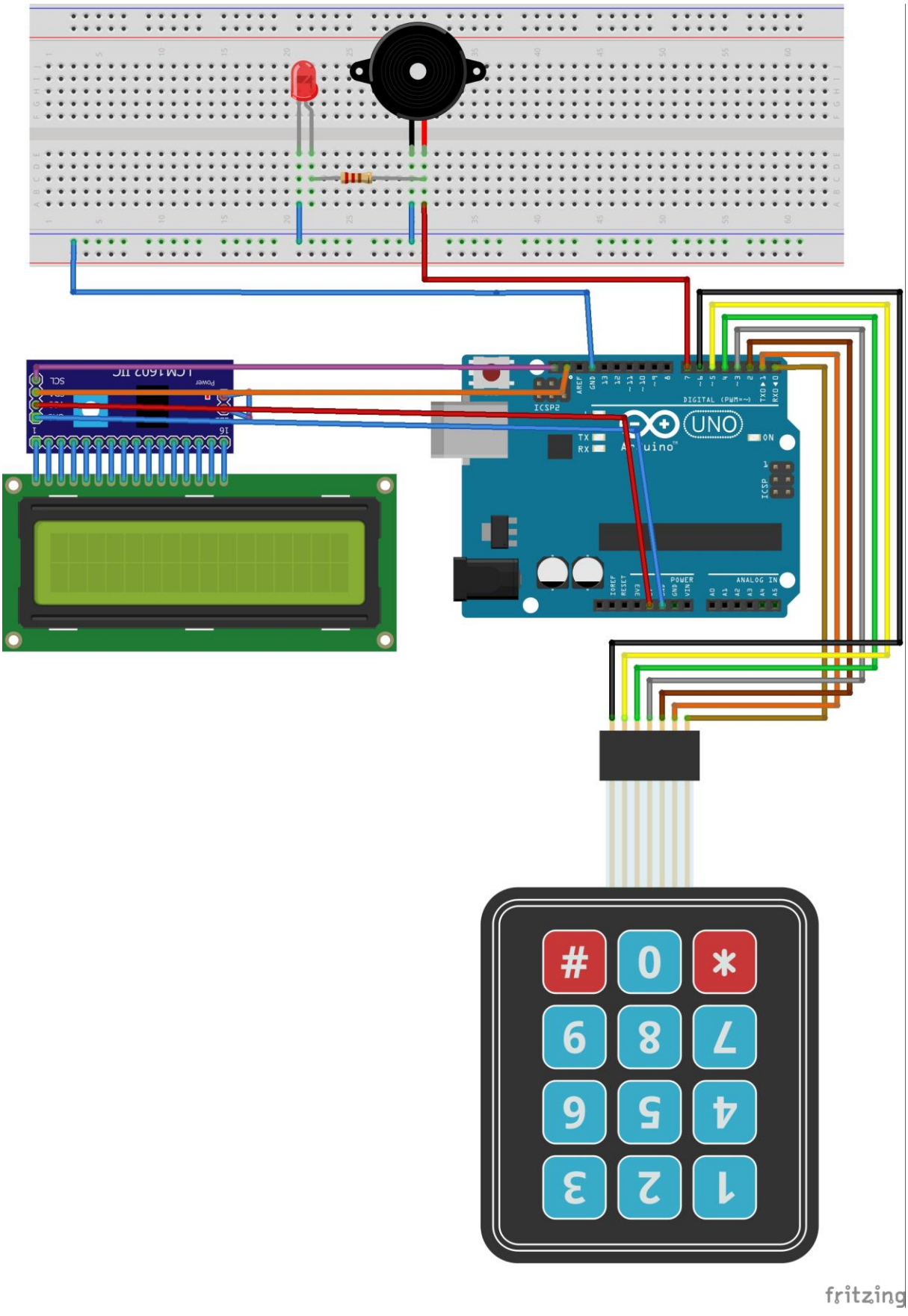
Jako opcjonalne i dające mobilność źródło zasilania została wykorzystana bateria 9V wraz ze złączem umożliwiającym podpięcie jej do modułu Arduino.

3. O kodzie Morse’a

Kod ten zawdzięcza swoją nazwę twórcy Samuelowi Morse. Na początku wieku XX kod ten był szeroko wykorzystywany głównie do telekomunikacji radiowej. Duża część komunikacji między krajami, która musiała zadziałać szybko była wykonywana właśnie przy użyciu alfabetu Morse’a. Jego podstawą są kreski i kropki. Kropka to wartość podstawowa, kreska powinna trwać 3 kropki, a przerwa pomiędzy wyrazami 7 kropek. Alfabet ten jest możliwy do zrozumienia przez człowieka, ale można też wykorzystywać specjalne urządzenia dekodujące. Poniżej spis liter używanych obecnie w krajach anglojęzycznych oraz odpowiadające im symbole alfabetu Morse’a.

A . –	J . – – –	S . . .
B – . . .	K – . –	T –
C – . – .	L . – . .	U . . –
D – . .	M – –	V . . . –
E .	N – .	W . – –
F . . – .	O – – –	X – . . –
G – – .	P . – – .	Y – . – –
H	Q – – . –	Z – – . .
I . .	R . – .	

4. Budowa układu



5. Oprogramowanie stacji

Spis użytych bibliotek oraz funkcji.

Wire.h – biblioteka potrzebna do komunikacji urządzeń z I2C

LiquidCrystal_I2C.h – biblioteka do obsługi wyświetlacza LCD
print() – wypisuje na wyświetlacz ciąg znaków podany jako parametr
setCursor(x, y) – ustawia kursor na pozycję podaną jako parametr
clear() – usuwa wszystkie znaki z wyświetlacza i ustawia kursor na początek
begin() – inicjuje pracę wyświetlacza
backlight() – ustawia podświetlenie ekranu

Keypad.h - biblioteka do obsługi klawiatury
getKey() – pobiera naciśnięty klawisz

avr/wdt.h – biblioteka wbudowana w Arduino IDE
wdt_enable() – funkcja resetująca urządzenie

Użyte funkcje nie wymagające dołączania bibliotek.
millis() – zwraca czas od uruchomienia modułu Arduino
length() – zwraca długość stringa
digitalWrite() – ustawia niskie bądź wysokie napięcie na określonym pinie
delay() – zatrzymuje pracę urządzenia na określoną liczbę milisekund
pinMode() – ustawia określony pin w określony stan np. OUTPUT albo INPUT

Kod programu:

```
// Dołączenie bibliotek
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <avr/wdt.h>

#define SIGNALPIN 7

// Tworzenie obiektu do zarządzania wyświetlaczem
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

// Tworzenie tablicy z alfabetem Morse'a
String morseArray[26] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",
```

```
"..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", " ", ".-.",
"...", "-", "..-", " ", ".--", "-..-", "-.--", "--.."};

// Towrzenie globalnych zmiennych pomocniczych
char lastKey = 'x';
int counter = 0;
int cursorPosition = -1;
int positionOfChar = -1;
unsigned long timeValue;
char wordToConvert[16];

// Ustawianie klawiatury
const byte ROWS = 4;
const byte COLS = 3;

char hexaKeys[ROWS][COLS] = {
  {'A','B','C'},
  {'D','E','F'},
  {'G','H','I'},
  {'J','K','L'}
};

char realKeys[8][3] = {
  {'A','B','C'},
  {'D','E','F'},
  {'G','H','I'},
  {'J','K','L'},
  {'M','N','O'},
  {'P','R','S'},
  {'T','U','W'},
  {'X','Y','Z'}
};

byte rowPins[ROWS] = {3, 2, 1, 0};
byte colPins[COLS] = {6, 5, 4};

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Funkcje obsługujące konkretne zdarzenia klawiatury
void printBackspace(){
  if (cursorPosition>=0) {
    lcd.setCursor(cursorPosition,1);
    lcd.print(' ');
    lcd.setCursor(cursorPosition,1);
    lastKey = 'x';
```

```
    counter = 0;
    cursorPosition--;
    wordToConvert[positionOfChar] = 'x';
    positionOfChar--;
  }
}

void resetArduino(){
  wdt_enable(WDTO_15MS);
}

void printSpace(){
  lcd.print(' ');
  lastKey = ' ';
  counter = 0;
  positionOfChar++;
  wordToConvert[positionOfChar] = ' ';
  cursorPosition++;
}

void printSelectedCharacter(char pressedKey){
  lcd.print(changeKeyToReal(pressedKey,lastKey));
  lastKey = pressedKey;
  timeValue = millis();
}

char changeKeyToReal(char pressedKey, char lastKey){
  if (pressedKey == lastKey && millis() - timeValue < 2000) {
    counter++;
    lcd.setCursor(cursorPosition,1);
  }
  else {
    counter = 0;
    cursorPosition++;
    positionOfChar++;
  }
  if (counter > 2){
    counter = counter % 3;
  }
  int asciiValueMinus65 = pressedKey - 65;
  char outputKey = realKeys[asciiValueMinus65][counter];
  wordToConvert[positionOfChar] = outputKey;
  return outputKey;
}
```

```
int howLongIsArray(char tab[]){
  int localCounter = 0;
  for(int i=0; tab[i]!='\0'; i++){
    localCounter = i;
  }
  localCounter++;
  return localCounter;
}

void sendSignal(){
  String encodedString = convertStringToMorse();
  lcd.clear();
  lcd.setCursor(3,0);
  lcd.print("Wysylanie");
  lcd.setCursor(4,1);
  lcd.print("sygnalu!");
  for(int i=0; i<encodedString.length(); i++){
    if(encodedString[i] == '.'){
      digitalWrite(SIGNALPIN, HIGH);
      delay(200);
      digitalWrite(SIGNALPIN, LOW);
      delay(300);
    }
    else if(encodedString[i] == '-'){
      digitalWrite(SIGNALPIN, HIGH);
      delay(600);
      digitalWrite(SIGNALPIN, LOW);
      delay(300);
    }
    else if(encodedString[i] == ' '){
      delay(1400);
    }
  }
  resetArduino();
}

 // Funkcja konwertująca ciąg znaków na kod Morse'a
String convertStringToMorse(){
  String stringToEncoding = "";
  for(int i=0; i < howLongIsArray(wordToConvert); i++){
    if(wordToConvert[i] != 'x'){
      stringToEncoding += morseArray[wordToConvert[i]-65];
      stringToEncoding += ' ';
    }
  }
```

```
  return stringToEncoding;
}

void setup()
{
  pinMode(SIGNALPIN,OUTPUT);
  MCUSR = 0;
  lcd.begin(16,2);
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Wpisz tekst:");
  lcd.setCursor(0,1);
}

void loop()
{
  char pressedKey = customKeypad.getKey();
    if (pressedKey){
      // Kontroler zarządzający zdarzeniami klawiatury
      switch (pressedKey - 65) {
        case 8:
          printBackspace();
          break;
        case 9:
          resetArduino();
          break;
          case 10:
          printSpace();
          break;
          case 11:
          sendSignal();
          break;
        default:
          printSelectedCharacter(pressedKey);
    }
  }
}
```

Kod (bez komentarzy) jest dostępny do pobrania w internecie:
https://pastebin.com/JX6fBm5f