Zadanie zaliczeniowe

Systemy wbudowane

Daniel Nowak
Bartłomiej Barański
27 July 2024

Grupa a2

Spis treści

Rozdział 0 Wstęp:

Celem zadania zaliczeniowego (końcowego) jest podsumowanie wiedzy zdobytej podczas zajęć z systemów wbudowanych (IV semestr). Wszystkie kody oraz nagrania, uporządkowane chronologicznie, prezentujące ich zastosowanie w rzeczywistych scenariuszach, są dostępne do obejrzenia ***tutaj.****

Celem niniejszego sprawozdania jest zaprezentowanie wyników zdobytej wiedzy z tego semestru oraz archiwizacja pracy w sposób umożliwiający odtworzenie zarówno kodu, jak i platformy w przyszłości.

Rozdział 1 Platforma Arduino - wykaz części:

Arduino Nano - R3

Ilość sztuk: 1

Arduino Nano to kompaktowa, pełna płytka oparta na mikrokontrolerze ATmega328. Nie posiada gniazda zasilania DC, zamiast tego używa kabla USB Mini-B. Pin 3.3V na płytce dostarcza napięcie tylko podczas zasilania przez USB.

LCD 16x2 I2C

Ilość sztuk: 1

LCD 16x2 I2C to mały ekran ciekłokrystaliczny, który umożliwia wyświetlanie krótkich komunikatów lub prostych grafik za pośrednictwem interfejsu I2C.

Rezystor 100 Ω

Ilość sztuk: 10

Rezystor jest pasywnym komponentem elektrycznym o dwóch końcówkach, który wprowadza opór w obwodzie elektrycznym. W obwodach elektronicznych rezystory służą do ograniczania przepływu prądu, regulacji poziomów sygnałów, dzielenia napięć, polaryzacji elementów aktywnych, zakończenia linii transmisyjnych oraz innych zastosowań.

Rezystor 10 kΩ

Ilość sztuk: 3

Rezystor to pasywny komponent elektryczny o dwóch końcówkach, który wprowadza opór w obwodzie elektrycznym. Jest używany do regulacji przepływu prądu, dostosowywania poziomów sygnałów,

dzielenia napięć, polaryzacji elementów aktywnych, zakończenia linii transmisyjnych oraz innych zastosowań.

Kabel USB Mini-B

Ilość sztuk: 1

Standardowy kabel USB Mini-B, służący do łączenia urządzeń z komputerem lub innymi urządzeniami.

Płytka stykowa (Breadboard)

Ilość sztuk: 1

Płytka stykowa pełnowymiarowa bez lutowania. Posiada 2 rozdzielone szyny zasilania, 10 kolumn i 63 wiersze. Wszystkie piny są rozmieszczone w standardowej odległości 0,1 cala.

Wyświetlacz LED linijka OSX10201-GYR1 - 10-segmentowy

10-segmentowy wyświetlacz LED linijka o kolorach 5 x zielony, 3 x żółty oraz 2 x czerwony. Może być wykorzystany np, jako wskaźnik temperatury lub poziomu cieczy.

Pakiet przewodów połączeniowych - M/M

Ilość sztuk: 2

Pakiet zawiera 20 standardowych przewodów połączeniowych męski-męski o długości 7 cali. Przewody te są używane do łączenia różnych komponentów w obwodach elektrycznych i są szczególnie wygodne w użyciu z płytką stykową.

Pakiet przewodów połączeniowych - M/F

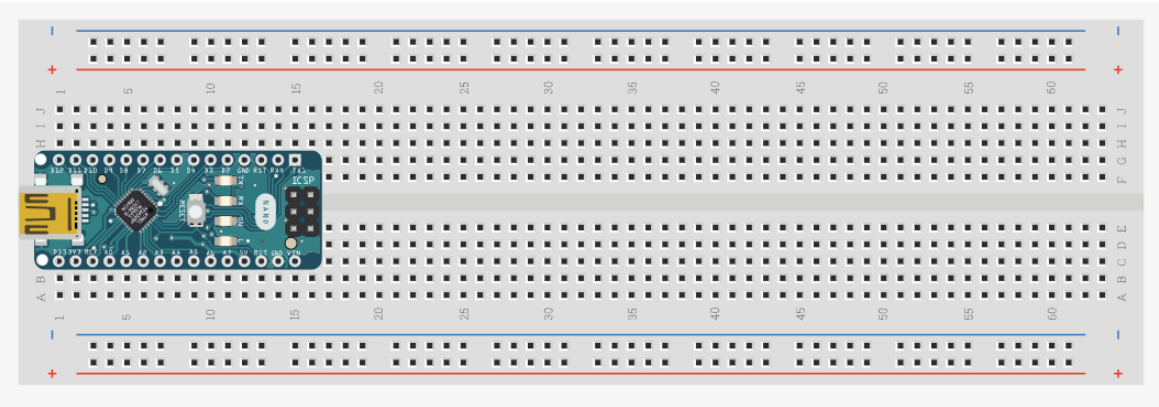
Ilość sztuk: 1

Pakiet zawiera 20 standardowych przewodów połączeniowych męski-żeński o długości 6 cali. Przewody te służą do łączenia

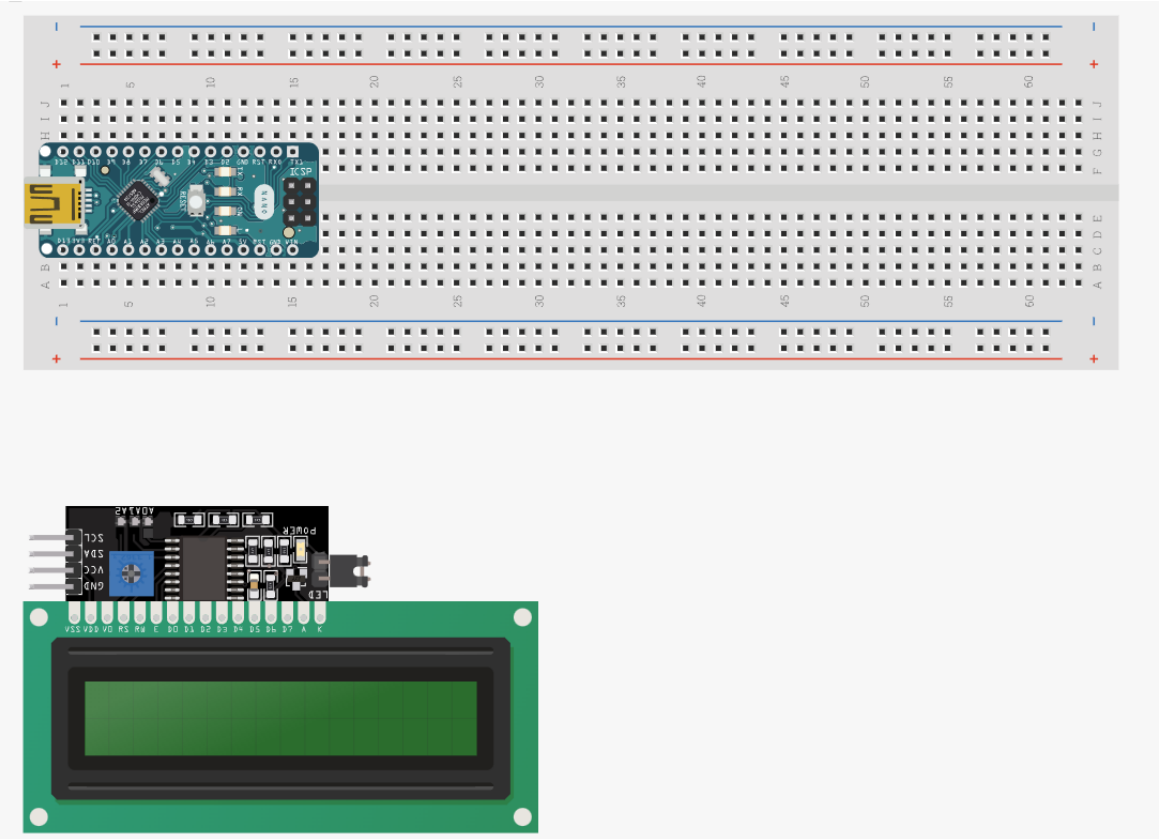
różnych komponentów w obwodach elektrycznych i są wygodne w użyciu z płytką stykową.

Rozdział 2 Instrukcja montażu:

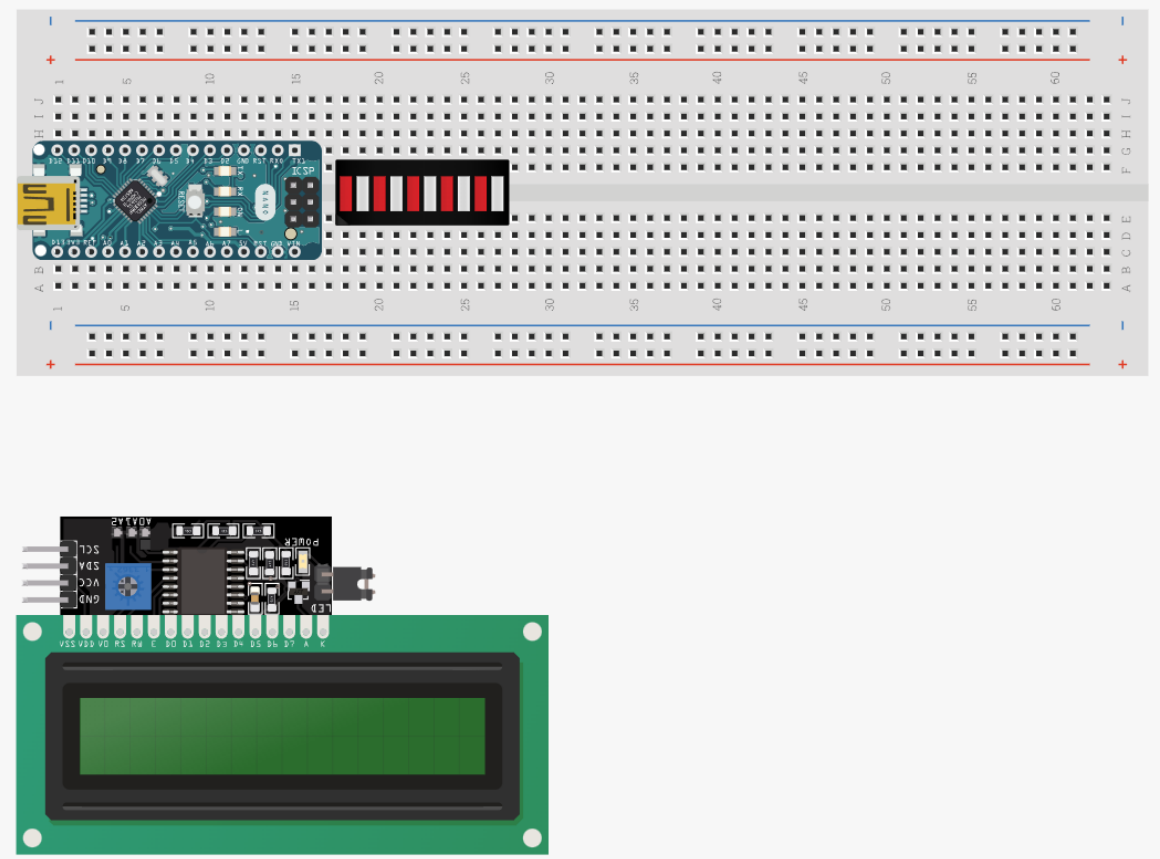
1.Umieść Arudino Nano



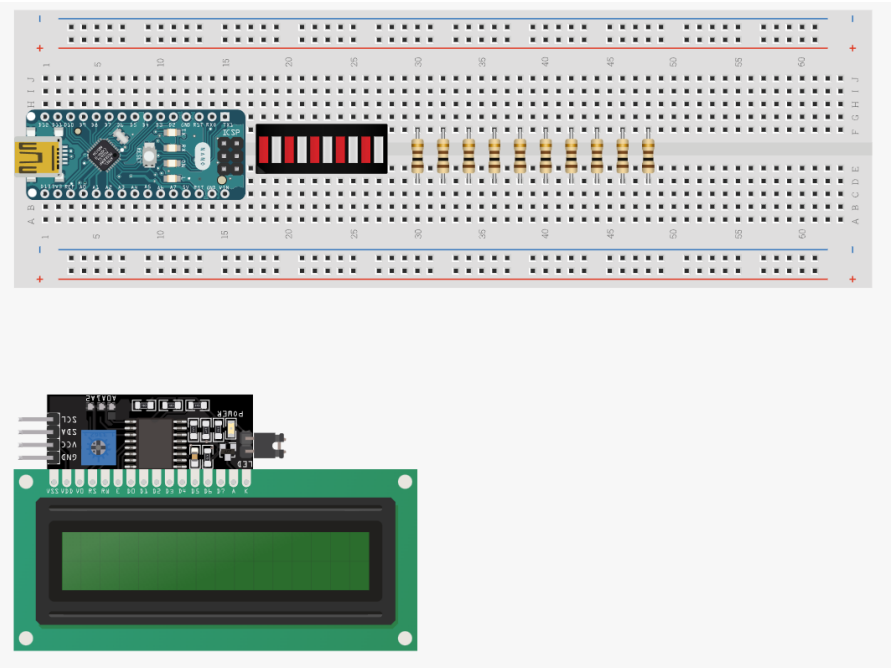
2. Umieść ekran LCD



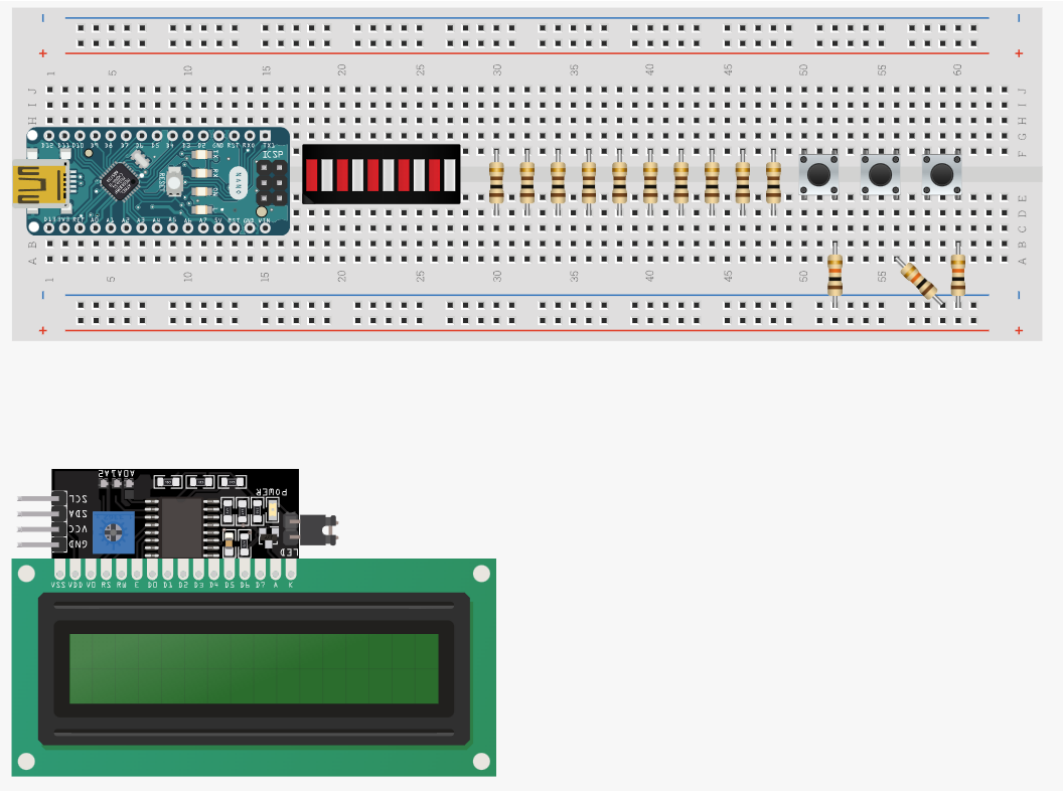
3. Umieść BAR Led



4. Umieść rezystory Place Res100O (x10)

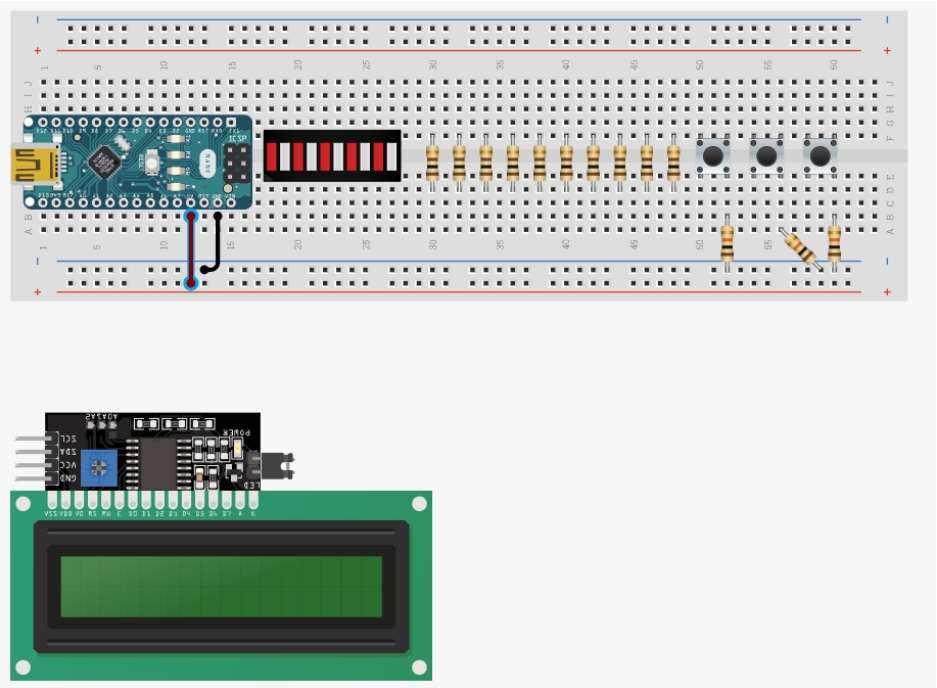


5.Umieść kolejno 3 przyciski i 3 rezystory Res10KO



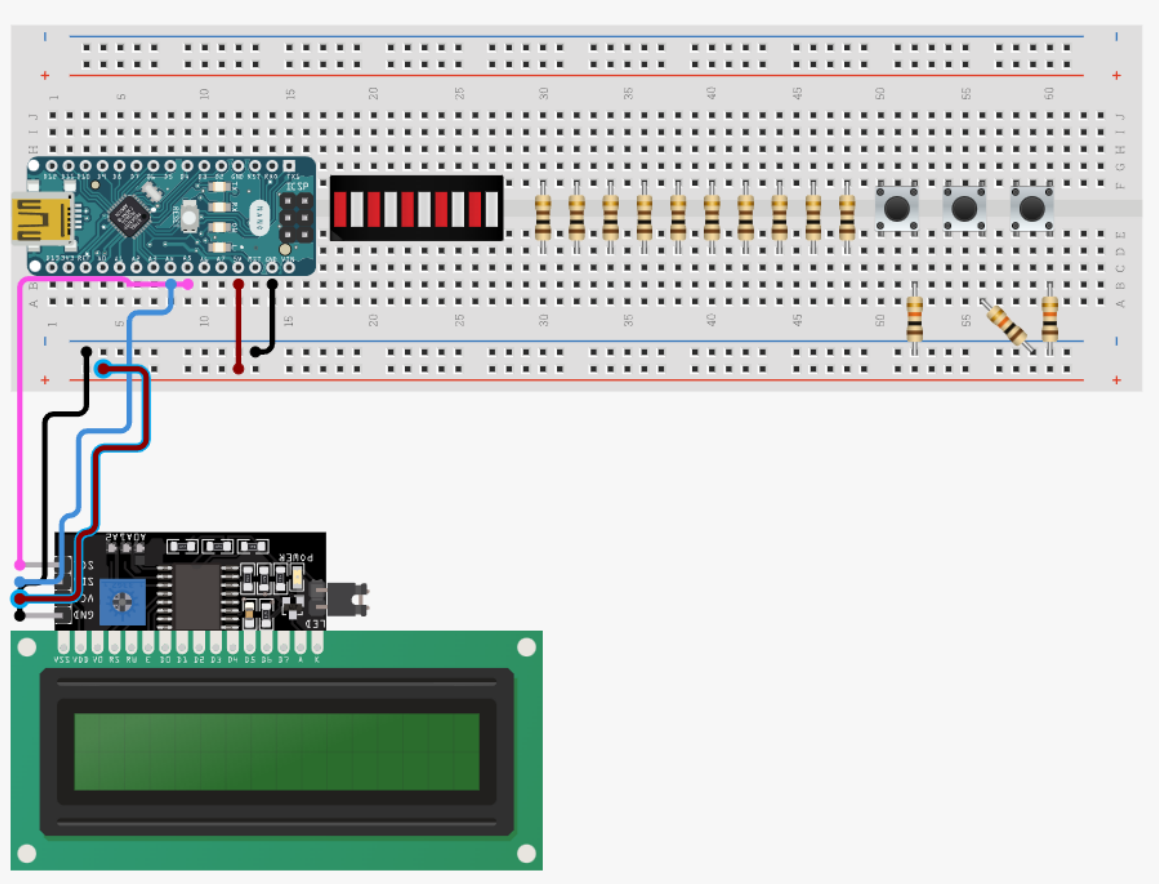
6. Podłącz Arduino Nano do:

Arduino Nano GND do GND szyny
Arduino Nano 5V do POS szyny



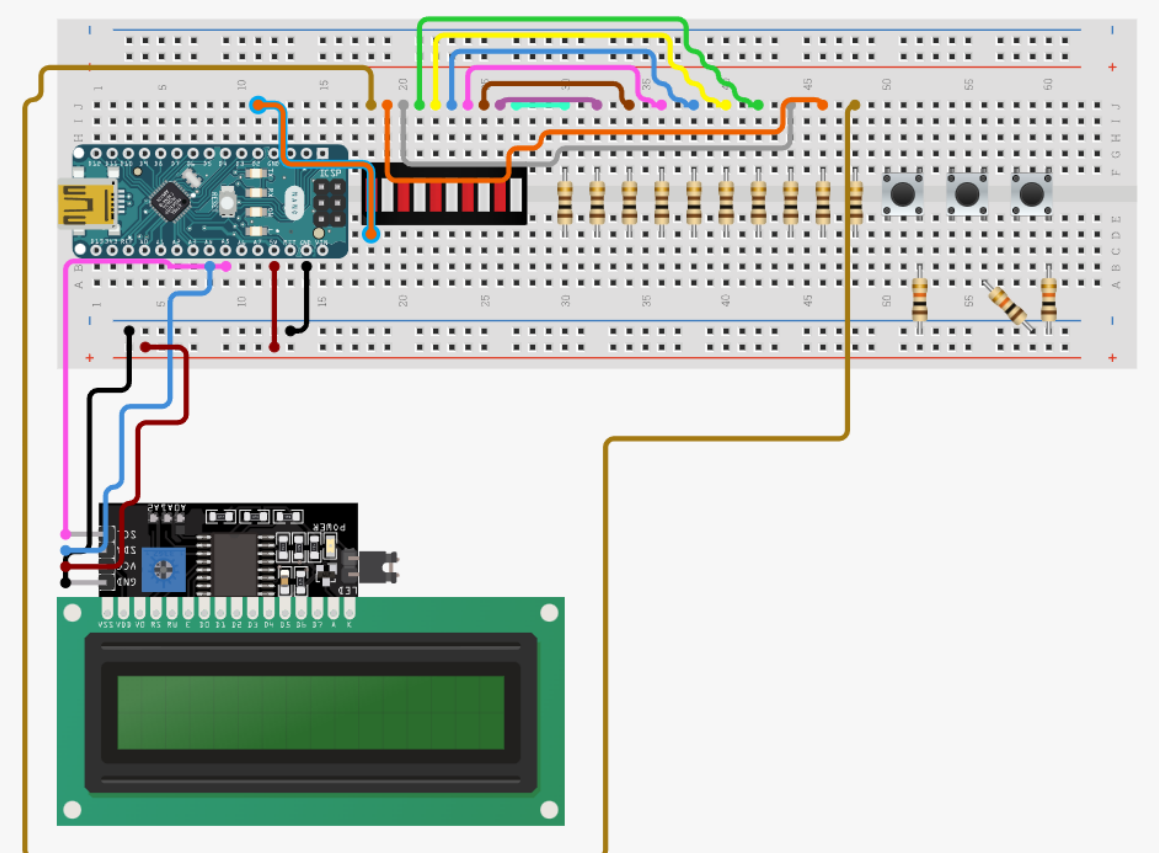
7. Podłącz LCD16X2I2C do:

LCD16X2I2C GND do GND szyny
LCD16X2I2C SCL do Arduino Nano A5
LCD16X2I2C SDA do Arduino Nano A4
LCD16X2I2C VCC do POS szyny



8. Podłącz pas LED do:

LEDBar JC do Res100Ω con0
LEDBar IC do Res100Ω con0
LEDBar HC do Res100Ω con0
LEDBar GC do Res100Ω con0
LEDBar FC do Res100Ω con0
LEDBar EC do Res100Ω con0
LEDBar DC do Res100Ω con0
LEDBar CC do Res100Ω con0
LEDBar BC do Res100Ω con0
LEDBar AC do Res100Ω con0
LEDBar AA do Arduino Nano 2



9. Podłącz pas LED do:

LEDBar BA do Arduino Nano 3

LEDBar CA do Arduino Nano 4

LEDBar DA do Arduino Nano 5

LEDBar EA do Arduino Nano 6

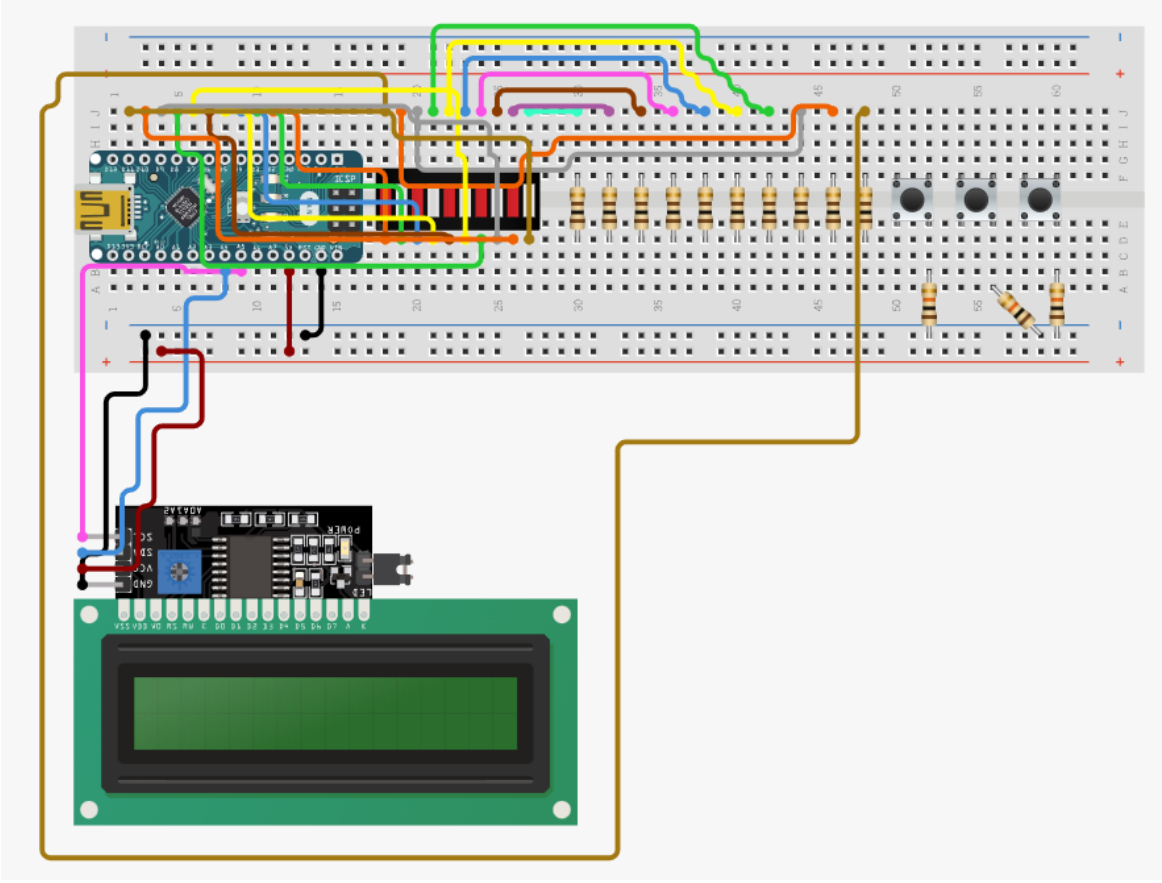
LEDBar FA do Arduino Nano 7

LEDBar GA do Arduino Nano 8

LEDBar HA do Arduino Nano 9

LEDBar IA do Arduino Nano 10

LEDBar JA do Arduino Nano 11



Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

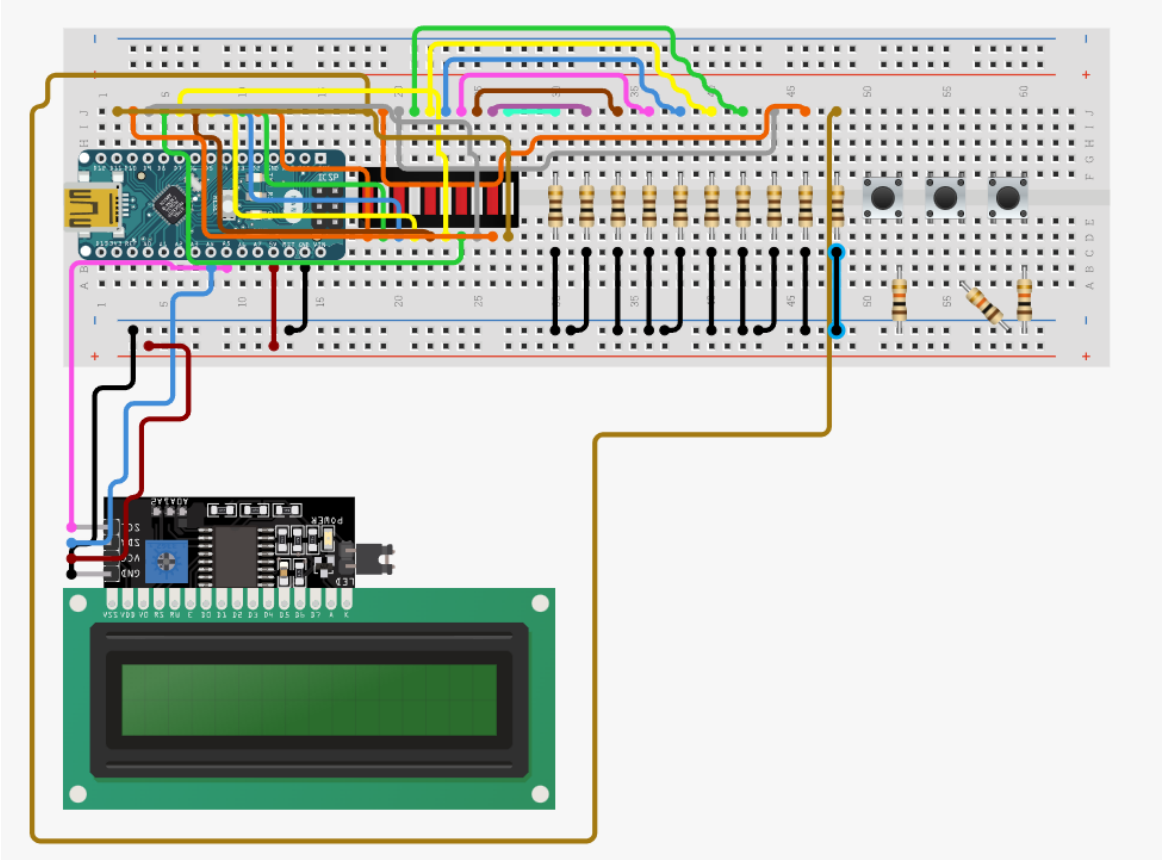
Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny

Podłącz rezystor 100 Ω

Res100Ω con1 do GND szyny



10. Podłącz:

przycisk 1:

PushButton 2 do Arduino Nano 12

PushButton 4 do szyny zasilania (POS)
przycisk 2:

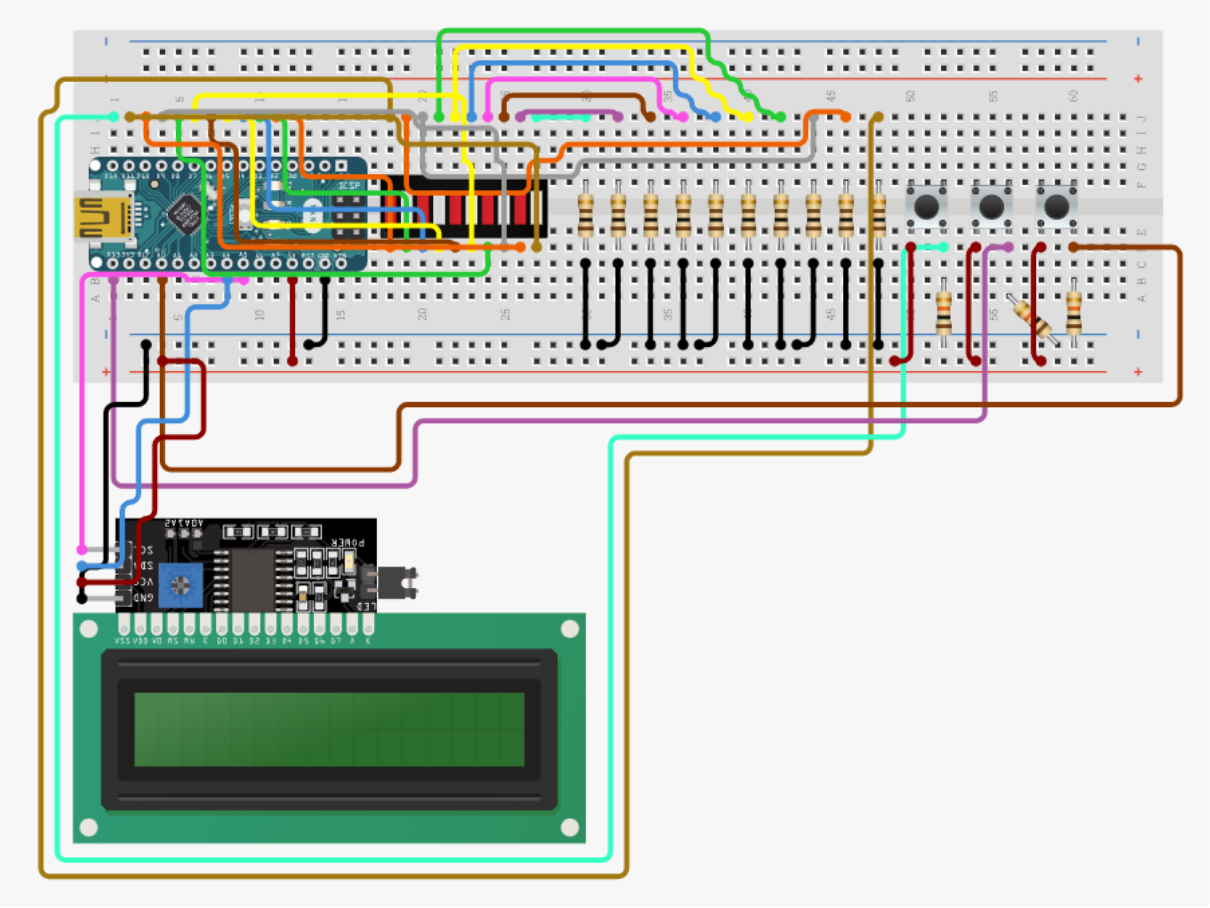
PushButton 2 do Arduino Nano 13

PushButton 4 do szyny zasilania (POS)

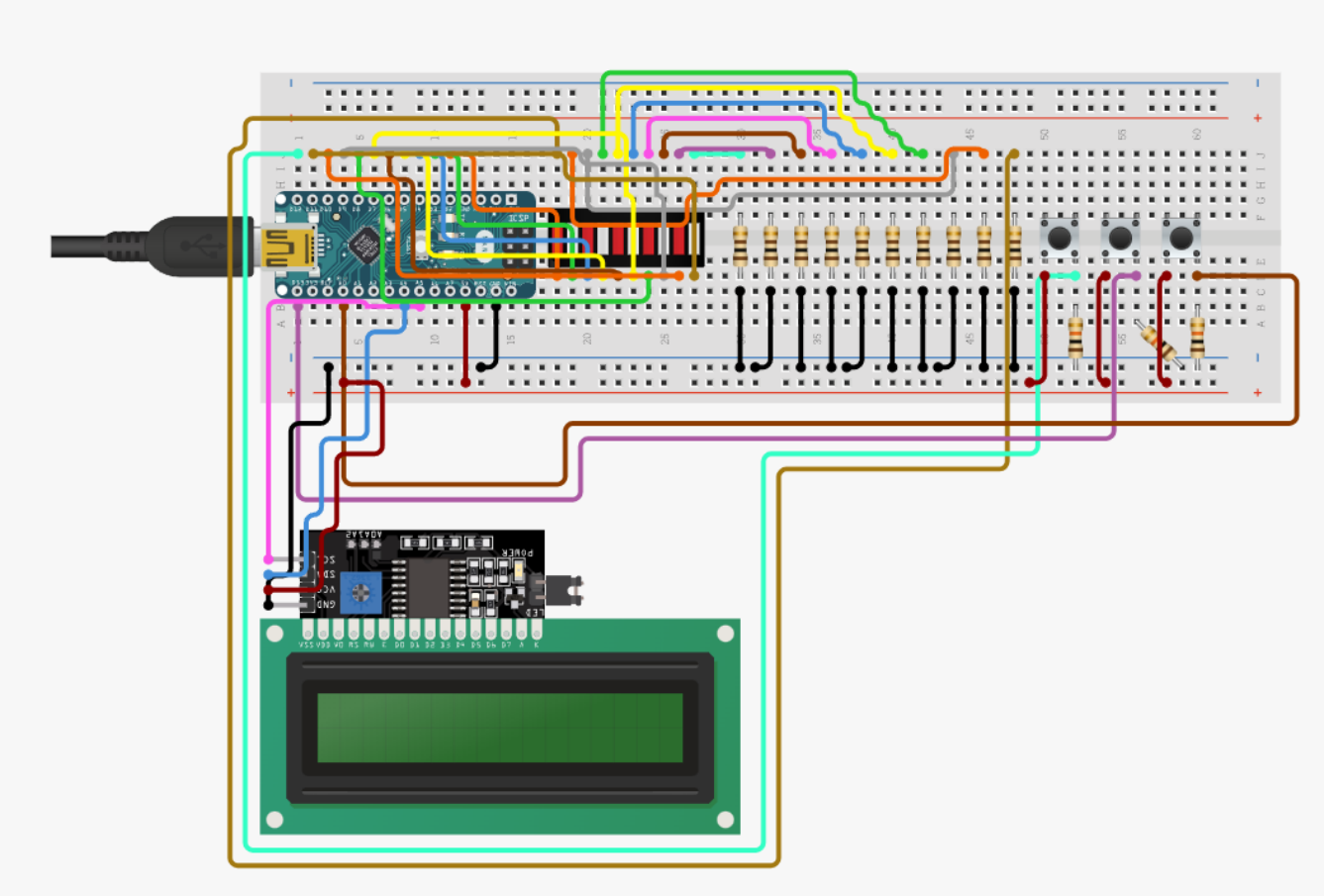
przycisk 2:

PushButton 2 do Arduino Nano A0

PushButton 4 do szyny zasilania (POS)



11. Podłącz do zasilania za pomocą kabla USB



Rozdział 3 Kod - wstęp:

Kod ma charakter edukacyjny i został zaprojektowany z myślą o nauce podstaw elektroniki oraz programowania mikrokontrolerów. Celem projektu jest ułatwienie zrozumienia działania systemów liczbowych, komunikacji I2C oraz obsługi podstawowych komponentów elektronicznych. Użytkownik może wprowadzać bity 0 lub 1 za pomocą dwóch przycisków, podczas gdy trzeci służy do resetu w przypadku pomyłki – wprowadzenia błędnego znaku lub chęci przekonwertowania kolejnej wartości. Wynik konwersji jest wyświetlany na ekranie LCD – w pierwszej linii widoczna jest wartość dziesiętna, a w drugiej linia wartość binarna. Panel LED, składający się z diod, ilustruje wartość binarną w formie świetlnej, aby użytkownik mógł łatwiej zwizualizować wynik w formie binarnej, a tym samym ćwiczyć umiejętność przeliczania wartości całkowitych między tymi dwoma systemami. Dodatkowo, kod zawiera mechanizmy obsługi błędów, takie jak wyświetlanie komunikatu, gdy liczba bitów przekracza dozwolony zakres (wymuszony przez rozmiar ekranu LCD (więcej na ten temat w rozdziale siódmym).

Rozdział 4 (kod wraz z komentarzami):

(! Kod (final.ino) można zobaczyć również *tutaj** w wygodniejszej formie, następny rozdział znajduje się na stronie:19)

```
#include <Wire.h> // Biblioteka do komunikacji I2C
#include <LCD_I2C.h> // Biblioteka do obsługi wyświetlacza LCD I2C

// Adres I2C dla wyświetlacza LCD
LCD_I2C lcd(0x3F, 16, 2); // Inicjalizacja wyświetlacza LCD z adresem 0x3F i rozmiarem 16x2

// Deklaracje pinów dla przycisków
const int buttonPin0 = 3;  // D3 - przycisk dla 0
const int buttonPin1 = 2;  // D2 - przycisk dla 1
const int buttonPinReset = A1; // A1 - przycisk reset

// Zmienna do przechowywania wartości binarnej
int binaryValue = 0; // Przechowywanie wartości binarnej
int bitCount = 0; // Licznik bitów

void setup() {
  // Konfiguracja pinów przycisków jako wejścia z wewnętrznym podciąganiem
  pinMode(buttonPin0, INPUT_PULLUP); // Konfiguracja D3 jako wejście
  pinMode(buttonPin1, INPUT_PULLUP); // Konfiguracja D2 jako wejście
  pinMode(buttonPinReset, INPUT_PULLUP); // Konfiguracja A1 jako wejście

  // Konfiguracja pinów LED jako wyjścia
  for (int i = 4; i <= 12; i++) {
    pinMode(i, OUTPUT); // Konfiguracja pinów D4 do D12 jako wyjścia
  }

  // Inicjalizacja wyświetlacza LCD
  lcd.begin(); // Inicjalizacja wyświetlacza LCD
  lcd.backlight(); // Włączenie podświetlenia wyświetlacza

  // Inicjalizacja wbudowanej diody LED
  pinMode(13, OUTPUT); // Konfiguracja D13 jako wyjście
```

```
  // Wyświetlenie początkowego stanu na LCD
  lcd.setCursor(0, 0); // Ustawienie kursora na pierwszą linię
  lcd.print("DEC: 0"); // Wyświetlenie "DEC: 0" na pierwszej linii
  lcd.setCursor(0, 1); // Ustawienie kursora na drugą linię
  lcd.print("(2)0"); // Wyświetlenie "(2)0" na drugiej linii
}

void loop() {
  // Sprawdzenie, czy przycisk 0 został naciśnięty
  if (digitalRead(buttonPin0) == LOW) { // Sprawdzenie, czy przycisk 0
jest wciśnięty
    while (digitalRead(buttonPin0) == LOW); // Czekanie na zwolnienie
przycisku
    addBit(0); // Dodanie bitu 0 do wartości binarnej
  }

  // Sprawdzenie, czy przycisk 1 został naciśnięty
  if (digitalRead(buttonPin1) == LOW) { // Sprawdzenie, czy przycisk 1
jest wciśnięty
    while (digitalRead(buttonPin1) == LOW); // Czekanie na zwolnienie
przycisku
    addBit(1); // Dodanie bitu 1 do wartości binarnej
  }

  // Sprawdzenie, czy przycisk reset został naciśnięty
  if (digitalRead(buttonPinReset) == LOW) { // Sprawdzenie, czy
przycisk reset jest wciśnięty
    while (digitalRead(buttonPinReset) == LOW); // Czekanie na
zwolnienie przycisku
    reset(); // Resetowanie wartości binarnej
  }
}

void addBit(int bit) {
  // Dodawanie bitu do wartości binarnej
  binaryValue = (binaryValue << 1) | bit; // Przesunięcie w lewo i
dodanie bitu
  bitCount++; // Zwiększenie licznika bitów
  updateDisplay(); // Aktualizacja wyświetlacza i panelu LED
}

void updateDisplay() {
  // Aktualizacja wyświetlacza LCD
```

```
  lcd.clear(); // Czyszczenie wyświetlacza
  lcd.setCursor(0, 0); // Ustawienie kursora na pierwszą linię
  lcd.print("DEC: "); // Wyświetlenie "DEC: " na pierwszej linii
  lcd.print(binaryValue); // Wyświetlenie wartości dziesiętnej

  // Konwersja wartości binarnej na string
  char binString[10]; // Tablica do przechowywania wartości binarnej
jako string
  itoa(binaryValue, binString, 2); // Konwersja wartości binarnej na
string

  // Wyświetlenie wartości binarnej na drugiej linii
  lcd.setCursor(0, 1); // Ustawienie kursora na drugą linię
  lcd.print("(2)"); // Wyświetlenie "(2)"
  lcd.print(binString); // Wyświetlenie wartości binarnej

  // Aktualizacja panelu LED
  for (int i = 0; i < 9; i++) { // Iteracja przez wszystkie piny LED
    int bit = (binaryValue >> (8 - i)) & 1; // Ekstrakcja odpowiedniego
bitu
    digitalWrite(4 + i, bit); // Ustawienie stanu diody LED
  }

  // Sprawdzenie, czy wartość przekroczyła zakres (9 bitów)
  if (bitCount >= 9) { // Sprawdzenie, czy liczba bitów przekroczyła 9
    lcd.clear(); // Czyszczenie wyświetlacza
    lcd.print("zaDuzaLiczbaBin"); // Wyświetlenie komunikatu błędu
    delay(2000); // Czekanie 2 sekundy przed resetem
    reset(); // Resetowanie wartości binarnej
  }
}

void reset() {
  // Resetowanie wartości binarnej i licznika bitów
  binaryValue = 0; // Resetowanie wartości binarnej
  bitCount = 0; // Resetowanie licznika bitów
  lcd.clear(); // Czyszczenie wyświetlacza
  lcd.setCursor(0, 0); // Ustawienie kursora na pierwszą linię
  lcd.print("DEC: 0"); // Wyświetlenie "DEC: 0" na pierwszej linii
  lcd.setCursor(0, 1); // Ustawienie kursora na drugą linię
  lcd.print("(2)0"); // Wyświetlenie "(2)0" na drugiej linii

  // Wyłączenie wszystkich diod LED
```

```
  for (int i = 4; i <= 12; i++) { // Iteracja przez wszystkie piny LED
    digitalWrite(i, LOW); // Wyłączenie diody LED
  }
}

void lightOnboard(void) {
  for (int j = 0; j < 10; j++) { // Miga wbudowaną diodą LED 10 razy
    digitalWrite(13, HIGH); // Włączenie diody LED
    delay(50); // Czekanie 50 ms
    digitalWrite(13, LOW); // Wyłączenie diody LED
  }
}
```

Rozdział 5:

Rozdział 5 znajduje się w pliku “Sprawozdanie - rozdział 5.pdf” z racji na swój rozmiar.

Rozdział 6 Omówienie problemów napotkanych podczas realizacji projektu wraz z ich rozwiązaniem:

Problem z zakresami bitów w tym projekcie wynika z ograniczonej liczby miejsc na wyświetlaczu LCD 16x2. Wyświetlacz ten może pomieścić jedynie 16 znaków w jednej linii, co oznacza, że przy próbie wyświetlenia zbyt dużej liczby bitów, na ekranie zabraknie miejsca.

Problem ten można rozwiązać za pomocą obsługi przewijania tekstu, jeżeli długość tekstu przekracza zakres wyświetlacza. Rozwiązanie problemu znajduje się w kodzie “rodzial7RozwiazanieProblemu.ino” ,
gdzie znajduje się zaktualizowana funkcja display

Rozdział 7 Podsumowanie:

Systemy wbudowane były jednym z najciekawszych i najbardziej rozwijających przedmiotów na IV semestrze moich studiów informatycznych. Dzięki nim mogłem odejść od teorii i skupić się na praktycznych rozwiązaniach, tworząc rzeczywiste platformy, a następnie je programować. Projekty, takie jak czujniki ruchu, parkowania czy potencjometr, którym po odpowiednim zakodowaniu można sterować przepustnicą w pojazdach z napędem elektrycznym, np. w hulajnogach czy skuterach, pozwoliły mi zrozumieć ich rzeczywiste zastosowanie.

Wielokrotnie w grupie napotkaliśmy sytuacje, w których kod, choć poprawnie napisany i skompilowany, nie działał prawidłowo. Często problem tkwił w fizycznej części projektu, np. błędnym podłączeniu elementów. Wspólnie z kolegą Bartłomiejem Barańskim musieliśmy dokładnie analizować zarówno kod, jak i sprzęt, aby znaleźć przyczynę problemu. Dodatkowo, pojawiały się też wyzwania związane z ograniczoną ilością pamięci urządzeń oraz przestrzenią na wyświetlanie znaków na ekranach.

Podczas ćwiczeń oraz wykładów prowadzonych przez dr. Pawła Janika, mogłem nie tylko zapoznać się z teorią, ale także zrozumieć wyzwania, jakie niosą ze sobą systemy wbudowane, zarządzanie energią, przetwarzanie sygnałów, integracja z sensorami (czujnikami ;) , oraz inne dziedziny nauk, takie jak np. mechatronika, automatyka, i robotyka. Jeden z wykładów, na którym omawiano konstrukcję samochodu elektrycznego, był moim zdaniem szczególnie interesujący. Pokazał zarówno aspekty technologiczne, jak i kwestie ekologiczne związane z tymi pojazdami.
Projekt był rozwijający i pozwolił mi oraz Panu Barańskiemu podsumować wiedzę zdobytą podczas całego semestru zajęć, zmierzyć się z napisanem kodu oraz zmontowaniem fizycznej platformy.

Rozdział 8:

* https://github.com/Danio4801/Systemy-wbudowane

https://drive.google.com/drive/folders/17Y_CqE10Q8emUEsLIPFU6E2uxEOQRN3L?usp=share_link

https://octopart.com/prt-12794-sparkfun-71808628

https://octopart.com/prt-12795-sparkfun-71808629

https://octopart.com/gs-830-global+specialties-11900592

https://octopart.com/3021009-06-qualtek-861525

https://octopart.com/cf14jt10k0-stackpole+electronics-19205381

https://octopart.com/cf14jt100r-stackpole+electronics-19205380

http://www.gearbest.com/other-accessories/pp_216639.html?wid=21

https://octopart.com/a000005-arduino-20172777

Daniel Nowak, Bartłomiej Barański Lipiec 2024