
	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych <b>Projekt SMiW</b>			
<b>Rok akademicki</b> <b>2021/2022</b>	<b>Rodzaj studiów*: SSI/NSI/NSM</b> <b>SSI</b>	<b>Numer ćwiczenia:</b> <b>41</b>	<b>Grupa</b> <b>5</b>	<b>Sekcja</b> <b>1</b>
<b>Data i godzina planowana:</b> dd/mm/rrrr - gg:mm	<b>28/02/2022-23:59</b>	<b>Prowadzący:</b> OA/JP/KT/GD/BSz/GB	<b>OA</b>	
<b>Data i godzina wykonania:</b> dd/mm/rrrr - gg:mm	<b>31/01/2022-12:45</b>			
<h1>Raport</h1>				
<b>Temat projektu:</b>  <div style="text-align: center;"> <b>Simon mówi</b>  <b>Gra dla dzieci</b> </div>				
<b>Skład sekcji:</b>	1. Kacper Nitkiewicz			

## 1. Opis projektu

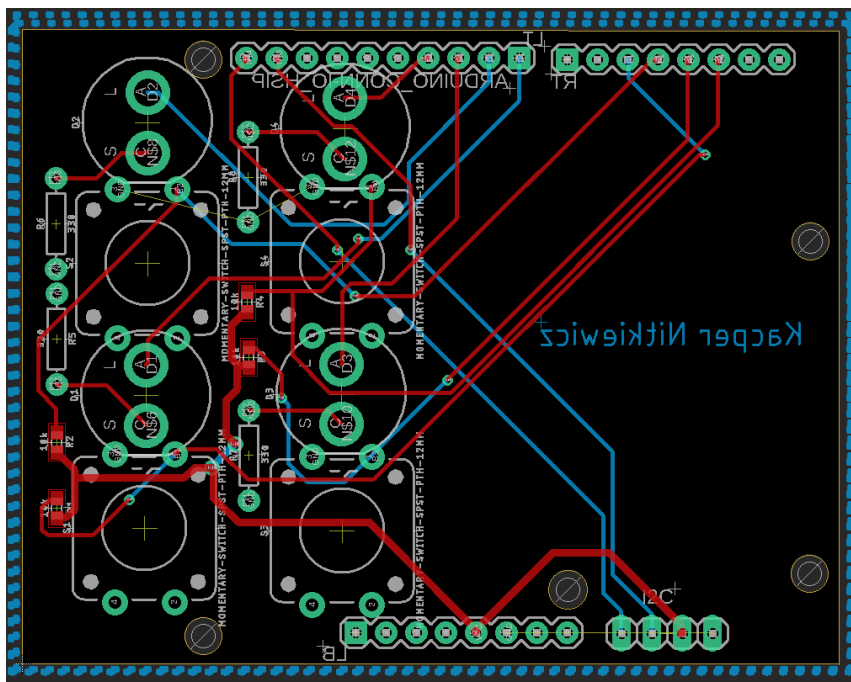
Gra Simon mówi (Simon says) polega wyborze odpowiednich przycisków w zależności od wyświetlonych diod. Wynik jest pokazywany na ekranie. Jeżeli popełnimy błąd to dostajemy informacje o błędzie. Projekt ma być wykonany na oddzielnej płytce, która może działać bez potrzeby być podpiętym do komputera.

## 2. Sposób wykonania

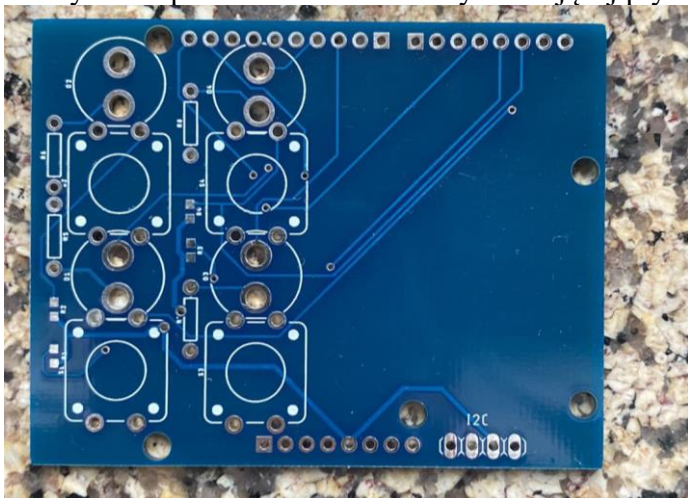
Do wykonania projektu zastosowałem płytkę Arduino Uno. Posiada ona zdolny do zaprogramowania procesor oraz możliwość dostarczenia energii do układu przez 12 V wejście Jack. Posiada guzik odpowiadający za reset układu.



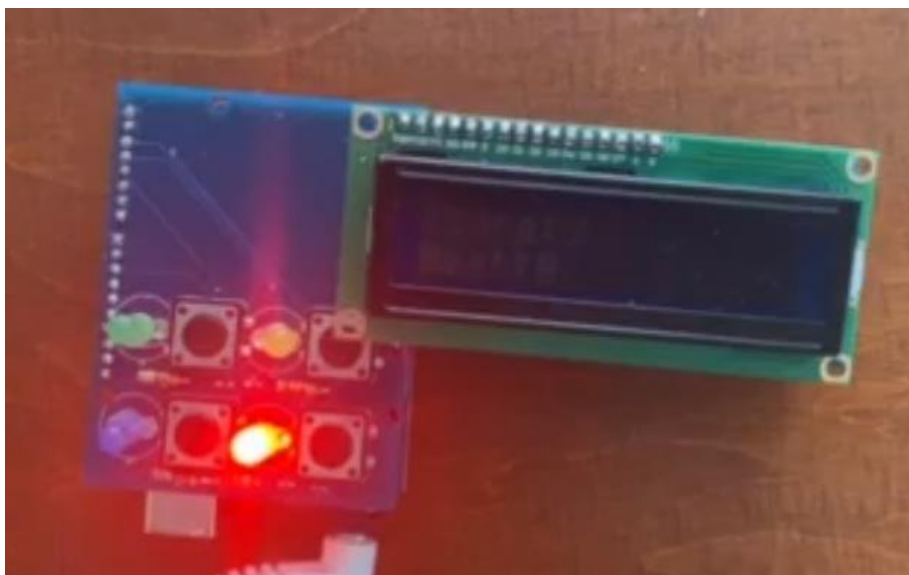
Na tą płytkę położyłem zaprojektowaną przeze mnie płytkę drukowaną, której projekt utworzyłem w aplikacji desktopowej EAGLE.



Po wysłaniu plików Gerber do firmy drukującej płytki PCB otrzymałem następującą część.



Do otrzymanej części dolutowałem gniazdo 1x4 w dolnej części oraz w górnej i dolnej części pin heady. W okrągłe miejsca zamontowałem diody, a w prostokąty – guziki. W odpowiednich miejscach przylutowałem również rezystory. Podczas lutowania okazało się, że jeden z rezystorów był uszkodzony przez co nie było obsługi jednego z guzików. Ostatecznie po zmontowaniu wyświetlacza otrzymałem następujący rezultat.



### 3. Kod źródłowy

Kolejnym zadaniem było stworzenie odpowiedniego kodu do obsługi płytki. Wykorzystałem port dla płytki Arduino, żeby napisać i przetestować kod.

```
#include <LiquidCrystal_I2C.h>
```

```
// Set the LCD address to 0x27 for a 16 chars and 2 line display  
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
#define L1 8
```

```
#define L2 9
```

```

#define L3 10
#define L4 11

#define B4 2
#define B3 3
#define B2 4
#define B1 5

bool value1 = 0;
bool value2 = 0;
bool value3 = 0;
bool value4 = 0;

//generated by program
short order[128];
short orderCount = 0;

//generated by player
short choice[128];
short choiceCount = 0;

int score = 0;
int bestScore = 0;
int errorValue = 0;

void switchDiode(int diode, int state) {
    switch (diode) {
        case 1:
            digitalWrite(L1, state);
            break;
        case 2:
            digitalWrite(L2, state);
            break;
        case 3:
            digitalWrite(L3, state);
            break;
        case 4:
            digitalWrite(L4, state);
            break;
    }
}

void showPreviousDiodes() {
    for (int i = 0; i < orderCount; i++)
    {
        switchDiode(order[i], 1);
        delay(1000);
        switchDiode(order[i], 0);
        delay(500);
    }
}

```

```
}  
}
```

```
void generateOneDiode()
```

```
{  
    //values 1-4  
    int diode = random(1, 5);  
  
    order[orderCount] = diode;  
    orderCount++;  
    switchDiode(diode, 1);  
    delay(1000);  
    switchDiode(diode, 0);  
    delay(500);  
}
```

```
int isError(int place) {  
    if (order[place] != choice[place]) {  
        errorValue = 1;  
        return 1;  
    }  
    return 0;  
}
```

```
void waitForButtonsRelease()
```

```
{  
    while (true)  
    {  
        int value = digitalRead(B1) && digitalRead(B2) && digitalRead(B3) && digitalRead(B4);  
        if (value) {  
            delay(50);  
            return;  
        }  
    }  
}
```

```
void waitForButtonClicks() {
```

```
    for (int i = 0 ; i < orderCount; i++)  
    {  
        int x = 1;  
        while (x)  
        {  
            value1 = !digitalRead(B1);  
            value2 = !digitalRead(B2);  
            value3 = !digitalRead(B3);  
            value4 = !digitalRead(B4);  
  
            if (value1) {  
                choice[choiceCount] = 1;
```

```

    }

    if (value2) {
        choice[choiceCount] = 2;
    }

    if (value3) {
        choice[choiceCount] = 3;
    }

    if (value4) {
        choice[choiceCount] = 4;
    }

    if (value1 || value2 || value3 || value4) {
        choiceCount++;
        waitForButtonsRelease();
        if (isError(choiceCount - 1))
            return;
        x = 0;
    }
}
}
}
}

```

```

void setup() {
    // put your setup code here, to run once:
    lcd.begin();
    lcd.backlight();
    pinMode(L1, OUTPUT);
    pinMode(L2, OUTPUT);
    pinMode(L3, OUTPUT);
    pinMode(L4, OUTPUT);

    pinMode(B1, INPUT);
    pinMode(B2, INPUT);
    pinMode(B3, INPUT);
    pinMode(B4, INPUT);

    randomSeed(analogRead(0));
}

```

```

void loop() {
    // put your main code here, to run repeatedly:
    lcd.clear();
    lcd.print("Score: ");
    lcd.setCursor(6, 0);
    lcd.print(score);
    lcd.setCursor(0,1);
}

```

```

lcd.print("Best:");
lcd.setCursor(5,1);
lcd.print(bestScore);

showPreviousDiodes();

generateOneDiode();

waitForButtonClicks();

choiceCount = 0;
score++;

if (errorValue == 1) {
    orderCount = 0;
    bestScore = max(score,bestScore);
    score = 0;
    errorValue = 0;
    lcd.setCursor(0, 1);
    lcd.print("Wrong Answer!");
    delay(2000);
}
}

```

## 5. Wnioski

Projekt, który wybrałem okazał się wyzwaniem. Była to mój pierwszy fizyczny projekt z płytką Arduino, choć wcześniej pracowałem już na kabelkach to sprawa z lutowaniem wyglądała zupełnie inaczej. Całkowity czas lutowania wynosił około 2 h. Wydaje mi się, że podejmując się tego wyzwania drugi raz zrobił bym to szybciej. Pisanie kodu w aplikacji Arduino to czysta przyjemność. Można sprawdzać czy wszystkie komponenty działają wysyłając dane do portu szeregowego który jest podpięty do komputera i płytki. Jednym z większych minusów jest brak podpowiedzi w przypadku wykorzystywania bibliotek do obsługi wyświetlacza LCD oraz automatycznej korekcji błędów.