# Karta projektu zaliczeniowego

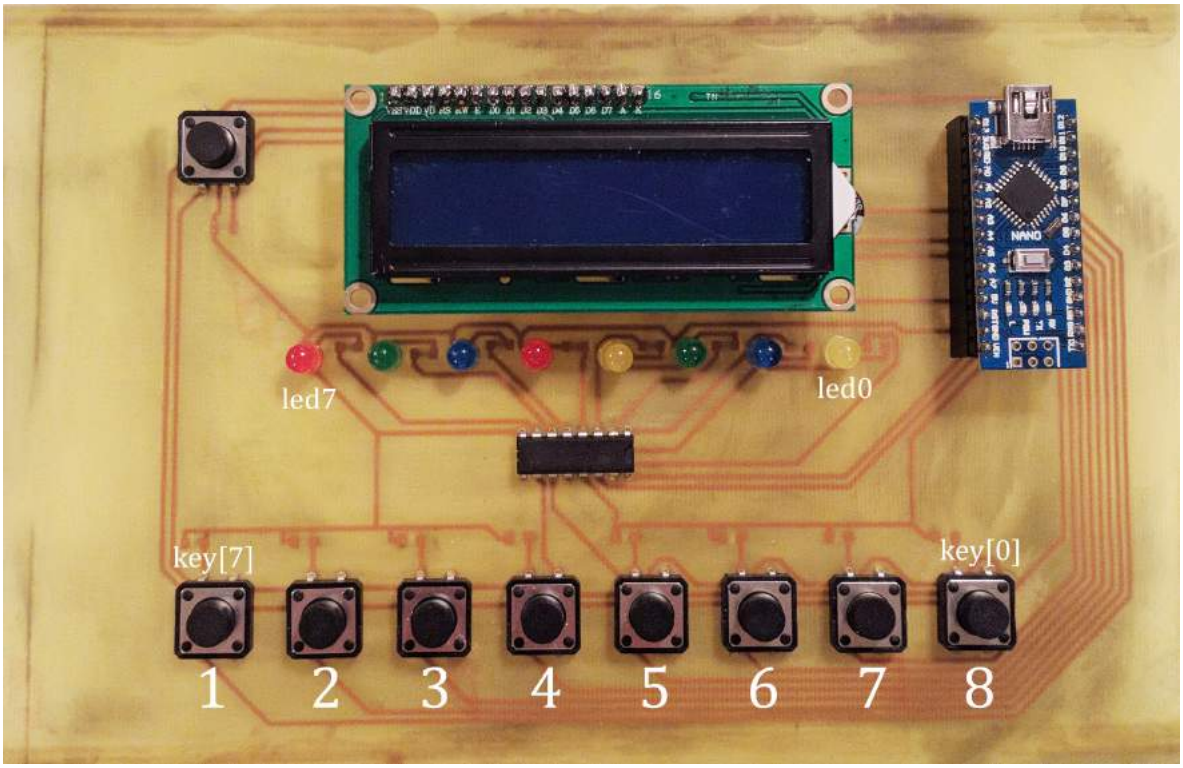
# Systemy mikroprocesorowe - 2019

Temat projektu: **Platforma do minigier**
Imię i nazwisko: **Tomasz Jankowski**
**Politechnika Poznańska**
kierunek: **AiR**, grupa: **A3**, nr albumu: **132062**

## Spis treści

## 1. Opis projektu

Układ - jak sama nazwa mówi - jest platformą do minigier. Wszelkie sterowanie odbywa się za pomocą ośmiu przycisków umieszczonych na jednej linii u dołu płytki oraz jednego umieszczonego w górnym lewym rogu. Skrócone instrukcje, aktualnie wybrany tryb oraz wynik są wyświetlane na wyświetlaczu LCD ze sterownikiem HD44780, a komunikacja odbywa się za pomocą magistrali I2C, co umożliwia wlutowany w układ wyświetlacza konwerter LCM1602. Diody podłączone do rejestru przesuwnego służą do informowania użytkownika, który przycisk powinien zostać wciśnięty i zostały rozmieszczone w małych odstępach od siebie, na odcinku jak najbliższym środka płaszczyzny płytki, aby wprowadzić dodatkowe utrudnienie do gry. Całość jest kontrolowana przez układ zgodny z Arduino Nano, który wymaga zewnętrznego zasilania za pomocą kabla mini USB. Platforma oferuje szerokie możliwości oraz łatwość w dodawaniu kolejnych trybów gry - mogą się one opierać na gotowych już funkcjach, takich jak odczyt i zapis do pamięci, czy sposób załącznia diod. Rolę komend mogą nie tylko spełniać załączane diody, ale również instrukcje z wyświetlacza LCD. Ograniczeniem jest jedynie kreatywność oraz czas, którym dysponujemy.



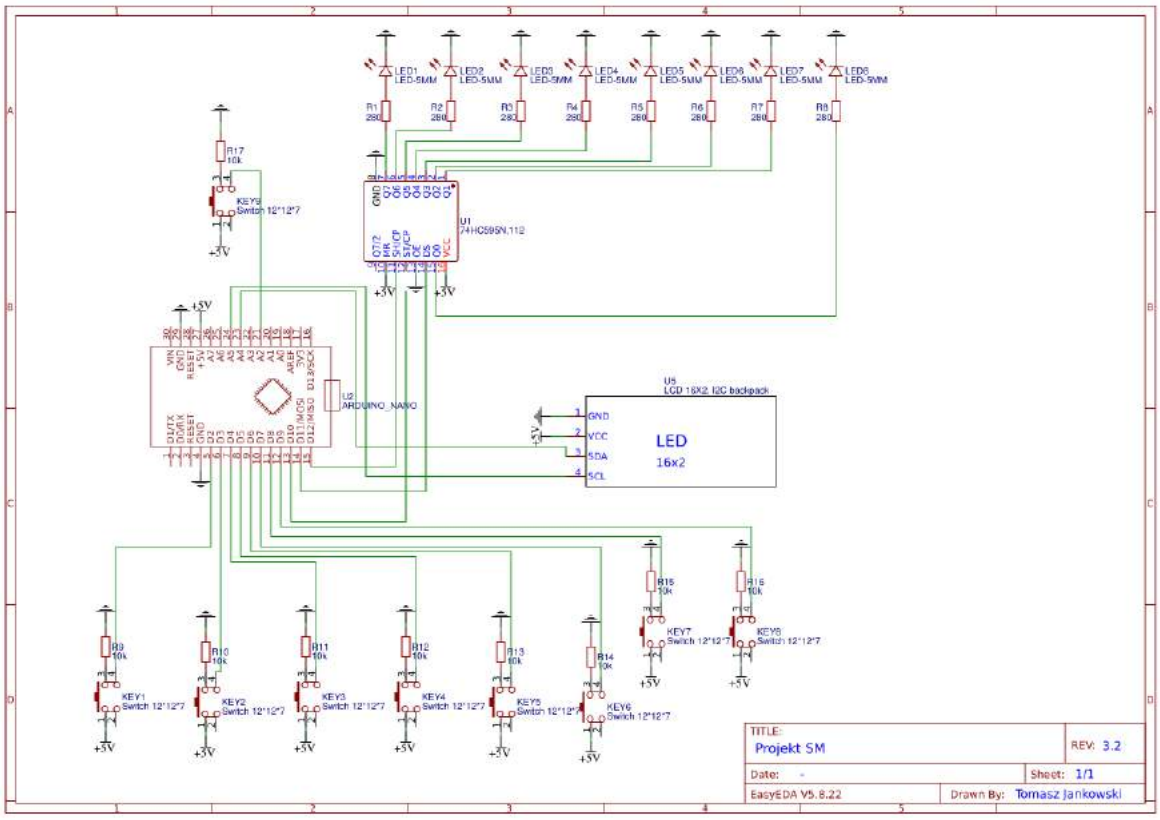

Rysunek 1: *Widok układu z góry*

## 2. Budowa układu

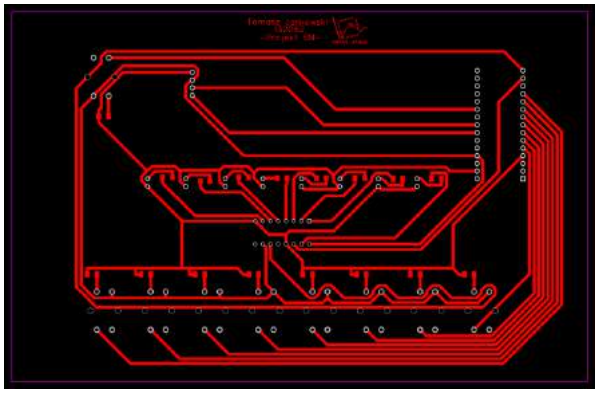
W układzie zostały wykorzystane następujące elementy elektroniczne:

- układ zgodny z Arduino Nano v3.0 (klon),
- wyświetlacz LCD 16x2 ze sterownikiem HD44780,
- konwerter magistrali I2C LCM1602,
- rejestr przesuwny 74HC595N,
- 8 diod LED 5mm w różnych kolorach,
- 9 przycisków typu tact switch o wymiarach 12x12x5 mm,
- rezystory SMD o różnych wartościach ($220\Omega, 360\Omega, 470\Omega, 681\Omega, 1k\Omega, 10k\Omega$,
- listwy goldpin.

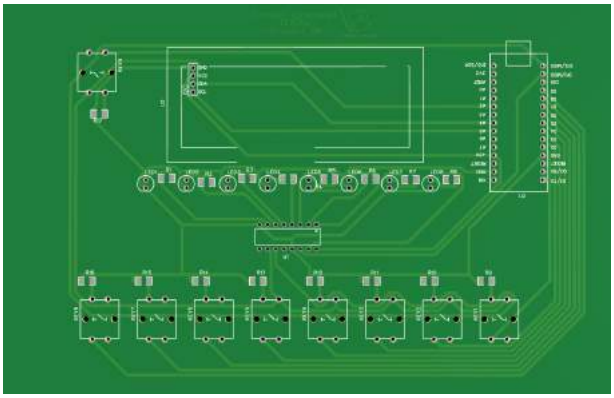
Takie zróżnicowanie wartości rezystorów pozwala na ujednolicenie poziomu jasności świecenia diod. Rezystory 10kΩ używane są w układzie każdego z przycisków jako pull-down.
Projekt został wykonany na własnoręcznie wykonanej płytce PCB, a obudowa stworzona jest z użyciem elementów z odzysku - dwóch kawałków deski wyciętych wyrzynarką do drewna oraz szkła akrylowego, sklejonych ze sobą klejem. Schemat elektryczny oraz projekt PCB zostały wykonane w środowisku EasyEDA oraz są dołączone wraz z dokumentacją.




Rysunek 2: *Schemat elektryczny*

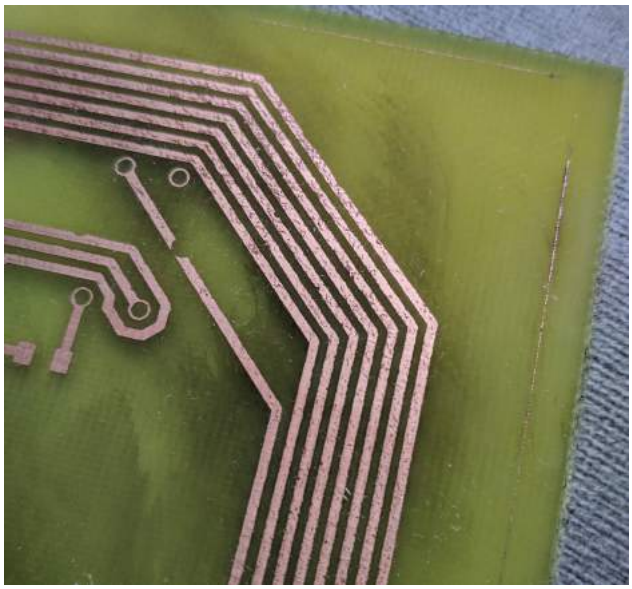


Rysunek 3: *Projekt płytki PCB*



Rysunek 4: *Wizualizacja*


Rysunek 5: *Proces termotransferu*



Rysunek 6: *Wytrawiona płytka PCB*

# 3. Elementy oprogramowania

## 3.1. Użyte biblioteki

- Wire - komunikacja z urządzeniami I2C (obsługa LCD)
- LiquidCrystal_I2C - sterowanie wyświetlaczem LCD
- EEPROM - obsługa pamięci

## 3.2. Definicje zmiennych

```
// REJESTR PRZESUWNY
int latchPin = 10;
int dataPin = 11;
int clockPin = 12;

// PRZYCISKI
int key[] = {2, 3, 4, 5, 6, 7, 8, 9}, button = A2;

// LCD
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

// ZMIENNE
int mode = 0, tab[15];
bool state = 0;
```

### 3.3. Funkcja void setup()

```
void setup()
{
  // lcd
  lcd.begin(16, 2);

  // rejestr
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);

  // przyciski
  for (int i = 0; i < 8; i++)
  {
    pinMode(key[i], INPUT);
  }

  // GENERATOR PSEUDO-LOSOWY
  randomSeed(analogRead(0));

  // START
  Reset();
  lightShiftRight();
  lightShiftLeft();
  blinkAll(1,200);
  Reset();
  chooseMode();
}
```

### 3.4. Funkcja void loop()

Funkcja ta jest pusta, program nigdy do niej nie wchodzi.

### 3.5. Start programu

Funkcja **void chooseMode()** - funkcja uruchamiająca się na początku działania programu (po funkcji void setup()), tłumacząca zasady wyboru trybu gry, a następnie przechodząca do funkcji **void chooseMode1()**:

```
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Wybierz tryb");
  delay(2500);
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print("<-----");
  lcd.setCursor(1, 1);
  lcd.print("Akceptuj tryb");
  delay(2500);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Przycisk 1-lewo");
  lcd.setCursor(0, 1);
  lcd.print("Przycisk 2-prawo");
  delay(2500);
  lcd.clear();
  chooseMode1();
```

### 3.6. Wybór trybu

Funkcja **void chooseMode1()** służy do wyboru trybu pierwszego **Trening** poprzez naciśnięcie przycisku w lewym górnym rogu, bądź zmiany trybu używając przycisków 1 i 2 (pierwsze dwa przyciski z ośmiu umieszczonych w jednej linii na dole). Przycisk 1 powoduje przejście do możliwości wyboru trybu trzeciego, a przycisk 2 wyboru drugiego. W sposób analogiczny działają funkcje **void chooseMode2()** oraz **void chooseMode3()**.

```
void chooseMode1()
{
  state=0;
  delay(200);
  lightPin(7);
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print("Tryb 1");
  lcd.setCursor(4, 1);
  lcd.print("Trening");

  while (!state)
  {
    if (digitalRead(key[6]) == HIGH)
      chooseMode2();

    if (digitalRead(key[7]) == HIGH)
      chooseMode3();

    if (analogRead(button) == 1023)
      Mode1();
  }
}
```

### 3.7. Zapis, odczyt oraz czyszczenie pamięci EEPROM

Funkcja **void EEPROMWritelong**(int address, long value) służy do zapisywania do pamięci EEPROM wartości zmiennych typu long na 4 bajtach.

```
void EEPROMWritelong(int address, long value)
      {
      //Decomposition from a long to 4 bytes by using bitshift.
      //One = Most significant -> Four = Least significant byte
      byte four = (value & 0xFF);
      byte three = ((value >> 8) & 0xFF);
      byte two = ((value >> 16) & 0xFF);
      byte one = ((value >> 24) & 0xFF);

      //Write the 4 bytes into the eeprom memory.
      EEPROM.write(address, four);
      EEPROM.write(address + 1, three);
      EEPROM.write(address + 2, two);
      EEPROM.write(address + 3, one);
      }
```

Funkcja **long EEPROMReadlong**(long address) służy do odczytywaia z pamięci EEPROM wartości zmiennych typu long zapisanych na 4 bajtach.

```
long EEPROMReadlong(long address)
      {
      //Read the 4 bytes from the eeprom memory.
      long four = EEPROM.read(address);
      long three = EEPROM.read(address + 1);
      long two = EEPROM.read(address + 2);
      long one = EEPROM.read(address + 3);

      //Return the recomposed long by using bitshift.
      return ((four << 0) & 0xFF) + ((three << 8) & 0xFFFF) + ((two << 16) & 0xFFFFFF) + ((one << 24) & 0xFFFFFFFF);
      }
```

Funkcja **void clearEEPROM**() służy do wyczyszczenia pamięci EEPROM, jednocześnie informując o operacji za pomocą wyświetlacza LCD.

```
void clearEEPROM()
{
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("EEPROM");
  lcd.setCursor(2,1);
  lcd.print("Clearing");
  delay(500);
  lcd.setCursor(10,1);
  lcd.print(".");
  delay(500);
  lcd.setCursor(11,1);
  lcd.print(".");
  delay(500);
  lcd.setCursor(12,1);
  lcd.print(".");
  delay(500);
  EEPROMWritelong(0,0);
  EEPROMWritelong(4,0);
  EEPROM.write(8,0);
  setup();
}
```

Funkcja ta jest możliwa do wykonania w funkcji wyboru trybu 2, czyli **void chooseMode2()** korzystając z kombinacji przycisków 6 oraz 8. Wiedza o kombinacji jest ukryta dla użytkownika, powinna być jawna jedynie dla "administratora".

```
if (digitalRead(key[2]) == HIGH && digitalRead(key[0]) == HIGH)
      clearEEPROM();
```

### 3.8. Realizacja załączania diod

Funkcja **void lightPin**(int p) służy do zaświecenia jedną z diod LED - korzysta z przesuwania bitów.

```
void lightPin(int p)
{
  int pin;
  pin = 1 << p;   // mnożenie przez 2
  digitalWrite(latchPin, LOW);    // włączenie nasłuchiwania
  shiftOut(dataPin, clockPin, MSBFIRST, pin);   // przesunięcie bitów
  digitalWrite(latchPin, HIGH);   // wyłączenie nasłuchiwania
}
```

Funkcja **void lightShiftRight**() służy do zaświecenia kolejno diod led7-led0 (od pierwszej do ósmej), zachowując czas świecenia każdej z diod na poziomie 75 ms. Analogicznie działa funkcja **void lightShiftLeft**().

```
void lightShiftRight()
{
  for (int i = 7; i >= 0; i--) {
    lightPin(i);
    delay(75);
  }
}
```

Funkcja **void Reset**() służy do zresetowania rejestru przesuwnego 74HC595N w celu zgaszenia wszystkich diod.

```
void Reset()
{
  for (int x = 0; x < 8; x++)
  {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, 0);
    digitalWrite(latchPin, HIGH);
  }
}
```

Funkcja **void blinkAll**(int n, int d) służy do zaświecenia wszystkich diod jednocześnie. Zmienna wejściowa n odpowiada za ilość takich zaświeceń, a d za czas świecenia.

```
void blinkAll(int n, int d)   // n - ilość, d - delay
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, 0);
  digitalWrite(latchPin, HIGH);
  delay(200);
  for (int x = 0; x < n; x++) {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, 255);
    digitalWrite(latchPin, HIGH);
    delay(d);
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, 0);
    digitalWrite(latchPin, HIGH);
    delay(d);
  }
}
```

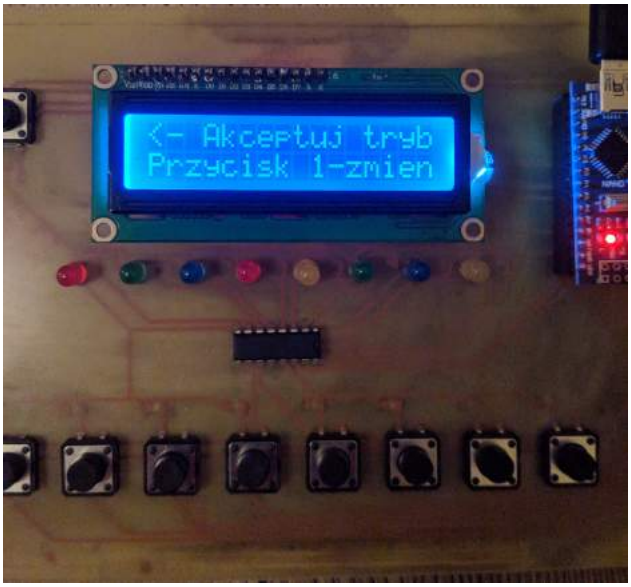
Funkcje opisujące działanie poszczególnych trybów gry to **void Mode1()**, **void Mode2()** oraz **void Mode3()**. Ich kod jest za długi na umieszczenie go w dokumentacji, ale można zapoznać się z nim w dołączonym do dokumentacji pliku źródłowym **projekt-SM.ino**.

## 4. Wykorzystane narzędzia projektowe

Do realizacji projektu wykorzystane zostały środowiska **EasyEDA** w wersji 5.8.22 (schemat elektryczny oraz projekt płytki PCB) oraz **Arduino IDE** w wersji 1.8.7 (kod programu).

## 5. Weryfikacja poprawności działania układu

Przy podłączeniu Arduino Nano do zasilania, program uruchomi się, uruchamiając sekwencję startową, czyli zaświecenie kolejno wszystkich diod od led7 do led0 i z powrotem, a następnie zaświecenie wszystkich diod jednocześnie. Instrukcje na wyświetlaczu LCD przedstawiają nam sposób wyboru trybu gry. Tryby gry zmienia się przy użyciu przycisków 1 oraz 2, czyli key[7] oraz key[6] (opis - *Rysunek 1*). Zależnie od trybu załącza się również jedna z diod led7, led6 lub led5. Akceptacja i rozegranie danego trybu odbywa się poprzez kliknięcie przycisku umieszczonego w lewym górnym rogu. Rozpoczęcie rozgrywki w danym trybie jest sygnalizowane poprzez jednoczesne zaświecenie wszystkich diod. Opisy każdego z trybów znajdują się poniżej.




Rysunek 7: *Instrukcja wyboru trybu*


Rysunek 8: *Wybór trybu 1*

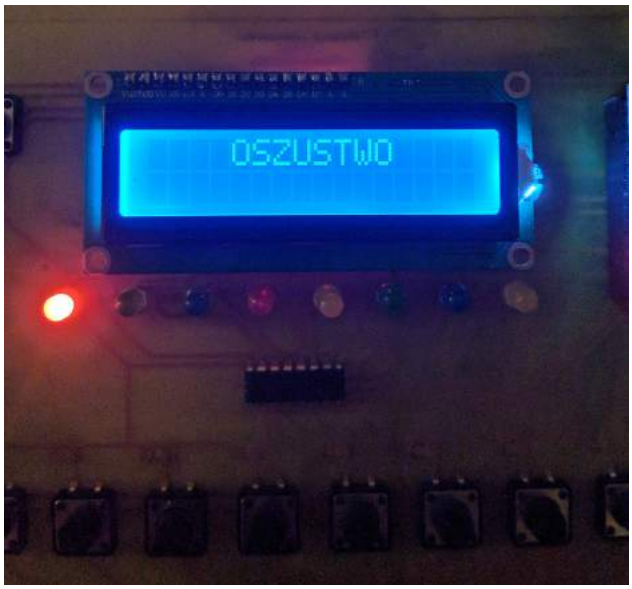
### 5.1. Tryb 1 - Trening

Ten tryb pozwala na przygotowanie się do trybu drugiego - jest rozgrzewką. Podczas rozgrywki zapalają się pojedyncze diody i zadaniem gracza jest wciśnięcie odpowiedniego przycisku, przykładowo: świecąca się dioda 3 (led[5]) sygnalizuje o konieczności wciśnięcia przycisku 3 (key[5]). Gdy prawidłowy przycisk został wciśnięty, na wyświetlaczu LCD pojawi się czas reakcji. Tym samym kończy się pierwsza runda. Trening trwa przez 10 rund, po którym pojawia się komunikat o jego zakończeniu oraz załącza się pierwsza z zielonych diod. Następnie ukazuje się ekran wyboru trybu.

### 5.2. Tryb 2 - Czas rekacji

Zasada działania tego trybu jest bardzo zbliżona do trybu 1. Należy wcisnąć przycisk odpowiadający załączonej diodzie, jednak po poprawnym ukończeniu pierwszej rundy, nie wyświetla się uzyskany czas reakcji. Zamiast tego, sygnalizacja ukończenia rundy jest zrealizowana poprzez zaświecenie wszystkich diod jednocześnie, po czym następuje druga runda. Jeżeli gracz wcisnął poprawny przycisk przed zaświeceniem się diody, wykrywana jest próba oszustwa - sygnalizowana załączeniem czerwonej diody oraz komunikatem na wyświetlaczu. Po tym zdarzeniu tryb gry jest natychmiastowo wyłączany i następuje przejście do ekranu wyboru trybu. Jeżeli gracz ukończy 15 rund, wyświetlany jest wynik będący sumą wszystkich międzyczasów, czyli czasów reakcji w każdej z rund. Następnie ukazuje się średni wynik ogólnego czasu reakcji graczy, a jeśli gracz pobił obecny rekord (lub ustanowił go jako pierwszy) - wyświetlany jest stosowny komunikat. Odczytywanie oraz zapisywanie średniego czasu reakcji graczy, rekordu oraz ilości rozegranych ogółem gier są zrealizowanie za pomocą obsługi pamięci EEPROM (odpowiednie fragmenty programu znajdują się w rozdziale 3).


Rysunek 9: *Wybór trybu 2*



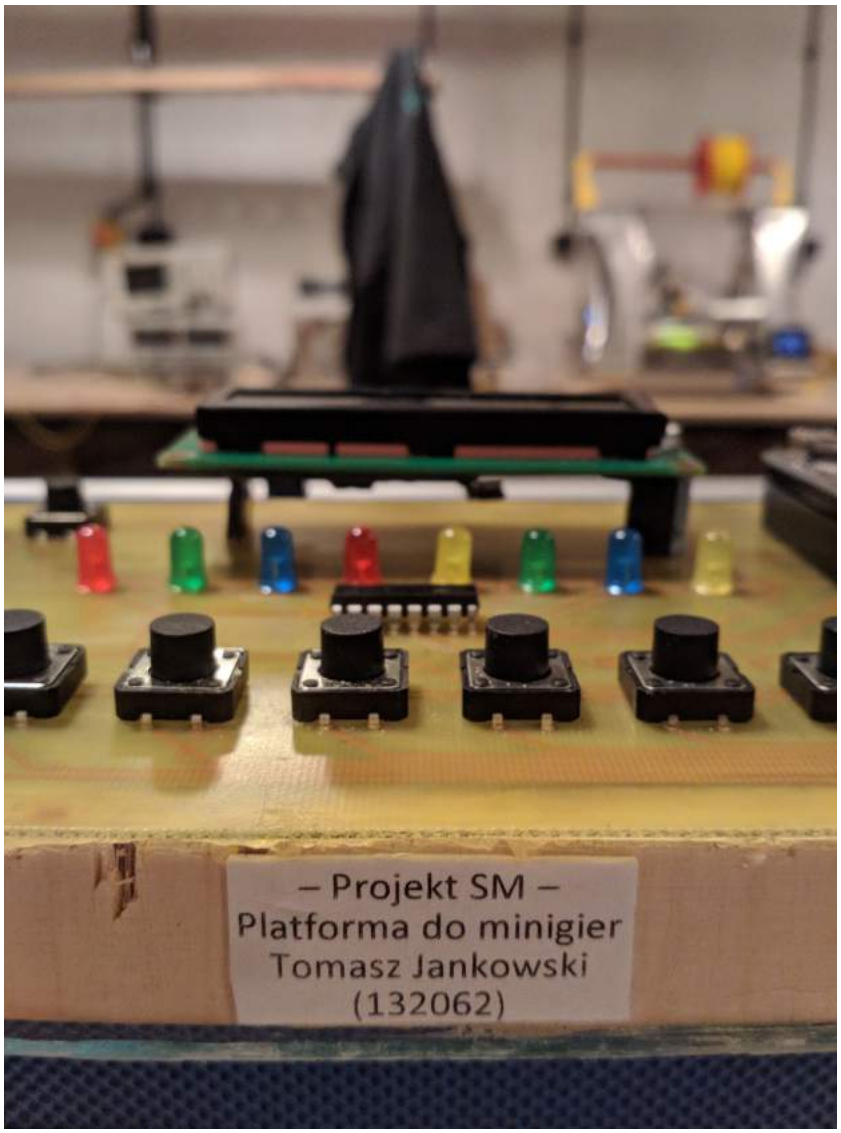

Rysunek 10: *Wykrycie próby oszustwa*

### 5.3. Tryb 3 - Multiplayer

W tym trybie bierze udział dwóch graczy. Gracz pierwszy dysponuje przyciskiem 1 (key[7]), a gracz 2 przyciskiem 8 (key[0]). Na wyświetlaczu pokazana jest aktualna punktacja każdego z graczy. Rundę wygrywa gracz, który jako pierwszy kliknie swój przycisk, gdy zaświeci się dowolna z ośmiu diod - uwaga, tutaj również zrealizowane jest wykrywanie wszelkich prób oszustwa, które sygnalizowane jest - tak jak poprzednio - komunikatem oraz załączeniem czerwonej diody. Za każdą wygraną rundę, dany gracz uzyskuje punkt, a za każdą próbę oszustwa jeden punkt jest mu odejmowany. Koniec każdej z rund jest sygnalizowany zaświeceniem się wszystkich diod jednocześnie. Zwycięzcą zostaje gracz, który jako pierwszy zdobędzie 10 punktów na swoim koncie, a informacja o jego wygranej zostanie wyświetlona na LCD. Po zakończeniu tego trybu gry, przechodzi się do ekranu wyboru trybu.

## 6. Literatura


1. http://dsp.org.pl/Systemy_mikroprocesorowe/42/
2. https://www.arduino.cc/reference
3. http://playground.arduino.cc/
4. https://www.norwegiancreations.com
5. https://stackoverflow.com/
6. home.agh.edu.pl/bartus/arduino/9
7. http://www.ti.com/lit/ds/symlink/sn74hc595.pdf
8. https://www.instructables.com/id/How-Shift-Registers-Work-74HC595/
9. https://forbot.pl/forum/topic/910-dla-poczatkujacych-wykonywanie-pcb-termotransferem/
10. https://www.overleaf.com
11. https://easyeda.com/components


Rysunek 11: Jeszcze jedno zdjęcie projektu