Projekt Mikrokontrolery

Temat projektu

- Temat 3 – „Przesuwanie znaków”

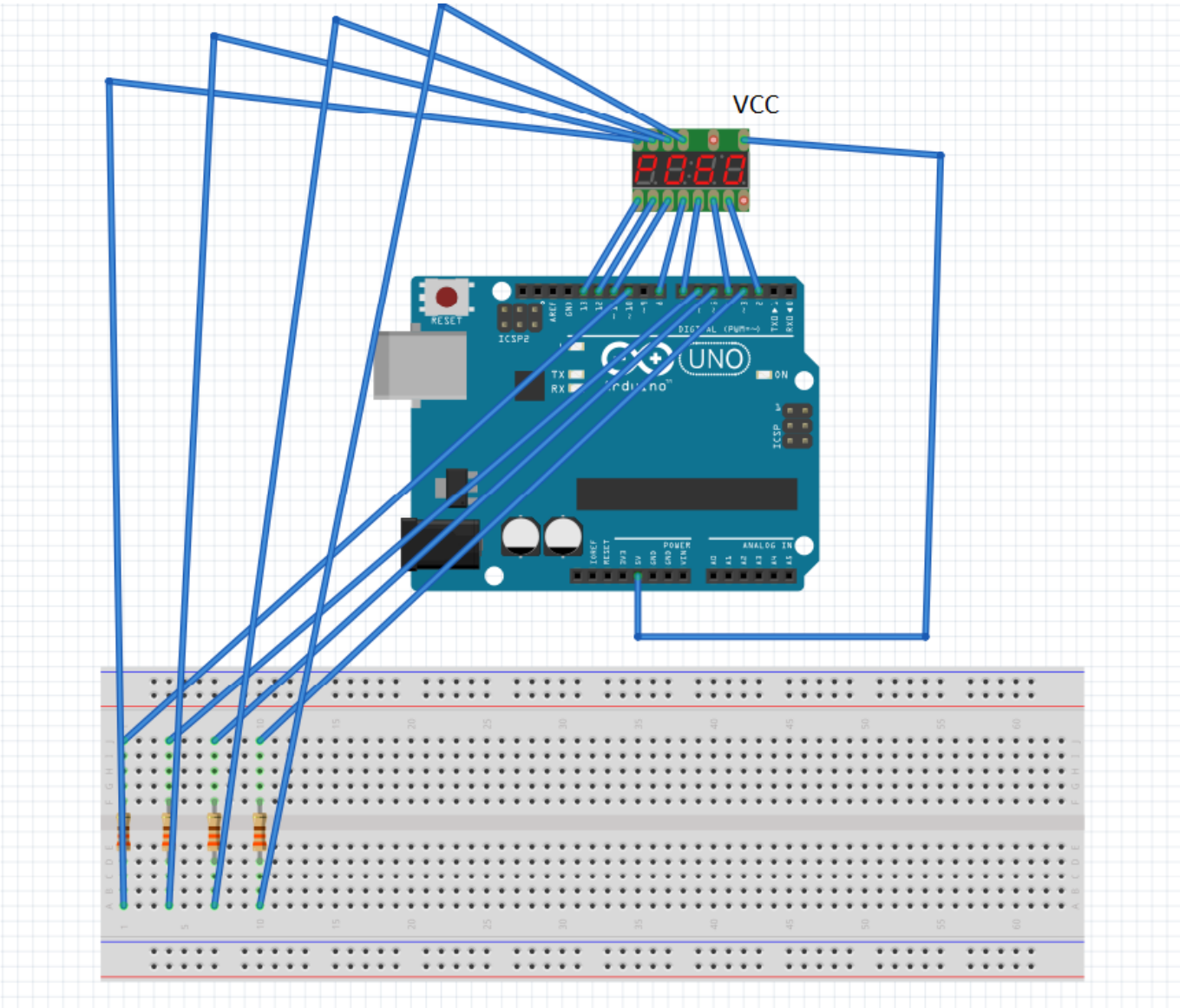
Skład osobowy grupy

- Szymon Sadowski
- Filip Prasałek
- Krzysztof Szczyrbak

Kierunek studiów, rok studiów i rok akademicki

- Informatyka, rok II, rok akademicki 2016/2017

Schemat połączeń płytki i modułu



Opis algorytmu

Algorytm wykorzystuje operacje bitowe na typie 'byte', aby przechowywać binarne reprezentacje danych symboli, co pozwala na zaoszczędzenie pamięci.

Przygotowane zostały funkcje do wyświetlenia pojedynczego symbolu, następnie wygaszenia segmentu, dzięki czemu możliwe było napisanie funkcji, wyświetlającej w przeplocie 4 symbole na 4 segmentach dając wrażenie wyświetlania 4 symboli na raz.

Aby kontrolować jasność poszczególnych segmentów użyto funkcji analogWrite. Aby mozna jej było użyć, konieczne jest odpowiednie podpięcie wyświetlacza (digity muszą być podłączone do wejść z symbolem tyldy).

Do reprezentacji pustego symbolu (zerowy bajt) użyto wartości -1 dzięki czemu możliwe było stworzenie animacji "wyłaniania się ciągu znaków".

Istotne elementy programu

Istotne stałe

- BLANK - wartość reprezentująca pusty symbol

Istotne zmienne

- byte pins[] - kolejność połączenia digits
- byte segments[] - kolejność połączenia segmentów
- byte nums_syms[] - reprezentacje binarne symboli
- byte BNESS_MAPPING[] - mapowanie jasności dla drugiego argumentu dla funkcji analogWrite, gdzie 0-dark 1-medium 2-light
- char TEXT_BUFF[14] - bufor do wyświetlania znaków na wyświetlaczu, reprezentuje on znaki, które kolejno będą wyświetlane
- byte TEXT_BUFF_INDICATOR - wskaźnik, od którego miejsca czytamy TEXT_BUFF aby wyświetlić następne 4 symbole

Funkcje

```
Ta funkcja wyciąga symbol jasności [0|1|2] z tabeli 'BRIGHT'

getBrightForSegment(n)
```

```
/*
 * Updatuje TEXT_BUFF w odniesieniu do zmiennej TEXT
 */
void updateTextBuff();
```

```
/*
```

```
  *Zwraca n-ta wartosc calkowita pobrana z 'TEXT'
 */
char getTextCharacter(byte n);
```

```
/*
 * Przesuwa wskaźnik aktualnej pozycji w TEXT_BUFF o 1 lub zeruje go
 */
void moveTextBuff();
```

```
/*
 * Wyciaga n-ty bit z bajta
 */
bool getNthBitFromByte(byte n, byte a);
```

```
/*
 * Wyswietla te segmenty, gdzie ustawiony jest bit w 'number'. Wyswietla w
segmencie o nr. 'dig' o zadanej jasnosci 'bness'
 */
void disp(byte number, byte dig, byte bness=255);
```

```
/*
 * Wyswietla wartosc calkowita podana w 'num' np. jak podamy num=12 to wyswietli
'c'
 */
void dispNum(char num, byte dig, byte bness=255);
```

```
/*
 * Wyswietla naraz 4 liczby calkowite podane w 'fst' 'snd' 'trd' 'fou'
 * o jasnosciach podanych odpowiednio w zmiennych z prefixem bness_
 * duration - czas trwania calego wyswietlania
 * DEL_INTER = delay interval czyli czestotliwosc odswiezania
 * Gdy podamy na danym miejscu -1 to nic sie tam nie wyswietli
 */
void disp4NumsBrightness(char fst, char snd, char trd, char fou, byte bness_fst,
byte bness_snd, byte bness_trd, byte bness_fou, short int duration, byte DEL_INTER
= 1) ;
```

```
/*
 * Wyswietla cykl pojawiania sie 6 liczb hexadecymalnych od prawej do lewej z
wylanianiem sie
 */
void dispCycleWithBness();
```

```
/*
 * "Czysci" caly digit o podanym numerze
 */
void turnOff(byte dig);
```

```
/*
 * Funkcja która sprawdza, czy na wejsciu serialowym jest jakis input
 * jesli jest to zwraca true, false w przeciwnym wypadku
 */
bool inputCheck();
```

```
/*
 * Funkcja wcielajaca w zycie komendy podane przez uzytkownika, parser
 */
void parseCommand(String input);
```

Informacje o wykorzystywanych bibliotekach

- brak użycia zewnętrznych bibliotek

Informacje o sposobie budowy kodu wykonywalnego

- kod nie wymaga żadnych zależności, budowa jest standardowa

Listing kodu źródłowego stworzonego w ramach projektu

- Projekt składa się z tylko jednego pliku ' src_SadowskiSczyrbakPrasalek.ino'

Pozostałe

Oto wykaz objętości kodu wykonywalnego:

```
Sketch uses 5266 bytes (16%) of program storage space. Maximum is 32256 bytes.
Global variables use 374 bytes (18%) of dynamic memory, leaving 1674 bytes for
local variables. Maximum is 2048 bytes.
```