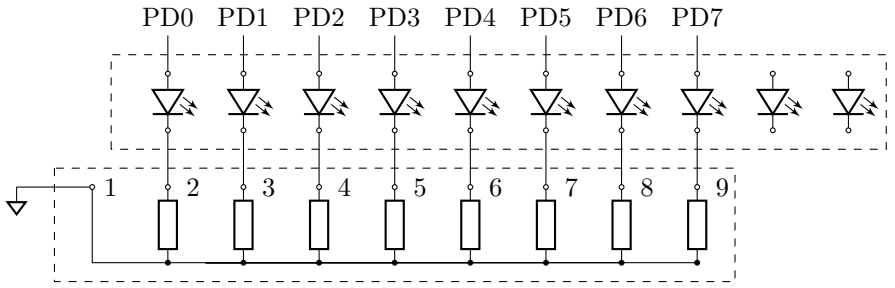
Systemy wbudowane

Lista zadań nr 1

14, 15 i 16 października 2024

1. Napisz program tłumaczący tekst na kod Morse'a. Tekst wejściowy powinien być odczytywany przez standardowe wejście skonfigurowane jak w programie przykładowym, wynik powinien być wyświetlany na wbudowanej diodzie LED lub podłączonej zewnętrznej diodzie (pamiętaj o rezystorze 220Ω w szeregu z diodą, bez niego dioda **może ulec uszkodzeniu!**)

2. W zestawie elementów znajduje się linijka LED oraz drabinka rezystorowa 220Ω. Podłącz je jak na poniższym schemacie. Uwaga – pin 1 drabinki rezystorowej, oznaczony kropką, jest pinem wspólnym. Nie należy do niego podłączać diody! Powinien on być podłączony do masy (linia GND, szyna (-) na płytce stykowej). Nieprawidłowe podłączenie **może uszkodzić** linijkę LED!



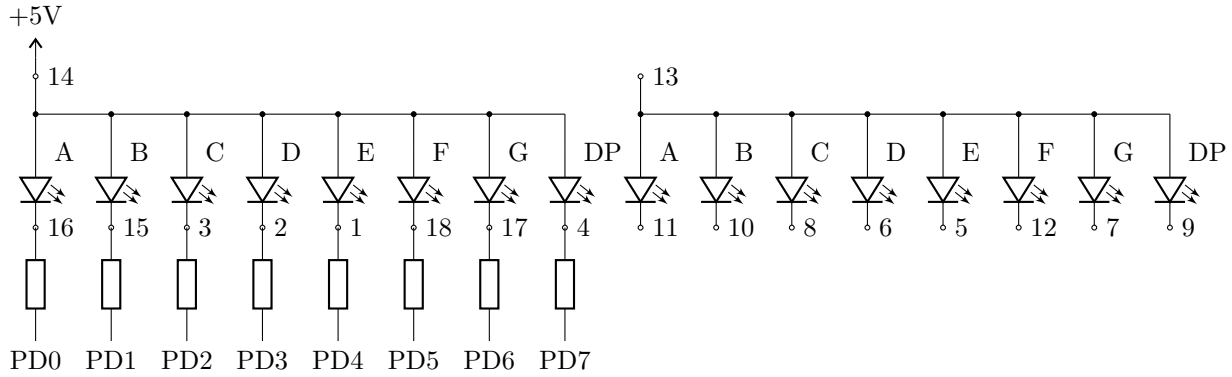
   Za ich pomocą zrealizuj efekt przewijającej się diodki, przypominającej światła KITT z serialu „Nieustraszony" („Knight Rider") lub oczy Cylonów z serialu „Battlestar Galactica". Wykorzystaj do tego piny z jednego portu, dzięki temu będzie można użyć operacji bitowych (np. przesunięcia bitowego), aby uprościć implementację.

   Uwaga – piny PD0 i PD1 są domyślnie wykorzystywane przez interfejs szeregowy, można wykorzystać je do zapalania diod wykonując na początku programu następującą instrukcję:

```
UCSR0B &= ~_BV(RXEN0) & ~_BV(TXEN0);
```

   Oczywiście użycie funkcji `printf` będzie wtedy niemożliwe. Uwaga – może się zdarzyć, że po podpięciu diod według schematu nie będzie można programować mikrokontrolera. Należy wtedy na czas programowania odłączyć pin 1 drabinki od masy.

3. Wyświetlacz 7-segmentowy z zestawu (FJ5261B) podłącz następująco używając rezystorów 220Ω. Nieprawidłowe uszkodzenie **może uszkodzić** wyświedlacz!



   Napisz program, który odlicza co sekundę kolejną cyfrę od 0 do 9 i wyświetla ją przy użyciu wyświetlacza. Wyświetlenie cyfry powinno następować przez wykonanie pojedynczego zapisu do odpowiedniego rejestru, potrzebne wartości warto zapisać w tablicy.

4. Napisz program, który dla każdego z wymienionych typów danych (`int8_t`, `int16_t`, `int32_t`, `int64_t`, `float`) wczyta z standardowego wejścia dwie liczby i wypisze wynik operacji: dodawania, mnożenia i dzielenia.

   Używając polecenia `make lst` wygeneruj plik `.lst`, zawierający kod wynikowy zapisany w formacie czytelnym dla człowieka. Co możesz powiedzieć o czasie wykonania i długości kodu dla tych operacji? (Nie jest wymagane dogłębne zrozumienie asemblera AVR ani wygenerowanego kodu, ale należy umieć wskazać, które fragmenty kodu wynikowego odpowiadają istotnym fragmentom kodu w C.)

   Uwaga: wprowadzanie i wyprowadzanie danych 64-bitowych nie jest bezpośrednio wspierane, można dla tych typów danych użyć 32-bitowego wejścia/wyjścia. Dla liczb zmiennoprzecinkowych `float` domyślnie brak takiego wsparcia, ale można je uzyskać dodając opcje:

   ```
   -Wl,-u,vfprintf -lprintf_flt -Wl,-u,vfscanf -lscanf_flt -lm
   ```

   do linijki `LIBS` w pliku Makefile (kosztem zwiększenia rozmiaru kodu wynikowego).