Linux i Python w Elektronicznej Sieci – ćwiczenia #12: Programowanie mikrokontrolerów STM32

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2021-06-01

1 Zadania

Zadanie 1.0.1

Zmień program `10_blink` tak, aby dioda LED migała około dwa razy wolniej.

Zadanie 1.0.2

Zastanów się jakie zmiany należy wykonać w programie `11_di`, aby zamiast dodawać zewnętrzny rezystor podciągający użyć wbudowanego podciągania wejść. Sprawdź swoje przypuszczenia odpowiednio modyfikując układ i program.

Zadanie 1.0.3

Zastanów się jakie zmiany należy wykonać w programie `11_di`, aby reagował on na przycisk podłączony do pinu A1. Sprawdź swoje przypuszczenia odpowiednio modyfikując układ i program.

Zadanie 1.0.4

Zmodyfikuj program `11_di` tak, aby dioda zapalała się tylko, gdy stan logiczny wejścia się zmienia.
Wskazówka: co zawiera zmienna ***`stan_a`*** *tuż przed załadowaniem jej nowym stanem? Pamiętaj o stanach nieustalonych.*

Zadanie 1.0.5

Wiedząc, że wartość 4096 odpowiada napięciu 3.3V, a 0 napięciu 0V, zmień przykładowy program ADC (`30_adc`) tak, aby zamiast surowej wartości wypisywał wartość napięcia.
Wskazówka: Aby uprościć obliczenia (uniknąć działań na liczbach zmiennoprzecinkowych), Twój program może podawać wartość w mV.

Zadanie 1.0.6

Zmodyfikuj rozwiązanie zadania 1.0.5 tak aby wypisywało wynik w V, bez stosowania arytmetyki zmiennoprzecinkowej.
Wskazówka: Wypisz osobno część całkowitą i część ułamkową

Zadanie 1.0.7

Zmodyfikuj program `21_uart_receiver` tak aby odbierał poprzez port szeregowy liczbę w zakresie od 2 do 9 i odpowiadał na nią trójkątem z gwiazdek odpowiedniej wielkości. Na przykład gdy otrzyma "3" powinien to być:

```
*
**
***
```

Zadanie 1.0.8

Zmodyfikuj rozwiązanie zadania 1.0.5 tak aby obsługiwać liczby wielocyfrowe (możemy ograniczyć się np. do dwucyfrowych). Odczyt liczby powinien kończyć w momencie wczytania znaku nowej linii. Wtedy też powinno następować generowanie trójkąta z gwiazdek.
Wskazówka: zrwóć uwagę na kodowanie nowej linii - `'\n'` vs `'\r\n'` vs `'\n\r'`, etc

Zadanie 1.0.9

Zmień funkcję realizującą logikę slave'a w przykładowym kodzie I2C (`40_i2c`) tak, aby zamiast mnożyć otrzymaną liczbę przez 2, dodawał do niej jakąś (dowolną) stałą.

Zadanie 1.0.10

Zapoznaj się z dokumentacją posiadanego rejestru z interfejsem I2C. Podłącz swój układ do magistrali `I2C1` mikrokontrolera (pin B7 jako SDA, pin B6 jako SCL). Zmodyfikuj program `40_i2c` tak aby obsługiwać posiadany rejestr - jako układ wejścia (odczyt stanu jego GPIO) oraz wyjścia (ustawianie stanu jego GPIO).
Wskazówka: STM32 będzie działać jako master magistrali I2C, a slavem będzie układ IO. Zatem nie będziesz potrzebować części kodu związanego z obsługą slave, działającego na `I2C2`.

2 Zadania dodatkowe

Zadanie 2.0.1

W programowaniu mikrokontrolerów często zachodzi potrzeba ustawienia pojedynczego bitu rejestru na 0 lub 1, albo zmiany jego wartości. Sprawdź (np. rozpisując ich działanie), które wyrażenie w rejestrze `rejestr`, na podstawie maski `maska`:

```
rejestr |= maska;  // <- Wyrażenie 1
rejestr ^= maska;  // <- Wyrażenie 2
rejestr &= ~maska; // <- Wyrażenie 3
```

- Ustawia te bity na 1,
- Ustawia te bity na 0,
- Odwraca wartości tych bitów.

Zadanie 2.0.2

Jaką maską bitową można sprawdzić czy bit nr. 1 jest w stanie wysokim? A jaką można sprawdzić to samo, ale dla wszystkich bitów parzystych (na pozycjach 0, 2, 4 ... 14[a])?

a. 16 bitowa liczba ma bity "ponumerowane" od 0 do 15

Zadanie 2.0.3

Napisz wyrażenia, które nie znając poprzedniej wartości 8-bitowego rejestru `XYZZY`, wykona operacje:

1. Ustawi jego drugi najmłodszy bit jako 1
2. Ustawi jego piąty najmłodszy bit jako 0
3. Odwróci jego najstarszy bit
4. Wyzeruje jego dolną połowę

Wskazówka: Możesz wygenerować maskę z ustawionym bitem n za przesuwając jedynkę o n miejsc w lewo: `(1<<n)`