Linux i Python w Elektronicznej Sieci – ćwiczenia #13: Programowanie mikrokontrolerów STM32

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

2023-06-25

1 Zadania

Zadanie 1.0.1

1. pobierz źródła biblioteki *libopencm3* (poprzez `git clone`) i skompiluj je uzywając polecenia `make`
2. pobierz źródła przykładowych programów do dzisiejszych zajęć
3. utwórz link symboliczny `libopencm3` w katalogu z źródłami przykładów wskazujący na poprawną lokalizację *libopencm3* (zauważ że przykładowe polecenia zawierają przykładowe lokalizacje - Ty mogłe(a)ś użyć innej).

Zadanie 1.0.2

Zgodnie z opisem w skrypcie wykładowym podłącz płytkę z mikrokontrolerem do przejściówki USB-UART (pełniącej rolę źródła zasilania i programatora).
Wskazówka: Sugerujemy wykonanie połączenia przy użyciu kabelków żeńsko-żeńskich, bez płytki stykowej. Sugerujemy aby po sprawdzeniu poprawności działania tego podłączenia nie rozłączać go w trakcie wykonywania kolejnych zadań. W trakcie naszych zajęć piny Rx i Tx będą służyły wyłącznie komunikacji z komputerem z użyciem tej przejściówki. Natomiast inne układy z płytki stykowej można dołączać do mikrokontrolera przy pomocy przewodów męsko-żeńskich.

1.1 dioda

Zadanie 1.1.1

Zapoznaj się z opisem programu `10_blink` w rozdziale *Pierwszy program* skryptu wykładowego. Skompiluj i wgraj do mikrokontrolera ten program (za pomocą polecenia `make install` wykonanego w katalogu w którym znajdują się jego źródła).

Zadanie 1.1.2

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

1.2 przycisk

Zadanie 1.2.1

Zapoznaj się z opisem programu `11_di` w rozdziale *Obsługa wejść* skryptu wykładowego. Skompiluj i wgraj do mikrokontrolera ten program. Zobacz jak działa.

Zadanie 1.2.2

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 1.2.3

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 1.2.4

Zmodyfikuj program `11_di` tak, aby przycisk działał jak przełącznik - tzn. jedno naciśnięcie włącza diodę, kolejne ją wyłącza, kolejne znowu włącza, itd.
Wskazówka: co zawiera zmienna ***`stan_a`*** *tuż przed załadowaniem jej nowym stanem?*

Zadanie 1.2.5

Poeksperymentuj z kodem z zadania 1.2.4 modyfikując wartość opóźnienia w pętli głównej. Co się dzieje gdy wartość ta jest zbyt mała? Co jest przyczyną takiego zachowania?

1.3 port szeregowy (UART)

Zadanie 1.3.1

Zapoznaj się z opisem działania UART w rozdziale *UART* skryptu wykładowego. Skompiluj i wgraj do mikrokontrolera program `21_uart_receiver`. Zobacz jak działa.

Zadanie 1.3.2

Zmodyfikuj program `21_uart_receiver` tak aby odbierał poprzez port szeregowy liczbę w zakresie od 1 do 9 i odpowiadał na nią trójkątem z gwiazdek odpowiedniej wielkości. Na przykład gdy otrzyma ”3” powinien to być:

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

Zadanie 1.3.3

Zmodyfikuj rozwiązanie zadania 1.3.2 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 1: wartość pierwszej cyfry przechowuj w zmiennej globalnej do czasu otrzymania kolejnej cyfry
Wskazówka 2: zwróć uwagę na kodowanie nowej linii - ***`'\n'`*** *vs* ***`'\r\n'`*** *vs* ***`'\n\r'`****, etc*

1.4 przetwornik analogowo-cyfrowy

Zadanie 1.4.1

Zapoznaj się z opisem działania przetwornika analogowo-cyfrowego w rozdziale *przetwornik analogowo-cyfrowy* skryptu wykładowego. Skompiluj i wgraj do mikrokontrolera program `30_adc`. Zobacz jak działa.

Zadanie 1.4.2

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.

1.5 I^2C

Zadanie 1.5.1

Zapoznaj się z opisem działania interfejsu I^2C w rozdziale I^2C skryptu wykładowego. Skompiluj i wgraj do mikrokontrolera program `40_i2c`. Zobacz jak działa.

Zadanie 1.5.2

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 - ustaloną przez programistę) stałą.

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)`

Zadanie 2.0.4

Zmodyfikuj rozwiązanie zadania 1.4.2 tak aby wypisywało wynik w V, bez stosowania arytmetyki zmiennoprzecinkowej.
Wskazówka: Wypisz osobno część całkowitą i część ułamkową

Zadanie 2.0.5

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`.