

Programowanie systemów internetu rzeczy i aplikacji sieciowych

Laboratorium 3

Piotr Niewiński
Bartosz Stachyra
Szymon Wiśniewski

Spis treści

Zadanie 1	2
Kod do zadania	2
Zadanie 2	3
Kod do zadania	3
Wynik Putty	4
Zadanie 3	4
Kod do zadania	4
Wynik Putty	6
Zadanie 4	6
Kod do zadania	6
Wynik Putty	8

Zadanie 1

Celem pierwszej części jest utworzenie wątku i uruchomienie w nim funkcjonalności polegającej na miganiu zieloną diodą na płytce

Kod do zadania

```
char stack_thread_blinking_green[THREAD_STACKSIZE_MAIN];

void *thread_blinking_green(void* arg) {
    (void) arg;
    xtimer_ticks32_t time_start;
    time_start = xtimer_now();

    while(1) {
        GREEN_LED_TOGGLE;
        xtimer_periodic_wakeup(&time_start, GREEN_LED_PERIOD);
    }

    return NULL;
}

int main(void)
{
    thread_create(stack_thread_blinking_green,
        sizeof(stack_thread_blinking_green),
        THREAD_PRIORITY_MAIN - 1, THREAD_CREATE_STACKTEST,
        thread_blinking_green, NULL,
        "thread_blinking_green");
    return 0;
}
```

Fragment kodu tworzy wątek w systemie wbudowanym do kontrolowania migającej zielonej diody LED. Stos dla wątku jest alokowany, a funkcja wątku wykorzystuje nieskończoną pętlę do przełączania stanu diody. Przełączanie diody odbywa się w regularnych odstępach czasu, kontrolowanych przez timer. Wątek jest tworzony w funkcji `main`, z określonym priorytetem i rozmiarem stosu.

Po kompilacji dioda zaczęła migać

Zadanie 2

Celem drugiej części jest napisanie kodu obsługującego przerwanie wywoływane przez naciśnięcie przycisku na płytce mikrokontrolera.

Kod do zadania

```
static void user_button_callback(void *arg){
    (void) arg;
    static unsigned button_pressed = 0;
    static xtimer_ticks32_t start;
    xtimer_ticks32_t stop;

    if(button_pressed == 0){
        start = xtimer_now();
        button_pressed = 1;
    } else {
        stop = xtimer_now();
        /*zmodyfikuj zapisywanie czasu przycisniecia przycisku */
        DEBUG("User button being pressed down time[us]: %ld\n",
xtimer_diff(stop, start).ticks32);
        button_pressed = 0;
    }
}

char stack_thread_blinking_green[THREAD_STACKSIZE_MAIN];

void *thread_blinking_green(void* arg){
    (void) arg;
    xtimer_ticks32_t last_start;
    last_start = xtimer_now();
    GREEN_LED_ON;

    while(1){
        GREEN_LED_TOGGLE;
        xtimer_periodic_wakeup(&last_start, GREEN_LED_PERIOD);
    }

    return NULL;
}

int main(void)
{
    thread_create(stack_thread_blinking_green,
sizeof(stack_thread_blinking_green),
                                THREAD_PRIORITY_MAIN - 1, THREAD_CREATE_STACKTEST,
                                thread_blinking_green, NULL, "green");
    gpio_init_int(USER_BUTTON, GPIO_IN_PU, GPIO_BOTH, user_button_callback,
(void*)0);
    return 0;
}
```

Funkcja `user_button_callback` mierzy czas trzymania wciśniętego przycisku użytkownika, używając timera do zapisania czasu wciśnięcia i zwolnienia. W funkcji `main` przycisk użytkownika jest inicjalizowany do wywoływania funkcji zwrotnej (callback) przy jego użyciu.

Wynik Putty



```
main(): This is RIOT! (Version: dc78b-main)
User button being pressed down time[us]: 157400
User button being pressed down time[us]: 173688
User button being pressed down time[us]: 187016
User button being pressed down time[us]: 184461
```

Po kompilacji i wciśnięciu a następnie puszczeniu przycisku, na konsoli wyświetlana jest informacja o czasie, przez przycisk który był przyciśnięty.

Zadanie 3

Celem tej części laboratorium jest stworzenie systemu w którym czas przyciśnięcia przycisku będzie wyświetlany przez wątek do którego wartość tego czasu zostanie przesłana poprzez system wiadomości systemu RIOT.

Kod do zadania

```
long user_button_pressed_time = 0;

static void user_button_callback(void *arg) {
    long* pressed_time = (long*)arg;
    static unsigned button_pressed = 0;
    static xtimer_ticks32_t start;
    xtimer_ticks32_t stop;
```

```

    if(button_pressed == 0){
        start = xtimer_now();
        button_pressed = 1;
    } else {
        stop = xtimer_now();
        /*zmodyfikuj zapisywanie czasu przycisnienia przycisku */
        *pressed_time = xtimer_diff(stop, start).ticks32;
        button_pressed = 0;
    }
}

char stack_thread_blinking_green[THREAD_STACKSIZE_MAIN];

void *thread_blinking_green(void* arg){
    (void)arg;
    long* pressed_time = (long*)arg;
    long previous_pressed_time = 0;

    xtimer_ticks32_t last_wakeup = xtimer_now();
    GREEN_LED_ON;
    while(1){
        GREEN_LED_TOGGLE;
        xtimer_periodic_wakeup(&last_wakeup, GREEN_LED_PERIOD);
        if (previous_pressed_time!=*pressed_time){
            previous_pressed_time=*pressed_time;
            DEBUG("User button being pressed down time[us]: %ld\n",
previous_pressed_time);
        }

        return NULL;
    }

}

int main(void)
{
    thread_create(stack_thread_blinking_green,
sizeof(stack_thread_blinking_green),
                THREAD_PRIORITY_MAIN - 1, THREAD_CREATE_STACKTEST,
                thread_blinking_green,
                (void*)&user_button_pressed_time, "green");

    gpio_init_int(USER_BUTTON, GPIO_IN_PU, GPIO_BOTH, user_button_callback,
(void*)&user_button_pressed_time);

    /* jeśli wykonanie kodu dotrze do tego miejsca,
zmienne zdefiniowane w main() przestaną być dostępne */
    return 0;
}

```

Ten fragment kodu w C tworzy system wbudowany z funkcją obsługi przycisku i kontrolą migającej diody LED. Funkcja `user_button_callback` mierzy czas wciśnięcia przycisku, zapisując go w zmiennej `user_button_pressed_time`. Wątek

thread_blinking_green używa tej zmiennej do rejestrowania i wyświetlania czasu wciśnięcia przycisku. W funkcji main, wątek diody LED jest tworzony i przycisk użytkownika jest inicjalizowany do wywołania funkcji zwrotnej z odpowiednim argumentem.

Wynik Putty

```
User button being pressed down time[us]: 176310
User button being pressed down time[us]: 156254
User button being pressed down time[us]: 188441
User button being pressed down time[us]: 188520
```

Po zaimplementowaniu obsługi kilku wątków i dodaniu drukowaniu przycisku do funkcji obsługującej wątek migającej diody, migła ona podczas wciskania przycisku.

Zadanie 4

Kod do zadania

```
char stack_thread_red[THREAD_STACKSIZE_MAIN];
kernel_pid_t red_pid;

void *thread_red(void* arg) {
    (void) arg;
    msg_t msg;
    while(1) {
        msg_receive(&msg);
        unsigned time = msg.content.value;
        DEBUG("User time[us]: ");
        DEBUG("%d\n", time);
    }

    return NULL;
}

static void user_button_callback(void *arg) {
```

```

static unsigned button_pressed = 0;
static xtimer_ticks32_t start;
xtimer_ticks32_t stop;
msg_t msg;

if(button_pressed == 0){
    start = xtimer_now();
    button_pressed = 1;
} else {
    stop = xtimer_now();
    msg.content.value = xtimer_diff(stop, start).ticks32;
    msg_send(&msg, *(kernel_pid_t*)arg);
    button_pressed = 0;
}
}

char stack_thread_blinking_green[THREAD_STACKSIZE_MAIN];

void *thread_blinking_green(void* arg){
    (void)arg;
    xtimer_ticks32_t last_wakeup = xtimer_now();
    GREEN_LED_ON;
    while(1){
        GREEN_LED_TOGGLE;
        xtimer_periodic_wakeup(&last_wakeup, GREEN_LED_PERIOD);
    }
}

int main(void)
{
    thread_create(stack_thread_blinking_green,
sizeof(stack_thread_blinking_green),
        THREAD_PRIORITY_MAIN - 1, THREAD_CREATE_STACKTEST,
        thread_blinking_green, (void*)0, "green");

    red_pid = thread_create(stack_thread_red, sizeof(stack_thread_red),
        THREAD_PRIORITY_MAIN - 2, THREAD_CREATE_STACKTEST,
        thread_red, (void*)&red_pid, "red");

    gpio_init_int(USER_BUTTON, GPIO_IN_PU, GPIO_BOTH, user_button_callback,
(void*)&red_pid);

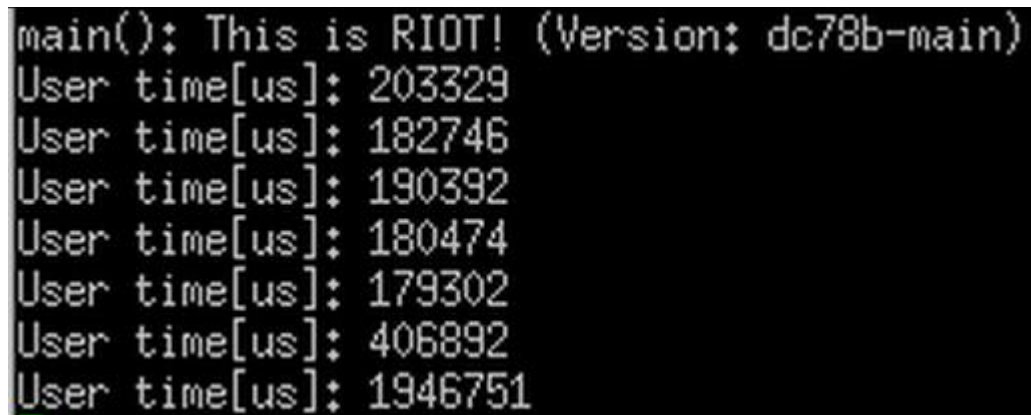
    return 0;
}

```

W tym kodzie C dla systemu wbudowanego tworzone są dwa wątki: jeden do obsługi migającej zielonej diody LED, a drugi do odbierania wiadomości. Wątek `thread_red` oczekuje na wiadomości z czasem trzymywania wciśniętego przycisku, a następnie wyświetla ten czas. Funkcja `user_button_callback` mierzy czas trzymywania przycisku i wysyła go jako wiadomość do wątku `thread_red`. Wątek `thread_blinking_green` kontroluje migającą zieloną diodę LED, przełączając

ją w regularnych odstępach czasu. W funkcji `main`, oba wątki są inicjalizowane i przycisk użytkownika jest skonfigurowany do wywołania funkcji zwrotnej. Kod ten demonstruje komunikację między wątkami za pomocą wiadomości w systemie mikrokontrolera.

Wynik Putty



```
main(): This is RIOT! (Version: dc78b-main)
User time[us]: 203329
User time[us]: 182746
User time[us]: 190392
User time[us]: 180474
User time[us]: 179302
User time[us]: 406892
User time[us]: 1946751
```

Po kompilacji rezultatem jest wyświetlenie czasu przytrzymania przycisku w konsoli.