„Ha beérjük annyival, hogy elátkozzuk vagy dicsőítjük a technikát, akkor sohasem jutunk el lényegének a megragadásához.”
Martin Heidegger

MIKROVEZÉRLŐS RENDSZERFEJLESZTÉS

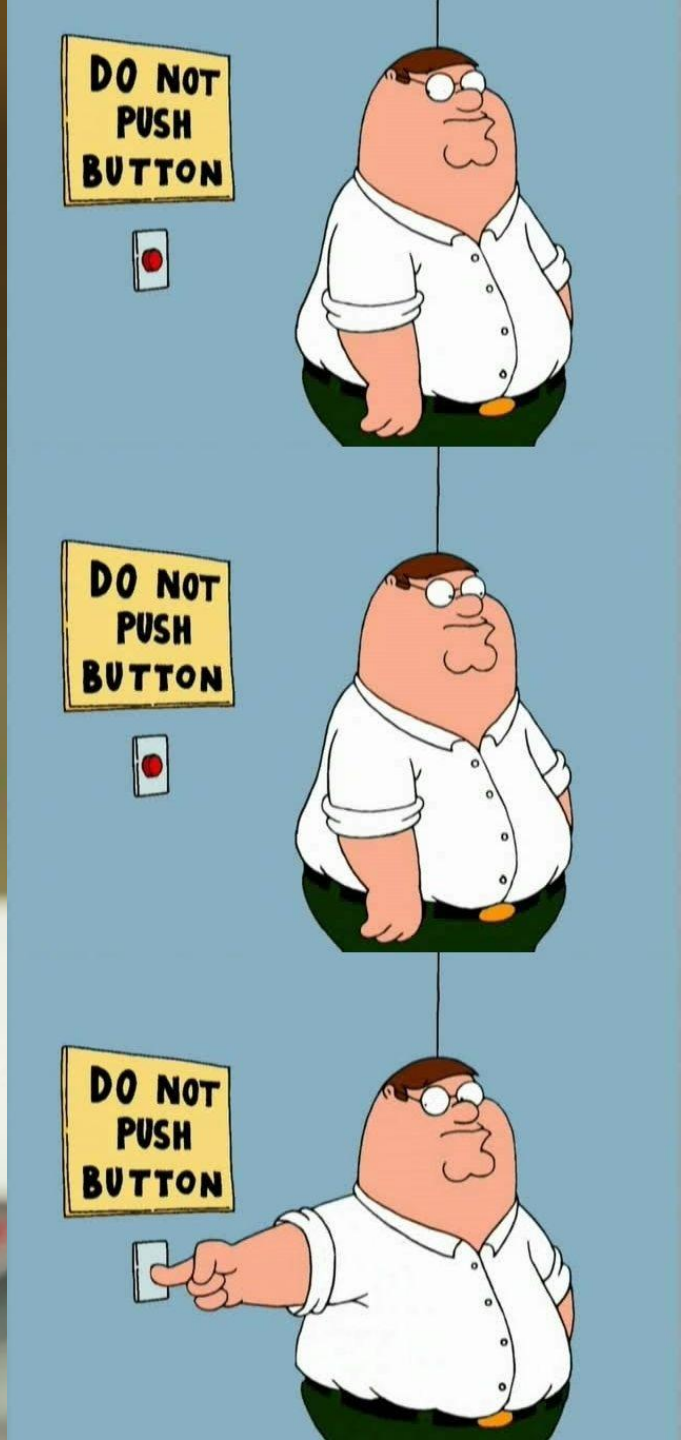
ChibiOS/RT

ChibiOS PAL: Nyomógomb kezelés kihívásai

Zsupányi Krisztián

Chibios PAL: Port abstraction layer

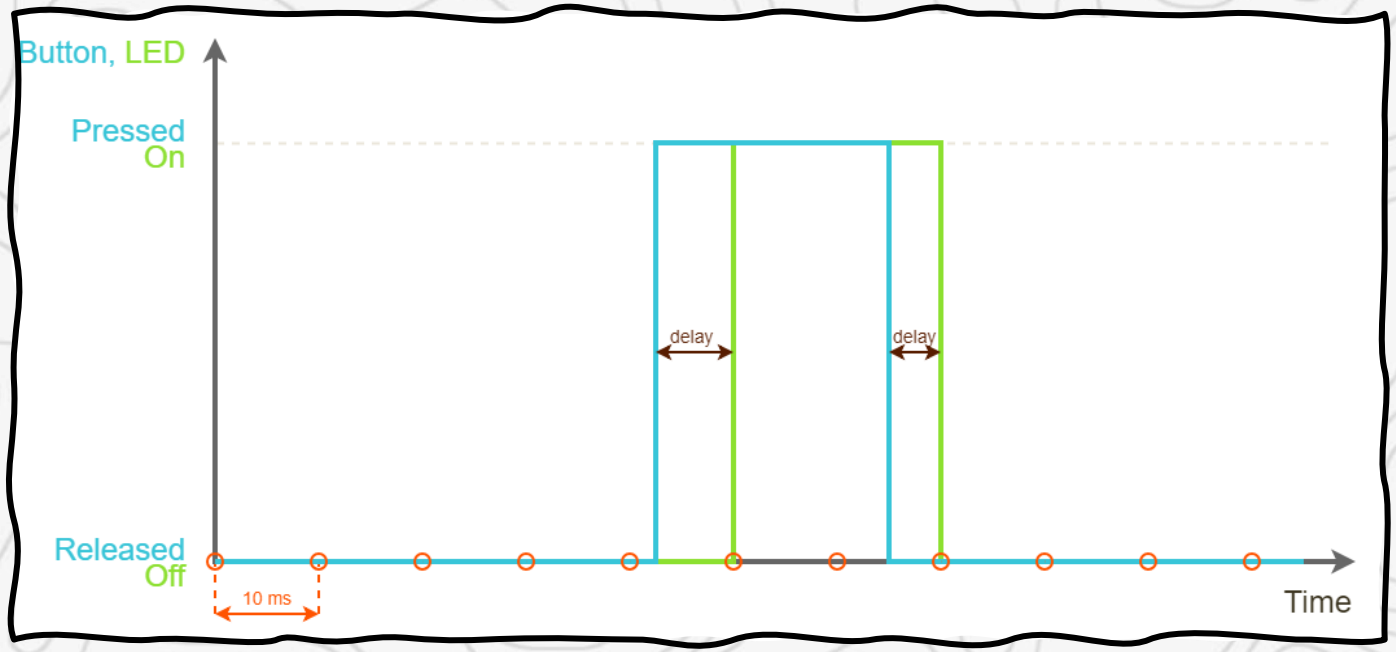
CHIBIOS PAL: NYOMÓGOMB KEZELÉS KIHÍVÁSAI



NYOMÓGOMB: POLLING, TOTÁLIS KÁOSZ

User gomb: GPIOA port 0-s pin,

- 10 millisec-es időzítés a beolvasáshoz
- Nem hatékony, folyamatos CPU terhelést jelent
- Van késleteltetés, akár le is maradhatunk
- Pergésre érzékeny, ha gyorsan mintavételezzük




```c
#include "ch.h"
#include "hal.h"

/* Application entry point. */
int main(void) {

  /* ChibiOS/HAL and ChibiOS/RT initializatio
  halInit();
  chSysInit();

  /* main() thread loop. */
  while (true) {
    /* Checking the button status. */
    if (palReadPad(GPIOA, 0U) == PAL_HIGH) {
      /* Button is pressed: turning the LED c
      palSetPad(GPIOE, 8U);
    }
    else {
      /* Button is released: turning the LED
      palClearPad(GPIOE, 8U);
    }
    chThdSleepMilliseconds(10);
  }
}
```

ChibiOS PAL: Nyomógomb kezelés kihívásai

OTT vagyunk MÁÁÁÁR?

NYOMÓGOMB: ESEMÉNY ALAPÚ

Megszakításokkal vezérelt nyomógomb kezelés
PAL_USE_WAIT aktiválás**: halconf.h-ban**

EXTened Interrupt and events controller (EXTI)

- **16 csatorna, pl EXT2: PA2, PB2, PC2, PD2, PE2…**
- **Hardveres, él vezérelt:** felfutó(rising), lefutó(falling) vagy mindkettő

Esemény beállítása:

palEnablePadEvent(<port>,<pin>,<edge config>);

- **PAL_EVENT_MODE_RISING_EDGE** -> **Felfutó él**
- **PAL_EVENT_MODE_FALLING_EDGE** -> **Lefutó él**
- **PAL_EVENT_MODE_BOTH_EDGES** -> **Mindkettő**

Várakozás az eseményre:

palWaitPadTimeout(<port>,<pin>,<timeout>);

Időtúllépés(timeout) ms-ben vagy:

- **TIME_INFINITE végtelen, TIME_IMMEDIATE azonnal**

```
#include "ch.h"
#include "hal.h"

/* Application entry point. */
int main(void) {

  /* ChibiOS/HAL and ChibiOS/RT initialization. */
  halInit();
  chSysInit();

  /* Enabling events on both edges of PA0 signal.*/
  palEnablePadEvent(GPIOA, 0U, PAL_EVENT_MODE_BOTH_EDGES);

  /* main() thread loop. */
  while (true) {
    /* Waiting for an edge on the button.*/
    palWaitPadTimeout(GPIOA, 0U, TIME_INFINITE);

    /* The action depends on the button state.*/
    if (palReadPad(GPIOA, 0U) == PAL_HIGH) {
      /* Button is pressed: turning the LED on. */
      palSetPad(GPIOE, 8U);
    }
    else {
      /* Button is released: turning the LED off. */
      palClearPad(GPIOE, 8U);
    }
  }
}
```


Majd szólok, csak várd meg!

NYOMÓGOMB: ESEMÉNY ALAPÚ, CALLBACK

Megszakításokkal vezérelt nyomógomb kezelés
PAL_USE_WAIT aktiválás**: halconf.h-ban**

- Nincs várakozás a hívó szálban
- Hardveres megszakítást használ
- Pergésre továbbra is érzékeny

Esemény beállítása:

palEnablePadEvent(<port>,<pin>,<edge config>);

- **PAL_EVENT_MODE_RISING_EDGE** -> **Felfutó él**
- **PAL_EVENT_MODE_FALLING_EDGE** -> **Lefutó él**
- **PAL_EVENT_MODE_BOTH_EDGES** -> **Mindkettő**

Callback beállítása:

palSetPadCallback(<port>,<pin>,<callback fv>, <arg>);

```c
#include "ch.h"
#include "hal.h"

/* Callback associated to the event. */
static void button_cb(void *arg) {

  (void)arg;
  /* Changing the LED status.*/
  palTogglePad(GPIOE, 8U);
}

/* Application entry point. */
int main(void) {

  /* ChibiOS/HAL and ChibiOS/RT initialization. */
  halInit();
  chSysInit();

  /* Enabling event on falling edge of PA0 signal.*/
  palEnablePadEvent(GPIOA, 0U, PAL_EVENT_MODE_FALLING_EDGE);

 /* Assigning a callback to PA0 passing no arguments.*/
  palSetPadCallback(GPIOA, 0U, button_cb, NULL);

  /* main() thread loop. */
  while (true) {
    /* Doing nothing. This thread can be used for any other
       user activity.*/
    chThdSleepMilliseconds(1000);
  }
}
```


Majd szólok, aludj nyugodtan!

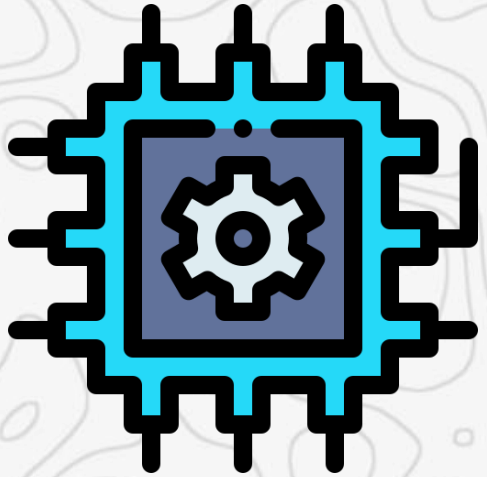
NYOMÓGOMB: PERGÉSMENTESÍTÉS

Hardveres

RC szűrő áramkör

Schmitt trigger áramkör

Flip-Flop áramkör



Szoftveres

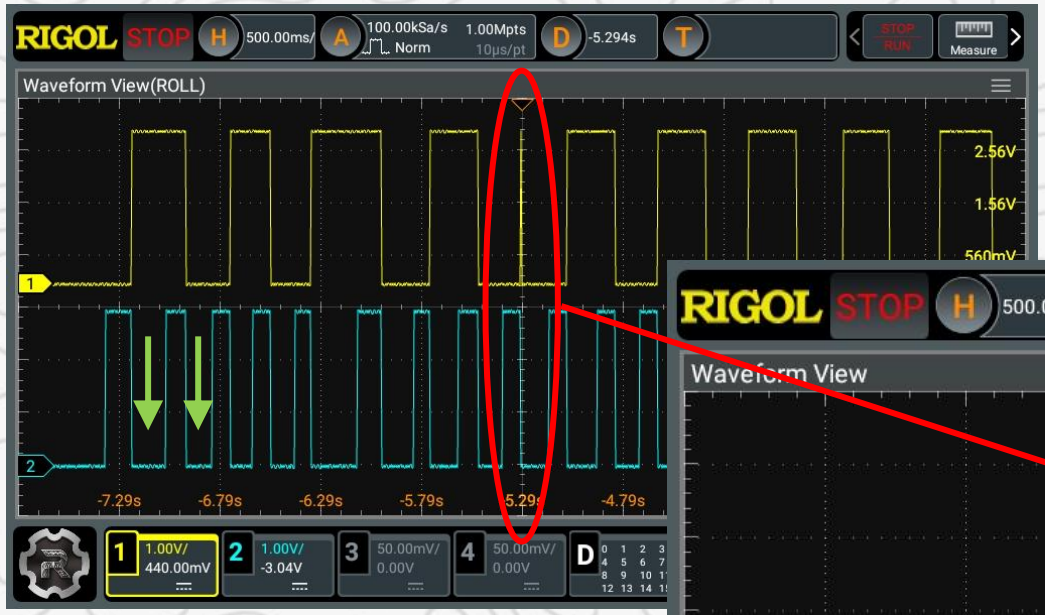
Késleltetés használata

Időzítők használata

Mozgóátlag (digitális szűrő)

Shift-regiszteres (bit-eltolásos)

SZOFTVERES PERGÉSMENTESÍTÉS



Gomb **lefutó** élre **invertálja** a LED-et

Egyszeri gombnyomásra is előfordul **két LED invertálás**

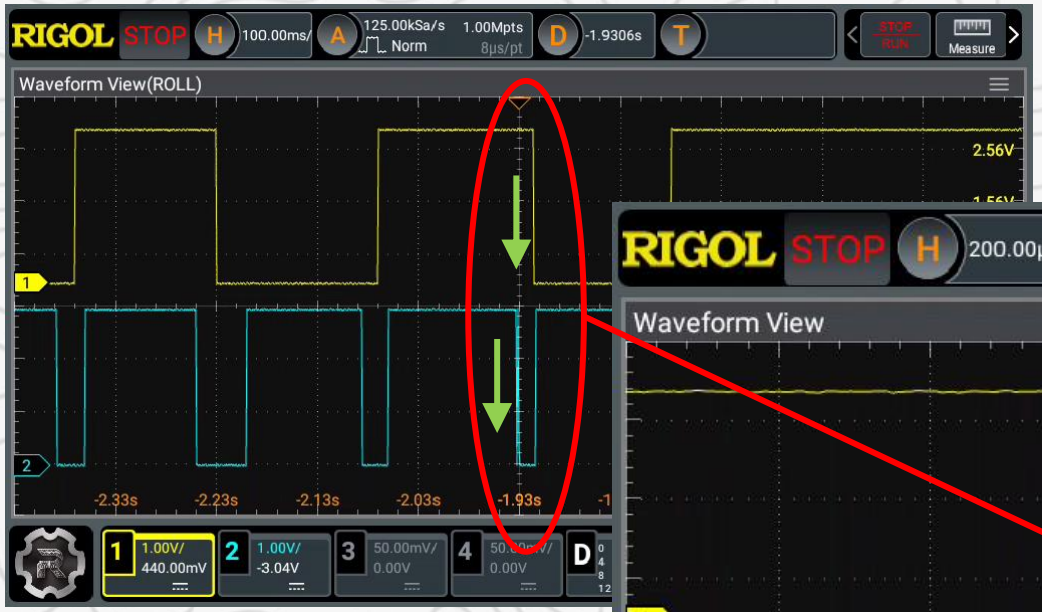
Nélkül

SZOFTVERES PERGÉSMENTESÍTÉS

```
/* LED kimenet alapállapotban kikapcsolva */
palClearPad(LED1_PORT, LED1_PAD);
bool prevState = true; /* feltételezzük: pulldown → nyugalmi LOW */

while (true) {

    bool currState = palReadPad(BUTTON1_PORT, BUTTON1_PAD);
    /* gomb aktív HIGH → akkor van lenyomva, ha currState == 1 */

    if ((prevState == true) && (currState == false)) {
        /*lefutó élváltás: HIGH → LOW → gomb elengedve */
        palTogglePad(LED1_PORT, LED1_PAD);

    }
    prevState = currState;
}
```

A probléma

SZOFTVERES PERGÉSMENTESÍTÉS



Gomb **lefutó** élre **invertálja** a LED-et

Egy bitmintát keresünk pl: **0xFFF0**

Egyszeri gombnyomásra kis késleltetéssel invertálja a LED-et

Vele

SZOFTVERES PERGÉSMENTESÍTÉS

Gomb olvasása while ciklusban 5ms-enként, a következő függvénnyel:

```
/* Debounce függvény (ugyanaz a logika mint az eredeti) */
static bool BtnDebounce(void) {

    static uint16_t Btn1_States = 0;
    /* palReadPad visszaadja 0 vagy 1: 1 = pad logikai magas */
    /* aktív HIGH feltételezve */

    Btn1_States = (uint16_t)((Btn1_States << 1) |
    (palReadPad(BUTTON1_PORT, BUTTON1_PAD)));

    return (Btn1_States == 0xFFF0);
}
```

A megoldás

„Ha beérjük annyival, hogy elátkozzuk vagy dicsőítjük a technikát, akkor sohasem jutunk el lényegének a megragadásához.”

Martin Heidegger

KÖSZÖNÖM A FIGYELMET!

Zsupányi Krisztián

ChibiOS PAL: Nyomógomb kezelés kihívásai