Projektno poročilo: Alarmni sistem na STM32H750B-DK

Predmet: Organizacija računalnikov

Mentor: viš. pred. dr. Robert Rozman

Avtor: Adam Bajt (63210380)

Datum: 29.5.2025

1. Uvod

Koda je dostopna na https://github.com/ByteAdam/stm32h750-alarmni-sistem

Na githubu je prav tako povezava do youtube posnetka: https://youtu.be/KxlVeEdbE8g

V tem projektu sem razvil kompleten alarmni sistem na platformi STM32H7, ki vključuje ultrazvočni senzor HC-SR04 za natančno merjenje razdalje objekta pred napravo, večfazni 60-sekundni odštevalnik z dinamičnimi zvočnimi in svetlobnimi signalizacijami (buzzer in tri rdeče LED), ter mehanizem za vnos šestmestne kode prek fizičnih tipk, zaznavanih z enojnimi EXTI prekinitvami. Sistem samodejno preklopi iz normalnega spremljanja, ko razdalja pade pod prag 100 cm, pri čemer se aktivirajo utripajoče LED in takojšen zvočni signal. Med odštevanjem se signalizacija prilagaja preostalim sekundam (številni kratki pisk in postopno prižiganje LED), v kolikor šestkrat pritisnem pravo kombinacijo tipk (6× gumb 1), se sproži deaktivacija – rdeče lučke ugasnejo, prižge se zelena LED in se zvočni alarm zaustavi. Če kode ni ali je napačna, se števec pritiskov ponastavi ali pa sistem po izteku odštevanja preide v neprekinjen alarm, dokler ni vnesena pravilna koda.

Za komunikacijo z zunanjo aplikacijo sem uvedel UART3 na 115 200 baud s sprejemom prek DMA in prekinitvijo ob idle pogojih, kar omogoča hiter in neblokirajoč prenos ukazov (BTN1…BTN4) ter stalni prenos statusnih sporočil (D: xx cm, T-xx s, Gx:y). Na spletni strani, dostopni v brskalniku Chrome prek Web Serial API, se v realnem času izrisujejo podatki o razdalji, poteku odštevanja, stanju LED in zgodovini dogodkov, hkrati pa lahko prek gumbov na zaslonu simuliram fizične pritiske. Vmesnik podpira tudi uvoz/izvoz CSV dnevnikov serijskih sporočil in dogodkov. Celoten sistem sem zasnoval modularno, z uporabo strojnih časovnikov (DWT za µs natančnost), EXTI prekinitvenega upravljanja vhodov, DMA za minimalno obremenitev CPU in enostavne HTML/JavaScript aplikacije za nadzor in vizualizacijo.

2. Uporabljena oprema in povezave

Funkcija	Arduino Dn	MCU port/pin	Opombe
Ultrazvočni TRIG	—	PC0	2 kΩ upor
Ultrazvočni ECHO	—	PF8	No push/pull, TTL
LED1 (rdeča)	D8	PE3	
LED2 (rdeča)	D9	PH15	
LED3 (rdeča)	D10	PB4	
LED4 (zelena)	D11	PB15	
Gumb 1	D7	PI2	EXTI2, pull-up, GND
Gumb 2	D6	PE6	EXTI6, pull-up, GND
Gumb 3	D5	PA8	EXTI8, pull-up, GND
Gumb 4	D4	PK1	EXTI1, pull-up, GND
Buzzer	D2	PG3	Push-pull digital output
UART3 TX/RX	—	PB10/TX, PB11/RX	115200 baud, 8N1

NVIC – aktivirani interrupti na line1, line2, line[9:5]

DMA – USART3_TX Memory to peripheral, USART3_RX Circular

3. Programska struktura

Program je organiziran kot strojni avtomat s štirimi stanji:
- **MONITOR**: vsakih 500 ms pošljemo impulz ultrazvočnemu senzorju in preberemo razdaljo.
- **COUNTDOWN**: sproži se odštevanje 60 s, LED diode in buzzer signalizirajo po fazah:
  - 60–55 s: vse tri rdeče LED počasi utripajo, kratek beep pri začetku.
  - 55–40 s: ena rdeča LED stalno prižgana, beep pri vsakih 5 s.
  - 40–20 s: dve rdeči LED stalno prižgani, beep vsake 5 s.
  - 20–10 s: tri rdeče LED stalno prižgane, beep pri 20 s in 15 s.
  - 10–0 s: tri rdeče LED hitro utripajo, beep vsako sekundo.
- **ALARM**: po 0 s se prižge neprekinjen alarm (utelovljeno utripanje + stalno piskanje),
- **DISARMED**: po pravilnem vnosu kode (6× Gumb 1) se izklopijo rdeče LED, prižge zelena za 10 s.

4. Podrobnosti implementacije

4.1 Inicializacija periferij

V funkciji `MX_GPIO_Init()` so vse LED in buzzer nastavljeni kot Push-Pull output, ultrazvočni TRIG kot output, ECHO kot input, tipke pa kot GPIO_MODE_IT_FALLING z notranjim pull-up. V `MX_USART3_UART_Init()` je UART3 nastavljen na 115200 baud. V `MX_DMA_Init()` sta inicializirani DMA tokom za UART3 RX in TX. SysTick poskrbi za ms zakasnitve, DWT se inicializira ročno za podmilisekunde.

4.2 DWT za merjenje mikrosekund

DWT (Data Watchpoint & Trace) CPU modul omogoča natančno štetje CPU ciklov. Za merjenje ultrazvočnega signala je bistveno, da znamo izmeriti čas impulza z natančnostjo vsaj nekaj mikrosekund.

```
static inline void DWT_Init(void) {
  CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
  DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
  DWT->CYCCNT = 0;
}
```

Funkcija `micros()` vrne število mikrosekund od inicializacije:
```c
static inline uint32_t micros(void) {
  return DWT->CYCCNT / (SystemCoreClock/1000000);
}
```

4.3 Ultrazvočno merjenje

Funkcija `us_read_cm()` pošlje 10 µs impulz na TRIG, nato čaka na ECHO high, meri čas do padajočega roba in izračuna razdaljo v centimetrih glede na hitrost zvoka (~340 m/s).

```
static uint32_t us_read_cm(void) {
  uint32_t t0, t1, t2;
  HAL_GPIO_WritePin(US_TRIG_PORT, US_TRIG_PIN, GPIO_PIN_SET);
  t0 = micros(); while (micros() - t0 < 10);
  HAL_GPIO_WritePin(US_TRIG_PORT, US_TRIG_PIN,
GPIO_PIN_RESET);
  t0 = micros();
  while (HAL_GPIO_ReadPin(US_ECHO_PORT, US_ECHO_PIN) ==
GPIO_PIN_RESET)
    if (micros() - t0 > 30000) return UINT32_MAX;
  t1 = micros();
  while (HAL_GPIO_ReadPin(US_ECHO_PORT, US_ECHO_PIN) ==
GPIO_PIN_SET)
    if (micros() - t1 > 30000) break;
  t2 = micros();
  return (t2 - t1) / 58;
}
```

4.4 EXTI prekinitve za tipke

Vsaka tipka sproži prekinitveno rutino `HAL_GPIO_EXTI_Callback`, kjer se štejejo pritiski v tabeli `pressCnt[]`. Po vsakem pritisku se izpiše stanje vse štirih gumbov.

```
void HAL_GPIO_EXTI_Callback(uint16_t pin) {
  if (pin == BTN1_PIN) pressCnt[0]++;
  else if (pin == BTN2_PIN) pressCnt[1]++;
  else if (pin == BTN3_PIN) pressCnt[2]++;
  else if (pin == BTN4_PIN) pressCnt[3]++;
  attemptCnt++;
  printf("G1:%lu G2:%lu G3:%lu G4:%lu\r\n",
      pressCnt[0], pressCnt[1], pressCnt[2], pressCnt[3]);
}
```

4.5 UART RX z DMA in WebApp

Za sprejem serijskih ukazov iz brskalnika (WebApp) uporabljamo DMA v "receive-to-idle" načinu. Funkcija `HAL_UARTEx_ReceiveToIdle_DMA` omogoči prekinitve ob prejmu podatkov, kar preprečuje polling.

```
HAL_UARTEx_ReceiveToIdle_DMA(&huart3,  rxBuf,
RX_DMA_BUFSIZE);

void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef  *hu,
uint16_t size) {
  static char line[64]; static uint8_t pos = 0;
  for (uint16_t i = 0; i < size; i++) {
    char c = rxBuf[i];
    if (c == '\r' || c == '\n') {
      line[pos]  = 0;
      if (!strcmp(line, "BTN1")) HAL_GPIO_EXTI_Callback(BTN1_PIN);
      // ... analogno za BTN2–BTN4
      pos = 0;
    } else if (pos < sizeof(line)-1) {
      line[pos++]  = c;
    }
  }
  HAL_UARTEx_ReceiveToIdle_DMA(&huart3,  rxBuf,
RX_DMA_BUFSIZE);
}
```

V WebApp JavaScript del pošilja ukaze "BTNn" prek serial.write in posodobi UI z razredi CSS `led-on` / `led-off`, beleži dogodke in omogoča izvoz CSV.

5. Zaključek

Projekt uspešno združuje vse zahtevane periferije in funkcionalnosti: natančno merjenje razdalje, strojni avtomat z večfaznim odštevanjem, prekinitve tipk, DMA za UART sprejem, blokirajoče UART pošiljanje, spletni nadzor prek Web Serial API in dodatne možnosti izpisa v CSV. Možne nadgradnje vključujejo DMA za TX, kompozitni signal na WebApp ter strojno dekodiranje tipk.

Celotna koda:

https://github.com/ByteAdam/stm32h750-alarmni-sistem/blob/main/Core/Src/main.c