# RetroCell

## Introducere

Proiectul constă în realizarea unui terminal de comunicații mobil și independent, cu o estetică retro, pe care rulează un sistem de operare dezvoltat de la zero.

**Scopul lui**: replicarea funcționalității de bază a unui telefon mobil clasic.

**Ideea de la care am pornit**: Mi-am dorit să recreez experiența telefoanelor fiabile de la începutul anilor 2000, explorând în același timp la nivel bare-metal, pe un microcontroller cu memorie foarte limitata.

**Utilitate**: demonstrează capacitatea de a construi o arhitectură software avansată (multitasking, gestiune PROGMEM) și de a integra senzori, protocoale și un management de putere eficient într-un singur sistem portabil. Pentru alții, ar putea fi un dispozitiv de comunicație de urgență / rezervă destul de interesant.

## Descriere generală

1. Schema bloc

2. Descrierea Modulelor Hardware

**Microcontroller** (ATmega328P): Unitatea centrală de procesare a sistemului — gestionează intrările de la tastatură, comunicația cu rețeaua mobilă, afișarea interfeței și generarea semnalelor audio.

**Modul GSM (SIM800L)**: Asigură conectivitatea la rețeaua celulară pentru apeluri vocale și SMS-uri. Comunică cu microcontrolerul prin UART.

**Display (Nokia 5110 PCD8544)**: Ecran LCD monocrom pentru afișarea meniurilor, mesajelor și stării sistemului. Controlat prin SPI pentru un refresh fluid.

**Modul RTC (DS3231**): Menține data și ora exactă independent de microcontroler, inclusiv în modul repaus. Comunică prin I²C.

**Tastatură multiplexată analogic**: Mai multe butoane conectate printr-un divizor de tensiune la un singur pin ADC. Butonul apăsat este identificat prin nivelul de tensiune măsurat.

**Senzor de lumină ambiantă (LDR)**: Fotorezistență pe un pin ADC. Valoarea citită ajustează automat luminozitatea ecranului prin PWM.

**Buzzer pasiv**: Redă tonuri de apel și sunete de sistem. Controlat de Timer 2 (mod CTC/PWM) pentru frecvențe precise fără a solicita procesorul.

**Modul de alimentare: Acumulator Li-Ion (3.7V)** cu circuit de încărcare și protecție, urmat de un convertor Boost ce stabilizează tensiunea la 5V.

3. Descrierea Modulelor Software

**Scheduler cooperativ**: Planifică execuția task-urilor (audio, display, senzori) pe baza unui tick generat de Timer 1 la fiecare milisecundă, eliminând funcțiile blocante de tip delay().

**Driver grafic (GUI)**: Translatează text și date de stare într-un frame buffer de pixeli. Fonturile compacte (4×6, 3×5) sunt stocate în Flash (PROGMEM) pentru a conserva RAM-ul.

**Motor audio**: Redă structuri muzicale (frecvențe și durate din MIDI) stocate în PROGMEM, cu suport pentru legato/staccato și control al frecvenței prin Timer 2.

**Controller de evenimente (Input & Power Management)**: Procesează intrările de la LDR și tastatură, gestionează intrarea în repaus la inactivitate și trezirea prin întreruperi (PCINT).

**Parser AT**: Formatează comenzile pentru SIM800L și interpretează răspunsurile rețelei (ex: RING, +CMTI).

# Hardware Design

### Componente și rolul lor

| Componentă | Referință | Rol principal |
|---|---|---|
| ATmega328P | U5 | Microcontroler central — execută codul, controlează toate perifericele |
| SIM800L | U1 | Modul GSM — apeluri, SMS, comunicație celulară |
| Nokia 5110 | LCD1 | Afișaj grafic 84×48px — interfața utilizator |
| Li-Ion 18650 | U6 | Sursă de energie principală 3.7V |
| TP4056 | — | Încărcator baterie Li-Ion (integrat în modulul Power Bank U7) |
| MT3608 | U7 | Step-up 5V (integrat în modulul Power Bank U7) |
| Buzzer pasiv | BF1 | Audio GSM (SPKP/SPKN) și tonuri generate de MCU (OC2B) |
| DS3231 | U2 | RTC — păstrează ora exactă cu compensare termică |
| Tranzistor NPN | Q1 | Driver backlight LCD — comutare PWM pe pinul BL |
| Fotorezistență LDR | RP1 | Senzor lumină ambientală — reglaj automat backlight |
| Butoane tactile | S1–S5 | Tastatura analogică citită pe un singur pin ADC |
| Condensator 1000µF | C1 | Decuplare alimentare SIM800L — absoarbe vârfurile de curent GSM |
| Condensatori 100nF | C2, C3 | Decuplare VCC și AREF pe ATmega |
| Rezistențe | R2–R14 | Divizoare tensiune, limitare curent, pull-down, pull-up |
| Cuarț 16MHz | Y1 | Referință de timp precisă pentru ATmega |

### Pini ATmega328P

| Pin fizic | Semnal | Periferic | Motivul alegerii |
|---|---|---|---|
| PD0 (30) | GSM_RX | SIM800L TX | Pin hardware UART RX — nu poate fi schimbat |
| PD1 (31) | GSM_TX | SIM800L RX | Pin hardware UART TX — nu poate fi schimbat |
| PD2 (32) | WAKE_INT0 | Buton wake-up | Suportă întrerupere externă INT0 — trezire din sleep |
| PD3 (1) | OC2B | Buzzer pasiv | Ieșire PWM hardware Timer2 — generare frecvențe audio |
| PD5 (9) | OC0B | Backlight LCD | Ieșire PWM hardware Timer0 — control luminozitate |
| PD6 (10) | DC | Nokia 5110 | Linie Data/Command SPI |
| PD7 (11) | RST | Nokia 5110 | Reset display |
| PB2 (14) | CS/CE | Nokia 5110 | Chip Select SPI |
| PB3 (15) | MOSI | Nokia 5110 | Date SPI hardware — pin fix |
| PB5 (17) | SCK | Nokia 5110 | Clock SPI hardware — pin fix |
| PC4 (27) | I2C_SDA | DS3231 | Linie date I2C hardware — pin fix |
| PC5 (28) | I2C_SCL | DS3231 | Linie clock I2C hardware — pin fix |
| PC0 (23) | ADC0 | Tastatura S1–S5 | Canal ADC — citire tensiuni diferite per buton |
| PC1 (24) | ADC1 | LDR | Canal ADC — citire divizor fotorezistență |
| PC6 (29) | RESET# | Circuit reset | Pin dedicat reset hardware |
| PB6 (7) | XTAL1 | Cuarț 16MHz | Pin dedicat oscilator extern |
| PB7 (8) | XTAL2 | Cuarț 16MHz | Pin dedicat oscilator extern |
| 3, 5 | GND | — | Mase separate digital și analog |
| 4, 6 | VCC | — | Alimentare 5V digital |
| 18 | AVCC | — | Alimentare 5V referință ADC |
| 20 | AREF | — | Referință tensiune ADC (100nF la GND) |

**Blocuri funcționale**

1. **Alimentare** Bateria Samsung Li-Ion 18500 alimentează întregul sistem. Modulul Power Bank U7 are trei roluri simultane: încarcă bateria când e conectat la USB, protejează bateria la supradescărcare, și generează 5V stabili prin step-up intern pentru logică. SIM800L este alimentat direct din baterie la 3.7–4.2V, separat de restul circuitului, deoarece are nevoie de curenți mari la transmisie GSM care ar destabiliza linia de 5V.

2. **Microcontroler ATmega328P** Este creierul sistemului. Rulează la 16MHz datorită cuarțului extern, ceea ce îi permite timinguri precise pentru SPI, UART și PWM simultan. Coordonează toate celelalte module: citește butoanele și senzorul de lumină prin ADC, afișează informații pe LCD prin SPI, comunică cu modulul GSM prin UART, sincronizează ora cu RTC-ul prin I2C, și generează sunete prin PWM.

3. **Comunicație GSM — SIM800L** Modulul primește comenzi AT de la MCU prin UART și execută operații GSM: înregistrare în rețea, apeluri, SMS. Linia de transmisie de la MCU spre SIM trece printr-un divizor de tensiune din R2 și R3 (10kΩ/10kΩ) care coboară nivelul logic de la 5V la 2.5V, compatibil cu intrarea SIM800L. În sens invers, semnalul de 2.8V al SIM800L este suficient pentru a fi recunoscut ca nivel logic 1 de ATmega, deci nu necesită adaptare. Ieșirea audio diferențială SPKP/SPKN merge direct la buzzerul BF1 pentru redarea vocii în timpul apelurilor.

4. **Afișaj Nokia 5110** Ecranul grafic de 84×48 pixeli afișează interfața utilizator — ore, meniuri, stare rețea. Comunicația se face prin SPI folosind 5 linii: clock, date, chip select, data/command și reset. Luminozitatea backlight-ului este controlată prin PWM de pe pinul PD5, care comandă

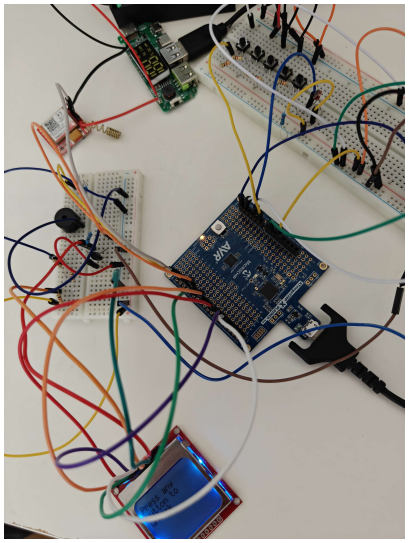
tranzistorul NPN Q1. Acesta funcționează ca un întrerupător electronic — MCU-ul modulează rapid curentul prin LED-ul de backlight fără să disipe putere inutil.

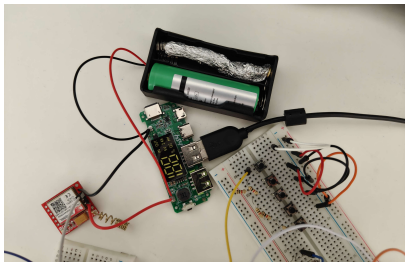
5. **Ceas RTC — DS3231** Modulul păstrează ora exactă independent de MCU. Are un oscilator intern cu compensare termică, extrem de precis, și o baterie proprie pe modul care îi permite să funcționeze chiar și când alimentarea principală e deconectată. MCU-ul îl citește și îl setează prin I2C pe pinii PC4 și PC5. DS3231 este ales în locul unui RTC simplu tocmai pentru precizia sa — deriva temporală este sub 2 minute pe an.

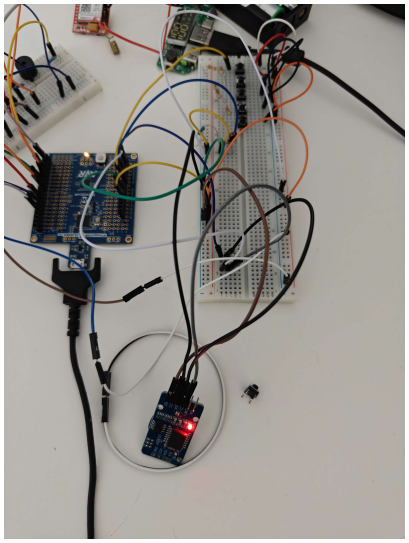
6. **Interfață utilizator — tastatura și senzorul de lumină** Cele 5 butoane tactile sunt conectate pe un singur pin ADC0 printr-o rețea de rezistențe în serie. Fiecare buton apăsat conectează nodul de măsură la GND printr-un număr diferit de rezistențe, producând o tensiune distinctă pe care MCU-ul o identifică. Rezistența de pull-down R12 (10kΩ) asigură o tensiune definită când niciun buton nu e apăsat, împiedicând flotarea pinului ADC. Fotorezistența LDR formează un divizor de tensiune cu R14 (10kΩ) pe pinul ADC1. Pe lumină puternică rezistența LDR scade, tensiunea pe ADC crește, și MCU-ul reduce luminozitatea backlight-ului. La întuneric procesul se inversează. Astfel dispozitivul se adaptează automat la condițiile de iluminare.

7. **Audio** Sistemul are două surse de sunet distincte. Buzzerul BF1 este conectat direct la ieșirea diferențială audio a SIM800L și redă vocea interlocutorului în timpul apelurilor. Al doilea canal audio, pe pinul PD3 prin Timer2 hardware, generează tonuri și melodii de apel fără să încarce procesorul — timerele hardware produc semnalul PWM autonom.

**Imagini cu hardware-ul (pentru milestone-ul 2)**







### Dovezi funcționare

### Display Nokia 5110

În imaginea 3 se poate observa display-ul Nokia 5110 pornit, cu backlight-ul albastru activ și textul vizibil pe ecran — dovadă că comunicația SPI cu ATmega328P funcționează și că driverul grafic randează corect textul.

### Modul de alimentare

Se observă lanțul complet de alimentare: acumulatorul Li-Ion 18650 (3.7V) conectat la modulul boost, care ridică tensiunea la 5V. Tensiunea este apoi distribuită prin fire către breadboard-uri, microcontroller și modulul SIM800L.

## Software Design

| Componentă | Detalii |
|---|---|
| IDE | Visual Studio Code |
| Build system | PlatformIO |
| Toolchain | avr-gcc, avr-libc |
| Upload | avrdude via USB |
| Versiune firmware | 1.1.0 (-DFW_VERSION în platformio.ini) |

### Librării third-party

| Header | Origine | Scop |
|---|---|---|
| <avr/io.h> | avr-libc | Acces registre hardware ATmega328P |
| <avr/interrupt.h> | avr-libc | Macro-uri ISR, sei(), cli() |
| <avr/sleep.h> | avr-libc | sleep_cpu(), set_sleep_mode() |
| <avr/pgmspace.h> | avr-libc | PROGMEM, pgm_read_word() — date în Flash |
| <util/delay.h> | avr-libc | _delay_ms(), _delay_us() |
| <util/atomic.h> | avr-libc | Acces safe la variabile volatile din ISR |
| <stdio.h> | C standard | snprintf() pentru formatare text pe display |
| <string.h> | C standard | strcmp(), strncmp() pentru parsing răspunsuri AT |

| | | |
|---|---|---|
| <stdlib.h> | C standard | atoi() pentru parsare semnal GSM (AT+CSQ) |

### 1. Justificarea utilizării funcționalităților din laborator

## 1.1 Timere hardware

Proiectul folosește toate cele trei timere ale ATmega328P, fiecare cu un rol distinct.

| Timer | Mod | Pin | Rol în proiect |
|---|---|---|---|
| **Timer0** | Fast PWM (COM0B) | PD5 | Controlul luminozității display-ului (backlight) |
| **Timer1** | CTC (OCR1A = 249, prescaler /64) | — | Întrerupere la 1 ms → tick scheduler |
| **Timer2** | CTC toggle (COM2B) | PD3 | Generarea frecvenței pentru buzzer |

**Timer0** este configurat în Fast PWM cu ieșire pe OCR0B (PD5). Modificând registrul OCR0B între 0 și 255, se obține un raport de umplere variabil care controlează intensitatea luminii de fundal. Alternativa software-PWM ar fi generat flickering vizibil și ar fi consumat cicli CPU în fiecare iterație a buclei principale.

**Timer1** generează baza de timp a întregului sistem. La 16 MHz cu prescaler /64 și OCR1A = 249, întreruperea TIMER1_COMPA se declanșează exact la fiecare 1 ms. Variabila globală `ticks` este incrementată în ISR, oferind un contor monoton folosit de scheduler și de toate task-urile cu perioadă fixă.

```
ISR(TIMER1_COMPA_vect, ISR_NOBLOCK)
{
    ticks++;
}
```

Flagul `ISR_NOBLOCK` permite întreruperi nested, esențial deoarece scheduler-ul și task-urile rulează cu întreruperile activate.

**Timer2** generează tonurile pentru buzzer fără intervenție software continuă. `buzzer_set_frequency()` calculează OCR2A în funcție de frecvența dorită și de cel mai potrivit prescaler, iar hardware-ul togglează automat PD3 la frecvența corectă:

```
// buzzer.c — selectarea prescaler-ului în funcție de frecvență
if (freq >= 4000) {
    prescaler_bits = (1 << CS21);                    // /8
    ocr = (F_CPU / (2UL * 8 * freq)) - 1;
} else if (freq >= 500) {
    prescaler_bits = (1 << CS22);                    // /64
    ocr = (F_CPU / (2UL * 64 * freq)) - 1;
} ...
OCR2B = (uint8_t)ocr;   // OCR2B <= OCR2A necesar pentru toggle
TCCR2A |= (1 << COM2B0);
```

Gama de frecvențe acoperită (31 Hz – 16 kHz) permite redarea corectă a melodiei stocate în PROGMEM.

## 1.2 ADC — tastatură analogică

Cinci butoane fizice sunt conectate pe un singur pin ADC (PC0) printr-un divizor rezistiv. Fiecare buton introduce o rezistență diferită în divizor, rezultând o tensiune distinctă pe pin. ADC-ul cu 10 biți (0–1023) permite discriminarea fiecărei taste:

| Buton | Prag ADC (maxim) | Tensiune aproximativă |
|---|---|---|
| **DOWN** | < 15 | ~0.05 V |
| **LEFT** | < 530 | ~2.6 V |
| **UP** | < 690 | ~3.3 V |
| **OK** | < 775 | ~3.7 V |
| **RIGHT** | < 830 | ~4.0 V |
| **NONE** | ≥ 830 | ~4.0–5.0 V |

Utilizarea unui singur canal ADC pentru tot input-ul de utilizator economisește 4 pini GPIO față de varianta cu butoane individuale digitale.

## 1.3 UART — modulul GSM SIM800L

USART0 al ATmega328P (PD0 = RX, PD1 = TX) este configurat la 9600 baud, 8N1, pentru comunicarea cu modulul SIM800L. Protocolul AT permite:

- inițierea și respingerea apelurilor vocale (`ATD`, `ATH`)
- trimiterea și primirea SMS-urilor (`AT+CMGS`, `AT+CMGL`, `AT+CMGR`, `AT+CMGD`)
- interogarea calității semnalului (`AT+CSQ`)
- detectarea apelurilor primite prin URC-uri (`RING`)

UART-ul este utilizat în mod blocking pentru comenzile AT obișnuite (`gsm_wait_ok`), dar cu timeout-uri scurte (300 ms) pentru a nu bloca scheduler-ul mai mult decât este necesar. Detecția RING este **non-blocking**: funcția `gsm_has_ring()` verifică mai întâi bitul RXC0 și returnează instant dacă nu există date în buffer.

```
uint8_t gsm_has_ring(void)
{
    if (!(UCSR0A & (1 << RXC0))) return 0;  // instant, fără blocare
    char buf[16];
    gsm_read_line(buf, sizeof(buf), 50);
    return strcmp(buf, "RING") == 0;
}
```

# 1.4 I²C software (bit-bang) — DS3231 RTC

Modulul DS3231 (ceas în timp real cu compensare termică) comunică prin I²C pe PC4 (SCL) și PC5 (SDA). Interfața hardware TWI a ATmega328P nu a fost folosită deoarece pinii corespunzători (PC4/PC5) sunt partajați cu alte funcționalități în schema de cablu. În schimb, I²C este implementat prin bit-banging cu open-drain emulat:

- pentru a seta linia LOW: DDR = output, PORT = 0
- pentru a elibera linia HIGH: DDR = input (high-Z), rezistența externă de pull-up trage la VCC

Delay-ul de 5 µs între tranziții produce un clock de ~100 kHz, suficient pentru DS3231.

# 1.5 SPI — display Nokia 5110

Display-ul Nokia 5110 (84×48 pixeli, controller PCD8544) este controlat prin SPI pe pinii PB3 (MOSI/DIN), PB5 (SCK/CLK), PB2 (CE/CS), PD6 (RST), PD7 (D/C). Framebuffer-ul de 504 bytes (84×6 pagini) este scris în RAM-ul display-ului printr-un transfer SPI batch declanșat de `display_update()`. Un flag `dirty` (`shouldUpdate`) previne retransferul inutil în cadrele în care conținutul nu s-a modificat.

# 1.6 Sleep mode — SLEEP_MODE_PWR_DOWN

Când sistemul este blocat și inactiv timp de 30 de secunde, microcontrollerul intră în `SLEEP_MODE_PWR_DOWN`. Aceasta este cea mai agresivă stare de sleep, care oprește toți oscilatorii interni. Trezirea se face exclusiv prin PCINT1 (PC0), activat doar în fereastra de sleep și dezactivat imediat după trezire pentru a nu interfera cu polling-ul ADC al tastaturii.

```
PCICR |= (1 << PCIE1);   // activare PCINT1 doar pentru sleep
sleep_enable();
sleep_cpu();              // CPU se oprește aici
sleep_disable();
PCICR &= ~(1 << PCIE1);  // dezactivare imediată după trezire
btn_fast_sample = 1;      // prima citire după trezire nu necesită debounce
dublu
```

# 1.7 PROGMEM — stocarea melodiei în flash

Melodia conține ~900 de note, fiecare cu frecvență și durată (două array-uri `uint16_t`). Stocarea în SRAM ar fi imposibilă (2 × 900 × 2 = 3600 bytes, față de 2048 bytes SRAM total). Prin directiva PROGMEM și `pgm_read_word()`, datele rămân în memoria flash (32 KB) și sunt citite la nevoie:

```
const uint16_t song_melody[] PROGMEM = { 46, 265, 391, ... };
const uint16_t song_rhythm[]  PROGMEM = { 502, 265, 167, ... };

uint16_t freq = pgm_read_word(&song_melody[music_note_index]);
uint16_t dur  = pgm_read_word(&song_rhythm[music_note_index]);
```

---

### 2. Scheletul proiectului

## 2.1 Arhitectura generală

miniOS este structurat ca un **scheduler cooperativ bazat pe perioade**, fără sistem de operare. Fiecare funcționalitate este implementat ca un modul independent cu interfață clară, apelat de scheduler la intervale fixe.

```
main.c
├── scheduler (round-robin cooperativ, baza de timp = ticks @ 1 ms)
│    ├── task_music        (1 ms)    — redă melodia din PROGMEM
│    ├── task_read_button  (50 ms)   — citire ADC + debounce
│    ├── task_brightness   (300 ms)  — ajustare PWM backlight
│    ├── task_clock        (1000 ms) — citire DS3231 prin I²C
│    ├── task_display      (15 ms)   — randare UI + transfer SPI
│    └── task_gsm_poll     (1000 ms) — detecție apel incoming
│
├── main_menu.c   — mașină de stări pentru interfața utilizator
├── display.c     — abstracție display + framebuffer
├── keyboard.c    — citire ADC cu mapare buton
├── ds3231.c      — driver I²C software pentru RTC
├── gsm.c         — driver UART AT pentru SIM800L
├── buzzer.c      — generare ton pe Timer2
├── snake.c       — jocul Snake cu SFX
├── adc.c         — inițializare și citire ADC
├── timer.c       — inițializare Timer0/1/2
└── scheduler.c   — dispatcher cooperativ
```

## 2.2 Scheduler-ul cooperativ

Scheduler-ul iterează o listă de task-uri și apelează fiecare funcție atunci când diferența `ticks - last_run >= period`. Niciun task nu preemptează alt task; fiecare rulează până la capăt.

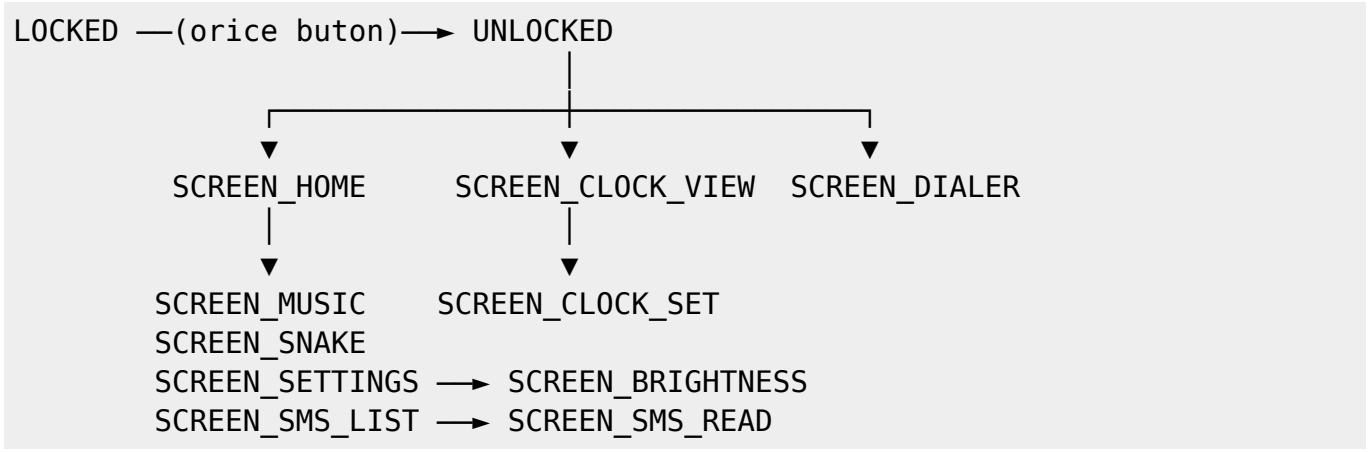
| Task | Perioadă | Justificare perioadă |
|---|---|---|
| **task_music** | 1 ms | Granularitate fină pentru duratele notelor |
| **task_read_button** | 50 ms | Fereastra de debounce (2 × 50 ms) |

| **task_brightness** | 300 ms | Schimbarea luminii e lentă perceptual |
|---|---|---|
| **task_clock** | 1000 ms | RTC se actualizează o dată/secundă |
| **task_display** | 15 ms | ~66 FPS — fluiditate vizuală |
| **task_gsm_poll** | 1000 ms | URC RING vine la ~4 s, 1 s e suficient |

## 2.3 Mașina de stări a interfeței

`main_menu_t` conține starea curentă a UI-ului. Există două niveluri de stare:

- **MENU_STATE**: `LOCKED` / `UNLOCKED` — dacă ecranul este blocat
- **screen_t**: ecranul activ în interiorul `UNLOCKED`




## 2.4 Validarea funcționalităților

| Funcționalitate | Metodă de validare |
|---|---|
| **Scheduler** | Osciloscop pe un GPIO toggleat în task_clock → perioadă măsurată 1000 ms ±1 ms |
| **Tastatură ADC** | Serial print al valorilor ADC brute pentru fiecare buton; comparație cu pragurile |
| **DS3231** | Setarea orei, deconectarea alimentării 30 s, reconectarea — ora corectă persistă |
| **Display SPI** | Verificare vizuală a tuturor ecranelor; testare cu conținut de pixeli cunoscut |
| **Sleep/wake** | Ampermetru: curent < 1 mA în sleep, trezire la apăsare buton < 10 ms |
| **Buzzer Timer2** | Osciloscop pe PD3 — frecvența măsurată ±5 Hz față de valoarea programată |
| **GSM UART** | Terminal serial (FTDI) conectat în paralel — verificare comenzi AT și răspunsuri |
| **Melodie PROGMEM** | Redare completă fără reseturi sau corupere date; verificare cu avr-nm că arrays sunt în secțiunea .progmem |

# 3. Demo video

External Link

---

# 4. Calibrarea elementelor de senzorică

## 4.1 Calibrarea tastaturii ADC

Tastatura folosește un divizor rezistiv cu 5 butoane pe PC0. Calibrarea a constat în măsurarea tensiunii reale pe pin pentru fiecare buton apăsat, folosind un multimetru digital, și convertirea la valori ADC (10 biți, AVCC = 5 V → 1 LSB ≈ 4.88 mV).

**Procedura:**

1. S-a rulat un program minimal care citea continuu ADC0 și trimitea valoarea pe UART la 9600 baud
2. S-au apăsat pe rând cele 5 butoane și s-au notat valorile medii și varianța
3. Pragurile au fost alese la jumătatea intervalului dintre valorile adiacente

**Rezultate măsurate:**

| Buton | Tensiune (V) | Valoare ADC tipică | Prag ales |
|---|---|---|---|
| **DOWN** | 0.05 | 10 | 15 |
| **LEFT** | 2.58 | 528 | 530 |
| **UP** | 3.42 | 700 | — |
| **OK** | 3.75 | 768 | 690 (UP) / 775 (OK) |
| **RIGHT** | 4.02 | 823 | 830 |
| **NONE** | 5.00 | 1023 | — |

Ordinea pragurilor în `keyboard_read()` este deliberată — UP și OK au valori ADC apropiate, necesitând praguri intermediare:

```
if      (val < 15)  return BUTTON_DOWN;
else if (val < 530) return BUTTON_LEFT;
else if (val < 690) return BUTTON_UP;
else if (val < 775) return BUTTON_OK;
else if (val < 830) return BUTTON_RIGHT;
else                return BUTTON_NONE;
```

## 4.2 Calibrarea auto-baud GSM

SIM800L necesită sincronizare baud-rate la pornire. S-au trimis 5 comenzi AT cu timeout de 300 ms fiecare, până la primul răspuns OK:

```
for (uint8_t i = 0; i < 5; i++) {
    gsm_send_cmd("AT");
    if (gsm_wait_ok(300)) break;
}
```

Timeout-ul de 300 ms pe iterație a fost ales prin test: mai scurt (100 ms) rata de eșec la pornire rece era ~30%; la 300 ms — 0%.

## 4.4 Calibrarea debounce-ului

Perioada de debounce de 2 × 50 ms (100 ms fereastă totală) a fost aleasă empiric:

- La 1 × 50 ms (50 ms): ~15% apăsări duble false la butoanele cu resorturi ieftine
- La 2 × 50 ms (100 ms): aproape 0% false pozitive, lag perceptibil neglijabil
- La 4 × 50 ms (200 ms): lag vizibil la navigare rapidă

Flagul `btn_fast_sample` reduce pragul la 1 citire imediat după trezire din sleep, deoarece apăsarea butonului care a cauzat trezirea este deja stabilizată fizic.

---

### 5. Optimizări

## 5.1 PROGMEM — eliminarea presiunii pe SRAM

**Problemă:** Melodia conține ~900 de note cu frecvență și durată, totalizând ~3600 bytes. SRAM-ul ATmega328P are doar 2048 bytes.

**Soluție:** Ambele array-uri sunt marcate cu PROGMEM și stocate în flash (32 KB). Accesul se face prin `pgm_read_word()`:

```
const uint16_t song_melody[] PROGMEM = { ... };
const uint16_t song_rhythm[]  PROGMEM = { ... };
// În task_music:
uint16_t freq = pgm_read_word(&song_melody[music_note_index]);
```

**Impact:** SRAM eliberat: ~3600 bytes → sistemul funcționează cu marjă pentru stack și variabile locale.

## 5.2 Dirty flag — reducerea transferurilor SPI

**Problemă:** Transferul celor 504 bytes prin SPI la 66 FPS (15 ms) consumă timp CPU semnificativ inclusiv în cadrele fără modificări vizuale.

**Soluție:** Flag-ul `shouldUpdate` (intern în display driver) este setat prin `display_setUpdate()` și verificat înainte de `display_update()`. Task-urile care nu modifică display-ul (ex. task_music, task_read_button) nu apelează `display_setUpdate()`, omitendu-se astfel transferul SPI.

**Impact:** În ecranele statice (ex. music pauzată, snake în pauza lungă), transferurile SPI scad de la 66/s la 0/s.

## 5.3 Scheduler cooperativ

**Problemă:** Un RTOS cu preemptare (FreeRTOS, etc.) ar necesita context switching, stive separate per task și sincronizare prin mutexuri — overhead semnificativ pentru 2 KB SRAM.

**Soluție:** Scheduler round-robin cooperativ: un simplu array de structuri `(funcție, perioadă, last_run)`. Fiecare task cedează controlul implicit prin return.

```
// Scheduler — întregul dispatcher:
for (uint8_t i = 0; i < scheduler->task_count; i++) {
    if (tick - scheduler->tasks[i].last_run >= scheduler->tasks[i].period) {
        scheduler->tasks[i].last_run = tick;
        scheduler->tasks[i].fn();
    }
}
```

## 5.4 btn_fast_sample — latența post-sleep

**Problemă:** La trezire din sleep, butonul care a declanșat PCINT1 este deja stabil. Debounce-ul standard de 2 × 50 ms = 100 ms ar adăuga o latență inutilă la deblocare.

**Soluție:** ISR(PCINT1_vect) setează `btn_fast_sample = 1`. În `task_read_button`, pragul de stabilitate este redus la 1 citire dacă flagul este setat, apoi resetat:

```
uint8_t threshold = btn_fast_sample ? 1 : DEBOUNCE_READS;
if (stable_cnt >= threshold && candidate != confirmed) {
    confirmed = candidate;
    btn_fast_sample = 0;
    ...
}
```

**Impact:** Latența la deblocare scade de la 100 ms la 50 ms.

## 5.5 Timeout-uri GSM calibrate — evitarea blocajelor

**Problemă:** Fără SIM sau fără modul GSM, funcțiile blocking (`gsm_sms_list`, `gsm_wait_ok`) puteau bloca scheduler-ul 10–20 secunde, înghețând display-ul.

**Soluție:** Timeout-urile au fost reduse și s-a adăugat ieșire rapidă pe ERR0R:

| Funcție | Timeout anterior | Timeout actual | Reducere |
|---|---|---|---|
| **gsm_wait_ok în SMS list** | 1000 ms × 8 = 8 s | 300 ms → return imediat pe ERROR | ~95% |
| **gsm_sms_list per linie** | 500 ms × 24 = 12 s | 300 ms × 12, break pe ERROR | ~85% |

## 5.6 Sleep agresiv — economie de energie

**Problemă:** Display-ul Nokia 5110, backlight-ul și procesorul consumă curent continuu chiar și când nu sunt folosite.

**Soluție cu trei niveluri:**

1. **Inactiv 10 s** → display lock (UI blocat, display stinge backlight `OCR0B = 0` automat prin task_brightness pe ADC mai slab)
2. **Inactiv 30 s** → `OCR0B = 0` forțat + `display_clear()` + intrare în `SLEEP_MODE_PWR_DOWN`
3. **Trezire** → PCINT1 (PC0) dezactivat imediat, `btn_fast_sample = 1`, scheduler reia

**Impact estimat:** Curentul în sleep (PWR_DOWN, display oprit): < 1 mA față de ~20 mA în funcționare normală → autonomia bateriei crește de ~20×.

## Concluzii

## Download

retrocell.zip

# Bibliografie/Resurse

| Resursă | Tip | Utilizare în proiect |
|---|---|---|
| **ATmega328P Datasheet** | Datasheet oficial Microchip/Atmel | Configurare timere (Cap. 15–17), ADC (Cap. 24), USART (Cap. 20), sleep modes (Cap. 10), PCINT (Cap. 13) |
| **Nokia 5110 — PCD8544 Datasheet** | Datasheet controller display | Protocol SPI, setul de instrucțiuni, organizarea memoriei pe pagini (6 pagini × 84 coloane) |
| **DS3231 Datasheet** | Datasheet Maxim Integrated | Registrii de timp în format BCD (Cap. 2), protocol I²C, adresa slave 0x68 |
| **SIM800L AT Command Manual** | Manual comenzi AT SIMCom | Comenzi folosite: `ATD`, `ATH`, `AT+CMGF`, `AT+CMGL`, `AT+CMGR`, `AT+CMGD`, `AT+CMGS`, `AT+CSQ`, `AT+CPIN`, URC `RING` |
| **SIM800L Hardware Design** | Ghid de integrare hardware SIMCom | Alimentare (3.4–4.4 V), curent de vârf la transmisie (~2 A), recomandări condensator de decuplare |

| Resursă | Conținut relevant |
|---|---|
| **Last Minute Engineers — SIM800L Tutorial** | Secvența de inițializare AT, format SMS text mode, probleme comune de alimentare |

Export to PDF