SmartRVM - Simulator de Reverse Vending Machine cu Platforma Web

Introducere

SmartRVM este un simulator miniaturizat de reverse vending machine (masina de colectat sticle PET), construit in jurul unui microcontroller **ATmega328P Xplained Mini** conectat la o platforma web care ruleaza pe PC.

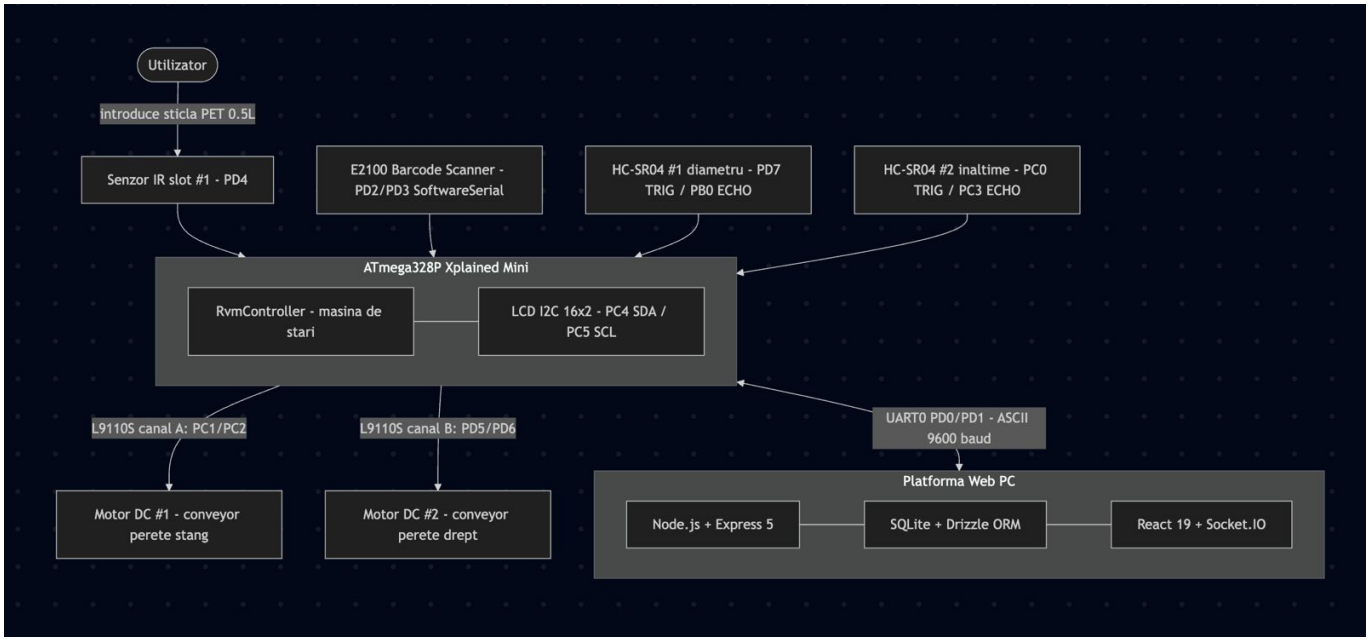
- **Ce face:** utilizatorul introduce o sticla PET, masina o scaneaza, masoara dimensiunile si decide acceptarea sau respingerea ei pe baza unei baze de date configurabile.
- **Scopul:** demonstrarea integrarii mai multor periferice hardware (UART, SoftwareSerial, senzori ultrasonici HC-SR04, motoare DC cu L9110S H-bridge, display I2C) si aplicarea principiilor DDD (Domain-Driven Design) atat pe firmware cat si pe platforma web.
- **Ideea de pornire:** aparatele de colectare a ambalajelor din supermarketuri.
- **Elementul distinctiv:** frameworkul de simulare complet — firmware-ul poate rula cu toti senzorii inlocuiti de stubs software controlabile din interfata web, permitand testarea logicii fara niciun circuit conectat.

Descriere generala

Flux de functionare:

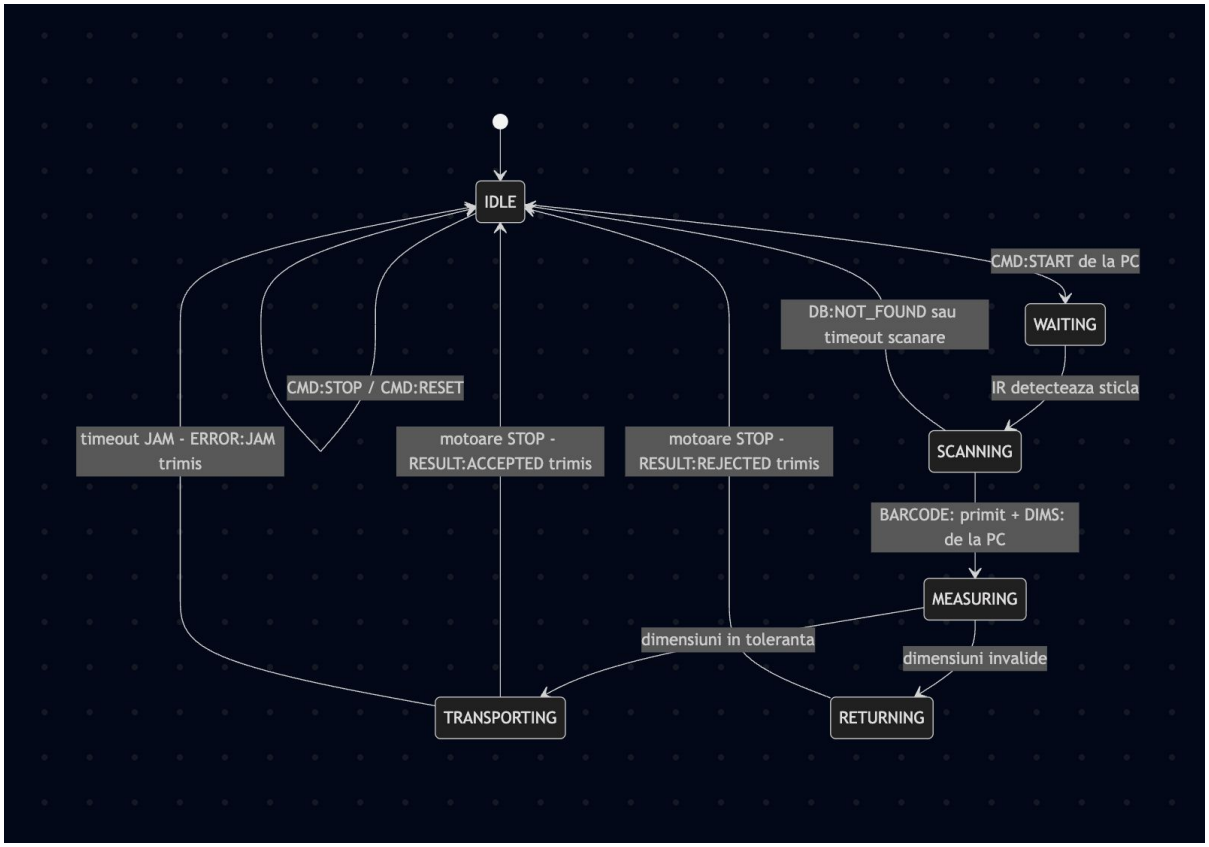
1. Operatorul porneste tranzactia din platforma web (CMD:START)
2. ATmega detecteaza sticla prin senzorul fotoelectric (IR slot) pe PD4
3. Scannerul E2100 citeste barcode-ul prin SoftwareSerial (PD2/PD3), ATmega il trimite catre PC
4. PC-ul cauta dimensiunile in baza de date si le trimite inapoi (`DIMS:`) sau `DB:NOT_FOUND`
5. HC-SR04 masoara inaltimea (PB1/PB2) si diametrul (PD7/PB0), firmware-ul valideaza dimensiunile
6. Daca acceptata: cele doua motoare DC (L9110S canal A PC1/PC2 + canal B PD5/PD6) pornesc inainte si antreneaza banda conveyor; LCD afiseaza “ACCEPTED”
7. Daca respinsa: motoarele pornesc in sens invers si returneaza sticla; LCD afiseaza “REJECTED”
8. PC-ul primeste `RESULT:ACCEPTED` / `RESULT:REJECTED` si actualizeaza dashboard-ul

Schema bloc — Arhitectura sistemului




```
flowchart TB
    User([Utilizator]) -->|introduce sticla PET 0.5L| irSensor

    irSensor["Senzor IR slot #1 - PD4"]
    e2100["E2100 Barcode Scanner - PD2/PD3 SoftwareSerial"]
    hcsr1["HC-SR04 #1 diametru - PD7 TRIG / PB0 ECHO"]
    hcsr2["HC-SR04 #2 inaltime - PC0 TRIG / PC3 ECHO"]

    irSensor --> ATmega
    e2100 --> ATmega
    hcsr1 --> ATmega
    hcsr2 --> ATmega

    subgraph ATmega["ATmega328P Xplained Mini"]
        rvm["RvmController - masina de stari"]
        lcd["LCD I2C 16x2 - PC4 SDA / PC5 SCL"]
        rvm --- lcd
    end

    ATmega -->|"L9110S canal A: PC1/PC2"| motorA["Motor DC #1 - conveyor
perete stang"]
    ATmega -->|"L9110S canal B: PD5/PD6"| motorB["Motor DC #2 - conveyor
perete drept"]
    ATmega <-->|"UART0 PD0/PD1 - ASCII 9600 baud"| web

    subgraph web["Platforma Web PC"]
        nodejs["Node.js + Express 5"]
        sqlite["SQLite + Drizzle ORM"]
        react["React 19 + Socket.IO"]
        nodejs --- sqlite
        sqlite --- react
    end
```

Masina de stari (RvmController)




```
stateDiagram-v2
    [*] --> IDLE
    IDLE --> WAITING : CMD:START de la PC
    WAITING --> SCANNING : IR detecteaza sticla
    SCANNING --> MEASURING : BARCODE: primit + DIMS: de la PC
    SCANNING --> IDLE : DB:NOT_FOUND sau timeout scanare
    MEASURING --> TRANSPORTING : dimensiuni in toleranta
    MEASURING --> RETURNING : dimensiuni invalide
    TRANSPORTING --> IDLE : motoare STOP - RESULT:ACCEPTED trimis
    RETURNING --> IDLE : motoare STOP - RESULT:REJECTED trimis
    TRANSPORTING --> IDLE : timeout JAM - ERROR:JAM trimis
    IDLE --> IDLE : CMD:STOP / CMD:RESET
```

Hardware Design

Lista de componente

Componenta	Rol	Pret
ATmega328P Xplained Mini	Microcontroller principal (AVR 8-bit, 16 MHz, 2KB RAM, 32KB Flash)	existent in laborator
E2100 Barcode Scanner	Citire coduri de bare 1D/2D prin UART TTL (SoftwareSerial 9600 baud)	59.90 RON
Adaptor FPC 12 pini (x2)	Conectare cablu FPC al modulului E2100 la pini standard 2.54mm	7.37 RON
HC-SR04 senzor ultrasonic (x2)	Masurare inaltime si diametru sticla (pulseIn, 2-450 cm, 0.3 cm precizie)	11.30 RON
Senzor fotoelectric slot IR (x2)	Detectie prezenta sticla la intrare (iesire digitala, LOW = sticla)	5.80 RON
Motor DC 3V-6V cu reductor 1:48 (x2)	Doua motoare pe peretii opusi ai cutiei pentru antrenarea conveyor belt-ului	14.52 RON
Modul L9110S H-Bridge dual	Driver pentru ambele motoare DC (canal A motor #1, canal B motor #2)	8.17 RON
Kit Breadboard 830p + 65 jumpers + sursa	Prototipare + modul sursa de alimentare 5V cu intrare barrel jack	42.94 RON
40x fire Dupont mama-tata 20cm	Interconectare placa cu module	16.17 RON
Rezistori 220 ohm (x10)	Current limiting pentru IR si semnale de control	1.20 RON
Rezistori 10k ohm (x10)	Pull-up / pull-down pe intrari digitale	1.20 RON
Condensatori ceramici 100nF (x10)	Decuplare alimentare langa module	2.42 RON
Condensatori electrolitici 100uF 25V (x10)	Filtrare bulk pe linia de alimentare a motoarelor	2.72 RON
Diode 1N4148 (x4)	Protectie flyback pe bobinele motoarelor	2.18 RON
LCD I2C 16×2 (driver PCF8574, backlight albastru)	Display local pentru starea masinii si rezultatul tranzactiei	14.90 RON
Total comanda online		**~158 RON + livrare**

Mapare pini (ATmega328P Xplained Mini)

Placa ATmega328P Xplained Mini are debugger/programator onboard (mEDBG) hardwired pe UART0 (PD0/PD1). Pinii PD5/PD6 (eliberati din configuratia anterioara) sunt folositi pentru motorul de conveyor #2. PC4/PC5 sunt dedicati I2C hardware.

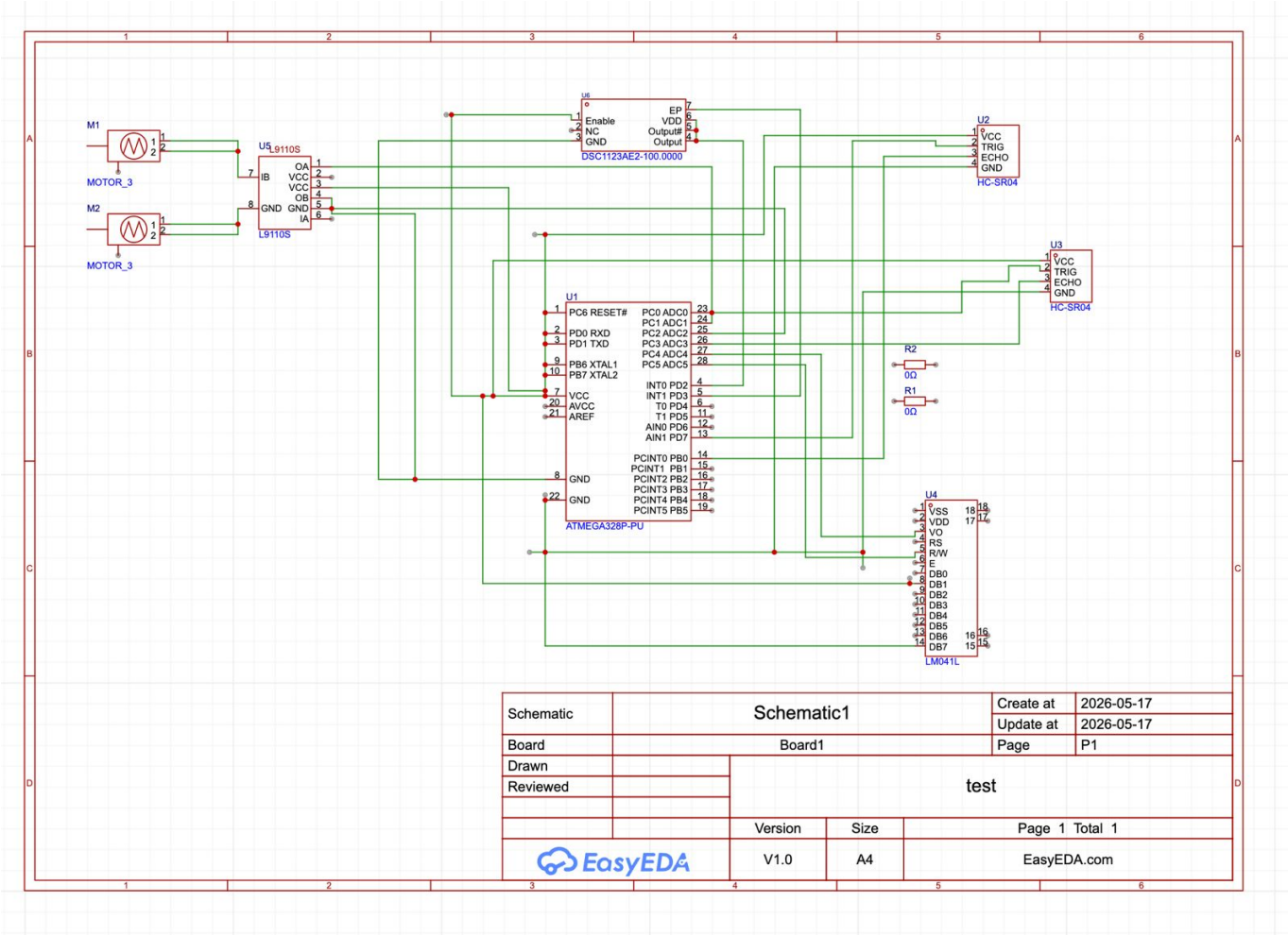
```
Pin    Functie              Modul conectat
-----------------------------------------
PD0    UART0 RX             mEDBG USB (serial la PC)
PD1    UART0 TX             mEDBG USB (serial la PC)
PD2    SoftwareSerial RX    E2100 TX
PD3    SoftwareSerial TX    E2100 RX
PD4    GPIO input           Senzor fotoelectric slot #1 (LOW = sticla
detectata)
PD5    GPIO output          L9110S B-IA (motor conveyor #2 directie A)
PD6    GPIO output          L9110S B-IB (motor conveyor #2 directie B)
```

```
PD7    GPIO output          HC-SR04 #1 TRIG (diametru)
PB0    GPIO input           HC-SR04 #1 ECHO (diametru)
PB1    GPIO output          HC-SR04 #2 TRIG (inaltime)
PC1    GPIO output          L9110S A-IA (motor conveyor #1 directie A)
PC2    GPIO output          L9110S A-IB (motor conveyor #1 directie B)
PB2    GPIO input           HC-SR04 #2 ECHO (inaltime)
PC4    I2C SDA              LCD I2C 16x2 (display stare masina)
PC5    I2C SCL              LCD I2C 16x2 (display stare masina)
```

Justificarea alegerii pinilor

Pin	Tip	De ce acest pin
PD0 / **PD1**	UART0 RX/TX	Singurul UART hardware al ATmega328P este hardwired la mEDBG. Nu poate fi reasignat.
PD2 / **PD3**	SoftwareSerial	INT0/INT1 — singurele pini cu intreruperi externe bine suportate, necesare pentru receptie SoftwareSerial fiabila la 9600 baud.
PD4	GPIO input	Semnal digital simplu de la senzorul IR (nu necesita PWM, ADC sau intreruperi).
PD5 / **PD6**	GPIO output (motor #2)	Eliberati din configuratia anterioara. Raman in Port D, consistent cu restul actuatorilor.
PD7	GPIO output	TRIG necesita doar puls simplu 10µs — orice GPIO functioneaza.
PB0	GPIO input	ICP1 (Input Capture Pin al Timer1) — permite migrare la masurare hardware precisa cu timer capture fara a modifica pinout-ul.
PB1 / **PB2**	GPIO (HC-SR04 #2 TRIG/ECHO)	Pini digitali Port B liberi. Relocati de la PC0/PC3 dupa ce acestia au dovedit probleme hardware pe board. Adiacenti cu PB0 (ECHO senzor #1), grupand logic toti pinii HC-SR04 in Port B.
PC1 / **PC2**	GPIO output (motor #1)	Consecutivi cu PC0/PC3 — toata sectiunea sensor+motor grupata in Port C.
PC4 / **PC5**	I2C SDA/SCL	**Singura optiune posibila**: TWI (I2C hardware) al ATmega328P este implementat fix pe acesti doi pini. Nu exista alternativa hardware.

Schema electrica



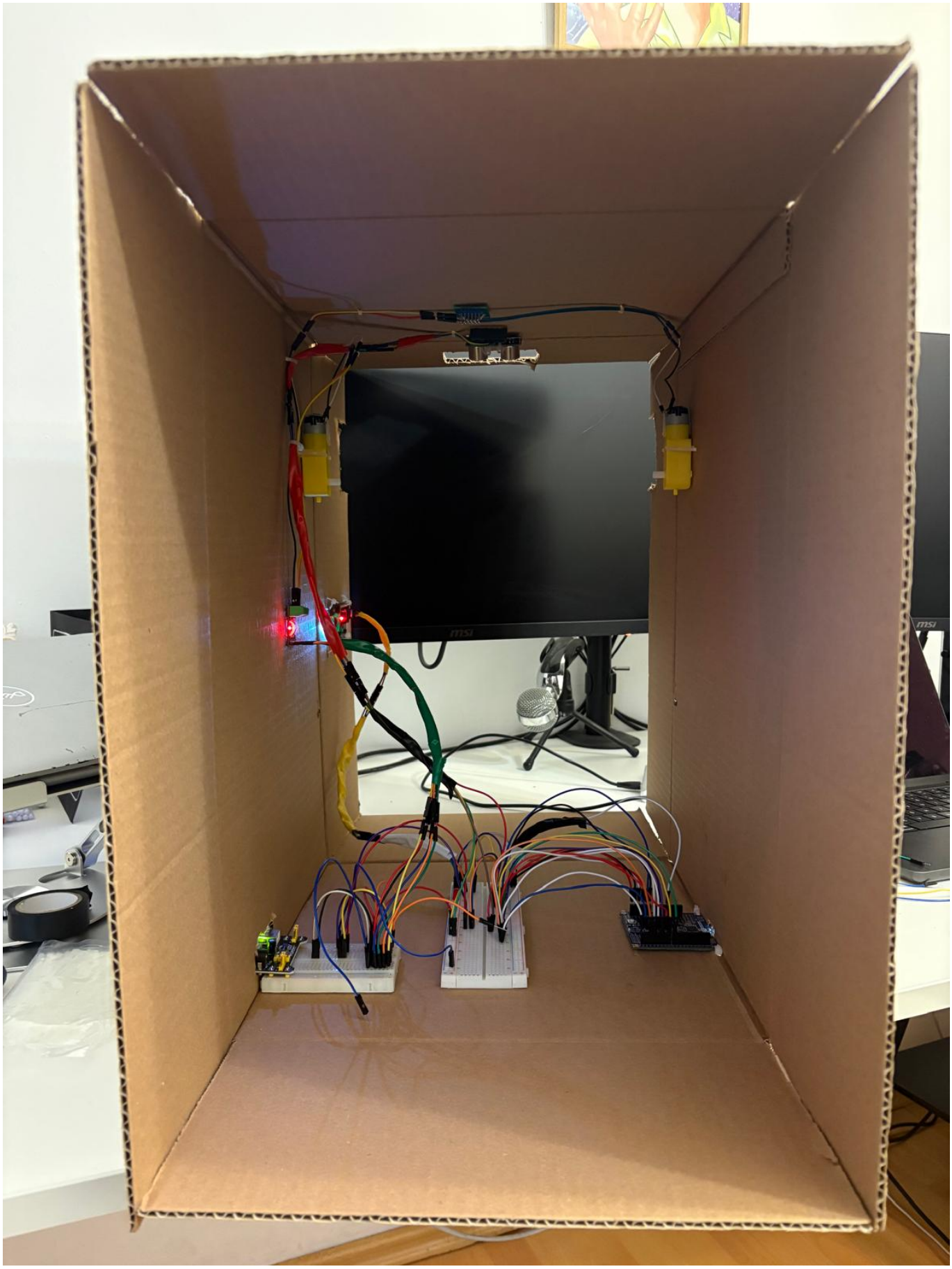
Tabel conexiuni:

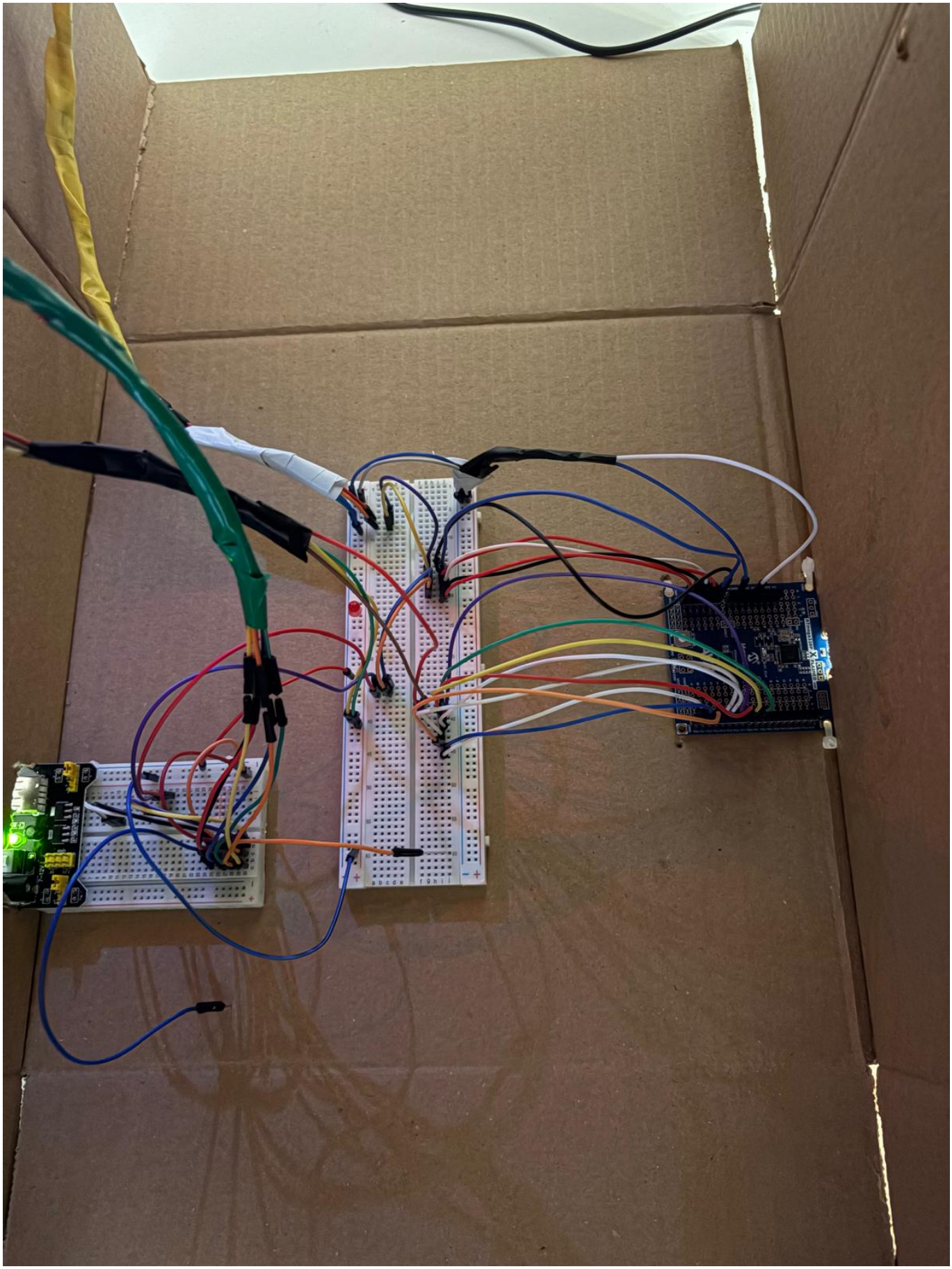
Componenta	Pin componenta	Pin ATmega	Note
E2100 Scanner	TX	PD2	SoftwareSerial 9600 baud
E2100 Scanner	RX	PD3	
IR Slot Sensor #1	OUT	PD4	LOW = sticla prezenta
HC-SR04 #1 (diametru)	TRIG	PD7	Puls 10µs HIGH
HC-SR04 #1 (diametru)	ECHO	PB0	pulseIn(HIGH), 58µs/cm
HC-SR04 #2 (inaltime)	TRIG	PC0	Puls 10µs HIGH
HC-SR04 #2 (inaltime)	ECHO	PC3	pulseIn(HIGH)
L9110S H-Bridge	A-IA	PC1	Motor #1 directie A
L9110S H-Bridge	A-IB	PC2	Motor #1 directie B
L9110S H-Bridge	B-IA	PD5	Motor #2 directie A
L9110S H-Bridge	B-IB	PD6	Motor #2 directie B
LCD I2C 16×2	SDA	PC4	TWI hardware, adresa 0x27
LCD I2C 16×2	SCL	PC5	

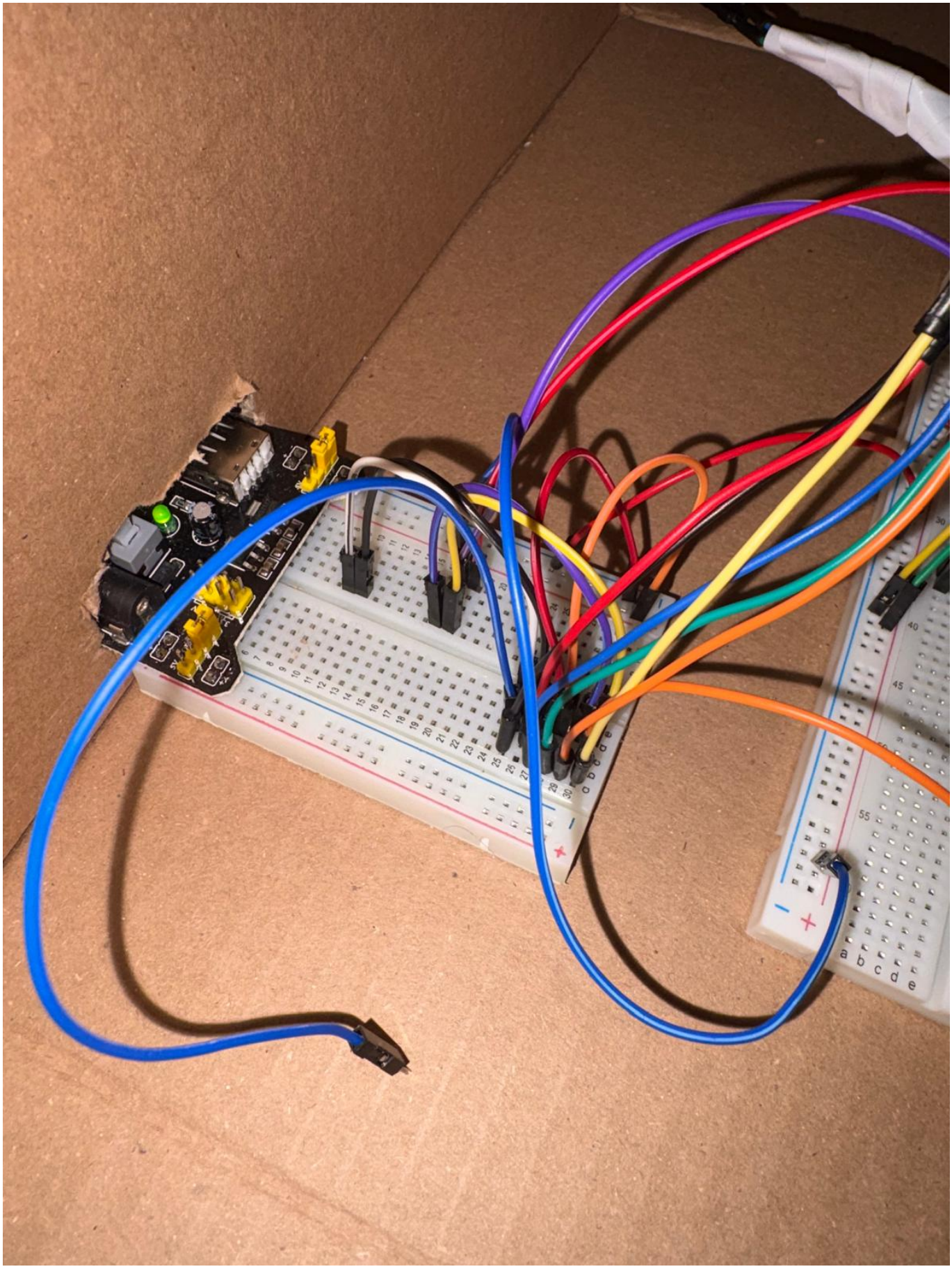
> **Nota alimentare:** Motoarele si LCD-ul sunt alimentate de pe sina 5V a breadboard-ului (incarcator 5V 2A), nu de pe ATmega (limitat la 500mA USB). GND comun intre surse.

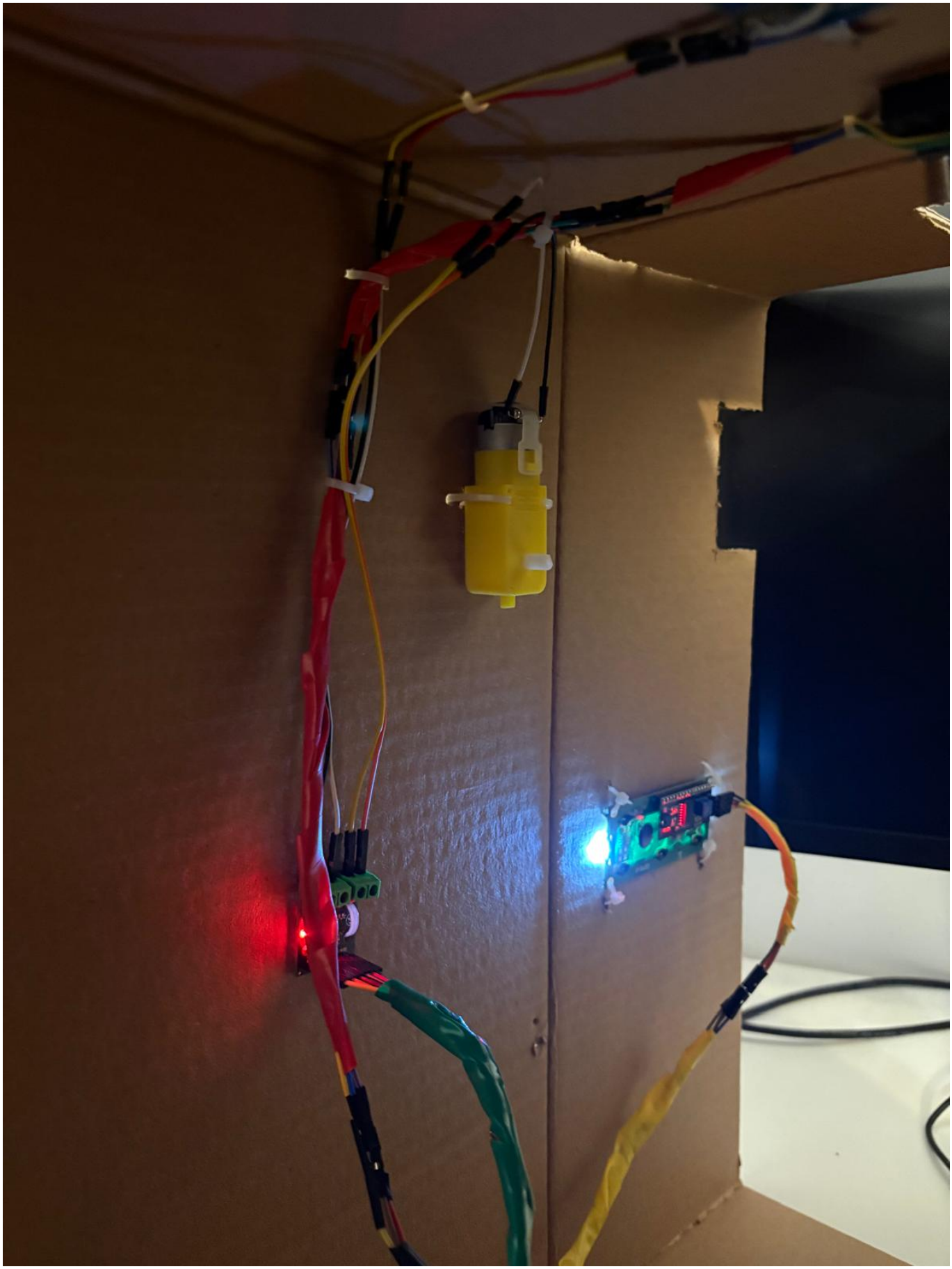
Dovezi de Functionare

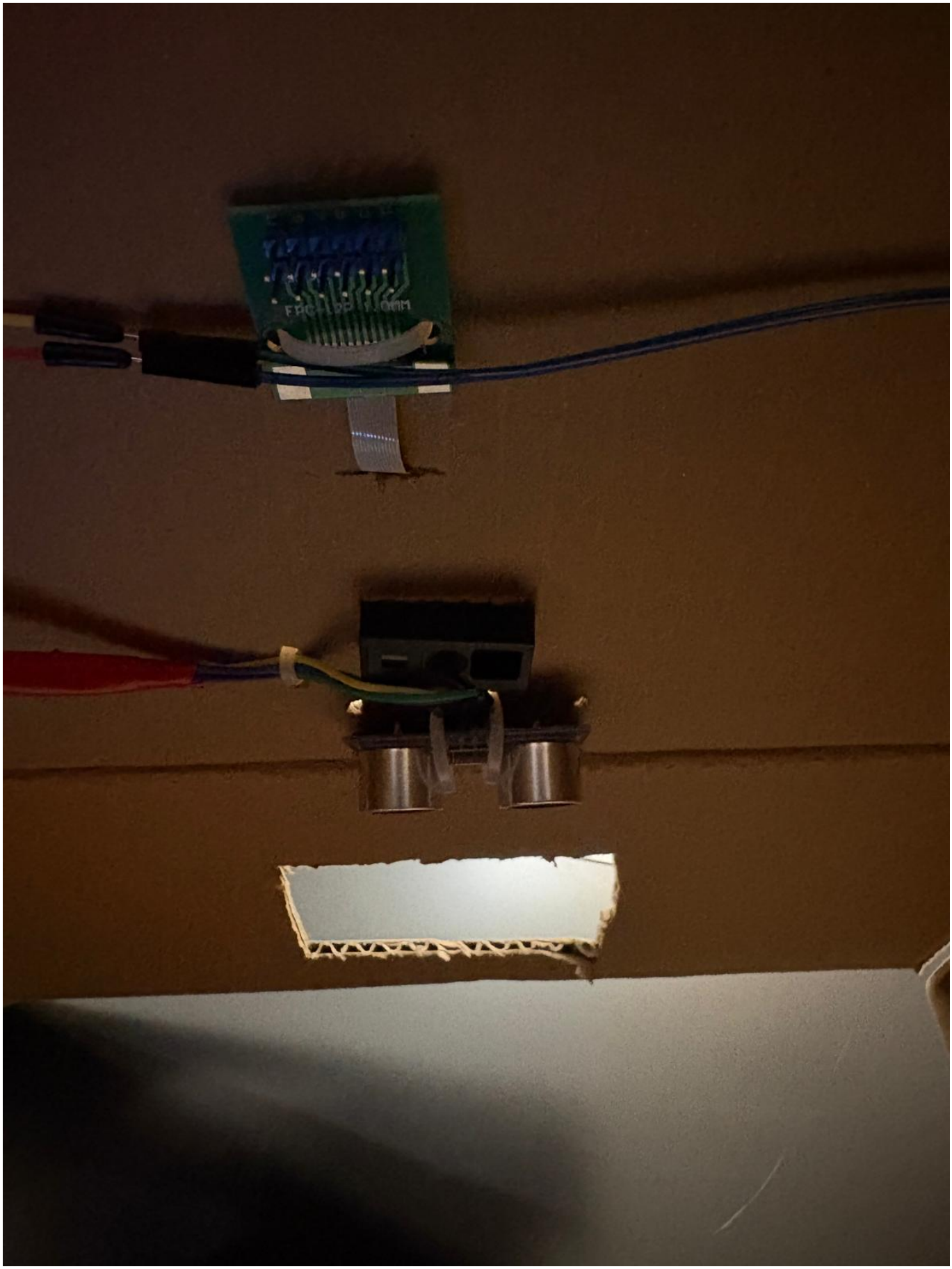
Componente conectate

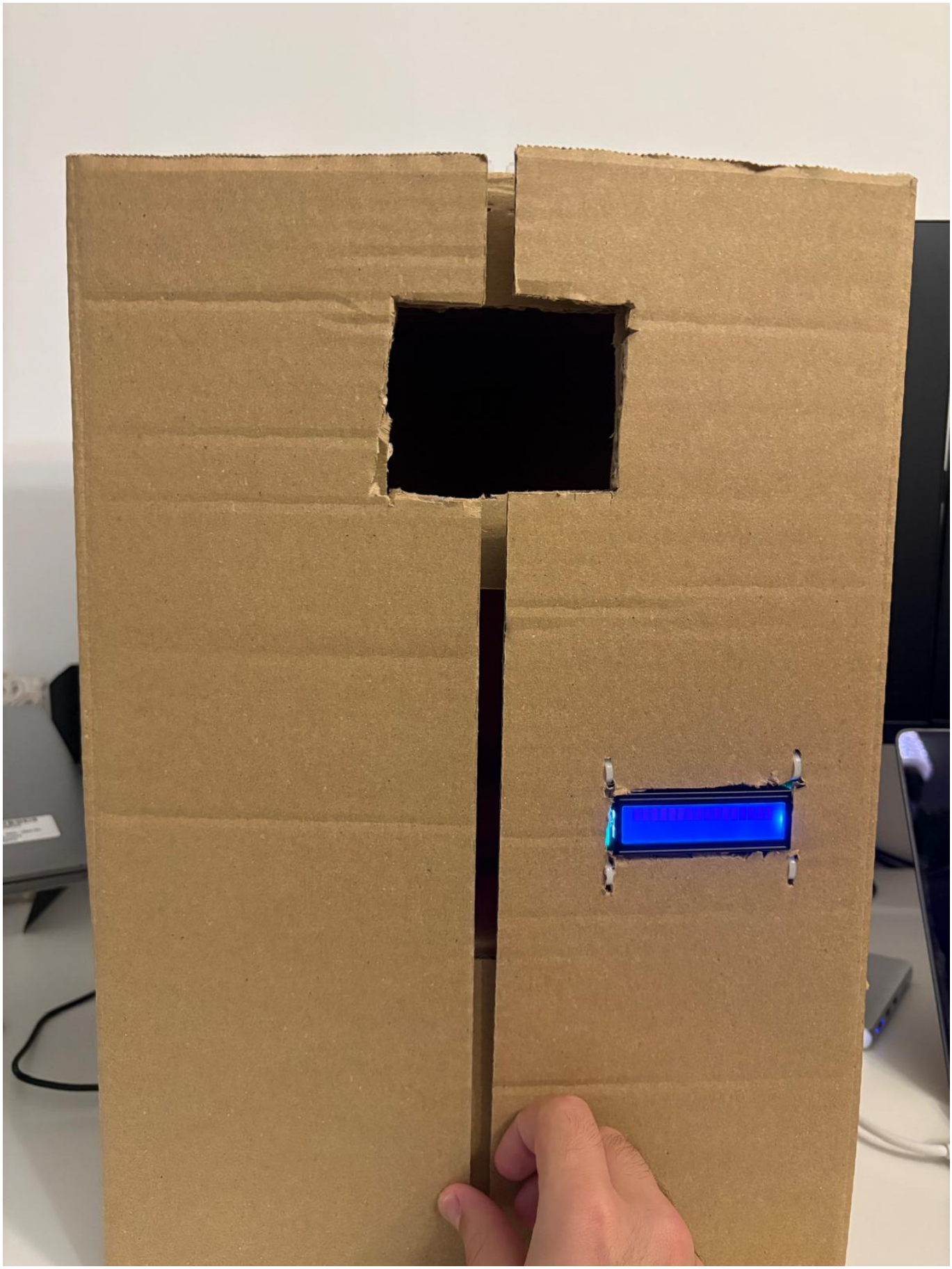

msi
msi

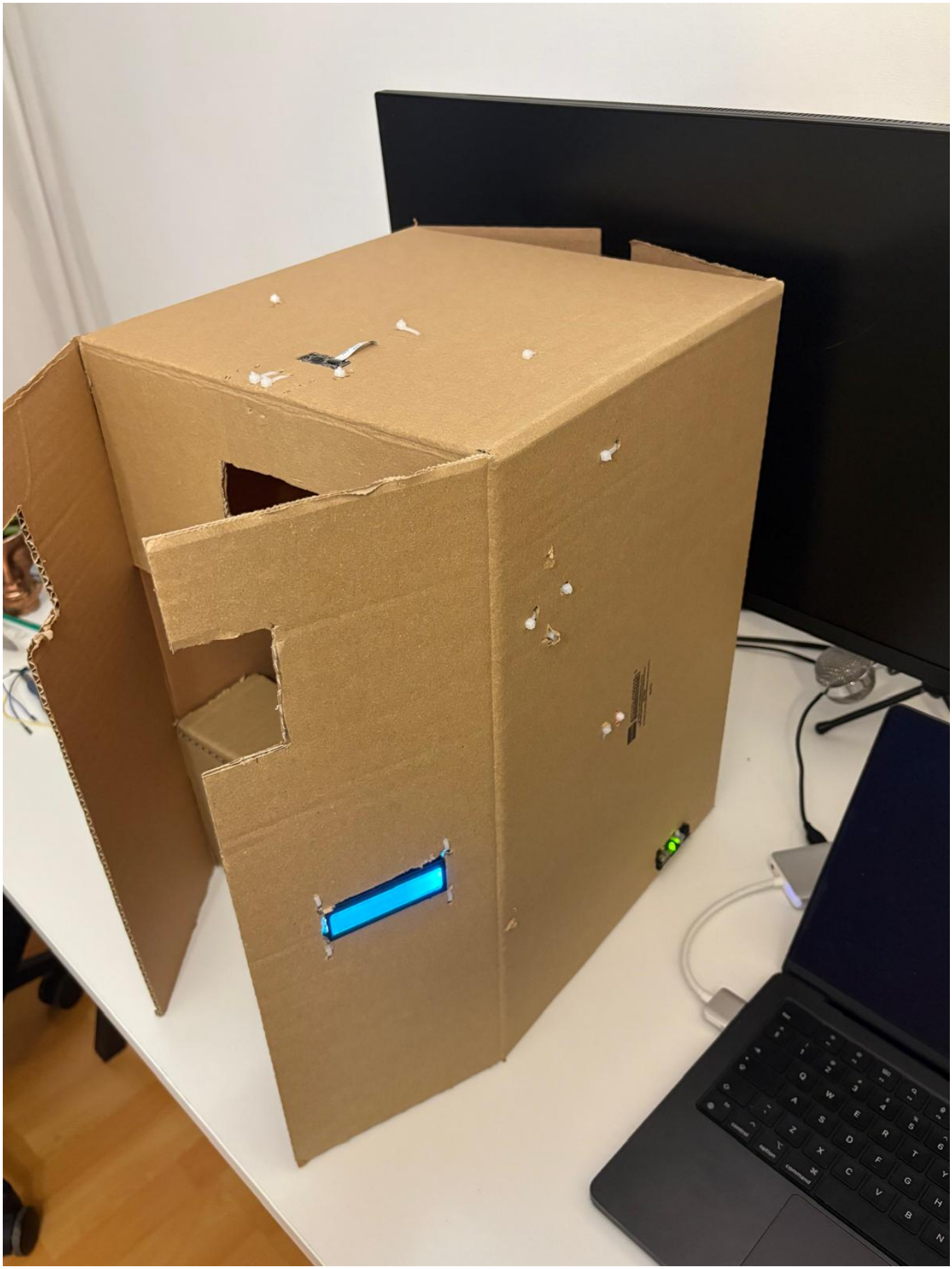












Software Design

Mediu de dezvoltare:

- **PlatformIO** cu framework Arduino, compilator avr-gcc 7.x, C++14
- Doua environment-uri: `xplained_mini` (hardware real) si `xplained_mini_sim` (simulare cu stubs)
- Script custom Python (`set_upload_port.py`) pentru upload prin mEDBG (interfata USB HID, nu serial clasic)

Arhitectura firmware (DDD pe ATmega):

- `common/` — `FixedString` (string pe stack, fara alocare dinamica), `RingBuffer`, `Timer`
- `domain/` — interfete pure C++: `ICommChannel`, `IScanner`, `IBottleDetector`, `IHeightSensor`, `IDiameterSensor`, `ITransport`, `IDisplay`
- `application/` — `RvmController` (masina de stari IDLE→WAITING→SCANNING→MEASURING→TRANSPORTING/RETURNING→IDLE), `BottleValidator`
- `infrastructure/` — `UartProtocol`, `Gm65Scanner` (E2100), `IrBreakBeam`, `DcConveyor` (L9110S dual motor), `LcdI2cDisplay` (PCF8574)
- `stubs/` — `StubScanner`, `StubHeightSensor`, `StubDiameterSensor`, `StubBottleDetector`, `StubTransport`, `StubDisplay`, `SimBridge`

Decizii cheie firmware:

- Niciun `delay()` in logica aplicatiei — toate operatiile sunt non-blocking cu `tick()`
- Fara alocare dinamica (nu exista `new`, `String`, `std::vector`) — pe 2KB RAM, fragmentarea heap-ului e fatala
- Erori returnate prin enum-uri, nu exceptii (compilat cu `-fno-exceptions`)
- `DcConveyor` controleaza simultan ambele motoare: inainte (ACCEPT), inapoi (RETURN), stop

Platforma web (Node.js + React):

- `shared` — tipuri TypeScript comune (protocol serial, DTO-uri REST)
- `server` — Node.js 22, Express 5, SQLite + Drizzle ORM, serialport, Socket.IO 4, Zod
- `client` — React 19, Vite, Tailwind CSS v4, shadcn/ui, Recharts

Pagini: **Dashboard** (status live, START/STOP), **Barcodes** (CRUD), **Transactions** (istoric), **Statistics** (grafice), **Simulation** (control stubs)

Protocol serial ASCII (ATmega ↔ PC):

```
PC → ATmega:  CMD:START / CMD:STOP / CMD:RESET
              DIMS:220:65:10     (inaltime:diametru:toleranta in mm)
              DB:NOT_FOUND

ATmega → PC:  STATUS:IDLE / WAITING / SCANNING / MEASURING / TRANSPORTING /
RETURNING
              BARCODE:5941234567890
              MEASURED:H=218:D=63
              RESULT:ACCEPTED / RESULT:REJECTED:oversized
```

```
                ERROR:JAM / ERROR:SCANNER_FAIL

Simulare:       SIM:BEAM:ON/OFF   SIM:BARCODE:<val>   SIM:HEIGHT:<mm>
                SIM:DIAM:<mm>     SIM:JAM             SIM:RESET_STUBS
                ATmega: SIM:ACK:<msg> / SIM:ERR:<motiv>
```

Rezultate Obtinute

- Firmware-ul compileaza fara warninguri in ambele configuratii (productie si simulare)
- Build-ul de simulare testat pe ATmega328P Xplained Mini: comunicatia seriala, masina de stari si SimBridge functioneaza corect
- Platforma web functionala cu toate cele cinci pagini
- Script custom Python (`set_upload_port.py`) rezolva limitarea PlatformIO cu protocolul mEDBG
- Componentele hardware au sosit; urmeaza conectarea fizica si calibrarea

Milestone 3 — Implementare Software

Stadiul actual

Tot codul este finalizat si functional. Firmware-ul compileaza in toate trei configuratiile:

Environment	Scop
`xplained_mini`	hardware real
`xplained_mini_sim`	simulare cu stubs software
`xplained_mini_diag`	diagnostic senzori fara actuatori

Masina de stari trece corect prin toate starile, motoarele sunt controlate prin `DcConveyor`, LCD-ul afiseaza starea si rezultatul.

Pe platforma web: server Express + SQLite functional cu toate rutele REST, WebSocket cu update-uri in timp real (< 50ms), toate paginile operationale — Dashboard, Barcodes, Transactions, Statistics, Simulation si `/diag`.

Biblioteci folosite

Firmware:

- **SoftwareSerial** — UART0 e ocupat de mEDBG (USB la PC), deci scannerul E2100 a trebuit conectat pe PD2/PD3 (INT0/INT1) prin SoftwareSerial. INT0/INT1 sunt necesare pentru receptie fiabila prin intreruperi.
- **LiquidCrystal_I2C** — LCD-ul are driver PCF8574 si comunica prin I2C. Varianta 4-bit paralel ar fi consumat 6 pini in plus, inacceptabil pe ATmega328P.
- **FixedString<N>, RingBuffer<T,N>, Timer** (proprii) — clasa `String` din Arduino aloca dinamic si fragmenteaza heap-ul. Pe 2KB RAM asta se termina cu crash dupa ore de rulare. Toate stringurile sunt pe stack.

Platforma web:

- **serialport** — singura biblioteca Node.js serioasa pentru USB serial cross-platform.
- **Socket.IO** — serverul impinge evenimentele la browser imediat ce le primeste de la ATmega (3-8ms latenta pe localhost), fara polling.
- **Drizzle ORM** — schema TypeScript-first, SQLite embedded, zero configurare server.
- **Zod** — validare body request; schema e si tip TypeScript direct, fara duplicare.
- **Vite, Tailwind v4, shadcn/ui, Recharts** — development rapid, componente accesibile, grafice SVG native React.

Ce e diferit la proiect

Framework de simulare integrat in firmware. Inainte sa soseasca componentele, `xplained_mini_sim` rula pe placa cu toti senzorii inlocuiti de stubs controlabile din interfata web:

```
SIM:BEAM:ON → ATmega simuleaza detectia sticlei
SIM:BARCODE:5941 → StubScanner returneaza barcode-ul
SIM:HEIGHT:220 / SIM:DIAM:65 → dimensiuni simulate
→ RESULT:ACCEPTED (fara nicio piesa conectata)
```

Asta a permis testarea integrala a masinii de stari — timeout, jam, barcode inexistent — cu saptamani inainte de hardware. Aceeasi logica din `RvmController` ruleaza identic in ambele build-uri.

Arhitectura DDD pe ATmega cu 2KB RAM. Domain, application, infrastructure si stubs sunt straturi separate; niciun header Arduino nu ajunge in logica de business. Mai greu de pus la punct, dar debugging-ul a fost mult mai simplu.

Firmware de diagnostic separat. `xplained_mini_diag` trimite snapshot-uri cu toti senzorii la 500ms fara sa activeze actuatori. Prin pagina `/diag` am verificat fiecare senzor individual si am descoperit rapid ca pinul PC0 era defect pe board — HC-SR04 #2 a fost mutat pe PB1/PB2.

Functionalitatile din laborator

Laborator	Cum e folosita
Lab 1 — UART	UART0 (PD0/PD1) pentru protocolul ATmega↔PC (`CMD:`, `DIMS:`, `BARCODE:`, `RESULT:`). SoftwareSerial pe PD2/PD3 pentru scannerul E2100.
Lab 2 — Debugging	Toate tranzitiile vizibile in Serial Monitor. `MockConnection` pe server injecteaza secvente fara ATmega conectat.
Lab 4 — Timere	Timeout-urile din `RvmController` (scanare 5s, masurare 2s, transport 10s) folosesc `Timer::elapsed()` cu `millis()`. Niciun `delay()` — loop la ~0.3ms/iteratie.
Lab 5 — GPIO	TRIG HC-SR04 (PD7, PB1): puls 10µs. ECHO (PB0, PB2): `pulseIn()`. IR slot (PD4): `digitalRead()`. L9110S (PC1, PC2, PD5, PD6): directia motoarelor.
Lab 7 — I2C/TWI	LCD I2C cu driver PCF8574 pe TWI hardware (PC4/PC5), adresa 0x27.

Structura si interactiunea dintre componente

```
device/lib/
    common/         FixedString, RingBuffer, Timer
    domain/         interfete pure: IScanner, ITransport, IDisplay,
ICommChannel...
    application/    RvmController (masina de stari), BottleValidator
    infrastructure/ UartProtocol, Gm65Scanner, HcSr04Sensor, DcConveyor,
LcdI2cDisplay
    stubs/          StubScanner, StubHeightSensor, StubTransport, SimBridge
device/src/         main.cpp / main_sim.cpp / main_diag.cpp

platform/
    shared/         serial-protocol.ts (tipuri comune server+client)
    server/         domain, application, infrastructure (serial, db,
websocket), routes
    client/         Dashboard, Barcodes, Transactions, Statistics,
Simulation, SensorDiag
```

Flux normal: sticla → IR(PD4) → `RvmController.tick()` → E2100 → `BARCODE:` pe UART0 → server cauta in SQLite → `DIMS:` inapoi → ATmega masoara HC-SR04 → `MEASURED:` → validare → motoare → `RESULT:` → server salveaza → Socket.IO emite catre browser.

Validare in trei pasi:

1. *Simulare firmware* (`xplained_mini_sim`): CMD:START → SIM:BEAM:ON → SIM:BARCODE: → SIM:HEIGHT: → SIM:DIAM: → RESULT:ACCEPTED testata inainte de hardware.
2. *Mock connection server*: `MockConnection` injecteaza fluxuri predefinite, a permis testarea WebSocket si a persistentei SQLite independent de placa.
3. *Firmware diagnostic* (`xplained_mini_diag`): fiecare senzor verificat individual prin `/diag`, fara masina de stari.

Demo

Video demonstratie SmartRVM — pornire sistem, conectare seriala, pagina Simulation cu stubs, pagina Sensor Diag cu citire live senzori si control motoare.

Calibrare senzori

HC-SR04: formula `distanta_mm = durata_us / 58 * 10`. Testat la distante cunoscute (50, 100, 200mm) — eroare ±3mm la 100mm, conform specificatiei. Offsetul de montaj a fost masurat empiric si adaugat ca `constexpr` in `main.cpp`. `pulseIn` are timeout de 30ms — fara el, o citire invalida ar bloca loop-ul pana la 1 secunda.

IR slot: activ LOW (LOW = sticla prezenta). Debounce 20ms in `RvmController` pentru a ignora tranzitiile false la introducerea sticlei.

E2100: 9600 baud, EAN-13 terminat CR+LF. `Gm65Scanner` elimina CR+LF si returneaza sirul pur. Timp de raspuns ~100ms de la detectia sticlei.

Optimizari

Fara alocare dinamica. `FixedString<N>` in loc de `String`, buffere si cozi statice. La build, avr-gcc raporteaza ~980 bytes SRAM din 2048 — cu `String` am fi ajuns la overflow dupa ore de rulare.

Non-blocking loop. Niciun `delay()` in logica aplicatiei. Toate timeout-urile folosesc `Timer::elapsed()` cu `millis()`. Loop la ~0.3ms/iteratie indiferent de starea masinii.

Alimentare separata pentru motoare. La pornire, un motor poate trage > 1A. Prin USB, ATmega are 500mA — varful de curent ii scade tensiunea si il reseteaza (brownout). Dupa separarea pe sina dedicata 5V 2A, problema a disparut.

Tip-safe protocol intre server si client. `shared/serial-protocol.ts` defineste toate tipurile de mesaje. Daca serverul emite un camp cu alt nume decat ce asteapta clientul, TypeScript raporteaza eroare la compilare, nu la runtime.

WebSocket in loc de polling. Serverul emite evenimentele imediat la primirea de la ATmega. Latenta medie: 3-8ms pe localhost, fata de 0-1s cu polling REST.

Concluzii

Separarea clara intre domain si infrastructure a permis testarea logicii masinii de stari inainte de sosirea hardware-ului. `DcConveyor` controleaza ambele motoare printr-o singura interfata `ITransport`, mentinand logica aplicatiei independenta de hardware. Frameworkul de simulare (stubs + SimBridge + pagina web Simulation) a validat integral fluxul de tranzactie.

Download

```
SmartRVM/
    device/          -- firmware ATmega (PlatformIO)
    platform/        -- platforma web (pnpm monorepo)
    wokwi/           -- schema electrica Wokwi (diagram.json)
    DOCUMENTATION.md -- documentatie Markdown (diagrame Mermaid)
    DOKUWIKI.txt     -- versiune DokuWiki pentru OCW
```

```
cd device && pio run -e xplained_mini_sim -t upload
cd platform && pnpm install
pnpm --filter server dev && pnpm --filter client dev
```

Bibliografie / Resurse

Resurse Hardware:

- ATmega328P Datasheet
- ATmega328P Xplained Mini User Guide
- HC-SR04 Datasheet
- Motor DC 3V-6V cu reductor 1:48 (ArduShop)
- PCF8574 I2C Expander Datasheet (LCD driver)
- L9110S H-Bridge Datasheet

Resurse Software:

- PlatformIO Documentation
- avrdude xplainedmini protocol
- Arduino SoftwareSerial Library
- LiquidCrystal_I2C Library
- serialport npm package
- Drizzle ORM Documentation

- Socket.IO Documentation
- shadcn/ui Components
- Wokwi — simulator Arduino online (schema electrica)

Cursuri PM relevante:

- Cursul 1 (UART): comunicatie seriala ATmega-PC prin UART0 si SoftwareSerial pentru scanner (PD2/PD3)
- Cursul 4 (Timere, PWM): Timer0 pentru timeout-urile non-blocking din RvmController
- Cursul 5 (GPIO): pinii PB1/PB2 GPIO digital pentru HC-SR04 #2; PC1/PC2 GPIO pentru motoare
- Cursul 7 (I2C): bus I2C hardware TWI pe PC4/PC5 pentru LCD-ul PCF8574
- Cursul 2 (Debugging, UART): debug prin Serial.println in faza de simulare

Export to PDF