

DOCUMENT DE PROIECT

Digital Challenge

NUMĂRUL PROIECTULUI

2024-2-RS01-KA210-SCH-000271542

Învățați prin exemple

Exerciții de bază pentru Arduino + Processing

Workshop Digital Challenge

#	Proiect	Teme abordate
1	Temperatură și Umiditate — DHT11	Senzori, comunicare serială, vizualizare în Processing
2	Senzor de lumină — LDR	Intrare analogică, divizor de tensiune, grafic cu bare
3	Sonar — Senzor cu ultrasunete HC-SR04	Distanță, control servomotor, afișaj radar
4	Ohmmetru	Divizor de tensiune, Legea lui Ohm, afișarea rezistenței
5	Sistem automat de irigații	Umiditatea solului, releu, controlul pompei

Introducere: Primii pași în programarea microcontrolerelor cu Arduino

Descriere

În această lecție, elevilor le este prezentată placa Arduino Uno, elementele de bază ale unui sistem cu microcontroler și primii pași în lucrul cu circuite electronice. Accentul este pus pe conectarea software-ului cu hardware-ul, deoarece aceasta reprezintă temelia fiecărui proiect STEM — de la idee, prin simulare, până la asamblarea în lumea reală.

Lecția este pregătită pentru documentația STEM Erasmus+ într-un mod care încurajează învățarea exploratorie, aplicarea practică a cunoștințelor și înțelegerea modului în care un program simplu poate controla un dispozitiv real.

Acest modul servește ca bază introductivă pentru toate proiectele ulterioare cu senzori și Processing din seria de ateliere *Digital Challenge*.

Obiective

Elevii vor:

- înțelege ce este Arduino și la ce folosește o placă cu microcontroler,
- deosebi componentele de bază ale plăcii Arduino Uno,
- învăța cum să folosească un breadboard și un rezistor într-un circuit simplu,
- conecta codul de programare cu cablarea componentelor fizice reale,
- încărca primul lor program folosind mediul Arduino IDE,
- dezvolta obiceiul de a verifica mai întâi schema circuitului înainte de a realiza conexiunile fizice.

Rezultate educaționale

După această lecție, elevii vor fi capabili să:

- explice scopul de bază al platformei Arduino,
- identifice cele mai importante părți ale plăcii Arduino Uno,
- assembleze un circuit simplu pe un breadboard,
- ruleze și să modifice un program Arduino de bază,
- utilizeze concepte precum pinMode, digitalWrite și delay,
- facă legătura între simularea și construcția reală a aceluiași circuit.

Echipament necesar

Componentă	Cantitate	Purpose
Arduino Uno	1	Placa principală de dezvoltare
Cablu USB	1	Conectare la calculator și încărcarea programului
Breadboard	1	Conectare rapidă fără lipire
Diodă LED	1–2	Indicator vizual al execuției programului
Rezistor 220Ω	1–2	Limitarea curentului pentru LED
Fire de conexiune (Jumper)	~4	Conectarea componentelor
Arduino IDE	1	Scrierea și încărcarea programelor
Tinkercad sau Circuit Designer	1	Simularea și verificarea circuitului

Cadrul teoretic

Ce este Arduino?

Arduino este o platformă hardware și software bazată pe un microcontroler. În practică, acest lucru înseamnă că o placă mică poate citi semnale de la senzori, le poate procesa și apoi poate controla LED-uri, motoare, relee, afișaje sau alte dispozitive.

Datorită cablării simple, numărului mare de exemple și suportului extins prin biblioteci, Arduino este extrem de potrivit pentru începători, precum și pentru prototiparea rapidă în activitățile școlare și de proiect.

 Gândiți-vă la Arduino ca la creierul proiectului vostru — citește intrările (senzori, butoane) și controlează ieșirile (LED-uri, motoare, afișaje)

De ce Arduino Uno?

Arduino Uno este cea mai frecventă alegere pentru primele exerciții deoarece este stabilă, bine documentată și ușor de utilizat. Are suficiente pini de intrare/ieșire pentru experimente școlare, fără a copleși elevul cu o complexitate inutilă.

Odată ce noțiunile de bază sunt învățate pe placa Uno, trecerea la alte modele precum Nano, Mega sau ESP32 devine mult mai ușoară.

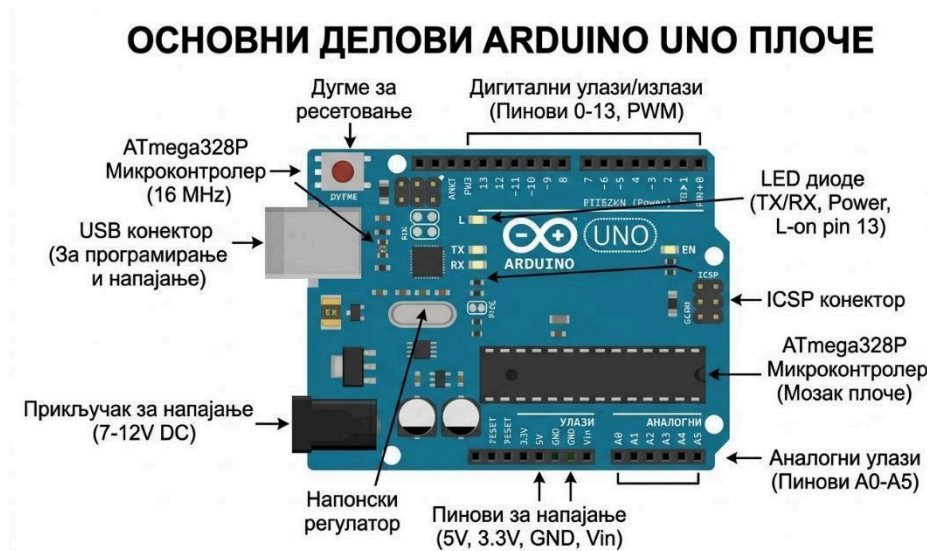
Componentele principale ale plăcii Arduino Uno

Înainte de prima conectare, este important ca elevii să știe unde se găsesc:

- Conectorul USB — pentru conectarea la calculator și încărcarea programelor,
- Pinurile digitale (0–13) — pentru semnale digitale de intrare și ieșire,
- Pinurile analogice (A0–A5) — pentru citirea valorilor analogice de la senzori,
- GND — împământare / masă (referință negativă),

- 5V și 3.3V — pinuri de alimentare,
- LED-ul integrat (Built-in LED) — conectat la pinul 13, util pentru primul test,
- Butonul de Reset — repornește programul,
- Cipul microcontroler (ATmega328P) — procesorul plăcii.

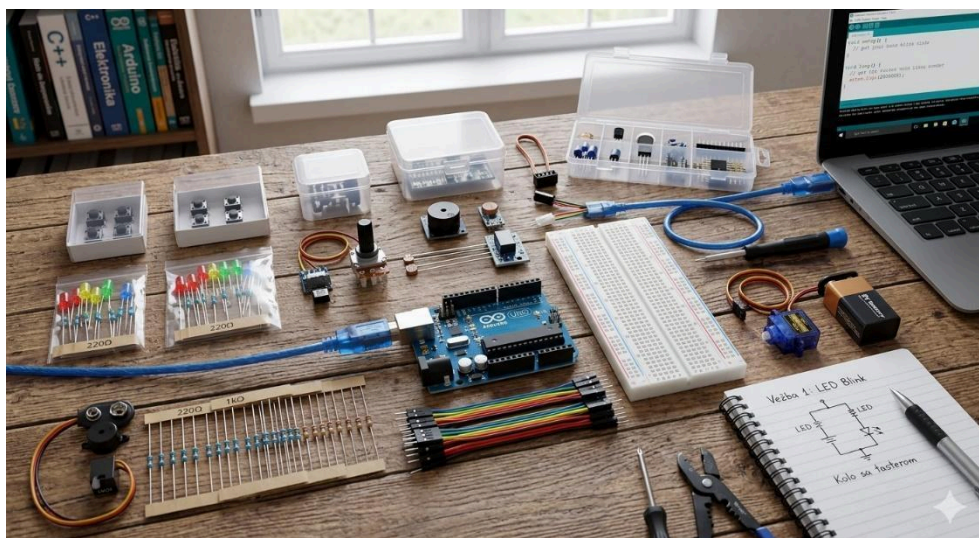
✓ Sfat: Identificați întotdeauna Pinii GND și de alimentare înainte de a conecta orice componentă.



Kitul de bază pentru electronică

Pentru primele exerciții se folosește un kit de pornire (starter kit), care include de obicei o placă Arduino, cablu USB, breadboard, fire de conexiune, rezistoare, diode LED, butoane, senzori și alte componente electronice mici. Un astfel de kit permite realizarea unui număr mare de experimente interesante chiar de la început.

Odată ce elevii învață să recunoască componentele și să le conecteze corect, pot realiza o serie întreagă de exerciții fără teama de a conecta greșit.



Breadboard-ul, Rezistori și LED-uri

Un breadboard este o placă pentru conectarea rapidă a componentelor electronice fără lipire. Conexiunile pot fi modificate ușor, ideile testate și circuitele redefinite. Rezistorul limitează curentul — pentru un LED, acesta este necesar pentru a preveni arderile.

Când se lucrează cu un LED, este important să se știe care picioruș este anodul (+) (mai lung) și care este catodul (–) (mai scurt), și cum se utilizează corect GND.



Anodul LED-ului (piciorul mai lung) se conectează la pinul de semnal prin rezistor. Catodul LED-ului (piciorul mai scurt) se conectează la GND.

Medii de dezvoltare: Arduino IDE, Tinkercad și Circuit Designer

Programele pentru Arduino sunt cel mai frecvent scrise în mediul Arduino IDE. Aici codul este scris, verificat și încărcat pe placă. Tinkercad și Circuit Designer sunt utile pentru planificarea circuitelor și verificarea inițială, deoarece permit simularea înainte de conectarea fizică.

Arduino IDE

Arduino IDE este mediul oficial de programare. În el se scrie codul, se verifică erorile și se trimite programul către microcontroler. Mediul este simplu de utilizat și adecvat pentru începători.

Mediul permite:

- scrierea și editarea programelor (sketch-uri),
- încărcarea programelor pe placa Arduino,
- monitorizarea execuției prin Serial Monitor (Monitorul Serial),
- adăugarea de biblioteci pentru senzori și alte componente.

Acest software este gratuit și poate fi descărcat de pe site-ul oficial Arduino. După instalare, selectați tipul corect de placă și portul COM, apoi încărcați programul pe device.

Tinkercad & Circuit Designer

Tinkercad și Circuit Designer permit conectarea virtuală a componentelor și testarea circuitelor înainte de lucrul practic, lucru util în special în predare.



Cea mai bună practică: Simulați întotdeauna circuitul mai întâi în Tinkercad, apoi construiți-l fizic pe breadboard.

Senzori și actuatori utilizați în proiectele ulterioare

În exercițiile următoare, elevii vor lucra cu o varietate de senzori și actuatori, de exemplu:

Sensor / Actuator	Type	Used In
DHT11	Senzor de temperatură și umiditate	Proiectul 1
LDR (Fotorezistor)	Senzor de lumină	Proiectul 2

Sensor / Actuator	Type	Used In
HC-SR04	Senzor ultrasonic de distanță	Proiectul 3
Modul Releu	Comutator electric (actuator)	Proiectul 5
Senzor de umiditate a solului	Senzor Capacitiv	Proiectul 5
Servomotor SG90	Actuator de poziție	Proiectul 3

Exercițiul 1: Încărcarea primului program — Blink

Acest exercițiu verifică dacă placa Arduino este conectată corect la calculator, dacă portul COM potrivit este selectat și dacă programul se încarcă cu succes. În loc de un circuit extern, se folosește LED-ul integrat pe pinul 13.

Cod Arduino

```
const int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Explicația codului

Comandă	Explicație
const int ledPin = 13;	Definește numărul pinului unde este conectat LED-ul.
setup()	Rulează o singură dată, la pornirea programului.
pinMode(ledPin, OUTPUT);	Setează pinul ca ieșire.
loop()	Se repetă la nesfârșit cât timp placa este alimentată.
digitalWrite(ledPin, HIGH/LOW);	Aprinde (HIGH) sau stinge (LOW) LED-ul.
delay(1000);	Pune o pauză în execuție de 1 secundă (1000 milisecunde).

Cum se încarcă programul

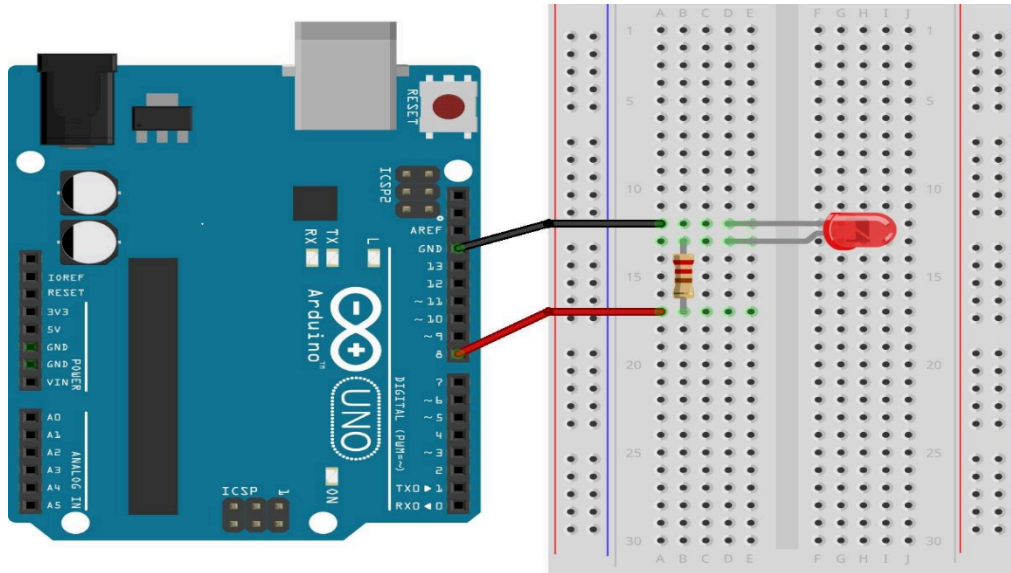
- Deschideți Arduino IDE.
- Mergeți la Tools (Instrumente) → Board (Placă) → Arduino Uno.
- Mergeți la Tools (Instrumente) → Port → selectați portul COM corect.
- Apăsați butonul Upload (Încărcare) (pictograma cu săgeată →).
- LED-ul integrat de pe placă ar trebui să înceapă să clipească o dată pe secundă.

✓ Dacă LED-ul clipește — succes! Placa este conectată și funcționează corect.

Exercițiul 2: Conectarea unui LED extern pe un breadboard

În acest exercițiu, elevul trece de la LED-ul integrat la un LED extern, asamblând pentru prima dată un circuit electronic real. Aici legăturile dintre schemă, conexiunile fizice și program sunt clar demonstrate.

Schema de conectare



Imagine 3: Conectarea reală pe breadboard

Arduino Uno	Componentă	Conexiune
Pin 8	Anod LED (+)	Prin rezistor de 220Ω
GND	Catod LED (-)	Direct la masă (GND)

Codul Arduino

```
const int ledPin = 8;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

Ce face acest program?

- Aprinde LED-ul de pe pinul 8,
- Așteaptă jumătate de secundă (500 ms),
- Stinge LED-ul,
- Repetă întregul proces la nesfârșit.

Sarcini pentru elevi

Nivel	Sarcină	Obiectiv
 De bază	Modificați valoarea pauzei (delay)	Înțelegerea efectului duratei pauzei
 Intermediar	Mutați LED-ul pe alt pin	Exersarea atribuirii pinilor în cod
 Avansat	Adăugați un buton pentru a controla LED-ul LED	Învățarea noțiunilor de INPUT_PULLUP și digitalWrite
 Provocare	Creați un efect de lumină secvențială cu 3 LED-uri effect	leșiri multiple, logică de temporizare

Sfaturi și greșeli frecvente

Acordați atenție următoarelor aspecte:

- Identificați întotdeauna unde sunt Pinii GND și de alimentare înainte de a conecta ceva.
- Nu conectați niciodată un LED direct fără un rezistor de limitare a curentului.
- Înainte de a încărca codul, verificați dacă este selectat portul COM corect.
- Dacă circuitul nu funcționează, verificați mai întâi conexiunile fizice, apoi codul.
- Începeți cu circuite simple înainte de a trece la unele mai complexe.

 O abordare sistematică: Schemă → Simulare (Tinkercad) → Conectare fizică → Încărcare cod → Testare.

Aplicații în lumea reală

Această abordare de învățare este utilizată în:

- ateliere educaționale și activități STEM,
- dezvoltarea prototipurilor pentru case inteligente (smart home),
- primele proiecte IoT și dispozitive conectate,
- sisteme meteorologice și de măsurare,
- soluții robotice și automatizate,
- instalații artistice interactive și de lumini,
- electronică purtabilă (wearables) și proiecte maker.

Conexiuni interdisciplinare

Materie	Conexiune	Exemple
Fizică	Măsurarea și înțelegerea fenomenelor	Tensiune, curent, rezistență, senzori
Informatică	Programare și algoritmi	Setup, loop, condiții, bucle
Tehnologie & Inginerie	Electronică și muncă practică	Conectarea componentelor, asamblarea circuitelor
Matematică	Logică și analiză	Temporizare, compararea valorilor, stări binare
Artă & Design	Aplicații creative	Efecte LED, afișaje interactive

Competențe Digitale

Lecția dezvoltă următoarele competențe din cadrul DigComp 2.2:

Domeniu DigComp	Competență dezvoltată
1. Alfabetizarea în domeniul datelor	Înțelegerea semnalelor digitale și a datelor de la senzori
2. Comunicare și colaborare	Prezentarea rezultatelor și lucrul în perechi/grupuri
3. Creare de conținut digital	Scrierea și modificarea programelor Arduino (sketch-uri)
4. Siguranță	Utilizarea în siguranță a componentelor electronice, niveluri corecte de tensiune
5. Rezolvarea de probleme	Depanarea sistematică a circuitelor și a codului

Mai specific, lecția dezvoltă:

- alfabetizare digitală și introducere în gândire computațională
- rezolvare de probleme prin depanare sistematică,
- abilități bazice de programare cu microcontrolere reale,
- Înțelegerea interfațelor hardware-software,
- lucru în echipă și colaborare într-un mediu STEM.

Ideii de extindere a proiectului

Elevii care finalizează ambele exerciții pot extinde proiectul prin:

- adăugarea unui servomotor și controlul mișcării prin cod,
- conectarea unui senzor de lumină (LDR) pentru a controla automat luminozitatea LED-ului,
- utilizarea Monitorului Serial pentru a afișa mesaje și valori de la senzori,
- adăugarea unui buton și programarea unui comportament de comutator (toggle),
- construirea unui sistem de semafor cu 3 LED-uri și secvență temporizată.

Extindere	Dificultate	Abilități noi
Mesaje de Monitorizare Serială	 de bază	Serial.begin(), Serial.println()
Comutator cu buton	 Intermediară	digitalRead, INPUT_PULLUP, logică de stare
Control de luminozitate cu LDR	 Intermediară	analogRead, map(), calibrarea senzorilor
Controlul unui servomotor	 Avansată	Librării servo, semnale PWM
Controlul unui LED prin WiFi (cu ESP32)	 Provocare	Librării WiFi, server web, concepte IoT

Concluzii

Această lecție reprezintă primul pas în lucrul cu platforma Arduino și pune bazele pentru toate activitățile STEM ulterioare din seria Digital Challenge. Prin exercițiul Blink și conectarea fizică a LED-ului, elevii înțeleg cum programul, componentele electronice și circuitul fizic funcționează ca un sistem unic

Competențele învățate aici - citirea unei scheme, setarea unor circuite, realizarea de bucle, utilizarea funcțiilor digitalWrite() și delay(), precum și realizarea de circuite pe un breadboard - sunt aplicabile în mod direct în proiecte ulterioare, de la senzori de temperatură sau sonare și sisteme automate de irigare.

Lecturi ulterioare și pași următori

Ca p continuare a acestei lecții, următoarele proiecte din seria Digital Challenge sunt recomandate:

Proiect ulterior	Subiect	Noi concepte
Proiect 1	Măsurare de temperatură & umiditate — senzor DHT11	Librării, comunicare serială, procesare
Proiect 2	Senzor de lumină — LDR	Intrare analogică, divizor de tensiune, grafic cu bare

Proiect ulterior	Subiect	Noi concepte
Proiect 3	Sonar — HC-SR04	Măsurarea distanței, servomotor, radar display
Proiect 4	Ohmmeter	Legea lui Ohm, măsurarea unui divizor de tensiune
Proiect 5	Sistem de irigație automat	Releu, controlul unei pompe, prag logic

✓ Traseu recomandat: Completează exercițiul de semnalizare → urmează la Proiectul 1 (DHT11) → continuă cu toate cele 5 proiecte în secvență

Introducere: Processing — Vizualizare în timp real pentru Proiecte Arduino

Descriere

Processing este un mediu de programare gratuit și open-source conceput pentru creație vizuală, animație și aplicații interactive. În contextul proiectelor Arduino, Processing acționează ca o interfață vizuală (front-end): primește datele trimise de Arduino prin conexiunea serială și le afișează în timp real sub formă de text, bare, grafice, culori, forme sau animații.

Această introducere pregătește elevii pentru toate cele cinci proiecte din seria *Digital Challenge*, unde Processing este utilizat pentru vizualizarea datelor culese de senzori — de la temperatură și umiditate la distanță, intensitate luminoasă, rezistență electrică sau umiditate a solului.

Această lecție urmează aceeași structură ca și introducerea în Arduino și este creată pentru documentație STEM Erasmus+, încurajând învățarea prin explorare și înțelegerea practică a modului în care datele sunt preluate din lumea fizică și transpuse în interfațe vizuale.

Obiective

Elevii vor:

- înțelege ce este Processing și cum diferă de Arduino IDE,
- instala și configura mediul Processing,
- înțelege structura unui sketch Processing (`setup()` și `draw()`),
- învața să primească date seriale de la Arduino,
- afișa valorile senzorilor sub formă de text, forme și bare colorate,
- utiliza funcția `serialEvent()` pentru gestionarea eficientă a datelor
- vor crea o vizualizare completă, în timp real conectată la un senzor Arduino.

Rezultate educaționale

După această lecție, elevii vor fi capabili să:

- explice rolul mediului Processing într-un sistem Arduino + PC,
- scrie și să ruleze o schiță Processing de bază,
- folosească biblioteca serială pentru a primi date de la Arduino,
- analizeze valorile comma-separated (CSV) trimise de Arduino,
- afișeze citirile senzorilor folosind `text()`, `rect()`, `ellipse()` și `fill()`
- răspundă unor date de intrare utilizând `serialEvent()`,
- construiască vizualizări în timp real pentru cel puțin un proiect utilizând senzori.

Software și echipament necesare

Componentă	Versiune / Detalii	Scop
Processing IDE	4.x (cea mai recentă) — processing.org	Scrierea și rularea vizualizărilor
Arduino IDE	2.x (cea mai recentă) — arduino.cc	Programarea plăcii Arduino
Arduino Uno	Placă hardware	Trimiterea datelor de la senzori prin serial
Cablu USB	Standard A-B sau A-micro	Conectarea Arduino la calculator
Orice senzor	DHT11, LDR, HC-SR04, etc.	Generarea datelor de vizualizat

✓ Processing e gratuit și rulează sub Windows, macOS și Linux. Poate fi descărcat de pe: processing.org/download

Fundamente teoretice

Ce este Processing?

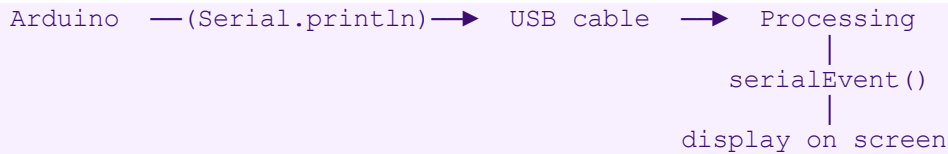
Processing este un limbaj și mediu de programare construit pe baza Java, creat special pentru design vizual și aplicații interactive. A fost creat la MIT Media Lab în 2001 și este utilizat extensiv în artă digitală, educație și programare creativă.

În ecosistemul Arduino, Processing joacă rolul unui dashboard vizual — nu poate controla în mod direct elemente hardware, dar excelează la primirea de date și transformarea unor date numerice în elemente vizuale relevante de ieșire.

Caracteristică	Arduino IDE	Processing
Scop principal	Control hardware (senzori, motoare)	Creare interfață vizuală (grafică, diagrame)
Rulează pe	Microcontroler (placă Arduino)	Calculator (Windows, macOS, Linux)
Ieșire	Semnale către componente	Grafică pe ecran
Flux de date	Trimite date prin Serial	Primește date prin Serial
Limbaj de bază	C / C++	Java
Structură sketch	setup() + loop()	setup() + draw()

Cum funcționează împreună Arduino și Processing

Cele două medii comunică prin comunicare serială prin cablul USB:



Fluxul de date în toate cele cinci proiecte *Digital Challenge* urmează acest model:

- Senzorul citește o valoare fizică (temperatură, distanță, lumină...),
- Arduino procesează valoarea și o trimite prin `Serial.println()`,
- Processing primește șirul de caractere (string-ul), îl analizează și îl convertește într-un număr,
- Processing desenează pe ecran text, bare, culori sau forme..

💡 Ambele programe trebuie să folosească aceeași rată de transfer (baud rate, de exemplu 9600).

The Processing Sketch Structure

Structura unui sketch în Processing

Fiecare program Processing (numit sketch) are două funcții obligatorii — exact ca Arduino:

```

void setup() {
  // Runs ONCE at the start
  // Set window size, initialize serial port
}

void draw() {
  // Runs CONTINUOUSLY - like loop() in Arduino
  // Draw shapes, text, update the display
}
  
```

Funcție	Când rulează	Utilizare tipică
setup()	O dată, la start	size(), configurare Serial, variabile inițiale
draw()	Continuu (~60 fps)	background(), text(), rect(), forme
serialEvent()	Când sosesc date seriale noi	Citirea și analiza datelor de la Arduino
mousePressed()	La click pe mouse	Butoane UI, interacțiuni
keyPressed()	La apăsarea unei taste	Trimiterea de comenzi înapoi la Arduino

Biblioteca Serială

Processing comunică cu Arduino folosind biblioteca sa integrată Serial. Pașii cheie:

```
import processing.serial.*; // 1. Import the library

Serial myPort;              // 2. Declare serial port object

void setup() {
  size(400, 300);
  // 3. List available ports (helpful for finding the right one)
  printArray(Serial.list());
  // 4. Open the port — index [0] is usually the first available
  myPort = new Serial(this, Serial.list()[0], 9600);
}
```

✓ Dacă `Serial.list()[0]` este eronat, verifică indexul. Rulează `printArray(Serial.list())` pentru a vedea care sunt toate porturile disponibile.

⚠ Închideți Serial Monitor din Arduino IDE înainte de a rula Processing — nu pot folosi ambele același port în același timp.

Funcții principale de desenare în Processing

Funcție	Descriere	Exemplu
<code>size(w, h)</code>	Setează lățimea și înălțimea ferestrei	<code>size(600, 400)</code>
<code>background(r,g,b)</code>	Umple fundalul cu o culoare	<code>background(255)</code> — alb
<code>fill(r,g,b)</code>	Setează culoarea de umplere pentru forme	<code>fill(255, 0, 0)</code> — roșu
<code>stroke(r,g,b)</code>	Setează culoarea conturului	<code>stroke(0)</code> — negru
<code>text(str, x, y)</code>	Desenează un text la o poziție dată	<code>text("Salut", 50, 100)</code>
<code>textSize(n)</code>	Setează dimensiunea fontului	<code>textSize(20)</code>
<code>rect(x, y, w, h)</code>	Desenează un dreptunghi	<code>rect(50, 100, 200, 30)</code>
<code>ellipse(x, y, w, h)</code>	Desenează o elipsă / cerc	<code>ellipse(200, 200, 50, 50)</code>
<code>line(x1,y1,x2,y2)</code>	Desenează o linie	<code>line(0, 0, 200, 200)</code>
<code>map(val,l1,h1,l2,h2)</code>	Scalează o valoare la un interval nou	<code>map(temp, 0, 100, 0, 300)</code>
<code>nf(n, d1, d2)</code>	Formatează un număr ca text	<code>nf(3.14159, 1, 2)</code> → "3.14"

Culorile în Processing

Processing utilizează valori RGB de culoare (0–255) pentru toate funcțiile de colorare:

```
fill(255, 0, 0);    // Red
fill(0, 255, 0);    // Green
fill(0, 0, 255);    // Blue
fill(255, 255, 0);  // Yellow
fill(0);            // Black (shorthand)
fill(255);          // White (shorthand)
fill(128, 0, 128);  // Purple

// With transparency (alpha channel 0-255):
fill(255, 0, 0, 150); // Semi-transparent red
```

Funcția map() — Esențială pentru vizualizare

Funcția map() scalează o valoare dintr-un domeniu de valori într-altul. Acest lucru este esențial când sunt vizualizate date culese de senzori în format bară pentru că valorile citirii senzorului (0–1023) trebuie să încapă în ecran:

```
// Valoarea senzorului 0-1023 scalată la lățimea barei de 0-300
// pixeli:

float barWidth = map(sensorValue, 0, 1023, 0, 300);
rect(50, 100, barWidth, 30);

// Temperatura 0-50°C scalată la intensitatea culorii roșu:
float r = map(temperature, 0, 50, 0, 255);
fill(r, 0, 0);
```

💡 map() este utilizată în fiecare dintre proiectele seriei Digital Challenge. Trebuie stăpânită cât mai curând!

Mediul de Dezvoltare Processing (PDE)

Mediul de Dezvoltare Processing (PDE) include un editor de text, un compilator și o fereastră de afișare. Acesta permite crearea de software în cadrul unui set de restricții conceput cu atenție.

PDE constă dintr-un editor de text simplu pentru scrierea codului, o zonă de mesaje, o consolă de text, taburi pentru gestionarea fișierelor, o bară de instrumente cu butoane pentru acțiuni frecvente și o serie de meniuri.

Consultați linkul: <https://processing.org/environment>

Component PDE	Scop
Editor de Text	Scrie și editează cod pentru Processing sketch
Aria de mesaje	Furnizează feedback pe durata salvării, exportului și afișează erori
Consola	Output de tip text din sketches — rezultate print() și println()
Bara de Instrumente	Butoane de Run și Stop pentru controlul rapid al sketch
Tabs	Asigură managementul multiplelor files dintr-un sketch
Meniuri	File, Edit, Sketch, Debug, Tools, Help

 In Processing, un program realizat este denumit “sketch”. Sketch-urile sunt “depozitate” în Sketchbook — un director de pe calculatorul utilizatorului..

Butoanele barei de instrumente

Buton	Acțiune
► Run	Compilează codul și deschide o nouă fereastră de afișare (modul Java)
■ Stop	Oprește un sketch care rulează a running sketch

Meniul File

Comandă	Descriere
New	Creează un sketch nou denumit cu data curentă (format: sketch_AAMMDDa)
Open...	Deschide un sketch într-o fereastră nouă
Open Recent	Selectează din sketch-urile închise recent
Sketchbook...	Arată lista tuturor sketch-urilor din dosarul sketchbook
Examples...	Răsfoiește exemple de cod integrate și furnizate de comunitate
Close	Închide fereastra sketch-ului curent
Save	Salvează sketch-ul deschis în starea sa actuală

Comandă	Descriere
Save As...	Salvează cu un alt nume (nu înlocuiește versiunea anterioară)
Export Application...	Exportă ca aplicație executabilă de sine stătătoare și deschide dosarul de ieșire
Preferences...	Configurează setările Processing (pe macOS: meniul Processing)
Quit	Părăsește Processing și închide toate ferestrele (pe macOS: meniul Processing)

Meniul Edit

Comandă	Descriere
Undo	Anulează ultima comandă sau introducere
Redo	Reface ultima acțiune anulată
Cut	Elimină și copiază textul selectat în clipboard
Copy	Copiază textul selectat în clipboard
Copy as HTML	Copiază codul formatat ca HTML (cu același aspect ca în PDE)
Paste	Inserează conținutul clipboard-ului la poziția cursorului
Select All	Selectează tot textul din fișierul deschis în prezent
Auto Format	Formatează codul într-un aspect mai ușor de citit (denumit anterior Beautify)
Comment / Uncomment	Comentează sau decommentează textul selectat
Increase Indent	Mărește indentarea textului selectat cu două spații
Decrease Indent	Micșorează indentarea textului selectat cu două spații
Find...	Caută și opțional înlocuiește un șir de text
Find Next	Caută următoarea apariție a șirului căutat
Find Previous	Caută apariția anterioară a șirului căutat

Meniul Sketch

Comandă	Descriere
Run	Compilează, deschide fereastra de afișare și rulează sketch-ul
Present	Rulează sketch-ul pe tot ecranul pe un fundal de culoare unică (Esc pentru ieșire)
Tweak	Rulează cu editare în timp real a culorilor și valorilor variabilelor (sketch-ul trebuie salvat mai întâi)
Stop	Oprește sketch-ul care rulează în prezent
Import Library...	Adaugă instrucțiuni de import; selectați Add Libraries... pentru a instala altele noi
Show Sketch Folder	Deschide dosarul care conține fișierele sketch-ului curent
Add File...	Adaugă o imagine, un font sau un fișier media în dosarul data/ al sketch-ului

Meniul Debug

Comandă	Descriere
Enable Debugger	Activează depanatorul (butonul Run se schimbă în Debug)
Continue	Avansează execuția codului până la următorul punct de oprire (breakpoint)
Step	Avansează câte o linie
Step Into	Intră în apelul unei funcții definite de utilizator
Step Out	Iese dintr-o funcție înapoi în zona apelantă
Toggle Breakpoint	Adaugă sau elimină un punct de oprire (numărul liniei este înlocuit cu <>)



Depanatorul este activat din Debug → Enable Debugger, sau făcând click pe pictograma cu fluture din colțul din dreapta-sus al PDE-ului..

Meniul Tools

Comandă	Descriere
Archive Sketch	Salvează o copie .zip a sketch-ului curent în același dosar
Color Selector...	Alege culori și copiază valorile lor HSB, RGB sau Hex
Create Font...	Convertește fonturile de sistem în format Processing VLW pentru utilizare în sketch-uri
Theme Selector	Alege sau creează teme de culori personalizate pentru PDE
Movie Maker	Creează un film QuickTime dintr-o secvență de imagini exportate
Manage Tools...	Deschide Managerul de Contribuții pentru a răsfoi și instala instrumente

Setări / Preferințe

PDE-ul este extrem de configurabil. Cele mai comune preferințe se modifică în fereastra Preferences (meniul File pe Windows/Linux, meniul Processing pe macOS):

Setare/Preferință	Descriere
Sketchbook location	Setează dosarul utilizat ca Sketchbook
Language	Selectează limba meniului (necesită repornire)
Editor font size	Setează dimensiunea fontului în editorul de cod
Console font size	Setează dimensiunea fontului în consolă
Background color when Presenting	Culoarea utilizată la rularea unui sketch în modul Present
Continuously check for errors	Activează/dezactivează verificarea erorilor în timp real
Code completion with Ctrl-Space	Activează/dezactivează autocompletarea codului
Suggest import statements	Sugerează automat biblioteci când este detectat codul acestora
Increase maximum available memory	Alocă mai multă memorie RAM pentru sketch-urile care folosesc fișiere media mari
Run sketches on display	Selectează monitorul pe care se va deschide fereastra sketch-ului (configurații multi-monitor)

Sketch-uri și Sketchbook

Toate proiectele Processing se numesc sketch-uri. Fiecare sketch are propriul său dosar — fișierul principal are același nume cu dosarul și are extensia .pde (Processing Development Environment).

```
Sketch name:  MyProject
Folder:      MyProject/
Main file:   MyProject/MyProject.pde
Media files: MyProject/data/ ← images, fonts, sounds go here
```

Sketch-urile sunt salvate implicit în Sketchbook. Pentru a-i găsi locația:

- Mergeți la **File** → **Preferences** (sau **Processing** → **Preferences** pe macOS).
- Verificați câmpul **Sketchbook location** din partea de sus.

✓ Sketch-urile sunt salvate implicit în Sketchbook. Pentru a-i găsi locația:

Mergeți la File → Preferences (sau Processing → Preferences pe macOS).
Verificați câmpul Sketchbook location din partea de sus.

Motoare de randare (Renderers)

Processing are trei motoare de randare pe ecran integrate, selectate prin funcția size():

Motor de randare	Descriere	Cel mai bun pentru
Default (fără argument)	Motor 2D standard	Majoritatea sketch-urilor 2D și afișajelor de senzori
P2D	Motor 2D mai rapid, dar mai puțin precis (OpenGL)	Grafică 2D de mare viteză
P3D	Geometrie 3D cu cameră, iluminare, materiale (OpenGL)	Vizualizări 3D, radar
FX2D	Motor 2D de înaltă calitate (bibliotecă separată JavaFX)	Afișaje mari sau de înaltă rezoluție

```
// Default 2D renderer:
void setup() { size(400, 300); }

// P3D renderer for 3D graphics:
void setup() { size(400, 300, P3D); }
```

Sistemul de coordonate

Processing folosește un sistem de coordonate cu **originea (0, 0)** în colțul din stânga-sus. X crește spre dreapta; Y crește în jos.

```
// In a 320×240 window:  
// (0, 0)      = upper-left corner  
// (320, 240) = lower-right corner  
// (319, 239) = last visible pixel (pixels draw right and below the  
coordinate)  
  
// In P3D, Z=0 is at the screen surface; negative Z goes into the scene.
```

💡 De reținut: Y crește în jos. `rect(50, 100, 200, 30)` începe desenarea de la 100 de pixeli față de SUS — nu de jos.

Stiluri de programare

Processing suportă două niveluri de complexitate pentru sketch-uri:

Stil	Descriere	Când se folosește
Static sketch	Fără <code>setup()</code> sau <code>draw()</code> — codul rulează o dată și se oprește	Învățarea noțiunilor de bază: coordonate, forme, culori
Active sketch	Folosește bucla <code>setup()</code> + <code>draw()</code> pentru animație și interacțiune	Date de la senzori, afișaj în timp real, interfață interactivă

Exemplu de Sketch Static

```
// Runs once – draws a yellow rectangle:  
size(200, 200);  
background(255);  
noStroke();  
fill(255, 204, 0);  
rect(30, 20, 50, 50);
```

Exemplu de Sketch Activ — urmărește cursorul mouse-ului

```
void setup() {  
  size(200, 200);  
  noStroke();  
  fill(0, 102, 153, 204);  
}  
  
void draw() {  
  background(255);  
  rect(width - mouseX, height - mouseY, 50, 50);  
  rect(mouseX, mouseY, 50, 50);  
}
```

Taburi și fișiere multiple

Pentru sketch-uri mai mari, codul poate fi împărțit în mai multe taburi. Apăsați butonul cu săgeată din dreapta taburilor pentru a crea, renumi sau șterge taburi.

Când un program cu mai multe taburi este rulat, toate taburile sunt compilate împreună. Clasele din alte taburi devin clase interioare — ele nu pot avea variabile statice. Pentru fișiere separate în stil Java complet, adăugați un tab cu extensia .java.

✓ Taburile sunt destinate utilizatorilor avansați. Pentru proiectele Digital Challenge, un sketch într-un singur fișier este suficient.

Adăugarea bibliotecilor, instrumentelor și modurilor

Ce doriți să adăugați	Cum se adaugă
Bibliotecă (Library)	Sketch → Import Library → Add Library... → select → Install
Instrument (Tool)	Tools → Manage Tools... → select → Install
Mod (Mode)	Mode menu (upper-right) → Add Mode... → select → Install
Exemple (Examples)	File → Examples... → Add Examples button → select → Install

Pentru proiectele Digital Challenge, instalați biblioteca Serial (integrată) și verificați dacă este disponibilă:

```
// At the top of every project sketch:
import processing.serial.*;
```

Command Line Interface (CLI)

Processing include o interfață CLI pentru rularea sketch-urilor din terminal — utilă pentru automatizare și sisteme fără interfață grafică (headless):

Sistem de operare	Comandă
Windows	processing --help more
Linux	processing --help
macOS	cd /Applications/Processing.app/Contents/MacOS then ./Processing --help

Exercițiul 1: Primul tău Sketch în Processing

Înainte de conectarea la Arduino, elevii învață mediul Processing creând un sketch animat simplu care desenează forme și text — fără a fi nevoie de hardware.

Scop

Desenarea unei ferestre cu un dreptunghi colorat, puțin text și un cerc care își schimbă culoarea în timp.

Cod Processing

```
int frameCount = 0;

void setup() {
  size(500, 300);
  textAlign(CENTER, CENTER);
}

void draw() {
  background(30);           // Dark gray background
  frameCount++;

  // Animated color using sin wave
  float r = 127 + 127 * sin(frameCount * 0.05);
  float g = 127 + 127 * sin(frameCount * 0.03);

  // Colored rectangle
  fill(r, g, 100);
  rect(50, 80, 400, 60, 10); // Rounded corners

  // White text centered in rectangle
  fill(255);
  textSize(22);
  text("Processing + Arduino = Power!", 250, 110);

  // Animated circle
  fill(r, 50, 200);
  ellipse(250, 220, 60, 60);
}
```

Ce demonstrează acest Sketch

Concept	Unde în cod
Setarea dimensiunii ferestrei	size(500, 300) in setup()
Animație continuă	draw() runs ~60 times per second
Culoare dinamică folosind matematică	sin() function on frameCount

Concept	Unde în cod
Desenarea unui dreptunghi	rect(x, y, width, height, radius)
Desenarea textului	text(), textSize(), textAlign()
Desenarea unui cerc	ellipse(cx, cy, width, height)

✓ Apasă Ctrl+R (Windows/Linux) sau Cmd+R (Mac) pentru a rula. Apasă Ctrl+. pentru a opri.

Exercițiul 2: Primirea datelor de la Arduino

Acest exercițiu conectează Processing la o placă Arduino. Arduino trimite un număr prin conexiunea serială la fiecare 500 ms, iar Processing îl afișează pe ecran.

Pasul 1 — Cod Arduino (încarcă mai întâi)

```
int counter = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(counter); // Send number to Processing
  counter++;
  if (counter > 100) counter = 0;
  delay(500);
}
```

Încarcă pe Arduino și închide Serial Monitor înainte de a rula Processing.

Pasul 2 — Cod Processing

```
import processing.serial.*;

Serial myPort;
String incomingData = "";
int receivedValue = 0;

void setup() {
    size(400, 250);
    printArray(Serial.list());          // Check available ports
    myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
    background(240);

    // Display received value as text
    fill(50);
    textSize(20);
    text("Received from Arduino: " + receivedValue, 50, 80);

    // Display as a bar
    fill(100, 150, 255);
    float barW = map(receivedValue, 0, 100, 0, 300);
    rect(50, 120, barW, 40);

    // Label on bar
    fill(0);
    textSize(16);
    text(receivedValue + " / 100", 50 + barW/2, 140);
}

void serialEvent(Serial p) {
    incomingData = p.readStringUntil('\n');
    if (incomingData != null) {
        incomingData = trim(incomingData);
        if (incomingData.matches("\\d+")) {
            receivedValue = int(incomingData);
        }
    }
}
```

Explicația codului

Parte	Explicație
import processing.serial.*	Importă biblioteca de comunicare serială
Serial myPort	Obiect ce reprezintă conexiunea pe portul serial

Parte	Explicație
<code>Serial.list()[0]</code>	Selectează primul port COM disponibil
<code>serialEvent(Serial p)</code>	Apelat automat când sosesc date noi de la Arduino
<code>readStringUntil('\n')</code>	Citește o linie de text (până la caracterul linie nouă)
<code>trim()</code>	Elimină spațiile albe și caracterele linie nouă din șir
<code>matches("\\d+")</code>	Verifică dacă șirul conține doar cifre (conversie sigură)
<code>map(value, 0,100, 0,300)</code>	Scalează valoarea 0–100 la o lățime de pixeli de 0–300

Exercițiul 3: Analiza valorilor multiple (Format CSV)

În proiectele 1, 3 și 5, Arduino trimite mai multe valori într-o singură linie, separate prin virgulă. Acest exercițiu îi învață pe elevi cum să proceseze (parseze) acel format în Processing.

Cod Arduino — Trimiterea a două valori

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  float temperature = 23.5;    // In real project: sensor reading  
  float humidity     = 61.0;  
  Serial.print(temperature);  
  Serial.print(",");           // Comma separator  
  Serial.println(humidity);    // Newline at end  
  delay(1000);  
}
```

Cod Processing — Analiza a două valori

```
import processing.serial.*;

Serial myPort;
float temperature = 0;
float humidity    = 0;

void setup() {
    size(500, 300);
    myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
    background(255);
    fill(0);
    textSize(22);
    text("Temperature: " + nf(temperature, 1, 1) + " \u00B0C", 50,
100);
    text("Humidity:      " + nf(humidity, 1, 1) + " %", 50, 150);

    // Temperature bar (red)
    fill(220, 50, 50);
    rect(50, 170, map(temperature, 0, 50, 0, 380), 25);

    // Humidity bar (blue)
    fill(50, 100, 220);
    rect(50, 210, map(humidity, 0, 100, 0, 380), 25);
}

void serialEvent(Serial p) {
    String data = p.readStringUntil('\n');
    if (data != null && data.contains(",")) {
        String[] values = split(trim(data), ",");
        if (values.length == 2) {
            temperature = float(values[0]);
            humidity    = float(values[1]);
        }
    }
}
}
```

Modelul de analiză CSV — Utilizat în toate proiectele

Pas	Cod	Explicație
1. Citire linie	<code>readStringUntil('\n')</code>	Așteaptă linia completă ce se termină cu linie nouă
2. Verificare non-null	<code>if (data != null)</code>	Evită erorile la citiri goale
3. Verificare format	<code>data.contains(",")</code>	Confirmă că este în format CSV
4. Separare șir	<code>split(trim(data), ",")</code>	Divide într-un tablou la caracterul virgulă
5. Verificare lungime	<code>values.length == 2</code>	Asigură numărul corect de valori
6. Conversie	<code>float(values[0])</code>	Convertește șirul de caractere în număr



Acest model exact apare în Proiectele 1, 3 și 5. Odată ce îl înveți aici, îl vei recunoaște peste tot.

Exercițiul 4: Processing Interactiv – Trimiterea de comenzi înapoi la Arduino

Processing poate trimite date și înapoi către Arduino — de exemplu, pentru a porni/opri o pompă (Proiectul 5) sau pentru a controla un servomotor (Proiectul 3). Acest exercițiu arată cum să folosești click-urile de mouse și apăsările de taste pentru a trimite comenzi

```
void setup() {  
  Serial.begin(9600);  
  pinMode(13, OUTPUT);          // Built-in LED  
}  
void loop() {  
  if (Serial.available() > 0) {  
    char cmd = Serial.read();  
    if (cmd == '1') digitalWrite(13, HIGH); // LED ON  
    else if (cmd == '0') digitalWrite(13, LOW); // LED OFF  
  }  
}
```

Cod Arduino — Primirea comenzii

Cod Processing — Butoane și control din tastatură

```
import processing.serial.*;  
Serial myPort;  
boolean ledOn = false;  
  
void setup() {  
  size(400, 220);  
  myPort = new Serial(this, Serial.list()[0], 9600);  
  textAlign(CENTER, CENTER);  
}  
  
void draw() {  
  background(240);  
  fill(30);  
  textSize(18);  
  text("Arduino LED Control", 200, 30);  
  text("Click buttons or press 1 / 0 on keyboard", 200, 60);  
}
```

```

// ON button
fill(ledOn ? color(0, 200, 0) : color(180));
rect(60, 100, 120, 50, 8);
fill(255);
text("ON", 120, 125);

// OFF button
fill(!ledOn ? color(200, 50, 50) : color(180));
rect(220, 100, 340, 50, 8);
fill(255);
text("OFF", 340, 125);

// Status
fill(30);
textSize(16);
text("Status: LED is " + (ledOn ? "ON" : "OFF"), 200, 190);
}

void mousePressed() {
  if (mouseX>60&&mouseX<180&&mouseY>100&&mouseY<150) {
myPort.write('1'); ledOn=true;  }
  if (mouseX>220&&mouseX<340&&mouseY>100&&mouseY<150) {
myPort.write('0'); ledOn=false; }
}

void keyPressed() {
  if (key=='1') { myPort.write('1'); ledOn=true;  }
  if (key=='0') { myPort.write('0'); ledOn=false; }
}

```

Sarcini pentru elevi

Nivel	Sarcină	Competențe exersate
 Bazic	Afișează un singur număr de la Arduino ca text	Serial, parseInt, text()
 Intermediar	Desenează o bară colorată proporțională cu valoarea senzorului	map(), rect(), fill()
 Avansat	Analizează două valori (CSV) și afișează-le pe ambele cu bare	split(), multiple visual elements

Nivel	Sarcină	Competențe exersate
● Provocare	Adaugă butoane ON/OFF care trimit comenzi la Arduino	mousePressed(), Serial.write()
● Challenge	Create a real-time graph of the last 100 readings	Arrays, shifting values, line()

Erori comune & Depanare

Eroare / Problemă	Cauză probabilă	Soluție
Portul nu a fost găsit / lipsă <code>Serial.list()</code>	Arduino nu este conectat sau lipsește driverul	Verifică conexiunea USB, instalează driverul CH340 dacă este necesar
Port ocupat / în uz	Serial Monitor din Arduino este încă deschis	Închide mai întâi Serial Monitor din Arduino IDE
Nu se primesc date	Index greșit de port în <code>Serial.list()[0]</code>	Rulează <code>printArray(Serial.list())</code> și verifică indexul
Baud rate necorespunzător	Arduino folosește 9600, Processing folosește o altă valoare	Asigură-te că ambele folosesc <code>Serial.begin(9600)</code> și <code>new Serial(...,9600)</code>
NullPointerException	Datele sunt nule (<code>data</code> este null – linie incompletă)	Verifică întotdeauna: <code>if (data != null)</code>
Valori eronate / neinteligibile pe ecran	Analiza eșuează pe șiruri non-numerice	Adaugă: <code>if (data.matches("\\d+"))</code>
Sketch-ul Processing îngheață	<code>draw()</code> este blocat de calcule grele	Mută citirea serială în funcția <code>serialEvent()</code>

✓ Regula de aur: Dacă nimic nu funcționează, repornește atât Arduino, cât și Processing și încearcă din nou. Majoritatea problemelor de comunicare serială se rezolvă la reconectare.

Processing vs. Arduino IDE — Ghid Rapid

Concept	Arduino IDE	Processing
Punct de intrare în program	setup() + loop()	setup() + draw()
Afișare în consolă	Serial.println()	println()
Citire date de intrare	Serial.read()	myPort.readStringUntil()
Întârziere / pauză	delay(ms)	delay(ms) (avoid in draw())
Tipuri de variabile	int, float, char, bool	int, float, String, boolean
Desenare dreptunghi	— (no screen)	rect(x, y, w, h)
Desenare text	— (no screen)	text(str, x, y)
Culoare	— (no screen)	fill(r, g, b) / stroke(r, g, b)
Gestionarea evenimentelor	—	mousePressed(), keyPressed()
Importare bibliotecă	#include	import library.*

Aplicații în lumea reală

Vizualizarea bazată pe Processing este utilizată în:

- tablouri de bord pentru senzori industriali și sisteme de monitorizare în timp real,
- vizualizarea datelor științifice în laboratoare de cercetare,
- expoziții interactive în muzee și instalații educaționale,
- programare creativă și artă digitală generativă,
- jurnalism de date și vizualizarea informațiilor,
- sisteme de monitorizare a sănătății și dispozitive de biofeedback,
- monitorizarea mediului (calitatea aerului, stații meteorologice),
- prototiparea pentru dezvoltarea jocurilor și media interactivă.

Conexiuni Interdisciplinare

Materie	Conexiune	Example in proecte
Informatică	Programare, structuri de date, gestionarea evenimentelor	<code>serialEvent()</code> , tablouri (arrays) pentru grafice
Matematică	Sisteme de coordonate, scalare, funcții	<code>map()</code> , <code>sin()</code> , reprezentarea grafică a datelor
Fizică	Interpretarea datelor de la senzori, măsurători	Bare de temperatură, radar de distanță
Arte Vizuale	Teoria culorilor, compoziție, design	Design interfață utilizator (UI), afișaje codificate prin culori
Tehnologie	Gândire sistemică, integrare hardware-software	Fluxul integrat Arduino + Processing

Competențe DigComp

Lecția dezvoltă următoarele competențe din cadrul DigComp 2.2:

Aria DigComp	Competență dezvoltată
1. Alfabetizare în domeniul informațiilor și datelor	Recepționarea, analiza și afișarea datelor de la senzori în timp real
2. Comunicare și colaborare	Proiectarea de interfețe vizuale clare pentru comunicarea datelor
3. Creare de conținut digital	Scrierea de sketch-uri în Processing și crearea de programe vizuale
4. Siguranță	Înțelegerea validării datelor și a gestionării erorilor
5. Rezolvarea de probleme	Depanarea comunicării seriale și a logicii de vizualizare

În mod specific, lecția dezvoltă:

- gândirea computațională prin programare bazată pe evenimente,
- alfabetizarea datelor — transformarea numerelor brute în semnificație vizuală,
- programarea creativă și abilitățile de design al interfețelor,
- rezolvarea sistematică a problemelor și depanarea (debugging),
- gândirea integrativă — conectarea hardware-ului cu software-ul.

Idei de extindere a proiectelor

Elevii care stăpânesc noțiunile de bază își pot extinde abilitățile de Processing prin:

- desenarea unui grafic derulant în timp real al ultimelor 100 de citiri ale senzorului,
- adăugarea de alerte sonore folosind biblioteca audio Minim,
- salvarea datelor senzorului într-un fișier CSV folosind `PrintWriter`,
- crearea unui tablou de bord complet cu afișaje pentru senzori multipli,
- adăugarea unui ceas și a unei mărci temporale (timestamp) pentru fiecare citire înregistrată,
- construirea unei versiuni web folosind p5.js (versiunea JavaScript a Processing),
- conectarea mai multor plăci Arduino la un singur sketch Processing.

Extindere	Dificultate	Concept-cheie
Grafic derulant în timp real	● Intermediar	Tablouri (arrays), deplasarea valorilor, <code>line()</code>
Salvarea datelor în fișier CSV	● Intermediar	<code>PrintWriter</code> , <code>saveStrings()</code>
Alarmă sonoră la prag	● Avansat	Biblioteca Minim, redare audio
Tablou de bord multi-senzor	● Avansat	Porturi seriale multiple sau date multiplexate
Versiune web p5.js	● Provocare	JS, HTML Canvas, WebSerial API

Cum se conectează această introducere la cele cinci proiecte

Proiect	Senzor	Vizualizare Processing	Tehnică cheie
1 DHT11	Temperatură și umiditate	Text cu două valori + bare	Analiză CSV, 2 valori
2 LDR	Intensitatea luminii	Bară unică + schimbare de culoare	<code>map()</code> , culoare de fundal
3 HC-SR04	Distanță (sonar)	Afișaj tip baleiaj radar	Trigonometrie, elipsă, linie
4 Ohmetru	Rezistență	Afișaj numeric + indicator (ceas)	<code>nf()</code> , <code>arc()</code> , text
5 Iriagare	Umiditatea solului	Afișaj stare + bară	Logica pragurilor, culoarea de umplere (<code>fill</code>)

✓ Toate cele cinci proiecte folosesc același tipar de bază: `serialEvent()` → analiză → `draw()`. Stăpânește tiparul aici, aplică-l peste tot.

Concluzie

Processing este puntea vizuală dintre lumea fizică și ecran. Odată ce elevii înțeleg modul în care datele curg de la un senzor prin portul serial al Arduino în funcția `draw()` din Processing, ei au fundația necesară pentru a construi orice vizualizare în timp real pe care și-o pot imagina.

Această introducere acoperă tot ce este necesar pentru toate cele cinci proiecte Digital Challenge: configurarea conexiunii seriale, citirea și analiza datelor, desenarea formelor și textului, utilizarea funcției `map()` pentru scalare și trimiterea de comenzi înapoi la Arduino. Aceste abilități, combinate cu cunoștințele de Arduino din introducerea anterioară, formează un set complet de instrumente STEM pentru proiecte interactive bazate pe senzori.

Pașii Următori

După finalizarea acestei introduceri, treceți la proiecte în următoarea ordine:

Pas	Proiect	Noi abilități Processing
Introducere	Intro în Arduino	Noțiuni de bază despre comunicarea serială, <code>setup/loop</code>
← Ești aici	Intro în Processing	<code>setup/draw</code> , <code>serialEvent</code> , <code>map</code> , forme vizuale
Proiectul 1	Temperatură & Umiditate DHT11	Analiză CSV, afișaj cu două bare
Proiectul 2	Senzor de lumină — LDR	Valoare unică, culoare de fundal dinamică
Proiectul 3	Sonar — HC-SR04	Afișaj radar, trigonometrie, control servomotor
Proiectul 4	Ohmetru	Numere formate, vizualizare tip indicator
Proiectul 5	Iriagare Automată	Afișare stare, interfață control pompă

Proiectul 1: Temperatură și Umiditate — DHT11

Descriere

În acest exercițiu, studenții construiesc un sistem simplu pentru măsurarea temperaturii și a umidității aerului folosind un Arduino Uno și un senzor DHT11. Datele colectate sunt transferate către un calculator prin comunicare serială și sunt vizualizate în timp real folosind Processing.

Exercițiul conectează noțiunile fundamentale de electronică, programare, fizică și vizualizare a datelor, servind ca o introducere în sistemele IoT (Internet of Things) și lucrul cu senzori.

Obiective

Students will:

- înțelege modul în care funcționează senzorii digitali,
- conecta senzorul DHT11 la un microcontroler Arduino,
- citi date de la senzor,
- utiliza comunicarea serială,
- vizualiza datele în timp real,
- dezvolta un sistem simplu de alertă bazat pe valori prag prestabilite.

Rezultatele învățării

După acest atelier, studenții vor fi capabili să:

- conecteze corect un senzor digital la un microcontroler,
- utilizeze biblioteci în mediul Arduino IDE,
- citească și să proceseze date de la senzor,
- înțeleagă principiul comunicării seriale,
- vizualizeze datele în mediul Processing,
- analizeze modificările de temperatură și umiditate,
- modifice independent sistemul prin adăugarea de noi funcționalități.

Echipament necesar

Componentă	Cantitate	Scop
Arduino Uno	1	Microcontroler
Senzor DHT11	1	Măsurarea temperaturii și umidității
Rezistență 10kΩ	1	Rezistență de pull-up
Breadboard	1	Conectarea componentelor
Fire de conexiune (jumper)	~5	Conectarea componentelor

Componentă	Cantitate	Scop
Cablu USB	1	Conectare la calculator
Arduino IDE	1	Programare Arduino
Processing	1	Vizualizarea datelor

Context teoretic

Ce este DHT11?

DHT11 este un senzor digital care măsoară temperatura și umiditatea relativă a aerului, trimițând datele în format digital. Senzorul este potrivit pentru începători deoarece nu necesită o conversie analog-digitală, este simplu de conectat și are la dispoziție biblioteci gata făcute.

Ce este Umiditatea Relativă?

Umiditatea relativă reprezintă procentul de vapori de apă din aer raportat la cantitatea maximă pe care aerul o poate reține la o anumită temperatură și presiune atmosferică normală.

Comunicarea Serială

Arduino trimite date către calculator în mod serial, prin intermediul portului USB. De exemplu:

```
Serial.println("24.5,60");
```

Acest lucru înseamnă: temperatură = 24.5 °C, umiditate = 60 %

Exercițiul 1: Conectarea Senzorului DHT11

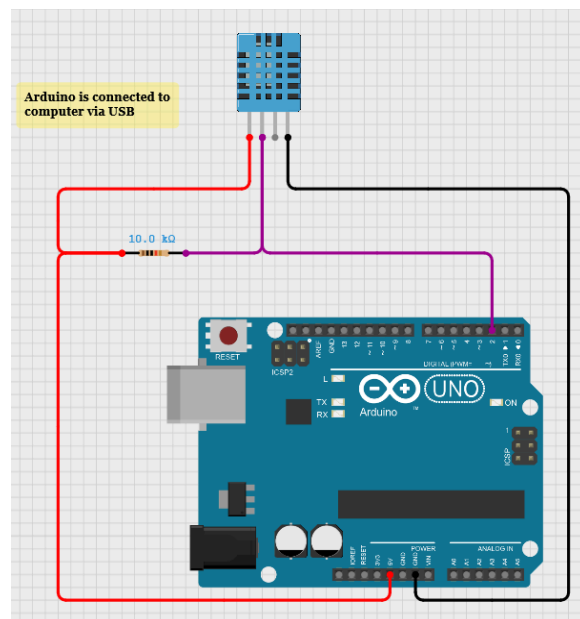
Conexiuni

Pin DHT11	Pin Arduino
VCC	5V
GND	GND
DATA	Digital pin 2

Dacă folosiți versiunea "simplă" a senzorului 3 pini:

- amplasați o rezistență de 10kΩ între VCC și DATA.

✓ Sfat: Modulul DHT11 cu 4 pini are deja o rezistență de pull-up încorporată.



Programul Arduino

Pasul 1: Instalarea bibliotecii

În Arduino IDE:

- Sketch → Include Library → Manage Libraries
- Search for: "DHT sensor library" by Adafruit
- Install the library

Pasul 2: Codul Arduino

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print(temperature);
  Serial.print(",");
  Serial.println(humidity);

  delay(2000);
}
```

Explicația codului

Comandă	Explicație
dht.readTemperature()	Citește temperatura
dht.readHumidity()	Citește umiditatea
Serial.print()	Trimite datele către calculator
delay(2000)	Așteaptă 2 secunde

✓ Datele sunt trimise ca valori separate prin virgulă, deoarece Processing le procesează cu ușurință.

Exercițiul 2: Vizualizarea datelor în Processing

Cod Processing

```

Serial myPort;
String data;
float temp, hum;

void setup() {
  size(400, 200);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(255);
  fill(0);
  textSize(20);
  text("Temperature: " + temp + " °C", 50, 80);
  text("Humidity: " + hum + " %", 50, 120);

  if (myPort.available() > 0) {
    data = myPort.readStringUntil(' ');
    if (data != null && data.contains(",")) {
      String[] values = split(trim(data), ",");
      if (values.length == 2) {
        temp = float(values[0]);
        hum = float(values[1]);
      }
    }
  }
}

```

Ce afișează programul?

- Citește datele trimise de Arduino
- Afișează temperatura și umiditatea
- Vizualizează valorile folosind bare grafice

Sarcini pentru elevi

Nivel	Sarcină
 Bază	Afișarea temperaturii și a umidității
 Intermediar	Adăugarea de bare de progres grafice
 Avansat	Adăugarea unei alarme pentru temperatură ridicată
 Provocare	Salvarea datelor într-un fișier CSV
 Provocare	Desenarea unui grafic în timp real

Exemplu de sistem de alertă

```

if (temperature > 30) {

```

```
fill(255, 0, 0);
text("WARNING: HIGH TEMPERATURE!", 40, 50);
}
```

Depanare

Problemă	Cauză posibilă
Nu se primesc date	Port COM incorect
Mesaj „Eroare senzor”	Senzor conectat necorespunzător
Valori instabile	Contact slab pe breadboard
Processing nu primește date	Neconcordanță între vitezele de comunicare serială

✓ Sfat: Folosește `Serial.list()` pentru a găsi portul COM corect.

Aplicații în lumea reală

Astfel de sisteme sunt utilizate în: case inteligente, stații meteo, sere, săli de servere, sisteme IoT și automatizări industriale.

Conexiuni interdisciplinare

Materie	Conexiune
Fizică	Temperatură, umiditate, măsurare
Informatică	Programare și procesarea datelor
Tehnologie & Inginerie	Electronică și senzori
Matematică	Analiza datelor și reprezentarea grafică

Competențe DigComp

Acest atelier dezvoltă:

- alfabetizarea digitală,
- abilitățile de rezolvare a problemelor,
- analiza datelor,
- competențele de programare,
- colaborarea într-un mediu STEM.

Idei pentru extinderea proiectului

- Adăugarea unui afișaj OLED
- Trimiterea datelor printr-o rețea Wi-Fi
- Crearea unui panou de control (dashboard) web

- Conectarea sistemului la o platformă cloud
- Construirea unei mini-stații meteo

Concluzie

Acest exercițiu servește ca o introducere în lucrul cu senzori, microcontrolere și vizualizarea datelor. Prin combinarea platformei Arduino Uno cu mediul de dezvoltare Processing, elevii dobândesc cunoștințe practice în electronică, programare și analiza datelor prin intermediul unui proiect STEM aplicativ.

Proiectul 2: Senzor de Lumină — Fotorezistență (LDR)

Descriere

Acest proiect citește intensitatea luminii ambientale folosind o fotorezistență (LDR) conectată la Arduino. Valoarea analogică (0–1023) este trimisă prin conexiunea serială către Processing, care o afișează ca text și sub forma unei bare vizuale în timp real.

Obiective

Elevii vor:

- înțelege modul în care o fotorezistență (LDR) detectează modificările intensității luminoase,
- învăța cum funcționează senzorii analogici cu un microcontroler Arduino Uno,
- realiza un circuit divisor de tensiune,
- citi și interpreta valorile senzorilor analogici,
- utiliza comunicarea serială între Arduino și calculator,
- vizualiza datele senzorilor în timp real folosind Processing,
- dezvolta vizualizări interactive simple pe baza datelor din mediu.

Rezultate educaționale

După finalizarea atelierului, elevii vor fi capabili să:

- conecteze corect un senzor LDR folosind un circuit divisor de tensiune,
- citească valori analogice de la o intrare analogică Arduino,
- explice relația dintre intensitatea luminii și rezistența electrică,
- transfere date de la senzori folosind comunicarea serială,
- creeze vizualizări de bază în timp real în Processing,
- analizeze modul în care condițiile de lumină din mediu afectează citirile senzorului,
- modifice proiectul prin adăugarea de noi funcționalități vizuale sau interactive.

Echipament necesar

Componentă	Cantitate	Scop
Arduino Uno	1	Microcontroler
Fotorezistență (LDR)	1	Detecție lumină
Rezistență 10kΩ	1	Divizor de tensiune
Breadboard	1	Conectare componente
Cabluri de conexiune (Jumper)	~5	Conectare componente
Cablu USB	1	Conectare la calculator
Processing IDE	1	Vizualizare date

Bazele Teoretice

Cum funcționează o fotorezistență?

O fotorezistență (LDR) își modifică rezistența în funcție de cantitatea de lumină. La lumină puternică, rezistența scade; la întuneric, rezistența crește.

Circuitul Divizor de Tensiune

LDR-ul este conectat în serie cu o rezistență fixă de 10kΩ. Tensiunea în punctul de mijloc la A0 variază în funcție de intensitatea luminii.

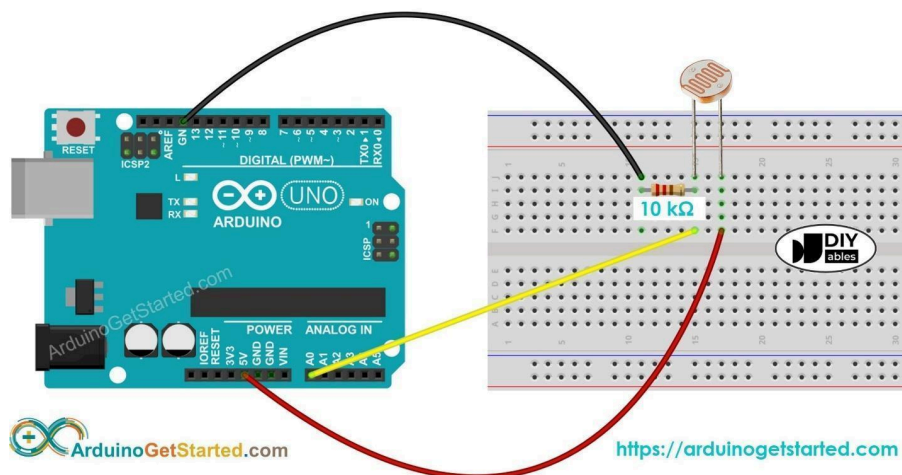
| 💡 $V_{out} = V_{in} \times R_2 / (R_1 + R_2)$. Aici R2 este LDR-ul.

Exercițiul 1: Configurarea Circuitului și Codul Arduino

Conexiuni (Cablare)

Componentă	Se conectează la
LDR (pin 1)	5V
LDR (leg 2)	A0 și Rezistență 10kΩ (pin 1)
Rezistență 10kΩ (pin 2)	GND

| ✅ Notă: Joncțiunea dintre LDR și rezistență se conectează la pinul A0 de pe Arduino.



Cod Arduino

```
const int sensorPin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int lightValue = analogRead(sensorPin); // 0 (dark) to
  1023 (bright)
  Serial.println(lightValue);
  delay(200);
}
```

Exercițiul 2: Vizualizare în Processing

Cod Processing — Afișare Text

```
import processing.serial.*;
Serial myPort;
String val = "";
int lightLevel = 0;

void setup() {
  size(400, 200);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(255);
  fill(0);
  textSize(20);
  text("Light Intensity: " + lightLevel, 50, 100);

  if (myPort.available() > 0) {
    val = myPort.readStringUntil('\n');
    if (val != null) { lightLevel = int(trim(val)); }
  }
}
```

Cod Processing — Bară + Eveniment Serial

```
import processing.serial.*;
Serial myPort;
String val = "";
int lightLevel = 0;

void setup() {
  size(400, 200);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(255);
  fill(0);
  textSize(20);
  text("Light Intensity: " + lightLevel, 50, 50);

  fill(0, 100, 255);
  rect(50, 100, map(lightLevel, 0, 1023, 0, 300), 30);
}

void serialEvent(Serial p) {
  val = p.readStringUntil('\n');
  if (val != null) {
    val = trim(val);
    if (val.matches("\\d+")) { lightLevel = int(val); }
  }
}
```

Sarcini pentru Elevi

Level	Task
● Nivel Bazic	Afișează valoarea numerică a luminii
● Nivel Intermediar	Desenează o bară colorată proporțională cu nivelul de lumină
● Nivel avansat	Schimbă culoarea fundalului în funcție de intensitatea luminii
● Provocare	Salvează valorile cu marcaje temporale (timestamps) într-un fișier text

Depanare (Troubleshooting)

Problem	Possible Cause
Valoarea rămâne la 0	LDR-ul nu este conectat sau circuitul este deschis
Valoarea rămâne la 1023	Rezistența lipsește sau GND nu este conectat
Citiri instabile	Contact slab pe breadboard

Ideii pentru extinderea proiectului

- Adăugarea unui LED care reacționează automat la intensitatea luminii
- Crearea unui sistem automat de lumină de veghe (pentru noapte)
- Utilizarea mai multor senzori LDR pentru detecția direcțională a luminii
- Conectarea sistemului la un modul WiFi pentru monitorizare de la distanță
- Construirea unui prototip simplu de iluminat pentru case inteligente (Smart Home)
- Afișarea datelor pe un ecran OLED sau LCD
- Înregistrarea datelor despre intensitatea luminii pentru analiză pe termen lung

Concluzie

Acest proiect le prezintă elevilor senzorii analogici, circuitele divizoare de tensiune și vizualizarea datelor în timp real. Combinând platforma Arduino Uno cu mediul Processing, elevii acumulează experiență practică în electronică, programare și sisteme interactive cu senzori printr-o activitate aplicată STEM.

Proiect 3: Sonar — Senzorul Ultrasonic HC-SR04

Descriere

Acest proiect utilizează senzorul ultrasonic HC-SR04 pentru a măsura distanța. Datele sunt vizualizate în Processing în cadrul a trei exerciții avansate progresiv: afișare de text, controlul servomotorului și un afișaj complet de tip radar/sonar.

Obiective

Elevii / studenții vor:

- înțelege principiul de funcționare al măsurării distanței cu ultrasunete,
- învăța cum măsoară senzorul ultrasonic HC-SR04 distanța folosind unde sonore,
- conecta și programa senzori ultrasonici și servomotoare utilizând o placă Arduino Uno,
- folosi comunicarea serială pentru transmiterea datelor de măsurare,
- vizualiza datele de la senzor în timp real folosind Processing,
- dezvolta vizualizări interactive de tip radar și sonar,
- explora legătura dintre fizică, electronică și programare prin intermediul unui proiect STEM practic.

Rezultate educaționale

După finalizarea atelierului, participanții vor fi capabili să:

- explice modul în care funcționează detecția cu ultrasunete,
- calculeze distanța pe baza timpului de întoarcere a ecoului,
- conecteze și să programeze corect senzorul HC-SR04,
- controleze un servomotor folosind Arduino,
- transmită datele senzorului către Processing prin comunicare serială,
- creeze vizualizări grafice în timp real ale măsurătorilor de distanță,
- dezvolte interfețe simple de tip radar,
- analizeze modul în care poziția senzorului și distanța până la obiect influențează măsurătorile,
- extindă în mod independent sistemul cu funcționalități și interacțiuni suplimentare.

Fundamente teoretice

Fizica detecției ultrasonice

Senzorii cu ultrasunete emit unde sonore de înaltă frecvență (~40 kHz). Când aceste unde lovesc un obiect, ele se reflectă înapoi. Senzorul măsoară timpul necesar pentru ca ecoul să revină:

$$\text{Distanța} = (\text{Viteza sunetului} \times \text{Timpul}) / 2$$

Viteza sunetului este de ≈ 343 m/s în aer. Împărțirea la 2 are loc pentru a lua în calcul drumul dus-întors al undei.

Echipamente necesare

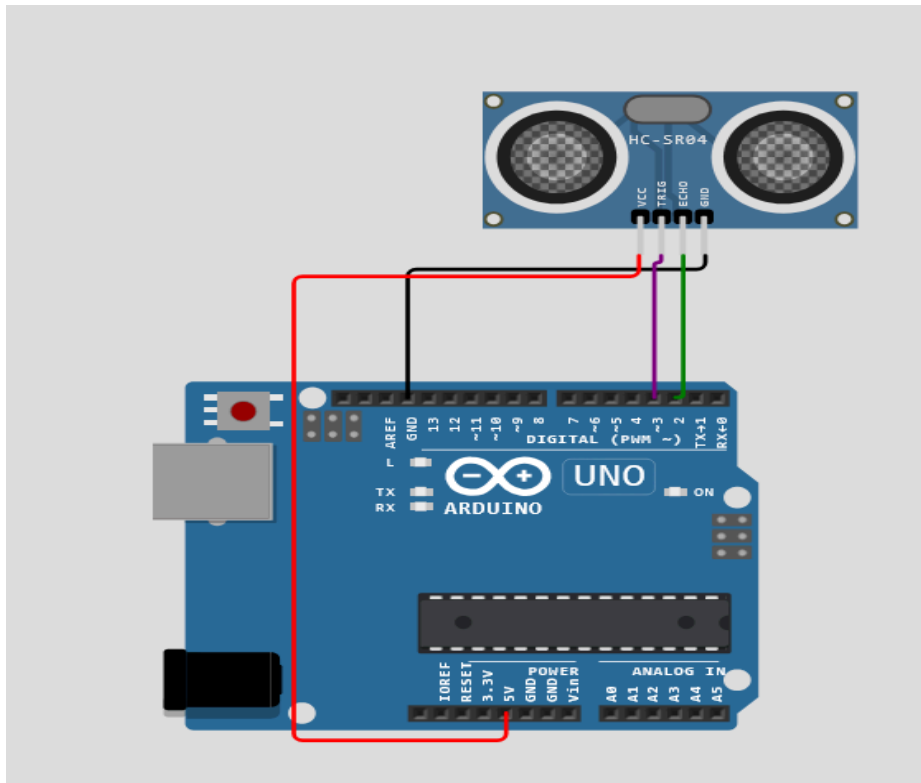
Componentă	Cantitate	Scop
Arduino Uno	1	microcontroler
HC-SR04 senzor ultrasonic	1	măsurarea distanței
SG90 servomotor	1	radar cu baleiaj (Ex. 2 & 3)
Suport din plastic pentru senzor	1	montarea senzorului pe servomotor
Breadboard + conectori jumper	1 set	conectarea componentelor
cablu USB	1	Conectarea la computer
Processing IDE	1	Vizualizare

Exercițiul 1: Măsurarea Distanței & Afișare în Processing

Componente Conexiuni

HC-SR04 Pin	Arduino Pin
VCC	5V
GND	GND
TRIG	Pin 3
ECHO	Pin 2

- Arduino Uno
- HC-SR04 ultrasonic sensor
- USB cable
- Processing IDE



Cod Arduino

```
#define trigPin 3
#define echoPin 2
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.0343 / 2;
  Serial.println(distance);
  delay(200);
}
```

Cod Processing – Vizualizarea Distanței

```
import processing.serial.*;
Serial myPort;
String val;
void setup() {
    size(400, 200);
    myPort = new Serial(this, Serial.list()[0], 9600);
}
void draw() {
    if (myPort.available() > 0)
    {
        val = myPort.readStringUntil('\n');
        if (val != null) {
            background(255);
            fill(0);
            textSize(20);
            text("Distance: " + val.trim() + " cm", 50, 100);
        }
    }
}
```

Cod Processing – Vizualizarea distanței cu bare orizontale

```
import processing.serial.*;
Serial myPort;
float distance;

void setup() {
    size(600, 200);
    printArray(Serial.list());
    myPort = new Serial(this, Serial.list()[0], 9600);
    myPort.bufferUntil('\n');
}
void draw() {
    background(255);
    fill(0);
    textSize(24);
    text("Distance: " + nf(distance, 0, 2) + " cm", 50, 50);
    // Horizontal bar
    fill(100, 200, 255);
    rect(50, 100, map(distance, 0, 200, 0, width - 100), 30);
}

void serialEvent(Serial myPort) {
    String data = myPort.readStringUntil('\n');
    if (data != null) {
        data = trim(data);
        distance = float(data);
    }
}
```

Exercițiul 2: Controlul SG90 Servo din Processing

Componente

- Arduino Uno
- SG90 Servo motor
- USB cable

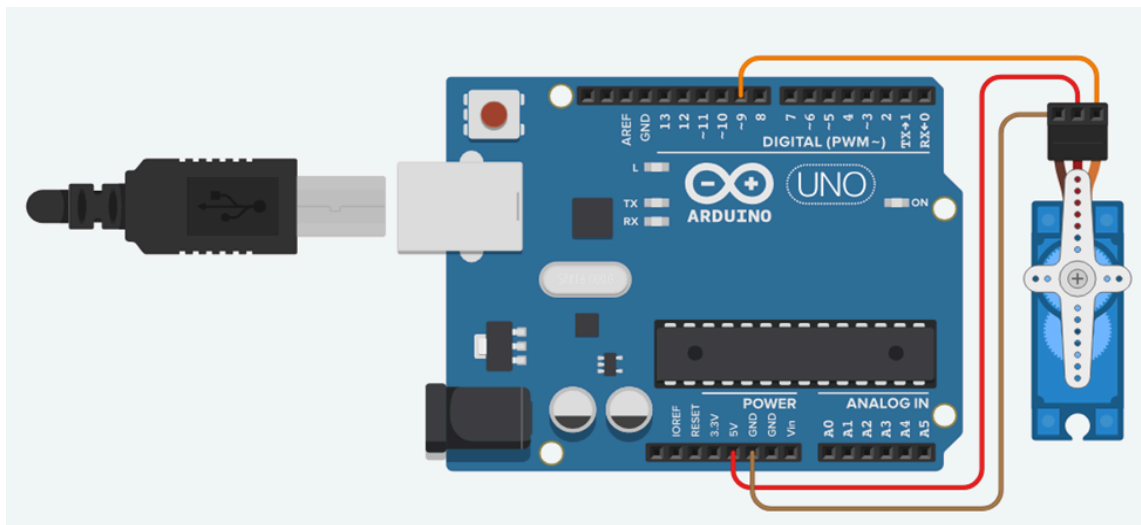
Conexiuni

Pin Servo	Pin Arduino
Signal (orange)	Pin 9
VCC (red)	5V
GND (brown)	GND

Referințe

[The Beginners Guide to Micro Servos](#)

<https://docs.arduino.cc/learn/electronics/servo-motors/>



Cod Arduino

```
#include <Servo.h>
Servo servo;
String command = "";
bool sweeping = false;
int angle = 0;
void setup() {
  Serial.begin(9600);
  servo.attach(9);
}
void loop() {
  if (Serial.available()) {
    command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "1") sweeping = true;
    else if (command != "1") sweeping = false;
  }
  if (sweeping) {
    for (angle = 0; angle <= 180 && sweeping; angle++) {
      servo.write(angle);
      delay(10);
    }
    for (angle = 180; angle >= 0 && sweeping; angle--) {
      servo.write(angle);
      delay(10);
    }
  }
}
```

Cod Processing — Butoane START / STOP

```
import processing.serial.*;

Serial myPort;
boolean isSweeping = false;

void setup() {
    size(400, 200);
    myPort = new Serial(this, Serial.list()[0], 9600);
    textAlign(CENTER, CENTER);
}

void draw() {
    background(240);
    textSize(16);
    fill(0);
    text("Servo Control", width/2, 30);

    drawButton("START", 100, 100, isSweeping);
    drawButton("STOP", 250, 100, !isSweeping);
}

void drawButton(String label, int x, int y, boolean active) {
    fill(active ? color(0, 200, 0) : color(180));
    rect(x, y, 100, 40, 7);
    fill(0);
    text(label, x + 50, y + 20);
}

void mousePressed() {
    if (mouseX > 100 && mouseX < 200 && mouseY > 100 && mouseY < 140) {
        myPort.write("1\n");
        isSweeping = true;
    } else if (mouseX > 250 && mouseX < 350 && mouseY > 100 && mouseY < 140) {
        myPort.write("0\n");
        isSweeping = false;
    }
}
```

Mod de funcționare

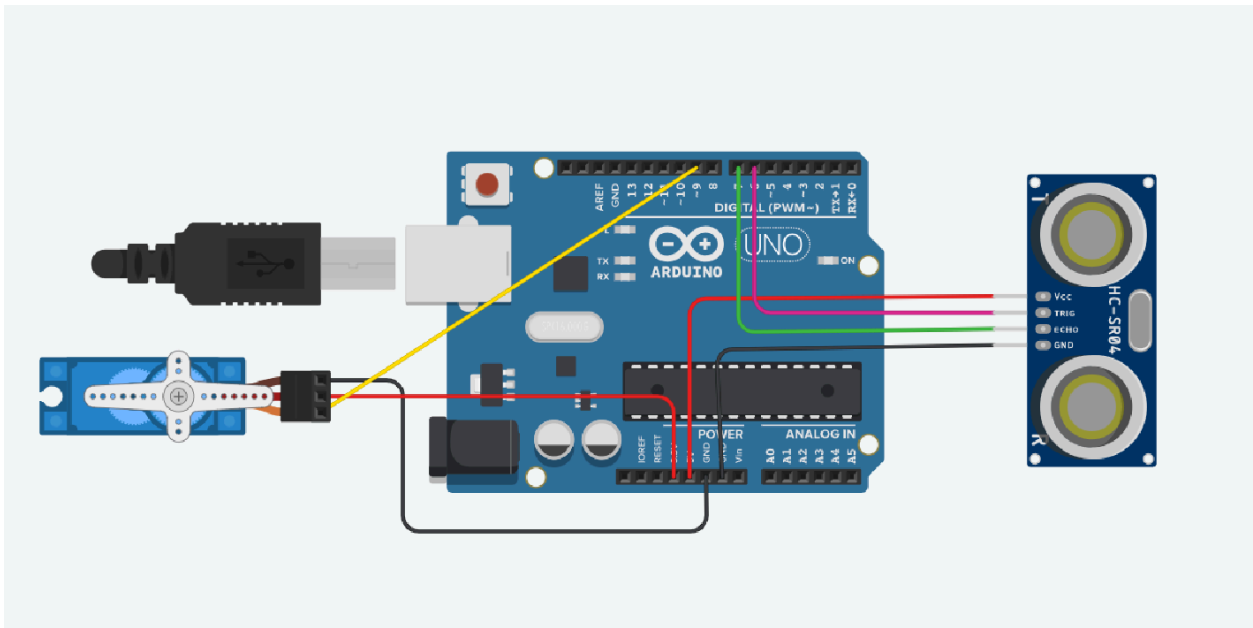
- Butonul Start trimite "START" → Arduino inițiază mișcarea de baleiaj a servomotorului.
- Butonul Stop trimite "STOP" → Arduino oprește mișcarea de baleiaj a servomotorului.
- Suplimentar pot fi programate limitări ale unghiului de baleiaj, control al vitezei unghiulare de baleiaj, respectiv feedback în timp real

Exercițiul 3: Controlul Servomotorului cu Senzor Ultrasonic

Servomotorul se rotește la 90° atunci când un obiect este detectat la la mai puțin de 50 cm și se repoziționează la 0° în alte cazuri.

Cod Arduino

```
#include <Servo.h>
// constants won't change
const int TRIG_PIN = 6; // Pin connected to Ultrasonic Sensor's TRIG pin
const int ECHO_PIN = 7; // Pin connected to Ultrasonic Sensor's ECHO pin
const int SERVO_PIN = 9; // Arduino pin connected to Servo Motor's pin
const int DISTANCE_THRESHOLD = 50; // centimeters
Servo servo; // create servo object to control a servo
// variables will change:
float duration_us, distance_cm;
void setup() {
    Serial.begin(9600); // initialize serial port
    pinMode(TRIG_PIN, OUTPUT); // set arduino pin to output mode
    pinMode(ECHO_PIN, INPUT); //set arduino pin to input mode
    servo.attach(SERVO_PIN); // attaches the servo on pin 9 to the servo
    object
    servo.write(0);
}
void loop() {
    // generate 10-microsecond pulse to TRIG pin
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    // measure duration of pulse from ECHO pin
    duration_us = pulseIn(ECHO_PIN, HIGH);
    // calculate the distance
    distance_cm = 0.017 * duration_us;
    if(distance_cm < DISTANCE_THRESHOLD)
        servo.write(90); // rotate servo motor to 90 degree
    else    servo.write(0); // rotate servo motor to 0 degree
    // print the value to Serial Monitor
    Serial.print("distance: ");
    Serial.print(distance_cm);
    Serial.println(" cm");
    delay(500);
}
```



Componente

- Arduino Uno
- ultrasonic senzor HC-SR04
- Servomotor SG90
- Suport din plastic (pentru rotirea senzor)
- Processing IDE

Referințe

[Arduino Ultrasonic Sensor + Servo Motor Tutorial](#)

Sarcini pentru elevi

Scrieți un program Processing care afișează distanțe măsurată în două format vizuale:

- a) ca o valoare numerică, utilizând text
- b) ca o bară orizontală cu lungime proporțională cu distanța

Exercițiul 4: Construirea unui scanner radar (Sonar)

|  Acest exercițiu este adecvat ca temă a unui proiect.

Cod Arduino — Baleiaj Radar

```
#include <Servo.h>
Servo myServo;
#define trigPin 9
#define echoPin 10
void setup() {
  Serial.begin(9600);
  myServo.attach(6);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  for (int angle = 0; angle <= 180; angle += 2) {
    myServo.write(angle);
    delay(50);

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    float distance = duration * 0.0343 / 2;

    Serial.print(angle);
    Serial.print(",");
    Serial.println(distance);
  }
}
```

Cod Processing — Radar Display

```
import processing.serial.*;

Serial myPort;
String data;
int angle;
float distance;

void setup() {
    size(600, 600);
    myPort = new Serial(this, Serial.list()[0], 9600);
    background(0);
}

void draw() {
    if (myPort.available() > 0) {
        data = myPort.readStringUntil(' ');
        if (data != null && data.contains(",")) {
            String[] parts = split(trim(data), ",");
            if (parts.length == 2) {
                angle = int(parts[0]);
                distance = float(parts[1]);

                float r = map(distance, 0, 100, 0, 250);
                float x = width/2 + r * cos(radians(angle));
                float y = height - r * sin(radians(angle));

                stroke(0, 255, 0);
                fill(0, 255, 0);
                ellipse(x, y, 5, 5);
            }
        }
    }
}
```

Componente

- Arduino Uno
- Senzor ultrasonic HC-SR04
- servomotor SG90
- Suport din plastic (pentru rotirea senzor)
- Processing IDE

Sarcini pentru elevi

Nivel	Sarcină
 Bazic	Afișează distanța ca text pe ecranul radarului
 Intermediar	Desenează grila și cercurile radarului
 Avansat	Codează culoarea punctelor detectate în funcție de distanță (verde/galben/roșu)
 Provocare	Adaugă o animație a liniei de baleiaj și o alertă sonoră de distanță

Idei pentru Extinderea Proiectului

- Adăugarea de efecte sonore sau alarme la detectarea obiectelor
- Crearea unei interfețe de radar complet animate
- Adăugarea mai multor senzori cu ultrasunete pentru o acoperire mai largă
- Conectarea sistemului la un modul WiFi pentru monitorizare de la distanță
- Construirea unui prototip de asistent inteligent de parcare Integrarea de LED-uri pentru a indica distanța până la obiect
- Stocarea datelor radarului pentru o analiză ulterioară
- Dezvoltarea unui robot simplu care evită obstacolele
- Combinarea sistemului cu senzori de temperatură sau de lumină pentru monitorizare multifuncțională

Proiectul 4: Ohmmetru Arduino cu afișare în Processing

Descriere

Acest proiect folosește un divizor de tensiune pentru a măsura o rezistență necunoscută. Arduino calculează rezistența folosind Legea lui Ohm și trimite rezultatul către Processing, care îl afișează în timp real.

Obiective

Students levii vor:

- înțelege principiile divizoarelor de tensiune și ale Legii lui Ohm,
- învăța cum să măsoare o rezistență necunoscută folosind o placă Arduino Uno,
- utiliza intrările analogice pentru a citi valorile de tensiune,
- calcula valorile rezistențelor pe baza tensiunilor măsurate,
- transmite datele calculate prin comunicare serială,
- vizualiza măsurătorile în timp real folosind Processing,
- dezvolta o înțelegere practică a sistemelor de măsurare electrică.

Rezultate educaționale

După finalizarea atelierului, studenții vor fi capabili să:

- explice modul în care funcționează un circuit divizor de tensiune,
- aplice Legea lui Ohm pentru a calcula o rezistență necunoscută,
- conecteze corect rezistoarele într-o configurație de divizor de tensiune,
- utilizeze intrările analogice Arduino pentru măsurarea tensiunii,
- proceseze și să transmită datele măsurate prin comunicare serială,
- creeze vizualizări de bază în timp real în Processing,
- interpreteze modul în care valorile rezistoarelor influențează tensiunea din circuit,
- îmbunătățească în mod independent proiectul prin adăugarea de funcționalități suplimentare de vizualizare sau măsurare.

Echipamente necesare

Componentă	Cantitate	Scop
Arduino Uno	1	microcontroler
Rezistor cu valoare cunoscută a rezistenței (ex, 1kΩ)	1	Referință pentru divizorul de tensiune
Rezistor cu valoare necunoscută a rezistenței	1	Componenta de măsurat

Componentă	Cantitate	Scop
Breadboard + conductori jumper	1 set	Componente de conexiune
Cablu USB	1	Conectare la computer
Processing IDE	1	Afișare

Fundamente teoretice

Divizorul de tensiune și legea lui Ohm

Un divizor de tensiune utilizează doi rezistori conectați în serie. Tensiunea la joncțiunea celor doi rezistori depinde de raportul rezistențelor:

$$V_{out} = V_{in} \times R_2 / (R_1 + R_2) \rightarrow R_2 = R_1 \times (V_{in}/V_{out} - 1)$$

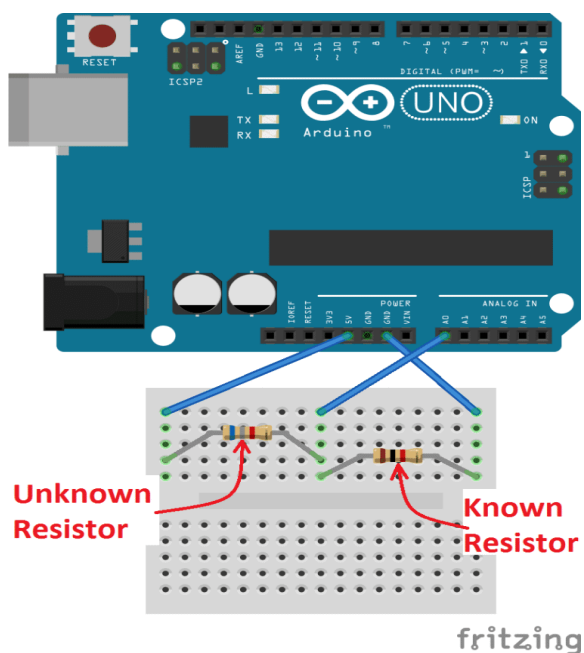
Arduino citește V_{out} prin pinul analog A0 și calculează R_2 (rezistența rezistorului necunoscut).

Pasul 1: Realizarea circuitului

Conexiuni

Conexiune	Descriere
R1 (cunoscută, ex. 1kΩ)	Între 5V și A0
R2 (necunoscut)	Între A0 și GND
A0	Joncțiunea dintre R1 și R2

💡 Cu cât R_2 este mai mare relativ la R_1 , cu atât mai mare va fi tensiunea măsurată în A0.



Pasul 2: Cod Arduino

```
vconst int analogPin = A0;
const float Vin = 5.0;
const float R1 = 1000.0; // Known resistor in ohms

void setup() {
  Serial.begin(9600);
}

void loop() {
  int raw = analogRead(analogPin);
  float Vout = (raw * Vin) / 1023.0;
  float R2 = R1 * (Vin / Vout - 1);

  Serial.println(R2); // Send resistance to Processing
  delay(500);
}
```

Pasul 3: Cod Processing

```
import processing.serial.*;

Serial myPort;
String val = "";
float resistance = 0;

void setup() {
  size(400, 200);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(255);
  fill(0);
  textSize(20);
  text("Measured Resistance:", 50, 80);
  text(nf(resistance, 1, 2) + " Ω", 50, 120);

  if (myPort.available() > 0) {
    val = myPort.readStringUntil('&apos;\n&apos;');
    if (val != null) {
      resistance = float(trim(val));
    }
  }
}
```

Modul de funcționare

- Arduino citește tensiunea de la joncțiunea divizorului de tensiune.
- Calculează rezistența necunoscută, utilizând Legea lui Ohm.
- Valoarea rezistenței este trimisă prin portul serial la Processing.
- Processing afișează valoarea în timp real.

Sarcini pentru elevi

Nivel	Task
 Bazic	Afișează valoarea rezistenței ca text
 Intermediar	Colorează textul afișat în funcție de apartenența valorii măsurate a rezistenței în intervale de mărime
 Avansat	Realizează o vizualizare de tip cadran/ceas de măsură
 Provocare	Memorează multiple măsurări și desenează un grafic

Depanare

Problemă	Cauze posibile
Măsurarea este indicată ca fiind 0 sau infinit	R2 nu este conectată sau este în scurtcircuit
Citirea nu este conformă valorii cunoscute a rezistenței R2	Valoarea R1 este introdusă incorect în cod
Valoare fluctuantă a citirii	Contact necorespunzător pe breadboard

Ideii pentru Extinderea Proiectului

- Adăugarea clasificării automate a rezistoarelor după intervalul de rezistență
- Crearea unui indicator grafic sau a unui aparat de măsură în stil analogic
- Afișarea codurilor de culori ale rezistoarelor alături de valorile măsurate
- Măsurarea rezistoarelor variabile, cum ar fi potențiometrele
- Salvarea măsurărilor de rezistență într-un fișier CSV
- Construirea unui prototip simplu de multimetru digital
- Adăugarea suportului pentru afișaje OLED sau LCD
- Crearea unui mod de calibrare pentru măsurători mai precise
- Compararea valorilor măsurate cu toleranțele nominale ale rezistoarelor

Concluzii

Acest proiect îi familiarizează pe elevi cu tehnicile de măsurare electrică, circuitele divizoare de tensiune și vizualizarea datelor în timp real. Prin combinarea platformei Arduino Uno cu mediul Processing, elevii obțin experiență practică în electronică, programare și fizică aplicată, prin intermediul unei activități STEM interactive axate pe măsurarea și analiza rezistenței electrice.

Proiectul 5: Sistem automat de irigare

Descriere

Acest proiect folosește un senzor capacitiv de umiditate a solului pentru a măsura conținutul de apă din sol. Studenții construiesc un sistem automat de irigare care monitorizează condițiile solului în timp real și controlează o pompă de apă în consecință. Datele senzorului sunt transferate către un calculator prin comunicare serială și sunt vizualizate folosind Processing.

Proiectul integrează electronica, programarea, fizica și științele mediului, servind ca o introducere practică în automatizări, sisteme IoT (Internet of Things) și control pe bază de senzori.

Obiective

Elevii vor:

- înțelege modul de funcționare al senzorilor capacitivi de umiditate a solului,
- conecta senzorul și modulul de releu la un microcontroler Arduino,
- citi și interpreta datele analogice furnizate de senzor,
- utiliza comunicarea serială pentru a transmite date către un calculator,
- vizualiza datele privind umiditatea solului în timp real folosind Processing,
- controla o pompă de apă prin intermediul unui modul de releu, atât din Arduino, cât și din Processing,
- construi un sistem de irigare complet automat, bazat pe o logică cu praguri de decizie.

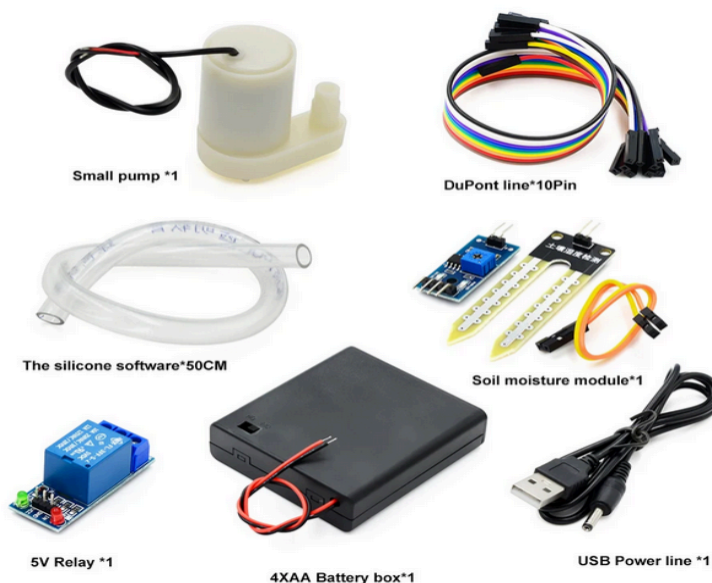
Rezultate educaționale

După atelier, elevii vor fi capabili:

- să conecteze corect un senzor capacitiv de umiditate a solului și un releu la Arduino,
- să citească și să scaleze valorile analogice ale senzorilor,
- să înțeleagă și să aplice logica de control bazată pe praguri,
- să utilizeze comunicarea serială pentru a trimite date între Arduino și Processing,
- să vizualizeze datele senzorului sub formă de text, bare și indicatori de stare în Processing,
- să trimită comenzi de control de la Processing la Arduino,
- să proiecteze un sistem automatizat complet care combină intrările senzorilor cu ieșirile actualelor.

Echipamente necesare

Componentă	Cantitate	Scop
Arduino Uno	1	microcontroler
Senzor capacitiv de umiditate a solului v1.2	1	Masurare umidității solului
Modul Releu	1	Pornire/oprire a pompei
Pompă de irigare (mini)	1	Irigare
Circuit de alimentare pompă	1	Sursă de energie electrică pentru pompă
Furtun din silicon (50 cm)	1	Distribuire a apei
Breadboard + conectori jumper	1 set	Conectare componente
Cablu USB	1	Conexiune la computer
Processing IDE	1	Vizualizare



Fundamente teoretice

Modul de funcționare a senzorului de umiditate a solului?

Senzorul capacitiv de umiditate a solului emite o tensiune analogică ce scade pe măsură ce umiditatea solului crește. Sol uscat → valoare mare (~1023). Sol ud → valoare mică (~300). Spre deosebire de senzorii rezistivi, acesta nu corodează în timp.

💡 Fizică: Solul ud are o permitivitate mai mare, modificând capacitatea și tensiunea de ieșire a circuitului senzorului.

Modulul releu

Un releu este un comutator acționat electric. Un semnal mic de 5V de la Arduino controlează un circuit de pompă cu tensiune mai mare. Când intrarea releului este HIGH, comutatorul este de obicei deschis (pompa OPRITĂ); când este LOW, comutatorul se închide (pompa PORNITĂ). Verificați întotdeauna logica activă a modului dumneavoastră specific de releu.

Comunicarea Serială

Arduino trimite datele senzorului către calculator în mod serial, prin portul USB. De exemplu:

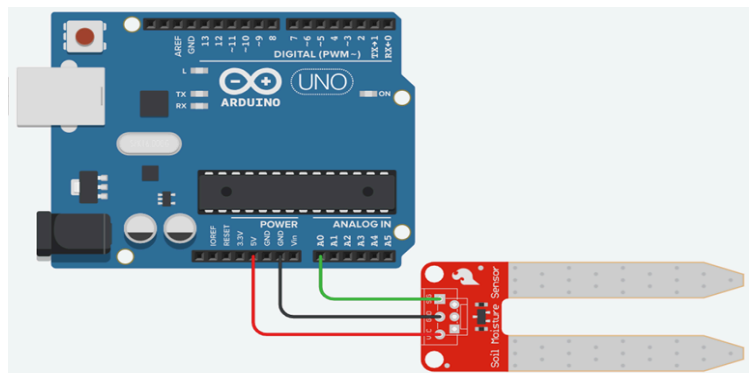
```
Serial.println(moisture);
```

Processing citește această valoare și o folosește pentru a actualiza afișajul vizual și pentru a determina starea pompei.

Exercițiul 1: Achiziția de date și vizualizare

Conexiuni — Sensor

Pin Sensor	Conectat la
VCC	3.3V sau 5V
GND	GND
AOUT	Arduino A0



Cod Arduino

```
const int sensorPin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int moisture = analogRead(sensorPin);
  // 0 (wet) to 1023 (dry)
  Serial.println(moisture);
  delay(300);
}
```

Senzorul generează o tensiune analogică care scade pe măsură ce umiditatea solului crește

Cod Processing — Afișare Text + Bare

```
import processing.serial.*;

Serial myPort;
String val = "";
int moisture = 0;

void setup() {
    size(400, 200);
    myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
    background(255);
    fill(0);
    textSize(20);
    text("Soil Moisture: " + moisture, 50, 60);

    // Visual bar
    fill(0, 100, 255);
    rect(50, 100, map(moisture, 1023, 300, 0, 300), 30);
    // Adjust range as needed
}

void serialEvent(Serial p) {
    val = p.readStringUntil('&apos;\n&apos;');
    if (val != null) {
        val = trim(val);
        if (val.matches("\\d+")) {
            moisture = int(val);
        }
    }
}
```

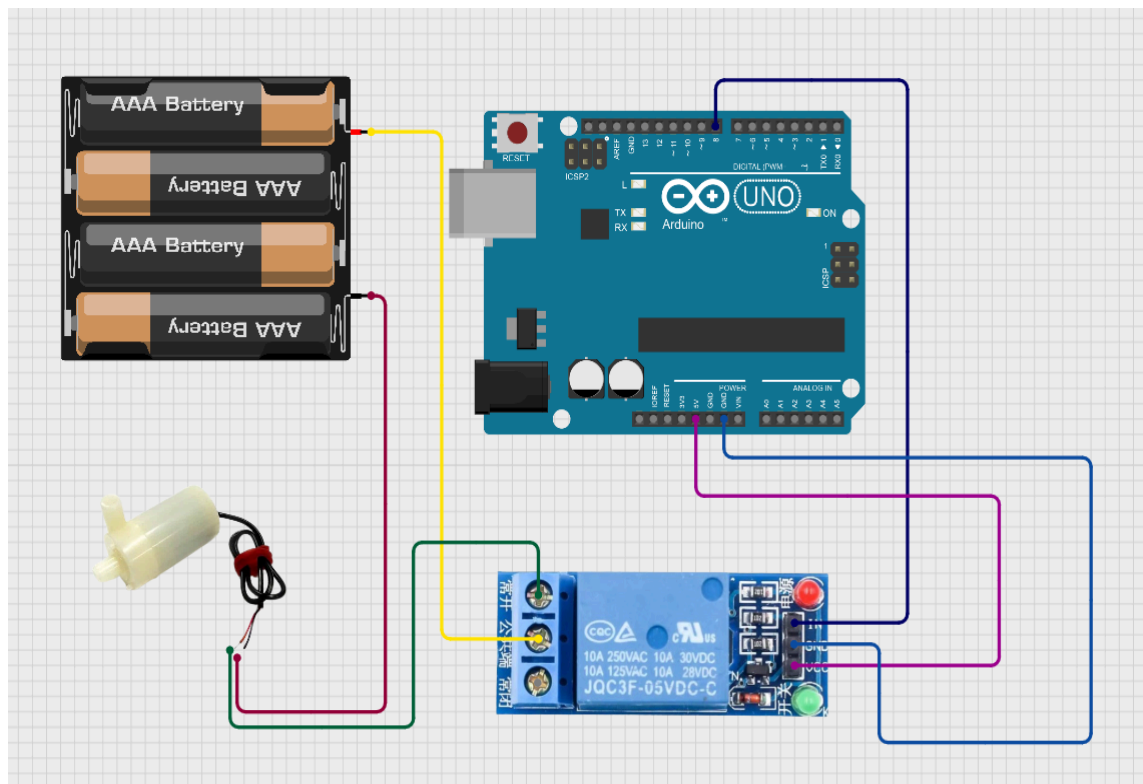
Exercițiul 2: Controlul pompei de irigare din Processing

Componente Adiționale:

- Modul Releu
- Pompă de irigare
- Sursă externă de 12V (pentru pompă)

Conexiuni — Releu și Pompă

Conecxiune	Descriere
Relay IN	Arduino D8
Relay VCC	Arduino 5V
Relay GND	Arduino GND
Relay COM	+ terminal of external power supply
Relay NO	Pump (+) terminal
Pump - terminal	Power Supply GND



Cod Arduino

```
void setup() {  
  Serial.begin(9600);  
  pinMode(9, OUTPUT);  
  digitalWrite(9, HIGH); // Ensure Pump starts OFF  
}  
  
void loop() {  
  if (Serial.available() > 0) {  
    while (Serial.available() > 1) { // Flush extra  
bytes  
      Serial.read();  
    }  
  
    char cmd = Serial.read();  
  
    if (cmd == '1') {  
      digitalWrite(9, LOW); // Pump ON  
    } else if (cmd == '0') {  
      digitalWrite(9, HIGH); // Pump OFF  
    }  
  }  
}
```

Cod Processing

```
import processing.serial.*;

Serial myPort;

void setup() {
  size(300, 100);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

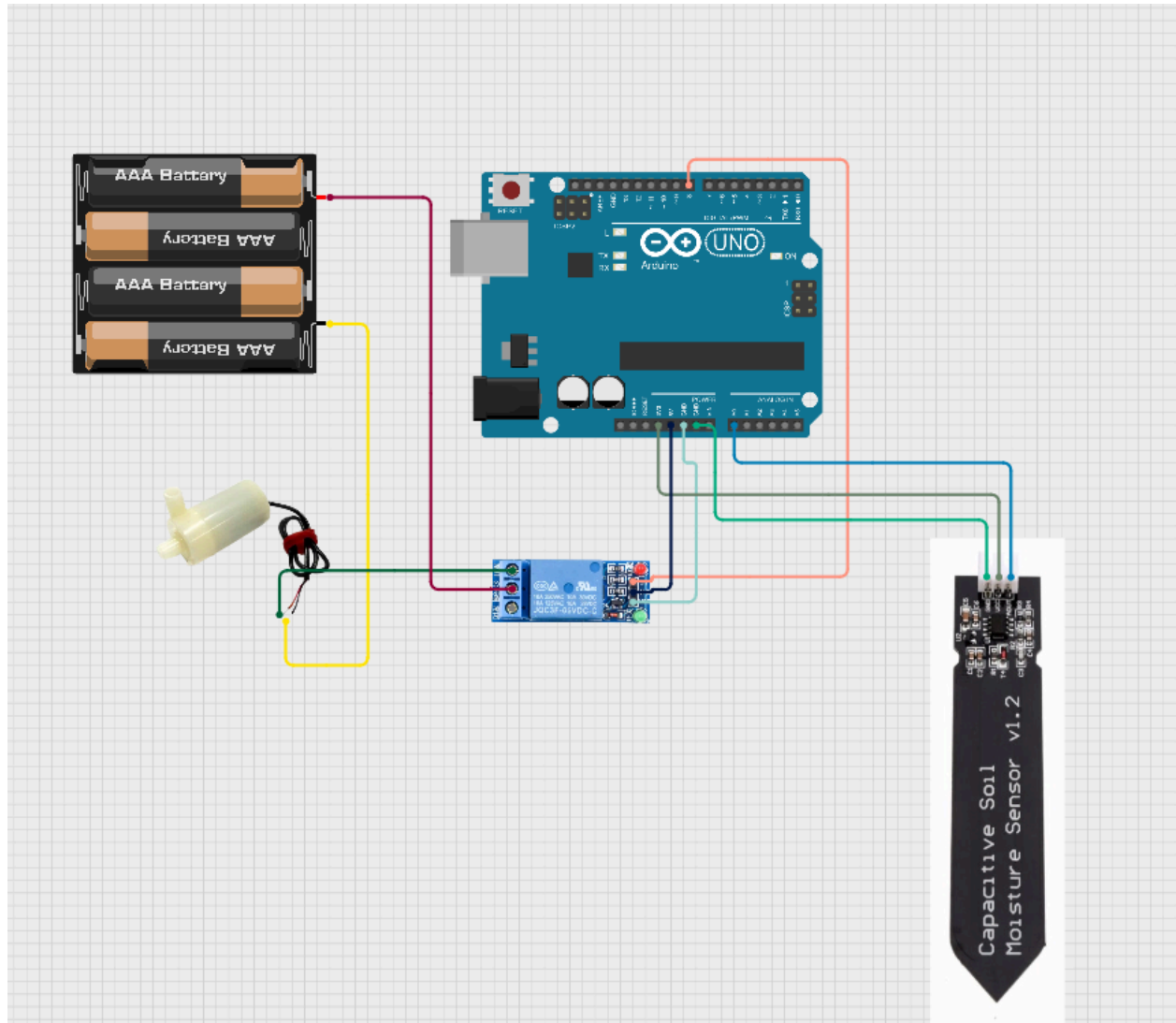
void draw() {
  background(200);
  fill(0);
  text("Press '1' to turn ON pump, '0' to turn OFF", 20, 50);
}

void keyPressed() {
  if (key == '1' || key == '0') {
    myPort.write(key);
  }
}
```

Exercițiul 3: Sistem complet automat de irigare

| 🚀 Acest exercițiu este adecvat ca proiect complet.

Scop: integrarea citirii senzorilor și controlul pompei astfel încât irigarea să aibă loc automat atunci când solul este uscat, cu afișarea în timp real prin Processing.



Cod Arduino

```
#define RELAY_PIN 8
#define SENSOR_PIN A0
#define THRESHOLD 600

void setup() {
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);
}

void loop() {
  int moisture = analogRead(SENSOR_PIN);
  Serial.println(moisture);

  if (moisture > THRESHOLD) {
    digitalWrite(RELAY_PIN, HIGH); // Dry → Water ON
  } else {
    digitalWrite(RELAY_PIN, LOW); // Wet → Water OFF
  }

  delay(1000);
}
```

Cod Processing — Afișarea statusului

```
import processing.serial.*;

Serial myPort;
String val;

void setup() {
    size(400, 200);
    myPort = new Serial(this, Serial.list()[0],
9600);
}

void draw() {
    if (myPort.available() > 0) {
        val = myPort.readStringUntil('&apos;\n&apos;');
        if (val != null) {
            int moisture = int(trim(val));
            background(255);
            fill(0);
            textSize(20);
            text("Soil Moisture: " + moisture, 50, 100);
            if (moisture > 600) {
                fill(255, 0, 0);
                text("Status: DRY - Watering", 50, 140);
            } else {
                fill(0, 150, 0);
                text("Status: WET - Idle", 50, 140);
            }
        }
    }
}
```

Sarcini pentru elevi

Nivel	Sarcina
 Bazic	Afișează nivelul umidității și statusul în format text
 Intermediar	Codează prin culoare bara de afișare a umidității (albastru = umed, roșu = uscat)
 Avansat	Adaugă un buton pentru declanșarea manuală a pompei (override button) în Processing
 Provocare	Salvează valorile umidității cu timestamps într-un fișier CSV

Depanare

Problemă	Cauze posibile
Pompa nu pornește	Conexiuni incorecte la releu sau neconcordanțe între pini. Baterii descărcate
Citiri maxime ale senzorului, permanent	Senzorul nu este plasat în sol sau este defect
Pompa funcționează încontinuu	Pragul de declanșare prea ridicat — reduceți valoarea THRESHOLD
Processing nu furnizează date	Port COM sau baud rate dezaliniat

✓ **Sfat:** Folosește `Serial.list()` pentru a găsi portul COM corect. Ajustează `Serial.list()[0]` dacă este necesar.

Aplicații în lumea reală

Sistemele de irigare automată și monitorizarea umidității solului sunt utilizate în:

- grădini inteligente și automatizarea locuinței
- agricultură și irigare de precizie
- sisteme de control al climatului în sere
- agricultură urbană și grădini verticale
- platforme IoT pentru monitorizarea mediului
- stații de cercetare pentru știința solului și a plantelor
- sisteme de economisire a apei în regiunile aride

Conexiuni transcurriculare

Materie	Conexiune
Fizică	Capacitate electrică, tensiune analogică, circuite electrice, comutare prin releu
Biologie și Ecologie	Biologia plantelor, compoziția solului, circuitul apei
Informatică	Programare, comunicare serială, procesare de date
Tehnologie și Inginerie	Electronică, integrarea senzorilor, controlul actuatorilor
Matematică	Scalarea datelor, valori de prag, interpretarea graficelor
Științele mediului	Conservarea apei, agricultură sustenabilă, climă

Competențe DigComp

Proiectul dezvoltă următoarele competențe DigComp 2.2:

Domeniul DigComp	Competențe dezvoltate
1. Informație și alfabetizare digitală	Collecting and interpreting real-time sensor data
2. Comunicare și colaborare	Prezentarea rezultatelor și lucrul în echipă
3. Creare de conținut digital	Programare în mediile Arduino și Processing
4. Siguranță	Înțelegerea utilizării în siguranță a componentelor electrice și a reelelor
5. Rezolvarea de probleme	Diagnosticarea defecțiunilor hardware și depanarea codului

Atelierul dezvoltă în mod specific:

- alfabetizarea digitală și gândirea computațională,
- rezolvarea de probleme și depanarea sistematică,
- achiziția și analiza datelor în timp real,
- programarea microcontrolerelor și a interfețelor vizuale,
- colaborarea într-un mediu practic de tip STEM.

Ideii de extindere a proiectului

Students Studenții care finalizează toate exercițiile pot extinde proiectul prin:

- Adăugarea unui afișaj OLED pentru a indica nivelul de umiditate direct pe dispozitiv
- Trimiterea datelor de irigare printr-o rețea WiFi folosind ESP8266 sau ESP32
- Crearea unui panou de control web pentru monitorizarea de la distanță a umidității solului
- Conectarea la o platformă cloud (de exemplu, ThingSpeak, Blynk) pentru înregistrarea datelor
- Construirea unui sistem de irigare pe mai multe zone pentru mai multe recipiente de plante
- Adăugarea unui grafic în timp real al umidității în Processing
- Integrarea unui API meteo pentru a omite udarea în zilele ploioase
- Adăugarea unei baterii de rezervă și a unui panou solar pentru un sistem autonom (off-grid)

Extensie	Dificultate	Noi deprinderi dobândite
OLED display	● Intermediar	comunicare I2C, librării de display
Trimitere de date prin WiFi	● Avansat	ESP32/ESP8266, programarea rețelelor
Cloud dashboard	● Avansat	Integrare de API, data visualization
Multi-zone system	● Provocare	Senzori multipli, control logic
Solar-powered	● Provocare	Management de putere electrică, design hardware

Concluzii

Acest proiect servește ca o introducere mecanică și practică în automatizări, control bazat pe senzori și vizualizarea datelor în timp real. Combinând mediile Arduino și Processing cu un senzor de umiditate a solului și un modul de releu, studenții acumulează experiență practică în electronică, programare și monitorizarea mediului.

Sistemul automat de irigații demonstrează modul în care instrumentele digitale simple pot rezolva probleme din lumea reală — de la economisirea apei în grădinile casnice până la gestionarea operațiunilor agricole la scară largă. Prin trei exerciții progresive, studenții își dezvoltă abilități care se conectează direct cu dezvoltarea IoT, ingineria automatizărilor și tehnologiile sustenabile.

Provocări de tip Hackathon și Integrarea Învățării Aplicate

Pe lângă activitățile ghidate din cadrul atelierelor, programul Digital Challenge include o serie de provocări competitive de tip hackathon, concepute pentru a încuraja creativitatea, rezolvarea independentă a problemelor, prototiparea rapidă și aplicarea practică a competențelor STEM acumulate în scenarii realiste.

Aceste provocări marchează tranziția de la exerciții de învățare structurate către sarcini deschise de inginerie și rezolvare de probleme. Elevii sunt încurajați să combine cunoștințe din electronică, fizică, programare, sisteme de senzori, comunicare serială, vizualizare și automatizare pentru a dezvolta soluții funcționale contra cronometru.

Formatul de tip hackathon promovează:

- învățarea aplicată,
- rezolvarea colaborativă a problemelor,
- gândirea de tip engineering design,
- creativitatea și inovația,
- identificarea și remedierea defecțiunilor (troubleshooting) sub presiune,
- și luarea deciziilor în timp real.

Spre deosebire de exercițiile tradiționale de la clasă, provocările simulează în mod intenționat situații ingineresti autentice în care elevii trebuie să:

- analizeze problema în mod independent,
- selecteze instrumentele hardware și software adecvate,
- proiecteze circuite,
- dezvolte algoritmi,
- testeze și să depaneze sistemele,
- și să prezinte prototipuri funcționale.

Sarcina fiecărei provocări este direct conectată la conceptele dezvoltate pe parcursul seriei de ateliere.

Tematici ale atelierelor	Provocări conexe în hackathon
Divizoare de tensiune și măsurători analogice	Măsurarea unei rezistențe necunoscute
LED-uri și măsurători electrice	Dimensionarea tensiunii de deschidere a LED-ului
Senzori de lumină și procesarea semnalelor	Comunicare optică prin codul Morse
Sisteme de monitorizare a temperaturii	Provocarea detectării extratereștrilor prin măsurarea temperaturii
Senzori cu ultrasunete și vizualizare	Senzori de parcare și sisteme radar
Vizualizare în timp real în Processing	Toate sarcinile bazate pe Processing
Automatizări și sisteme cu praguri de decizie	Logica de control pentru irigații și parcare

Structura progresivă a provocărilor permite elevilor să evolueze de la:

- măsurători simple
- către sisteme interactive
- în final, la proiecte complet autonome și bazate pe comunicare.

O atenție deosebită este acordată:

- stabilității sistemului
- organizării codului
- fiabilității măsurărilor
- clarității vizualizării
- demonstrării cu succes a prototipurilor funcționale.

Scenariile ficționale de tip science-fiction utilizate în cadrul mai multor sarcini (invazie extraterestră, roboți invizibili, comunicare securizată prin lumină) sunt concepute intenționat pentru a spori implicarea și motivația studenților, necesitând totodată aplicarea unor principii științifice și ingineresti reale.

Prin intermediul acestor activități, studenții își dezvoltă competențe practice legate de:

- integrarea senzorilor
- programarea sistemelor embedded
- comunicarea serială
- prelucrarea semnalelor
- Vizualizare
- Automatizare
- interacțiunea om-mașină.

Componenta de hackathon consolidează, de asemenea:

- lucrul în echipă
- gestionarea timpului
- abilitățile de prezentare
- reziliența în procesul de depanare
- fluxurile de lucru ingineresti colaborative.

Evaluarea în timpul activităților de tip hackathon se bazează nu doar pe obținerea unor rezultate tehnice corecte, ci și pe:

- calitatea designului
- funcționalitate
- creativitate
- eficiență
- structura codului
- stabilitate
- capacitatea de a explica și de a demonstra soluția dezvoltată.

Această abordare susține învățarea autentică de tip STEM prin conectarea conceptelor teoretice cu experimentarea practică, prototiparea rapidă și practicile ingineresti din lumea reală, într-un mediu de proiect extrem de motivant.

Provocări pentru Hackathon – Ziua 1

Provocarea 1

Sarcina ta este să determini valorile rezistențelor necunoscute ale rezistoarelor albastru, roșu și verde utilizând software-ul instalat, sistemele de dezvoltare și componentele disponibile la stația ta de lucru.

10 puncte în total

- **2 puncte** — circuit conectat corect
- **2 puncte** — software corespunzător
- **3 puncte** — rezistorul albastru determinat corect R1= _____
- **1.5 puncte** — rezistorul roșu determinat corect R2= _____
- **1.5 puncte** — rezistorul verde determinat corect R3= _____

Provocarea 2

Folosind componentele, sistemele de dezvoltare și software-ul instalat disponibile, determinați tensiunea de deschidere (turn-on voltage) a diodelor LED roșie / galbenă / verde (tensiunea la bornele diodei la care aceasta începe să emită lumină). Curentul prin LED nu trebuie să depășească 50 mA.

12 puncte în total

- **4 puncte** — circuit realizat fără a depăși 50 mA
- **3 puncte** — metodă corectă de măsurare
- **2 puncte** — rezultatul pentru LED-ul roșu: U1= _____
- **1 punct** — rezultatul pentru LED-ul galben: U2= _____
- **1 punct** — rezultatul pentru LED-ul verde: U3= _____
- **1 punct** — explicație / prezentarea măsurătorii

Provocarea 3

În viitorul apropiat, are loc o invazie extraterestră. Singurul canal de comunicare pe care invadatorii nu îl pot controla este comunicarea pe bază de lumină. Sarcina ta este să trimiți un apel general, CQ, în codul Morse către oricine îți poate vedea semnalul. Curentul optim pentru funcționarea LED-ului utilizat la transmiterea semnalului este de 15 mA. Un punct durează 200 ms. Folosind software-ul instalat, sistemele de dezvoltare și componentele disponibile la stația ta de lucru, rezolvă această problemă.

Total: 15 puncte

- **3 puncte** — calculul corect al rezistorului

- **2 puncte** — circuit conectat corect
- **4 puncte** — codul Morse corect pentru litera C
- **4 puncte** — codul Morse corect pentru litera Q
- **2 puncte** — spațiere corectă / repetarea semnalului

Provocarea 4

Extraterestrii dețin roboți invizibili care pot fi detectați doar prin măsurarea temperaturii dintr-o încăpere. Dacă temperatura, fără o cauză externă vizibilă, depășește 30°C, există posibilitatea ca un robot invizibil să se afle în apropiere. Măsoară temperatura din sala de derularea a hackathonului, folosind componentele disponibile, sistemele de dezvoltare și software-ul instalat.

Cerințe

- Sistemul trebuie să funcționeze independent de un calculator și să ofere un semnal luminos atunci când temperatura depășește 30°C.
- Afișează valorile măsurate pe un calculator folosind Processing.

a) 10 puncte	b) 10 puncte
• 2 puncte — senzor conectat corect • 3 puncte — citire precisă a temperaturii • 3 puncte — răspuns corect la 30°C • 2 puncte — funcționare stabilă fără calculator	• 3 puncte — transmisie de date în serie (sau <i>transmisie serială de date</i>) • 3 puncte — afișarea corectă a valorilor • 2 puncte — actualizări în timp real • 2 puncte — claritatea afișajului

Provocarea 5

Din cauza aglomerației din trafic, este necesară parcare a vehiculelor cât mai aproape unele de altele. Prin urmare, sunt necesari senzori de parcare care să indice distanța dintre un vehicul și un obstacol. Sarcina ta este să proiectezi un senzor de parcare simplu folosind resursele disponibile, care să afișeze apropierea vehiculului prin intermediul unui LED verde și al unui LED roșu.

LED-ul verde luminează la luminozitate maximă atâta timp cât distanța dintre vehicul și obstacol este mai mare de 15 cm. Luminozitatea acestuia scade pe măsură ce vehiculul se apropie de obstacol. Pentru distanțe mai mici de 10 cm, LED-ul roșu se aprinde, iar luminozitatea sa crește până la nivelul maxim pe măsură ce distanța scade la 5 cm. Pentru distanțe mai mici de 5 cm, LED-ul roșu începe să lumineze intermitent (să clipească) rapid.

Cerințe

- Sistemul trebuie să funcționeze independent de un calculator.
- Proiectează propriul tău mod de afișare a distanței și prezintă-l pe un calculator folosind Processing.

a) 15 puncte • 3 puncte — măsurarea corectă a distanței • 3 puncte — LED-ul verde funcționează pentru >15 cm • 3 puncte — intensitatea LED-ului verde scade pe măsură ce obstacolul se apropie • 3 puncte — intensitatea LED-ului roșu crește în intervalul 10–5 cm • 3 puncte — LED-ul roșu clipește rapid pentru <5 cm	b) 10 puncte • 4 puncte — afișarea distanței în Processing • 3 puncte — vizualizare logică și clară • 3 puncte — originalitatea modului de afișare
--	--

Provocare Specială

Singurul canal de comunicare securizat în timpul unei invazii extraterestre este lumina. Codul Morse și primele cinci litere ale alfabetului sunt folosite pentru a trimite mesaje codificate prin semnale luminoase. Un punct în codul Morse durează 200 ms. Folosind software-ul instalat, sistemele de dezvoltare și componentele disponibile la stația ta de lucru, creează un dispozitiv prototip care va afișa mesajul luminos primit. Pentru decodificare, folosește tabelul:

Cod	Simbol	Cod	Simbol	Cod	Simbol
AAA	A	ADA	M	BAC	Y
AAB	B	ADB	N	BAD	Z
AAC	C	ADC	O	EAA	0
AAE	D	ADD	P	EAB	1
ABA	E	ADE	Q	EAC	2
ABC	F	AEA	R	EAD	3
ABD	G	AEB	S	EAE	4
ABE	H	AEC	T	EBA	5
ACA	I	AED	U	EBB	6
ACB	J	AEE	V	EBC	7
ACD	K	BAA	W	EBD	8
ACE	L	BAB	X	EBE	9

50 de puncte în total

- **4 puncte** — detectarea corectă a impulsurilor luminoase
- **4 puncte** — diferențierea corespunzătoare între semnalele scurte și cele lungi
- **10 puncte** — recunoașterea corectă a literelor
- **10 puncte** — recunoașterea corectă a codurilor din trei litere
- **5 puncte** — stabilitatea sistemului
- **7 puncte** — organizarea programului
- **10 puncte** — demonstrația reușită pe un exemplu de test

Provocările Hackathon-ului – Ziua 2

Provocarea 1: Măsurarea distanței cu ESP32 și HC-SR04

Obiectiv: Conectarea unui senzor cu ultrasunete la placa ESP32, măsurarea distanței și vizualizarea datelor.

Punctaj total: 35 puncte

Sarcina 1 — Desenează schema circuitului [5 pct]

Folosind Circuit Designer (<https://app.circuitdesigner.com/>), desenează schema electrică de conectare a plăcii ESP32 la senzorul cu ultrasunete HC-SR04.

- Etichetează clar toți pinii: VCC, GND, TRIG, ECHO.
- Exportă schema sau fă o captură de ecran și adaug-o în folderul pentru predare.

💡 *Placa ESP32 funcționează cu semnal logic de 3.3 V; pinul ECHO al senzorului HC-SR04 scoate un semnal de 5 V — folosește un divizor de tensiune sau un adaptor de nivel logic (level shifter) dacă este necesar.*

Sarcina 2 — Asamblarea hardware [5 pct]

Conectează fizic componentele ESP32 și HC-SR04 pe o placă de testare (breadboard), respectând schema desenată.

- Verifică atent conexiunile VCC (5 V sau 3.3 V) și GND înainte de a alimenta placa.
- Evaluatorul va inspecta conexiunile de pe breadboard.

Sarcina 3 — Programul în Arduino IDE [10 pct]

Scrie un program C++ (sketch Arduino) care:

- Trimite un impuls de declanșare (trigger) de 10 μ s pe pinul TRIG.
- Măsoară durata ecoului pe pinul ECHO folosind funcția pulseIn().
- Calculează distanța în centimetri (viteza sunetului = 0.034 cm/ μ s).
- Afișează distanța în Serial Monitor la fiecare 500 ms.

Testează la cel puțin două distanțe diferite și compară rezultatele cu o riglă.

Sarcina 4 — Verificarea funcționării [5 pct]

Deschide Serial Monitor (setat la viteza de 9600 baud). Fă o captură de ecran care să arate:

- Cel puțin 5 citiri consecutive.
- Două valori de distanță clar diferite.

Sarcina 5 — Vizualizarea în Processing [10 pct]

Scrie un program în Processing (modul Java) care:

- Citește valorile distanței primite prin portul serial.
- Afișează pe ecran valoarea numerică a distanței.
- Desenează un dreptunghi plin a cărui lățime este proporțională cu distanța măsurată.

💡 *Folosește biblioteca Serial din Processing. Asigură-te că viteza de transmisie (baud rate) se potrivește cu cea din programul Arduino (9600).*

Provocarea 2: Controlul unui servomotor în funcție de distanță

Obiectiv: Utilizarea citirii de la senzorul ultrasonic pentru a controla poziția unui servomotor.

Punctaj total: 30 puncte

Sarcina 1 — Desenarea circuitului [5 pct]

În Cirkuit Designer, desenează conexiunile dintre plăcuța ESP32, servomotorul SG90 și senzorul HC-SR04.

- Etichetează clar toate conexiunile.
- Evidențiază conexiunea pinului de semnal al servomotorului.

Sarcina 2 — Montajul hardware [10 pct]

Conectează ESP32, servomotorul SG90 și senzorul HC-SR04 pe un breadboard.

- Alimentează servomotorul dintr-o sursă externă de 5 V (sau USB), NU direct dintr-un pin GPIO al plăcii ESP32.
- Verifică funcționarea stabilă a circuitului înainte de a trece la programare.

💡 *Servomotorul SG90 poate consuma până la 250 mA — folosește o sursă de alimentare externă pentru a evita repornirea (resetarea) plăcii ESP32.*

Sarcina 3 — Programul în Arduino IDE [10 pct]

Scrie un program C++ care:

- Citește continuu distanța furnizată de HC-SR04.
- Rotește servomotorul la 0° (stânga) când distanța este mai mică de 10 cm.
- Rotește servomotorul la 180° (dreapta) când distanța este mai mare de 20 cm.
- Menține servomotorul la 90° (centru) pentru distanțe cuprinse între 10 și 20 cm.
- Afișează distanța și unghiul servomotorului în Serial Monitor.

Sarcina 4 — Verificare [5 pct]

Demonstrează că servomotorul reacționează corect la cele trei intervale de distanță. Evaluatorul va ține un obiect la diferite distanțe în timpul evaluării.

Provocarea 3: Scanner Sonar cu Vizualizare în Processing

Obiectiv: Construiește un sistem sonar rotativ și redă un afișaj în stil radar în Processing.

Punctaj total: 50 puncte

Sarcina 1 — Desenarea Schemei Electrice [5 pt]

În Cirkuit Designer, desenează conexiunea completă (ESP32 + SG90 + HC-SR04).

- Senzorul și servomotorul trebuie să fie reprezentate montate împreună pe un suport.

Sarcina 2 — Asamblarea Hardware [10 pt]

Asamblează fizic unitatea sonar:

- Montează senzorul HC-SR04 pe brațul servomotorului SG90, astfel încât să se poată roti stânga/dreapta.
- Conectează cablurile de alimentare și semnal fără a restricționa rotația.

Sarcina 3 — Programul în Arduino IDE [15 pt]

Scrie un program C++ care:

- Rotește servomotorul de la 0° la 180° și înapoi, în pași de 1°.
- Măsoară distanța la fiecare unghi.
- Trimite datele prin interfața Serială în formatul: UNGHI,DISTANȚĂ\n
- Include o scurtă întârziere (≤ 30 ms per pas) pentru a permite stabilizarea citirilor senzorului.

Sarcina 4 — Verificarea [5 pt]

Deschide Serial Monitor. Confirmă că valorile unghiului variază între 0 și 180, iar valorile distanței sunt plauzibile.

Sarcina 5 — Vizualizarea în Processing [15 pt]

Scrie un cod în Processing care:

- Citește liniile de tip UNGHI,DISTANȚĂ de pe portul serial.
- Redă un afișaj în stil radar: o linie verde de scanare pe un fundal negru.
- Plotează un punct sau un arc la distanța măsurată pentru fiecare unghi.
- Adaugă etichete pentru unghi și distanță în jurul afișajului.

💡 *Folosește conversia din coordonate polare în coordonate carteziane: $x = \text{dist} * \cos(\text{unghi})$, $y = \text{dist} * \sin(\text{unghi})$.*

Provocarea 4 (Avansată): Sonar Wi-Fi cu procesare pe Android

Obiectiv: Transmiterea datelor de la sonar prin Wi-Fi (UDP) și vizualizarea acestora într-o aplicație Android realizată în Processing.

Punctaj total: 30 puncte | Provocare bonus — încercați-o doar după finalizarea Provocărilor 1–3.

Sarcina 1 — Programul în Arduino IDE [15 pt]

Scrieți un program C++ care:

- Conectează placa ESP32 la o rețea Wi-Fi cunoscută (SSID-ul și parola vor fi definite în cod).
- Roteste servomotorul continuu între 0° și 180° și măsoară distanța la fiecare pas.
- Trimite datele sub formă de pachet UDP cu formatul UNGHI,DISTANȚĂ către IP-ul și portul specificate ale dispozitivului Android.
- Repetă baleierea (măturarea) în mod continuu.

💡 *Indiciu: Folosiți librăria WiFiUDP pe ESP32. Asigurați-vă că placa ESP32 și dispozitivul Android sunt conectate la aceeași rețea.*

Sarcina 2 — Vizualizarea în Processing pentru Android [15 pt]

Scrieți un sketch în Processing (modul Android) care:

- Ascultă pachetele UDP primite pe portul specificat.
- Extrage valorile UNGHI și DISTANȚĂ din fiecare pachet primitor.
- Afișează un ecran complet în stil radar, cu o linie de baleiere și puncte corespunzătoare distanței.
- Rulează fluent la cel puțin 20 fps pe un dispozitiv Android.

💡 *Indiciu: Folosiți librăria UDP ketai sau processing.net pentru modul Android. Testați pe un dispozitiv real dacă este posibil.*

Cadrul pedagogic și metodologic al proiectului Digital Challenge

Introducere

Colecția de ateliere *Digital Challenge* este concepută ca un program educațional STEM practic, ce combină electronica, programarea, fizica, ingineria, vizualizarea datelor și rezolvarea de probleme din lumea reală prin activități de învățare bazate pe proiecte.

Atelierele se bazează pe experimentare practică folosind microcontrolere, senzori, actuatori și instrumente de vizualizare, permițându-le elevilor să își dezvolte atât competențe tehnice, cât și transversale, prin participare activă și muncă în echipă.

Abordarea educațională folosită promovează:

- învățarea prin practică (*learning by doing*)
- învățarea bazată pe investigație
- experimentarea
- gândirea computațională
- rezolvarea interdisciplinară a problemelor
- creativitatea și inovația

Materialele sunt adecvate pentru o mare diversitate de contexte educaționale :

- educația formală
- activități STEM extracurriculare
- spații de tip *makerspace*
- hackathoane
- cluburi de robotică
- medii de învățare non-formale
- activități de formare a cadrelor didactice

Filosofia educațională

Colecția de ateliere se bazează pe ideea că elevii învață cel mai eficient atunci când proiectează, construiesc, testează, analizează și îmbunătățesc în mod activ sisteme reale.

În loc să se concentreze doar pe explicații teoretice, elevii:

- interacționează cu dispozitive fizice
- colectează și interpretează date din lumea reală
- dezvoltă prototipuri funcționale
- vizualizează informații
- rezolvă probleme autentice de inginerie

Metodologia încurajează:

- experimentarea
- proiectarea iterativă
- depanarea (*troubleshooting*)
- munca în echipă
- învățarea reflexivă

O atenție deosebită este acordată conectării conceptelor științifice cu aplicațiile practice din viața de zi cu zi și tehnologia modernă.

Abordarea metodologică

Atelierele combină mai multe abordări educaționale complementare:

Învățarea colaborativă și între egali (Peer Learning)

Atelierele sunt concepute pentru a susține:

- munca în echipă
- rezolvarea colaborativă a problemelor
- mentoratul între egali
- schimbul de cunoștințe
- prezentarea rezultatelor proiectelor

Elevii sunt încurajați să discute soluții, să compare abordări și să depaneze împreună sistemele hardware și software.

Învățarea bazată pe proiecte (PBL)

Elevii învață prin dezvoltarea unor proiecte funcționale complete, cum ar fi:

- sisteme de monitorizare a mediului
- simulări de radar
- sisteme inteligente de irigații
- sisteme automatizate cu senzori
- interfețe vizuale interactive

Proiectele evoluează de la funcționalități de bază către extinderi mai avansate și provocări deschise (*open-ended*).

Învățarea bazată pe investigație

Elevii sunt încurajați să:

- exploreze comportamentul senzorilor
- testeze ipoteze
- compare măsurători
- calibreze sisteme
- identifice erori
- îmbunătățească performanța prin experimentare

Atelierele promovează gândirea științifică și raționamentul bazat pe dovezi.

Învățarea bazată pe provocări (Challenge-Based Learning)

Fiecare atelier include:

- exerciții cu dificultate progresivă
- sarcini practice
- situații de depanare
- provocări avansate

Această structură permite elevilor cu niveluri diferite de experiență să participe cu succes, oferind totodată oportunități pentru o explorare mai aprofundată.

Integrarea STEM

Colecția de ateliere integrează multiple discipline STEM într-un cadru practic unitar.

Disciplină	Examples
Fizică	Propagarea sunetului, tensiune, rezistență, capacitate, senzori
Electronică	Circuite, divizoare de tensiune, relee, actuatoare
Informatică	Programare, comunicare serială, procesarea datelor
Inginerie	Integrarea sistemelor, automatizare, prototipare
Matematică	Analiza datelor, scalare, reprezentare grafică
Științele mediului	Sisteme de irigații, sustenabilitate, monitorizarea mediului

Structura interdisciplinară îi ajută pe elevi să înțeleagă modul în care conceptele științifice și tehnologice interacționează în sistemele din lumea reală.

DigComp și competențele digitale

Atelierele susțin dezvoltarea competențelor aliniate cu cadrul **DigComp 2.2**, inclusiv:

DigComp Area	Competencies Developed
Alfabetizare digitală și gestionarea datelor	Colectarea, interpretarea și vizualizarea datelor de la senzori
Comunicare și colaborare	Munca în echipă și prezentarea rezultatelor proiectelor
Creare de conținut digital	Programarea microcontrolere-lor și a interfețelor vizuale
Siguranță	Manipularea în siguranță a componentelor și sistemelor electronice
Rezolvarea de probleme	Depanarea sistemelor hardware și software

În plus, atelierele dezvoltă:

- gândirea computațională
- raționamentul algoritmic
- alfabetizarea digitală
- creativitatea
- gândirea critică
- abilitățile de proiectare în inginerie

Competențele secolului XXI

Activitățile din cadrul atelierelor promovează:

- gândirea critică
- creativitatea
- colaborarea
- comunicarea
- adaptabilitatea
- inițiativa
- rezolvarea de probleme
- alfabetizarea tehnologică

Elevii își dezvoltă atât competențe tehnice, cât și abilități transversale relevante pentru educația și carierele viitoare în domeniul STEM.

Metodologia de evaluare

Evaluarea poate include:

- implementarea hardware reușită
- funcționalitatea codului
- procesul de rezolvare a problemelor
- capacitatea de depanare
- interpretarea datelor
- creativitatea soluțiilor
- munca în echipă și colaborarea
- prezentarea proiectului
- calitatea documentației

Atelierele susțin:

- **evaluarea formativă** în timpul activităților
- **evaluarea sumativă** prin proiecte finale, demonstrații și competiții

Incluziune și accesibilitate

Structura atelierelor susține educația STEM incluzivă prin:

- niveluri de dificultate progresive
- sarcini de la nivel începător la avansat
- muncă colaborativă
- sprijin între egali (*peer support*)
- abordări vizuale ale învățării
- activități practice direct pe echipament

Materialele sunt concepute pentru a încuraja participarea elevilor cu niveluri diferite de experiență anterioară și pentru a susține o implicare mai largă în domeniile STEM.

O atenție deosebită poate fi acordată:

- încurajării participării fetelor la proiecte din domenii STEM
- susținerii grupurilor subreprezentate
- adaptării activităților pentru diferite contexte educaționale

Ghid de siguranță

Elevii trebuie să:

- deconecteze alimentarea înainte de a modifica circuitele
- evite scurtcircuiturile
- țină lichidele departe de componentele electronice
- manipuleze cu atenție senzorii și actuatoarele
- verifice conexiunile (*cablajul*) înainte de a alimenta sistemele

Proiectele care implică relee, pompe sau surse de alimentare externe necesită supraveghere suplimentară și o atenție sporită la siguranța electrică.

Reflecție și evaluare

Sunt incluse activități de reflecție pentru a încuraja o înțelegere mai profundă a:

- comportamentului sistemului,
- deciziilor de inginerie,
- proceselor de depanare,
- aplicațiilor din lumea reală.

Exemple de întrebări pentru reflecție:

- Care a fost cea mai mare provocare tehnică?
- Cum ar putea fi îmbunătățit sistemul?
- Ce problemă din lumea reală rezolvă acest proiect?
- Cât de sigure/precise sunt măsurătorile?
- Ce senzori sau funcționalități suplimentare ar putea fi adăugate?

Relevanța în lumea reală

Colecția de ateliere demonstrează modul în care sistemele digitale simple pot fi aplicate în:

- monitorizarea mediului
- agricultura inteligentă
- automatizări
- robotică
- case inteligente (*smart homes*)
- sisteme IoT
- monitorizare industrială
- vizualizare interactivă

Elevii acumulează experiență cu tehnologii și concepte conectate direct cu ingineria modernă și sistemele digitale.

Replicare și adaptare

Atelierele sunt concepute pentru a fi:

- accesibile financiar, utilizând materiale și echipamente cu costuri reduse
- modulare
- scalabile
- replicabile

Acestea pot fi adaptate pentru:

- ateliere scurte
- proiecte semestriale
- hackathoane
- tabere STEM
- predare la clasă
- medii de învățare online sau hibride

Majoritatea activităților utilizează componente hardware ușor accesibile pe piață și instrumente software cu sursă deschisă (*open-source*).