***Proiect Sistem Incorporate***

***Student: Profesor coordonator:***

***MURTAZA MIHAI Enescu Nicolae***

***Cerinta:***

Sistem gonire rozatoare si pasari (buton, lcd, difuzor, senzor lumina) - afiseaza starea curenta a senzorului la un interval de 2 secunde (ca o medie de 5 citiri consecutive). Daca luminozitatea este sub 15 se porneste o alarma la fiecare 500ms cu pauza de 100ms cu o frecventa peste 20kHz (ultrasunete). Cand luminozitatea creste peste valoarea 15 se porneste o alarma la fiecare 300ms cu pauza de 100ms cu o frecventa de 15kHz alternata de o alarma la fiecare 500ms cu pauza de 100ms cu o frecventa peste 20kHz. Sistemul porneste si se opreste la apasarea butonului. Cand e oprit sistemul, nu se citesc senzorii, difuzorul si lcd-ul sunt oprite.

***Cuprins:***

1. ***Componentele necesare rezolvarii proiectului.***
2. ***Descrierea circuitului.***
3. ***Solutie software. Codul sursa.Explicatii.***
4. ***Despre Arduino.***
5. ***Bibliografie.***

1. ***Componente necesare rezolvarii proiectului:***

***Placuta ARDUINO***

O plăcuță Arduino este compusă dintr-un microcontroler [Atmel](https://ro.wikipedia.org/w/index.php?title=Atmel&action=edit&redlink=1) [AVR](https://ro.wikipedia.org/w/index.php?title=AVR&action=edit&redlink=1) de 8-, 16- sau 32-biți (deși începând cu 2015 s-au folosit microcontrolere de la alți producători) cu componente complementare care facilitează programarea și încorporarea în alte circuite. Un aspect important la Arduino este că acesta dispune de conectori standard, care permit utilizatorului să conecteze plăcuța cu procesorul la diferite module interschimbabile numite shield-uri. Unele shield-uri comunică cu Arduino direct prin pinii digitali sau analogici, dar altele sunt adresabile individual prin magistrala serială [I²C](https://ro.wikipedia.org/wiki/I%C2%B2C) permițând utilizarea mai multor module în paralel. Până în anul 2015 plăcuțele Arduino oficiale au folosit cipuri Atmel din seria megaAVR, în special ATmega8, ATmega168, ATmega328, ATmega1280 și ATmega2560, iar în 2015 au fost adăugate cipuri de la alți producători. O multitudine de alte procesoare au fost folosite de dispozitive compatibile Arduino. Multe plăcuțe includ un regulator liniar de 5 V și un oscilator cu cuarț de 16 MHz (sau un rezonator ceramic în unele variante), deși anumite plăcuțe, cum ar fi LilyPad, funcționează la 8 MHz și nu necesită regulator, datorită restricțiilor de formă. Un microcontroler instalat pe Arduino vine preprogramat cu un [bootloader](https://ro.wikipedia.org/w/index.php?title=Bootloader&action=edit&redlink=1) care simplifică încărcarea programelor pe memoria flash a cipului, în comparație cu alte dispozitive care necesită programatoare externe. Acest aspect face Arduino o soluție simplă, permițând programarea de pe orice computer ordinar. În prezent, bootloader-ul optiboot este bootloader-ul implicit instalat pe Arduino UNO.[[7]](https://ro.wikipedia.org/wiki/Arduino#cite_note-7)

La nivel conceptual, când se folosește mediul de dezvoltare integrat Arduino, programarea tuturor plăcuțelor se face prin conexiune serială. Implementarea acesteia diferă în funcție de versiunea hardware. Unele plăcuțe Arduino au implementate convertoare de nivel logic pentru a realiza conversia între nivelele logice RS-232 și cele TTL. Plăcuțele Arduino din prezent sunt programate prin USB, având integrate cipuri de conversie USB-serial, cum ar fi FTDI FT232. Unele modele UNO, mai noi, folosesc un cip AVR separat programat să funcționeze ca un convertor USB-serial, care poate fi reprogramat printr-un port ICSP dedicat. Alte variante, cum ar fi Arduino Mini și versiunea neoficială Boarduino, folosesc adaptoare detașabile USB-serial, cabluri, Bluetooth sau alte metode.

Plăcuța Arduino are expuși mulți dintre pinii de intrare/ieșire ai microcontrolerului, pentru ca aceștia să fie folosiți de alte circuite. Diecimila, Duemilanove și UNO oferă 14 pini digitali de intrare/ieșire, dintre care 6 pot produce semnale PWM și 6 intrări analogice care, de asemenea, pot fi folosite ca intrări/ieșiri digitale. Acești pini sunt accesibili prin partea superioară a plăcuței, prin intermediul unor barete mamă cu pasul între pini de 2,54 mm.

***Buton***

Butonul unui dispozitiv electronic, de obicei, funcționează ca un întrerupător  electric, adică are în interior două contacte, fiind pe unul, se va activa funcția inversă in momentul acesta care este realizat, în cazul în care acesta este un dispozitiv (normal deschis) va fi închis, în cazul unui dispozitiv de NI (normal închis) va fi deschis

"Butonul" a fost folosit în [calculatoare](https://ro.wikipedia.org/wiki/Calculator), [telefoane](https://ro.wikipedia.org/wiki/Telefon), [aparate](https://ro.wikipedia.org/w/index.php?title=Aparate&action=edit&redlink=1), și diverse alte dispozitive mecanice și electronice, acasă și comercial.

În aplicații industriale și comerciale, butoanele pot fi legate între ele printr-o legătură mecanică în care actul de pulsare a unui buton de așa natură încât celălalt buton să nu se mai presioneze. În acest fel, un buton de oprire poate "forța" un buton de pornire pentru a fi lansat. Această metodă de legare este utilizat în operațiuni manuale simple, în care aparatul sau procesul nu conțin [circuite electrice](https://ro.wikipedia.org/wiki/Circuit_electric) pentru control.

***LCD 16 x 2***

**Afișajul cu cristale lichide** ([engleză](https://ro.wikipedia.org/wiki/Limba_englez%C4%83) *Liquid Crystal Display*, prescurtat *LCD*) este un [dispozitiv](https://ro.wikipedia.org/w/index.php?title=Dispozitiv&action=edit&redlink=1) de afișare pentru litere, cifre, grafică și imagini, fiind constituit dintr-o matrice de celule lichide care devin opace sau își schimbă culoarea sub influența unui curent sau câmp electric.[[1]](https://ro.wikipedia.org/wiki/Afi%C8%99aj_cu_cristale_lichide#cite_note-1) Din p.d.v. [fizic](https://ro.wikipedia.org/wiki/Fizic%C4%83) [fenomenul](https://ro.wikipedia.org/wiki/Fenomen) se explică prin proprietatea cristalelor lichide de a influența direcția de polarizare a luminii atunci când ele sunt puse sub o anumită tensiune electrică. Afișajele cu cristale nu produc ele însele [lumină](https://ro.wikipedia.org/wiki/Lumin%C4%83), și au un consum de energie foarte mic. Un afișaj LCD se prezintă sub forma unui [ecran](https://ro.wikipedia.org/wiki/Ecran) afișor (display) care este comandat electronic printr-un [decodificator](https://ro.wikipedia.org/w/index.php?title=Decodificator&action=edit&redlink=1) de caractere numerice și alfabetice. Este folosit frecvent în construcția [ceasurilor](https://ro.wikipedia.org/wiki/Ceas) digitale (ceasuri care au în locul [acelor arătătoare](https://ro.wikipedia.org/w/index.php?title=Ac_ar%C4%83t%C4%83tor&action=edit&redlink=1) un afișor de tip LCD), la afișările de date la mașini CNC, mașini de uz casnic, inscripții și semnalizări electronice.

Cristalele lichide sunt, în cazul de față, [combinații](https://ro.wikipedia.org/w/index.php?title=Combina%C8%9Bie&action=edit&redlink=1) chimice de natură organică aflate în stare lichidă. Ele au proprietatea de a putea fi comandate de o corespunzătoare [tensiune electrică](https://ro.wikipedia.org/wiki/Tensiune_electric%C4%83), astfel încât își ordonează [moleculele](https://ro.wikipedia.org/wiki/Molecul%C4%83) trecând de la stare „transparentă” la stare „netransparentă”. Concret, este vorba de o [polarizare](https://ro.wikipedia.org/wiki/Polarizare) electrică a unor molecule lichide care în contrast cu restul „câmpului” formează o imagine vizibilă.

***Difuzor***

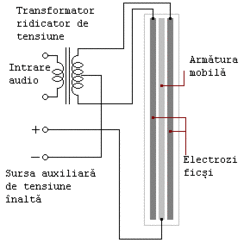
**Difuzorul** este un dispozitiv în care [energia electrică](https://ro.wikipedia.org/wiki/Electricitate) de audiofrecvență de la ieșirea receptorului radio, TV sau amplificatorului se transformă în sunet. Această transformare se face prin mai multe sisteme și anume:*electromagnetic, electrodinamic, piezoelectric și electrostatic*.

Constructiv, difuzorul are o parte fixă, *carcasa*, care susține partea *mobilă*. Descriu mai jos cum funcționează fiecare sistem de difuzor.

Difuzorul bazat pe acest sistem este o construcție simplă care se bazează pe *acțiunea câmpului magnetic alternativ*. Se compune dintr-un [magnet](https://ro.wikipedia.org/wiki/Magnet) permanent care între cei doi poli are montată o bobină prin care circulă un curent de audiofrecvență emis de amplificator. Prin mijlocul bobine se găsește montată, circulând liber, o lamelă elastică de oțel care la un capăt este cuplată de o membrană în formă de pâlnie (poate pentru anumite cerințe să fie și plană). Datorită curentului de audiofrecvență și sub acțiunea câmpului magnetic alternativ, lamela vibrează antrenând membrana difuzorului emițând sunete. Cu toate că acest tip de difuzor are o sensibilitate bună, caracteristica de redare a frecvențelor este foarte proastă și cu distorsiuni mari. Nu se mai folosește, el a fost difuzorul începuturilor.

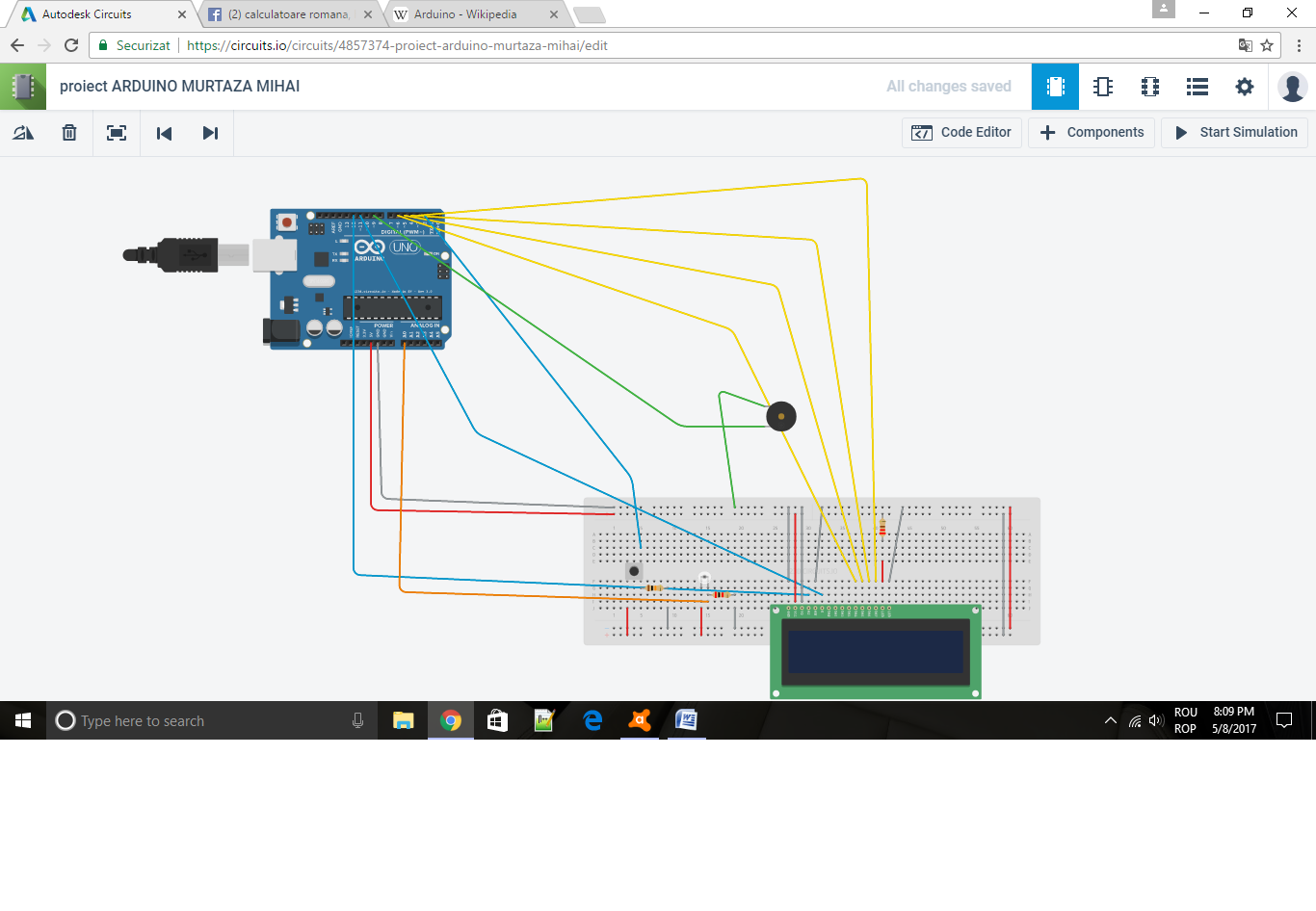
Sistemul piezolectric

Principiul de funcționare: pe armăturile elementului piezoelectric se aplică tensiunea alternativă de audiofrecvență. Elementul începe să oscileze mecanic în aceeași frecvență. Oscilațiile sunt transmise membranei fixate rigid de elementul piezoelectric, care produce vibrații sonore. Acest tip de difuzoare se pretează pentru redarea frecvențelor audio superioare peste 8000 Hz. Puterea lor este oarecum limitată de rezistența mecanică a elementului piezoelectric.

******

***Senzor de Lumina***

***2)Descrierea circuitului:***

**

***3) Solutie Software:***

Programele Arduino pot fi scrise în orice [limbaj de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) cu un [compilator](https://ro.wikipedia.org/wiki/Compilator) capabil să producă un [cod mașină](https://ro.wikipedia.org/w/index.php?title=Cod_ma%C8%99in%C4%83&action=edit&redlink=1) binar. Atmel oferă un mediu de dezvoltare pentru microcontrolerele sale, AVR Studio și mai nou, Atmel Studio.[[15]](https://ro.wikipedia.org/wiki/Arduino#cite_note-AutoF7-38-16)[[16]](https://ro.wikipedia.org/wiki/Arduino#cite_note-AutoF7-39-17)

Proiectul Arduino oferă un mediu integrat de dezvoltare (IDE), care este o aplicație [cross-platform](https://ro.wikipedia.org/wiki/Cross-platform), scrisă în [Java](https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare)). Acesta își are originile în mediul de dezvoltare pentru limbajul de programare Processing și în proiectul Wiring. Este proiectat pentru a introduce programarea în lumea artiștilor și a celor nefamiliarizați cu dezvoltarea software. Include un editor de cod cu funcții ca evidențierea sintaxelor, potrivirea acoladelor și spațierea automată și oferă mecanisme simple cu un singur click, pentru a compila și a încărca programele în plăcuța Arduino. Un program scris în IDE pentru Arduino se numește *sketch*.[[17]](https://ro.wikipedia.org/wiki/Arduino#cite_note-18)

Arduino IDE suportă limbajele de programare [C](https://ro.wikipedia.org/wiki/C_(limbaj_de_programare)) și [C++](https://ro.wikipedia.org/wiki/C%2B%2B) folosind reguli speciale de organizare a codului. Arduino IDE oferă o librărie software numită Wiring, din proiectul Wiring, care oferă multe proceduri comune de intrare și ieșire. Un sketch tipic Arduino scris în C/C++ este compus din două funcții care sunt compilate și legate cu un ciot de program ***main()***, într-un program executabil cu o execuție ciclică:

* *setup()*: o funcție care este rulată o singură dată la începutul programului, când se inițializează setările.
* *loop()*: o funcție apelată în mod repetat până la oprirea alimentării cu energie a plăcuței.

După compilarea și legarea cu GNU toolchain inclus, de asemenea, în IDE, mediul de dezvoltare Arduino trimite comandă către programul avrdude pentru a converti codul executabil într-un fișier text codat hexazecimal, care poate fi încărcat în placa Arduino de un program de încărcare.

Un program Arduino tipic pentru un programator începător face ca un LED să se aprindă intermitent. Acest program este încărcat pe placă, în mod normal, de către producător. În mediul de dezvoltare Arduino, utilizatorul ar trebui să scrie un astfel de program după cum urmează:[[18]](https://ro.wikipedia.org/wiki/Arduino#cite_note-Blink_Tutorial-19)

Multe plăcuțe Arduino conțin un LED, împreună cu un rezistor în serie, între pinul 13 și masă (GND), ceea ce este un amănunt util pentru multe teste.[[18]](https://ro.wikipedia.org/wiki/Arduino#cite_note-Blink_Tutorial-19)

***Codul Sursa:***

***#include <LiquidCrystal.h>***

***LiquidCrystal lcd(12, 11, 6,5, 4, 3);***

***const int sensorReadings = 5;***

***int readings[sensorReadings]; // citirile analogice***

***int readIndex = 0; //indexul citirii curente***

***int total = 0; //variabila in care facem suma citirilor***

***int average = 0; // variabila in care salvam media citirilor pt a o afisa pe LCD***

***int lightSensorPin = A0;***

***int btn=2;***

***int buzzer=9;***

***int i=0;***

***int scroll = 0; //variabila pt afisarea de la stanga la dreapta circular pe LCD***

***//variabilele pt citirea pushbutonului***

***int buttonState = 0;***

***int old\_buttonState = 0;***

***int stare = 0;***

***void setup()***

***{***

***lcd.begin(16, 2);***

***pinMode(btn, INPUT); //setam pinul pushbutonului ca fiind INPUT***

***pinMode(buzzer, OUTPUT); //setam buzzerul ca OUTPUT***

***pinMode(lightSensorPin,INPUT); //setam pinul analogic A0senzorului de lumina ca OUTPUT***

***}***

***void loop(){***

***if(stare % 2 == 0){ // verificam daca butonul nu s-a apasat sau este in starea de oprire a sistemului***

***noTone(buzzer);***

***lcd.clear();***

***lcd.noDisplay();***

***}***

***buttonState = digitalRead(btn);//citim valoarea pushbutonului***

***if((buttonState == HIGH)&&(old\_buttonState == LOW)) { //verificam daca sa apasat butonul***

***stare++;***

***}***

***old\_buttonState = buttonState; //verificam care a fost ultima stare a butonului***

***if(stare % 2 != 0){***

***for(i=0; i < sensorReadings; i++){ //citim de 5 ori valoare senzorului de lumina***

***readings[readIndex] = analogRead(lightSensorPin); // facem citirea de pe pinul A0 pe care avem senzorul de lumina***

***total = total + readings[readIndex]; // adunam la total ultima valoare citita***

***readIndex = readIndex + 1; //trecem la urmatoare citire***

***average = total / sensorReadings; // media a 5 citiri consecutive***

***delay(400); // o citire la 0,4 secunde pentru a avea 5 citiri la 2 secunde***

***}***

***lcd.display();***

***lcd.clear();***

***lcd.setCursor(scroll, 0);***

***lcd.print(average); //afisam valoare media a 5 citiri ale senzorului pe LCD***

***scroll++;***

***if(scroll > 13){ //daca scroll ajunge la 13 o resetam pt a nu depasii fomatul de afisare al LCD-ului***

***scroll = 0;***

***}***

***sonerie(); // apelam functia sonerie pt pornirea alarmei potrivite***

***total=0; //resetam totalul pt o noua serie de citiri***

***}***

***}***

***void sonerie(){***

***if(average<15){***

***tone(buzzer ,22000, 500); // Daca luminozitatea este sub 15 se porneste o alarma la fiecare 500ms cu pauza de 100ms cu o frecventa peste 20kHz (ultrasunete)***

***delay(100);***

***}***

***if(average>=15){***

***tone(buzzer ,15000, 300);***

***delay(100); //Cand luminozitatea creste peste valoarea 15 se porneste o alarma la fiecare 300ms cu pauza de 100ms cu o frecventa de 15kHz***

***// alternata de o alarma la fiecare 500ms cu pauza de 100ms cu o frecventa peste 20kHz.***

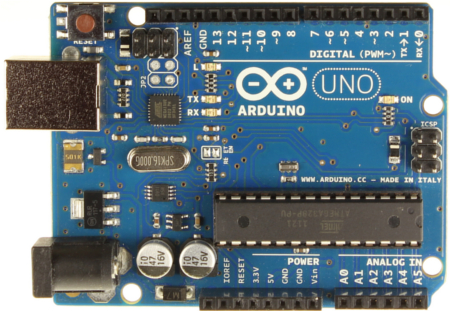
***tone(buzzer ,22000, 500);***

***delay(100);***

***}***

***}***

***4) Despre Arduino***

******

Începând de sus, există 14 pini digitali de intrare / ieşire (I/O - input/output). Aceştia operează la o tensiune de 5 volţi şi pot fi controlaţi cu una din funcţiile pinMode(), digitalWrite() şi digitalRead(). Fiecare pin poate primii sau trimite o intensitate de maxim 40 mA şi au o rezistenţă internă între 20-50 kOhmi (default deconectată). În afară de semnalul standard I/O, unii dintre pini mai au şi alte funcţii specializate, care sunt descrise mai jos:

0. (serial) RX - pin serial, utilizat în special pentru recepţia (intrare - **Rx**) datelor seriale asincrone ([asynchronous serial communication](http://en.wikipedia.org/wiki/Asynchronous_serial_communication)) Protocolul serial asincron este o metodă foarte răspândită în electronică pentru a trimite şi recepţiona date între dispozitive. Acest protocol este implementat în dispozitiv numit [UART](http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter) (Universal Asynchronous Receiver/Transmitter)

**1.** (serial) TX - pin serial, utilizat pentru trimiterea datelor asincrone (ieşire - **Tx**). [TTL](http://en.wikipedia.org/wiki/Transistor-transistor_logic) vine de la transistor-transistor logic.

**2.** (External Interrupts) întrerupere externă. Acest pin poate fi configurat pentru a declanșa o întrerupere la o valoare mică, un front crescător sau descrescător, sau o schimbare în valoare. Vezi detalii despre posibile comenzi la [attachInterrupt()](http://arduino.cc/en/Reference/AttachInterrupt)

**3.** (External Interrupts + PWM) întrerupere externă. Identic cu pinul 2. Suplimentar, toţi pinii marcaţi cu semnul ~ pot fi folosiţi şi pentru **PWM** ([pulse with modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation))

**4.** (I/O) pin standard intrare/iesire

**5.** (PWM) poate furniza control de ieşire pe 8-bit pentru controlul PWM. Vezi detalii despre posibile comenzi la [analogWrite()](http://arduino.cc/en/Reference/AnalogWrite)

**6.** (PWM)

**7.** (I/O) pin standard intrare/ieşire

**8.** (I/O) pin standard intrare/ieşire

**9.** (PWM)

**10.** (PWM + SPI) - suportă comunicare prin interfaţa serială ([Serial Peripheral Interface](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus)). SPI-ul are patru semnale logice specifice iar acest pin se foloseste pentru SS - Slave Select (active low; output din master). Pinii SPI pot fi controlaţi folosind [libraria SPI](http://arduino.cc/en/Reference/SPI).

**11.** (PWM + SPI) - suportă SPI, iar acest pin se foloseşte pentru MOSI/SIMO - Master Output, Slave Input (output din master)

**12.** (SPI) - suportă SPI, iar acest pin se foloseşte pentru MISO/SOMI - Master Input, Slave Output (output din slave)

**13.** (LED + SPI) - suportă SPI, iar acest pin se foloseşte pentru SCK/SCLK - Ceas serial (output din master). De asemenea, pe placă este încorporat un LED care este conectat la acest pin. Când pinul este setat pe valoarea HIGH este pornit, când are valoarea LOW este oprit.

**14.** (GND) - împământare. Aici se pune negativul.

**15.** (AREF) - Analog REFference pin - este utilizat pentru tensiunea de referinţă pentru intrările analogice. Se poate controla folosind funcţia [analogReference()](http://arduino.cc/en/Reference/AnalogReference).

**16.** (SDA) - comunicare I2S

**17.** (SCL) - comunicare I2S

Jos, există o serie de 6 pini pentru semnal analogic, numerotaţi de la A0 la A5, fiecare din ei poate furniza o rezoluţie de 10 biţi (adică maxim 1024 de valori diferite). În mod implicit se măsoară de la 0 la 5 volţi, deşi este posibil să se schimbe limita superioară a intervalului lor folosind pinul 15 AREF şi funcţia analogReference(). De asemenea, şi aici anumiţi pini au funcţii suplimentare descrise mai jos:

**0.** standard analog pin

**1.** standard analog pin

**2.** standard analog pin

**3.** standard analog pin

**4.** (SDA) suportă comunicarea prin 2 fire (**I2S** (I-two-C) sau **TWI** (Two wire interface)). Acest pin este folosit pentru SDA (Serial Data) la TWI.

**5.** (SCL) identic cu pinul 4, doar că acest pin este folosit pentru **SCL** (Serial Clock) la TWI. Pentru controlul TWI se poate folosi [librăria Wire](http://arduino.cc/en/Reference/Wire).

Lângă pinii analogici de jos mai există o secţiune de pini notată POWER. Mai jos sunt prezentaţi începând de lângă pinul analog A0:

* Vin - intrarea pentru tensiune din sursă externă (input Voltage)
* GND - negativul pentru tensiune din sursă externă (ground Voltage)
* GND - negativ. Se foloseşte pentru piesele şi componentele montate la arduino ca şi masă/împământare/negativ.
* 5V - ieşire pentru piesele şi componentele montate la arduino. Scoate fix 5V dacă placa este alimentată cu tensiune corectă (între 7 şi 12 v)
* 3.3V - ieşire pentru piesele şi senzorii care se alimentează la această tensiune. Tensiunea de ieşire este 3.3 volţi şi maxim 50 mA.
* RESET - se poate seta acest pin pe LOW pentru a reseta controlerul de la Arduino. Este de obicei folosit de shield-urile care au un buton de reset şi care anulează de obicei butonul de reset de pe placa Arduino.
* IOREF - este folosit de unele shield-uri ca referinţă pentru a se comuta automat la tensiunea furnizată de placa arduino (5 volţi sau 3.3 volţi) (Input/Output Refference Voltage)
* pin neconectat, este rezervat pentru utilizări ulterioare (la reviziile următoare ale plăcii probabil).

Comunicarea cu calculatorul, altă placă arduino sau alte microcontrolere se poate realiza fie prin portul USB (şi este văzut ca un port standard serial COMx), fie prin pinii 0 şi 1 (RX şi TX) care facilitează comunicarea serială **UART TTL** (5V). Folosind [librăria SoftwareSerial](http://www.arduino.cc/en/Reference/SoftwareSerial) se poate face comunicaţii seriale folosind oricare din pinii digitali. Pentru comunicarea **I2C** (**TWI**) este inclusă o [librărie Wire](http://arduino.cc/en/Reference/Wire). Pentru comunicarea **SPI** se poate folosi [librăria SPI](http://arduino.cc/en/Reference/SPI).

După cum vedeţi în imaginea de mai sus, placa mai are o serie de pini marcaţi **ICSP** ([In-Circuit Serial Programming](http://en.wikipedia.org/wiki/In_Circuit_Serial_Programming_(ICSP))). Aceşti pini pot fi folosiţi pentru [reprogramarea microcontrolerului](http://www.instructables.com/id/Using-an-Arduino-board-as-an-ISP-via-the-ICSP-head/), sau ca pini de expansiune cu alte microcontrolere compatibile. Sunt conectaţi standard şi se poate folosi un cablu de 6 fire (MOSI, MISO, SCK, VCC, GND, şi pinul RESET) la fel ca în imagina de pe wikipedia (vezi linkul de la ICSP de mai sus).

**Intrare / Iesire la sistemele Arduino**

• Pini de intrare/iesire digitali, conectati la porturile microcontrollerului

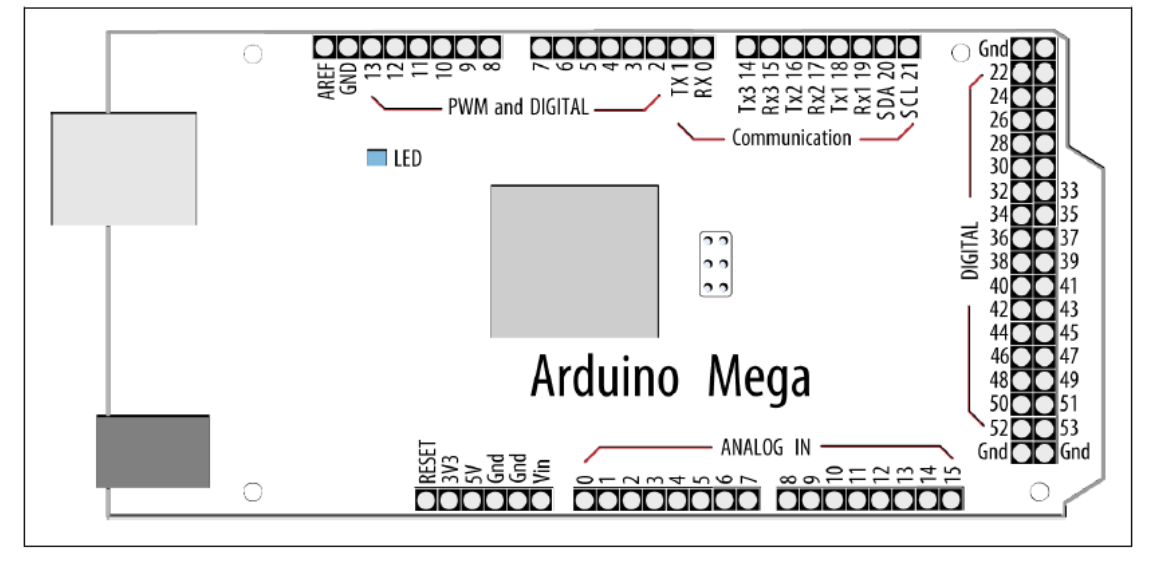
• Mediul de dezvoltare se ocupa de problema corespondentei

• Logica de programare este orientata pe numarul pin-ului

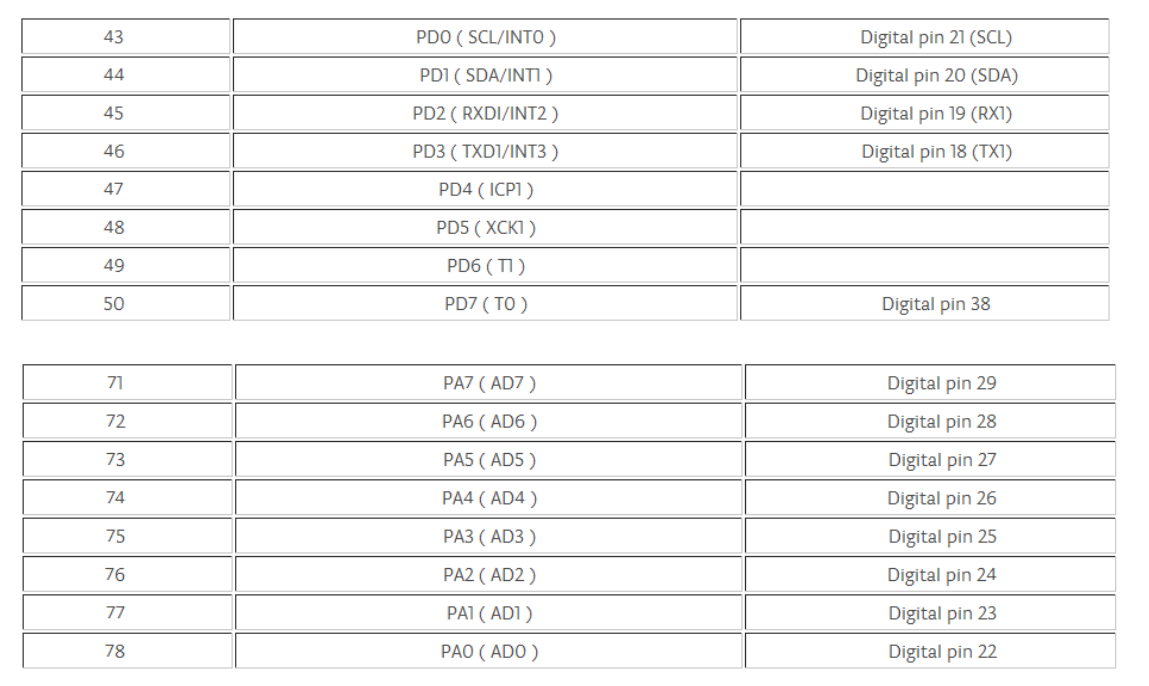
• O parte din pini au functii speciale (comunicatie seriala UART sau I2C, generator de unda PWM, sau semnale analogice)

• Pinii care au functia RX si TX trebuie evitati! Sunt rezervati pentru comunicarea seriala prin USB, care include programarea placii

• De obicei exista un LED pe placa, conectat la pin-ul 13

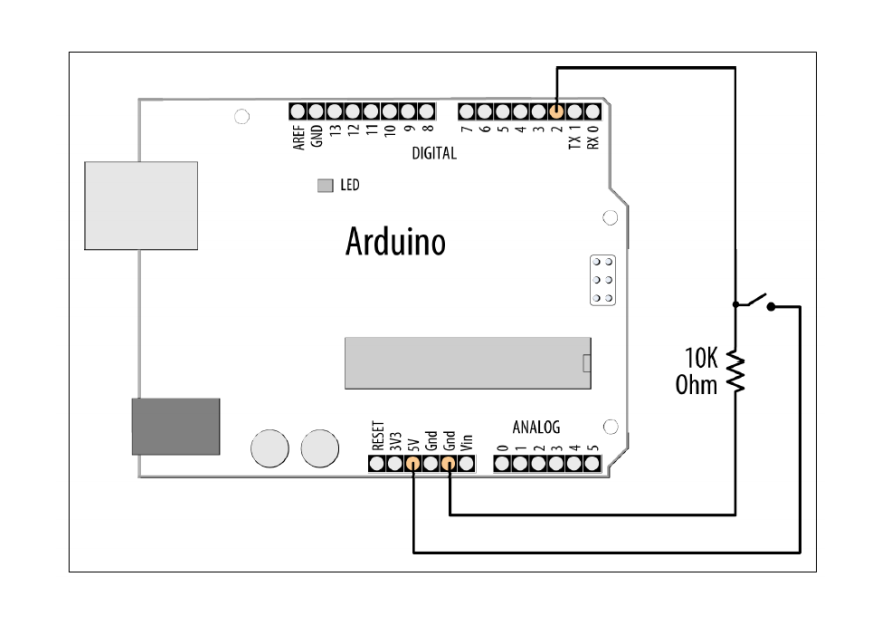


* Corespondenta pinilor cu porturile microcontrollerului ATMega2560
* Selectie:



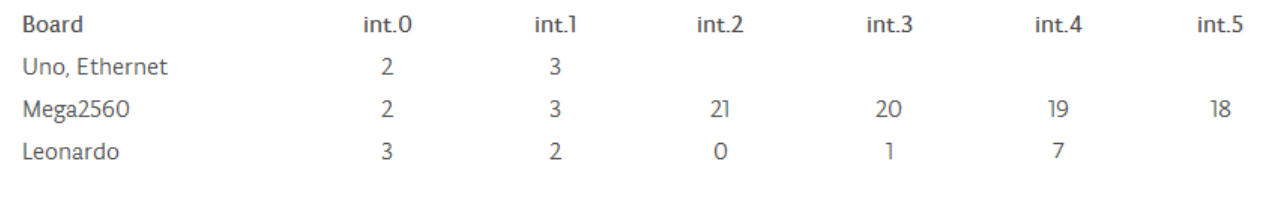
Sursa elementara de semnal: un buton conectat la un pin de intrare digital.

Se foloseste o rezistenta “pull down”, pentru ca atunci cand butonul nu este apasat, semnalul de intrare sa fie nivel logic ‘0’.

Pentru iesire, se foloseste led-ul de pe placa.

**Tratarea intreruperilor externe**

• Detectarea unor evenimente pe pini, fara a verifica in permanenta starea acestora prin digitalRead

• In functie de placa Arduino folosita, numarul intreruperilor externe disponibile este variabil:

Pentru utilizarea unei intreruperi, acestei intreruperi trebuie sa i se ataseze o procedura de tratare (Interrupt Service Routine - ISR). In acest scop, se foloseste functia attachInterrupt(), cu sintaxa: attachInterrupt(interrupt, ISR, mode)

interrupt - numarul intreruperii externe (0, 1, 2, …)

ISR – numele procedurii de tratare a intreruperii (o procedura din program)

mode – modul de declansare:

LOW – declansare pe nivel ‘0’

CHANGE – declansare cand se schimba nivelul pinului

RISING – declansare pe front crescator

FALLING – declansare pe front descrescator

• Dezactivarea tratarii unei intreruperi se face prin apelul functiei detachInterrupt(), cu sintaxa:

detachInterrupt(interrupt)

interrupt - numarul intreruperii

• Daca se doreste dezactivarea temporara a tuturor intreruperilor, se apeleaza functia noInterrupts(), fara parametri. Pentru re-activarea intreruperilor, se apeleaza functia interrupts().

• Intreruperile sunt active implicit! Dezactivarea lor trebuie facuta doar pentru perioade scurte de timp, in caz contrar alte functii arduino pot fi afectate.

***Atentie:***

• Toate variabilele care se pot modifica intr-o functie de tip ISR trebuie sa fie declarate ca “volatile”. Astfel, compilatorul va sti ca aceste variabile se pot modifica in orice moment, si nu va optimiza codul prin atasarea acestora la registri, ci le va stoca intotdeauna in RAM

• Doar o procedura ISR poate rula la un moment dat, celelalte intreruperi fiind dezactivate in acest timp

• Deoarece delay() si millis() folosesc la randul lor intreruperi, ele nu vor functiona in timp ce o procedura de tip ISR este in executie.

• Pentru intarzieri scurte intr-o procedura ISR, se poate folosi functia delayMicroseconds(), care nu foloseste intreruperi.

• Nu se recomanda scrierea datelor prin interfata seriala intr-o procedura ISR

***5)Bibliografie:***

[***https://www.arduino.cc/***](https://www.arduino.cc/)

[***https://circuits.io/***](https://circuits.io/)

[***https://ro.wikipedia.org/wiki/Arduino***](https://ro.wikipedia.org/wiki/Arduino)

[***https://en.wikipedia.org/wiki/Buzzer***](https://en.wikipedia.org/wiki/Buzzer)

[***https://www.youtube.com/watch?v=z9PPfE8dvGs***](https://www.youtube.com/watch?v=z9PPfE8dvGs)