Electronică și Robotică: Arduino, de la 0 și 1 la infinit.

de Cătălin Cazan-Gheorghiu

EDITURA CREATOR
BRAȘOV, 2023

Cuprins

1. Să facem cunoștință cu Arduino...

2. Semnale de intrare și de ieșire, analogice și digitale.

3. Afișaje.

PREFAȚĂ

Despre acest proiect

Am scris această carte cu scopul de a explora împreună lumea fascinantă a dispozitivelor electronice, fără de care viața noastră ar fi mult mai complicată și mult mai anostă. Iar cea mai bună variantă pentru a studia modalitatea în care funcționează dispozitivele electronice este să utilizăm platforma de dezvoltare Arduino care ne permite să ne conectăm la aproximativ orice tip de senzor pentru a prelua informații din mediul înconjurător: temperatură, umiditate, intensitate a sunetului, intensitate a luminii, presiune atmosferică, tensiune electrică, apăsarea unui buton de către un utilizator, deplasarea cursorului unui potențiometru pe care să le procesăm, utilizând un program creat de noi și apoi să trimitem o comandă către un alt dispozitiv: să prezentăm informații pe un afișaj, să aprindem sau să stingem un bec sau un LED, să acționăm un releu, să modificăm viteza de rotație a axului unui motor electric, să modificăm intensitatea luminii emise de un LED și așa mai departe. Toate acestea sunt exact elementele dispozitivelor electrice și electronice, casnice sau industriale, pe care le utilizăm acasă sau la serviciu. Mai mult decât atât, vom studia modalitatea în care putem programa telefonul mobil astfel încât să poată trimite comenzi dispozitivului Arduino sau să recepționeze informații de la senzorii conectați la acesta.

Evident că numărul proiectelor pe care le putem dezvolta este atât de mare încât nu putea fi cuprins într-un singur volum și probabil că nu am avea suficientă hârtie să descriem toate proiectele ce pot fi realizate utilizând platforma de dezvoltare Arduino, deoarece numărul acestora depinde doar de imaginația voastră (de aici provine utilizarea termenului infinit din titlul cărții). Drept urmare, vom prezenta câteva proiecte mai simple în acest volum, pe care, așa cum veți observa pe parcurs, le putem combina sau modifica pentru a realiza altele mai complexe și, după ce ne vom însuși câteva noțiuni generale, vom încerca, în volumele următoare să realizăm proiecte din ce în ce mai complexe și, implicit, mai interesante.

Cum este structurată cartea

Am organizat această carte în jurul activităților practice astfel încât să descoperim noțiunile teoretice pe măsură ce efectuăm experimente. Iar orice proiect care integrează dispozitivul Arduino conține două componente esențiale: circuitul electric și programarea platformei de dezvoltare, mai precis, a microcontrolerului conținut de aceasta. Am abordat programarea din două perspective: Prima

variantă a fost utilizarea blocurilor funcționale, similare unor piese de puzzle pe care le combinăm pentru a realiza programul nostru, și, așa cum veți observa pe parcursul cărții, acest sistem de programare este foarte util pentru începători deoarece lucrează cu elemente predefinite pe care trebuie doar să le personalizăm, adică să modificăm anumiți parametri, și să le combinăm într-o structură ce poate realiza funcția pe care ne-am propus-o. A doua variantă o reprezintă modalitatea tradițională de programare, sub formă de text care însă presupune să urmăm un anumit protocol: să cunoaștem structura unui program, tipul variabilelor, anumite funcții, să respectăm introducerea unor semne de punctuație care delimitează comenzile și așa mai departe, deoarece, altfel, codul sursă nu va fi compilat, adică nu va putea fi interpretat de către aplicația Arduino IDE și nu va fi transformat în program. Chiar dacă pare mai eficientă, prima variantă este limitată de blocurile funcționale care au fost definite de creatorii aplicației pe care o utilizăm, în cazul de față, Mixly, și nu poate rezolva chiar orice situație. În plus, programarea text oferă avantajul bibliotecilor de funcții, care conțin o serie de funcții predefinite pentru diverse dispozitive ce pot fi conectate la Arduino. Și, pe măsură ce realizăm mai multe proiecte, vom reutiliza o mare parte dintre funcții ceea ce înseamnă că tot procesul de programare a platformei de dezvoltare va dura din ce în ce mai puțin chiar dacă proiectele vor fi mai complexe (evident că, la un moment dat, după ce ne vom fi familiarizat cu întreaga procedură și vom fi realizat un număr suficient de mare de proiecte, nu vom rescrie toate liniile de cod pentru fiecare proiect în parte, adică nu vom reinventa roata, ci le vom refolosi pe cele din proiectele anterioare).

Fiecare dintre activitățile practice conține informații generale despre proiect, detalii despre componentele electronice, senzorii sau modulele pe care le utilizăm pentru prima dată, schema electrică, o descriere a bibliotecilor utilizate, împreună cu lista funcțiilor relevante pentru proiectul nostru, un algoritm sau o strategie de dezvoltare a codului sursă, codul sursă pentru Arduino IDE sau modalitatea de programare cu blocuri funcționale utilizând aplicația Mixly și una dintre variantele de aranjare pe breadboard (suportul pe care îl vom utiliza pentru a conecta între ele toate elementele noastre) a componentelor circuitului, realizată cu aplicația Fritzing. Pentru fiecare dintre blocurile funcționale și funcțiile codului sursa am oferit câte o scurtă descriere în scopul utilizării mai facile a acestora.

Ultima parte este dedicată aplicațiilor mobile: am descris modalitatea în care poate fi utilizată conexiunea Bluetooth a telefonului mobil pentru a transmite comenzi către dispozitivul Arduino care să determine activarea sau dezactivarea unor ieșiri de semnal sau pentru a recepționa informații de la senzorii conectați la dispozitivul Arduino.

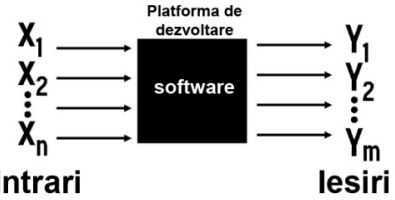
1

Să facem cunoștință cu Arduino...

În acest capitol vom încerca să ne familiarizăm cu platforma de dezvoltare Arduino, aflând detalii despre structura acesteia și modalitatea în care poate fi programată conform unor specificații.

Având în vedere că acest volum este dedicat platformei de dezvoltare Arduino, vom analiza fiecare element de circuit din punct de vedere al conectării la această platformă de dezvoltare, ceea ce înseamnă că, pentru fiecare element nou vor fi prezentate informații despre rolul acestuia în circuit, tensiunea de alimentare, semnalele de ieșire pe care le generează senzorii și modalitatea în care Arduino le poate interpreta sau semnalele de intrare pe care le utilizează elementele de acționare și modalitatea în care Arduino le poate genera, precum și alte aspecte esențiale pentru dezvoltarea proiectelor noastre. Veți regăsi mai multe informații despre fiecare element de circuit în cartea *Electronică și Robotică. Primii pași.*

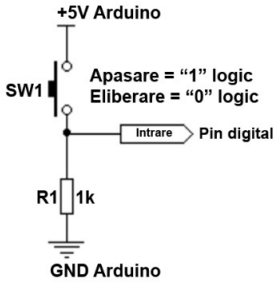
În cea mai simplă formă a sa, platforma de dezvoltare Arduino poate fi privită ca o cutie neagră, care recepționează ***semnale de intrare*** $(x_1, x_2, \cdots, x_n)$ precum: parametri preluați de la senzori – informații despre mediul înconjurător referitoare la temperatură, intensitate a luminii, distanța față de obstacole, formă și culoare a obiectelor din imediata apropiere sau informații introduse de utilizatori, precum apăsarea unor butoane sau rotirea axului unui potențiometru, pe care le procesează utilizând un ***sistem software***, definit de programatorul platformei conform unor specificații, și le transformă în ***semnale de ieșire*** $(y_1, y_2, \cdots, y_m)$, care vor fi transferate, sub formă de comenzi, unor elemente de acționare, cum ar fi motoarele electrice.



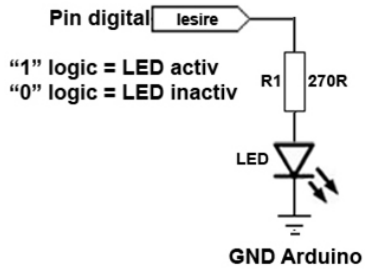
Totuși, lucrurile nu sunt chiar atât de simple: atât semnalele de intrare cât și cele de ieșire pot fi ***analogice*** sau ***digitale***. Vom încerca să descriem, pe scurt, aceste noțiuni pentru a ne forma o idee despre semnificația lor, însă le vom descrie în amănunt în capitolele care vor urma și veți constata că, pe măsură ce realizăm mai multe experimente, lucrurile ni se vor părea din ce în ce mai simple și mai ușor de înțeles. Un **semnal digital** poate avea doar două valori: ***„0" logic*** sau ***LOW***, asimilat unui comutator deschis (decuplat) sau conectării la terminalul

negativ al sursei de alimentare și ***„1” logic*** sau ***HIGH***, asimilat unui comutator închis (cuplat) sau conectării la terminalul pozitiv al sursei de alimentare.

Cel mai bun exemplu de **intrare digitală** îl reprezintă conectarea unui întrerupător cu revenire la una dintre intrările Arduino, așa cum este prezentat în schema alăturată. Apăsarea sa va conduce la alimentarea pinului platformei Arduino cu o tensiune de +5V și, astfel, va fi detectată o stare „1” logic. Eliberarea butonului va conduce la conectarea pinului platformei Arduino la terminalul negativ al sursei de alimentare, adică 0V, și, astfel, va fi detectată o stare „0” logic.

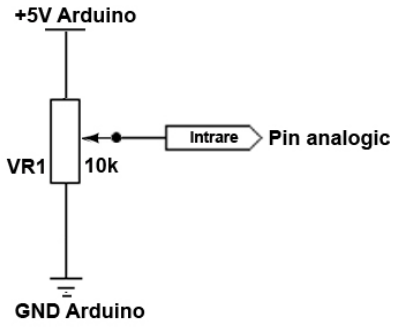


Cel mai bun exemplu de **ieșire digitală** îl reprezintă conectarea unui LED la una dintre ieșirile Arduino, așa cum este prezentat în schema alăturată. Activarea ieșirii (starea „1” logic) va presupune generarea unei tensiuni de +5V și ar fi similară conectării ansamblului format din LED și rezistor la o sursă de alimentare de 5V, ceea ce va determina LED-ul să ilumineze la intensitatea maximă. Dezactivarea ieșirii (starea „0” logic) va presupune conectarea la polul negativ al sursei de alimentare; astfel, lipsa tensiunii electrice va determina LED-ul să nu mai ilumineze.

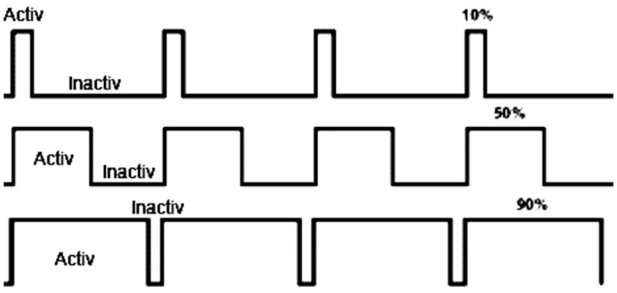


Un semnal analogic poate avea o multitudine de valori (limitată, în principiu, de conversia tensiunilor electrice în valori numerice) situate între „0” logic (0V) - conectarea la terminalul negativ al sursei de alimentare și „1” logic (+5V în cazul Arduino Uno) - conectarea la terminalul pozitiv al sursei de alimentare. Platforma Arduino Uno oferă posibilitatea de a converti acest interval de tensiune: [0; 5V] în numere naturale de la 0 până la 1023, deci are o rezoluție de aproximativ 4,9mV.

Un exemplu de **intrare analogică** îl reprezintă conectarea unui potențiometru la una dintre intrările analogice ale Arduino (doar anumite intrări ale platformei Arduino sunt analogice), așa cum este prezentat în schema alăturată. În cazul schemei prezentate, potențiometrul este conectat pentru a forma un divizor de tensiune și, astfel, rotirea axului său va determina modificarea valorii rezistenței electrice dintre o extremitate și cursor și, implicit, a tensiunii de la bornele intrării analogice a platformei Arduino. Apoi, tensiunea electrică măsurată va fi convertită, prin intermediul sistemului software, într-un număr natural de la 0 până la 1023. Această valoare poate fi utilizată ulterior pentru a regla viteza unui motor, luminozitatea unui LED sau pentru a afișa un număr care să reprezinte poziția în care se regăsește axul potențiometrului.



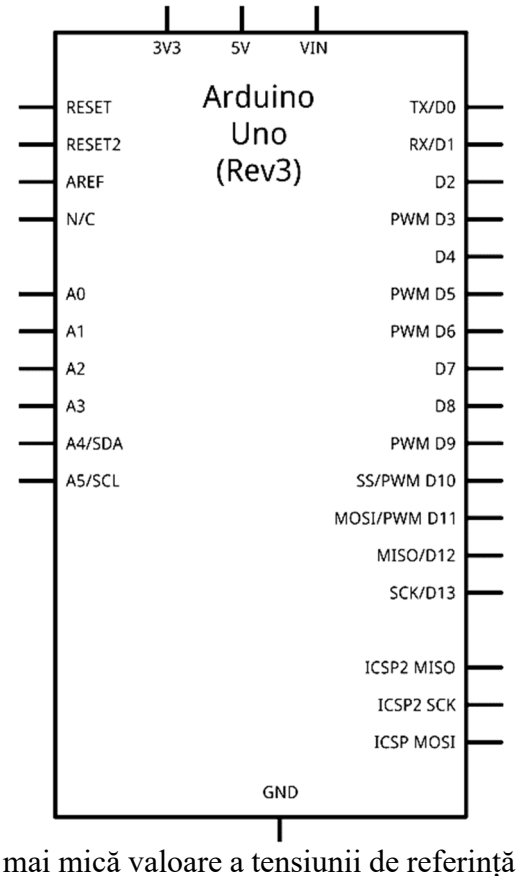
Cel mai bun exemplu de **ieșire analogică** îl reprezintă semnalul PWM (*Pulse Width Modulation – variația duratei impulsului, adică a perioadei cât ieșirea este activă).* Să presupunem că dorim să controlăm luminozitatea unui LED fără a modifica tensiunea de alimentare (deoarece nu întotdeauna reducerea tensiunii de alimentare va conduce la scăderea luminozității LED-ului). Soluția este să controlăm frecvența cu care LED-ul este aprins și stins. Raportul dintre perioada în care LED-ul rămâne aprins și cea în care LED-ul rămâne stins este denumită factor de umplere (*duty cycle*) și, cu cât va fi mai lungă perioada în care LED-ul este aprins, comparativ cu aceea în care LED-ul este stins cu atât ni se va părea că LED-ul ilumineză mai intens.



PWM - controlul factorului de umplere

Schema electrică Arduino

Cel mai răspândit model al platformei Arduino este Arduino Uno și acesta va fi modelul pe care îl vom utiliza în aplicațiile practice pe care urmează să le prezentăm. Întrucât trebuie să ne concentrăm mai întâi asupra schemei electrice a circuitului și, după aceea, asupra codului sursă, vom analiza pentru început dispunerea pinilor acestui dispozitiv.



V_{IN} – tensiune de alimentare pentru Arduino, atunci când este utilizată o sursă externă de alimentare.

5V – tensiune de ieșire de +5V pentru alimentarea dispozitivelor conectate la platformă.

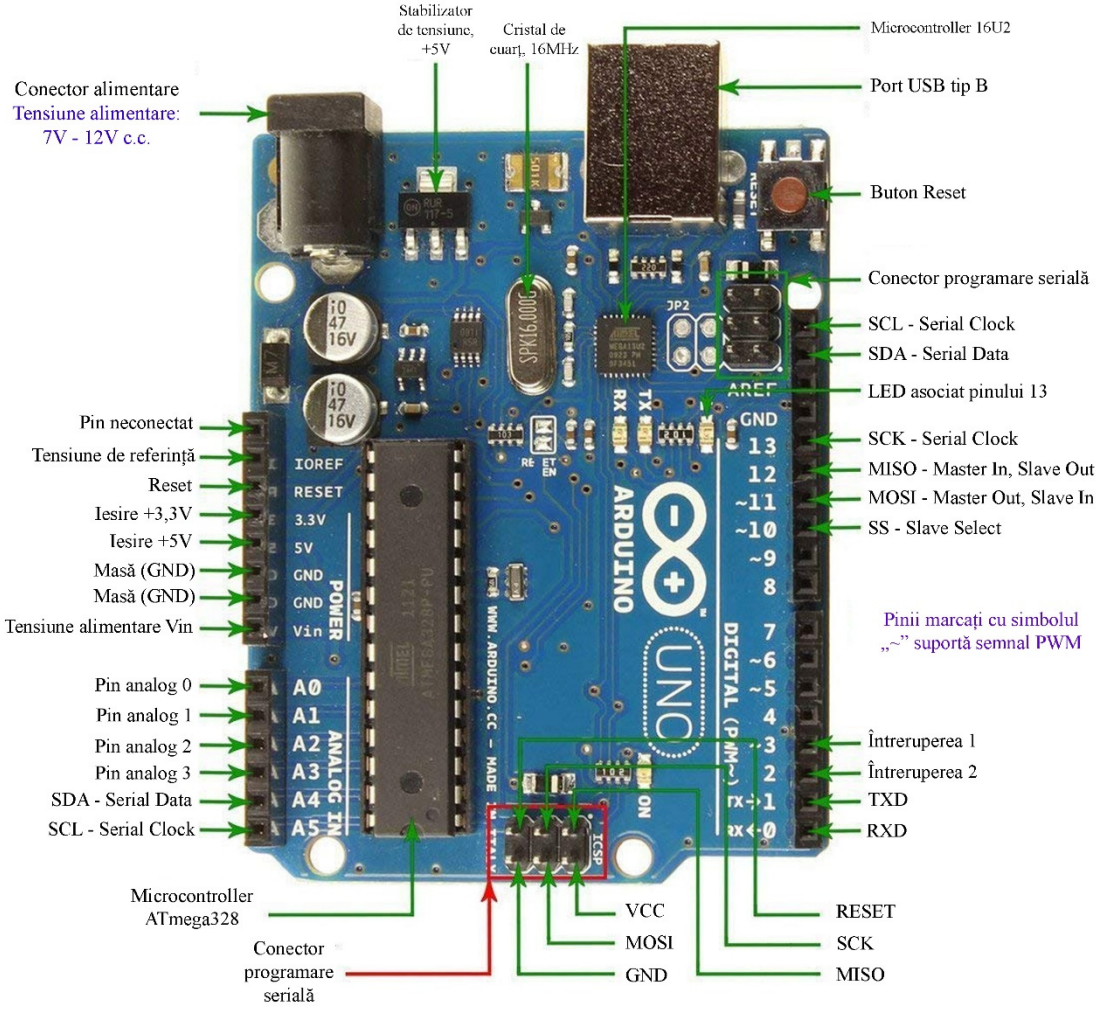
3.3V – tensiune de ieșire de +3,3V, cu un curent maxim suportat de 50mA, utilizată pentru a alimenta anumite dispozitive care necesită tensiune de alimentare de 3,3V.

RESET – oferă posibilitatea de a reseta dispozitivul. Pentru a reseta (aduce la starea inițială) platforma, pinul trebuie conectat la masă (GND).

AREF – permite furnizarea unei tensiuni, mai mică de 5V, ca referință pentru conversia semnalului analogic. De exemplu, dacă furnizăm pinului AREF o tensiune electrică de 3,3V acesta va converti intervalul [0; 3,3V] în numere naturale de la 0 până la 1023. Cea mai mică valoare a tensiunii de referință este de 1,1V.

D0÷D13 – *pini digitali*, care pot fi programați, utilizând sistemul software, ca semnale de intrare sau de ieșire, în funcție de ceea ce dorește programatorul. După cum observați în schema electrică, doar o parte dintre aceștia permit generarea unui semnal PWM: D3, D5, D6, D9, D10 și D11. În plus, pinii D0 (*Tx – Transmit*) și D1 (*Rx – Receive*) pot fi utilizați pentru comunicație serială (transfer de date) între un Arduino și dispozitive periferice sau între două platforme Arduino.

A0÷A5 – *intrări analogice*, utilizate, de exemplu, pentru a conecta un potențiometru. Pinii A4 (*SDA - Serial Data Line*) și A5 (*SCL -Serial Clock Line*) pot fi utilizați pentru stabilirea unei conexiuni seriale de tip master-slave denumită I²C sau IIC (*Inter-Integrated Circuit*). Acest tip de comunicație este utilizat pentru a transfera date între un Arduino și dispozitive care permit utilizarea acestui protocol, precum senzori, elemente de acționare sau alte microcontrolere.



Pinii D12 (*MISO - Master In Slave Out*), D13 (*SCK - System Clock*), D11(*MOSI - Master Out Slave In*) și D10 (*SS - Slave Select*) pot fi utilizați pentru stabilirea unei conexiuni seriale de tip SPI (*Serial Peripheral Interface*), utilizată pentru a transfera date între un Arduino și anumite dispozitive periferice.

Pinii D2 și D3 pot fi utilizați ca întreruperi, adică oferă posibilitatea de a recepționa semnale de la un dispozitiv periferic ce vor conduce la suspendarea firului normal de execuție al programului și lansarea în execuție a unei rutine de tratare a întreruperii (la sfârșitul execuției rutinei de tratare a întreruperii programul principal este reluat din punctul de unde a fost întrerupt, cu datele care fuseseră procesate până în momentul producerii întreruperii).

Tensiunea recomandată de alimentare este de 7 – 12V curent continuu și nu poate depăși 20V, iar ieșirile digitale suportă o sarcină de cel mult 40mA la tensiunea de 5V, ceea ce înseamnă că, dacă veți avea un consumator mai mare, trebuie să îl conectați la Arduino prin intermediul unui tranzistor de putere sau al unui releu.

Înainte de prima utilizare...

Am aflat până acum informații despre structura platformei de dezvoltare, adică despre partea hardware, însă, pentru a o putea utiliza, vom avea nevoie și de un sistem software. Drept urmare, înainte de a utiliza platforma de dezvoltare Arduino va trebui să pregătim calculatorul personal pentru a accesa sistemul software al acesteia.

Vom încerca să utilizăm în paralel, pentru a observa avantajele și dezavantajele fiecăreia, două aplicații: **Arduino IDE** și **Mixly**. În plus, vom prezenta și o aplicație web, **Tinkercad**, cu ajutorul căreia putem simula funcționarea circuitelor noastre electrice înainte de a le instala efectiv.

Utilizarea Arduino IDE presupune faptul că suntem oarecum familiarizați cu utilizarea limbajelor de programare și cu protocolul pe care îl impun acestea, adică faptul că va fi necesar să memorăm anumite instrucțiuni pe care le vom utiliza frecvent și să respectăm anumite reguli: practic, va fi ca și cum am învăța o limbă străină. În plus, este necesar și să păstrăm o anumită structură a programului, pentru a ne asigura că nu vor exista erori de compilare (*adică de transformare a codului scris de noi în instrucțiuni pe care le poate procesa platforma Arduino*).

În cazul Mixly, vom utiliza blocuri funcționale pentru a construi programele și este necesar doar să le asociem într-o secvență corectă și să modificăm anumiți parametri astfel încât programul să ruleze conform specificațiilor.

Instalare Arduino IDE

Kitul de instalare al sistemului software Arduino IDE (*integrated development environment* – o grupare de sisteme software suficiente pentru a elabora și testa aplicații) poate fi descărcat de pe site-ul oficial: https://www.arduino.cc/ secțiunea Software → Downloads (https://www.arduino.cc/en/Main/Software). Având în vedere că, uneori, este posibil să utilizați o clonă (adică un dispozitiv similar celui original, cu același microcontroler, dar cu o altă interfață de conectare la computer – *este vorba despre microcontroler-ul 16U2, care poate fi înlocuit și de alte microcontrolere, precum CH340*) este recomandat să descărcați și un driver

CH340 sau CH341. După ce veți instala programul descărcat, conectați placa de dezvoltare la computer, prin intermediul cablului USB și așteptați să se instaleze driver-ul. Pentru a avea acces la toate facilitățile pe care le poate oferi platforma de dezvoltare Arduino și pentru a fi siguri că aplicațiile și circuitele electrice pe care le proiectați și realizați funcționează la parametri normali, **este recomandat să utilizați întotdeauna dispozitive originale** și nu *clone* ale acestora.

În cazul unor versiuni de Windows, este posibil ca driverul să nu se instaleze corect. În această situație, veți recepționa mesajul „*Device driver software not successfully installed*". Pentru a rezolva problema, trebuie să accesați Windows Device Manager (*Start>Control Panel>Hardware>Device Manager* – pentru Windows 7 sau *Settings* pentru Windows 10 și, în bara de căutare, să scrieți „Device Manager") și să identificați Arduino Uno în lista dispozitivelor conectate la computer. După ce l-ați identificat, apăsați click dreapta și alegeți **Update driver.**

Selectați opțiunea **Install from a list of specific location (Advanced)** în fereastra următoare apoi apăsați **Next >.** În următoarea fereastră asigurați-vă că ați selectat opțiunea **Include this location** și căutați folderul în care ați descărcat sau copiat fișierele de instalare ale Arduino. Selectați folderul **drivers** și apăsați **OK**.

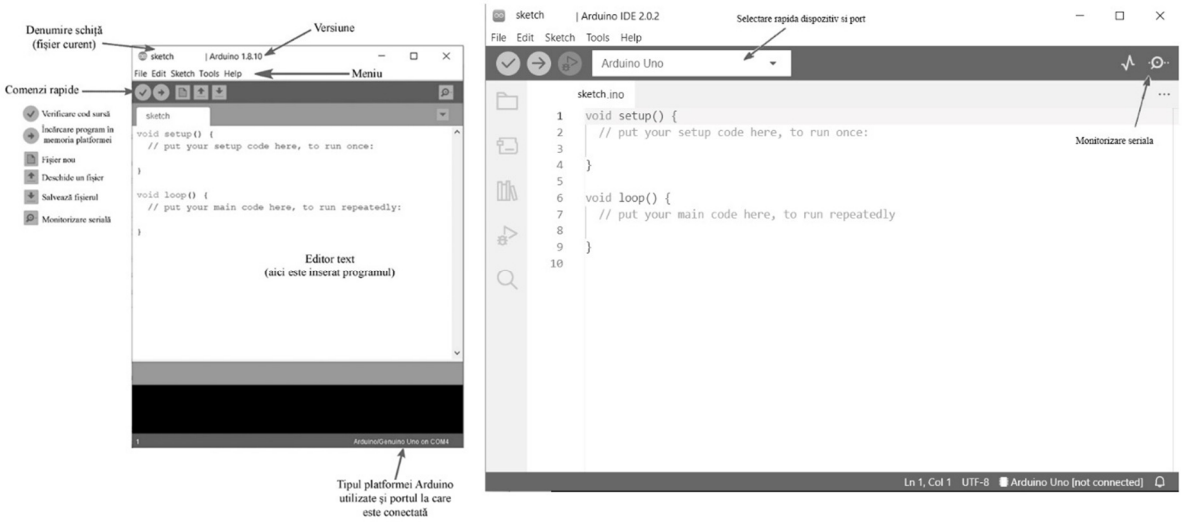
Dacă ați efectuat corect algoritmul de instalare, sistemul de operare va copia fișierele necesare și va instala driver-ul. În timpul instalării este posibil să apară o fereastră de avertizare care vă informează că urmează să instalați un dispozitiv care nu a fost testat pentru compatibilitate; apăsați **Continue Anyway** pentru a continua instalarea dispozitivului. După finalizarea instalării driverului, apăsați **Finish**.

Este necesar să reporniți computerul pentru instalarea corectă a driver-ului. După repornire, asigurați-vă de faptul că dispozitivul Arduino este conectat la computer prin intermediul cablului USB. LED-ul verde trebuie să ilumineze continuu. În acest moment va trebui să identificăm portul COM asociat dispozitivului Arduino Uno, deoarece îl vom selecta în Arduino IDE atunci când vom transfera codul sursă microcontrolerului. Pentru aceasta, va trebui să deschideți, din nou, *Device Manager* (din **Start Menu**, selectați **Settings → Control Panel**, dublu click pe opțiunea **System** și selectați **Hardware**. Apoi apăsați pe **Device Manager –** în cazul sistemului de operare Windows 7 sau, în cazul utilizării sistemului de operare Windows 10 – click pe Start Menu, aflat în partea din stânga jos a ecranului, apoi selectați *Settings* și, în bara de căutare, să scrieți „Device Manager").

Mediul de lucru este minimalist, fiind orientat pe o accesare cât mai simplă a funcțiilor. Astfel, meniul de sus include: ***File*** *(fișier nou, încărcare fișier existent, salvare fișier curent);* ***Edit*** *(copiere, lipire, selectare, căutare);* ***Sketch*** *(verificare și compilare a codului sursă,*

încărcare program în memoria platformei de dezvoltare, adăugarea unei biblioteci de funcții); ***Tools*** *(instrumentele necesare pentru testarea proiectelor) și* ***Help*** *(informații utile despre aplicație).* Partea de sub acest meniu este dedicată verificării codului sursă și încărcării acestuia pe dispozitiv.

În meniul *File* există un meniu secundar care conține diferite exemple, considerate relevante de autorii acestui sistem software. Aceste exemple sunt detaliate pe site-ul oficial al Arduino și constituie o metodă destul de eficientă de a vă familiariza cu platforma de dezvoltare.



Pentru a realiza corelația dintre intrări și ieșiri, va trebui să programăm microcontrolerul, adică să scriem cod sursă și să îl înregistrăm în memoria acestuia. Limbajul de programare este un limbaj C simplificat și, așa cum am specificat și anterior, atunci când vom utiliza Arduino IDE, vom presupune că aveți cunoștințe minime de programare. Drept urmare, nu ne vom concentra asupra noțiunilor de bază ci asupra celor specifice programării platformelor de dezvoltare. În cazul aplicației Mixly situația este mult mai simplă, nefiind necesare cunoștințe de programare încă de la început ci doar pe măsură ce avansăm.

După cum veți observa, atunci când deschidem o schiță nouă în aplicația Arduino IDE, aceasta conține deja două funcții, absolut necesare pentru funcționarea corectă: *void setup()* care stabilește parametrii inițiali – aici vor fi definite funcțiile fiecăruia dintre pini – adică se va stabili care dintre pini constituie intrare sau ieșire; va fi inițializată comunicația serială, prin

intermediul căreia sunt transmise computerului informații preluate de platforma de dezvoltare, utilizând conexiunea USB; va fi definită o temporizare sau o întrerupere și funcția *void loop(),* funcția principală, o buclă ce se repetă la nesfârșit și care conține informații despre procesarea semnalelor de intrare și conversia lor în semnale de ieșire. Pe lângă acestea, pot exista și alte funcții, care vor fi accesate în secvența principală, *void loop()*, deoarece trebuie să țineți cont că funcția *void setup()* rulează o singură dată, la punerea în funcțiune a dispozitivului sau, după caz, la resetarea acestuia.

Veți găsi o parte dintre codurile sursă ale activităților practice pe care le vom prezenta la adresa web http://www.electronicasirobotica.ro/coduri-sursa-arduino însă vă recomand să scrieți codul sursă de la tastatură în Arduino IDE, să îl verificați și să îl încărcați pe platforma de dezvoltare pentru a implementa proiectele. Astfel, scriind personal codul sursă, veți încerca să înțelegeți fiecare linie a acestuia; în plus, există o probabilitate destul de mare să mai greșiți și, fiind nevoiți să corectați greșelile, veți învăța.

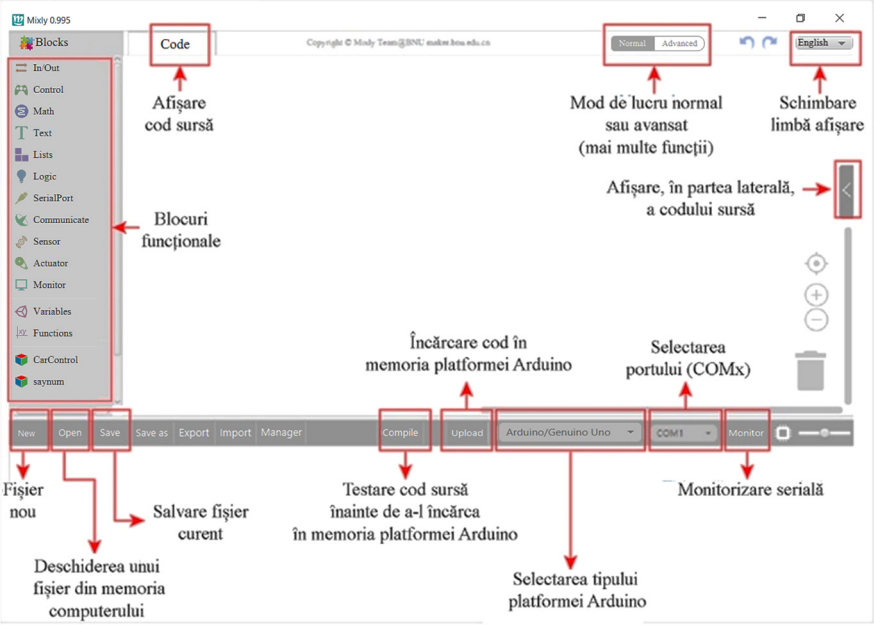
Trebuie să rețineți ceea ce nu este obligatoriu să copiați: comentariile. Comentariile sunt delimitate de caractere speciale: ceea ce se regăsește între /* și */ sau linia de cod care este urmată de //. (/* acesta este un comentariu */ și //acesta este un comentariu). Sistemul de marcare /* și */ este utilizat dacă vom avea un comentariu care se extinde pe mai multe linii, adică pe mai multe rânduri ale textului.

Instalare Mixly

Înainte de a prezenta modalitatea de instalare a aplicației, vă voi explica de ce am ales Mixly și nu altă aplicație. Două dintre aspectele esențiale sunt acelea că aplicația poate fi descărcată gratuit și că funcționează independent de Arduino IDE (marea majoritate a aplicațiilor similare depind de instalarea Arduino IDE). În plus, Mixly posedă o gamă foarte largă de blocuri funcționale și permite monitorizarea codului sursă odată cu plasarea blocurilor funcționale (astfel, vom putea învăța și limbajul de programare necesar pentru a utiliza Arduino IDE, care ne va fi necesar pentru a dezvolta aplicații mai complexe). Oricum, dacă veți decide să utilizați alte aplicații, precum: Ardublockly, BlocklyDuino, miniBloq, Blockly@rduino, Scratch for Arduino, Tinkercad Circuits sau altele similare, principiile vor fi aceleași.

Kitul de instalare al aplicației Mixly poate fi descărcat de pe site-ul oficial: https://wiki.keyestudio.com/Download_Mixly_Software. După descărcarea arhivei .rar sau .zip veți extrage informațiile și veți rula fișierul Mixly.exe.

Pentru aplicațiile pe care urmează să le prezentăm în capitolele următoare, am ales utilizarea variantei 0.995 a sistemului software Mixly.



Mediul de lucru este construit astfel încât suprafața cea mai mare să fie ocupată de blocurile funcționale; în același timp, dacă dorim, putem afișa și codul sursă asociat blocurilor plasate în fereastra principală. Blocurile funcționale pe care le putem utiliza se află în partea stângă a ferestrei și sunt organizate astfel încât să poată fi identificate cât mai ușor. Elementele prezentate mai jos fac parte din grupul de funcții ***In/Out***, care poate fi utilizat pentru operații simple asociate intrărilor și ieșirilor.

Stare a unei ieșiri digitale: HIGH - 1 logic, LOW - 0 logic

Schimbarea stării unei ieșiri digitale în LOW sau HIGH.
Parametrii sunt: numărul ieșirii și starea ieșirii.

Citirea stării unei intrări digitale, precum cea conectată la un senzor.
Parametrii sunt: numărul intrării și starea detectată.

Schimbarea stării unei ieșiri analogice: valoarea poate fi un număr natural între 0 și 255.
Această funcție este valabilă doar pentru ieșirile care suportă semnal PWM.
Parametrii sunt: numărul ieșirii și starea ieșirii.

Citirea stării unei intrări analogice, precum cea conectată la un potențiometru.
Valoarea poate fi un număr natural cuprins între 0 și 1023 iar funcția este valabilă doar pentru intrările analogice (pentru Arduino Uno acestea sunt de la A0 la A5).
Parametrii sunt: numărul intrării și starea detectată.

După cum observați, aceste blocuri funcționale sunt similare unor piese de puzzle, permițând astfel interconectarea lor cu alte blocuri funcționale. Vom prezenta însă în detaliu toate aceste informații în capitolele care vor urma.

2

Semnale de intrare și de ieșire, analogice și digitale.

În cadrul acestui capitol ne propunem să aflăm, cu ajutorul aplicațiilor practice, cum putem exploata intrările și ieșirile, analogice și digitale, ale platformei de dezvoltare Arduino.

Vom încerca să prezentăm informațiile și, în același timp, să efectuăm câte un experiment pentru a studia modalitatea în care sunt utilizate anumite componente electronice sau anumiți senzori împreună cu Arduino și modalitatea în care putem programa platforma de dezvoltare pentru proiectele pe care ne propunem să le dezvoltăm. Fiecare componentă electronică și fiecare funcție nouă va fi evidențiată prin utilizarea unei pictograme. Pentru fiecare proiect vom prezenta pe rând: componentele noi, funcțiile noi, schema electrică, algoritmul, implementarea sa și, unde va fi cazul, modalitatea de a realiza circuitul. Pentru început, vom programa dispozitivul Arduino mai întâi cu Mixly și apoi cu Arduino IDE, pentru a ne familiariza atât cu blocurile funcționale, cât și cu limbajul de programare sub formă de text.

Pentru primele experimente ne propunem să analizăm modalitatea de a utiliza ieșirile digitale astfel încât să controlăm diverse dispozitive, precum LED-uri și difuzoare. Apoi, vom modifica intensitatea luminii emise de un LED utilizând un semnal PWM, adică o ieșire analogică, vom prelua informații de la un buton pentru a modifica starea unui LED (învățând astfel cum putem utiliza intrările digitale) și, la final, vom prelua informații de la un potențiometru (pe care l-am putea înlocui și cu un fotorezistor sau oricare alte componente electronice a căror rezistență electrică variază în anumite condiții), utilizând intrările analogice, pentru a controla luminozitatea unui LED.

Activitate practică Nº. 1: Lumină dinamică - LED intermitent.

Ce ne propunem să realizăm pentru primul nostru experiment? Dorim să construim un circuit electric care asigură funcționarea intermitentă, la interval de o secundă, a unui LED (adică LED-ul va ilumina pentru o secundă, apoi se va stinge pentru o secundă, va ilumina din nou pentru o secundă și ciclul se va repeta la nesfârșit, până la întreruperea alimentării circuitului electric).

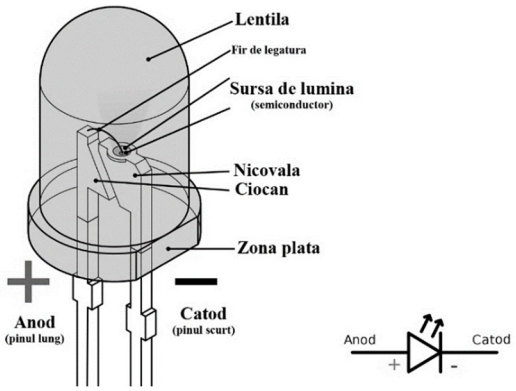
Pentru fiecare proiect vom avea nevoie de cel puțin două elemente: schema electrică a circuitului pe care urmează să îl construim și programul pe care îl vom încărca în memoria platformei Arduino.

Înainte de a prezenta schema electrică, vom analiza componentele electronice pe care urmează să le utilizăm: un LED (elementul de circuit care produce lumină atunci când este alimentat) și un rezistor, al cărui scop este de a adapta funcționarea LED-ului la parametrii sursei de alimentare a circuitului electric pe care îl construim.

Așa cum am precizat la început, ne vom concentra doar asupra aspectelor strict necesare despre componentele electronice pe care urmează să le utilizăm pentru prima dată: **LED-urile** (*light-emitting diode - dioda electroluminiscentă*) sunt diode proiectate pentru a emite lumină – atunci când LED-ul este în conducție directă (adică alimentat corespunzător), energia eliberată de deplasarea și schimbarea stării electronilor este transformată, în mare parte, în lumină (o cantitate destul de mică de energie este convertită și în căldură). Culoarea luminii emise depinde de materialul semiconductor utilizat: de exemplu, pentru culoarea roșie este utilizat un aliaj format din galiu, aluminiu și arsen iar pentru culoarea albastră este utilizat un aliaj zinc-seleniu.

Pentru a funcționa corect, LED-ul trebuie alimentat într-o anumită direcție (acesta posedă un pin denumit **anod** - care va fi conectat la polul pozitiv al sursei de alimentare și altul denumit **catod** - care va fi conectat la polul negativ al sursei de alimentare; conectarea greșită la sursa de alimentare poate conduce la defectarea iremediabilă a LED-ului). Pentru a evita posibilitatea alimentării eronate, avem câteva metode de a identifica polii LED-ului:

- terminalul pozitiv ***(anod)*** este mai lung decât terminalul negativ ***(catod)***
- capsula prezintă o teșitură în dreptul terminalului negativ (catod)
- dacă veți privi în interiorul capsulei LED-ului, veți observa că unul dintre pini are formă de ciocan și celălalt de nicovală; forma de ciocan este asociată terminalului pozitiv (anod), iar forma de nicovală - terminalului negativ (catod).



Structura și reprezentarea grafică a unui LED.

Informațiile de catalog esențiale pentru utilizarea corespunzătoare sunt: tensiunea directă, notată cu U_F (tensiunea la care LED-ul începe să emită lumină) și curentul direct, notat cu I_F (curentul optim de funcționare a LED-ului). Dacă nu avem la dispoziție informații specifice despre LED-ul pe care îl utilizăm și acesta este unul obișnuit, vom considera $U_F = 3V$ și $I_F = 10{\sim}20mA$.

Pentru a nu defecta LED-ul atunci când îl alimentăm trebuie să îi asigurăm condiții optime de funcționare: o intensitate a curentului și o tensiune electrică adecvată. Astfel, pentru a adapta un LED la parametrii electrici ai platformei Arduino, este necesar să utilizăm un **rezistor**. Rolul unui rezistor este de a limita intensitatea curentului electric pe o anumită porțiune de circuit, deoarece unele componente necesită un anumit nivel al intensității curentului electric la bornele lor pentru a funcționa corect. În plus, în anumite configurații, acestea pot reduce inclusiv tensiunea electrică dintr-o anumită regiune a circuitului. Drept urmare, rezistoarele sunt utilizate pentru a adapta anumite componente electronice la condițiile oferite de sursa de alimentare prezentă în circuit (având în vedere că nu putem să adaptăm sursa de alimentare la condițiile necesare pentru funcționarea fiecărui element de circuit și nici să utilizăm câte o sursă de alimentare dedicată pentru fiecare componentă a circuitului ci trebuie să adaptăm fiecare element de circuit la funcționarea optimă în condițiile oferite de circuit). Mărimea electrică utilizată pentru a cuantifica opoziția rezistorului asupra circulației curentului electric într-un circuit se numește rezistență electrică și este măsurată utilizând unitatea de măsură denumită Ohm (notată utilizând litera grecească Ω - omega).

Există două probleme esențiale pe care le întâmpinăm în cazul rezistoarelor: prima este aceea că nu există rezistoare disponibile pentru fiecare valoare a rezistenței electrice și, de aceea, de fiecare dată când efectuăm un calcul, dacă valoarea nu este critică, trebuie să o aproximăm la o valoare disponibilă. Astfel, există un set de valori standard: 1Ω; 1,2Ω; 1,5Ω; 2,2Ω; 2,7Ω; 3,3Ω; 3,9Ω; 4,7Ω; 5,6Ω; 6,8Ω; 8,2Ω și multipli ai acestora (valori pe care le obținem înmulțind aceste valori cu 10, 100, 1000 și așa mai departe). A doua problemă este că, din cauza gabaritului redus, majoritatea rezistoarelor nu sunt marcate alfanumeric (adică utilizând cifre și litere) ci prin intermediul unui cod al culorilor. Mai precis, cel mai răspândit sistem de marcare utilizează patru benzi colorate, fiecare dintre acestea înconjurând rezistorul.

Primele două benzi reprezintă valoarea semnificativă, din două cifre, a rezistorului. Următoarea bandă, a treia, este coeficientul de multiplicare cu 10, adică numărul de ori cu care vom înmulți valoarea semnificativă cu 10 (de exemplu, dacă a treia banda are o culoare asociată cifrei 2 vom efectua o înmulțire cu 100, adică înmulțim de două ori cu 10). Ultima bandă reprezintă toleranța sau precizia rezistorului (cât de apropiată poate fi de valoarea marcată – de exemplu un rezistor cu valoarea rezistenței de 1000Ω și cu o toleranță de 10% poate avea o valoare cuprinsă între 900Ω și 1100Ω).

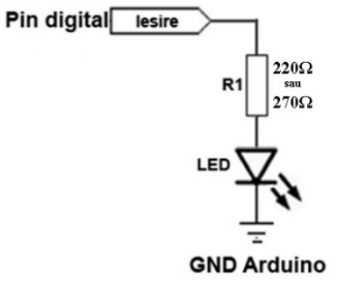
Asocierea cifră - culoare este următoarea: 0 – negru, 1 – maro, 2 – roșu, 3 – portocaliu, 4 – galben, 5 – verde, 6 – albastru, 7 – violet, 8 – gri, 9 – alb. Există două culori suplimentare, folosite cel mai frecvent pentru a delimita toleranța: auriu (5%) și argintiu (10%). Aceste culori sunt utilizate și pentru reprezentarea rezistențelor cu valori mai mici decât 1Ω reprezentând factori de multiplicare de 0,1 (auriu) și 0,01 (argintiu). Pentru a reține mai ușor asocierea cifră-culoare țineți cont că la extremități, 0 și 9, se află negru și alb, iar de la 2 la 7 sunt culorile curcubeului (R.O.G.V.A.I.V.), mai puțin indigo, înglobat în violet, deoarece, dacă erau utilizate ambele culori, nu am fi putut să le deosebim. Va trebui să mai rețineți doar că maro este asociat cu 1 și gri cu 8.

Culoare	Prima banda	A doua banda	Multiplicator	Toleranta
Negru	0	0	1Ω	
Maro	1	1	10Ω	± 1%
Rosu	2	2	100Ω	± 2%
Portocaliu	3	3	1kΩ	
Galben	4	4	10kΩ	
Verde	5	5	100kΩ	± 0.5%
Albastru	6	6	1MΩ	± 0.25%
Violet (mov)	7	7	10 MΩ	± 0.1%
Gri	8	8		± 0.05%
Alb	9	9		
Auriu			0.1Ω	± 5%
Argintiu			0.01Ω	± 10%

Codul culorilor pentru un rezistor marcat cu 4 inele colorate.

Există și alte modalități de marcare a rezistoarelor iar, pentru a afla mai multe detalii, puteți accesa pagina web: http://www.electronicasirobotica.ro/rezistenta-electrica-si-rezistoarele

Cum adaptăm LED-ul la tensiunea de alimentare a dispozitivului Arduino? În primul rând, cunoaștem faptul că tensiunea furnizată de platforma Arduino Uno este 5V și, așa cum am specificat anterior, în general, tensiunea directă a unui LED este $U_F = 3V$ și vom alege o valoare pentru curentul direct: $I_F = 10mA$. Vom utiliza următoarea relație matematică:



$$R_{LED} = \frac{U_{Arduino} - U_{F\,LED}}{I_{F\,LED}} = \frac{5V - 3V}{10mA} = \frac{2V}{0{,}01A} = 200\Omega$$

În practică, vom utiliza un rezistor fix cu o rezistență de 220Ω (marcat *roșu-roșu-maro-auriu/argintiu)* sau 270Ω (marcat *roșu-violet-maro-auriu/argintiu)*, conectat între ieșirea digitală a platformei Arduino și polul pozitiv al LED-ului, așa cum este prezentat în imaginea alăturată.