Traian Anghel

PROGRAMAREA PLĂCII ARDUINO

Ediția a II-a

Editura Paralela 45

Editura Paralela 45 este recunoscută de
Consiliul Naţional al Cercetării Ştiinţifice (CNCS)

Redactare: Bianca Vişan
Tehnoredactare & pregătire de tipar: Marius Badea
Design copertă: Ionuţ Broştianu

Descrierea CIP a Bibliotecii Naţionale a României
ANGHEL, TRAIAN
Programarea plăcii Arduino / Traian Anghel. - Ed. a 2-a. - Piteşti : Paralela 45, 2020
Conţine bibliografie
ISBN 978-973-47-3204-3

004

Cuprins

Introducere

„Lucrul cel mai important este să nu încetezi să pui întrebări.
Curiozitatea are propria raţiune de a exista."

Albert Einstein

Astăzi, singura constantă este schimbarea, lumea transformându-se într-un ritm nemaiîntâlnit. În zilele noastre, suntem martorii începutului celei de-a patra revoluţii industriale[1], care constă în pătrunderea rapidă a tehnologiilor digitale în toate domeniile. „Viitorul este aici, doar că nu este uniform distribuit", spunea cândva William Gibson, scriitor canadian de origine americană. La fel se întâmplă şi acum, când noua revoluţie industrială se manifestă deja în zone diverse ale activităţii omului, cuprinzând treptat un număr mare de sectoare ale activităţii economice, sociale, dar şi personale a acestuia.

Tehnologia avansată pătrunde în toate domeniile, inclusiv în cele care până nu de mult păreau a fi „ferite" de aceasta. Automatizarea casei (*home automation*) este un concept care se dezvoltă extrem de rapid. De exemplu, se pot controla lumininile din casă, precum şi sistemele de încălzire, de aer condiţionat şi de securitate ale acesteia. De asemenea, se pot utiliza diverse *gadgeturi* în apropierea casei (cu care, de exemplu, sunt controlate temperatura şi umiditatea aerului din sera de flori sau de legume) sau în drumul către casă (e.g. se porneşte maşina de spălat).

În următorul deceniu, mai mult de un trilion de senzori vor fi conectaţi la Internet, conform afirmaţiilor făcute la Forumul Economic Mondial de la Davos, 2016. Aceştia vor trimite diverse informaţii omului, dar vor comunica şi între ei, scopul acestei uriaşe „desfăşurări de forţe" fiind de a oferi omului un trai mai bun, mai confortabil şi mai sigur.

---

[1] Această idee a fost avansată la cea de-a 46-a ediţie a Forumului Economic Mondial, desfăşurată la Davos (Elveţia), în perioada 20-23 ianuarie 2016.

Conform *Business Insider* (*www.businessinsider.com/*), în 2015 erau conectate la Internet circa 15 miliarde de dispozitive (dintre care numai jumătate erau calculatoare, tablete şi telefoane mobile), iar în 2020 vor fi conectate 34 de miliarde. O parte dintre acestea sunt ceasuri inteligente, frigidere, brăţări de fitness, camere de supraveghere, televizoare *smart*, dar şi o mulţime de dispozitive construite chiar de către utilizatori. Acea parte a Internetului la care sunt conectate dispozitivele enumerate este cunoscută sub numele de *Internetul lucrurilor* (*Internet of Things*, IoT).

Una dintre caracteristicile definitorii ale tehnologiei actuale constă în aceea că este accesibilă practic oricărei persoane. O altă caracteristică importantă este adaptabilitatea ei la necesităţile personale. Din acest motiv, este posibilă construirea unor dispozitive electronice în propria casă (garaj[2]) sau în laboratorul de fizică al şcolii[3]. Astfel, bazându-se pe cunoştinţele şi deprinderile obţinute în gimnaziu şi liceu, dar şi pe cele dobândite din diverse surse aflate la îndemână, oricine are posibilitatea de a construi dispozitive pe care le poate utiliza în cele mai variate scopuri.

Este adevărat că multe dintre dispozitivele de care avem nevoie la un moment dat pot fi cumpărate. Dar ne putem imagina unele extrem de personalizate pe care ori nu le putem găsi în comerţ, ori – dacă le găsim sau le putem obţine la comandă – sunt destul de scumpe. Iată argumente care vin în sprijinul ideii de a ne crea propriul „laborator" în care să realizăm dispozitivele care ne sunt necesare sau măcar o parte dintre ele.

Deoarece aceste dispozitive electronice îndeplinesc sarcini repetitive, ele trebuie să funcţioneze pe baza unor programe. De asemenea, dacă se doreşte schimbarea sarcinilor îndeplinite sau cel puţin a unor parametri ai acestora, programele pe care se bazează funcţionarea lor trebuie înlocuite sau cel puţin modificate, ceea ce înseamnă că dispozitivele trebuie să fie programabile. Iar dacă programarea lor poate fi realizată de către utilizator, atunci dispozitivele pot fi adaptate sau modificate în funcţie de nevoile acestuia. Este situaţia cea mai convenabilă pentru utilizatorii interesaţi de crearea propriilor dispozitive electronice.

---

[2] Steve Jobs a început să asambleze calculatoare, la mijlocul anilor 1970, împreună cu Steve Wozniak, în garajul casei părinteşti. Ambii sunt cofondatori ai companiei Apple, înfiinţată în anul 1976.

[3] Construirea unor dispozitive chiar de către utilizatori este facilitată în cea mai mare măsură de structura modulară a acestora. Adesea, modulele componente pot fi tratate drept cutii negre, fiind considerate importante pentru utilizatori numai operaţiile pe care acestea le efectuează asupra datelor de intrare.

Capitolul 1. Noţiuni de electricitate

„Studiază mai întâi ştiinţa şi continuă apoi cu
practica născută din această ştiinţă!"

Leonardo da Vinci

În primul capitol sunt prezentate succint o serie de noţiuni de electricitate necesare pentru buna înţelegere a lucrării, în special a exemplelor practice oferite în cadrul acesteia. Precizăm că noţiunile prezentate în capitolul de faţă se regăsesc între cele incluse în programele de fizică pentru gimnaziu şi de liceu, fiind astfel accesibile tuturor persoanelor interesate de buna înţelegere a lucrării.

1.1. Structura corpurilor

Eforturile depuse de savanţi de-a lungul secolelor – în special fizicieni şi chimişti – au permis obţinerea unei imagini bogate şi complexe a realităţii înconjurătoare, în particular a structurii corpurilor.

> Astăzi, se ştie că toate corpurile, indiferent de starea de agregare a acestora, sunt compuse din atomi şi molecule. Moleculele sunt structuri alcătuite din atomi uniţi prin câteva tipuri de legături chimice (e.g. covalentă, ionică). Un tip special de legătură este cea metalică.

După încercarea de pionierat[1] din 1904 a lui J.J. Thomson (care a elaborat un model simplificat al atomului, supranumit „cozonacul cu stafide"), structura atomului a fost dezvăluită în 1911 de către fizicianul Ernest Rutherford (1871-1937), laureat al premiului Nobel pentru chimie (1908), considerat „părintele" fizicii nucleare şi cel mai mare experimentator după Michael Faraday. Rutherford a formulat, ca urmare a celebrelor sale experienţe de împrăştiere a particulelor alfa[2] pe foiţe metalice subţiri,

[1] Menţionăm că primul model atomic a fost cel al lui Dalton (1766-1844). Conform acestuia, atomul este sferic şi fără structură internă, atomii aceluiaşi element fiind identici, dar diferiţi de atomii altor elemente.

[2] Particulele alfa (α) sunt nuclee de heliu (i.e. atomi de heliu care au cedat prin ionizare ambii electroni), cu o sarcină pozitivă de două ori mai mare decât cea a electronului şi cu o masă de circa 7300 de ori mai mare decât masa unui electron.

modelul planetar al atomului (figura 1). În cadrul acestui model, atomul este similar cu sistemul planetar, în cadrul căruia – după cum se ştie – planetele (inclusiv Pământul) se rotesc pe orbite aproximativ circulare în jurul Soarelui.

În cadrul modelului elaborat de Rutherford, atomul (având raza de circa $10^{-10} \div 10^{-9}$ m) este alcătuit dintr-un înveliş electronic şi un nucleu (având raza de circa $10^{-15} \div 10^{-14}$ m), în jurul căruia se rotesc pe orbite circulare electronii. În modelul planetar al atomului a fost introdusă prima oară în fizică noţiunea de nucleu atomic.

Electronul (cu simbolul e^-) este o particulă elementară, cu sarcina electrică negativă şi egală în modul cu sarcina electrică elementară (pentru detalii, a se citi secţiunea următoare), având masa foarte mică[3]:

$$e = -1,6 \cdot 10^{-19}\,\mathrm{C} \qquad m = 9,1 \cdot 10^{-31}\,\mathrm{kg} \tag{1}$$

Electronii unui atom formează învelişul electronic al acestuia. Numărul electronilor unui atom se numeşte număr atomic şi se notează cu *Z*.

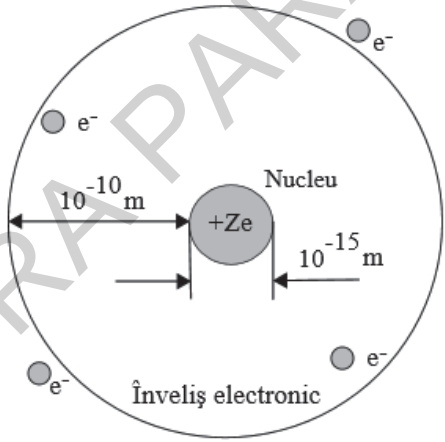

Figura 1. Modelul planetar al atomului

Nucleul atomic, conţinând aproape întreaga masă a atomului[4], este încărcat pozitiv, fiind alcătuit din protoni[5] (Rutherford, 1920) şi neutroni

[3] Masa unui electron este de circa 1840 de ori mai mică decât masa nucleului celui mai uşor atom din natură (atomul de hidrogen).

Capitolul 3. Prezentarea plăcii şi a mediului de dezvoltare Arduino

„Fără o mare perseverenţă nu există talente, nici genii."

Dmitri Mendeleev

Pentru a programa plăcile de dezvoltare din familia Arduino este necesar ca pe calculator să se instaleze mediul de dezvoltare Arduino IDE (*Integrated Development Environment*), dar şi o serie de biblioteci, precum şi software suplimentar. Arduino IDE are versiuni pentru platformele software uzuale (Windows, Linux, Mac OS). Înainte de analiza procedurii de instalare a software-ului, este realizată o prezentare a familiei de plăci Arduino, în special a versiunii Arduino UNO[1] R3 (Revision 3), utilizată cu precădere în lucrarea de faţă.

3.1. Placa de dezvoltare Arduino

Arduino este o familie de plăci de dezvoltare (*development boards*) cu microcontroler, comercială, accesibilă ca preţ, care poate fi programată folosind un limbaj de programare similar cu C/C++, cu ajutorul mediului de dezvoltare integrat Arduino IDE.

> Arduino a fost dezvoltată începând cu anul 2005, sub forma unui proiect al lui Masimmo Banzi, student italian în Ivrea, Italia. Din echipa iniţială au mai făcut parte încă patru persoane. Numele Arduino provine de la un bar în care aceştia obişnuiau să se întâlnească.

Plăcile de dezvoltare Arduino pot fi echipate cu un număr impresionant de senzori (e.g. de temperatură, umiditate, presiune, forţă, distanţă, nivel de iluminare), actuatori (e.g. motoare de curent continuu, motoare pas cu pas sau servomotoare) şi plăci de extensie (cunoscute sub denumirea de *shields*), cu scopul de a construi dispozitive (pe care le vom numi în cele ce urmează proiecte) pentru obţinerea informaţiilor din mediul extern, pentru controlul acestuia sau pentru comunicarea cu alte echipamente, folosind diverse medii de comunicaţie (Bluetooth, Ethernet, WI-FI, GSM,

[1] Numele UNO a fost dat plăcii pentru a celebra prima versiune a Arduino IDE (1.0), care – până atunci – era în versiune Beta.

GPS). Pentru afişarea informaţiilor primite din mediul extern sau a altor tipuri de informaţii, plăcile din familia Arduino folosesc ecrane LCD (*Liquid-Crystal Display*), de la cele mai simple (pe care se pot afişa informaţii text), până la cele grafice.

Se poate spune – fără teama de a greşi – că limitele care pot să apară în dezvoltarea proiectelor cu plăcile din familia Arduino sunt determinate numai de imaginaţia celor care le folosesc, acestea putând fi utilizate de profesionişti care activează într-un număr mare de domenii, inclusiv de artişti, dar şi de către oricine altcineva, în scop personal sau ca hobby.

Arduino este un echipament cu sursă deschisă (*open-source hardware*). Din acest motiv, designul plăcii, schemele şi codul sursă sunt disponibile în mod gratuit oricui doreşte să le folosească. Printre altele, aceasta înseamnă că, alături de producătorul italian al plăcilor Arduino, există numeroşi alţi fabricanţi de clone (ale căror preţuri sunt adesea mai avantajoase), dar şi de variante îmbunătăţite. Din punct de vedere legal, numai plăcile fabricate de firma mamă din Italia pot purta numele Arduino. Alegerea între placa originală şi o clonă este determinată în primul rând de bugetul de care dispune cel care dezvoltă proiecte cu Arduino. Pentru dezvoltarea proiectelor incluse în lucrarea de faţă au fost utilizate atât plăci originale Arduino, cât şi clone, fără a observa diferenţe semnificative între performanţele şi funcţionarea acestora.

Arduino este fabricată în două formate: formatul de dezvoltare (folosit în lucrarea de faţă) şi formatul de producţie. Plăcile realizate în cele două formate sunt total compatibile din punct de vedere funcţional, ele fiind deosebite numai în ceea ce priveşte dimensiunile, plăcile de producţie având dimensiuni mai mici decât cele de dezvoltare. De asemenea, plăcile de producţie nu conţin modulul de comunicaţie USB şi sunt mai ieftine. În general, proiectele sunt create pe o placă de dezvoltare, fiind transferate după finalizare pe o placă de producţie (e.g. Arduino Nano, Arduino Mini).

„Creierul" unei plăci Arduino este microcontrolerul (MCU, *microcontroler unit*). Cele mai multe plăci Arduino, inclusiv versiunea UNO, folosesc un microcontroler AVR ATmega[2], dezvoltat de compania Atmel. Acesta este bazat pe o arhitectură Harvard RISC pe opt biţi, modificată. De exemplu, Arduino UNO, dar şi Arduino Nano folosesc microcontrolerul

[2] Arduino Due foloseşte un microcontroler ARM Cortex, pe 32 de biţi.

Capitolul 4. Bazele programării plăcii Arduino

„Mai multe lucruri a dus la capăt iscusinţa decât forţa.”

Baltasar Gracián

Interacţiunea dintre Arduino şi dispozitivele conectate la pinii acesteia este comandată prin intermediul unui program încărcat pe placă. În capitolul de faţă sunt abordate subiecte care se pot constitui ca punct de plecare în studiul programării plăcii Arduino (e.g. structura unui program, intrări/ieşiri digitale, intrări analogice, generarea semnalelor PWM, utilizarea întreruperilor, afişarea datelor).

4.1. Structura unui program Arduino

Un program scris pentru placa Arduino poartă denumirea de *sketch* şi are extensia *.ino*. După lansarea Arduino IDE, este creat automat un *sketch* „gol”, al cărui nume implicit este *sketch_MonthDayChar*, unde *Month* este luna curentă, *Day* este ziua curentă şi *Char* este o literă (e.g. *sketch_mar20a*). Oricare *sketch* are următoarea structură:

```
void setup() {
  // put your setup code here, to run once
}
void loop() {
  // put your main code here, to run repeatedly
}
```

În programul anterior au fost păstrate comentariile în limba engleză, aşa cum apar ele în *sketch*-ul creat automat la pornirea Arduino IDE. Iată şi traducerea în limba română:

```
void setup() {
  /* pune aici codul de initializare, pentru a rula
     o singura dată: */
}
void loop() {
  /* pune aici codul principal, pentru a rula în
     mod repetat: */
}
```

După cum se poate observa, spre deosebire de un program scris în limbajul C/C++ standard, un *sketch* care rulează pe o placă Arduino are două secţiuni sau blocuri standard, obligatorii: *setup* şi *loop*.

În secţiunea *setup* este realizată iniţializarea conţinutului variabilelor şi este configurat comportamentul porturilor plăcii. Codul inclus în această secţiune va fi executat o singură dată (la alimentarea plăcii sau la apăsarea butonului *Reset* al acesteia). În schimb, secţiunea *loop* include partea principală a codului, care va fi executată repetitiv (acesta este motivul pentru care secţiunea poartă numele *loop*, adică buclă). Cele două secţiuni sunt implementate sub forma a două rutine (funcţii), `setup()` şi `loop()`, ambele fiind de tipul `void` (i.e. nu întorc nimic).

Dacă *sketch*-ul foloseşte variabile partajate între secţiunile *setup* şi *loop*, acestea trebuie definite în afara lor, deoarece sunt variabile globale. Definirea lor se va face într-o a treia secţiune, situată înaintea celor două secţiuni standard. De asemenea, în cadrul acesteia se pot folosi directive standard de preprocesor (e.g. `#define`, `#include`, `#if`, `#ifdef`, `#ifndef`, `#endif`, `#else`).

Directiva de preprocesor `#include` face posibilă includerea unor fişiere şi este utilizată astfel:

```
#include "numeFisier"
```

sau

```
#include <numeFisier>
```

O linie `#include` este înlocuită cu conţinutul integral al fişierului specificat. Un fişier inclus poate conţine la rândul său alte directive `#include`.

Folosind directiva de preprocesor `#define` se definesc macrouri, care, ulterior, pot fi apelate când se doreşte. Un macro constituie o modalitate de prelucrare a textelor bazată pe substituţie, ceea ce înseamnă că în definiţia unui macro se specifică textul care urmează a se substitui la fiecare apel al acestuia. În particular, se recomandă definirea constantelor simbolice folosind directiva `#define` şi nu ca variabile, deoarece în ultimul caz se consumă mai multă memorie din memoria SRAM.

Pentru compilarea condiţionată se folosesc directivele de preprocesor `#if`, `#ifdef`, `#ifndef`, `#endif`, `#else` etc. Acest tip de compilare permite să se aleagă dintr-un program părţile care se vor compila împreună.