UNIVERSITATEA POLITEHNICA TIMIŞOARA

Facultatea de Automatică şi Calculatoare

Sisteme Încorporate

**Proiect: Realizarea unei versiuni simplificate a jocului Arkanoid**

Robert – Bogdan CAZACU

Alexandra – Mirela MOCANU

„Echipa Rachetă”

An III CTI

An universitar 2017-2018

1. **Enunțul proiectului:**

* Se vor defini 3 tipuri de blocuri în funcție de duritate;
* Se va utiliza o matrice cu LED-uri (de dimensiune 8\*8 sau mai mare)/ afișaj OLED, TFT

etc., pentru afişarea pieselor de joc;

* Jucătorul va fi reprezentat de un bloc aflat la baza ecranului;
* Deplasarea la stânga sau la dreapta a jucătorului se va realiza prin acţionarea câte unui

buton corespunzător fiecărui sens (o apăsare corespunde unei deplasări cu o poziţie);

* Opțional se pot implementa diferite bonusuri (viață extra, minge mai puternica, etc.).

1. **Descrierea plăcii de dezvoltare utilizate:**

Pentru realizarea acestui proiect s-a ales placa de dezvoltare Arduino Uno.

1. Caracteristici generale ale plăcii Arduino Uno [1] :

* este o platformă cu microcontroler foarte simplă de utilizat din familia Arduino;
* informaţia este preluată de la o gamă variată de elemente de intrare (senzori şi comutatoare), se procesează în interiorul microcontrolerului şi este transmisă către o gamă la fel de variată de elemente de ieşire: leduri, motoare, actuatoare, etc.;
* poate fi folosit pe orice sistem de operare (Linux, Windows sau MacOS). Majoritatea plăcilor de dezvoltare fiind limitate la sistemul de operare Windows;
* un microcontroler instalat pe platforma Arduino vine preprogramat cu un bootloader care simplifică încărcarea programelor pe memoria flash a cipului, în comparaţie cu alte dispozitive care necesită programatoare externe;
* programele Arduino pot fi scrise în orice limbaj de programare cu un compilator capabil să producă un cod maşină binar;
* pentru programare se utilizează software-ul Arduino IDE (Integrated Development Environment) care suportă limbajele de programare C şi C++ folosind reguli speciale de organizare a codului.

1. Tipul de microcontroler:

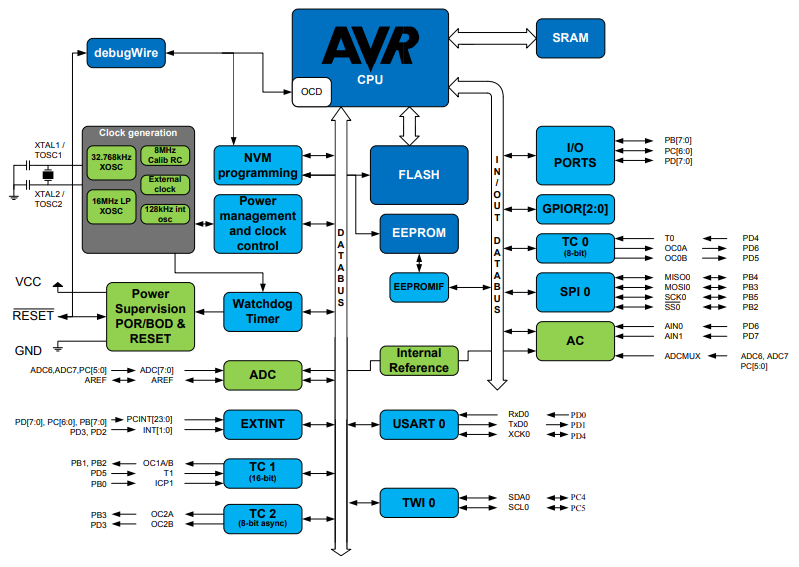
O platformă Arduino este compusă dintr-un microcontroler Atmel AVR de 8, 16 sau 32 biţi, precum şi componente complementare care facilitează programarea şi încorporarea în alte circuite.

Arduino Uno este o placa de dezvoltare bazată pe microcontrolerul ATmega328P, având 6 intrări analogice, 14 de intrări digitale/pini de ieşire (din care 6 pot fi utilizate ca ieşiri PWM), un oscilator cu quart de 20 MHz, o conexiune USB, o mufa de alimentare, şi un buton de resetare. [1]

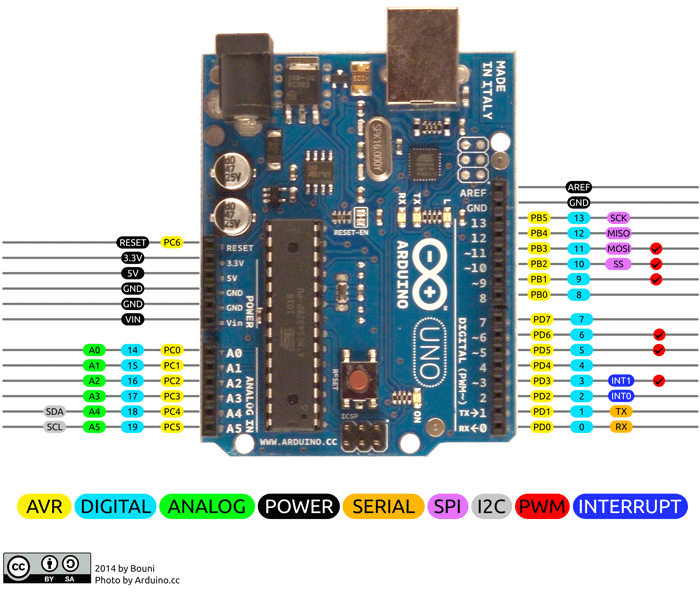
Particularități a microcontrolerului Atmega328p [1]:

|  |  |
| --- | --- |
| Frecvenţa | 20 MHz |
| Tensiunea de alimentare | 1,8 – 5,5 V |
| Număr de pini | 28 |
| Memorie SRAM | 2 kB |
| Memorie EEPROM | 1 kB |
| Memorie Flash | 32 kB |
| Cicluri scriere/citire | 10000 Flash/100000 EEPROM |
| Număr de pini intrare - ieşire | 23 |
| Canale conversie analog - digitală | 8 |
| Timere pe 8 biţi | 2 |
| Timer pe 16 biţi | 1 |
| Consum de energie | * Funcţionare 0,2 mA * Stand-bye 0,1 μA |

Schema bloc a microcontrolerului Atmega328p [2]:



Schema pinilor microcontrolerului Arduino [3]:



1. **Descrierea detaliată a modulelor implicate:**

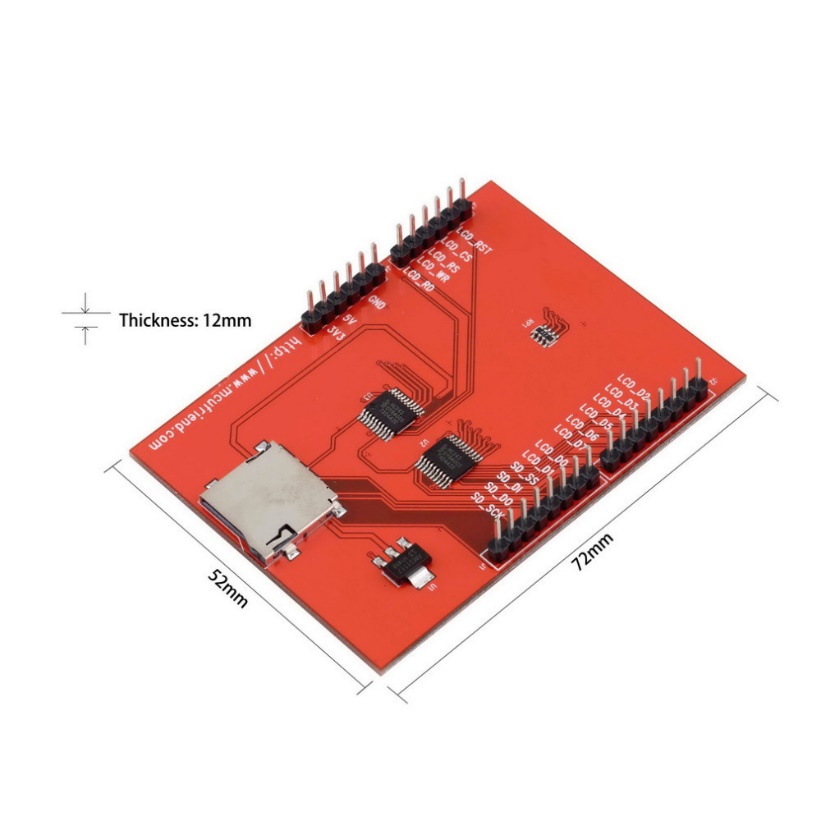
Module implicate:

* Display 2.4’’ TFT LCD Shield:

Caracteristici[4]:

* + - 2.4″ TFT LCD display
    - 240 x 320 rezoluție
    - Luminozitate : 4 LED-uri albe de iluminare, 18-bit 262000 culori  
      touchscreen rezistiv (Ok deget)
    - 8 biți interfață digitala, plus 4 linii de control
    - Pentru Arduino UNO R3 se utilizează pinii 5-13 si A0-A3
    - 71 x 52 x 7mm

Schemă pini[4]:



1. **Explicarea funcţionării sistemului de afişaj ales:**

Sistemul de afișare ales este un display 2.4’’ TFT LCD Shield. Acesta este și modalitatea de control a barei de joc de către utilizator prin intermediul touchscreen-ului modulului.

Modul de funcționare a jocului:

* + - Programul va începe cu 5 randuri de blocuri reprezentând zidurile ce trebuie distruse;

Fiecare bloc dintr-un zid are 3 „vieți”, adică, pentru a fi distrus și deci pentru a dispărea de pe ecran jucătorul va trebui să-l lovească de 3 ori cu ajutorul mingii. Jucătorul va fi reprezentat de o bară poziționată în centrul ultimului rând al ecranului, iar mingea va fi poziționată inițial pe mijlocul barei.

* + - Jucătorul și el are 3 vieți, de fiecare dată când nu „prinde” mingea va pierde o viață.
    - Jucătorul va mișca bara cu ajutorul degetului sau un creion stânga/dreapta pentru a prinde și pentru a trimite mingea înapoi catre blocuri.
    - Jocul se termină atunci când toate blocurile tutoror zidurilor au fost lovite, caz in care jucătorul a câștigat, sau se termină atunci când jucătorul a rămas fără nici o viață.

1. **Conectarea hardware:**



Modulul este conectat la pinii 5-13 și A0-A3.

1. **Programe:**
2. **Programul principal *arkanoid.ino***

// library SPFD5408

#include <SPFD5408\_Adafruit\_GFX.h> // Core graphics library

#include <SPFD5408\_Adafruit\_TFTLCD.h> // Hardware-specific library

#include <SPFD5408\_TouchScreen.h> // Touch library

// game files

#include <block.h>

#include <globals.h>

// Calibrates value

#define SENSIBILITY 300

#define MINPRESSURE 10

#define MAXPRESSURE 1000

#define XM A1 // must be an analog pin, use "An" notation!

#define YP A2 // must be an analog pin, use "An" notation!

#define XP 7 // can be a digital pin

#define YM 6 // can be a digital pin

//second

short TS\_MINX=153;

short TS\_MINY=923;

short TS\_MAXX=933;

short TS\_MAXY=223;

TouchScreen ts = TouchScreen(XP, YP, XM, YM, SENSIBILITY);

// LCD Pin

#define LCD\_CS A3

#define LCD\_CD A2

#define LCD\_WR A1

#define LCD\_RD A0

#define LCD\_RESET A4

// Init LCD

Adafruit\_TFTLCD tft(LCD\_CS, LCD\_CD, LCD\_WR, LCD\_RD, LCD\_RESET);

// lastTouch

TSPoint lastTouch;

void(\* resetFunc) (void) = 0;

//-- Setup

int frameStart, frameTime, frameCount;

int frameRate = 60;

int MS\_PER\_FRAME = 1000 / frameRate;

void setup(void) {

Serial.begin(9600);

tft.reset();

tft.begin(0x9341); //hardcoded due to idk

tft.setRotation(3); // Need for the Mega, please changed for your choice or rotation initial

width = tft.width() - 1;

height = tft.height() - 1;

// Border

drawBorder();

// Initial screen

tft.setCursor(65, 50);

tft.setTextSize (4);

tft.setTextColor(RED);

tft.println("Arkanoid");

tft.setCursor(70, 100);

tft.setTextSize (2);

tft.setTextColor(BLACK);

tft.println("Touch to start!");

tft.setCursor(150, 210);

tft.setTextSize (1);

tft.setTextColor(BLACK);

tft.println("Made by: Cazacu Robert");

tft.setCursor(203, 220);

tft.println("Mocanu Alexandra");

lastTouch = waitOneTouch();

drawGameBorder();

gameSetUp();

}

// -- Loop

void loop(0)

{

processInput();

preUpdateCleanup();

update();

render();

}

void update() {

if (life\_count <= 0)

resetFunc();

ball -> move();

}

TSPoint tsp;

void processInput() {

tsp = currentTouch();

if( abs(lastTouch.x - tsp.x) > 10 ) {

deleteOldBar();

lastTouch = tsp;

showTouched(lastTouch);

bar\_dx = bar\_x;

bar\_x = getNewBar();

bar\_dx = -(bar\_dx - bar\_x) / abs(bar\_dx - bar\_x);

} else {

bar\_dx = 0;

}

}

void preUpdateCleanup() {

deleteOldBall();

}

void render() {

for(uint32\_t i = 0; i < box\_count; i++)

if( box[i] -> needsRedraw && !box[i] -> dead)

box[i] -> draw(tft);

bar\_draw();

ball -> draw(tft);

drawLifes();

}

void drawLifes() {

tft.setCursor(10, 6);

tft.setTextSize (1);

tft.setTextColor(WHITE);

tft.println("Lifes: ");

tft.fillRect(50, 6, 5, 5, BLACK);

tft.setCursor(50, 6);

tft.println(life\_count);

}

void deleteOldBar() {

tft.fillRect(bar\_x, bar\_y, bar\_l, bar\_t, BLACK);

}

void deleteOldBall() {

tft.fillCircle(ball -> x, ball -> y, ball -> r, BLACK);

}

void bar\_draw() {

tft.fillRect(bar\_x, bar\_y, bar\_l, bar\_t, RED);

}

void gameSetUp() {

ball = new Ball();

box = new Block\*[box\_count];

for(uint32\_t st = 0; st < box\_count; st++) {

box[st] = new Block();

}

for(uint32\_t p = 0; p < box\_row; p++){

for(uint32\_t q = 0; q < box\_column; q++)

{

box[p\*box\_column + q]->x = offsetX + box[p\*box\_column + q]->l \* q;

box[p\*box\_column + q]->y = offsetY + box[p\*box\_column + q]->t \* p;

}

}

}

int getNewBar() {

return clamp( 5 - ((26 \* (lastTouch.y - 900)) / 69),

5,

264);

}

int clamp (int x, int left, int right) {

if (x < left) x = left;

if (x > right) x = right;

return x;

}

TSPoint currentTouch() {

TSPoint p = ts.getPoint();

pinMode(XM, OUTPUT); //Pins configures again for TFT control

pinMode(YP, OUTPUT);

if ((p.z > MINPRESSURE ) && (p.z < MAXPRESSURE))

return p;

else

return lastTouch;

}

// wait 1 touch to return the point

TSPoint waitOneTouch() {

TSPoint p;

do {

p= ts.getPoint();

pinMode(XM, OUTPUT); //Pins configures again for TFT control

pinMode(YP, OUTPUT);

} while((p.z < MINPRESSURE )|| (p.z > MAXPRESSURE));

return p;

}

void drawGameBorder () {

uint16\_t width = tft.width() - 1;

uint16\_t height = tft.height() - 1;

uint8\_t border = 5;

tft.fillScreen(RED);

tft.fillRect(border, border, (width - border \* 2), (height - border + 1), BLACK);

}

// Draw a border

void drawBorder () {

uint16\_t width = tft.width() - 1;

uint16\_t height = tft.height() - 1;

uint8\_t border = 10;

tft.fillScreen(RED);

tft.fillRect(border, border, (width - border \* 2), (height - border \* 2), WHITE);

}

// Show the coordinates

void showTouched(TSPoint p) {

uint8\_t w = 40; // Width

uint8\_t h = 10; // Heigth

uint8\_t x = (width - (w\*2)); // X

uint8\_t y = 11; // Y

tft.fillRect(x, y, w\*2, h, WHITE); // For cleanup

tft.drawRect(x, y, w, h, RED); // For X

tft.drawRect(x+w+2, y, w\*2, h, RED); // For Y

tft.setTextColor(BLACK);

tft.setCursor(x+2, y + 1);

tft.print("X: ");

showValue(p.x);

tft.setCursor(x+2+w+2, y + 1);

tft.print("Y: ");

showValue(p.y);

}

// Show a value of TSPoint

void showValue (uint16\_t value) {

if (value < 10)

tft.print("00");

if (value < 100)

tft.print("0");

tft.print(value);

}

// Map the coordinate X

uint16\_t mapXValue(TSPoint p) {

uint16\_t x = map(p.x, TS\_MINX, TS\_MAXX, 0, tft.width());

return x;

}

// Map the coordinate Y

uint16\_t mapYValue(TSPoint p) {

uint16\_t y = map(p.y, TS\_MINY, TS\_MAXY, 0, tft.height());

return y;

}

1. **Fișierul *ball.cpp* ce descrie obiectul *ball* (mingea):**

#include "ball.h"

#include "globals.h"

Ball::Ball(){

this -> x = width / 2;

this -> y = height \* 3 / 4;

this -> e = 1;

this -> r = 5;

this -> vx = game\_speed;

this -> ball\_speed = game\_speed;

this -> vy = this -> ball\_speed;

this -> add\_vx = 0.2;

this -> mybox = box;

}

void Ball::move(){

this -> x = this -> x + this -> vx;

this -> y = this -> y + this -> vy;

// left of screen

if(this -> x < (r/3 + offsetX + 1)) {

this -> vx = -e\*this -> vx;

this -> x = r/3 + offsetX + 1;

}

// right of screen

if(this -> x > (width - r/3 - offsetX - 1)){

this -> vx = -e\*this -> vx;

this -> x = width - r/3 - offsetX - 1;

}

//Bottom - you died son

if(this -> y > height - r/3){

life\_count -= 1;

this -> x = width / 2;

this -> y = height \* 3 / 4;

this -> vx = ball\_speed;

this -> vy = ball\_speed;

}

// top of screen

if(this -> y < (r/3 + offsetY + 1 )){

this -> vy = -e\*this -> vy;

this -> y = r/3 + offsetY + 1 ;

}

//bar impact

float bar\_distance = this->x - (bar\_x + bar\_l / 2);

bar\_distance = abs(bar\_distance);

if ( bar\_distance < (bar\_l / 2 + r)) { //if on bar

if ( this->y > (bar\_y - r) ) { //if low enough

this -> y = bar\_y - this -> r;

this -> vy = - (this -> vy);

this -> vx = this -> vx + ( this -> add\_vx \* bar\_dx );

}

}

//boxes impact

int box\_distance;

int y\_distance;

for (int i = 0; i < box\_count; ++i) {

if (box[i] -> dead)

continue;

//from below box

box\_distance = abs(this -> x - (box[i] -> x + box[i] -> l / 2));

if ( box\_distance < (box[i] -> l / 2 + r / 2)) { // ball has same x as box

y\_distance = this -> y - (box[i] -> y + (box[i] -> t / 2));

//check if close enough, if yes -> we got a hit

if ( abs(y\_distance) <= (box[i] -> t / 2 + r)) {

box[i] -> hit();

// from below

if (y\_distance <= (box[i] -> t / 2 + r + 1) && abs(y\_distance) > (box[i] -> t / 2)) {

this -> vy = -e \* this->vy;

}

//from above

else if ( y\_distance >= (box[i] -> t / 2 - r - 1) && abs(y\_distance) > (box[i] -> t / 2)) {

this -> vy = -e \* this->vy;

}

// from aside

else if ( abs(y\_distance) <= (box[i] -> t / 2)) {

this -> vx = -e \* this -> vx;

}

}

}

}

}

void Ball::draw(Adafruit\_TFTLCD &tft){

tft.fillCircle(this -> x, this -> y, this -> r, WHITE);

}

1. **Fișierul *block.cpp* ce descrie obiectul *block* (bloc din cadrul unui zid):**

#include "block.h"

#include "globals.h"

Block::Block() {

this->x = 0;

this->y = 0;

this->contact = 0;

this->interval = 1;

this->l = (width - 2\*offsetX) / box\_column;

this->t = (height - offsetY) / 3 / box\_row;

this->needsRedraw = true;

this->dead=false;

}

void Block::draw(Adafruit\_TFTLCD &tft) {

if(this->contact == 0)

this->color = MAGENTA;

else if(this->contact == 1)

this->color = BLUE;

else if(this->contact == 2)

this->color = GREEN;

tft.fillRect(this->x + this->interval, this->y + this->interval, this->l - 2\*this->interval, this->t - 2\*this->interval, this->color);

// tft.drawRect(this->x + this->interval, this->y + this->interval, this->l - 2\*this->interval, this->t - 2\*this->interval, this->color);

if(this->contact > 2){

tft.fillRect(this->x + this->interval, this->y + this->interval, this->l - 2\*this->interval, this->t - 2\*this->interval, BLACK);

this->dead = true;

return;

}

this -> needsRedraw = false;

}

void Block::hit() {

this -> contact++;

this -> needsRedraw = true;

}

1. **Fișierul *globals.cpp* ce descrie valorile inițiale ale unor variabile globale:**

#include "globals.h"

extern int set\_no = 1;

extern int i = 0;

extern int game\_speed = 1;

extern int start\_key = 0;

extern int life\_count = 3;

extern int p = 0;

extern float p\_x = 0;

extern float d\_x = 0;

extern float bar\_y = 225;

extern float bar\_l = 50;

extern float bar\_x = 100 - bar\_l/2;

extern float bar\_t = bar\_l/5;

extern float bar\_v = 15;

extern float bar\_dx = 0;

// Dimensions

extern int width = 0;

extern int height = 0;

extern int offsetX = 10;

extern int offsetY = 15;

extern Ball\* ball = nullptr;

extern Block\*\* box = nullptr;

extern int box\_row = 5;

extern int box\_column = 5;

extern int box\_count = box\_row \* box\_column;

1. **Fișiere Header:**

**5.1 *ball.h***

#ifndef BALL\_H

#define BALL\_H

#include <SPFD5408\_Adafruit\_GFX.h> // Core graphics library

#include <SPFD5408\_Adafruit\_TFTLCD.h> // Hardware-specific library

#include "block.h"

class Ball {

public:

float x, y, e, r, vx, vy, box\_l, box\_t, ball\_speed, add\_vx;

Block\*\* mybox;

Ball();

void draw(Adafruit\_TFTLCD &tft);

void move();

// void swap(float x, float y); //TODO: fix this, make it globalsmh

};

#endif /\* BALL\_H \*/

**5.2 *block.h***

#ifndef BLOCK\_H

#define BLOCK\_H

#include <SPFD5408\_Adafruit\_GFX.h> // Core graphics library

#include <SPFD5408\_Adafruit\_TFTLCD.h> // Hardware-specific library

class Block {

public:

float x, y, l, t, contact, interval;

int index, color;

bool needsRedraw, dead;

Block();

void draw(Adafruit\_TFTLCD &tft);

void hit();

};

#endif /\* BLOCK\_H \*/

**5.3 *global.h***

#pragma once

#ifndef GLOBAL\_H

#define GLOBAL\_H

#include "ball.h"

#include "block.h"

extern int set\_no;

extern int i;

extern int game\_speed;

extern int start\_key;

extern int life\_count;

extern int p;

extern float p\_x;

extern float d\_x;

extern float bar\_y;

extern float bar\_l;

extern float bar\_x;

extern float bar\_t;

extern float bar\_v;

extern float bar\_dx;

// Dimensions

extern int width;

extern int height;

extern int offsetX;

extern int offsetY;

extern Ball\* ball;

extern Block\*\* box;

extern int box\_row;

extern int box\_column;

extern int box\_count;

// Assign human-readable names to some common 16-bit color values:

#define BLACK 0x0000

#define BLUE 0x001F

#define RED 0xF800

#define GREEN 0x07E0

#define CYAN 0x07FF

#define MAGENTA 0xF81F

#define YELLOW 0xFFE0

#define WHITE 0xFFFF

#endif /\* GLOBAL\_H \*/

1. **Bibliografie:**

[1] <http://retele.elth.ucv.ro/Bratu%20Cristian/MAP/004%20-%20Curs%20004%20-%20MAP%20-%20Arduino.pdf>

[2] <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>

[3] <https://components101.com/microcontrollers/arduino-uno>

[4] <http://roboromania.ro/produs/display-2-4-tft-lcd-shield-compatibil-arduino/>

1. **Link proiect Github:**

***https://github.com/AlexandraMocanu/proiectSI***