UNIVERSITATEA POLITEHNICA TIMIŞOARA

Facultatea de Automatică și Calculatoare

Secția: Calculatoare și Tehnologia Informației

**SISTEME ÎNCORPORATE**

**Monitorizarea numărului de locuri libere într-o parcare cu barieră**

Ciulu Miruna Dalila

Cosma Patricia-Giorgiana

Damian Melisa-Dana

An III, AC, CTI, gr. 2

Anul universitar

2017 – 2018

Tema proiectului

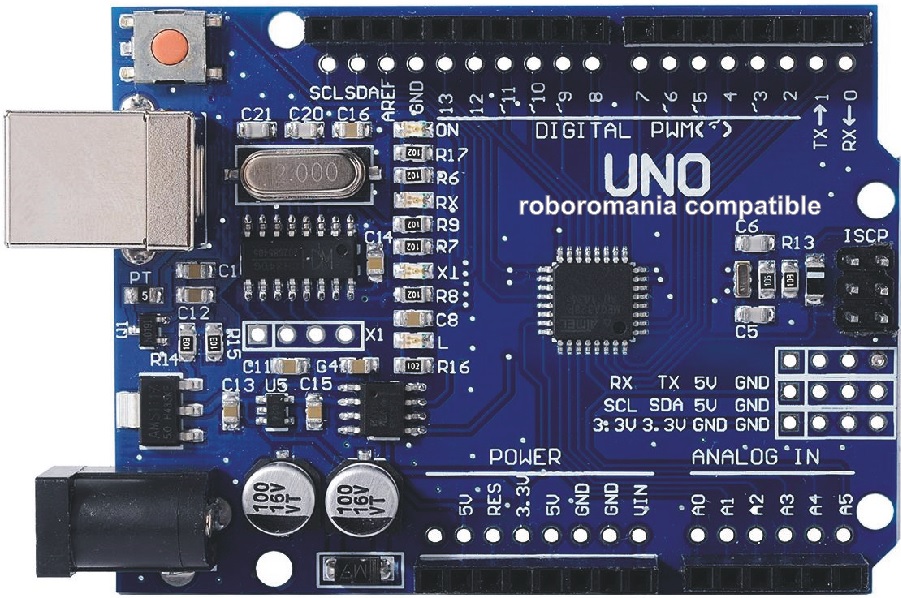
Caracteristici:

* Se vor utiliza fotodiode sau senzori de detecţie a obiectelor reflectorizante („reflective object sensors”) pentru detectarea maşinilor care intră si ies. Se pot folosi alte tipuri de senzori conform cu decizia proiectantului.
* Modalitatea de legare a senzorilor va cuprinde cât mai puţine fire (se vor alege interfeţe seriale, precum I2C, CAN, LIN etc).
* Se va realiza o machetă a parcării, cu o barieră la intrare, care va fi comandată de către microcontroler cu ajutorul unui motoraş. La fiecare maşină care intră în parcare, bariera se va ridica şi se va lăsa la loc.
* Se va asigura afişarea numărului de locuri libere din parcare (afişaj cu segmente).
* Numărul iniţial al locurilor libere de parcare va fi prestabilit.
* Aplicaţia va dispune de 2 LED-uri: unul din ele va fi pornit atâta vreme cât mai există locuri libere, al doilea va fi pornit când nu vor mai fi locuri disponibile în cadrul parcării. Nu este posibilă pornirea concomitentă a celor două LED-uri.
* Codul sursă va trebui să ţină cont în redactare de constrângerile specifice care pot apărea în cadrul unui sistem încorporat.

**Placa de dezvoltare Arduino UNO R3**

Placa de dezvoltare este compatibilă cu Arduino. Aceasta este o versiune de UNO, cu o interfață USB noua și îmbunătățită. Ca și UNO, are slot de expansiune cu suport pentru 3,3 V, pin de RESET și un sistem de selecție automat pentru sursa de alimentare USB sau DC. UNO este compatibil din perspectiva codului și a formatului cu Duemilanove, Diecimilla și alte versiuni mai vechi de Arduino, deci toate librăriile și modulele vor funcționa adecvat. UNO R3 oferă în plus o concesiune USB îmbunătățită, pini în plus pentru suportul I2C și IORef.

Hardware-ul constă dintr-o platformă de procesare open-source cu un microcontroler Atmel AVR pe 8 biţi sau un Atmel ARM pe 32 de biţi.



**Specificaţii hardware:**

• Microcontroler: ATmega328  
• Voltaj de operare: 5V  
• Voltaj de intrare (recomandat): 7-10 V

• Limitele voltajului de intrare: 6-20 V  
• 14 pini de intrare/ieșire digitali (cu suport pentru PWM)  
• 6 intrări analogice  
• Curent continuu per pin de intrare/ieșire: 40 mA

• Curent continuu per pin de 3.3V: 50 mA  
• Memorie Flash: 32 KB  
• SRAM: 2 KB  
• EEPROM: 1 KB  
• Frecvenţă de funcţionare: 16 MHz

**Descriere pini**

Există **14 pini digitali** de intrare / ieşire (I/O sau input/output). Aceştia operează la o tensiune de 5V. Fiecare pin poate primii sau trimite o intensitate de maxim 40 mA şi au o rezistenţă internă între 20-50 kOhmi (implicit deconectată). În afară de semnalul standard I/O, unii dintre pini mai au şi alte funcţii specializate:

* 0 (serial) **RX** – pin serial, utilizat în special pentru recepţia datelor seriale asincrone. Protocolul serial este de tip UART.
* 1 (serial) **TX** – pin serial, utilizat pentru trimiterea datelor asincrone
* 2,3 (**External Interrupts**) - întrerupere externă
* 3,5,6,9,10,11 (**PWM**) - toţi pinii marcaţi cu semnul ~ pot fi folosiţi şi pentru PWM ([pulse with modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation)) pe 8 biti
* 10 (**SPI**) – suportă comunicare prin interfaţa serială (**[Serial Peripheral Interface](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus)**). SPI-ul are 4 semnale logice specifice iar acest pin se foloseste pentru **SS** – Slave Select (activ pe nivel 0; ieşire din master).
* 11 (**SPI**) – suportă SPI, iar acest pin se foloseşte pentru **MOSI**/SIMO – Master Output, Slave Input
* 12 (**SPI**) – suportă SPI, iar acest pin se foloseşte pentru **MISO**/SOMI – Master Input, Slave Output
* 13 (**SPI**) – suportă SPI, iar acest pin se foloseşte pentru SCK/SCLK – Ceas serial
* 13(**LED**) - pe placă este încorporat un LED. Când pinul este setat pe valoarea HIGH este pornit, când are valoarea LOW este oprit.
* 14 (**GND**) – masa
* 15 (**AREF**) – tensiunea de referinţă pentru intrările analogice
* 16 (**SDA**) – comunicare I2S
* 17 (**SCL**) – comunicare I2S

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.

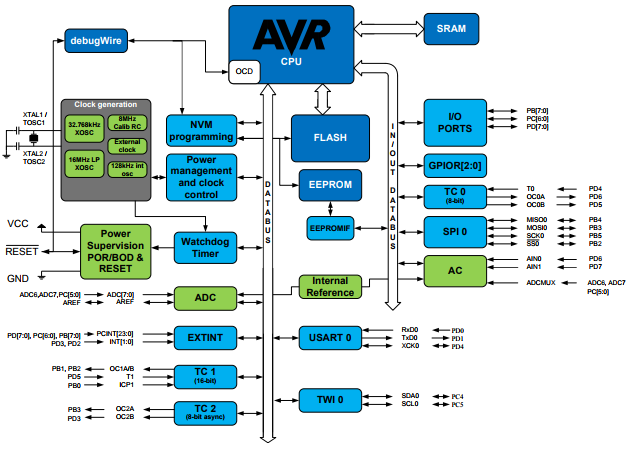
Pe lângă pinii analogici mai există o secţiune de pini notată **POWER**.

* 1**- Vin** – intrarea pentru tensiune din sursă externă (input Voltage)
* 2,3 **- GND** – negativ
* 4 **- 5V** – ieşire pentru piesele şi componentele montate la Arduino
* 5 **- 3,3V** – ieşire pentru piesele şi senzorii care se alimentează la această tensiune.
* 6 **- RESET** – se poate reseta controlerul de la Arduino
* 7 **-** **5VREF** – 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 Reference Voltage)
* 8**- pin** **neconectat**, este rezervat pentru utilizări ulterioare (la reviziile următoare ale plăcii probabil)

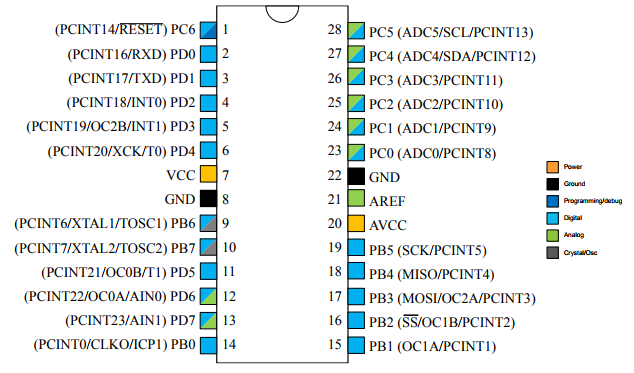
Comunicarea cu calculatorul, altă placă Arduino sau alte microcontrolere se poate realiza fie prin portul **USB**, fie prin pinii 0 şi 1 (**RX** şi **TX**) care facilitează comunicarea serială **UART TTL**.

După cum vedeţi în imagine, în partea dreapta, placa mai are o serie de pini marcaţi **ICSP** ([**In-Circuit Serial Programming**](http://en.wikipedia.org/wiki/In_Circuit_Serial_Programming_%28ICSP%29)), care pot fi folosiţi pentru [reprogramarea microcontrolerului](http://www.instructables.com/id/Using-an-Arduino-board-as-an-ISP-via-the-ICSP-head/).

**Schema bloc a microcontrolerului ATmega328**

****

**Configuraţia pinilor**

****

**Particularităţi ale microcontrolerului ATmega328**

**ATmega328** este un [cip](https://ro.wikipedia.org/wiki/Circuit_integrat) [microcontroler](https://ro.wikipedia.org/wiki/Microcontroler) creat de către [Atmel](https://ro.wikipedia.org/w/index.php?title=Atmel&action=edit&redlink=1" \o "Atmel — pagină inexistentă) și face parte din seria [megaAVR](https://ro.wikipedia.org/w/index.php?title=MegaAVR&action=edit&redlink=1).

Microcontrolerul Atmel pe 8 biți AVR RISC combină memoria ISP flash de 32 kB cu capabilități de scriere în timp real, 1 kB EEPROM, 2 kB SRAM, 23 linii intrare/ieşire de uz general, 32 registre de lucru cu scop general, contoare cu moduri de comparare, întreruperi interne și externe, programare USART serial, interfață serială orientată pe două fire, port serial SPI, convertor A/D cu 6 canale de 10 biți (8 canale în pachete TQFP și QFN/MLF), cronometru programabil de supraveghere cu oscilator intern și cinci moduri de economisire a energiei selectabile.

Dispozitivul funcționează între 1,8 și 5,5 V. Acesta atinge o viteză de transfer apropiată de 1 MIPS per MHz.

ATmega328 este utilizat, în mod obișnuit, în multe proiecte și în sisteme autonome unde este nevoie de un microcontroler simplu, cu putere redusă și cu costuri reduse. Cea mai comună implementare a acestui cip este pe platforma populară de dezvoltare Arduino, și anume modelele Arduino Uno și Arduino Nano.

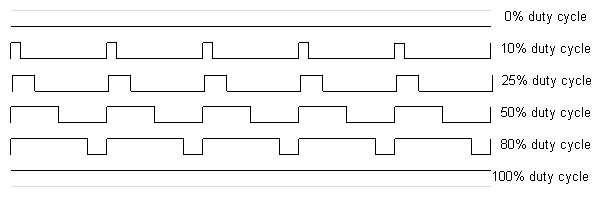
**PWM**

Termenul de PWM provine din limba engleză de la Pulse Width Modulation ceea ce înseamnă Modulația Pulsurilor în Lungime și este o tehnică de a simula un semnal analogic, folosindu-ne de un semnal digital, adică de a varia în mod controlat tensiunea dată unui dispozitiv electronic.

Semnalul digital poate lua doar 2 stari: ON sau OFF ; „1“ sau „0“ ; 5V sau 0V. Perioada de timp corespunzătoare valorii ON dintr-un ciclu ON-OFF se numește factor de umplere (duty cycle) și reprezintă, în medie, ce tensiune va primi dispozitivul electronic. Un semnal PWM ajută la obținerea unei mulțimi de valori cuprinse între 5V și 0V. De exemplu, se pot obține 2.5V.

**Duty\_Cycle = [ON\_time / (ON\_time + OFF\_time) ] \* 100**

**Output\_Voltage = Duty\_Cycle \* Input\_Voltage**



Semnalele PWM sunt utile pentru a varia viteza de rotație a unui motor sau a controla unghiul unui servomotor (În cazul unui motor, căruia i se aplică un semnal PWM cu factor de umplere de 0%, viteza de rotație a acestuia va fi egală cu 0 rpm. Un factor de umplere de 100% va duce la o turație maximă a acestuia), dar și pentru a controla intensitatea luminoasă a unui led, la producerea sunetelor de frecvențe diferite.

O placa Arduino UNO poate genera pana la 6 semnale PWM prin pinii 3, 5, 6, 9, 10 și 11, deoarece microcontroller-ul este echipat cu un modul hardware special care se ocupă exclusiv de aceste semnale.

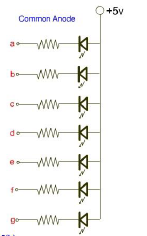
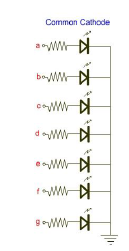
Lucrul cu PWM-ul presupune inițializarea unui timer și configurarea output-ului pe pini. Fiecare timer are 3 moduri de lucru: Fast PWM, Phase Corrected PWM, Phase and Frequency Phase Corrected PWM, acesta din urmă fiind recomandat pentru controlul unui motor:

**PWM\_frequency = clock\_speed / (2 \* Prescaller\_value \* TOP\_value).**

**Funcționarea afișajului cu 7 segmente**

Disponibile în diverse mărimi, formate și culori, afișajele cu 7 segmente (LED-uri) sunt ușor de folosit. Acestea sunt disponibile ca module individuale, precum și mai multe într-un singur modul. În mod tipic, cele 7 segmente care formează afișajul sunt identificate dupa 7 litere (eventual și un punct pentru zecimale daca este cazul) dispuse ca in Fig.1.

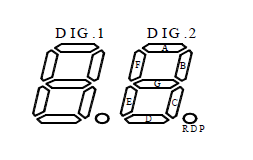
Cele 7 (sau 8) LED-uri din interiorul unui astfel de afișaj au câte un terminal scos afară individual pentru fiecare LED, restul terminalelor fiind unite între ele la un singur pin ce iese în afară. În funcție de modul de conectare putem avea anod comun sau catod comun. Afișajele cu **catod comun** au nevoie pentru aprindere de 1 logic pentru segmente, catodul fiind legat la masă și 1 logic pentru selecție. Afișajele cu **anodul comun** au nevoie de 0 logic pentru aprindere, anodul fiind legat la plus și 0 logic pentru selecție. În proiectul nostru, am ales varianta cu anod comun (rezistente de 220 ohmi).

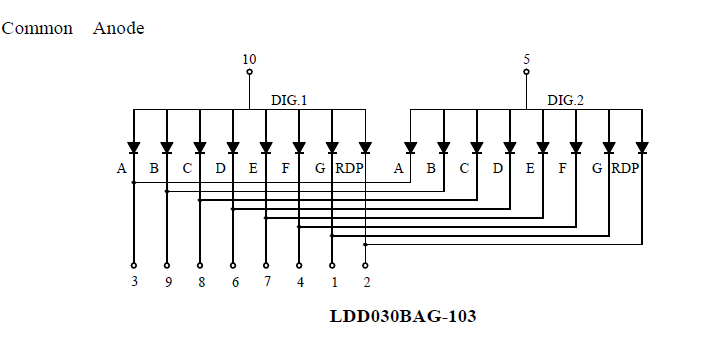


Afișarea pe mai multe segmente a unor valori diverse în același timp, teoretic, nu este posibilă, deoarece, la un moment dat, doar unul dintre module poate fi selectat pentru afișare. Practic însă, dacă rata de refresh este destul de mare (în jur de 50 rate pe secundă ~ 20 ms) ochiul uman nu percepe pâlpâirea, astfel încât valorile se văd în același timp.

Pași:

* Se configureaza ca ieșiri toți pinii legați la toate modulele.
* Se baleiază fiecare segment pe rând, într-un ciclu.
* Pentru fiecare segment se validează pinul de selecție (se pune pe 1 logic).
* Se invalidează pinul de selecție pentru celelalte module.
* Se trimite valoarea care se vrea afișată la portul respectiv.
* Se face o întârziere de 10 ms (daca avem de afișat pe două segmente -20 ms).
* Se trece la următorul segment.

Afișajul ales pentru proiect este LDD030BAG-103, un afișaj cu 7 segmente si 2 digits, cu anod comun.

**Schema internă** (cum este prezentată în documentația atașată)

**Modul Senzor cu fotodiodă de detectare a**

**nivelului de lumină**

Senzorul ales pentru proiect este LM393, un senzor cu fotodiodă pentru detectarea nivelului de lumină.

Acest modul de senzor de lumină combină modelul KY-018 și modelul LM393 cu 3 pini pentru a crea un modul universal care poate funcționa ca un comutator digital, dacă este conectat la pinul DO, sau ca senzor analogic dacă este conectat la AO (ieșire analogică). În plus, senzorul se poate calibra utilizând potențiometrul, pentru a regla sensibilitatea de declanșare sau pentru a schimba spectrul de sensibilitate spre niveluri întunecate sau luminoase.

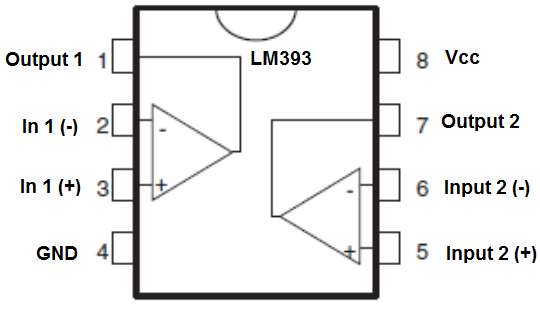
Când există lumină, rezistența LDR va scădea în funcție de intensitatea luminii. Cu cât intensitatea luminii este mai mare, cu atât rezistența LDR este mai mică. Senzorul are un buton de potențiometru care poate fi reglat pentru a schimba sensibilitatea LDR spre lumină.

**Funcționare:**

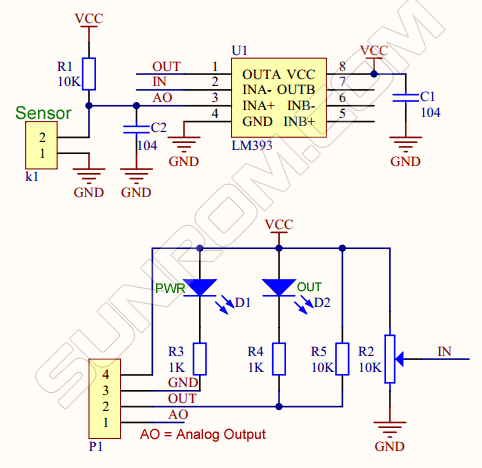
* Modulul se ”uită” la nivelul de lumină care ajunge la senzor și îl transmite la ieșirea analogică;
* Valoarea citită este comparată cu valoarea presetată de utilizator cu ajutorul potențiometrului;
* Dacă nivelul luminii este mai mare (lumină mai puternică) decât valoarea setată, ieșirea digitală devine 0 logic.

**Caracteristici:**

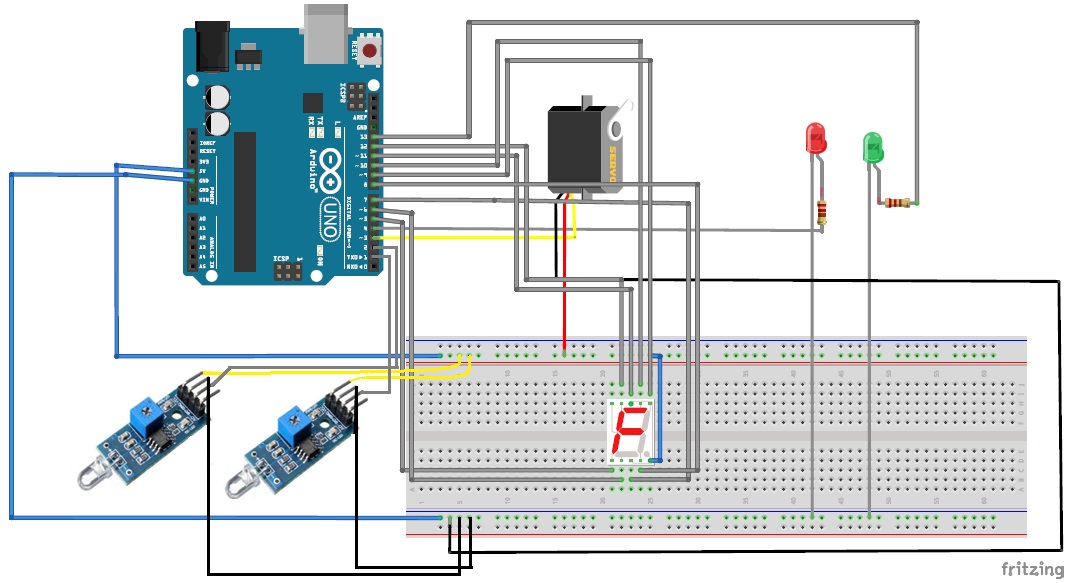
* Tensiunea de operare 3,3V-5V
* Curent de operare: 15 mA
* Ieșire digitală DO – 0 corespunde luminii puternice, 1 luminii slabe
* Ieșire analogică AO – ia valori între 0 și 1023 (10 biți); valoare mica la lumina intensă
* 2 Led-uri - indică dacă senzorul este pornit și ieșirea
* Potențiometru
* Comparator LM393



**Schema bloc:**



**Conectarea hardware**



**Descriere:**

* Pinul 1 – ieșirea digitală a diodei folosite la ieșirea din parcare
* Pinul 2 – ieșirea digitală a diodei folosite la intrarea în parcare
* Pinul 3 – servomotor
* Pinul 4 – led-ul roșu
* Pinii 5-12 – segmente afișaj
* Pinul 13 – led-ul verde

Între pini si led-uri a fost introdusă câte o rezistență de 220 Ohmi pentru a reduce intensitatea curentului sub maximul admis de led (22,7 < 25 mA).

Fotodiodele și servomotorul sunt conectate la VCC și la GND. Afișajul fiind cu anod comun e conectat la VCC. Led-urile sunt conectate la GND.

**Program**

#include <Servo.h> // includere bibliotecă Servo

int servoPin = 3; // declarare pin servomotor

Servo Servo1; // creare obiect Servo

int DIODE1 = 2; // declarare pini fotodiode – dioda intrare

int DIODE2 = 1; // dioda ieşire

int RED\_LED = 4; // declarare pin led rosu

int GREEN\_LED = 13; // declarare pin led verde

void setup() {

pinMode(RED\_LED, OUTPUT); // definire led-uri

pinMode(GREEN\_LED,OUTPUT);

digitalWrite(GREEN\_LED, HIGH); // iniţializare led-uri: led verde aprins

digitalWrite(RED\_LED,LOW); // led roşu stins

Servo1.attach(servoPin); // ataşare servomotor la pinul corespunzător

Servo1.write(0); // poziţia iniţială a servomotorului la 0 grade

pinMode(DIODE1, INPUT); // definire fotodiode

pinMode(DIODE2, INPUT);

digitalWrite(DIODE1,HIGH); // iniţializare diode

digitalWrite(DIODE2,HIGH);

pinMode(5,OUTPUT); // definire afişaj cu 7 segmente

pinMode(6,OUTPUT);

pinMode(7,OUTPUT);

pinMode(8,OUTPUT);

pinMode(9,OUTPUT);

pinMode(10,OUTPUT);

pinMode(11,OUTPUT);

pinMode(12,OUTPUT);

digitalWrite(5,HIGH); // stingerea tuturor segmentelor

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

digitalWrite(8,HIGH);

digitalWrite(9,HIGH);

digitalWrite(10,HIGH);

digitalWrite(11,HIGH);

digitalWrite(12,HIGH);

}

const int number[10] = {0b00000011, 0b00111111, 0b01001010, 0b00101010, 0b00110110, 0b10100010, 0b10000010, 0b00111011, 0b00000010, 0b00100010}; // configuraţiile cifrelor 0-9 (0 – aprins, 1 - stins)

int parkingSpaces = 2; // număr iniţial de locuri libere

int nrMaxParkSpaces = 6; // număr maxim de locuri

void display\_freeParkingSpaces(int x){

int pin,b;

for(pin = 5, b = 0; pin <=12 ; pin++, b++){

digitalWrite(pin, bitRead(number[x], b));

}

}

int diode1\_val1=0, diode1\_val2=0; // diodă intrare

int diode2\_val1=0, diode2\_val2=0; // diodă ieşire

void control\_parking\_lot(){

diode1\_val1 = digitalRead(DIODE1); // citire 2 valori consecutive

diode1\_val2 = digitalRead(DIODE1);

if (diode1\_val2 - diode1\_val1 == 1 && parkingSpaces){

Servo1.write(90); // servomotorul ridică bariera la 90 grade

delay(4000); // bariera e ridicată timp de 4 secunde

Servo1.write(0); // servomotorul coboară bariera la 0 grade

parkingSpaces--; // decrementare număr de locuri libere

display\_freeParkingSpaces(parkingSpaces);

if(!parkingSpaces){ // dacă nu mai există locuri libere

digitalWrite(GREEN\_LED, LOW); // led-ul verde se stinge

digitalWrite(RED\_LED, HIGH); // led-ul roşu se aprinde

}

}

diode2\_val1 = digitalRead(DIODE2); // analog pentru dioda de ieşire

diode2\_val2 = digitalRead(DIODE2);

if (diode2\_val2 - diode2\_val1 == 1){

Servo1.write(90);

delay(4000);

Servo1.write(0);

if(parkingSpaces < nrMaxParkSpaces)

parkingSpaces++;

display\_freeParkingSpaces(parkingSpaces);

if(parkingSpaces){

digitalWrite(GREEN\_LED, HIGH); // led-ul verde se aprinde

digitalWrite(RED\_LED, LOW); // led-ul roşu se stinge

}

}

}

void loop(){

display\_freeParkingSpaces(parkingSpaces);

control\_parking\_lot();

}

**Bibliografie**

**Hardware:**

<http://roboromania.ro/produs/afisaj-2-digiti-led-display-verde-cu-anod-comun/>

<http://www.etc.ugal.ro/cchiculita/students/pam/Afisaj7Seg.pdf>

<https://sites.google.com/site/qeewiki/books/avr-guide/pwm-on-the-atmega328>

<http://www.robofun.ro/docs/curs/64_600f0173-a7ca-47d0-a7e8-7244f6f95991/Arduino-SoftwarePWM.pdf>

<http://cs.curs.pub.ro/wiki/pm/lab/lab3>

<https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM>

<http://roboromania.ro/produs/placa-de-dezvoltare-uno-r3-placa-de-dezvoltare-compatibila-arduino/>

<https://en.wikipedia.org/wiki/ATmega328>

<http://www.notenoughtech.com/raspberry-pi/rpi-sensors/light-sensor-lm393-ky018/>

http://www.instructables.com/id/LDR-Sensor-Module-Users-Manual-V10/

<https://www.sunrom.com/p/light-sensor-module-photodiode>

**Software:**

<http://www.instructables.com/id/Arduino-Servo-Motors/>

<http://www.instructables.com/id/Using-IR-Sensor-TCRT-5000-With-Arduino-and-Program/>

<https://arduino-info.wikispaces.com/Brick-LightSensor-Analog-Digital>

<https://defendtheplanet.net/2016/01/01/5v-light-detector-analog-digital-flying-fish-mh-sensor-series/>

**Datasheets:**

**datasheet microcontroler Atmel** <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>

**datasheet afișaj cu 7 segmente** <https://www.tme.eu/ro/Document/0eb377cc0087a59474afac5c7c87fae0/LDD030BAG-103.pdf>

**datasheet comparator fotodiodă LM393**

<https://www.onsemi.com/pub/Collateral/LM393-D.PDF>

**Link GitHub:**

https://github.com/cosmapatricia/Arduino-Parking-Lot-with-Barrier.git