**Universitatea Politehnica din Timișoara**

**Facultatea de Automatică și Calculatoare**

**Sisteme încorporate**

**Aplicația “Automotive 2”: Facilitarea manevrelor de parcare a unui automobil: măsurarea distanței rămase până la un obstacol folosind un microcontroler și senzori de distanță.**

*Pocol Bogdan-Adrian*

*anul III, subgrupa 5.1*

*Raț Alexandru*

*anul III, subgrupa 5.1*

**Anul universitar**

**2018-2019**

**2. Aplicaţia „Automotive 2”: Facilitarea manevrelor de parcare a unui automobil:**

**măsurarea distanţei rămase până la un obstacol folosind un microcontroler şi**

**senzori de distanţă (2 studenți)**

Caracteristici:

• Senzorii de distanţă se pot lega la o interfaţă serială, precum CAN, I2C sau SPI, la

alegerea proiectantului;

• Se va măsura o distanţă cuprinsă între 10 şi 70 – 100 cm;

• Măsurarea distanţei va fi însoţită de o alarmă sonoră: frecvenţa sunetului va creşte pe

masură ce distanţa până la obstacol scade;

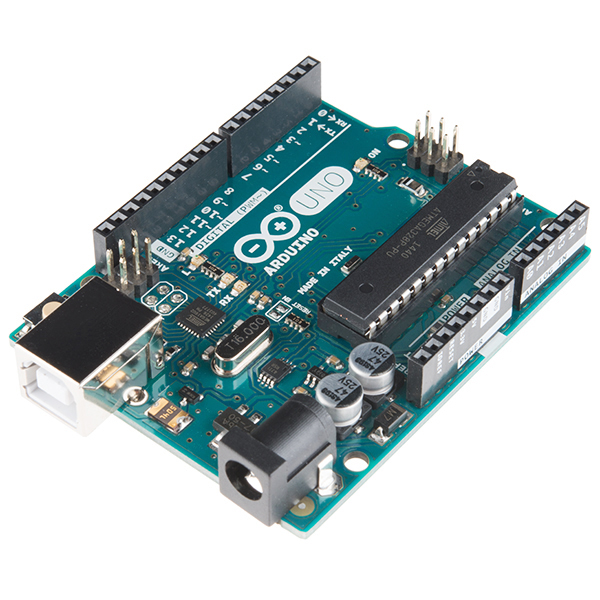
• Valoarea distanţei măsurate va fi afişată, la alegerea studentului, pe afişaje cu 7 segmente

sau matrice de LED-uri sau afişaj LCD;

• Se vor utiliza cel puțin 2 senzori de distanță, iar aplicația va avea un prototip practic.

**Echipament necesar**

Se va folosi o placă de dezvoltare Arduino Uno R3, un difuzor piezoelectric TDK PS1240P02BT, un senzor ultrasonic de distanță Elecfreaks HC-SR04, un Breadboard de dimensiuni 82x52x10mm și un modul LCD Tinsharp TC1602A-01T.

**Placa de dezvoltare Arduino Uno R3**

Placa este bazată pe un microcontroller ATmega328P, are 14 pini de intrare/ieșire digitală (6 dintre aceștia putând fi folosiți ca ieșiri PWM), 6 intrări analogice, un cristal de cuarț de 16 MHz, o conexiune USB, un port de alimentare, un circuit ICSP, și un buton pentru resetare.

Specificații:

* Microcontroller: ATmega328P
* Tensiune de lucru: 5V
* Tensiune de intrare: 6-20V
* Pini I/O digitali: 14 (6 ieșiri PWM)
* Intrări analogice: 6
* Curent per pin I/O: 20 mA
* Curent per pin 3.3V: 50 mA
* Memorie flash: 32 KB (ATmega328P), 0.5 KB pentru bootloader
* SRAM: 2 KB (ATmega328P)
* EEPROM: 1 KB (ATmega328P)
* Frecvență: 16 MHz

**Funcționalități folosite**: timer integrat, porturi digitale de intrare/ieșire.

**Timer integrat**

Uno are trei numărătoare, numite timer0, timer1 și timer2. Fiecare dintre numărătoare are un contor care este incrementat la fiecare ciclu de tact al timer-ului. Întreruperile **Clear Timer on Compare (CTC)** ale timer-ului sunt declanșate atunci când numărătorul ajunge la o valoare specificată întru registru de comparare. Odată ajuns la această valoarea, numărătorul își va reseta valoarea internă, la următorul ciclu de tact, cu valoarea 0 și va continua numărarea. În funcție de valoarea setată în registrul comparator și frecvența incrementărilor se poate controla frecvența întreruperilor.

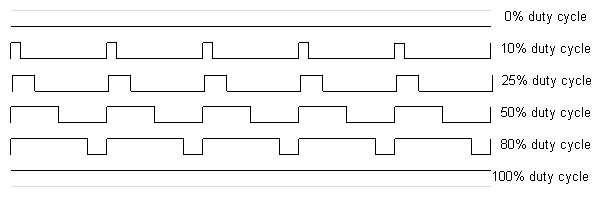
**Porturi digitale de intrare/ieșire**

Porturile digitale au două nivele de tensiune, 0V și 5V.

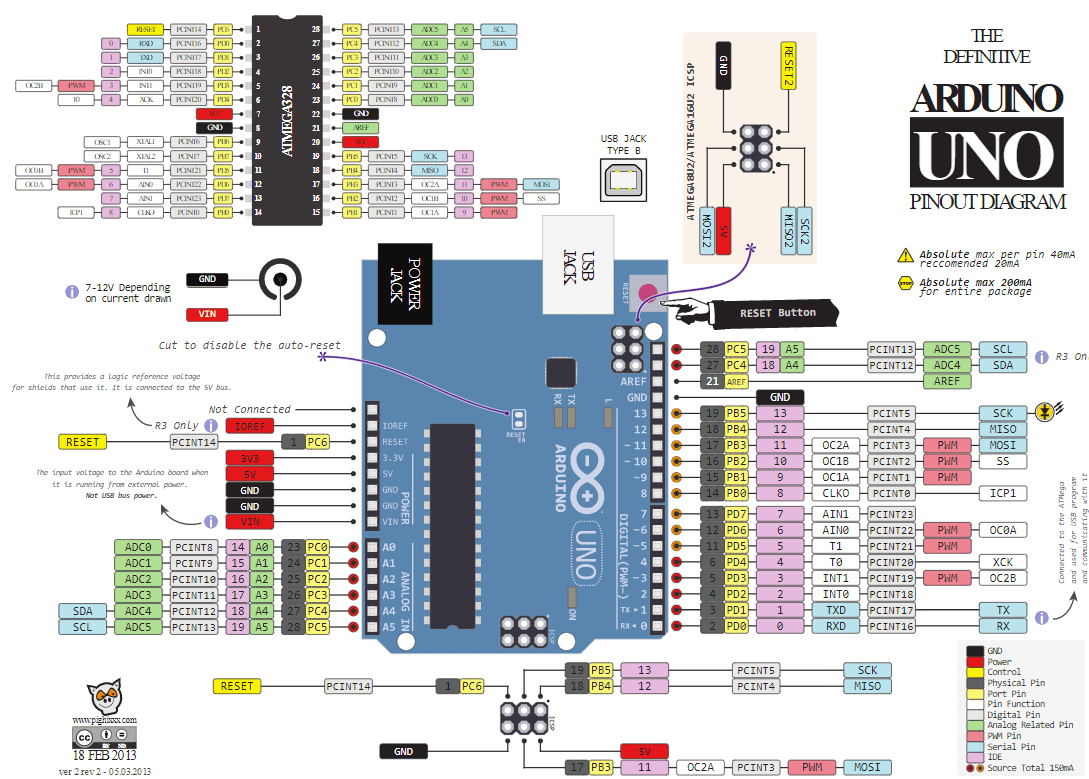
Terminalele din aceste porturi au 3 moduri de funcționare:

* INPUT: modul intrare standard, în care terminalul este lăsat în starea a 3-a și necesita o rezistenta externa de ridicare sau coborare
* OUTPUT: modul iesire
* INPUT\_PULLUP: modul intrare, în care terminalul este trecut printr-o rezistenta de ridicare (incorporata în microcontroller) de 20 kΩ

6 dintre aceste porturi suporta generarea de semnale PWM (Pulse Width Modulation), cu o precizie de 8 biti, folosind timerele din microcontrollerul ATmega328P. Semnalele PWM reprezinta o metoda de a genera comportamente analogice utilizand semnale digitale, modulate în timp la frecventa ridicate.

**Exemplu de semnale PWM:**

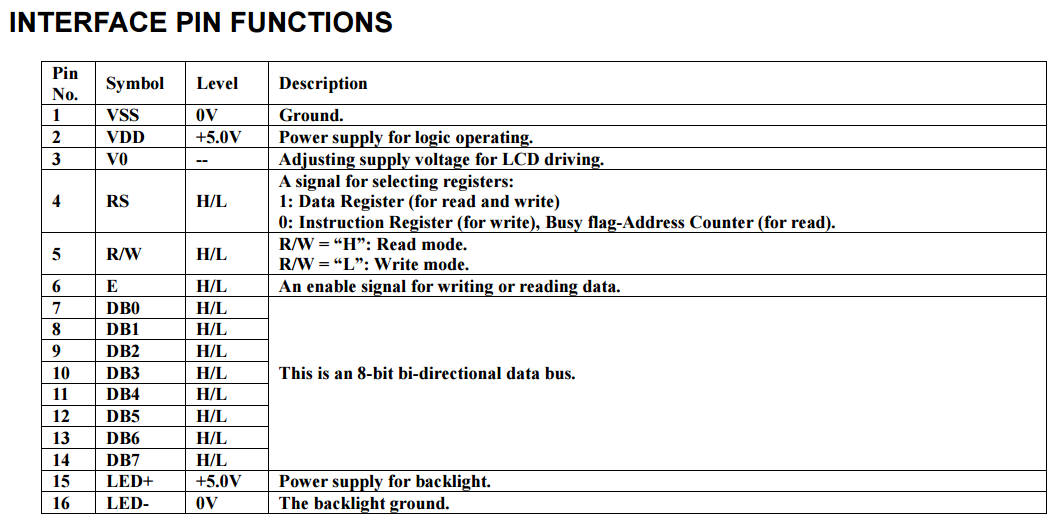
**Schema placii cu toti pinii disponibili:**

****

**Funcționarea sistemului cu afișaj LCD (16x2 caractere) :**

LCD-urile au o interfață paralelă, ceea ce înseamnă că microcontrolerul trebuie să manipuleze mai multe terminale de interfață dintr-o dată pentru a controla afișarea. Interfața este format din următoarele terminale:





Registru Selector (**RS**): acest terminal controlează unde în memoria ecranului LCD se scriu datele. Se poate selecta fie registrul de date, care reține ceea ce va apărea pe ecran, sau un registru de instrucțiuni, care este folosit în atunci când controlerul ecranului LCD caută urmatoarele instrucțiuni.

Citire/Scriere (**R/W**): terminal care selectează modul citire sau modul scriere

**Enable**: terminalul care permite scrierea în registre

8 terminale de date (**D0-D7**). Stările acestora (0 sau 1 logic) sunt biții care sunt scriși într-un registru atunci când lucrăm în modul scriere, sau valorile care au fost citite când lucrăm în modul citire.

Există, de asemenea, un terminal pentru constrastul LCD-ului (**Vo/VEE**), terminalul de alimentare (**+5V** și **GND**) și terminalele pentru LED-urile de iluminare din spate (**Bklt+** și **BKlt-**).

Aceste terminale sunt utilizate pentru a alimenta LCD-ul, pentru a controla contrastul afișajului și pentru a activa și dezactiva iluminarea din spate a afisajului.

Procesul de control al afișajului implica punerea datelor care formeaza imaginea a ceea ce trebuie să fie afișat în registrele de date, apoi punerea instrucțiunilor în registrul de instrucțiuni.

LCD-urile cu driver Hitachi HD44780, sau LCD-urile compatibile cu acestea, pot fi controlate în două moduri: 4-biți sau 8 biți. Modul pe 4 biți necesită șapte terminale digitale de intrare/iesire din Arduino pentru a fi controlate, în timp ce modul de 8-biți necesită 11 terminale. Pentru afișarea textului pe ecran este suficient controlul în modul 4 biți.

**Diagrama de control pentru scriere (de la microcontroller la afisaj)**

# 

# 

# 

# 

# **Senzor ultrasonic HC-SR04**

Acest senzor poate măsura distante intre 2 cm și 400 cm cu precizie care poate ajunge la 3 mm. Modulul HC-SR04 include un transmitator ultrasonic, un receptor și un circuit de comanda.

Pentru functionare senzorul are nevoie de 4 terminale: **VCC** (Power), **Trig** (Trigger), **Echo** (Primire), și **GND** .

**Caracteristici:**

* Tensiune de operare: DC 5V
* Curentul de functionare: 15mA
* Unghi de functionare: 15 °
* Distanta: 2cm - 4m
* Durată minima semnal trigger: 10us.

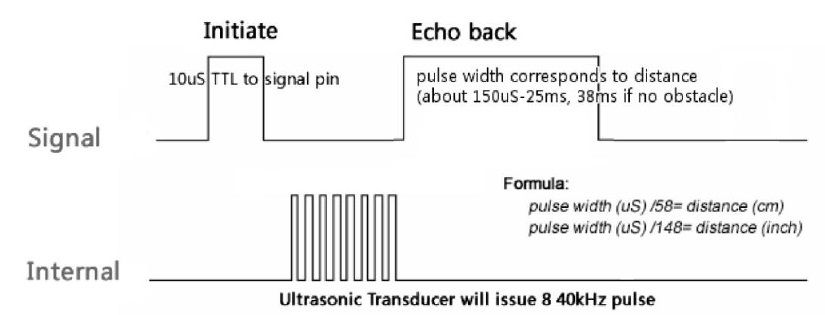
**Modul de functionare**:

* Se trimite un semnal pe nivel inalt pentru cel putin 10us pe terminalul Trig
* Modulul trimite automat opt semnale dreptunghiulare cu frecventa de 40 kHz și asteapta să primească semnalul ultrasonic reflectat (un impuls) înapoi
* Când receptorul primește impulsul înapoi, va seta terminalul Echo pe nivel înalt pentru un timp proporțional cu distanța reflectată

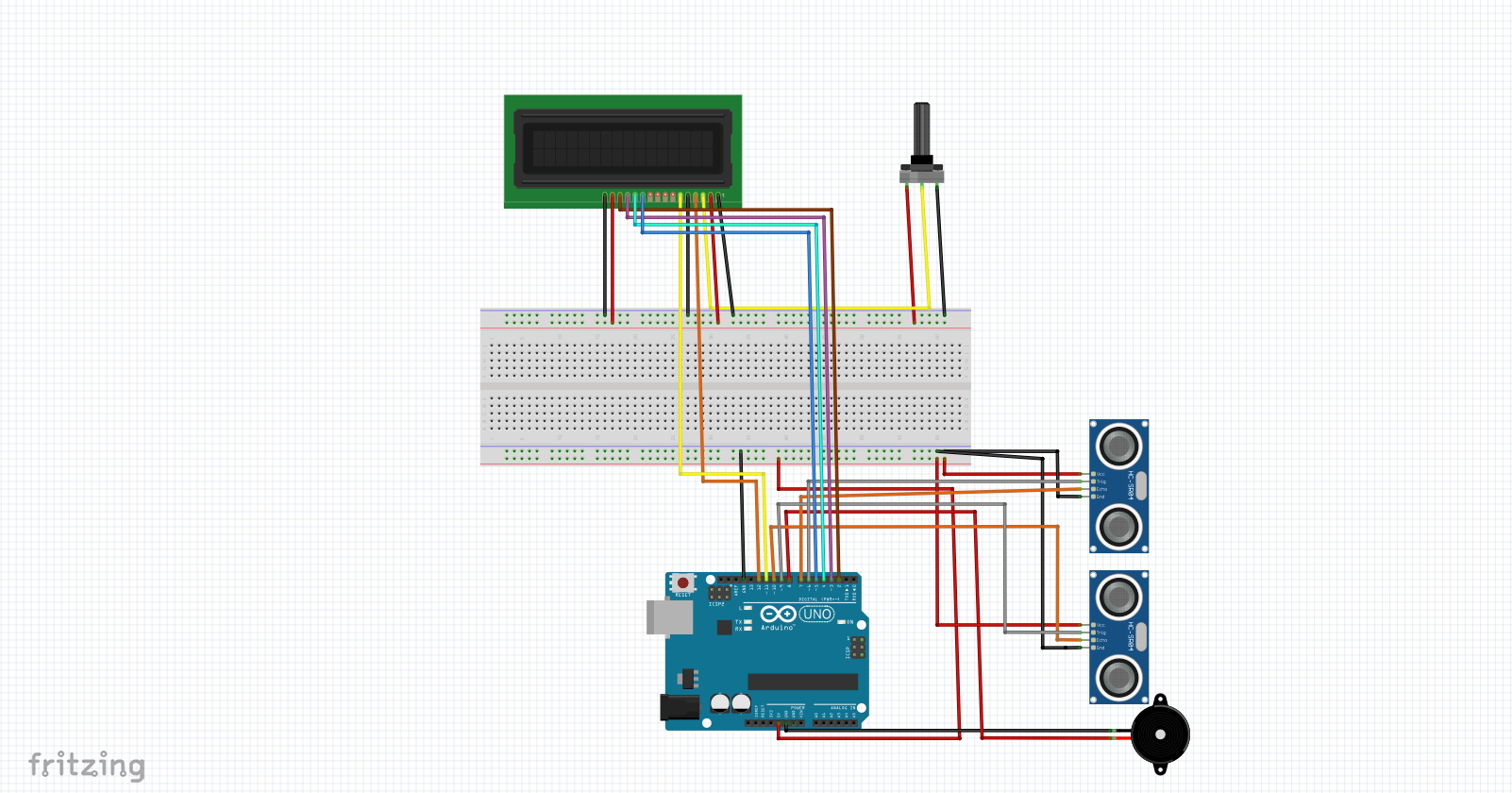
**Calculul distanței:**

Distanța= (timpul pe nivel înalt \* viteza sunetului (340 m/s) ) / 2

Distanța în centimetri = timpul pe nivel înalt / 58



**Schema circuitului**



#include <LiquidCrystal.h>

#define trigPin1 6

#define echoPin1 7

#define buzzPin 8

#define trigPin2 9

#define echoPin2 10

#define minDistance 10

#define maxDistance 100

#define minFrequency 31

#define maxFrequency 4095

// LiquidCrystal(rs, enable, d4, d5, d6, d7)

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

lcd.begin(16, 2);

pinMode(trigPin1, OUTPUT);

pinMode(echoPin1, INPUT);

pinMode(trigPin2, OUTPUT);

pinMode(echoPin2, INPUT);

pinMode(buzzPin, OUTPUT);

}

long readDistance(int echoPin, int trigPin) {

// triggers internal command for the sensor to know that after this it will come a "read" operation

long distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads a pulse (either HIGH or LOW) on a pin and returns the length of the pulse in microseconds

distance = pulseIn(echoPin, HIGH);

distance = (distance/2) \* 0.34;

return distance;

}

void loop() {

long distance1, distance2, distance;

distance1 = readDistance(echoPin1, trigPin1);

distance2 = readDistance(echoPin2, trigPin2);

distance = (distance1 + distance2) / 2;

if(distance > minDistance && distance < maxDistance) {

lcd.setCursor(0, 0);

lcd.print(" ");

lcd.setCursor(0, 0);

lcd.print(distance);

lcd.print(" cm");

distance = map(distance, minDistance, maxDistance, maxFrequency, minFrequency); //maps the values of minDistance to maxFreq and vice-versa

//Serial.println(distance);

tone(buzzPin, distance);

}

else {

lcd.setCursor(0, 0);

lcd.print("Safe to proceed!");

noTone(buzzPin);

}

delay(20);

}

**Bibliografie**

* https://www.google.com/
* https://www.pololu.com/product/2191
* http://maxembedded.com/2011/07/avr-timers-ctc-mode/
* https://cdn-shop.adafruit.com/datasheets/TC1602A-01T.pdf
* https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
* https://product.tdk.com/info/en/catalog/datasheets/ef532\_ps.pdf
* https://arduino-info.wikispaces.com/Timers-Arduino
* https://www.arduino.cc/en/main/arduinoBoardUno
* https://www.arduino.cc/en/uploads/Main/Arduino\_Uno\_Rev3-schematic.pdf