Proiect Sisteme Incorporate

Aplicaţie Sonar / Radar folosind un senzor ultrasonic, un servomotor și o interfață grafică

Goțiu Bettina-Cătălina

Grosu Andrei

CTI-RO, AN III, SG. 3.1

1. Tema proiectului :

a. Sistemul va dispune de un senzor ultrasonic care se poate roti, cu o viteză constantă, în jurul unui ax, pentru a acoperi un anumit unghi de vizibilitate, de exemplu de la 15 grade până la 165 grade. Aceasta se poate realiza montând senzorul ultrasonic pe un servomotor comandat de microcontroller.
b. Se va măsura o distanţă cuprinsă între 10 şi 70 – 100 cm; senzorul ultrasonic va măsura în permanență distanța până la obiectele din jurul lui, în timp ce se rotește cu viteză constantă, realizând astfel o baleiere permanentă a zonei de vizibilitate, similar cu un sistem radar. Pe baza măsurătorilor de distanță obținute, se va realiza o cartografiere în timp real a zonei din jurul senzorului, care va fi reprezentată grafic printr-o aplicație care va rula pe computer / laptop / tabletă / smartphone (de exemplu, în varianta cea mai simplă, se afișează cu verde zonele în care nu au fost detectate obiecte și cu roșu zonele în care este câte un obiect, afișând și valoarea unghiului sub care este detectat acel obiect).
c. Simpla afișare a distanței până la obiecte nu este suficientă pentru promovarea proiectului. Se cere obligatoriu și realizarea aplicației grafice. Modul de implementare al acesteia este la alegerea studenților, de exemplu poate fi un program C# care rulează pe un laptop sau computer desktop, poate fi o aplicație web grafică în browser sau poate fi o aplicație grafică Android care rulează pe un dispozitiv mobil.

2. Problema abordată de proiect:

Prin acest proiect, rezolvăm problema creării unui sistem de radar folosind un Arduino Uno R3, senzor ultrasonic și servomotor, conectate la o breadboard. Aplicațiile practice ale acestui sistem sunt diverse și includ:

a. Securitatea și supravegherea

Sistemul poate fi utilizat pentru monitorizarea și detectarea obiectelor într-o anumită zonă, fiind util în aplicații de securitate și supraveghere, cum ar fi protecția perimetrelor sau a proprietăților.

b. Asistență în navigație

Radarul poate ajuta la detectarea obstacolelor în timpul navigației, fiind util pentru drone, roboți sau vehicule autonome pentru a evita coliziunile.

c. Asistență pentru persoane cu dizabilități

Radarul poate fi folosit pentru a crea dispozitive de asistență care ajută persoanele cu deficiențe de vedere să navigheze mai sigur în mediul înconjurător.

Aceste aplicații demonstrează versatilitatea și utilitatea unui sistem de radar bazat pe Arduino în diverse domenii, contribuind la creșterea siguranței, eficienței și autonomiei în multiple sectoare.

3. Funcționalitate :

Acest proiect utilizează un Arduino Uno R3 conectat la o breadboard pentru a organiza conexiunile și a asigura o implementare mai clară. Mai întâi, Arduino-ul este conectat la breadboard astfel:

- Pinul de alimentare 5V al Arduino este conectat la linia de alimentare pozitivă (+) a breadboard-ului.
- Pinul de masă (GND) al Arduino este conectat la linia de masă (-) a breadboard-ului.

Aceste conexiuni furnizează o sursă stabilă de alimentare pentru toate componentele conectate la breadboard.

În continuare, pentru a conecta servomotorul și senzorul ultrasonic la breadboard, urmăm aceeași logică de conectare:

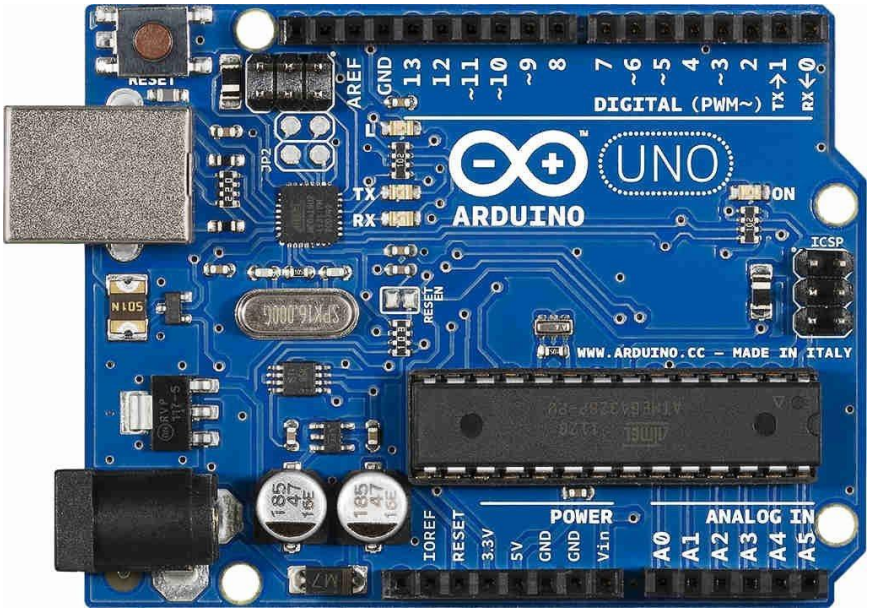
- Firul de alimentare (VCC) al servomotorului și al senzorului ultrasonic este conectat la linia de alimentare pozitivă (+) a breadboard-ului, asigurându-le astfel o sursă de alimentare constantă de 5V.
- Firul de masă (GND) al servomotorului și al senzorului ultrasonic este conectat la linia de masă (-) a breadboard-ului, oferindu-le un drum de întoarcere pentru curent.
- Firul SIGNAL (portocaliu) al servomotorului este conectat la pinul 12 de pe Arduino, ceea ce permite controlul poziției acestuia.
- Firul ECHO (alb) de la senzorul ultrasonic este conectat la pinul 11 de pe Arduino, permițând transmiterea semnalului de întoarcere pentru măsurarea distanței.
- Firul TRIG (mov) de la senzorul ultrasonic este conectat la pinul 10 de pe Arduino, utilizat pentru a trimite semnale de ultrasunete și a măsura distanța până la obiecte.

Aceste conexiuni completează configurarea hardware a proiectului, permițând Arduino-ului să comunice eficient cu servomotorul și senzorul ultrasonic, și să efectueze scanarea și măsurarea distanțelor înconjurătoare.

După încărcarea codului pe placa Arduino cu ajutorul Arduino IDE, proiectul utilizează Processing 4.3 pentru a afișa grafic informațiile colectate de radarul cu senzorul ultrasonic și servomotorul. Astfel, utilizatorul poate vizualiza și interacționa cu datele într-un mod grafic și intuitiv.

3.Descrierea placii de dezvoltare:

Arduino Uno R3



Placa de dezvoltare Arduino Uno R3 este o placa cu microcontroller bazar pe ATmega328. Are 20 de pini intrare/ieșire digitali(migrare care 6 pot fi utilizați ca PWM(pulse with modulation) și 6 pot fi utilizați ca intrări analogice), un rezonator de 16 MHz ,o conexiune USB, o mufa de alimentare și un buton de resetare.

Uno diferă de toate plăcile precedente prin faptul că nu folosește cipul de driver FTDI USB-to-serial. În schimb, are un ATmega16U2 programat ca un convertor USB-la-serial. Acest microcontroler auxiliar are propriul său bootloader USB, care permite utilizatorilor avansați să-l reprograma.

Caracteristici tehnice principale:

• Microcontroller: ATmega328

• Operating Voltage: 5V

• Input Voltage (recommended): 7-12V

• Input Voltage (limits): 6-20V

• Digital I/O Pins: 14 (of which 6 provide PWM output)

• Analog Input Pins: 6

• DC Current per I/O Pin: 40 mA

• DC Current for 3.3V Pin: 50 mA

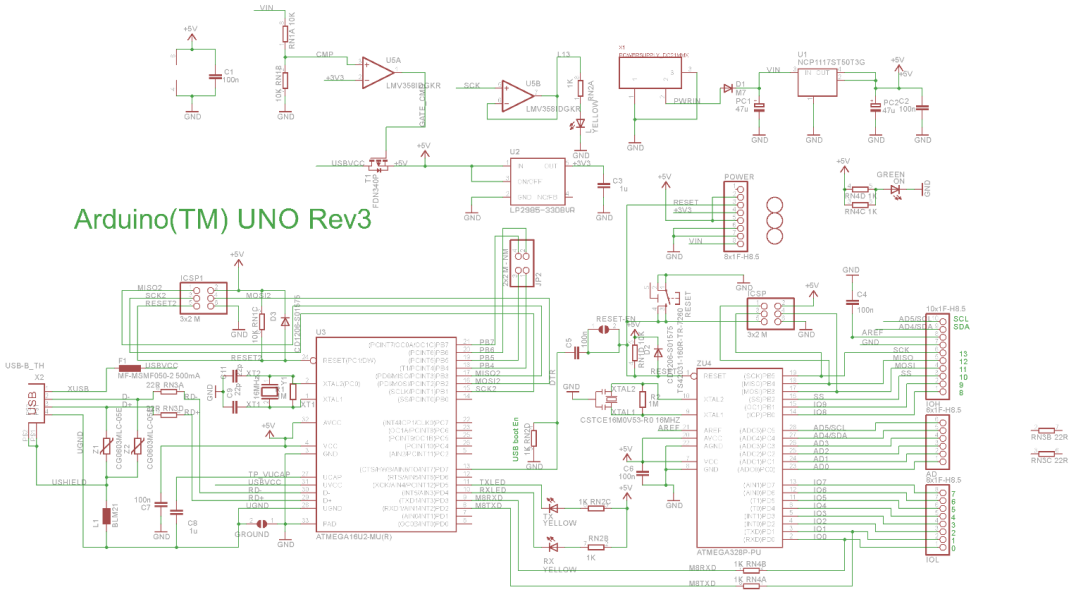
• Flash Memory: 32 KB of which 0.5 KB used by bootloader

• SRAM: 2 KB (ATmega328)

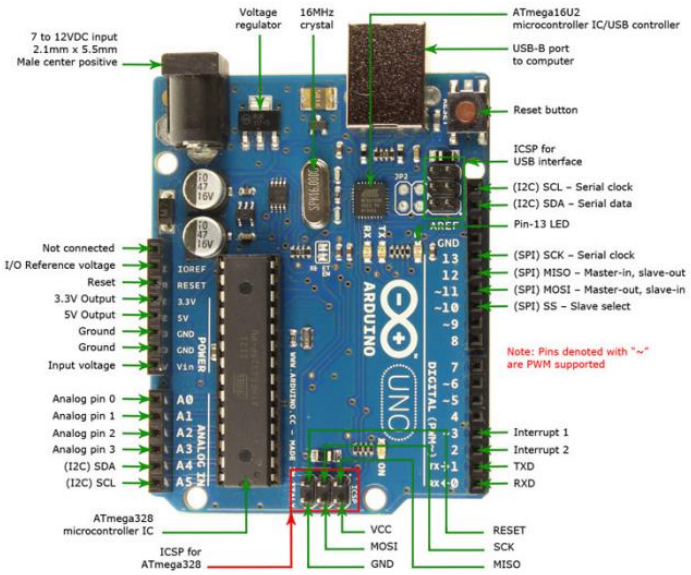
• EEPROM: 1 KB (ATmega328)

• Clock Speed: 16 MHz

Schema bloc:



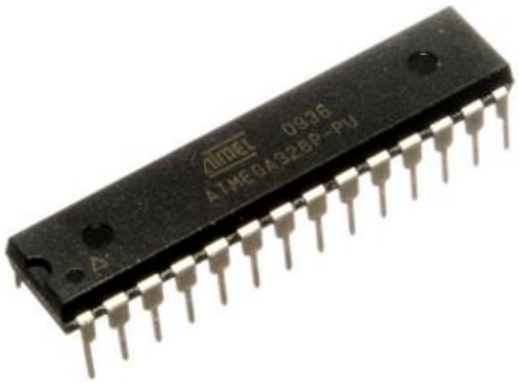
Descrierea pinilor:



Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

Microprocesorul ATmega328P



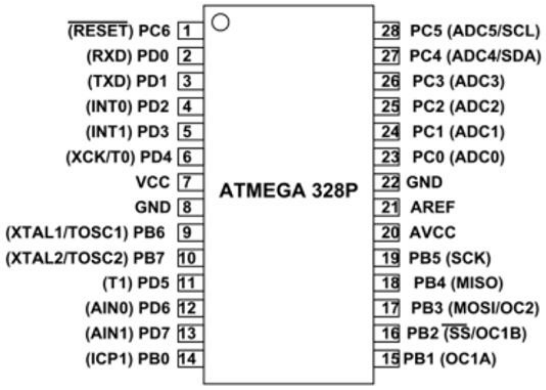
ATmega328 este un microcontroler cu un singur cip creat de Atmel în familia megaAVR. Are un nucleu de procesor RISC pe 8 biți cu arhitectură Harvard modificat .

Specificatii:

Microcontrolerul Atmel pe 8 biți AVR RISC combină memorie flash ISP de 32 KB cu capacități de citire în timp ce scriere, 1 KB EEPROM , 2 KB SRAM , 23 de linii I/O de uz general, 32 de registre de lucru de uz general , 3 flexibile temporizator/ contoare cu moduri de comparare, întreruperi interne și externe , USART serial programabil , o interfață serială cu 2 fire orientată pe octeți, port serial SPI , convertor A/D

pe 6 canale pe 10 biți (8 canale în pachetele TQFP și QFN / MLF), programabilTimer watchdog cu oscilator intern și 5 moduri de economisire a energiei selectabile prin software. Aparatul funcționează între 1,8 și 5,5 volți. Dispozitivul atinge un randament de aproximativ 1 MIPS /MHz.

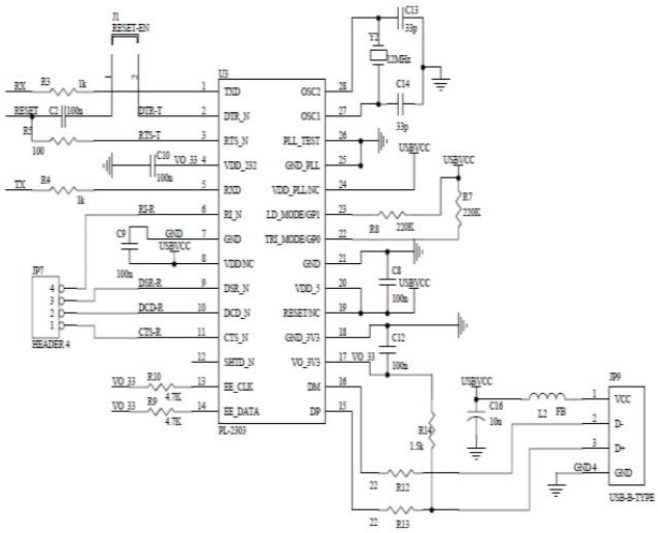
Descrierea pinilor:



Pin No.	Pin name	Description	Secondary Function
1	PC6 (RESET)	Pin6 of PORTC	Pin by default is used as RESET pin. PC6 can only be used as I/O pin when RSTDISBL Fuse is programmed.
2	PD0 (RXD)	Pin0 of PORTD	RXD (Data Input Pin for USART) USART Serial Communication Interface [Can be used for programming]
3	PD1 (TXD)	Pin1 of PORTD	TXD (Data Output Pin for USART) USART Serial Communication Interface [Can be used for programming] INT2(External Interrupt 2 Input)
4	PD2 (INT0)	Pin2 of PORTD	External Interrupt source 0
5	PD3 (INT1/ OC2B)	Pin3 of PORTD	External Interrupt source1 OC2B(PWM - Timer/Counter2 Output Compare Match B Output)
6	PD4 (XCK/ T0)	Pin4 of PORTD	T0(Timer0 External Counter Input) XCK (USART External Clock I/O)
7	VCC		Connected to positive voltage
8	GND		Connected to ground
9	PB6 (XTAL1/ TOSC1)	Pin6 of PORTB	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
10	PB7 (XTAL2/ TOSC2)	Pin7 of PORTB	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
11	PD5 (T1/OC0B)	Pin5 of PORTD	T1(Timer1 External Counter Input) OC0B(PWM - Timer/Counter0 Output Compare Match B Output)
12	PD6 (AIN0/ OC0A)	Pin6 of PORTD	AIN0(Analog Comparator Positive I/P) OC0A(PWM - Timer/Counter0 Output Compare Match A Output)
13	PD7 (AIN1)	Pin7 of PORTD	AIN1(Analog Comparator Negative I/P)

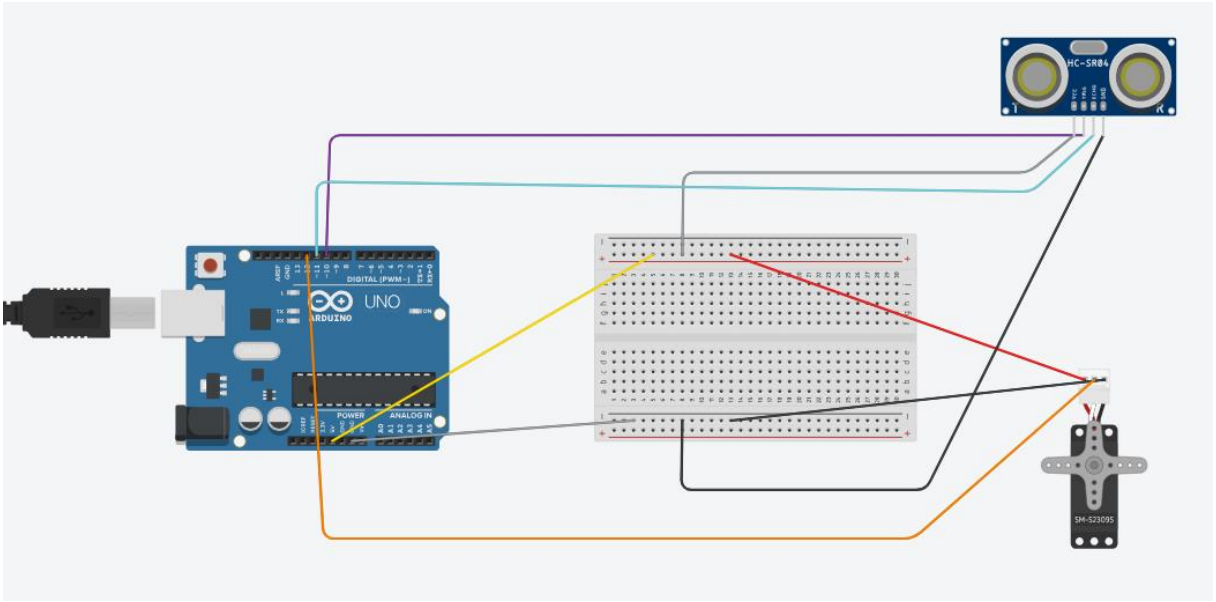
14	PB0 (ICP1/ CLKO)	Pin0 of PORTB	ICP1(Timer/Counter1 Input Capture Pin) CLKO (Divided System Clock. The divided system clock can be output on the PB0 pin)
15	PB1 (OC1A)	Pin1 of PORTB	OC1A (Timer/Counter1 Output Compare Match A Output)
16	PB2 (SS/ OC1B)	Pin2 of PORTB	SS (SPI Slave Select Input). This pin is low when controller acts as slave. [Serial Peripheral Interface (SPI) for programming] OC1B (Timer/Counter1 Output Compare Match B Output)
17	PB3 (MOSI/ OC2A)	Pin3 of PORTB	MOSI (Master Output Slave Input). When controller acts as slave, the data is received by this pin. [Serial Peripheral Interface (SPI) for programming] OC2 (Timer/Counter2 Output Compare Match Output)
18	PB4 (MISO)	Pin4 of PORTB	MISO (Master Input Slave Output). When controller acts as slave, the data is sent to master by this controller through this pin. [Serial Peripheral Interface (SPI) for programming]
19	PB5 (SCK)	Pin5 of PORTB	SCK (SPI Bus Serial Clock). This is the clock shared between this controller and other system for accurate data transfer. [Serial Peripheral Interface (SPI) for programming]
20	AVCC		Power for Internal ADC Converter
21	AREF		Analog Reference Pin for ADC
22	GND		GROUND
23	PC0 (ADC0)	Pin0 of PORTC	ADC0 (ADC Input Channel 0)
24	PC1 (ADC1)	Pin1 of PORTC	ADC1 (ADC Input Channel 1)
25	PC2 (ADC2)	Pin2 of PORTC	ADC2 (ADC Input Channel 2)
26	PC3 (ADC3)	Pin3 of PORTC	ADC3 (ADC Input Channel 3)
27	PC4 (ADC4/ SDA)	Pin4 of PORTC	ADC4 (ADC Input Channel 4) SDA (Two-wire Serial Bus Data Input/output Line)
28	PC5 (ADC5/ SCL)	Pin5 of PORTC	ADC5 (ADC Input Channel 5) SCL (Two-wire Serial Bus Clock Line)

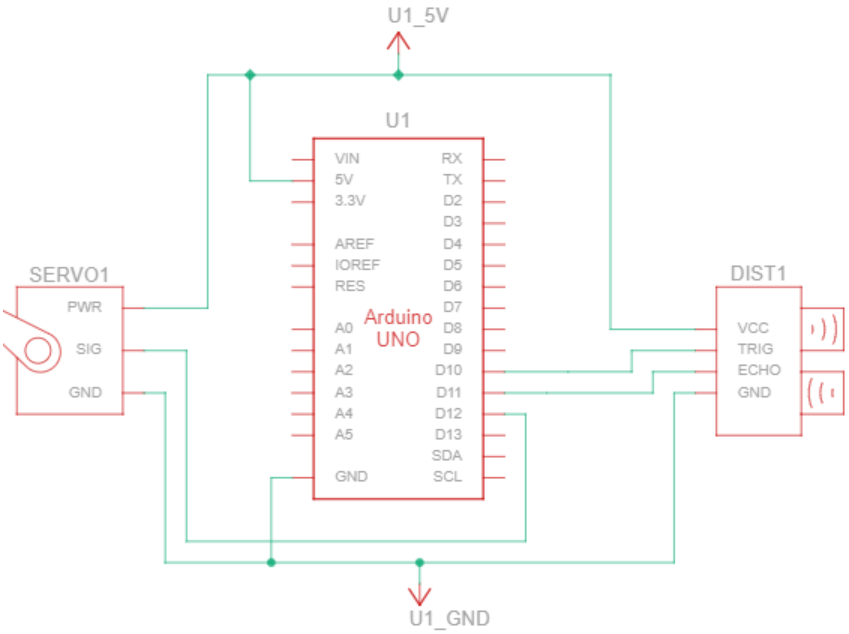
Schema bloc:



4. .Arhitectura sistemului:

Schema bloc a sistemului:





5. Descrierea modulelor folosite:

5.1 Senzor de distanta HC-SR04

Senzorul ultrasonic HC-SR04 este unul dintre cei mai utilizați senzori pentru aflarea distanței. În special folosit pentru proiectele cu plăci de dezvoltare Arduino, are avantaje față de senzorii analogici, necesitând doar pini I/O digitali și are imunitate mai mare la zgomotul din jur. Senzorul emite ultrasunete la o frecvență de 40000Hz care circulă prin aer, iar dacă întâlnește un obstacol, acesta se va întoarce înapoi spre modul, astfel, luând in considerare viteza sunetului se poate calcula distanța până la obiect.

5.1.1 Caracteristici tehnice:

Tensiune de alimentare: 5V

Curent consumat: 15mA

Distanță de funcționare: 2cm - 4m

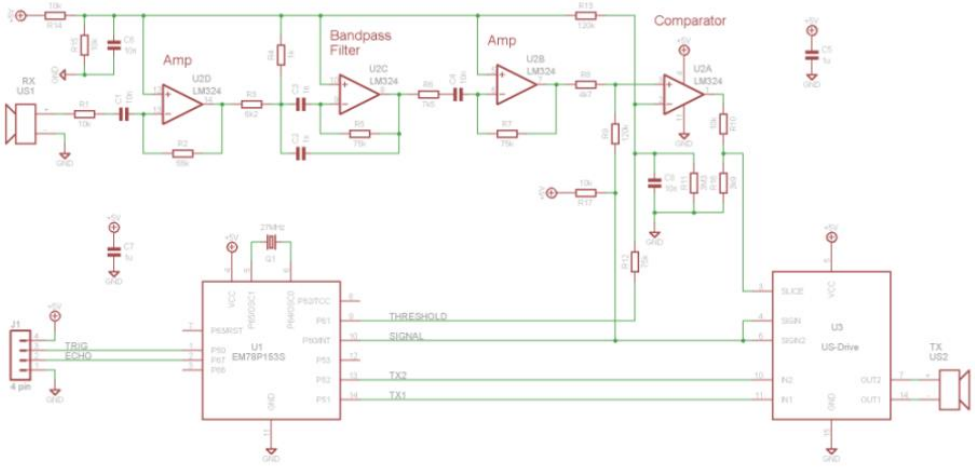
Unghi de măsurare: 15 grade

Eroare de doar 3mm

Durată semnal input: 10us

Dimensiuni: 45mm x 20mm x 15mm

5.1.2 Schema:

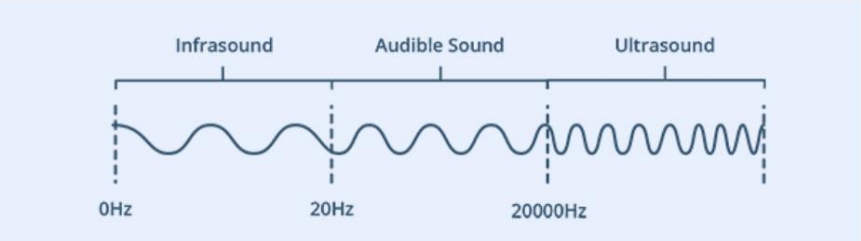


5.1.3 Descrierea pinilor:

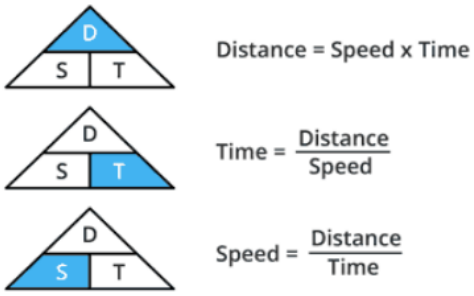
Vcc- Connects to 5V of positive voltage for power
Trig- A pulse is sent here for the sensor to go into ranging mode for object detection
Echo- The echo sends a signal back if an object has been detected or not. If a signal is returned, an object has been detected. If not, no object has been detected.
GND- Completes electrical pathway of the power.

5.1.4 Descrierea hardware:

Ultrasunetele sunt unde sonore înalte cu frecvențe mai mari decât limita audibilă a auzului uman.

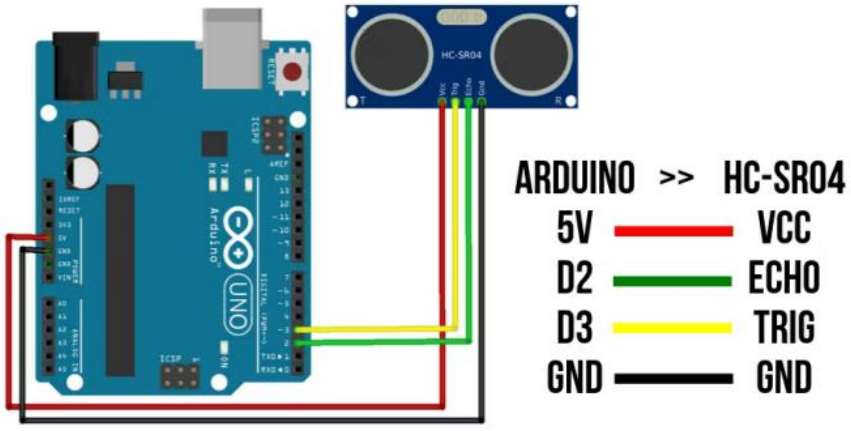


Urechile umane pot auzi unde sonore care vibrează în intervalul de la aproximativ 20 de ori pe secundă (un zgomot profund) până la aproximativ 20.000 de ori pe secundă (un fluier înalt). Cu toate acestea, ultrasunetele au o frecvență de peste 20.000 Hz și, prin urmare, sunt inaudibile pentru oameni.În esență, senzorul de distanță cu ultrasunete HC-SR04 este format din două traductoare cu ultrasunete. Unu acționează ca și un transmițător care convertește semnalul electric în impulsuri sonore ultrasonice de 40 KHz. Receptorul ascultă impulsurile transmise. Dacă le primește, produce un impuls de ieșire a cărui lățime poate fi folosită pentru a determina distanța parcursă de impuls. Totul începe cu un impuls de cel putin 10 microsecunde aplicat pinului de declasare. Ca răspuns la aceasta, senzorul transmite o unsa sonora forfota din 8 impulsuri la 40 KHz(Acest model de 8 impulsuri face ca "semnătură de ultrasunete" de la dispozitiv sa fie unica, permițând receptorului sa diferențieze modelul transmis de zgomotul ambiental. Cele 8 impulsuri se deplasează prin aer și între timp pinul ECHO devine HIGH pentru a începe sa formeze începutul semnalului eco-back.



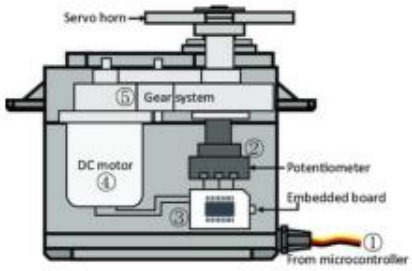
Latimea impulsului primit este apoi utilizată pentru a calcula distanta pana la obiectul reflectat.

5.1.5 Legarea cu arduino

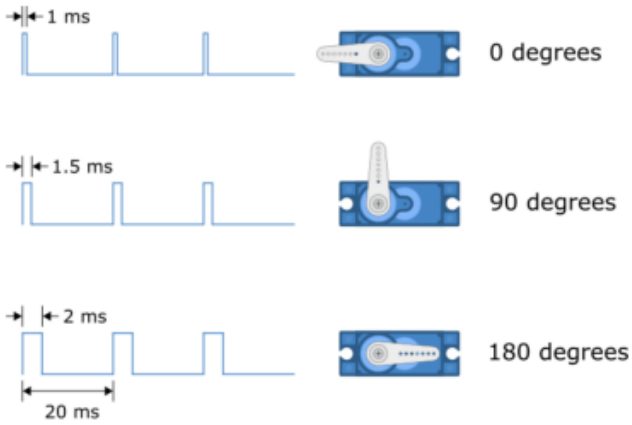


5.3 Servomotoare:

5.3.1 Schema :



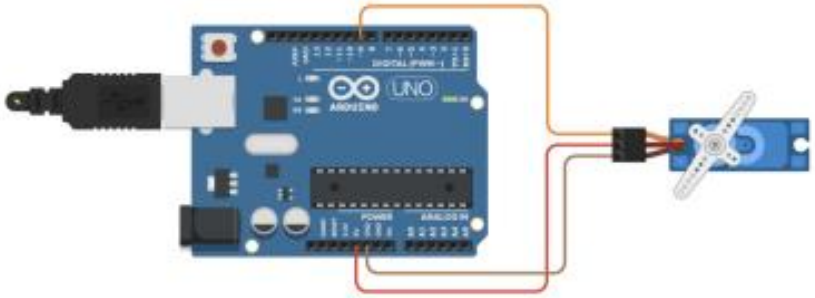
Servomotoarele sunt controlate prin trimiterea unui semnal PWM (modulație pe lățime a impulsului) către linia de semnal a servo. Lățimea impulsurilor determină poziția arborelui de ieșire. Când trimiteți servo un semnal cu o lățime a impulsului de 1,5 milisecunde (ms), servo se va muta în poziția neutră (90 de grade). Poziția minimă (0 grade) și maximă (180 de grade) corespund de obicei unei lățimi a impulsului de 1 ms și, respectiv, 2 ms. Rețineți că acest lucru poate varia ușor între diferitele tipuri și mărci de servomotoare (de exemplu, 0,5 și 2,5 ms). Multe servo-uri se rotesc doar cu aproximativ 170 de grade (sau chiar doar 90), dar poziția de mijloc este aproape întotdeauna la 1,5 ms.



5.3.2 Descrierea pinilor:

Servo motor	Connection
Power (red)	5 V power supply
Ground (black or brown)	Power supply ground and Arduino GND
Signal (yellow, orange or white)	Pin 9 Arduino

5.3.3 Legarea cu Arduino:



6. Cod Arduino IDE:

```
#include <Servo.h>.
// Definește pinii Trig și Echo ai senzorului ultrasonic
const int trigPin = 10;   // Pinul Trig este conectat la pinul 10 de pe Arduino
const int echoPin = 11; // Pinul Trig este conectat la pinul 11 de pe Arduino

// Variabile pentru durata și distanța măsurată
long duration;
int distance;

Servo myServo; // Creează un obiect servo pentru a controla servomotorul

void setup() {
  pinMode(trigPin, OUTPUT);  // Setează pinul Trig ca ieșire
  pinMode(echoPin, INPUT);  // Setează pinul Echo ca intrare
  Serial.begin(9600); // Inițializează comunicația serială la 9600 baud
  myServo.attach(12);  // Atașează servomotorul la pinul 12
}

void loop() {
  // Rotește servomotorul de la 15 la 165 de grade
  for(int i=15;i<=165;i++){
  myServo.write(i); // Setează servomotorul la unghiul i
  delay(30); // Așteaptă 30 de milisecunde
  distance = calculateDistance // Apelează funcția pentru a calcula distanța măsurată de senzorul ultrasonic pentru fiecare grad
```

```
 Serial.print(i);  // Trimite gradul curent prin portul serial
 Serial.print(","); // Trimite un caracter separator pentru a delimita valorile
 Serial.print(distance); // Trimite valoarea distanței prin portul serial
 Serial.print(".");  // Trimite un caracter separator pentru a delimita valorile
 }
 // Repetă liniile anterioare de la 165 la 15 grade
 for(int i=165;i>15;i--){
 myServo.write(i);
 delay(30);
 distance = calculateDistance();
 Serial.print(i);
 Serial.print(",");
 Serial.print(distance);
 Serial.print(".");
 }
}

 int calculateDistance(){  // Funcție pentru calcularea distanței măsurate de senzorul ultrasonic
 digitalWrite(trigPin, LOW);  // Pune pinul Trig pe LOW pentru 5 microsecunde pentru a reseta senzorul
 delayMicroseconds(5);
 digitalWrite(trigPin, HIGH);  // Pune pinul Trig pe HIGH pentru 10 microsecunde pentru a trimite un impuls
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);  // Pune pinul Trig din nou pe LOW
 duration = pulseIn(echoPin, HIGH); // Citește pinul Echo, returnează durata în microsecunde a impulsului
 distance= duration*0.034/2; // Calculează distanța în centimetri pe baza duratei impulsului
 return distance; // Returnează distanța calculată
}
```

7. Cod Processing pentru reprezentare grafica:

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial myPort;
String angle = "";
String distance = "";
String data = "";
```

```
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1 = 0;
int index2 = 0;
PFont orcFont;

void setup() {
 size(1920, 1080);
 smooth();
 myPort = new Serial(this, "COM3", 9600);
 myPort.bufferUntil('.');
}

void draw() {
 fill(98, 245, 31);
 noStroke();
 fill(0, 4);
 rect(0, 0, width, 1010);
 fill(98, 245, 31);
 drawRadar();
 drawLine();
 drawObject();
 drawText();
}

void serialEvent(Serial myPort) {
 data = myPort.readStringUntil('.');
 data = data.substring(0, data.length()-1);
 index1 = data.indexOf(",");
 angle = data.substring(0, index1);
 distance = data.substring(index1+1, data.length());
 iAngle = int(angle);
 iDistance = int(distance);
}

void drawRadar() {
 pushMatrix();
```

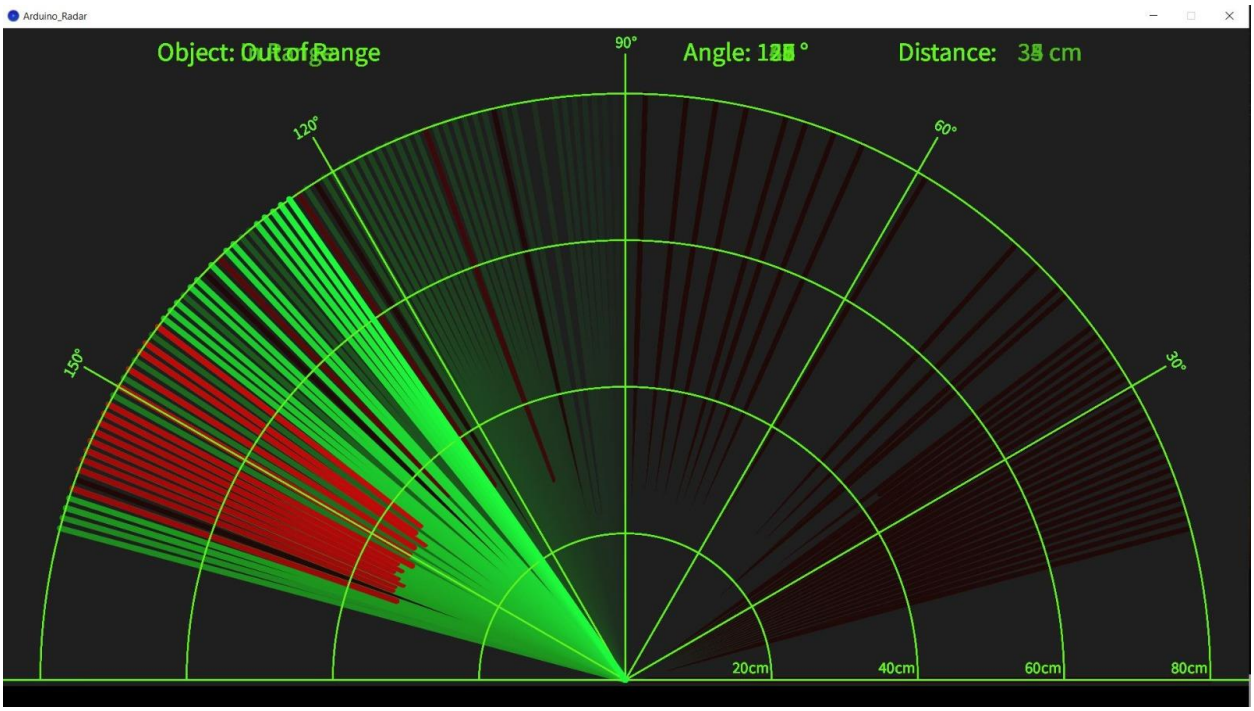
```
  translate(960, 1000);
  noFill();
  strokeWeight(2);
  stroke(98, 245, 31);
  arc(0, 0, 1800, 1800, PI, TWO_PI);
  arc(0, 0, 1350, 1350, PI, TWO_PI);
  arc(0, 0, 900, 900, PI, TWO_PI);
  arc(0, 0, 450, 450, PI, TWO_PI);  // Updated to accommodate 80cm range
  line(-960, 0, 960, 0);
  line(0, 0, -960*cos(radians(30)), -960*sin(radians(30)));
  line(0, 0, -960*cos(radians(60)), -960*sin(radians(60)));
  line(0, 0, -960*cos(radians(90)), -960*sin(radians(90)));
  line(0, 0, -960*cos(radians(120)), -960*sin(radians(120)));
  line(0, 0, -960*cos(radians(150)), -960*sin(radians(150)));
  line(-960*cos(radians(30)), 0, 960, 0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(960, 1000);
  strokeWeight(9);
  stroke(255, 10, 10);
  pixsDistance = map(iDistance, 0, 80, 0, 900);
  if (iDistance <= 80) {
    line(pixsDistance*cos(radians(iAngle)), -pixsDistance*sin(radians(iAngle)), 900*cos(radians(iAngle)), -
900*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30, 250, 60);
  translate(960, 1000);
  line(0, 0, 900*cos(radians(iAngle)), -900*sin(radians(iAngle)));
  popMatrix();
}
```

```
void drawText() {
  pushMatrix();
  if (iDistance > 80) {
    noObject = "Out of Range";
  } else {
    noObject = "In Range";
  }
  fill(0, 0, 0);
  noStroke();
  rect(0, 1010, width, 1080);
  fill(98, 245, 31);
  textSize(25);
  text("20cm", 1125, 990);
  text("40cm", 1350, 990);
  text("60cm", 1575, 990);
  text("80cm", 1800, 990);
  textSize(40);
  text("Object: " + noObject, 240, 50);
  text("Angle: " + iAngle + " °", 1050, 50);
  text("Distance: ", 1380, 50);
  if (iDistance <= 80) {
    text("       " + iDistance + " cm", 1500, 50);
  }
  textSize(25);
  fill(98, 245, 60);
  translate(961+960*cos(radians(30)), 982-960*sin(radians(30)));
  rotate(-radians(-60));
  text("30°", 0, 0);
  resetMatrix();
  translate(954+960*cos(radians(60)), 984-960*sin(radians(60)));
  rotate(-radians(-30));
  text("60°", 0, 0);
  resetMatrix();
  translate(945+960*cos(radians(90)), 990-960*sin(radians(90)));
  rotate(radians(0));
  text("90°", 0, 0);
  resetMatrix();
```

```
translate(935+960*cos(radians(120)), 1003-960*sin(radians(120)));
rotate(radians(-30));
text("120°", 0, 0);
resetMatrix();
translate(940+960*cos(radians(150)), 1018-960*sin(radians(150)));
rotate(radians(-60));
text("150°", 0, 0);
popMatrix();
```



Referințe Bibliografice:

1. **Arduino Uno**

Arduino. (n.d.). Arduino Uno Rev3 [Fișă tehnică].
https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

2. **ATmega328P**

Wikipedia. (n.d.). ATmega328. https://ro.wikipedia.org/wiki/Atmega328

Components101. (n.d.). ATmega328P: Pinout, Features, and Datasheet.
https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet

3. **Senzor de distanță HC-SR04**

Random Nerd Tutorials. (n.d.). Complete Guide for Ultrasonic Sensor HC-SR04.
https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

Zhu, Z. (n.d.). HC-SR04 User Manual: https://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf

4. **Servomotor**

SunFounder. (n.d.). Servo [Documentație]: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/27-component_servo.html