Senzor de parcare

-Proiect ARDUINO MEGA-

Disciplina: Proiectare cu microprocesoare

Student: Mătieși Darius-Andrei

Grupa:30235

An universitar: 2020 - 2021

Profesor îndrumător laborator și curs: Itu Răzvan

Contents

[1.Introducere în tema proiectului 3](#_Toc60685022)

[2. Prezentarea soluției 3](#_Toc60685023)

[2.1 Materiale necesare: 3](#_Toc60685024)

[2.2 Descrierea soluției abordate 4](#_Toc60685025)

[2.2.1 Senzorul ultrasonic 4](#_Toc60685026)

[2.2.2 Registrul de shiftare, LED-urile și buzzerul 4](#_Toc60685027)

[3. Montajul proiectului. 6](#_Toc60685028)

[4.Poze implementare 7](#_Toc60685029)

[Bibliografie 8](#_Toc60685030)

# 1.Introducere în tema proiectului

Obiectivul principal al acestui proiect îl reprezintă simularea unui senzor de parcare care funcționează pe baza unor led-uri și unui buzzer care semnalizează treptat apropierea de un obstacol. Cele 8 LED-uri se aprind rand pe rand odata cu apropierea unui corp și buzzer-ul sună din ce în ce mai des.

# 2. Prezentarea soluției

## 2.1 Materiale necesare:

+

Pentru realizarea acestui proiect a fost nevoie de următoarele componente:

* o placa de dezvoltare Arduino Mega 2560;
* un cablu pentru alimentarea plăcuței
* un buzzer;
* un senzor ultrasonic HC-SR04 pentru detectarea obstacolelor;
* 8 led-uri pentru semnalarea stării sistemului;
* 8 rezistențe de 220 ohmi pentru led-uri;
* un registru de shiftare SN74HC595;
* breadboard(mai mare) sau 2 breadboard mai mici;
* fire tată-tată/jumper pentru legarea componentelor.

## 2.2 Descrierea soluției abordate

Pentru realizarea senzorului de parcare, a fost necesară realizarea cuplării dintre două părți majore, și anume : senzorului ultrasonic și respectiv, registrul de shiftare pe 8 biți alături de LED-uri și buzzer

### 2.2.1 Senzorul ultrasonic

Senzorul de distanță ultrasonic HC-SR04 reprezintă un dispozitiv care calculează distanța de la acesta până la primul obstacol. Acesta prezintă 4 pini, VCC și Ground care sunt legate la plăcuță la 5V, respectiv Gnd, Trig care este legat la pinul 11 și Echo care este legat la pinul 12 al Arduino. În partea de setup(), aceștia trebuie setați ca intrare(echo) respectiv ieșire(trig) cu ajutorul funcției pinMode().

Principiul de funcționare este bazat pe utilizarea formulei vitezei( viteza sunetului fiind egala cu 340 m/s).In partea de loop(), prin intermediul pinului trig care se pune pe valoare ‚1’ timp de 10 microsecunde, se trimite un semnal de 8 microsecunde, recepționat de pinul echo care va calcula timpul

pe care sunetul l-a făcut până la corpul imediat apropiat. Timpul înscris pe pinul echo se extrage folosind funcția pulseIn(), care așteaptă valoarea ‚1’ al pinului echo, aceasta semnificând finalizarea înscrierii timpului respectiv. Pentru calcularea distanței, se face înmulțirea cu 0.034 al acestui semnal captat, după care se împarte la 2 deoarece valoarea salvată de pinul echo este dublă, deoarece se calculează timpul parcurs de sunet dus-întors la senzor.

### 2.2.2 Registrul de shiftare, LED-urile și buzzerul

Registrul de shiftare SN74HC595 reprezintă un circuit cu 16 pini, și anume:

* Q1-Q8: biții pentru shiftare- conectați prin rezistență la intrările Low ale LED-urilor;
* VCC-conectat la 5V;
* Gnd- conectat la 0V;
* SER-conectat la pinul 8 al plăcuței, reprezintă intrarea de date serială.
* - conectat la 0V, reprezintă pinul de output enable.
* RCLK- conectat la pinul 9 al plăcuței, reprezintă bitul stocarea ceasului registrului. Acesta trebuie ținut pe High în momentul în care se nu se așteaptă scrierea și comutat pe Low pentru momentul în care se face shiftarea
* SRCLK- conectat la pinul 10 al plăcuței, reprezintă clock-ul sistemului.
* -conectat la 0V.

Funcția principală pe care o implementează acest circuit este *shiftOut*(), care primește 4 parametrii(în această ordine):

**dataPin**: pinul pe care trebuie setați ca output fiecare bit. Tipuri de date permise: int.

**clockPin**: pinul pentru a comuta odată ce dataPin a fost setat la valoarea corectă. Tipuri de date permise: int.

**bitOrder**: care ordine pentru a deplasa biții; fie MSBFIRST, fie LSBFIRST. (Cel mai semnificativ bit întâi sau, cel mai puțin semnificativ primul bit).

**valoare**: datele care trebuie schimbate. Tipuri de date permise: octet.

Pentru buzzer, am ales să folosesc funcția millis() care calculează câte milisecunde au trecut de la începerea execuției programului și până în prezent. Aceasta este apelată de două ori pentru a se controla intervalul de timp la care buzzerul sună.

# 3. Montajul proiectului.

Diagram

Description automatically generated

# 4.Poze implementare

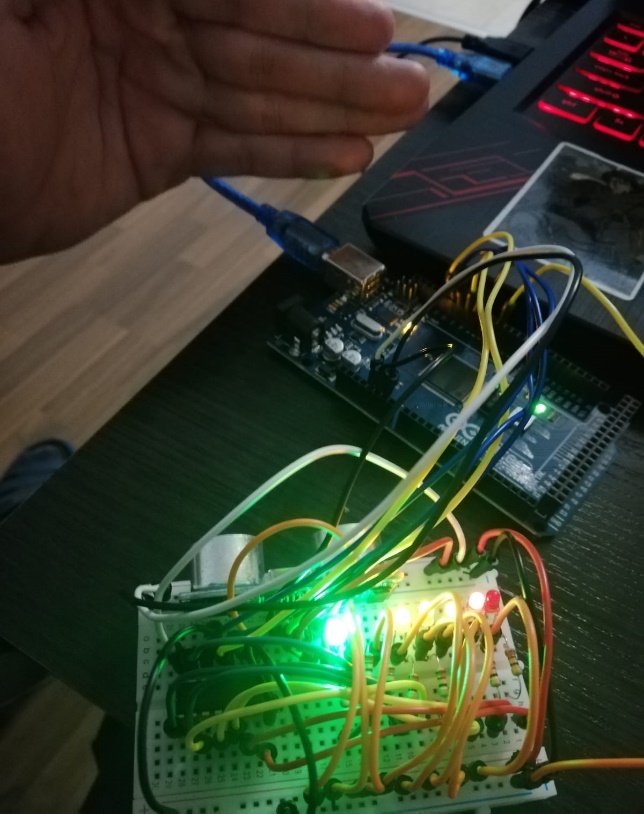
A picture containing electronics

Description automatically generatedA picture containing text, electronics

Description automatically generatedA picture containing electronics

Description automatically generated

A picture containing indoor

Description automatically generated 

# Bibliografie

**[1]**Dejan Nedelkovski. „Ultrasonic Sensor HC-SR04 and Arduino Tutorial.” -*HowToMechatronics.com.* [https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/.](https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/)

**[2]**Texas Instruments-SNx4HC595 8-Bit Shift Registers With 3-State Output Registers. <https://datasheetspdf.com/pdf-file/1398338/etcTI/SN74HC595/1>

**[3]**Tr3v3n\_Jaganath-Parking Sensor Assistant-create.arduino.cc-<https://create.arduino.cc/projecthub/Tr3v3n_Jaganath/parking-sensor-assistant-fe5778>

**[4]** Serial to Parallel Shifting-Out with a 74HC595- create.arduino.cc-<https://www.arduino.cc/en/Tutorial/Foundations/ShiftOut>