A black and white text on a black background

Description automatically generated

**Proiect practică**

Proiectarea unui releu de protecție de tensiune si curent în PROTEUS

Iordache Cristian-Sebastian

Mituleci Tudor-Darius

Homeghiu Mihnea

Iordache Decebal-Andrei

Facultate: IESC

Specializarea: ETTI

Contents

[Descrierea proiectului 3](#_Toc201827931)

[Despre Arduino Uno 3](#_Toc201827932)

[Conectori și componente importante 4](#_Toc201827933)

[Ce poți face cu un Arduino Uno? 4](#_Toc201827934)

[Accesorii compatibile 4](#_Toc201827935)

[Arduiono Uno in programare 5](#_Toc201827936)

[Obiectivul proiectului 5](#_Toc201827937)

[Componentele utilizate 5](#_Toc201827938)

[Funcționarea circuitului 5](#_Toc201827939)

[1) Măsurarea tensiunii și curentului: 5](#_Toc201827940)

[2) Procesarea datelor**:** 6](#_Toc201827941)

[3) Controlul becului prin releu: 6](#_Toc201827942)

[4) Afișare LCD: 6](#_Toc201827943)

[Circuitul Proteus 7](#_Toc201827944)

[Codul proiectului in Arduino Uno 8](#_Toc201827945)

# Descrierea proiectului

Acest proiect realizat în Proteus simulează un sistem de monitorizare și control al tensiunii și curentului electric, utilizând un releu, un microcontroller (Placuta Arduino), un senzor de tensiune, un senzor de curent, un bec de test și un afișaj LCD 16x2.

# Despre Arduino Uno

**Arduino Uno** este o placă de dezvoltare bazată pe microcontrolerul **ATmega328P**, folosită pentru prototiparea rapidă a circuitelor electronice și învățarea programării embedded. Este una dintre cele mai populare plăci din familia Arduino.

| **Caracteristică** | **Valoare** |
| --- | --- |
| Microcontroler | ATmega328P |
| Tensiune de operare | 5V |
| Tensiune de alimentare (recomandată) | 7–12V |
| Pini digitali I/O | 14 (dintre care 6 pot fi PWM) |
| Pini analogici de intrare | 6 |
| Curent maxim pe pin I/O | 40 mA |
| Memorie Flash | 32 KB (0.5 KB ocupat de bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Viteza de ceas | 16 MHz |
| Interfețe | UART, SPI, I2C |
| Port USB | Type-B pentru programare/alimentare |
|  |  |

# Conectori și componente importante

* **Port USB** – pentru încărcarea codului și alimentare.
* **Conector de alimentare (baril)** – pentru alimentare externă (7–12V).
* **Regulator de tensiune** – asigură 5V și 3.3V pentru componente.
* **Pini de I/O** – controlați senzori, LED-uri, motoare etc.
* **Pini analogici** – citesc valori de la senzori analogici.
* **Buton de reset** – reinitializează programul de pe placă.

# Ce poți face cu un Arduino Uno?

* Controlul LED-urilor, motoarelor, afișajelor LCD etc.
* Citirea datelor de la senzori (temperatură, lumină, accelerație).
* Proiecte de automatizare (ex: uși automate, sisteme de irigare).
* Sisteme IoT (cu module externe de rețea, ca ESP8266).
* Proiecte educaționale și experimentale.

# Accesorii compatibile

* Breadboard-uri, rezistențe, LED-uri, butoane
* Module: Bluetooth, Wi-Fi, GPS, RFID, RTC (ceasuri în timp real)
* Shield-uri: plăci ce se conectează direct pe Arduino (ex: Ethernet Shield, Motor Shield)

# Arduiono Uno in programare

Limbaj: C/C++ (Arduino Language – un subset cu funcții specifice).

Mediu de dezvoltare (IDE): Arduino IDE (disponibil gratuit pentru Windows, macOS, Linux).

Upload: Se face prin USB, iar IDE-ul include un bootloader pentru scrierea directă pe microcontroler

# Obiectivul proiectului

Sistemul are ca scop:

* Să monitorizeze în timp real valorile tensiunii și curentului de intrare.
* Să comute un releu pentru a bloca aprinderea unui bec în cazul în care tensiunea depășește un prag de siguranță prestabilit.
* Să afișeze pe un ecran LCD valorile măsurate și starea becului (**Pornit/Oprit**).

# Componentele utilizate

* Placa Arduino UNO 1.0
* Releu intelligent ()
* Senzor de tensiune
* Senzor de curent (ACS712x5A.MDF)
* Bec (sarcină de test)
* Ecran LCD 16x2

# Funcționarea circuitului

## Măsurarea tensiunii și curentului:

Tensiunea de intrare este preluată de un senzor de tensiune și convertită într-un semnal analogic care este citit de microcontroller.

În paralel, curentul consumat este măsurat cu un senzor de curent.

## Procesarea datelor**:**

Microcontrollerul citește valorile analogice, le convertește în valori reale (V și A), le compară cu pragurile prestabilite.

Dacă tensiunea depășește valoarea critică (de exemplu 240V), microcontrollerul dezactivează releul, întrerupând alimentarea becului.

## Controlul becului prin releu:

Releul este comandat de un tranzistor controlat de microcontroller.

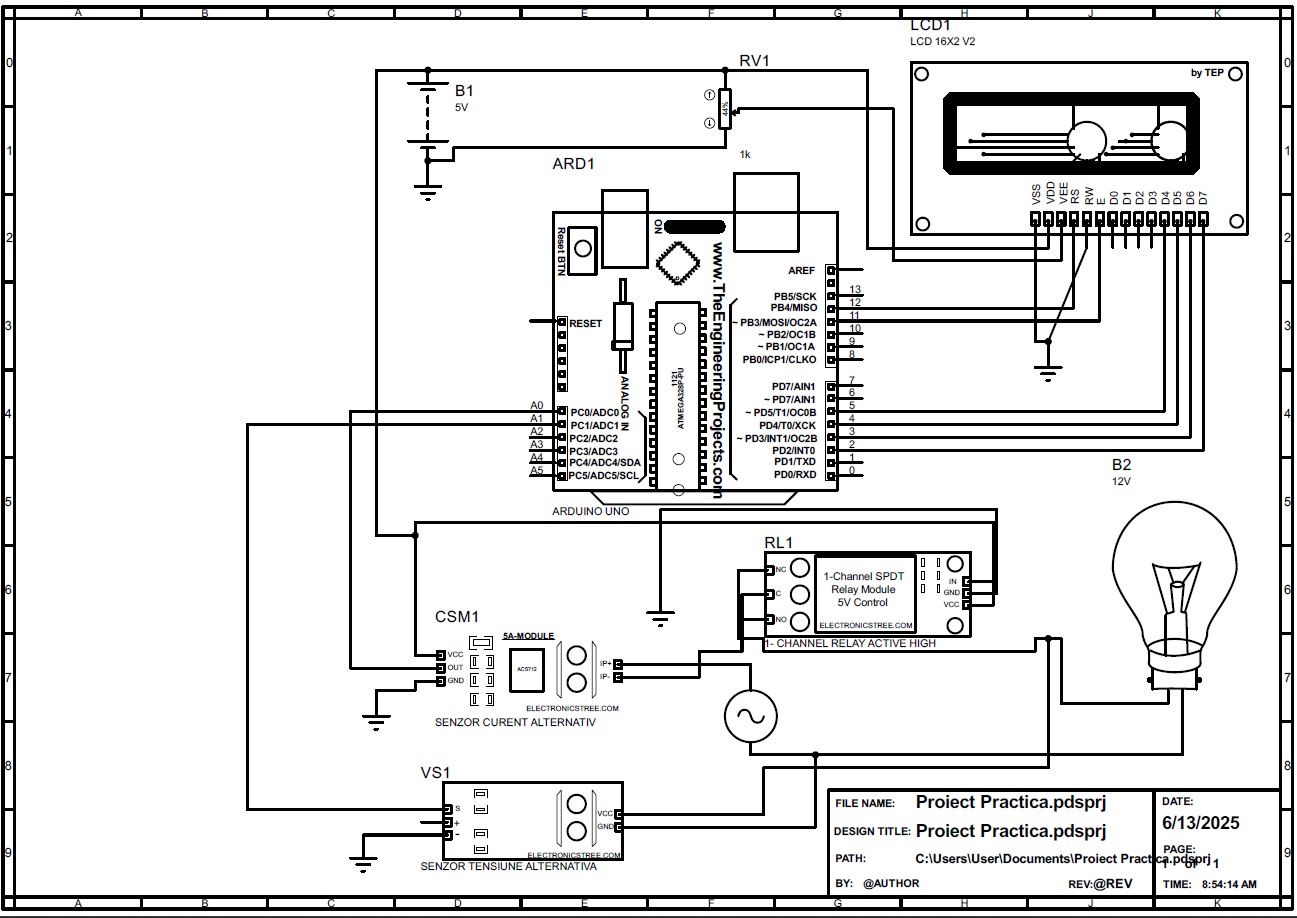
Dacă tensiunea este în limite normale, releul permite aprinderea becului. În caz contrar, becul este oprit pentru protecție.

## Afișare LCD:

Pe ecranul LCD sunt afișate:

* Valoarea tensiunii (ex: „Tensiune: 230V”)
* Valoarea curentului (ex: „Curent: 0.45A”)
* Starea becului (ex: „Bec: PORNIT” sau „Bec: OPRIT”)

# Circuitul Proteus



# Codul proiectului in Arduino Uno

#include <LiquidCrystal.h>

// LCD: RS, E, D4, D5, D6, D7

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

// Pini

const int voltagePin = A1;

const int currentPin = A0;

const int relayPin = 0;

//const int potPin = A2;

// Praguri

float voltageThresholdHigh = 240.0;

float voltageThresholdLow = 180.0;

float currentThreshold = 4.5;

void setup() {

lcd.begin(16, 2);

pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, LOW); // releu pornit (normal ON)

lcd.print("Sistem Protectie");

delay(2000);

lcd.clear();

}

void loop() {

// Citiri analogice

int voltageRaw = analogRead(voltagePin);

int currentRaw = analogRead(currentPin);

// int potRAW = analogRead(potPin);

// Conversie la valori reale (calibrate manual)

float voltage = voltageRaw \* (250.0 / 1023.0); // Ex: 0-250V

float current = (currentRaw - 512) \* (5.0 / 1023.0) \* 5.0; // Ex: +/-5A

// float treshold=map(analogRead(A2),0,1023,0,265);

// Afi?are pe LCD

lcd.setCursor(0, 0);

lcd.print("U:");

lcd.print(voltage);

lcd.print("V ");

lcd.setCursor(0, 1);

lcd.print("I:");

lcd.print(current);

lcd.print("A ");

// Control releu

if (voltage > voltageThresholdHigh || voltage < voltageThresholdLow || current > currentThreshold) {

digitalWrite(relayPin, HIGH); // Oprire sarcina

lcd.setCursor(10, 1);

lcd.print("STOP");

} else {

digitalWrite(relayPin, LOW); // Activare sarcina

lcd.setCursor(10, 1);

lcd.print(" OK ");

}

delay(500);

}