Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator Nr.4.1

Disciplina: IoT

Tema: Actuatori cu interfată binară. Releu

A efectuat: st.gr.TI-212,

Muntean Mihai

A verificat : asist. univ.

Lupan Cristian

Chișinău 2024

Definirea problemei:

Să se realizeze o aplicație in baza de MCU care va controla dispozitivele de acționare cu comenzi recepționate de la interfața serială și raportare către LCD.

Dispozitivele de acționare vor fi următoarele:

* un bec electric prin intermediul releului cu comenzi de ON si OFF

Obiective:

1. Implementarea comunicării seriale pentru recepționarea comenzilor;
2. Controlul releului pentru acționarea becului electric;
3. Afișarea stării pe LCD.

INTRODUCERE

Actuatorii reprezintă componente esențiale în sisteme automatizate, având rolul de a transforma un semnal de control (electronic sau mecanic) într-o acțiune fizică. În esență, un actuator este un dispozitiv care primește o comandă de la un sistem de control (precum un microcontroler) și o convertește într-o mișcare sau într-o altă formă de energie utilă, cum ar fi activarea unui releu pentru a aprinde un bec sau pentru a mișca un motor electric.

Într-un sistem automatizat, actuatoarele permit interacțiunea cu mediul fizic, permițând funcții precum mișcarea mecanică, controlul poziției, aprinderea sau stingerea unui dispozitiv și multe altele. Controlul precis al actuatorilor este important în aplicații variate, de la automatizarea industrială și robotică până la dispozitive inteligente de uz casnic.

Exemple de actuatori:

* Releurile – permit controlul dispozitivelor de putere (ex. becuri, motoare) printr-un semnal de control de joasă tensiune. Un releu poate fi acționat pentru a închide sau deschide un circuit electric.
* Motoarele electrice – utilizate pentru mișcarea componentelor, fiind acționate în funcție de intensitatea și direcția curentului.

EFECTUAREA LUCRĂRII

Materiale necesare:

* **Microcontroler** (Arduino Mega) folosit ca microcontroler principal pentru controlul releului și afișarea stării pe LCD. Acesta va primi comenzi prin interfața serială și va controla becul;
* **Releu** va fi utilizat pentru a comanda aprinderea și stingerea LED-ului, care simulează becul electric în această configurație;
* **LED** folosit pentru a reprezenta vizual becul aprins/stins. Acesta va fi conectat la terminalul NO (normally open) al releului, ceea ce permite LED-ului să se aprindă doar când releul este activat;
* **Display LCD I2C** pentru a afișa starea becului (aprins sau stins). Interfața I2C simplifică conexiunile necesare pentru LCD, reducând numărul de pini necesari pe Arduino Mega;
* **Surse de alimentare și fire de conectare** asigură alimentarea circuitului și conectarea între componente;
* **Modul de alimentare (VCC)** – în acest caz, simbolul de alimentare „VCC” este un nod virtual ce asigură o conexiune constantă de 5V.

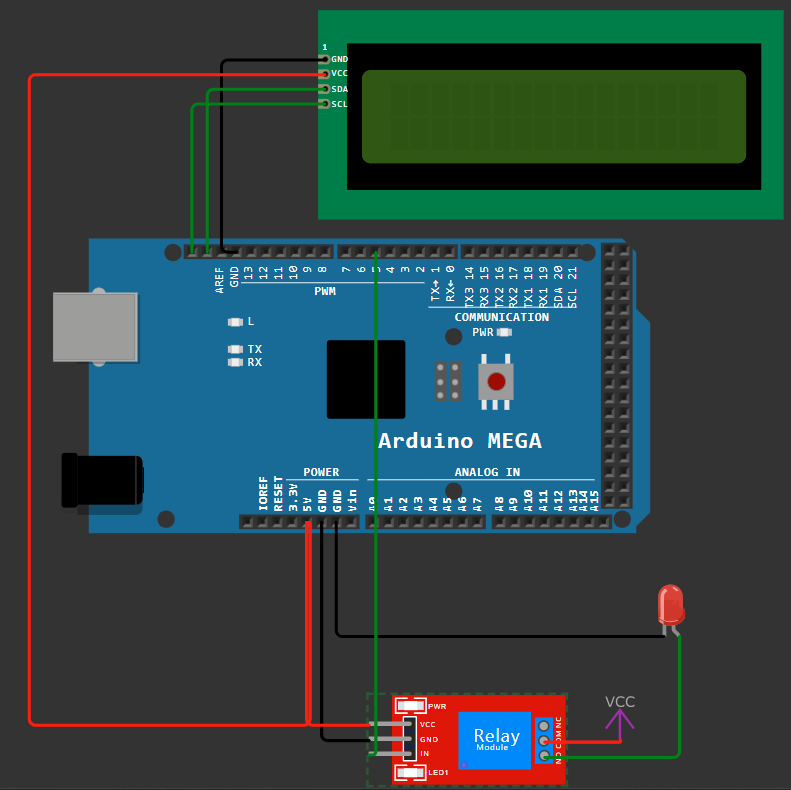


Figura 1 – Ansamblarea circuitului virtual

Modul de lucru

Utilizând această aplicație în bază de MCU, putem gestiona starea unui LED (aprins/stins), utilizând un releu comandat de microcontroler prin intermediul terminalului **Serial.**

#include "relay.h"

#include "lcd\_utils.h"

#define RELAY\_PIN 5

void serial\_read();

void HandleSerialCommand(char command);

bool relayState = false;

char command;

void setup() {

**Serial**.begin(15200);

  setup\_relay(RELAY\_PIN);

  initializeLcd(relayState);

}

Mai sus are loc instanțierea stărilor primare a componentelor, configurând microcontrolerul pentru a putea controla releul și LCD-ul.

void loop() {

  if (**Serial**.available() > 0) { // Verifică dacă există date disponibile în interfața serială

    serial\_read(); // Citește comanda de la utilizator

    HandleSerialCommand(command); // Procesează comanda primită și schimbă starea releului

    resetLcd(relayState); // Actualizează LCD-ul pentru a afișa starea curentă a releului

  }

}

// Functie care procesează comanda de la utilizator

void HandleSerialCommand(char command) {

  if (command == '1') { // Dacă comanda este '1'...

    relayState = turnRelayOn(RELAY\_PIN); // ...activează releul (ON)

  } else if (command == '0') { // Dacă comanda este '0'...

    relayState = turnRelayOff(RELAY\_PIN); // ...dezactivează releul (OFF)

  }

}

// Functie care citește comanda de la serial și verifică dacă este validă

void serial\_read(){

  scanf("%s", &command); // Citește comanda de la utilizator prin interfața serială

  if(command != '1' && command != '0'){ // Dacă comanda nu este '1' sau '0', este invalidă

    int c;

    while ((c = getchar()) != '\n' && c != EOF); // Curăță restul input-ului din buffer

    printf("Comanda invalida! Folositi '1' pentru ON si '0' pentru OFF.\n"); // Afișează mesaj de eroare

  }

}

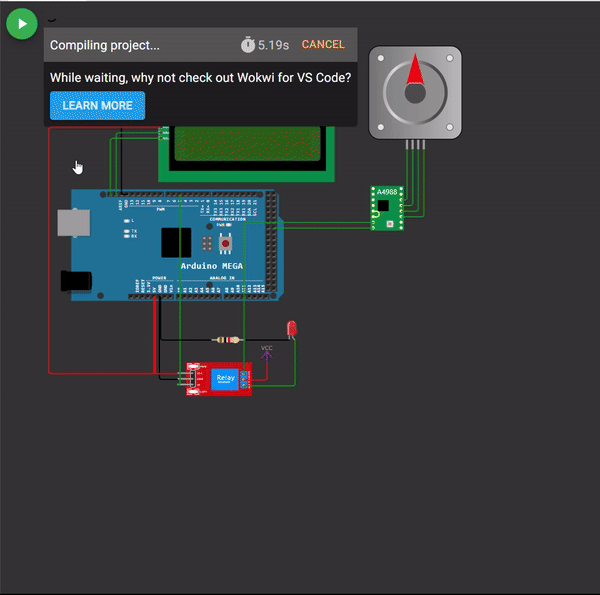


Figura 2 – Simularea aplicației în wokwi

CONCLUZIE

Lucrarea de laborator a demonstrat aplicarea practică a controlului unui actuator simplu (releu) prin intermediul unui microcontroler Arduino Mega. Prin utilizarea interfeței seriale, sistemul primește comenzi binare (ON/OFF) pentru a controla un bec electric, simulând funcționalitatea unui sistem IoT de automatizare. Afișajul LCD a oferit o modalitate eficientă de feedback vizual, permițând utilizatorului să vizualizeze starea actuală a releului în timp real.

BIBLIOGRAFII

1. Resursa electronică: <https://docs.wokwi.com/?utm_source=wokwi> – Regim de acces;
2. Resursa electronică: <https://else.fcim.utm.md/course/view.php?id=343> – Regim de acces;
3. Resursa electronică: <https://forum.arduino.cc/t/serial-print-and-printf/146256/14> - Regim de acces;
4. Proiectul pe GitHub, Resursa electronică: <https://github.com/MunMihai/Anul4/tree/8c7af8c43b8c728ff28e1ee6c75a63f997659015/Semestrul_7/Internetul_Lucrurilor%20(IoT)/Laborator/Laborator4> - Regim de acces;

Anexa 1

Fișierul stdinout.h

#ifndef \_STDINOUT\_H

#define \_STDINOUT\_H

// no need to make an instance of this yourself

class initializeSTDINOUT

{

        static size\_t initnum;

public:

        // Constructor

        initializeSTDINOUT();

};

// Call the constructor in each compiled file this header is included in

// static means the names won't collide

static initializeSTDINOUT initializeSTDINOUT\_obj;

#endif

Fișierul stdinout.cpp

#if ARDUINO >= 100

#include "Arduino.h"

#else

#include "WProgram.h"

#endif

#include <stdio.h>

#include "stdinout.h"

// Function that printf and related will use to print

static int serial\_putchar(char c, FILE \*f)

{

  if (c == '\n') {

    serial\_putchar('\r', f);

  }

  return **Serial**.write(c) == 1 ? 0 : 1;

}

// Function that scanf and related will use to read

static int serial\_getchar(FILE \*)

{

  // Wait until character is avilable

  while (**Serial**.available() <= 0) { ; }

  return **Serial**.read();

}

static FILE serial\_stdinout;

static void setup\_stdin\_stdout()

{

  // Set up stdout and stdin

  fdev\_setup\_stream(&serial\_stdinout, serial\_putchar, serial\_getchar, \_FDEV\_SETUP\_RW);

  stdout = &serial\_stdinout;

  stdin  = &serial\_stdinout;

  stderr = &serial\_stdinout;

}

// Initialize the static variable to 0

size\_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout

initializeSTDINOUT::initializeSTDINOUT()

{

  if (initnum++ == 0) {

    setup\_stdin\_stdout();

  }

}

Anexa 2

main.cpp

#include "relay.h"

#include "lcd\_utils.h"

#define RELAY\_PIN 5

#define STEP\_PIN 31

#define DIR\_PIN 33

void serial\_read();

void HandleSerialCommand(char command);

void runMotorStepper();

void changeDirection();

bool relayState = false;

char command;

bool motorState = 0;

bool dirState = 0;

void setup() {

**Serial**.begin(15200);

  setup\_relay(RELAY\_PIN);

  initializeLcd(relayState);

  pinMode(STEP\_PIN, OUTPUT);

  pinMode(DIR\_PIN, OUTPUT);

  printf("Session starts:\n");

}

void loop() {

  if (**Serial**.available() > 0) {

    serial\_read();

    HandleSerialCommand(command);

    resetLcd(relayState);

  }

  runMotorStepper();

}

void HandleSerialCommand(char command) {

  if (command == '1') {

    relayState = turnRelayOn(RELAY\_PIN);

  } else if (command == '0') {

    relayState = turnRelayOff(RELAY\_PIN);

    changeDirection();

  }

}

void serial\_read() {

  scanf("%s", &command);

  if (command != '1' && command != '0') {

    int c;

    while ((c = getchar()) != '\n' && c != EOF);

    printf("Comanda invalida! Folositi '1' pentru ON si '0' pentru OFF.\n");

  }

}

void runMotorStepper() {

  digitalWrite(STEP\_PIN, motorState == HIGH ? LOW : HIGH);

  motorState = !motorState;

}

void changeDirection() {

  digitalWrite(DIR\_PIN, dirState == HIGH ? LOW : HIGH);

  dirState = !dirState;

}

relay.h

#ifndef RELAY\_H

#define RELAY\_H

#include <Arduino.h>

bool turnRelayOn(short pin);

bool turnRelayOff(short pin);

void setup\_relay(short pin);

#endif

relay.cpp

#include "relay.h"

void setup\_relay(short pin){

  pinMode(pin, OUTPUT);

  digitalWrite(pin, LOW);

}

bool turnRelayOn(short pin) {

  digitalWrite(pin, HIGH);

  printf("Relay ON\n");

  return true;

}

bool turnRelayOff(short pin) {

  digitalWrite(pin, LOW);

  printf("Relay OFF\n");

  return false;

}

lcd\_utils.h

#ifndef LCD\_UTILS\_H

#define LCD\_UTILS\_H

#include <LiquidCrystal\_I2C.h>

#define I2C\_ADDR    0x27

#define COLUMNS 16

#define ROWS   2

void initializeLcd(bool relayState);

void resetLcd(bool relayState);

#endif // LCD\_UTILS\_H

lcd\_utils.cpp

#include "lcd\_utils.h"

#include <stdio.h>

LiquidCrystal\_I2C lcd(I2C\_ADDR, COLUMNS, ROWS);

void initializeLcd(bool relayState){

  lcd.begin(16, 2);

  lcd.backlight();

  resetLcd(relayState);

}

void resetLcd(bool relayState) {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Relay Status:");

  lcd.setCursor(0, 1);

  if (relayState) {

    lcd.print("ON");

  } else {

    lcd.print("OFF");

  }

}

ANEXA 3

diagram.json

{

  "version": 1,

  "author": "Mihai Muntean",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-arduino-mega", "id": "mega", "top": -47.4, "left": -3.6, "attrs": {} },

    { "type": "wokwi-relay-module", "id": "relay1", "top": 249.8, "left": 201.6, "attrs": {} },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": 169.2,

      "left": 378.2,

      "attrs": { "color": "red" }

    },

    {

      "type": "wokwi-lcd1602",

      "id": "lcd1",

      "top": -195.2,

      "left": 168.8,

      "attrs": { "pins": "i2c" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r1",

      "top": 205.55,

      "left": 259.2,

      "attrs": { "value": "1000" }

    },

    {

      "type": "wokwi-stepper-motor",

      "id": "stepper1",

      "top": -293.99,

      "left": 528.43,

      "attrs": { "size": "17", "arrow": "red", "gearRatio": "5:1" }

    },

    { "type": "wokwi-a4988", "id": "drv1", "top": -52.8, "left": 532.8, "attrs": {} },

    { "type": "wokwi-vcc", "id": "vcc1", "top": 221.56, "left": 345.6, "attrs": {} }

  ],

  "connections": [

    [ "relay1:GND", "mega:GND.2", "black", [ "h0" ] ],

    [ "relay1:NO", "led1:A", "green", [ "h87.6", "v-78.6" ] ],

    [ "mega:5V", "relay1:VCC", "red", [ "v131.7", "h40.7" ] ],

    [ "lcd1:GND", "mega:GND.1", "black", [ "h-67.2", "v105.6" ] ],

    [ "lcd1:VCC", "mega:5V", "red", [ "h-192", "v422.5", "h182.4", "v-131.7" ] ],

    [ "lcd1:SDA", "mega:SDA", "green", [ "h0" ] ],

    [ "lcd1:SCL", "mega:SCL", "green", [ "h0" ] ],

    [ "mega:5", "relay1:IN", "green", [ "v0" ] ],

    [ "r1:2", "led1:C", "black", [ "v0" ] ],

    [ "r1:1", "mega:GND.2", "black", [ "h-86.4", "v-74.1" ] ],

    [ "drv1:2B", "stepper1:A-", "green", [ "h0" ] ],

    [ "drv1:2A", "stepper1:A+", "green", [ "h0" ] ],

    [ "stepper1:B+", "drv1:1A", "green", [ "v0" ] ],

    [ "stepper1:B-", "drv1:1B", "green", [ "v0" ] ],

    [ "drv1:SLEEP", "drv1:RESET", "yellow", [ "v0", "h9.6", "v-9.6" ] ],

    [ "vcc1:VCC", "relay1:COM", "red", [ "v0" ] ],

    [ "drv1:DIR", "mega:33", "green", [ "h0" ] ],

    [ "relay1:NC", "drv1:STEP", "green", [ "v0" ] ],

    [ "mega:31", "drv1:STEP", "green", [ "v0" ] ]

  ],

  "dependencies": {}

}