**НП „Обучение за ИТ умения и кариера“**

**Модул 8: Въведение в операционни и вградени системи**

**КУРСОВ ПРОЕКТ**на тема: **Система за умен контрол на температура и осветление на стая**

**Изготвил:**  
Светлозар Сталев

**Преподавател:**

Коля Петрова

**Група 08**

**Документация на „Система за умен контрол на температура и осветление на стая“**

**Съдържание:**

1. Описание на проекта
2. Блокова схема
3. Електрическа схема
4. Списък съставни части
5. Сорс код – описание на функционалността
6. Заключение
7. **Описание на проекта**

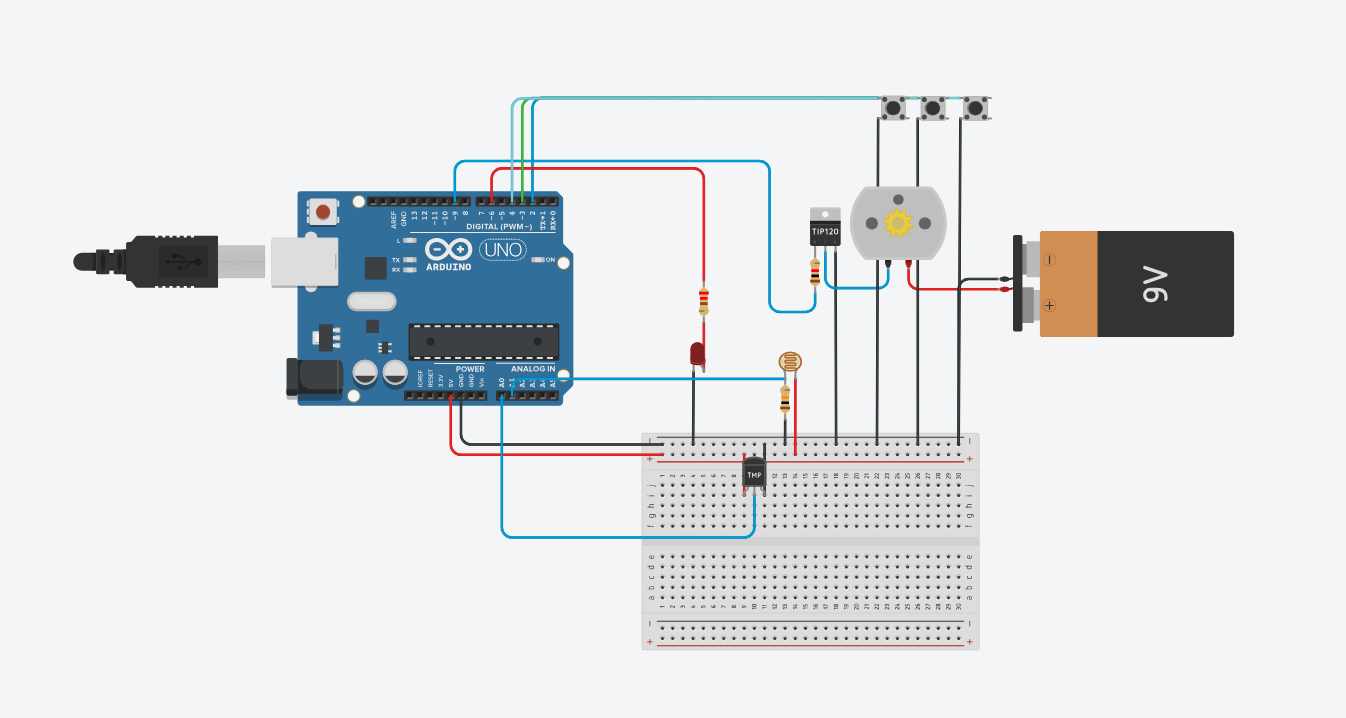
Проектът представлява симулация на автоматизирана система за контрол на светлина и задвижване на мотор, в зависимост от околната осветеност и температура. Целта на системата е да следи параметрите на средата и автоматично да включва LED светлина (която изпълнява ролята на осветление) и DC мотор (който изпълнява ролята на вентилатор) при необходимост.

**Основна функционалност:**  
• **Измерване на осветеност:** Системата използва фоторезистор (LDR), чрез който следи нивото на светлина. При ниска осветеност автоматично се активира светодиод (LED - осветление).  
• **Измерване на температура:** Температурният сензор TMP36 следи околната температура. Когато тя надвиши определена стойност, се активира DC мотор (вентилатор).  
• **Задвижване на мотор:** DC моторът се управлява чрез транзисторен ключ TIP120, който осигурява необходимата мощност при включване.  
• **Ръчно управление:** Добавени са бутони, които могат да служат за ръчно тестване или алтернативно управление на светлината и мотора.

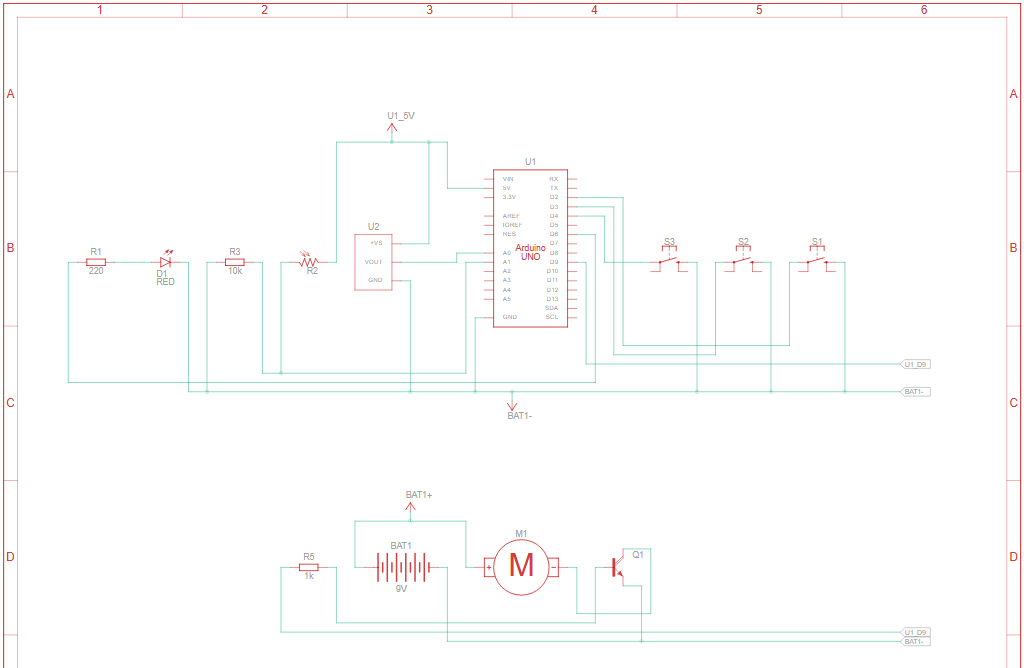
**Управляващ модул:**  
Сърцето на системата е микроконтролерна платка **Arduino Uno**, която чете аналогови стойности от сензорите, обработва ги и подава подходящи сигнали към светлината и мотора.

Системата е проектирана да бъде напълно автономна. При намалена осветеност LED светлината се включва автоматично. При повишена температура се задейства моторът за охлаждане или вентилация. Проектът е реализиран чрез симулация в **Tinkercad** и може лесно да бъде адаптиран за реална употреба в интелигентни домове или технически лаборатории.

1. **Блокова схема**

****

1. **Електрическа схема**

****

|  |  |  |
| --- | --- | --- |
| **Компонент (Component)** | **Пин на Компонента (Pin)** | **Свързан към пин на Arduino (Arduino Pin)** |
| **Arduino UNO (U1)** | **5V** | **Всички VCC пинове на сензори, LDR** |
|  | **GND** | **Всички GND пинове на сензори, LED, TIP120** |
|  | **A0** | **TMP36 – OUT** |
|  | **A1** | **Делител между LDR и 10kΩ резистор** |
|  | **D2** | **Бутон** |
|  | **D3** | **Бутон** |
|  | **D4** | **Бутон** |
|  | **D6** | **LED (анод през резистор)** |
|  | **D9** | **База на TIP120 (през 1kΩ резистор) – за мотора** |
| **Фоторезистор (LDR)** | **Единият край** | **5V** |
|  | **Другият край** | **A1 и 10kΩ към GND** |
| **Резистор (10kΩ за LDR)** | **Единият край** | **A1 (заедно с LDR)** |
|  | **Другият край** | **GND** |
| **Температурен сензор (TMP36)** | **VCC** | **5V** |
|  | **GND** | **GND** |
|  | **OUT** | **A0** |
| **Бутон (Button)** | **Единият край** | **GND** |
|  | **Другият край** | **D2** |
| **Бутон (Button)** | **Единият край** | **GND** |
|  | **Другият край** | **D3** |
| **Бутон (Button)** | **Единият край** | **GND** |
|  | **Другият край** | **D4** |
| **LED (червен)** | **Анод (дълъг пин)** | **D6 през резистор (220Ω)** |
|  | **Катод (къс пин)** | **GND** |
| **Транзистор TIP120** | **База** | **D9 през резистор 1kΩ** |
|  | **Колектор** | **Минус на мотора** |
|  | **Емитер** | **GND** |
| **Мотор (DC мотор)** | **Плюс** | **5V или външно захранване** |
|  | **Минус** | **Колектор на TIP120** |

**4. Списък съставни части**

|  |  |  |
| --- | --- | --- |
| **Name** | **Quantity** | **Component** |
| **U1** | **1** | **Arduino Uno R3** |
| **U2** | **1** | **Temperature Sensor [TMP36]** |
| **D1** | **1** | **Red LED** |
| **R1** | **1** | **220 Ω Resistor** |
| **R2** | **1** | **Photoresistor (LDR)** |
| **R3** | **1** | **10 kΩ Resistor** |
| **M1** | **1** | **DC Motor** |
| **Q1** | **1** | **TIP120 Transistor** |
| **BAT1** | **1** | **9V Battery** |
| **R5** | **1** | **1 kΩ Resistor** |
| **S1** | **3** | **Pushbutton** |
| **S2** | **3** | **Pushbutton** |
| **S3** | **3** | **Pushbutton** |

**5. Сорс код – описание на функционалността**

// Дефиниции на пиновете

const int TEMP\_SENSOR\_PIN = A0;

const int LDR\_PIN = A1;

const int FAN\_PIN = 9;

const int LED\_PIN = 6;

const int BUTTON\_PIN = 2;

const int BUTTON\_FAN\_PIN = 3;

const int BUTTON\_LED\_PIN = 4;

// Константи за настройки

const float TEMP\_THRESHOLD = 28.0;

const int LIGHT\_THRESHOLD = 300;

const int FAN\_SPEED = 200;

const int LED\_BRIGHTNESS = 150;

// Променливи за състояние

bool fanOn = false;

bool ledOn = false;

bool manualMode = false;

unsigned long lastButtonPress = 0;

const unsigned long DEBOUNCE\_DELAY = 50;

int buttonFanState = HIGH;

int lastButtonFanState = HIGH;

int buttonLedState = HIGH;

int lastButtonLedState = HIGH;

unsigned long lastFanButtonPress = 0;

unsigned long lastLedButtonPress = 0;

// Променливи за сензори

float temperature = 0.0;

int lightLevel = 0;

int buttonState = HIGH;

int lastButtonState = HIGH;

void setup() {

Serial.begin(9600);

Serial.println("=== Smart Room Control ===");

Serial.println("System starting...");

pinMode(FAN\_PIN, OUTPUT);

pinMode(LED\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

pinMode(BUTTON\_FAN\_PIN, INPUT\_PULLUP);

pinMode(BUTTON\_LED\_PIN, INPUT\_PULLUP);

digitalWrite(FAN\_PIN, LOW);

digitalWrite(LED\_PIN, LOW);

Serial.println("System is ready!");

Serial.println("Temperature threshold: " + String(TEMP\_THRESHOLD) + "°C");

Serial.println("Light threshold: " + String(LIGHT\_THRESHOLD));

Serial.println("------------------------");

}

void loop() {

// Четене на сензорите

readSensors();

// Проверка на бутона

checkButton();

checkManualControlButtons();

// Автоматично управление (ако не е в ръчен режим)

if (!manualMode) {

automaticControl();

}

// Показване на информация

displayStatus();

// Кратка пауза

delay(1000);

}

void readSensors() {

// Четене на температурата

int tempReading = analogRead(TEMP\_SENSOR\_PIN);

// Конвертиране за TMP36 сензор

float voltage = (tempReading \* 5.0) / 1024.0;

temperature = (voltage - 0.5) \* 100.0;

// Четене на светлината

lightLevel = analogRead(LDR\_PIN);

}

void checkButton() {

int reading = digitalRead(BUTTON\_PIN);

// Debounce логика

if (reading != lastButtonState) {

lastButtonPress = millis();

}

if ((millis() - lastButtonPress) > DEBOUNCE\_DELAY) {

if (reading != buttonState) {

buttonState = reading;

if (buttonState == LOW) {

toggleManualMode();

}

}

}

lastButtonState = reading;

}

void checkManualControlButtons() {

if (manualMode) {

// --- ВЕНТИЛАТОР БУТОН ---

int fanReading = digitalRead(BUTTON\_FAN\_PIN);

if (fanReading != lastButtonFanState) {

lastFanButtonPress = millis();

}

if ((millis() - lastFanButtonPress) > DEBOUNCE\_DELAY) {

if (fanReading != buttonFanState) {

buttonFanState = fanReading;

if (buttonFanState == LOW) {

if (fanOn) {

turnOffFan();

} else {

turnOnFan();

}

}

}

}

lastButtonFanState = fanReading;

// --- LED БУТОН ---

int ledReading = digitalRead(BUTTON\_LED\_PIN);

if (ledReading != lastButtonLedState) {

lastLedButtonPress = millis();

}

if ((millis() - lastLedButtonPress) > DEBOUNCE\_DELAY) {

if (ledReading != buttonLedState) {

buttonLedState = ledReading;

if (buttonLedState == LOW) {

if (ledOn) {

turnOffLED();

} else {

turnOnLED();

}

}

}

}

lastButtonLedState = ledReading;

}

}

void toggleManualMode() {

manualMode = !manualMode;

if (manualMode) {

Serial.println(">>> MANUAL MODE ACTIVATED <<<");

// Изключване на всички устройства в ръчен режим

turnOffFan();

turnOffLED();

} else {

Serial.println(">>> AUTOMATIC MODE ACTIVATED <<<");

}

}

void automaticControl() {

// Управление на вентилатора според температурата

if (temperature > TEMP\_THRESHOLD && !fanOn) {

turnOnFan();

} else if (temperature <= (TEMP\_THRESHOLD - 2.0) && fanOn) {

turnOffFan();

}

// Управление на LED според светлината

if (lightLevel < LIGHT\_THRESHOLD && !ledOn) {

turnOnLED();

} else if (lightLevel > (LIGHT\_THRESHOLD + 50) && ledOn) {

turnOffLED();

}

}

void turnOnFan() {

fanOn = true;

analogWrite(FAN\_PIN, FAN\_SPEED);

Serial.println("Fan is ON");

}

void turnOffFan() {

fanOn = false;

analogWrite(FAN\_PIN, 0);

Serial.println("Fan is OFF");

}

void turnOnLED() {

ledOn = true;

analogWrite(LED\_PIN, LED\_BRIGHTNESS);

Serial.println("LED light is ON");

}

void turnOffLED() {

ledOn = false;

analogWrite(LED\_PIN, 0);

Serial.println("LED light is OFF");

}

void displayStatus() {

Serial.println("------------------------");

Serial.println("Mode: " + String(manualMode ? "MANUAL" : "AUTOMATIC"));

Serial.println("Temperature: " + String(temperature, 1) + "°C");

Serial.println("Light level: " + String(lightLevel) + " (0-1023)");

Serial.println("Fan: " + String(fanOn ? "ON" : "OFF"));

Serial.println("LED: " + String(ledOn ? "ON" : "OFF"));

// Предупреждения

if (temperature > TEMP\_THRESHOLD + 5) {

Serial.println("WARNING: Temperature too high!");

}

if (lightLevel < 100) {

Serial.println("WARNING: Room is too dark");

}

Serial.println("========================");

}

void calibrateSensors() {

Serial.println("=== SENSOR CALIBRATION ===");

int tempSum = 0;

int lightSum = 0;

int samples = 10;

for (int i = 0; i < samples; i++) {

tempSum += analogRead(TEMP\_SENSOR\_PIN);

lightSum += analogRead(LDR\_PIN);

delay(100);

}

int avgTemp = tempSum / samples;

int avgLight = lightSum / samples;

Serial.println("Average temperature value: " + String(avgTemp));

Serial.println("Average light value: " + String(avgLight));

Serial.println("===============================");

}

**Глобални променливи:**

* **TEMP\_SENSOR\_PIN, LDR\_PIN, FAN\_PIN, LED\_PIN, BUTTON\_PIN, BUTTON\_FAN\_PIN, BUTTON\_LED\_PIN:  
  Дефинират пиновете, към които са свързани температурният сензор, сензорът за светлина, вентилаторът, LED осветлението и бутоните.**
* **TEMP\_THRESHOLD, LIGHT\_THRESHOLD:  
  Прагова стойност за температура и светлина, използвана за автоматично включване на устройства.**
* **FAN\_SPEED, LED\_BRIGHTNESS:  
  Задават PWM стойности за интензитета на вентилатора и LED осветлението.**
* **fanOn, ledOn, manualMode:  
  Булеви флагове, указващи състоянието на вентилатора, LED-а и активния режим (автоматичен/ръчен).**
* **lastButtonPress, lastFanButtonPress, lastLedButtonPress:  
  Използват се за debounce логика при работа с бутоните.**
* **temperature, lightLevel:  
  Текущо измерени стойности от температурния сензор и LDR.**

**• Функция setup():**

* **Изпълнява се еднократно при стартиране на микроконтролера.**
* **Инициализира серийния монитор, пиновете и началните състояния на вентилатора и LED-а.**
* **Извежда в серийния монитор начална информация за системата и зададените прагове.**

**• Функция loop():**

* **Изпълнява се непрекъснато в цикъл.**
* **Основни стъпки:**
  1. **Четене на сензорите чрез readSensors() (температура и осветеност).**
  2. **Проверка за натискане на бутона за смяна на режим – чрез checkButton().**
  3. **Ако е активиран ръчен режим, проверява бутоните за ръчно включване/изключване на вентилатора и LED-а – чрез checkManualControlButtons().**
  4. **Ако е в автоматичен режим, стартира автоматичното управление – чрез automaticControl().**
  5. **Извежда текущия статус на системата в серийния монитор – чрез displayStatus().**

**• Функция readSensors():**

* **Измерва аналоговата стойност от температурния сензор и я преобразува в градуси по Целзий (за TMP36).**
* **Чете светлината от фоторезистора и я съхранява като стойност от 0 до 1023.**

**• Функция checkButton():**

* **Следи бутон за превключване между автоматичен и ръчен режим.**
* **Използва debounce механизъм, за да избегне фалшиви натискания.**
* **При натискане се извиква toggleManualMode().**

**• Функция checkManualControlButtons():**

* **Активна само в ръчен режим.**
* **Проверява отделните бутони за:**
  + **Включване/изключване на вентилатора (пин 3).**
  + **Включване/изключване на LED осветлението (пин 4).**
* **Всеки бутон използва независима debounce логика.**
* **При натискане извиква turnOnFan(), turnOffFan(), turnOnLED() или turnOffLED().**

**• Функция toggleManualMode():**

* **Превключва между ръчен и автоматичен режим.**
* **При включване на ръчен режим изключва всички устройства.**
* **Извежда съобщение в серийния монитор.**

**• Функция automaticControl():**

* **Стартира се само ако не е активиран ръчен режим.**
* **Управлява вентилатора:**
  + **Включва при температура > TEMP\_THRESHOLD.**
  + **Изключва при температура < TEMP\_THRESHOLD - 2.0 (хистерезис).**
* **Управлява LED осветлението:**
  + **Включва при ниска осветеност (< LIGHT\_THRESHOLD).**
  + **Изключва при достатъчна светлина (> LIGHT\_THRESHOLD + 50).**

**• Функции turnOnFan(), turnOffFan(), turnOnLED(), turnOffLED():**

* **Управляват съответно включване и изключване на вентилатора и LED-а.**
* **Използват analogWrite() за регулиране на мощността.**
* **Показват статус в серийния монитор.**

**• Функция displayStatus():**

* **Показва в серийния монитор:**
  + **Текущ режим.**
  + **Температура и осветеност.**
  + **Състояние на вентилатора и LED-а.**
  + **Предупреждения при висока температура или ниска осветеност.**

**• Функция calibrateSensors() *(не се използва в основния цикъл)*:**

* **Провежда 10 последователни измервания на температура и светлина.**
* **Изчислява и извежда средните стойности.**
* **Подходяща за калибриране на сензорите при настройка.**

**6. Заключение**

Разработеният проект успешно симулира работата на автоматизирана система за контрол на светлина и мотор (вентилатор). Постигнати са основните цели: надеждно отчитане на нивото на осветеност и температурата, автоматично управление на LED осветление и мотор (вентилатор) чрез транзисторен ключ. Системата демонстрира стабилна работа и реална приложимост в сферата на домашната автоматизация и енергийно ефективни решения.

1. **Връзки към проекта**

Tinkercad: <https://www.tinkercad.com/things/7euqibyS97l-smart-home>

Github: <https://github.com/TheCrock1/SmartHome>