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

гр. Хасково
2024/2025 г.

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

Съдържание:

1. 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.

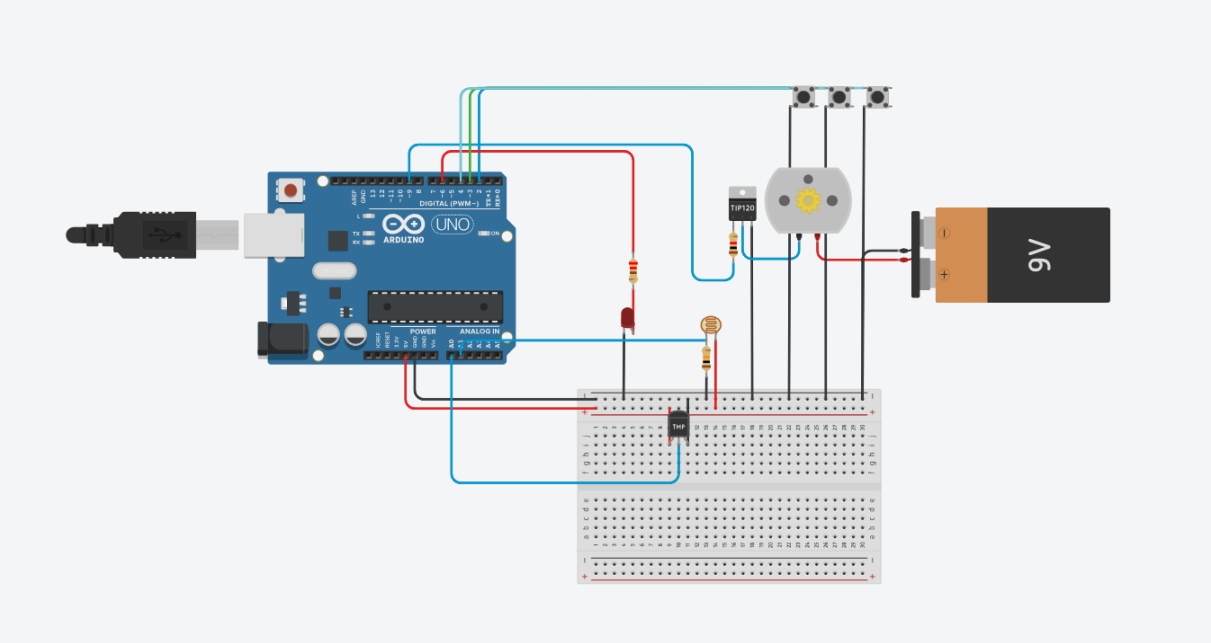
Основна функционалност:
• **Измерване на осветеност:** Системата използва фоторезистор (LDR), чрез който следи нивото на светлина. При ниска осветеност автоматично се активира светодиод (LED - осветление).
• **Измерване на температура:** Температурният сензор TMP36 следи околната температура. Когато 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.

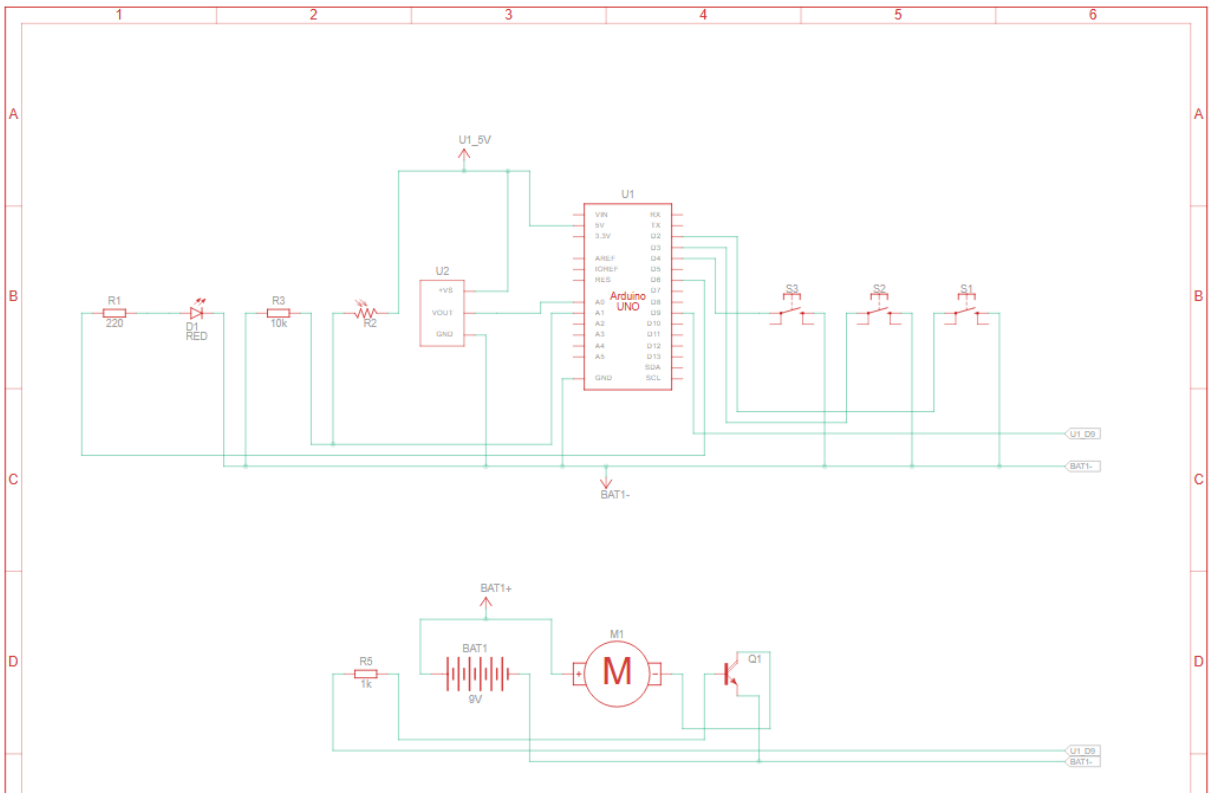
Системата е проектирана да бъде напълно автономна. При намалена осветеност LED светлината се включва автоматично. При повишена температура се задейства моторът за охлаждане или вентилация. Проектът е реализиран чрез симулация в **Tinkercad** и

може лесно да бъде адаптиран за реална употреба в интелигентни домове или технически лаборатории.

2. Блокова схема



3. 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("=== Умен контрол на стая ===");
  Serial.println("Стартиране на системата...");

  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("Системата е готова!");
```

```
  Serial.println("Температурен праг: " + String(TEMP_THRESHOLD)
+ "°C");
  Serial.println("Светлинен праг: " + 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(">>> РЪЧЕН РЕЖИМ АКТИВИРАН <<<");
    // Изключване на всички устройства в ръчен режим
    turnOffFan();
```

```
    turnOffLED();
  } else {
    Serial.println(">>> АВТОМАТИЧЕН РЕЖИМ АКТИВИРАН <<<");
  }
}

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("Вентилаторът е ВКЛЮЧЕН");
}

void turnOffFan() {
  fanOn = false;
  analogWrite(FAN_PIN, 0);
  Serial.println("Вентилаторът е ИЗКЛЮЧЕН");
}

void turnOnLED() {
  ledOn = true;
  analogWrite(LED_PIN, LED_BRIGHTNESS);
  Serial.println("LED осветлението е ВКЛЮЧЕНО");
}

void turnOffLED() {
  ledOn = false;
  analogWrite(LED_PIN, 0);
  Serial.println("LED осветлението е ИЗКЛЮЧЕНО");
}

void displayStatus() {
  Serial.println("------------------------");
  Serial.println("Режим: " + String(manualMode ? "РЪЧЕН" :
"АВТОМАТИЧЕН"));
  Serial.println("Температура: " + String(temperature, 1) +
"°C");
  Serial.println("Светлина: " + String(lightLevel) + " (0-
1023)");
```

```
  Serial.println("Вентилатор: " + String(fanOn ? "ВКЛ" :
"ИЗКЛ"));
  Serial.println("LED: " + String(ledOn ? "ВКЛ" : "ИЗКЛ"));

  // Предупреждения
  if (temperature > TEMP_THRESHOLD + 5) {
    Serial.println("ВНИМАНИЕ: Много висока температура!");
  }

  if (lightLevel < 100) {
    Serial.println("Много тъмно в помещението");
  }

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

void calibrateSensors() {
  Serial.println("=== КАЛИБРИРАНЕ НА СЕНЗОРИ ===");

  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("Средна стойност температура: " +
String(avgTemp));
  Serial.println("Средна стойност светлина: " +
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.

7. Връзки към проекта

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

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