Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

ЗВІТ

з лабораторної роботи №4

з дисципліни «Розробка застосунків інтернету речей та сенсорних мереж»

Тема: «Дослідження аналого-цифрового перетворення і розробка взаємодії мікроконтролера з датчиками на платформі Arduino»

Варіант №17

Виконав:
Студент групи ТР-12
Ковальов Олександр Олексійович

Дата здачі: 04.03.2025

КИЇВ – 2025

Мета роботи. Дослідити концепцію і принципи функціонування аналогоцифрового перетворювача (АЦП), а також розробити схему для взаємодії мікроконтролера з датчиками, використовуючи платформу Arduino.

Індивідуальне завдання:

1) Ознайомлення. Провести детальне вивчення поняття аналогоцифрового перетворювача (АЦП) і засвоїти основні принципи його функціонування.
2) Використовуючи вбудований АЦП у платформі Arduino, провести конвертацію вхідного напругового значення в бітовий формат. Перетворити отримане значення в процентне відношення до максимально можливого значення. Вивести результати вимірювань на цифровий екран для подальшого аналізу.
3) Виконати згідно з непарним варіантом (17) – розробити схему для взаємодії з датчиком температури і модулем годинника. Програмно аналізувати дані з датчика і виводити на екран інформацію про температурний стан. Якщо температура нижче 16 градусів – вивести «Холодно», якщо вище 28 – «Жарко», в іншому випадку – «Прийнятно». Частота оновлення результатів даних – один раз за хвилину.

Хід роботи.

Першим завданням є відображення конвертації напруги. Відповідно, був написаний код призначений зчитувати аналоговий сигнал з датчика, підключеного до піну A0, та відображати отримані значення на OLED-дисплеї SSD1306.

Спочатку підключаються необхідні бібліотеки: Wire.h для роботи з шиною I2C, Adafruit_GFX.h для графічних операцій та Adafruit_SSD1306.h для керування дисплеєм. Визначаються розміри екрану та задається макрос ADC_PIN, що відповідає аналоговому входу A0. Створюється об'єкт display, який представляє OLED-дисплей.

У функції setup() ініціалізується дисплей. Якщо ініціалізація не вдається, програма зупиняється у безкінечному циклі. Дисплей очищується, встановлюється розмір шрифту 1 та колір тексту білий.

У функції loop() виконується зчитування аналогового значення з піну A0. Отримане значення перетворюється у напругу за допомогою формули voltage = adcValue * (5.0 / 1023.0), де 5.0 В – це опорна напруга, а 1023 – максимальне значення АЦП. Також обчислюється відсоткове значення рівня сигналу відносно 1023 за формулою (adcValue / 1023.0) * 100.0.

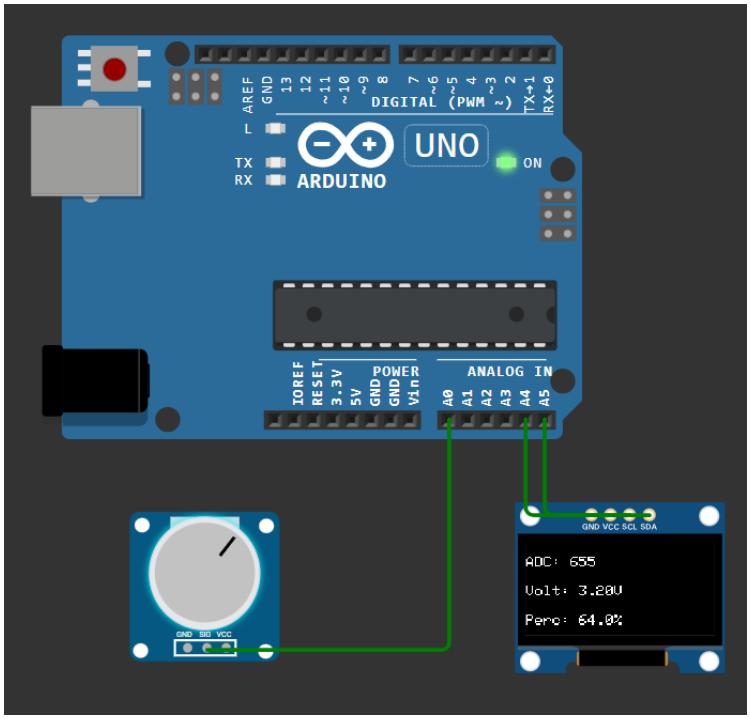
Далі очищується дисплей, встановлюється позиція курсора та виводяться три рядки тексту: значення АЦП, напруга та відсотковий еквівалент. Після цього дисплей оновлюється методом display.display(), а перед наступним зчитуванням даних виконується пауза у 500 мс.

Код:

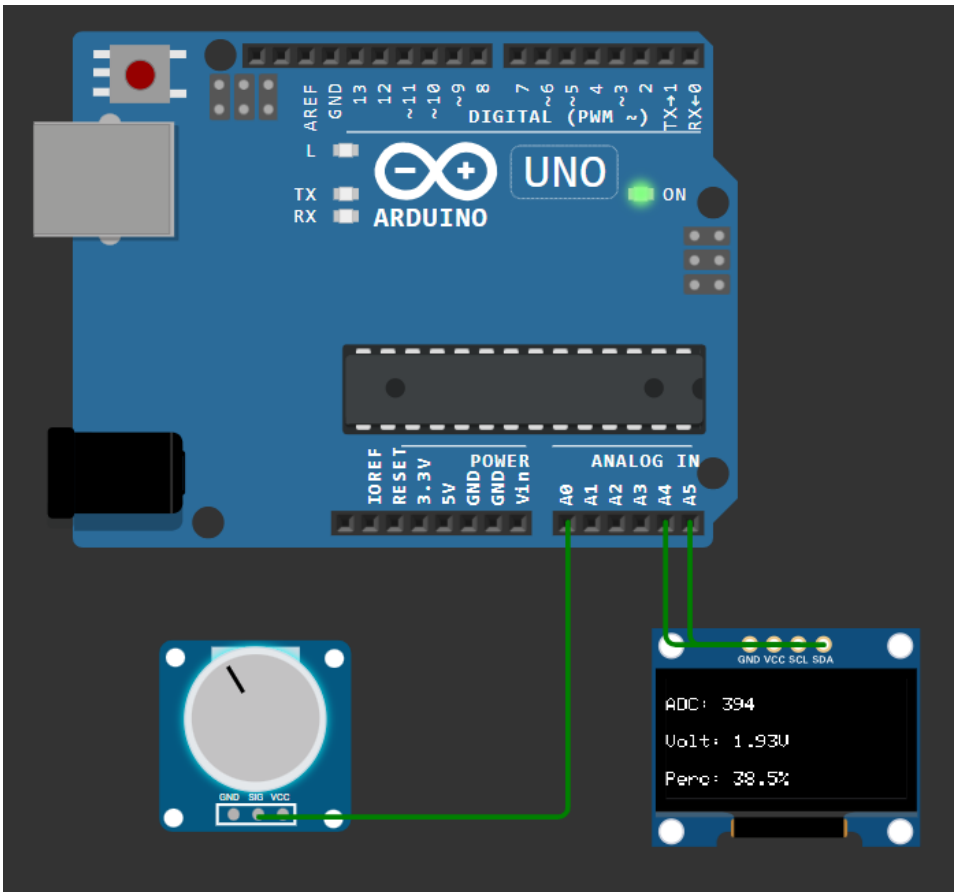
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define ADC_PIN A0
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        for (;;);
    }

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
}

void loop() {
    int adcValue = analogRead(ADC_PIN);
    float voltage = adcValue * (5.0 / 1023.0);
    float percentage = (adcValue / 1023.0) * 100.0;

    display.clearDisplay();
    display.setCursor(0, 10);
    display.setTextSize(1);
    display.print("ADC: ");
    display.println(adcValue);
    display.setCursor(0, 30);
    display.setTextSize(1);
    display.print("Volt: ");
    display.print(voltage, 2);
    display.println("V");

    display.setCursor(0, 50);
    display.setTextSize(1);
    display.print("Perc: ");
    display.print(percentage, 1);
    display.println("%");

    display.display();

    delay(500);
}
```

Відповідно, результат при, наприклад, 3.20 V:



При 1.93 V:



Друге завдання – розробити схему для взаємодії з датчиком температури і модулем годинника, програмно аналізувати дані з датчика і виводити на екран інформацію про температурний стан. Якщо температура нижче 16 градусів – вивести «Холодно», якщо вище 28 – «Жарко», в іншому випадку – «Прийнятно». Частота оновлення результатів даних – один раз за хвилину.

Був написаний код призначений для отримання поточного часу з модуля RTC DS1307, вимірювання температури за допомогою датчика DHT22 та виведення отриманих даних на OLED-дисплей SSD1306, а також у серійний монітор.

На початку підключаються необхідні бібліотеки: Wire.h для роботи з шиною I2C, RTClib.h для роботи з модулем реального часу, Adafruit_SSD1306.h для керування OLED-дисплеєм і DHT.h для роботи з температурним датчиком. Оголошується об'єкт display для OLED-дисплея з розмірами 128x64 пікселів, об'єкт rtc для модуля RTC та об'єкт dht для датчика температури, підключеного до цифрового піну 2.

У функції setup() ініціалізується серійний зв’язок зі швидкістю 9600 бод, OLED-дисплей і модуль RTC. Якщо дисплей або RTC не знайдені, виводиться повідомлення в серійний монітор, і програма зупиняється. Якщо RTC не працює, встановлюється час компіляції. Також ініціалізується датчик DHT22.

У функції loop() отримується поточний час з модуля RTC та вимірюється температура з датчика DHT22. Якщо датчик не може передати значення, виводиться повідомлення про помилку, і функція переривається.

Далі аналізується температура: якщо вона нижча за 16°C, встановлюється статус "Cold", якщо вища за 28°C — "Hot", в іншому випадку — "Comfortable". Отримані значення температури та статусу виводяться в серійний монітор.

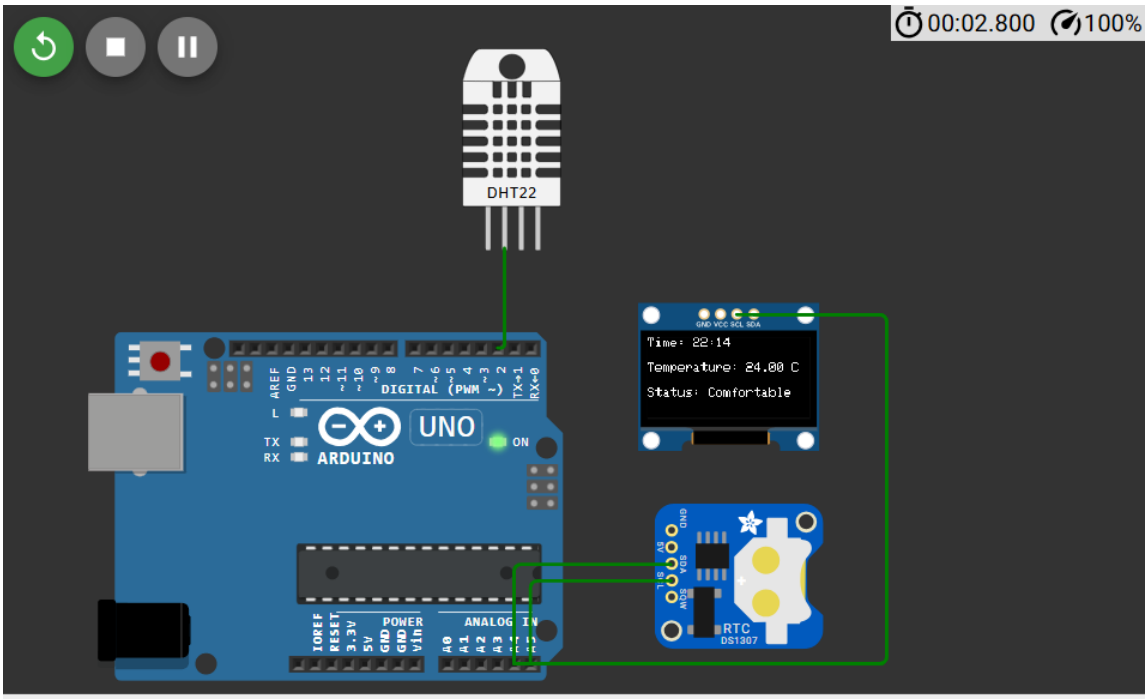
На дисплей виводиться поточний час у форматі ГГ:ХХ, значення температури з одиницями вимірювання С, а також статус температури. Дані оновлюються методом display.display(). Після цього програма чекає 5 секунд перед наступним вимірюванням.

Код:

```
#include <Wire.h>
#include <RTClib.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

RTC_DS1307 rtc;

#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("OLED display not found");
    while (1);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  if (!rtc.begin()) {
    Serial.println("RTC module not found!");
    while (1);
  }

  if (!rtc.isrunning()) {
    Serial.println("RTC is not set! Setting to compile time...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  dht.begin();
}

void loop() {
  DateTime now = rtc.now();
  float temperature = dht.readTemperature();

  if (isnan(temperature)) {
    Serial.println("DHT reading error");
    return;
  }

  String state;
  if (temperature < 16) {
    state = "Cold";
  } else if (temperature > 28) {
    state = "Hot";
  } else {
    state = "Comfortable";
  }
```

```
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" C, Status: ");
  Serial.println(state);

  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("Time: ");
  display.print(now.hour());
  display.print(":");

  if (now.minute() < 10) display.print("0");
  display.print(now.minute());

  display.setCursor(0, 20);
  display.print("Temperature: ");
  display.print(temperature);
  display.print(" C");

  display.setCursor(0, 40);
  display.print("Status: ");
  display.print(state);

  display.display();

  delay(60000);
}
```

Демонстрація роботи:



Висновок: У ході виконання лабораторної роботи було досліджено принципи функціонування аналогоцифрового перетворювача (АЦП) та його застосування в мікроконтролерах на базі платформи Arduino. Було розглянуто методи конвертації аналогового сигналу у цифровий формат та проведено практичну реалізацію зчитування вхідної напруги за допомогою вбудованого АЦП.

Розроблено схему взаємодії мікроконтролера з датчиком температури DHT22 та модулем годинника реального часу RTC DS1307. Програмно реалізовано аналіз отриманих даних, визначено логіку обробки температурних показників та їхнього виводу на OLED-дисплей. Відповідно до індивідуального завдання, розроблено алгоритм класифікації температурного стану, який дозволяє інформувати користувача про рівень комфорту навколишнього середовища.

В результаті проведених експериментів отримано працездатну систему, що періодично вимірює температуру, визначає поточний час та виводить отриману інформацію на екран. Це дозволило не лише закріпити теоретичні знання про АЦП, а й отримати практичний досвід роботи з периферійними модулями та розширити навички програмування мікроконтролерів.

Контрольні питання:

1. *Що таке АЦП і як він працює в контексті мікроконтролерів, таких як Arduino?*

   Аналогоцифровий перетворювач (АЦП) – це електронний компонент, що конвертує аналогові сигнали (наприклад, напругу) у цифрові значення. У мікроконтролерах, таких як Arduino, АЦП використовується для зчитування рівня напруги з аналогових датчиків. Він працює шляхом дискретизації вхідного сигналу та його перетворення у цифровий код відповідно до заданої розрядності та опорної напруги.
2. *Опишіть, як визначається роздільна здатність АЦП і як це впливає на точність вимірювань.*

   Роздільна здатність АЦП визначається кількістю бітів, які використовуються для представлення результату вимірювання. В Arduino Uno використовується 10-бітний АЦП, що означає розбиття діапазону вимірювань на 1024 рівні (від 0 до 1023). Чим вища роздільна здатність, тим точніше можна виміряти невеликі зміни сигналу, але збільшується час перетворення та обсяг обчислень.
3. *Які три режими опорної напруги доступні в Arduino і як кожен з них впливає на вимірювання АЦП?*

   В Arduino доступні три режими опорної напруги: DEFAULT (за замовчуванням 5V або 3.3V залежно від плати), INTERNAL (внутрішня опорна напруга, наприклад, 1.1V для ATmega328P) і EXTERNAL (зовнішня напруга, підключена до піну AREF). Вибір опорної напруги впливає на діапазон вимірювань: нижча опорна напруга підвищує точність вимірювання малих сигналів, а вищі значення дозволяють працювати з ширшим діапазоном напруг.
4. *Які фактори необхідно враховувати при виборі вхідного піна для аналогових вимірювань на Arduino?*

   При виборі вхідного аналогового піна необхідно враховувати його електричні характеристики, наявність шумів, внутрішній опір джерела сигналу та необхідність використання конденсаторів для згладжування коливань. Важливо уникати одночасного використання пінів для цифрового та аналогового вводу/виводу, щоб не спричинити конфлікти сигналів.

5. *Які основні принципи роботи шини I2C та її роль у комунікації між компонентами в системах на базі Arduino?*
   Шина I2C (Inter-Integrated Circuit) – це двопровідний протокол зв'язку, який використовується для обміну даними між мікроконтролерами та периферійними пристроями. Вона складається з ліній SDA (дані) та SCL (синхронізація) і підтримує підключення декількох пристроїв на одній лінії. I2C дозволяє передавати дані з використанням унікальних адрес пристроїв, що робить її ефективною для зв'язку з датчиками, екранами та іншими модулями в Arduino.
6. *Обговоріть різні методи калібрування сенсорів, які підключаються до Arduino, і як це впливає на точність даних.*
   Калібрування сенсорів може включати методи програмного коригування, використання відомих еталонних значень або фізичне налаштування датчика. Наприклад, температурні сенсори калібрують шляхом порівняння з еталонним термометром і введення поправочних коефіцієнтів. Коректна калібровка зменшує похибки вимірювань і підвищує точність отриманих даних.
7. *Які типові застосування датчиків температури в системах на базі Arduino?*
   Датчики температури широко використовуються в автоматизованих системах моніторингу, кліматичних контролерах, розумних будинках, системах безпеки, сільському господарстві та біомедичних приладах. У проектах на базі Arduino вони можуть застосовуватися для керування вентиляцією, підтримки стабільної температури обладнання або створення системи сигналізації у разі критичних змін температури.