**Національний технічний університет України**

**«Київський політехнчний 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»**

Варіант №16

Виконав:

студент групи ТР-23

Ровний Г.О.

Дата здачі: 17.09.2025

КИЇВ – 2025

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

**Поставлене завдання:**

1) Ознайомлення. Провести детальне вивчення поняття аналого цифрового 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 варіантом: — парні варіанти (2, 4, 6…). Розробити схему взаємодії мікроконтролера з датчиком відстані та модуля годинника. Програмно реалізувати зчитування даних з датчика і виведення їх на цифровий екран. Частота оновлення результатів даних — один раз за хвилину. Якщо об’єкт розміщується на відстані до 2 метрів, вивести повідомлення про близькість, в іншому випадку — про віддаленість.

**Результат виконання роботи**

**Завдання Перше**

Для виконання першого завдання спершу треба створити схему з використанням контролера Arduino і штучного джерела живлення. Необхідно з’єднати його з аналоговим входом ардуіно.

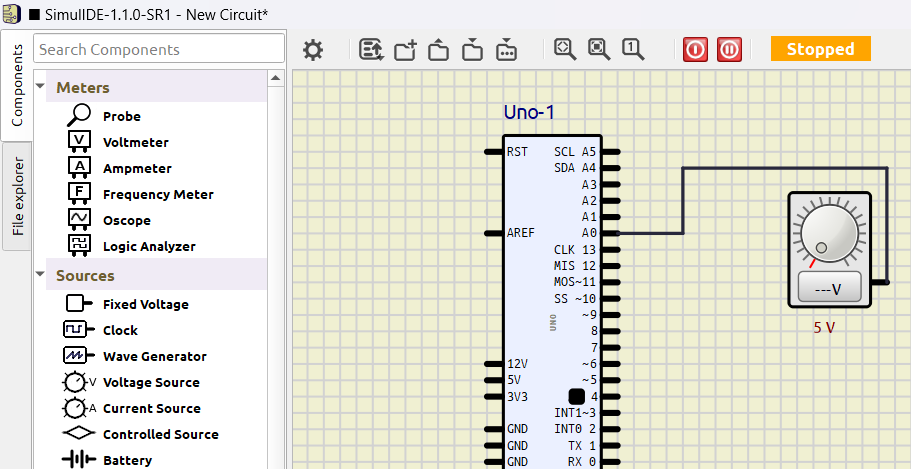
****

Рис. 4.1. Джерело штучного живлення, під’єднане до плати ардуіно

Необхідно зауважити, що аналоговий роз’єм Arduino має допустиме отримане значення від 0 до 1023, що означає пропускний обсяг до 10 біт. Схема АЦП в Arduino дає можливість обирати джерело опорної напруги (для встановлення допустимих меж вимірювання). Для зміни цього параметра існує метод analogReference(mode), де mode — один з трьох доступних режимів: DEFAULT — напруга джерела живлення контролера (5 Вольт); INTERNAL — напруга стабілізатора контролера (1.1 Вольта); EXTERNAL — зовнішне джерело опорної напруги. Для цього необхідно подати напругу до 5 Вольт на порт контролера AREF (рис. 4.8). Зчитування даних з порту відбувається за допомогою методу analogRead(pin), де pin — це номер аналогового порту. Отримане в результаті роботи АЦП побітове значення автоматично конвертується в десятковий вигляд (від 0 до 1023). Конвертація відбувається за допомогою взаємодії зі знайденою одиницею АЦП.

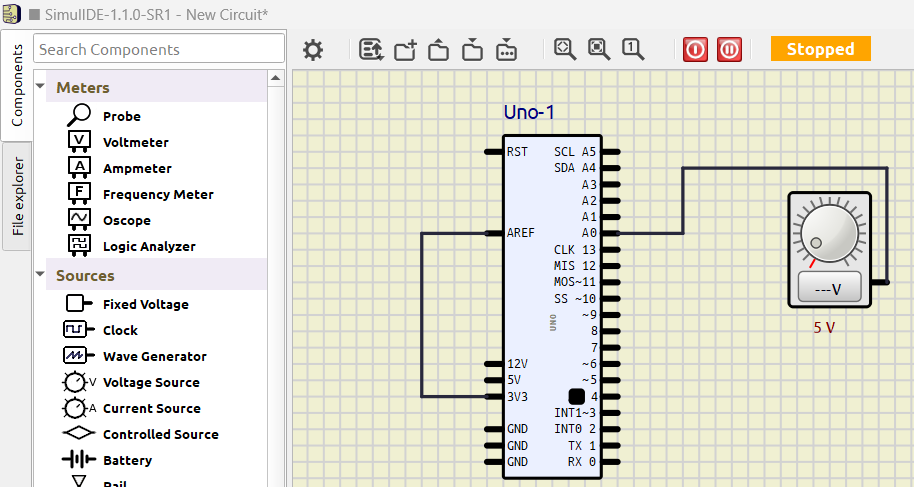


Рис. 4.2. Приклад використання зовнішнього джерела опорної напруги (3.3 Вольта)

Наприклад: — знайдена напруга ((Вольт) \* (Отримане значення АЦП)) / (Загальне значення АЦП(1023)). — відсоткове відношення ((100%) \* (Отримане значення АЦП)) / (Загальне значення АЦП(1023)). — переведення в n біт ((2^n) \* (Отримане значення АЦП)) / (Загальне значення АЦП(1023)). Вище подано текст, який описує процес конвертації аналогових значень, зчитаних з датчика за допомогою аналого-цифрового перетворювача (АЦП) мікроконтролера Arduino, у різні формати. Цей процес необхідний, тому що АЦП перетворює аналоговий сигнал (який може мати будь-яке значення в межах певного діапазону напруги) у цифрове значення з фіксованим діапазоном (для Arduino, як правило, від 0 до 1023 для 10-бітного АЦП).

1. Знайдена напруга. Ця формула перетворює цифрове значення, отримане від АЦП, назад у напругу. Це робиться за допомогою множення значення АЦП на максимально можливу напругу, яку може виміряти АЦП (як правило, 5 В для Arduino), і поділу результату на максимальне значення АЦП (1023). Таким чином, якщо АЦП повертає 512, це відповідатиме половині максимальної напруги.

2. Відсоткове відношення. Ця формула перетворює значення АЦП у відсотки від максимально можливого значення. Це корисно, коли потрібно визначити, яка частка максимального сигналу подана поточним зчитуванням. Наприклад, якщо АЦП повертає 512, це буде 50% від максимального сигналу.

3. Переведення в n бітів. Ця формула дає можливість перетворити значення АЦП у діапазон, визначений кількістю бітів. Це може бути корисним для інтерфейсів, які приймають дані у форматах, менших за 10 бітів. Наприклад, якщо потрібно перетворити значення АЦП у 8-бітне значення, використовують (28), щоб отримати значення від 0 до 255.

Ці формули важливі для різних застосувань, таких як калібрування сенсорів, перетворення даних для відображення, або для подальшої цифрової обробки у мікроконтролері чи комп’ютері. Вони дають можливість працювати з аналоговими входами мікроконтролера більш гнучко, адаптуючи вхідні дані до потреб конкретної задачі.

**Для другої частини першого завдання** необхідно розмістити на схемі екран (ssd1306) і з’єднати його шинами стандарту i2o із мікросхемою:

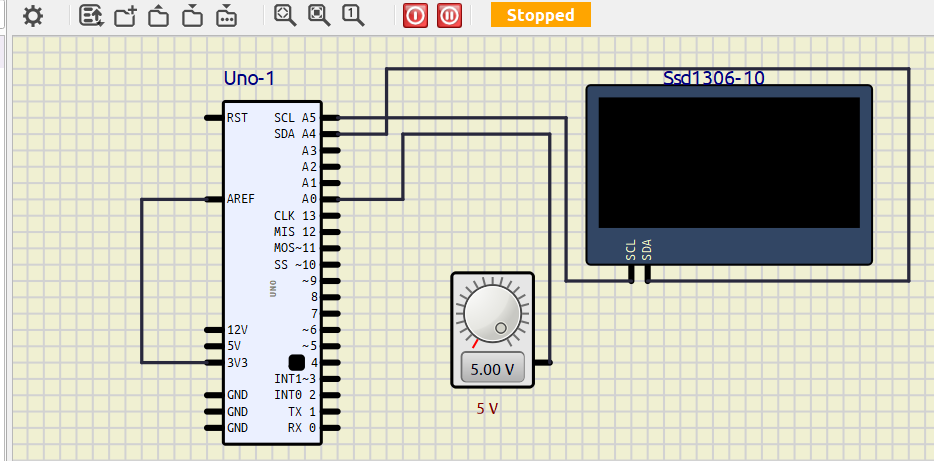


Рис 4.3. Схема з використанням контролера, штучного джерела живлення та екрана

Екран типу (ssd1306) при роботі вимагає наявності відповідних бібліотек. Як приклад було використано бібліотеку Adafruit\_SSD1306. Для відображення на екрані необхідно виконати такі дії:

— ініціалізувати екземпляр класу Adafruit\_SSD1306, конструктор якого приймає два параметри: Ширина екрана та Висота \*Примітка: ssd1306 є розміри 128х32\* ;

— увімкнути передачу даних дисплею за допомогою методу класу begin() ;

— виставити колір тексту за допомогою методу класу setTextColor(COLOR), де COLOR — це назва кольору;

— для кожної операції відображення виставити координати курсору введення за допомогою методу класу setCursor(Х, У), де Х та У — це відповідні координати;

— надіслати необхідне повідомлення до буфера екрана за допомогою методу класу print(MASSAGE), де MASSAGE — повідомлення типу string;

— після заповнення буфера відобразити зміни на екрані за допомогою методу класу display() ;

У результаті виконання має бути створена схема обчислення вхідної напруги і виведення результатів на екран. Напишемо наступний код:

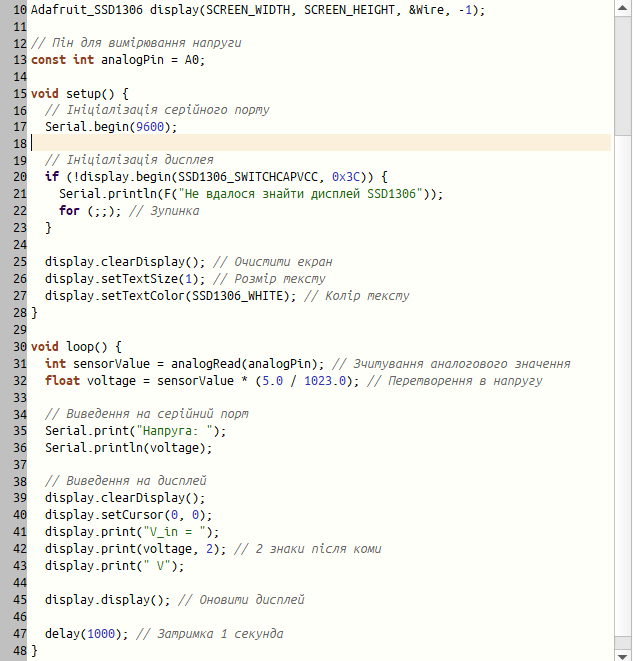


Рис 4.4. Код для виведення результату на екран

Спочатку ми підключаємо бібліотеки Adafruit\_GFX та Adafruit\_SSD1306. Вони дозволяють нам спілкуватися з OLED-дисплеєм, передавати йому текст, координати, кольори — усе, що потрібно для виводу інформації.

Ми створюємо об’єкт display, вказуючи розміри екрана — 128 пікселів у ширину і 32 у висоту. Це стандарт для SSD1306, і ми точно знаємо, що наш дисплей буде працювати в цих межах.

У функції setup() ми ініціалізуємо дисплей методом begin(). Якщо все гаразд — дисплей готовий до роботи. Ми очищаємо його, встановлюємо розмір тексту та задаємо білий колір для виводу. Тепер екран чекає на наші дані.

У loop() ми постійно зчитуємо аналогове значення з піну A0. Це значення — просто число від 0 до 1023, яке ми перетворюємо на напругу за формулою:  
voltage = sensorValue \* (5.0 / 1023.0)  
Таким чином, ми отримуємо реальну напругу, яка подається на вхід:

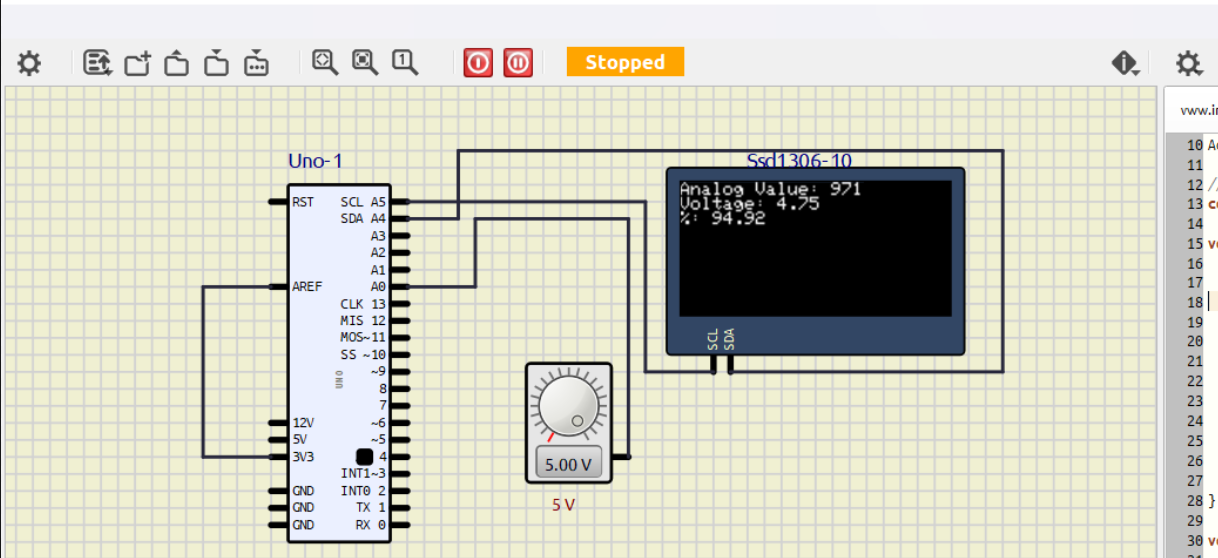


Рис 4.5. Приклад роботи схеми

**Завдання Друге**

Для другого завдання, робимо схему зчитування даних з датчика та їхнього аналізу. На схемі розміщаємо мікроконтролер, екран 99 (ssd1306), модуль годинника (ds1307) і відповідний до варіанта сенсор. Переконайтеся, що всі елементи з’єднані правильно.

1. Підключаємо датчик DHT до мікроконтролера Arduino, визначивши відповідні порти для передачі даних.

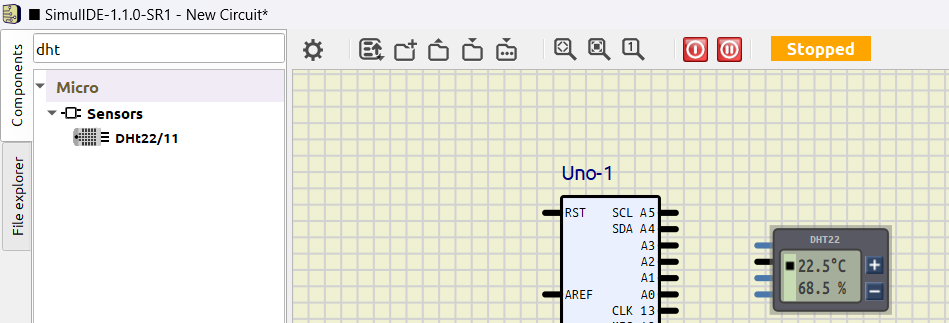


Рис. 4.6. Підключений датчик DHT до мікроконтролера Arduino

2. Встановимо необхідні бібліотеки в інтегроване середовище розробки (IDE), такі як DHT.h для датчика температури та Adafruit\_SSD1306 для роботи з OLED-дисплеєм.

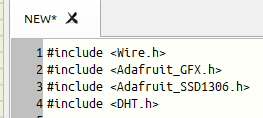


Рис. 4.7. Підключення необхідних бібліотек

3. Ініціалізуйте датчик DHT за допомогою класу DHT, вказавши тип датчика (наприклад, DHT22) і пін, до якого він підключений. 4. Запустіть серію команд для зчитування даних з датчика, використовуючи методи бібліотеки DHT, такі як readTemperature() для отримання температури. 5. Ініціалізуйте дисплей SSD1306 і відображайте зібрані дані на екрані, використовуючи методи бібліотеки Adafruit SSD1306, такі як display() для показу даних. Збираємо наступну схему:

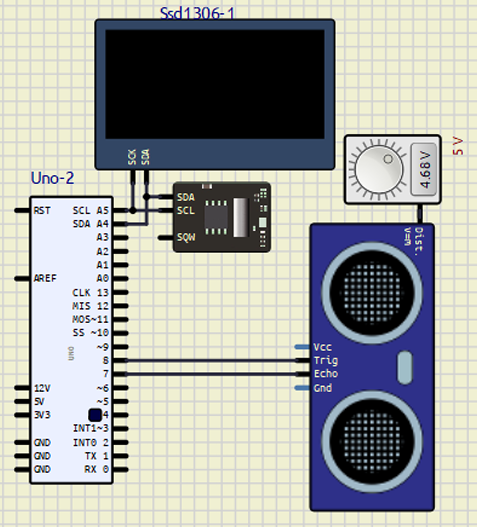


Рис. 4.8. Схема на основі ардуіно для Парного (16) варіанту

**Для парних варіантів (Варіант 16)**

Датчик відстані отримує та ретранслює отриманий сигнал для отримання часу відкліку, який використовується для обчислення відстані. Для отримання значення Часу відгуку необхідно використати функцію pulseIn(echoPin,HIGH), яка вираховує час активного імпульсу в мілісекундах. Підставивши дані в формулу, знаходимо результат: (час імпульсу/29)/2 в сантиметрах. Для конвертування значення до метра можемо скористатись відношенням: 1 метр = 100 сантиметрів.

Ми визначаємо:

- Розміри дисплея

- Пін для DHT22

- Тип датчика (DHT22, можна змінити на DHT11)

- Піни для ультразвукового датчика HC-SR04

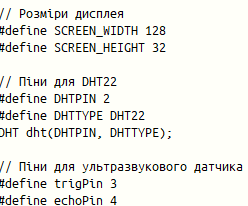


Рис 4.9. Визначення основних компонентів датчиків на схемі

Далі створюємо два об'єкти: dtf для зчитування температури та вологості, display — для керування OLED-дисплеєм. Визначаємо ф-цію setup(), де запускаємо серійний монітор для дебагу, ініціалізуємо датчик DHT, та дисплей і перевіряємо його доступність. Далі налаштовуємо пін trigPin як вихід, а echoPin як вхід — для роботи HC-SR04.

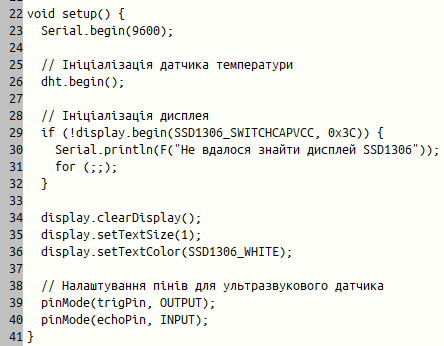


Рис 4.10. Налаштування ф-ції setup

Далі потрібно прописати основну логіку програми в loop. Ми зчитуємо температуру та вологість. Якщо значення NaN, тобто нечислове — це означає, що датчик не відповідає, і ми пропускаємо цикл. Надсилаємо короткий імпульс на trigPin, потім отримаємо, скільки часу сигнал повертається на echoPin, а pulseIn() повертає тривалість імпульсу в мікросекундах. І також - використовуємо фізичну формулу для обчислення відстані в сантиметрах, ділимо на 100, щоб отримати результат в метрах. В кінці прописуємо виводи на дисплей.

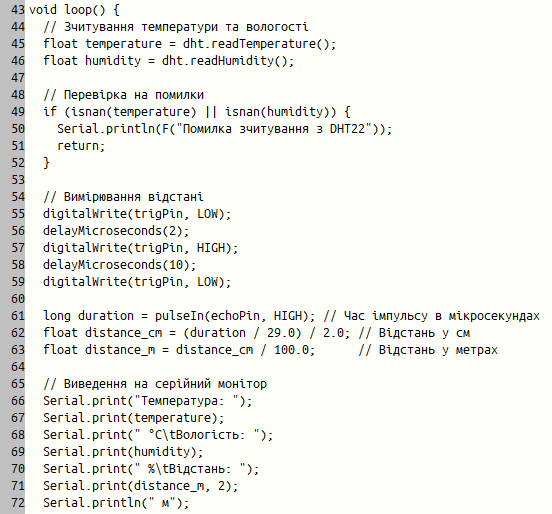


Рис 4.11. Програмування основної логіки програми

Після чого компілюємо наш код та отримаємо наступну інформацію на нашому дисплеї:

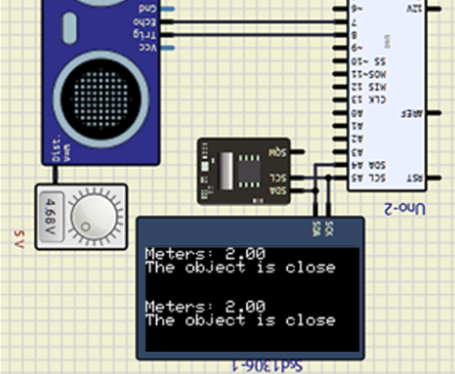


Рис.4.12 Успішна компіляція коду

**Висновок:**

У результаті виконання лабораторної роботи було реалізовано мікросистему збору та виводу даних на базі Arduino. Було успішно підключено датчик температури та вологості DHT22, ультразвуковий датчик HC-SR04 для вимірювання відстані, а також OLED-дисплей SSD1306 для візуалізації результатів. Ми навчилися зчитувати аналогові та цифрові сигнали, обробляти їх у коді, та виводити значення температури, вологості й відстані у зручному форматі.