Міністерство освіти і науки України

**Прикарпатський національний університет**

**імені В.Стефаника**

*Факультет математики та інформатики*

*Кафедра інформаційних технологій*

*Програмування вбудованих систем*

Лабораторна робота № 5

Тема: «Створення погодного логгера»

Варіант : **21**

Виконав: **Рижкін О. К.**

Група ІПЗ-23

Дата: 12 листопада 2023р.

Викладач: Лазарович І.М.

Івано-Франківськ – 2023

**Мета:** навчитись працювати з датчиками для вимірювання параметрів оточуючого середовища та енергонезалежною пам’яттю Arduino.

**Завдання на виконання**

Використовуючи теоретичний матеріал лекції 6 спроектувати схему вимірювання рівня освітленості, температури і вологості повітря на основі Arduino UNO та з передаванням даних в послідовний порт після натиснення кнопки. Мікроконтролер повинен опитувати датчики з періодом, що задається в програмі константою (10сек).   
  
 Дані повинні накопичуватись в EEPROM, після її заповнення дані повинні перезаписуватись в початок діапазону EEPROM. При натисненні на кнопку всі накопиченні дані повинні передаватись в порт в наступному форматі: T=xx.x гр.С, H=xx%, L=xxx @ Година:Хвилина:Секунда … де xx.x гр.С – значення температури з 1 знаком після коми, H=xx% - значення вологості (ціле), L=xxx – рівень освітленості (в умовних одиницях чи відсотках)   
Година:Хвилина:Секунда – час від моменту першого включення Arduino У випадку вимкнення живлення і повторного ввімкнення дані попередні дані повинні бути збережені, а запис повинен відбуватись в наступні вільні комірки EEPROM. Після передавання даних в порт, EEPROM повинна очиститись, а час повинен обнулитися.   
  
 В звіті навести тему, мету, сформульоване завдання, програмний код, схему з’єднань (можна фото), скріншот результатів виведення з послідовного порта (не менше 15-20 рядків), висновки.

**Code**

#include "DHT.h"

#include <EEPROM.h>

#define DHT\_PIN 2

#define DHT\_TYPE DHT22

DHT dht(DHT\_PIN, DHT\_TYPE);

const float GAMMA\_VALUE = 0.7;

const float RL\_VALUE = 50;

unsigned long current\_millis = 0;

const long update\_interval = 3000; // для швидших замірів

int eeprom\_address;

unsigned long start\_time;

unsigned long reset\_time = 0;

unsigned long previousMillis = 0;  // Added declaration

void configurePins() {

  pinMode(2, INPUT);

  pinMode(4, INPUT\_PULLUP);

}

void initializeSerial() {

**Serial**.begin(115200);

}

void initializeEEPROM() {

**EEPROM**.get(0, eeprom\_address);

**EEPROM**.get(sizeof(int), start\_time);

  if (eeprom\_address < 0) {

    eeprom\_address = sizeof(int) + sizeof(unsigned long);

    start\_time = 0;

  }

}

void setup() {

  configurePins();

  initializeSerial();

  initializeEEPROM();

  dht.begin();

}

void readFromEEPROM() {

  int stored\_temperature;

  int stored\_humidity;

  long stored\_light\_level;

  unsigned long stored\_time;

**EEPROM**.get(eeprom\_address, stored\_temperature);

  eeprom\_address += sizeof(int);

  stored\_humidity = **EEPROM**.read(eeprom\_address);

  eeprom\_address += 1;

**EEPROM**.get(eeprom\_address, stored\_light\_level);

  eeprom\_address += sizeof(long);

**EEPROM**.get(eeprom\_address, stored\_time);

  eeprom\_address += sizeof(unsigned long);

**Serial**.print("Temperature=");

  if (stored\_temperature >= -100 && stored\_temperature <= 100) {

**Serial**.print(stored\_temperature / 10.0, 1);

  } else {

**Serial**.print("Error");

  }

**Serial**.print("C, Humidity=");

**Serial**.print(stored\_humidity);

**Serial**.print("%, Light\_Level=");

**Serial**.print(stored\_light\_level);

**Serial**.print("LUX, ");

**Serial**.print((stored\_time) / 3600000);

**Serial**.print(":");

**Serial**.print(((stored\_time) % 3600000) / 60000);

**Serial**.print(":");

**Serial**.println(((stored\_time) % 60000) / 1000);

}

void resetEEPROM() {

  eeprom\_address = sizeof(int) + sizeof(unsigned long);

  for (int i = 0; i < **EEPROM**.length(); i++) {

**EEPROM**.write(i, 0);

  }

}

void loop() {

  current\_millis = millis() + start\_time - reset\_time;

  int current\_temperature = dht.readTemperature() \* 10;

  int current\_humidity = dht.readHumidity();

  int analog\_value = analogRead(A0);

  float current\_voltage = analog\_value / 1024.0 \* 5;

  float current\_resistance = 2000 \* current\_voltage / (1 - current\_voltage / 5);

  long current\_light\_level = pow(RL\_VALUE \* 1e3 \* pow(10, GAMMA\_VALUE) / current\_resistance, (1 / GAMMA\_VALUE));

  if (digitalRead(4) == LOW && (current\_millis - reset\_time > 200 || !reset\_time)) {

**Serial**.println("Кнопку успішно натиснуто!");

    int eeprom\_addr = sizeof(int) + sizeof(unsigned long);

    unsigned long stored\_time;

    reset\_time = millis();

    current\_millis = 0;

    while (eeprom\_addr < eeprom\_address) {

**EEPROM**.get(eeprom\_addr, current\_temperature);

      eeprom\_addr += sizeof(int);

      current\_humidity = **EEPROM**.read(eeprom\_addr);

      eeprom\_addr += 1;

**EEPROM**.get(eeprom\_addr, current\_light\_level);

      eeprom\_addr += sizeof(long);

**EEPROM**.get(eeprom\_addr, stored\_time);

      eeprom\_addr += sizeof(unsigned long);

**Serial**.print("T=");

**Serial**.print(current\_temperature / 10.0, 1);

**Serial**.print("C, H=");

**Serial**.print(current\_humidity);

**Serial**.print("%, L=");

**Serial**.print(current\_light\_level);

**Serial**.print("LUX, ");

**Serial**.print((stored\_time) / 3600000);

**Serial**.print(":");

**Serial**.print(((stored\_time) % 3600000) / 60000);

**Serial**.print(":");

**Serial**.println(((stored\_time) % 60000) / 1000);

    }

    start\_time = 0;

    eeprom\_address = sizeof(int) + sizeof(unsigned long);

    resetEEPROM();

  } else if (current\_millis - previousMillis >= update\_interval) {

    previousMillis = current\_millis;

**EEPROM**.put(eeprom\_address, current\_temperature);

    eeprom\_address += sizeof(int);

**EEPROM**.put(eeprom\_address, current\_humidity);

    eeprom\_address += 1;

**EEPROM**.put(eeprom\_address, current\_light\_level);

    eeprom\_address += sizeof(long);

**EEPROM**.put(eeprom\_address, current\_millis + start\_time);

    eeprom\_address += sizeof(unsigned long);

**EEPROM**.put(0, eeprom\_address);

**EEPROM**.put(sizeof(int), current\_millis);

    if (eeprom\_address + sizeof(int) + 1 + sizeof(long) + sizeof(unsigned long) > **EEPROM**.length()) {

**Serial**.println("Overflow");

      eeprom\_address = sizeof(int) + sizeof(unsigned long);

**EEPROM**.put(0, eeprom\_address);

    }

  }

}

**Screenshot of the scheme**



