Звіт з курсового проекту з дисципліни «Мікроконтролери»

Тема: Комп’ютеризована система керування кліматом з використанням давача DHT22

Схема №1

Завдання:

****

Порядок виконання роботи:

1. Вибір контролера – оскільки тип мікроконтролера не вказано, то необхідно його вибрати. Критеріями вибору мікроконтролера є:
   1. Наявність UART для забезпечення зв’язку по інтерфейсу RS-485
   2. Інтефейс 1-Wire для підключення сенсора DHT22
   3. Достатня кількість цифрових ліній портів ввод/виводу інформації для підключення: механічний енкодер – 4 шт., рідкокристалічний індикатор – 7 шт. (або 11 при 8-бітній схемі підключення), реле включення виконавчих пристроїв – 3 шт.

Можливим варіантом (а для реалізації прототипу пристрою - найдоцільнішим) є вибір плати сімейства Arduino. Це, крім модульного принципу побудови, забезпечує можливість використання безкоштовних засобів розробки програмного забезпечення та бібліотек периферійних пристроїв Arduino IDE. Вказаним критеріям відповідає плата Arduino Mega2560.

1. В середовищі Proteus 8.13 SP0 інсталюємо бібліотеки компонентів системи:
   1. Плати Arduino - <https://musbench.com/all/downloads/>;
2. В середовищі Proteus складаємо схему апаратного забезпечення системи в якій
   1. інтерфейс RS-485 емулюється віртуальним терміналом;
   2. оскільки при симуляції сенсора типу DHT 22 він не відповідав на запити від плати Arduinо його було замінено на сенсор типу DHT 11;
   3. Механічний енкодер емулюємо схемою на базі елементу MOTOR-ENCODER представленою на Рис.1, де: ключ SW1 типу SW-DPDT-MOM забезпечує реверс енкодера, а кнопка з фіксацією типу BUTTON забезпечує його зупинку (чомусь після першого включення цієї кнопки виключити мотор-енкодер неможливо, тому на схемі присутній додатковий ключ SW2, що відключає вихід енкодера від входів мікроконтролера), інша кнопка SW\_MODE емулює вимикач енкодера;



Рис. 1. Схема емуляції енкодера

* 1. Спрацювання реле детектується світлодіодом в колі бази транзистора керування реле



Рис. 2. Схема детектора спрацювання реле

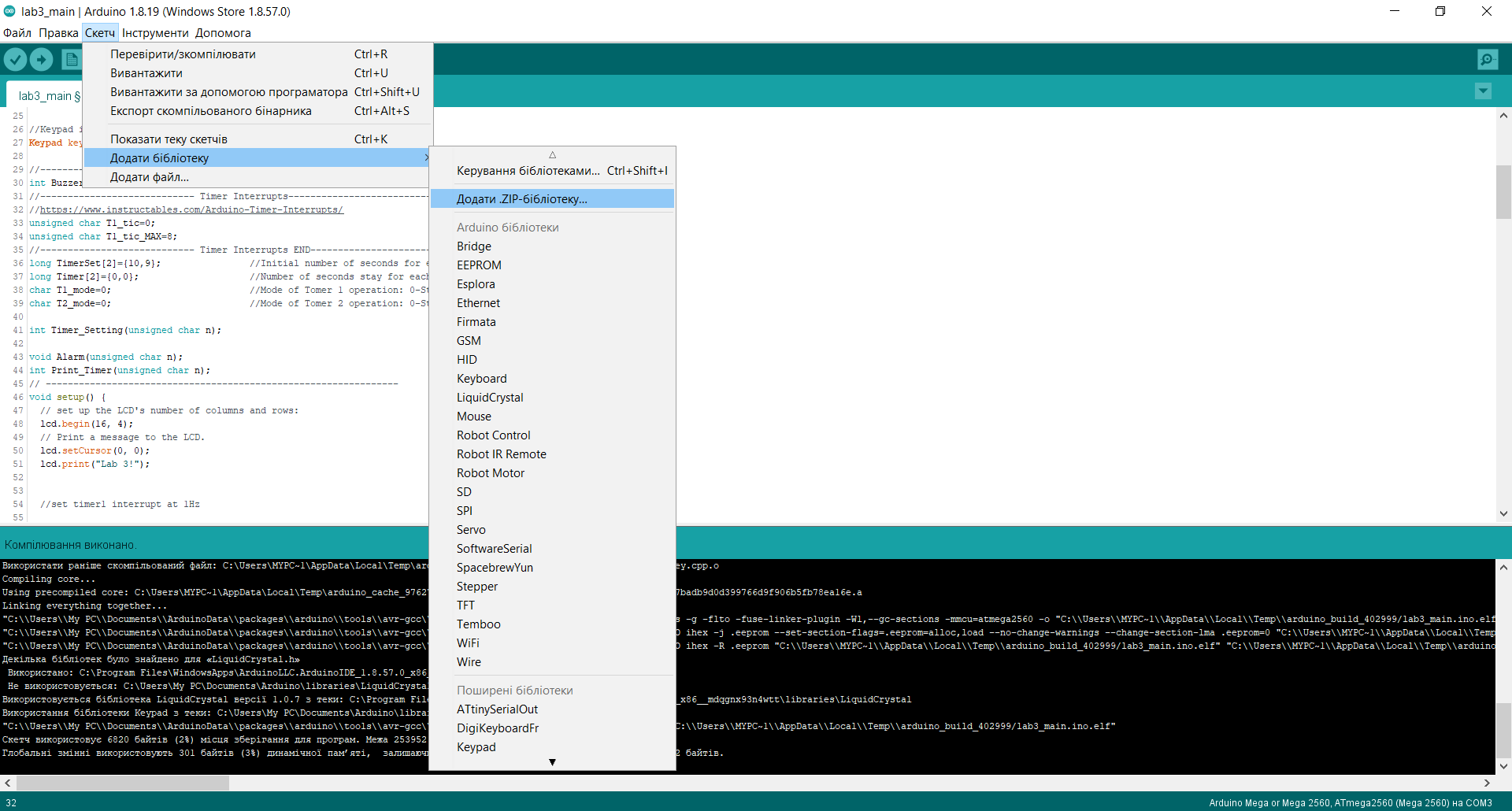


Рис. 3. Схема емуляції в програмному пакеті Proteus апаратного   
забезпечення проектованої системи

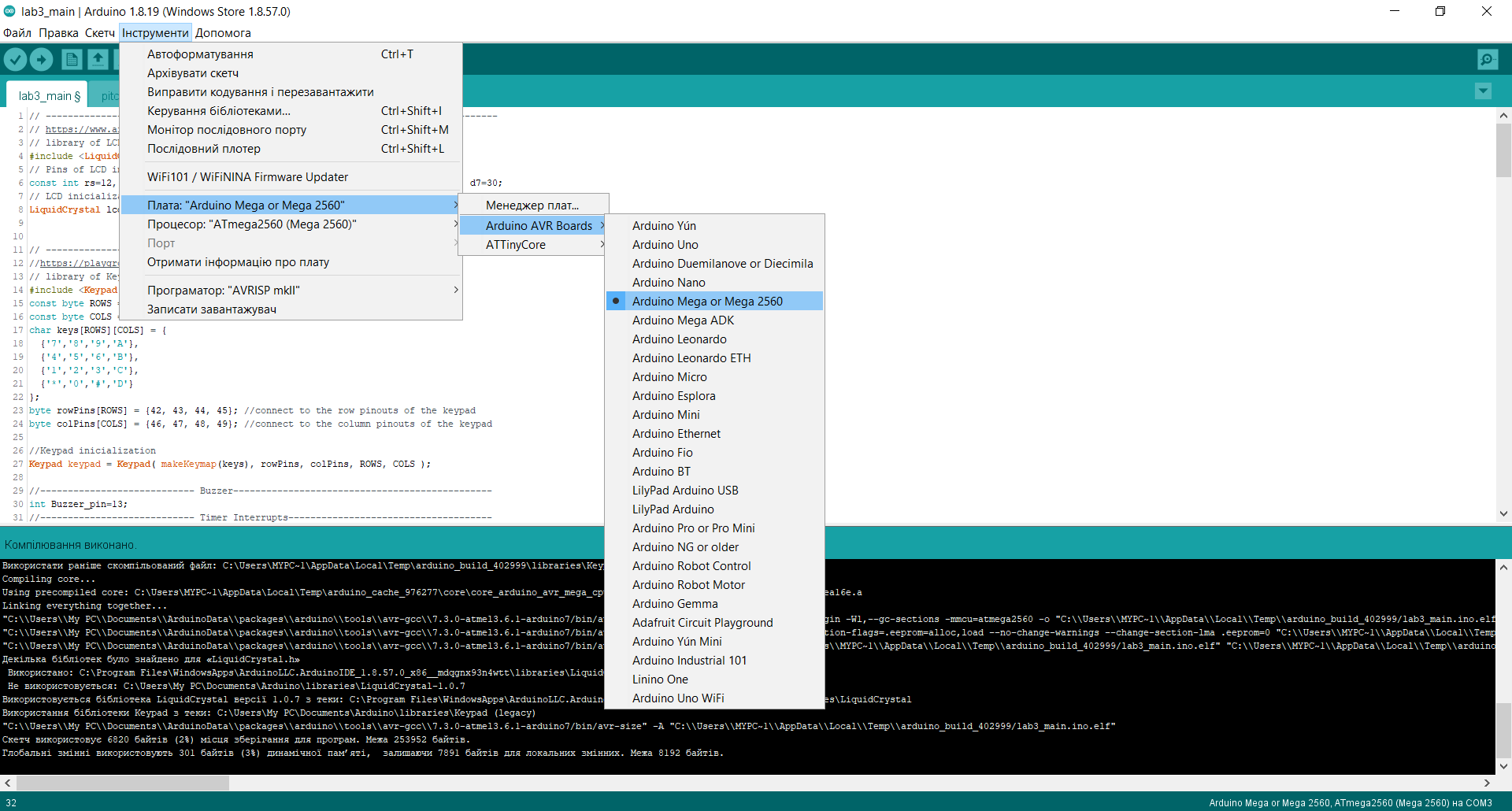
1. Скачуємо набір бібліотек представлених в таблиці

|  |  |  |  |
| --- | --- | --- | --- |
| № | Периферійний пристрій | Інтернет адреса | Файл |
| 1. | Рідкокристалічний індикатор LCD | <https://www.arduino.cc/reference/en/libraries/liquidcrystal/> | LiquidCrystal-1.0.7.zip |
| 2. | Механічний енкодер | <https://www.arduino.cc/reference/en/libraries/rotaryencoder/> | RotaryEncoder-1.5.2.zip |
| 3. | Сенсор DHT22 | <https://www.arduino.cc/reference/en/libraries/dht-sensor-library/> | DHT\_sensor\_library-1.4.3.zip |
| 4. | Adafruit Unified Sensor | <https://www.arduino.cc/reference/en/libraries/adafruit-unified-sensor/> | Adafruit\_Unified\_Sensor-1.1.5.zip |

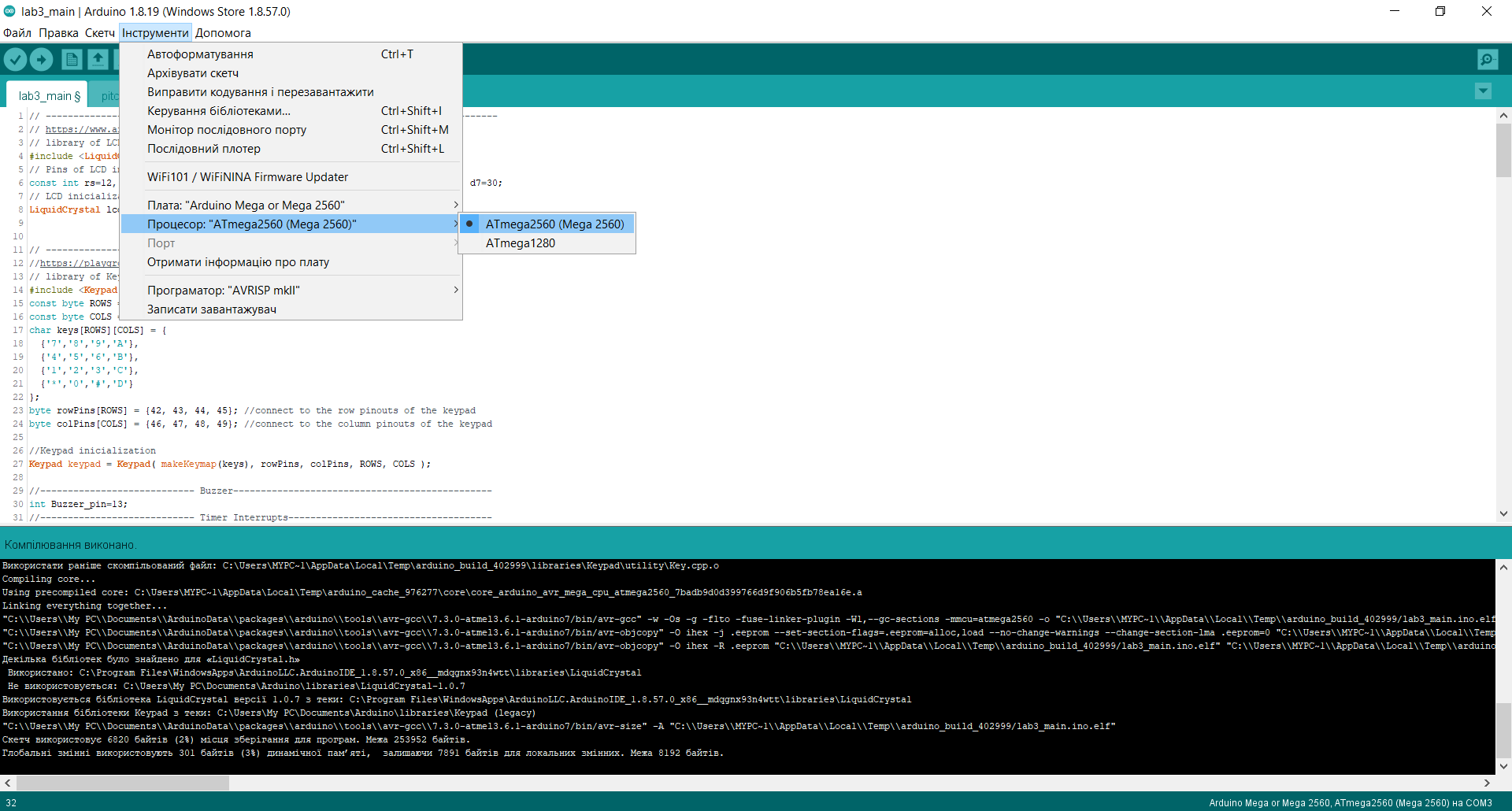
1. В середовищі програмування Arduino IDE інсталюємо скачані в п.4 бібліотеки (вибираємо шлях до скачаних архівів)



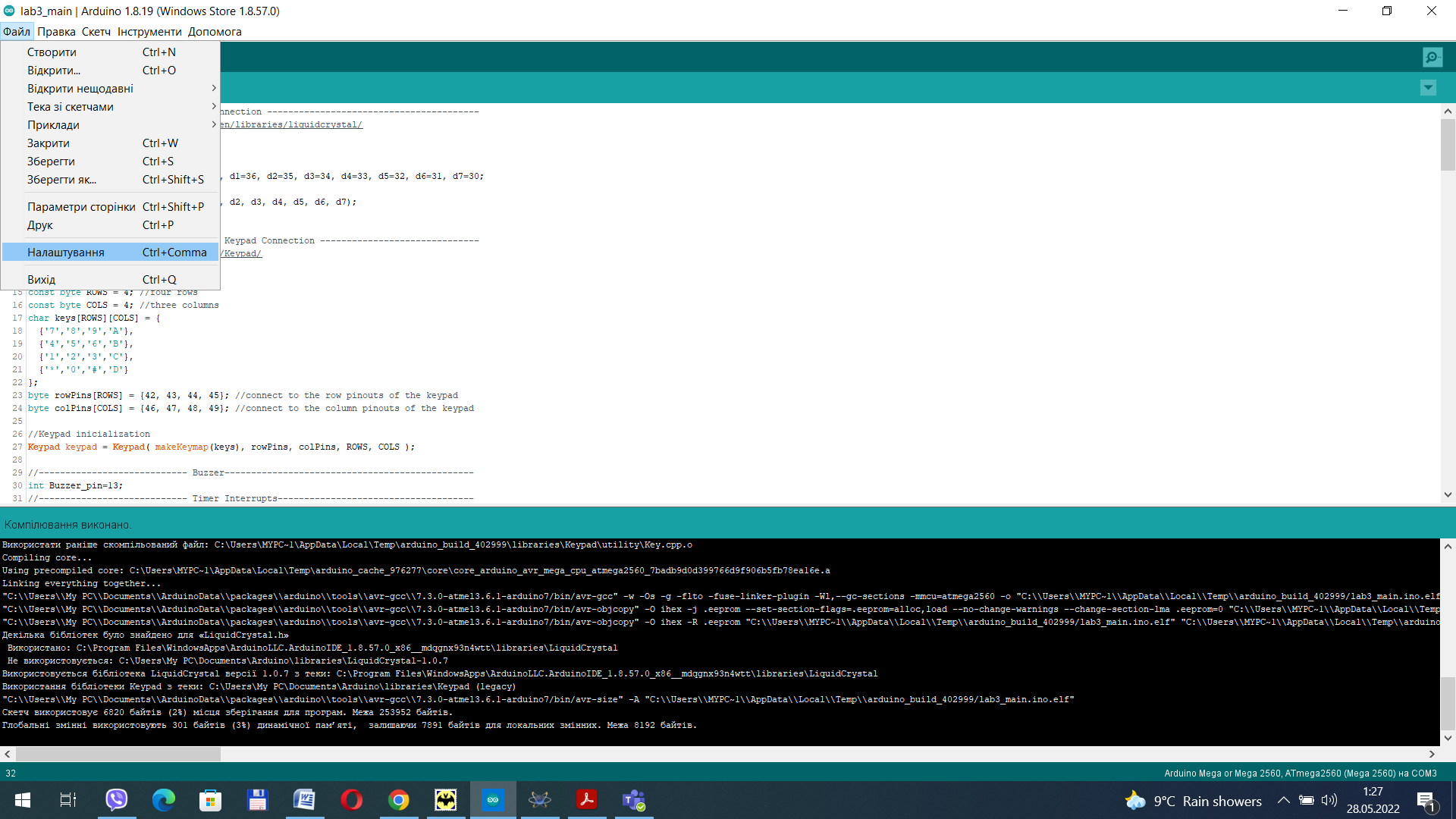
1. В середовищі програмування Arduino IDE задаємо обрану плату типу Arduino Mega2560

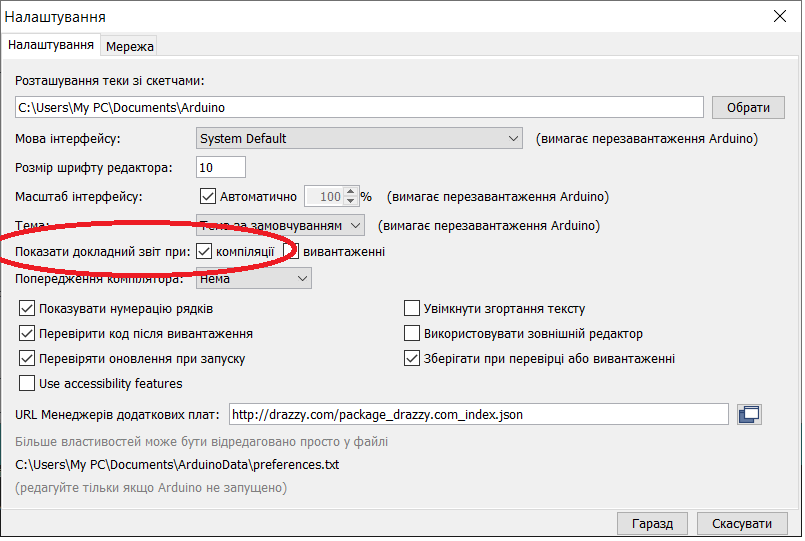


1. В середовищі програмування Arduino IDE задаємо тип мікроконтролера типу Mega2560



1. Викликаємо вікно налаштувань і задаємо режим «Показати докладний звіт при компіляції»





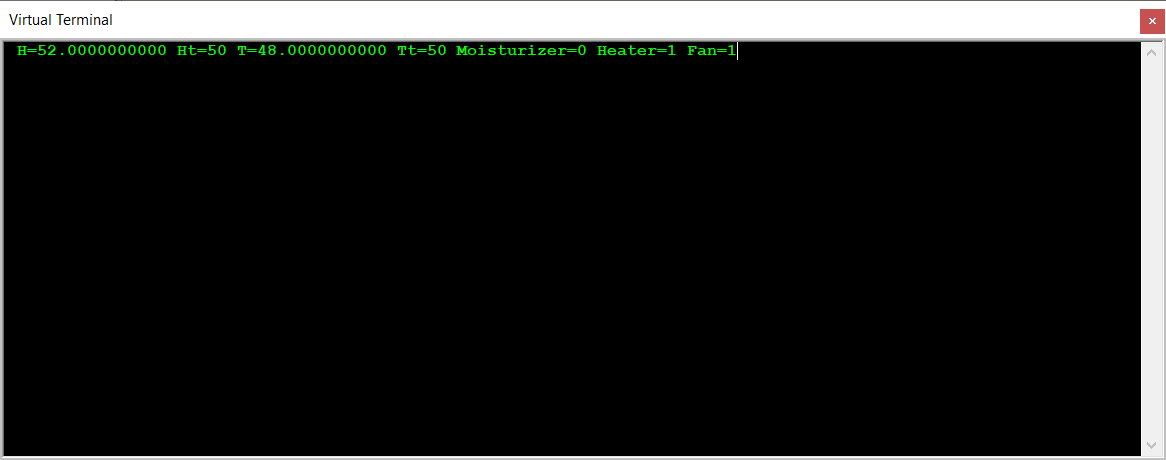
1. Задаємо назву проекту та набираємо текст програми роботи мікроконтролера

|  |
| --- |
| // -------------------------- LCD Connection ----------------------------------------  // https://www.arduino.cc/reference/en/libraries/liquidcrystal/  // library of LCD  #include <LiquidCrystal.h>  // Pins of LCD interface  const int rs=30, rw=31, en=32, d0=22, d1=23, d2=24, d3=25, d4=26, d5=27, d6=28, d7=29;  // LCD inicialization  LiquidCrystal lcd(rs, rw, en, d0, d1, d2, d3, d4, d5, d6, d7);  // --------------------------- Relays Connection -------------------------------------  #define RelayT 51 //Relay Temp  #define RelayF 52 //Relay Fan  #define RelayH 53 //Relay Hum  char HumSwOn;  char TempSwOn;  char FanSwOn;  // --------------------------- Encoder Connection ------------------------------------  //#include <Arduino.h>  #include <RotaryEncoder.h>  #define TempSetStep 1  #define TempSetMin 0  #define TempSetMax 100  int TempSetVal=50; //Theshold value of Temp  float TempCur; //Current value of Temp  #define HumSetStep 1  #define HumSetMin 0  #define HumSetMax 100  int HumSetVal=50; //Theshold value Humidity  float HumCur; //Current value of Humidity  #define ROTARYSTEPS 1  #define ROTARYMIN 0  #define ROTARYMAX 100  int EncoderStart=50;  int EncoderPos;  #define REnc\_PIN\_A 36  #define REnc\_PIN\_B 37  #define REnc\_SW\_PIN 38 //Switch Temp\_setting/Humidity\_setting  unsigned char REnc\_Temp\_Set\_Mode=1; //1-Temp\_setting, 0-Humidity\_setting  RotaryEncoder encoder(REnc\_PIN\_A, REnc\_PIN\_B, RotaryEncoder::LatchMode::TWO03);    // --------------------------- DHT22 Connection ------------------------------------  #include "DHT.h"  #define DHTPIN 43 // Digital pin connected to the DHT sensor  #define DHTTYPE DHT11 // DHT 11  DHT dht(DHTPIN, DHTTYPE);  // --------------------------- Serial Port ------------------------------------  // https://www.arduino.cc/reference/en/language/functions/communication/serial/available/  int incomingByte=0;  //-------------------------------------------------------------------------  int EncoderReading(void){  int Delta = 0;  encoder.tick();  int EncoderNewPos = encoder.getPosition() \* ROTARYSTEPS;  if (EncoderNewPos < ROTARYMIN) {  encoder.setPosition(ROTARYMIN / ROTARYSTEPS);  EncoderNewPos = ROTARYMIN;  } else if (EncoderNewPos > ROTARYMAX) {  encoder.setPosition(ROTARYMAX / ROTARYSTEPS);  EncoderNewPos = ROTARYMAX;  }  if (EncoderPos != EncoderNewPos) { //Encoder was changed  Delta=(EncoderNewPos-EncoderPos)/ROTARYSTEPS;  EncoderPos = EncoderNewPos;    /\* Serial1.print("Delta:");  Serial1.print(Delta);  Serial1.print("pos:");  Serial1.print(EncoderNewPos);  Serial1.print(" dir:");  Serial1.println((int)(encoder.getDirection()));\*/  }  return Delta;  }  //-------------------------------------------------------------------------  void SetTempHum(int Delta)  {  REnc\_Temp\_Set\_Mode=digitalRead(REnc\_SW\_PIN);    if(REnc\_Temp\_Set\_Mode==1) TempSetVal+=Delta \* TempSetStep;  else HumSetVal+=Delta \* HumSetStep;  if(TempSetVal>TempSetMax)TempSetVal=TempSetMax;  if(HumSetVal>HumSetMax)HumSetVal=HumSetMax;  if(TempSetVal<TempSetMin)TempSetVal=TempSetMin;  if(HumSetVal<HumSetMin)HumSetVal=HumSetMin;  /\* Serial1.print("\n\r TSet=");  Serial1.print(TempSetVal);  Serial1.print(" HumSet=");  Serial1.print(HumSetVal);\*/  }  //-------------------------------------------------------------------------  void ReatTempHum(void)  {  // --------------------- DHT sensor test ------------------------------  // Reading temperature or humidity takes about 250 milliseconds!  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)  float h = dht.readHumidity();  // Read temperature as Celsius (the default)  float t = dht.readTemperature();  // Check if any reads failed and exit early (to try again).  if (isnan(h) || isnan(t))  {  Serial1.println(F("Failed to read from DHT sensor!"));  return;  }  TempCur=t;  HumCur=h;  /\* Serial1.print(F("\n\rHumidity: "));  Serial1.print(HumCur);  Serial1.print(F("% Temperature: "));  Serial1.print(TempCur);  Serial1.print(F("°C "));\*/  }  //-------------------------------------------------------------------------  void printLCD(void)  {  int tt;  //----------- Print Hummidity on LCD ----------------  lcd.setCursor(0, 0);  lcd.print("H= . % %");  tt=HumCur;  lcd.setCursor(2, 0);  lcd.print(tt,DEC);  tt=(HumCur-tt)\*10;  lcd.setCursor(5, 0);  lcd.print(tt,DEC);  lcd.setCursor(9, 0);  lcd.print(">");  tt=HumSetVal;  lcd.print(tt,DEC);  //----------- Print Temp on LCD ----------------  lcd.setCursor(0, 1);  lcd.print("T= . oC oC");  tt=TempCur;  lcd.setCursor(2, 1);  lcd.print(tt,DEC);  tt=(TempCur-tt)\*10;  lcd.setCursor(5, 1);  lcd.print(tt,DEC);  lcd.setCursor(9, 1);  lcd.print(">");  tt=TempSetVal;  lcd.print(tt,DEC);  }  //-------------------------------------------------------------------------  void RelaySet(void)  {  if(TempCur<TempSetVal)TempSwOn=1;  else TempSwOn=0;  if(HumCur<HumSetVal)HumSwOn=1;  else HumSwOn=0;  FanSwOn=HumSwOn|TempSwOn;  if(TempSwOn==1)digitalWrite(RelayT, HIGH); //To Low Temp - Switch ON Heater  else digitalWrite(RelayT, LOW);  if(HumSwOn==1)digitalWrite(RelayH, HIGH); //To Low Humidity - Switch ON Moisturizer  else digitalWrite(RelayH, LOW);    if(FanSwOn==1)digitalWrite(RelayF, HIGH); //Switch ON Fan for smoothing Temperature & Hummidity  else digitalWrite(RelayF, LOW);  }  //-------------------------------------------------------------------------  void SerialTest(void)  {  if (Serial1.available() > 0)  {  // read the incoming byte:  incomingByte = Serial1.read();  if(incomingByte=='?') //Request for printing values '?'  {  Serial1.print(" H=");  Serial1.print(HumCur, DEC);  Serial1.print(" Ht=");  Serial1.print(HumSetVal);  Serial1.print(" T=");  Serial1.print(TempCur, DEC);  Serial1.print(" Tt=");  Serial1.print(TempSetVal,DEC);  Serial1.print(" Moisturizer=");  Serial1.print(HumSwOn,DEC);  Serial1.print(" Heater=");  Serial1.print(TempSwOn,DEC);  Serial1.print(" Fan=");  Serial1.print(FanSwOn,DEC);  }  if(incomingByte=='!') //Request for Setting Thresholds "!HHTT": HH-HummSetVal, TT-TempSetVal (if H or T is not digit it is restored 0 - !A45K -> HumSetVal=04 & TempSetVal=50)  {  // Serial1.print("\n\rEnter Hummidity threshold:");  while(Serial1.available() == 0); //HumSetVal reading  incomingByte = Serial1.read();  if (!isDigit(incomingByte))incomingByte='0';  HumSetVal=10\*(incomingByte-'0');  while(Serial1.available() == 0);  incomingByte = Serial1.read();  if (!isDigit(incomingByte))incomingByte='0';  HumSetVal+=(incomingByte-'0');  while(Serial1.available() == 0); //TempSetVal reading  incomingByte = Serial1.read();  if (!isDigit(incomingByte))incomingByte='0';  TempSetVal=10\*(incomingByte-'0');  while(Serial1.available() == 0);  incomingByte = Serial1.read();  if (!isDigit(incomingByte))incomingByte='0';  TempSetVal+=(incomingByte-'0');    Serial1.print("\n\r Ht=");  Serial1.print(HumSetVal);  Serial1.print(" Tt=");  Serial1.print(TempSetVal);  }  }  }  //-------------------------------------------------------------------------  void setup() {  // ------------- LCD Init ----------------------------------  // set up the LCD's number of columns and rows:  lcd.begin(16, 4);  // Print a message to the LCD.  /\* lcd.setCursor(0, 0);  lcd.print("Circuit #1 Project");  delay(100);  \*/  // ------------- Encoder Init ----------------------------------  pinMode(REnc\_SW\_PIN, INPUT\_PULLUP);  encoder.setPosition(EncoderStart / ROTARYSTEPS);  encoder.tick();  EncoderPos = encoder.getPosition() \* ROTARYSTEPS;  // ------------- Serial Communication Init ---------------------  Serial1.begin(9600);  // Serial1.println(F("Test!"));  // ------------- DHT11 Sensor Init ------------------------------  dht.begin();  // ------------- Relays Init ------------------------------  pinMode(RelayT, OUTPUT);  digitalWrite(RelayT, LOW);  pinMode(RelayF, OUTPUT);  digitalWrite(RelayF, LOW);  pinMode(RelayH, OUTPUT);  digitalWrite(RelayH, LOW);  }  //-------------------------------------------------------------------------  void loop() {  // delay(200);  int n=EncoderReading(); //Read Encoder's increment  if(n!=0) SetTempHum(n); //Set new Thresholds  ReatTempHum(); //Read sensor  printLCD(); //Renew LCD  RelaySet(); //Send control signals to Relays  SerialTest(); //Test serial port status & parse if is not empty  }  //-------------------------------------------------------------------------  //------------------------------------------------------------------------- |

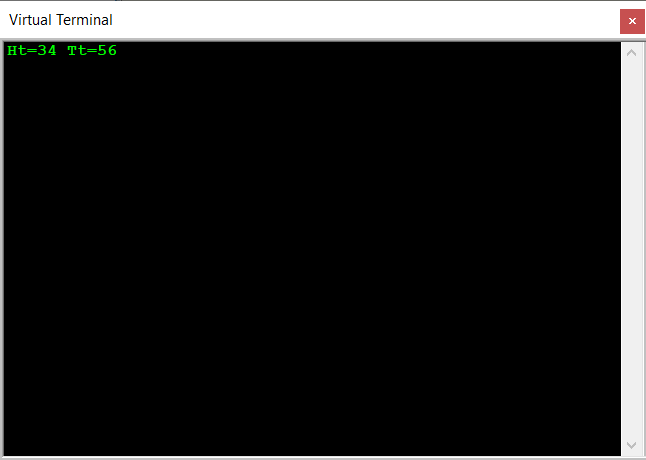
**\* протокол звязку між мікроконтролером та програмним забезпеченням сервера описано при описі функції void SerialTest(void)**

Програма роботи мікроконтролера складається з наступних функцій:

* void setup() – забезпечує початкову ініціалізацію ресурсів мікроконтролера: рідкокристалічного індикатора, енкодера, послідовний інтерфейс (режим роботи 9600 8N1), сенсор DHT 11, входи керування реле, виключає всі реле.
* void loop() – виконується циклічно. Забезпечує:
  + Виклик функції зчитування енкодера - EncoderReading();
  + При ненульовому значенні приросту енкодера викликає функцію SetTempHum – встановлення порогових значень температури і вологості;
  + Виклик функції ReatTempHum() – зчитування показів сенсора;
  + Виклик функції printLCD() – вивід на рідкокристалічний індикатор;
  + Виклик функції RelaySet() – формування сигналів керування реле;
  + Виклик функції SerialTest() – зчитування даних з буферу послідовного інтерфейсу і опрацювання запиту
* void SerialTest(void) – функція забезпечує зчитування даних з буферу послідовного інтерфейсу і опрацювання запиту. Передбачається 2 можливі запити:
  + символ «?» - запит значень. В відповідь формується стрічка формату «H=52.0000 Ht=50 T=48.00000 Tt=50 Moisturizer=0 Heater=1 Fan=1», де значення після: «Н=» - поточне значення вологості, «Нt=» - порогове значення включення зволожувача, «Т=» - поточне значення температури, «Tt=» - порогове значення включення нагрівача, «Moisturizer=» - значення виводиться на вивід керування транзисторним каскадом включення реле зволожувача (1 – зволожувач включений, 0 – виключений), «Heater=» -/- нагрівача, «Fan» - -/- вентилятора.



* + послідовність з 5-ти символів, що починається символом «!» - «!HHTT» - команда встановлення порогових значень вологості та температури (НН – вологість, ТТ - температура). Символи Н та Т повинні бути цифровими ASCII-символами. Якщо якийсь з цих символів не є цифровим то замість нього записується символ «0». Відповідь на запит формується стрічка з пороговими значеннями. Наприклад відповідь на запит «!3456» є стрічка «Ht=34 Tt=56»



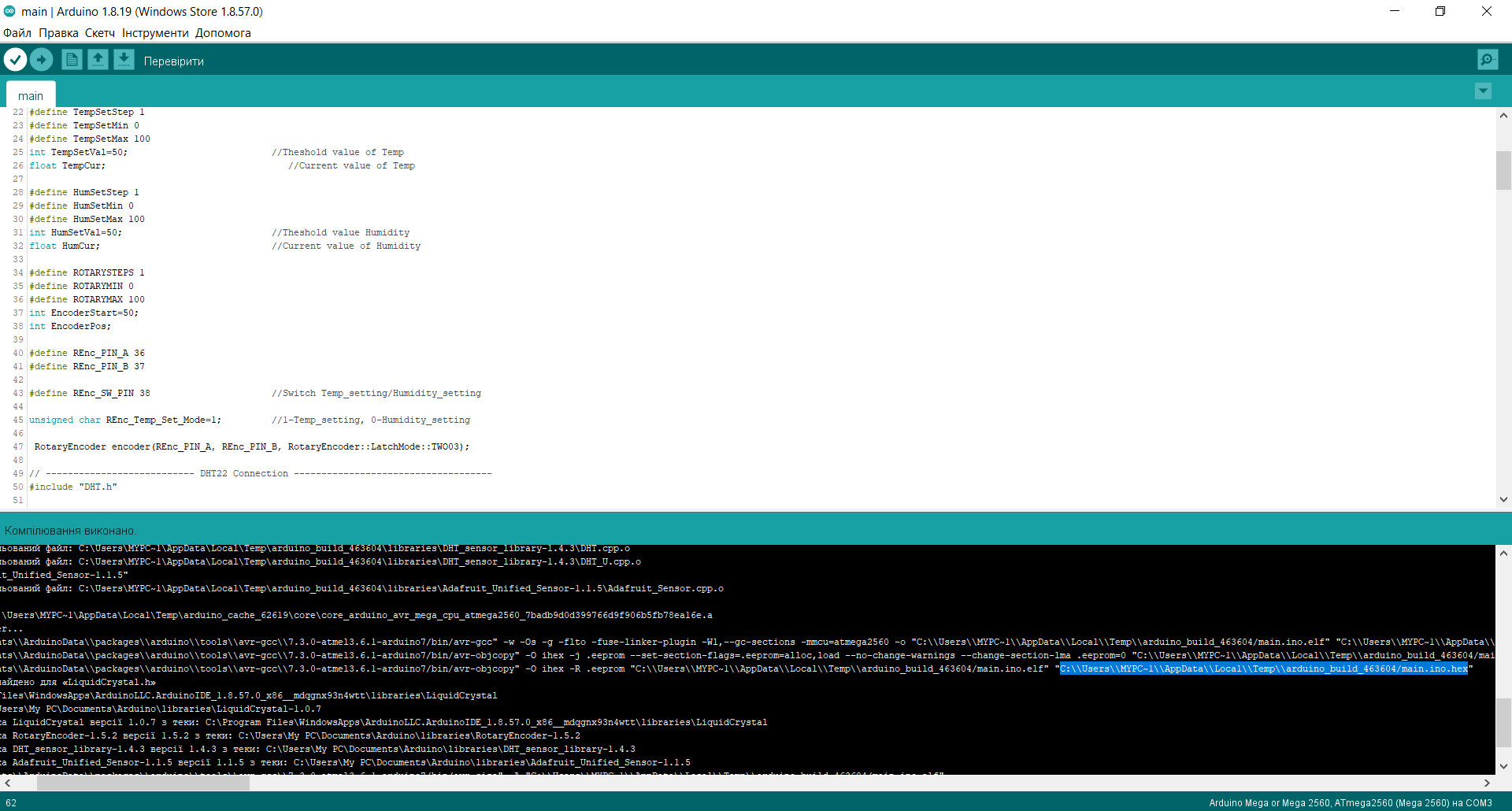
* void RelaySet(void) – функція формування сигналів керування реле. Якщо поточне значення температури перевищує порогове, то реле керування нагрівачем – включається, якщо поточне значення вологості перевищує порогове значення, то реле керування зволожувачем – включається. Якщо включене реле керування нагрівачем або зволожувачем, то включається вентилятор для вирівнювання температури та вологості в кімнаті.
* void printLCD(void) – функція виведення інформації на рідкокристалічний індикатор. В верхній стрічці індикатора виводяться поточне та порогове значення вологості, а в нижній – поточне та порогове значення температури.



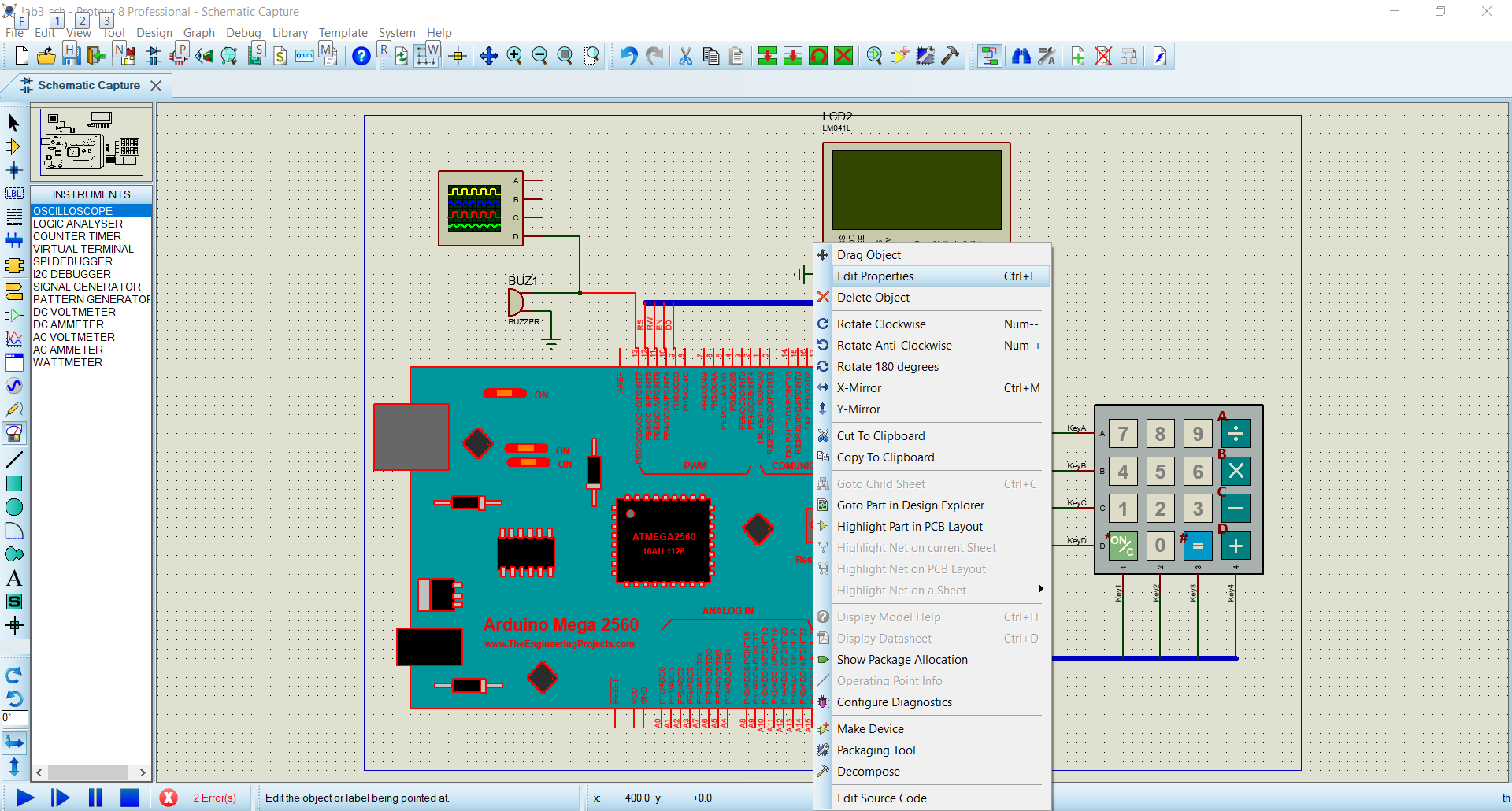
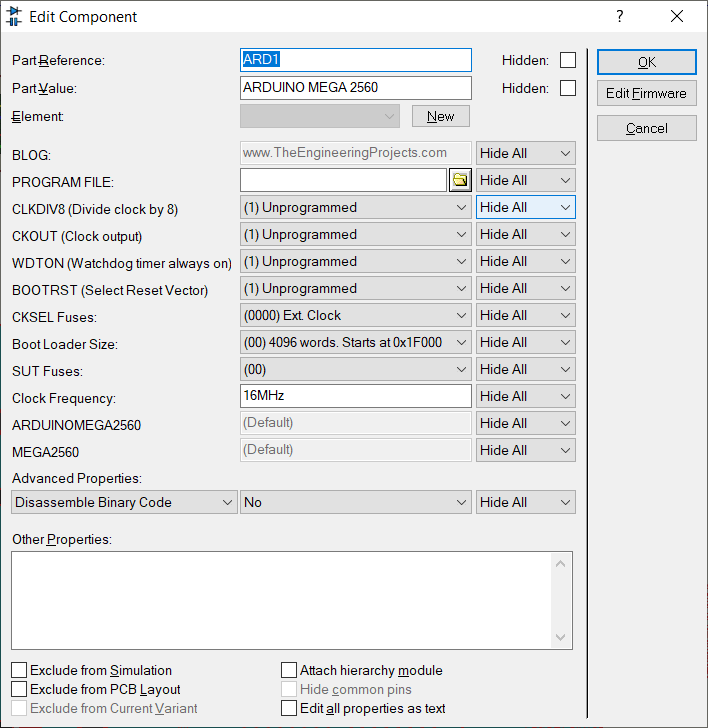
* void ReatTempHum(void) – функція зчитування даних сенсора DHT 11 (сенсор заданий в завданні – DHT 22 не запрацював в режимі емуляяції в пакеті Proteus);
* void SetTempHum(int Delta) – функція запису порогових значень вологості та температури, яка приріст коду енкодера записує в приріст порогового значення вологості якщо кнопка SW\_MODE підключена до виводу 38 плати ArduinoMega - натиснута, або в приріст порогового значення температури якщо ця кнопка не натиснута. Порогові значення мають обмеження на максимальне та мінімальне значення.
* int EncoderReading(void) – функція зчитування сигналів енкодера. Повертає значення приросту коду енкодера (якщо енкодер обертати в іншу сторону, то повертається від’ємне значення).

Для запуску симуляції роботи системи потрібно підключити виконавчий файл програми роботи мікроконтролера, що генерується Arduino IDE до МК в программі Proteus. Для цього потрібно:

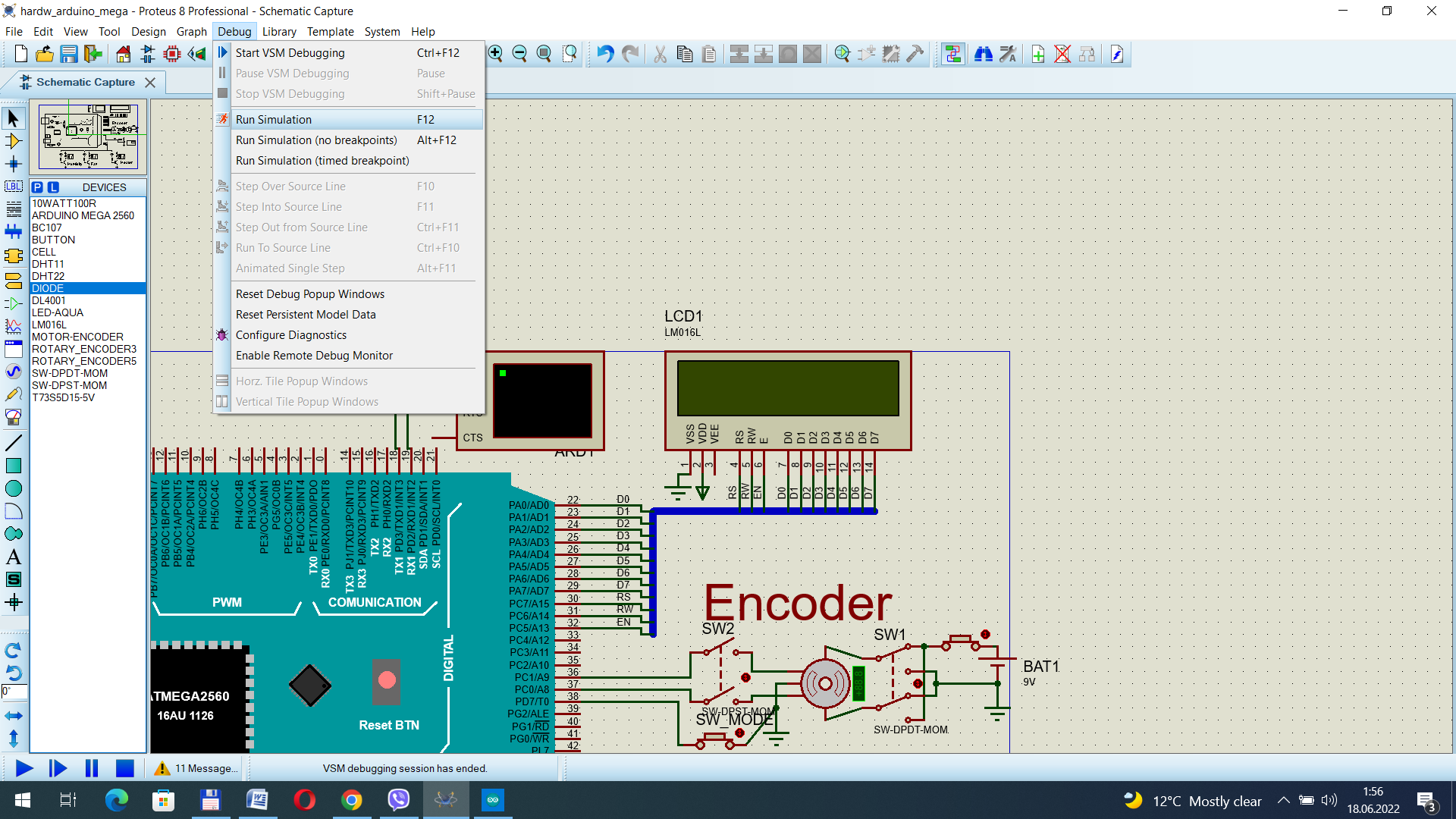
* в вікні статусу виділити шлях до виконавчого файлу (файл з розширенням НЕХ) і зкопіювати його в буфер обміну (після кожного запуску Arduino IDE цей шлях змінюється, тому це потрібно роити після кожного перезапуску Arduino IDE)



* В програмі Proteus викликати контекстне меню плати Arduino та вибрати пункт Edit Properties

* В полі «PROGRAM FILE» вставити інформацію з буферу обміну
* Вибрати пункт меню «Debug»-«Run Simulation»



При симуляції для зміни порогових значень з допомогою енкодера:

* для зменшення порогового значення вологості потрібно, щоби органи керування енкодером були в положенні



* для збільшення порогового значення вологості потрібно, щоби органи керування енкодером були в положенні



* для зменшення порогового значення температури потрібно, щоби органи керування енкодером були в положенні



* для збільшення порогового значення температури потрібно, щоби органи керування енкодером були в положенні



* для того, щоби порогові значення не змінювалися потрібно, щоби ключ SW2 був виключений



Висновки: під час виконання курсової роботи було виявлено наступні помилки в роботі програмного пекету Proteus 8.13 SP0:

* некорректна симуляція елементу MOTOR-ENCODER – обертається ротор при виключеному живленні;
* некоректна симуляція сенсора типу DHT 22 – відсутність відповіді на запит.