```
/*
  * This is the first firmware version for a home weather station.
  * This firmware was written in May-June 2022 as part of a course project.
  * It is planned to release upgraded firmware.
*/

#include <Wire.h>               // library for working with I2C modules
#include <LiquidCrystal.h>      // library for working with LCD display
#include <TroykaMeteoSensor.h>  // library for working with a weather sensor
#include <TroykaMQ.h>           // library for working with a carbon dioxide sensor

// pin detection for connecting the LCD display
constexpr uint8_t PIN_RS = 6;
constexpr uint8_t PIN_EN = 7;
constexpr uint8_t PIN_DB4 = 8;
constexpr uint8_t PIN_DB5 = 9;
constexpr uint8_t PIN_DB6 = 10;
constexpr uint8_t PIN_DB7 = 11;

// pins to which the carbon dioxide sensor is connected
#define PIN_MQ135 A0
#define PIN_MQ135_HEATER 5

LiquidCrystal lcd(PIN_RS, PIN_EN, PIN_DB4, PIN_DB5, PIN_DB6, PIN_DB7);  // assigning
a variable to an LCD display and assigning its pins
TroykaMeteoSensor meteoSensor;                                          // defining a
variable for a weather sensor
MQ135 mq135(PIN_MQ135);                                                 // sensor
variable MQ-135

// ---------------------------- Tuning function ----------------------------
void setup() {
  lcd.begin(16, 2);       // determining the size of the LCD display
  meteoSensor.begin();    // initialization of the weather sensor
  mq135.heaterPwrHigh();  // voltage supply to the carbon dioxide sensor heater
}

// -------------------------- Cyclic execution function -------------------------
void loop() {
  co2();        // Calling the carbon dioxide sensor function
  meteo();      // Calling the weather sensor function

  delay(3000);  // delay function, measurements take place every 3 seconds
}

// ----------------------- Function carbon dioxide sensor -----------------------
void co2(){
  /* if the sensor heating interval has passed
  and the calibration was not completed */
  if (!mq135.isCalibrated() && mq135.heatingCompleted()) {
    mq135.calibrate();    // calibrate the sensor in clean air
    lcd.setCursor(0, 1);  // we output the data to the second line of the LCD display
    /* if the resistance of the sensor in clean air is known
    you can specify it manually, for example 160
    mq135.calibrate(160);
    we display the sensor resistance in clean air (Ro) on the LCD display*/
    lcd.print("Ro = ");
    lcd.print(mq135.getRo());
  }

  /* if the sensor heating interval has passed
  and the calibration was done */
  if (mq135.isCalibrated() && mq135.heatingCompleted()) {
    lcd.setCursor(0, 1);  // I output the data to the second line of the LCD display
    // I output the values of carbon dioxide in ppm
    lcd.print("CO2: ");
    lcd.print(mq135.readCO2());
    lcd.print(" ppm");
    delay(100);
  }
}

// ---------------------------- Weather sensor function -------------------------
void meteo(){
  int stateSensor = meteoSensor.read();

  switch (stateSensor) {
    // I display humidity and temperature readings
    case SHT_OK:
      lcd.setCursor(0, 0);
      lcd.print(meteoSensor.getTemperatureC());
      lcd.print("C ");
      lcd.print(meteoSensor.getHumidity());
      lcd.print("%");
    break;

    // data error or sensor is not connected
    case SHT_ERROR_DATA:
      lcd.setCursor(0, 0);
      lcd.print("Data error");
      lcd.setCursor(0, 1);
      lcd.print("Or not connected");
    break;

    // checksum error
    case SHT_ERROR_CHECKSUM:
      Serial.println("Checksum error");
    break;
  }
}
```