Indoor weather station

Embedded system project

AiR – CPS 2022

by

Agnieszka Welian

Mikołaj Szopa

Bartosz Trynda

Github repository: https://github.com/tenkinogawa/Embedded-project

13-06-2022

Contents

1. Introduction

Smart ideas are nowadays more and more popular not only in major companies but also in private homes. One of the desired solution is a monitoring of the basic parameters as temperature, humidity and air quality.

The aim of the project is to create a weather station which will allow to monitor several parameters in the house. Sensors are placed in the especially created housing which allows particles in the air to freely flow in and out.

2. Hardware

2.1 General information

The whole project is based on the Arduino MKR 1010 WiFi board and Nextion NX3224k028_011 screen (with screen diagonal equal 2.8'). Measurements are gather each 5s. Thie choice of period was dictated by the constantly changing indoor environment and safety considerations. In case of particles in the air, every second can matter when it comes to poisoning.

2.2 Simplified diagram – system composer

To illustrate what kind of connections are present within the project, the system diagram was prepared. Diagram is simplified – contains only names of the sensors, used protocols and voltage levels.

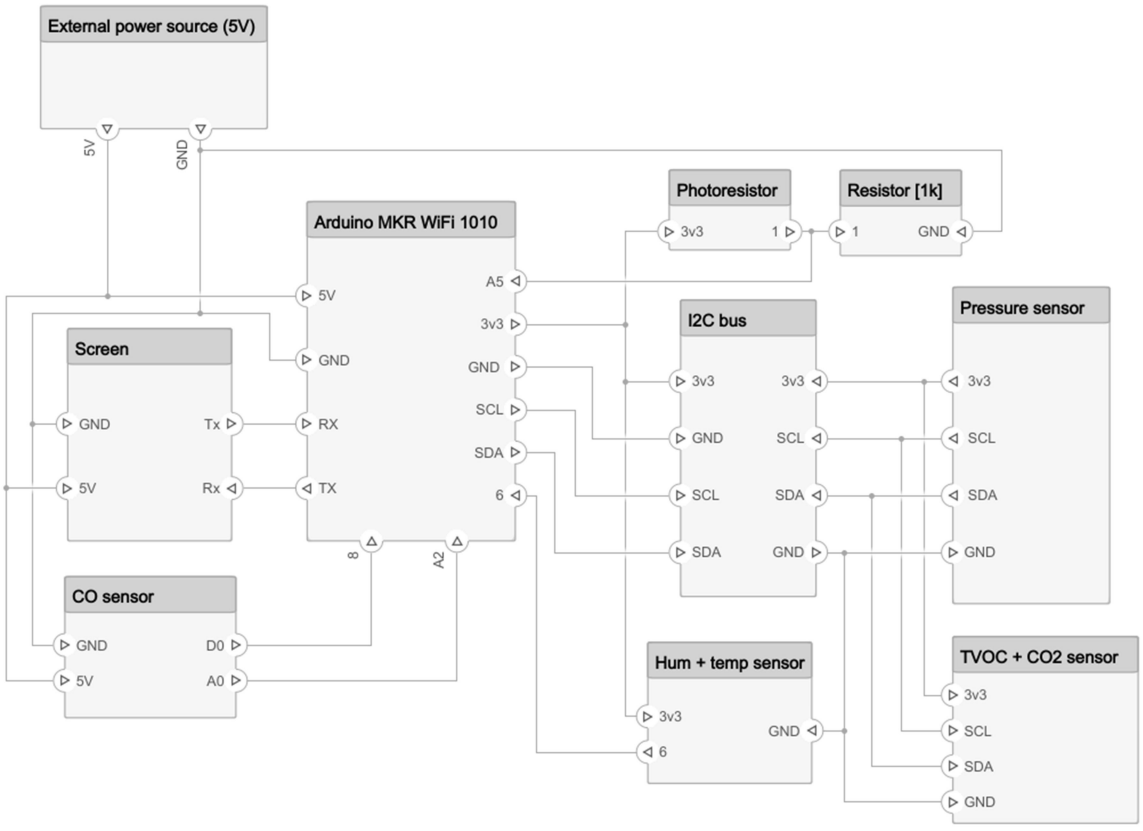


Figure 1 – System composer view

2.3 Components list

Below all used components are presented:

Table 1 – Components list

Name	Catalogue number	Link to the website
Microcontroller	Arduino MKR1010 ABX00023 - WiFi ATSAMW25 + ESP32	https://botland.com.pl/arduino-seria-mkr-oryginalne-plytki/12945-arduino-mkr1010-abx00023-wifi-atsamw25-esp32-ze-zlaczami-7630049200258.html
LCD screen	Nextion Enhanced 2.8" 320x240 NX3224K028	https://elty.pl/pl/p/Wyswietlacz-Nextion-Enhanced-2.8-320x240-NX3224K028-rezystancyjny-panel-dotykowy/1815
Temperature + humidity sensor	DHT11	https://botland.com.pl/czujniki-multifunkcyjne/1886-czujnik-temperatury-i-wilgotnosci-dht11-modul-przewody-5903351242448.html
Pressure + Altitude sensor	BMP280	https://botland.com.pl/czujniki-cisnienia/7245-bmp280-cyfrowy-barometr-czujnik-cisnienia-110kpa-i2cspi-33v-5904422310042.html
CO2 + TVOC sensor	CCS811 - SparkFun SEN-14193	https://botland.com.pl/czujniki-czystosci-powietrza/9144-ccs811-czujnik-czystosci-powietrza-co2-i2c-sparkfun-sen-14193-5904422366797.html
CO sensor	MQ-9	https://botland.com.pl/czujniki-gazow/3029-czujnik-tlenku-wegla-i-latwopalnych-gazow-mq-9-polprzewodnikowy-modul-niebieski-5904422359287.html
Photo resistor	GL5616	https://botland.com.pl/fotorezystory/1564-fotorezystor-5-10k-gl5616-10szt-5903351245739.html
Resistor	None – value 1k	https://botland.com.pl/rezystory-oporniki/8283-rezystor-tht-cf-weglowy-1w-1k-200szt-5904422305482.html
Rectifier diode	1N4148	https://botland.com.pl/diody-prostownicze/4927-dioda-prostownicza-1n4148-100v-015a-10szt-5903351244442.html
Electrolytic capacitor	105C [10uF/50V]	https://botland.com.pl/kondensatory-elektrolityczne-tht/211-kondensator-elektrolityczny-10uf-50v-5x11mm-105c-tht-10szt-5903351248259.html
LDO stabilizer	NCP1117DT18G	https://botland.com.pl/napiecie-wyjsciowe-18v/15251-stabilizator-ldo-18v-ncp1117dt18g-smd-to-252-5szt-5904422323516.html
Transistor	P-MOSFET IRF9Z34	https://botland.com.pl/p-mosfet/1661-tranzystor-p-mosfet-irf9z34-tht-5-szt-5904422356125.html?gclid=CjwKCAjw14uVBhBEEiwAaufYx7R7m-KRggiqfd4AHkD8akQQIZAEx2Z3mSz-jeujs7tDBocTxpYpaRoCPDoQAvD_BwE
Housing	None	Prepared by authors

2.4 Housing

Housing for all the elements was prepared in Solidworks environment. This is an environment used for creating the 3D models of numerous elements and then eventually prepare the technical drawing of the earlier prepared model. Housing is crated in the 3d printing technique.

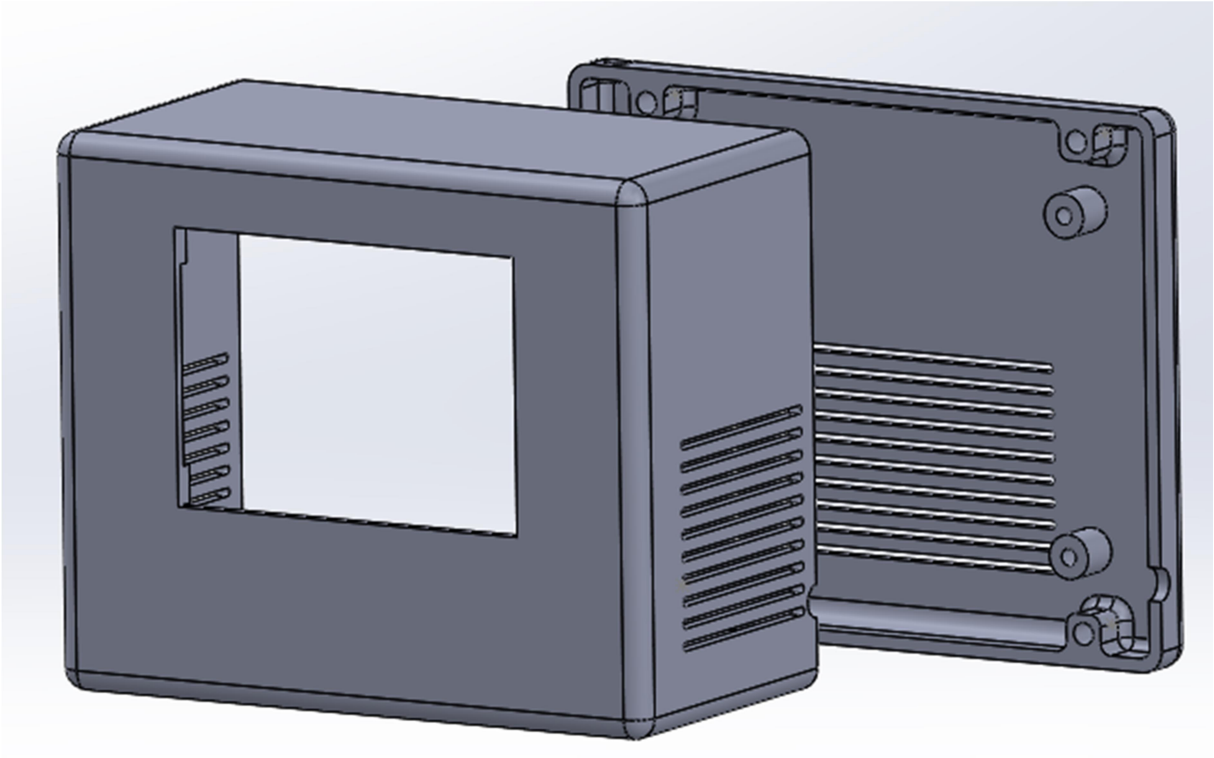


Figure 2 – Housing prepared in Solidworks environment

All components are supposed to be put into the housing and then attached with a screw.

3. Software

3.1 Basic software information

Software was written in Arduino IDE with usage of external, open-source libraries. Mostly, the libraries were used to manage the proper work of the sensors as well as the screen. For managing the screen, Nextion Editor and Nextion.h library were used – there is a need to add the Nextion libraries manually since libraries are not present in the libraries manager in the Arduino IDE.

3.2 Includes and global variables

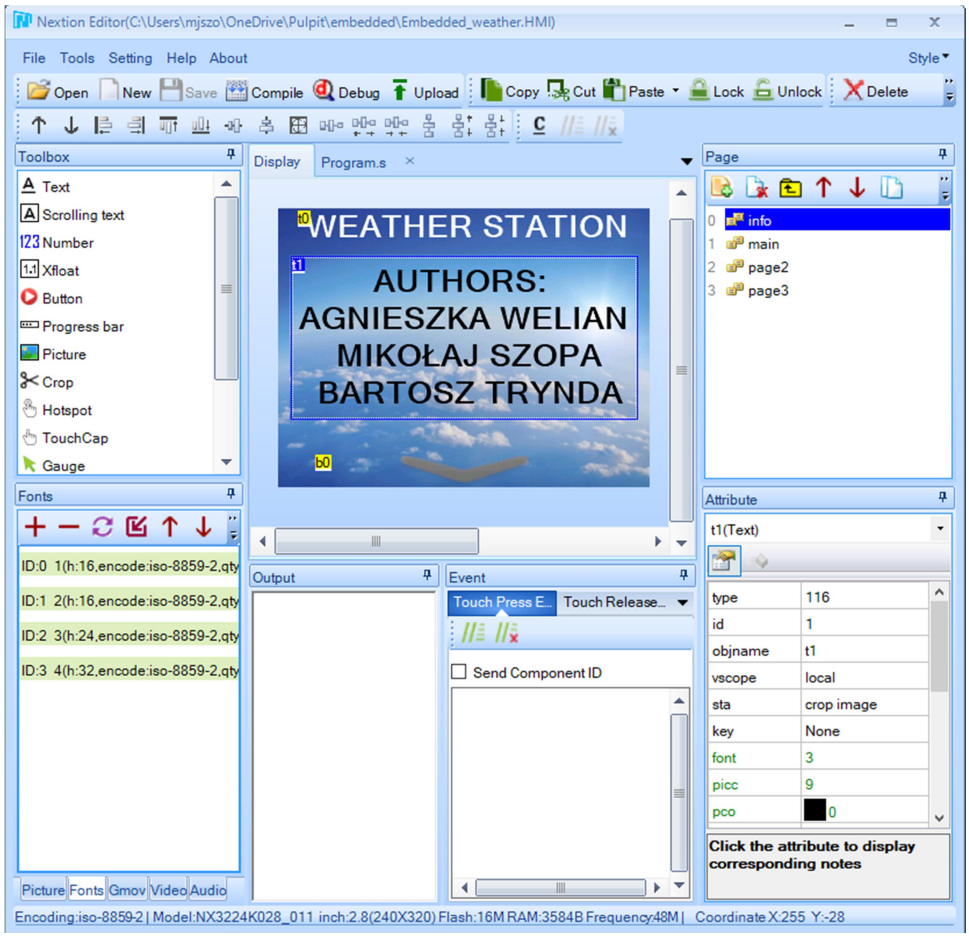
Below all the includes and global variables were defined:

```
#include <DFRobot_DHT11.h>
#include "DFRobot_CCS811.h"
#include <Adafruit_BMP280.h>
#include "Nextion.h"

DFRobot_CCS811 CCS811; //TVOC + CO2 sensor
DFRobot_DHT11 DHT;     //Humidity + Temperature sensor
#define DHT11_PIN 6

//CO sensor variables declaration
#define TIME_OF_BURNING 90000
#define TIME_OF_ACCUMULATING 60000
#define mosfetPIN 8
#define analogIN A2

//---------------------------------- Display variables-------------------------------------------------
// Declare your Nextion objects - Example (page id = 0, component id = 1, component name = "b0")
NexText tTempIn = NexText(1, 3, "tTempIn");
NexText tHum = NexText(1, 4, "tHum");
NexText tTVOC = NexText(2, 2, "tTVOC");
NexText tCO2 = NexText(2, 3, "tCO2");
NexText tCO = NexText(2, 4, "tCO");
NexText tPress = NexText(3, 1, "tPress");
NexText tAlt = NexText(3, 2, "tAlt");
NexText tLight = NexText(3, 6, "tLight");

// Register a button object to the touch event list.
NexTouch *nex_listen_list[] = {
  NULL
};

//loop variables
unsigned long t_now_loop,t_previous_loop, t_diff_loop;

//global variables------------------
unsigned long time_now,time_previous,time_difference;
bool burn;
float ppmvalue,sensor_volt,RS_gas,ratio,R0;
int ppmvalue_int,sensorValue;

//char array for display communication
char buff[20];
```

Figure 3 – Part of code: Includes and global variables

3.3 Connection with the screen

Nextion Editor is a program developed specially for designing and programming GUI application for Nextion screens. When creating new objects, special unique id is given, that is later used in Arduino code. It is possible to program screen directly from computer as well as download compiled program on microSD card and insert it into card slot in digital screen. Producer of the screen prepared set of libraries available online. In our case small changes were required to established serial connection including changing default Serial port numbers.



In Arduino code in the first step a declaration of used widgets is required. Type of widget and name for Arduino is written, then reference to GUI. In that place must be specified type of widget, page id on which widget is located, id of widget on particular page and given name. In case of buttons that have impact on Arduino program, example activate function, additional table with touch objects need to be filled, but in weather station project we use screen only for displaying information from sensors.

```
//--------------------------------- Display variables---------------------------------------------
// Declare your Nextion objects - Example (page id = 0, component id = 1, component name = "b0")
NexText tTempOut = NexText(1, 2, "tTempOut");
NexText tTempIn = NexText(1, 3, "tTempIn");
NexText tHum = NexText(1, 4, "tHum");
NexText tTVOC = NexText(2, 2, "tTVOC");
NexText tCO2 = NexText(2, 3, "tCO2");
NexText tCO = NexText(2, 4, "tCO");
NexText tPress = NexText(3, 1, "tPress");
NexText tAlt = NexText(3, 2, "tAlt");
NexText tLight = NexText(3, 6, "tLight");

// Register a button object to the touch event list.
NexTouch *nex_listen_list[] = {
  NULL
};
```

Figure 4 – Part of code: Connection with the screen

To write some data to label, we must choose created object, and use some of it build-in functions like setText.

```
tPress.setText(buff);
```

3.4 CO sensor

In case of CO sensor, no libraries were provided. This is a so called “burn-off” sensor, it means it require be powered with two different voltage values separately to collect particles and then to clear itself. To make it possible additional system consisting of diodes, Mosfet transistor and voltage stabilizer was prepared according to components documentation. The sensor works as a voltage divider, according to collected particles the voltage value on analog output changes. In Arduino code we must perform several calculations to obtain current resistance of sensor. Then the number of particles of specific gas can be calculated according to charts attached to sensor documentation.

```
//---------------------------------- Burnoff function ---------------------------------------------
void burnoff(unsigned long time_burn){
  time_now=millis();
  time_difference=time_now-time_previous;

  sensorValue = analogRead(A1);
  sensor_volt = ((float)sensorValue / 1024) * 4.8;

  R0 = (998-998*sensor_volt)/sensor_volt;
  R0 =700;
  RS_gas = (3.6/sensor_volt-1)*996; // Depend on RL on yor module
  ratio = RS_gas / R0; // ratio = RS/R0

  ppmvalue=595*pow(ratio,-2.24);
  ppmvalue_int=round(ppmvalue);
  ppmvalue_int= ppmvalue_int-1;
  sprintf (buff, "CO: %d", ppmvalue_int);
  strcat(buff," ppm");
  tCO.setText(buff);
  memset(buff, 0, sizeof(buff));

  if (time_difference > time_burn) {
    digitalWrite(mosfetPIN,HIGH);
    sprintf (buff, "CO: No data");
    time_previous=millis();
    burn=false;
  }}
```

Figure 5 – Part of code: CO sensor

3.5 Pressure sensor

Communication with pressure sensor is based on the I2C protocol (Master-Slave communication protocol in which Master sends a request with a proper time). To receive that data correctly, the Wire.h library was used. The transmission is started with the previously specified address (found in documentation) - after establishing the connection, values are read and connection ends. Unless the error is set to 1, the data is displayed on the screen.

```
//---------------------------------- Pressure and altitude function ----------------------------------------
void pressure_sensor(){
  Adafruit_BMP280 bmp280;
  bool error = false;
  if (!bmp280.begin(0x76)){
    error = true;
  }

  Wire.beginTransmission(0x76);

  //Read values
  float pressure    = bmp280.readPressure()/100 + 35;
  float altitude    = bmp280.readAltitude(1013.25) + 40;

  if (error == true) {
    strcat(buff,"Pressure: No data");
    tPress.setText(buff);
    memset(buff, 0, sizeof(buff));
    strcat(buff,"Altitude: No data");
    tAlt.setText(buff);
    memset(buff, 0, sizeof(buff));
    }
  else {
    sprintf (buff, "Pressure: %.2f", pressure);
    strcat(buff," hPa");
    tPress.setText(buff);
    memset(buff, 0, sizeof(buff));

    sprintf (buff, "Altitude: %.2f", altitude);
    strcat(buff," m");
    tAlt.setText(buff);
    memset(buff, 0, sizeof(buff));

    Serial.println();  // nowa linie
  }

  Wire.endTransmission(0x76);
}
```

Figure 6 – Part of code: Pressure sensor

3.6 CO2 + TVOC sensor

The sensor works in a similar way to the sensor from point 3.5 since communication is based on the I2C communication protocol. Unless the data is not read, sensor sends to screen an information "No data".

```
//---------------------------------- TVOC and CO2 function -----------------------------------------
void TVOC() {

  if(CCS811.checkDataReady() == true){
        float CO2 = CCS811.getCO2PPM();
        sprintf (buff, "eCO2: %.2f", CO2);
        strcat(buff," pmm");
        tCO2.setText(buff);
        memset(buff, 0, sizeof(buff));

        float TVOC = CCS811.getTVOCPPB();
        sprintf (buff, "TVOC: %.2f", TVOC);
        strcat(buff," ppb");
        tTVOC.setText(buff);
        memset(buff, 0, sizeof(buff));
    } else {
        strcat(buff,"TVOC: No data");
        tTVOC.setText(buff);
        memset(buff, 0, sizeof(buff));

        strcat(buff,"CO2: No data");
        tCO2.setText(buff);
        memset(buff, 0, sizeof(buff));
    }

    CCS811.writeBaseLine(0x447B);
    delay(1000);

  }
```

Figure 7 – Part of code: CO2 and TVOC sensor

3.7 Temperature + humidity sensor

Code for the humidity sensor was based on the DFRobot_DHT11.h. The library establishes the whole connection between the sensor and microcontroller. After reading the value, read value is assigned to another variable with float type and then sent to the screen. With the conditions authors made assumption that having the humidity value equal to 0 is not possible in the home conditions (due to the Earth's atmosphere).

```
//---------------------------------- Temperature and humidity function -----------------------------------------
void temperature_and_humidity() {
  DHT.read(DHT11_PIN);

  float temperatureDHT = DHT.temperature;
  float humidityDHT = DHT.humidity;

  if (humidityDHT == 0){
    strcat(buff,"Indoor: No data");
    tTempIn.setText(buff);
    memset(buff, 0, sizeof(buff));

    strcat(buff,"Humidity: No data");
    tHum.setText(buff);
    memset(buff, 0, sizeof(buff));
    }
  else {
    sprintf (buff, "Indoor: %.2f", temperatureDHT);
    strcat(buff," ^C");
    tTempIn.setText(buff);
    memset(buff, 0, sizeof(buff));

    sprintf (buff, "Humidity: %.2f", humidityDHT);
    strcat(buff," %");
    tHum.setText(buff);
    memset(buff, 0, sizeof(buff));
  }
}
```

Figure 8 – Part of code: Temperature and humidity sensor

3.8 Photoresistor

Photoresistor is a resistor which change its resistance with the change of light which falls on it. The code is simple – it consists of reading an analogue value (from 0 to 1023) and setting conditions in which the exposure is none, weak or strong from the sun or any other kind of light source. Values in conditions were determined experimentally.

```
//---------------------------------- Photoresistor function ---------------------------------------------
void photoresistor() {
  int read_value = 0;
  Serial.println("Photo");
  read_value = analogRead(A5);

  if (read_value <= 75) {
    tLight.setText("Exposture: None");
    memset(buff, 0, sizeof(buff));
  }
  else if (read_value > 75 && read_value < 350){
    tLight.setText("Exposture: Weak");
    memset(buff, 0, sizeof(buff));
  }
  else {
    tLight.setText("Exposture: Strong");
    memset(buff, 0, sizeof(buff));
  }
}
```

Figure 9 – Part of code: Photoresistor

3.9 Setup and loop functions

In the setup part, the communication with the screen is initialized and all parameters of deciding the period of the executing a program are set. The request for each sensor is sent within the 5 seconds by the execution of the functions which were mentioned before.

```
void setup(void)
{
    Serial1.begin(9600);
    nexInit();
    if(CCS811.begin() != 0){
        delay(1000);
    }
    t_now_loop=millis();
    t_previous_loop = t_now_loop;
}
void loop()
{
    t_now_loop=millis();
    t_diff_loop=t_now_loop-t_previous_loop;
    if(t_diff_loop >= 5000)
    {
      t_previous_loop = millis();
      TVOC();
      pressure_sensor();
      temperature_and_humidity();
      photoresistor();
      if (burn==true) burnoff(TIME_OF_BURNING);
      if (burn==false) accumulate(TIME_OF_ACCUMULATING);

  }
}
```

Figure 10 – Part of code: Setup and loop functions

4. Conclusions

Project was successfully performed – all the assumptions were met, each problem has been solved, and end-user device plays role of the smart home device. Convenient graphical user interface allows user to check whole parameters in the fast, and easy way. The compact housing with the modern design makes the device fit into any interior of the home.