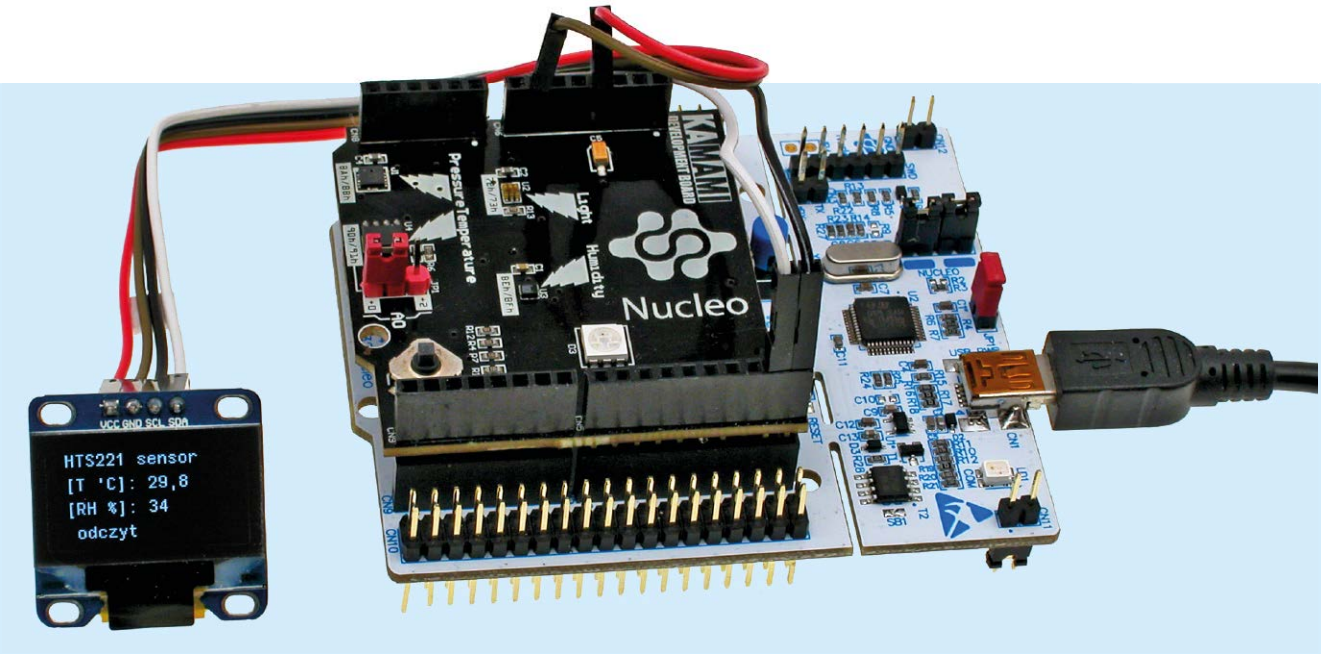


HTS221 mały czujnik o sporych możliwościach

Układ HT221 jest zintegrowanym, miniaturowym czujnikiem przeznaczonym do pomiaru wilgotności względnej i temperatury. Nadaje się on nie tylko do zastosowań związanych z pogodą – czułość i szybkość reakcji pozwalają na zastosowanie go w roli detektora obecności wody, np. w systemach ostrzegających przed zalaniem. Interfejs I²C i ogólnodostępne biblioteki oprogramowania pozwalają na szybkie wykonanie własnego urządzenia z HTS221.

Układ scalony czujnika HTS221 charakteryzuje się małymi wymiarami, szerokim zakresem napięcia zasilającego i małym poborem prądu. Ułatwia to stosowanie tego sensora w urządzeniach przenośnych, zasilanych z baterii lub akumulatorów. Najważniejsze parametry czujnika są następujące:

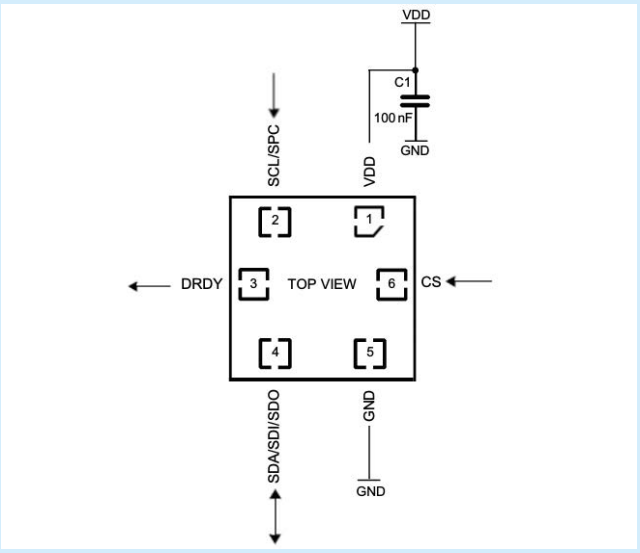
- Zakres pomiarowy wilgotności względnej: 0…100%.
- Dokładność pomiaru wilgotności: ±4,5% rH, w zakresie 20…80% rH, ±6% w całym zakresie.
- Dokładność pomiaru temperatury: ±0,5°C w zakresie +15…+40°C, ±1°C w zakresie 0…+60°C.
- Zakres temperatury pracy: -40…+120°C.
- Napięcie zasilania: 1,7…3,6 V.
- Średni prąd zasilający w trybie aktywnym: 2 µA przy częstotliwości odczytu 1 Hz.
- Interfejsy komunikacyjne: SPI i I²C.
- Wymiary: 2 mm×2 mm×0,9 mm.

Czujnik jest fabrycznie skalibrowany dla uzyskania maksymalnej dokładności pomiarów.

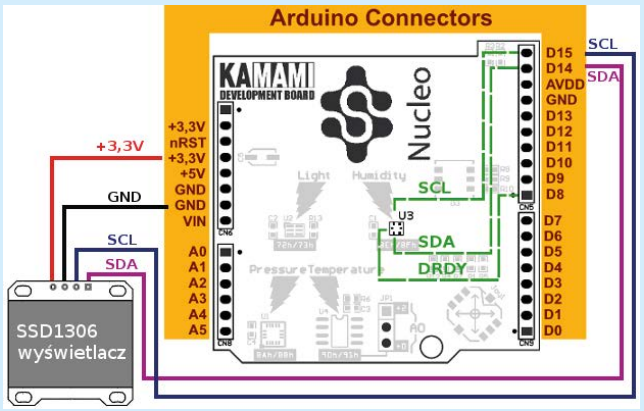
Miniaturowa obudowa HTS221 ma 5 wyprowadzeń umieszczonych na spodzie. Ich rozmieszczenie w widoku z góry pokazano na **rysunku 1**. Czujnik stanowi kompletny system, a jego obudowa zawiera wszystkie niezbędne bloki funkcjonalne. Jedynym dodatkowym komponentem zewnętrznym wymaganym przez jego aplikację jest kondensator o pojemności 100 nF filtrujący napięcie zasilające, który powinien być dołączony możliwie blisko doprowadzenia napięcia zasilającego. Interfejs komunikacyjny wybiera się za pomocą poziomu logicznego na wejściu CS. Logiczna „1” uaktywnia interfejs I²C, natomiast logiczne „0” – SPI. Szczegółowe informacje techniczne czujnika, dostępne są na stronie internetowej producenta pod adresem *https://goo.gl/2RwybH*.

Konfiguracja sprzętowa

Obudowa czujnika HTS221 jest zbyt mała, aby móc swobodnie dołączyć ją za pomocą przewodów do systemu



Rysunek 1. Rozmieszczenie wyprowadzeń sensora HTS221



Rysunek 2. Połączenie płytki KA-NUCLEO-Weather czujnikiem HTS221

z mikrokontrolerem. Na szczęście są dostępne moduły z wygodnymi do przyłączenia wyprowadzeniami, na których ten czujnik jest zamontowany. Ostatecznie, zestaw sprzętu użytego do testowania możliwości czujnika i pracy przy oprogramowaniu demonstracyjnym wyglądał następująco:

1. Płytka KA-NUCLEO-Weather z zamontowanym czujnikiem HTS221. Gniazda płytki w standardzie Arduino. Czujnik przystosowany do pracy z magistralą I²C.
2. Moduł STM32F411 NUCLEO-64 ze złączami w standardzie Arduino do osadzenia płytki KA-NUCLEO-Weather. Na module zamontowano kontroler STM32F411RE.
3. Miniaturowy wyświetlacz OLED SSD1306 z interfejsem I²C do prezentowania wyników pomiarów.

Na **rysunku 2** pokazano schemat połączeń wykonanych pomiędzy gniazdami a czujnikiem i wyświetlaczem. Zielonymi liniami przerywanymi zaznaczono połączenia pomiędzy zamontowanym na płytce KA-NUCLEO-Weather czujnikiem HTS221 i wyprowadzeniami na złączach. Do tych samych wyprowadzeń magistrali I²C, do których przyłączono czujnik, doprowadzone są przewody linii SCL i SDA wyświetlacza. Dodatkowo, wyświetlacz jest przyłączony do masy GND i pobiera napięcie zasilania z jednego z dostępnych wyprowadzeń +3,3 V. Zasilanie czujnika jest poprowadzone na płytce KA-NUCLEO-Weather i na rysunku nie zostało zaznaczone. Dodatkowa linia DRDY połączona z wyjściem D8 na złączu Arduino, służy do sygnalizowania gotowości czujnika do odczytu danych kolejnego pomiaru wilgotności i temperatury. Na płytce KA-NUCLEO-Weather zamontowano obowiązkowe dla interfejsu I²C rezystory zasilające linie SCL i SDA.

Narzędzia programistyczne i biblioteki

Napisanie funkcjonalnego oprogramowania dla czujnika HTS221 nie jest skomplikowane, jeśli korzysta się z dostępnych bibliotek i narzędzi programistycznych do automatycznego generowania kodu. Podczas pracy nad projektem korzystałem z następujących pakietów oprogramowania i bibliotek:

- Narzędzia do generowania szkieletu oprogramowania opartego o sterowniki HAL *STM32CubeMX* w wersji 4.13.0 (***https://goo.gl/AnLRuY***).
- Pakietu kompilatora *AC6 System Workbench* (*SW4STM32*, ***https://goo.gl/tm054L***).
- Biblioteki *STM32_LIBRARIES.zip* (***https://goo.gl/3cw7KS***). Niekomercyjna, darmowa biblioteka funkcji dla mikrokontrolerów STM32F4xx oparta na sterownikach HAL. Między innymi, w bibliotece można znaleźć gotowe funkcje do obsługi wyświetlacza SSD1306.
- Biblioteki funkcji do obsługi czujnika HTS221 *hts221_lib_stm32.zip* (***https://goo.gl/SYkbN0***).
- Dokumentacji biblioteki HAL (*UM1884: Description of STM32L4 HAL and Low-layer drivers*, ***https://goo.gl/AoCgow***).

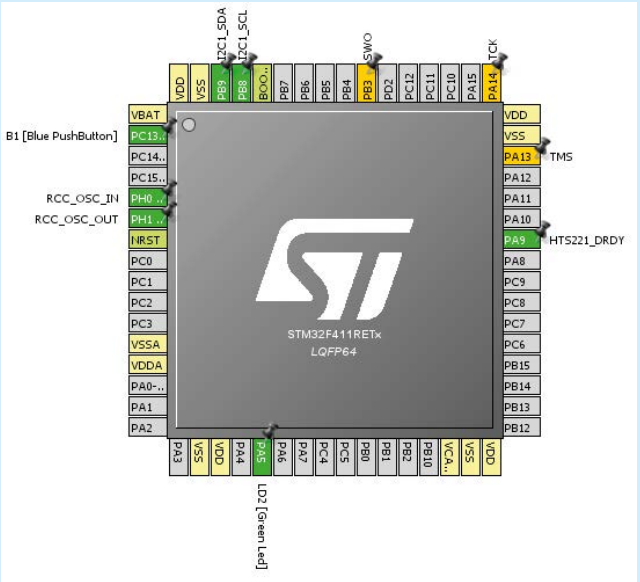
Założenia projektowe

Założeniem projektu było wykonanie w pełni funkcjonalnego miernika wilgotności i temperatury. Miernik miał działać w trybie pomiaru ciągłego i wyświetlać wyniki na wyświetlaczu OLED SSD1306. W kolejnych punktach, krok po kroku pokażę, jak wykonać oprogramowania takiego przyrządu.

STM32CubeMX – szkielet oprogramowania miernika wilgotności

Posługując się narzędziem programistycznym STM32CubeMX, w łatwy sposób można wygenerować szkielet oprogramowania dla wybranego typu mikrokontrolera lub płytki ewaluacyjnej. Szkielet będzie zawierał procedury inicjujące, utworzone w formacie plików odpowiednim dla wskazanego typu kompilatora.

Po uruchomieniu STM32CubeMX należy wybrać płytkę za pomocą menu *Board Selector* → *Type of Board: NUCLEO64* → *NUCLEO-F411RE*. Zostanie wyświetlony rysunek obudowy mikrokontrolera zamontowanego na płytce NUCLEO-F411RE, jak pokazano na **rysunku 3**. Ponieważ jako platformę docelową projektu wybrano określony typ płytki, widoczny na rysunku kontroler jest wstępnie skonfigurowany. Między innymi, zaznaczono dwa porty przyłączone na płytce do przycisku (B1[Blue PushButton]) i do diody LED (LD2[Green Led]). Można to pozostawić bez zmian.



Rysunek 3. Rysunek obudowy wyświetlony przez STM32CubeMX

Konfigurowanie peryferiów dodatkowych

Konfigurację wyprowadzenia portu, które będzie służyło do odczytywania poziomu logicznego wyjścia DRDY czujnika HTS221, należy wykonać w następujący sposób:

- Wskazujemy kursorem na oznaczenie wyprowadzenia portu PA9 i klikamy lewym przyciskiem myszy.
- Z wyświetlonej listy wybieramy *GPIO_Input*. Kolor wyprowadzenia zmieni się na zielony.
- Wybieramy zakładkę *Configuration* → *System* → *GPIO*. Na wyświetlonej liście klikamy na pozycję PA9.
- Zmieniamy ustawienia: *GPIO mode = Input mode, GPIO Pull-up/Pull-down = Pull-up, User Label = HTS221_DRDY.*

Wyprowadzenia portów, które będą pracowały „na usługach" interfejsu I²C (SDA, SCL) należy skonfigurować w następujący sposób:

- Na zakładce *Pinout* i liście *Peripherals* wybieramy pozycję I2C1 i zaznaczamy I²C.
- Wskazujemy kursorem wyprowadzenie portu PB9 i wybieramy opcję I2C1_SDA.
- Podobnie dla portu PB8, dla którego wybieramy opcję I2C1_SCL.

Dodatkowo, można zmienić ustawienia wewnętrznych przebiegów zegarowych:

- Na zakładce *Pinout* i liście *Peripherals* wybieramy pozycję RCC i zmieniamy ustawienie *High Speed Clock* (HSE) na *BYPASS Clock Source*. Odpowiada to sytuacji na płytce NUCLEO-F411RE, w której impulsy taktujące mikrokontroler są podawane na wejście OSC_IN. Gdyby kontroler miał pracować z dołączonym rezonatorem kwarcowym, należałoby wybrać opcję *Crystal/Ceramic Resonator*.
- Następnie otwieramy zakładkę *Clock Configuratin*.
- W polu *Input Frequency* wpisujemy częstotliwość impulsów zegarowych w MHz, czyli w tym przypadku „8".
- Zaznaczamy na schemacie impulsów zegarowych opcję *PLL Source Mux* na HSE, a *System Clock Mux* na PLLCLK. Jeżeli system wewnętrznych podzielników pętli PLL będzie miał ustawione następujące parametry: */M=4, *N=84, /P=2* to impulsy HCLK będą miały częstotliwość 84 MHz.

Główne ustawienia projektu

Aby wprowadzić ustawienia główne, należy wybrać *Project* → *Settings*. W polu *Project Name* wpisać wybraną nazwę dla projektu, a w polu *Project Location* podać ścieżkę dostępu do katalogu, w którym mają być zapisane pliki projektu. Wybrać z listy *Tolchain/IDE* nazwę pakietu kompilatora, w którego formacie zostaną wygenerowane pliki projektu. W naszym wypadku będzie to SW4STM32.

Generowanie plików projektu

Po wybraniu z menu *Project* → *Generate Code* zostanie wygenerowany szkielet programu i zapisany jako pliki w wybranym lub podanym folderze projektu.

Otwarcie wygenerowanych plików projektu za pomocą System Workbench for STM32

Należy otworzyć System Workbench for STM32 (AC6) podając w *Workspace Launcher* ścieżkę dostępu do katalogu, w którym znajduje się utworzony szkielet oprogramowania. Wybrać *File* → *Import* → *General* → *Existing Projects into Workspace* → *Next* i podać ścieżkę dostępu do projektu.

Import do projektu plików z procedurami obsługi wyświetlacza SSD1306

Zamiast pisać samodzielnie procedury obsługi wyświetlacza, skorzystamy z gotowych procedur znajdujących się w plikach **STM32_LIBRARIES.zip**. Najpierw należy rozpakować bibliotekę w wybranym katalogu. Przeglądając jej zawartość znajdziemy interesujące nas pliki (**tm_stm32_ssd1306.h** i **tm_stm32_ssd1306.c**). Aby móc posłużyć się funkcjami obsługi wyświetlacza należy najpierw je zaimportować do utworzonego projektu programu. Potrzebny będzie także import kilku dodatkowych plików z biblioteki. Można to zrobić w następujący sposób:

- W katalogu projektu środowiska System Workbench for STM32 tworzymy podkatalog dla plików procedur wyświetlacza SSD1306. W tym celu na wyświetlonym w *Project Explorer* drzewie katalogów projektu wskazujemy kursorem *Application*. Klikamy prawym przyciskiem myszki na *Application New* → *Folder* i w polu *Folder name* wpisujemy nazwę dodawanego folderu TM_SSD1306.
- Klikając prawym przyciskiem myszki na nazwie utworzonego folderu wybieramy *Import* → *File System* → *Next*.
- Wskazujemy katalog, w którym znajdują się rozpakowane pliki biblioteki STM32_LIBRARIES i zaznaczamy kwadraciki przy pozycjach: **tm_stm32_fonts.h**, **tm_stm32_fonts**, **tm_stm32_ssd1306.h** i **tm_stm32_ssd1306.c**.
- Po naciśnięciu *Finish* pliki zostaną przekopiowane do utworzonego w projekcie podkatalogu TM_SSD1306.

Należy jeszcze w projekcie zadeklarować ścieżkę dostępu do dodanych plików. W *Project Explorer* wskazujemy kursorem nazwę projektu. Po kliknięciu prawym przyciskiem wybieramy *Properities* → *C/C++ Build* → *Tool Settings* → *Includes*. Po naciśnięciu *Add* → *Workspace* należy wskazać nazwę folderu, w którym znajdują się dodane pliki. Po zatwierdzeniu wyboru nowa ścieżka dostępu zostanie dodana do listy. Można się o tym przekonać rozwijając w drzewie katalogu projektów pozycję *Includes*. W podobny sposób do opisanego należy utworzyć w projekcie kolejne podkatalogi i zaimportować z biblioteki kolejne pliki:

1. Utworzyć katalog TM_DELAY, zaimportować ***TM_STM32_DELAY.C***, ***TM_STM32_DELAY.H***.
2. Utworzyć katalog TM_GPIO, zaimportować ***TM_STM32_GPIO.C***, ***TM_STM32_GPIO.H***.
3. Utworzyć katalog TM_I2C, zaimportować ***TM_STM32_I2C.C***, ***TM_STM32_I2C.H***.
4. Utworzyć katalog TM_INC, zaimportować ***ATTRIBUTES.H***.

Do każdego utworzonego podkatalogu należy w opisany wcześniej sposób podać ścieżkę dostępu.

Korzystając z plików bibliotecznych STM32_LIBRARIES, dla prawidłowego przebiegu kompilacji należy jeszcze dodać do projektu plik definicji. Po wskazaniu kursorem podkatalogu TM_INC przyciska się prawy

Listing 1. Modyfikacja pliku nagłówkowego

```
/* Put your global defines for all libraries here used in your
project */
/*
        |PINSPACK 2    |
I2CX    |SCL    SDA    |
        |              |
I2C1    |PB8    PB9    |
*/
#define SSD1306_I2C     I2C1
#define SSD1306_I2C_PINSPACK     TM_I2C_PinsPack_2
//deklaracja dla transmisji do HTS221
#define HTS221_I2C     I2C1
//deklaracje wyprowadzenie DRDY HTS221
#define HTS221_DRDY_Port     GPIOA
#define HTS221_DRDY_Pin     GPIO_PIN_9
```

przycisk myszy i dalej *New → Header File*. W polu *Header File* należy wpisać *defines.h*. Po zatwierdzeniu nazwy za pomocą *Finish* zostanie utworzony szablon pliku nagłówkowego o podanej nazwie. Należy do niego wkleić tekst z **listingu 1**. Dzięki tym deklaracjom zostanie prawidłowo zainicjowany interfejs I2C1 oraz port połączony z wyprowadzeniem DRDY. Wygląd drzewa projektu z prawidłowo zaimportowanymi plikami bibliotecznymi pokazano na **rysunku 4**.

Od tego momentu program jest w stanie zainicjować SSD1306 i wyświetlić na ekranie komunikaty. Aby sprawdzić połączenie z wyświetlaczem, należy w pliku *main()* dodać kilka programu – pokazano je na **listingu 2**. Powinny one być umieszczone w sekcji USER CODE BEGIN 2. Po skompilowaniu programu i zaprogramowaniu mikrokontrolera na wyświetlaczu powinien zostać wyświetlony komunikat „HTS221 sensor temperatury i wilgotności”.

Pliki procedur czujnika HTS221

W opisany wcześniej sposób należy utworzyć w projekcie podkatalog HTS221 i z biblioteki HTS221_LIB_STM32 zaimportować pliki HTS221_i2c_lib.c i HTS221_i2c_lib.h. Tym razem bibliotekę procedur należy przystosować do współpracy ze sterownikami HAL. Sprowadza się to do zmiany dwóch plików odpowiedzialnych za odczyt i zapis do rejestrów czujnika. Na **listingu 3** pokazano sposób modyfikacji funkcji HTS221_I2C_Register_Read dostępnej w pliku HTS221_i2c_lib.c. Na **listingu 4** pokazano zmodyfikowana funkcję HTS221_I2C_RegisterWrite, która również jest dostępna w pliku HTS221_i2c_lib.c. Dodatkowo, w pliku HTS221_i2c_lib.c należy zmienić dyrektywę dyrektywę #include w następujący sposób:

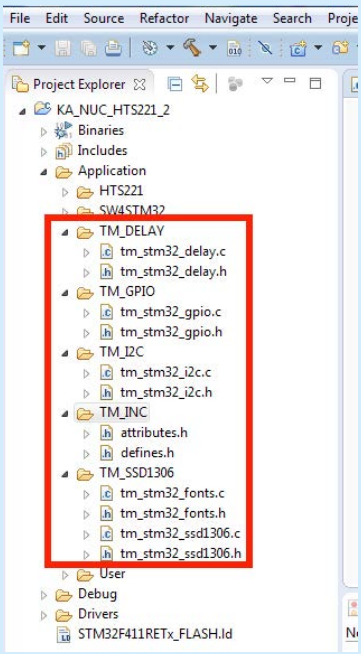
```
//#include „stm32f10x_i2c.h”  //wersja dla bibliotek CMSIS
#include „stm32f4xx_hal.h”  //wersja dla sterowników HAL
```

W pliku nagłówkowym HTS221_i2c_lib.h należy dodać następujące linie:

```
#include „defines.h”
#include „tm_stm32_i2c.h”
```

Zmiany w funkcji main()

Aby uruchomić program miernika wilgotności i temperatury, należy w funkcji *main()* dopisać kilka linijek kodu. Najpierw należy uruchomić procedurę inicjacji czujnika HTS221, a gdy się powiedzie odczytać jego



Rysunek 4. Wygląd drzewa projektu z zaimportowanymi plikami bibliotecznymi

Listing 2. Modyfikacja funkcji main()

```
/* USER CODE BEGIN 2 */
/* Init SSD1306 LCD 128 x 64 px */
if (TM_SSD1306_Init())
{
/* SSD1306 is connected */
    TM_SSD1306_GotoXY(30, 4);
} else
{
    /* SSD1306 is not connected */
}
TM_SSD1306_GotoXY(8, 4);
TM_SSD1306_Puts(„HTS221 sensor”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
TM_SSD1306_GotoXY(8, 20);
TM_SSD1306_Puts(„temperatury i”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
TM_SSD1306_GotoXY(8, 36);
TM_SSD1306_Puts(„wilgotnosci”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
/* Update screen, send changes to LCD */
TM_SSD1306_UpdateScreen();
/* USER CODE END 2 */
```

Listing 3. Modyfikacja funkcji HTS221_I2C_Register_Read

```
//wersja dla sterowników HAL
void HTS221_I2C_Register_Read(unsigned char regaddr, unsigned char *regval, unsigned char nob)
{
    uint8_t x;

    if (nob ==1)
    {
        TM_I2C_Read(HTS221_I2C, HTS221_I2C_Addr_R, regaddr, regval);
    }
    else
    {
        for (x=0; x<nob; x++)
        {
            TM_I2C_Read(HTS221_I2C, HTS221_I2C_Addr_R, regaddr+x, regval+x);
        }
    }
    return;
}
```

Listing 4. Modyfikacja funkcji HTS221_I2C_Register_Write

```
//wersja dla sterowników HAL
void HTS221_I2C_Register_Write(unsigned char regaddr, unsigned char *regval, unsigned char nob)
{
    uint8_t val, x;

    if (nob ==1)
    {
        val =*regval;
        TM_I2C_Write(HTS221_I2C, HTS221_I2C_Addr_W, regaddr, val);
    }
    else
    {
        for (x=0; x<nob; x++)
        {
            val =*(regval+x);
            TM_I2C_Write(HTS221_I2C, HTS221_I2C_Addr_W, regaddr+x, val);
        }
    }
    return;
}
```

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **66838**, pass: **643ttoo**

rejestry kalibracyjne. Kod źródłowy programu do dopisania w sekcji USER CODE BEGIN 2, za wcześniej dopisanym kodem uruchamiającym wyświetlacz, pokazano na **listingu 5**. Dalej, należy dopisać kod odczytujący wyniki pomiarów z czujnika, dokonujący konwersji i wyświetlający dane na wyświetlaczu. Kod należy umieścić w nieskończonej pętli while() pokazanej na **listingu 6**. Dodatkowo, należy zadeklarować użyte w procedurze odczytu zmienne. Prywatne zmienne globalne dodajemy w sekcji USER CODE BEGIN PV:

```
/* USER CODE BEGIN PV */
/* Private variables */
float tempCelsius = 25.50;
float humi = 55;
char tmp _ char[20];
unsigned char z;
GPIO _ Pin-
State stan _ HTS221 _ DRDY;
/* USER CODE END PV */
```

Zmienne lokalne dopisujemy w funkcji main():

```
  /* USER CODE BEGIN 1 */
int tmp _ int;
unsigned char tmp _ uchar;
  /* USER CODE END 1 */
```

Po skompilowaniu i wczytaniu pliku wynikowego do pamięci mikrokontrolera na wyświetlaczu w powinny być pokazywane wyniki pomiaru wilgotności względnej w procentach i temperatury z dokładnością do dziesiątej części stopnia.

Listing 5. Modyfikacja funkcji main() w celu uruchomienia miernika wilgotności i temperatury

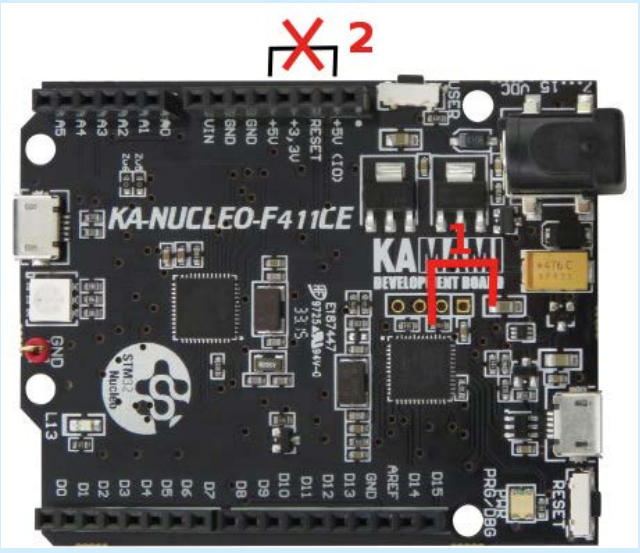
```
if(HTS221_I2C_Init())
{
    TM_SSD1306_GotoXY(15, 52);
    TM_SSD1306_Puts(„ID bledny”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
    /* Update screen, send changes to LCD */
    TM_SSD1306_UpdateScreen();
    while(1);
}
else
{
    TM_SSD1306_GotoXY(15, 52);
    TM_SSD1306_Puts(„ID prawidlowy”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
    /* Update screen, send changes to LCD */
    TM_SSD1306_UpdateScreen();
}
HTS221_I2C_Calib();


//Listing 6. Konwersja i wyświetlanie wyników pomiarów
/* Infinite loop */
/* USER CODE BEGIN WHILE */
z =0;
while (1)
{

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    stan_HTS221_DRDY =HAL_GPIO_ReadPin(HTS221_DRDY_Port, HTS221_DRDY_Pin);
    while (stan_HTS221_DRDY ==GPIO_PIN_RESET)
    {
        z++;
        HAL_Delay(10);
        if(z==200)
        {
            TM_SSD1306_GotoXY(15, 52);
            TM_SSD1306_Puts(„blad DRDY”, &TM_Font_7x10, SSD1306_COLOR_WHITE);
            TM_SSD1306_UpdateScreen();
            while (1);
        }
        TM_SSD1306_GotoXY(15, 52);
        TM_SSD1306_Puts(„pomiar        „, &TM_Font_7x10, SSD1306_COLOR_WHITE);
        TM_SSD1306_UpdateScreen();
        stan_HTS221_DRDY =HAL_GPIO_ReadPin(HTS221_DRDY_Port, HTS221_DRDY_Pin);
    }
    TM_SSD1306_GotoXY(15, 52);
    TM_SSD1306_Puts(„odczyt        „, &TM_Font_7x10, SSD1306_COLOR_WHITE);
    TM_SSD1306_UpdateScreen();
    HTS221_I2C_Read_Temp_Humi(&tempCelsius, &humi);
    tmp_int =(tempCelsius *10);
    tmp_uchar =tmp_int % 10;
    snprintf(&tmp_char[0], 20, „[T ,C]: %2d,%d  „, (int)tempCelsius,tmp_uchar);
    TM_SSD1306_GotoXY(8, 20);
    TM_SSD1306_Puts(&tmp_char[0], &TM_Font_7x10, SSD1306_COLOR_WHITE);
    snprintf(&tmp_char[0], 20, „[RH %C]: %d  „, ,%’,(int)humi);
    TM_SSD1306_GotoXY(8, 36);
    TM_SSD1306_Puts(&tmp_char[0], &TM_Font_7x10, SSD1306_COLOR_WHITE);
    TM_SSD1306_UpdateScreen();
    z =0;
}
```

Płytka alternatywna KA-NUCLEO-F411CE

Istnieje możliwość uruchomienia opisywanego oprogramowania na alternatywnej płytce rozwojowej z mikrokontrolerem STM32F411CE o nazwie KA-NUCLEO-F411CE. Jest to platforma sprzętowa o podobnych do NUCLEO-F411RE możliwościach, ze złączami w standardzie Arduino. Decydując się na tę płytkę, należy pamiętać o dwóch rzeczach.

Zgodnie z oznaczeniem na rys. 2 wyprowadzenie złącza CN6-2 jest połączone z napięciem +3,3 V. Na płytce KA-NUCLEO-F411CE w tym miejscu oznaczonym +5V(IO) jest doprowadzone napięcie +5 V. Dla uniknięcia konfliktu należy na płytce KA-NUCLEO-F411CE odciąć połączenie tego wyprowadzenia od napięcia +5 V. Najłatwiej można to zrobić w miejscu pokazanym na **rysunku 5** i oznaczonym cyfrą „2”. Przecinana ścieżka znajduje się na górnej części płytki drukowanej pomiędzy złączem a krawędzią. Do normalnej pracy płytka KA-NUCLEO-F411CE potrzebuje stałego połączenia z interfejsem USB komputera PC. Jej autonomiczna praca jest jednak możliwa. Należy tylko wlutować zworę zaznaczoną na rys. 5 kolorem czerwonym i cyfrą „1”.

RYSZARD SZYMANIAK, EP



Rysunek 5. Do normalnej pracy płytka KA-NUCLEO-F411CE potrzebuje stałego połączenia z USB –należy wlutować zworę zaznaczoną kolorem czerwonym i cyfrą „1”

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 66838, pass: 6433ttoo