**Listing 1. Konfigurowanie I2C**

void PmodHYGRO\_Config**()**

**{**

pmodHygroI2c**.**Instance **=** I2C2**;**

pmodHygroI2c**.**Init**.**AddressingMode **=** I2C\_ADDRESSINGMODE\_7BIT**;**

pmodHygroI2c**.**Init**.**DualAddressMode **=** I2C\_DUALADDRESS\_DISABLE**;**

pmodHygroI2c**.**Init**.**GeneralCallMode **=** I2C\_GENERALCALL\_DISABLE**;**

pmodHygroI2c**.**Init**.**NoStretchMode **=** I2C\_NOSTRETCH\_DISABLE**;**

pmodHygroI2c**.**Init**.**OwnAddress1 **=** 0x01**;**

// Set the I2C\_TIMINGR register to fulfill the HDC1080 timing requirements (max 400kHz).

pmodHygroI2c**.**Init**.**Timing **=** 0x10563046**;**

HAL\_I2C\_Init**(&**pmodHygroI2c**);**

// The HDC1080 requires at most 15 ms start-up delay.

HAL\_Delay**(**15**);**

**}**

**Listing 2. Konfigurowanie GPIO**

void HAL\_I2C\_MspInit**(**I2C\_HandleTypeDef **\***hi2c**)**

**{**

\_\_HAL\_RCC\_I2C2\_CLK\_ENABLE**();**

\_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE**();**

GPIO\_InitTypeDef GPIO\_InitStruct**;**

GPIO\_InitStruct**.**Mode **=** GPIO\_MODE\_AF\_OD**;**

GPIO\_InitStruct**.**Pull **=** GPIO\_NOPULL**;**

GPIO\_InitStruct**.**Speed **=** GPIO\_SPEED\_FREQ\_VERY\_HIGH**;**

GPIO\_InitStruct**.**Alternate **=** GPIO\_AF4\_I2C2**;**

GPIO\_InitStruct**.**Pin **=** GPIO\_PIN\_0 **|** GPIO\_PIN\_1**;**

HAL\_GPIO\_Init**(**GPIOF**,** **&**GPIO\_InitStruct**);**

**}**

**Listing 3. Pomiar i odczyt wartości temperatury i wilgotności**

void PmodHYGRO\_GetMeasurements(int16\_t\* temperature, int16\_t\* humidity)

**{**

// Write the address 0x00 pointing to temperature register in order to execute the measurements.

uint8\_t data**[**4**]** **=** **{**0**};**

HAL\_I2C\_Master\_Transmit**(&**pmodHygroI2c**,** PMODHYGRO\_ADDRESS**,** data**,** 1**,** 100**);**

// Wait for the measurements to complete.

HAL\_Delay**(**13**);**

// Read the four bytes starting from address 0. First two bytes contain the 14-bit temperature

// value, the last two bytes - humidity value.

HAL\_I2C\_Master\_Receive**(&**pmodHygroI2c**,** PMODHYGRO\_ADDRESS**,** data**,** 4**,** 100**);**

uint16\_t temperatureRaw **=** **(**data**[**0**]** **<<** 8**)** **|** data**[**1**];**

uint16\_t humidityRaw **=** **(**data**[**2**]** **<<** 8**)** **|** data**[**3**];**

// Calculate the real temperature and humidity values according to the formula described on the

// website: https://reference.digilentinc.com/reference/pmod/pmodhygro/reference-manual

float temp **=** **(**temperatureRaw **\*** 1.0 **/** **(**1 **<<** 16**)** **\*** 165**)** **-** 40**;**

**\***temperature **=** temp **\*** 10**;**

temp **=** **(**humidityRaw **\*** 1.0 **/** **(**1 **<<** 16**))** **\*** 100**;**

**\***humidity **=** temp**;**

**}**

**Listing 4. Funkcja wyświetlająca znaki**

void Oledrgb**::**DrawChar**(**uint8\_t col**,** uint8\_t row**,** char character**,** uint16\_t color**,** uint16\_t bgColor**)**

**{**

uint8\_t bitmap**[**FONT\_WIDTH **\*** FONT\_HEIGHT **\*** 2**]** **=** **{**0**};**

**for(**int i **=** 0**;** i **<** FONT\_WIDTH **\*** FONT\_HEIGHT**;** i**++)**

**{**

bitmap**[**2 **\*** i**]** **=** **(**bgColor **>>** 8**)** **&** 0xFF**;**

bitmap**[**2 **\*** i **+** 1**]** **=** bgColor **&** 0xFF**;**

**}**

uint8\_t charIndex **=** character **-** ' '**;**

**for(**int i **=** 0**;** i **<** 6**;** i**++)** **{**

**for(**int j **=** 0**;** j **<** FONT\_HEIGHT**;** j**++)**

**if** **((**font**[**charIndex**][**i**]** **<<** j**)** **&** 0x8000**)** **{**

bitmap**[**2 **\*** **(**i **+** **(**j **\*** FONT\_WIDTH**))]** **=** **(**color **>>** 8**)** **&** 0xFF**;**

bitmap**[**2 **\*** **(**i **+** **(**j **\*** FONT\_WIDTH**))** **+** 1**]** **=** **(**color**)** **&** 0xFF**;**

**}**

**}**

this**->**DrawBitmap**(**col**,** row**,** col **+** FONT\_WIDTH **-** 1**,** row **+** FONT\_HEIGHT **-** 1**,** bitmap**);**

**}**