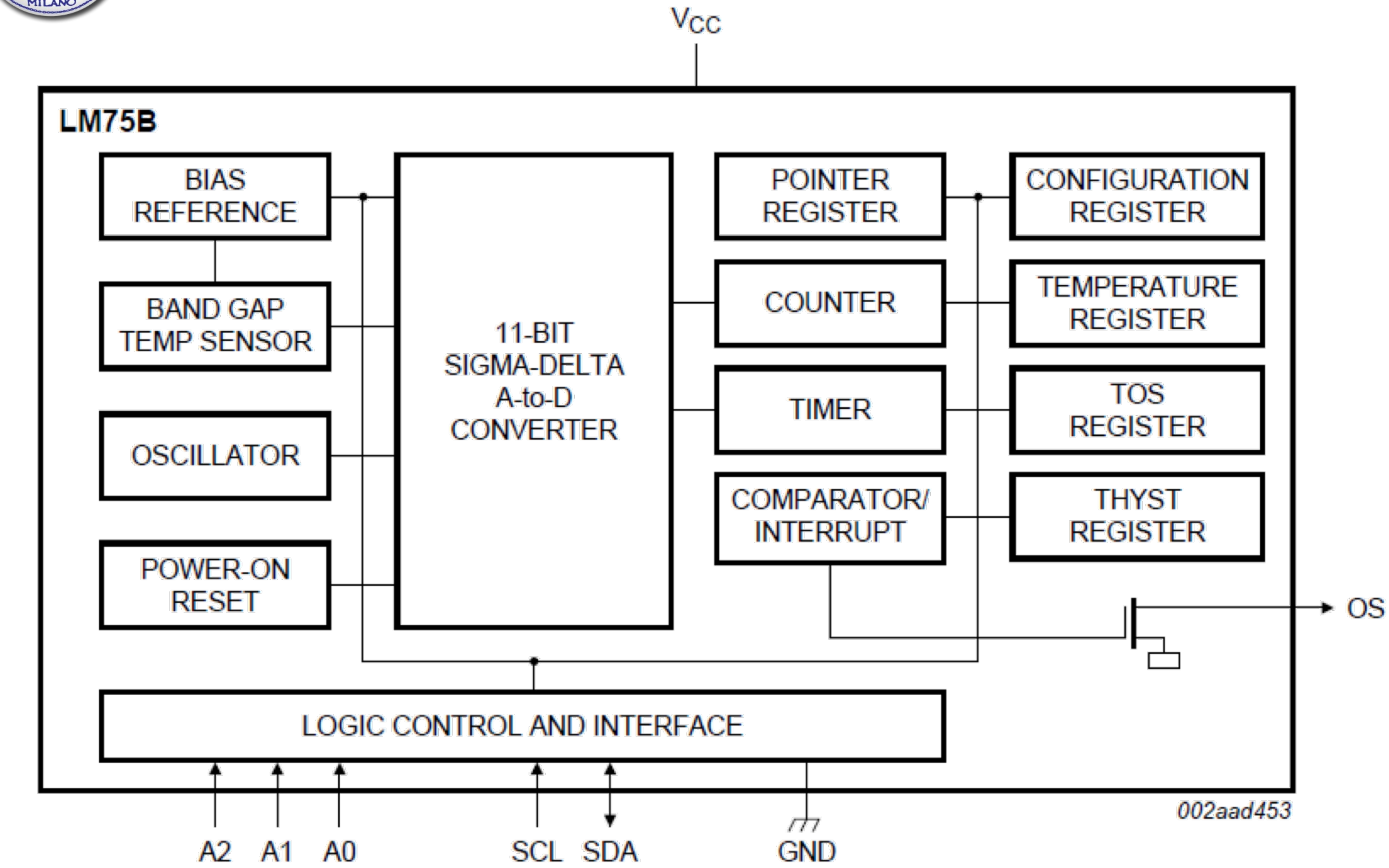
POLITECNICO
DI MILANO

www.polimi.it

STM32 – Temperature sensor

Federica Villa

Temperature sensor on POLIMI board



I^2C interface

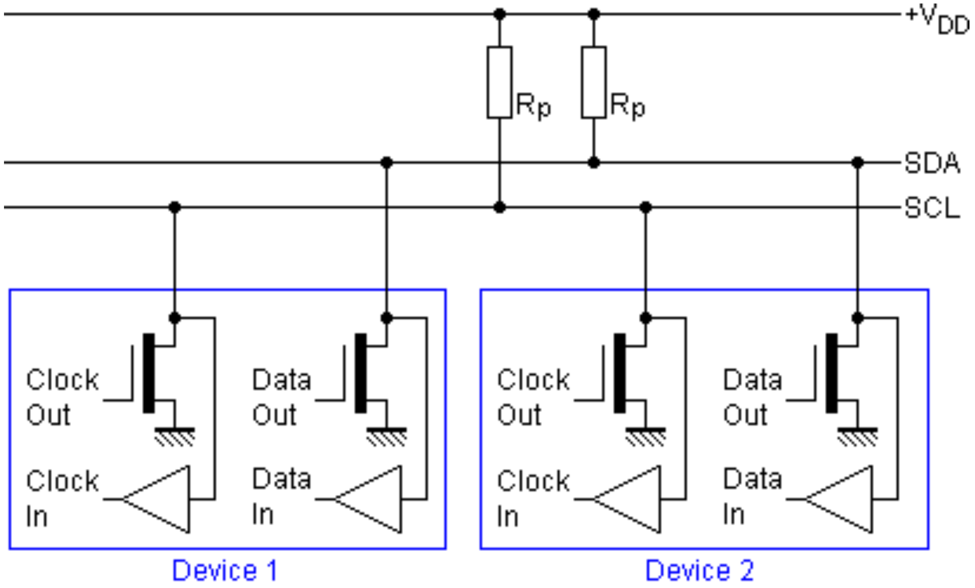
I^{2}C protocol basics

- Inter Integrated Circuit → IIC → I^{2}C
- Many devices share same bus
- **Multi-master, multi-slave protocol** → Open drain outputs (non-destructive arbitration, zero wins)

- Transmission rates: **100 kbps** (Standard mode), 400 kbps (Fast mode), 3.4 Mbps (High speed mode), 5 Mbps (Ultra-fast mode)
  Not supported by the STM32F401

- **Two wires** connection:
  - SDA (for data)
  - SCL (clock)
  - (+ common ground)

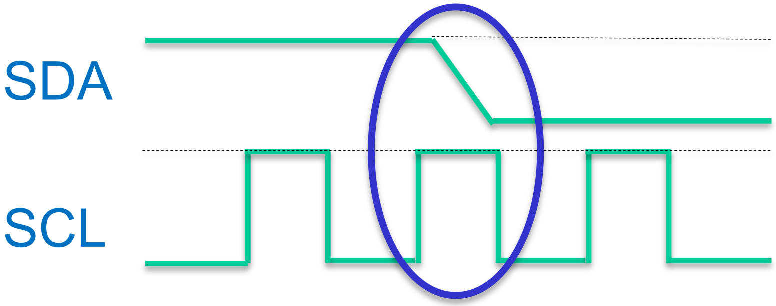
Both SDA and SCL are **bidirectional**



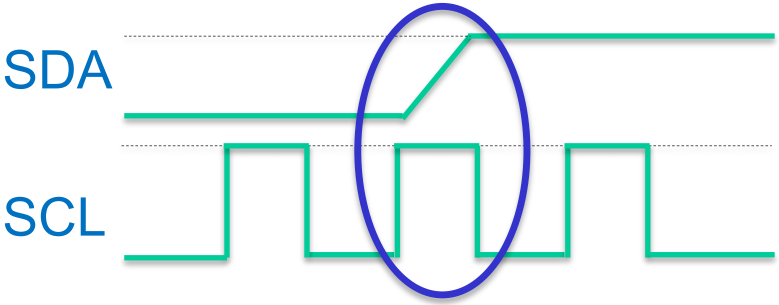
I^{2}C protocol: START and STOP

When there's no trasmission, both SCL and SDA are kept «high».

START: transition from high to low of SDA, while SCL is kept high.



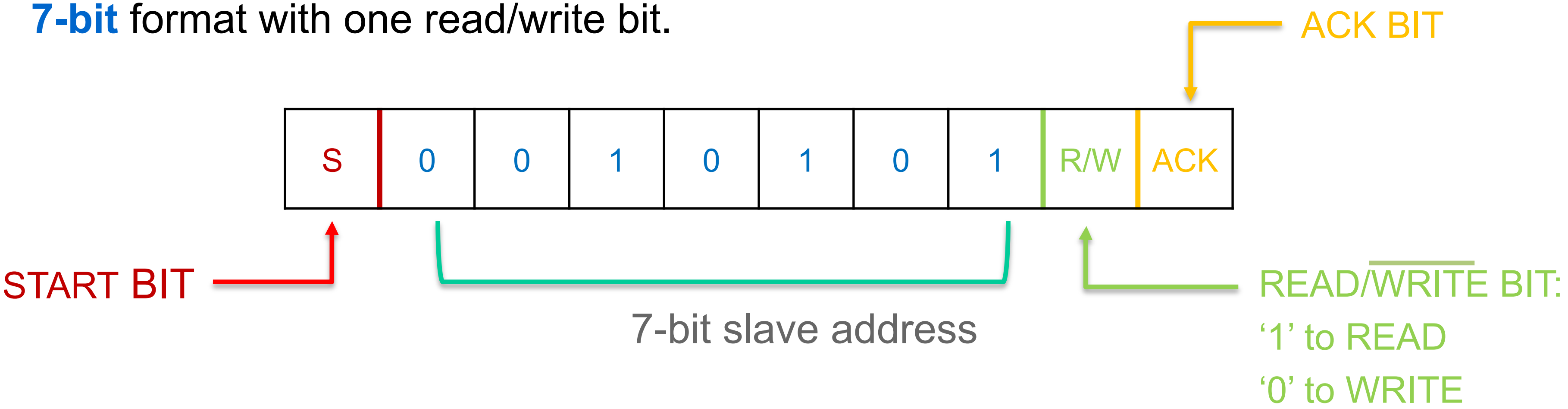
STOP: transition from low to high of SDA, while SCL is kept high.



I^{2}C protocol: addressing

Each device connected to the I2C bus has a **unique address**, in either of 2 formats:

- **7-bit** format with one read/write bit.



S	0	0	1	0	1	0	1	R/W	ACK

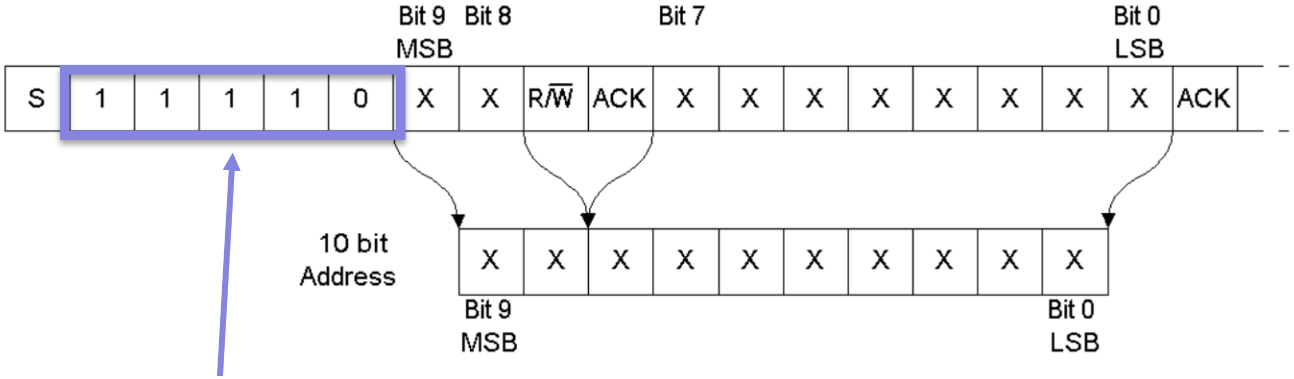
ACK BIT

START BIT

7-bit slave address

READ/$\overline{\text{WRITE}}$ BIT:
'1' to READ
'0' to WRITE

- **10-bit** format with one read/write bit



S	1	1	1	1	0	X	X	R/$\overline{\text{W}}$	ACK	X	X	X	X	X	X	X	X	ACK

X	X	X	X	X	X	X	X	X	X

This sequence tells that we are going to transmit a 10-bit address

I^{2}C protocol: data packets and speed

Data is packed in **bytes**, there's no limitation on the number of bytes.

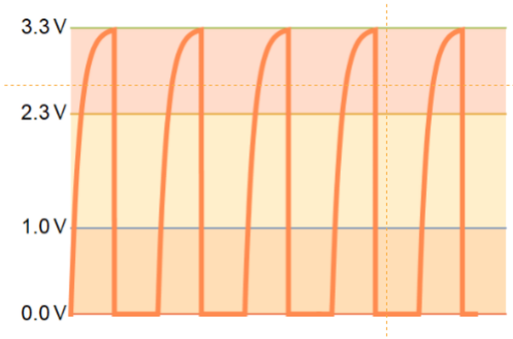
Acknowledge bit controls transmissions: if no ACK, communication stops:

- In **slave-receiver** mode, the slave necessarily needs to acknowledge the byte; otherwise the communication stops (e.g. if the address is wrong, no ACK is raised and communication halts)
- In **master-receiver** mode, the master deliberately doesn't ack the last byte in order to stop the communication.

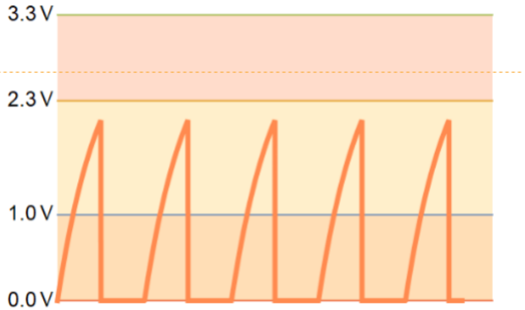
High-Low transitions are **fast** (driven by an NMOS with low impedance).

Low-High transitions limited by circuit **RC**:

pull-up resistor value typ: 1 to 10 kΩ / bus capacitance typ. 10 to 100 pF



Normal operation!

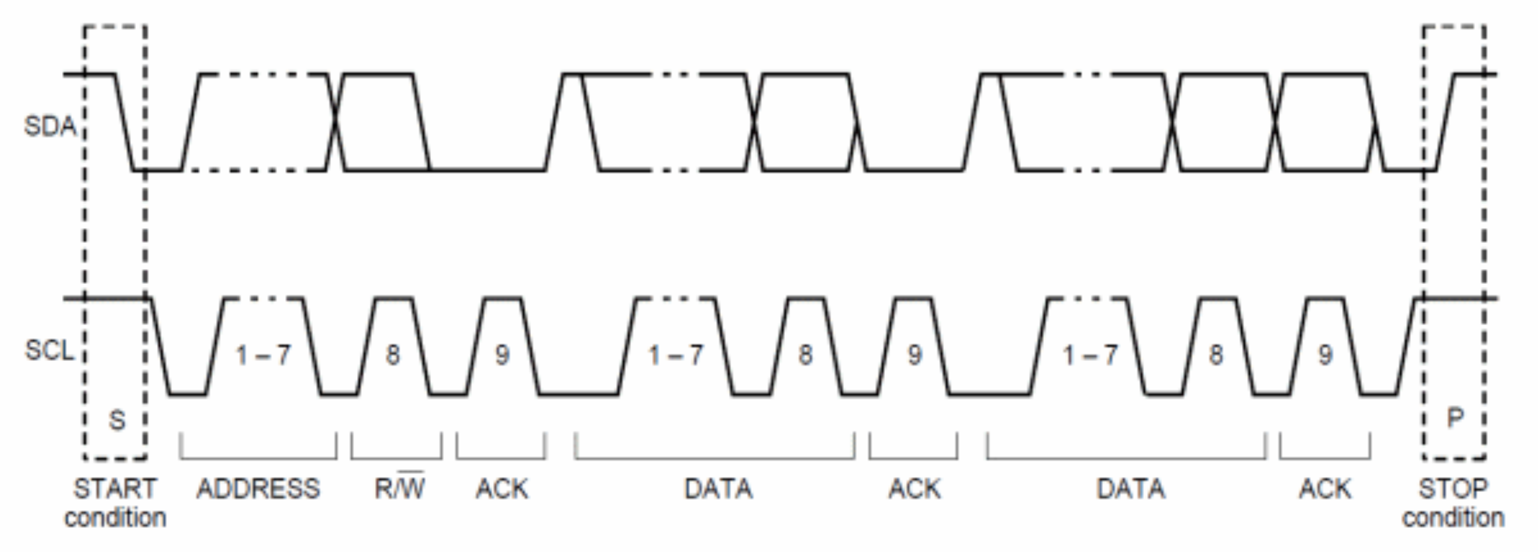


Too fast transitions!

I^{2}C protocol: transmission example

Since a SDA transition during SCL high will bring to a START/STOP condition, the **data transfer** necessarily occurs **when SCL is low**.

Example of transmission:



ADC HAL functions

HAL functions for I^{2}C:

HAL_StatusTypeDef **HAL_I2C_Master_Transmit**(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)

HAL_StatusTypeDef **HAL_I2C_Master_Receive**(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)

uint16_t DevAddress is the device address

uint8_t *pData is the data to be sent/received

uint16_t Size is the number of bytes to be sent/received

LM75/LM75B

- Band gap temperature sensor + 9 (B: 11) bit ADC
  LSB =0.5 °C (B: 0.125°C), range -55°C to 125°C
  8 bit integer temperature in °C (MSB sign bit)
  (two's complement digital data)

+ 3 bit decimal (LM75**B**)
+ 1 bit decimal (LM75)
Different board batches!

	MSByte								LSByte							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LM75B	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X
LM75	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X	X	X

- Temperature register (address 0x00) stores the converted temperature
- I^{2}C interface
- LM75/LM75B address: 1001 + 000 → 1001000 (7 bit address)

Project 1

Read the **temperature**
measured by the LM75
and send it to a remote terminal
every 1 second.

As a first step we will read only the MSB 8 bit.

Project hints

1. In CUBE configure the I2C in standard mode and enable the USART.

2. Generate the c code.

3. In the while loop read the MSByte of the temperature register using the I2C functions. When you declare the variables, watch out for the data types needed by the I2C functions.

4. Send the value to the PC using the USART interface and add a proper delay to repeat the loop every 1 s.

5. Do you need to do any conversions in order to obtain the value of temperature in °C?

Project 1b

Now we will modify the code
to read all 11 bits within an interrupt
routine

Project hints

1. Modify the code to read both MSByte and LSByte of the temperature register.

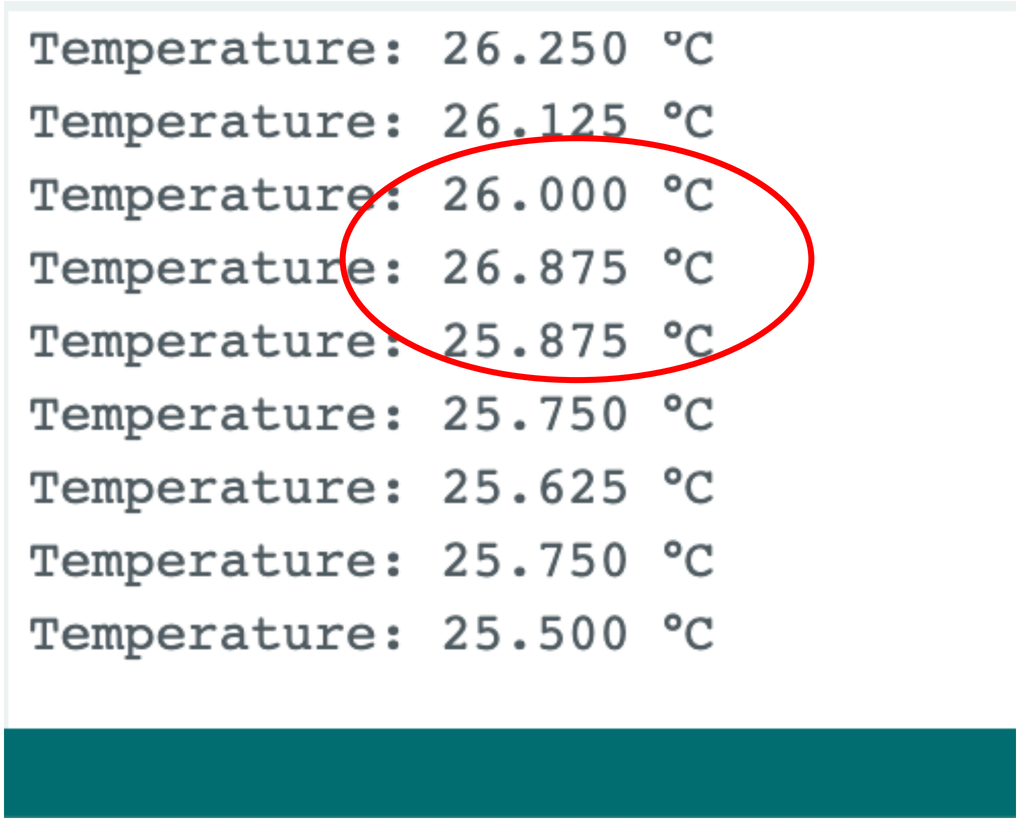
2. Convert the values of MSByte and LSByte in a value of temperature with 3 decimal places.
   Hint: verify your conversion function with the example values reported on the datasheet in Table 10 (pp. 9-10)

3. Send the value to the PC using the USART interface using a timer to generate an interrupt every second.

Bug to be checked

If you only read once the temperature bytes, you might observe this bug.

In this case the temperature was decreasing, but for some reasons (it is up to you to find the reason) it passes from 26 °C to 26.875 °C instead of 25.875 °C.



```
Temperature: 26.250 °C
Temperature: 26.125 °C
Temperature: 26.000 °C
Temperature: 26.875 °C
Temperature: 25.875 °C
Temperature: 25.750 °C
Temperature: 25.625 °C
Temperature: 25.750 °C
Temperature: 25.500 °C
```