Nuxtu SAS
Technical examination
Software Engineer - Low-level programming
14/Jan/2022
Presentation day: 20/Jan/2022 - 10:00 to 17:00
Presentation location: CL 109 # 18C-17 OF 301

For this technical examination you may use any books, notes, web pages, tutorials, documentation, or any other type of information source. You **may not** discuss this exam or questions related to the exam with your fellow contestants. It is also prohibited to ask for help from anyone you know that has experience on these topics, doing so will disqualify you from this and any future selection processes at *Nuxtu*.

Feel free to ask any question related to this examination to our *Chief Technology Officer* through WhatsApp or phone call at *+57 (301) 658 - 3977*, remember: **the only bad question is an unasked one**.

At the end of this examination, the following deliverables are expected:

- A forked GitHub repository from this original repository, with continuous commits on your progress. **Your last commit before the deadline will be evaluated**.
- A well-documented C or C++ project in your chosen IDE that solves the proposed challenge. Code complexity and design, execution time, coding style and resilience will be taken into account during the grading process.
- Be prepared to answer any questions related to your solution. This will show us that you completely understand your implementation, which is the most important part of this examination.

Work hard. Have fun. Make history!

Jeff Bezos

Your challenge will be to implement the low level code that runs inside one of the *Nuxtu* gas sensor arrays. The sensor array is controlled by a STM32F303VCT6 micro-controller that will be emulated by a STM32F3DISCOVERY board for the purpose of this challenge (Further reading on STM32F3DISCOVERY: https://www.sigmaelectronica.net/manuals/STM32F3DISCOVE.pdf).

Your code will be tested using the STM32F3DISCOVERY board to simulate the sensor array as slave, connected via I^2C with a Nvidia Jetson AGX Xavier as master. The Jetson device will be running our main software (*SensesCube*) which will try to communicate with your code by using a predefined protocol.

1 Understanding the *Nuxtu* gas sensor array

The gas sensor array is a device based on the micro-controller STM32F303VCT6 designed to read environmental variables and multiple gas sensor analog outputs. The gas sensor array PCB is equipped with sensors SHT31-ARP-B and KP229-E2701-XTMA1 for external temperature, pressure and humidity readings, as well as the internal STM32F303VCT6 temperature sensor for PCB temperature reading. All this information is sent to *SensesCube*, the main program used to read and control *Nuxtu* sensors.

The general architecture of a simple gas sensor array is shown on Figure 1.

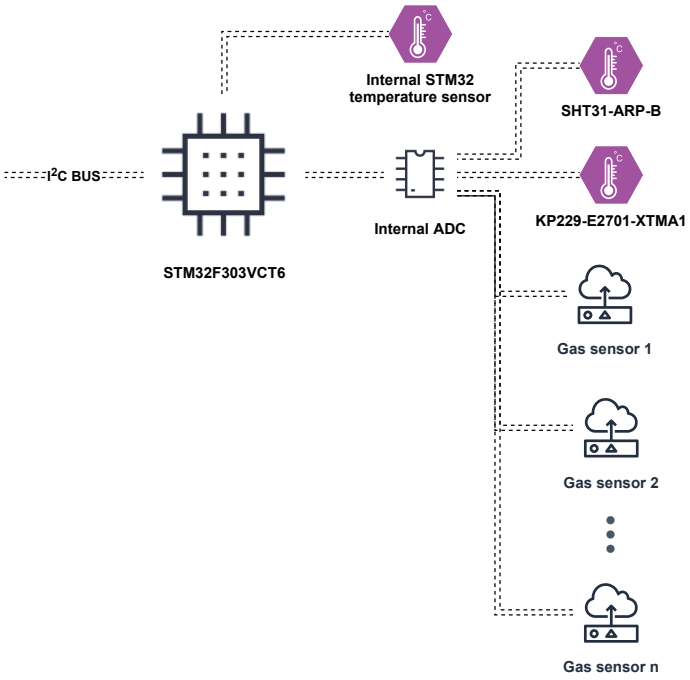

Figure 1: Gas sensor array general connections diagram

It is important to take into account when programming the micro-controller that the following components are variable and may **not** be present on all gas sensor arrays. (Specific characteristics of the sensor array will be given on test day, so code must be as generic as possible).

- **SHT31-ARP-B**
- **KP229-E2701-XTMA1**
- **Gas sensors:** The number of gas sensors is variable, and the ADC pins where those sensors are connected is also variable. There will always be at least **one** sensor on the gas sensor array.

2 Connections on the I^2C bus

Every sensor array that is used in a *Nuxtu* device implementation is connected with the main device (Jetson AGX Xavier) through an I^2C multi-device bus. Figure 2 shows the electronic architecture that will be used to test your implementation.

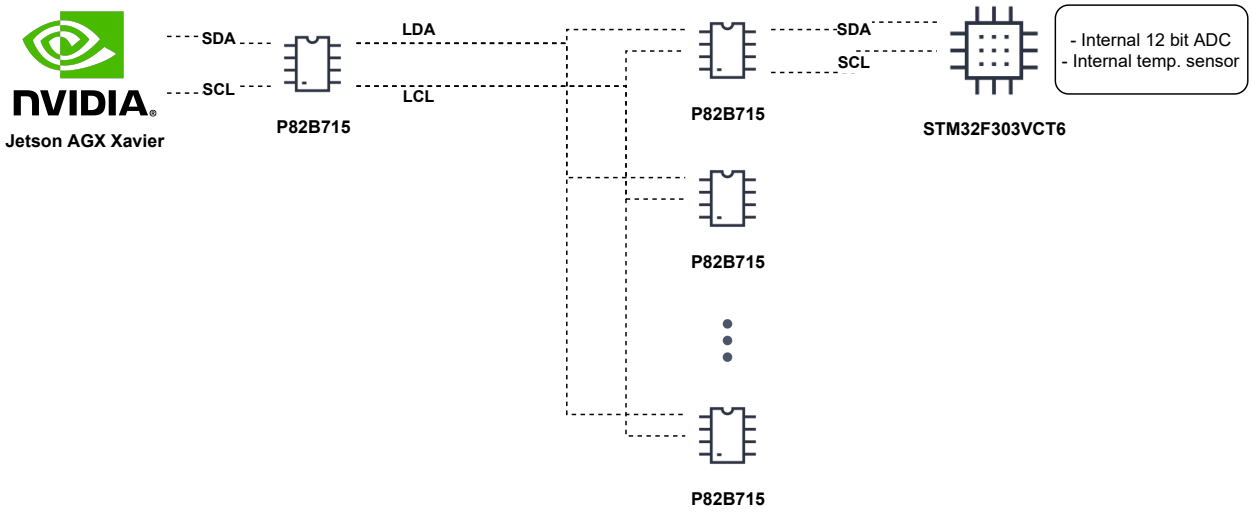

Figure 2: Connections between the STM32F3DISCOVERY and the Nvidia Jetson

According to Figure 2, the Jetson AGX Xavier device is the master of the I^2C communication, and every STM32F303VCT6 slave represents a gas sensor array. On a typical implementation, the master will request device metadata information such as device type, id and sensor types once a I^2C connection is detected and after that it'll continuously request all sensor voltages and meteorological information. Further details about this protocol are shown on section 4.

Note 1: It is completely **MANDATORY** to implement all the I^2C communications by using interrupts. Polling implementations will not be graded.

3 Sensor readings

Each sensor array is capable of reading the following data from their environment:

1. n sensor voltages between 0mV and 3300mV, where n is the number of sensors connected to the micro-controller ADC.

2. Environment temperature, from an analog temperature sensor (SHT31-ARP-B) connected to the micro-controller ADC.

3. Micro-controller temperature, from the internal STM32F303VCT6 temperature sensor.

4. Environment humidity, from an analog humidity sensor (SHT31-ARP-B) connected to the micro-controller ADC.

5. Environment absolute pressure, from an analog absolute pressure sensor (KP229-E2701-XTMA1) connected to the micro-controller ADC.

The specific value of n and the analog ADC pins where each sensor (including environmental sensors) is connected will be given at the examination time an must be left as general as possible in the code, since you will have to set the specific pins and number of sensors (n) on that day.

Note 2: Using a RTOS (e.g. FreeRTOS) to schedule the multiple tasks will grant additional points in the test, but it is not mandatory.

Note 3: For the SHT31-ARP-B and KP229-E2701-XTMA1 sensors, assume that the output will never be above 3.3 V.

Note 4: Not all sensor arrays have the environment temperature, micro-controller temperature, environment humidity and environment absolute pressure measurements. This will be further explained on section 4.

4 Communication protocol with SensesCube

SensesCube is the main software used by *Nuxtu* to communicate with every sensor device using a highly standardized set of instructions. For this test, you will have to program the micro-controller so that it is capable of responding to the following messages:

- `uint8 REQUEST_DEVICE_TYPE = 1`
- `uint8 REQUEST_DEVICE_METADATA_BASIC = 2`
- `uint8 REQUEST_DEVICE_METADATA_COMPLETE = 3`
- `uint8 REQUEST_DEVICE_VOLTAGE_DATA = 4`

A detailed explanation of each command is given in the following subsections.

Note 5: SensesCube uses the Python implementation of `smbus2` to communicate with each peripheral device, this communication is achieved using the `i2c_rdwr` instruction. In the detection process over the I^2C bus, a single byte is requested from the micro-controllers by using the `read_byte(address)` (offset 0) instruction, the micro-controller must answer with 0 or 255 to be recognized by SensesCube.

4.1 REQUEST_DEVICE_TYPE

This message is used to acknowledge the device type that has been detected, since multiple device types can be connected to the same I^2C bus. For the nose sensor array, the SensesCube identifier is `0`.

- **Jetson sends:** `REQUEST_DEVICE_TYPE` (uint8 - 1 byte)
- **Expected response:** `0` (uint8 - 1 byte)

4.2 REQUEST_DEVICE_METADATA_BASIC

This message is used to get the most basic sensor array metadata from the device. The most important part of the message is the number of sensors (n), since this will be used to calculate the length of the other 2 messages.

- **Jetson sends:** `REQUEST_DEVICE_METADATA_BASIC` (uint8 - 1 byte)
- **Expected response:** 14 bytes as follows: — PCB unique ID (int16 - 2 bytes) — Number of sensors (int8 - 1 byte) — Manufacturing date (char[10] - 10 bytes) — PCB capabilities (bool[8] (only 4 used) - 00001111 - 1 byte) —

These are the characteristics of each part of the message:

- **PCB unique ID:** Consecutive for the PCB ID, will be given on test day.
- **Number of sensors:** Number of sensor ADC readings, will be given on test day.
- **Manufacturing date:** PCB manufacturing date as string on `dd/mm/yyyy` format. Set it to the test day.
- **PCB capabilities:** Only the 4 LSBs are used to specify the environmental sensors that the PCB has, since not all readings are mandatory. The order is as follows: `temperature_degC`, `temperaturePCB_degC`, `humidity_percent`, `absolutePressure_kPa` `(00001111)`. Capabilities will be given on test day and must be left parametrized.

4.3 REQUEST_DEVICE_METADATA_COMPLETE

This message is used to get the metadata of each sensor that is connected to the gas sensor array. The length of this message is defined by n, since the complete information must be retrieved for all sensors.

- **Jetson sends:** `REQUEST_DEVICE_METADATA_COMPLETE` (uint8 - 1 byte)
- **Expected response:** (47 x n) bytes as follows: — Sensor name (char[11] - 11 bytes) — Sensor type (char[14] - 14 bytes) — Main gas (char[20] - 20 bytes) — Response time (int16 - 2 bytes) — x n

These are the characteristics of each part of the message:

- **Sensor name:** Commercial reference of current sensor as string, will be given on test day.
- **Sensor type:** Current sensor technology type as string, will be given on test day.

- **Main gas:** Current sensor main response gas as string, will be given on test day.
- **Response time:** Current sensor response time (t_{90}) in seconds, will be given on test day.

These four characteristics are sent n times in the same message, once for each sensor.

4.4 REQUEST_DEVICE_VOLTAGE_DATA

This message is used to continuously get the voltage for each sensor and the device atmospherical data. It is important to take into account that the length of this message is **not** dependent on the PCB capabilities variable, if the PCB does not have an specific atmospherical data capability any value can be sent on the corresponding field, SensesCube will automatically ignore this value.

- **Jetson sends:** `REQUEST_DEVICE_VOLTAGE_DATA` (uint8 - 1 byte)
- **Expected response:** 16 + (4 x n) bytes as follows: — Temperature (float32 - 4 bytes) — Micro-controller temperature (float32 - 4 bytes) — Humidity (float32 - 4 bytes) — Absolute pressure (float32 - 4 bytes) — [Voltage (float32 - 4 bytes) x n] —

These are the characteristics of each part of the message:

- **Temperature:** Measurement from SHT31-ARP-B in Celsius.
- **Micro-controller temperature:** Measurement from the internal temperature sensor in Celsius.
- **Humidity:** Measurement from SHT31-ARP-B in % from 0% to 100%.
- **Absolute pressure:** Measurement from KP229-E2701-XTMA1 in kPa.
- **Voltage:** The n voltage measurements from sensors in mV.