DRIXIT TECHNOLOGIES

Firmware Engineer - Development study case

This document summarizes the main design decisions taken for the development of the solution to the given case study.

User requirements

The main user design inputs were given in the PDF document Firmware Engineer – Ejercicio.

Technical specifications

The solution focuses on the firmware, but there are several important hardware considerations:

1- The microcontroller selected is the ARM Cortex M4 (+M0) LPC4367 from NXP. It is a mature product with a rich development ecosystem, has enough hardware power to support FreeRTOS and a very wide range of peripherals to meet the application needs. Among them: UART, I2C, SPI and SPIFI.
2- On the other hand, to speed up the development phase of the solution, it is considered that the firmware developer has an OM13088 board that incorporates this microcontroller and a flash memory equivalent the one required by the user.

Regarding firmware development, there are three main components to be considered: LIS3MDL sensor driver, flash memory W25Q80DV driver and FreeRTOS main application.

LIS3MDL sensor driver design considerations:

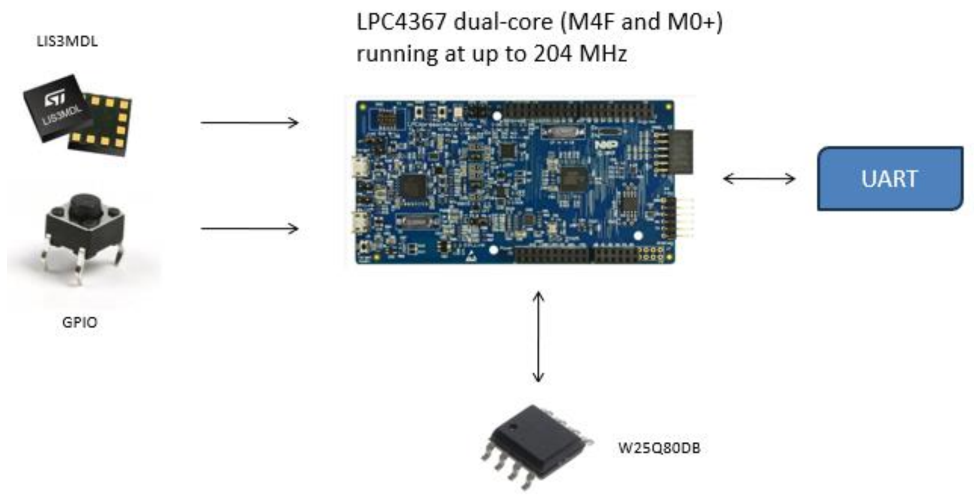
- Hardware interface: I2C transfer oriented.
- Sensor sampling rate: 80Hz
- Driver architecture: state machine driven by I2C interruptions.
- Multi-instance driver to allow multiple LIS3MDL sensors on one I2C port (at least two instances because there is only one configurable address bit).

Flash memory W25Q80DV driver design considerations:

- Handle small sets of data (tens of bytes)
- Scalable for other data
- Error correction (reliability)
- The driver is SPIFI based. This peripheral supports several flash types and vendors. The driver is hardware-dependent so can only be instantiated once. But with this approach it is very easy to access as on-chip memory with direct address mapping.
- It is necessary to erase (0xFF) the smallest possible memory sectors and read/write to without affecting the rest of the memory.
- For simplicity, the driver provides access to any part of the memory in “lots” of 32 bytes. With this strategy, the user doesn’t need to know about the memory sectors, pages and maps. Each memory lot is addressable with a single ID number.
- The above assumption allows to save data read from the sensor with an easy consecutive ID number. In case to have multiple sensors and need to add more information to the raw sensor data (for example the sensor type or a timestamp) there are several reserved bytes in the data structure. Also, it is possible to split the memory map in ”sectors lots“ to save different sensor data in different parts of the memory and retive the data without the need of a search algorithm.

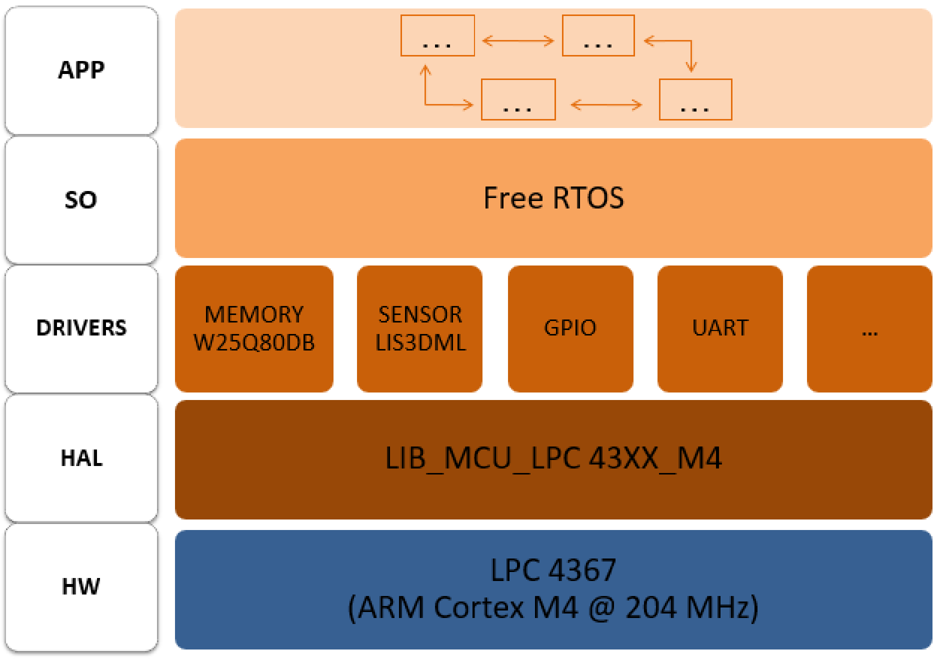
Firmware development testbench

To test some of the features on real hardware, the NXP OM13088 board was used. The block diagram is shown below.



Firmware layers organization

The proposal for the firmware structure is shown below.

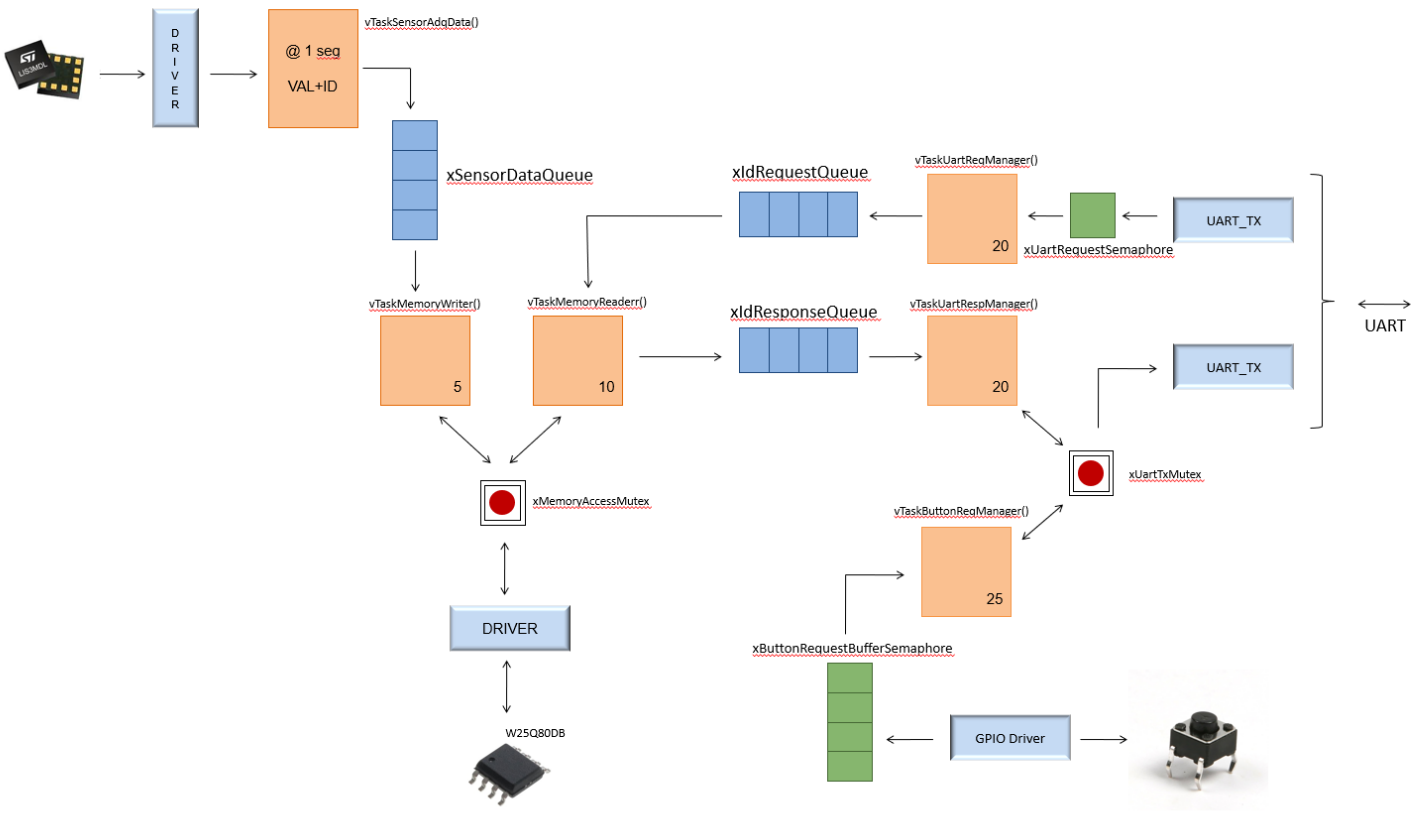


The architecture is composed of 5 layers to encapsulate the drivers and provide a certain level of hardware independence. In addition, this approach allows porting the application code to other platforms and to perform tests over each part of the code independently.

FreeRTOS solution diagram

The next diagram presents the complete firmware architecture integrated by:

1- Five tasks
2- Three queues
3- Two mutexs
4- Two semaphores
5- The flash memory and magnetic sensor drivers
6- The GPIO and UART drivers



There are some design considerations:

1- Each hardware shared resource (flash memory and UART Tx channel) has its own mutex.
2- Data producing tasks have their own output buffer to handle multiple events without loss of information. For example, the application code samples the LIS3MDL sensor data @1Hz, this is a very low frequency process, so the output data queue has only 10 positions to allow the consuming task to delay up to 10 seconds in processing the data.
3- The task priorities were defined according to the expected behavior of the system according to the user requirements.