Evaluation of Miniaturized Inertial-Measurement-Units

Practical Project Computational Neuro Engineering
Final Report

Klaus Wackerbarth
Thomas Wildgruber
Xinhua Liao

Lehrstuhl für
STEUERUNGS- UND REGELUNGSTECHNIK
Technische Universität München
Univ.-Prof. Dr.-Ing../Univ. Tokio Martin Buss
Prof. Dr. sc.nat. Jörg Conradt

Betreuer/-in:	Dipl.-Inf. Nicolai Waniek
Beginn:	23.10.2013
Abgabe:	14.01.2014

Contents

1 Introduction

1.1 About the Project

Autonomous aerial robots typically rely on integrated inertial measurement units (IMU) for estimation of the vehicle's attitude (roll, pitch and yaw). Recently new IMU modules have become available that integrate 9 degrees of freedom measurement systems (3x Accelerometer, 3x Gyroscope and 3x Compass) with microcontrollers, to directly compute attitude (given e.g. by Euler Angles). This Project deals with the communication between the LPC1769 and the IMU9250. We should get the data from the MPU-9250 with the help of the LPC1769. The Project is then divided into mainly two parts, the software development and the hardware development. For each part it has its own goal and tasks, which will be talked about later.

1.2 LPC1769

The LPC1769 is an ARM Cortex-M3 based microcontroller for embedded applications featuring a high level of integration and low power consumption. The ARM Cortex-M3 is a next generation core that offers system enhancements such as enhanced debug features and a higher level of support block integration.
The LPC1769 operates at CPU frequencies of up to 120 MHz. The LPC1769 includes up to 512 kB of flash memory, up to 64 kB of data memory, Ethernet MAC, USB Device/Host/OTG interface, 8-channel general purpose DMA controller, 4 UARTs, 2 CAN channels, 2 SSP controllers, SPI interface, 3 I2C-bus interfaces, 2-input plus 2-output I2S-bus interface, 8-channel 12-bit ADC, 10-bit DAC, motor control PWM, Quadrature Encoder inter face, four general purpose timers, 6-output general purpose PWM, ultra-low power Real-Time Clock (RTC) with separate battery supply, and up to 70 general purpose I/O pins.
(see: [1] NXP Semiconductors).

1.3 MPU-9250

The MPU-9250 is the world’s smallest 9-axis Motion Tracking device for smartphones, tablets, wearable sensors, and other. It is only 3x3x1mm. The MPU-9250 includes a 3-axis accelerometer, a 3 axis-gyroscope and a 3-axis magnetometer.
(see: [2] InvenSense)

2 Goal of this Project

2.1 Original tasks

In this project our goal was to implement hard- and software for small test systems to obtain values from available IMUs, program a microcontroller and/or PC software to read and store values, and compare the performance of multiple such miniaturized IMU sensors during stand-alone operation. So the tasks were:

- Acquire or build test systems for multiple IMUs; including MPU6050, MPU9150 and MPU9250
- Develop software for recording of data
- Evaluate and compare the systems' performances against ground-truth-data on a bench setup
- Time permitting record data from in-flight and compare against ground truth from overhead tracking

2.2 Refined tasks

The main task in our project was to compare the performance of the new MPU9250 with the older version MPU9150. Our project started in October. A big problem with our project was that the MPU9250 chip was delivered not before December. Another problem was that there are no documentations about the new MPU9250 chip so it was not possible to write software for the new board because we had no register map. The only thing we got after a while was a sheet with the pin outputs of the MPU9250. Because of these problems our tasks changed.

- create a platinum with the LPC17969 and the MPU9250
- develop a software for a demo setup with the LPC-Expresso and the MPU9159
- deliver the values of the sensor board to the PC

3 Our Progress

To fulfill our tasks in this project it was necessary to divide them into two major groups. One part was to develop new platinum with the LPC-Microprocessor and the MPU-Unit on it. The other part was to develop software for the LPC-Microprocessor to read the sensor data from the IMU and send them to a PC via UART connection. The problem with these two things is that they are directly connected to each other so you need a finished platinum to get the software running but the other thing is you need a running software to test the new created platinum.
For this reason we started to develop the hardware and software separately.

3.1 Hardware

3.1.1 Schematic Design

The Schematic Design should include most importantly two parts: The MPU9250 and the LPC1769. Here we had to consider about how to integrate the two parts in one circuit board. We use the important data from the Datasheet of each component as an instruction to build the board.

3.1.1.1 Important Data from Datasheet of LPC1769

The important data of the device LPC1769 are the block diagram and the pin outputs. They determined how the device is connected with other devices.
The block diagram for the LPC1769 is shown in figure 3.1.
In this diagram we found out the name for each block easily. It then provided a clear view of the structure of the device. This makes it more useable when we tried to use some function of this device. Otherwise we had to use additional hardware pieces for the design. The parts we needed for our platinum were I2C bus, UART connections, JTAG connections, GPIOs for LEDs and external interrupt. Based on the above selections, it was easy to take a look at each part in the pin outputs part of the device. There we found the corresponding pin outputs for each function.

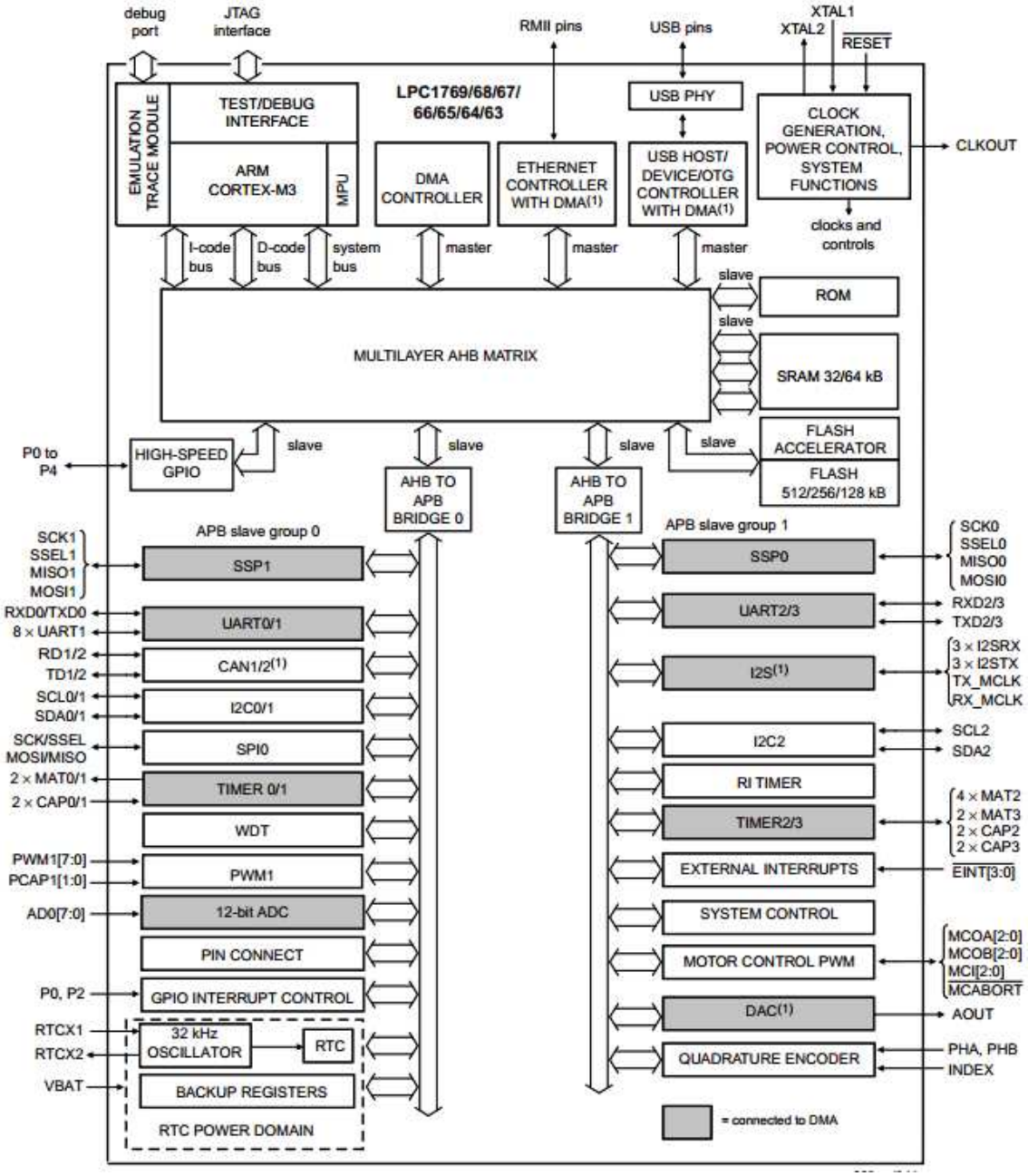

Figure 3.1: Block diagram for the LPC1769

3.1.1.2 Important Data from Datasheet of MPU9250

Similar to the LPC1769 device, the block diagram of the MPU9250 should also be included in the hardware design. The block diagram of the MPU9250 is shown in figure 3.2.

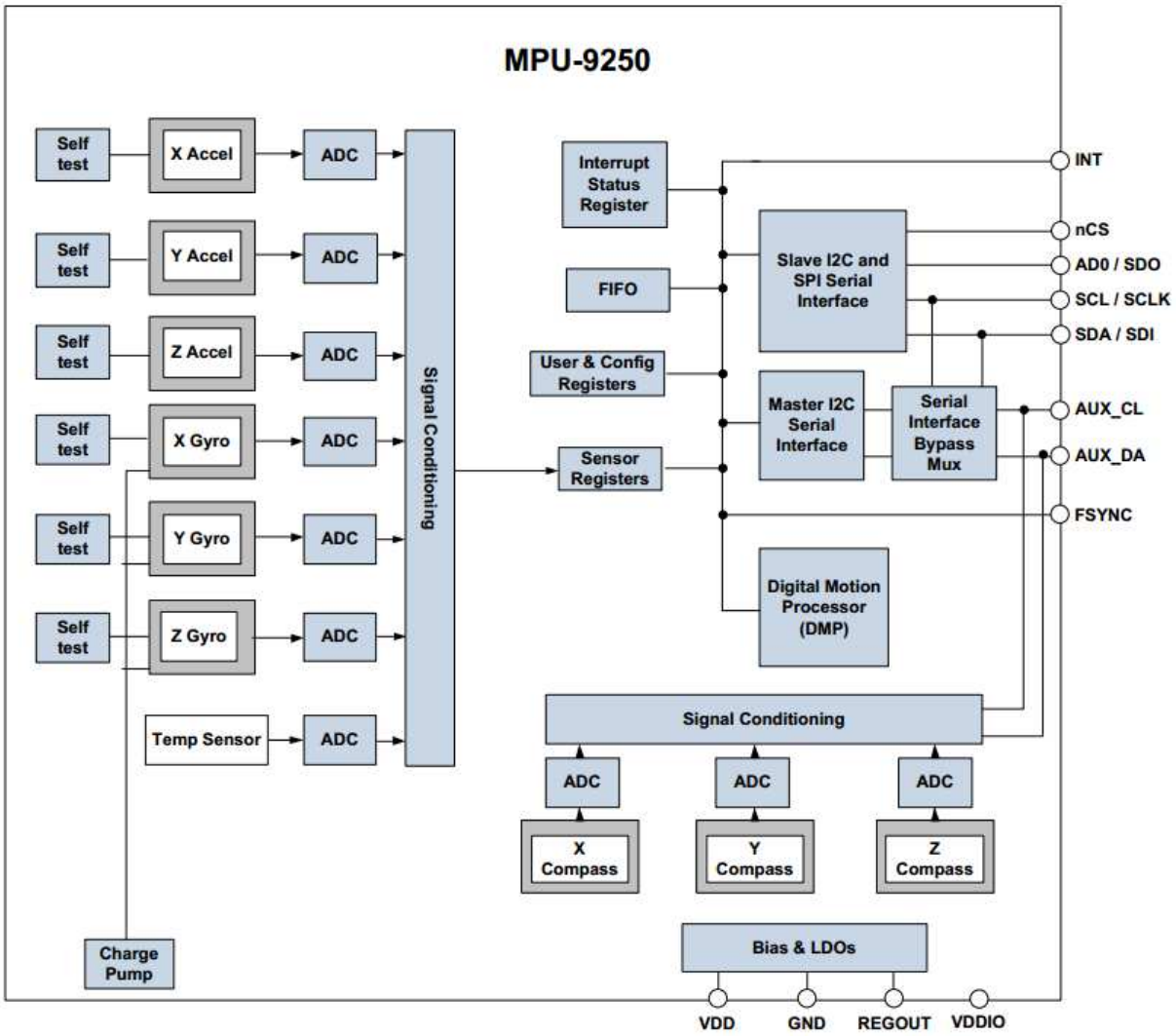

Figure 3.2: Block diagram for the MPU9250

It is also important to use the pin outputs of the MPU and the typical operating circuit information of this device by designing.
The pin outputs of the MPU9250 are shown in figure 3.3.
The typical operating circuit of this device is shown in figure 3.4.

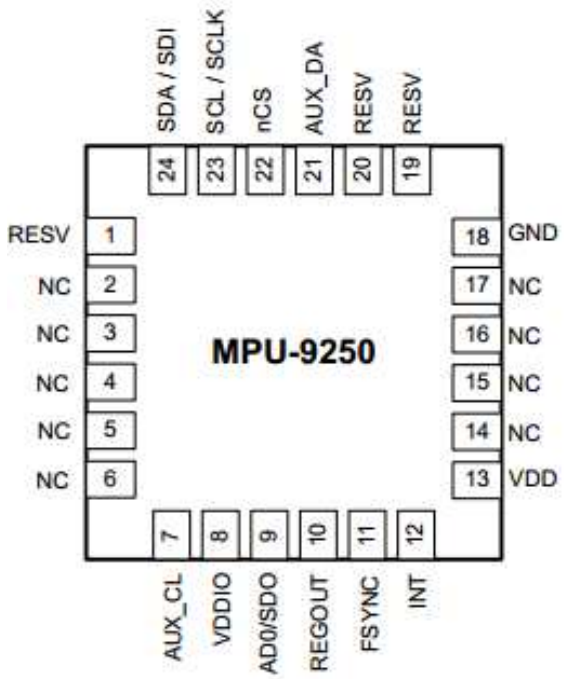

Figure 3.3: The pin outputs for the MPU9250

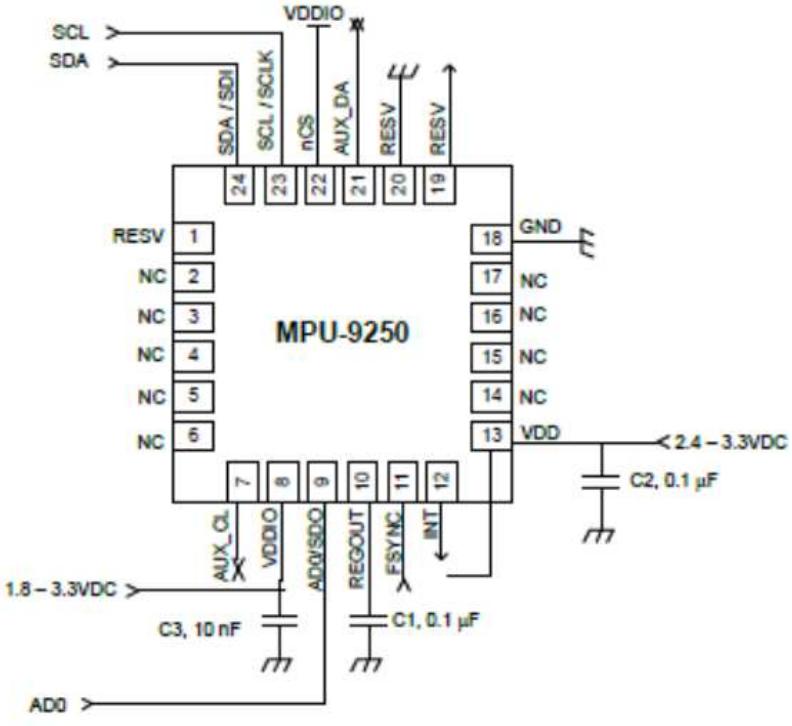

Figure 3.4: The typical operating circuit of this device

3.1.1.3 Connection between LPC1769 and MPU9250

After we collected all necessary data for the design, we started with the actual design by connecting the two parts. The I2C bus is considered as the main data transfer channel for the communication. It should be also considered that the LPC1769 device can catch the begin signal of the MPU9250. So there should also be an interrupt line between the two devices. So when the MPU is ready to send data, it sends an interrupt signal to the LPC to inform it about the message.

3.1.1.4 Peripheral equipment on the schematic

By peripheral equipment it is required that the device should work with a power supply, which is suitable for both devices. In this sense, a more detailed look into the datasheet should be made. Generally the working condition for the LPC and MPU are 3.3V. But it should be also considered that the LPC and MPU device will need a hardware interface for programming. So the power supply for these interfaces should also be considered.
For the LPC device, it uses a UART interface to get the power needed. The power supply for this UART interface is 5.0V. Other than that, the MPU device also needs the power supply from 3.3V.
It should also be taken into consideration, that the LPC should be programmable. In this case, a JTAG interface for programming the software into the hardware is also needed.
Other than these parts, a reset function for the LPC devices is also favorable. It can reset the microprocessor when something goes wrong. This should be not hard to make, because the LPC already provide a pin for the reset function.
Apart from all the above, some technical details should also be paid attention to. For example, a buffer should be set to buffer the power from the pins of the microchips. It should also be added a function for the LPC device by using the LEDs to instruct that the device is functioning.
After careful consideration of all the functions and combining the data needed the schematic design should not be hard to draw with the help of a PCB design tool, called EAGLE.
There's a free version for this design tool, which contains only two layers design possibility, but it is already good enough for this work.
The overall design of the schematic is shown in figure 3.5.
By checking the violation of the design rules, then the schematic design for this Project should be done.

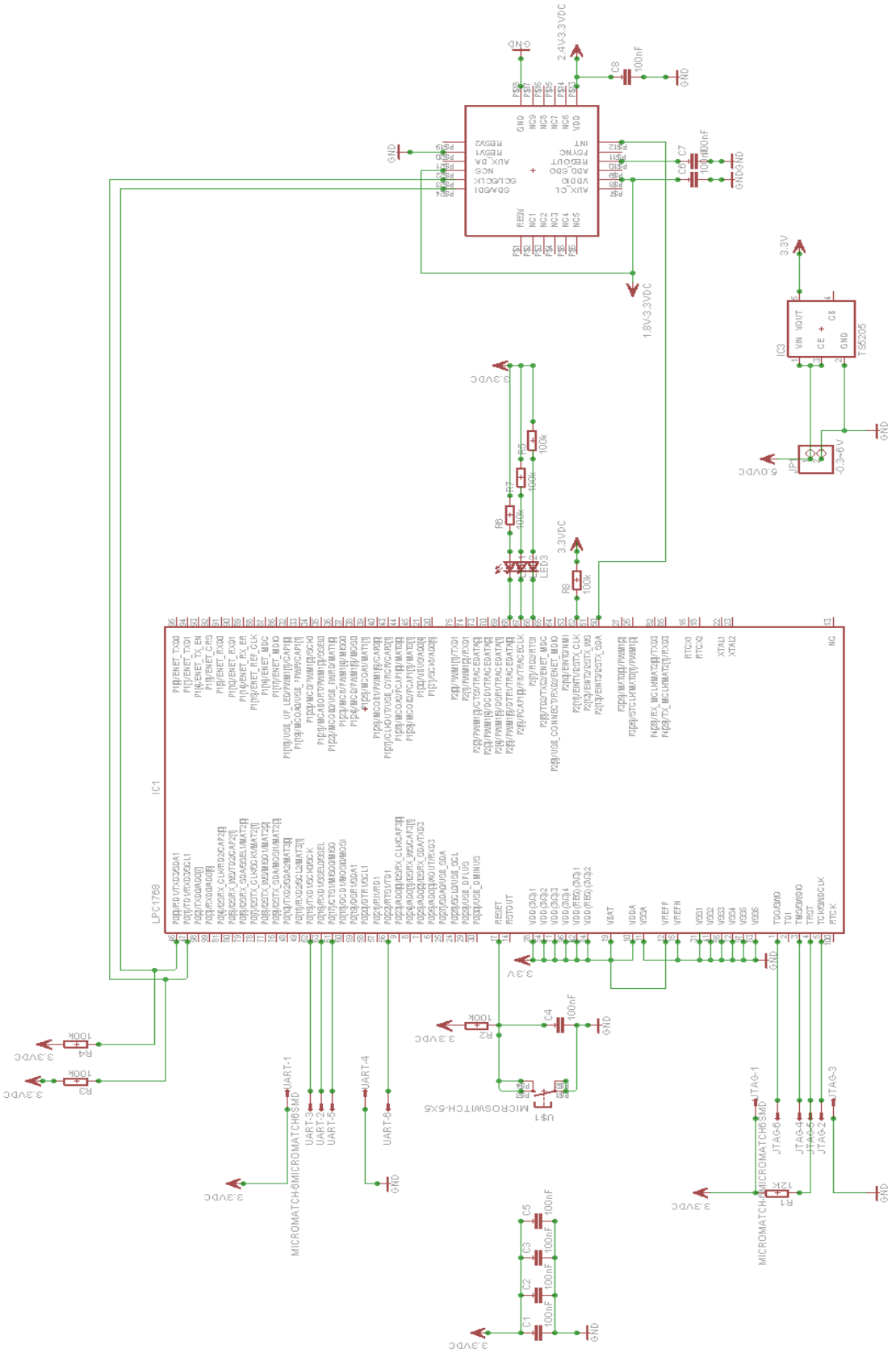

Figure 3.4: Schematic design overview

3.1.2 Board Design

3.1.2.1 Principle of the Board Design

The principle of the board design has the following rules:

- Firstly the route wire in each layer should not cross each other; otherwise we should make a hole through the layer to connect the wire in another layer.
- Secondly the board should be made as small as possible, and the wires should be as short as possible so that the cost of production can be low.
- Thirdly the angle of the wires should be in 45 degree, so that it is easier to make in manufacture.

There is also an additional principle that should be kept, which is to try not to use an auto router to connect the wires, because it sometimes fails and the design will then be destructed. In this sense, by connecting the wires, the basic principle is to do it by hand.
Finally, the holes though the layers should be as less as possible so that it is more safety by producing the board, which reduces the risks by breaking the board.

3.1.2.2 Error Correction

When we are connecting the wires by following the above principles, it should not be hard to finish the board design. Normally we should only follow the connecting instruction lines on the board to connect each component.
But it is not easy work, because we can't do it well on first time. It always left some room for perfecting the design, to make it smaller and better.
Error Correction is one such part. By this round, the errors in the rule checking should all be corrected and be avoided. This takes a lot of time to be finished. But on the other hand, the board is getting smaller and looks better.
Before handling the board to production, some of the basic principles listed above should be checked again. And the serious errors should also be checked. For example, the short cuts between the wires or an opening of the wire and so on.
After carefully avoiding all the possible errors and following the design of the schematic, the board design is then not hard to be done. It is shown in figure 3.6.

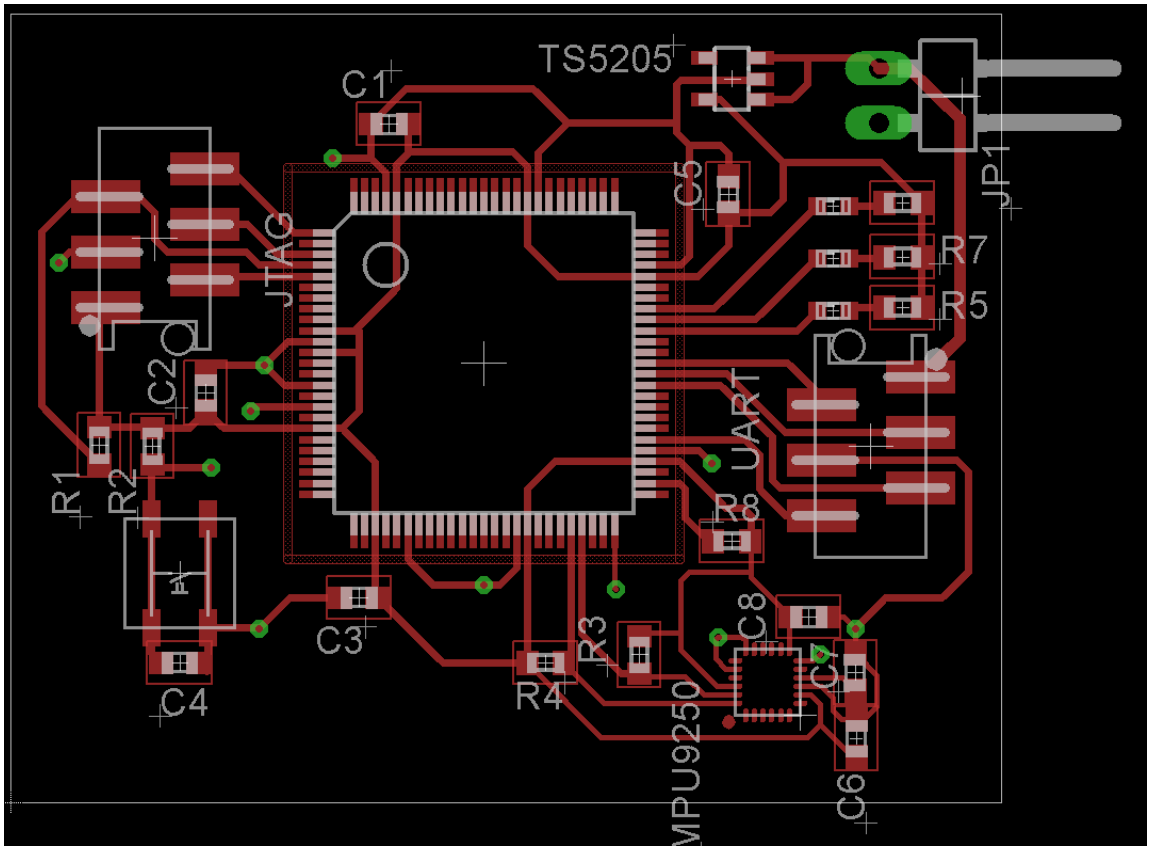

Figure 3.6: Outlook of the board design

3.1.3 Board production and testing

3.1.3.1 Precautions by board Production

The Board Production part is also full of principles, which should also be carefully kept.

- Firstly, the drills for the production machines should be placed probably and tested before usage. Because when used improperly, the drills will break, which costs money and is dangerous for the people on the scene.
- Secondly, by soldering the pads, the short cut of the wires that lye beside should also be avoided. It is easy to cross the wires when doing soldering by hand. Especially by soldering the pins of the microchips. Since they are really small, it would be easy to cross one pin with another through soldering.
- Thirdly, the epoxy on the board which is produced by the production process is harmful for the body, so to wash the hand after the production is necessary for health reason.

The final product for the board for this project is shown in figure 3.7.

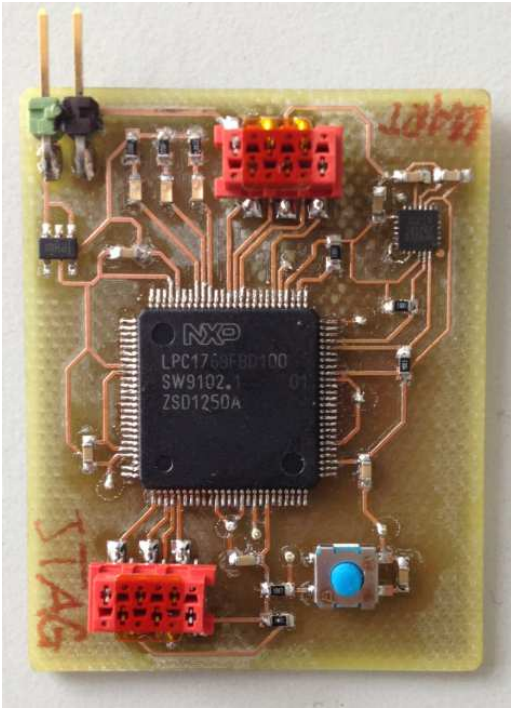

Figure 3.7: Product of this project

3.1.3.2 Hardware Testing

After production the board should be tested then. The basic test for the board consists of two parts. Tests of the hard- and software.
It is quite simple by doing the hardware testing. By connecting it to the actual power, it is easy to see if there is some short cut exists, because the board will smoke for this reason.
Otherwise it is more important to combine the software test with the hardware. It is the only way to find out whether the board is actually working or not.

3.1.3.3 Problems with the Hardware

This hardware design is basically finished. However it still leaves some problems. The board now passed the basic software test and the hardware test. But the study of the new MPU has just begun.
This is because the MPU9250 is very new on the market. The datasheet is not complete by finishing the hardware design.
We are trying to compare the difference between the MPU9150 and the newest one. And then try to flash the images into the MPU9250. But it takes lots of experiments by trying this. Because the pin outputs for the MPU9150 are quite different from the newest MPU version. So the register structure may also be different. In this sense, the software testing for the difference of the two type of MPU still takes time to complete.

3.2 Software

3.2.1 Developing Environment

As described in the introduction of chapter three, we needed an independent developing hardware platform, to work parallel on the PCB-Sensorboard design and the software, controlling the micro- controller in- and outputs. Therefor we used the LPCXpresso LPC1769 Development-Board with the LPCXpresso IDE and the MPU-9150 9-Axis Evaluation Board. Because this configuration was the same like in our sensor board hardware configuration, we were able to develop all software functionality completely independent to the PCB-Board layer development. The only problem in our PCB-Board configuration was that we used the MPU-9250 chip. This IMU was pretty new and not already available on the market, so we had no complete documentation about it. But to make progress, we act on the assumption that both IMU specifications are about the same. In figure 3.8 you can see the complete hardware setup for the software development.

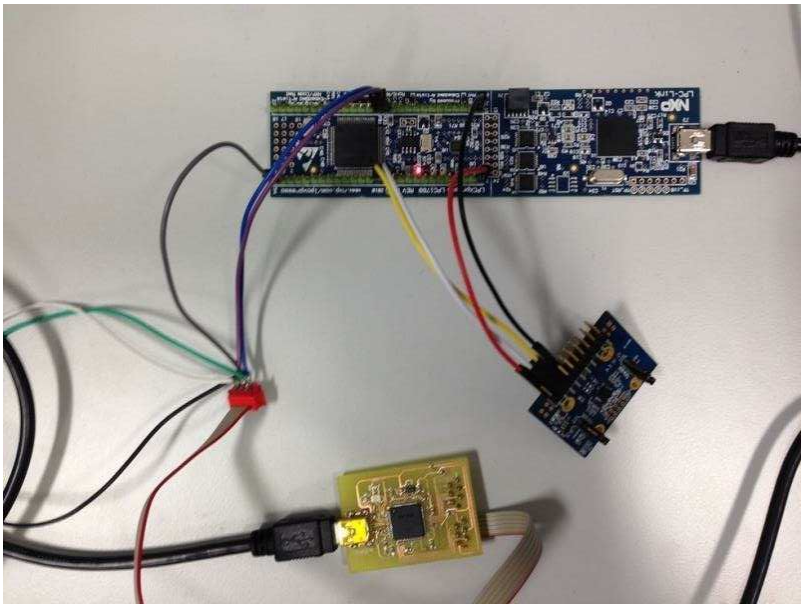

Figure 3.8: Hardware setup for software development

To flash and program the LPC1769 microcontroller, we used the appending LPCXpresso IDE. This IDE is a Eclipse based, highly integrated software development environment for NXP's LPC processors and as many enhancements to simplify development with NXP LPC microcontrollers. LPCXpresso also features the industry-standard GNU tool chain, with a choice of a proprietary optimized C libraries or the standard newlib-library and is on this account most suitable for our requirements.
For the Hardware Abstraction Layer we hark back to the standard CMSIS-Library for ARM- Cortex processors, to get a simple programmable basis. One problem at the beginning of the project was to find a workable product version of the CMSIS-Library,

which is compatible to the LPCXpresso libraries. This was not as easy as predicted, because the libraries were particularly defected and worked only fractional with the LPC 1769 microcontroller. At least we decided us for the version 2.0 and after some bug research we made this library running, but it costs us some not scheduled time.

3.2.2 Organisation of the Microcontroller Firmware

Figure 3.9 shows the two main hardware interfaces, UART and I2C, which allowed the communication between the microcontroller and the MPU and the communication to an external PC. Both interfaces have been managed by the firmware of the LPC1769 microcontroller. For this specification you can divide the tasks of the firmware altogether into four separate main parts:

- the initialization and specification of the used GPIO's
- the communication and initialization of the MPU over the I2C bus protocol
- the interaction and data exchange to an external device with the UART interface
- the internal data handling and processing of the firmware

The first part handles and initializes all the different modules, GPIO's and communication layers of our microcontroller. This section has been very important for our project, because the specifications of the developing-platform are a little bit different to the specifications of the final version on the self-made PCB-Sensorboard. So we had to be able to change all specifications easily, without writing a complete new source code. You can find the general specification map in figure 3.5 of chapter 3.1.

Figure 3.9: Hardware interfaces

To create a communication interface between the LPC and the MPU, we had to enable the I2C bus, which is the second and at the same time the biggest part of our firmware development. Therefor LPCXpresso provides some drivers libraries, which conduced us as a basic layer. Constitutive to them we build up our own read and write functions, to send and receive data over the I2C bus. The I2C driver therefor can only send and receive packages of eight bit length. For this reason, all data has to be parsed or transferred, from or into the right generic data type. To afford this we implemented some functionality, which ensure, that this happens in the right way.

After power up the system, the MPU first-of-all has to be initialized. To afford this, we had to flash some specific registers, to get the functionality of the MPU we want to have. The specification possibilities can be gleaned at the product documentation sheets. Two examples cohesive to this were, to activate the self-test-mode and to deactivate the sleep mode of the MPU. Other functions can be added in the initial part of the source code easily afterward.
After this initialization steps, we was able to read the sensor values from the MPU. Therefor we call a data-polling function for all sensor coordinates during every cpu-clock cycle. Of course this method is not the best in performance but for our purpose sufficient. For later more complex enhancement we attached an interrupt reserve wire, which can be programmed to recognize, if new data on the MPU is available.

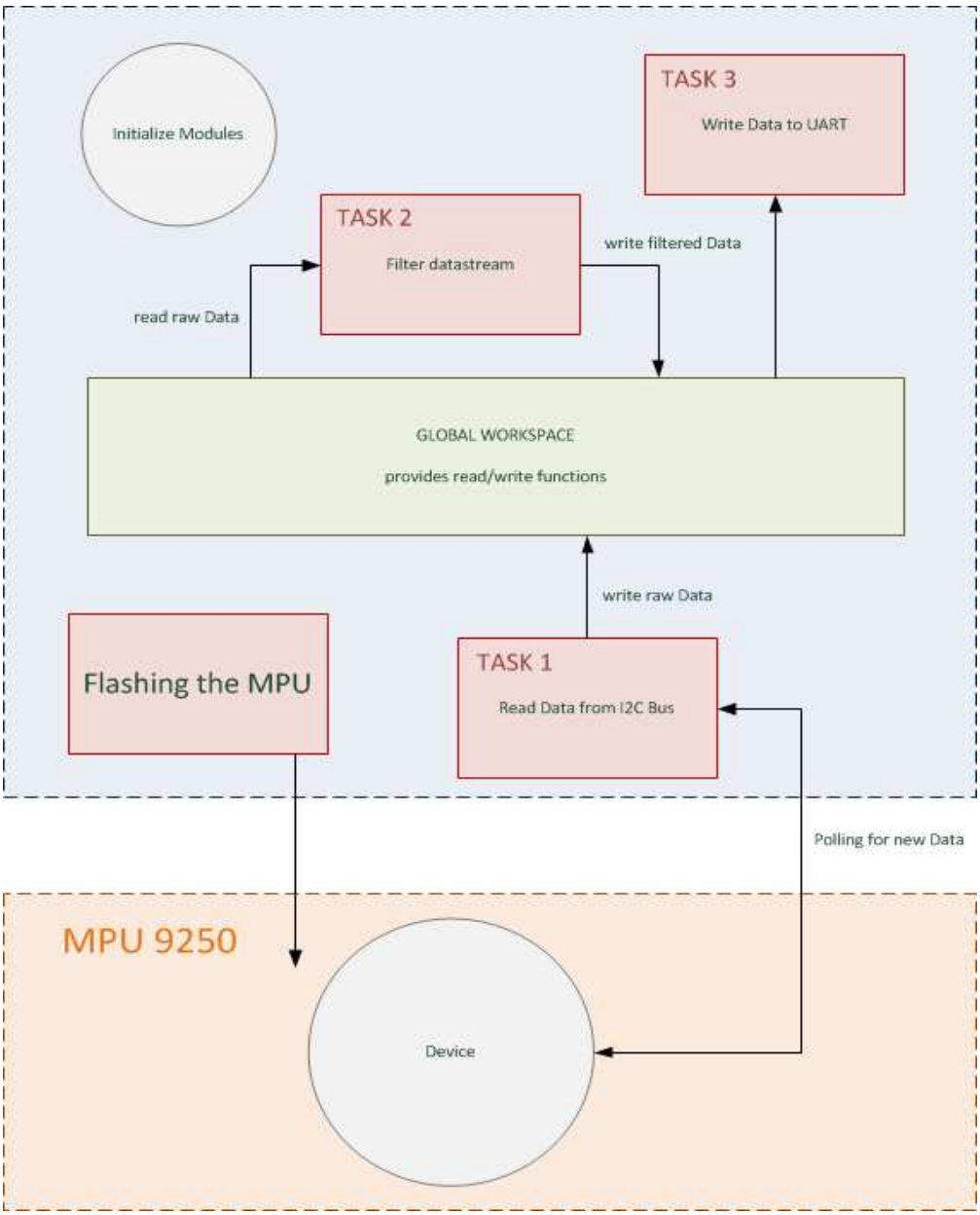

Figure 1.10: Organization of the firmaware

The third task handles the communication between the LPC microcontroller and any external devices, which needs the sensor data. In order to make the sensor data visible and recognizable, we used the UART interface to transfer the sensor values to an external PC. Because UART is an asynchronous receiver/transmitter piece of computer hardware, data is transferred connectionless between two devices. One problem in this case was, that we had to pack the data into a parse able package, so that an external visualization program known which value is associated with which sensor data. Our solution was, to revert to the JSON string format. In this case, we had to add a routine in our source code, which converts the sensor data into a first defined strict string format. As JSON is a very commonly used format, in many programming languages does exist parser to dissolve this string construction. For this reason the JSON format was well suited to our requirements.
The last part contains the core and global design of our software, as you can see in figure 3.10. Our requirements included, that we wanted to have a standard and easy to use interface, to exchange data between every function and module we wanted to integrate and want to integrate in future to our firmware. This organization cost us some developing time and a little bit of hardware performance, but on the other hand side we got many advantages of this design. With this organization several different persons were able to develop different functions independently, took hold on this standard interface. Another convenience is, that later plugins can be integrated easily in the whole software scheme, without changing anything in the core of our source code. With this software design our requirements were realized in a very simple and also efficient way.

3.2.3 Sensor data visualization at the PC

For the visualization and the data evaluation we had been looking for an easy way, to represent the sensor data in real-time in a graphical surface. Matlab/Simulink provides therefor user-friendly appealing methods, to read the serial UART interface and to process the sensor data. Therefor the Instrumental-Simulink toolbox has integrated all functionality we needed.
The only problem we got with this implementation was, that Simulink do not support char inputs from a serial device. So our solution with the JSON sensor data stream is not be supplied by this application. Therefor we changed our source code in the firmware and now we sent the sensor data in a predefined order after the firmware got an acknowledge request from Simulink. This solution of course is not globally adaptive, but in our specific case inevitable.
Figure 3.11 shows the main functional blocks we used in our Simulink application. As you can see the signals and scope inputs for visualization can be easily changed, so everybody can configure his personal signal output.

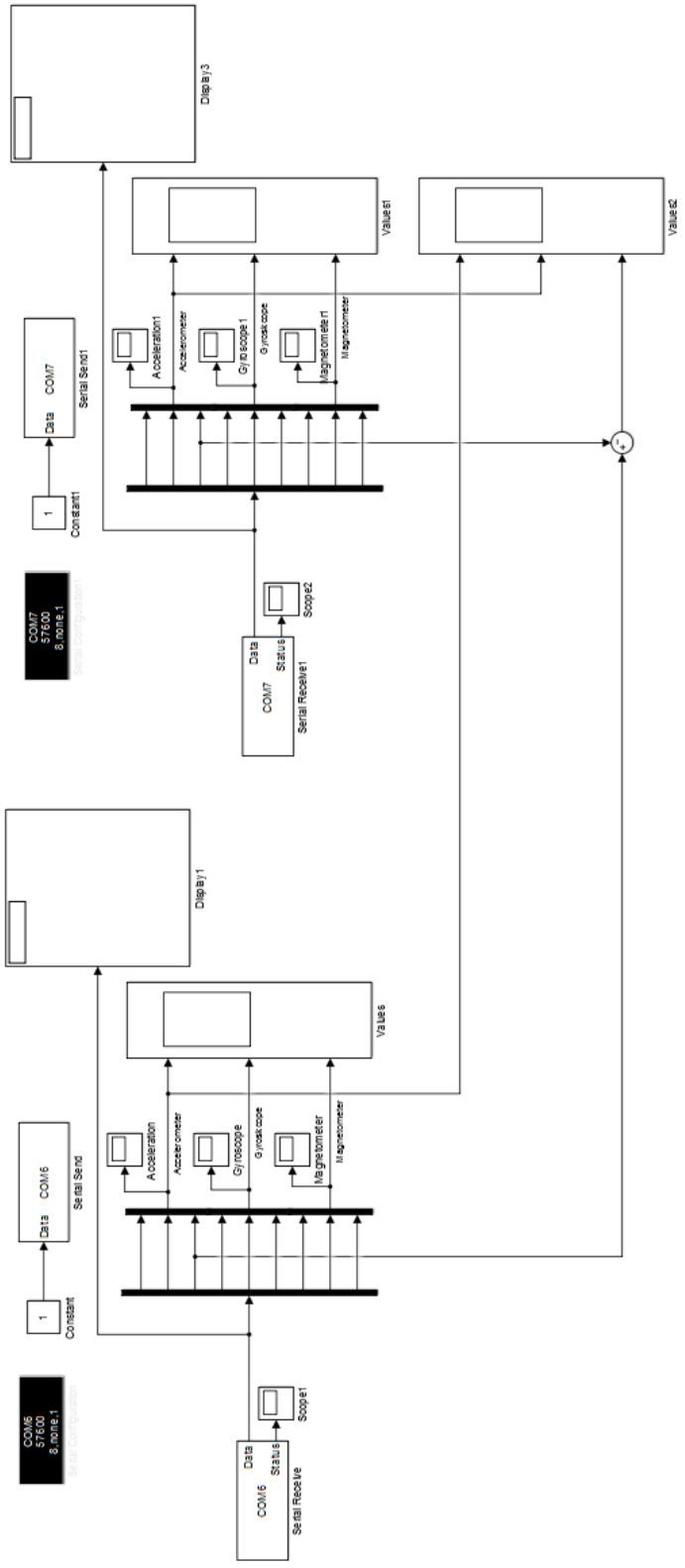

Figure 3.11: Simulink model for value visualization

4 Summary and look-out

4.1 Results

In our project we developed a schematic for a platinum consisting of a LPC-1769 microprocessor and the new MPU-9250 sensor board. We used this schematic to actually build this platinum.
For the software part we wrote software for the demo setup described in chapter 3.2. The software is running and able to read all sensor data of the MPU9150. The data can be sent to a PC via UART connection. On the PC we were able to visualize the data on Simulink. Figure 4.1 shows an example of the recorded acceleration values. The y-axis is scaled in mg.

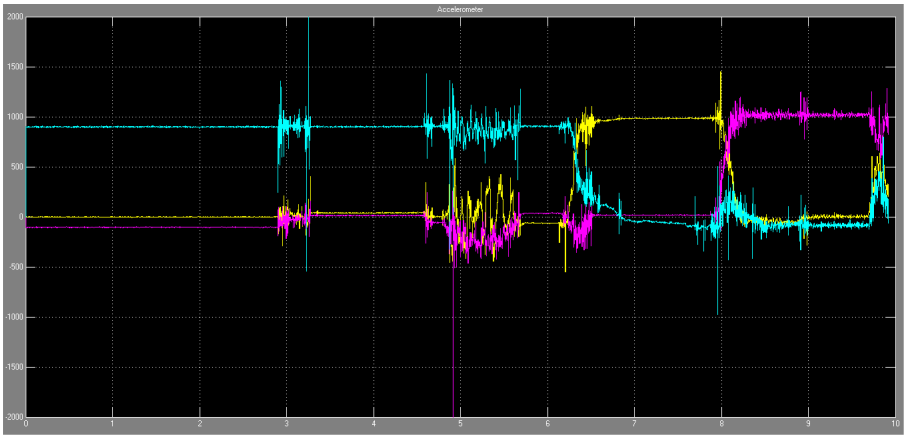

Figure 4.1: Raw-Values of the accelerometer

The datasheet of the new MPU9250 is not published yet so we weren't able to compare the registers of the old and the new sensor board and verify if the code runs on the new created platinum. Therefore we just tried it and it worked. So we were able to get all sensor data from our new created platinum.
Because of our problems with the library, the late delivery and missing datasheets of the sensor board we lost very much time so we weren't able to compare the two sensor boards an validate the data.

4.2 Look-out

Future tasks of this project can be:

- Creating a platinum with the MPU9150
- Build an appliance in a calibrated environment to get sensor data from both boards and compare them
- Try to optimize the sensor data with filters (onboard; programmable)
- Sensor fusion of the 9 values

Appendix

Bibliography

[1] NXP Semiconductors, „LPC1769/68/67/66/65/64/63” LPC1769_68_67_66_65_64_63 datasheet, Oct. 2013

[2] InvenSense, Inc., “MPU-9150 9-Axis Evaluation Board User Guide” AN-MPU-9150EVB-00 datasheet, Nov. 2011