UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

Design and implementation of a femto-satellite technology demonstrator

Victor Kravchenko

SUPERVISED BY

Joshua Tristancho

Universitat Politècnica de Catalunya
Master in Aerospace Science & Technology

June 30, 2011

Design and implementation of a femto-satellite technology demonstrator

BY

Victor Kravchenko

DIPLOMA THESIS FOR DEGREE

Master in Aerospace Science and Technology

AT

Universitat Politècnica de Catalunya

SUPERVISED BY:

Joshua Tristancho

Applied Physics department

ABSTRACT

The present work is aimed to demonstrate the development and implementation of a scalable femto-satellite technology demonstrator, experimentation and development platform in order to practically prove the possibility of meeting the N-Prize competition requirements as well as introducing some possible scientific applications of a femto-satellite as a satellite-on-a-board spacecraft. In order to achieve these goals the low-cost low-power self-sufficient platform with communication, data processing position determination and keeping abilities was designed and implemented. A set of subsystem qualification tests was designed and performed in order to obtain numerical results and calls for the future studies. The ability to meet the general space qualification and particular N-Prize competition-related requirements was studied and the overall conclusions were stated.

Keywords: Femto-satellite, satellite-on-a-board, PCBSat, COTS, MEMS, OSHW, Open Source, Mini-Launcher, N-Prize, WikiSat, EETAC, UPC

Acknowledgements

I'm grateful to my parents and my friend Elena for encouraging and supporting me during this work.

I'm very grateful to my tutor Joshua Tristancho for the opportunity to do this research as well as his help and inspiration in this work.

I'm very grateful to Roger Jove for introducing me to the WikiSat research group.

I want to thank members of the WikiSat research group who helped me with this work implementing the low-cost tools, providing the supplies and prototyping the boards, specially Joshua Tristancho, Roberto Rodríguez, Esteve Bardolet, Sonia Pérez and Jordi Gutiérrez.

I'm very grateful to people who helped and participated in the long range link test campaign, specially Raquel González, Enric Fernández, Roberto Rodríguez, Joan Martínez and Javier Pérez.

I want to thank the EETAC school for the laboratories and equipment they have provided to support this research.

Special thanks to Luis Izquierdo from the Universidad Nebrija and Ángel Esteban from the D47 company for their interest in this research and the opportunity to participate in the TURISMAP project funded by AVANZA grant, project code F-00300.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

A significant amount of breakthrough achievements in both science and technology in areas such as electronics, telecommunications and materials for the last decade have provoked a dramatically change in the modern understanding of a spacecraft as a technical system. A tendency to decrease mass, volume and power consumption of a spacecraft in space market areas such as Earth observation, remote sensing and scientific in general has led to the rapid development of a generation of micro, nano and pico satellites. Up to the date, there are dozens of CubeSats with the mass varying from 100 to a few kilograms orbiting the Earth and performing various scientific, governmental and commercial missions. The new generation of university CubeSats in a mass range from 1 to $10\,kg$ has shown ability to successfully perform Earth observation and other scientific missions in a budget below $100000\,USD$ [17, 6] that a few years ago were possible only with million USD budgets.

It has been theoretically proved as well as practically demonstrated that the modern Commercial Of The Shelf (COTS) electronic products are sufficient to meet the requirements of the space environment [31, 32, 17] and enable a design and implementation of the next-generation spacecrafts for scientific purposes, femto-satellites with a mass below $100\,g$ though allowing significant decrease in production cost and design time [6]. A feasibility study [4] was done to prove a possibility of creating a fully functional satellite-on-a-chip for the Earth observation purposes with a mass of $1\,g$. A low-cost satellite-on-a-board was proposed and the prototype was demonstrated [6] with a cost below $300\,USD$ though has not flown to the date. A few distributed sensor array and cloud-missions for pico and femto satellites were proposed and proved to be feasible [6, 5]. A major interest of both governmental and industrial sectors to the emerging new technologies in the satellite market has been demonstrated by a growing amount of research groups involved in the studies with the financial support of aforementioned institutions. All the facts stated above show the tendency of shift in advance driving force role from governmental and industrial institutions to university research groups.

It is also very important to note that shifting the payload mass to the range below $100\,g$ opens completely new space of launch opportunities based on the re-utilization of old and well known low cost (in comparison to the conventional launchers) chemical rocket propulsion technologies for low mass payloads (also known as missiles) and stratospheric balloons as the launchpad. The feasibility study of the low-cost launch technology based on the aforementioned studies was made [8] and demonstrated the high success probability.

Public interest to the ongoing progress was demonstrated by a challenge given by the N-Prize competition [10]: to put into the orbit a satellite with a mass between 9.99 and 19.99 grams and to prove its completion of at least 9 orbits assuming that no part of any orbit can be lower than $99.99\,km$. It is also required that total cost of the launcher and satellite must be within a budget of $999.99\,GBP$. As the response to the call of the N-Prize competition, a research work has been performed and a feasibility study was done [28] showing the possibility to achieve the goals of the competition.

The present work is aimed to demonstrate the development and implementation of a scalable femto-satellite technology demonstrator, experimentation and development platform

in order to practically prove the possibility of meeting the N-Prize competition requirements as well as introducing some possible scientific applications of a femto-satellite as a satellite-on-a-board spacecraft. In order to achieve these goals the low-cost low-power self-sufficient platform with communication, data processing position determination and keeping abilities was designed and implemented using affordable technologies and open source or free-ware tools along with the community knowledge and technological experience.

The Chapter 1 will synthesize a set of requirements to the femto-satellite technology demonstrator capable of achieving the N-Prize goals. The Advanced Wireless Inertial Platform (AWIP) will be introduced as the affordable low-cost multi-functional development and experimentation platform capable of meeting the aforementioned requirements.

The technical design and implementation of the AWIP will be described in the Chapter 2. The concurrency-based component selection and functional analysis will be presented as the design optimization driving force. A set of free-ware, Open Source and Open Source Hardware (OSHW) tools used in the platform development will be introduced. The hardware implementation technique will be demonstrated including low-cost and professional implementation approaches. The prototyping budget will be shown and justified. The design and technology related difficulties and mistakes observed will be discussed. Flash programming methods including In-System Programming (ISP) and High Voltage Parallel Programming (HVPP) will be discussed along with the rapid firmware prototyping methods and low-cost OSHW tools.

The 3rd Chapter will demonstrate the test and validation methods developed in order to obtain experimental data proving the device operability. The observations of the qualification tests performed will be discussed in this chapter as well. The hardware-in-the-loop development and simulation concept will be demonstrated with the platform and the Moon2.0 space mission simulation software [30].

The Chapter 4 will provide a set of application scenarios for the AWIP as a femto-satellite technology demonstration and experimentation platform as well as some other possible spin-off applications. A set of very basic implementation guidelines for each scenario will be given.

CHAPTER 1. SYSTEM DEFINITION

The scope of this chapter is to define the femto-satellite technology demonstrator following the requirements analysis guidelines for the system engineering in general [20] as well as spacecraft-specific [12, 23] resulting in the product design specification (PDS) [27] to be followed during the design and implementation processes. It is also worth mentioning that the preliminary definition of the femto-satellite technology demonstrator is a satellite-on-a-board or a PCBSat system and, thus, we have to assume the engineering design process as a design of embedded electronic system with all the corresponding requirements and constraints [14]. Another set of constraints and requirement is presented by the N-Prize competition rules and the WikiSat team internal agreements and regulations [29, 28] and will be introduced in the requirements analysis and PDS synthesis as well.

1.1 Requirements synthesis

To achieve the goal of the N-Prize competition the preliminary requirements analysis for the femto-satellite was made [28]. However, it is necessary to make a revision of the requirements analysis in order to take into account technology-specific criteria related to implementation of the satellite-on-a-board in general and technology demonstration, experimentation and development platform in particular. The target platform is aimed to represent the station-keeping functionality.

There exist different requirements and product design strategies, and among them there are at least 3 strategies applicable to the subject of the current study: general product engineering, spacecraft and embedded electronic system design strategies with corresponding requirements analysis methods. Practically, it is possible to combine some parts of different design strategies. For instance, satellite-on-a-board may be viewed as a device in general as well as spacecraft and printed circuit board providing additional flexibility to the definition process. It is important to note that the object of development is a technology demonstrator and developer platform all-in-one, thus some femto-satellite requirements are to be reasonably suppressed. A requirements synthesis strategy based on reasonable combination of the aforementioned ones was used.

One important constraint source for the requirements synthesis to mention is introduced by the WikiSat team rules list, stated in [28, 29]. Among them there are three to be considered as the most affecting the requirements design:

Geographical Availability: There must be at least one alternative for each component selected and there must be at least 1 company in the European community able to provide this component.

Zero Redundancy: Redundant hardware is not allowed in the design, however some redundancy may be presented in the software if necessary and possible.

Simplicity: The design should be Kept Simple and Safe (KISS rule).

Taking into account the requirements stated in [28] for the femto-satellite as well as the available technical and technological solutions and a set of geographical and political constraints, a list of requirements to the satellite-on-a-board technology demonstrator for the N-Prize mission is summarized and presented below:

1.1.1 High Level Requirements

HL0: The device should be able to provide bi-directional link capability on the $2.4\,GHz$ frequency with at least $1200\,bps$ bandwidth in order to transmit the station-keeping data including at least the determined position and to receive the possible updates about the target to point the antenna.

HL1: The device should be able to determine its own position with respect to the center of gravity of the Earth by inertial means only.

HL2: The device should be able to keep tracking the selected target pointing its antenna by the means of magnetic actuators.

HL3: The processing unit should be able to perform the necessary computations in order to process the sensor data, generate the control instructions for the pointing system and keep receiving or transmitting data fast enough to fulfill the N-Prize requirements.

1.1.2 Additional Requirements

The fact that the device is not only a PCBSat technology demonstrator, but also a software development and hardware experimentation kit applies additional set of requirements that is important to mention.

AR0: It should be possible to re-program the device using the on-board USB interface.

AR1: It should be possible to perform the debugging operations using the on-board USB interface.

AR2: The device may be powered using the on board USB interface

AR3: The device may be powered by any kind of battery source with the input voltage varying from 3.4 to $12\,V_{dc}$

AR4: The device should have at least 2 GPIO pins with analog or PWM output capabilities to interface with any possible simulator of the magnetic actuation system for the pointing control.

AR5: The device should have a low-level programming interface to enable firmware recovery in case of damage.

AR6: The device should have a power-up LED indicator providing the visual proof of life to the operator/developer.

1.1.3 Technological Constraints

It is important to mention a list of technological constraints based on some internal agreements of the WikiSat team. One of the noteworthy key statements is one defining the design rules and affecting the selection of the technological solution, which states that it should be possible to manufacture the device using amateur tools and low-cost means. The study of the low cost and amateur PCB manufacturing technologies was done and reported in [3]. The resulting list of technological constraints was defined and stated below:

TC0: It should be possible to mask the PCB for etching using toner-transfer masking technology.

TC1: It should be possible to etch the PCB using $FeCl_3$ heated water solution.

TC2: The vias should have internal diameter of at least $0.6mm$ to allow low-precision drilling technique.

TC3: The PCB track width can't be small than $0.2mm$ (forced by TC0).

TC4: The gap between tracks can not be smaller than $0.15mm$ (forced by TC0, TC1).

TC5: Low temperature soldering paste (with the temperature peak at $230^{\circ}C$) is allowed to be used for the in-house prototyping purposes only due to the limitations of consumer electric ovens used for amateur reflow-based SMD PCBA processes.

TC6: Any component package can be used except for array-based like BGA, LGA or PGA due to the limitations of amateur pick and place tools.

TC7: The smallest standard passive component package can be used is 0201 due to the technological limitations of the most industrial and amateur pick and place tools.

1.2 Product design specification synthesis

The constraint and requirements definition allows establishing the final product design specification. The specification model can be defined with respect to subsystems in a way similar to the Low Level requirements definition shown in [28] or can be performed component-wise taking into account that the PCB is a target device, thus it is possible to define the component list with high precision. It is also important to note that some PCBSat components may play a role of a complete subsystem. Following the component-based specification definition method will simplify the following component selection task decreasing though overall design time.

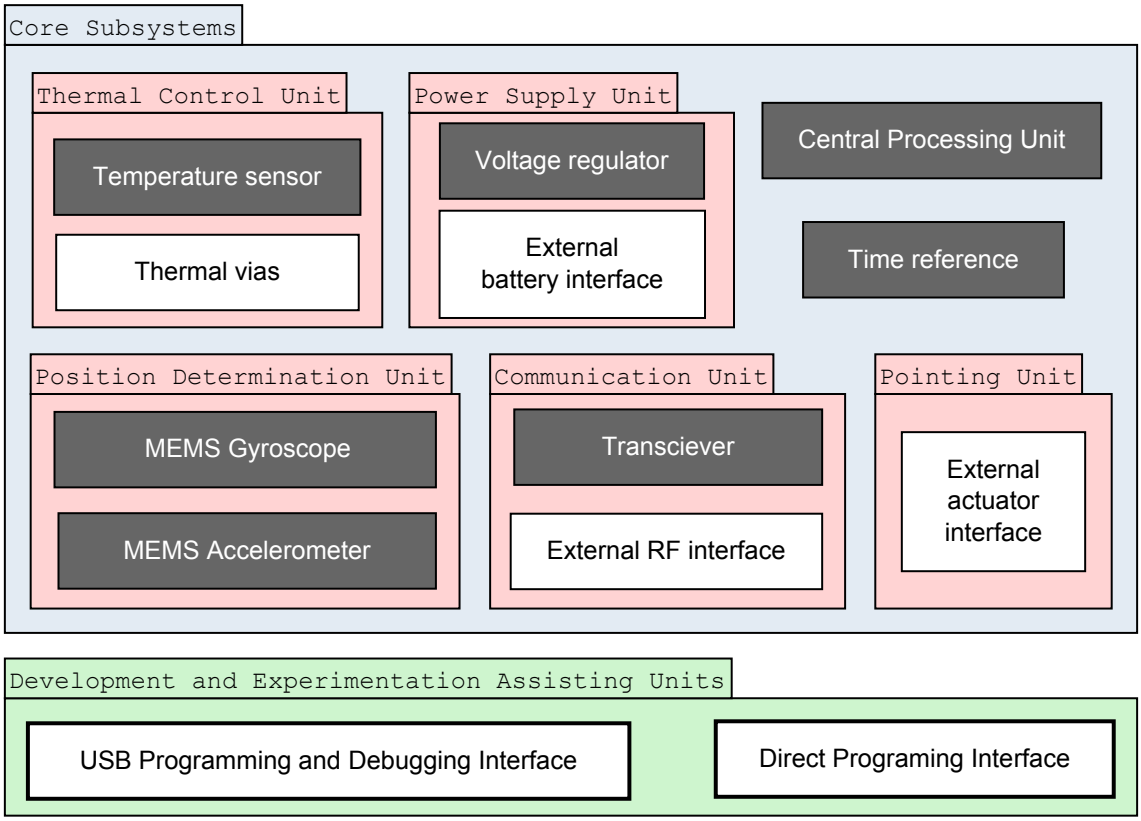


Figure 1.1: The definition of the subsystem-component relation model

As shown in the Figure 1.1, the component and subsystem relation is very strong as in the most cases a subsystem is defined by a single component (assuming that the component is an integrated circuit and its support elements). It is well seen that the component-based specification design in this case can be more efficient than a classical subsystem-based approach.

1.2.1 Central processing unit

The design specification for the desired central processing unit is one of the key issues of this work as it needs both hardware compatibility and computational performance requirements justification. It has been observed that for this kind of short-term missions an optimal solution for the role of the central processing unit is a micro-controller [17, 6, 4] as basically the accuracy, computational power, IO and memory are flexible enough to fulfill a wide range of possible requirements.

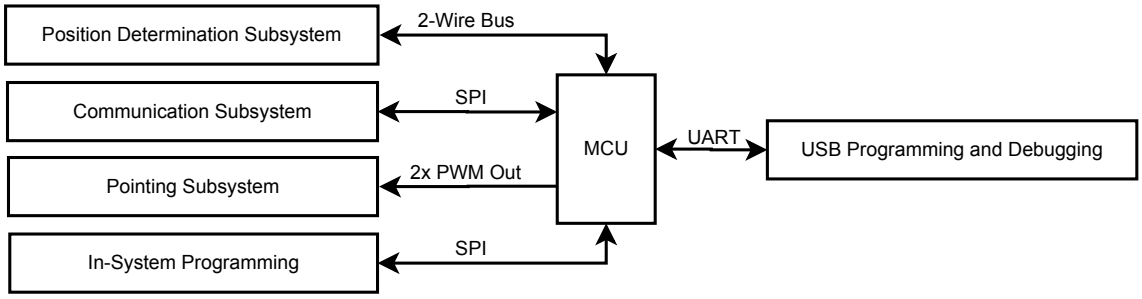


Figure 1.2: The definition of the MCU IO requirements

The hardware specification of the MCU IO ports is well defined by the needs of the subsystems as illustrated in the Figure 1.2. It is necessary to note that the dedicated SPI bus for the communication subsystem is preferable and may be software-emulated by hardware design if needed so. The preferable low or direct memory programming interface is a serial in-system (ISP) for the efficient pin-space utilization.

The definition of the performance specification for the MCU appears to be the most critical part as its optimal selection strongly affects overall system specification. There are basically two MCU families in the scope of this study - 8-bit low power RISC MCUs and 16/32 bit high performance MCUs. For the N-Prize satellite-on-a-board it is necessary to choose an MCU which is sufficient to fulfill the mission requirements. It should have the smallest form-factor, lowest power consumption and optimal performance. As the only data sources are 2-wire bus MEMS sensors with the size of a single value equal to 16 bits and transfer rate about 100 - $300 kHz$ it is possible to assume that management of these sensors and data acquisition can be performed with a low-power 8-bit MCU.

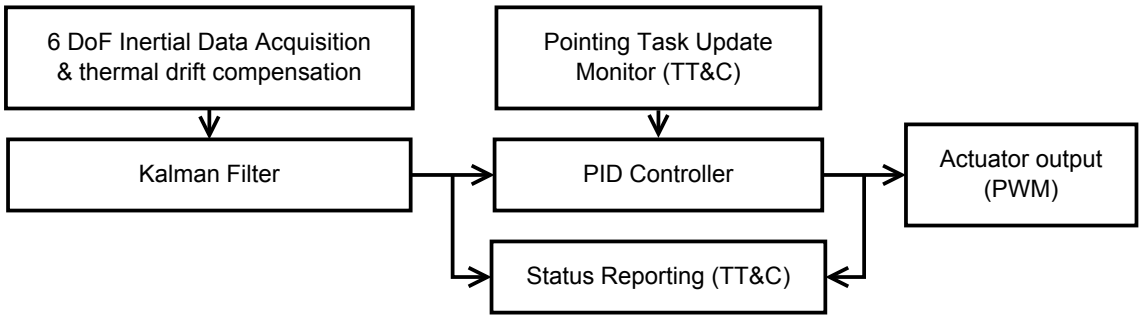


Figure 1.3: Structural overview of the satellite firmware for the N-Prize mission

The internal flash memory size and computational power (architecture, number of operations per second, mathematical co-processing) may be specified according to the requirements of the algorithms that should be implemented in order to achieve the mission tasks.

The software structure overview shown in the Figure 1.3 illustrates the key software functions necessary to accomplish the N-Prize mission. An estimation of their impact on the system performance, based on the benchmark comparison is summarized in the Table 1.1. It is necessary to take into account that this data is an initial guess and the performance characteristics are to be improved.

Table 1.1: Estimated performance and memory costs of some functions

Function	**Memory, kB**	**Execution time[1], ms** Generic 8-bit RISC
Kalman filter[2]	5.8	5.2 (192 Hz)
PID[3]	5.6	2.1 (476 Hz)
Total	11.4	7.9 (126 Hz)

Assuming that the efficient programming will be an additional challenge or even a call for future work, it is concluded that 8-bit MCU can perform all the required tasks with the lowest consumption profile and can be found in a smallest package. It is also clear that using for instance an ARM Cortex M3 or similar MCU the programming and code efficiency requirements are non significant due to the extensive computational power, though it is hard to find a representative model of an MCU in this class with a comparable power profile or suitable form factor: most of MLF, QFN and TQFP packages for such MCUs starts with the side length of 7 mm, when an 8-bit MCU can be found even in 3x3 mm QFN package.

As stated in the table above, a 16 MIPS 8-bit MCU and $12\,kB$ memory space is sufficient to perform the most expensive functions in terms of required memory capacity and execution time. Assuming that 1 program cicle is performed at least at 100 Hz frequency it is possible to consider an 8-bit low-power MCU a suitable target platform. Applying all the requirements and constraints, the resulting MCU design specification was summarized and stated in the Table 1.2.

Table 1.2: Central processing unit specification

Parameter	**Min**	**Max**	**Unit**
In-system programable flash		16	KB
Internal SRAM		1	KB
IO interfaces	1xUART, 1xSPI, 1xI2C(TWI), 6xPWM		
Computational power	1 MIPS/MHz, up to 20 MIPS		
Clock frequency	0.4	16	MHz

[1]The benchmark results were selected only for MCUs equipped with an 16-bit MPU that were tested at $16\,MHz$ of the clock frequency.

[2]The Kalman filter implementation for the 8-bit platform used in the performance test was taken from http://www.starlino.com/imu_guide.html.

[3]The PID controller source code used for the performance test was provided by Brett Beauregard at http://www.arduino.cc/playground/Code/PIDLibrary

1.2.2 Position determination subsystem

In case of the inertial measurement unit, the subsystem design specification was done and the components selection was justified [3]. The aforementioned work was performed as the part of WikiSat research group activities so it is assumed that the selected components are meeting all the requirements of this work as well. The resulting design specification of the position determination subsystem is presented in the Table 1.3.

Table 1.3: Position determination subsystem specification

Parameter	Accelerometer			Rate Gyroscope		
	Min	Max	Unit	Min	Max	Unit
Measurement range	± 6	± 24	g	± 2000		deg/sec
Data rate	50	1000	Hz	1000	8000	Hz
Sensitivity	± 6	± 3	$mg/digit$	14.3		deg/sec
Resolution	16 bits					

1.2.3 Thermal control subsystem

The thermal control subsystem is considered to be passive and used only for data correction purposes. A temperature sensor is necessary to monitor the system temperature in order to predict and compensate for thermal drifts in sensors or adjust the transmission channel.

Nowadays, temperature sensors are embedded in most MEMS motion sensors for thermal drift compensation purposes. To ensure homogeneous heat distribution between the key subsystems a strong common ground plane as well as thermal vias are to be used. The resulting design specification of the thermal control subsystem are presented in the Table 1.4.

Table 1.4: Thermal control subsystem specification

Parameter		MCU	Accelerometer	Gyro	Transciever	Resulting/
Operating	Max	$+85$	$+85$	$+85$	$+85$	**+85**
Temperature, ℃	Min	-40	-40	-40	-40	**-40**
On-board temperature sensor characteristics						
Parameter				**Min**	**Max**	**Units**
Measurement range				-30	$+85$	℃
Resolution					16	bit

1.2.4 Pointing subsystem

The study of the magnetic actuators for the N-Prize satellite-on-a-board orientation in the magnetic field of Earth was performed [21] explaining the design methods and requirements to the subsystem. In order to provide additional flexibility in the final actuator selection and related experiments, it was decided to use a raw non-amplified analog output

or PWM interface able to provide at least $40\,mA$ in peak. Such a decision will provide additional flexibility for the device in its development and experimentation applications. The optimal solution in this case can be achieved using an MCU with 2 analog output or raw PWM output lines.

1.2.5 Communication subsystem

The carrier frequency for the communication subsystem is one of the key specification defining criteria for the transceiver. The $2.4\,GHz$ frequency was selected as the current radio link budget study for the N-Prize femto-satellite [28] has shown that this frequency is preferable for the platform because it can meet the requirements of a wide range of possible missions. The high gain antenna for the satellite-on-a-board concept has been developed [9] to work on this frequency as well. The transceiver should have a sleep mode and allow power consumption profile.

The RF output of the transceiver should be compatible with a standard $50\,Ohm$ antenna RF interface as it will be used to work with different amplifiers and/or antennas. The digital interface between the processing unit and the transceiver should be based on one SPI bus for the programming simplicity and efficiency reasons. The communication protocol may be proprietary but it should be documented good enough to be decoded by a software radio if needed. The transceiver should be able to work in a mesh or p2p network mode with up to at least 100 of instances. The resulting design specification of the communication subsystem is presented in the Table 1.5.

Table 1.5: Communication subsystem specification

Parameter	Min	Max	Unit
Operating frequency	2400	2525	MHz
Air data rate	250	2000	$kbps$
Frequency deviation	± 160	± 320	kHz
Non-overlapping channel spacing	1	2	MHz
Maximum output power		$+4$	dBm
Bandwidth (depending on data rate)	700	2000	kHz
Sensitivity	-94	-82	dBm
external crystal frequency		16	MHz
external crystal tollerance		± 60	ppm
modulation type		GFSK	

1.2.6 Time reference unit

As the mission duration is limited to a maximum one week, it is not necessary to install a special RTC for time determination and it is possible to use high quality crystal oscillator. It also was observed that many transceivers for $2.4\,GHz$ carrier frequency are requiring a $16\,MHz$ clock source as well as most of the MCU in the class of interest. The decision was made to use a $16\,MHz$ crystal oscillator with the smallest possible form-factor and highest

accuracy. To ensure high stability and coexistence with the transceiver, the frequency tolerance value of the oscillator selected should not be higher than $\pm 60 \times 10^{-6}$.

1.2.7 Power supply subsystem

As the N-Prize femto-satellite mission is limited to 9 spins and previous research [28] has shown that other similar low-cost missions based on the same technology should be limited to a week, the power source for the satellite should be a battery and there is no need in its recharging or solar panel interfacing. This assumption allows limitation of the power supply subsystem to a battery interface and a voltage regulator, which should have low noise properties and should have good thermal drift characteristics.

The power supply for the pointing actuators should be provided by a separate regulator with a consumption limit which should be defined during the pointing actuation system development. This fact allows limiting the power supply subsystem to a single voltage regulator which should provide a stable source for communication, position determination, time reference subsystems and to the central processing unit as well.

An overview of the low power low cost MCU market shows that a typical 20 - $25 MIPS$ MCU with a 8 - $20 MHz$ clock source is powered in a range from 2.7 to $5.5 V_{dc}$. For position determination and communication subsystems it is optimal to use a $3.3 V_{dc}$ power source. It is also important to take into account the development and experimentation platform functionality of the target device which makes it interesting to use a USB programming and debugging interface as an alternative power source for the core subsystems. The resulting specification of the power supply subsystem is presented in the Table 1.6.

Table 1.6: Power supply subsystem specification

Parameter		MCU	Accelerometer	Gyro	Transciever	Common/ Total
Supply voltage, V	Min	1.8	2.16	2.1	1.9	**2.16**
	Max	5.5	3.6	3.6	3.6	**3.6**
Current consumption, mA	Min	1.2	0.25	6.5	0.4	**8.35**
	Max	5.0			13.5	**25.25**

1.2.8 Development support subsystem

Most of the modern 8-bit RISC MCU platforms have an in-system self programmable flash program memory feature that basically allows making MCU re-programmable via any interface by means of the proper boot-loader. For the simplicity of implementation and wide community support reasons it was decided to follow the approach of using the UART interface for boot-loader-based in-system programming and debugging purposes.

The USB to UART bridge was a necessary solution in order to provide a comfortable development and experimentation environment allowing the developer to perform software experiments in a fast and safe way. Support of the bit-banging mode as well as the 500000

baud per second operating mode and existence of the on-board 3.3 Vdc voltage regulator with at least $100\,mA$ output were the selection criteria for the USB to UART bridge.The resulting design specification for the development support subsystems is presented in the Table 1.7.

Table 1.7: Communication subsystem specification

Parameter	Min	Max	Unit
Data rate	300	576000	$baud/s$
Voltage requlator output,	3.0	3.6	V
Max voltage regulator output current		100	mA

1.2.9 The AWIP definition

So far, taking into account all aforementioned facts regarding the required system structure, functionality and performance it is clearly seen that the desired satellite-on-a-board technology demonstrator that is capable of meeting the N-Prize requirements as well as playing a role of development and experimentation toolkit is nothing more than an Advanced Wireless Inertial Platform (AWIP).

This definition allows extending the initial aims of the device, making it an interesting technical solution for a variety of scientific, educational and engineering problems. It can possibly become a helpful tool speeding up the development of motion processing, feedback control and other low-cost embedded applications with wireless communication capabilities. The target applications may vary from a femto-satellite or a UAV autopilot to an ultra-compact object tracker to even a geo-tagging camera plug-in for 3D-imaging applications.

1.3 Components selection

Following the design specification stated above the component selection was performed. The primary and backup (or secondary) component instances were selected according to the *Geographical Availability* rule of the WikiSat research group taking as well into account their cost characteristics. The summary of the component selection work is summarized in the Table 1.8

One of the selection reasons that is worth mentioning in case of the MCU selection was that the Atmel ATMEGA168 MCU was selected as the primary candidate has a strong community support in terms of available hardware and firmware solutions. The Arduino[1] development IDE as well as boot-loader firmware and corresponding libraries can significantly simplify the final firmware development process.

Table 1.8: Key platform components list

Component candidate		Mfg	Model	Size mm	Package Type	Price[2] €
MCU	Primary	Atmel	ATMEGA168	5×5	MLF32	3.07
	Secondary	ST	STM8L151G6	4×4	QFN28	2.35
Accelerometer	Primary	ST	LIS331HH	3×3	QFN14	3.64
	Secondary	ST	LIS3DH	3×3	QFN14	3.64
Rate Gyroscope	Primary	InvenSense	ITG-3200	4×4	QFN24	25.39
	Secondary	ST	L3G4200D	4×4	LGA16	13.72
Transciever	Primary	Nordic	nRF24L01+	4×4	QFN20	3.78
	Secondary	TI	CC2500	4×4	MLF20	3.67
USB to UART	Primary	Silabs	CP2103	5×5	QFN28	4.05
	Secondary	FTDI	FT232RQ	5×5	QFN32	5.93
Clock	Primary	Epson	TSX-3225	2.5×3.2	QFN28	2.54
	Secondary	Kyocera	PBRV16.00MR	4.5×1.2	QFN32	1.52
					Total components cost, minimum:	28.95
					Total components cost, maximum:	44.35
					Total components cost, primary:	42.47

[1] Open source and open hardware development and educational platform based on ATMEGA MCU family, http://www.arduino.cc

[2] The prices are given for a reference purposes only; the price information was obtained from local websites of Farnell and RS-Online electronic component distributors and are taken for the quantities below 10 units.

CHAPTER 2. TECHNICAL IMPLEMENTATION

The technical implementation of the AWIP was split into the following parts: hardware design, prototyping, hardware debugging and firmware implementation for the life-proof tests. The aforementioned implementation steps were performed and documented below. One of the key aims of the technical implementation part was to perform it with minimum expenses and using low-cost in-house manufacturing tools that could be afforded by a particular hobbyist, open source and open hardware programming and debugging tools.

2.1 Hardware design

One of the most important tools in the hardware design process is the CAD software used for schematics and board design, electrical and manufacturing-related rules compliance revision and CAM output synthesis for prototyping or outsourced manufacturing. The study of available open source and freeware tools was performed and the CadSoft Eagle CAD was selected[1] due to its advantages such as freeware license; wide community support and amount of component libraries; user-scripting language support making the CAD highly customizable and others.

This software package is able to produce CAM output in GERBER RS-274[2] and EXCELLON[3] providing though a high compatibility with industrial PCB manufacturing and assembling services as well as in-house prototyping using even a toner-transfer method. It is also important to note that the package is equipped with an auto-routing tool which allows achieving the lowest development time.

2.1.1 System busses

The hardware design of the system was based on the data sheets and application notes of the components selected. The interconection of the key component and interface selection was performed according to the corresponding specifications as well as following the interfacing scheme presented in the Figure 1.2. The pin numbers referenced below correspond to the MLF-32 package of the ATMEGA168 MCU.

2.1.1.1 Serial ISP bus

The hardware SPI[4] bus interface of the MCU was used for the low-level in-system programming (ISP) mode natively supported by the MCU. This mode is toggled if the proper

[1] Official website: http://www.cadsoft.de/

[2] PCB manufacturing CAM format, http://en.wikipedia.org/wiki/Gerber_format

[3] PCB drilling and routing CAM format, http://en.wikipedia.org/wiki/Excellon_Format

[4] Stands for Serial Peripheral Interface bus, http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

command is received by the MCU on the SPI bus in a few milliseconds after the reset event. The MISO (master in - slave out, pin 16), MOSI(master out - slave in, pin 15), SCK (serial clock, pin 17) and RESET (29) pins of the MCU are used for this interface.

By means of this interface it is possible to re-program the boot-loader section. For this purpose, a proper hardware adaptor and corresponding programming software are required.

2.1.1.2 USART bus

The USART[5] bus was selected for intrfacing with the USB to UART bridge for boot-loader based in-system programming and debug terminal purposes. The RX(30), TX(31) and DTR (connected to the reset pin 29 through the $0.1\mu F$ capacitor) data lines were used in this bus. The boot-loader program will begin the firmware update procedure if in the first few seconds after the power-on or reset event the correct STK-500 command is received through the RX pin. The DTR pin is used to provide a remote control of the MCU reset event to the programming host computer as it allows toggling the programming mode on demand.

According to the factory default settings the serial ISP programming mode is disabled and can be enabled only through the low-level high-voltage parallel programming which can be performed using low-cost hardware tools as well. The listening time may be defined by programming the MCU configuration registers.

2.1.1.3 Two-wire bus (I2C)

The 2-wire bus (TWI) or I2C was used to interface with the accelerometer and gyroscope. The bus can simultaneously contain up to 125 devices with the same address space and can be operated at up to 400 kHz frequency. Native hardware interface was used assigning the serial clock role (SCL) to the pin 28 and the serial data (SDA) to the pin 27.

2.1.1.4 Soft SPI and transceiver control

To ensure safety of the native SPI interface used for serial ISP and due to the neglectable losses as well as pin-spacing reasons, the decision was made to use the transceiver interface SPI bus in the software mode connecting the transceiver to general input-output pins instead of a native SPI. Software mode basically means that the role of the shift registers will be performed by the code. Practically it means that instead of $10ns$ for data-shifting transaction it will take from 160 to 80 depending on the code efficiency.

Transmission enable (CE pin of the transceiver) and interrupt (IRQ pin of the transceiver) pins are also required to control the transceiver. Additional pins are required to control the transceiver. The soft SPI role will be performed by pin 9 (MISO), 10 (MOSI), 11 (SCK) and 13 for chip-select (CSN).

[5] Stands for universal asynchronous receiver/transmitter, http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

2.1.1.5 PWM actuator outputs

Due to the very limited pin-space, the decision was made to use pins 11 and 12 necessary for the serial ISP as the PWM outputs for the actuators as the usage overlap is impossible. Both pins are able to provide a pulse-width modulated output with the 8-bit resolution by the hardware means of the MCU.

2.1.1.6 GPS input

An external GPS receiver may be connected to the pin 13 used as the serial clock of the serial ISP interface. In such a case the soft UART bus should be implemented. For the best performance of the soft UART the usage of a state-machine and interrupt based algorithm is assumed.

2.1.2 Components conditioning and interfacing

The conditioning of all the components was performed according to the manufacturers recommendations of the implementation methods stated in the corresponding data sheets [2, 26, 15, 22, 11, 25] and taking into account the interfacing agreements mentioned above. The values of the passive components used were preferably selected according to the corresponding component manufacturers recommendations, though package types were selected preferring the smallest acceptable form factor if can be provided by any local distributor.

The MCU conditioning and interface scheme is presented in the Figure 2.1 as an example of the schematics design process. The complete schematics design document is presented in the Appendix A.

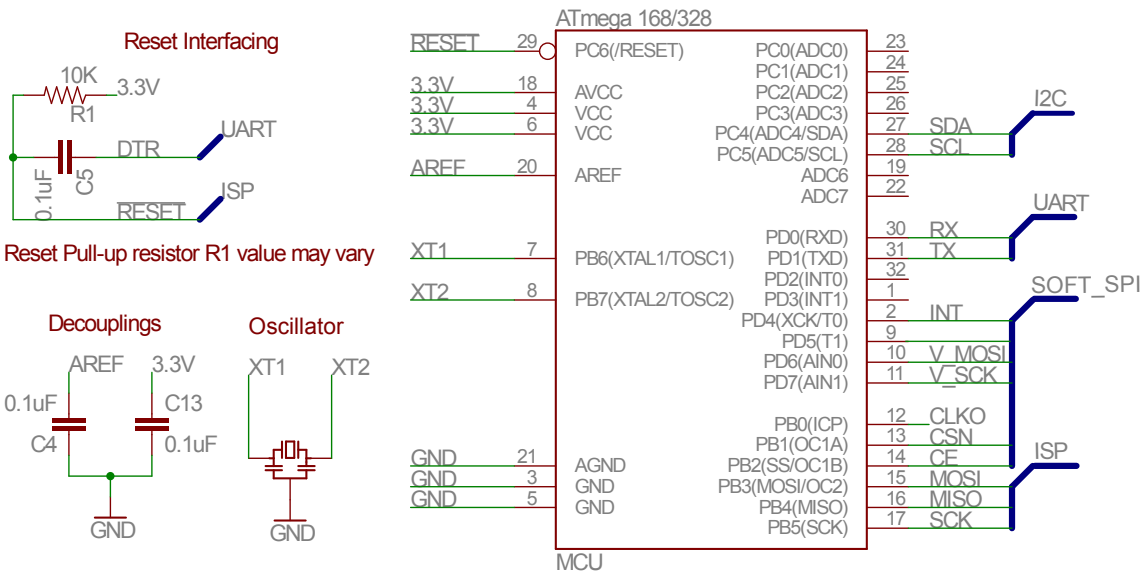


Figure 2.1: MCU interfacing and conditioning schematics

2.1.3 The bill of materials

Once the schematics design was finished the bill of materials (BOM) was created using an embedded script “bom.ulp”. The BOM was completed with the data of component providers as well as pricing and delivery time information. The study of local electronic components providers was performed in order to obtain lowest price values and fastest delivery. As regards accelerometers and transceivers it was possible to obtain development samples from the manufacturing instance. The complete BOM is shown in the Table 2.1. The component prices shown in the table were taken assuming production of at least 10 units.

Table 2.1: The complete AWIP bill of materials

No	Part	Mfg	Value	Case	Qty	Supplier	Reference	Price, €
1	IC1	Atmel	Atmega168	MLF-32	1	Farnell	1455105	2.92
2	IC2	ST	LIS331HH	QFN-12	1	Farnell	1838525	3.95
3	IC3	InvenSense	ITG3200	QFN-24	1	Farnell	1858279	18.79
4	IC4	Nordic	nRF24L01P	QFN-24	1	Farnell	1346268	4.87
5	IC5	SiLabs	CP2103	QFN-28	1	Farnell	1291538	3.25
6	CLK	Epson	TSX-3225	custom	1	Farnell	1712841	1.74
7	C3-5, 13, 14, 16, 20, 21	AVX	$0.1\,\mu F$	0402	8	Farnell	1833862	0.023
8	R1, 3, 6	Vishay	$10\,K\Omega$	0402	3	Farnell	1738864	0.028
9	C6, 17	MURATA	$10\,nF$	0201	2	Farnell	1776000	0.03
10	C1, 2 (polar)	KEMET	$10\,\mu F$	1206	2	Farnell	1457414	0.27
11	C11	AVX	$1.5\,pF$	0201	1	Farnell	1327609	0.022
12	C7	KEMET	$1\,nF$	0201	1	Farnell	1535558	0.3
13	C12	AVX	$1\,pF$	0201	1	Farnell	1327608	0.031
14	C22	AVX	$1\,\mu F$	0402	1	Farnell	1327658	0.195
15	C9, 15	AVX	$2.2\,nF$	0201	2	RS	698-3033	0.055
16	C10	MURATA	$4.7\,pF$	0201	1	Farnell	1776018	0.029
17	C18	AVX	$10\,\mu F$	0603	1	Farnell	1833804	0.27
18	C8	MURATA	$33\,nF$	0402	1	RS	723-5244	0.022
19	C19	AVX	$100\,nF$	0402	1	RS	698-3162	0.045
20	R4	Multicomp	$22\,K\Omega$	0201	1	Farnell	1851022	0.04
21	L1	Johanson	$8.2\,nH$	0201	1	Farnell	1865740	0.047
22	L2	Johanson	$2.7\,nH$	0201	1	Farnell	1865731	0.047
23	L3	Johanson	$3.9\,nH$	0201	1	Farnell	1865735	0.047
24	IC_VREG	MICREL	MIC5205	custom	1	Farnell	1663083	0.82
25	USB_CON	KYOCERA	USB mini B	custom	1	Farnell	1558179	1.69
26	PCB	MICRON-20	AWIP F1v1	custom	1	micron-20		15.33
							Total:	55.19

2.1.4 Board design process

The Eagle CAD tool-set allows converting the device schematics into the board design template hardware links between all the components in the form of wires, directly connecting the corresponding pins in the nets. The board design process can be represented as a position optimization task.

The PCB design process can be significantly simplified if the proper design procedure is followed. The procedure used in the current work was graphically represented in the Figure 2.2 and described by the following text. First, the design rules were introduced to the system specifying all the tolerances, minimums and maximums as well as preferred tracing strategies for each layer. Than the optimal position of each component was manually defined taking into account the optimal spacing inter-connection simplicity factors. The backup image of the board design was done right after the component positioning process in order to provide a safe-restore point for the following routing procedures.

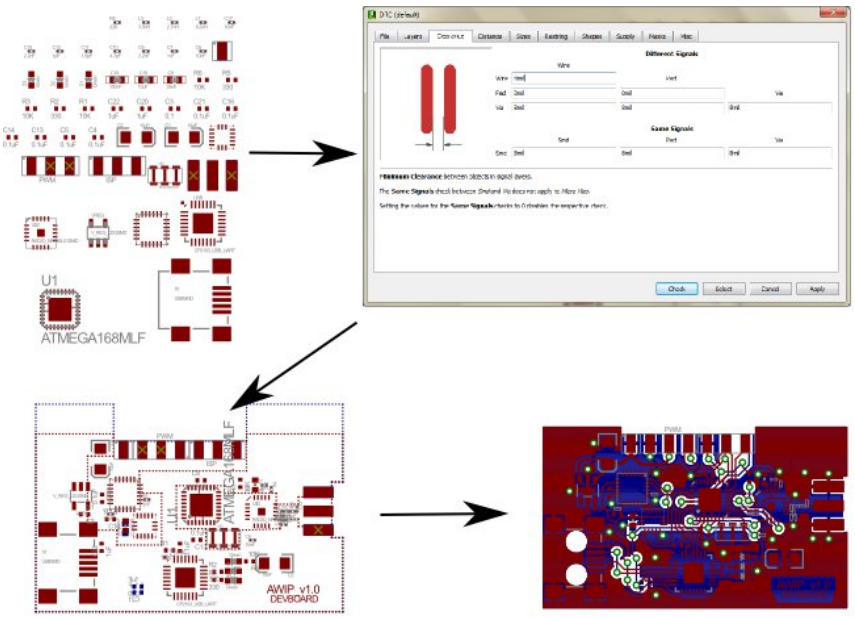


Figure 2.2: A simplified diagram of the PCB design process

If the design rules are correctly defined and the component positions are selected properly the embedded auto-routing script can be applied. Auto-routing script is based on the iterative algorithm and can be run a few times in order to improve the solution. When the best possible result is achieved the design rule check (DRC) script should be run. If the DRC has detected any errors that are to be revised manually. The DRC script should be run after each significant design check or correction. Once all the DRC issues are resolved the board design can be considered finished.

As the PCB assembling process was decided to be done in-house, the assembling procedure was created using the final BOM as a design reference. The assembly form is shown in Appendix B.

2.2 PCB prototyping procedure

In order to fulfill the N-Prize requirements of being low-cost and easy to manufacture, it was decided to make a prototype using the amateur technology. The toner-transfer masking and $FeCl3$ etching methods were selected. A toner-transfer device and etching tank were manufactured and used for the prototype. A 2-axis drilling semi-automatic CNC was built and used as well. The design and implementation processes for each of the aforementioned tools are described in [28, 3]. Some of those tools are shown in the Figure 2.3.

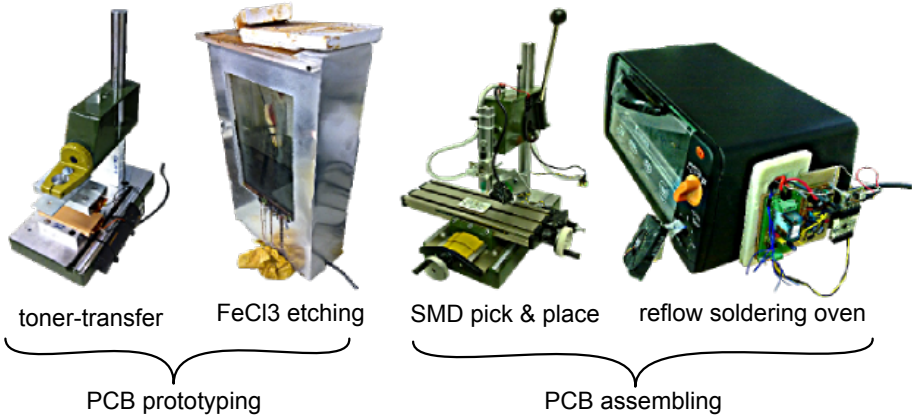


Figure 2.3: The low-cost PCB prototyping and assembling tools used in this work

It is important to mention some of the technology-related observations obtained during the in-house prototyping of the AWIP board. It was observed that the toner-transfer masking was not very reliable and played a weak point role, though it was possible to obtain a stable mask and achieve an acceptable result after etching. It was also observed that the top-bottom layers inter-connections (vias) were to be done after the assembling of the board in the reflow oven if the wire-based inter-connection technology was used, though it was acceptable to do the layer inter-connection before assembling if it was based on the riveting technology. The chemical and electroplating technologies are out of the scope of the in-house manufacturability study.

The in-house prototyping of the PCB including only toner transfer and etching took 18 hours due to the toner-transfer masking failures related to inhomogeneous heating, over-exposure, inhomogeneous toner distribution (printing cartridge blade quality to be checked). Taking into account all the aforementioned facts it is concluded that low-cost in-house prototyping of the board is possible though it is not efficient in terms of time and quality.

Due to the necessity of the PCB rapid prototyping the market study of European rapid prototyping services was performed. The offers for the shortest lead time were compared resulting with selection of a Bulgarian PCB prototyping and assembling company[6] Micron-20. The final manufacturing price proposed for an 8-boards batch was $4.19€/board$ and the complete price with shipping and artwork production included was $15.33€/board$.

[6]Company website: http://www.micron20.com/en

2.3 Post-factory MCU initialization

As mentioned above, the ATMEGA MCU comes with a serial ISP feature disabled and the only possible way to enable this feature is to re-program the MCU fuse registers using the high-voltage programming method (HVPP) which cannot be done in system.

The implementation of the HVPP method requires a special device capable to control the procedure. Using the official datasheet, a few on-line papers explaining the experiences of HVPP method implementation and taking into account the needs of the AWIP project, the HVPP Toolkit was designed. This method is used not only to enable serial ISP feature, but also to set the correct external oscillator and enable CLOCK_OUT feature. The aforementioned features were configured by setting the corresponding bits in the fuse registers of the MCU to the following values provided by the on-line fuse calculator[7]: $0x87$ for the low fuse, $0xDD$ for high and $0x00$ for extended.

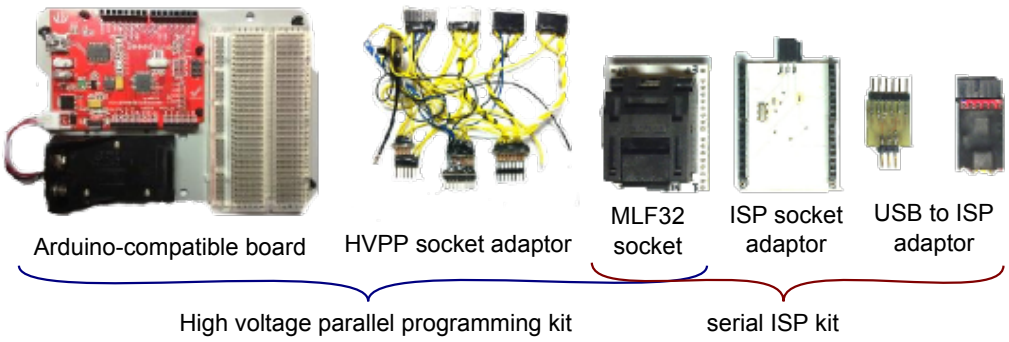


Figure 2.4: Low-cost programming tools for MLF-32-encapsulated ATMEGA MCUs

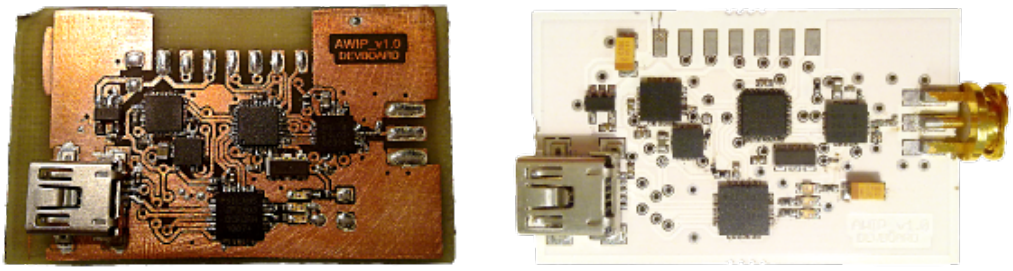
The low cost programming tools used in the HVPP process are shown in the Figure 2.4. The key hardware component used in the setup was the MLF-32 spring-loaded burn-in socket[8] which was used to connect the MCU to the programmer in a mechanically safe and non-destructive way. An Arduino-compatible micro-controller board and an adapting circuit were used to ensure a safe interaction between the programmer and the target MCU. The programming firmware was derived from an open-source HVPP project [16].

The serial ISP target board compatible with the MLF-32 socket adapter was designed in order to allow testing the MCU in the same socket-based non-destructive manner before the assembling as well as allowing the boot-loader programming. Programming the boot-loader before the assembling process makes the board ready to be used with the USB right after the post-assembling check procedure. The open-source *avrdude*[9] tool was used as the programming software to flash the boot-loader image.

[7] The calculator can be found at http://www.engbedded.com/fusecalc
[8] The low-cost MLF-32 spring-loaded socket was obtained from http://www.chinics.com/
[9] The homepage of the *avrdude* project can be found at http://www.nongnu.org/avrdude/

2.4 PCB assembling procedure

The PCB assembling process was done in-house using a set of very basic tools. The process was split into 3 main parts: soldering paste application; components positioning; reflow soldering. An important part of the process was preparing the components for the assembly in the previously defined assembling order. The aforementioned BOM-based printed form was made for keeping the components with the help of double-sided duct tape. The assembly examples of both manually prototyped PCB and the one provided by an external company are shown in the Figure 2.5.



Low-cost manual in-house PCB prototyping Low-cost industrial PCB prototyping

Figure 2.5: Example of AWIP assemblies with different PCB prototyping methods used

In order to follow the hard low-cost manufacturing style manual needle-based paste dispensing method was used instead of the common stencil-printing method. The average time[10] spent on the paste application is 25 minutes. It was also considered possible to use a stencil-printing method with a laser-engraved stencil made by external commercial provider and corresponding GERBER files were produced though were not used due to the time constraints.

As the smallest part used in the assembly had a size of 0.2 by 0.5 mm it was necessary to use a pick and place tool for the assembling process. Due to a high cost of the professional pick-and-place equipment and low efficiency of hobbies-oriented versions, a decision was made to build a low-cost manual pick-and-place tool (see Figure 2.3) able to operate 0201 packaged[11] components . For this reason a special tool, based on the precision bench and an amateur vacuum pen was designed and implemented. Due to the significant difference between 0201 package handling and packages above 0402 a set of vacuum pen adapters was made.

In order to improve the assembling time characteristics, the manual pick and place tool was equipped with a gear-motor controlled by a keypad. The visual guidance and assembly inspection modes are provided by a adjustable web camera mounted close to the vacuum pen tool.

A low-cost reflow oven shown in Figure 2.3 and based on the consumer toaster device was made in order to ensure a quality reflow soldering. The oven was equipped with a thermoresistive temperature sensor attached to the 8-bit ADC of the oven controller. The reflow oven controller was based on the same ATMEGA168 hardware and Arduino firmware. The embedded software allows programming the heat treatment procedure according to

[10] The measurement was done through assembling of 5 units

[11] according to SMD component package specification by JEDEC, commonly used international standard.

the typical requirements of the reflow soldering process (preheating temperature and time, soldering temperature and time and final cooling time). The feedback of the reflow process was provided to the PC terminal in the real time during the soldering process via USB to UART bridge interface. More information on the low-cost assembling tools used in this work can be found in [3].

2.5 Post-production treatment

The post-production visual inspection was performed for all the prototypes using the camera of the pick-and-place machine. The defects detected during the visual inspection were corrected using low-cost soldering station with both hot-air rework tool and high precision soldering iron depending on the defect type observed. A short-circuit caused by an excessive amount of the soldering paste was one of the most commonly observed defects.

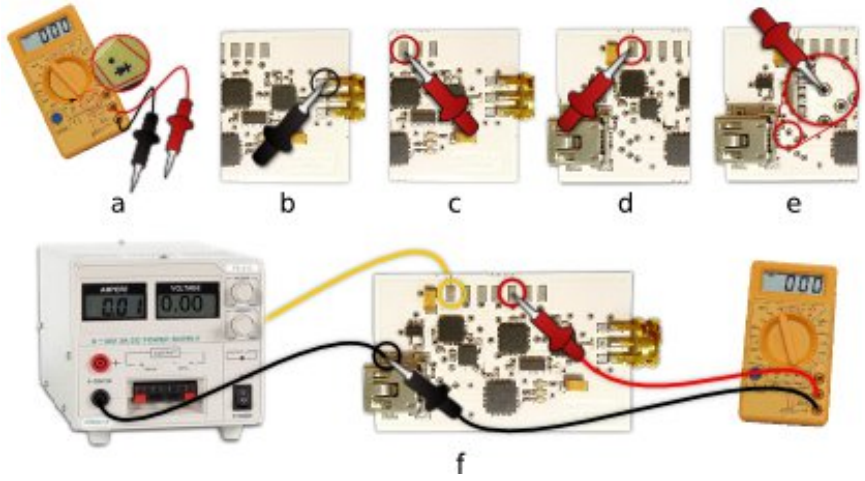


Figure 2.6: Post-production electrical safety test procedures

In order to ensure that the device is safe to operate or interface with other electronic equipment a set of basic electrical safety test routines was developed and implemented. Those routines were designed to ensure absence of short-circuits in the power supply lines and correct operation of the on-board voltage regulators.

The supply lines short-circuit test routine is illustrated in the Figure 2.6. Points (b,c), (b,d) and (b,e) were probed for a short-circuit between using a generic digital multimeter (a) with short-circuit detection function. The setup, illustrated in Figure 2.6(f) was used to test the on-board voltage regulator in a safe way, varying the input voltage from 0 to $12V$ and checking the output value and stability.

2.6 Environmental Impact

The impact of this type of technology demonstrators can be taken in terms of saving power during the manufacturing process and the life cycle in general. Hazardous materials were used during the manufacturing process including ferric acid, fluxes and lead-free soldering pastes that could be harmful for the environment in certain conditions.

A set of safety measures was taken in order to ensure safety of operators. Air extractor was used in the assembly line and personal protection equipment was used including the safety gloves, plastic glasses during the work with acids as well as respirators during the soldering works.

Low cost and lead-free materials were used in this project in order to minimize environmental impact. In order to minimize the amount of waste produced during the work some recycled components were used. All the boards with production defects were recycled and reused in other prototypes as well.

CHAPTER 3. TESTING AND EVALUATION

This chapter will demonstrate the test and evaluation procedures developed in order to assess operability of the device and the results analysis will be stated. In order to improve credibility of the results obtained all the tests were done for at least 2 devices (up to 6, depending on the availability). Basically the aforementioned procedures were split into 2 groups: proof of alive tests and performance evaluation. It is also important to mention that most test procedures are based on the firmware and requires no additional hardware instrumentation (except for a few RF tests).

3.1 Proof of alive test group

The group of proof of alive tests was designed as an advanced continuation of the post-production tests intended to provide a quality-control-like feedback for hardware debugging procedures. The necessity to track and eliminate hardly traceable hardware defects was an additional factor forcing the development and implementation of this test routine group. The tests were grouped by the data bus types instead of subsystems.

3.1.1 USB-UART interface test

As the USB to UART interface was designed for programming and debugging purposes it was necessary to ensure its proper functioning. Basically, the key idea of the test procedure was to determine if the bridge is operating properly when plugged in the USB bus and that it is responding correctly to discovery and status requests. There were 6 boards subjected to this test and no hardware or performance defects were detected among them. The USB discovery test was performed in the Linux OS using the "*lsusb*" tool, though it could be performed in Microsoft Windows by means of "*usbview*" application.

3.1.2 Subsystem discovery test procedure

In order to ensure that inertial sensors and transceiver are soldered, configured and conditioned properly it was necessary to develop a test procedure. It was observed that due to mechanical properties of the sensor packages and limitations of the low-cost positioning and soldering techniques for such kind of packages there was a common soldering defect when the pins of both sensors used for slave address selection appeared to be floating or unsoldered resulting in unpredictable slave address change.

To provide a solution for the problem mentioned above an I2C discovery test routine was implemented. When loaded to the MCU the program scans each address of the I2C bus device address space and echoes to the UART the ones that are found to be occupied. If the device address is floating in certain conditions or none of possible slave addresses are present the mechanical correction is done using a high precision soldering iron or

reworking the sensor IC with a hot air gun depending on the type of soldering defect observed.

The firmware code for this test routine was derived from the I2C discovery code project developed for the Arduino platform[19]. Some changes were made to the original code in order to fix the address space limitation issue as the gyro slave address appeared to be out of the initial scan range as well as to output the addresses found in a hex for reading simplicity.

To ensure that transceiver was correctly conditioned and its interface with the MCU was working well the register scanning test procedure was implemented. The operational theory of the test routine can be presented as reading the transceiver register values and comparing them with the defaults. If an incorrect response is observed the hardware inspection of the IC pads soldering is considered necessary in order to track the defect source. For all the six boards produced the unsoldered MISO pin was detected only in one but tracked and repaired during the post-assembly testing procedure.

```
AWIP Post-production test protocol

1. I2C discovery test
1.1 ST Accelerometer found at 0x19
1.2 InvenSense Gyro found at 0x69

2. nRF24L01 register value test
2.1 Register 0x0 value is 0x8 as expected; PASSED
2.2 Register 0x1 value is 0x3F as expected; PASSED
2.3 Register 0x2 value is 0x3 as expected; PASSED
2.4 Register 0x3 value is 0x3 as expected; PASSED
2.5 Register 0x4 value is 0x3 as expected; PASSED
2.6 Register 0x5 value is 0x2 as expected; PASSED
2.7 Register 0x7 value is 0xE as expected; PASSED

Post-production test finished, session can be closed.
```

Figure 3.1: Post-production subsystem discovery test output example

The I2C and transceiver discovery tests were combined together in a single source in order to simplify the test procedure. For the simplicity reasons and due to the time constraints the aforementioned test procedure was implemented in the Arduino environment. The typical output protocol obtained via a serial port terminal is illustrated in the Figure 3.1. The source code is presented in the Section C.1.

The "*GTKTerm*" package was used for the Linux-based tests though any terminal software can be used to communicate with the board during this test. In order to provide an improved software compatibility, a decision was taken to use 115200 baud rate. For the Microsoft Windows based test session the "*WikiTerminal*"[1] package was used as it supports advanced streaming modes and higher data-rates.

[1]The program can be found in http://code.google.com/p/moon-20/downloads/detail?name=wikiterminal_win32.zip.

3.2 Subsystem performance evaluation and testing

The purpose of this section is to describe methods, procedures and results obtained during the performance evaluation tests of the AWIP subsystems. Those tests were done to determine minimum operational capabilities of the platform and prove that it is enough to fulfill the initial design requirements stated in the first chapter of this work.

3.2.1 UART to USB performance evaluation

A set of tests was designed and performed in order to determine the data transfer constraints related to USB to UART bridge the CPU time cost of a data package transfer from a platform to PC. The test was performed for 2 boards only as all the hardware defects that could possibly affect the test performance were assumed filtered during the post-production testing making the performance characteristics variation neglectable.

Table 3.1: The UART communication test summary

Baudrate	**Message rate,** Hz	**MCU time cost,** s/msg	**Datarate,** $kbps$
1200	8	0.13128	0.117
4800	31	0.03322	0.454
9600	61	0.01660	0.893
19200	121	0.00830	1.772
38400	241	0.00415	3.530
57600	368	0.00271	5.390
115200	736	0.00135	10.781
230400	1389	0.00071	20.346
250000	1563	0.00063	22.895
500000	3125	0.00031	45.776
1000000	5750	0.00017	84.228

The transmission evaluation test was performed for a constant length payload of 15 bytes simulating a data packet containing acceleration and angular rates for 3 axis and temperature of the gyroscope package. The test results are summarized in the Table 3.1. It is necessary to note that communication test failed for 460800, 576000 and 921600 baud rates as the error rate in this modes is higher than acceptable[2]. The optimal rate in terms of MCU time cost was 1000000 mode which allows transmitting 15 bytes message in $170, \mu s$. The source code of the firmware used to perform this test is presented in the Section C.2. It is also important to mention that a special USB to UART bridge configuration tool, which is described in [24], is necessary to enable the $1 Mbit$ mode.

[2] More information about baud rate modes and errors as well as error estimation formulas are available in the “The Baud Rate Generator” subsection of the [2, p. 172]

3.2.2 IMU data acquisition performance evaluation

The purpose of this experiment was to perform a study of the DAQ system of sensors and MCU determining the maximum data acquisition and streaming rates through the UART interface varying the methods of reading the FIFO registers of the sensors and I2C bus parameters. It was also necessary to determine an MCU time cost of a single DAQ operation at the maximum rate. The payload for this experiment was defined as the 3 values of acceleration and 3 values of angular velocity measurements resulting in a message size of 12 bytes.

Due to the possibility of externally-induced UART communication errors causing loose of the synchronization in the binary communication mode, a 2-byte message header was added as it was considered to be more efficient in the experiment conditions than the check-sum method. A performance comparison at maximum DAQ rate was made for text[3] and binary modes was summarized in the Table 3.2.

Table 3.2: AWIP DAQ evaluation results

Stream data to:	MCU, RAM	UART, binary mode	UART, text mode
Data packet size, B	12	14	$13-32$, variable
Data packets/s	$1,628$	$1,353$	$374-510$, variable
Total transmitted, kB	19.07	18.49	13.62, approx.
Message CPU time cost, μs	610	740	$2,740$, approx.

The source code of the aforementioned experiment is presented in the Section C.3. The results obtained allowed to perform a DAQ code optimization resulting in an Arduino library AWIPIMU designed to handle all the sensor operations in the simplified way though keeping the performance level as high as it was demonstrated in this experiment.

The experiment results showed that the best performance in terms of MCU time was achieved when the I2C bus was operated in Fast mode $(400\,kHz)$ and that the CPU time cost for reading a 16 bit value was equal for both sensors. The total cost for acquiring 12 bytes of data (only accelerations and angular rates were selected as the thermal changes are considered to happen at lower frequency) assuming the bus operation in Fast mode was equal to $610\,\mu s$. The maximum data rate of streaming complete 6-DoF message from sensors to the MCU RAM achieved was $1,628\,Hz$ while in binary streaming mode through UART at $1,000,000\,baud/s$ the value of $1,353\,Hz$ was obtained.

The summary of data streaming tests is presented in the Table 3.2. Taking into account the aforementioned results decision was taken to operate the I2C bus in the Fast mode. It was also concluded that the sensors output data rates could be set to the $1\,kHz$ mode to ensure loss-less data streaming for future experiments, though it may be interesting in some cases to slow down to $0.3\,kHz$ mode.

[3] The comma separated value (CSV) text format was used for encoding the message in the text communication mode

3.2.3 IMU data processing demo

The purpose of this experiment was to demonstrate the way of interfacing the IMU data with the common scientific tools stating an example of the IMU noise distribution model study in the non-disturbed condition. A binary data stream of 6-DoF[4] (12 bytes) with a 2 bytes synchronization header was recorded for a period of $1\,s$ while keeping the board in a neutral position with minimum of disturbing factors. The δt was not added to the data stream as it was observed to be equal for all measurements in the binary streaming mode if the DAQ and streaming operations are performed at the maximum rate (described in the previous section).

The data recording was done using the *WikiTerminal* program while the unpacking of the binary data was performed with a specially made $bin2CSV$ converter[5]. The CSV format was chosen as a final storage format for the data due to the simplicity of the import procedure and compatibility with most of the data processing programs. The synchronization header was necessary to be used in order to ensure data integrity due to the UART error factor [2] of at least 1% for the data-rate of $1\,Mbps$ All the further processing of the data obtained was performed in the $matlab$ environment.

$$\omega_i[^\circ/s] = value_i \times \left(\frac{1}{scale\,factor}\right) \tag{3.1}$$

$$a_i[m/s^2] = value_i \times g_0 \times \left(\frac{g_{range}}{2^{15}}\right) \tag{3.2}$$

The conversion of the raw gyroscope data to $^\circ/s$ was done for each axis according to the equation 3.1, where the scale factor was taken from the sensor datasheet [15] for normal temperature of $25\,^\circ C$ without any additional thermal correction. The accelerometer raw data was converted according to the equation 3.2 where the g_{range} defines the measurement range of the sensor and in this test was equal to $\pm 6\,g$. The standard gravity g_0 is used as additional multiplier to convert the value from g to m/s^2. The data collection part of the test was performed in the normal conditions of $25\,^\circ C$ so no thermal correction or compensation method was used.

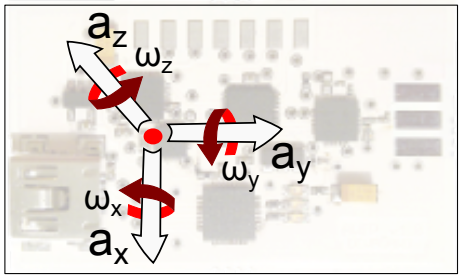


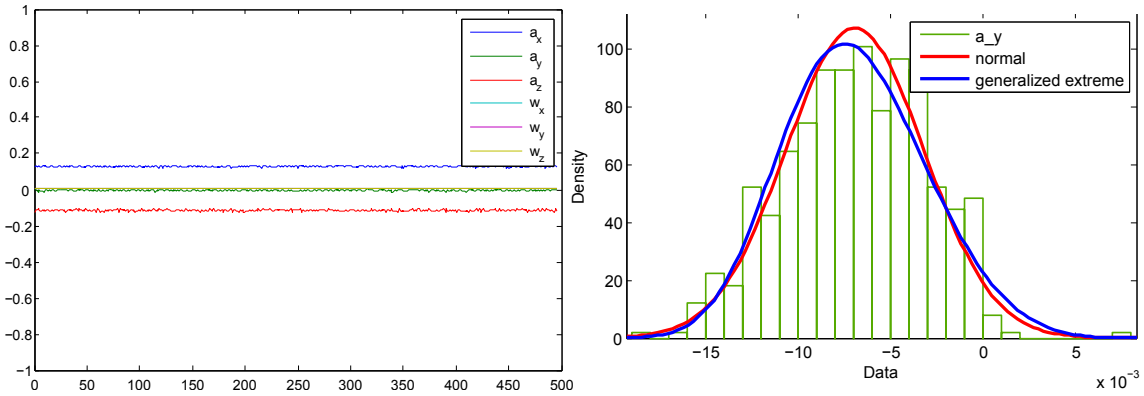
Figure 3.2: IMU axis alignment

[4] DoF in this context stands for “degrees of freedom”.

[5] The program and its sources are available at http://code.google.com/p/moon-20/source/browse/#svn/gadgets/FAWIP_F1/bin2CSV.

The alignment of the IMU axis with respect to the board is shown in Figure 3.2 and must be taken into account while reading processing the IMU data. The code for the data collection routine is presented in the Section C.3 switched to the binary output mode with the 2 byte synchronization header and $1\,Mbit$ UART data-rate in order to gain the highest resolution.

The corresponding code for data post-processing in the $matlab$ is not attached due to its simplicity, though it is necessary to mention that the CSV data may be imported to the workspace using the GUI of the workspace window only. An example of the raw data and the $dfittool$ output for the data obtained from one of the accelerometer channels is shown in the Figure 3.3.



(a) $6DoF$ normalized data example **(b)** Probability density view example

Figure 3.3: IMU data processing visualization examples in MATLAB

3.2.4 RF output power and offset test

An experiment was carried out in order to ensure that the output power of the transceiver is correct and constant in all the bands as declared in the datasheet. It was also necessary to determine the carrier frequency offset levels in all the bands of the transceiver to ensure its proper operability.

The corresponding test program was implemented and is presented in the Section C.4. The RF spectrum analyzer for $2.4\,GHz$ and a serial data monitoring tool (laptop) were the only additional hardware resources involved in the test.

$$F_0 = 2400 + CH\,[MHz] \tag{3.3}$$

The output carrier frequency start, stop and step parameters as well as the time that specified frequency is active are defined in the constants setup section of the code. The expected frequency F_0 was calculated for each channel according to the formula 3.3, where the CH was defined as a 7 bit number[22].

The observation of the experiment results showed that the output power was constant for all the channels tested in the range from $2400\,MHz$ to $2525\,MHz$ and was equal to $0\,dbm$ as expected according to the datasheet. The frequency offset observed was also constant

for all the bands tested and was equal to $0.25\,MHz$ which was within the specification limit of $1\,MHz$.

3.2.5 Wireless Link Test

The wireless link validation procedure was designed in order to ensure the correct functioning of the transceiver as well as the related HAL and control library made for it in various communication modes. The parameters varied were the data-rate, payload size, control check-sum presence and its size, acknowledged and non-acknowledged modes. The variation of the distance was not scope of this test as the antenna and RF front-end may vary depending on the desired satellite architecture though it was observed that with a typical dipole antenna for $2.4\,GHz$ band the $20\,m$ distance may be achieved at least.

The platform combined with external low-noise amplifier and a typical $24\,dB$ parabolic antenna for $2.4\,GHz$ band[6] has been used in a long-range link test campaign in a role of ground station (setup shown in the Figure 3.4) achieving the desired result of $7\,km$. A break-board for an RF front-end[7] was used for both ground station and the remote node in this experiment. The maximum power amplifier gain achieved was $7.8\,dBm$ while the receiver sensitivity was supposed to be equal to $-104\,dB$. The antenna of remote node was custom made and is a part of the work presented in the [9]. The source codes of the programs for RX and TX modes are presented in the Appendix C.8 and Appendix C.9.

Figure 3.4: AWIP ground station setup for a long-range communication test

A ground station and the remote node programs were implemented in order to perform the test. The main role of the ground station program was to receive the package from a specific sender, convert it from binary to text format or add binary synchronization header if the binary output was desired, and echo the final output to the UART interface. The source code of the ground station role is presented in the Section C.5. The varied parameters were the node and local address, payload size and structure, RF configuration and the message representation method.

The source code of the remote node program is presented in the Section C.6. The payload

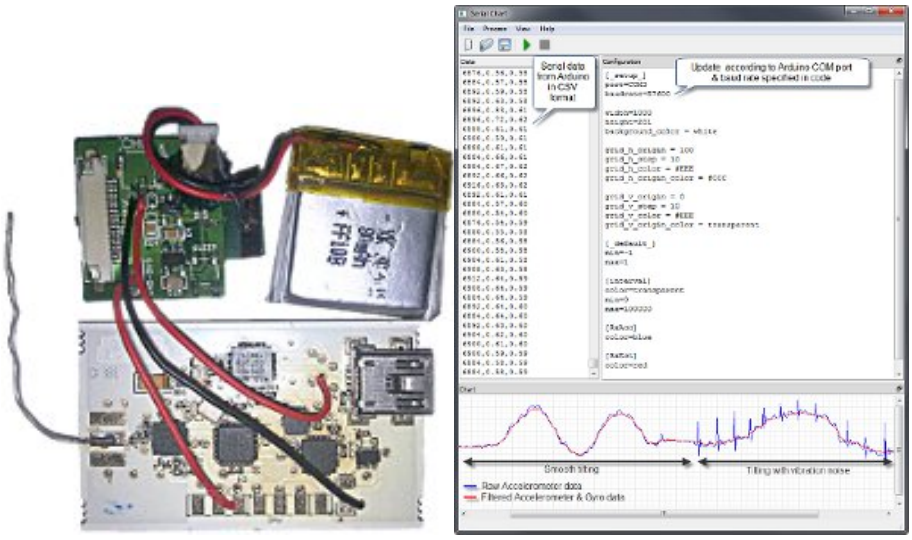
[6] The antenna used in this experiment was found in http://www.cablematic.es/Antena-2_dot_4-GHz--802_dot_11_hyphen_b_-_g_-_n/Antena-parabolica-de-2_comma_4-GHz-y-24-dBi/?pag=13.

[7] The datasheet of the front-end used can be found at http://maxiamp.com/downloads/MCP01_r5.pdf.

data sent by the remote node was taken from various sources during the test, including the raw IMU data, GPS NMEA stream and a static binary message (to have a measurement of the real data rate excluding the processing losses). In case of the IMU data streaming the DAQ rate was limited to $100\,Hz$ in order to simplify the real-time visualization process. During all the tests accelerometer was setup to the $\pm 6\,g$ though it was considered possible to implement an automatic measurement range adjustment algorithm if needed in the future experiments. The test-ready remote node setup is shown in the Figure 3.5a

The *SerialChart*[8] program was used in order to visualize graphically the received IMU stream in the ground station computer. The program is able to read the data stream from the serial port interface of the PC and interpret CSV-encoded data. The data is visualized than in an oscilloscope style with an option to configure all the display parameters, including the individual color representation for each channel. An example of the UI is shown in the Figure 3.5b.

The experiment performed has confirmed that the wireless link subsystem is functioning properly and can be easily re-configured to work with any type of data payload. The hardware abstraction layer and the link library were significantly improved in the sense of performance and code density during the test development resulting in the compiled library size of $2.1\,KB$. The source code of the link library is presented in the Section C.7



(a) AWIP remote node setup

(b) IMU stream visualization with SerialChart[3]

Figure 3.5: Wireless link test elements

[8] The program and the Figure 3.5b are taken from http://code.google.com/p/serialchart/.

CHAPTER 4. APPLICATION SCENARIOS

The following chapter will introduce a set of application scenarios of the platform developed. Those scenarios are covering various development applications related to the development process of a satellite-on-a-board, its subsystems and corresponding software as well as test setups for hardware-in-a-loop test and evaluation procedures. Also some scenario proposals related to aeronautic applications in general.

4.1 Femto-satellite and ground station development kit

The femto-satellite application development and experimentation kit is the primary role and application scenario of the device. Though the combination of external components may vary depending on the development scope, the global target application scope is the TT&C, payload management, power and thermal management and attitude control development. The key hardware elements and their relations for this application scenario are shown in the Figure 4.1.

In this scenario the 6-DoF scalable motion measurement system is used to determine the satellites trajectory and provide the data for the trajectory estimator and the control loop which than can process the data and send the control command to the attitude control subsystem able to actuate 2 servos used as a representation of the final actuator. Additional GPIO pin may be used as an input for an external GPS unit in order to gain higher precision of the trajectory estimation over a long time period.

It is known that the modern sensor fusion algorithms for 6-DoF systems are quite limited in the sense of long-time performance and drifts, though it may be assumed enough for the femto-satellite mission as the motion model is well known and may be fitted to the trajectory estimation algorithm. It is also important to note that there are various research projects working on the trajectory estimation issues and successfully minimizing the thermal and integration drifts using various mathematical models [13, 1, 7].

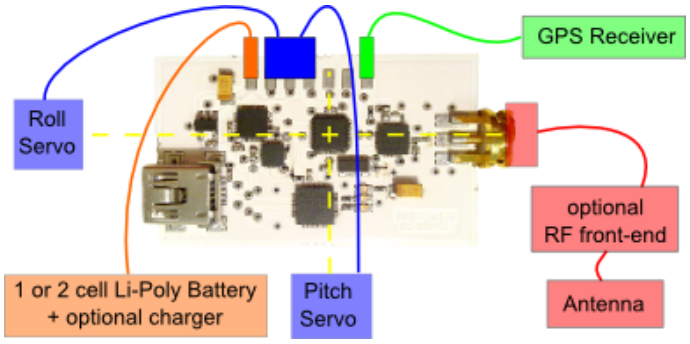


Figure 4.1: Femto-satellite SDK hardware inter-connections map

The determined and estimated trajectory data and the control history are supposed to be packaged and sent to the ground station using the embedded radio-link subsystem in order to provide the ground crew with the proof-of-life message. The pointing of antenna and an

optional payload sensor is supposed to be performed by 2 servos in order to simplify the control development process.

The code for driving thermal and power management subsystems may be developed in this scenario as well. The hardware means of the control over the current consumption and sleep-mode of all the components is provided. The thermal data is supposed to be used in the development process of the thermal drift compensator for the IMU (the initial compensator parameters are provided for the gyro, accelerometer and the clock in the corresponding data-sheets [15, 26, 11]) and the system clock source.

Any device compatible with Nordic NRF24 protocol system can be used as a ground station receiver (including the software defined radio) though it is recommended to use a different AWIP board for the ground station development purposes due to the code compatibility and task similarity.

In order to simplify the end RF components development such as power amplifier, low noise amplifier and switch or complete RF front-end and antenna the system was equipped with an RP-SMA RF connector.

4.2 Secondary application scenarios

The purpose of this work was to design and implement a satellite-on-a-board development and experimentation kit, though it was observed that the resulting product may serve to perform some additional tasks and may be used as a part of other complex systems or as a stand-alone product. Some of the key secondary application scenarios are stated in the following sections.

4.2.1 PicoRover autopilot and remote control unit

The PicoRover is a robotic system designed for lunar exploration and developed by the WikiSat research group as a form of participation in the Team FredNet group, the official competitor of the Google Lunar X-Prize. The AWIP platform fits perfectly the role of the autopilot and remote control for this mission as it has an interface for the rover's actuators as well as the IMU to provide the inertial reference for the navigation purposes. The UART interface of the system may be used to interface with the imaging payload if needed. An example of the AWIP platform used as a PicoRover autopilot is shown in the Figure 4.2.

It is also necessary to note that the AWIP application for the PicoRover allows development of the scalable rover networks of multiple architectures. The development process for such application will not require any additional hardware as the embedded hardware resources are considered sufficient for this role.

(a) AWIP board as an autopilot **(b)** PicoRover field test

Figure 4.2: Examples of AWIP used as an autopilot for a PicoRover

4.2.2 PID tuning bench

The AWIP platform may be used as a development platform for an implementation of a high efficiency PID control algorithm for a low-cost energy-efficient aeronautic system. A single-engine stabilization test bench was proposed as a low-cost PID development, tuning and experimentation platform. An example of a hardware setup proposed is shown in the Figure 4.3.

The key hardware elements involved in this setup are the brushless motor with a propeller, interfaced with an electronic speed control unit (ESC), a high Li-Poly battery with a fast discharge capability and an AWIP platform as a control unit and a remote command interface. The IMU is supposed to be the sensor data provided to the PID control loop while the ESC unit will play a role of actuator interface. The wireless link interface is supposed to be used as a state setup and monitoring interface.

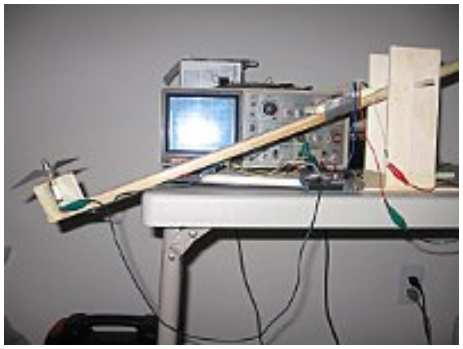


Figure 4.3: PID test bench setup[1]

[1]The image was taken from http://www.nesinfinity.com/Projects-PID.html.

4.2.3 IMU-Assisted GPS (GAP)

The GAP (high precision geo-locator, Spanish abbreviation) is a device developed by the WikiSat research group in collaboration with Nebrija university and a D47 R&D company as a part of the TURISMAP project funded by AVANZA grant. The purpose of the device is to provide a GPS-like information with a high precision using the IMU and sensor fusion algorithms. The location data should be available even in the areas without GPS coverage or in RF jamming environment.

GAP is considered to be a multipurpose high precision tracker that can be used for different final applicastions. The key applications considered was 3D imaging assistant (recording the position and the heading of a typical camera in order to use the dato for the 3D scene reconstruction) and the high precision personal tracker for the crew of various tactical teams of disaster-management organizations.

In this application scenario the only additional hardware element required is an external GPS unit as the rest necessary resources are embedded in the AWIP platform. The implementation of the filtering and sensor fusion algorithms is the only work to be done in order to implement the aforementioned device. An example of the AWIP platform used as a GAP device is shown in the Figure 4.4

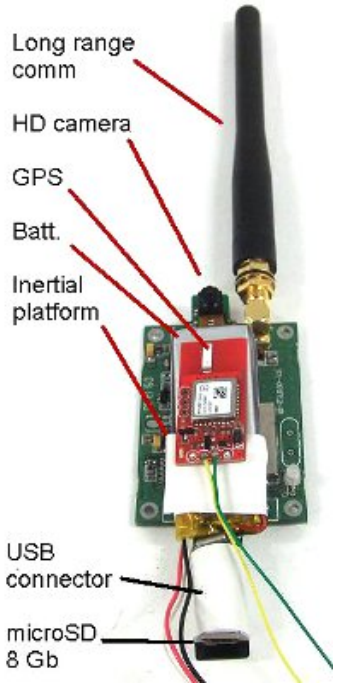


Figure 4.4: AWIP board used in a GAP device

CONCLUSIONS

The present work has shown the necessity of the satellite-on-a-board development and experimentation kit as well as the process of its design and technical implementation. A set of hardware testing and validation procedures were presented and the results were discussed and published by [18]. A group of primary and secondary application scenarios were shown the possibility of practical usage of the device as a development platform for various aerospace applications as well as a stand-alone product. It was also shown that the proposed hardware system is enough to achieve the N-Prize competition goal.

It is important to note that the platform developed has opened a set of new challenges for the future research related to the development of the flight software for the femto-satellite. The implementation of the flight software for the N-Prize mission appears to be another call for future work based on this platform. Some other members all ready started to study the aforementioned issues using the presented platform.

It is also important to mention that further experimentation and research work is required in such fields like trajectory estimation and sensor fusion algorithms, thermal and power management algorithms in order to fulfill the N-Prize requirements. Another development challenge will be the real time operating system able to manage the states of the final satellite subsystem which can also be developed using the presented platform.

BIBLIOGRAPHY

[1] *E. Foxlin. Inertial head-tracker sensor fusion by a complementary separetebias Kalman flter. In Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium* (1996).

[2] ATMEL. Atmega 168 data sheet. http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf.

[3] BARDOLET, E. Study of a low cost inertial platfom for a femto-satellite deployed by a mini-launcher. Master's thesis, Universitat Politècnica de Catalunya, 2010.

[4] BARNHART, D., VLADIMIROVA, T., AND SWEETING, M. Satellite-on-a-chip: a feasibility study. In *Fifth Round Table on Micro/Nano Technologies for Space, Nordwijk* (2005).

[5] BARNHART, D., VLADIMIROVA, T., AND SWEETING, M. Design of self-powered wireless system-on-a-chip sensor nodes for hostile environments. In *Circuits and Systems ISCAS, IEEE International Symposium* (2008).

[6] BARNHART, D. J., VLADIMIROVA, T., BAKER, A. M., AND SWEETING, M. N. A low-cost femto-satellite to enable distributed space missions. *Acta Astronautica 64* (2009), 1123 – 1143.

[7] BARSHAN, B., AND DURRANT-WHYTE, H. F. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation 11* (1995), 328–342.

[8] BONET, L. High altitude balloon mission design and implementation for a mini-launcher. Master's thesis, Universitat Politècnica de Catalunya, 2011.

[9] DE LAS HERAS LÓPEZ, A. Design and implementation of a synthetic aperture antenna for a femto-satellite. Master's thesis, Universitat Politècnica de Catalunya, 2011.

[10] DEAR, P. H. N-prize, rules in full. http://www.n-prize.com/assets/rules_in_full.pdf, 2008.

[11] EPSON TOYOCOM CORPORATION. Tsx-3225. http://www.epsontoyocom.co.jp/english/product/Crystal/set02/tsx3225/index.html.

[12] FORTESCUE, P., STARK, J., AND SWINERD, G. *Spacecraft Systems Engineering*. Wiley, 2003.

[13] G.S.W. KLEIN, T. D. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing 22* (2004), 769–776.

[14] GUTIERREZ, K., AND COLEY, G. Pcb design guidelines. Tech. rep., Texas Instruments, 2009.

[15] INVENSENSE. Itg-3200 data sheet. http://invensense.com/mems/gyro/documents/EB-ITG-3200-00-01.1.pdf.

[16] KEYZER, J. Avr hv rescue shield 1. http://mightyohm.com/blog/products/avr-hv-rescue-shield/.

[17] KLOFAS, B., ANDERSON, J., AND LEVEQUE, K. A survey of cubesat communication systems. In *California Polytechnic State University* (2008).

[18] KRAVCHENKO, V., AND TRISTANCHO, J. Advanced wireless inertial platform: A femto-satellite toolkit development platform. In *50th AIAA Aerospace Science Meeting* (2012).

[19] KURT, T. E. Arduino as i2c bus scanner (blog post). http://todbot.com/blog/2009/11/29/i2cscanner-pde-arduino-as-i2c-bus-scanner/, 2009.

[20] LIGHTSEY, B. *Systems Engineering Fundamentals*. Defense Acquisition University Press, Fort Belvoir, Virginia, 22060-5565, 2001.

[21] NAVARRO MORCILLO, L. Use of hardware-on-the-loop to test missions for a low cost mini-launcher. Master's thesis, Universitat Politècnica de Catalunya, 2011.

[22] NORDIC SEMICONDUCTOR. nrf24l01+ data sheet. http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P.

[23] SHISHKO, R., AND ASTER, R. *NASA systems engineering handbook*, vol. 6105. NASA Center for AeroSpace Information, 2007.

[24] SILICON LABORATORIES. An205: Cp210x baud rate support. Tech. rep., Silicon Laboratories, 2007.

[25] SILICON LABS. Cp2103. http://www.silabs.com/products/interface/usbtouart/Pages/usb-to-uart-bridge.aspx.

[26] ST ELECTRONICS. Lis331hh data sheet. http://www.st.com/stonline/books/pdf/docs/16366.pdf.

[27] SUMPNER, D. Manufacturing. http://openlearn.open.ac.uk/mod/oucontent/view.php?id=399740&direct=1, Retrieved April 2011.

[28] TRISTANCHO, J. Implementation of a femto-satellite and a mini-launcher. Master's thesis, Universitat Politècnica de Catalunya, 2010.

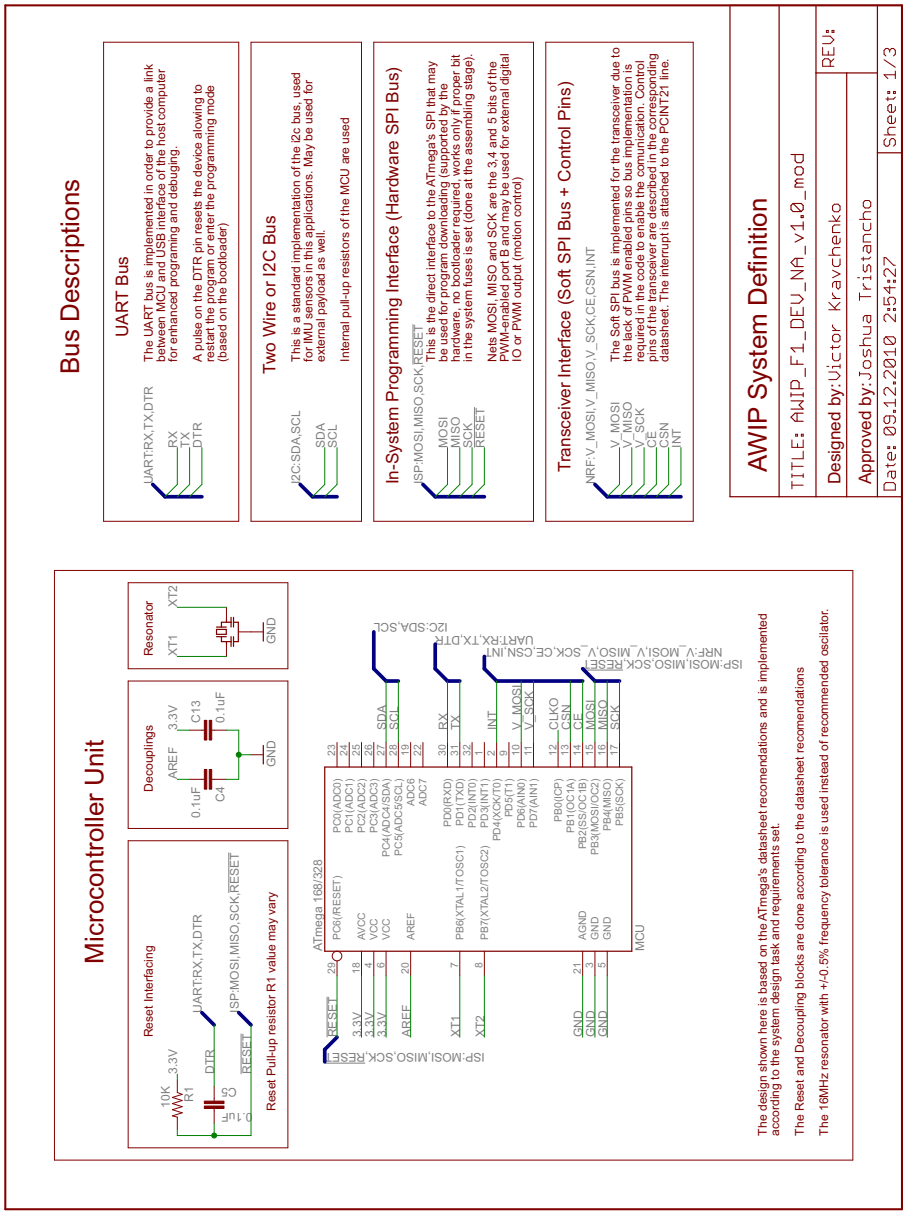
[29] TRISTANCHO, J. Wikisat team engineering management plan. http://code.google.com/p/moon-20/wiki/WikiSat_Engineering_Management_Plan, Retrieved April 2011.

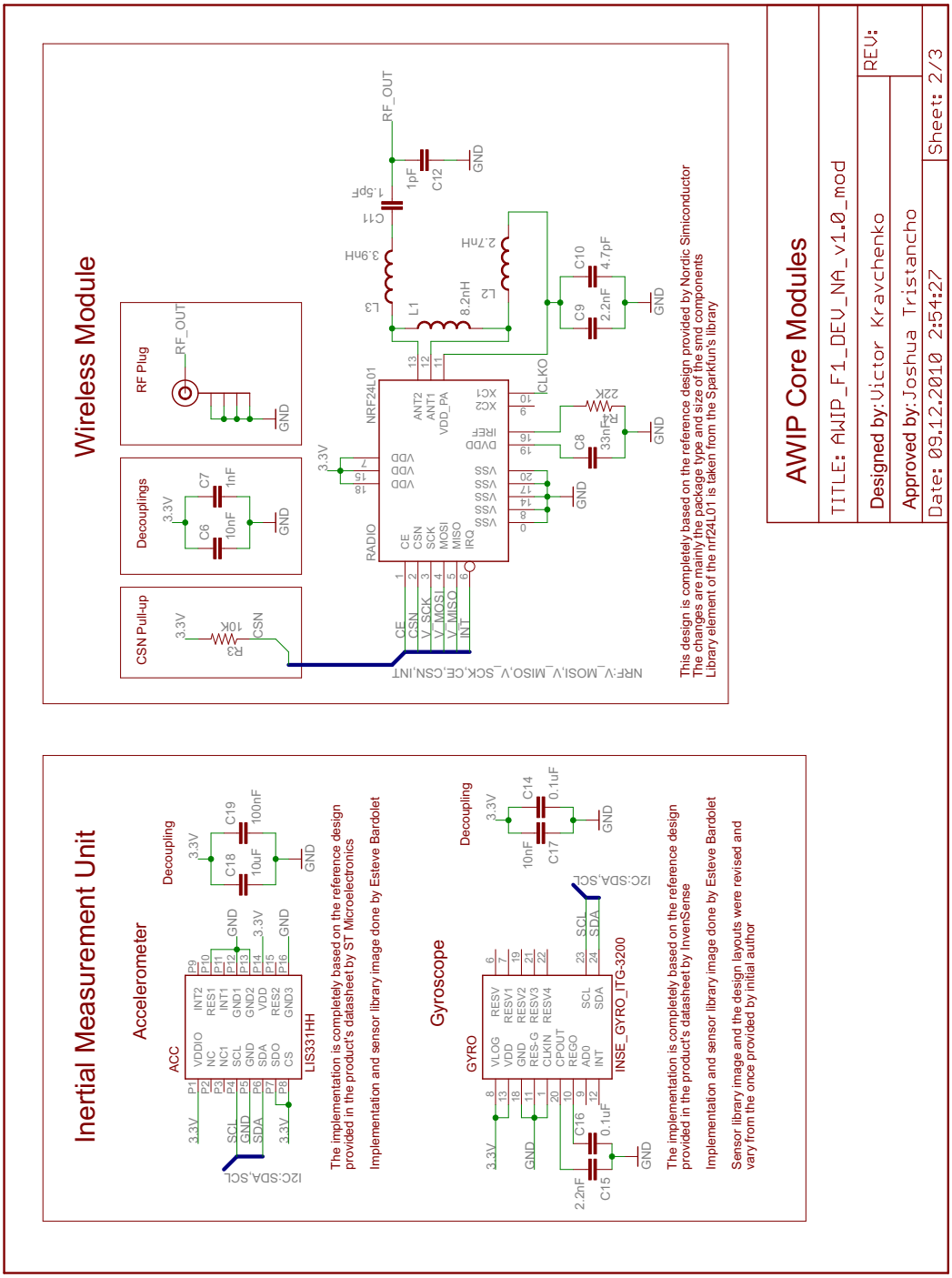
[30] TRISTANCHO, J., MARTINEZ, J., ET AL. Moon-20 home page. http://code.google.com/p/moon-20, Retrieved April 2011.

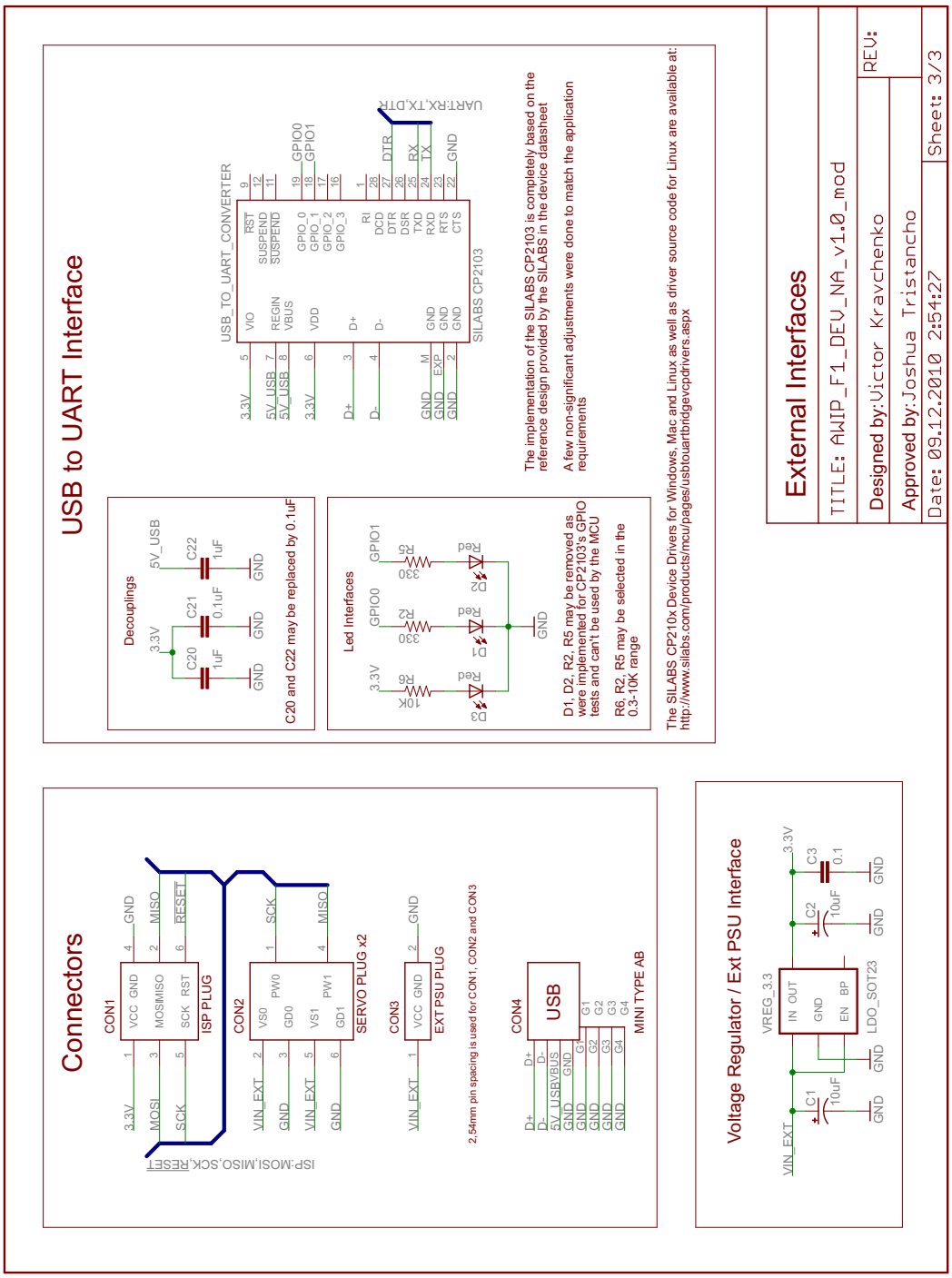
[31] UNDERWOOD, C., UNWIN, M., SORENSEN, R., FRYDLAND, A., AND JAMESON, P. Radiation testing campaign for a new miniaturised space gps receiver. *Radiation Effects Data Workshop, 2004 IEEE 22* (2004), 120 – 124.

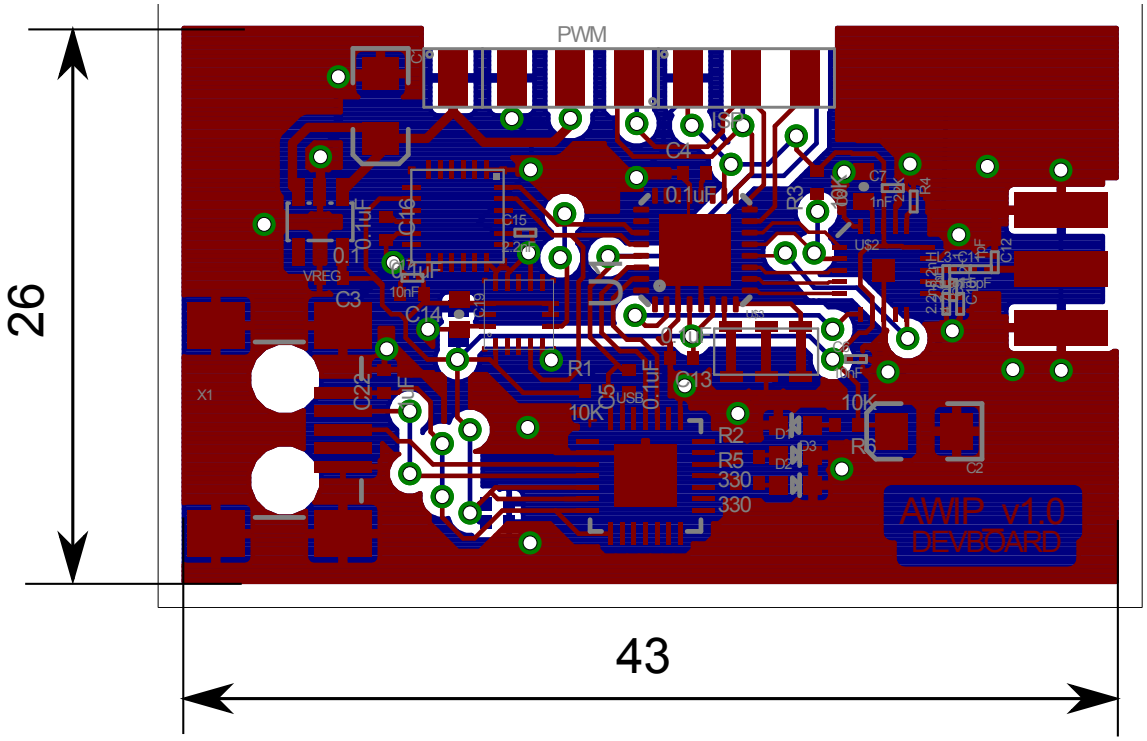
[32] UNDERWOOD, C. I., RICHARDSON, G., AND SAVIGNOL, J. In-orbit results from the snap-1 nanosatellite and its future potential. *Phil. Trans. R. Soc. Lond. A 361* (2003), 199–203.

APPENDIX A. AWIP SCHEMATICS




Wireless Module
RF Plug
RF_OUT
GND
Decouplings
3.3V
C7
1nF
C6
10nF
GND
CSN Pull-up
3.3V
R3
10K
CSN
NRF24L01
RADIO
CE
CSN
SCK
MOSI
MISO
IRQ
VDD
VSS
ANT2
ANT1
VDD_PA
DVDD
REF
XC2
XC1
CLKO
1.5pF
C11
3.9nH
L3
L1
8.2nH
L2
2.7nH
1pF
C12
C10
4.7pF
C9
2.2nF
C8
33nF
R2
22K
NRF:V_MOSI,V_MISO,V_SCK,CE,CSN,INT
This design is completely based on the reference design provided by Nordic Simiconductor
The changes are mainly the package type and size of the smd components
Library element of the nrf24L01 is taken from the Sparkfun's library
Inertial Measurement Unit
Accelerometer
ACC
LIS331HH
VDDIO
NC
NC1
SCL
GND
SDA
SDO
CS
INT2
RES1
INT1
GND1
GND2
VDD
RES2
GND3
Decoupling
C18
10uF
C19
100nF
I2C:SDA,SCL
The implementation is completely based on the reference design provided in the product's datasheet by ST Microelectronics
Implementation and sensor library image done by Esteve Bardolet
Gyroscope
GYRO
INSE_GYRO_ITG-3200
VLOG
VDD
GND
RES-G
CLKIN
CPOUT
REGO
AD0
INT
RESV
RESV1
RESV2
RESV3
RESV4
SCL
SDA
C14
0.1uF
10nF
C17
C16
0.1uF
2.2nF
C15
The implementation is completely based on the reference design provided in the product's datasheet by InvenSense
Implementation and sensor library image done by Esteve Bardolet
Sensor library image and the design layouts were revised and vary from the once provided by initial author
AWIP Core Modules
TITLE: AWIP_F1_DEV_NA_v1.0_mod
Designed by:Victor Kravchenko
REV:
Approved by:Joshua Tristancho
Date: 09.12.2010 2:54:27
Sheet: 2/3


USB to UART Interface
Connectors
CON1
ISP PLUG
CON2
SERVO PLUG x2
CON3
EXT PSU PLUG
CON4
USB
MINI TYPE AB
2.54mm pin spacing is used for CON1, CON2 and CON3
ISP:MOSI,MISO,SCK,RESET
Voltage Regulator / Ext PSU Interface
VREG_3.3
LDO_SOT23
Decouplings
C20 and C22 may be replaced by 0.1uF
Led Interfaces
D1, D2, R2, R5 may be removed as were implemented for CP2103's GPIO tests and can't be used by the MCU
R6, R2, R5 may be selected in the 0.3-10K range
USB_TO_UART_CONVERTER
SILABS CP2103
UART:RX,TX,DTR
The implementation of the SILABS CP2103 is completely based on the reference design provided by the SILABS in the device datasheet
A few non-significant adjustments were done to match the application requirements
The SILABS CP210x Device Drivers for Windows, Mac and Linux as well as driver source code for Linux are available at: http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx
External Interfaces
TITLE: AWIP_F1_DEV_NA_v1.0_mod
Designed by:Victor Kravchenko
REV:
Approved by:Joshua Tristancho
Date: 09.12.2010 2:54:27
Sheet: 3/3


PWM
26
43
AWIP v1.0
DEVBOARD

APPENDIX B. AWIP ASSEMBLY FORM

AWIP F1 Developer's Board v1.0 BOM/Assembly Part List

Type	Qty	Case	Value	Part
Passive components – top mount – needle 0201				
CAP	1	0201	1.5pF	C11
CAP	1	0201	1nF	C7
CAP	1	0201	1pF	C12
CAP	2	0201	2.2nF	C9, C15
CAP	1	0201	4.7pF	C10
CAP	2	0201	10nF	C6, C17
RES	1	0201	22K	R4
IND	1	0201	2.7nH	L2
IND	1	0201	3.9nH	L3
IND	1	0201	8.2nH	L1
Passive components – top mount – needle 0402+				
CAP	7	0402	0.1uF	C3, C4, C5, C13, C14, C16, C21
RES	2	0402	10K	R1, R3
RES	1(+2)	0402	1K	R2, R5, R6
CAP	1	0603	33nF	C8
CAP	1	0603	100nF	C19
LED	1(+2)	0603	RED	D1,D2,D3
POLC	1	1206	10uF or 4.7uF	C1, C2
Passive components – bottom mount – needle 0402+				
CAP	2	0402	0.1uF	C20, C22
CAP	1	***1206***	10uF	C18
Active components – top mount – needle 0402+				
IC	1	QFN20	nRF24L01P	
IC	1	MLF32	ATmega168	
IC	1	Resonator 16MHz		
IC	1	QFN28	CP2103	
IC	1	LGA16	LIS3DH(331HH)	
IC	1	QFN36	ITG3200	
IC	1	Voltage Regulator 3.3V		
CON	1	USB-MINI-B (AB)		

CAP = ceramic capacitor, RES = resistor, IND = inductive coil, POLC = tantalum capacitor, polarized, IC = integrated circuit, CON = connector

APPENDIX C. SOURCE CODES

This appendix contains source codes used to perform proof-of-alive and performance evaluation tests of the AWIP platform. Codes are written in C++ language using AVRC and Arduino libraries. It is recommended to compile the codes listed below with a avr-gcc compiler. For the using simplicity using of Arduino IDE for project management and programming is advised.

C.1 AWIP Post-production Discovery Test, Source Code

The AWIPRF Arduino library is necessary to compile this code. The test procedure is fully automatic and requires only a terminal connection to display the test results.

```
#include <AWIPRF.h>
#include <nRF24L01p.h>

#include "Wire.h"
extern "C" {
#include "utility/twi.h"
}

byte i2cScan(byte start_addr, byte stop_addr)
{
  byte result = 0;
  byte data  = 0;
  byte found = 0;
  for( byte addr = start_addr; addr <= stop_addr; addr++ ) {
    result = twi_writeTo(addr, &data, 0, 1);
    if (result == 0){ Serial.print("1."); Serial.print(found+1, DEC);
      if ((addr == 0x19)||(addr == 0x18)){Serial.print(" ST Accelerometer");}
      else if ((addr == 0x68)||(addr == 0x69)){Serial.print(" InvenSense Gyro");}
      else {Serial.print(" Unknown device");}
    Serial.print(" found at 0x"); Serial.println(addr, HEX);
    found++; }} return found;
}

void setup(){
  Serial.begin(115200);
  Serial.println("AWIP Post-production test protocol\n\n1. I2C discovery test");
  Wire.begin();
  if (i2cScan(1, 120) != 2) Serial.println("1. Hardware revision is advised");
  Serial.println("\n2. nRF24L01 register value test");
  nrf.init();
  byte nrf_regs[7]     = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x07};
  byte nrf_reg_vals[7] = {0x08, 0x3F, 0x03, 0x03, 0x03, 0x02, 0x0E};
  boolean passed = true;
  for (byte i = 0; i < 7; i++){
    Serial.print("2."); Serial.print(i+1, DEC);
    Serial.print(" Register 0x"); Serial.print(nrf_regs[i], HEX);
    byte value = nrf.getReg(nrf_regs[i]);
    Serial.print(" value is 0x");  Serial.print(value, HEX);
    if (value != nrf_reg_vals[i]){ passed = false; Serial.print(" when expected 0x");
      Serial.print(nrf_reg_vals[i], HEX); Serial.println("; FAILED");}else{
      Serial.println(" as expected; PASSED"); }
  }
  if (!passed) Serial.println("2. Hardware revision is advised");
  Serial.println("\nPost-production test finished, session can be closed.");
}

void loop(){}
```

C.2 AWIP UART Performance Test, Source Code

The desired baudrate to test should be specified as a constant before compiling and downloading this code to the target board. The test procedure is fully automatic and requires only a terminal connection to display the test results.

```
//Constants
#define BAUDRATE 500000
#define TESTMSG  "XXYYZZXXYYZZTT"

//Global variables
unsigned long time  = 0;
unsigned long cost  = 0;
double        count = 0;
boolean       stat  = false;

void setup(){
  Serial.begin(BAUDRATE);
}

void loop(){
  time = micros();
  unsigned long local_cost = time;
  if (time < 1000001){
    count++; Serial.println(TESTMSG);
    cost += micros() - local_cost;
  }else if (!stat){ stat = !stat;
    Serial.println("\n\r\n\r================================");
    Serial.print("Baudrate: "); Serial.println(BAUDRATE);
    Serial.print("Message count, total: "); Serial.println(count, 0);
    Serial.print("Message cost, s: ");
    double avg_cost = cost/((count)*1000000); Serial.println(avg_cost, 5);
    Serial.print("Message size, bytes: "); Serial.println(sizeof(TESTMSG));
    Serial.print("Transmited total, bytes: ");
    Serial.println(sizeof(TESTMSG)*count, 0);
  }

}
```

C.3 AWIP DAQ Performance Test, Source Code

The desired baudrate may be selected in the *define* section. The output mode may be selected by uncomenting or commenting the corresponding lines in the text. The output mode may be switched between none (stream to RAM), stream to UART in binary and stream to UART in text. It is also possible to switch the I2C bus frequency mode between *Normal* and *Fast*.

```
//I2C bus frequency should be configured below
#define AWIP_TWI_FREQ 400000L

//UART baud rate should be configured below
#define BAUDRATE 1000000

#include <AWIPIMU.h>
#include <Wire.h>
#include "imu.h"

void setup(){
  Wire.begin();
```

```
  Serial.begin(BAUDRATE);

  //Reconfigures the I2C bus freq. to AWIP_TWI_FREQ
  imu.init();

  //Configure the sensors to default DAQ modes
  //More about sensor setup should be found in
  //the corresponding data-sheets
  imu.setReg(ACC_START);
  imu.setReg(ACC_SET_WIN_06G);
  imu.setReg(GYRO_SET_R22);
  imu.setReg(GYRO_SET_R62);
}

//Global variables
int     count     = 0;
boolean stat      = false;
boolean start     = false;
unsigned long time = 0;
int data[] = {0,0,0,0,0,0};

//Reads the data from sensor $device FIFO to the $out buffer
//The $addr specifies the start register and $len set the
//register autoincrement limit, should be equal to the $out size
void read_stream(int device, byte addr, byte *out, byte len)
{
 Wire.beginTransmission(device); Wire.send(addr);
 twi_readFrom(device, out, 6); Wire.endTransmission();
}

//Stream the $data to the UART in a binary format,
//length limited by the $len
void write_data(byte* data, byte len){
  for (byte i = 0; i < len; i++) Serial.write(*(data+i));
}

void loop()
{
  if (!start){time = micros(); start = true; }
  if (micros() - time < 1000001) // - start
  {
    //Stream data from sensors to the MCU RAM
    read_stream(ACC_ID, B10101000,  (byte *) &data, 6);
    read_stream(GYRO_ID, B10011110, (byte *) &data+6, 6);

    //Stream data from MCU RAM to the UART in binary mode
    //Serial.print("SP");
    //write_data((byte *) &data, sizeof(data));

    //Stream data from MCU RAM to the UART in binary mode
    for (byte i = 0; i < 5; i++){ Serial.print(data[i]); Serial.print(", "); }
    Serial.println(data[5]);

    //Increment packet counter
    count++;
  }else if (!stat){ stat = !stat;
    Serial.println("\n\r================================");
    Serial.print("Count: "); Serial.println(count);
    Serial.print("Avg   cost, us: "); Serial.println((double)1/count, 5);
  }
}
```

C.4 AWIP RF Carrier Test, Source Code

The desired baudrate may be selected in the $define$ section. Additional hardware is needed to perform this test: RF spectrum analyzer able to work with 2.4 - $2.5\,GHz$ bands and with a peak search feature. An RP-SMA adapter may be needed to interface the board with a spectrum analyzer.

```
/*************************************************************

  AWIP RF CARRIER OUTPUT TEST

  Device will transmit a carrier wave at each channel starting
  from RF_CHANNEL_START and finishing at RF_CHANNEL_STOP
  with a step = RF_STEP, MHz holding the channel active
  for HOLD_TIME, ms. Once the RF_CHANNEL_STOP is reached, the
  test will be restarted from the begining.

  SPECTRUM ANALYZER SETUP:
  center frequency = 2.42GHz, span = 100 MHz,
  capture mode = peak hold (TRACE or AVERAGE menu)

*************************************************************/
//TEST CONFIGURATION SECTION
#define RF_CHANNEL_START 0     //F = 2400 + X MHz
#define RF_CHANNEL_STOP  125    //F = 2400 + X MHz
#define HOLD_TIME        5000  //ms
#define RF_STEP          1     //MHz
#define BAUDRATE         115200 //bps
/*************************************************************/
#include <AWIPRF.h>
#include "nRF24L01p.h"

void setup()
{
  Serial.begin(BAUDRATE);
  nrf.init();
  nrf.setReg(CONFIG,   0x02);
  nrf.setReg(RF_CH,    RF_CHANNEL_START);
  nrf.setReg(RF_SETUP, 0x9F);
  nrf.printSTATUS();
  nrf.printCONFIG();
  nrf.printRFSETUP();
  delayMicroseconds(1500);
  ceHi();
}

byte freq = RF_CHANNEL_START;

void loop(){
  Serial.print("Channel (register val) active: ");
  Serial.println(nrf.getReg(RF_CH), DEC);
  delay(HOLD_TIME);
  ceLow();
  freq +=RF_STEP;
  if (freq > RF_CHANNEL_STOP) freq = RF_CHANNEL_START;
  nrf.setReg(RF_CH, freq);
  ceHi();
}
```

C.5 AWIP RF Link Test - GS Role, Source Code

The desired baudrate may be selected in the *define* section. The channel number, local and remote addresses, desired CRC and ACK modes as well as data-rate and payload type are setup and may be redefined if needed in the *define* section of the code.

```
#include <AWIPRF.h>
#include <nRF24L01p.h>

#define LOCAL_ADDR  "AWsrv" //must be 5 bytes
#define REMOTE_ADDR "AWIP0" //target addr, must be 5 bytes
#define RF_CHANNEL  75      // Freq = 2400 + RF_CHANNEL and always < 2525, MHz
#define PL_SIZE     14      //Payload packet size, must be same for both boards

void setup()
{
  Serial.begin(115200);
  nrf.init(); //initialize the pin modes and pull-ups
  nrf.configure((byte *)LOCAL_ADDR, (byte *)REMOTE_ADDR, RF_CHANNEL, PL_SIZE);
  nrf.stateStandbyI();
  nrf.setReg(CONFIG, 0x0F);
  delayMicroseconds(130);
  ceHi();
  Serial.println("Last known status is:");
  nrf.printSTATUS(); nrf.printCONFIG();
  Serial.println("Listening started...");
}

byte stat = B00001110;

void loop(){
  ceHi();
  //nrf.printSTATUS(nrf.getReg(STATUS));
  //nrf.printCONFIG();
  int acc[8] = {0,0,0,0,0,0,0,0};
  byte tmp = nrf.getReg(STATUS);
  /*if (stat != tmp){
    stat = tmp;*/
  if (nrf.dataReady()){
    //ceLow();
    nrf.getData((byte *) &acc, 16);
    //for (int i = 0; i < PL_SIZE; i++) Serial.print(buff[i]);
    Serial.print(acc[0]);
    Serial.print(",");
    Serial.print(acc[1]);
    Serial.print(",");
    Serial.print(acc[2]);
    Serial.print(",");
    Serial.print(acc[3]);
    Serial.print(",");
    Serial.print(acc[4]);
    Serial.print(",");
    Serial.print(acc[5]);
    Serial.print(",");
    Serial.print(acc[6]);
    Serial.println();
    //nrf.setCmd(FLUSH_RX);

    //nrf.printRFSETUP();
    //nrf.printCONFIG();
    //delayMicroseconds(1500);
    //ceHi();
  }
  //delay(1);
}
```

C.6 AWIP RF Link Test - Remote Node Role, Source Code

The desired baudrate may be selected in the *define* section. The channel number, local and remote addresses, desired CRC and ACK modes as well as data-rate and payload source are setup and may be redefined if needed in the *define* section of the code.

```
#include <Wire.h>
extern "C" {
#include "utility/twi.h"  // from Wire library, so we can do bus scanning
}
#include <AWIPRF.h>
#include <nRF24L01p.h>

#define AccID 0x19
#define ACC_START 0x20, B00111111
#define ACC_WINDOW 0x23, B00000000
#define AccX 0x29, 0x28
#define AccY 0x2B, 0x2A
#define AccZ 0x2D, 0x2C

#define GyroID B1101001
#define R22 0x16, B00011011
#define R62 0x3e, B00000001
#define Temp  0x1b, 0x1c
#define GyroX 0x1d, 0x1e
#define GyroY 0x1f, 0x20
#define GyroZ 0x21, 0x22

#define URX_ADDR   (byte *)"AWIP0"
#define UTX_ADDR   (byte *)"AWsrv"
#define RF_CHANNEL 75

#define PL_SIZE    14

void i2cSetRegister(int device, int address, int value)
{
 Wire.beginTransmission(device);
 Wire.send(address);
 Wire.send(value);
 Wire.endTransmission();
}

byte i2cGetRegister(int device, int address)
{
 Wire.beginTransmission(device);
 Wire.send(address);
 Wire.endTransmission();
 Wire.requestFrom(device, 1);
 if(Wire.available())
 return Wire.receive();
 return B00000000;
}

int i2cGetValue(int device, int addressH, int addressL)
{
 int i=((unsigned int)(i2cGetRegister(device, addressH))<<8) +
i2cGetRegister(device, addressL);
 return i;
}

boolean sendd(byte* data, byte psize){
  //byte stat = nrf.getReg(STATUS);
  //if ((stat & ((1 << TX_DS)  | (1 << MAX_RT)))) return false;

  ceLow();
  nrf.setReg(CONFIG, 0x0e);//?
  nrf.setCmd(FLUSH_TX);
  nrf.setData(data, psize);
```

```
  ceHi();
  delayMicroseconds(10);
  ceLow();
}

void setup()
{
  Serial.begin(1000000);
  nrf.init();
  nrf.configure(URX_ADDR, UTX_ADDR, RF_CHANNEL, PL_SIZE);
  nrf.stateStandbyI();

  Wire.begin();
  i2cSetRegister(AccID, 0x20, B00111111); // Initialize at 1000 Hz
  i2cSetRegister(AccID, 0x23, B00000000); // From -6g to +6g. g=5461.3f;
  i2cSetRegister(GyroID, R62);
  i2cSetRegister(GyroID, R22);
}

byte stat = B00001110;
int i = 0;
void loop(){
  unsigned long time = millis();
  //int acc[3] = {i,i*10,i*100};
  int x,y,z,i,j,k,t;

  //i++;
  //x = i;
  //if (i > 100) i = 0;
  x=i2cGetValue(AccID, AccX);
  y=i2cGetValue(AccID, AccY);
  z=i2cGetValue(AccID, AccZ);
  i=i2cGetValue(GyroID, GyroX);
  j=i2cGetValue(GyroID, GyroY);
  k=i2cGetValue(GyroID, GyroZ);
  t=i2cGetValue(GyroID, Temp);
  //delay(10);
  //
  int acc[7] = {x,y,z,i,j,k,t};

/*  Serial.print(acc[0]);
    Serial.print(",");
    Serial.print(acc[1]);
    Serial.print(",");
    Serial.print(acc[2]);
    Serial.println();
  */
  sendd((byte *)&acc, 14);
  while (!((nrf.getReg(STATUS) & ((1 << TX_DS)  | (1 << MAX_RT)))))
  {}
  //nrf.printSTATUS();
  //ceLow();
  nrf.setReg(STATUS,(1 << TX_DS) | (1 << MAX_RT));
  //nrf.printSTATUS();
  //nrf.setReg(CONFIG, 0x0F);
  //delay(2);
  //ceHi();*/
  Serial.println(millis()-time);
}
```

C.7 AWIP RF Link Library, Source Code

In order to use files mentioned below correctly with the *Arduino* environment it is necessary to create a folder with them called *AWIPRF* and then locate it in the *libraries* folder of the *Arduino*. The library consists of the following files:

AWIPRF.h, the header file

```
#ifndef _AWIPRF_H_
#define _AWIPRF_H_

#include <WProgram.h>
#include "nRF24L01p.h"

#define pCE     10
#define pCSN    9

#define SOFT_MOSI 6
#define SOFT_MISO 5
#define SOFT_SCK  7

#define ceHi() { digitalWrite(pCE, HIGH); }
#define ceLow() { digitalWrite(pCE, LOW); }
#define csnLow() { digitalWrite(pCSN, LOW); }
#define csnHi() { digitalWrite(pCSN, HIGH); }

#define AWIP_CONFIG ((1<<EN_CRC) | (0<<CRCO) )

class AWIP_nRF24L01 {
    public:
        AWIP_nRF24L01(){};
        void init();

        //BASIC FUNCTIONS
        uint8_t setCmd(uint8_t reg);
        uint8_t setReg(uint8_t reg, uint8_t value);
        uint8_t setReg(uint8_t reg, uint8_t * value, uint8_t len);
        uint8_t getReg(uint8_t reg);
        uint8_t getReg(uint8_t reg, uint8_t * value, uint8_t len);

        //ADVANCED, TESTING
        uint8_t tx_payload(uint8_t * data, uint8_t len, boolean transmit);
        uint8_t setData(uint8_t * data, uint8_t len);
        uint8_t getData(uint8_t * data, uint8_t len);
        bool dataReady();
        void configure(uint8_t * myaddr, uint8_t * addr, uint8_t channel, uint8_t pl_size);
        void stateStandbyI();


    private:
        //LOW LEVEL
        uint8_t softSpiSend(uint8_t data);
        void transferSync(uint8_t *dataout,uint8_t *datain,uint8_t len);
        void transmitSync(uint8_t *dataout,uint8_t len);
};

extern AWIP_nRF24L01 nrf;

#endif /* _AWIPRF_H_ */
```

AWIPRF.cpp,the source file

```
/*
 BLA BLA BLA
*/
#include "AWIPRF.h"

AWIP_nRF24L01 nrf = AWIP_nRF24L01();

void AWIP_nRF24L01::init()
{
    pinMode(SOFT_MOSI, OUTPUT);
    pinMode(SOFT_MISO, INPUT);
    pinMode(SOFT_SCK, OUTPUT);
    digitalWrite(SOFT_MOSI,LOW);
    digitalWrite(SOFT_SCK,LOW);
    pinMode(pCE,OUTPUT);
    pinMode(pCSN,OUTPUT);
    ceLow();
    csnHi();
}

/*↩
    ============================================================================================↩

    LOW LEVEL
============================================================================================↩
    */
uint8_t AWIP_nRF24L01::softSpiSend(uint8_t data)
{
  uint8_t buff = 0;
  for (int i = 7; i>=0; i--)
  {
    digitalWrite(SOFT_MOSI, (data>>i&1));
    digitalWrite(SOFT_SCK,HIGH);
    buff =buff | ((digitalRead(SOFT_MISO)&1)<<i);
    digitalWrite(SOFT_SCK,LOW);
  }
  return buff;
}

void AWIP_nRF24L01::transferSync(uint8_t *dataout,uint8_t *datain,uint8_t len){
    uint8_t i;
    for(i = 0;i < len;i++){
        datain[i] = softSpiSend(dataout[i]);
    }
}

void AWIP_nRF24L01::transmitSync(uint8_t *dataout,uint8_t len){
    uint8_t i;
    for(i = 0;i < len;i++){
        softSpiSend(dataout[i]);
    }
}

/*↩
    ============================================================================================↩

    BASIC FUNCTIONS
============================================================================================↩
    */
uint8_t AWIP_nRF24L01::setCmd(uint8_t reg)
{
    csnLow();
    status = softSpiSend(reg);
    csnHi();
    return status;
}

uint8_t AWIP_nRF24L01::setReg(uint8_t reg, uint8_t value)
{
```

```
    csnLow();
    status = softSpiSend(W_REGISTER | (REGISTER_MASK & reg));
    softSpiSend(value);
    csnHi();
    return status;
}

uint8_t AWIP_nRF24L01::setReg(uint8_t reg, uint8_t * value, uint8_t len)
{
    uint8_t stat = 0;
    csnLow();
    status = softSpiSend(W_REGISTER | (REGISTER_MASK & reg));
    transmitSync(value,len);
    csnHi();
    return status;
}

uint8_t AWIP_nRF24L01::getReg(uint8_t reg)
{
    uint8_t stat = 0;
    csnLow();
    status = softSpiSend(R_REGISTER | (REGISTER_MASK & reg));
    stat = softSpiSend(reg);
    csnHi();
    return stat;
}

uint8_t AWIP_nRF24L01::getReg(uint8_t reg, uint8_t * value, uint8_t len)
{
    uint8_t stat = 0;
    csnLow();
    status = softSpiSend(R_REGISTER | (REGISTER_MASK & reg));
    transferSync(value,value,len);
    csnHi();
    return status;
}

/*↩
    ================================================================================================↩

    ADVANCED, TESTING
================================================================================================↩
    */

uint8_t AWIP_nRF24L01::setData(uint8_t * data, uint8_t len)
{
    csnLow();
    status = softSpiSend(W_TX_PAYLOAD);
    transmitSync(data, len);
    csnHi();
}

uint8_t AWIP_nRF24L01::getData(uint8_t * data, uint8_t len)
{
    csnLow();
    status = softSpiSend(R_RX_PAYLOAD);
    transferSync(data, data, len);
    csnHi();
    setReg(STATUS, (1<<RX_DR));
}

bool AWIP_nRF24L01::dataReady()
{
    byte tmp = getReg(STATUS);
    if ( status & (1 << RX_DR) ) return 1;
    byte fifoStatus = getReg(FIFO_STATUS);
    return !(fifoStatus & (1 << RX_EMPTY));
}

void AWIP_nRF24L01::configure(uint8_t * myaddr, uint8_t * addr, uint8_t channel, uint8_t pl_size↩
    )
{
    ceLow();
```

```
    setReg(RX_ADDR_P0, addr, 5);
    setReg(RX_ADDR_P1, myaddr, 5);
    setReg(TX_ADDR,    addr, 5);
    setReg(EN_AA,      0x03); //auto_ack enabled for both rx pipes 0 and 1
    setReg(EN_RXADDR,  0x03); //both pipes P0 and P1 are enabled to recieve
    setReg(RX_PW_P0, pl_size);
    setReg(RX_PW_P1, pl_size);
    setReg(SETUP_RETR, 0x1A);
    setReg(RF_CH,      channel);
    setReg(RF_SETUP,   0x07);
}

void AWIP_nRF24L01::stateStandbyI()
{
    ceLow();
    setReg(CONFIG,     0x0E);
    delayMicroseconds(130);
}

uint8_t AWIP_nRF24L01::tx_payload(uint8_t * data, uint8_t len, boolean transmit)
{
    csnLow();
    status = softSpiSend(W_TX_PAYLOAD);
    transmitSync(data, len);
    csnHi();

    if(transmit == true){
        ceHi();
        delayMicroseconds(100);
        ceLow();
    }
    return status;
}
```

$nRF24L01p.h$, the file that contains device-specific constants

```
/* Memory Map */
#define CONFIG      0x00
#define EN_AA       0x01
#define EN_RXADDR   0x02
#define SETUP_AW    0x03
#define SETUP_RETR  0x04
#define RF_CH       0x05
#define RF_SETUP    0x06
#define STATUS      0x07
#define OBSERVE_TX  0x08
#define CD          0x09
#define RX_ADDR_P0  0x0A
#define RX_ADDR_P1  0x0B
#define RX_ADDR_P2  0x0C
#define RX_ADDR_P3  0x0D
#define RX_ADDR_P4  0x0E
#define RX_ADDR_P5  0x0F
#define TX_ADDR     0x10
#define RX_PW_P0    0x11
#define RX_PW_P1    0x12
#define RX_PW_P2    0x13
#define RX_PW_P3    0x14
#define RX_PW_P4    0x15
#define RX_PW_P5    0x16
#define FIFO_STATUS 0x17

/* Bit Mnemonics */
#define MASK_RX_DR  6
#define MASK_TX_DS  5
#define MASK_MAX_RT 4
#define EN_CRC      3
#define CRCO        2
#define PWR_UP      1
#define PRIM_RX     0
```

```
#define ENAA_P5         5
#define ENAA_P4         4
#define ENAA_P3         3
#define ENAA_P2         2
#define ENAA_P1         1
#define ENAA_P0         0
#define ERX_P5          5
#define ERX_P4          4
#define ERX_P3          3
#define ERX_P2          2
#define ERX_P1          1
#define ERX_P0          0
#define AW              0
#define ARD             4
#define ARC             0
#define PLL_LOCK        4
#define RF_DR           3
#define RF_PWR          1
#define LNA_HCURR       0
#define RX_DR           6
#define TX_DS           5
#define MAX_RT          4
#define RX_P_NO         1
#define TX_FULL         0
#define PLOS_CNT        4
#define ARC_CNT         0
#define TX_REUSE        6
#define FIFO_FULL       5
#define TX_EMPTY        4
#define RX_FULL         1
#define RX_EMPTY        0

/* Instruction Mnemonics */
#define R_REGISTER      0x00
#define W_REGISTER      0x20
#define REGISTER_MASK   0x1F
#define R_RX_PAYLOAD    0x61
#define W_TX_PAYLOAD    0xA0
#define FLUSH_TX        0xE1
#define FLUSH_RX        0xE2
#define REUSE_TX_PL     0xE3
#define NOP             0xFF
```

C.8 AWIP RF Link Test with GPS payload, RX role

The desired baudrate may be selected in the *define* section. The channel number, local and remote addresses, desired CRC and ACK modes as well as data-rate and payload type are setup and may be redefined if needed in the *define* section of the code.

```
#include <AWIPRF.h>
#include <nRF24L01p.h>

#define LOCAL_ADDR  "AWsrv" //must be 5 bytes
#define REMOTE_ADDR "AWIP0" //target addr, must be 5 bytes
#define RF_CHANNEL  73      // Freq = 2400 + RF_CHANNEL and always < 2525, MHz
#define PL_SIZE     32      //Payload packet size, must be same for both boards

void setup()
{
  Serial.begin(115200);
  nrf.init(); //initialize the pin modes and pull-ups
  nrf.configure((byte *)LOCAL_ADDR, (byte *)REMOTE_ADDR, RF_CHANNEL, PL_SIZE);


  nrf.setReg(EN_AA, 0x00);
  nrf.setReg(SETUP_RETR, 0x00);
  nrf.setReg(RF_SETUP, B00100111);



  nrf.stateStandbyI();
  nrf.setReg(CONFIG, 0x0F);
  delayMicroseconds(130);
  ceHi();
  Serial.println("Last known status is:");
  nrf.printSTATUS(); nrf.printCONFIG();
  Serial.println("Listening started...");
}

byte stat = B00001110;

void loop(){
//  ceHi();
  char buff[PL_SIZE];
  if (nrf.dataReady())
  {
    //ceLow();
    nrf.getData((byte *) &buff, PL_SIZE);
    //for (int i = 0; i < PL_SIZE; i++) Serial.print(buff[i]);
    for (byte i = 0; i < PL_SIZE; i++) Serial.print(buff[i], BYTE);
  }
}
```

C.9 AWIP RF Link Test with GPS payload, TX role

The desired baudrate may be selected in the *define* section. The channel number, local and remote addresses, desired CRC and ACK modes as well as data-rate and payload source are setup and may be redefined if needed in the *define* section of the code.

```
#include <AWIPRF.h>
#include <nRF24L01p.h>

#define URX_ADDR    (byte *)"AWIP0"
```

```
#define UTX_ADDR    (byte *)"AWsrv"
#define RF_CHANNEL 73

#define PL_SIZE     32


boolean sendd(byte* data, byte psize){
  //byte stat = nrf.getReg(STATUS);
  //if ((stat & ((1 << TX_DS)  | (1 << MAX_RT)))) return false;

  ceLow();
  nrf.setReg(CONFIG, 0x0e);//?
  nrf.setCmd(FLUSH_TX);
  nrf.setData(data, psize);
  ceHi();
  delayMicroseconds(10);
  ceLow();
}

byte buff[PL_SIZE];

void setup()
{
  Serial.begin(4800);
  nrf.init();
  nrf.configure(URX_ADDR, UTX_ADDR, RF_CHANNEL, PL_SIZE);

  nrf.setReg(EN_AA, 0x00);
  nrf.setReg(SETUP_RETR, 0x00);
  nrf.setReg(RF_SETUP, B00100111);

  nrf.stateStandbyI();

 /* for (int i=0;i<PL_SIZE;i++){       // Initialize a buffer for received data
     buff[i]=' ';
   } */
}

byte stat = B00001110;


void loop(){
  int count = 0;
  while (count < PL_SIZE)
  {
    if (Serial.available() > 0)
    {
      buff[count] = byte(Serial.read());
      mcount++;
    }
  }

  sendd((byte *)&buff, PL_SIZE);
  while (!((nrf.getReg(STATUS) & ((1 << TX_DS)  | (1 << MAX_RT)))))
  {}
  nrf.setReg(STATUS,(1 << TX_DS) | (1 << MAX_RT));
}
```