Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Institute of Space Sciences

TREBALL FINAL DE GRAU

TFG TITLE: Disseny i desenvolupament del hardware del CubeSat Testbed Astro-01

DEGREE: Grau en Enginyeria d'Aeronavegació

AUTHOR: Ibrahima Mbaye

ADVISORS: Jordi Gutiérrez Cabello
Carles Sierra Roig

DATA: September 06, 2018

Títol: Disseny i desenvolupament del hardware del CubeSat Testbed Astr-01

Autor: Ibrahima Mbaye

Director: Jordi Gutiérrez Cabello
Carles Sierra Roig

Data: September 06, 2018

Resum

El projecte es basa a dissenyar i desenvolupar l'electrònica del Testbed Astr-01, un globus estratosfèric capaç d'embarcar experiments científics i comunicar les dades del seu entorn recopilades pels sensors i els experiments d'a bord a una estació terrestre. El sistema estarà format per diferents subsistemes cadascú amb una tasca determinada.
Durant les pràctiques d'empresa a l'Institut d'Estudis Especials es va estudiar el funcionament d'alguns models i prototips dels sistemes desenvolupats en aquest document. Aquest document fa referència a les millores proposades al final d'aquestes pràctiques.
L'Astro-01 durant tota la missió recopilarà i processarà les dades del payload i de telemetria i les guardarà en una memòria SD. Les dades de telemetria i algunes dades importants del experiment seran transmeses. El microprocessador gestionarà les dades i mesures científiques i les enviarà al sistema de comunicació que les transmetrà en paquets radio a una estació terrestre. El subsistema de comunicació incorpora, un receptor GPS, un transmissor de baixa potència i un connector per fer funcionar un Buzzer i un StrobeLight. La senyal digital transmesa està primer codificada fent servir el protocol AX.25 i després modulada en AFSK.
El subsistema elèctric és l'encarregat de subministrar potencia a tots els altres subsistemes del Globus. En aquesta fase del projecte s'han tingut en compte diverses fonts d'energia, i la més adient és una bateria amb carrega suficient per la durada del vol i la posterior recuperació de l'equip. L'anomenat placa d'expansió, és el subsistema dissenyat per connectar tots els sensors i experiments científics al microprocessador; aquesta té una capacitat de setze sensors digitals o dispositius que es poden comunicar al microprocessador fent servir el bus de comunicació digital en sèrie I2C i vuit dispositius analògics. S'han creat plaques d'adaptació per adaptar alguns components com el microprocessador principal, la Razor IMU, i els busos de comunicació.
Durant el test de l'electrònica dut a terme després d'ensamblar tots els subsistemes, la majoria d'objectius s'han assolit, tot i trobar-nos amb obstacles, i els que no s'han assolit ha sigut per falta de temps i la prematura renuncia de l'enginyer de software. Es proposen futurs tests i actualitzacions del hardware com a feina futura.
Al final del document es resumeix el procés de desenvolupament proposant algunes millores pel futur. A l'annex, s'ha afegit un manual per l'usuari a l'hora de configurar l'electrònica, així com els recursos necessaris per millorar la comprensió del sistema i la feina feta.

Title: Disseny i desenvolupament del hardware del CubeSat Testbed Astr-01

Author: Ibrahima Mbaye

Director: Jordi Gutiérrez Cabello
Carles Sierra Roig

Date: September 07, 2018

Overview

The Objective of this project is to design and develop the Hardware Testbed Astro-01, an unmanned stratospheric balloon outfitted with scientific experiments and able to exchange such as scientific data, Housekeeping data and in some cases, commands to modify some states with the ground stations. The system consists into different, each with its own task.
During the Internship to the Institute of Space Sciences, we studied several prototypes of the subsystems developed in this document and their feasibility. This document takes in account the enhancements proposed at the end of the Internship and future work.

Scientist data and housekeeping data will be recollected during the whole flight and delivered to the microprocessor, the brain of the hardware. These data will be processed and communicated to the communication subsystem. The communication subsystem which incorporates a low power transmitter sends the radio packet to the ground stations. The data sent is first encoded using AX.25 protocol and then modulated in AFSK.

The electrical subsystem is responsible for providing power to all other subsystems of the stratospheric balloon. In this Unit, we took in account several Energy sources. However, from a given input voltage the EPS is able to fix the output voltage into three defined values that remain constant and stable. The expansion Board is designed to connect all the peripherals to the MCU. It can connect up to fifteen digital peripherals and ten analog devices to the MCU. Two adaptation boards were designed to adapt the peripherals to the hardware as the MCU, the RAZOR IMU, and the communication serial bus.
While testing the hardware, we found some unexpected results. Most of our objectives were achieved, even if there were some issues and unexpected cases.
To the annex is added a manual user for the hardware setting. And some improvements are proposed looking forward

For everyone who has believed in me

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Since the beginning of space-era, small satellites were the most sought investment of the countries and companies with a limited budgets or a little experience in space technology. The design and construction of satellites involve many aspects of engineering, physics and computer science and usually have a very high cost.

Dr. Jordi Puig-Suari of California Polytechnic State University San Luis Obispo (Cal Poly) and Dr. Twiggs of Stanford University developed for a class of picosatellites and established the CubeSat standard in 1999 in order to reduce the cost and development time which ease the access to space for students and researchers. [1]

Space exploration and research is one of the most prestigious undertaking within Aerospace engineering, furthermore due the high cost of launching, designing and constructing satellites many students usually have not the opportunity to experience with space mission designing before starting to work in a space agency.

The Objective of this project is to design and develop a hardware Testbed according to the CubeSat standard. This project was born from the interest of the Institute of Space Science (ICE) to acquire a Know-How of small satellites design. Nowadays the stratospheric balloons are widely used by scientist and technologist researchers and students to perform in situ scientific experiments, for example studying the particles of the atmosphere and the chemical reaction that occurs to the stratosphere. In fact, the era of scientific ballooning started in 1783 when the French scientist Jacques Charles and the Robert brothers lifted of a hydrogen balloon *Charlière* from Paris [2]. This Testbed platform will aim the Institute of Space Sciences to outfit a variety of scientific experiments. The ease of creating an operational stratospheric scientist balloon using available technology and electronics can offer us an experience of mission planning and space platform design and building to the stratosphere.

The objective of this project is to design and develop several structural subsystems for an unmanned stratospheric balloon, including the mechanic and the structural part. Therefore, the project was initially divided into three blocks: the software block carried by an informatics engineering student who had to develop the on board software; and I had to develop the others ones, the hardware and the structural blocks. Since the informatics student left the project, the objectives were reduced to the hardware design and development.

This document is divided in four parts: the it starts with the settlement of the objectives, requirements and constraints within the first chapter. In the second chapter, we studied the proposals solutions from the ICE advanced engineer unit and also study some baseline solutions arguing their design and feasibility. The third and fourth chapter embodies all the topics related to the design and development of the Hardware. and in the Last chapter we discuss the results and the unforeseen cases we had. This structure reflects the chronological order of the Testbed building.

One of the functionality of the Astro-01 is to communicate with the ground stations. Communication includes an information exchange such as scientific data, Housekeeping data and in some cases, commands to modify some states. Therefore, it will be necessary to design a communication subsystem.

As all electronics devices, a sufficient and stable energy is necessary for a small satellite to work. The Electrical power is for instance key element and fundamental for the platform hardware such as the communication subsystem and the microcontrollers and others components to operate.

The objective of this part of the project is to design a subsystem that allows the integration and adaption of any sensors and scientific experiment to the platform, the payload with others words. For that, it is primordial to design a board that allows to outfit the Testbed Astro01 with a variety of scientific instruments.

OBJECTIVES

The diagram shown below is the first baseline structure of the hardware system. The microprocessor is the brain of the hardware. The payload, the housekeeping sensors and the communication system are connected to the MCU through the I^2C serial buses. All the measurements are stored in a subsystem memory to avoid data loss in case of communication interruption or others unexpected cases that can occur during the mission. The saved data also can be useful for the post flight data analysis. The MCU can communicate with COM system and the memory requesting our sending some data through the serial, therefore the connection between them is represented by a bidirectional arrow.

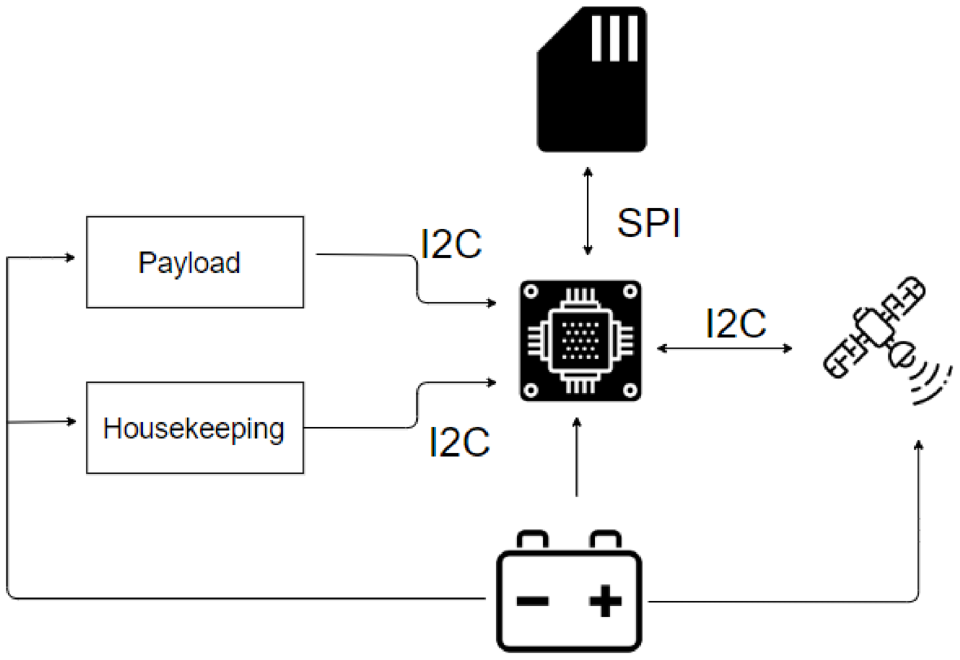


Figure 0.1 Baseline Structural Solution

With this first approach, it seems convenient to divide the whole system into different subsystems. An Electrical subsystem for the power control and distribution. The communication subsystem in charge of the ground-air link. And finally an expansion board to connect all the subsystems to the MCU. In the next chapters the content and how they are connected will be discussed

0.1. Electrical Power Subsystem

The purpose of designing an electrical power subsystem (EPS) is to produce and distribute power to all the subsystems that need it. Obviously, the design requirements will include some details concerning the quantity and the type of power supplied and the power needed by the others subsystems.

The requirements in the file provided by Cal Poly imposes that no electronics shall be active during launch to prevent any interference. But in this first approach as won't transfer control commands from ground to space to turn on the power subsystem, the EPS design won't take in account this requirement.

0.2. Communication Subsystem

The communication subsystem provides a link between the ground station and the space platform. This subsystem should have three primary functions:

- Transmission of GPS data
- Download telemetry and payload scientific data to a ground station
- Ease the payload location after the flight

The objective is to redesign and suit the trackuino to the project needs. The trackuino is an Arduino shield combining a GPS receiver with a low-power radio transmitter. It drives also a buzzer to support to ease payload location after the flight.
As studied during the internship at the Institute of Space Science, the trackuino is the communication system that entirely complies with the requirements of the project using the Automatic Packet Radio System to the frequency of 144,8 MHz in compliance with the IARU conference 2002. This part of the project consists in designing and improving some features of the Open-source APRS Based of Arduino platform to suit them to the project's needs and purposes.

0.3. Serial Bus Data Platform – Expansion Board

The serial data buses are almost used in all satellites for the communication between subsystems. Nowadays the I2C serial data bus is the most used in nanosatellites missions due to its low power consumption. For that, it is primordial to design a board that allows to outfit the Astro01 with a variety of scientific instruments, but it is important to note that nanosatellites typically support one or two instruments due to the amount of power available. This board aims to connect the scientific instruments to the microcontroller.

The Institute of Space Science

The Institute of Space Science belongs to the CSIC, the Spanish National Research Council, it is the largest public institution dedicated to research in Spain and the third largest in Europe. The Institute articulates the CSIC participation in the confederation of independent institutes affiliated with the non-for-profit foundation IEEC (“Institut d’Estudis Espacials de Catalunya”).

The Institute of Space Sciences is interested in developing both theoretical models as well as technologies to scrutinize their subject study being it a planet, a galaxy, star or the whole universe and also to use space technologies to better understand more about our own Earth. The Institute experienced a growth of considerable importance in staff and of course in papers, which situate it among the best Institutes of scientific and technology research in Europe.

Nowadays the Institute participate in more than a dozen missions and ground-based experiments with significant participation in mission like LISA Pathfinder and LISA for a detection of gravitational waves, the telescope ARIEL (Atmospheric Remote-sensing Infrared Exoplanet Large) etc. [3]

CHAPTER 1. REQUIREMENTS AND CONSTRAINTS

1.1. Communications Subsystem

1.1.1. The International Telecommunication Union

The ITU (International Telecommunication Union) is the agency of the United Nations in charge of coordinating telecommunication operations and services throughout the world. *Our mission is to ensure the rational, equitable, efficient and economical use of the radio-frequency spectrum by all radiocommunication services, including those using satellite orbits, and to carry out studies and approve Recommendations on radiocommunication matters [4].*

The main objective of the ITU is to ensure free interference operations for this purpose the organization implemented the Radio Regulations and Regional Agreements. The Radio Regulations incorporates the decisions of the World Radiocommunication Conferences, it was established by the ITU member's states and contain the main principles and settle down the specific regulations like the frequency spectrum allocations to different radiocommunication services [5].

1.1.2. The Current Regulations and Rules

In the Radio Regulations are collected all the information needed to use the frequencies of the ITU RR. In the article Nº. 9 it is explained the procedure for effecting coordination with or obtaining agreement of the administrations, it contains the main regulatory procedures applicable to the space systems. It exposes that states members of the ITU should endeavour to limit the of frequencies and the spectrum used to the minimum essential to provide in a satisfactory manner the necessary services.

Nanosatellites and picosatellites using frequency assignments, with those operating in the amateur-satellite service, are into the category of a space radiocommunication service and are required to be notified under RR Article 11. Prior to notification, in accordance with RR No. 9.1, the notifying administration of such networks is required to send to the BR (Radiocommunication Bureau) the advance publication information (API) not earlier than seven years and preferably not later than two years before the date of bringing into use. This last requirement is stated into the 11.2. [6]

We could discuss more about the topic of frequencies assignment but it is not rocket science to notice that there is a lot of procedures to carry out. Also there is an exception in the amateur service in the amateur-satellite service, so using amateur service like Ham radio.

By the way for the time and the budget we have, the use of the amateur frequency band is practically necessary. To use the band allocated to amateur satellite service: on the one hand a national radio amateur license is needed and on the other hand, the rules and regulations of the RR articles 1.56 and 1.57 shall be obeyed.

However, the use of amateur or amateur satellite service spectrum, under the amateur service, is only appropriate if the definition of the amateur service (RR No. 1.56) is met:

“A radiocommunication service for the purpose of self-training, intercommunication and technical investigations carried out by amateurs, that is, duly authorized persons interested in radio technique solely with a personal aim and without pecuniary interest.” [7]

Anyways, for the space activities it is recommended by the European Space Agency to check first the governmental treaties ratified by the state for space activities, space and satellites missions, contact with the appropriate telecommunication entity.

1.1.3. Ham Radio License

The international Amateur Radio Union is a federation of national associations of certified radio amateurs. The IARU divided the world* in three different regions according the structure of the International Telecommunications Union.

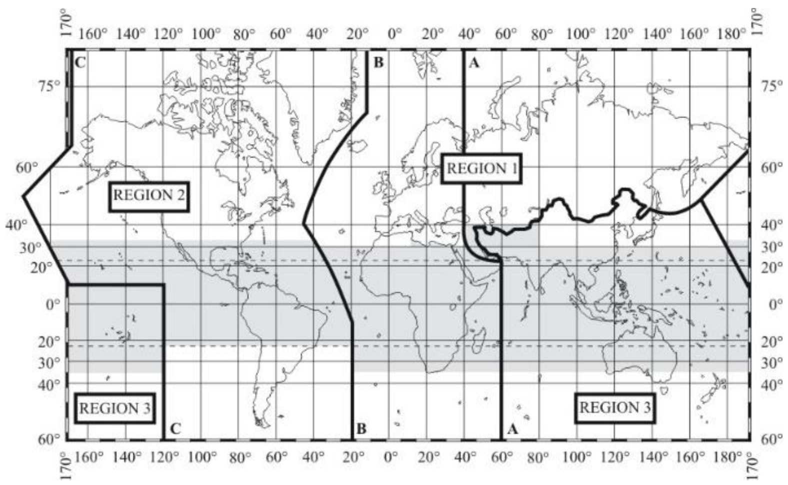


Figure 1. 1: The IARU Regions

Some frequencies are reserved by the IARU in the 2m VHF band for satellite telemetry. This band is fully used throughout the world for short-range communications including the use of repeaters*. The 2m band is one of the most heavily used for satellites amateur operations. The band 145.806 -146MHz is exclusively used for satellite communication in the first Region. [8]

In Spain the CNAF (Cuadro Nacional de Atribución de frequencia) is the legal instrument, under the authority of the ministry of industry for the allocation of the different frequencies bands for the different radio communications services (fix services, fix services by satellite, Radio navigation, radiolocation, amateur's services, space operations etc..) from 8.3kHz to 3000GHz according to the ITU structure. [9]

1.1.4. Position Reporting

Some limitations are found where consulting the trackuino components. One of them is the GPS used for the position reporting. There is some restrictions or operational limits in GPS technology, like the COCOM limits. In GPS, the term COCOM limit refers to the limit placed on GPS tracking devices that shuts off when the dispositive is moving at an altitude higher than 18Km to a velocity higher than 1000 knots (515 m/s). The COCOM limits was intended to prevent the use of GPS in intercontinental ballistic missile applications. They are some manufacturers enable tracking when both are exceeded, while others disable tracking when one of the limit is reached.

[10] The SparkFun Venus GPS (The Venus638FLPx) is a high performance and low cost GPS receiver. It has a low power consumption, high sensitivity and contains all the necessary components of a GPS receiver, this module also minimizes RF Layout design issues. The operational limits found to the datasheet, shows that the GPS receiver fulfil the COCOM limits, but news the good is that the GPS only shuts off when ***both*** (altitude and speed) limits are reached although one of the limit will be reached, the altitude of 18km.The speed will depend of the following forces: the upward buoyant force, the gravitational pull downwards, the drag force acting and some others forces. So the value of the ascent rate of the balloon can be approximately known solving the net force, the result of adding all the forces acting (even if we know that some the parameters varies with the altitude). It is necessary to ensure that the GPS device will work during the whole flight, or at least under 18km, to be sure that the payload will be recovered otherwise this process can't be carried out.

1.2. Electrical Power Subsystem (EPS)

1.2.1. Power Consumption

Each subsystem consumes power: transmitter, GPS, payload and sensors on-board etc. The power consumption of the whole system depends of the power consumption of each component and any additional payload throughout the entire flight. The power is substantial in electronics design, power usage must be moderated and well managed. It is recommended to know the power limits of your mission to remain within them. As someone said, a nanosatellite launched with a negative power budget is "space debris". During the design the power consumption should be minimized and the component selection carefully taken in account. It is recommended to switch OFF non-essential subsystems and payloads by software [11]. As the flight will take some hours, the design doesn't contemplate the use of power generators such as solar cells or fuel cells. Instead, a battery with sufficient charge to the last hours will be implemented once the power consumption of the whole system is quantified

1.2.2. Outputs Voltage

The EPS has to be designed to power the whole system. Not all the subsystems and components are working at the same voltage. Therefore, the EPS will have different outputs voltage. Most of the components used in this platform are working at 3,3 or 5V. The MCU maybe will require more than 5V so a third higher output voltage is needed.

1.3. The Expansion Board

The Testbed Astro01 will be outfitted, together to other subsystems, with a payload and sensors to study the environment along the flight. All the sensors are connected to the microcontroller. The sensors communicate to the microcontroller their measurements to be broadcasted by radio or stored to the subsystem memory.

As discussed, there is a serial data bus that intents to allow multiple "slave" digital integrated circuits to communicate with a master device. This protocol is called Inter-Integrated Circuit, I^2C. [12] It only requires two wire and can support up to 1024 (1008) slave devices when using 10 bits addressing. Most I^2C devices are able to communicate at 100kHz or 400kHz (nowadays it supports devices from 0 kHz to 5MHz) and can support multi-master system. As we are using the mode with 8 bits addressing the serial buses can support 256 slave devices. To make the system more flexible, the board is also designed to connect analog output sensors, if they lack an I2C interface.

1.4. 9DoF Razor IMU Adaptation

The Board adaptation is designed and built to fit the 9DoF Razor IMU subsystem into the whole hardware structure as the brain or computer of the subsystem, the software part of the project has been developed by the informatics student but due to personal reasons he decided to leave. The adaptation board aims to connect the master IDC connector of the Expansion board, which connects the

I2C serial bus of all devices that can be connected, to the Astro-01 using I2C interface and Analogous devices.

The MPU-9250 9Dof sensor is connected to the Atmel SAMD21 (SAMD21G18) through the I2C serial bus, thus the idea is to connect the Master IDC connector (of the expansion board) I2C serial buses to the Razor IMU I2C serial buses. Both devices will be considered are slaves and the Atmel microprocessor as the Master. It’s important to remember that the MPU-9250 share the same I2C bus while programming the board to ensure that any additional I2C devices using the bus doesn’t share the IMU’s 7-bit address (0x68), specially made for the MPU-9250 sensor. The Razor can be powered through the EPS, thus the external power pin of the 9DoF shall be connected to an input voltage connector.

CHAPTER 2. BASELINE OF SOLUTIONS

2.1. Communications Subsystem

To implement the communication system of the payload, we have studied two options according to the project budget:

- Customise a communication system using the XTend 900 from Sparkfun House.
- Trackuino: An Arduino shield combining a GPS receiver with a low-power radio transmitter (using the Radiometrix Hx1 or the SBR MX145 transmitter).

2.1.1. The XTend 900 custom

[13] The XTend Modem packs a ton of RF punch into a small, low-power, easy-to-use, and reliable module. The XTend modem can communicate up to 40 miles and operates within the ISM 900 MHz. What makes this module to be a good candidate for our mission is his transmit power that is software adjustable from 1mW to 1W (0-30dBm). In the table below it's shown some technical characteristics of the XTend module.

Table 2.1 XTend Technical specifications

XTend 900 custom	
Range	*9600 bps - 40 Miles (LOS)* *115200 bps - 20 Miles (LOS)*
Transmit Power Output	*0 to 30 dBm*
Antenna	*RPSMA Connector – 50 Ohm*
Maximum current	*950mA*
Frequency band	*902 MHz – 928 MHz*
Modulation	*FSK*
Operating Temperature	*-40ºC to 85ºC*
Cost	*194,95 €*

No configuration is necessary for out of-the-box RF communication, this module from Sparkfun has an advanced configuration that can be implemented using simple AT or binary commands and needs a receiver sensitivity between -100dBm to -110dBm. Bellow it's shown the Internal data flow of the module.

Figure 2-03. Internal Data Flow Diagram (The five most commonly-used pin signals shown)

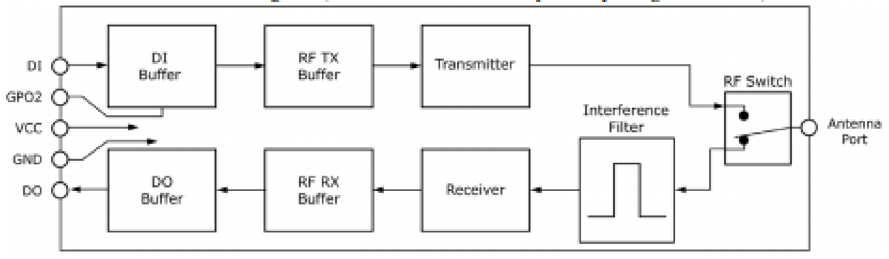
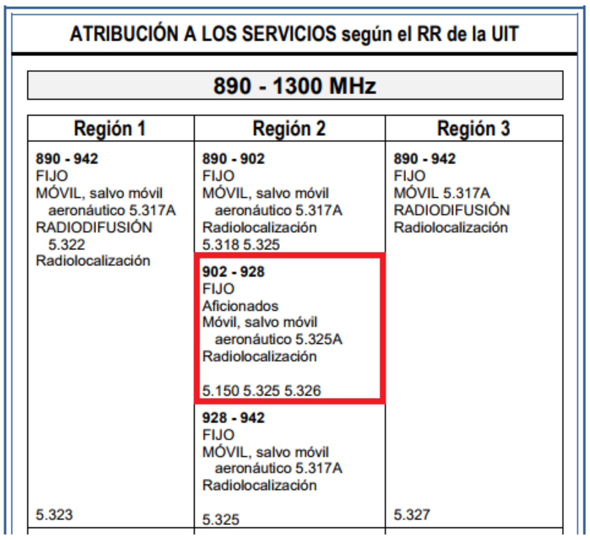


Figure 2.1 Internal Data Flow Diagram

[14] When serial data enters the module through the Data IN pin, the data is stored in the DI Buffer until it can be processed. When Transmit Mode, the module attempts to initialize a Radio Frequency connection. If the module is already receiving RF data, the serial data is stored in the module's DI Buffer. The capacity of the DI buffer is 2.1 KB. The hardware or software flow control can be implemented to prevent overflow. The Data OUT Buffer has the same capacity.
This module can be used in Spain. The inconvenient is that in the region 1 the frequency 900MHz is reserved to fix services, telephony, mobile services broadcasting and radiolocation (see the allocation to services according to the UIT RR) which makes impossible the use of the XTend 900. But it would be a good option if the launch was carried out somewhere to the region 3.



ATRIBUCIÓN A LOS SERVICIOS según el RR de la UIT

890 - 1300 MHz

Región 1	Región 2	Región 3
890 - 942 FIJO MÓVIL, salvo móvil aeronáutico 5.317A RADIODIFUSIÓN 5.322 Radiolocalización	**890 - 902** FIJO MÓVIL, salvo móvil aeronáutico 5.317A Radiolocalización 5.318 5.325	**890 - 942** FIJO MÓVIL 5.317A RADIODIFUSIÓN Radiolocalización
	902 - 928 FIJO Aficionados Móvil, salvo móvil aeronáutico 5.325A Radiolocalización 5.150 5.325 5.326	
	928 - 942 FIJO MÓVIL, salvo móvil aeronáutico 5.317A Radiolocalización	
5.323	5.325	5.327

Figure 2.2 CNAF 890 - 1300 MHz Services

2.1.2. Trackuino

The trackuino is an open-source source APRS tracker based on the Arduino platform. The APRS (Automatic Packet Reporting System) is a Radio based system for digital communications of information of immediate value in the local area. With other words: APRS is a packet communications protocol for disseminating live data to everyone on network in a real time. In Europe the frequency for the APRS is 144,8 MHz in compliance with the IARU conference of 2002 [15]. The trackuino was designed to support a Radiometrix Hx1 (300mW) transmitter and the Argentdata's MX146-8v (500mW). The Argentdata's MX146-8v (500mW) is not available anymore to the market, it was replaced by the MX145-5v. Before explaining the enhancement and how the communication system will be built it seems necessary to show a block diagram of this application.

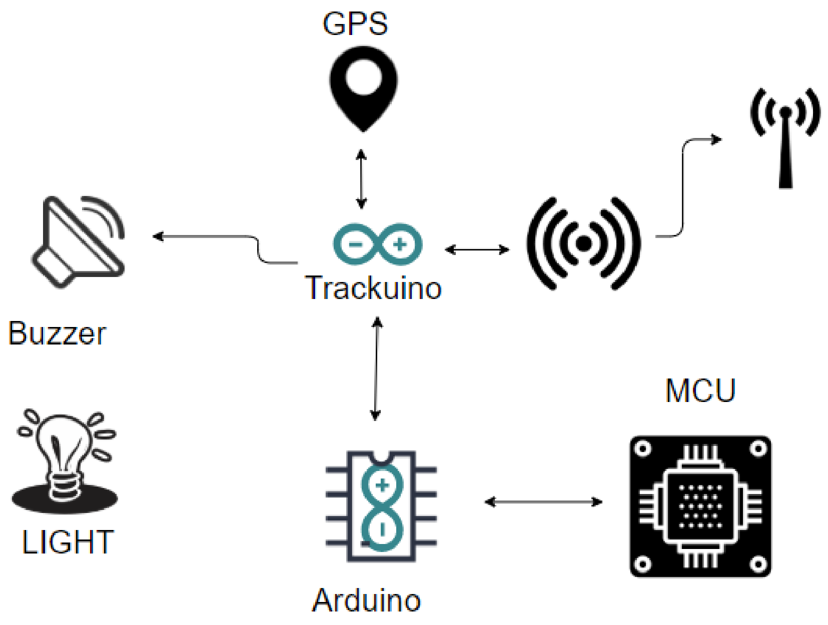


Figure 2.3 Communication Block Diagram

We definitely decide to use the trakuino. The main reason is because it's an open source and has a very low cost. The hardware can modified and suit to our project requirements. And the frequency at which it works fulfils also our project requirements. Due to the time requested by this project, as discussed, it seems that using the amateur-radio frequency will save us time instead of considering starting a process for obtaining a frequency band which requires a wide knowledge of Radio telecommunications and take several years to get the license for using a specific frequency band. The background and the know-how acquired from the previous project (Astro-01 approach) aims us to implement a communication system.

2.1.2.1. Trackuino with SBR MX145 Transmitter

The SBR MX145 is an embeddable VHF transmitter form SBR Electronics programmable from any frequency between 144 to 148MHz. The SBR MX145 have the following characteristics:

Table 2.2 SBR MX145 Technical Specifications [16]

SBR MX145	
Channel spacing	*2.5kHz*
Modulation:	*digital injection modulation*
Modulation Bandwidth:	*>20kHz*
modulation Sensitivity:	*23kHz/V (typ)*
Input impedance:	*~600Ω*
Spurious suppression:	*> 80dB (channel spacing > 10kHz, typ)*
Harmonic suppression:	*45dB*
Frequency stability:	*+/5ppm (typ)*
Furn on delay (after PTT):	*25msec (typ)*
Output power:	*> 350mW (minimum) into 50Ω*
Programming:	*SPI® and I2C® interface or 16 pre-programmed frequencies, pin selectable. 3.3V CMOS level*
Supply voltage:	*+5VDC (4.7V to 6V)*
Price:	*59,36 euros*

What makes the SBR MX145 to be the better option than the Radiometrix's HX1 is his output power that can reach 500mW. The output power is important when computing the range of the RF signal.

2.1.2.2. Trackuino with Radiometrix Hx1 Transmitter

[17] This transmitter offers a 300mW RF output VHF data link in Radiometrix SIL standard pin-out and footprint. For its low power output, it is suited for those low power applications. Below is shown a summary of the technical characteristics:

- Transmit power: 300mW (24.7dBm)
- Operating frequency: 144.390, 144.800 and 169.4125MHz
- Channel spacing: 25kHz
- Supply: 5V (regulated)

- Current consumption: 140mA nominal transmit
- Data bit rate: 3kbps or 10kbps max.
- Size: 43 x 15 x 5mm

The block diagram of the Radiometrix is also shown to the next figure:

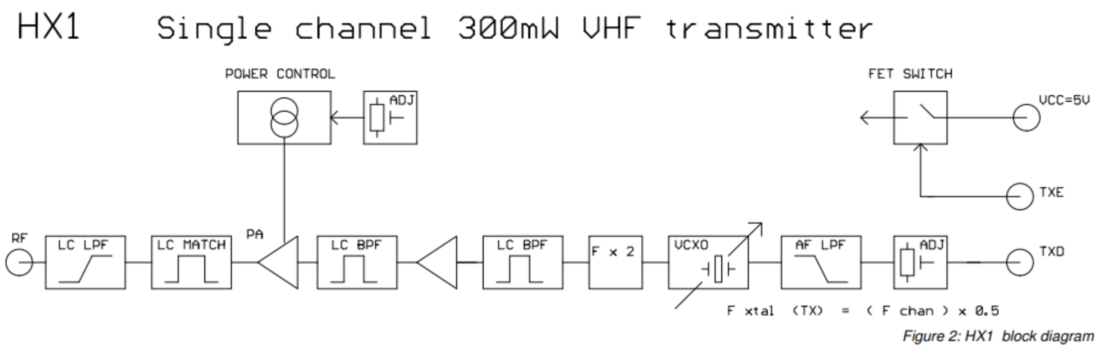

HX1 Single channel 300mW VHF transmitter
POWER CONTROL
ADJ
FET SWITCH
VCC=5V
TXE
RF
LC LPF
LC MATCH
PA
LC BPF
LC BPF
F x 2
VCXO
AF LPF
ADJ
TXD
F xtal (TX) = (F chan) x 0.5
Figure 2: HX1 block diagram

Figure 2.4 Internal Block Diagram of Hx1 300 mW Radiometrix

From the datasheet it is explained that this module consists of a frequency modulated Voltage Controlled Crystal oscillator (VCXO) feeding a frequency doubler with two stage amplifier and RF filter (as we can see in the block diagram). The frequency of oscillation of the VCXO is controlled by the input voltage. It means that the output frequency changes as the input voltage varies. The final power amplifier (PA) is pre-set to appropriate band power level.

2.1.3. Transmitter – Baseline Solution

We definitely decided to use the Radiometrix Hx1, because The SBR MX145 was out of stock. This last one has better features, beside of incorporating a receiver and a higher transmit power up to 1W it has a very low power consumption.

2.1.4. Ham Radio License

It's is forbidden to use any frequency band that requires a License. Therefore, to use the APRS, I contact with the president and the treasurer of the FEDI EA [3]. After a meeting to the Institute of Space science with the two directors of the project Astro-01 they accepted to give us a callsign in order to carry out the radio tests, as a collaboration.

2.1.5. Automatic Packet Reporting System (APRS)

The APRS is an amateur radio-based system for real time digital communications of information of immediate value in the local area [14]. The amazing thing with the APRS is when a packet is broadcasted everything else is “done”, it means that there is no need to design and implement a ground station. The packets will be transmitted to the ground receivers also called digipeaters (digital repeater), already implemented, that will automatically upload the RF signal received and retransmit them at the same frequency through a determinate path until they reach an I-Gate. An I-Gate is a type of gateway APRS station that allows the packets to cross by the RF local network to the APRS IS (APRS Internet Service) allowing to users with no APRS gear to still receive the packets, thanks to Ham radio amateur. A path is a list of intermediate stations that are supposed to digipeat the packets sent by an initiating station. This station includes the list of the intermediate stations.

2.1.5.1. AX25 PROTOCOL

For a reliable data transport data between two terminals it is primordial to define a protocol that can accept and deliver the packet data over different types of communication links. For that purpose, the AX.25 was designed to provide this service. At the link level APRS uses AX.25 protocol using Unnumbered Information frames. In the APRS protocol reference document (attached to this report) they explain the format of the packet data frames. [18]

2.1.5.2. Ground Stations

It is possible to set what kind and how many digital repeaters will be used to deliver a packet data to a specific I-GATE. when they operate with multiple frequencies they are called gateways.

As explained the digipeater receive the data packet and retransmits it through the network. When it receives a packet of data, it is stored in an internal memory before being transmitted to the same frequency. They are placed in strategical locations to cover a wide area and have a generic alias in APRS. When transmitting a radio packet there is no need to know the digipeaters callsign, but how much hop is needed (information found to the path). The station who receive the path will:

- Examine the path of the received packet.
- Decide if the packet is candidate for digipeating according its configuration information.
- Modify the packet addressing information.
- Send de modified packet.

2.1.5.3. APRS Internet Service

The APRS Internet Service or APRS-IS is the Internet-based network that interconnects radio networks throughout the world it is operated and maintained by Amateur Radio operators to provide world-wide capabilities to all the users and promote the Amateur Radio service as a whole. Some technical specifications can be found to the APRS-IS for software authors, these specifications are addressed to software designer.

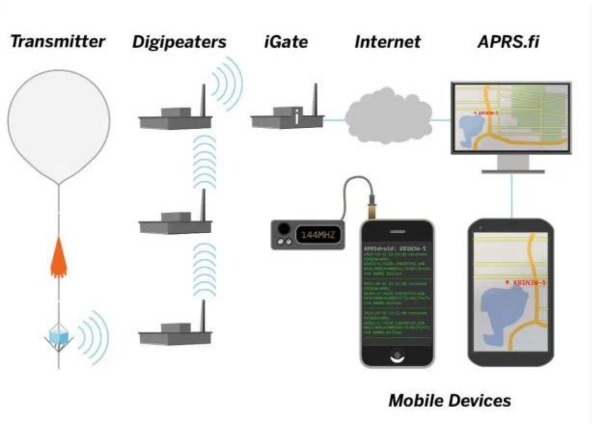


Figure 2.5 APRS Network

2.1.6. Redundancy – Mini GPS Tracker/ Mobile Phone

After analysing the risks that can occur during the flight or after, we found that there is a possibility to have a GPS failure due to the ground impact or for some reasons else. Therefore, it can be necessary or useful to outfit the electronics with a mini GPS tracker that work with a sim card. This module will send a message with the latitude and longitude of the target to ease the payload location.

The other idea was to use a mobile and set it to send the GPS coordinates by SMS every X times, the big disadvantages is the weight that can arrive to 150g, but due to the short life term of the mini GPS tracker, it might better option to use the it.

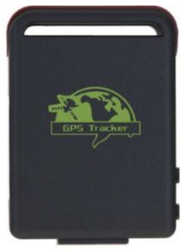


Figure 2.6 Mini GPS Tracker

Table 2 3. Mini GPS Tracker.

Dimensions	**70 x 47 x 17 mm**
Weight	38 g
Temperature	-25 ºC to 65 ºC
Velocity Limit	515 m/s
Price	19,14 €

2.2. Electrical Power Subsystem

2.2.1. Solar Energy

The main energy source in all nanosatellites and CubeSats is the sun, the power or energy limitations usually restrict nanosatellites functionality. Solar cells convert the energy found in sunlight into electricity using the photovoltaic effect, a creation of voltage and electric current through a physical and chemical phenomenon. As all electronics devices, a sufficient and stable energy is necessary for a small satellite to work. when the small satellite is in the shadow the energy shall be provided by the battery. Therefore, it’s important to ensure that the electrical power will be available for the system during the mission.

In this first phase of this project, our objective is to design and develop an EPS able to supply energy to the others subsystems. Even if we know that using battery has some limitations and also they have short life time.

2.2.2. Battery

The battery choice depends of the whole system consumption, including the payload (sensors and the scientist experiments). This part is going to be studied at the end of the project before the flight.

2.2.3. Fuel cells

Fuel cells is one of the best energy source nowadays. They produce an efficient energy with almost zero emissions compared to fossil fuel. They convert the hydrogen and oxygen into water. This conversion produces electricity. The conversion is done connecting both hydrogen and oxygen to an anode and cathode separated by an electrolyte. They never run down and don’t need to be recharged. So looking forward, the possibility of using this Energy source can be a good application for the Astro-01.

2.3. Board Interconnections

Electronics is all about interconnecting circuits and devices in order to swap information from one to another. So sharing a common communication is essential to the devices that are attempted to be connected. There a lot of communication protocols that have been defined for data exchange but we'll only study them for this project due to the project requirements and because they are commonly used nowadays. Obviously the connection of the analog devices also is discussed.

2.3.1. I2C Serial Bus

It consists of 2 signals: the *SCL* (Serial Clock) generated by the master although slave can force the clock low/ hold down the clock line until it is ready to continue communication, this is called "clock stretching", it able the salve to protect itself from being addressed quickly. The *SDA* (Serial Data). All data transfer among the devices occurs through this line. The two bus is used for communication between a microprocessor and the peripheral chips such as temperature sensors and actuators etc. I2C can support a multi-master system.

2.3.2. SPI Serial Bus

SPI (Serial Peripheral Interface) like is I2C is used to send data from devices as SD cards, shift registers, sensors to a microcontroller. It is a synchronous data bus. It's called synchronous because different lines are used for the data and the clock to keep both signal synchronized. The clock is an oscillating signal and defines when the receiver has to put data to the data line. Unlike I2C serial bus, the SPI interface only allows one master, and this one is the device that generates the serial clock and it uses to be the MCU. The data lines are two: MOSI and MISO (Master Out/ Slave In and Master In/ Slave Out). With SPI there is not a signal who tell the master the existence of data in the MISO lines, therefore the master has to knows if the slave is going to send a data answering to a master command. The last line and not less important is the SS (Slave Select) is used to wake up and receive or to send a data, normally used when there is more than one slave.

2.3.3. Analog devices

The design of the communication protocol also takes in account the analog device that produce an analog output signal. Usually this output signal is a voltage that change based on the measured variable. The Board connection will integrate connectors that aims to connect analog devices to the MCU. The microprocessor has ADC converters which convert the analog signal to a digital signal. So this value can be processed and transmitted by the trackuino with the data of the digital peripherals.

2.3.4. Interface – Baseline Solutions

For the digital serial Bus, we definitely decided to use the I2C serial because although allowing more than one master, It also require less signal lines than the SPI, in the case of having 100 slaves for instance, 100 SS lines are needed and which would be an important change in the architecture design of the hardware. And as discussed the SPI must be defined in advance, which doesn’t match with our first philosophy which was made every subsystem versatile looking forward.

CHAPTER 3. HARDWARE DESIGN

3.1. Background – Prototypes

3.1.1. Design execution

In the first phase of the Astro-01 hardware development, some prototypes were built using PCBs prototyping boards, soldering the components and routing with wires. The subsystems were designed using BlackBoard Circuit Designer which is a software open source for the design of electronic circuits for prototyping boards. To the annex are shown, as a sample, the prototype of the expansion board, the EPS and the Razor IMU adaptation board, designed with the BlackBoard Circuit Designer.

3.1.1. Prototypes conclusions

Some conclusions were made after the prototypes design:

For the EPS, we didn't prevent the capacitor from discharging to the regulator, and also were using electrolytes capacitors without checking their operational conditions (specifically the pressure and temperature).

The expansion board needs to follow the same principles of working than his prototype. But the pull-up resistor has to be well dimensioned due to the difficulties found during the prototypes working test. We added two pull up resistors and none of the slave devices could communicate with the master (Razor IMU). Reviewing the schematics of this last one we found out that the 9DoF Razor IMU already incorporates internal pull up resistors to the I2C serial buses.

The communication subsystem is outfitted by several components to provide an acoustic payload location and send the telemetry to the ground stations. We took the idea of the trackuino open source shield without any change. The original shield is outfitted with an internal and external temperature sensor. For a future work, we decided to remove both sensors (for the redundancy with some Housekeeping sensors) and add a visual payload location.

3.2. Communications Subsystems

The last version of the trackuino shield used in this project is the, version 2.2. The bill of materials is attached to this report. Apart from the transmitter from Radiometrix and the strobe light from RS, all the components was found to the Mouser distributor.

To suit the features of the Trackuino to the project needs, some modifications have been carried out:

- The internal and external sensors of temperature have been removed assigning the temperature measurement to another subsystem.
- A “Strobe” light is added to support to ease visual payload location.

Below we’ll make some specifications about the different components of the trackuino: the GPS, the transmitter used for the telemetry, and the enhancements.

In the GitHub, the leading software platform development, whose website link is annexed to this report, the firmware of the trackuino can be found. It explains the features of the trackuino in his basic set: the use of GPS allows to automatically receive the positioning of the balloon from the GPS constellation satellites. With the transceiver it is possible to carry out the telemetry of the sensors measurements and the GPS coordinates, using 1200 bauds FSK with 8-bit PWM and other features, that will be discussed later.

The shield and the bills of materials to build the hardware are also found to this repository, it contains the eagle schematic and PCB and the Gerber file. And Some advices are given for the hardware building and flashing.

Some enhancements are added to the board as, headers and jumpers to the level shifters to make the board more flexible and two decoupling capacitors also called bypass capacitors are added to suppress high frequency noise in power supply signals and additionally if supply drops its voltage (temporally) they can briefly supply power at the correct voltage. The buzzer, GPS, and strobe light test were successfully carried.

3.2.1. Adding a Strobe Light

One of the important part of this project is the payload location when the Strat01 comes back to earth. Along all the flight, the Strat01 will report some information as his positioning information (latitude and longitude), speed and course.

In fact, the process of localizing the payload can take place during the night or in low visibility conditions. Therefore, adding a strobe light will ease the payload location. “Strobe” light because it consists of a flashing light.

We will use the Arduino to control the light. The Arduino pins have a maximum current rating of 40mA but it is recommended to limit the current at 20mA to not damage the microcontroller. The output voltage that can give the Arduino pins is between 0 and 5V. A power Mosfet is connected to the microcontroller to control the led.

3.2.1.1. Power Mosfet

The power mosfet are electronics devices that aims to control high current, they have three pins: The Drain, the Source and the Gate. The main current goes between Source and drain while the control of this current is obtained applying a voltage to the Gate. This voltage is known as V_{GS}. As we are pretending to use the mosfet as a digital switcher, the mosfet will has two modes: The saturation mode (ON) and the cut-off mode (OFF).

When the mosfet is in saturation the value of the internal resistor between source and drain, R_{DS} is low thus the power the power dissipated is also low however the current that runs through the mosfet can be high. To make the mosfet in saturation mode the voltage applied to the Gate should be enough large than the mosfet gate threshold voltage. Otherwise the mosfet will work in cut-off mode.

The power mosfet with a low gate voltage are known as logic level power mosfet. When they change their logical level of control the mosfet absorbs some current that charges the internal capacitor of the gate, so a resistor is needed to control the current going through the mosfet. The value of this resistor determines the time commutation of the mosfet, while lower is this value faster is the commutation.

A resistor will be connected in series to the pin to limit any current going to the mosfet, this goes up the gate capacitance. And connect a 10K pull-down resistor between the gate and the ground to ensure that the device is off when the digital pin of the microcontroller is floating. A resistor in series of 100 Ohm would be enough. The resistors connected to the digital pin behave as a voltage divider. A voltage divider of 10 KΩ/200 Ω will reduce the voltage of:

$$V_{out} = \frac{R_5}{R_5 + R_4} V_{in} \tag{3.1}$$

$$V_{out} = \frac{10 \cdot 10^3}{10 \cdot 10^3 + 200} V_{in} = 0{,}98 \cdot V_{in} \tag{3.2}$$

Being the Voltage to the gate 98% of the voltage delivered by the microcontroller pin. This voltage drop won't have a relevant cost or affect the mosfet operation.

3.2.1.2. Eagle LED – Circuit Design

We use the eagle program to modify the eagle files of the trackuino. We had to learn to use this program, reading the manual of the software and some tutorial found to the web. Learning how to implement a circuit and designing a board.
The first step was to find the library which contains the transistors and the footprint to implement it into the PCB.

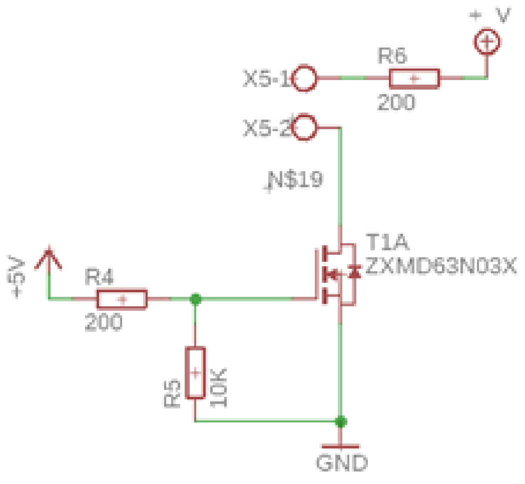


Figure 3.1 Eagle LED - Circuit Design

3.2.1.3. LED – Characteristics

Here are shown some technical parameters of the LED serie XP-C Cree XLamp of Cree manufacturer:

Table 3.1 LED Characteristics

Size	3.45 x 3.45 mm
Colour	White
Max Power	2W
Max Light	138lm
Typical Forward Voltage	3.4V @ 350mA
Max Reverse Voltage	5V
Max Junction Temperature	150 ℃
Max Drive current	500mA
Pin numbers	2
Viewing angle	115

3.2.2. Jumpers and Buffers configuration

The MCU can send some commands to the GPS when the Arduino TX pin is connected to GPS RX pin when the Jumper 5 (J5) is shorted. Shorting J5 implies that it is not possible to print any debug information out the serial port and into the Arduino IDE's console. We use jumpers because it is possible that we'll need to send some commands to the GPS.

3.2.2.1. The level shifters: IC4, IC3 & IC5

Since we're using the Trackuino Uno that can give 5V output the buffer IC4 should be soldered to do the down conversion of 5V to 3.3V for the GPS. All the buffers are connected to be sure that the board can work with both 3.3 and 5V logics.

As IC4, IC3 and IC5 downconvert the 3.3V to 5V for the radio HX1 (Radiometrix). As we'll use 5V output there is no need to solder or connect the jumpers J6 and J8, but to make the configuration of the board more flexible we'll also connect the jumpers for the same reasons as with IC4.

3.3. Electrical Power Subsystem

The EPS (Electrical Power System), which block diagram is shown below, was designed to power the whole system. A Battery of 12V output voltage is used to power the system. The EPS has three different Voltage output: 3.3V, 5V and 9V.

3.2.1. Output Voltage 3.3 – 5V

This circuit was designing to convert the 12V into 5 and 3.3V settable using jumpers. **R1**, **R3**, **R4** and **R5** are needed to set the output voltage. The output voltage value depends of the R2 and R1 and can be computed with the following equation:

$$V_{OUT} = V_{REF} \cdot \left(1 + \frac{R_2}{R_1}\right) + (I_{ADJ} \cdot R_2) \tag{3.3}$$

As we using two capacitors one for improving the ripple rejection and the other one due to the distance between the regulator and the power supply a protection diode is recommended in both cases to provide low-impedance discharge path preventing the capacitor discharge into the output regulator. The Figure 3.3 shows the schematic from of the 3.3 – 5 V output voltage.

F1 is a resettable fuse that is used to protect against overcurrent faults. The resistance of the PTC is lower than the resistance of the circuit. The PTC, as known, increases their resistance to reduce the current in the circuit to a value that can be safely carried by any elements of the circuit.

JP1 is used in order set the output voltage of the circuit. This value can be computed using the equation 1. Where:

$$V_{REF} = 1.25V,\ I_{ADJ} = 50\mu A \quad R_1 = 240\Omega \quad R_2 = \begin{cases} R_3 + R_4 \text{ for } V_{OUT} = 3.3\text{V} \\ R_3 + R_5 \text{ for } V_{OUT} = 5\text{V} \end{cases}$$

(3.4)

3.2.2. Output Voltage 9 – 5V

The same circuit is used but changing the value of the resistors.

$$V_{REF} = 1.25V,\ I_{ADJ} = 50\mu A \ ,\ R_1 = 240\Omega \quad \text{and} \quad R_2 = \begin{cases} R_7 + R_8 \text{ for } V_{OUT} = 9\text{V} \\ R_7 + R_6 \text{ for } V_{OUT} = 5\text{V} \end{cases}$$

(3.5

3.4. The Expansion Board

The expansion board is designed to connect all the sensors to the microcontroller. It is powered by the Electrical Power Subsystem through the Jack DC connectors and can run two different Voltage. The Power supply has three different output voltages (3.3, 5 and 9V). Almost all the components are working between 2 and 5V so only the output voltage of 3.3 and 5V will be connected to the expansion board. The first approach of the EB design is shown to the diagram below.

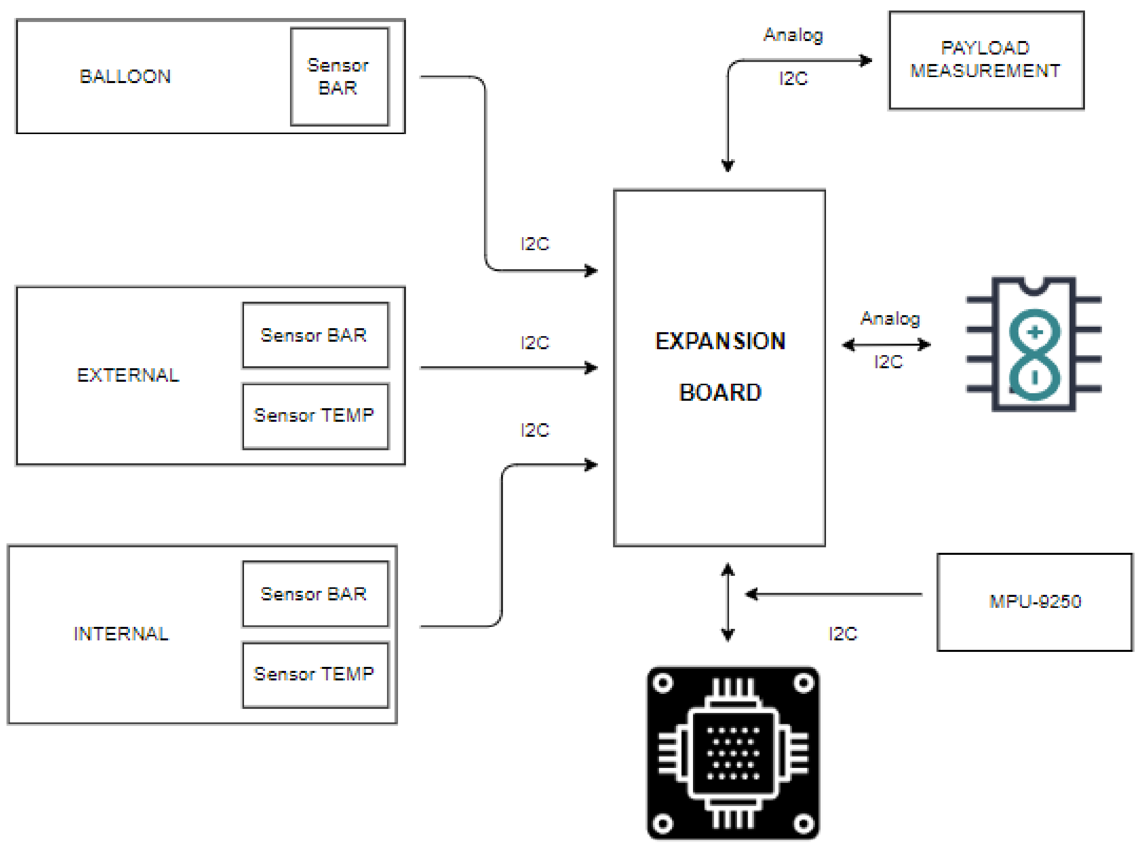


Figure 3.2 Baseline Solution – Expansion Board

3.4.1. I^2C Serial Bus – Pull-up Resistors

SCL and *SDA* are *open drain* (when talking about Mosfet) also known as open collector (when talking about BJT) what means that the master and the slave devices can put the two lines low but cannot drive them high. For these reason both lines are connected to a positive supply voltage through pull-up resistors. When a slave is using the line it drives it low so no one else can use the line.

The pull-up resistors are needed so that they can pull the line high when it is not driven low by an open drain interface.

The figure bellow shows a pull-up resistor connected to I2C bus:

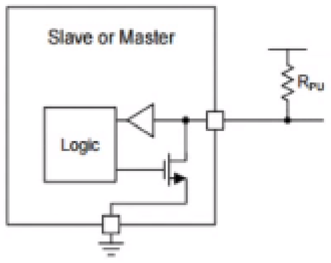


Figure 3.3 I2C Bus

The transistor pulls down the I2C bus when it turns on as shown in the figure bellow:

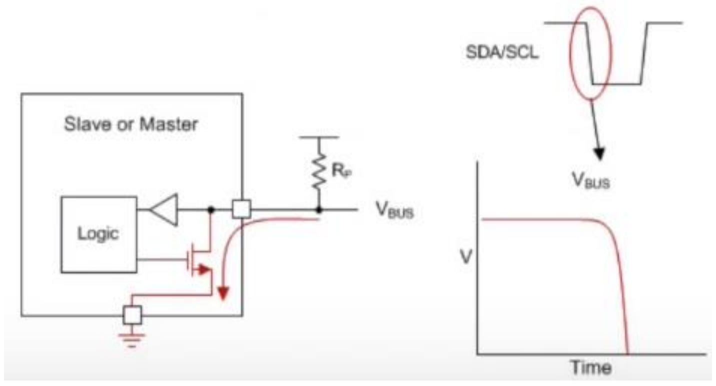


Figure 3.4 I2C Bus - Low

When the transistor turns off the I2C is turned up:

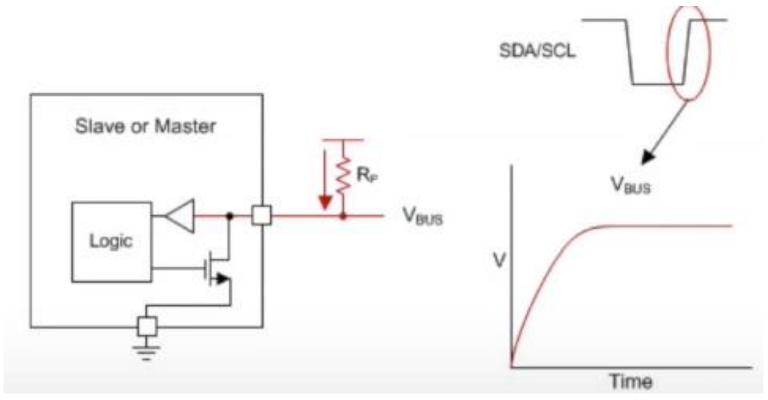


Figure 3.5 I2C Bus - High

The pull-up resistors value can be a value between a range of resistor values. The minimum value depends upon certain criteria. A small value of the resistor (stronger pull-up) may prevent the I2C pin from being able to drive low. The logical output voltage() determines the $R_P\, min$ which depends of the logic output low current and the supply voltage.

$$R_P \min = \frac{V_{CC} - V_{OL}\,max}{I_{OL}} \tag{3.1}$$

The maximum value requires knowledge of the parameters about the I2C bus such as the bus capacitance.

$$R_P \max = \frac{t_r}{0.8473 \cdot C_B} \tag{3.2}$$

Let's consider the pull-up resistor and de capacitance bus as a RC circuit. As the value of the voltage depends of the time (t), the pull-up response of a RC circuit to voltage step of amplitude V_{CC} starting at time t = 0 is written as follows:

$$V(t) = V_{CC} \cdot (1 - e^{\frac{-t}{RC}}) \tag{3.3}$$

As the V_{IH} (Logic Input High voltage) is:

$$V_{IH} = 0.7 \cdot V_{CC} = V_{CC} \cdot (1 - e^{\frac{-t1}{RC}}) \tag{3.4}$$

And the V_{IL} (Logic Input Low voltage) is:

$$V_{IL} = 0.3 \cdot V_{CC} = V_{CC} \cdot (1 - e^{\frac{-t2}{RC}}) \tag{3.5}$$

So we find the expression of the time:

$$t_2 = -R \cdot C \cdot \ln 0{,}3$$
$$t_1 = -R \cdot C \cdot \ln 0{,}7$$

The rise time is the difference of the two instances $t1$ and $t2$:

$$t_r = t_2 - t_1 = -R \cdot C \cdot \ln 0{,}3 + R \cdot C \cdot \ln 0{,}7$$

Where R is the pull-up (R_p) resistor and C the capacitance bus C_B

$$t_r = R \cdot C \cdot (\ln 0{,}7 -) = 0.8473 \cdot R_p \cdot C_B$$

$$\boxed{R_p = \frac{t_r}{0.8473 \cdot C_B}}$$

(3.6)

Two I2C pull-up resistors of 10K will be added to the board and be configurable with a two sets of headers and jumpers. This improvement makes the boards more flexible and usable when the inner circuit has not incorporated pull up resistors.

3.4.2. Jumpers & Headers

They'll aim us to set the voltage, what makes the design more flexible. During the internship the expansion board was designed without taking in account the pull-up resistors of the I2C but later were added to the board of the razor. In this new design, the pull-up resistors of the I^2C are included and connect to the data and clock line to the supply voltage.

3.4.3. Through Hole IDC Connector

The board is designed to connect the I2C and analogical sensors to the MCU. Since the dimensions of Arduino reference board aims us to solder 8 connectors, the capacity of the board is up to 16 sensors using I2C protocol and 8 analogic sensors that can communicate all with the microcontroller through the same canal/buses.

3.4.3.1. I2C Connectors – Pinout

The I2C connectors are set to connect to devices that use I2C protocol, so 8 pins of the 10 are connected and the pinout is shown to the table below:

Table 3.2 I2C Pure Pinout

Pin	1	2	3	4	5	6	7	8	9	10

J1,2,3,4	SDA	VCC	SCL	GND	-	-	SDA	SCL	VCC	GND

Note: The pin 5 and 6 of the connectors are floating.

3.4.3.2. I2C & Analog Connectors – Pinout

Table 3.3 I2C & Analog – Pinout

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
J5,6,7,8	SDA	VCC	SCL	GND	AN_X	AN_Y	SDA	SCL	VCC	GND

Note: AN_X and AN_Y are reserved to connect analog sensor or devices that need to communicate the MCU.

3.4.3.3. Master Connector J9 – Pinout

Table 3.4 Master Connector J9 – Pinout

Pin Master	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pin Analog	$J6_6$	$J6_5$	$J7_5$	$J5_6$	$J7_6$	$J5_5$	$J8_5$	SCL	$J8_6$	SDA

The master connector connects the I2C and analogical devices to the Razor of SparkFun house.

3.5. 9DoF Razor IMU – Board Adaptation Design

The SparkFun 9DoF Razor IMU M0 is an MPU-9250 9Dof sensor combined with an Atmel SAMD21 (SAMD21G18) microprocessor.

3.5.1. MPU-9250 9DoF

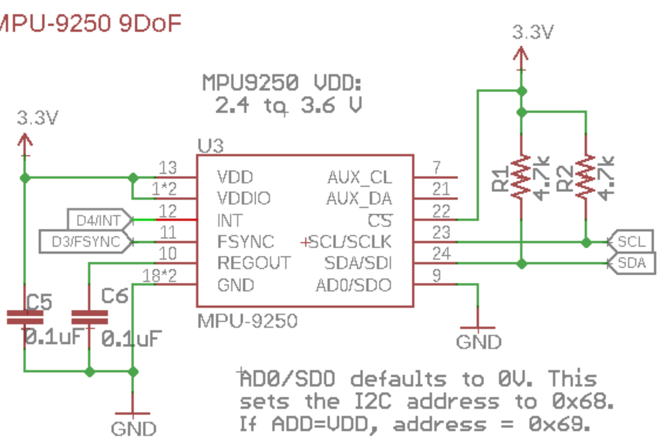


Figure 3.6 MPU9250 - Pinout

[16] The MPU-9250 is a 9-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and a Digital Motion Processor™ (DMP) all in a small 3x3x1mm package.

It features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for the accelerometer outputs, and three 16-bit ADCs for the magnetometer outputs.

The figure above shows the schematic of the MPU-9250. In the table below is shown the Pin out and Signal description:

Table 3.5 MPU92590 - Pinout Description & Connection

Pin Number	Pin Name	Pin Description	Connection
1*2	VDDIO	Digital I/O supply voltage	3.3V
7	AUX_CL	I2c Master serial Clock	-
9	AD0	I2C Serial Address LSB	GND
10	REGOUT	Regulator filter capacitor connection	3.3V through a capacitor of 0.1uF
11	FSYNC	Frame Synchronization In	D3/FSYNC
12	INT	Interrupt Digital Input	D4/INT
13	VDD	Power Supply Voltage	3.3V
18*2	GND	Power Supply Ground	GND
21	AUX_DA	I2C Master Serial Data	-
22	nCS	Chip Select	3.3V
23	SCL	I2C Serial Clock	3.3V through pull-up resistor 4.7k
24	SDA	I2C Serial Data	3.3V through pull-up resistor 4.7k

ABOVE, it’s explained the I2C protocol and how it works. When connected to a processor the MPU-9250 can operate as a slave device. As we explained, the I2C buses always needs pull-up resistors to define the logic level of the bus. In this case both have a value of 4.7k and are connected to a VDD of 3.3V. The slave address of the MPU-9250 is b110100X. The LSB bit is determined by the logic level on pin AD0. As the pin AD0 is connected to the ground so the MPU-9250 slave address is b1101000 (0x68).

3.5.1.1. Gyroscope – Features

The gyroscope consists of three independents vibratory MEMS rate gyroscope which detect rotation about the three axes. A Coriolis Effect causes a vibration when the gyros are rotated about the sense of the axes. This vibration is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered producing a voltage proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis.

The gyroscope includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- Low-pass filter digitally programmable and enable a wide range of cut-frequencies
- Gyroscope operating current: 3.2mA
- Sleep mode current: 8µA
- Factory calibrated sensitivity scale factor
- Self-test

3.5.1.2. Accelerometer – Features

It uses separate proof masses for each axis. Acceleration along one of the axis induces a displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The work principle of the accelerometer is represented to the figure below.

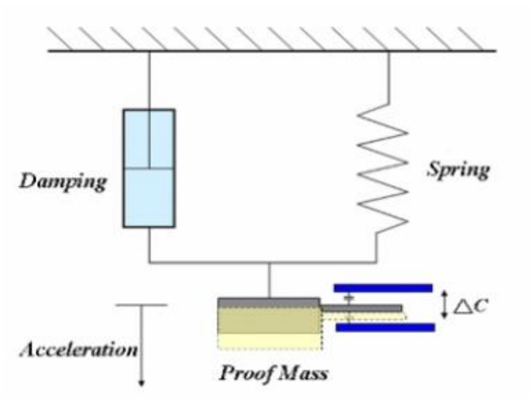


Figure 3.7 Accelerometer Work Principle

It includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g.
- Accelerometer normal operating current: 450µA
- Low power accelerometer mode current: 8.4µA at 0.98Hz, 19.8µA at 31.25Hz
- Sleep mode current: 8µA
- User-programmable interrupts
- Wake-on-motion interrupt for low power operation of applications processor
- Self-test

3.5.1.3. Magnetometer – Features

It includes the following features:

- 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
- Wide dynamic measurement range and high resolution with lower current consumption.
- Output data resolution of 14 bit (0.6µT/LSB) or 16 bit (15µT/LSB)
- Full scale measurement range is ±4800µT
- Magnetometer normal operating current: 280µA at 8Hz repetition rate
- Self-test

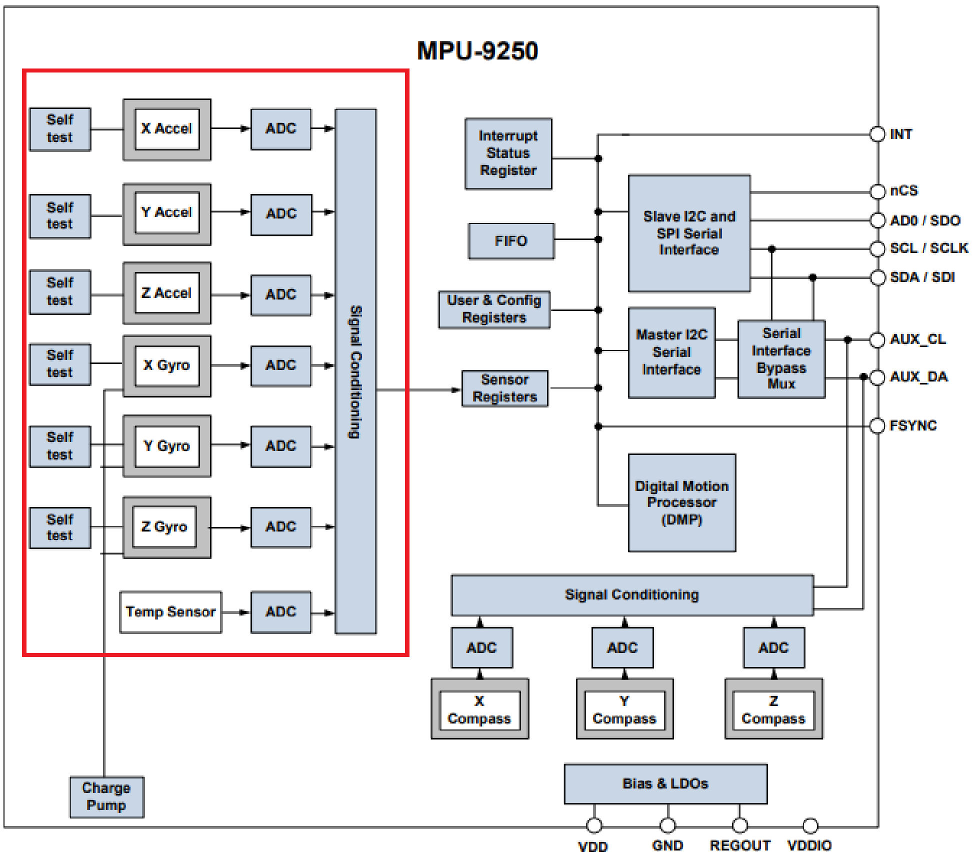


Figure 3.8 Internal Block Diagram - MPU9250

The sensors measurements are stored to the sensor registers. The register Map and description are annexed to this file.

3.5.1.4. Self-Test

The self-test aims us to test the electrical and the mechanical portions of the sensors embedded to the MP9250. They can be enabled and disabled by using the self-test registers. When activated, the output caused by the sensors is used to observe the sensor self-test response, which is defined as the difference between the sensor output with self-test-test enabled and with self-test disabled. This value should remain within the approximate limits (values generated during the manufacturing tests). When the self-test response exceeds the appropriate values it is deemed to have failed self-test.

3.5.2. The SAMD21G18

[17] The SAMD21 is based on an ARM (Advanced RISC Machine) architecture. It uses a processor ARM Cortex-M0+ CPU of 32-bits bus size compared to the

8-bits of the ATMEGA328. It runs at frequency up to 48MHz with a flash memory up to and 32KB SRAM. One of the features of this microprocessor is SERCOM, which is a set of 6 serial communication interfaces, each configurable to operate either: I2C master, USART, SPI master, USB, I2C slave and SPI Slave etc. Every port can be multiplexed giving a choice of which task each pin is assigned.

The SAMD21G18 has 14 ADC input pins. Each pin has 12-bit resolution which means that every bit represents about 0,49 mV (for a voltage range of 2,01 V) from the bit 0 to 2^{12} which allows more sensitive voltage measurements.

$$Resolution = \frac{3{,}63\ V - 1{,}62\ V}{2^{12}} = 0{,}49\ mV \tag{3.6}$$

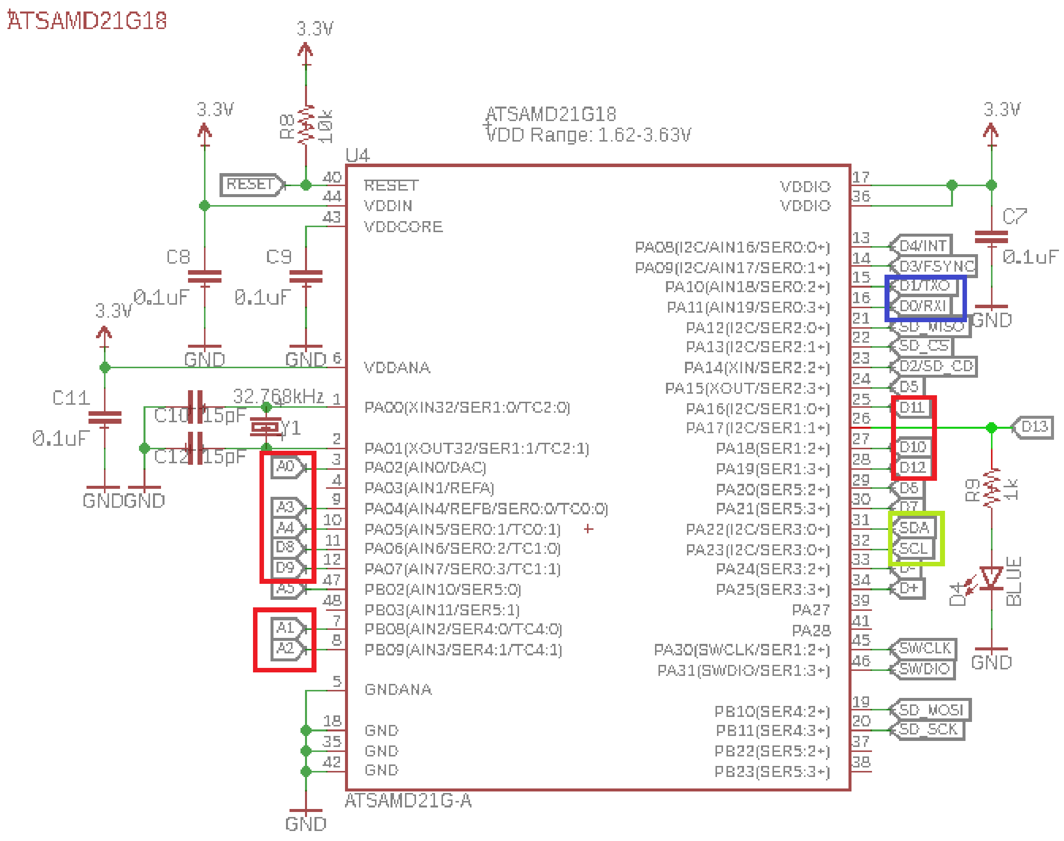


Figure 3.9 SAMD21G18 - Pinout & Internal Connection

The SAMD21G18 is the brain of the Razor. All the sensors will share their measurements through I2C bus to the SAMD21. In addition to a pair of ICs, the 9DoF Razor IMU includes a µSD card socket to log the measurements of the MPU-9250.

The pins shown in the figure above are connected to the PCB pinout. It includes pins 10-13, analog-to-digital converter inputs A0-A4, RX, TX, and the I2C pins, SDA and SCL.

It’s important to remember that the MPU-9250 share the same I2C bus, so it’s important to ensure that any additional I2C devices using the bus doesn’t share the IMU’s 7-bit address (0x68). The Analog sensors of the Astro01 are connected to the microprocessor through these pins. In the table below it is shown the PCB and the connectors relation.
There are connected through two IDC connectors, to the table below are shown the pins connections:

Table 3.6 RAZOR IDC Connector J0 – Pinout

Pin	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
J0	SDA	D10	SCL	A0	D09	A1	D08	A2	A4	A3

Note: The Pin A_ and D_ correspond respectively to the analog and digital pin of the 9DoF Razor IMU pins.

Table 3.7 RAZOR IDC Connector J1 – Pinout

Pin	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
J1	D13	D12	GND	D11	3.3V	-	RX	-	TX	-

Note: The pins 6, 8 and 10 are floating.

According to the manufacturer the VIN, VBAT, and GND pins can be used to supply the 9DoF Razor IMU’s 3.3V regulator, instead of the USB or LiPo JST inputs. And also the Voltage on the VIN pin should not exceed 6V, and the VBAT pin should only be connected to a single-cell LiPo battery.

CHAPTER 4. HARDWARE BUILDING

4.1. From the Design to Manufacturing

After the Gerber files of each subsystem were generated and all electrical and design test runt (with the Eagle commands ERC and DRC) they were sent to the university for manufacturing. One week later, we get our boards back from the university due to some mistakes that have been found when loading the files to the CircuitCAM Software.

CircuitCAM is a Computer Aid Manufacturing system for printing PCB. It is used to prepare the Gerber file generated by the EAGLE Autodesk program into a format, which the milling machine understands.

To make a little summary of what is the Gerber file. They are an intermediary manufacturing files (generated after design) and they describe the copper of every layer in the way that a CAM like CircuitCAM can understand. They are two different formats of the Gerber available with Autodesk EAGLE (Version 8.7.0), the Gerber RS-274D and the Gerber RS-274X. The Gerber RS-274D is the oldest one and use two files per layer to contain the information. The RS-274X use all the information of one layer in a single file. The Eagle Autodesk program is a Software design I never used during my Bachelor degree, so when I was designing the PCBs I had to draw on of the manual guide.

1st Intent. The Gerber files were generated following the information given by the manufacturer. The mistakes are due to the format of Gerber used (the RS-274D), which was incompatible with the CAM the university Laboratories Technical support uses.

2nd Intent. The drill copper wasn't defined. The CircuitCAM can't open the file.

3rd Intent. The distance between the lines and the polygon was short and doesn't fulfil the PCBs design rules.

Solutions. In order to generate the files in the correct format, which solves the first and the second mistake, the next Steps were followed:

1. Open the **CAM Processor**

2. Left click to the arrow at the top side of the CAM Processor window, and from the Local CAM jobs use the **gerb274x.cam**.

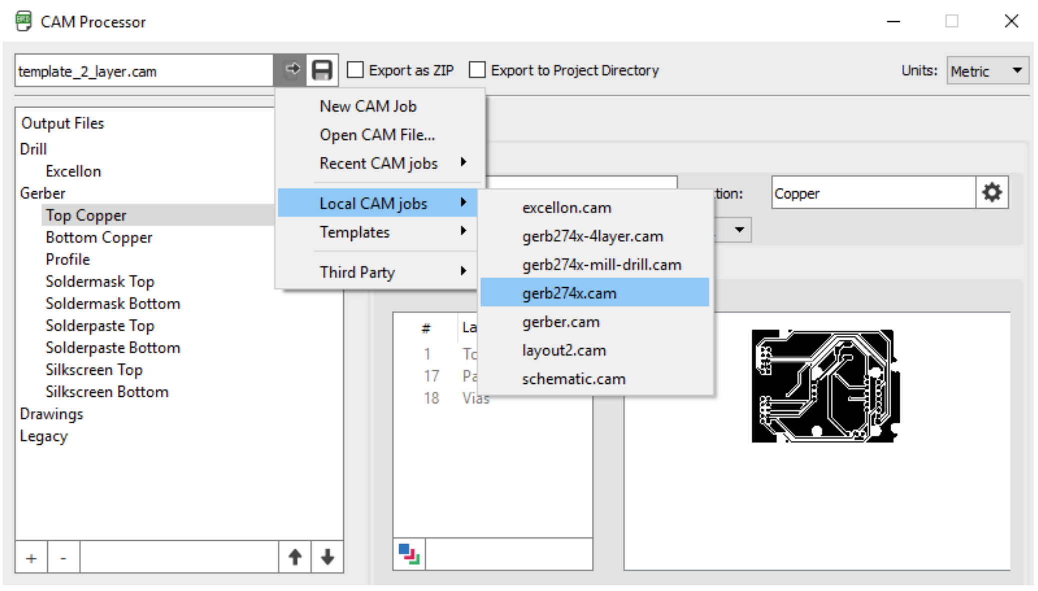


Figure 4.1 Gerber Format Select

3. After the last step will appear the windows below and the output type is set by default to the format **Gerber X2** which has to be changed to the format **Gerber RS-274X.**

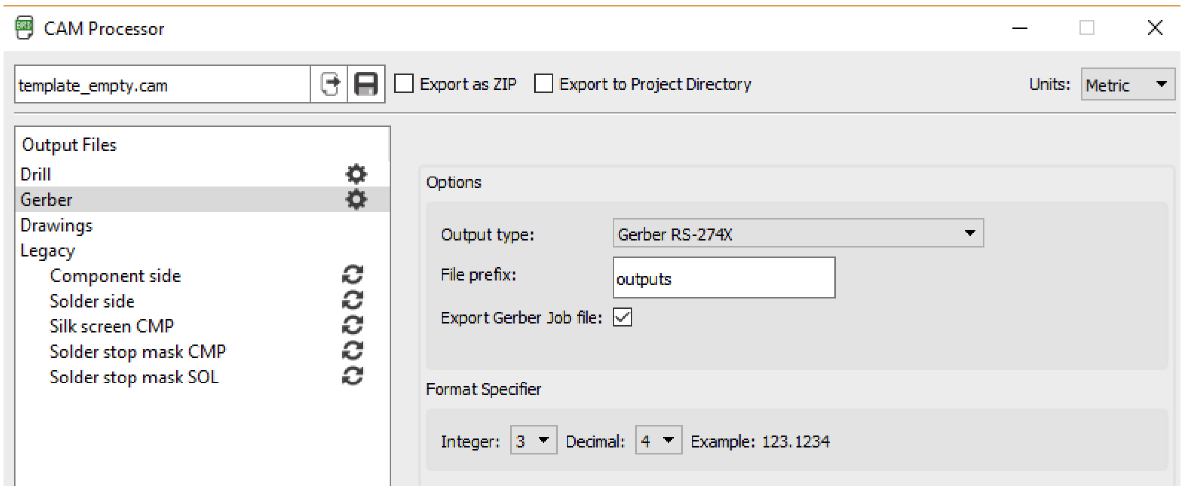


Figure 4.2 Generating Gerber Files

4. Select a file where to save Gerber Files, that are 6 and are explained below:

.cmp	**Top Copper**
.sol	**Bottom Copper**
.stc	Top Soldermask
.sts	Bottom Soldermask
.plc	Top Silkscreen

.gpi	Unknown

All the information about the Top and Bottom layer are contained respectively to the .cmp and the .sol files.

The files generated give information to the manufacturer about the individual layer but not about the drill holes. So the next step is to generate the drill files.

1. Open the **CAM Processor**
2. Left click to the arrow at the top side of the CAM Processor window, and from the Local CAM jobs, use the **excellon.cam**.

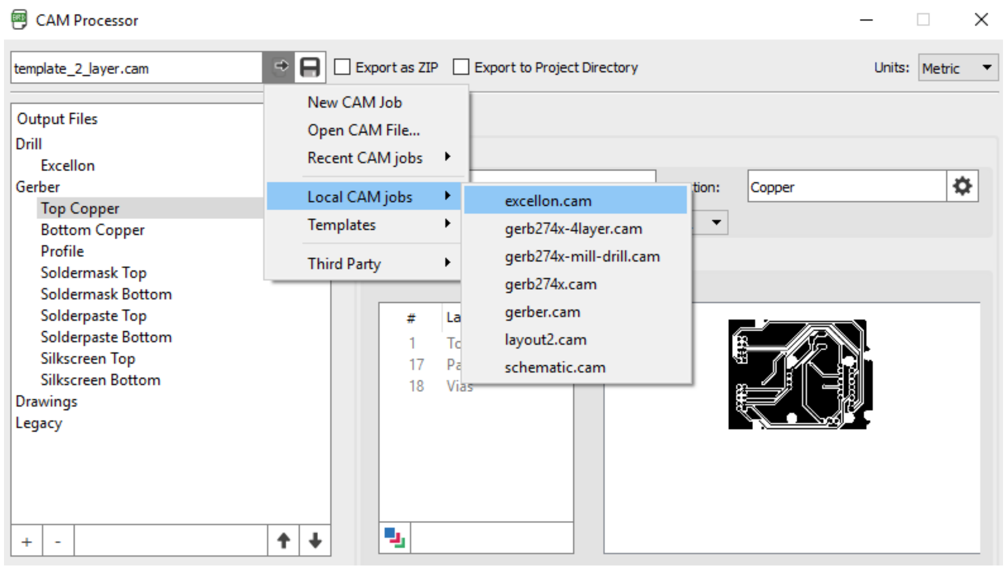


Figure 4.3 Generating Drill Files

3. After the last step will appear the windows below and the output type is set by default to the format **Gerber X2** which has to be changed to the format **Gerber RS-274X.**
4. Save the file to the same address than the Gerber Files. This step generates two files. The. drd file contains the information related to the drill Holes.

In order to solve the mistakes due to the distance between the polygon and the PCB's routes, the polygon isolation property has to be changed. This value represents the distance between the polygon and the routes. This value is zero by default and has to be at least equal to the thicker line.

4.2. Soldering and Wiring Test

4.2.1. Electrical Power Subsystem

Before starting soldering the components to the EPS board, first the continuity of the lines was checked (especially those that go from a layer to another one).

The next step was to start soldering the SMD resistors, which wasn't easy due to the component and the board size. And later the others components: diodes, capacitors and the PTC Fusible.

A DC Power supply is used to check the EPS operation. The Power Supply is set to +12V. The EPS is connected to the Power supply through a wire connected to a Barrel Jack DC connector. We check the outputs voltage with a Voltmeter, the results are shown in the figure below:

Table 4.1 Output Voltage 3.3 - 5 V

Outputs Voltages	*3.3 V*	*5 V*
3.3 – 5 V	3.25 V	5.03 V

Table 4.2 Output Voltage 5 - 9 V

Outputs Voltages	***5 V***	***9 V***
5 – 9 V	4.981 V	7.965 V

The output voltage of 9 V is 12% less than the expected value due to the resistors. In the subsection 3.2.2 are computed the value of the resistors to achieve an output voltage of 9 V. We didn't have the resistor of 1k1Ω so we decided to apply the trial error method and solder two resistors of 1K3 and 1K5 which gave respectively an output voltage of 9.6 and 10.25 V. Finally, we decided to use the method of connecting to resistors in parallel:

Be R_1 two resistors with same value, if we connect the two resistors the equivalent resistor is computed as:

$$R_{eq} = \frac{R_1 \cdot R_1}{R_1 + R_1} = \frac{{R_1}^2}{2 \cdot R_1} = \frac{1}{2} \cdot R_1 \tag{4.1}$$

The equivalent resistor is equal to the half of R1. We connect two resistors of 2K2 Ω so the equivalent resistor value is:

$$R_{eq} = \frac{1}{2} \cdot 2200 = 1K1\ \Omega$$

(4.2)

The output voltage obtained after soldering the new resistor is shown to the figure below:

Table 4.3 Final test - Output Voltage 5 - 9 V

Outputs Voltages	***5 V***	***9 V***
5 – 9 V	4.981 V	8.897 V

4.2.2. Communication Subsystem

In this section we'll first discuss the mistakes found while soldering the components to the board and their mitigations. As mentioned to the Chapter 3, not all the devices of the communication subsystem are working with the same voltage, so 3 level shifters are connected to down convert the 5V of the Arduino UNO to 3.3V, so it's important that the devices are powered with the correct voltage.

Regarding to the complexity and the number of component embedded, the communication subsystem is definitely the most complex subsystem. Beside of outfitting the transmitter Hx1, a low power transmitter, it also integrates the electronics that eases the payload location and the GPS from SparkFun. The continuity and the connectivity are checked and compared with the design as done with the other subsystems.

4.2.2.1. The SparkFun VENUS GPS

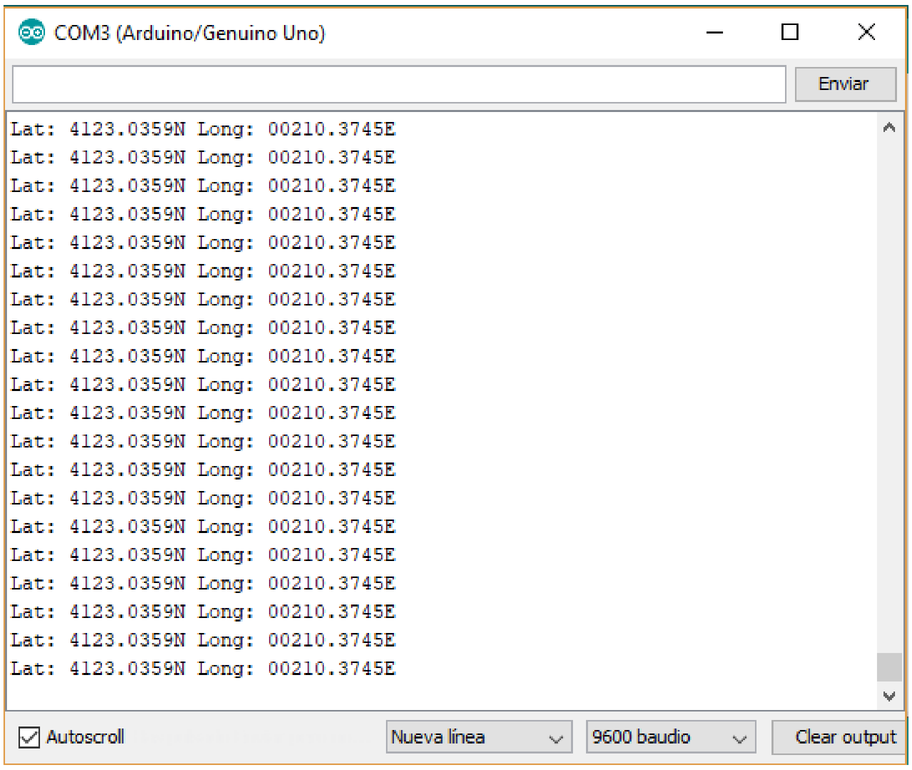
The GPS is tested separately with a protoboard powered by an external voltage supply module and connected to the Arduino. The test is done using the Software Serial Library on the Arduino. The sketch uploaded to the Arduino Board to test the VENUS GPS has been found to the manufacturer web page. It is attached to the ANNEX at the end of the document. The following connections are made:

- Connect with a wire the 3.3V pin of the Arduino to the 3.3V VENUS GPS in order to power the GPS module.

- Connect the ground (GND) of both board.

- And finally connect the Arduino pin 10 to TX0 of the VENUS GPS module.

Opening the Serial Monitor, our position appears in latitude and longitude:



```
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
Lat: 4123.0359N Long: 00210.3745E
```

Figure 4.4 GPS Latitude and Longitude with the VENUS module from SparkFun

According to the Google Earth our position is Latitude 41º23’00.21” N and Longitude 2º10’23.79” E. According to the Venus GPS the latitude and longitude is respectively 41º23’03.59” N and 2º10’37.45” E. Depending on the datasheet the accuracy of the GPS is 2.5m which is less than the accuracy measured which is around 330m.

4.2.2.2. Buzzer

The buzzer work with PWM (Pulse Width Modulation), the PWM also called as pulse-duration modulation used to encode a message into a pulsing signal. PWM varies the perceived power going to the electronic devices by very quickly turning the power on and off, the perceived output is changed by varying the duty cycle. The duty cycle describes how long the signal is turned on versus turned off and it is expressed in percent. When the duty cycle is 100% it means that the signal is always on and when it is 50% it means that the signal is turned on half the time and off the other half. An important part of duty cycle is the switching frequency or how fast the signal turns on and off.

In the trackuino code, the duty cycle is set according to the F_CPU which is the clock rate of the microcontroller. The buzzer is turn on two seconds and off 1 second. And the altitude at which it starts buzzing can be set according to the altitude given by the GPS. In this case we set it to the height of 1.000 meters (3.280,84 ft.).

It is also possible to use both active (DC) and passive (AC) buzzer. The program aims us to change the value of the BUZZER_TYPE which is the variable that describes if the buzzer is active (0) or passive (1). The main difference is that active buzzers take direct current voltage and have an internal oscillator that translates the DC into a fixed-frequency square wave whereas passive buzzers needs that square wave be generated externally.

After uploading the trackuino codes to the Arduino Uno board, the trackuino board is connected to the MCU. The Arduino code was modified removing to the altitude condition thus the buzzer's operation can be tested. This last one starts buzzing after connecting the device to the USB port of the computer.

4.2.2.3. Transmitter – Radiometrix Hx1

The latest version of the trackuino firmware is the 1.52. After downloading the firmware of this version, we proceeded to check the different sketches and study their functionality. We made some change in the original shield, therefore some changes had to be made in the firmware in order to fit it with our version. This part of the project had to be done by the informatics student.

The Config.h contains all the basic variables that have to be set before uploading the firmware to board. This sketch is attached to the document. The table below shows the basic

Table 4.4 Config.h Parameters

Sketch	***Functions***
Config.h	Configurate the basic: Callsign data, transmission delay, APRS slot and period, the GPS Baud rate, the type and Buzzer and strobe light altitude etc.…

To test the subsystem, we two antennas VHF/UHF with a frequency range of 144 to 430 MHz. The first antenna was connected to the trackuino shield and the second one to the spectrum analyser. The central frequency was set to 144,800 MHz. As result we didn't receive the signal, which means that the transmitter wasn't working properly.

After checking the electronics connection and compare the theoretical design with the actual board, we got to the conclusion that the issue wasn't in the Hardware. Thus we decided to check the software, for that we opened the main sketch and start debugging the setup functions but wasn't showing any flaws.

We noticed based on when bypassing the level shifters inputs, the signal the signal appeared to the screen and disappeared in few seconds after bypassing. In fact, we realized that two different signals (low and high) were arriving to the level shifter, therefore the Hx1 transmitter wasn't enabled.

Disabling the level shifters, we were able to see the signal to the spectrum analyser and the AFSK modulation result to the serial monitor, and the transmission started.

4.2.2.4. Strobe Light

Even if apparently the strobe light seems not to be an important piece of the puzzle, it is primordial for the payload location after the flight specially when the light conditions (visibility) are low. The strobe light has the same principle of work than the buzzer. So in order to test the LED, we can use the same code and set the LED_TYPE to 0. After connecting the board to the computer the light started blinking.

4.2.3. 9DoF Razor IMU Board

The same process is done to check the continuity of the lines. And in this case a discontinuity is found to the line that connects the Analog input A2 of the IDC connector J0_8 and the line that connects the digital pin 09 of the Razor to the IDC connector J0_5.

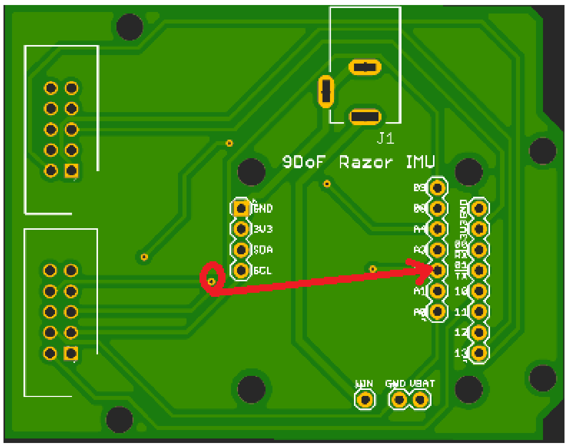


Figure 4.5 Razor Board from manufacturing

The discontinuity of A2 – J0_8 is due to the lack of copper in hole that connects both lines from the top and the bottom layer. To fix it a cooper is introduced into the hole and each extreme is connected to one layer.

The discontinuity of 09 – J0_5 is probably due to a manufacturing error.

After soldering all the components (The Jumpers, Headers and IDC connectors), we supply +3.3 V to the razor Board, and check if it is operating switching the ON/OFF button. The red led razor turned on, which means that the Razor is well powered. The next step was to check the connectivity of the Table 3.8 and Table 3.9.

Caution: When supplying voltage through the Jack Barrel DC connector don't connect none other power source (USB etc…).

4.2.4. Expansion Board

It's important to emphasize the importance of the expansion board. The payload and the peripheral chips are connected to the MCU through the Expansion board. Before starting to solder all the components as shown to the ANNEX as done with the others subsystems, the connectivity and continuity are checked.

CONCLUSION

The objective of the Astro-01 project was to design and develop the hardware and Software of a stratospheric scientist balloon as a Testbed of CubeSat standard for the Institute of Space sciences.

This Project was leaded and coordinated by Carles Sierra Roig, Electronics Engineer and member of the Advanced Engineering Unit to the Institute of Space Sciences (ICE). We were two students hired to carry out the software and hardware design and implementation. Julià Medina, student in informatics engineering from the Universitat Autònoma de Barcelona in charge of the software part and I, responsible of the hardware unit. Since Julià left the project, the objectives were reduced. The Objectives of the hardware unit were to design the main subsystems which are needed for any space platform to operates.

- The electrical power subsystem which power the whole system.
- The Communication subsystem which will send the measurements taken by the payload and scientist experiments to the ground stations.
- And the platform (The expansion Board) which will aim to connect all the experiments to the brain (The MCU of the 9DoF Razor IMU) of the Astro-01.

In this part of Astro-01, we decided to improve the Hardware first design using a scriptable EDA (Electronic Design Automation), Eagle Autodesk Version 8.7.0, to enhance the capacity, the stability, the size and others physical and geometrical characteristics of the hardware to make the subsystems more understandable and well documented.

As explained above we couldn't develop the software code. But the original open source code had been modified to carry out the operation test of the subsystems.

And finally after mounting all the subsystems, we were able to test them separately. The results were satisfactory although some of them weren't what we expected for to the first test: the EPS output voltage, the analogous pins that connects the razor, to the J01 IDC connectors, the GPS accuracy etc.

Looking forward:

- Add a connector to the shield of the trackuino to connects the Arduino I2C serial bus to the Razor IMU I2C serial bus. The design didn't take into account this connection. It will ease the data transfer from the MCU SAMD21 to Arduino microprocessor.
- When using the trackuino, don't add the level shifters. If it is wanted to use the shield for both with the Arduino UNO.

- Use the transmitter SBR MX145 which have better features regarding to the power transmitter. The SBR MX145 also include a receiver.

- As the strobe light added to the communication subsystem is turned on at the same time than the buzzer under 1km of height, it doesn’t make sense to ease visual location when there is a sunlight. Looking forward light detector can be used to switch on/off the strobe light.

Bibliography

[1] CubeSat, https://ca.wikipedia.org/wiki/CubeSat [visited 14/01/2018]

[2] Charlière, https://ca.wikipedia.org/wiki/Globus_de_gas [Visited 12/05/2018]

[3] Institute of Space Sciences, http://www.ice.csic.es/es/content/2/bienvenidos-a-nuestra-pagina-web [Visited 14/01/2018]

[4] About the ITU, https://www.itu.int/en/about/Pages/default.aspx [Visited 02/01/2018]

[5] ITU Radiocommunication Objectives, https://www.itu.int/en/ITU-R/Pages/default.aspx [Visited 02/01/2018]

[6] ITU Symposium and Workshop on small satellite regulation and communication systems, https://www.itu.int/en/ITU-R/space/workshops/2016-small-sat/Documents/01-AM-CHL-16.pdf [Visited 02/01/2018]

[7] Radio Regulations Nº1, http://search.itu.int/history/HistoryDigitalCollectionDocLibrary/1.43.48.en.101.pdf [Visited 09/01/2018]

[8] The IARU 2m band, https://www.iaru-r1.org/index.php/spectrum-and-band-plans/vhf/2-meter [Visited 03/04/2018]

[9] The CNAF, https://es.wikipedia.org/wiki/Cuadro_Nacional_de-Atribuci%C3%B3n_de_Frecuencias_(Espa%C3%B1a)

[10] SparkFun Venus GPS, https://www.sparkfun.com/products/11058 [Visited 04/04/2018]

[11] CubeSat Design Specification, https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf [Visited 12/04/2018]

[12] Learning I2C Serial Bus, https://learn.sparkfun.com/tutorials/i2c [Visited 22/08/2018]

[13] The XTend 900 Custom, https://www.sparkfun.com/products/retired/9411 [Visited 22/04/2018]

[14] XTend 900 Product manual, https://www.sparkfun.com/datasheets/Wireless/Zigbee/xtend-productmanual.pdf [Visited 22/04/2018]

[15] The Automatic Packet Reporting System definition, https://en.wikipedia.org/wiki/Automatic_Packet_Reporting_System [Visited 20/02/2018]

[16] SBR MX145 Technical Specifications, https://www.argentdata.com/catalog/product_info.php?products_id=134 [Visited 23/02/2018]

[17] The Radiometrix Hx1, http://www.radiometrix.com/content/hx1 [Visited 03/05/2018]

[18] Wertz J.R., Larson W.J. (Eds.), Space Mission Analysis and Design, 3rd ed., Microcosm Press, California, 1999

[19] Helvajian H., Janson S.W. (Eds.), Small Satellites: Past, Present, and Future, AIAA, Reston, Virginia, 2009

[20] Barnhart, D.J., Vladimirova T, Baker A.M., Sweeting, M.N., A low-cost femtosatellite to enable distributed space missions, Acta Astronautica, vol. 64, no. 11–12, June 2009, pp. 1123–1143

UPC eetac Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Institute of Space Sciences

ANNEXOS

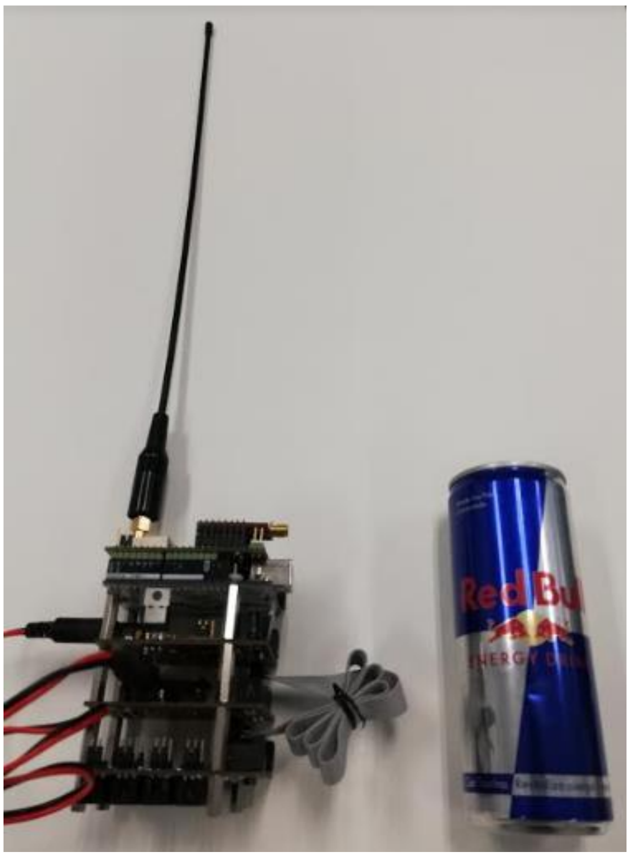
TFG TITLE: Disseny i desenvolupament del hardware del CubeSat Testbed Astr-01

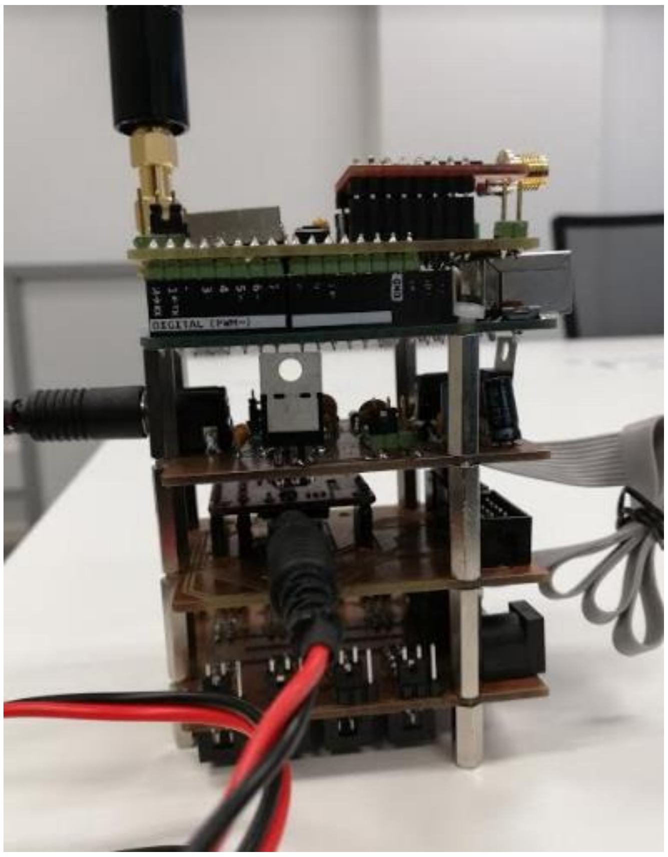
DEGREE: Grau en Enginyeria d'Aeronavegació

AUTHOR: Ibrahím Mbaye

ADVISORS: Jordi Gutiérrez Cabello
Carles Sierra Roig

DATA: September XX, 2018





Astro – 01

Electrical Power Subsystem - EPS User Guide

Description

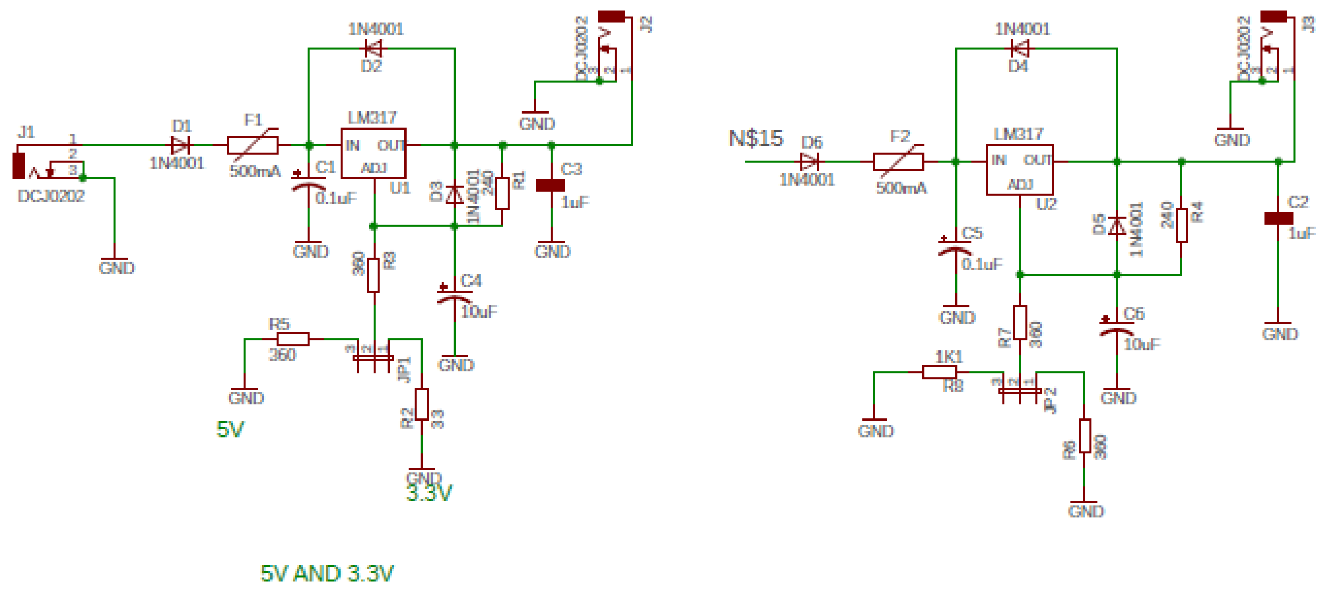
Here is a very simple power supply that takes power a DC battery and have three selectable outputs (5V, 3.3V and 9V). The design files are attached at the end of the document. The input voltage is 12/15V and can support until 37V.

EPS - Electrical Power Subsystem

Includes

- DC Barrel Connector (2.1mm center positive)
- TO-220 Voltage Regulator (LM317 1.5A max current)
- 6 x 1N4008 Reverse Protection Diode
- 100uF 25V Capacitor
- 2 x 10uF 25V Capacitor
- 0.1uF 50V Capacitor
- 2 x 3 Headers
- 2 Jumpers 2.54mm
- SMD 4 x 360 Resistor W
- SMD 30 Resistor 1%
- SMD 2 x 240 Resistor 1%
- SMD 2 x 2K2 Resistor 1/6W
- 2 x PTC resettable fuse

$$V_{OUT} = V_{REF} \cdot \left(1 + \frac{R_2}{R_1}\right) + (I_{ADJ} \cdot R_2)$$



Spark Fun Electronics

TITLE: arduino_Uno_ref2

Document Number:

REV: 1

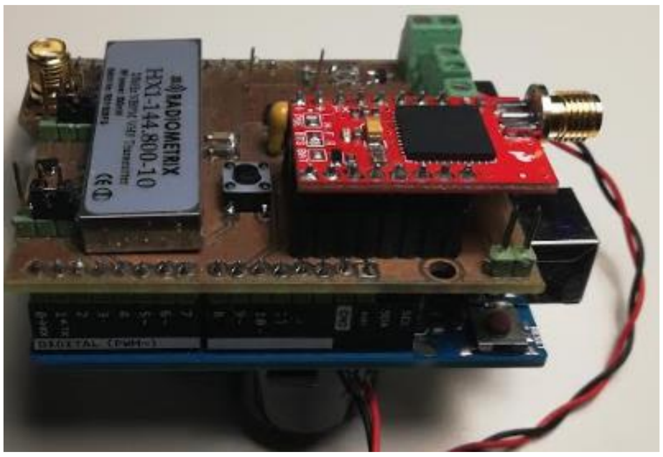
Date: 05/05/2018 22:13

Sheet: 1/1

Trackuino Board User Guide

Description

The trackuino is an open-source source APRS tracker based on the Arduino platform. The APRS (Automatic Packet Reporting System) is a Radio based system for digital communications of information of immediate value in the local area.



Trackuino - Arduino

Features

- Arduino shield form factor (you can stack more shields on it)
- GPS: Venus 638FLPx. Reports okay above 18 Km.
- Radio: Radiometrix's HX1 (300 mW).
- 1200 bauds AFSK using 8-bit PWM
- Sends out standard APRS position messages (latitude, longitude, altitude, course, speed and time).
- Active/passive buzzer support to ease acoustic payload location.
- Strobe light to support to ease visual payload location
- 2 x SMA female plugs (1 x GPS in + 1 x radio out

Jumpers configuration

IC3, IC4 and IC5 are Level Shift converters. IC3 and IC5 Convert 3.3V to 5V. and IC4 down converts 5V to 3.3V. These level shifters aim the board to be connected to two kinds of MCU, the 3.3V Arduino (The ChipKit Uno32) and 5V (UNO, Arduino Duemilanove).

- When using a 5V MCU:

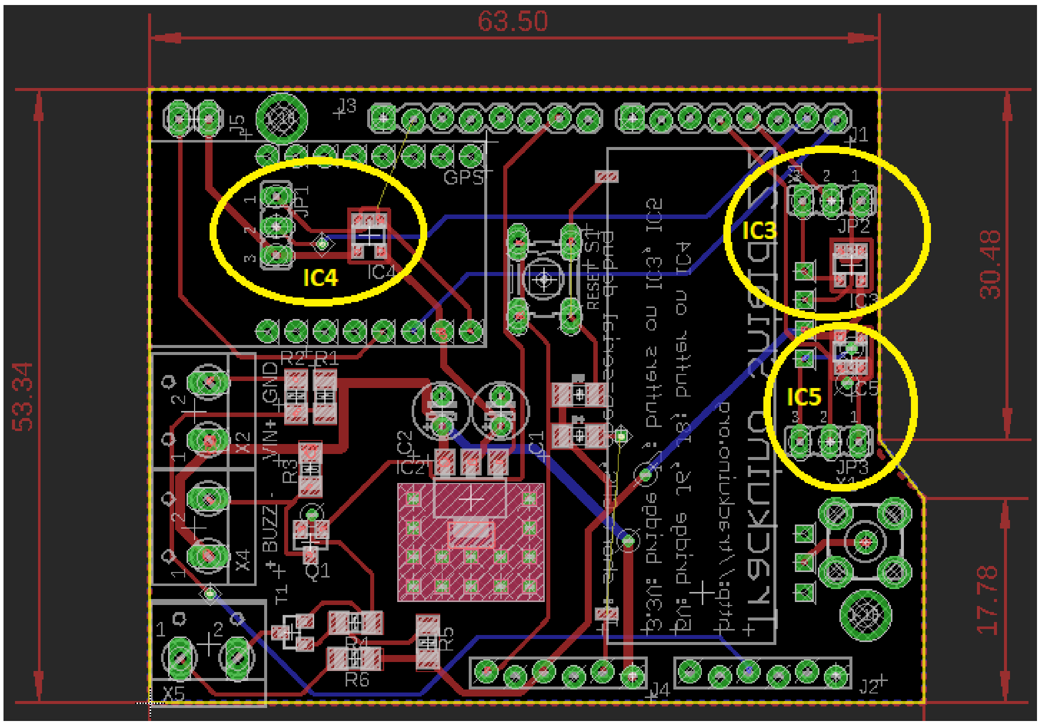
The pins 1 and 2 of JP1 have to be shorted thus IC4 provides the 5 -> 3.3V downconversion for the GPS which needs to powered with 3.3V.

The pins 2 and 3 of JP2 and JP3 have to be shorted, because the transmitter work with 5V and there is no need to down convert its input voltage.

- When using a 3.3V MCU:

The pins 2 and 3 of JP1 have to be shorted, because the GPS works with 5V and there is no need to down convert its input voltage.

The pins 1 and 2 of JP2 and JP3 have to be shorted thus IC3 and IC5 provide the 3.3 -> 5V downconversion for the Hx1 transmitter which needs to be powered with 5 V.



Eagle File: Trackuino Board design

When using a 7.5 DC voltage, waiting for GPS fix will have a consumption of 120mA and normal operation (inactive) and normal operation (Hx1 transmitter) will have respectively a consumption of 70mA and 200mA.

They are Three way to power this board using:

- The Arduino barrel Jack
- The VIN and GND clamp terminals, close to the buzzer clamps

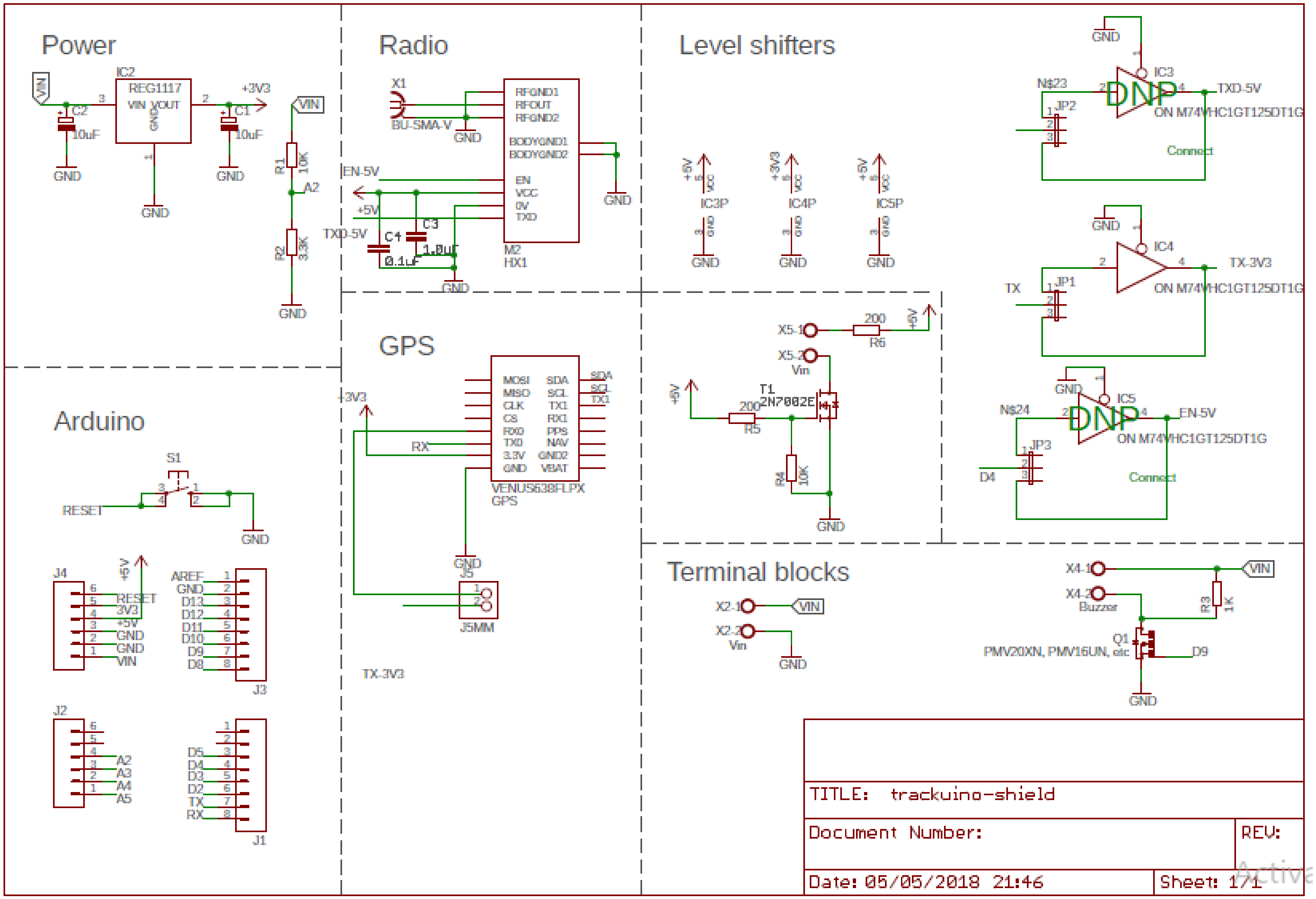
- A USB cable plugged to a host computer.

Caution

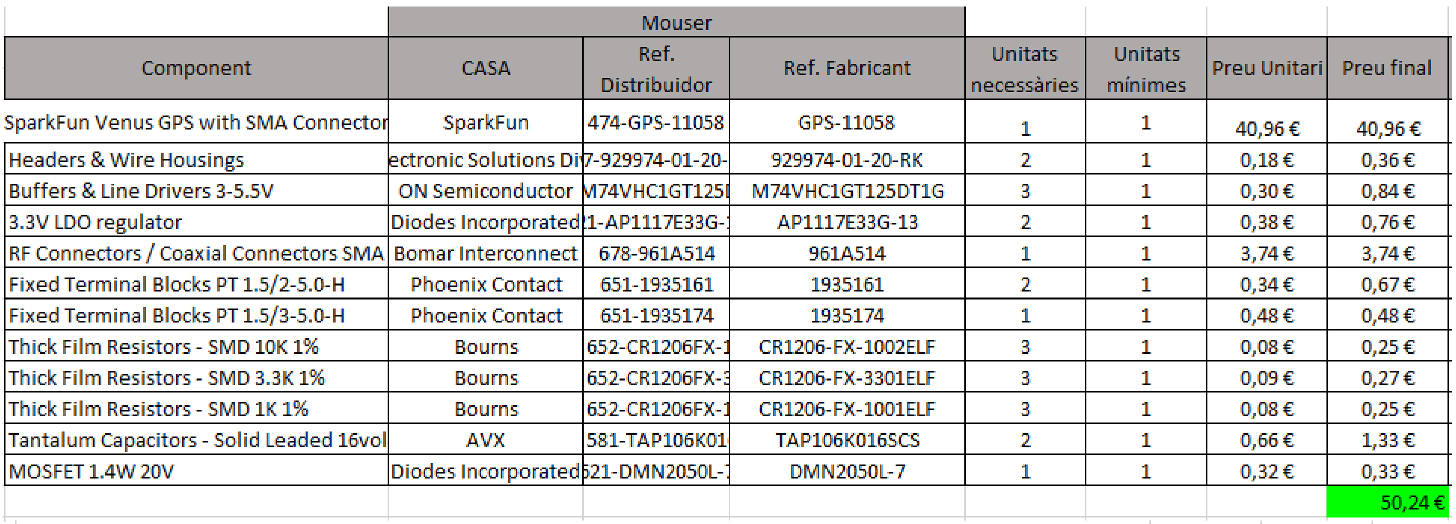
Do not turn on the tracker without a proper antenna because power not radiated by the HX1 will reflect back and cause overheating which might eventually fry the HX1 and/or the whole board.

Do not connect the trackuino board to the Arduino before uploading the sketch.

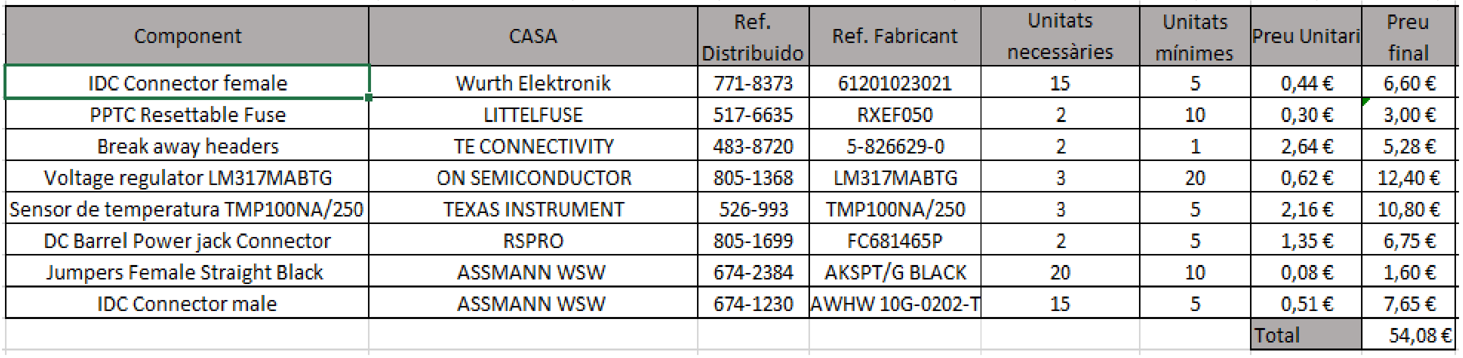
Do not transmit without an antenna to the antenna pin. Without an antenna, RF energy can reflect back into the module, damaging it


Power
IC2
REG1117
VIN VOUT
GND
C2
10uF
C1
10uF
+3V3
VIN
GND
R1
10K
A2
R2
3.3K
Radio
X1
BU-SMA-V
RFGND1
RFOUT
RFGND2
BODYGND1
BODYGND2
EN
VCC
0V
TXD
M2
HX1
EN-5V
+5V
TXD-5V
C3
1.0uF
C4
0.1uF
Level shifters
IC3P
IC4P
IC5P
+5V
+3V3
VCC
IC3
IC4
IC5
DNP
ON M74VHC1GT125DT1G
JP1
JP2
JP3
N$23
N$24
TXD-5V
TX-3V3
EN-5V
TX
D4
Connect
GPS
MOSI
MISO
CLK
CS
RXD
TXD
3.3V
SDA
SCL
TX1
RX1
PPS
NAV
GND2
VBAT
VENUS638FLPX
GPS
RX
J5
J5MM
TX-3V3
X5-1
X5-2
Vin
R6
200
T1
2N7002E
R5
R4
10K
Arduino
S1
RESET
J4
RESET
3V3
+5V
VIN
J3
AREF
D13
D12
D11
D10
D9
D8
J2
A2
A3
A4
A5
J1
D5
D4
D3
D2
RX
Terminal blocks
X2-1
X2-2
X4-1
X4-2
Buzzer
R3
1K
Q1
PMV20XN, PMV16UN, etc
D9
TITLE: trackuino-shield
Document Number:
REV:
Date: 05/05/2018 21:46
Sheet: 1/1

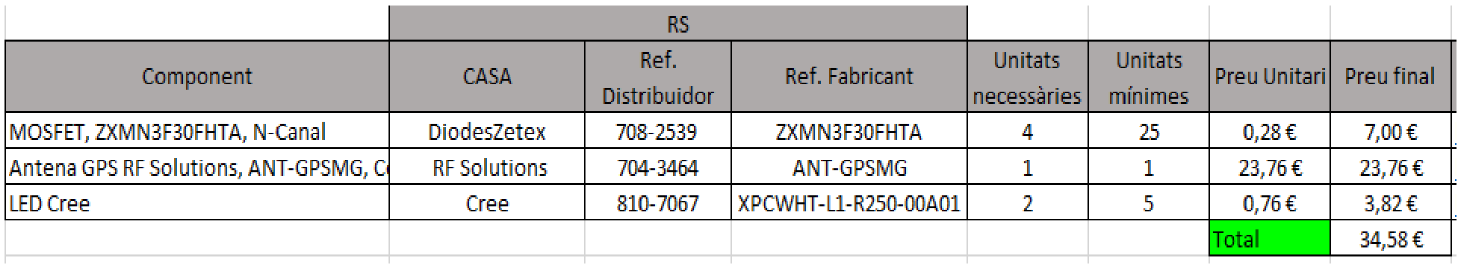
List of Materials



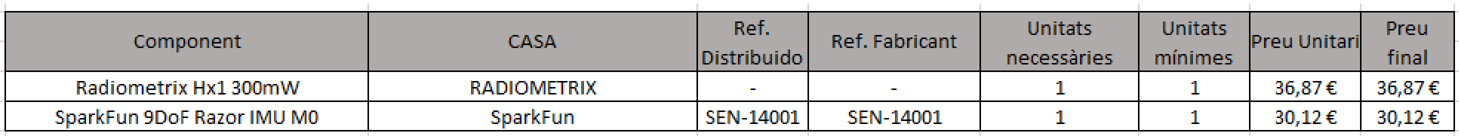
	Mouser						
Component	CASA	Ref. Distribuidor	Ref. Fabricant	Unitats necessàries	Unitats mínimes	Preu Unitari	Preu final
SparkFun Venus GPS with SMA Connector	SparkFun	474-GPS-11058	GPS-11058	1	1	40,96 €	40,96 €
Headers & Wire Housings	ectronic Solutions Di	7-929974-01-20-	929974-01-20-RK	2	1	0,18 €	0,36 €
Buffers & Line Drivers 3-5.5V	ON Semiconductor	M74VHC1GT125I	M74VHC1GT125DT1G	3	1	0,30 €	0,84 €
3.3V LDO regulator	Diodes Incorporated	1-AP1117E33G-	AP1117E33G-13	2	1	0,38 €	0,76 €
RF Connectors / Coaxial Connectors SMA	Bomar Interconnect	678-961A514	961A514	1	1	3,74 €	3,74 €
Fixed Terminal Blocks PT 1.5/2-5.0-H	Phoenix Contact	651-1935161	1935161	2	1	0,34 €	0,67 €
Fixed Terminal Blocks PT 1.5/3-5.0-H	Phoenix Contact	651-1935174	1935174	1	1	0,48 €	0,48 €
Thick Film Resistors - SMD 10K 1%	Bourns	652-CR1206FX-1	CR1206-FX-1002ELF	3	1	0,08 €	0,25 €
Thick Film Resistors - SMD 3.3K 1%	Bourns	652-CR1206FX-3	CR1206-FX-3301ELF	3	1	0,09 €	0,27 €
Thick Film Resistors - SMD 1K 1%	Bourns	652-CR1206FX-1	CR1206-FX-1001ELF	3	1	0,08 €	0,25 €
Tantalum Capacitors - Solid Leaded 16vol	AVX	581-TAP106K01	TAP106K016SCS	2	1	0,66 €	1,33 €
MOSFET 1.4W 20V	Diodes Incorporated	621-DMN2050L-	DMN2050L-7	1	1	0,32 €	0,33 €
							50,24 €



Component	CASA	Ref. Distribuido	Ref. Fabricant	Unitats necessàries	Unitats mínimes	Preu Unitari	Preu final
IDC Connector female	Wurth Elektronik	771-8373	61201023021	15	5	0,44 €	6,60 €
PPTC Resettable Fuse	LITTELFUSE	517-6635	RXEF050	2	10	0,30 €	3,00 €
Break away headers	TE CONNECTIVITY	483-8720	5-826629-0	2	1	2,64 €	5,28 €
Voltage regulator LM317MABTG	ON SEMICONDUCTOR	805-1368	LM317MABTG	3	20	0,62 €	12,40 €
Sensor de temperatura TMP100NA/250	TEXAS INSTRUMENT	526-993	TMP100NA/250	3	5	2,16 €	10,80 €
DC Barrel Power jack Connector	RSPRO	805-1699	FC681465P	2	5	1,35 €	6,75 €
Jumpers Female Straight Black	ASSMANN WSW	674-2384	AKSPT/G BLACK	20	10	0,08 €	1,60 €
IDC Connector male	ASSMANN WSW	674-1230	AWHW 10G-0202-T	15	5	0,51 €	7,65 €
						Total	54,08 €



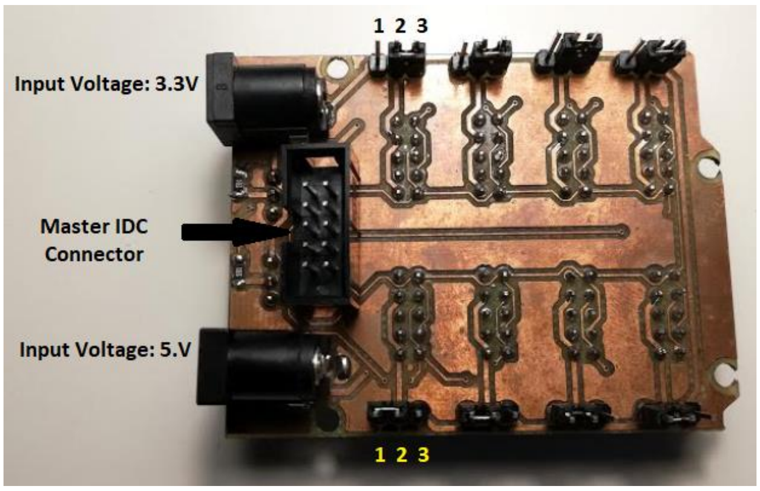
	RS						
Component	CASA	Ref. Distribuidor	Ref. Fabricant	Unitats necessàries	Unitats mínimes	Preu Unitari	Preu final
MOSFET, ZXMN3F30FHTA, N-Canal	DiodesZetex	708-2539	ZXMN3F30FHTA	4	25	0,28 €	7,00 €
Antena GPS RF Solutions, ANT-GPSMG, C	RF Solutions	704-3464	ANT-GPSMG	1	1	23,76 €	23,76 €
LED Cree	Cree	810-7067	XPCWHT-L1-R250-00A01	2	5	0,76 €	3,82 €
						Total	34,58 €



Component	CASA	Ref. Distribuido	Ref. Fabricant	Unitats necessàries	Unitats mínimes	Preu Unitari	Preu final
Radiometrix Hx1 300mW	RADIOMETRIX	-	-	1	1	36,87 €	36,87 €
SparkFun 9DoF Razor IMU M0	SparkFun	SEN-14001	SEN-14001	1	1	30,12 €	30,12 €

Description

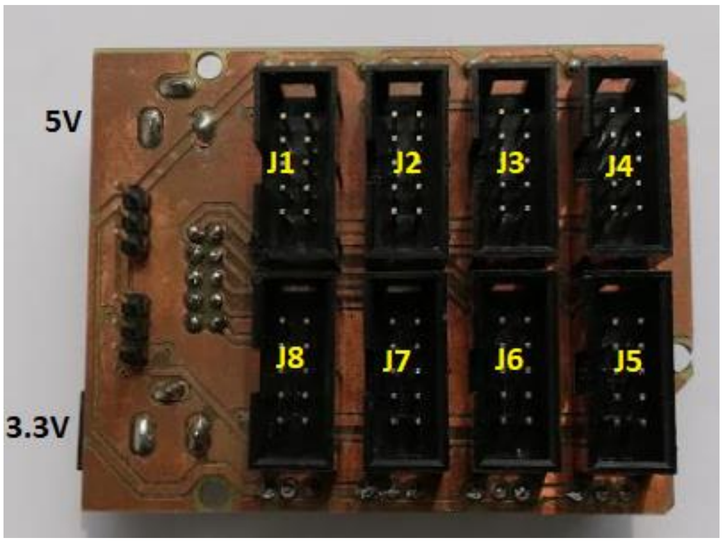
The expansion board is designed to connect all the sensors to the microcontroller Through the I2C serial Buses and analog pins. It is powered by the Electrical Power Subsystem through the Jack DC connectors and can run two different Voltage.



Bottom View Expansion board

The expansion board can run to two input voltage (3.3V and 5V) and has 8 sets of 3 headers which aim to set the desired voltage supply by connecting a jumper.

Every set of 3 headers is connected to one IDC connector (J1, J2, J3…., Jn). When the board is orientated like shown in the bottom view (figure above), the header's pins are numbered from 1 to 3.



When the pins 1 and 2 of each set of headers are connected, the table below

Top View Expansion Board

show the pinout configuration of the two types of connectors:

Table 1: I2C Pure Pinout

Pin	1	2	3	4	5	6	7	8	9	10
J1,2,3,4	SDA	5V	SCL	GND	-	-	SDA	SCL	5V	GND

Table 2: I2C & Analog – Pinout

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
J5,6,7,8	SDA	5V	SCL	GND	AN_X	AN_Y	SDA	SCL	5V	GND

When the pins 2 and 3 of each set of headers are connected, the table below show the pinout configuration of the two types of connectors:

Table 3: I2C Pure Pinout

Pin	1	2	3	4	5	6	7	8	9	10
J1,2,3,4	SDA	3.3V	SCL	GND	-	-	SDA	SCL	3.3V	GND

Table 4: I2C Pure Pinout

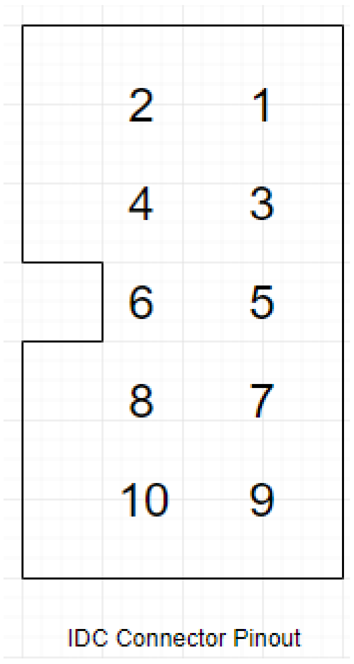
Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
J5,6,7,8	SDA	3.3V	SCL	GND	AN_X	AN_Y	SDA	SCL	3.3V	GND

Note: The pin 5 and 6 of the connectors of J1,2,3,4 are floating. And AN_X and AN_Y are reserved to connect analog sensor or devices that need to communicate to the MCU.

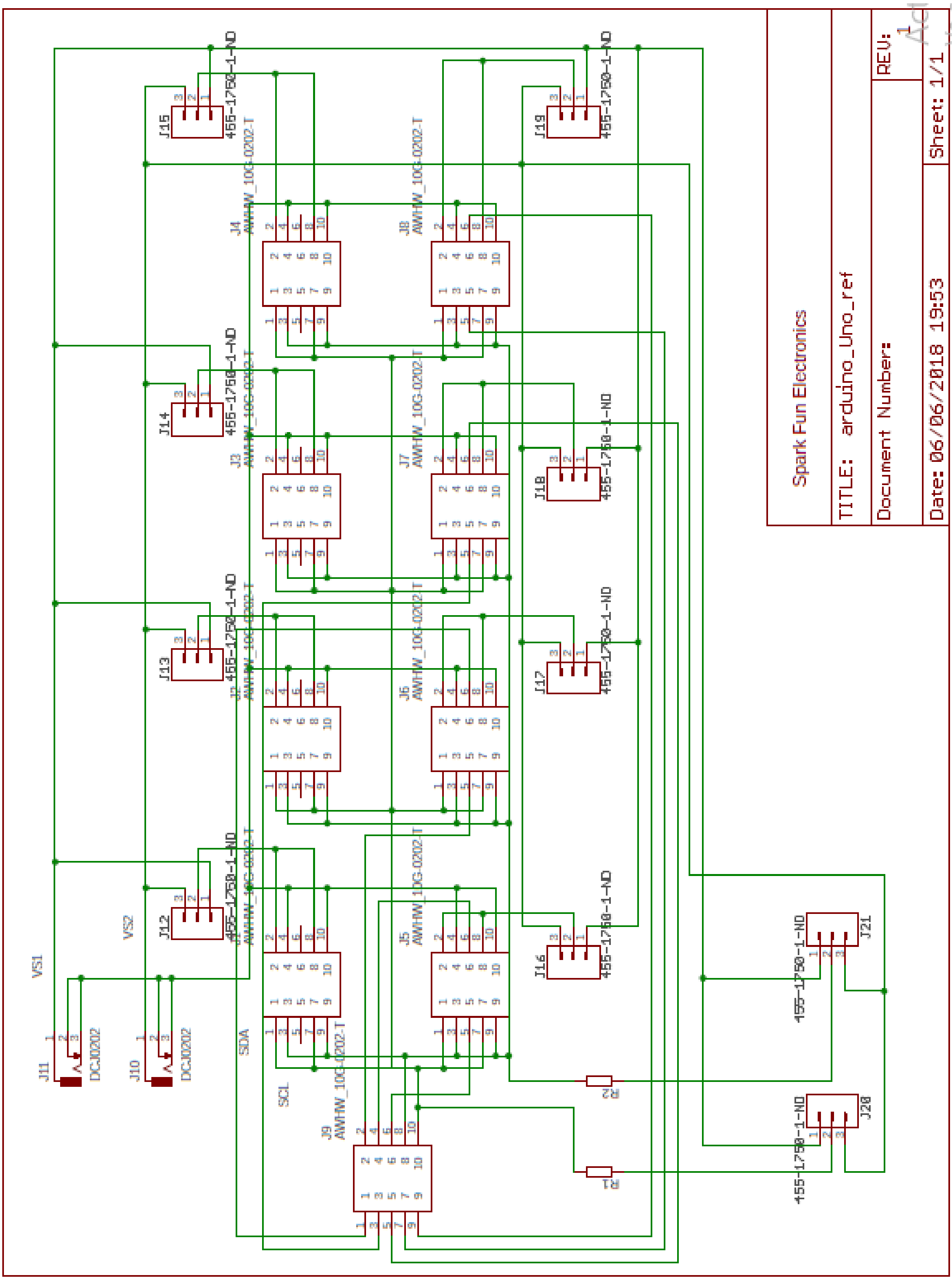
Table 4: The master connector Pinout

Pin Master	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pin Analog	$J6_6$	$J6_5$	$J7_5$	$J5_6$	$J7_6$	$J5_5$	$J8_5$	SCL	$J8_6$	SDA

Pinout Number



IDC Connector Pinout


Spark Fun Electronics
TITLE: arduino_Uno_ref
Document Number:
REV: 1
Date: 06/06/2018 19:53
Sheet: 1/1
J4 AWHW_10G-0202-T
J3 AWHW_10G-0202-T
J2 AWHW_10G-0202-T
J1 AWHW_10G-0202-T
J8 AWHW_10G-0202-T
J7 AWHW_10G-0202-T
J6 AWHW_10G-0202-T
J5 AWHW_10G-0202-T
J9 AWHW_10G-0202-T
J15 455-1750-1-ND
J14 455-1750-1-ND
J13 455-1750-1-ND
J12 455-1750-1-ND
J19 455-1750-1-ND
J18 455-1750-1-ND
J17 455-1750-1-ND
J16 455-1750-1-ND
J21 455-1750-1-ND
J20 455-1750-1-ND
J11 DCJ0202
J10 DCJ0202
VS1
VS2
SDA
SCL
R1
R2

Prototypes

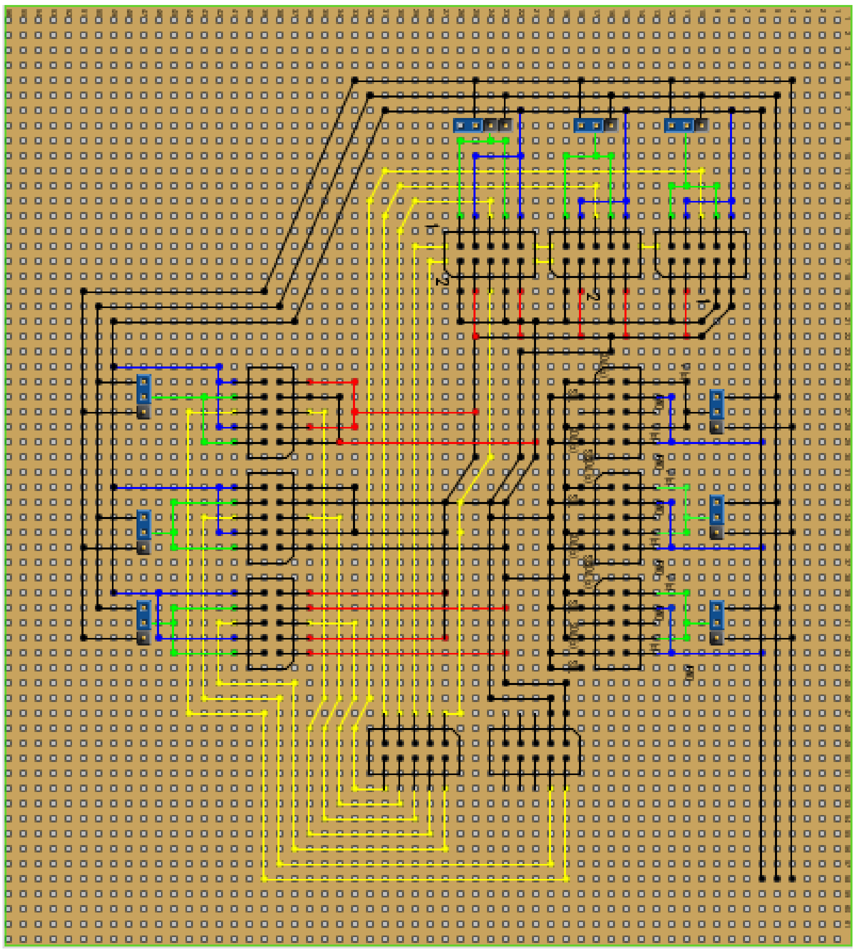


Figure A.1 Expansion Board - First Prototype

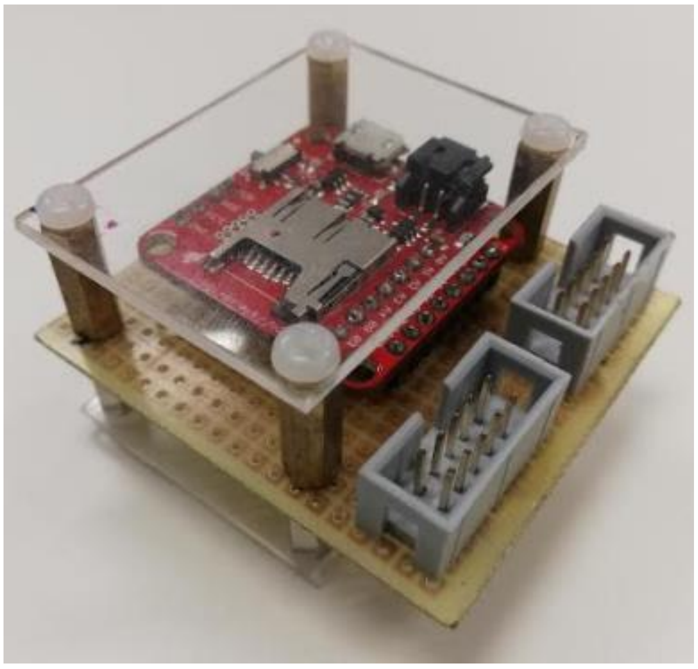


Figure A.2 Razor Adaptation - First Prototype

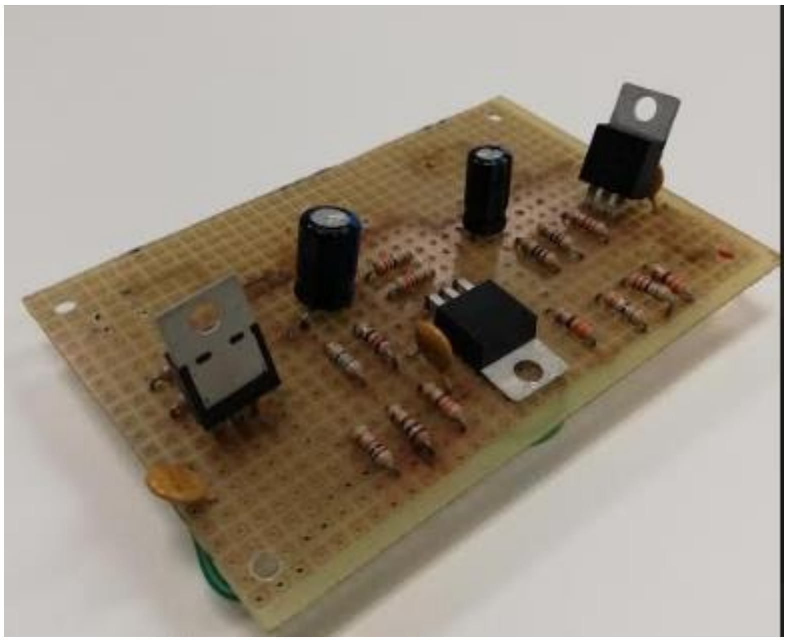


Figure A.3 Electrical Power Subsystem - First Prototype

CODE ARDUINO

SKETCH VENUS GPS TEST

```
#include <SoftwareSerial.h>

SoftwareSerial gpsSerial(10, 11); // RX, TX (TX not used)
const int sentenceSize = 80;

char sentence[sentenceSize];

void setup()
{
  Serial.begin(9600);
  gpsSerial.begin(9600);
}

void loop()
{
  static int i = 0;
  if (gpsSerial.available())
  {
    char ch = gpsSerial.read();
    if (ch != '\n' && i < sentenceSize)
    {
      sentence[i] = ch;
      i++;
    }
    else
    {
     sentence[i] = '\0';
     i = 0;
     displayGPS();
    }
  }
}

void displayGPS()
{
  char field[20];
  getField(field, 0);
  if (strcmp(field, "$GPRMC") == 0)
  {
    Serial.print("Lat: ");
    getField(field, 3);  // number
    Serial.print(field);
    getField(field, 4); // N/S
    Serial.print(field);
    Serial.print(" Long: ");
    getField(field, 5);  // number
    Serial.print(field);
    getField(field, 6);  // E/W
    Serial.println(field);
  }
}

void getField(char* buffer, int index)
{
  int sentencePos = 0;
  int fieldPos = 0;
  int commaCount = 0;
  while (sentencePos < sentenceSize)
  {
    if (sentence[sentencePos] == ',')
    {
      commaCount ++;
      sentencePos ++;
    }
    if (commaCount == index)
    {
      buffer[fieldPos] = sentence[sentencePos];
      fieldPos ++;
    }
    sentencePos ++;
  }
  buffer[fieldPos] = '\0';
}
```

BUZZER TEST

Buzzer.h

```
#ifndef __BUZZER_H__
#define __BUZZER_H__

void buzzer_setup();
void buzzer_on();
void buzzer_off();

#endif // ifndef __BUZZER_H__
```

Buzzer_avr.cpp

```
#ifdef AVR

#include "config.h"
```

```
#include "buzzer.h"
#include "pin.h"
#if (ARDUINO + 1) >= 100
#  include <Arduino.h>
#else
#  include <WProgram.h>
#endif
#include <stdint.h>
#include <avr/interrupt.h>
#include <avr/io.h>


// Module constants
static const unsigned long PWM_PERIOD = F_CPU / BUZZER_FREQ;
static  const  unsigned  long  ON_CYCLES  =  BUZZER_FREQ  *
BUZZER_ON_TIME;
static  const  unsigned  long  OFF_CYCLES  =  BUZZER_FREQ  *
BUZZER_OFF_TIME;
#if BUZZER_TYPE == 0  // active buzzer
static const uint16_t DUTY_CYCLE = PWM_PERIOD;
#endif
#if BUZZER_TYPE == 1  // passive buzzer
static const uint16_t DUTY_CYCLE = PWM_PERIOD / 2;
#endif

// Module variables
static volatile bool is_buzzer_on;
static volatile bool buzzing;
static volatile unsigned long alarm;

// Exported functions
void buzzer_setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
  pin_write(BUZZER_PIN, LOW);
  buzzing = false;
  is_buzzer_on = false;
  alarm = 1;

  // Top is ICR1 (WGM1=14), p.135
  TCCR1A = _BV(WGM11);
  TCCR1B = _BV(WGM13) | _BV(WGM12);

  // Set top to PWM_PERIOD
  ICR1 = PWM_PERIOD;

  // Enable interrupts on timer overflow
  TIMSK1 |= _BV(TOIE1);

  // Start the timer, no prescaler (CS1=1)
  TCCR1B |= _BV(CS10);
}

void buzzer_on()
{
  is_buzzer_on = true;
}

void buzzer_off()
{
  is_buzzer_on = false;
}

// Interrupt Service Routine for TIMER1. This is used to
switch between the
// buzzing and quiet periods when ON_CYCLES or OFF_CYCLES
are reached.
ISR (TIMER1_OVF_vect)
{
  interrupts();    // allow other interrupts (ie. modem)
  alarm--;
  if (alarm == 0) {
    buzzing = !buzzing;
    if (is_buzzer_on && buzzing) {
      switch(BUZZER_PIN) {
        case 9:
          // Non-inverting pin 9 (COM1A=2), p.135
          TCCR1A |= _BV(COM1A1);
          OCR1A = DUTY_CYCLE;
          break;
        case 10:
          // Non-inverting pin 10 (COM1B=2), p.135
          TCCR1A |= _BV(COM1B1);
          OCR1B = DUTY_CYCLE;
          break;
      }
      alarm = ON_CYCLES;
    } else {
      switch(BUZZER_PIN) {
        // Disable PWM on pin 9/10
        case 9:  TCCR1A &= ~_BV(COM1A1); break;
        case 10: TCCR1A &= ~_BV(COM1B1); break;
      }
      pin_write(BUZZER_PIN, LOW);
      alarm = OFF_CYCLES;
    }
  }
}

#endif // #ifdef AVR
```

Buzzer_pic32.cpp

```
#ifdef PIC32MX

#include "config.h"
#include "buzzer.h"
#include "pin.h"
#include <p32xxxx.h>
#include <plib.h>
#include <WProgram.h>

// Module constants
static  const  unsigned  long  PWM_PERIOD  =  F_CPU  /  8  /
BUZZER_FREQ;
static  const  unsigned  long  ON_CYCLES  =  BUZZER_FREQ  *
BUZZER_ON_TIME;
static  const  unsigned  long  OFF_CYCLES  =  BUZZER_FREQ  *
BUZZER_OFF_TIME;
#if BUZZER_TYPE == 0  // active buzzer
static const uint16_t DUTY_CYCLE = PWM_PERIOD;
#endif
#if BUZZER_TYPE == 1  // passive buzzer
static const uint16_t DUTY_CYCLE = PWM_PERIOD / 2;
#endif

// Module variables
static volatile bool is_buzzer_on;
static volatile bool buzzing;
static unsigned long alarm;

// Exported functions
void buzzer_setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
  pin_write(BUZZER_PIN, LOW);
  buzzing = false;
  is_buzzer_on = false;
  alarm = 1;

  // There are two timers capable of PWM, 2 and 3. We are
using 2 for the modem,
  // so use 3 for the buzzer.
  OpenTimer3(T3_ON | T3_PS_1_8, PWM_PERIOD);
  ConfigIntTimer3(T3_INT_ON | T3_INT_PRIOR_5);
}

void buzzer_on()
{
  is_buzzer_on = true;
}

void buzzer_off()
```

```
{
  is_buzzer_on = false;
}

// Interrupt Service Routine for TIMER 3. This is used to
switch between the
// buzzing and quiet periods when ON_CYCLES or OFF_CYCLES
are reached.
extern "C" void __ISR (_TIMER_3_VECTOR, ipl5) T3_IntHandler
(void)
{
  interrupts();   // allow other interrupts (ie. modem)
  alarm--;
  if (alarm == 0) {
    buzzing = !buzzing;
    if (is_buzzer_on && buzzing) {
      switch(BUZZER_PIN) {
        case 9:
          OpenOC4(OC_ON       |       OC_TIMER3_SRC       |
OC_PWM_FAULT_PIN_DISABLE, DUTY_CYCLE, 0);
          break;
        case 10:
          OpenOC5(OC_ON       |       OC_TIMER3_SRC       |
OC_PWM_FAULT_PIN_DISABLE, DUTY_CYCLE, 0);
          break;
      }
      alarm = ON_CYCLES;
    } else {
      switch(BUZZER_PIN) {
        case 9:  CloseOC4(); break;
        case 10: CloseOC5(); break;
      }
      alarm = OFF_CYCLES;
      pin_write(BUZZER_PIN, LOW);
    }
  }
  // Clear interrupt flag
  //  This  will  break  other  interrupts  and  millis()
(read+clear+write race condition?)
  //   IFS0bits.T3IF = 0; // DON'T!!
  // Instead:
  mT3ClearIntFlag();
}

#endif // #ifdef PIC32MX
```

Config.h

```
#ifndef __CONFIG_H__
#define __CONFIG_H__


// -------------------------------------------------------------
------------------
// THIS IS THE TRACKUINO FIRMWARE CONFIGURATION FILE. YOUR
CALLSIGN AND
// OTHER SETTINGS GO HERE.
//
// NOTE: all pins are Arduino based, not the Atmega chip.
Mapping:
// http://www.arduino.cc/en/Hacking/PinMapping
// -------------------------------------------------------------
------------------



// -------------------------------------------------------------
------------------
// APRS config (aprs.c)
// -------------------------------------------------------------
------------------

// Set your callsign and SSID here. Common values for the
SSID are
// (from http://zlhams.wikidot.com/aprs-ssidguide):
//
// - Balloons:  11
// - Cars:       9
// - Home:       0
// - IGate:      5
#define S_CALLSIGN      "MYCALL"
#define S_CALLSIGN_ID   11

// Destination callsign: APRS (with SSID=0) is usually okay.
#define D_CALLSIGN      "APRS"
#define D_CALLSIGN_ID   0

// Digipeating paths:
//    (read    more    about    digipeating    paths    here:
http://wa8lmf.net/DigiPaths/ )
// The recommended digi path for a balloon is WIDE2-1 or
pathless. The default
// is pathless. Uncomment the following two lines for WIDE2-
1 path:
#define DIGI_PATH1      "WIDE2"
#define DIGI_PATH1_TTL  1

// APRS comment: this goes in the comment portion of the
APRS message. You
```

```
// might want to keep this short. The longer the packet, the
more vulnerable
// it is to noise.
#define APRS_COMMENT   "Trackuino reminder: replace callsign
with your own"


// --------------------------------------------------------
------------------
// AX.25 config (ax25.cpp)
// --------------------------------------------------------
------------------

// TX delay in milliseconds
#define TX_DELAY      300

// --------------------------------------------------------
------------------
// Tracker config (trackuino.pde)
// --------------------------------------------------------
------------------

// APRS packets are slotted so that multiple trackers can be
used without
// them stepping on one another. The transmission times are
governed by
// the formula:
//
//         APRS_SLOT (seconds) + n * APRS_PERIOD (seconds)
//
// When launching multiple balloons, use the same APRS_PERIOD
in all balloons
// and set APRS_SLOT so that the packets are spaced equally
in time.
// Eg. for two balloons and APRS_PERIOD = 60, set APRS_SLOT
to 0 and 30,
// respectively. The first balloon will transmit at 00:00:00,
00:01:00,
// 00:02:00, etc. and the second balloon will transmit at
00:00:30, 00:01:30,
// 00:02:30, etc.
#define APRS_SLOT     0     // seconds. -1 disables slotted
transmissions
#define APRS_PERIOD   60    // seconds

// GPS baud rate (in bits per second). This is also the baud
rate at which
// debug data will be printed out the serial port.
#define GPS_BAUDRATE  9600
```

```
// -------------------------------------------------------------
------------------
// Modem config (afsk.cpp)
// -------------------------------------------------------------
------------------

// AUDIO_PIN is the audio-out pin. The audio is generated by
timer 2 using
// PWM, so the only two options are pins 3 and 11.
// Pin 11 doubles as MOSI, so I suggest using pin 3 for PWM
and leave 11 free
// in case you ever want to interface with an SPI device.
#define AUDIO_PIN       3

// Pre-emphasize the 2200 tone by 6 dB. This is actually
done by
// de-emphasizing the 1200 tone by 6 dB and it might greatly
improve
// reception at the expense of poorer FM deviation, which
translates
// into an overall lower amplitude of the received signal.
1 = yes, 0 = no.
#define PRE_EMPHASIS    1

// -------------------------------------------------------------
------------------
// Radio config (radio_hx1.cpp)
// -------------------------------------------------------------
------------------

// This is the PTT pin
#define PTT_PIN           4

// -------------------------------------------------------------
------------------
// Sensors config (sensors.cpp)
// -------------------------------------------------------------
------------------

// Most of the sensors.cpp functions use internal reference
voltages (either
// AVCC or 1.1V). If you want to use an external reference,
you should
// uncomment the following line:
//
// #define USE_AREF
//
// BEWARE! If you hook up an external voltage to the AREF
pin and
// accidentally set the ADC to any of the internal
references, YOU WILL
```

```
// FRY YOUR AVR.
//
// It is always advised to connect the AREF pin through a
pull-up resistor,
// whose value is defined here in ohms (set to 0 if no pull-
up):
//
#define AREF_PULLUP           4700
//
// Since there is already a 32K resistor at the ADC pin, the
actual
// voltage read will be VREF * 32 / (32 + AREF_PULLUP)
//
// Read more in the Arduino reference docs:
//
http://arduino.cc/en/Reference/AnalogReference?from=Referen
ce.AREF

// Pin mappings for the internal / external temperature
sensors. VS refers
// to (arduino) digital pins, whereas VOUT refers to
(arduino) analog pins.
#define INTERNAL_LM60_VS_PIN     6
#define INTERNAL_LM60_VOUT_PIN   0
#define EXTERNAL_LM60_VS_PIN     7
#define EXTERNAL_LM60_VOUT_PIN   1

// Units for temperature sensors (Added by: Kyle Crockett)
// 1 = Celsius, 2 = Kelvin, 3 = Fahrenheit
#define TEMP_UNIT 1

// Calibration value in the units selected. Use integer only.
#define CALIBRATION_VAL 0

// Resistors divider for the voltage meter (ohms)
#define VMETER_R1       10000
#define VMETER_R2       3300

// Voltage meter analog pin
#define VMETER_PIN      2

// --------------------------------------------------------
------------------
// Buzzer config (buzzer.cpp)
// --------------------------------------------------------
------------------

// Type of buzzer (0=active, 1=passive). An active buzzer is
driven by a
// DC voltage. A passive buzzer needs a PWM signal.
#define BUZZER_TYPE             0
```

```
// When using a passive buzzer, specify the PWM frequency
here. Choose one
// that maximizes the volume according to the buzzer's
datasheet. Not all
// the frequencies are valid, check out the buzzer_*.cpp
code. On Arduino,
// it must be between L and 65535, where L = F_CPU / 65535
and F_CPU is the
// clock rate in hertzs. For 16 MHz Arduinos, this gives a
lower limit of
// 245 Hz.
#define BUZZER_FREQ             895     // Hz

// These are the number of seconds the buzzer will stay
on/off alternately
#define BUZZER_ON_TIME          1       // secs
#define BUZZER_OFF_TIME         2       // secs

// This option disables the buzzer above BUZZER_ALTITUDE
meters. This is a
// float value, so make it really high (eg. 1000000.0 = 1
million meters)
// if you want it to never stop buzzing.
#define BUZZER_ALTITUDE          3000.0  // meters (1 ft =
0.3048 m)

// The options here are pin 9 or 10
#define BUZZER_PIN              9

// -------------------------------------------------------
------------------
// Debug
// -------------------------------------------------------
------------------

// This is the LED pin (13 on Arduinos). The LED will be on
while the AVR is
// running and off while it's sleeping, so its brightness
gives an indication
// of the CPU activity.
#define LED_PIN                 13

// Debug info includes printouts from different modules to
aid in testing and
// debugging.
//
// Some of the DEBUG modes will cause invalid modulation, so
do NOT forget
// to turn them off when you put this to real use.
//
```

```
// Particularly the DEBUG_AFSK will print every PWM sample
out the serial
// port, causing extreme delays in the actual AFSK
transmission.
//
// 1. To properly receive debug information, only connect
the Arduino RX pin
//     to the GPS TX pin, and leave the Arduino TX pin
disconnected.
//
// 2. On the serial monitor, set the baudrate to GPS_BAUDRATE
(above),
//    usually 9600.
//
// 3. When flashing the firmware, disconnect the GPS from
the RX pin or you
//    will get errors.

#define DEBUG_GPS      // GPS sentence dump and checksum
validation
// #define DEBUG_AX25   // AX.25 frame dump
// #define DEBUG_MODEM  // Modem ISR overrun and profiling
#define DEBUG_AFSK   // AFSK (modulation) output
#define DEBUG_RESET  // AVR reset
// #define DEBUG_SENS   // Sensors


#endif
```