UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

DESIGN OF A REMOTE INSTRUMENTATION SYSTEM AND A DEDICATED CIRCUIT FOR THE CONFIGURATION AND QUALITY ANALYSIS OF RF POWER AMPLIFIERS

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Marc Pradas

In partial fulfilment

of the requirements for the degree in

(AudioVisual systems) **ENGINEERING**

Advisor: Mireya Fernandez

Barcelona, January 2017

Abstract

The author was working at the Tryo Communications in the i+d department, after some analysis of the fabrication procedures a lag of automation in a specific configuration process for powers amplifiers was detected.

After the specifications and requirements compilation, a deeper analysis was done in order to design a system that would involve measure instruments, a server, an own PCB and power amplifiers. That system should be able to do all the configuration process by its own.

This initiative is a hardware and software solution that truly has 0 cost in comparison with the benefits provided with the integration of that technologies to the enterprise production chain. Most of the repetitive processes are becoming automated by machines to improve efficiency and serial manufacturing.

This is a concept that could be applied in many terms but this project is focused on the automation using electronics applied to RF and it also contains a software integration.

Resumen

El autor trabajaba en Tryo Communications en el departamento de i + d, tras un análisis de los procedimientos de fabricación se detectó una falta de automatización en un proceso de configuración específico para amplificadores de potencia.

Después de la compilación de especificaciones y requerimientos, se realizó un análisis más profundo para diseñar un sistema que involucraría instrumentos de medida, un servidor, una propia PCB y amplificadores de potencia. Ese sistema debe ser capaz de hacer todo el proceso de configuración por su cuenta.

Esta iniciativa es una solución de hardware y software que realmente tiene 0 costo en comparación con los beneficios proporcionados con la integración de esas tecnologías a la cadena de producción de la empresa. La mayoría de los procesos repetitivos están siendo automatizados por las máquinas para mejorar la eficiencia y la fabricación en serie.

Este es un concepto que se podría aplicar en muchos términos, pero este proyecto se centra en la automatización utilizando la electrónica aplicada a RF y también contiene una integración de software.

Acknowledgements and dedication

This page of the document is dedicated to all the people that helped in the project in all the aspects. There are some anonymous engineers if Tryo Communications from the i+d department as well as the production department that had to be mentioned in the dedication. Its also dedicated to the project tutors that guided me through this complicated tasks. It is also dedicated to the university that gave me the necessary knowledge and motivation.

Revision history and approval record

Revision	Date	Purpose
0	15/01/2017	Document creation
1	20/01/2017	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Marc Pradas	marc.pradas.barro@gmail.com
Mireya Fernandez	
Juan Miguel Vives	

Written by:		Reviewed and approved by:	
Date	15/01/2017	Date	20/01/2017
Name	Marc Pradas	Name	Mireya Fernandez
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

Each table in the thesis must be listed in the “List of Tables” and each must be given a page number for its easy location.

1. Introduction

1.1 Statement of purpose (objectives).

The main idea of the work emerged in an enterprise called Tryo Communications on the environment of the I+D department: This enterprise is mainly dedicated on the generation of highly qualified RF equipment. There are two main products produced on the factory: Transmitters (DAB, DVBT, DVBT2) and power amplifiers (VHF, UHF bands).

After an extensive analysis of the manufacturing processes of those devices, the author of this project realized that there was a need of acceleration and automation of a configuration protocol. The goal of this idea was to automate the configuration of all RF power limiters and all the ADC converters of a power amplifier, and so on, this was an objective that only could be reached with a hardware and software complete system.

1.2 Requirements and specifications.

1.2.1 Project requirements:

- Server (Windows OS executing the visual studio application), in order to make it work, with all the program functionalities it should be connected in the enterprise domain.
- Generation of an electronics board (see its requirements below***).
- Visual Studio software for the creation of a windows program.
- Use of PCAD-2006 software for the schematic design and PCB generation.
- Use of Arduino IDE software to program the microprocessor.
- IP(Ethernet), Serial Port (DB-9), USB connectivity between devices.
- Use of Aglient Wattimeter and Spectrum analyzers.

*****Electronic board requirements:**

- Arduino MICRO 5V with USB connectivity.
- PCB 4-layer with multiple RF paths.
- SMD Components(condensers, relays, coils, resistances, transistors, diodes).
- Coaxial switch.
- 1 DB-9, 2 SMA, 4 SMB connectors.

1.2.2 Project specifications:

- At least the minimum windows version in which the program can be executed is windows 7.
- The manufacturing process off the board should be done with some specific micro-welding tools. As well as, a machine to generate printed circuits would be needed to create the PCB.
- The Wattimeter used shall be from model: U2000 series. And have a measurement error of less than: +-0.05dB (RMS power precision for high frequency RF signal).
- The Spectrum Analyzer shall be from model: N9010A. And have measurement error of less than: +-0.3dB. (bandwidth obtained power)
- The board microprocessor should have at least 15 I/O controllable pins, as well as, a COM port in order to communicate with the server, the used COM connection is USB.
- The board shall include 4 automotive relays and all of them with a state indicator wich is obtained using another relay connected to each respective relays.
- The configured equipment 'Power amplifiers' have to include 'Gain control' board used to calibrate the inputs and outputs of the device and adjust its power limiters.
- The supply of the board is given through the USB and an auxiliary entrance of 12V.

1.3 Methods and procedures

A Power Amplifier contains a RF input that is directly passed through a 'Gain Control' programmable device used to regulate the output power of the PA, as well as activating and calibrate all the device power, current and voltage limiters or sensors. In order to do this, 'Gain control' also have two RF samples coming from an internal coupler of the PA.

The objective of the project only could be reached with the manufacturing of an electronic board that had to contain RF inputs, High RF Relays to commutate the carried signals with the minimum losses and a coaxial commuter to make the effect of power reflection connecting it to an impedance load. The best way to control the electronic Relays or commuters was using a microprocessor and adapting the signals sent from the pins of this micro to the operating range of the required device.

As the introduction of a microprocessor on the architecture of the board was totally necessary the programming of a firmware had to be done. 'Arduino micro 5V' that is a pinned programmable board, was used as the microprocessor and it was programmed

with Arduino IDE software, this platform only supports its own language that is mainly based and compatible with C.

The program used to develop the schematic and the PCB design of the electronic circuit was P-CAD 2006 it is an older version of the well known Altium Designer. After the PCB design obviously it should be manufactured, so some providers where searched. Once the best offer had been accepted by the department manager, the Gerber files for the manufacturability of the board were generated.

Therefore, the finalization of the hardware ended connecting the board to all the required devices. A whole system composed by the PCB, a computer Server, a potentiometer, a spectrum analyser and one IP switch was the final necessary hardware for the project execution.

In order to make it work the following connections had to be done: 3 Ethernet connections to the IP switch and coming from Server, PA and the Spectrum analyser. Four SMB connections from some internal RF signals samples of the PA's to the PCB, two SMA RF input from a transmitter or a signal generator. One USB connection between the PCB and the computer. Three N connections all coming from the coaxial commuter, where a RF coupler is connected too, the output sample of the coupler to a Spectrum Analyser, in the input sample of the coupler a wattimeter is inserted and finally the output of the PA is connected to the coaxial commuter, the coupler must have a plugged impedance load, this commuter should be also connected with a DB-9 connector with the computer that is executing the software (Server).

See above on the graphical description:

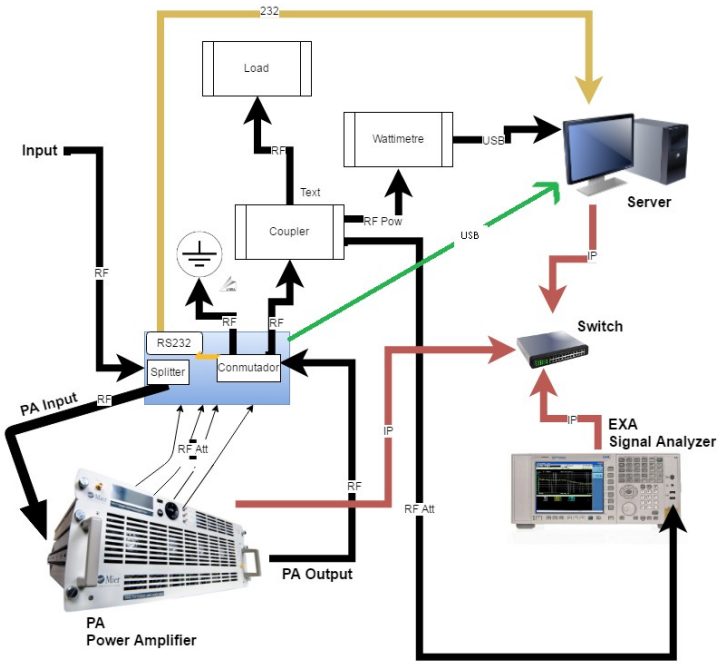


Figure 1: System connection diagram

Finally with all the hardware project segment closed a Windows compatible Server application was also programmed with the purpose of controlling the electronic board as well as all the electronic devices used in this process (Spectrum analysers, Wattimeters, and PA's).

This server application was implemented with Visual c#, at the beginning the author began to program some libraries to control spectrum analysers concretely Aglient (Keysight) N9010A through the IP port using SCPI protocol, at the same way a function to obtain potentiometers RF RMS power was programmed too, the communication port was the USB using SCPI protocol as well.

Talking about the front-end structure consists in a program of automation, it means that has some utils to extract data from analysis devices in a useful and a reliable way. The main purposes of the visual studio program are to take control of the switching electronic board previously designed, process the data received from the microprocessor, the spectrum analyser and the potentiometer, before sending the desired configuration sets through IP directly to the Power amplifiers (PA's).

This Windows compatible application is really an easy-to-use program, once the factory technicians have done the corresponding connections mentioned before, they should execute the program. An initial window is shown, there, they are able to select through an IP drop down selector the desired Spectrum analyser, this window also tracks the USB connections to the board and a wattimeter automatically. By the time everything is right the technician would be able to see a pushbutton called 'Power Amplifiers Autoconfiguration'. This button as well as the 'Start button' should be clicked and the process starts to do every step in an automatic way. After the execution of the configuration we can see a resume where all the steps are compiled, or an error message if any of the communications had failed.

Work Packages, Tasks and Milestones

Project:**AUTOCONFIG SYS**	WP ref: (WP1)	
Major constituent: SW	Sheet 1 of 6	
Short description: Visual studio structure programming for the server communication Planned start: 3/09/2016 Planned end:18/10/2016	Planned start: 3/09/2016 Planned end:18/10/2016	
	Start event:3/09/2016 End event:18/10/2016	
Internal task T1: libraries implementation Internal task T2: initial GUI structure Internal task T3: device communication tests	Deliverables:	Dates:

Project:**AUTOCONFIG SYS**	WP ref: (WP2)	
Major constituent: Electronics desing	Sheet 2 of 6	
Short description: Pcad Schematic desing	Start event: 24/09/2016 End event: 30/10/2016	
	Start event: 24/09/2016 End event: 30/10/2016	
Internal task T1: documentation and components specifications Internal task T2: Board schematic Design Internal task T3: Simulations	Deliverables:	Dates:

Project: **AUTOCONFIG SYS**	WP ref: (WP3)	
Major constituent: SW	Sheet 3 of 6	
Short description: Firmware implementation	Planned start: 30/10/2016 Planned end: 24/11/2016	
	Start event: 30/10/2016 End event: 9/12/2016	
Internal task T1: Creation of a communication protocol Internal task T2: Arduino Implementation of the micro-controller behavior (firmware).	Deliverables:	Dates:

Project: **AUTOCONFIG SYS**	WP ref: (WP4)	
Major constituent: Electronics desing	Sheet 4 of 6	
Short description: PCB design	Planned start: 22/11/2016 Planned end:22/12/2016	
	Start event: 8/12/2016 End event: 4/01/2016	
Internal task T1: Microstrip calculation of RF paths Internal task T2: Components positioning Internal task T3: Copper paths tracing and DXF generation	Deliverables:	Dates:

Project: **AUTOCONFIG SYS**	WP ref: (WP5)	
Major constituent: SW	Sheet 5 of 6	
Short description: Visual Studio server implementation	Planned start: 12/12/2016 Planned end: 3/01/2017	
	Start event: 18/12/2016 End event: 06/01/2016	
Internal task T1: Programming of the set of instructions that should be sent from the server to all the system components Internal task T2: Programming of the front-end graphical interface of the application	Deliverables:	Dates:

Project: **AUTOCONFIG SYS**	WP ref: (WP6)	
Major constituent: System testing	Sheet 6 of 6	
Short description: Once the design and manufacturing part project is finished the system should be tested with lots of devices in order to analyze its performance.	Planned start: 7 /01/2017 Planned end: 16/01/2017	
	Start event:18 /01/2017 End Event: 25/01/2017	
Internal task T1: Test the factory performance evaluating the parameters of the outcome devices.	Deliverables:	Dates:

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1	Libraries programming	-	1-2-3
1	2	Gui structure	-	4-5
1	3	Com tests	-	5
2	2	Docum & specs	-	3-4-5
2	2	Board schematic design	-	5-6-7
2	3	Simulations	-	7
3	1	Programming new serial protocol	-	8
3	2	Firmware programming	-	9-10
4	1	Microstrip RFdesign	-	10
4	2	Components positioning	-	11-12
4	3	Cooper paths and vias	-	12-13
5	1	Set of instructions	-	13-14
5	2	GUI Programming	-	14-15
6	1	Factory test	-	16

In the table above, the column of Date contain the week numbers which the tasks are done since the project start.

Time Plan (Gantt diagram)

	1st week	2nd week	3rd week	4th week	5th week	6th week	7th week	8th week	9th week	10th week	11th week	12th week	13th week	14th week	15th week	16th week
Visual studio structure programming for the server communication																
Libraries implementation	x	x	x													
Initial GUI structure				x	x											
Device communication tests					x											
P-cad Schematic design																
Documentation and components specs			x	x	x											
Board schematic design				x	x	x	x									
Voltage simulations							x									
Arduino IDE Firmware implementation																
Creation of a communication protocol								x	x							
Firmware behavior programming									x	x						
P-cad PCB design																
Microstrip calculation of RF paths										x						
Components positioning											x	x				
Cooper paths tracing, vias insertion, GND and supply planes.											x	x	x			
Visual Studio server implementation																
Set of instructions programming													x	x		
GUI development													x	x	x	
Manufacturing part																
PCB external fabrication													x	x		
Components soldering														x	x	
System testing																
Evaluation of factory performance																x

2. State of the art and theoretical background

2.1. Application of electronics systems on the industry

2.1.1 Micro-controller based technologies

The following illustration shows the block diagram of a typical micro-controller. All components are connected via an internal bus and are all integrated on one chip. The modules are connected to the outside world via I/O pins.

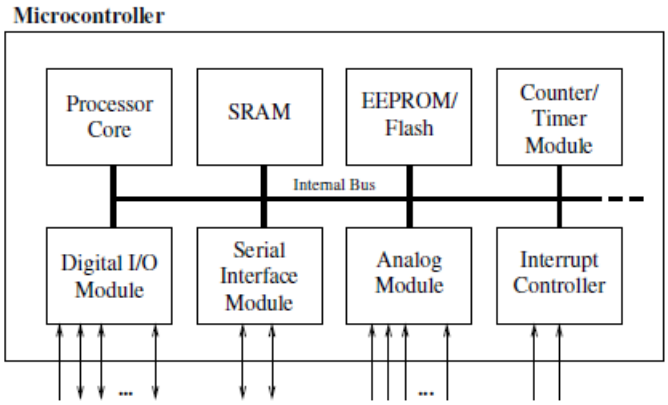


Figure 2: Basic layout of a micro-controller.

The following list contains the modules typically found in a microcontroller:

Processor Core: The CPU of the controller. It contains the arithmetic logic unit, the control unit, and the registers (stack pointer, program counter, accumulator register, register file,).

Memory: The memory is sometimes split into program memory and data memory. In larger controllers, a DMA controller handles data transfers between peripheral components and the memory.

Interrupt Controller: Interrupts are useful for interrupting the normal program flow in case of (important) external or internal events. In conjunction with sleep modes, they help to conserve power.

Timer/Counter: Most controllers have at least one and more likely 2-3 Timer/Counters, which can be used to timestamp events, measure intervals, or count events.

Many controllers also contain PWM (pulse width modulation) outputs, which can be used to drive motors or for safe breaking (anti lock brake system, ABS). Furthermore the PWM output can, in conjunction with an external filter, be used to realize a cheap digital/analog converter.

Digital I/O: Parallel digital I/O ports are one of the main features of micro-controllers. The number of I/O pins varies from 3-4 to over 90, depending on the controller family and the controller type.

To summarize, a microcontroller is a (stripped-down) processor which is equipped with memory, timers, (parallel) I/O pins and other on-chip peripherals. The driving element

behind all this is cost: Integrating all elements on one chip saves space and leads to both lower manufacturing costs and shorter development times. This saves both time and money, which are key factors in embedded systems. Additional advantages of the integration are easy upgradability, lower power consumption, and higher reliability, which are also very important aspects in embedded systems. On the downside, using a micro-controller to solve a task in software that could also be solved with a hardware solution will not give you the same speed that the hardware solution could achieve. Hence, applications which require very short reaction times might still call for a hardware solution. Most applications, however, and in particular those that require some sort of human interaction (microwave, mobile phone), do not need such fast reaction times, so for these applications micro-controllers are a good choice.

2.1.2 Electromechanics

In engineering, electromechanics combines electrical and mechanical processes and procedures drawn from electrical engineering and mechanical engineering. Electrical engineering in this context also encompasses electronics engineering.

Devices which carry out electrical operations by using moving parts are known as electromechanical. Strictly speaking, a manually operated switch is an electromechanical component, but the term is usually understood to refer to devices which involve an electrical signal to create mechanical movement, or mechanical movement to create an electric signal. Often involving electromagnetic principles such as in relays, which allow a voltage or current to control other, usually isolated circuit voltage or current by mechanically switching sets of contacts, and solenoids, by which a voltage can actuate a moving linkage as in solenoid valves.

.2.1 Relays

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal.

RF switches can be categorized as absorptive or reflective. Absorptive switches use a 50-Ω load termination in each output port which results in a low VSWR (voltage standing wave ratio) in both on and off states. In the terminated port, the terminating resistance absorbs the incident signal energy that would otherwise reflect from an unterminated port. When there is a signal on the input that must propagate through the switch, the open port is disconnected from the termination so all the signal energy can propagate through. Absorptive switches are used in applications where it's important to minimize reflections back to the RF source. Otherwise, reflective switches have no termination resistors. So their open ports have a lower insertion loss.

2.2. IP remote control for multiple type of devices

2.2.1 IVI.VISA libraries / IO libraries Suite

Keysight's I/O offerings are bundled into a single product called the Keysight IO Libraries Suite. It contains everything you need to get started controlling your instrument. The IO

Libraries Suite is included with each instrument and I/O interface that you purchase from Keysight.

An I/O library is software that runs on your computer to establish communication with an instrument. This communication could be over GPIB, USB, LAN, RS-232, VXI, or some other physical medium. If you write software, or use prewritten software, to control your instruments they will probably use one of the I/O libraries mentioned below. Instrument drivers, which provide a higher-level interface to the instrument, use I/O libraries internally to send and receive information from the instrument.

Starting with IO Libraries Suite 15.5, all of the Keysight I/O Libraries are available for use in both 32-bit and 64-bit systems. You can find specific information on how to use the libraries on 64-bit systems by searching the IO Libraries Suite 15.5 Help file for the topic "64-bit".

2.2.2 SCPI through IP for RF instruments

Socket connections for instruments are a convenient way to communicate with LAN enabled instrument without going through any I/O libraries. They are easy to use and are very fast for transporting large blocks of data. Plus sockets are supported on a wide variety of OS's and machines.

Agilent instruments have standardized on using port 5025 for SCPI socket services. Once a connection is made you simply send the SCPI strings to the instrument and read back responses over the socket connection. All "Program Messages" must be terminated with a newline for the message to be parsed. All responses that are read will also have a newline at the end.

Socket connections support a control connection. This second connection is used by client to send Device Clear and to listen for Service Requests. The initial connection is used to send and receive ASCII/SCPI commands, queries, and query responses. The user needs to query the initial connection for its control connection port number.

2.2.3 Formatted 488 through USB port

One of the primary ways to communicate with resources on INSTR sessions is to use the I488FormattedIO interface for formatted I/O. VISA COM I/O comes with a basic Formatted I/O Component that provides 488.2-style formatted I/O capabilities.

Currently there is only one interface with formatted I/O services defined, the Basic Formatted I/O component and the IFormattedIO488 interface. It is difficult to provide a complete formatted I/O solution as part of this standard due to the different formatted I/O paradigms of the various COM client platforms and the requirement that this be a freely distributable component. There are some specific programming rules above the use of the library functionalities. Id could be found at *reference* [12].

2.3.4 Instruments and functonalities

U2001A power meter

The power meters or wattimeters are really important in this project, the automation system is based on the carriage of specific RF signals that have a defined power trough the project's board calibrated paths, and then, set the converters parameters with the power measurements obtained from the power meters.

This devices mainly have to be plugged to a RF coupler and on the other hand to the computer with an USB connection (for supply and communication), the adjusted parameters are average, center frequency and zero calibration.

N9010A Spectrum analyzers

In this work they are used through IP with IVI.VISA libraries and its given ResourceManager objects. Basically the server realizes a capture of the spectrum centred at a given frequency and specific span, in order to appreciate the shoulders and flatness of the bandpass amplifier frequency response.

2.3. RF Electronic design

2.3.1 RF coupled-line couplers

Coupled lines are used in couplers (usually quadrature couplers) as well as transmission line filters. Coupled line couplers are not "DC connected", as opposed to "direct coupled" couplers such as the Wilkinson and the branchline. Coupled lines occur when two transmission lines are close enough in proximity so that energy from one line passes to the other. Usually we are talking about lines that are coupled over a quarterwave section, or multiple sections. Lines can be coupled in at least three ways:

- Edge coupled
- Broadside coupled
- End coupled (used in filters but not used in couplers)

In order to make a quadrature coupled-line coupler you need to couple a quarter-wave section; end-coupled structures are not useful in this case. That leaves two broad categories of coupled line couplers, edge coupled, and broadside coupled (and perhaps some gray territory in between!) Both can be realized in microstrip, stripline or even coax, but stripline is the go-to technology for coupled-line couplers.

Theory of coupled line couplers

For two transmission lines coupled together to form a four-port network, there are two things that have to occur with coupled lines to become a usable coupler with directivity and quadrature phase:

1. The coupled section should be a quarter-wave at center frequency
2. The product of even and odd mode impedances must be equal to Z02

It's time to define some port numbers. Let port 1 be the input port. The port that is directly coupled to port 2, which is one of the two output ports. The other output port is directly across from the input port, we'll call it port 3. Under ideal conditions, a signal incident on port 1 will transfer zero power to port 4; this is called the isolated port.

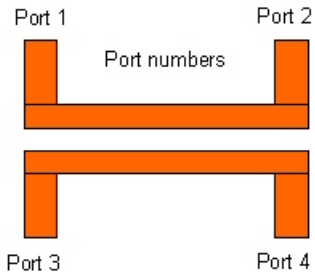


Figure 3: coupled lines graph

Advantages of coupled line couplers

Bandwidth is greater than in interconnected transmission line (uncoupled line) circuits like the branchline "coupler".

A natural consequence of coupled lines is a quadrature phase response.

Why Quadrature?

The quadrature property of the coupled lines is a subset of the amazing properties of lossless, symmetric four-port microwave circuits with or without coupling. For double symmetry circuits (like two coupled lines) the 16 S-parameters of the scattering matrix of the four-port reduce to 4 independent S-parameter values: $S_{11}, S_{12}, S_{13}, \& \ S_{14}$.

The resulting scattering matrix is shown below:

$$S_{Symmetric\ 4\ port} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{12} & S_{11} & S_{14} & S_{13} \\ S_{13} & S_{14} & S_{11} & S_{12} \\ S_{14} & S_{13} & S_{12} & S_{11} \end{bmatrix}$$

Because we are assuming that the circuit is lossless (no heat or radiation losses), conservation of power is applicable to the four ports. This means that the multiplication of the scattering matrix by its complex conjugate transposed matrix will be equal to the unity matrix. $[S]^{T^*}[S] = I$

2.3.2 RF ADC converters [Reference NUM]

The ADC is a key component in any radio that uses direct digitization of the RF input signal or that uses digitization after an initial downconversion to an IF. The other key component is the digital signal processor.

Bandpass Sampling for Direct Downconversion

Sampling at rates lower than $2f_{max}$ still can allow for an exact reconstruction of the information content of the analog signal if the signal is a bandpass signal. An ideal bandpass signal has no frequency components below a certain frequency f_l and above a certain frequency f_h. Typically, bandpass signals have $f_l \gg f_h - f_l$. So, if we modulate the signal to fl we would obtain a baseband signal of $BW=fh-fl$ and the Nyquist theorem tells us that we could be able to digitalize the signal only if our sample rate is equal or greater than $2*BW$.

Effects of Quantization Noise

The ADCs best suited to RF and IF processing that have widespread availability use uniform quantization. In uniform quantization, the voltage difference between each quantization level is the same. Other methods of quantization include logarithmic (A-law and μ-law), adaptive, and differential quantization.

In uniform quantization, the analog signal cannot be represented exactly with only a finite number of discrete amplitude levels. Therefore, some error is introduced into the quantized signal. The error signal is the difference between the analog signal and the quantized signal. Statistically, the error signal is assumed to be uniformly distributed within a quantization level.

Using this assumption, the mean squared quantization noise power P_{qn} is :

$P_{qn} = Q^2/12R$

Q is the quantization step size and R is the input resistance of the ADC.

2.3.3 SPINNER Coaxial commuter

SPINNER switches and relays have been designed and developed for excellent technical features. Some technical details are described below:

Bistable switching (latching)

Bistable switching (latching) is the switching from one stable condition into another whenever the control voltage is applied. Therefore a pulse is sufficient as a control signal, i. e. after switching presence of the control voltage is not required any longer. In the event of power failure the last switch position is retained. The same is true after re-starting the system.

Switches with solenoid drive are separated between impulse solenoid drive and lifting magnet drive. The operating voltage (preferably 24V DC) is used to control both drive systems. These drive systems can achieve extremely short switching times (e. g. < 40 ms for the N coaxial switch). Therefore solenoid drives are mainly used for relays and switches up to the 7-16 interface.

Impulse solenoid drive

Switches with solenoid drive generate the torque with a rotating magnet located in a stationary coil. The drive system is locked in both terminal positions (latching).

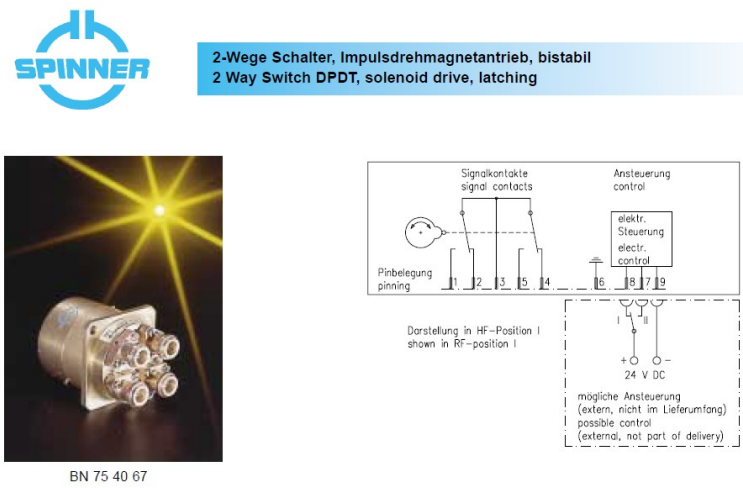


Figure 4: Used RF High power relay for the project's board

2.3.4 Microstrip designs

Much has been written about terminating PCB traces in their characteristic impedance, to avoid signal reflections. However, it may not be clear when transmission line techniques are appropriate.

A good guideline to determine when the transmission line approach is necessary for logic signals is as follows:

Terminate the transmission line in its characteristic impedance when the one-way propagation delay of the PCB track is equal to or greater than one-half the applied signal rise/fall time (whichever edge is faster).

MICROSTRIP PCB TRANSMISSION LINES

For a simple two-sided PCB design where one side is a ground plane, a signal trace on the other side can be designed for controlled impedance. This geometry is known as a surface microstrip, or more simply, microstrip.

A cross-sectional view of a two-layer PCB illustrates this microstrip geometry as shown in the following figure:

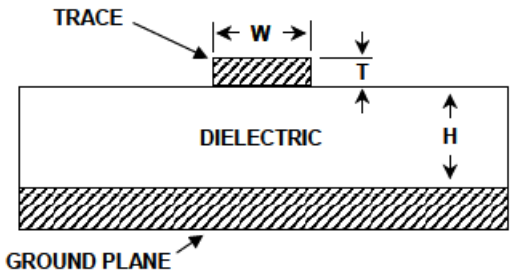


Figure 5: Microstrip Transmission Line with defined impedance

For a given PCB laminate and copper weight, note that all parameters will be predetermined except for W, the width of the signal trace can then be used to design a PCB trace to match the impedance required by the circuit. For the signal trace of width W and thickness T, separated by distance H from a ground (or power) plane by a PCB dielectric with dielectric constant εr, the characteristic impedance is:

$$Z_o(\Omega) = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln\left[\frac{5.98H}{(0.8W + T)}\right]$$

Note that the formula is expressed in mils. (1 mil ~ 0.0254 mm)

We also have to calculate the equivalent capacitance of the line, the definition of the formula is:

$$C_o(\text{pF/in}) = \frac{0.67(\varepsilon_r + 1.41)}{\ln[5.98H/(0.8W + T)]}$$

2.3.5 Use of transistors as logic circuits (MOSFET)

The MOSFET is a type of transistor used for amplifying or switching electronic signals which has an insulated gate whose voltage determines the conductivity of the device.

Although the MOSFET is a four-terminal device with source (S), gate (G), drain (D), and body (B) terminals, the body (or substrate) of the MOSFET is often connected to the source terminal, making it a three-terminal device like other field-effect transistors. Because these two terminals are normally connected to each other (short-circuited) internally, only three terminals appear in electrical diagrams.

Modes of operation

The operation of a MOSFET can be separated into three different modes, depending on the voltages at the terminals:

Cutoff

When VGS < Vth:

Where VGS is gate-to-source bias and Vth is the threshold voltage of the device. In this operation mode, there is no conduction between drain and source, so the transistor can be used as a turned-off switch.

Linear region

When VGS > Vth and VDS < VGS – Vth (VDS refers to saturation voltage)

The transistor is turned on, and a channel has been created which allows current to flow between the drain and the source. The MOSFET operates like a resistor, controlled by the gate voltage relative to both the source and drain voltages.

Saturation

When VGS > Vth and VDS ≥ (VGS – Vth):

The switch is turned on, and a channel has been created, which allows current to flow between the drain and source. In this case we could ideally model the response of the transistor as a breakdown voltage between gate and drain.

In this work we used this transistor functionalities to convert the outputted signal from digital I/O pins of the micro-controller to the desired signals on the relay ports in order to do the commutation operation.

2.4. Visual c# multi-platform control and use of MSND libraries

2.4.1 .NET framework

.NET Framework (pronounced dot net) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) known as Common Language Runtime (CLR), an application

virtual machine that provides services such as security, memory management, and exception handling.

FCL provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with .NET Framework and other libraries. .NET Framework is intended to be used by most new applications created for the Windows platform

2.5. Enterprise Power amplifiers basics

2.5.1 Device electronic description

The Power Amplifier Stage is directly plugged to the mains network. Its own power supply will deliver the necessary DC voltages (+12 and +VAMP) to feed the different modules and to generate, from these, additional DC voltages (+5V, +3V3) to feed some other boards. The signal path goes from the RF input connector (RF IN), throughout the gain control module which will perform the AGC (automatic gain control) using output signal samples delivered from the output coupler. At the output of the gain control board the signal will enter through the different amplification stages which will allow the signal to meet the expected power level at the output connector. The driver module provides a high gain which will deliver the proper level for the next stages, driver module and Final Amplifier module, which raise the signal power up to its nominal level. Finally, the signal will go through the output coupler, which will provide a reference signal to the gain control board and will allow the unit to have output signal samples at its front and back panel.

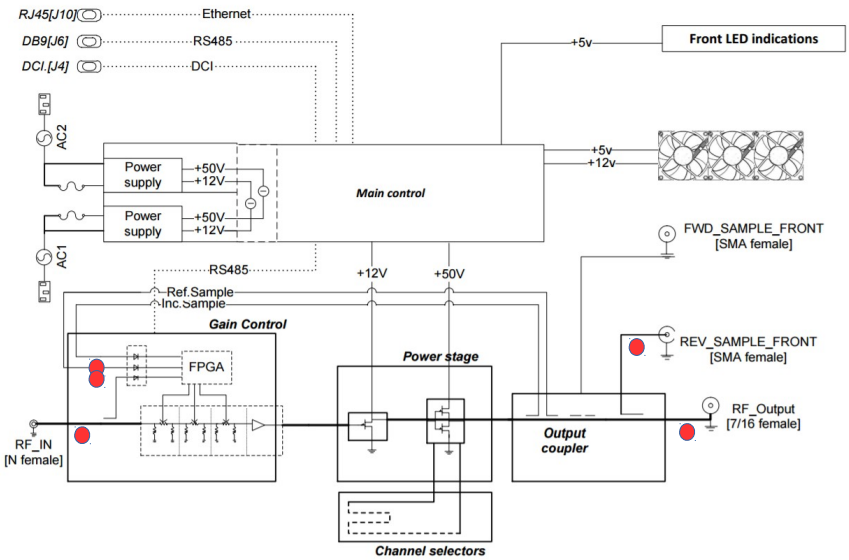


Figure 6: Power amplifier internal structure

As you can see in the previous figure, we appreciate that there are 5 red circles, the board designed in this project has the purpose to control automatically the routing of those signals for configuration purposes.

2.4.2 PA's communication protocol

Communication protocols are implemented on the external Gain Control via RS485 bus can read and write any register addresses between 0 and 127 and can read any record in the address between 128 and 255. However, it should bear in mind that using this protocol logs are exported as 16-bit registers (2 bytes) and 32 bits (4 bytes) are as physically. Thus, from the outside is given access only to 16-bit lighter weight of each record, reserving the 16-bit heavier Internal Firmware and Software and allowing them to interact with signals and parameters with which the external user has not done anything.

An example of the former would signal "enable" RS485 driver. This signal is controlled from the software to transmit information by bus but out of an FPGA pin mapping in the firmware. Thus, this signal is mapped bit 0 16 registry to allow software controlled using firmware. But this bit is not accessible from the RS485 bus is in 16-bit heavier registry

3. Methodology / project development:

3.1 Initial phase

As we have mentioned before, at Tryo where some programmed libraries and applications with Visual Studio for diverse purposes, like an application to adjust the RF parameters of the Power Amplifiers such as emitted power, frequency bands, internal DC supply for transistors…Taking in account that we could be able to interpret all the parameter sets of a Power Amplifier through IP using a library compatible with Visual studio, combined with the fact that we could localize all the needed connections to configure the PA's automatically within easily accessible connectors, we found a lack of optimization in the devices production chain.

After few planning weeks the idea was closed, the final system had to be composed at least by one own designed PCB with a switching function (See X.X for detailed explanation), one Server computer executing the Visual studio program, a potentiometer, a spectrum analyser and the PA to be configured, with all the required connections (See Figure 1 for detailed explanation).

For that reason, the first steps consisted in the implementation of the missing libraries for the future visual studio application. The work was basically done in this order because it would let the author to be familiarized with the libraries and the COM interface of the enterprise that were already programmed specifically for PA's with Visual Studio, this task also introduced the author to remote control instrumentation and SCPI protocol. Five classes were created for the instrumentation purposes:

- **Class Instruments**: It was a generic class or master class for all type of instruments, it contained all the attributes initialization, the other classes inherit their attributes form this class. It also contain some declared methods that are generic for all the instruments, like Identify (that is used to return the device type, model, Serial and firmware), as well as the constructors of methods that had been implemented in other classes. In the UML diagram we could be able to see three aggregated classes 'InstrumentExecuteResult' (used for determining the status of the commands sent to and received from the instruments, every time a command is sent we receive this type of response), 'InstrumentIdentity' (constructor of instrument identification class) and 'ExecutionChangedEventArgs' (define whenever a new command is sent and so the 'status change').

- **Class InstrumentIP**: This intermediate class initializes the connection parameters and functions for a specific instrument that is identified as an 'InstrumentIP', this identification is done by 'CreateInstance()' method and in this case it only admits the 'N9010A' parsed from the 'Model' attribute from 'InstrumentIdentity' class. Those previously mentioned parameters are IPAddr, PortNumber and Socket (the initialization of a new socket connection can be done with 'Connect()' method).

- **Class InstrumentUSB**: This intermediate class initializes the connection parameters and functions for a specific instrument that is identified as an 'InstrumentIP', this identification is done by 'CreateInstance()' method and in this case it only admits the 'U2001A' parsed from the 'Model' attribute from 'InstrumentIdentity' class. Those previously mentioned parameters are a 'ResourceManager', 'FormattedIO488' (which are class types of 'ivi.Visa' libs see 2.3.2 for more information), and 'UsbString' (it is the string used to create a FormattedIO488 connection through a ResourceManager). This class also includes the 'ExecuteCommand()' method that determines if there was an interruption on the USB communication, it is also used to change the 'ExecutionChanged' parameter and the definition of WriteLine and ReadLine for the FormattedIO488 connection.

- **Class U2001A**: This class inherits all the previous attributes from class 'InstrumentUSB' and its function is to give the necessary methods to extract configuration data from Power-meters of the model U2001A (Agilent). All the data extraction is used following the device manual instructions for SCPI protocol.

- **Class N9010A**: This class inherits all the previous attributes from class 'InstrumentIP' and its function is to give the necessary methods to extract configuration data from Spectrum Analyzers of the model U2001A (Agilent). Those methods are enumerated in the UML diagram.

Since we had all this classes implemented, we've been able to connect and read the Power Meters measurements, as well as set Spectrum Analysers span, center frequency and also get the screen captures or the MER measure analysis. This classes have to be imported in the main project and just executing a line we could get all the desired parameters, that would be the most comfortable way to program the main server application and its interaction with instruments.

The UML diagram of this classes can be seen in the following figure, the first square block represents the whole 'Instrument' *Namespace* with the main 'Instrument' class its aggregated classes and enumerations, in a lower level we can find 'InstrumentUSB' and 'instrumentIP' classes wich inherit from the main class, 'U2001A' and 'N9010A' inherit respectively the attributes from this second level classes:

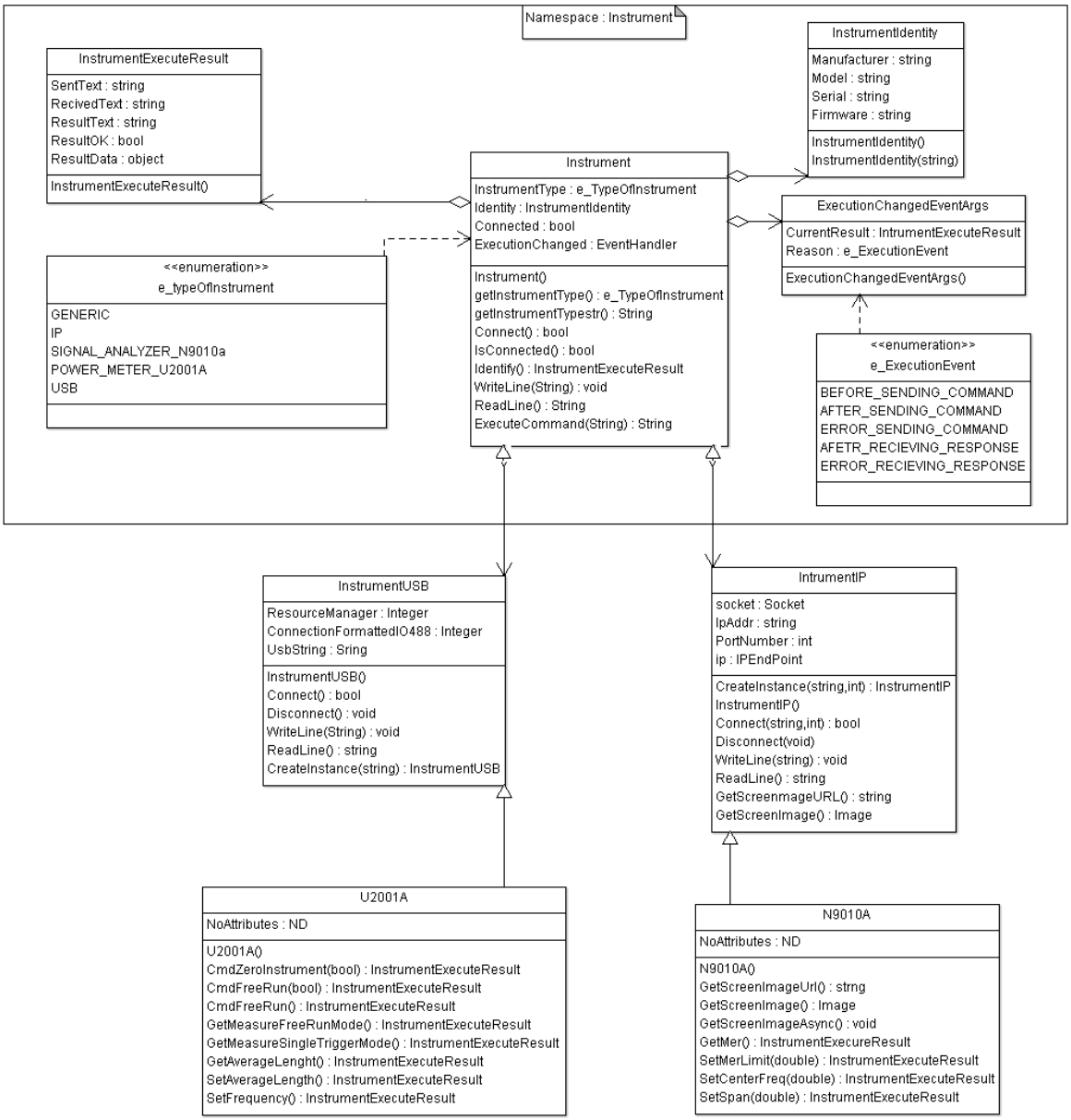


Figure 7: Visual Studio instrumentation UML class diagram

3.2 Schematic design

As we could see in chapter 2.5.1 of this document, in the board design 4 RF inputs and 2 outputs with adapted paths at a 50 ohms characteristic impedance for the following ranges would be needed:

- Signal power range: 5dBm – 30 dBm.
- Frequency range: 30 MHz – 900 MHz

Each output has to be able to receive 3 different input signals, taking into account this requirements we should dispose 4 relays in a specified order. All the relays have each respective indicator. For more understanding of the relays disposition see the figure below:

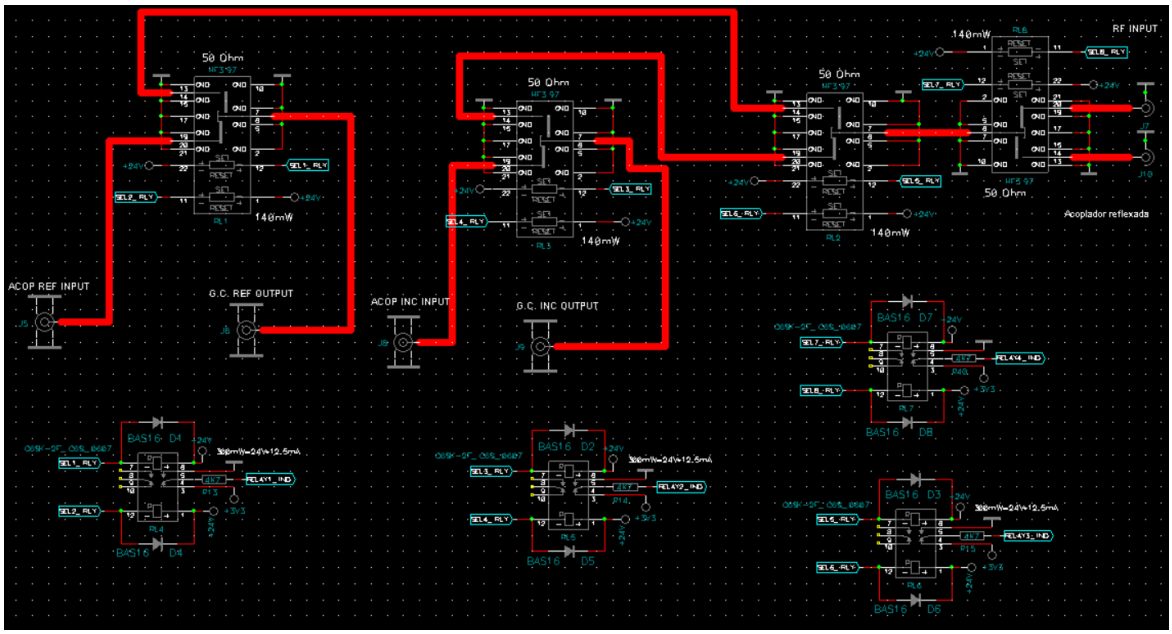


Figure 8: Relays disposition and RF connections capture from schematic

The relays are appreciated on the top part of the figure with its respective RF microstrip designed paths. Note that each relay is connected to two nets simultaneously, those nets come from the micro-controller, and are used to commute the relays bidirectionally. The digital I/O pins of the controller attached to this nets have to be at logical 1 by default and whenever we want to commute the relay, the micro have to put a 0 during an instant of time on the correct pin. This 'SELnº_RLY' nets are also connected to the pins of the indicators, if we have a 0 in one of those signals the internal switch of the indicators is commuted and it passes from GND pin to 3V3 pin or upside down.

In order to obtain the desired tension on the 'SELnº_RLY' nets mentioned above, the creation of a logical electronic network was done, this logical network comes from the pin of the micro and finishes onto the selection pin of the relay. On the following figure you should be able to appreciate the realization of this logical net for signal adaptation:

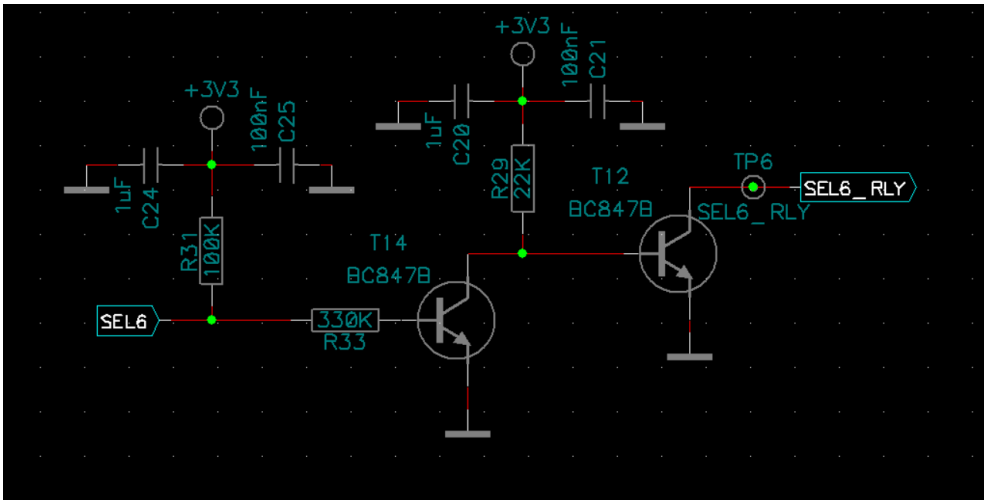


Figure 9: logical circuit for relay's commutation pins.

On the first BJT if a digital 1 is inserted to 'SEL6'(direct signal from the micro), that will enter in saturation region and that would cause a 0 on the second BJT gate, so this second transistor will be working in cutoff mode. Inversely if 'SEL6' is set to 0 we would obtain another 0 on the 'SEL6_RLY' and if we focus on the relays figure we can observe that the solenoid of the relay will generate a positive current between port 1 and 11, that would be equivalent to put the switch on RESET status.

After defining the commutation and routing part of the circuit we could understand much better the pin connections on the micro-controller stage. The following picture is a schematic capture of the micro part:

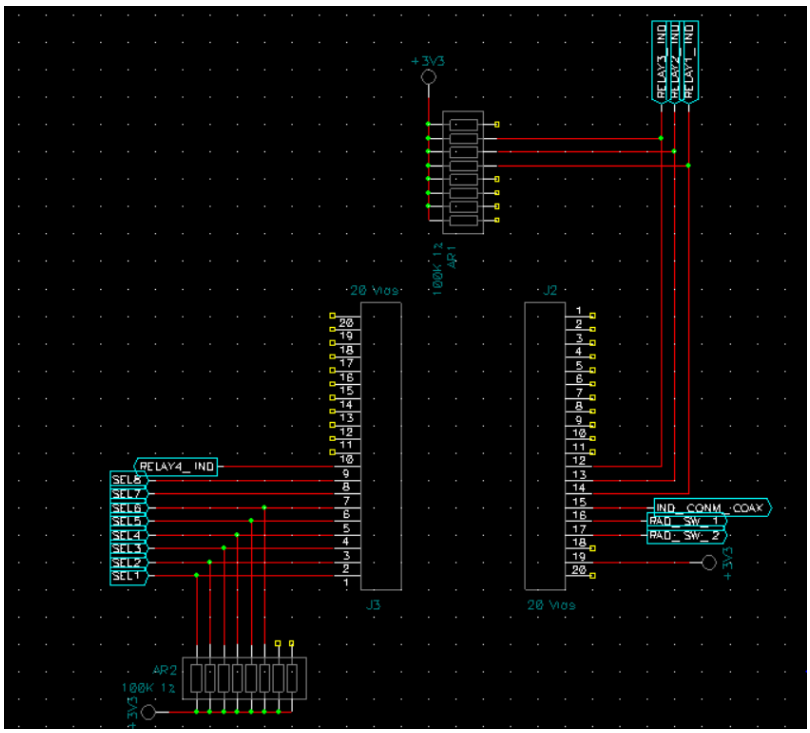


Figure 10: Microcontroller I/O connection pins.

The Arduino micro in this case will have 8 'SEL' pins connected to its digital I/O pins in order to control both possible positions of all the 4 relays, all those pins should be raised to 3.3V by the firmware when the device is current supplied. From those relays the micro would also have 4 relay indicator pins. Apart of it, there should be 3 digital I/O pins too, two of them are used to commute the coaxial switch and the other one is a position indicator pin. As we could se in section 2.3.3 the Spinner coaxial switch has to be DC supplied to 24V at pin 7 or pin 8 of its DB-9 connector depending on the position that we want to commute. That's why a design of a voltage adaptation circuit was done:

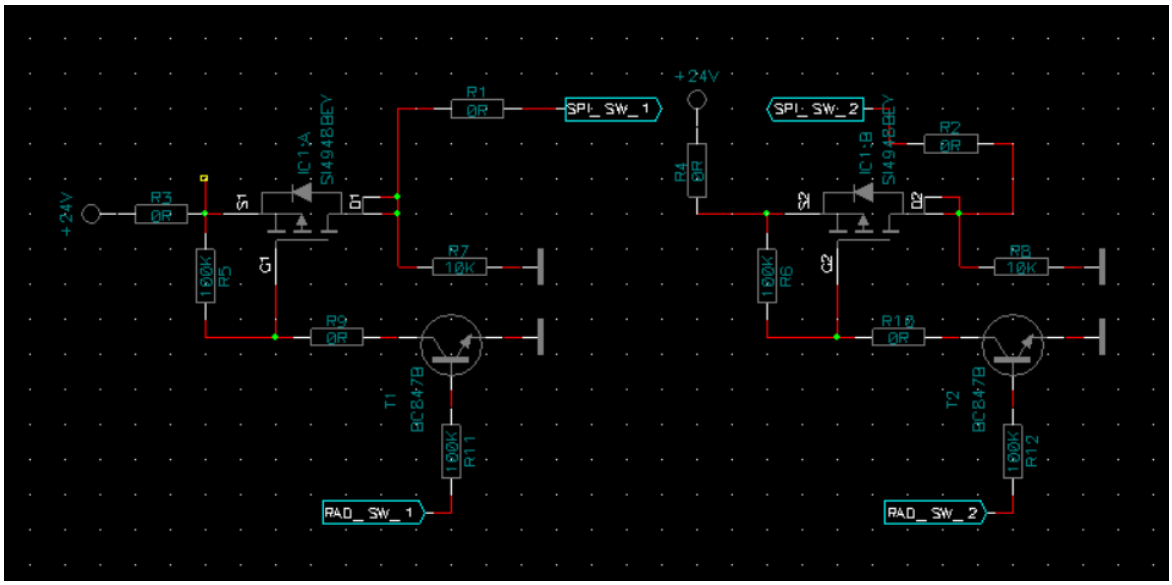


Figure 11: Voltage adapter circuits for coaxial commuter pins.

Once we send a 0 from 'RAD_SW' we make the tristable transistor enter on cut-off mode and that will generate a 24V pulse to the selected 'SPI_SW'. Additionally, we should also have an external supply port of 24V as well as a DB-9 connector to pass the signals between coaxial switch and the board. So now, the schematic design will have all the required connections for the accomplishment of the initial requirements.

3.3 Server-Board communication protocol & firmware

After designing the schematic a firmware for the micro-controller had to be implemented. Focusing on the specifications of Arduino Micro, the programming of this part was based on the creation of a serial communication protocol between the server and the board through the USB port. This communication protocol has a Master-slave structure and it has a periodic check of the serial port from the micro as it is the slave. The weft has a fixed size of 16 bytes and a CRC used to check the communication errors. All the hexadecimal characters of the packet bytes are encoded into 2 independent bytes, that is done for 'ETX' and 'STX' recognition, as they are equal to 0x02 and 0x03 and used for beginning and closing of the packet respectively. So the message structure of the protocol is the following:

Own communication protocol

	STX	read_or_write	Addr	Message	CRC	ETX
Value ranges	0x02	0 or 1	0 to 255	0 | 0 | 6 | 4	XOR of body bytes	0x03
Uncoded sequence length	1 byte	1 binary byte	1 byte	4 bytes word	1byte	1 byte

Figure 12: Byte structure of serial protocol's uncoded message.

The protocol was programmed inside 'Arduino IDE' software with C programming language, it has different implemented 7 methods:

- *unsigned int slave_char_encode(char c):* This is the character encoder, the input is a char of 8 bits and applying a cast and a mask there is a 16 bits unsigned int returned.

- *char slave_char_decode(char c0, char c1):* This is the character decoder, inverse function of pthe previously explained.

- *int slave_char_send(char c):* Function which executes the digital write of characters and have a communication error control with a maximum number of 50 errors per character.

- *int slave_char_send_encoded(char c):* Makes an encoding of the input char and sends both encoded character with the methods explained before.

- *slave_send(unsigned char read_or_write, unsigned long message, unsigned char addr):* In this case the whole packet sent with the corresponding message, the CRC is calculated, most of the sent characters are encoded but STX and ETX are never encoded, that's the reason of existence of the two methods explained before.

- *slave_receive():* On the receiver, unless the micro have received an STX character, a read or write byte, a known address, an interpretable message, a correct CRC and finally the ETX the message is discarted.

- *char slave_process_command():* This function is the 'action choice switch', the controller takes a decision depending on the read_or_write mode, the Addr and the body message of the packet.

Apart from those methods all the Arduino firmwares should include two other system methods by default:

- *void setup():* That is the designated function to declare all the initialization for communication buses and connected pins.

- *void loop():* Main function of the firmware, as its name indicates it is executed periodically by the micro.

The board is totally commanded by the Server (Visual Studio application ran on a windows computer), it sends all the read-write packets the board function slave_process_command() function of the firmware realizes the lectures of indicators or the commutation of relays depending on the instruction sent, and returns a message packet with its read-write byte (contrary of the one sent by the server) and the response message for the command.

3.4 PCB design

First of all, a small briefing of PCB providers and prices was done, during this investigation we noticed that there were some materials and thickness surfaces combinations much more economical than other. The chosen surface structure was:

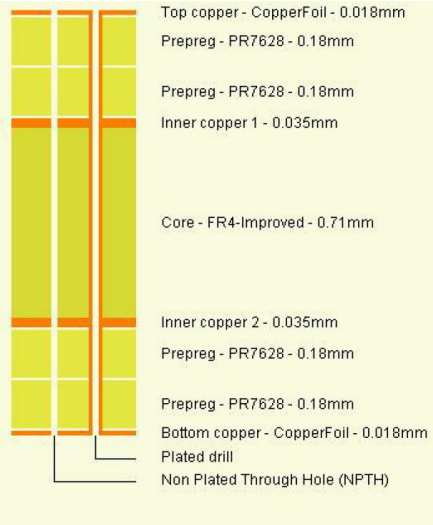


Figure 13: PCB surface structure

All the microstrip design should be based on the calculation of cooper traces width using the dimensions of the board for impedance adaptation, for more detailed information see 2.3.4. The surfaces were distributed by the following structure from top to bottom:

- 1st cooper surface: contains the RF paths that carry the signals commuted by the relays as well as most of the digital signals different from 3V3 supply.

- 2nd cooper surface: It is the GND plane, apart from giving the neutral charge to the circuit it has the function of RF isolation from other signal interferences.

- 3rd cooper surface: Used as a supply plane, the components are distributed in a previously studied way that lets us separate this surface in a 3.3V supply plane and a 24 supply plane too.

- 4th cooper surface: Contains the digital signals that couldn't be carried through the first cooper surface because of line-crossing reasons.

If we apply this configuration the H parameter of the equivalent impedance is equal to 2x0.18 mm = 0.36 mm as there are two Prepeg sheets between RF cooper lines and GND plane.

In this section is explained how was the layout of the PCB done using Pcad-2006. Once the schematic of the board was closed we could export the .dxf as well as the netlist files, those are necessary to start the design of the layout. On the design there were some critical factors and rules to take into account:

- **RF line shapes:** Should avoid angles greater than 45 degrees to minimize reflections.

- **Position of the connectors:** They should be disposed in a concrete way in order to avoid crossed cables plugging.

- **Minimum length for RF paths and maximum isolation from other signals:** with the aim of avoiding attenuation and uncoupling of the carried signals.

- **Relays distribution:** This is totally dependant of the two conditions explained before.

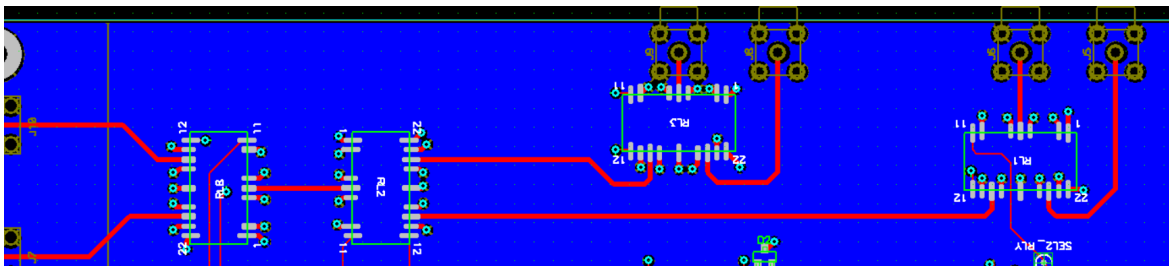


Figure 14: Capture of the relays and RF lines distribution.

- **Closeness of filter condensers:** All the condensers used as filters has to be at minimum distance from the filtered resistance.

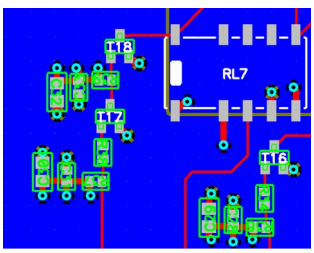


Figure 15: filtering condersors positioning of the net explained at figure N.

- **Components distribution for supply planes:** All the supplied components depending on they nominal voltage should be situated over the 24V or the 3.3V plane.

- **Proximity between relays and respective indicators:** This fact is important because the same signals used to commute the relays are used for indicator pins too.

When the design was completed, the Gerber files have been exported and sent to 'Eurocircuits' (PCB manufacturer). So in this way, the whole design of the board became totally closed, therefore we must have to wait for the ordered board.

The picture beneath shows the complete top layer with the board serigraphy, in the top layer are contained all the SMD pads for the components, the cooper of the 'vias', the cooper basis of connectors, the enterprise logo's...

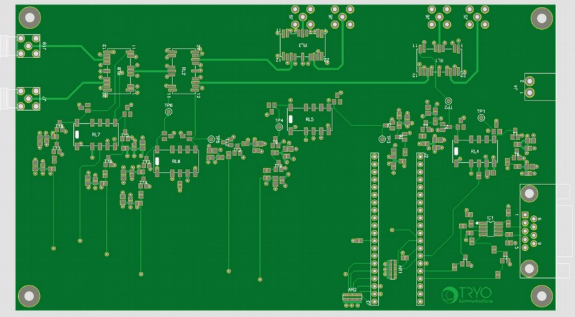


Figure 16: Gerber files for fabrication board (TOP layer & serigraphy)

The following captures are the final design of top and bottom layers, the horizontal lines are a defect caused by the 'Pcad' grid configuration:

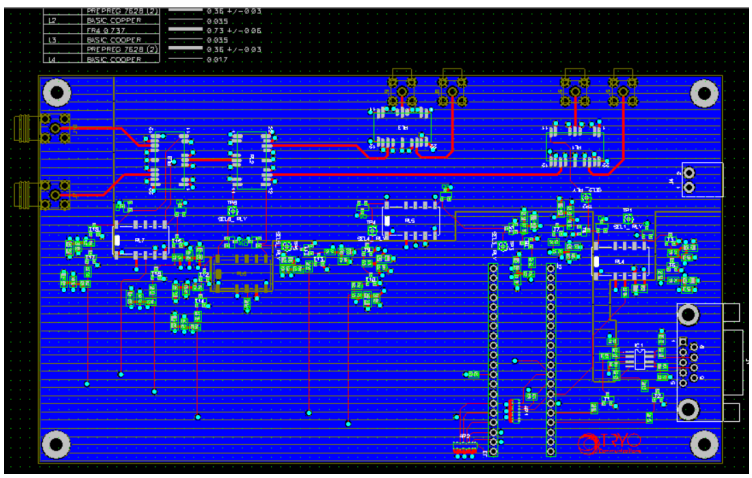


Figure 17: PCB top plane of the board with the positioned components and RF paths.

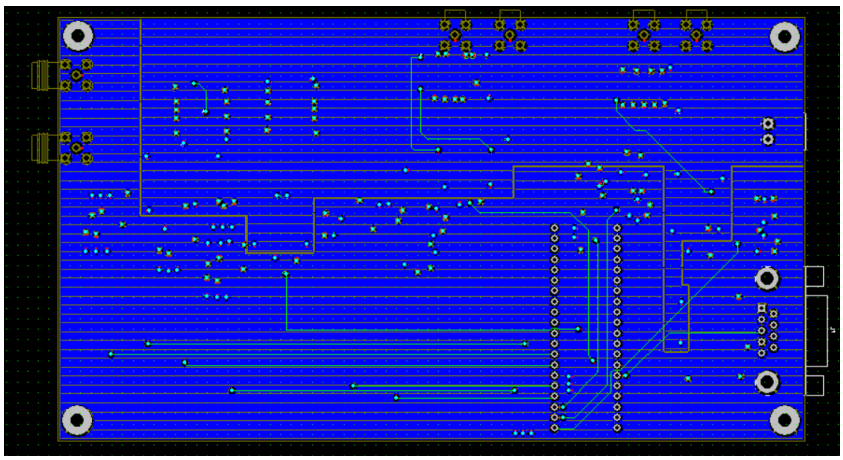


Figure 18: bottom layer PCB design capture.

4. Description of deviations and incidences.

This project couldn't have a normal development deadline, this was a work that has a duration of 4 moths taking into account that the author will be working for 'Tryo Communications' enterprise at that time. But unfortunately, the enterprise passed through some economical and financial problems and the author wasn't able to be contracted for the whole project period.

There were some legal problems about the medical insurance if the author wanted to finish this project in the enterprise without legal contraction. So during an entire month the project was frozen. Lately, an enterprise called 'Tryo Aerospace' from the same global group of enterprises contracted the author for other work purposes. But that, at the same time, legalized the confection of the project in the first enterprise. So, the author took advantage of this situation and tried to finish the whole work in the spare-time.

Finally, after loosing a month for contraction problems and elaborate this project apart from the work hours, the final deadline couldn't be accomplished and the total delay was around 10 days. The gantt diagram of this project reserved all the testing and simulations with the board during the last week but the board was ordered just one week before the document delivery and the final results of the autoconfiguration system were impossible to be added.

So, the electronic design and the system programmability parts were totally finished, but the mounting of the board was still missing and in this moment it is impossible to provide an exact validation of the system performance.

5. Results

In this section, the execution of the program with the whole system fully operative had to be presented, but considering the problems emerged on the project deadline execution (explained in section 4 of this document), we couldn't be able to get the ordered boards in time and they still couldn't be mounted when this thesis document was wrote.

In order to test the Server system communications a partial execution of the application was done, its realization was without the board, connecting the incident sample from the PA's internal coupler (see 2.5.1) to a power meter and also to an impedance load of 50 ohms, the spectrum analyser on a coupler that is plugged in the output of the PA and an enterprise transmitter in the PA's input.

The GUI (executed by the server) made with Visual Studio and C# language, only have two pages to ease the configuration project for factory technicians, but it has a huge background programming to reach the desired functionalities. As the final execution results couldn't be presented in this section the main background processes are explained.

The initial state of the program when is ran is the following:

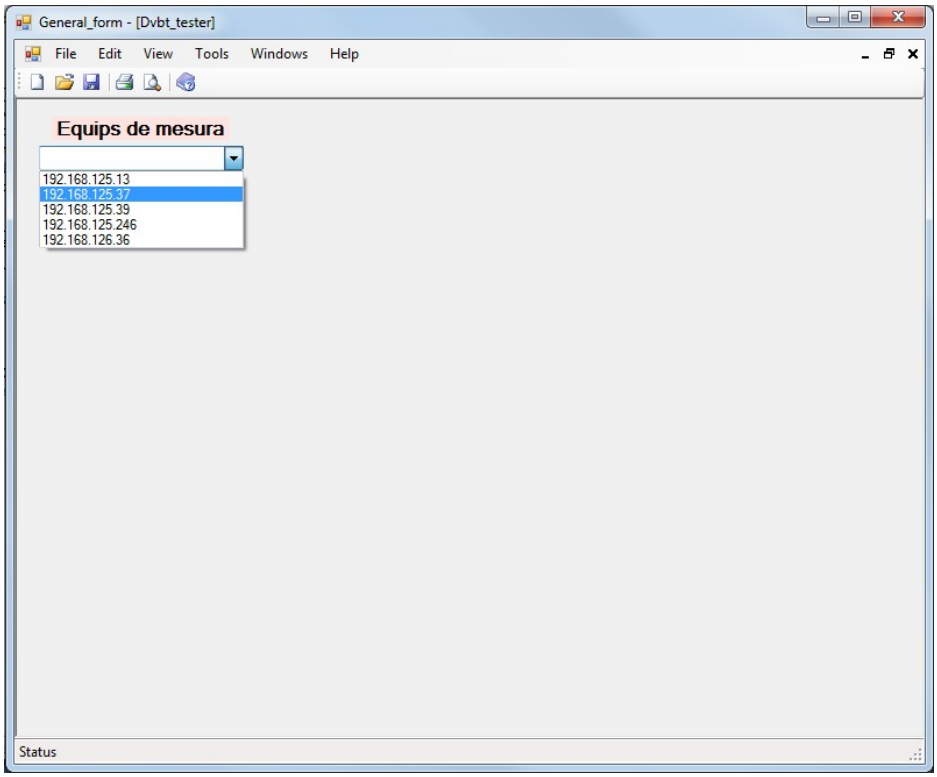


Figure 19: Initial state of the first GUI page.

At the beginning we only have a window with a drop-down selector, as well as an upper menu from Windows interface. This Windows menu is not implemented because it won't have sense in the application. The unique implemented tab from the menu is 'View' where technicians can select 'full screen mode' or 'reduced window mode'. This drop-down

selector only shows all the ip's of the Agilent devices that are connected to the same LAN. That can be accomplished declaring a new 'Process' of the command prompt, start it and use 'StandardInput' subclass to send an 'arp -a', with the response of the command we should parse the ip's using the MAC number, cause de MAC first numbers are unique depending on the fabricator owner. Then, when this ip's are caught there is an Instrument object declared (refers to 4.1 section) and from this object we execute an identification of all the devices models, after it we should be able to insert in the drop-down the ip's that are truly spectrum analysers.

The second capture of the program has a selected ip from the drop-down and shows the spectrum analyser screen extracted by GetScreenImage() method of 'N9010A' class, its span and centre frequency is also configured with the class methods.

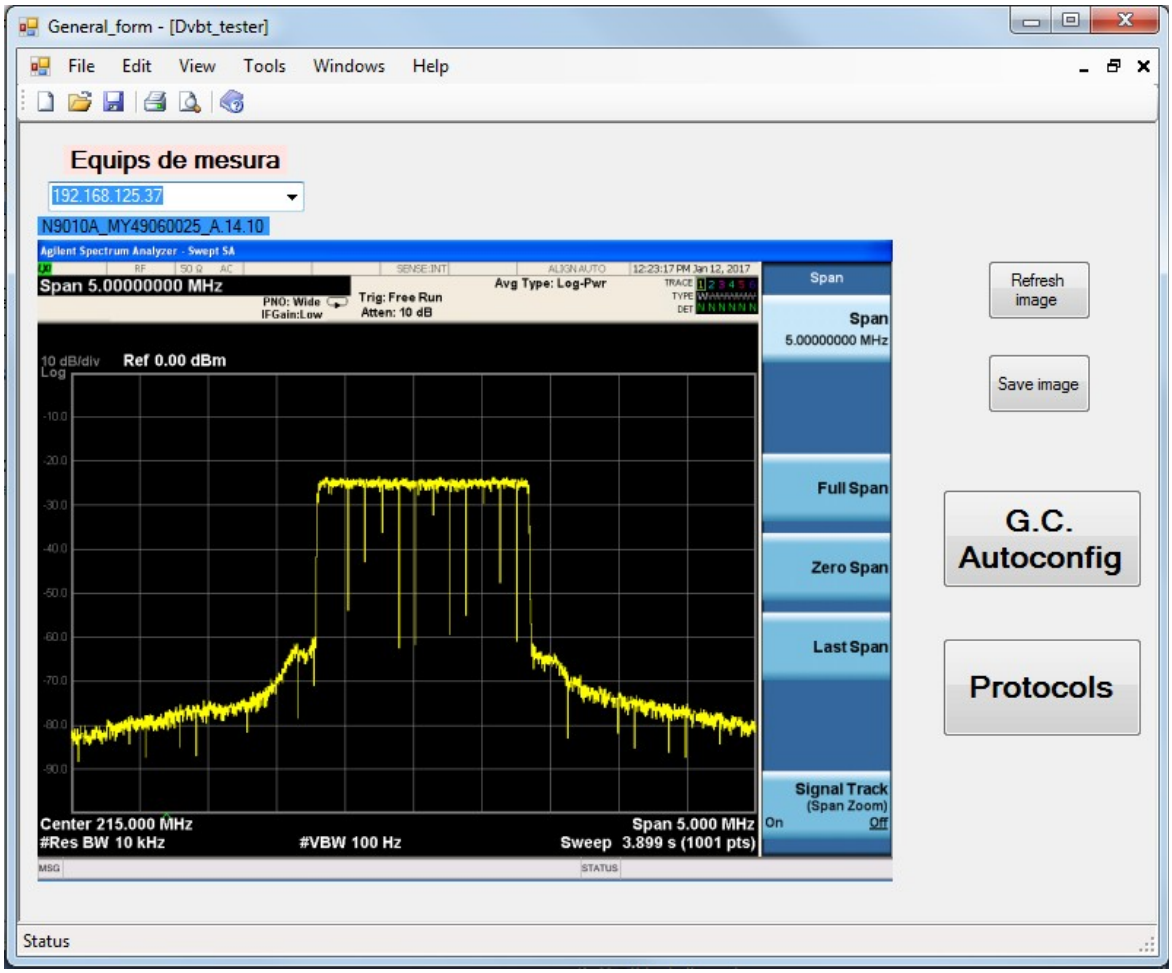


Figure 20: First page with selected spectrum.

After selecting the spectrum analyser IP the screen capture is shown and also 4 pushbuttons are shown. The 'Refresh image' button is used to refresh the screen capture and the 'Save image' shows a windows SaveFileDialog. There are also two extra buttons, 'G.C. Autoconfig' when it is pressed the automatic configuration process of the plugged power aplifier starts. 'Protocols' is explained later.

The next figure is the window that appears once we click over 'G.C. Autoconfig' button in the initial page of the GUI, it is mainly composed by a tab window that contain all the steps to configure a PA automatically, and a progress bar that grows during the configuration process:

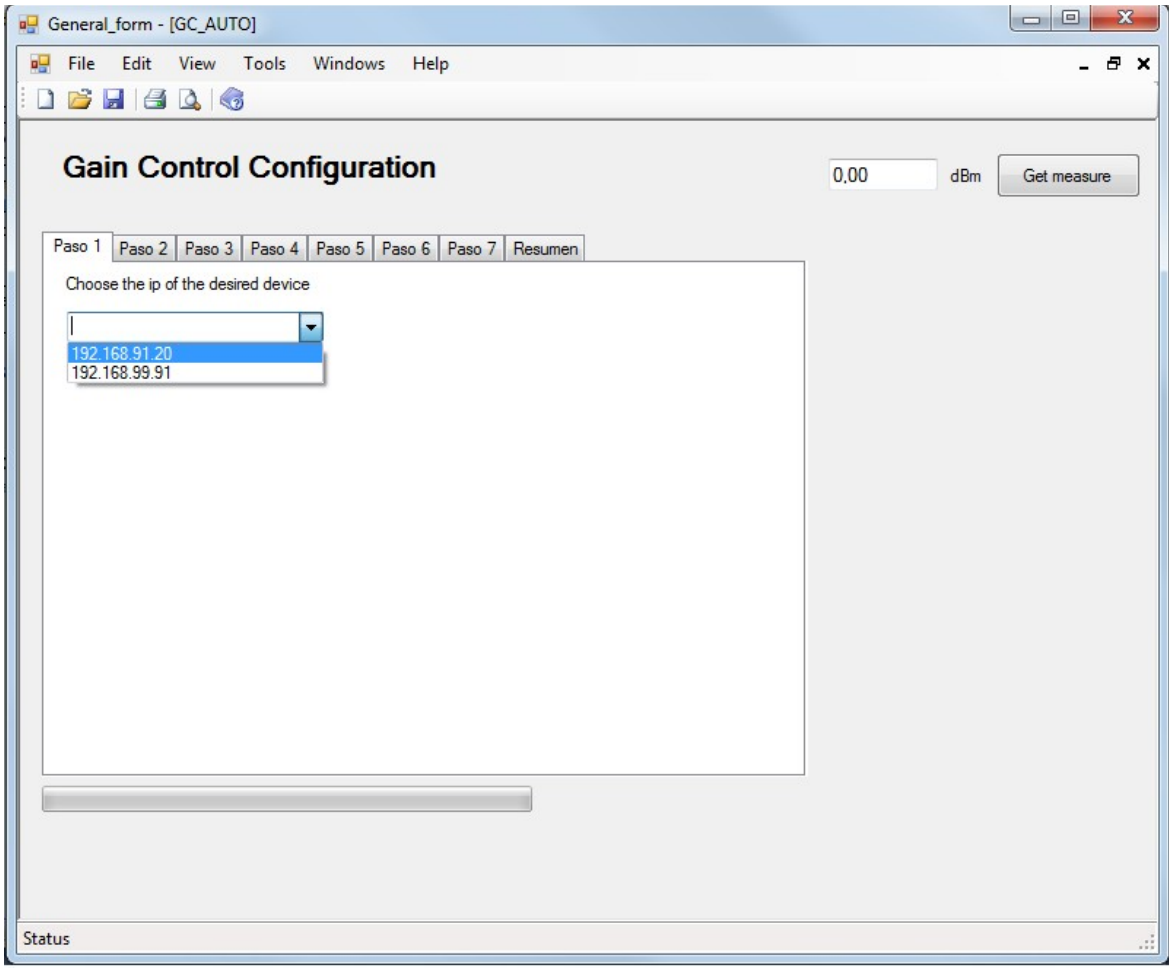


Figure 21: IP drop down selector for PA's connected to the LAN

It also have an ip drop-down selector used to connect to the configured power amplifier, this selector works with the same method explained before but it only look for MAC's corresponding to the enterprise power amplifiers. In the right corner of the window we can see the apparition of a power measurement, it comes from the power meter that is connected via USB to the server computer. There is a USB search process executing in second therm and if the connection isn't still plugged the drop down selector is not visible as well as the text box of the measure. When the power meter is detected in any of the USB ports by this background process, the program automatically enables the ip selector and the textbox.

After selecting the ip the server realizes a connection establishment with the PA and sends an identification command to detect the device model, depending on the PA model

different logos are shown. The system executes a general connection test and if everything is right, the 'START' button appears below the Power measurement.

The following image shows the first step of the configuration process, when technicians press 'START' button the Tab menu commute to Step2, and the first step is to calibrate the INC sample from PA, if we could test the process on the board it would be 1 of the RF inputs:

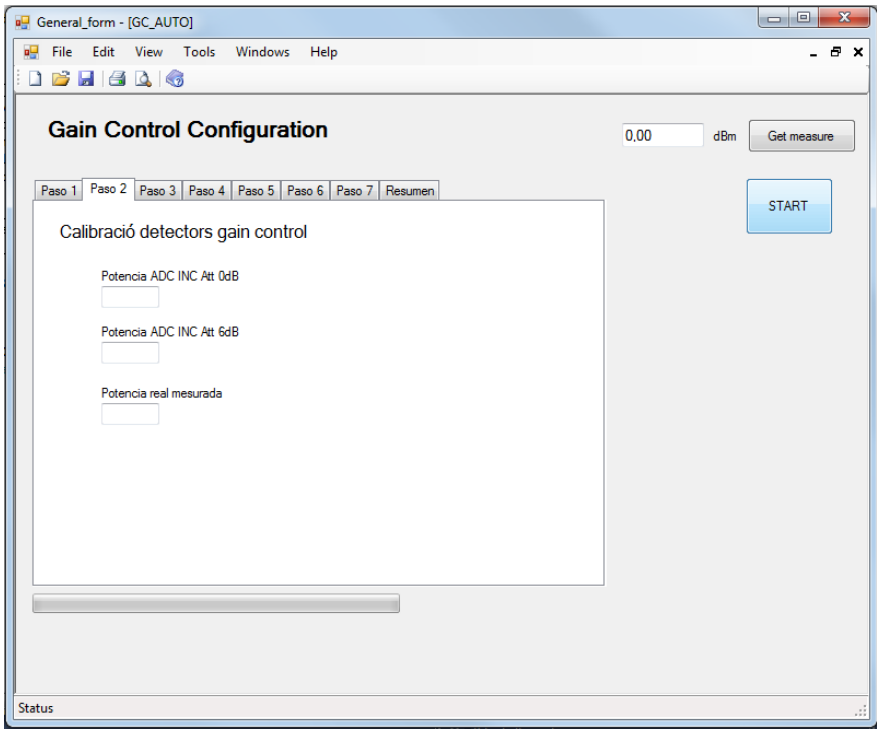
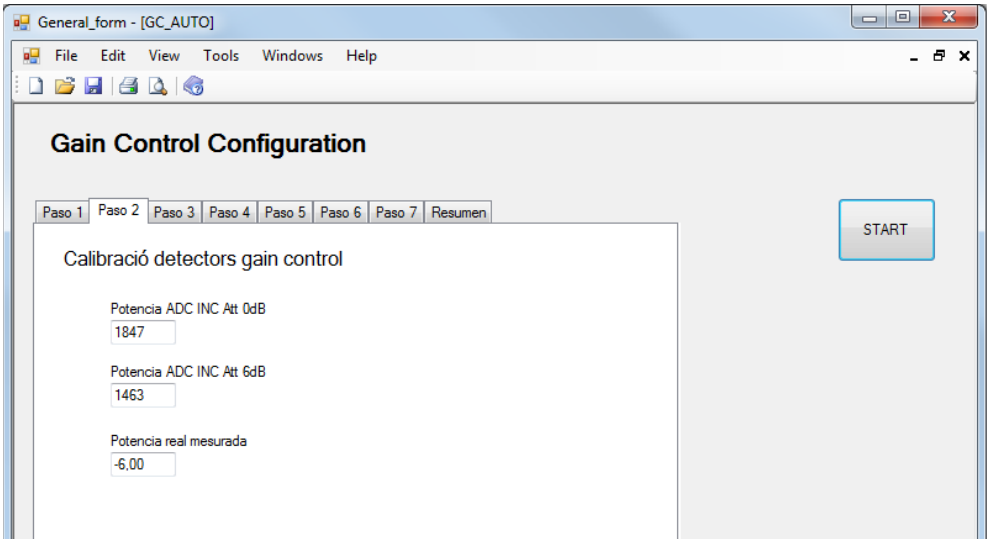


Figure 22: 1st calibration stage for the incident sample of the PA.

The server order to the transmitter a signal emission of 0 dBm RMS and also does an initial measurement of ADC steps for the zero calibration. Immediately it switches the nominal emitter power to -6dBm and takes the second measure again, the power measurement is done in ADC steps through the PA and in dBm by the power meter.



In this point we have calibrated the ADC input of the first incident coupler sample, but it was impossible to change all the code of the program in order to make the whole configuration manually, as it was designed for sending commands to the micro controller in the same main script, that's why I generated a system simulation program that is only able to execute the first step. This step corresponds to the default RF path of the board that is generated by the relay's initial position, it can be observed on the first schematic capture (see 3.2).

After this software system testing, we obtained satisfactory results in communication and reliability of the server control. It is clear that without the board this project of automation have no sense, but it is intended to present an entire demo of the system in the final presentation. The test execution was done without communication problems, and that guarantees a correct implementation of at least the calibration process, and also for functions and classes implemented between the Server and the instruments, as well as the communication of PA's and Transmitters with the Server. Until we won't get the PCB, the board communication tests are unrealisable.

The entire configuration process consist in 5 ADC calibration stages, 1 power limiters configuration and finally the frequency parameters extraction trough the spectrum analyser.

6. Budget

The budget of this project was reduced to the board costs that are about 150 euros. This costs are mainly the PCB manufacturing and the Arduino Micro. All the rest of the needed components were extracted from the enterprise component repository. Of course, it also had some symbolical costs of time for the engineers that gave advice in some project aspects. For the rest of aspects, the project developer worked without being contracted so it supposed a cost of time but an inversion on knowledge too.

7. Conclusions and future development:

If the objectives of this project are totally accomplished in the next coming days, this first automation system can open a door in the sense of devices confection for the enterprise. The technician using this program just have the role of observer and some other tasks can be done till the system configured the PA.

Since the last 80's there are lots of enterprises that used specific designs to accelerate and automate factory processes. Envolving RF devices in this kind of processes is quite complicated, but in future developments of the system, huge amounts of data from the measure instruments and from PA's could be compiled on the server and used to improve its configured parameters.

Although some complications in the project structure and elaboration, it was totally designed and tested with satisfactory results. In bigger projects that involve hardware, software and all that different knowledges is better to have a specialized team for each task, than trying to focus in every technical aspect of the project, because it makes a long time to acquire the necessary knowledge on each aspect.

Bibliography:

[1] https://www.microwaves101.com/microwave-encyclopedia/coupled-line-couplers

[2] N9020A/N9010A Spectrum Analyzer Mode User's and Programmer's Reference

[3] Agilent U2000 Series USB Power Sensors Programming Guide

[4] NTIA Report 96-328, RF and IF Digitization in Radio Receivers: Theory, Concepts, and Examples
J.A. Wepman, J.R. Hoffman

[5] https://en.wikipedia.org/wiki/Electromechanics

[6] Introduction to Microcontrollers, Courses 182.064 & 182.074
Vienna University of Technology, Institute of Computer Engineering, February 26, 2007

[7] https://www.pasternack.com/t-basics-of-rf-switches.aspx

[8] http://www.profesormolina.com.ar/tutoriales/trans_bipolar.htm
??
[9] https://en.wikipedia.org/wiki/MOSFET

[10] https://en.wikipedia.org/wiki/.NET_Framework

[11] https://msdn.microsoft.com/es-es/library/w0x726c2(v=vs.110).aspx

[12] Systems Alliance, VPP-4.3.4: VISA Implementation Specification for COM, June 9, 2010 (Section 7)

[13] Coaxial Switches – SPINNER

[14] Multiple references from device manuals of 'Tryo communications'

Glossary

- PA: Enterprise power amplifier
- RMS: Root mid square
- IP: Internet protocol
- SCPI: Standard Commans for Programmable Instruments
- RF: Radio frequency
- ADC: Analog/digital converter
- USB: Universal serial bus
- INC: incident signal of a coupler sample
- GND: ground plane
- GUI: Graphical user interface
- PCB: Printed Circuit Board
- Prepeg: Isolating material for PCB situated between cooper planes