Documentation - ADRF5720 Control Board

Table of contents

Documentation - ADRF5720 Control Board 1
1. General presentation 1
2. Description of the shield board 1
3. ADRF5720 Operation 4
4. Embedded code 4
4.1 Uploading the Embedded Code 4
5. Python Control Software 5
5.1 Running the Python Script 5
6. JSON Configuration File 5
7. Utilisation 6
8. Warning 6

1. General presentation

This document presents the shield board developed to control the numeric RF attenuator ADRF5720-EvalZ through an Arduino Uno or a similar STM32 microchip. The board allows the command of the ADRF5720 attenuation with a serial firmware thanks to a Python code and an embedded one on the Arduino/STM32. The aim of this control board is to adjust the power gain of a Gaussian noise signal at the entry of the ADC within the scope of ALMA project.

2. Description of the shield board

The shield board allows the power supply and the control of the ADRF572-EvalZ.

There is its principal's material characteristics:

- Connector CON2: allows the choice of power supply with a jumper (Arduino: 1-2 / External: 2-3)
- Terminal block: external source of power supply
- Connector CON6: link the control board and the ADRF5720 through a 18 pin connector to manage the attenuation

- Connector CON3: provide the voltage necessary to the ADRF5720 (VSS = -3.3V, GND, VDD = +3.3V)
- Test points: allow the correct sequence of ignition and shutdown (VDD before VSS to power on, VSS before VDD to power off)
- LEDs: 6 LEDs shows the attenuation send to the ADRF5720
  LED off = 0, LED on = 1
  Example: 000001 = 0.5 dB, 001100 = 6 dB, 111111 = 31.5 dB

Electronic schematic

You'll find the electronic schematic the pdf file : Schéma_électronique_ADRF5720

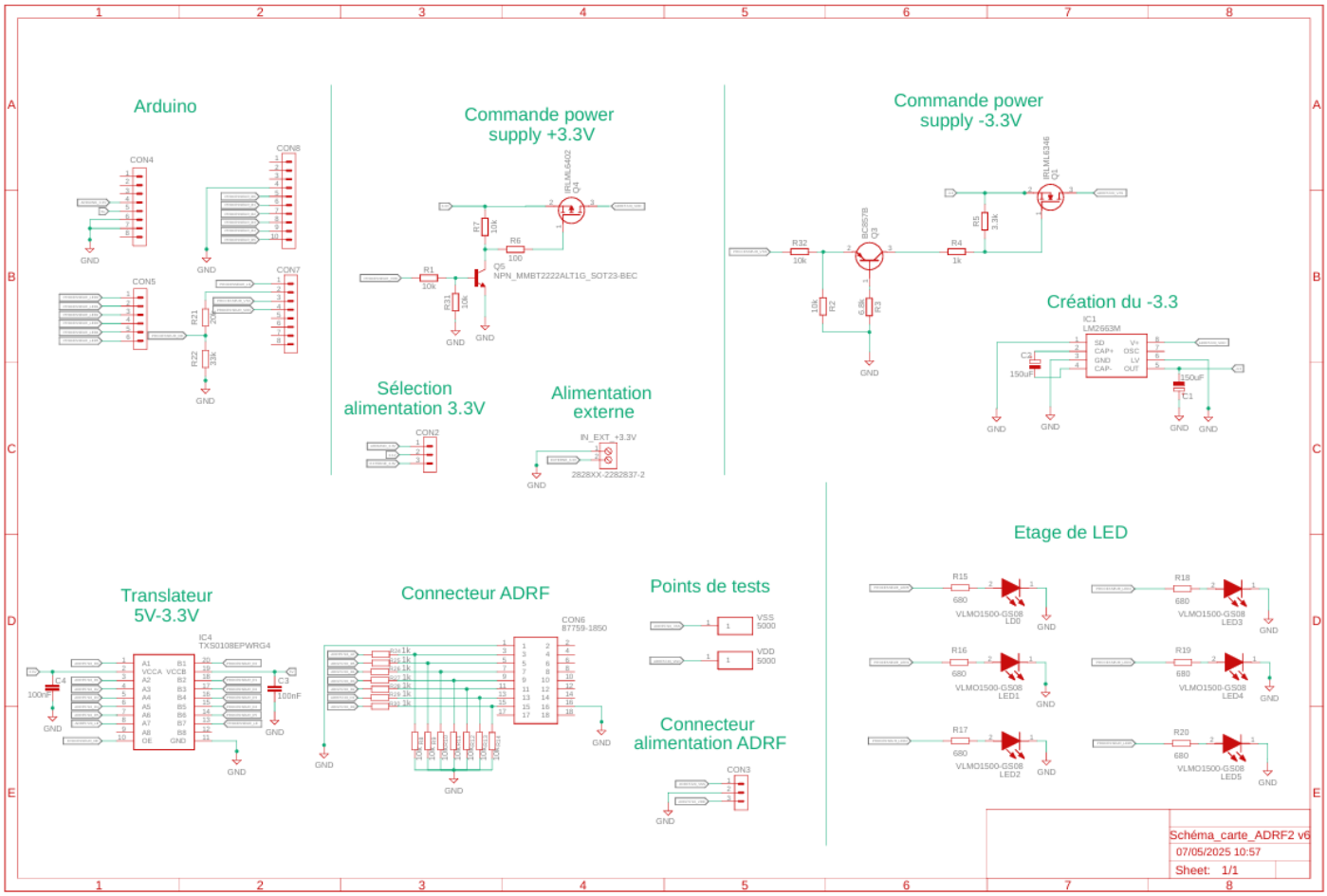

Figure 1: Electronic schematic of the ADRF5720 control shield board

3D view of the board

It's an example 3D view because some of the component don't have the 3D package linked to their library.

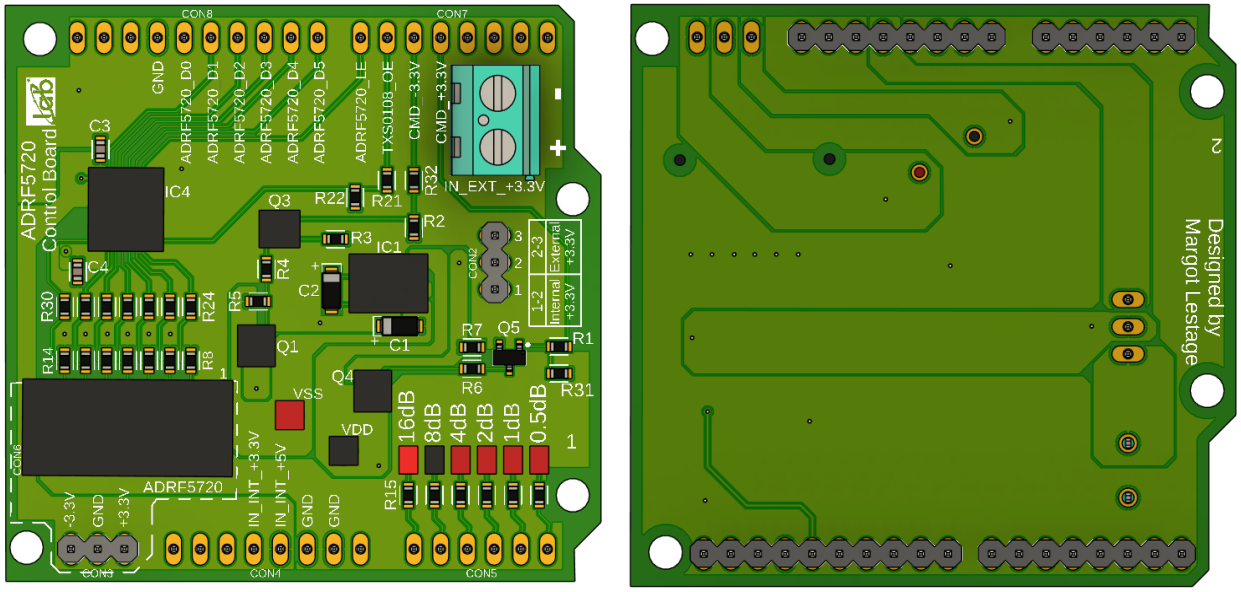

Figure 2: 3D view of the ADRF5720 control shield

Pictures of the board

Below are photos of the final shield board used to control the ADRF5720.

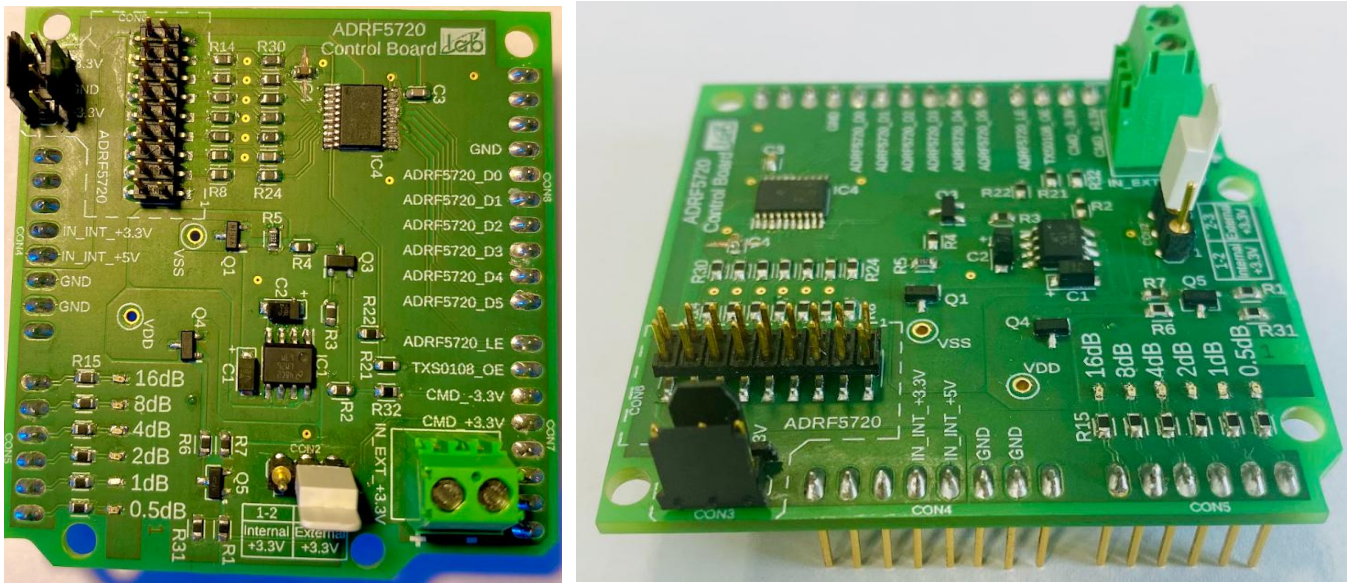

Figure 3: Photos of the ADRF5720 control shield

3. ADRF5720 Operation

The ADRF5720 is a digitally adjustable RF attenuator. It allows control of the RF signal attenuation from 0 to 31.5 dB with 0.5 dB steps. Its bandwidth is from 9 kHz to 40 GHz, which fits the needs of the ALMA project. On the control board, the module is controlled using a 6-bit parallel interface.

4. Embedded code

The firmware embedded in the Arduino or STM32 is responsible for the correct power-up and power-down sequences of the ADRF5720, receiving serial commands, and sending attenuation bits. It also includes protection against syntax or usage errors.

The embedded code manages:

- Power state (on/off)
- Sending attenuation values to the ADRF5720
- Synchronization using the LE (Latch Enable) signal
- LED visualization of the attenuation bits

4.1 Uploading the Embedded Code

➢ Case 1 – Arduino Uno:

1. Open the file Code_C_ADRF5720.ino in the Arduino IDE
2. In the Tools menu → Board, select Arduino Uno
3. In Tools → Port, select the COM port of the Arduino (visible in Device Manager)
4. Click the Upload button (arrow icon)

➢ Case 2 – STM32 Nucleo (STM32F103RB):

1. Open the file ADRF5720_code_STM320.ioc in STM32CubeMX
2. Generate the code via Project → Generate Code, then open the project in STM32CubeIDE
3. Compile and flash the board

The shield is connected with asymmetrical headers, so you cannot insert it the wrong way.

5. Python Control Software

A Python script lets the user communicate with the Arduino via serial communication. It supports individual attenuation commands and automatic attenuation sequences.

The script includes:

- Reading the port and settings from a JSON file
- Sending simple commands like att 6.0
- Running attenuation series (example: from 0 to 31.5 dB, 1 dB steps, 10s delay)
- Displaying responses and errors

5.1 Running the Python Script

1. Open the file Code_Python_ADRF5720_V5.py n your Python editor (VS Code, etc.)
2. Install the required library in the terminal: pip install pyserial
3. Make sure the file Parameter.json is in the same folder as the script
4. In VS Code, click on "Run Python File in Terminal" at the top right

→If you're using another terminal or editor: python Code_Python_ADRF5720_V5.py

6. JSON Configuration File

The file Parameter.json contains the serial communication parameters and custom error messages.
You must adapt this file to your system. Check which COM port is used by the Arduino or STM32 in Device Manager, and update the JSON file.

Example :

```
{
 "port": "COM3",
 "baudrate": 9600,
 "errors": {
    "ERR0": "Commande inconnue. Utilisez 'power' ou 'att'.",
    "ERR1": "Commande mal formée. Vérifiez la syntaxe.",
    "ERR2": "Valeur de power invalide.",
    "ERR3": "Erreur de format : utilisez '.' et non ',' ",
    "ERR4": "Valeur hors limite (0 - 31.5 dB par pas de 0.5).",
    "ERR5": "Alimentation éteinte : allumez-la d'abord."
  }

}
```

7. Utilisation

1. Plug the shield into the Arduino Uno or STM32 Nucleo-F103RB
2. Connect the board to the PC using a USB cable
3. Upload the embedded C code (see previous instructions)
4. Check the COM port in Windows Device Manager
5. Edit the JSON configuration file if necessary
6. Run the Python script
7. Select the power source using the jumper:
   - Internal: position 1-2
   - External: position 2-3
8. Use the commands to control the attenuation

Command	Description
att	Returns the current attenuation
att 'value'	Sets the attenuation (value between 0 and 31.5 dB)
power	Returns the current power status
power on/off (1/0)	Turns power ON or OFF
serie	Starts an attenuation series with user-defined step, delay and start value
exit	Quits the script and powers off the ADRF5720

9. Observe the LEDs to visually confirm the attenuation values
   (Each LED represents 1 bit: ON = 1, OFF = 0)

8. Warning

Do not change the embedded C code unless you fully understand the protection mechanisms, especially those related to power sequencing (VDD before VSS at startup, VSS before VDD at shutdown) and serial command validation.

Project developed by Margot Lestage