

# HANI Board

## User's Guide

## Outline

1. Introduction .....	4
1.1. Description .....	4
1.2. Kit contents .....	4
1.3. LCD mounting .....	5
2. System overview .....	6
2.1. Block diagram .....	6
2.2. Main devices .....	7
2.3. Debug probes.....	8
2.3.1 Keil ULink-ME .....	8
2.3.2 Segger JLink .....	8
3. Getting started .....	9
3.1. HANI pre-programmed demo .....	9
3.2. Creating and debugging a sample “Blinky” project.....	10
3.2.1 Prerequisites .....	10
3.2.2 Creating and building a demo project .....	12
3.2.3 Debugging the project .....	15
3.3. Developing and debugging on R41Z multi-protocol radio module.....	15
3.4. Communicating with default BLE demo firmware.....	16
3.5. Communicating with default Wi-Fi demo firmware .....	17
3.6. Communicating with NFC device.....	17
3.7. Creating and flashing an Embedded Wizard project .....	20
3.7.1 Prerequisites .....	20
3.7.2 Opening and building a demo project .....	20
4. Connectors .....	22
4.1. Arduino expansion connector .....	24
4.2. JTAG connectors.....	24
5. Usage .....	25
5.1. Power supply.....	25
5.2. Power enable and measurement.....	25
5.3. User interface.....	26
6. Board layout.....	27

## Revisions

REVISION	DATE	DESCRIPTION	STATUS	AUTHOR	REVISER
0.1	15/05/2018	Document created	draft	L. Dal Bello	
0.5	28/05/2018	Document released to Arrow	draft	L. Dal Bello	
0.6	14/06/2018	New sections added: Debug probe connection	draft	L. Dal Bello	
0.7	28/09/2018	Updates for new MCUXpresso and SDK Additional explanation for LCD mounting, BLE and Wi-Fi communication, NFC management Embedded Wizard integration guide	draft	L. Dal Bello	

## Disclaimer

All rights strictly reserved. Reproduction in any form is not permitted without written authorization from RELOC s.r.l.

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by RELOC s.r.l. without notice.

**RELOC s.r.l.**

HEADQUARTERS

Strada Langhirano, 264/3A - 43124 – Parma (Italy)

Phone +39-0521-649116

[www.reloc.it](http://www.reloc.it)

[info@reloc.it](mailto:info@reloc.it)

# 1. Introduction

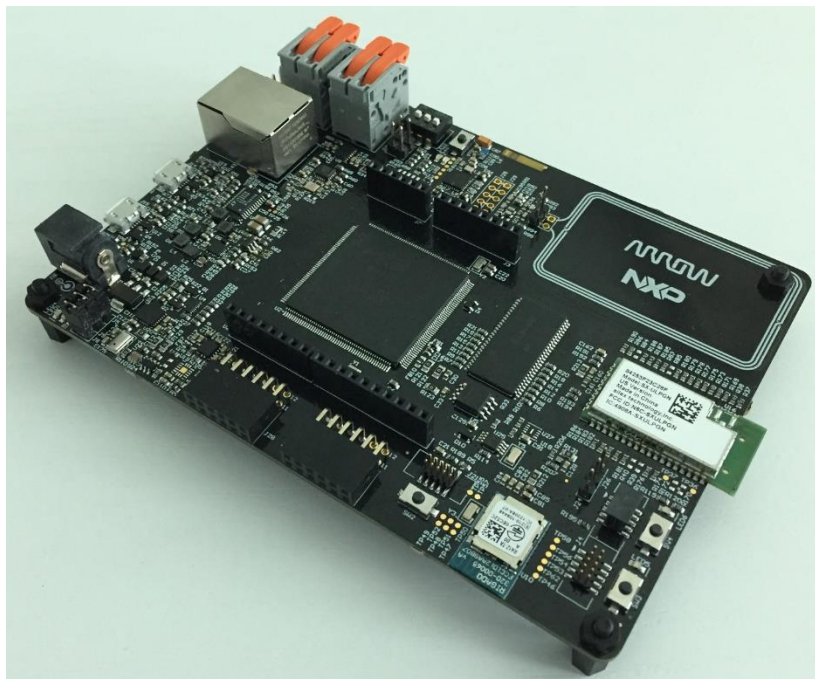
## 1.1. Description

HANI, developed by RELOC for Arrow Electronics, is a board focused on HMI (Human Machine Interface) supporting multiple display sizes and connectivity protocols with a multi-protocol wireless module, a Wi-Fi module, a NFC reader, CAN, Ethernet and USB interfaces. On-board sensors make the HANI board a robust IoT kit.

## 1.2. Kit contents

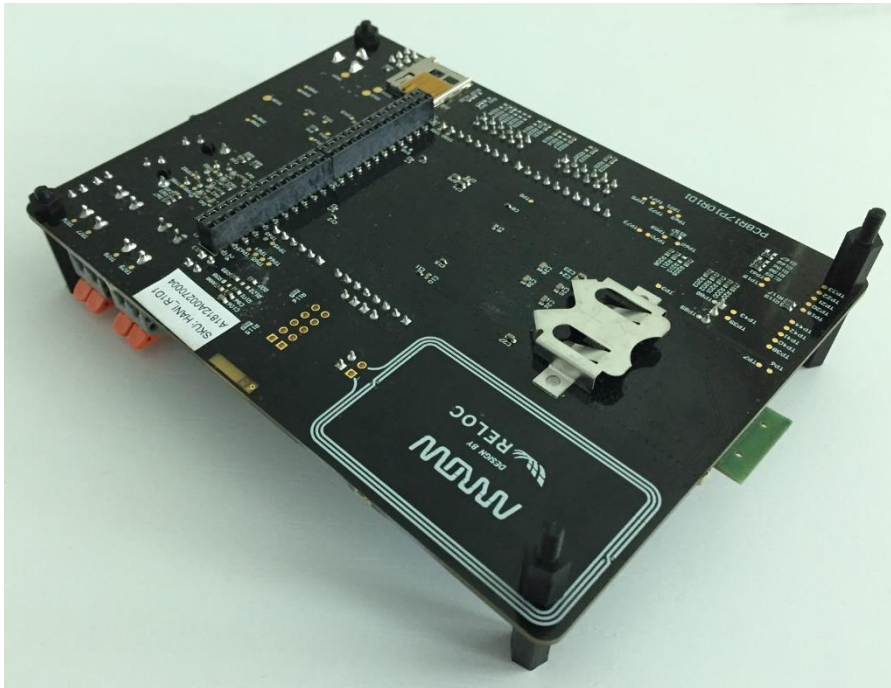
The following items are included in the box:

- 1x HANI board
- 1x 5V 2A power wall adapter (input: 100-240 V, 50/60 Hz)
- 4x 10mm plastic spacers mounted on the board, 8x additional 20mm plastic spacers



### 1.3. LCD mounting

In order to add optional LCD (sold separately), please remove the plastic spacers, reverse the board and mount additional 20 mm plastic spacers on the other side:



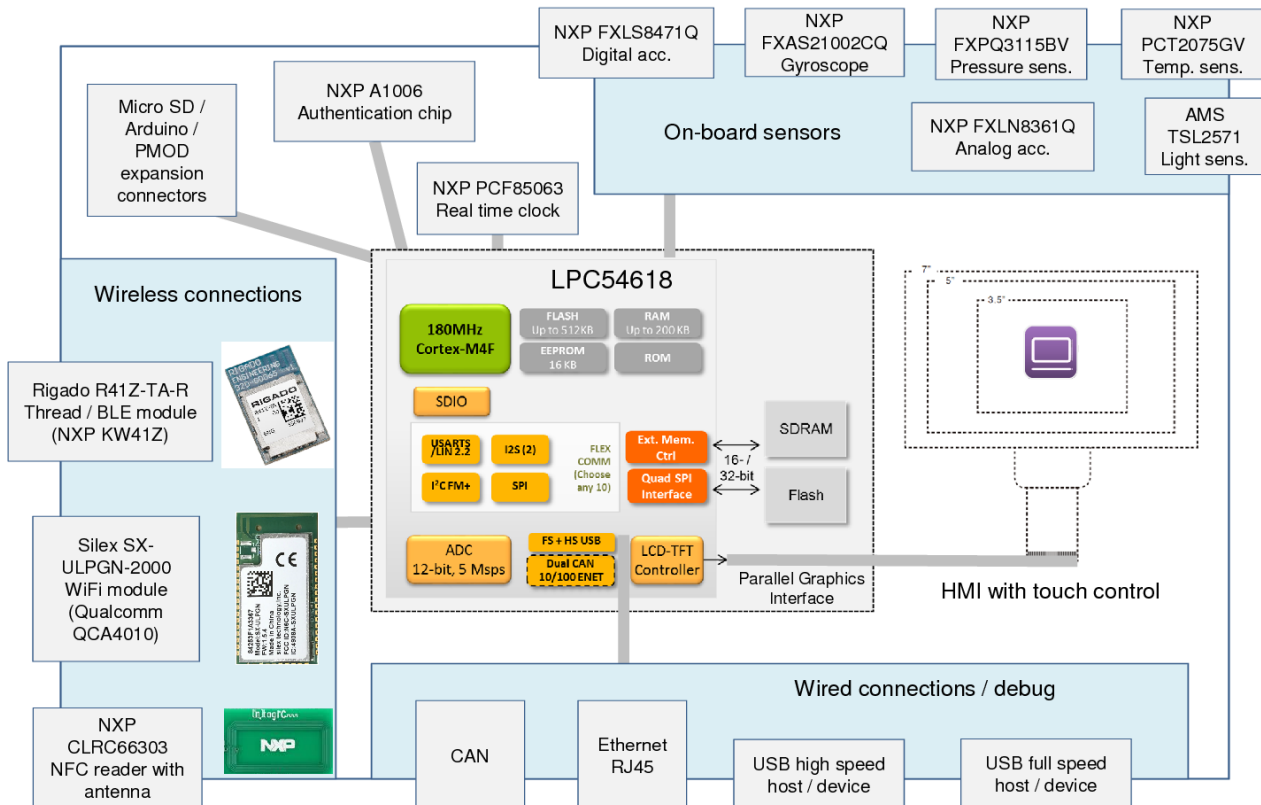
You can re-use 10 mm plastic spacers (like in the picture) to provide additional support for the LCD board; then place the display on the HANI board, checking proper orientation of the connector:



## 2. System overview

### 2.1. Block diagram

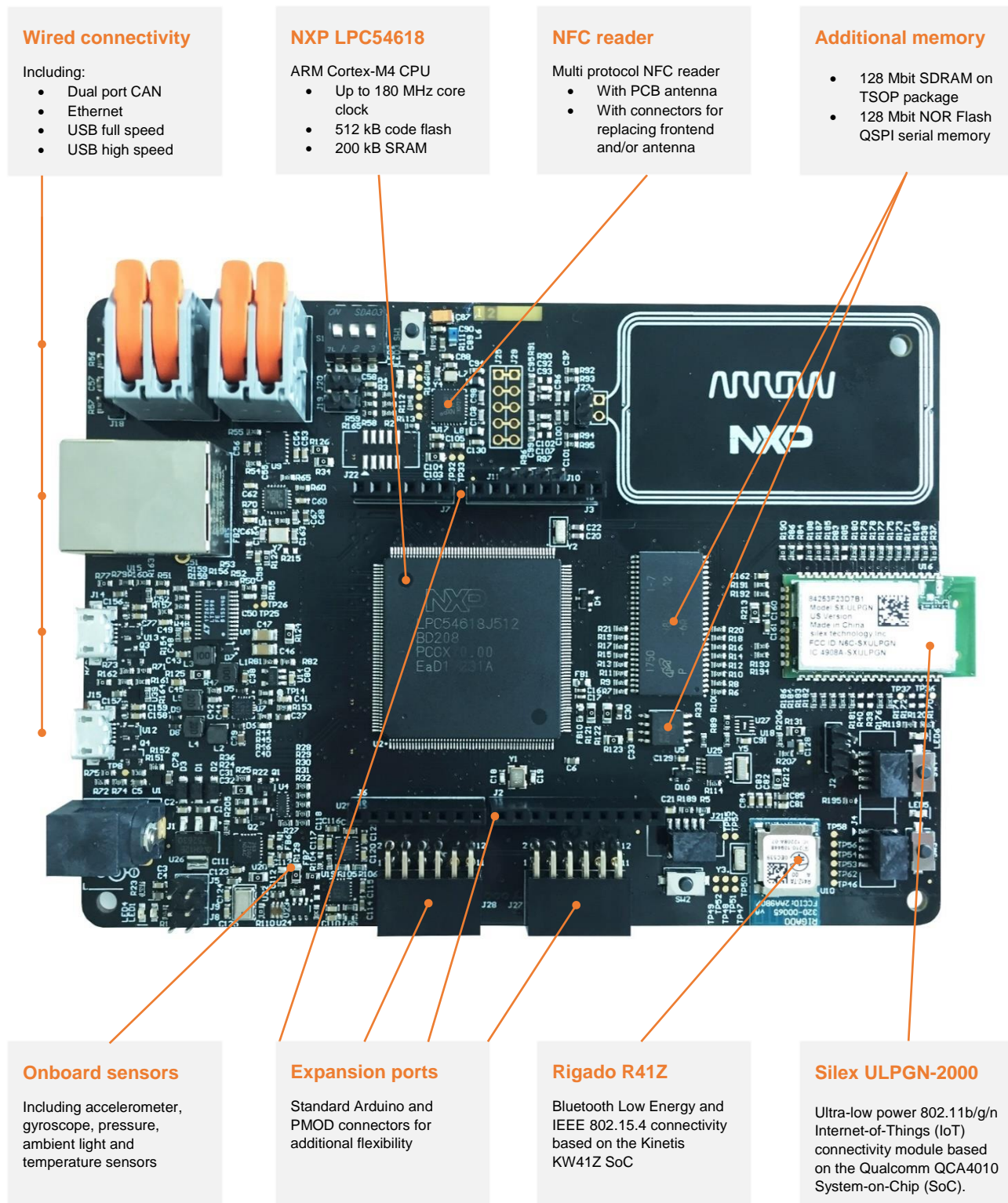
An overview of the functional blocks of HANI board is shown in the figure below.





## 2.2. Main devices

HANI board main features are summarized in the picture below.



## 2.3. Debug probes

An external debug probe is required to flash and debug the software; any device supported by NXP MCUXpresso should work properly, additional instructions for tested interfaces are provided below.

### 2.3.1 Keil ULink-ME

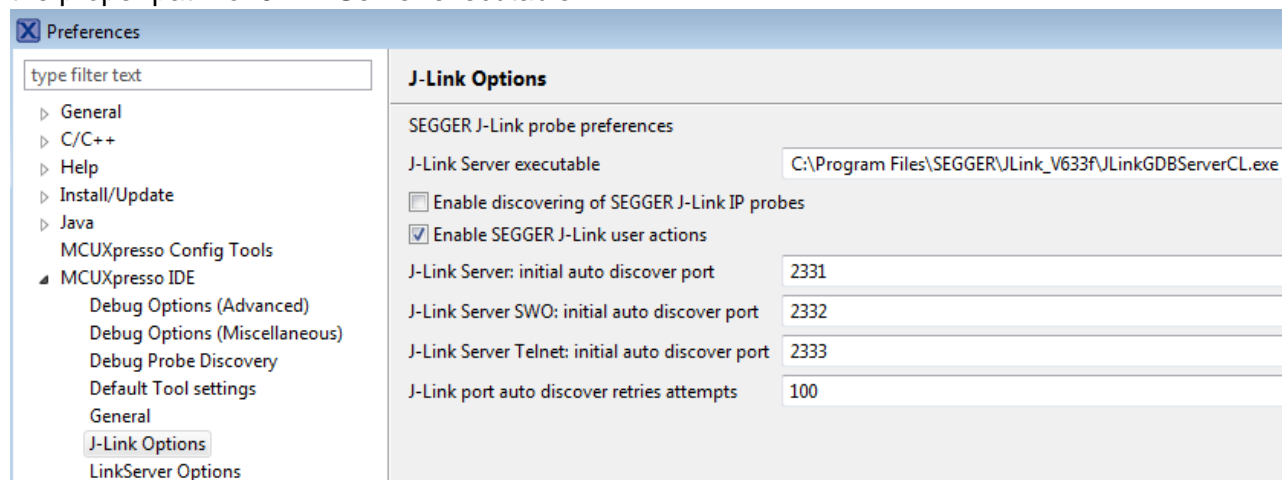
This probe should be automatically detected and used by MCUXpresso IDE.

### 2.3.2 Segger JLink

JLink V6.33f or later release can be used with HANI board; please download and install the software from Segger website:

<https://www.segger.com/downloads/jlink/>

After installation is complete, please open Window → Preferences menu in MCUXpresso and select the proper path for J-link Server executable:



Check also that discovery of Segger J-Link probes is enabled in Preferences → MCUXpresso IDE → Debug Probe Discovery.



## 3. Getting started

### 3.1. HANI pre-programmed demo

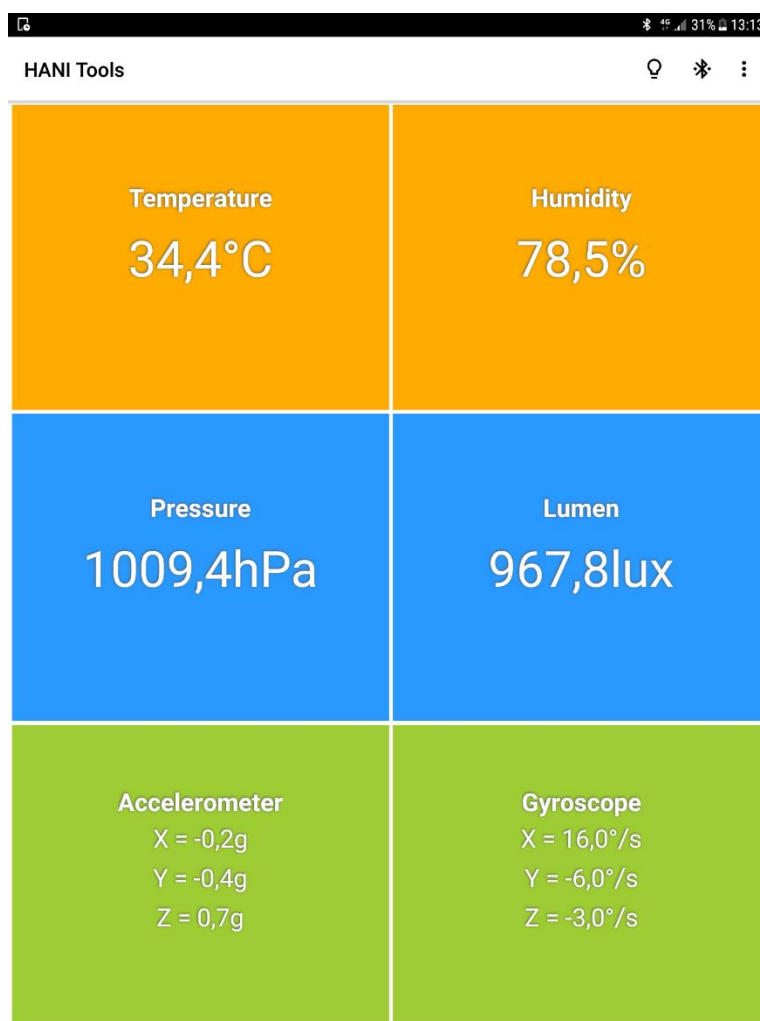
The HANI board is pre-programmed with a demo application which continuously monitors the on-board sensors, sends collected data on Bluetooth Low-Energy (BLE) network and shows them on 4.3" display (if connected).

The NXP LPC processor takes care of sampling sensors data whereas the KW41Z multi-protocol module establishes the BLE communication with a BLE client.

In order to test HANI functionalities by means of pre-programmed demo application

1. download the free HANI tools smartphone App from Google Play™ ( <https://play.google.com/store/apps/details?id=com.demo.rlc.hanitool> );
2. power supply the HANI board through J1 power connector
3. open the ARIS Tools smartphone App and select your HANI from the device list.

The application will show sensor values in real-time through the BLE communication channel.



## 3.2. Creating and debugging a sample “Blinky” project

### 3.2.1 Prerequisites

The following development tools should be installed before trying to either create a project or use any example based on HANI board:

- MCUXpresso Integrated Development Environment (IDE) v10.1.0 or later release

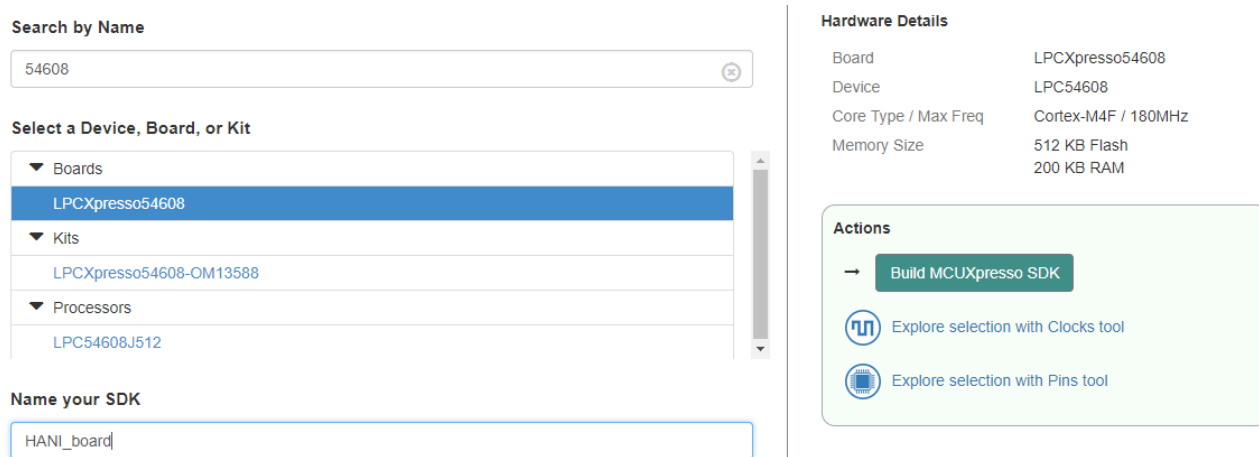
<http://www.nxp.com/mcuxpresso/ide>

- MCUXpresso SDK for LPC54608J512 v2.4.0 or later release

You can build it by yourself starting from this page:

<https://mcuxpresso.nxp.com/en/welcome>

“Select Development Board” and choose board LPCXpresso54608:



“Build MCUXpresso SDK”, “Add software component”, add “emWin” and “FreeRTOS Kernel” and “Save changes”

Middleware	
AWS IoT	
CMSIS DSP Library	
DMA Manager	
FatFS	✓
LVHB	
NTAG I2C	
QCA400x WiFi	
Secure Element	
Sigfox	
USB Type-C PD stack	
USB stack	✓
emWin	✓
littlefs	
lwIP	✓
mbedtls	
nand flash management stack	
sddmmc stack	
wolfssl	
Operating System	
Amazon-FreeRTOS Kernel	✓

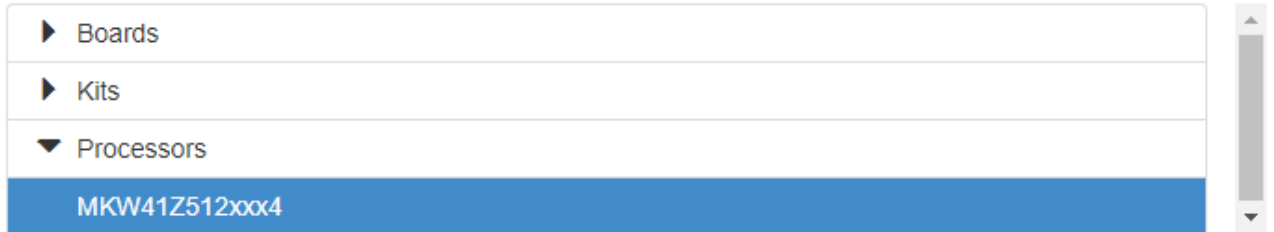
“Request build” and “Download SDK archive”

- MCUXpresso SDK for MKW41Z512xxx4 v2.2.0 or later release (optional, for KW41Z development)


<https://mcuxpresso.nxp.com/en/welcome>

“Select Development Board” and choose processor MKW41Z512xxx4:

**Select a Device, Board, or Kit**



**Name your SDK**



“Build MCUXpresso SDK” and “Download SDK”

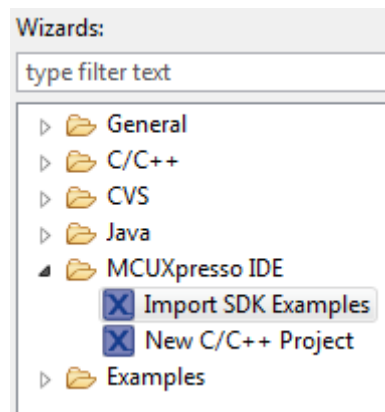
SKDs should be installed from within MCUXpresso, please refer to official documentation.

Please refer to NXP website ( [www.nxp.com](http://www.nxp.com) ) if previous links are outdated.

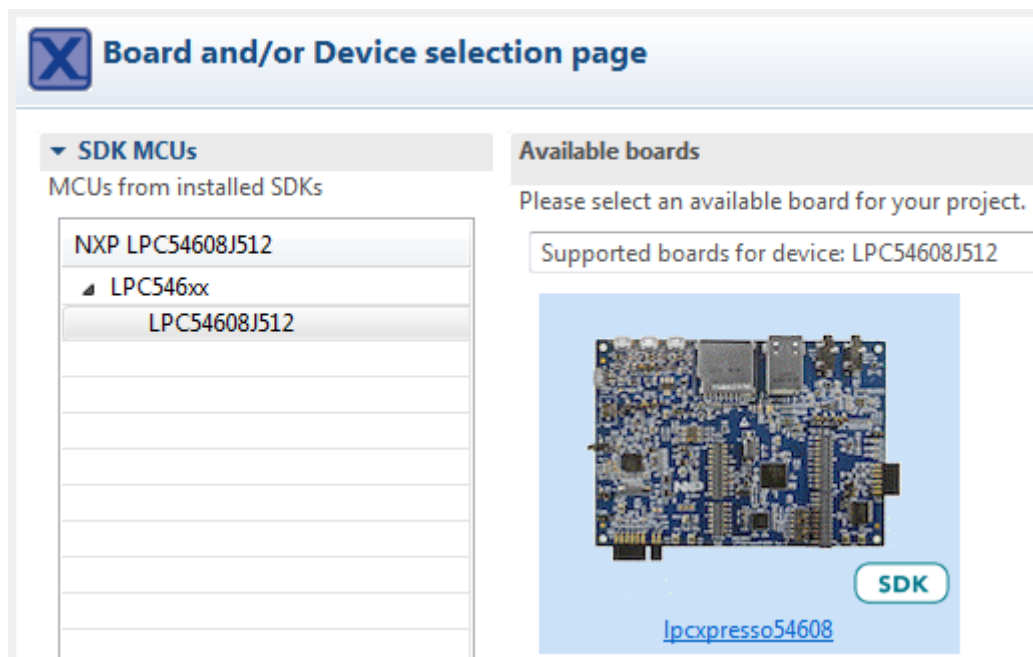
### 3.2.2 Creating and building a demo project

To create a new project with HANI board as a target, please follow these steps within MCUXpresso:

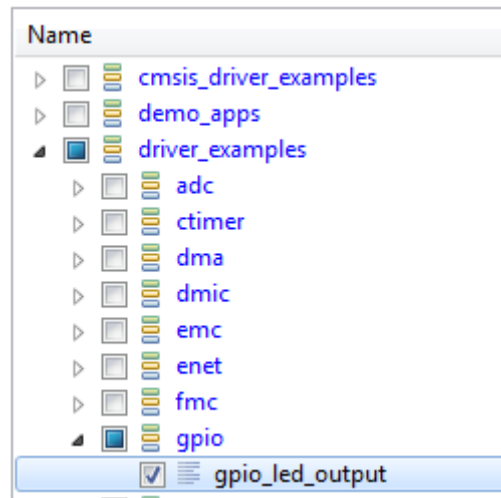
- Select: File → New → Project in the menu and then select: MCUXpresso IDE -> Import SDK Examples



- Select: lpcxpresso54608 board on the right panel and press Next to continue:



- Assign a project name and select an example (e.g. gpio\_led\_output) as a starting point:



- Press the “Finish” button twice to confirm the selection

Then you should change the source code, as required by the example and by your target application; for gpio\_led\_output example:

- Open source → gpio\_led\_output.c
- Modify existing Definitions:

```

/*****
 * Definitions
 *****/

```

```

#define APP_BOARD_TEST_LED_PORT 2U
#define APP_BOARD_TEST_LED_PIN 2U
#define APP_SW_PORT 0U
#define APP_SW_PIN 6U

```

With proper definitions for current board:

```

/*****
 * Definitions
 *****/

```

```

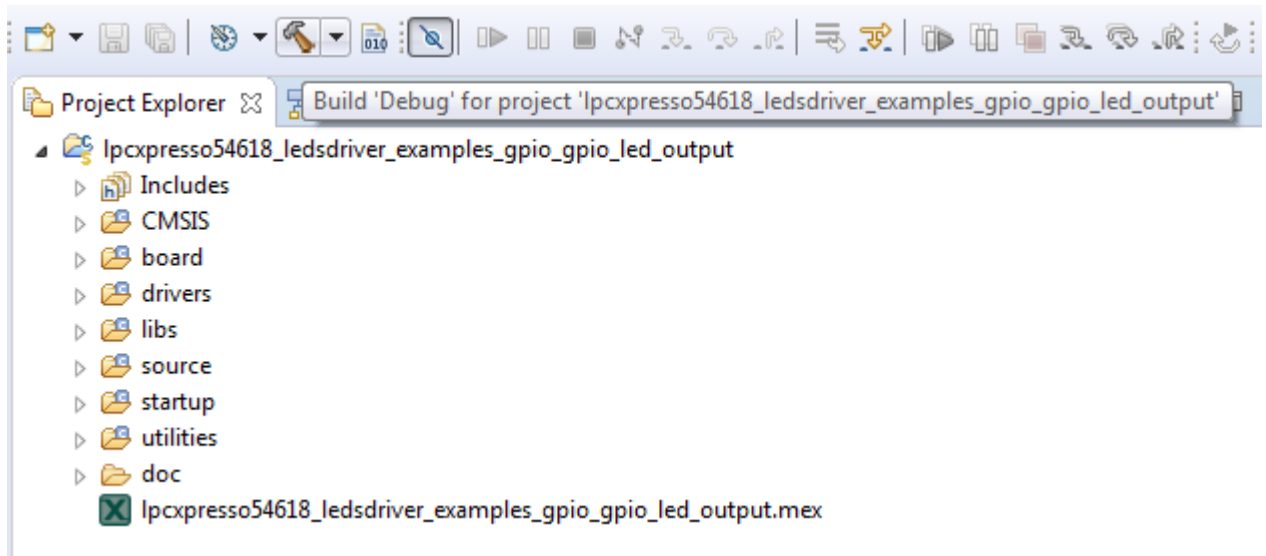
#define APP_BOARD_TEST_LED_PORT 5U
#define APP_BOARD_TEST_LED_PIN 2U
#define APP_SW_PORT 0U
#define APP_SW_PIN 29U

```



Modify existing Definitions:

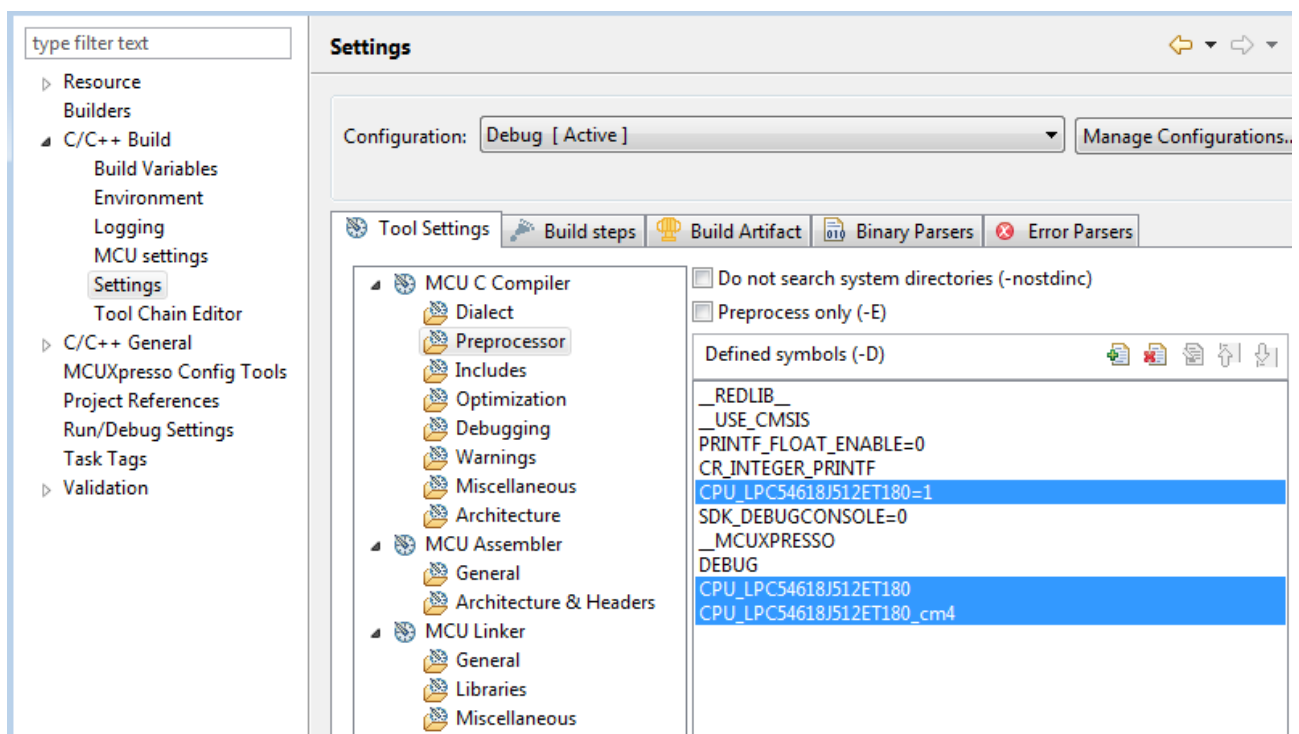
- You should now be able to build the project by clicking on the Build button:



Compilation should end without any error.

If you receive #error "No valid CPU defined!":

- Open Project setting and change all references from LPC54618J512ET180 to LPC54618J512BD208



If you receive #error ./log/fsl\_log.h: No such file or directory:

- Open Utilities → fsl\_debug\_console.c
- Change rows #69-70 from:

```
#include "../log/fsl_log.h"
#include "../str/fsl_str.h"

to:
#include "fsl_log.h"
#include "fsl_str.h"
```

If you receive `#error: ../io/fsl_io.h: No such file or directory`

- Open Utilities → `fsl_log.c`
- Change row #37 from:

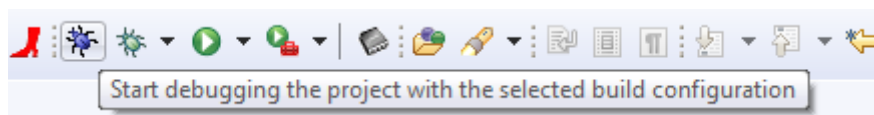
```
#include "../io/fsl_io.h"

to:
#include "fsl_io.h"
```

### 3.2.3 Debugging the project

Connect a Keil ULink-ME or compatible debugger to J21 pin header, then provide power to the board through J1 power connector.

In order to check the correct behavior of the sample application start the Debug by selecting this icon in the menu:



Select the appropriate debug interface if required, then click Resume button (F8) to start the execution:



Pressing SW1 switch on the board will blink LED2.

## 3.3. Developing and debugging on R41Z multi-protocol radio module

The same procedure can be used for the development on R41Z multi-protocol radio module, setting KW41Z as a target device.

### 3.4. Communicating with default BLE demo firmware

The default firmware pre-loaded on Rigado R41Z module creates a standard BLE GATT server, accessible from a single user, exposing the following UUIDs:

```
0000AA002016050500004E10C0000000 /* Environment Service */
0000AA0100001000800000805F9B34FB /* Temperature Characteristic */
0000AA0200001000800000805F9B34FB /* Humidity Characteristic */
0000AA0300001000800000805F9B34FB /* Pressure Characteristic */
0000AA0400001000800000805F9B34FB /* Lux Characteristic */

0000AA102016050500004E10C0000000 /* Accelerometer Service */
0000AA1100001000800000805F9B34FB /* ACC_X Characteristic */
0000AA1200001000800000805F9B34FB /* ACC_Y Characteristic */
0000AA1300001000800000805F9B34FB /* ACC_Z Characteristic */

0000AA202016050500004E10C0000000 /* Accelerometer Service */
0000AA2100001000800000805F9B34FB /* ACC_X Characteristic */
0000AA2200001000800000805F9B34FB /* ACC_Y Characteristic */
0000AA2300001000800000805F9B34FB /* ACC_Z Characteristic */

0000AA502016050500004E10C0000000 /* LED Service */
0000AA5100001000800000805F9B34FB /* LED Characteristic */
```

The LPC microcontroller communicates with Rigado module through an asynchronous channel (115200 bps, 8 bits data, 1 bit stop, no parity) on FlexComm 4 port.

The protocol from Rigado module to LPC MCU manages the following commands:

```
"B \r\n" = SW3 button pressed
"L1 0\r\n" = Command to deactivate the first LED managed by MCU
"L1 1\r\n" = Command to activate the first LED managed by MCU
"L2 0\r\n" = Command to deactivate the second LED managed by MCU
"L2 1\r\n" = Command to activate the second LED managed by MCU
"K \r\n" = Acknowledge of the value written
"C \r\n" = BLE slave device connected
"D \r\n" = BLE slave device disconnected
```

On the other direction, the communication from MCU to BLE module can contain:

```
"T nnnn\n" = Set temperature value to nnnn
"H nnnn\n" = Set humidity value to nnnn
"P nnnn\n" = Set pressure value to nnnn
"I nnnn\n" = Set illuminance value to nnnn
"AX nnnn\n" = Set X accelerometer value to nnnn
```

"AY nnnn\n" = Set Y accelerometer value to nnnn  
 "AZ nnnn\n" = Set Z accelerometer value to nnnn  
 "GX nnnn\n" = Set X gyroscope value to nnnn  
 "GY nnnn\n" = Set Y gyroscope value to nnnn  
 "GZ nnnn\n" = Set Z gyroscope value to nnnn  
 "MX nnnn\n" = Set X magnetometer value to nnnn  
 "MY nnnn\n" = Set Y magnetometer value to nnnn  
 "MZ nnnn\n" = Set Z magnetometer value to nnnn

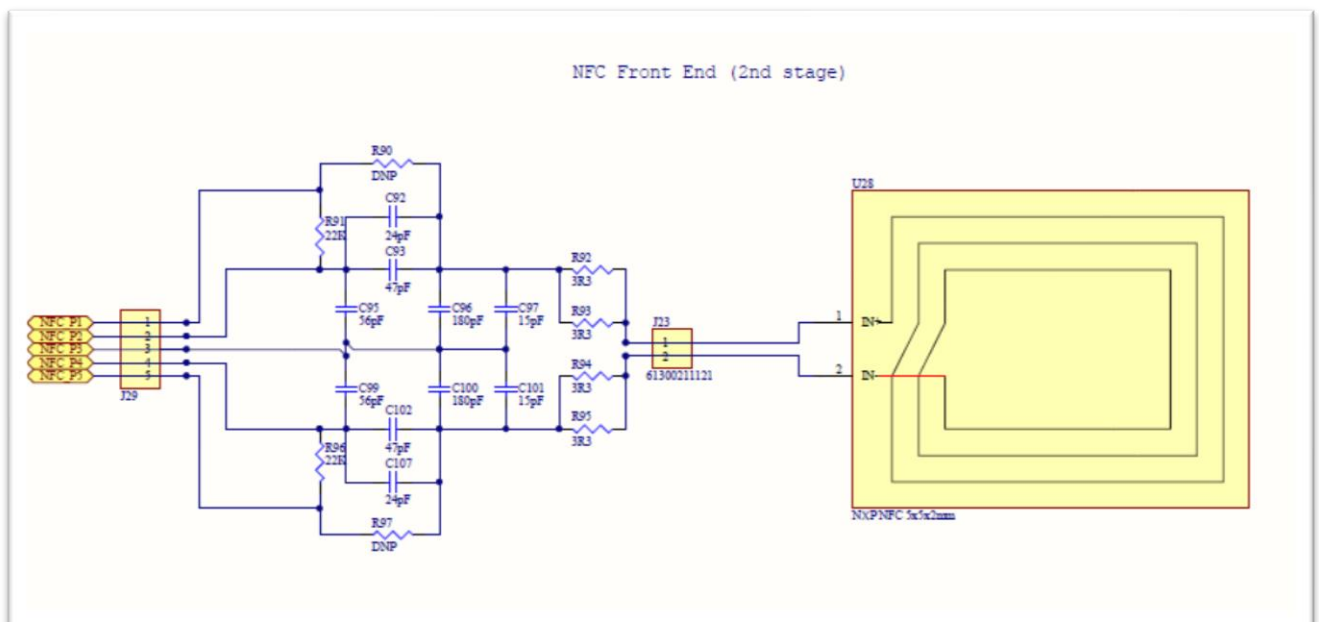
### 3.5. Communicating with default Wi-Fi demo firmware

LPC MCU communicates with Wi-Fi module through a serial interface, using AT commands according to "SX-ULPGN Silex AT Command Specification" r2.12.

<https://www.silextechnology.com/productspecs/silex-at-command-set-for-sx-ulpgn?hsLang=en-us>

### 3.6. Communicating with NFC device

The HANI board has integrated a CLRC663 as NFC Front-end reader and an embedded Nfc Antenna or the possibility to connect a custom external antenna by developer on connector J23.



The documentation, regarding to the Nxp chip, can be downloaded by the following Nxp link:

<https://www.nxp.com/products/identification-and-security/nfc/nfc-reader-ics/clrc663-iplus-i-high-performance-multi-protocol-nfc-frontend-for-gaming:CLRC66303HN>

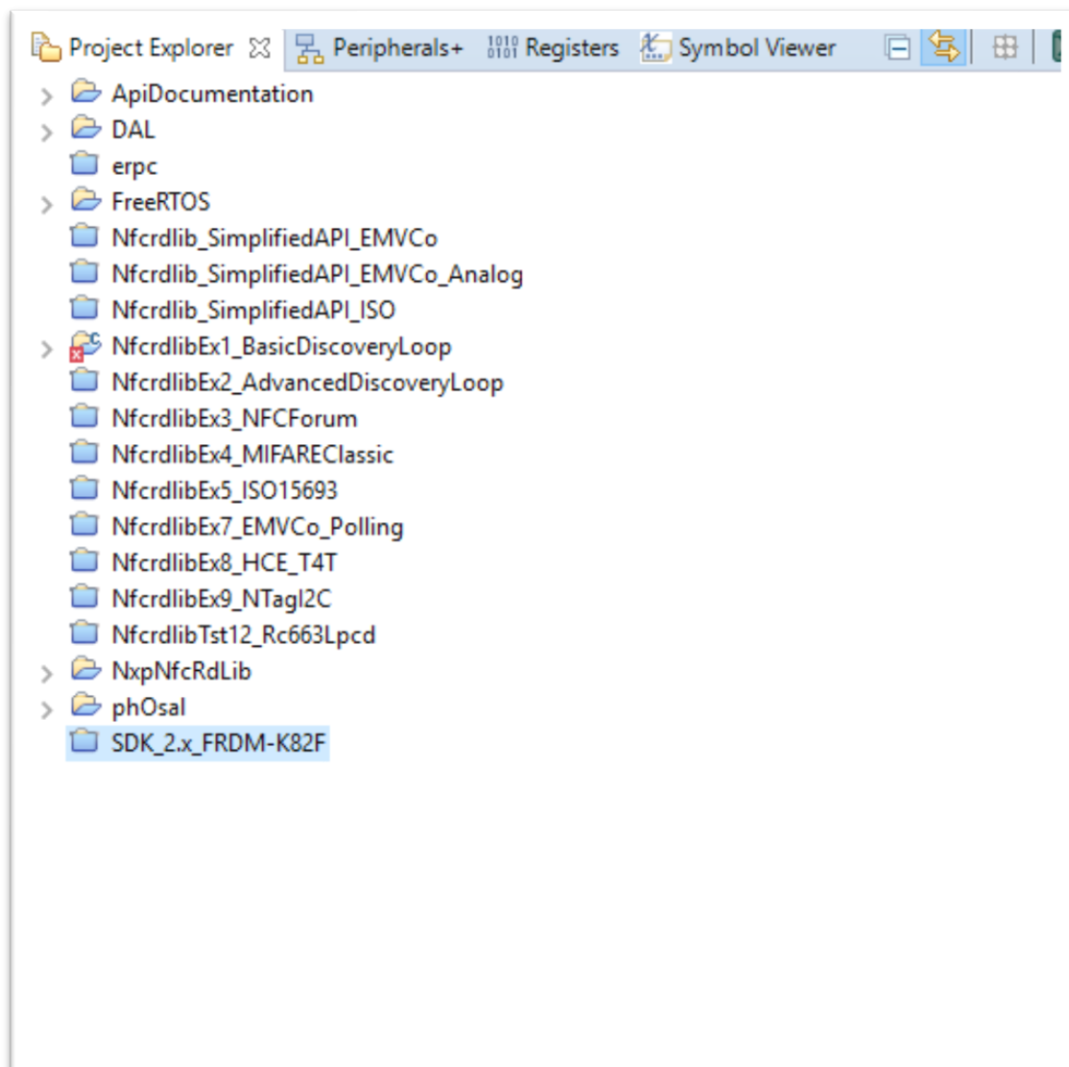
On other side, the LPC MCU communicates with the reader through a SPI interface (Flexcomm 1) and a generic GPIO that triggers an interrupt on pin 98 (port PIO1\_17).

NXP provides a dedicated library in C code to be integrated in your microcontroller and ready to manage the interface. It is composed by 3 sub-folders:

- DAL: Hardware Abstraction (GPIO, TIMERS, SPI,...)
- phOsal: RTOS wrapping on Freertos
- NxpNfcRdLib: the Nfc library

<https://www.nxp.com/products/identification-and-security/nfc/nfc-reader-ics/nfc-reader-library-software-support-for-nfc-frontend-solutions:NFC-READER-LIBRARY>

Moreover, the library has some examples in order to test all the different kinds of Nfc tags.



You can integrate this library in your application following the documentation released by NXP (for example, about the board K82L).

<https://www.nxp.com/docs/en/application-note/AN11908.pdf>



The project “NFC Task”, released in “HANI NFC.rar”, you can find the basic project “NfcrdlibEx1\_BasicDiscoveryLoop” that already provides the Nfc management on LPC54618, then the user can build on top layer his own application through the APIs placed in “../source/APP/NFC/nfc\_task.h” as following:

```
if(nfc_task_get_read_tag_is_new()){
    uint8_t pDst[64];
    uint32_t Len=0;
    if((Len=nfc_task_get_read_tag_payload(pDst, 64)){
        PRINTF("\r\nPayload: (%Lu)->%s\r\n", Len, pDst);

        /* ... Insert Application code ...*/
    }
}
```

As reference of a general purpose implementation, refer to “GameDemo800x480” example that integrates NFC, BLE and LCD.

## 3.7. Creating and flashing an Embedded Wizard project

### 3.7.1 Prerequisites

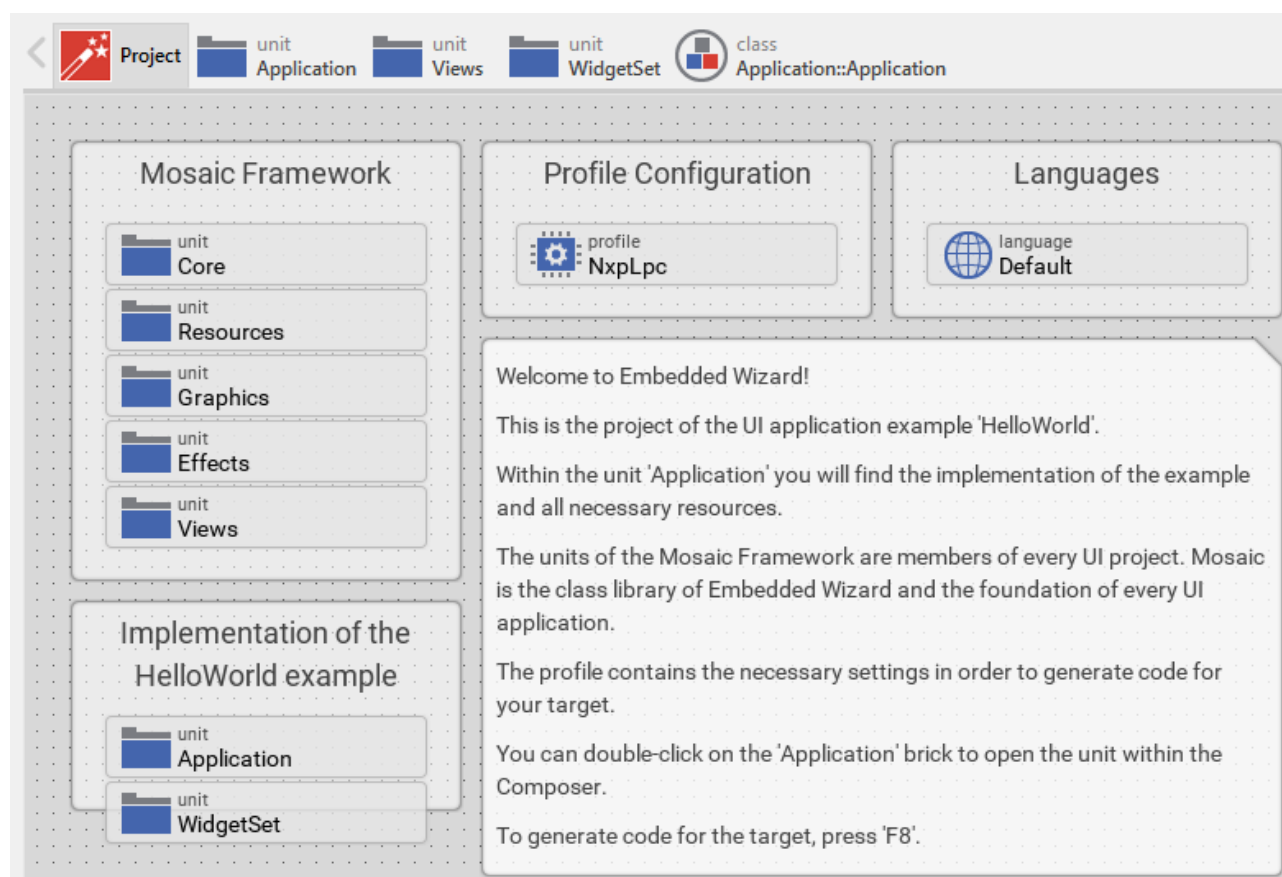
Additionally to what listed in section 3.2.1, in order to work with LCDs and the graphic library, Embedded Wizard v9.0 must be installed; you can download evaluation version here:

<https://www.embedded-wizard.de>

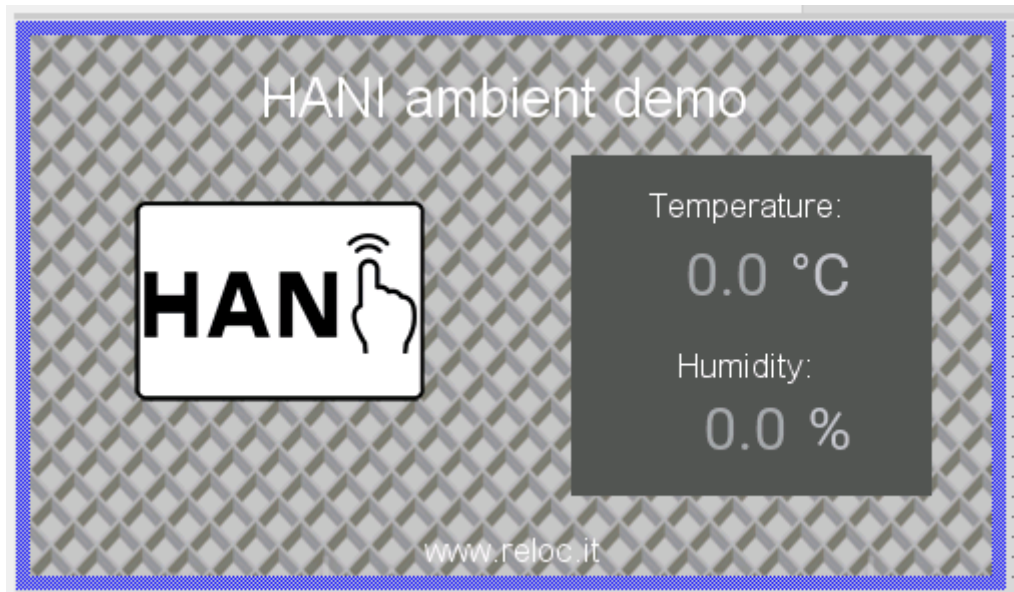
After that, the content of “HANI\_EmbeddedWizard\_1.1.3.zip” archive must be decompressed (typically in C:\NXP folder, an additional subfolder named HANI will be created).

### 3.7.2 Opening and building a demo project

Open an example project (default folder is C:\NXP\Hani\Examples), for example “HaniDemo.ewp” in “HaniDemo - LCD 4.3 inches” subfolder; Embedded Wizard will start showing the project components:



You can change the graphic opening unit Application and then class Application:



You can start simulator pressing CTRL + F5 to test the behavior of the GUI; when satisfied, press F8 to generate the project in the default folder.

After that, do following steps:

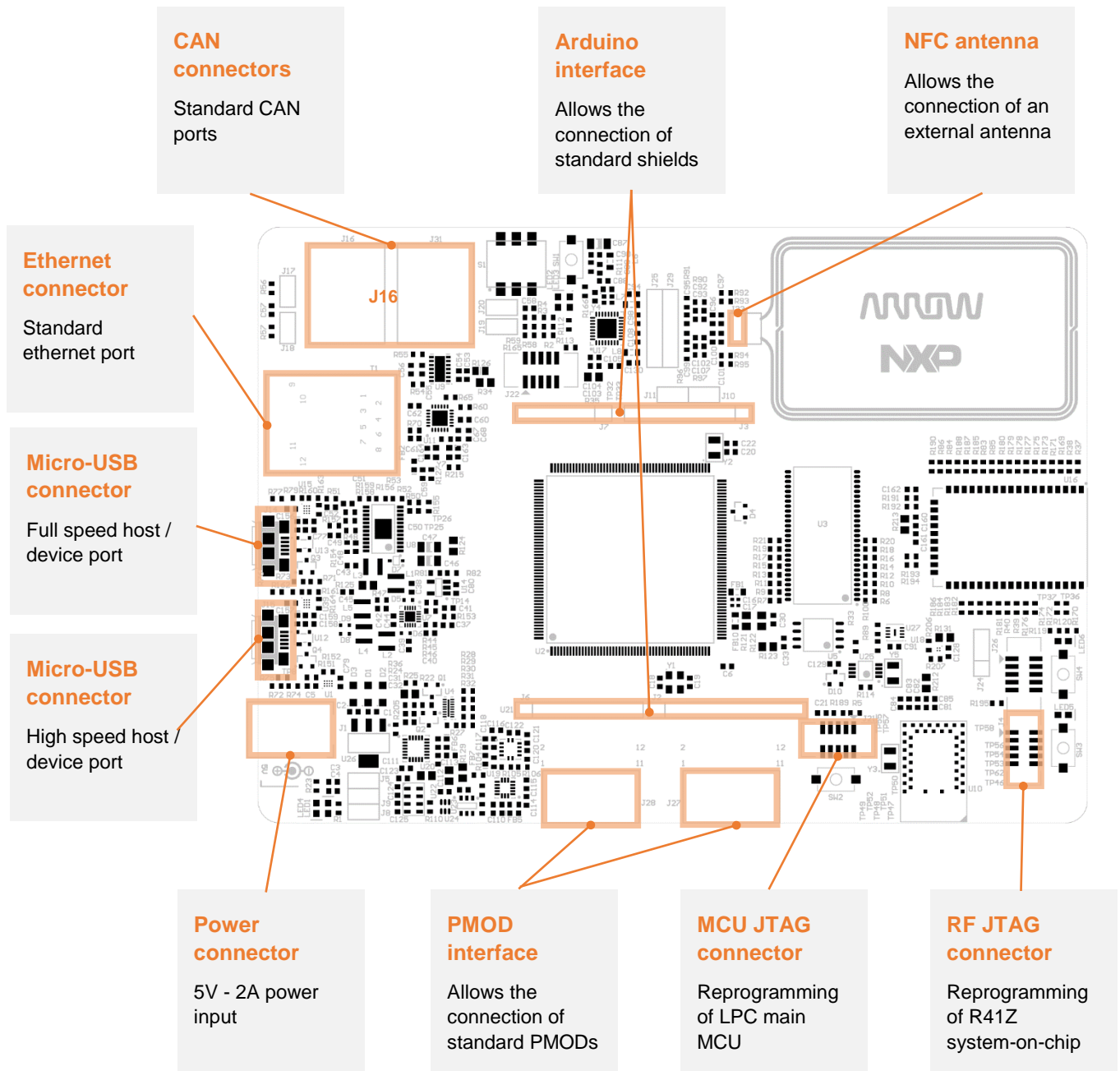
- Open NXP MCUXpresso IDE and select the directory \Application\Project\MCUXpresso as workspace directory.
- To import the C project, select the menu item File ► Import and choose General - Existing Projects into Workspace and press Next.
- Choose Select root directory - Browse and select the directory \Application\Project\MCUXpresso\LPCXpresso54608.
- Press Finish.
- Open TargetSpecific ► Drivers ► fsl\_lcd\_RK043.h and select proper define for specific LCD, example for 4.3" display:

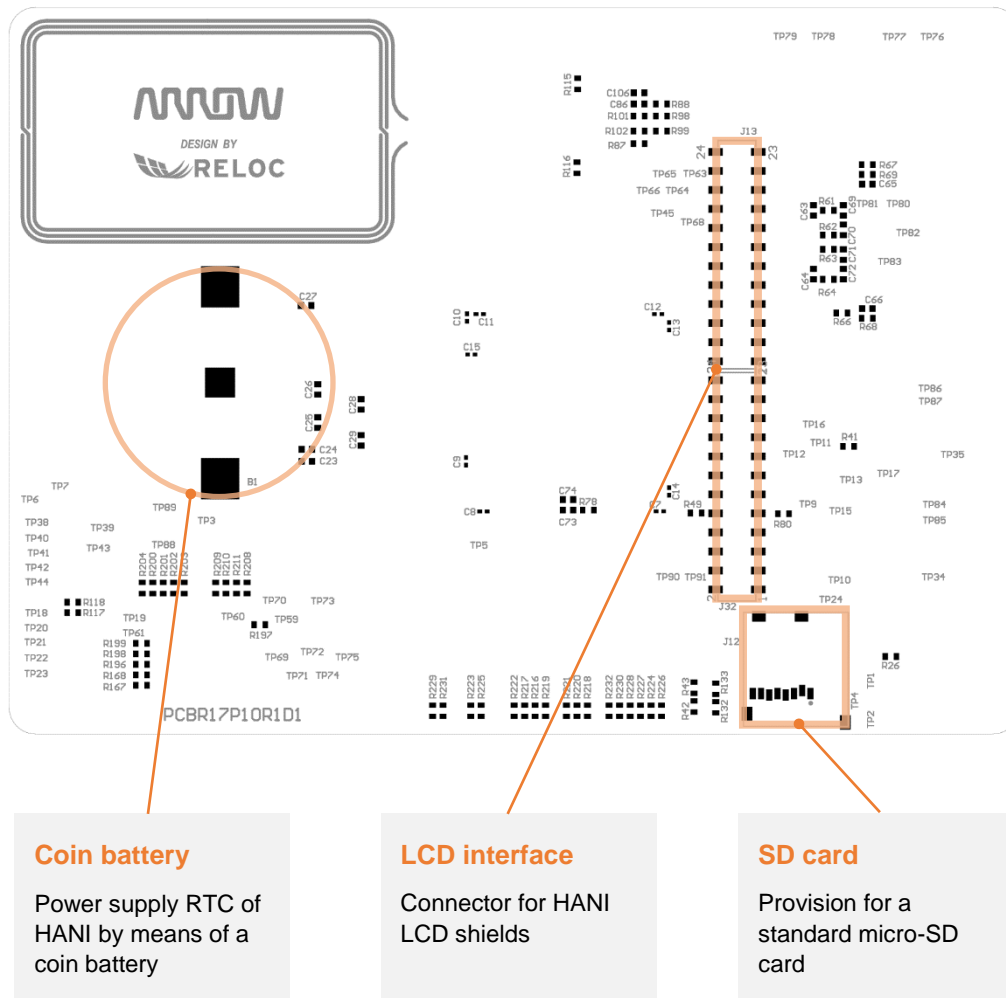
```
// #define LCD_3_5_320x240
#define LCD_4_3_480x272
// #define LCD_5_0_800x480
// #define LCD_7_0_800x480
```

- To compile the project select Project ► Build Project.
- Choose the GUI Flash Tool for running the application on the target or use the build in debug function for debugging the application on your target.

## 4. Connectors

This chapter gives an overview of the main HANI connectors and their functionalities.

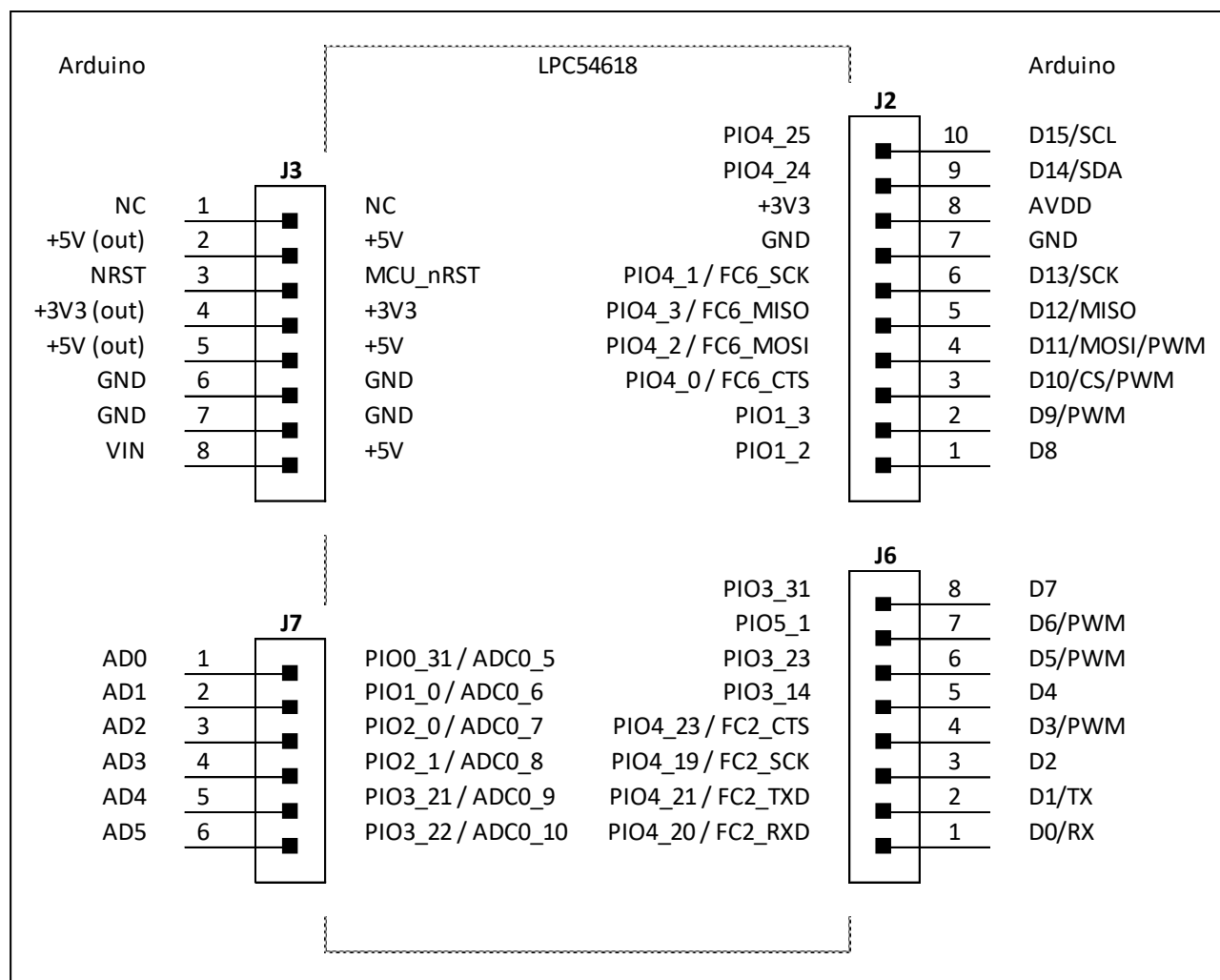






## 4.1. Arduino expansion connector

Connectors J2, J3, J6, J7 provide user with a standard Arduino shield expansion slot.



## 4.2. JTAG connectors

Several JTAG connectors are provided on HANI board for programming and debugging purposes, detailed below for reference:

- J21 – JTAG connector for NXP LPC54618 MCU;
- J4 – JTAG connector routed to Rigado R41Z system-on-chip;
- J26 – JTAG connector for Silex SX-ULPGN-2000 Wi-Fi module;
- J22 (not mounted) – optional JTAG for NXP CLRC6630 device.

## 5. Usage

This chapter describes how to connect, configure and interact with the HANI board.

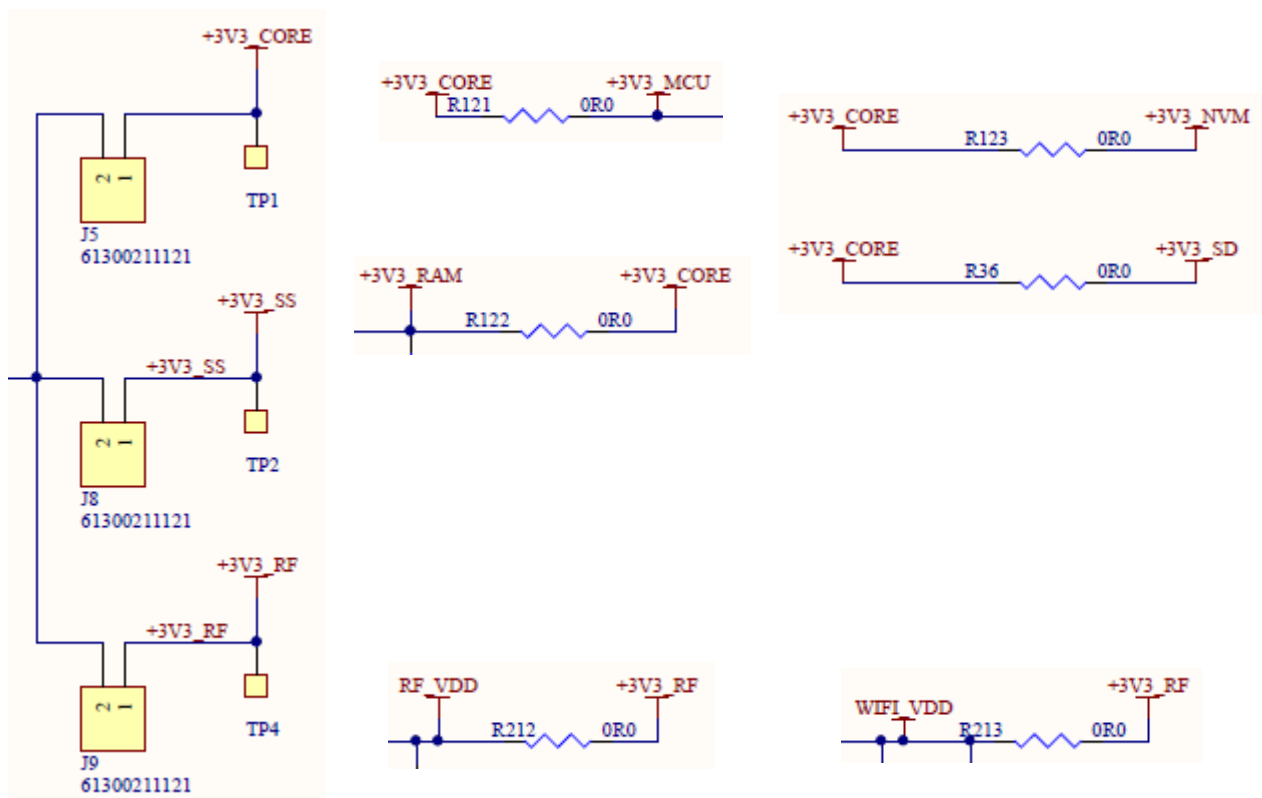
### 5.1. Power supply

The J1 power connector delivers power to the board; please use only the power adaptor provided with HANI board.

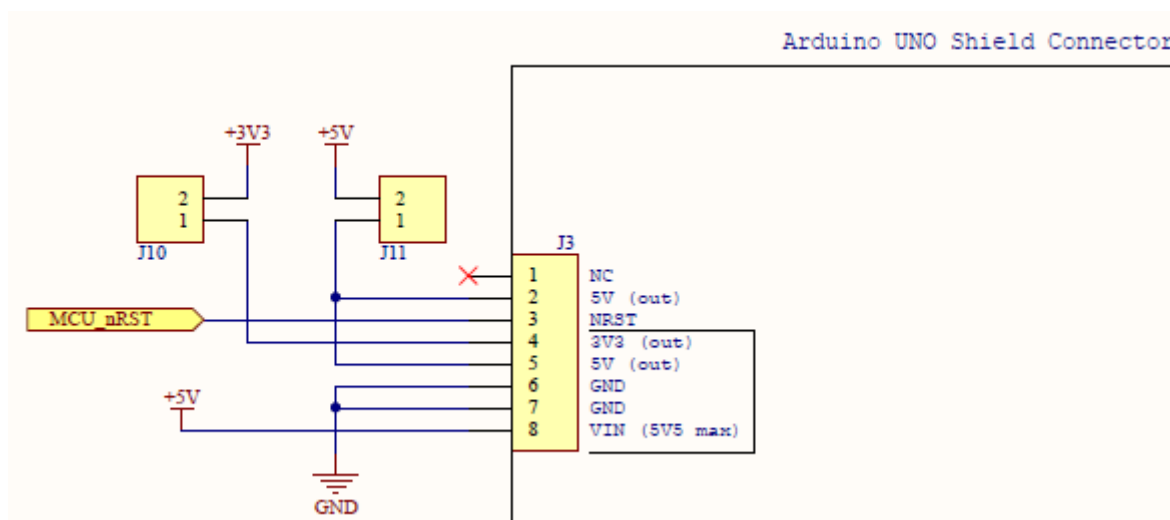
### 5.2. Power enable and measurement

J5, J8 and J9 (populated in the default configuration) provide power to the main logic devices (MCU, RAM and Flash memory), to sensors and radio modules, respectively (i.e. connect the input power sources to the board loads).

They can also be used to measure the current consumption of the HANI circuitry sections.



J10 and J11 (*not* populated in the default configuration) could be used to provide power to the Arduino connectors; before placing a jumper please check that current consumption of the shield is within maximum values provided by the +5V input and +3.3V regulated power supply.



### 5.3. User interface

HANI developer is provided with a basic user interface, including buttons and LEDs directly connected to both NXP LPC54618 MCU (SW1, SW2, LED2, LED3) and other devices.

Mapping between port numbers and the physical designator is provided below.

Device	Port	Schematic Signal	Description
System	-	+3V3	<b>LED1</b> Green LED
NXP LPC54618	PIO0_29	UI_SW1	<b>SW1</b> tactile switch
	RESETN	RESETN	<b>SW2</b> tactile switch connected to MCU reset signal (active low)
	PIO5_2	UI_LED2	<b>LED2</b> Red LED
	PIO4_10	UI_LED1	<b>LED3</b> Blue LED
SD card	-	SD slot VDD	<b>LED4</b> Yellow LED
Rigado R41Z	PCT5	RF_PCT5	<b>SW3</b> tactile switch
	PCT1	RF_PCT1	<b>LED5</b> Red LED
Sillex ULPGN-2000	GPIO[30]	WIFI_SW	<b>SW4</b> tactile switch
	GPIO[11]	WIFI_LED	<b>LED6</b> Red LED

