POLITECNICO DI TORINO

Electronics for Embedded Systems

Project of the course

Professor: Claudio Passerone

Data: November 14, 2019

Author: Alessandro Salvato

Contents

1 Introduction

I'm very glad to present by this report, my own project for Electronics for Embedded Systems course. The main idea consists in realizing a camera able to capture photos to be transferred to an host computer for further processing. Before describing in details the environmental structure, I guess it's fundamental to highlight that all the topics of the course have been covered in the following manner:

- **Memory**: memory controller as FSM
- **Programmable logic device**: acts as remote to send the user command
- **Interconnection protocol**: I2C, SCCB and UART
- **Peripheral management**: programming of camera peripheral
- **AD and DA converters**: usage of AD converter to sample the luminance of the environment
- **Power management**: Step-down converter to drive a LED

Each of them owns a proper section below. Let's have a look on the general architecture, listing the main parts of the system. The camera is connected to an ST microcontroller: a STM32F446ZET. The choice on this MCU has due to the fact that it integrates a very useful and widespread engaged peripheral in the video-capturing field: the Digital CaMera Interface (**DCMI**). All images are sent to the host computer by UART protocol, exploiting another embedded peripheral of the microcontroller. Computer reads UART's data plugging a USB-UART adapter. Then a small Python script converts those data into a BMP image. The command is generated by the user pushing a specific button. An 8 buttons keyboard is tied up to the FPGA: an Altera Cyclone IV on DE0-Nano board. The implemented VHDL is composed of the keyboard driver and a fully structural UART peripheral. When a button has clicked, an encoded 8-bit data is sent to the microcontroller, which, by an interrupt, recognizes that the user pushed something. A breadboard has been used to build circuits for AD and Bulk converters. I used STM32 CubeIDE to program the microcontroller , a software Eclipse based. To program the FPGA I exploited both Quartus II when uploading the *sof* file and Xilinx ISE Design Suite when simulating (just for my high familiarity with that). Moreover, as classical laboratory instruments I relied on an oscilloscope provided by GW Instek and a multimeter for resistor measurements. Moreover, I exploited the functionalities offered by GitHub, more specifically the desktop version that I found very easy and immediate to use.

For sure, the following image is more clarifying for the reader.

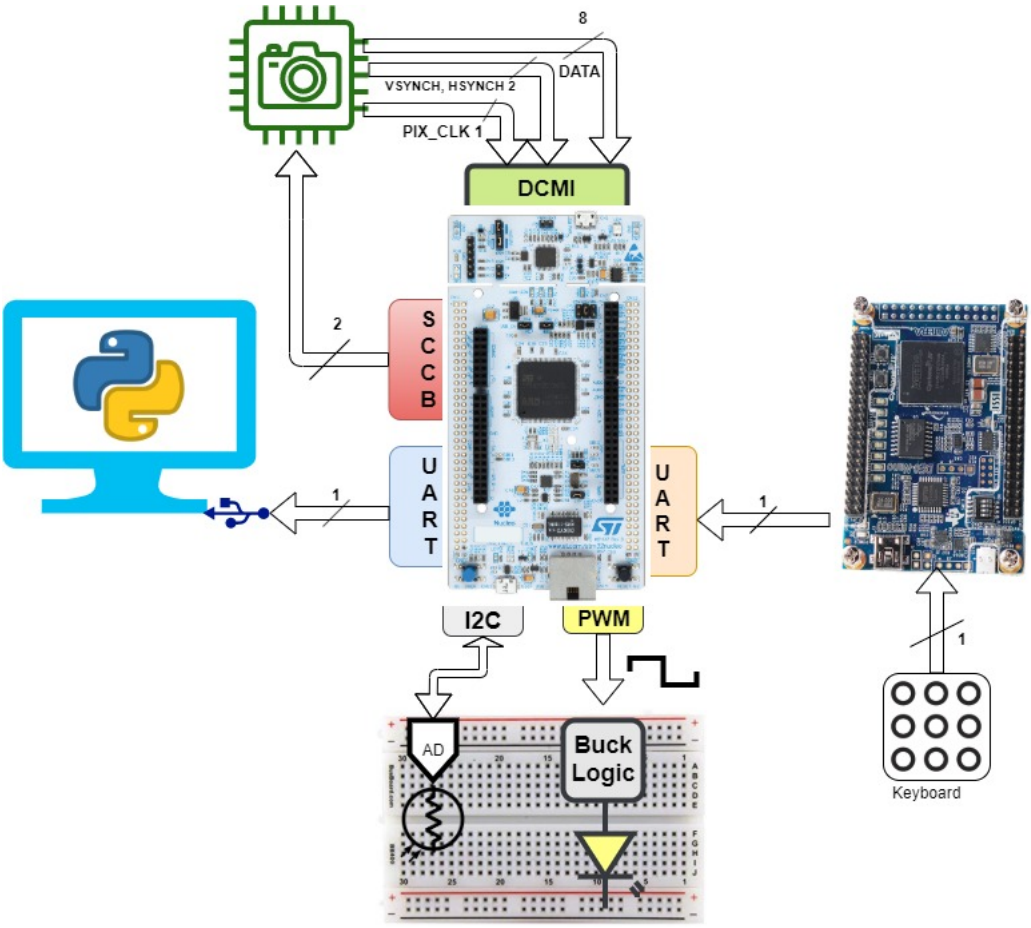

Figure 1: General block schema

Camera has been set to capture images in format QVGA (320x240) RGB 565 for an amount of 153600 bytes. However due to the poorness of memory availability, I capture the half of them. Then the Python script generates on the host a picture 640x120 without colors.

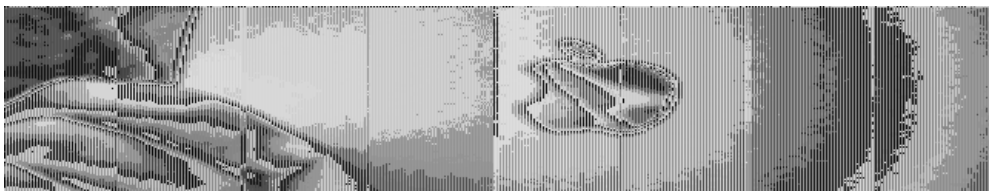

Figure 2: Apple logo captured by camera and then processed by software

Figure 3: Original Apple logo

What actually the system appears.... It's a little bit messy due to the high number of flying wires. Only the camera requires 18 links.

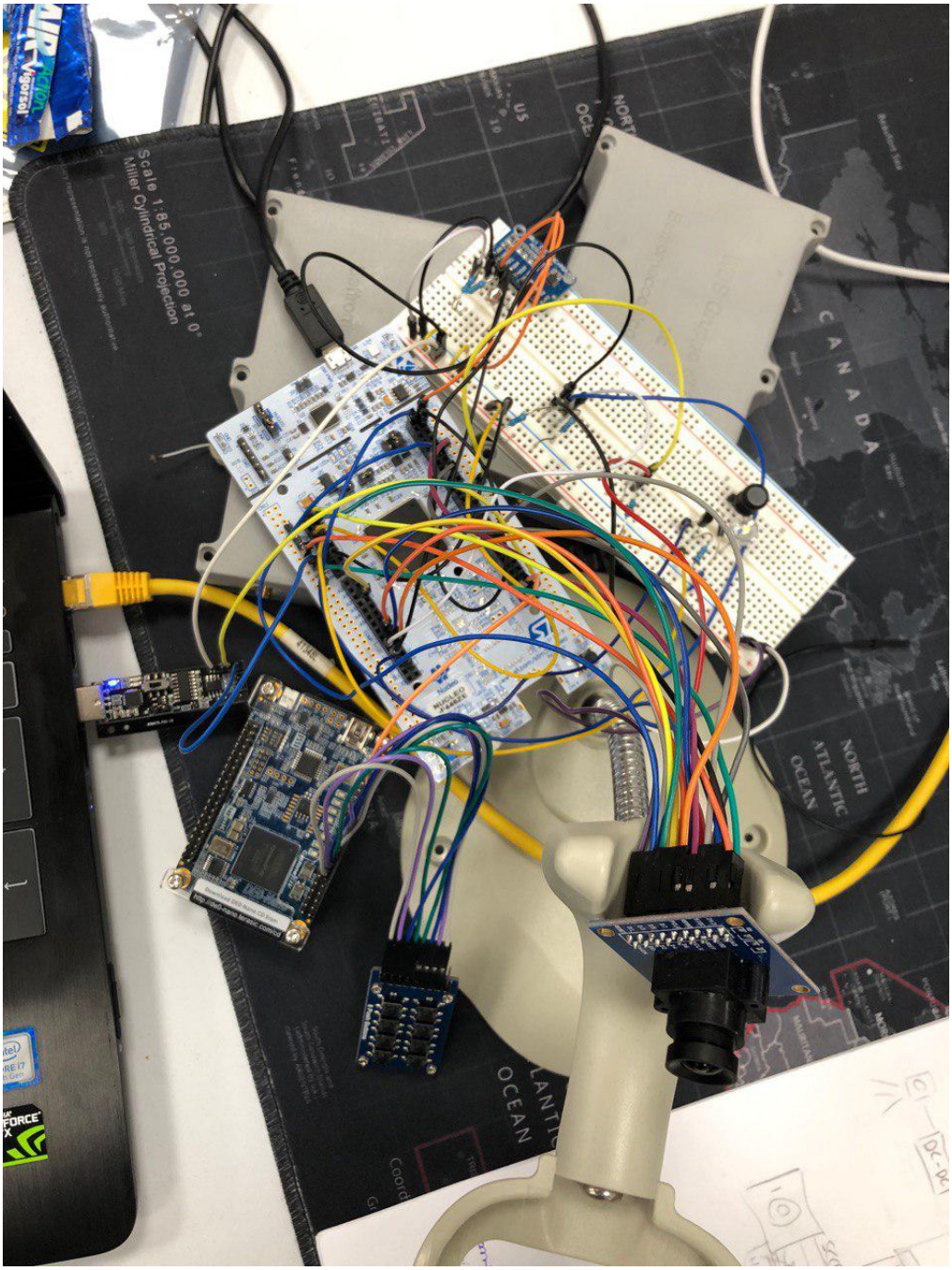

Figure 4: Real system

I'll proceed in the description in the same order I implemented each part of the project.

2 Analog to digital conversion

The analog to digital conversion aims at reading the luminance of the room, sending the sample to microcontroller. The input channel is feed by the voltage passing through a photoresistor. After having measured that voltage in condition of both full and absence of light, the dynamics has approximated at range [0.5 - 3.0] Volt.
The chip is ultra-small, low-Power, 16-Bit analog-to-digital converter with internal reference, provided by Texas Instrument at 3.50$. The microcontroller drives it by means of an I2C interface.

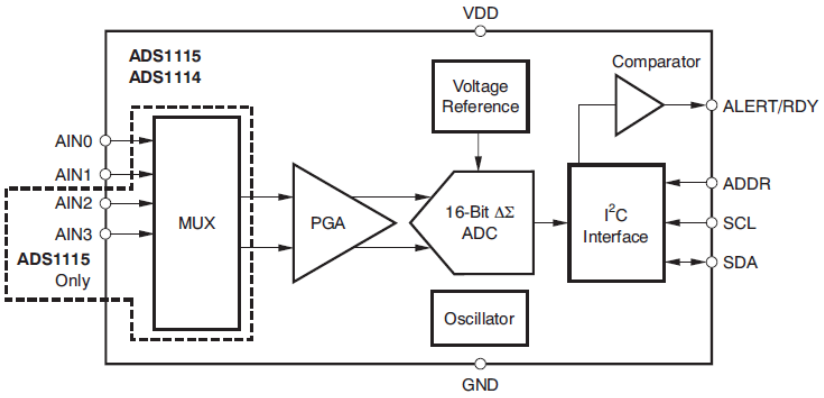

Figure 5: A2D internal architecture, from datasheet

The chip operates either in continuous conversion mode or a single-shot mode that automatically powers down after a conversion and greatly reduces current consumption during idle periods. I decided to work in the single shot one, sending the command directly from the microcontroller.

In general, to write and read 4 bytes must be sent, the string is composed of the **slave address** (0x48), the **register address**, and then in the big-endian format 2 bytes of data. Due to the fact I choose to operate in single shot mode, I have to send every time the command, so the protocol gets as follows: write 4 bytes (including the command), write the slave address and the register address containing the converted data and read 2 byte from it.

Bit(s) name	**Description**
Command	Begin conversion (automatically cleared when completed)
Input multiplexer configuration	Channel P: In0, Channel N: In1 (physically grounded on breadboard)
Programmable gain amplifier configuration	$\pm 2.048V$
Device operating mode	Power-down single shot mode
Data rate	8 sample per second
Comparator mode	not used
Comparator polarity	not used
Latching comparator	not used

Table 1: A2D configuration

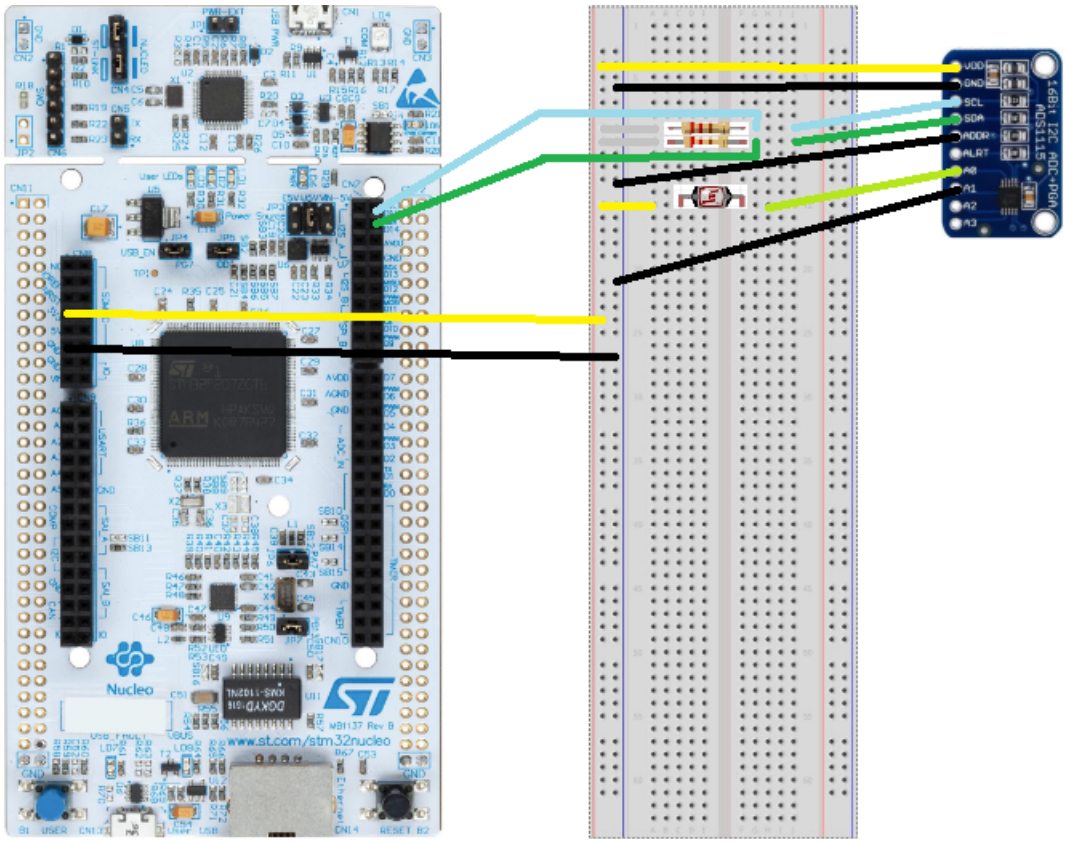

Figure 6: Connection between MCU and A2D

GPIO	Morpho Connector	Description
PB8	D15	I2C1_SCL
PB9	D14	I2C1_SDA

Table 2: A2D: GPIOs involved