**Crisan Paul Teodor - Grupa 5**

**Universitatea Tehnica din Cluj Napoca**

**18.12.2022**

**MicroBlaze**

# Introducere

Scopul proiectului este de a crea si de a programa un microprocesor MicroBlaze pentru controlul unui senzor de temperatura Pmod TMP3. Temperatura citita de microprocesor trebuie afisata atat pe ecran, cat si pe 7 segment display-ul FPGA-ului.

Protocolul de comunicare intre senzor si placa este I2C, prin urmare o provocare ridicata de aceasta tema este manipularea comunicarii [I](mailto:I@C)2C dintre master (FPGA) si slave (Pmod) in limbajul C, precum si manipularea afisorului pe 7 segmente.

In aria de solutii pentru problema proiectului gasim tehnologii precum:

* VHDL (VHSIC Hardware Description Language)
* Diligent’s Basys 3 FPGA ([Field-programmable Gate Array](https://en.wikipedia.org/wiki/Field-programmable_gate_array))
* IP (Intellectual Property)
* 7 Segment Display
* C Programming Language
* I2C Protocol
* Pmod (Peripheral Module) interface

# Fundamentare teoretica

Limbajul VHDL a fost utilizat pentru a [descrie si implementa microprocesorul MicroBlaze](https://digilent.com/reference/learn/programmable-logic/tutorials/basys-3-getting-started-with-microblaze/start) cu ajutorul IP-urilor puse la dispozitie de catre Xilinx. Design-ul a fost exportat in Vivado SDK unde vom programa microprocesorul sa indeplineasca misiunea proiectului.

Am gasit incarcate pe GitHub [o serie de librarii cu scopul realizarii comunicarii I2](https://github.com/Xilinx/embeddedsw/tree/master/XilinxProcessorIPLib/drivers/iic/src)C intre un master si un slave care au fost de mare ajutor solutiei proiectului. Pe langa IP-ul [Pmod TMP3](https://digilent.com/shop/pmod-tmp3-digital-temperature-sensor/) oferit de Xilinx, compania a pus la dispozitie si o librarie in C pentru manipularea comunicarii cu acest senzor.

Placa Basys 3 dispune de mai multe module interactive pe aceasta, precum LED-urile, butoanele, afisorul pe 7 segmente etc. Aceste module se clasifica sub numele de GPIO(General-purpose input/output), care de asemenea pot fi manipulate cu ajutorul unor librarii create de Xillinx.

Senzorlul de temperatura Pmod TMP3 comunica cu FPGA-ul prin protocolul I2C, un

protocol de comunicare sincron care permite controlarea mai multor dispozitive slave de catre un singur master. Are nevoie de doua linii de comunicare open-drain: SDA – serial data line, linia pentru trimiterea datelor; si SCL – serial clock line, linia pentru semnalul de ceas folosit pentru sincronizarea dispozitivelor slave la fracevnta masterului. Transmitera datelor este seriala, adica bit cu bit, iar initial se trimite o adresa pe care sa fie stocate/citite datele, apoi se trimite pachetul de date in sine.

Senzorul de presiune Pmod DPG1 comunica cu FPGA-ul prin protocolul SPI, un protocul de comunicare sincron folosit pentru comunicare la distante scurte, in principal pentru sisteme embedded. Ca si I2C, SPI foloseste arhitectura master-slave, si dispune de 4 linii de comunicare: SCLK – linia de clock a masterului (pentru sincronizare), MOSI – Master Out Slave In (data din master in slave), MISO – Master In Slave Out (data din slave in master), CS – chip select (un flag care indica trimiterea datelor).

# Proiectare si Implementare

Am inceput proiectul prin a ma documenta in amanunt ce inseamna MicroBlaze, acesta fiind un microprocesor general-purpose proiectat pentru FPGA-uri folosit in sisteme embedded.

Dupa ceam inteles despre ce e vorba, am trecut la treaba, primul pas fiind crearea unui proiect nou in Vivado 2017. Mi-am importat de pe GitHub fisierele necesare pentru ca Vivado sa imi detecteze placa Basys 3, fiindca fara acestea nu puteam sa realizez diagrama bloc.

Pasul doi a fost de a realiza un block diagram care sa cuprinda toate necesitatile temei. In primul rand am adaugat IP-ul MicroBlaze care face toata magia, si desigur semnalul de clock al placii. Am avut nevoie si de un mod de comunicare cu microprocesorul, urmand sa aleg USB UART. IP-urile pentru Pmod si 7 segment display au fost adaugate ulterior, momentan a trebuit sa testez ce am implementat so far. Am creat un block design wrapper pentru sistem, am generat bitstream-ul si am exportat toata magia in Xilinx SDK.

Pentru a vedea ce se intampla pe UART am folosit TeraTerm, o simpla consola care preia datele de pe USB si le afiseaza.

Initial am creat un proiect nou dupa un Hello World template predefinit de catre Xilinx.

Am conectat FPGA-ul la PC prin USB, am programat placa cu wrapper-ul exportat mai devreme, am build-uit proiectul si l-am rulat. Boom, “Hello World!” in consola de la TeraTerm.

My hopes were up, not for long.

A urmat sa incerc sa inteleg cum fac sa manipulez senzorul de temperatura din SDK. Protocolul acestuia de comunicare este I2C, pretty straight forward. Filip mi-a zis ceva de creat o librarie proprie care se ocupa de asta asa ca am inceput sa fac research. Initial am gasit o librarie care parea perfecta pentru ce aveam nevoie, doar ca era pentru Arduino. 99% din informatiile gasite erau pentru Arduino. Dupa ceva timp, am dat peste mina de aur, sau cel putin asa am crezut. Pe contul oficial Xilinx de GitHub erau postate niste librarii deosebite fix pentru protocolul I2C cu MicroBlaze. Am inclus librariile necesare in proiectul meu si am incercat sa le build-uiesc. Problema era ca acele librarii necesitau alte librarii ca sa compileze. Am inclus si acele librarii. La scurt timp m-am trezit intr-un cerc vicios de a include librarii peste librarii si de a exploata din plin Git-ul pentru si mai multe librarii. At some point aveam cred ca vreo +50 de librarii incluse in proiect si dura cam un minut sa incerce sa se build-uiasca. Pana la urma l-am scos la capat si am luat o pauza. Cand am revenit la proiect, surpriza, nu se mai build-uia din cauza altor librarii for some unknown reason, asa ca am luat-o de la capat.

Noroc cu Filip la laborator care mi-a propus sa vedem cum conectam senzorul la placa, so back to Vivado. Am avut nevoie de un IP specific senzorului Pmod TMP3 pe care nu il gaseam in lista pusa la dispozitie de Vivado, asa ca l-am cautat pe internet si l-am gasit. Am inclus IP-ul in diagrama cu iesirea la conectorul JA al placii. Am creat wrapper nou, bitstream nou, export si back to SDK.

Spre surprinderea mea, dupa ce am inclus Pmod-ul in block diagram si l-am exportat, am vazut niste librarii noi incluse in SDK si erau FIX pentru manipularea senzorului de temperatura Pmod TMP3 cu functii predefinite.

Urmatorul pas a fost de a conecta fizic senzorul la placa, unde am intampinat din nou dificultati din cauza ca au mufe diferite. Am folosit o legatura de cabluri curcubeu, conectand, 3V3 la 3V3, GND la GND, dar nu am vazut nicaieri cum ar trebui sa conectez SCL si SDA, asa ca am presupus ca se conecteaza in continuarea primelor doua cabluri.

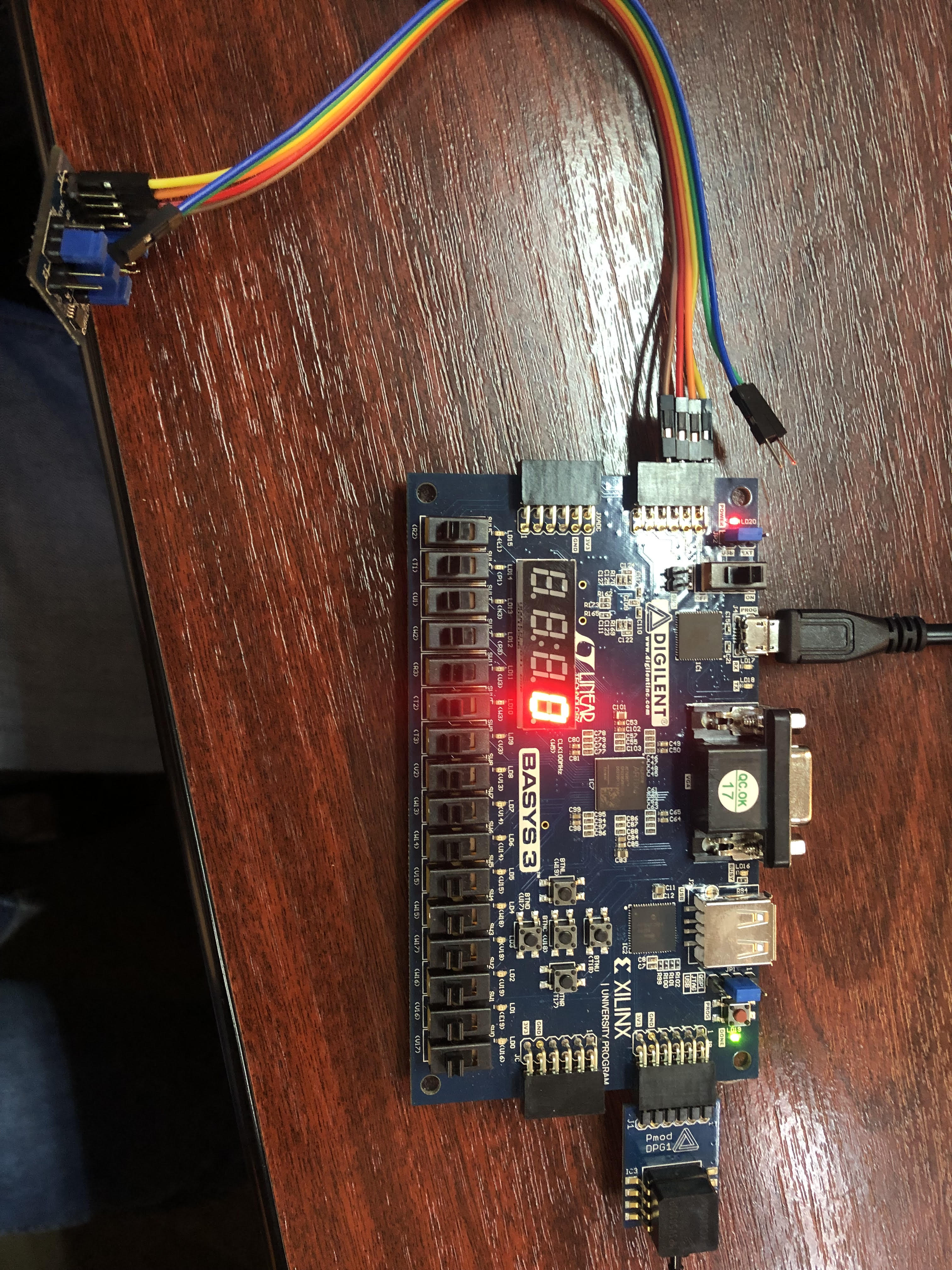
A urmat citirea propriu-zisa a temperaturii de pe senzor. Pentru ca aceasat sa fie posibila, am avut nevoie in primul rand sa setez jumperele de pe senzor in asa fal incat rezistentele pull-up sa fie pornite. In al doilea rand senzoul trebuie initializat din SDK prin instantierea unui obiect PmodTMP3 si cu ajutorul functiei TMP3\_begin care seteaza adresa I2C a senzorului(luata din block diagram) si adresa de chip a acestuia(configurata in functie de pozitia jumperelor, 0x48 by default). Configurarea senzorului este cea default. Am facut un loop infinit in care citeam in continuu temperatura cu functia TMP3\_getTemp. Senzorul s-a initializat si configurat corect, dar dintr-un motiv sau altul temperatura citita e mereu 0. Am intrat in adancurile functiilor cu debug-erul, problema e ca placa nu primeste date de la senzor, speculez ca e din cauza conexiunii fizice.

Ultimul pas a fost de a afisa pe 7 segment display informatia citita de la senzor. Am inceput prin a adauga in block diagram IP-ul pentru afisor, apoi aceeasi poveste, wrapper, bitstream si export. Din nou Xilinx a pus la dispozitie librarii pentru manipuarea GPIO, dar de data asta nu am mai cautat o luna dupa ele. IP-ul pentru afisor dispune de doua canale (anod si catod) pe care pot scrie date din SDK. Am initializat GPIO-ul si l-am setat ca si output. Apoi am creat o functie care converteste un numar intreg in codul hexa pentru afisor al numarului respectiv, iar la citirea de pe senzorul de temperatura, am convertit de asemenea rezultatul in codul pentru numarul respectiv si l-am scris pe canalul catodului.

# Concluzii

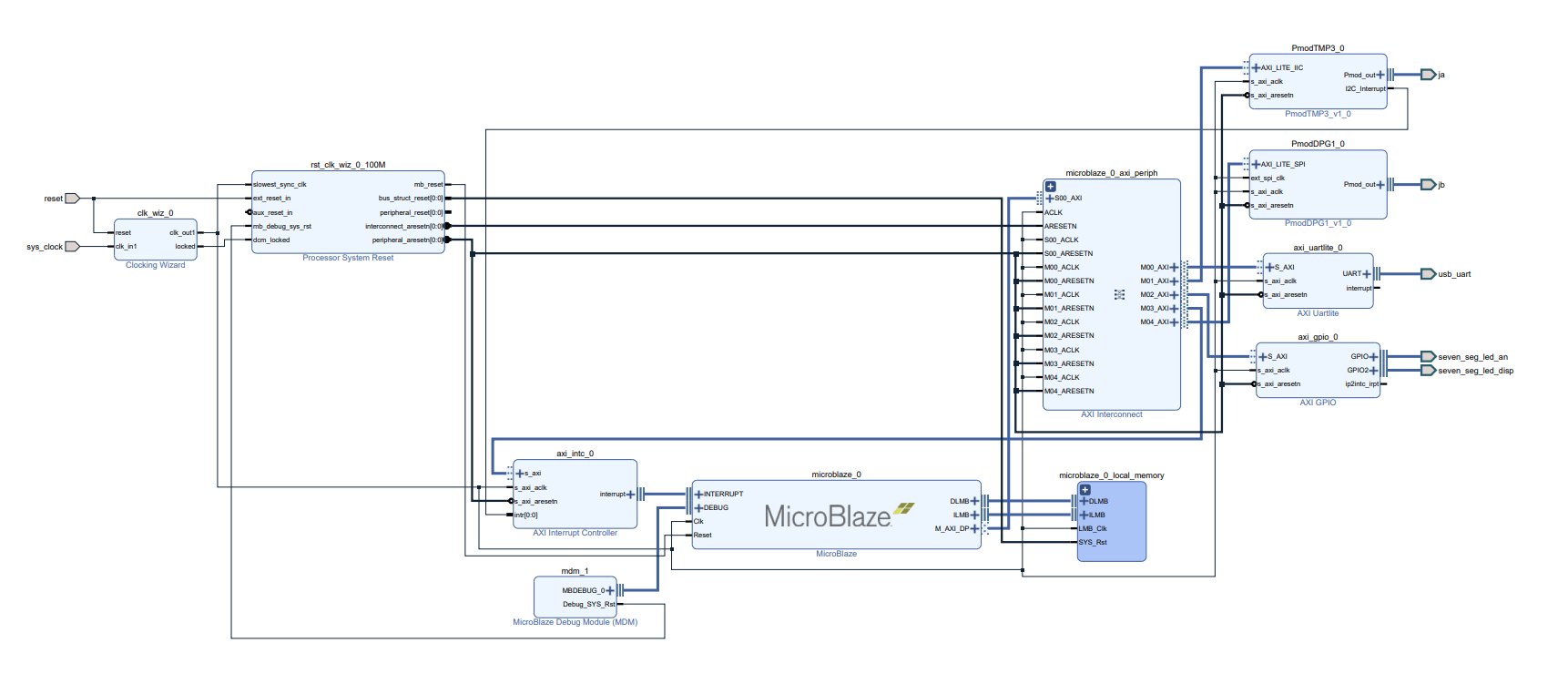
A fost printre cele mai interesante proiecte de pana acum, cel mai divers as putea spune. Am fost nevoit sa imi folosesc cam toata expertiza, si am avut norocul ca nu am prea chiulit in trecut. Invatatura de minte foarte importanta sa nu ma obsedez pe o singura problema cand mai sunt si altele de rezolvat, fiindca e foarte probabil sa fie legate intre ele.

# Anexa



Montaj:

Block diagram:



Cod:

**#include** "PmodTMP3.h"

**#include** "PmodDPG1.h"

**#include** "stdio.h"

**#include** "xil\_printf.h"

**#include** "xil\_types.h"

**#include** "xil\_cache.h"

**#include** "xparameters.h"

**#include** "xgpio.h"

**#include** "sleep.h"

**#include** "gpio/sseg.h"

**#include** "gpio/sseg\_loop.h"

XGpio gpio;

PmodTMP3 TemperatureSensor;

PmodDPG1 PressureSensor;

**#define** IIC\_ADDR 0x00020000

**#define** SPI\_ADDR 0x00030000

**#define** GPIO\_ADDR 0x40000000

**#define** GPIO\_DEVICE\_ID XPAR\_GPIO\_0\_DEVICE\_ID

**#define** GPIO\_DEVICE\_CAT XPAR\_GPIO\_1\_DEVICE\_ID

**#define** DELAY 99999999

**int** **main**(**void**) {

Xil\_ICacheEnable();

Xil\_DCacheEnable();

// initialize 7 segment display

**int** Status = XGpio\_Initialize(&gpio, GPIO\_DEVICE\_ID);

**if** (Status != XST\_SUCCESS) {

xil\_printf("Gpio Initialization Failed\r\n");

**return** XST\_FAILURE;

}

xil\_printf("Gpio Initialization Success\r\n");

XGpio\_SetDataDirection(&gpio, 1, 0x0); // set all 7-segment tristates to output

XGpio\_DiscreteWrite(&gpio, 1, 0xE); // enable the first digit

//initialize Pmod Temperature Sensor

TMP3\_begin(&TemperatureSensor, IIC\_ADDR, TMP3\_ADDR);

xil\_printf("Initialized temperature sensor\r\n");

xil\_printf("\*Address: 0x%x\r\n\n", TemperatureSensor.chipAddr & 0xff);

//initialize Pmod Pressure Sensor

DPG1\_begin(&PressureSensor, SPI\_ADDR);

xil\_printf("Initialized pressure sensor\r\n");

xil\_printf("\*Address: 0x%x\r\n\n", SPI\_ADDR & 0xff);

//reading temperature

xil\_printf("Reading data from sensors...\r\n");

**while**(1) {

**double** temp = TMP3\_getTemp(&TemperatureSensor);

**char** str1[100];

**sprintf**(str1, "%f", temp);

xil\_printf("Temperature: %s\r\n", str1); //print in console

**double** pressure = DPG1\_GetPressure(1, &PressureSensor);

**char** str2[100];

**sprintf**(str2, "%f", pressure);

xil\_printf("Pressure: %s\r\n\n", str2); //print in console

u32 tempDigit = tempToSSeg((**int**) temp); //convert to digit

XGpio\_DiscreteWrite(&gpio, 2, tempDigit); //show on 7 segment display

**for**(**int** i = 0; i < DELAY; i++);

}

Xil\_ICacheDisable();

Xil\_DCacheDisable();

**return** 0;

}

# Bibliografie

Microblaze Block Design for Basys 3: https://digilent.com/reference/learn/programmable-logic/tutorials/basys-3-getting-started-with-microblaze/start

Pmod TMP3: https://digilent.com/shop/pmod-tmp3-digital-temperature-sensor/

Pmod DPG1: <https://digilent.com/shop/pmod-dpg1-differential-pressure-gauge-sensor/>