Masurarea temperaturii cu senzorul de pe placa Nexys 4 DDR si transmiterea ei la un dispozitiv mobil

Autori: Rodanciuc Tiberiu-Gabriel, Stoian George-Iulian
Profesor indrumator: Lisman Dragos Florin
Grupa: 30237

Facultatea de Automatica
si Calculatoare

15 Ianuarie 2024

Cuprins

1 Rezumat

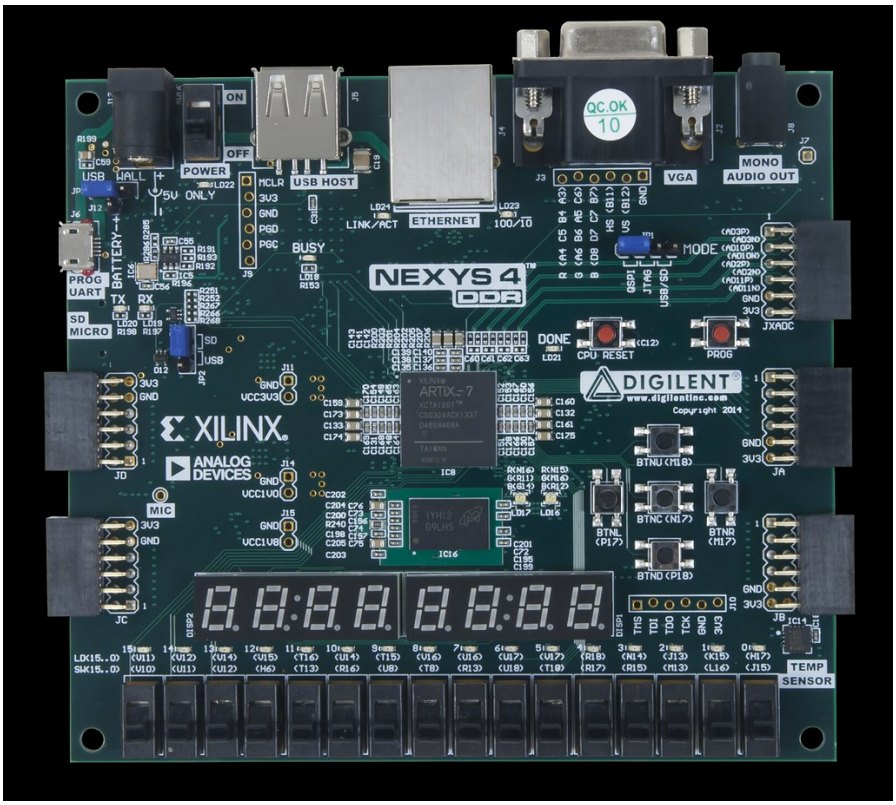

Figura 1: Placa Nexys4 DDR

În acest proiect, am utilizat placa Nexys 4 DDR pentru a măsura temperatura cu ajutorul senzorului de temperatură ADT7420. Acesta este un senzor de temperatură de înaltă acuratețe de la Analog Devices, integrat in placa Nexys, cunoscut pentru precizia și fiabilitatea sa într-o gamă largă de temperaturi. Acesta comunicață cu placa Nexys 4 DDR folosind protocolul I2C, un standard bine-cunoscut pentru comunicații seriale de mică viteză. Senzorul ADT7420 oferă o rezoluție de 16 biți, ceea ce permite detectarea schimbărilor fine de temperatură, fiind ideal pentru aplicații care necesită monitorizare precisă și constantă a temperaturii. După preluarea și procesarea datelor de temperatură, acestea au fost transmise printr-o conexiune UART către un modul Bluetooth.

PmodBT2 este un modul Bluetooth versatil, care permite plăcilor de dezvoltare, cum ar fi Nexys 4 DDR, să comunice wireless cu alte dispozitive. Acest modul utilizează protocolul UART pentru transmiterea datelor, un protocol serial care este bine-cunoscut pentru simplitatea și eficiența sa în transferul de date pe distanțe scurte. PmodBT2 nu doar că facilitează transferul de date, dar face și posibilă integrarea cu diverse dispozitive mobile, permițând aplicațiilor să primească date de temperatură în timp real.

Când un utilizator apasă un buton specific pe placa Nexys 4 DDR, declanșează un eveniment în sistemul VHDL. Acest eveniment inițiază o serie de acțiuni: în primul rând, datele de temperatură recent măsurate de senzorul ADT7420 sunt preluate. Aceste date sunt apoi procesate, convertindu-le într-un format adecvat pentru transmitere - adică, cu două zecimale pentru precizie sporită.

Odată procesate, aceste date sunt trimise prin modulul Bluetooth. În aplicația mobilă, datele primite sunt interceptate și afișate utilizatorului. Astfel, la apăsarea butonului pe placa Nexys

4 DDR, temperatura actuală - cu două zecimale pentru o precizie sporită - este afișată în timp real în aplicația mobilă.

2 Introducere

Am inceput prin a cauta site-urile potrivite pentru a ne forma o impresie despre acest proiect. Primul cod testat a fost protocolul I2C pentru transmiterea de informatii de la senzorul de temperatura la placa. Am reusit sa gasim acest cod pe site-ul Digikey care ne-a fost recomandat la laborator. La primele masuratori am adaugat si un afisor SSD pentru a afisa valorile. Formatul temperaturii era in hexazecimal ceea ce il facea putin mai greu de verificat. Apoi a urmat cautarea unui convertor pentru a putea trimite cifrele aplicatiei implementate. A urmat partea de conectare a senzorului HC05 care avea 4 pini: VCC, GND, TXD, RXT. Alimentarea am legat-o la 5V si GND la 0V, iar transmisia se face prin TXD si RXD. Acest senzor va primi informatii de la placuta si le va transmite catre aplicatia de pe Android.

Partea de cod pentru aplicatia a fost realizata in AndroidStudio cu care ne putem conecta la dispozitivul dorit si pe care se afiseaza tenperatura curenta.

In final ne-am folosit de aplicatia PUTTY pentru a verifica transmiterea de temperatura de la placa catre laptop. Dupa ce am reusit sa primit rezultatul primit, proiectul era apoape finalizat deoarece trecerea de la PUTTY la aplicatie fiind foarte asemanator. Valoarea finala a temperaturii transmise pe aplicatie a fost in decimal.

2.1 Interfata I2C

2.2 Interfata UART

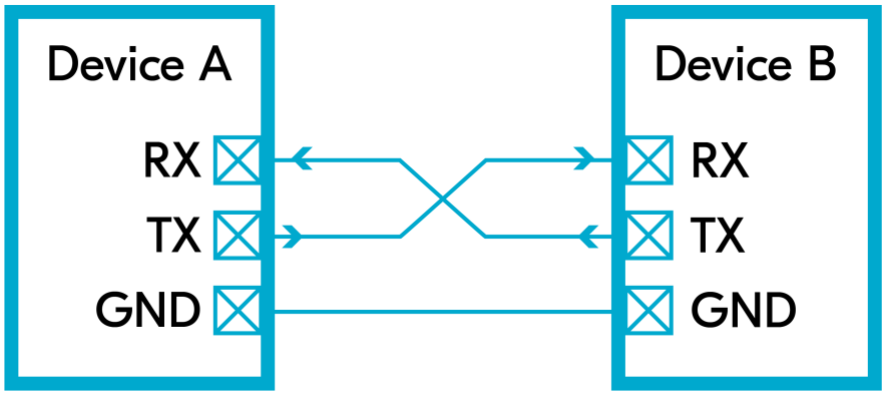

Figura 2: Interfata UART

2.3 Modulul Bluetooth Pmod BT2

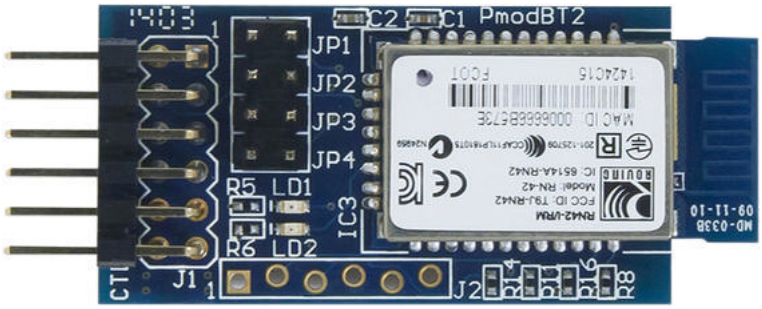

Figura 3: modulul PmodBT2

2.4 Senzorul de temperatura ADT7420

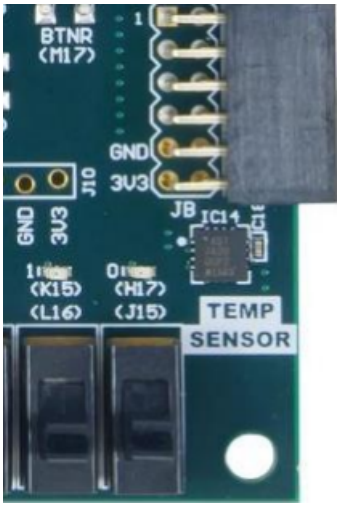

Figura 4: ADT7420

Bit	Default Value	Type	Name	Description
[1:0]	00	R/$\overline{W}$	Fault queue	These two bits set the number of undertemperature/overtemperature faults that can occur before setting the INT and CT pins. This helps to avoid false triggering due to temperature noise. 00 = 1 fault (default). 01 = 2 faults. 10 = 3 faults. 11 = 4 faults.
2	0	R/$\overline{W}$	CT pin polarity	This bit selects the output polarity of the CT pin. 0 = active low. 1 = active high.
3	0	R/$\overline{W}$	INT pin polarity	This bit selects the output polarity of the INT pin. 0 = active low. 1 = active high.
4	0	R/$\overline{W}$	INT/CT mode	This bit selects between comparator mode and interrupt mode. 0 = interrupt mode 1 = comparator mode
[6:5]	00	R/$\overline{W}$	Operation mode	These two bits set the operational mode for the ADT7420. 00 = continuous conversion (default). When one conversion is finished, the ADT7420 starts another. 01 = one shot. Conversion time is typically 240 ms. 10 = 1 SPS mode. Conversion time is typically 60 ms. This operational mode reduces the average current consumption. 11 = shutdown. All circuitry except interface circuitry is powered down.
7	0	R/$\overline{W}$	Resolution	This bit sets up the resolution of the ADC when converting. 0 = 13-bit resolution. Sign bit + 12 bits gives a temperature resolution of 0.0625°C. 1 = 16-bit resolution. Sign bit + 15 bits gives a temperature resolution of 0.0078°C.

Figura 5: Tabel configurare

3 Fundamentare teoretica

3.1 Automate cu stari finite

În hardware, un automat cu stări finite este un model conceptual utilizat pentru a proiecta logica unui circuit sau a unui sistem digital. FSM-ul este alcătuit dintr-un număr finit de stări, și schimbă aceste stări în funcție de intrările sale. Fiecare stare reprezintă o anumită configurație a sistemului sau o anumită etapă în procesarea sa.

Stările sunt diferitele moduri sau condiții în care un sistem hardware poate exista. De exemplu, într-un semafor, stările ar putea fi "Roșu", "Galben" și "Verde". În hardware, aceste stări sunt adesea reprezentate prin valori binare sau coduri care sunt stocate în registre sau memorii.

FSM-ul schimbă stările în răspuns la anumite evenimente sau intrări. De exemplu, un semafor s-ar putea schimba de la starea "Roșu" la "Verde" după un anumit interval de timp. În VHDL, aceste tranziții sunt descrise prin logica de control care evaluează intrările și starea curentă pentru a decide starea următoare.

În VHDL, un FSM este de obicei implementat folosind o combinație de procese și instrucțiuni de selecție (cum ar fi `case` sau `if-then-else`). Procesele sunt sincronizate de obicei cu un semnal de ceas pentru a asigura tranzițiile de stare la momente specifice. Fiecare posibilă stare a sistemului este descrisă în cadrul acestor instrucțiuni, împreună cu logica pentru tranziționarea între stări.

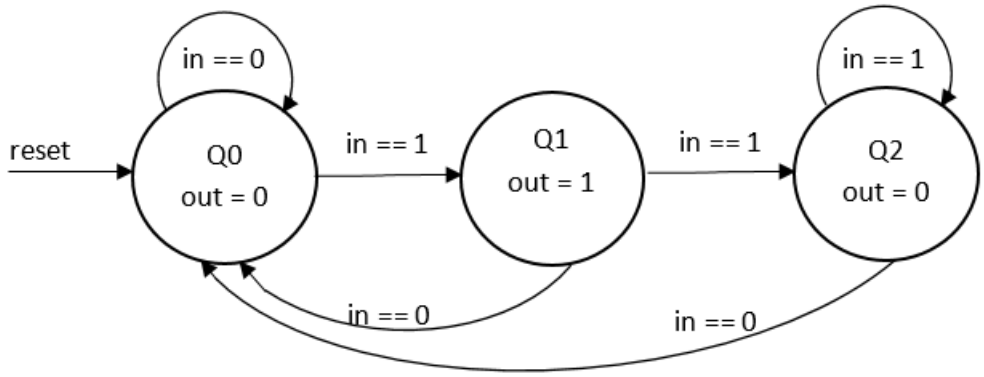

Figura 6: Exemplu FSM

3.2 Protocolul I2C

Protocolul de transmisie serială I2C (Inter-Integrated Circuit), dezvoltat inițial de Philips Semiconductors (acum NXP Semiconductors), este un standard larg răspândit pentru comunicații seriale între diferite dispozitive pe o placă de circuit. Acesta este un protocol de comunicație multi-master și multi-slave, ceea ce înseamnă că mai multe dispozitive (master) pot iniția comunicația cu unul sau mai multe dispozitive (slave) pe aceeași magistrală.

- **Două Linii de Comunicare:** I2C folosește doar două linii - SDA (Serial Data Line) pentru transmiterea datelor și SCL (Serial Clock Line) pentru sincronizarea ceasului.
- **Adresare:** Fiecare dispozitiv slave pe magistrală are o adresă unică. Masterul folosește această adresă pentru a selecta slave-ul cu care dorește să comunice.
- **Comunicare Master-Slave:** Un dispozitiv master generează semnalul de ceas și controlează accesul la magistrala I2C. El inițiază și termină sesiunile de comunicație cu slave-urile.
- **Transmisie Serială Sincronă:** Datele sunt transmise sincron cu semnalele de ceas.
- **Extensibilitate și Flexibilitate:** Magistrala poate fi extinsă pentru a include mai multe dispozitive slave, cu minimă modificare hardware.

Comunicația în protocolul I2C începe cu masterul care generează un **Start Condition**. Acesta se realizează prin tranzitarea liniei SDA de la nivel înalt la nivel jos (high-to-low) în timp ce linia SCL rămâne la nivel înalt, indicând începerea unei sesiuni de comunicație.

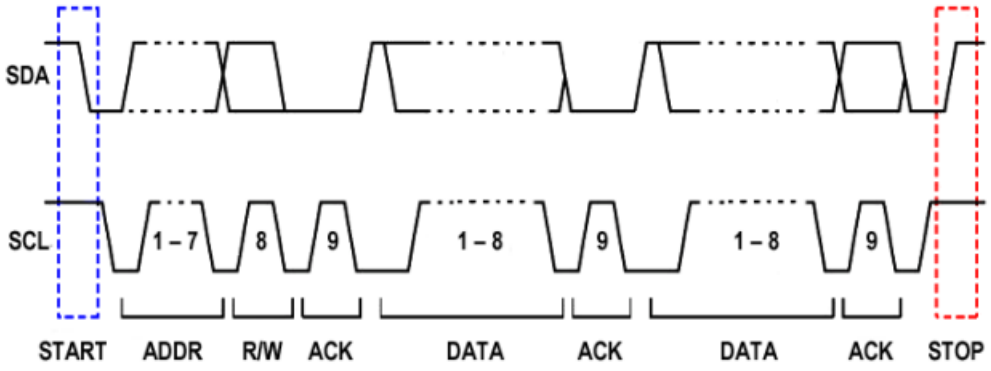

Figura 7: Modul de transmisie I2C

Masterul trimite un octet care conține adresa slave-ului dorit, urmată de un bit care indică modul de operare: citire sau scriere. Dispozitivele slave compară această adresă cu propria adresă și răspund cu un bit de recunoaștere (ACK) dacă adresa corespunde.

În modul de scriere, masterul trimite datele în octeți, cu slave-ul răspunzând cu un bit ACK după fiecare octet. În modul de citire, slave-ul trimite date către master, cu masterul generând un bit ACK după fiecare octet primit, exceptând ultimul octet, când se trimite un bit NACK. Datele sunt transmise serial pe linia SDA, sincronizate cu semnalele de ceas de pe linia SCL.

Masterul încheie sesiunea de comunicație prin generarea unei **Stop Condition**, realizată prin tranzitarea liniei SDA de la nivel jos la nivel înalt (low-to-high) în timp ce linia SCL este la nivel înalt, semnalând sfârșitul comunicației.

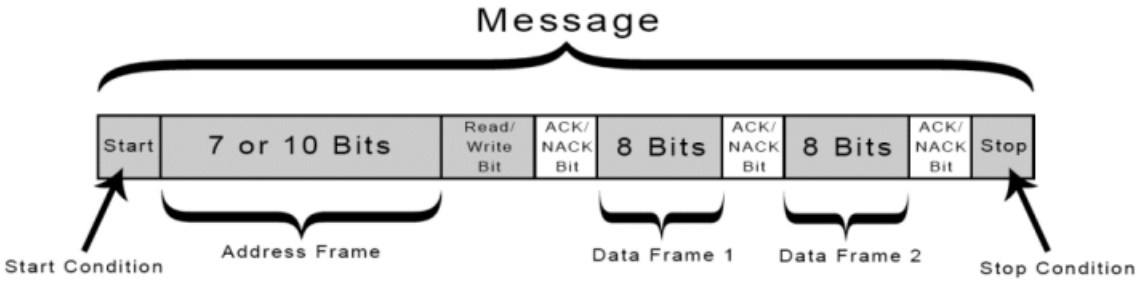

Figura 8: Pachet de date I2C

3.3 Protocolul UART

Protocolul de transmisie UART (Universal Asynchronous Receiver/Transmitter) este o formă de comunicație serială utilizată frecvent în domeniul hardware și comunicațiilor digitale.

Un cadru de date UART tipic începe cu un bit de start, urmat de 5-9 biți de date, un bit opțional de paritate, și se încheie cu 1 sau 2 biți de stop. Structura exactă a cadrelor depinde de configurația specifică a sistemului. Bitul de start este utilizat pentru a alerta receptorul că un nou cadru de date este pe cale să înceapă și pentru a sincroniza ceasul receptorului cu ceasul transmițătorului. Biții de stop semnalizează sfârșitul cadrelor de date și ajută la stabilirea unei perioade de pauză înainte de începerea transmiterii următorului cadru.

În momentul transmiterii, datele sunt trimise bit cu bit, începând cu bitul cel mai puțin semnificativ (LSB) și avansând spre bitul cel mai semnificativ (MSB). Fiecare bit este menținut pe linia de transmisie pentru o perioadă egală cu durata unui bit, care este inversă ratei de baud (de exemplu, la 9600 baud, durata unui bit este de aproximativ 104 microsecunde).

La recepție, fiecare bit este citit la mijlocul perioadei sale de timp alocată pentru a minimiza efectele oricăror variații de sincronizare între transmițător și receptor. Receptorul așteaptă un bit de start pentru a începe citirea cadrelor de date și apoi citește fiecare bit secvențial, reasamblând datele în formatul lor original.

Bitul de paritate, care este opțional, poate fi utilizat pentru a verifica integritatea datelor. Există diferite moduri de paritate (cum ar fi paritate pară, impară, sau fără paritate), iar verificarea se face comparând bitul de paritate cu datele din cadru. Dacă se detectează o eroare de paritate, acest lucru indică faptul că datele pot fi corupte.

Bitul de stop marchează sfârșitul unui cadru de date în comunicarea UART. După transmiterea biților de date (și a bitului de paritate, dacă este utilizat), bitul de stop este trimis pentru a indica finalizarea transmiterii cadrelor de date curente. În mod tipic, bitul de stop este setat la un nivel înalt (1). Dacă receptorul detectează o eroare în bitul de stop (de exemplu, dacă

bitul de stop nu este la nivel înalt așa cum este așteptat), acest lucru poate indica o eroare de transmisie, cum ar fi distorsiunea semnalului sau pierderea sincronizării.

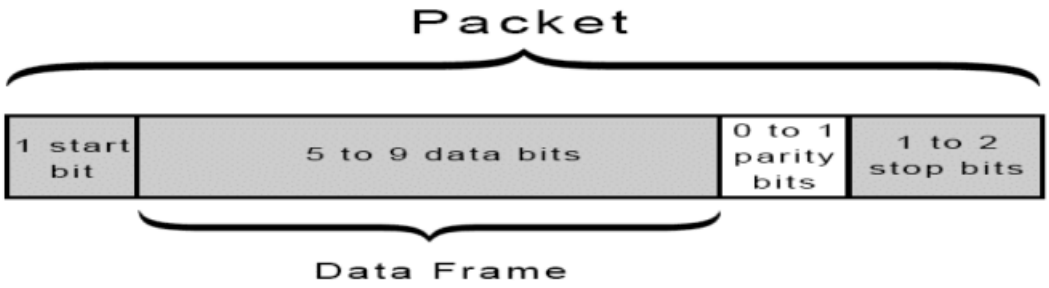

Figura 9: Pachet de date UART

3.4 Senzorul de Temperatura

Senzorul ADT7420 are un set de registre accesibile prin comunicația I2C. Fiecare registru are o adresă unică. Pentru a citi temperatura, trebuie să cunoașteți adresa specifică a registrului de temperatură. De exemplu, în unele implementări, adresa registrului de temperatură poate fi 0x00.

Datele de temperatură sunt oferite de ADT7420 într-un format de 16 biți. Aceasta înseamnă că fiecare măsurătoare a temperaturii este reprezentată printr-o valoare binară de 16 biți.

În cadrul acestor 16 biți, primele 13 biți (de la bitul cel mai semnificativ la bitul 3) reprezintă valoarea temperaturii. Restul de 3 biți sunt folosiți pentru a oferi o precizie suplimentară sau alte informații. De exemplu, pentru un senzor cu o rezoluție de 0.0625°C, fiecare increment al valorii de 13 biți corespunde unei schimbări de 0.0625°C în măsurarea temperaturii.

Valorile binare trebuie convertite într-o valoare de temperatură citibilă. Acest lucru necesită înțelegerea modului în care senzorul codifică datele de temperatură. De obicei, valoarea este exprimată în grade Celsius și poate necesita conversii suplimentare, cum ar fi aplicarea unui factor de scalare sau a unei formule matematice.

4 Proiectare si implementare

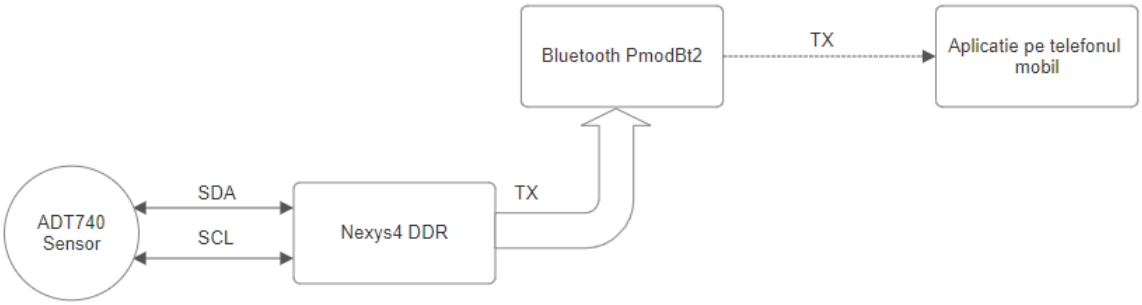

Figura 10: Diagrama simplificata a componentelor si transmisiei de date

In primul rand, trebuie sa citim temperatura de pe senzorul de pe placa. Pentru asta, folosim un modul numit **temp_sensor_adt7420**, destinat pentru interfațarea cu un senzor de temperatură ADT7420 printr-o comunicație I2C.

Funcționarea modulului este regizată de o mașină de stări finite, care navighează prin mai multe stări pentru a inițializa senzorul, a citi datele de temperatură și a trimite aceste date la ieșire. Aceste stări includ inițializarea (start), setarea rezoluției (set-resolution), o pauză scurtă (pause), citirea datelor de temperatură (read_data) și, în final, emiterea rezultatelor (output_result).

Starea 'start' este starea inițială în care FSM-ul începe după un reset sau la pornirea sistemului. În această stare, modulul așteaptă un interval de timp (100ms în exemplul dat), care permite senzorului ADT7420 să se inițializeze și să fie pregătit pentru comunicație. Acest interval este măsurat folosind un contor (counter), care crește la fiecare ciclu de ceas până când atinge valoarea dorită. După ce perioada de inițializare se încheie, FSM-ul trece la starea set_resolution.

In starea set_resolution modulul configurează rezoluția datelor de temperatură ale senzorului ADT7420. Procesul începe cu transmiterea adresei și a comenzilor de configurare prin intermediul componentei i2c_master. Aceasta include setarea registrului de configurare al senzorului pentru a folosi o rezoluție de 16 biți. FSM-ul urmărește statusul operației I2C (în principal dacă este ocupată sau nu) și, după finalizarea cu succes a configurării, trece la starea pause.

Starea 'pause' introduce o mică pauză între comenzi, necesară pentru a asigura că senzorul poate procesa comanda anterioară și este gata pentru următoarea operație. Durata pauzei este controlată de un contor, similar cu starea start.

In 'read_data' modulul începe efectiv procesul de citire a datelor de temperatură de la senzor. Inițial, se trimite adresa registrului de temperatură la care se dorește să se acceseze (de exemplu, registrul pentru valoarea MSB a temperaturii), urmat de citirea efectivă a datelor. Datele citite sunt stocate într-un buffer intern (temp_data). FSM-ul gestionează citirea atât a byte-ului MSB, cât și a celui LSB al temperaturii.

In output_result datele de temperatură sunt citite și acumulate si sunt trimise la ieșirea temperature. Aceasta înseamnă că datele de temperatură sunt acum disponibile pentru a fi utilizate de alte părți ale sistemului.

In **Convertor** modulul primește temperatura ca un vector de 16 biți (temperatura) și produce un vector de 40 de biți (asciiCodes), care reprezintă reprezentarea ASCII a temperaturii. Dupa ce convertim numarul din hexazecimal in zecimal, extragem fiecare cifra a acestuia in cifra1, cifra2, cifra3 si cifra 4. In acest moment, el are formatul "xx.xx". Semnalele codHex1, codHex2, codHex3, codHex4 reprezintă codurile ASCII ale fiecărei cifre. Fiecare cifră este convertită în codul ASCII corespunzător folosind o serie de instrucțiuni condiționale (when...else). De exemplu, dacă cifra1 este egală cu x"0" (adică 0 în format hexazecimal), codHex1 va fi setat la x"30", care este codul ASCII pentru caracterul '0'. Acest proces se repetă pentru fiecare cifră a temperaturii.

Semnalul concatenare este utilizat pentru a asambla codurile ASCII ale cifrelor în ordinea corectă, creând un șir de caractere ASCII care reprezintă valoarea temperaturii. Concatenarea include și adăugarea caracterului punct (.) pentru a separa cifrele, reprezentat de codul ASCII x"2E". În final, concatenare conține reprezentarea ASCII a temperaturii, gata de a fi trimisă către o aplicație sau afișată.

Semnalul de ieșire asciiCodes este setat la valoarea concatenare, furnizând astfel codul ASCII final al temperaturii pentru utilizare externă.

Urmatorul modul este numit **UART16** , si comunica direct cu modulul **UART**, care este proiectat pentru a transmite date ASCII prin UART (Universal Asynchronous Receiver/Transmitter) în pachete de 8 biți. Componenta esentiala a acestui modul este automatul de stari finite, care se asigura selectarea individuala a octetilor ce trebuie transmisi prin UART.

În această stare inițială, modulul se resetează și așteaptă un semnal de la Send pentru a începe transmisia. Numărătorul de octeți este setat la 7, pregătind modulul pentru a trimite șirul de 8 caractere (octeți).

Odată ce transmisia începe, modulul intră în starea trimit. Aici, începe efectiv trimiterea datelor prin selectarea octeților individuali din Data1 și plasarea lor în TxData pentru a fi transmiși. Fiecare octet este transmis unul câte unul. După fiecare transmisie, numărătorul de octeți este decrementat până când toate cele 8 octeți sunt trimiși.

După transmiterea tuturor octeților, modulul intră în starea stop, unde se resetează pentru a fi pregătit pentru următoarea secvență de transmisie.

5 Rezultate experimentale

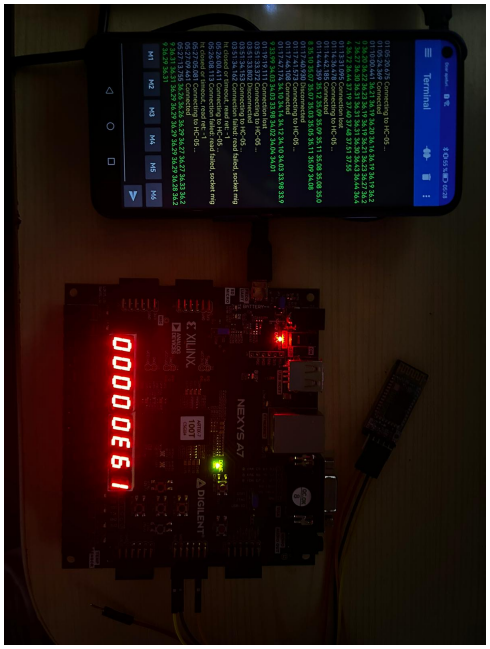

Figura 11: Rezultat experimental 1

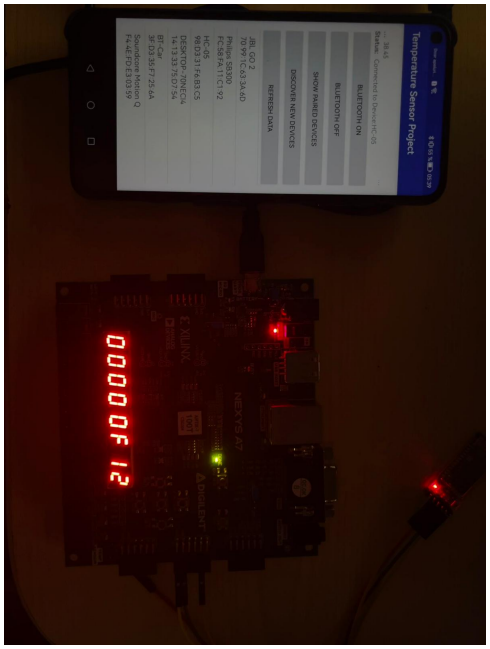

Figura 12: Rezultat experimental 2

Figura 13: Rezultat experimental 3

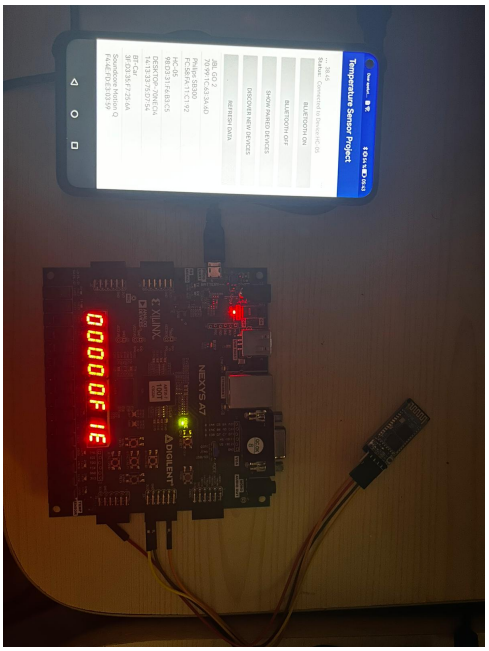

Figura 14: Rezultat experimental 4

6 Concluzii

Un aspect esențial al proiectului a fost implementarea comunicației seriale I2C si UART, o metodă eficientă de a transmite datele de temperatură către un dispozitiv mobil. În plus, o realizare importantă a fost conversia valorilor de temperatură în coduri ASCII, un proces care facilitează transmiterea și interpretarea datelor într-un mod accesibil.

Pe măsură ce proiectul nostru avansează, ne-am propus să explorăm posibilități de dezvoltare ulterioară, printre care se numără implementarea unui sistem de comunicație bidirecțională. Acest pas important ar permite utilizatorilor să solicite date de la senzor direct dintr-o aplicație

mobilă. O astfel de funcționalitate ar putea fi realizată prin dezvoltarea unei aplicații mobile cu o interfață de utilizator intuitivă, care să permită apăsarea unui buton pentru a cere datele de temperatură în timp real. Această caracteristică ar transforma experiența utilizatorului, oferindu-i un control mai mare și interactivitate sporită cu sistemul de măsurare a temperaturii.

7 Bibliografie

1. https://forum.digilent.com/topic/4543-sending-data-from-nexys-4-ddr-to-pc/
2. https://digilent.com/reference/learn/programmable-logic/tutorials/nexys-4-basic-user-demo/start
3. UART Protocol implementation on Nexys 4 DDR : r/FPGA
4. https://forum.digilent.com/topic/4543-sending-data-from-nexys-4-ddr-to-pc/
5. https://www.youtube.com/watch?v=EOfCEhWq8sg
6. https://www.youtube.com/watch?v=kMI2jy-WlGM
7. https://users.utcluj.ro/ onigaf/files/teaching/AC/AC-indrumator-laborator.pdf
8. https://forum.digikey.com/t/temperature-sensor-adt7420-pmod-controller-vhdl/20296