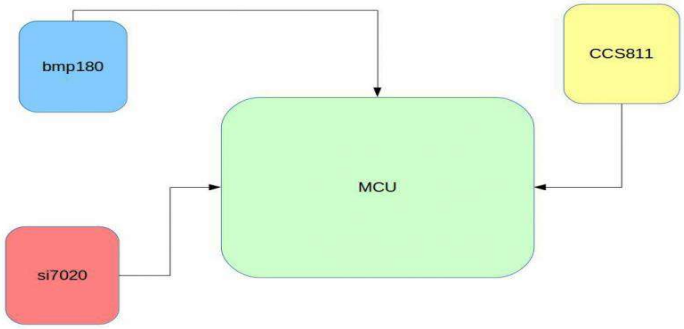
Laboratorul 09 - Comunicare I2C si SPI

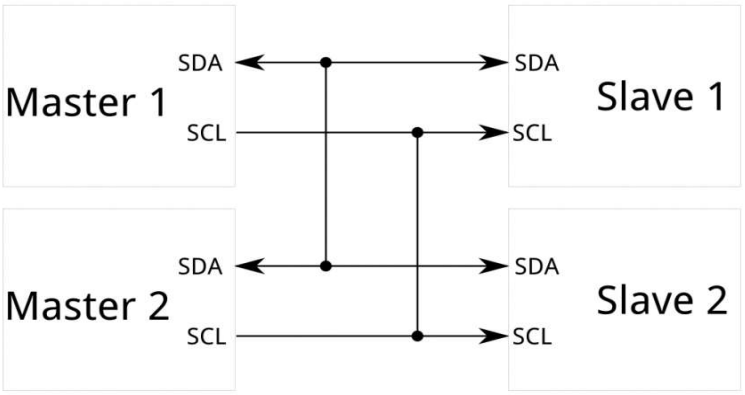
Orice sistem embedded presupune interconectarea mai multor circuite care, împreuna, sa contribuie la realizarea (de obicei) a unui singur scop. Sa luam spre exemplu un sistem a carui funcție este sa monitorizeze confortul ambiental (o stație meteo simplificata). Pentru a crea acest sistem avem nevoie de un microcontroller si mai mulți senzori: de temperatura, de umiditate, de presiune, de calitate a aerului, etc.



Microcontrollerul este „creierul” care colecteaza și proceseaza datele, iar senzorii sunt furnizorii datelor. Cum ajung datele de la senzori la microcontroller? Avem nevoie de o schema/un protocol pe care sa îl cunoasca atât uC cât și senzorii pentru a putea interschimba date. În prezent, exista numeroase astfel de standarde care descriu modalitati de transmisie de date, dar, în principiu, le putem grupa în 2 mari categorii: paralele sau seriale. Doua protocoale foarte populare ce se incadreaza în cea de-a 2 categorie sunt SPI si I2C.

I2C

Interfata I2C este un standard sincron multi-master, multi-slave pentru transmisia de date dezvoltat de Philips Semiconductor (NXP Semiconductors). Un master este componenta care initiaza transferul de date și genereaza semnalul de ceas necesar pentru sincronizare. I2C folosește doar 2 linii de comunicatie, una pentru semnalul de ceas (SCL) și alta pentru date (SDA), deci comunicatia este half-duplex, functionand cu pana la 400Kbps.

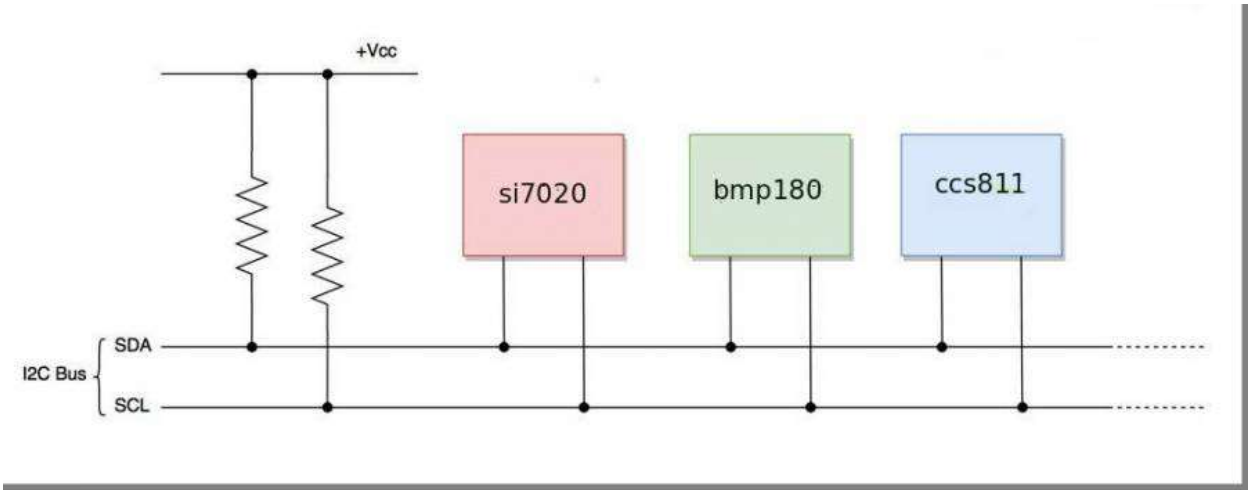


SDA și SCL sunt 2 linii bidirectionale, open-drain, conectate la o sursa pozitiva de tensiune, prin intermediul unor rezistente de pull-up.

Modul de functionare al I2C

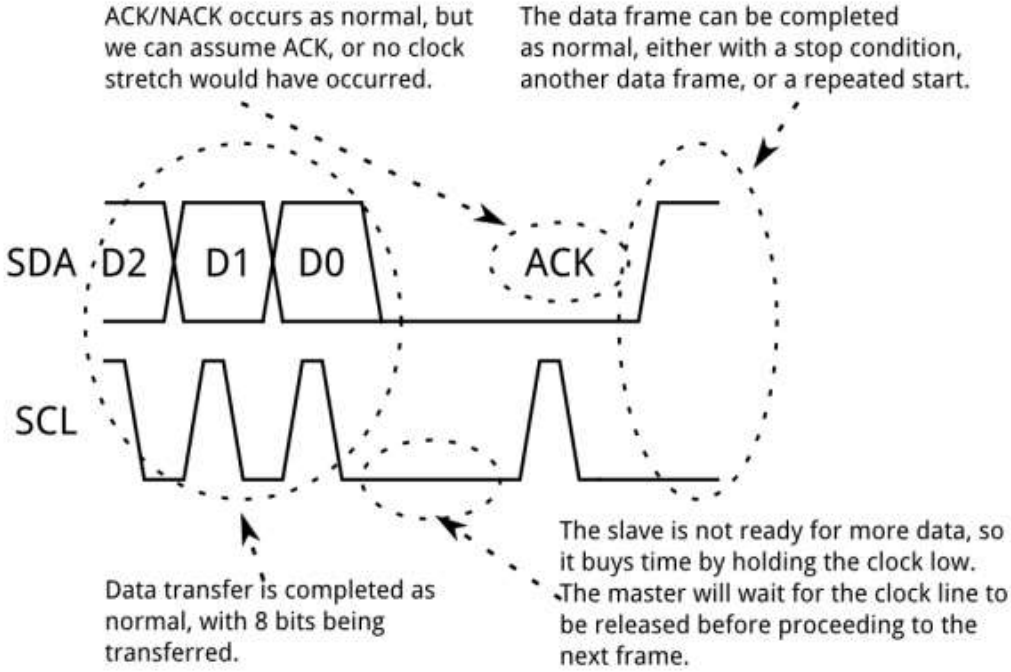
Cum începem o tranzactie de date în I2C? Când magistrala este libera, și SDA și SCL sunt în starea HIGH. Pentru a initia o tranzactie, trebuie sa trimitem conditia de START: SCL rămâne în starea HIGH, iar SDA trece din HIGH în LOW. Similar, dacă dorim sa încheiem o tranzactie, conditia de STOP corespunzatoare este reprezentata de trecerea SDA din LOW în HIGH, iar SCL rămâne HIGH. După trimiterea conditiei de START, magistrala este considerată a fi ocupata. Revenind la exemplul cu stația meteo, avem 3 senzori conectați la aceeași magistrala I2C a microcontrolerului, care are rolul de master. Dacă uC vrea sa citească date de la senzorul de presiune, bmp180, atunci va initia o tranzactie pe magistrala trimitand START.

Magistrala este aceeași pentru toți senzorii, deci cum îl alegem pe cel cu care vrem sa comunicam la un moment dat ? In I2C, fiecare „slave device” este referit printr-o adresa. Astfel, dacă avem mai multe circuite slave conectate pe magistrala, masterul alege cu care dintre ele dorește sa initieze comunicatia, precizand adresa specificata în datasheetul senzorului. Cel mai adesea, se folosesc adrese reprezentate pe 7 biti, iar cel mai nesemnificativ bit din octetul adresei ne arata daca urmeaza o operatie de scriere sau citire. După stabilirea device-ului cu care se comunica, se poate începe transimiterea datelor.



Clock stretching

Uneori "slave" device-ul are nevoie de mai mult timp pentru a trimite datele. Dupa cum am spus, masterul controleaza linia de ceas si asteapta ca slave-ul sa trimita datele, tinand cont de ciclii sai de ceas. Daca datele nu sunt gata, slave-ul trebuie sa mai "traga de timp" si face acest lucru aducand linia SCL in starea LOW, imediat dupa ce masterul a eliberat-o.



Exemplu de comunicare prin I2C

Sa consideram unul dintre senzorii din exemplul cu statia meteo: senzorul de umiditate si temperatura si7020 si sa-l conectam la MCU pentru a realiza citirea umiditatii.

Primul pas este conectarea fizica a liniilor de date si ceas la MCU, ce are rolul de master. In datasheet-ul senzorului gasim o descriere elaborata despre cum se realizeaza o masuratoare:

In figura de mai sus sunt realizate urmatoarele codificari: * S - conditia de START * P - conditia de STOP * R - Read (=1) * W - Write (=0)

Prin urmare, pentru citirea umiditatii masurate de senzor se procedeaza in felul urmator:

1. Masterul trimite conditia de START (S) 2. Masterul trimite un octet ai carui cei mai semnificativi 7 biti reprezinta adresa senzorului (0x40) (Slave Address), iar ultimul bit este 0 pentru a marca faptul ca urmeaza o operatie de scriere (W). Urmatorul bit este bitul ACK/NACK. Acest bit urmeaza intotdeauna, dupa orice frame, fie el de date sau adresa. Dupa ce adresa + bitul R/W au fost trimisi, controlul SDA-ului revine senzorului. Acesta trebuie sa aduca linia de date in starea LOW, pentru a marca ca a primit datele de la master si acestea sunt corecte (ACK) Daca SDA-ul nu este dus in starea LOW (NACK), transmisia se opreste, iar masterul decide ce se intampla in continuare. 3. Masterul trimite un octet ce reprezinta masuratoarea pe care vrea sa o citeasca de la senzor. Uitandu-ne in datasheet, in tabelul ce descrie comenzile pe care le poate primi si7020 observam ca pentru a citi umiditatea comanda corespunzatoare este: `0xE5 :

Sequence to perform a measurement and read back result (Hold Master Mode)

S	Slave Address	W	A	Measure Cmd	A	Sr	Slave Address	R	A	Clock stretch during measurement

MS Byte	A	LS Byte	NA	P

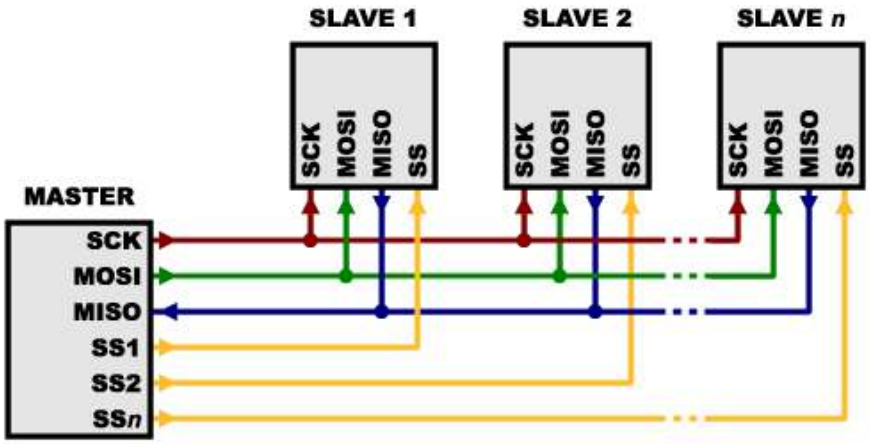
Measure Relative Humidity`. Prin urmare trimite 0xE5 si asteptam din nou ACK. (Measure Cmd) 4. Masterul trimite din nou octetul ce contine adresa slave-ului (Slave Address), de data aceasta cu bitul R/W setat pe 1, marcand faptul ca operatia dorita este de citire. 5. Senzorul aduce SCL in starea LOW, pentru a avea timp sa pregateasca datele (Clock stretch) 6. Senzorul trimite primul byte (MS) din valoarea umiditatii (este reprezentata pe 2 octeti) si asteapta ACK. 7. Senzorul trimite al 2-lea byte (LS) fara a mai fi nevoie de confirmare (NACK). 8. Masterul trimite conditia de STOP.

SPI

SPI se incadreaza tot în categoria standardelor seriale sincrone, conectand un master și mai multe device-uri „slave”. Spre deosebire de I2C nu sunt suportate mai multe dispozitive master, însă comunicatia este full-duplex. Folosește 4 legături pentru comunicare:

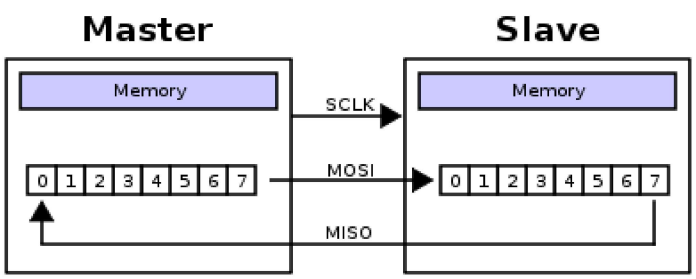
* SCK — Serial Clock (output de la master către slave) * MOSI — Master Output, Slave Input (output de la master către slave) * MISO — Master Input, Slave Output (output de la slave către master) * SS — Slave Select (activ pe 0; output de la master către slave)

Pentru fiecare dispozitiv slave este necesar câte un semnal de slave select separat, iar pentru selectarea slave-ului se pune linia SS corespunzatoare pe 0.

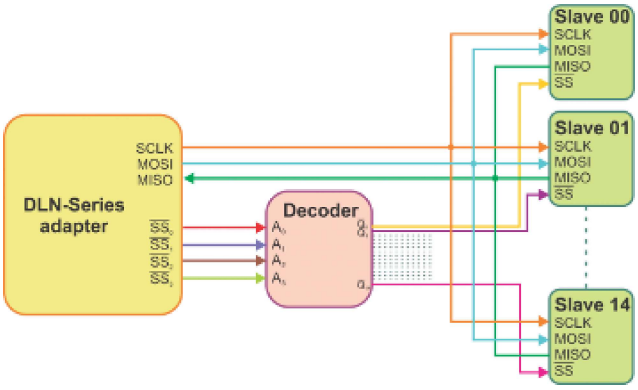


Modul de funcționare SPI

Mai întâi masterul pune pe 0 linia SS corespunzatoare slave-ului cu care dorește sa comunice. Apoi, transmisiunea fiind full-duplex, masterul va trimite biti pe linia MOSI, ce vor fi citiți de slave iar slave-ul va trimite, în același timp pe linia MISO, biti care vor fi interpretati de master. Acest lucru este posibil datorită unor registrii de shiftare, conectați circular.



Deși SPI-ul are numeroase avantaje, exista situații când avem nevoie sa conectam foarte multe dispozitive „slave”. Acest lucru se traduce în foarte multe linii de slave select. Atunci când nu avem suficiente linii disponibile, se pot folosi decodoare care sa permita referirea mai multor dispozitive slave decât linii SS disponibile.



Exerciții

- Hint: aveți nevoie de bibliotecile SparkFun-Sensors și SparkFun-IC-Microcontrollers.
- Pentru acest laborator este recomandat ca bibliotecile folosite să fie luate de pe GitHub (link [https://github.com/sparkfun/SparkFun-Eagle-Libraries]).
    1. (4p) Realizați schema si pcb-ul pentru statia meteo descrisa in acest laborator.
        - Pentru microcontroller puteti folosi ATMEGA328P
    2. (6p) Realizati schema si pcb-ul pentru un sistem cu microcontroller care are atasati mai multi senzori SPI:
        - Pentru presiune puteti folosi senzorul T5403
        - Pentru umiditate folositi senzorul BME280.
        - Conectati senzorul LSM9DS1.
        - Conectati senzorul L3G4200D.

Resurse

- https://www.nxp.com/docs/en/user-guide/UM10204.pdf [https://www.nxp.com/docs/en/user-guide/UM10204.pdf]
- https://www.silabs.com/documents/public/data-sheets/Si7020-A20.pdf [https://www.silabs.com/documents/public/data-sheets/Si7020-A20.pdf]
- https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus [https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus]

icalc/laboratoare/lab9.txt · Last modified: 2020/02/14 16:18 by ana.constantinescu