

Modulul 12

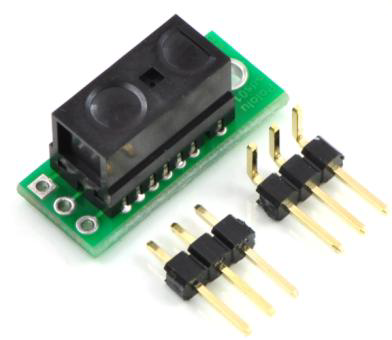
Programare: Abstractizarea HW

HAL: senzor de obstacol cu infrarosu

folosind Interrupt Service Routine (ISR)

# Obiectivele modulului

Construirea nivelului HAL: senzor de obstacol cu infrarosu



# Construirea modulului HAL: senzor de obstacol cu infrarosu

Pentru a începe construcţia modului de abstractizare a senzorului, trebuie să răspundem la 3 întrebări:

* Ce măsoară senzorul ?
* Cum comunică senzorul cu microcontroller-ul ?
* Cum interpretam semnalul primit de la senzor ?

Ce măsoară senzorul ?

Senzorul de obstacol POLOLU-1134 poate detecta obiecte la o distanţă cuprinsă între 2 cm şi 10 cm. Este un senzor cu un răspuns rapid, cu dimensiuni reduse şi cu consum minim de curent. Este excelent pentru detecţia obstacolelor datorită intervalului de detecţie. **Senzorul doar specifică dacă este sau nu un obiect în raza acestuia de detecţie, distanţa faţă de acesta nefiind precizată.**

Cum comunică senzorul cu microcontroller-ul ?

Acest senzor funcţionează cu o tensiune de alimentare cuprinsă între 2.7 V şi 6.2 V. Pe spatele plăcii cu senzorul se află şi un LED ce semnalizează prezenţa unui obstacol. Senzorul mai are şi un pin de “Enable” ce controleaza starea senzorului: activ sau inactiv. **In cazul nostru, pinul de “Enable” este conectat la Vcc, ceea ce face ca senzorul sa fie în permanență activ.**

Senzorul transmite pe un pin digital valoarea 0 logic sau 1 logic, dupa cum urmeaza:

Cand nu exista vreun obstacol, valoarea de pe pinul de iesire al senzorului este de 1 logic.

Cand se detecteaza un obstacol, pe pin apare o tranzitie de la 1 logic la 0 logic.

Cat timp obstacolul este prezent, valoarea de pe pin este de 0 logic.

Cand obstacolul dispare, pe pin apare o tranzitie de la 0 logic la 1 logic.

Cum interpretam semnalul primit de la senzor?

Pentru o eficienta si o viteza de reactie sporita a programului, se va folosi sistemul de intreruperi pentru lucrul cu acest senzor. Astfel, programul va fi instiintat imediat in momentul detectiei **prezentei** sau **absentei** obstacolului.

Daca nu am folosi sistemul de intreruperi, programul ar trebui:

* sa verifice periodic valoarea semnalului de pe pin.
* timp de executie irosit, in mare parte a timpului de functionare, cand nu exista obstacol.
* sa termine tot ce are de facut pana cand sa verifice iar valoarea semnalului de pe pin.
* reactie intarziata.

Tinand cont de aceste fapte si de forma semnalului primit, logica controlului senzorului trebuie sa jongleze cu modul de activare a **intreruperii:** la front pozitiv sau la front negativ.

Scenariu:

1. Initial, intreruperea va fi setata sa se activeze la frontul negativ (1 -> 0) al semnalului de pe pin.
2. Cand obstacolul va aparea in fata senzorului, se va genera intrerupere.
3. Pentru a sti cand obstacolul dispare din fata senzorului, trebuie asteptat frontul pozitiv (0 -> 1) al semnalului de pe pin. Asadar, intreruperea va fi setata sa se activeze la frontul pozitiv.
4. Astfel, cand obstacolul dispare, se va genera intrerupere.
5. Revino la pasul 1.

Pentru mai multe detalii legate de senzorul de obstacol cu IR, consultaţi fisa tehnică a dispozitivului :

<https://www.pololu.com/product/1134>

Impactul analizei în SW este urmatorul:

* *Modulul HAL: senzor obstacol IR* trebuie să ofere variabila de tip BOOLEAN care indica detectia obiectului.
* *Modulul HAL: senzor obstacol IR* se va folosi de modulul de intreruperi, folosind intreruperea externa INT0, care va fi generata de una dintre tranzitii: 0 -> 1 sau 1 -> 0, in functie de logica gandita.

Pentru implementarea functionalitatii se vor folosi urmatoarele functii:

|  |  |  |  |
| --- | --- | --- | --- |
| **\_INT0Interrupt** | | | |
| Funcția este una speciala, de tipul Interrupt Service Routine (ISR).  Este functia care se va executa in momentul producerii intreruperii.  Functiile de acest tip **NU** au parametrii si **NU** returneaza vreo valoare. (void) | | | |
| **Parametri** | | | |
| **Nume** | **Tip** | **Descriere** | |
| - | - | - | |
| **Valoare returnată** | | | |
| **Tip** | **Descriere** | | |
| - | - | | |
| **Reentrantă** | | | Nu |

|  |  |  |  |
| --- | --- | --- | --- |
| **INT0\_vSetPolarity** | | | |
| Functia seteaza frontul care va genera intreruperea: pozitiv sau negativ. | | | |
| **Parametri** | | | |
| **Nume** | **Tip** | **Descriere** | |
| bPolarity | BOOL | Poate lua valorile definite cu #define: NEG\_EDGE sau POS\_EDGE | |
| **Valoare returnată** | | | |
| **Tip** | **Descriere** | | |
| - | - | | |
| **Reentrantă** | | | Nu |

|  |  |  |  |
| --- | --- | --- | --- |
| **INT0\_bGetPolarity** | | | |
| Functia returneaza frontul care genereaza intreruperea. | | | |
| **Parametri** | | | |
| **Nume** | **Tip** | **Descriere** | |
| - | - | - | |
| **Valoare returnată** | | | |
| **Tip** | **Descriere** | | |
| BOOL | Frontul: NEG\_EDGE sau POS\_EDGE. | | |
| **Reentrantă** | | | Nu |

In final, modulul HAL: senzor obstacol pune la dispozitie o variabila de tip BOOLEAN care va indica prezenta/absenta obstacolului.

Aceasta variabila va fi folosita in logica de la nivelul APLICATIE, pentru constructia functionalitatii finale.

