Arhitectură și design pentru dezvoltarea aplicației client în Python

# Mediu de dezvoltare

Pentru dezvoltarea unei aplicații în Python se recomandă folosirea mediului de dezvoltare PyCharm (Community este suficient). Mediul de dezvoltare standard este IDLE dar acesta are funcționalități foarte limitate de editare a codului.

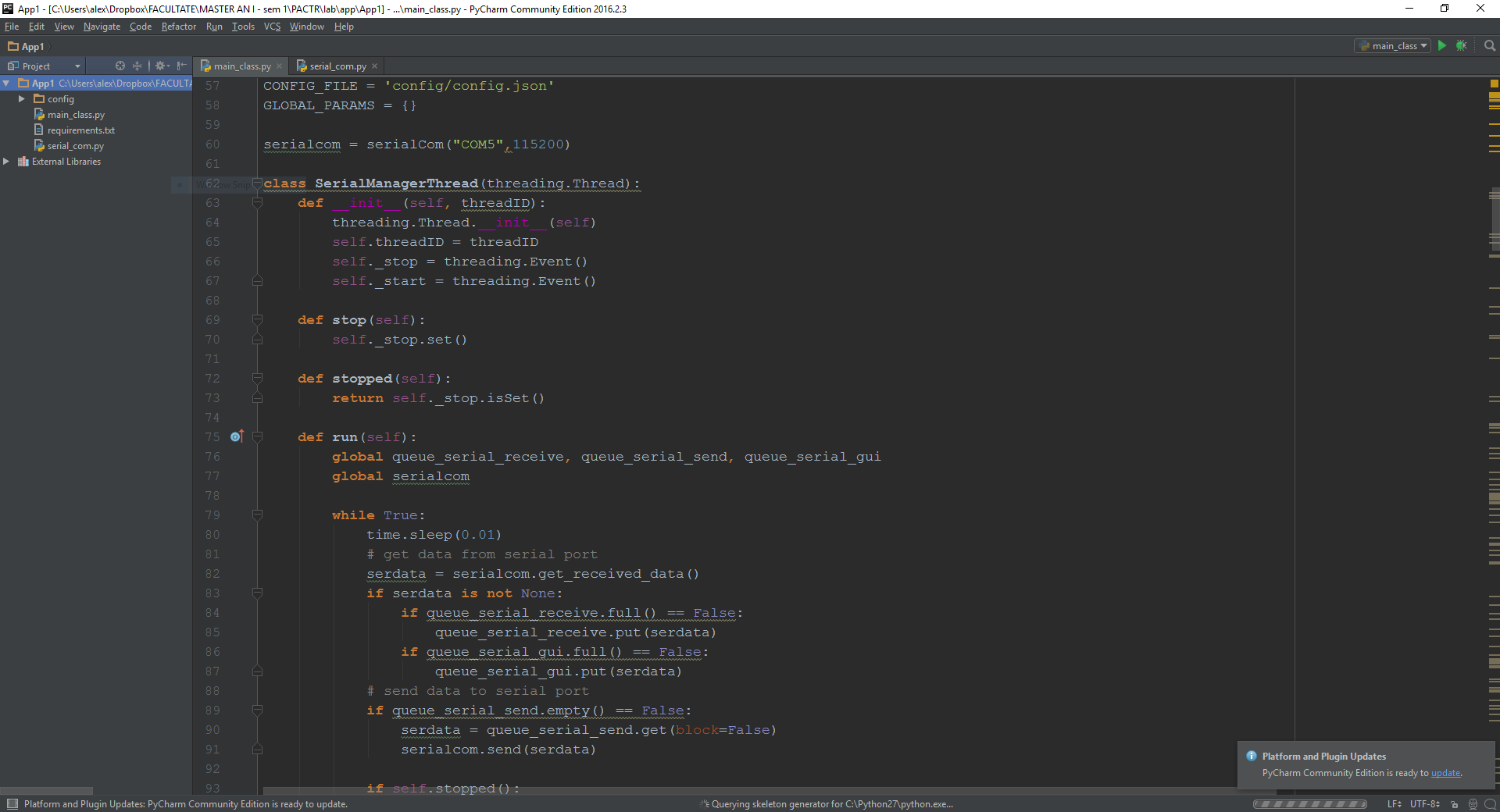


Figura 1 Dezvoltarea aplicației cu PyCharm

Se recomandă folosirea Python 2.7 din motive de compatibilitate cu bibliotecile utilizate.

# Module utilizate

În această secțiune sunt prezentate pes curt principalele module necesare pentru implementarea aplicației în Python

## Tkinter

Tkinter[[1]](#footnote-1) este un modul standard pentru Python în care se poate realiza interfața grafică a aplicației.

## Module adiționale

* threading: permite implementarea unei aplicații multi-thread
* json: permite folosirea formatului JSON pentru fișierele de configurare
* Queue: este utilizat pentru comunicația dintre firele de execuție ale aplicației folosind mesaje
* serial: modul pentru comunicație serială

## Comunicație serială

Modulul de comunicație serială este realizat într-un fișier separat

# Arhitectura soluției

În această secțiune este prezentată arhitectura de bază a aplicației

Funcționalitatea de bază a aplicației este reprezentată de comunicația prin bluetooth (serială bidirecțională) cu aplicația centrală. Aceste date sunt apoi utlizate în firul de execuție ce gestionează secvența de control și sunt afișate în interfața cu utilizatorul.

## Comunicație serială

Pentru comunicația serială am considerat 2 fire de execuție sub formă de clase (vezi implementare Python):

* SerialReadThread: are la bază o buclă infinită în care se așteaptă primirea datelor de la aplicația centrală (folosind adaptorul bluetooth care este emulat ca un port serial – COM). Datele primite sunt salvate într-o coadă de mesaje (read\_queue)
* SerialWriteThread: are la bază o buclă infinită în care se așteaptă primirea datelor de la aplicația client (aplicația descrisă aici). Astfel, atunci când se primesc date în coada de mesaje pentru scriere (write\_queue), aceste date sunt trimise la aplicația centrală prin bluetooth (pe portul serial)

## Interfața cu utilizatorul (GUI)

Aici se definesc elementele de interacțiune cu utilizatorul: butoane, text box, meniuri. De asemenea, pentru o interfață fluidă se recomandă folosirea firelor de execuție separate pentru operații consumatoare de timp (în caz contrar, aplicația se va bloca și nu va putea răspunde la comenzi). Tkinter nu permite însă actualizarea elementelor din interfață din fire de execuție separate. Din această cauză am folosit cozile de mesaje care sunt citite de bucla principală a interfeței.

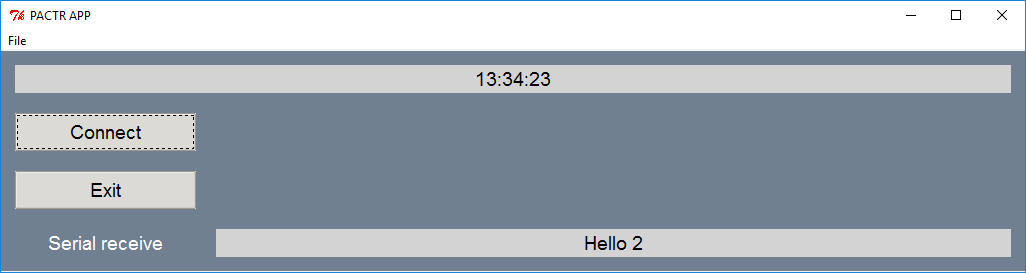


Figura 2 Schelet aplicație (GUI)

În Figura 2 aplicația afișează ora curentă și datele primite de la aplicația centrală. Pentru exemplificare am folosit un Arduino UNO care returnează un mesaj atunci când primește o anumită valoare pe interfața serială.

## Thread de gestionare a datelor pentru comunicația serială

Acest fir de execuție verifică dacă s-au primit date în coada de mesaje a portului serial și pune aceste date în cozile de mesaje corespunzătoare firului de execuție principal și ale interfeței grafice.

De asemenea se verifică dacă se dorește trimiterea datelor pe portul serial folosind coada de mesaje pentru scriere (queue\_serial\_send).

Citirea și scrierea datelor pe portul serial se realizează prin funcțiile implementate în modului de comunicație serială prezentat anterior.

## Thread de control

Aici se definește bucla de control principală a aplicației. Pentru exemplificare, se trimit periodic date la aplicația centrală. Aceasta reprezintă o cerere de date, iar aplicația centrală va trebui să returneze datele cerute.

Astfel, din punct de vedere al comunicației este realizată o arhitectură de tip client-server.

1. <https://wiki.python.org/moin/TkInter> [↑](#footnote-ref-1)