Architecture logicielle pour le contrôle d'exosquelettes d'assistance à la marche

Romain Baud (romain.baud@epfl.ch)

Groupe REHAssist, Biorob (rehassist.epfl.ch)

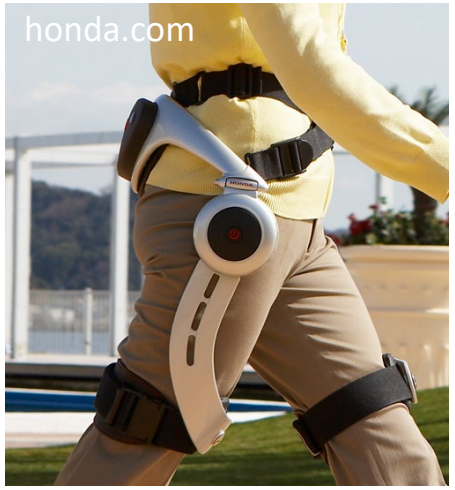
Exosquelettes, pourquoi faire ?

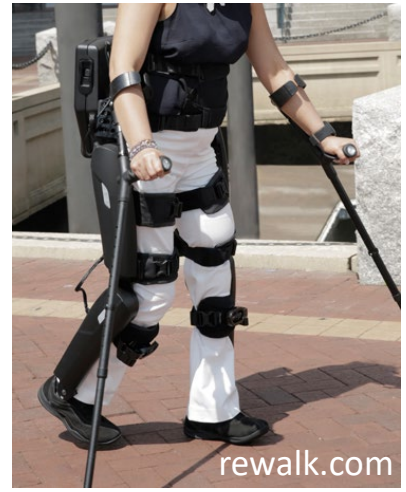
Exosquelettes pour membres inférieurs

- Utilisateurs-cible : difficultés pour la marche.
  - Personnes âgées : **faible endurance**, difficultés pour les escaliers.
  - AVC avec séquelles : perte de force et coordination, souvent d'**un seul côté**.
  - Myopathes : **perte partielle de force** et de coordination.
  - Paraplégiques : **perte totale de motricité** et sensation dans les jambes.

=> personnalisation requise pour être efficace.

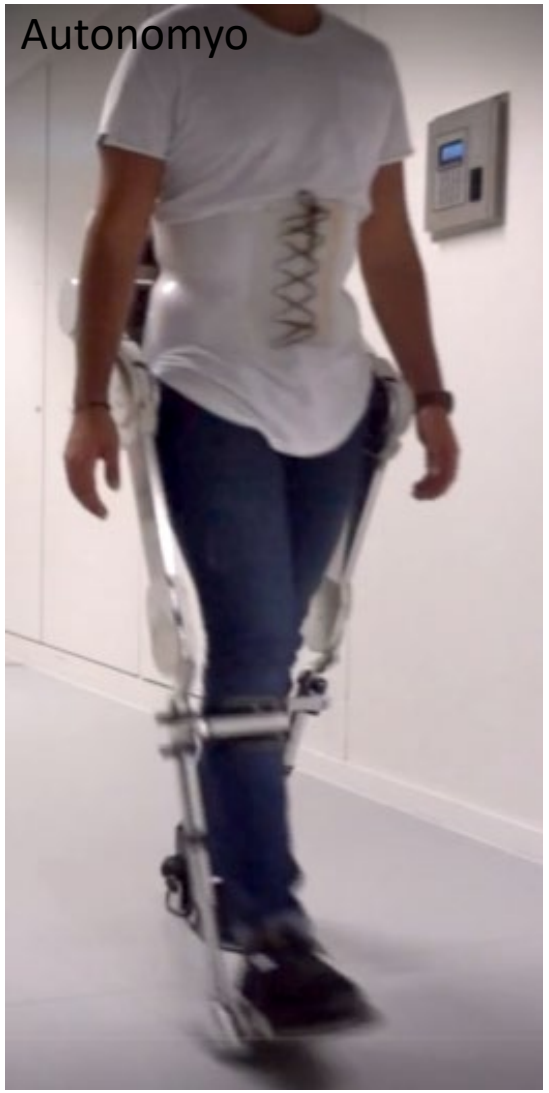
- Sujet actif de recherche !





Vidéo disponible sur :
https://actu.epfl.ch/news/un-pas-vers-l-independance-avec-l-exosquelette-t-3/ (Lien «Dossier de presse»)
https://drive.google.com/drive/folders/1A2kcd8kRn41joGe4qPWXaPmbxmuVIlzx

Exosquelettes du groupe REHAssist



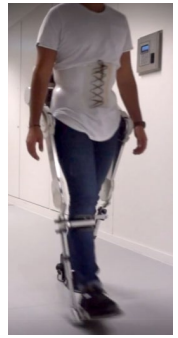
Architecture de commande

Modules

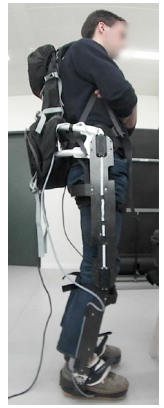
- TWIICE :
  - 4x moteur+réducteur+encodeur
  - boutons sur les béquilles
  - montre connectée
  - HUD dans les lunette AR (optionnel)

- HiBSO :
  - 2x moteur+réducteur+encodeur avec capteur de couple
  - capteurs de pression sous les pieds
  - cardiofréquencemètre

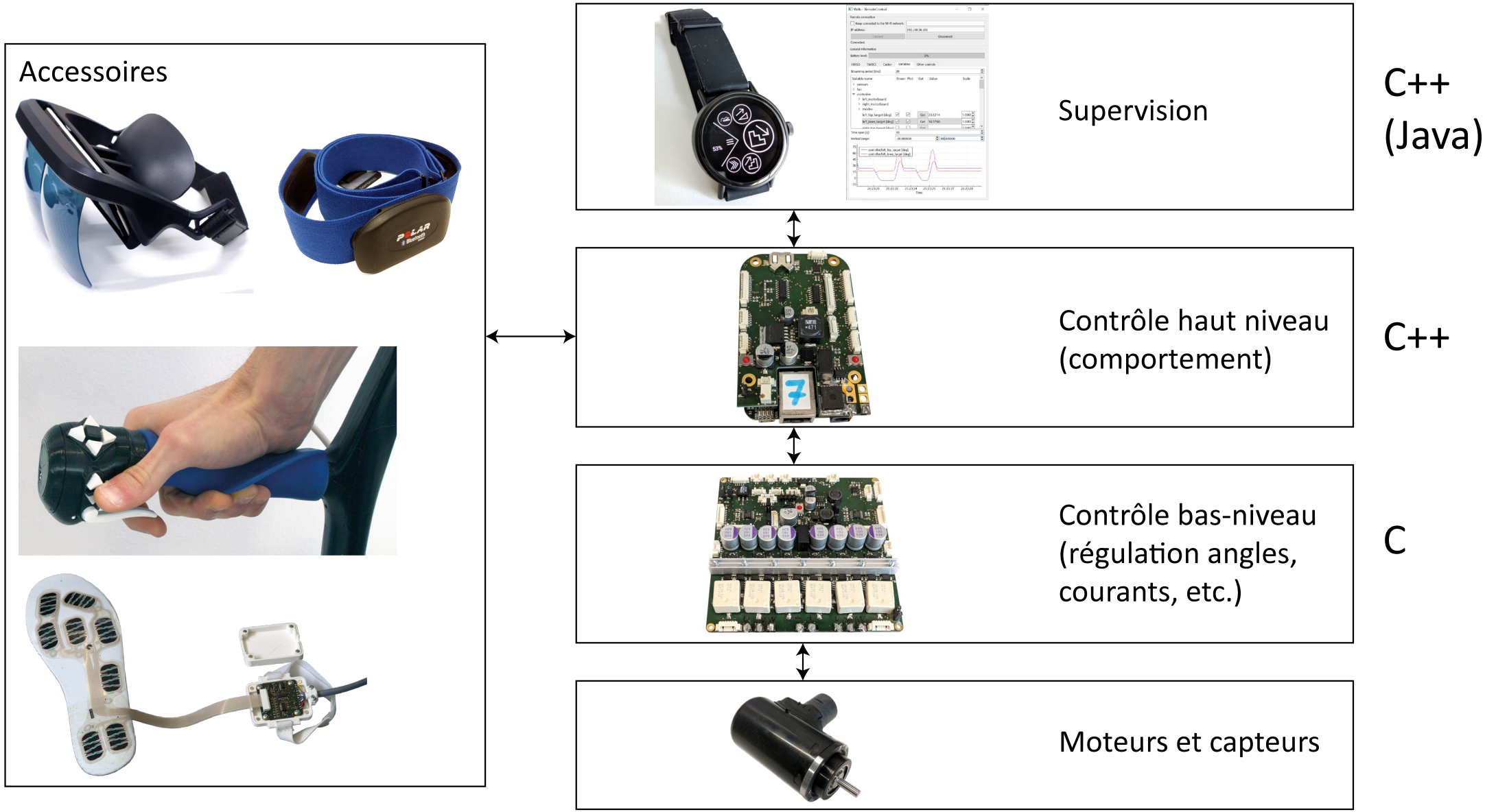
- Autonomyo :
  - 6x moteur+réducteur+encodeur
  - cellules de charge sous les pieds



- INSPIIRE :
  - 4x encodeurs
  - cellules de charge sous les pieds



Architecture matérielle modulaire



Pourquoi C++ ?

- Besoin de performance élevée.
- Besoin de structurer le code (plus de 15000 lignes de code!).
- Utilisable partout (PC, smartphone, montres connectées, etc.).
- Beaucoup de librairies disponibles.

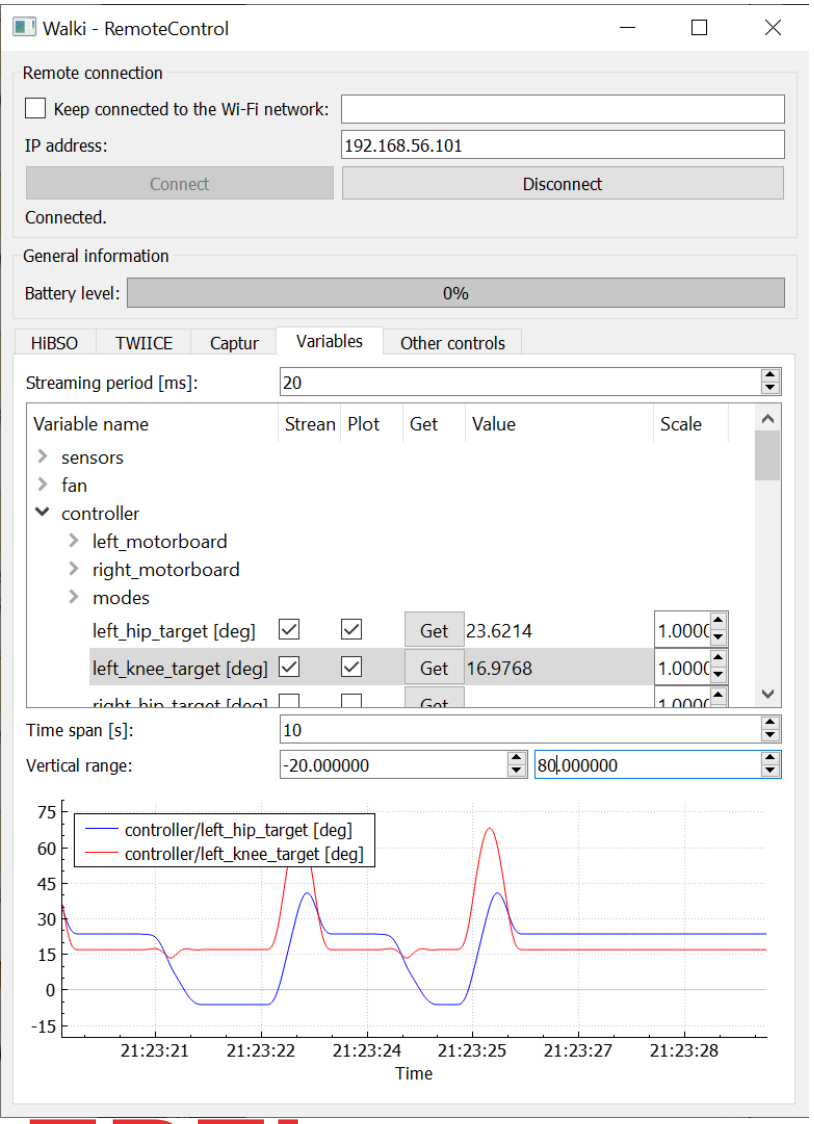
Importance d'une architecture logicielle modulaire

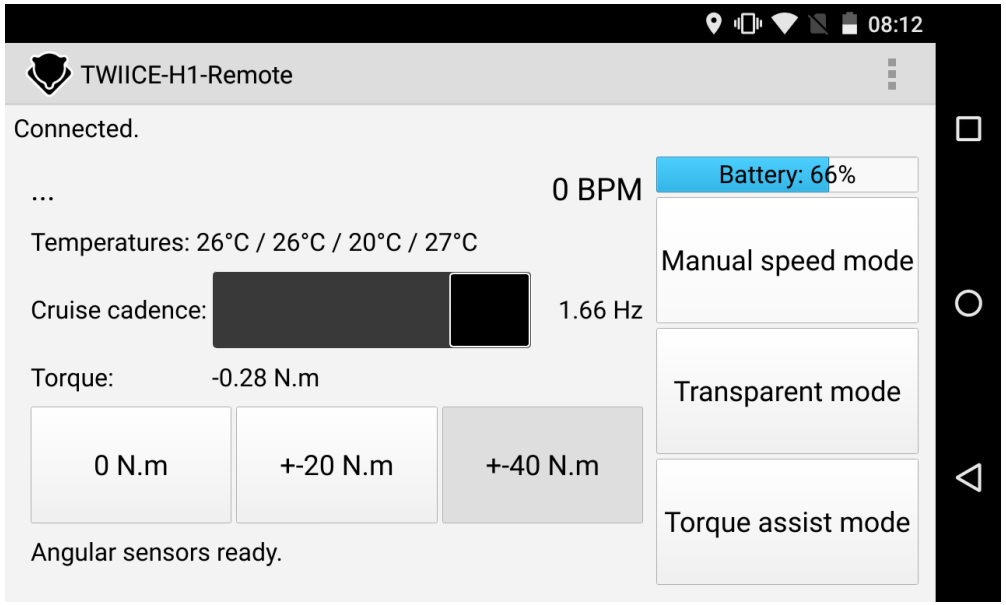
- Support de plusieurs exosquelettes :
  - doivent partager le même code.
  - tous différents.
  - constitués de différents modules communs (moteurs, capteurs, etc.).

- Développement facilité (beaucoup de code à gérer).

- Besoin de fiabilité :
  - un bug peut entraîner des blessures.
  - le temps des sessions de test avec un participant est précieux.

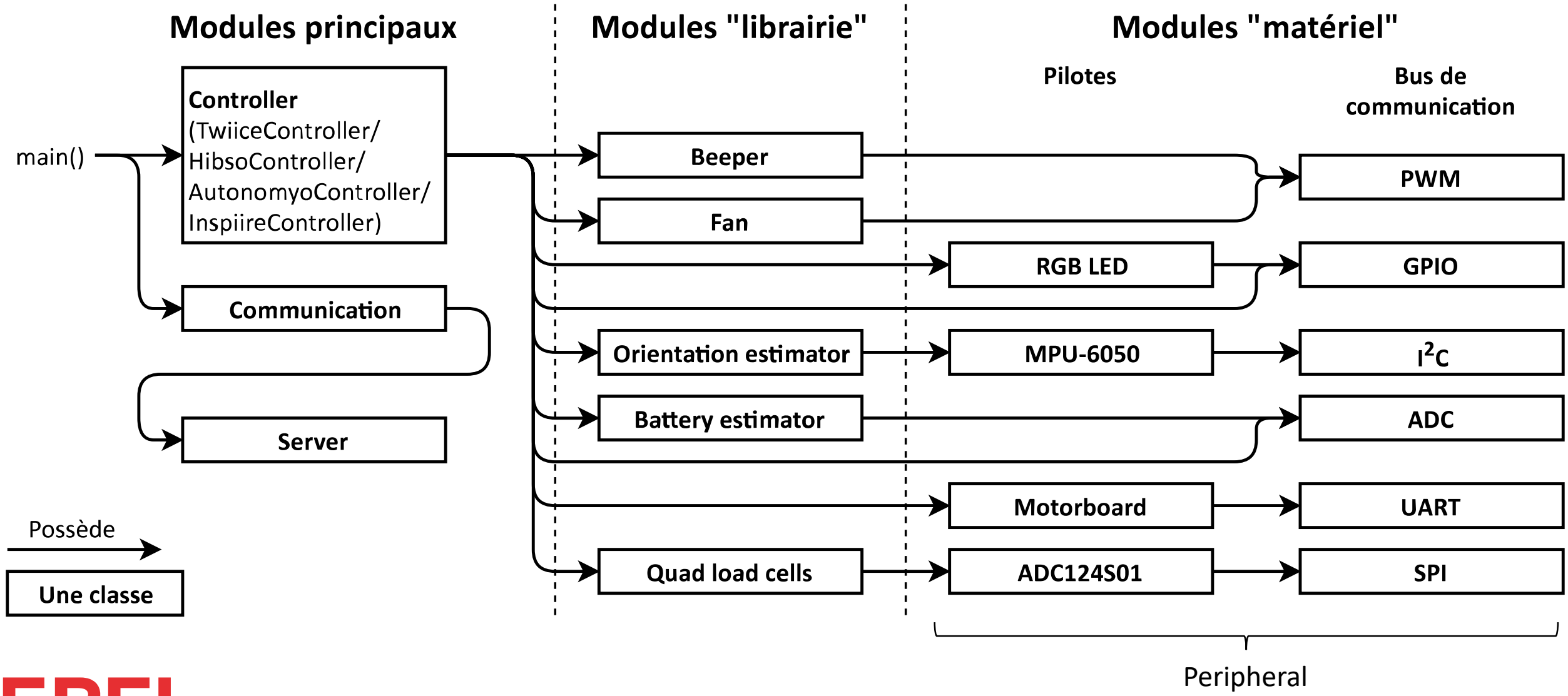
Logiciels de supervision - aperçu



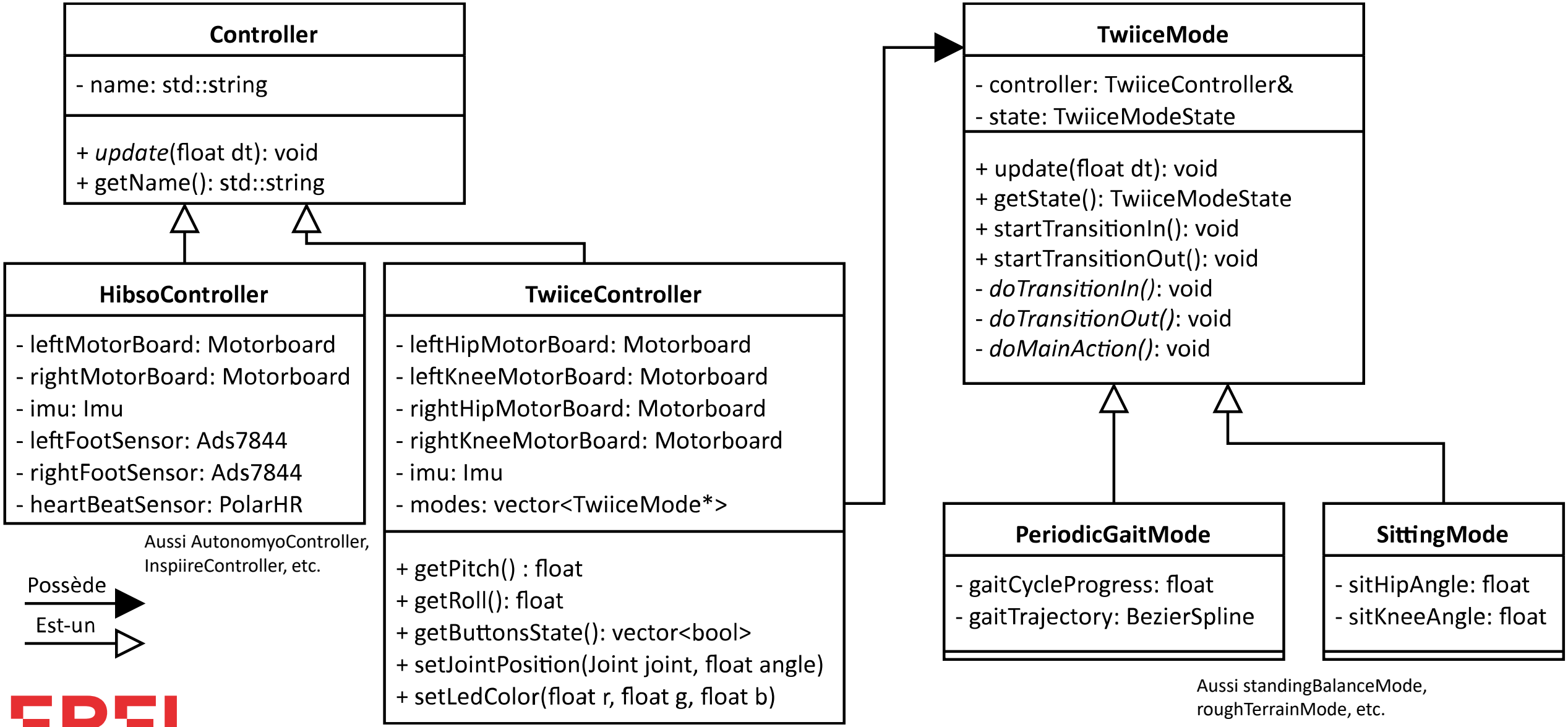


- Base commune (communication avec l'exo).
- Interfaces graphiques spécialisées.

Programme embarqué – modules principaux



Programme embarqué – Classes mères

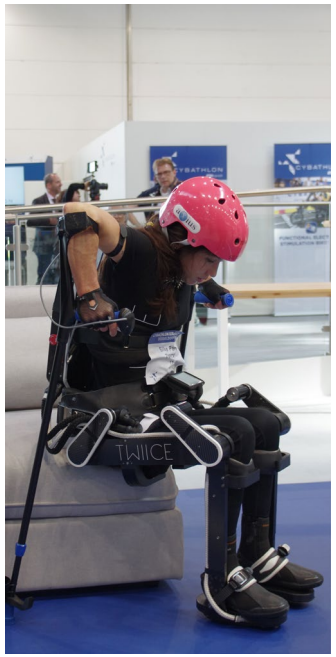


Programme embarqué - Peripheral

- Peripheral (classe mère)
  - attributs:
    - statut (DISABLED / ACTIVE / CALIBRATING / FAULT).
  - méthode:
    - virtual void update(float dt).

- Exemples de méthodes des classe dérivées :
  - get()
    - `const bool& get()` pour une entrée digitale.
    - `const Vec4f& get()` pour un capteur 4 canaux.
    - `const Vec3f& getAcceleration()` et `const Vec3f& getAngularSpeed()` pour un IMU.
  - `setIntensity()` (vibreur), `setTorque()` (moteur), `setColor()` (LED), `writeRegister()` (bus).

Programme embarqué - TwiiceMode

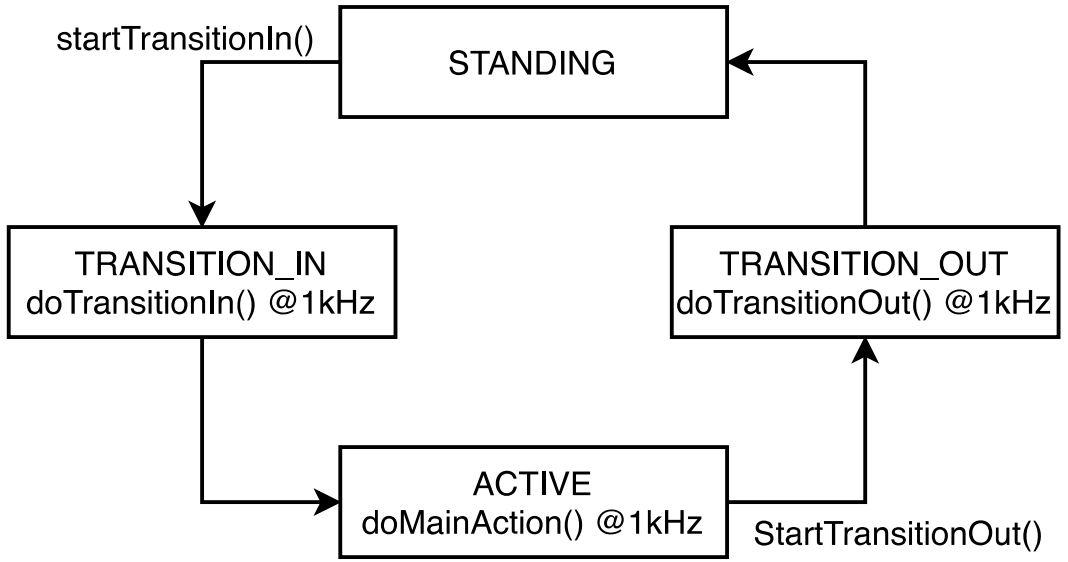

Fauteuil


Embûches


Rampe


Escaliers

TwiiceMode::update():



Conclusion

- Un projet de programmation n’est jamais fini !
  - Réfléchir à l’architecture AVANT de commencer à coder.
  - Soigner l’architecture pour une possible extension.
  - Pensez à ceux qui reprendront le projet après vous !