平成19年10月4日
電話会議

インテリジェント手術機器研究開発プロジェクト

コア制御ソフトウェア開発

現状と本年度実施計画

名古屋工業大学

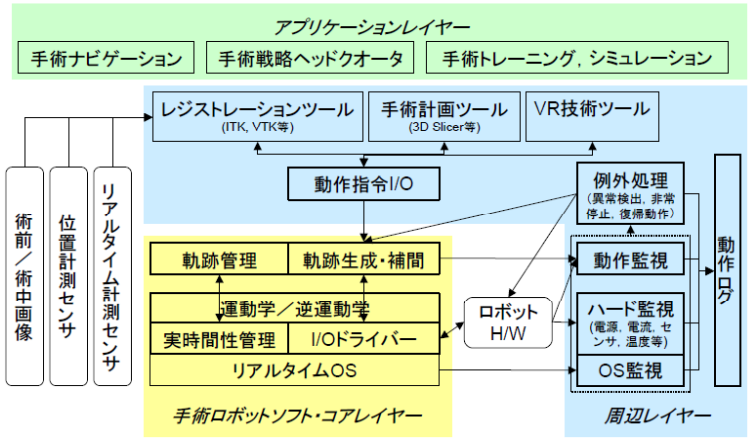
手術ロボット基盤ソフト技術開発

医療機器のソフトウェアでは，信頼性が極めて重要である．今日では系統的な動作検証を経ないソフトウェアを用いた医療機器は上市は論外として臨床研究に進むことも許されない．ところが，手術ロボットのソフトウェアはシステムの複雑化に伴い，その動作検証が煩雑化しており，これが開発上の阻害要因となっている．

これらの経験をふまえて，本提案で開発する「手術ロボット基盤ソフトウェア」は，医療機器としての高い信頼性を担保すると共に，本邦の手術ロボット開発の将来的な基盤となるソフトウェアを開発する．

・手術ロボット・ライブラリキット開発（ハーバード大学医学部）
・手術ロボット・周辺レイヤー開発（ハーバード大学医学部）
・手術ロボット・コアレイヤー実装（名古屋工業大学）
・医療機器ソフトウェア検証技法（産業総合技術研究所）

「インテリジェント手術機器」プロジェクト　ソフトウェア構成



手術ロボット制御オープンソフトウェア

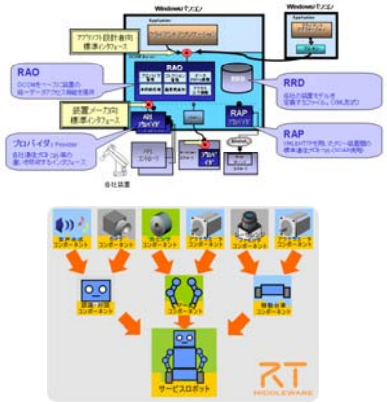
- 背景
  - 現在，Da VinciやZeusなどの手術ロボットを含め，様々なロボットが開発されている．

問題点：
- これらロボットの多くが実用化されているが，一方で，広く社会に普及していない．
- 手術ロボットにおいては，プログラムは非公開であり，安全性に不透明感が残る．
- 各ロボットの制御ソフトはおそらく似通ったものであり，共通する部分も多いと考えられるが，独自開発ゆえに効率が悪い．

ソフトウェア側での具体的な対策：
- 多種多様な顧客のニーズに対応できていないために普及しないと考えられ，迅速かつ安価に開発し，柔軟に要求に応えていく必要がある．
- プログラムを共用化・オープン化することで開発期間・効率を上げる．
- ロボットの安全性にソフトウェア側から共通の基準を示す．
- プログラム内部を機能別にモジュール（クラス）化し，簡単に脱着できるようにすることで，ロボットの機能・部品の差異，交換に迅速かつ柔軟に対応する．
- マルチプラットフォームにし，プログラム動作環境の違いにも対応する．

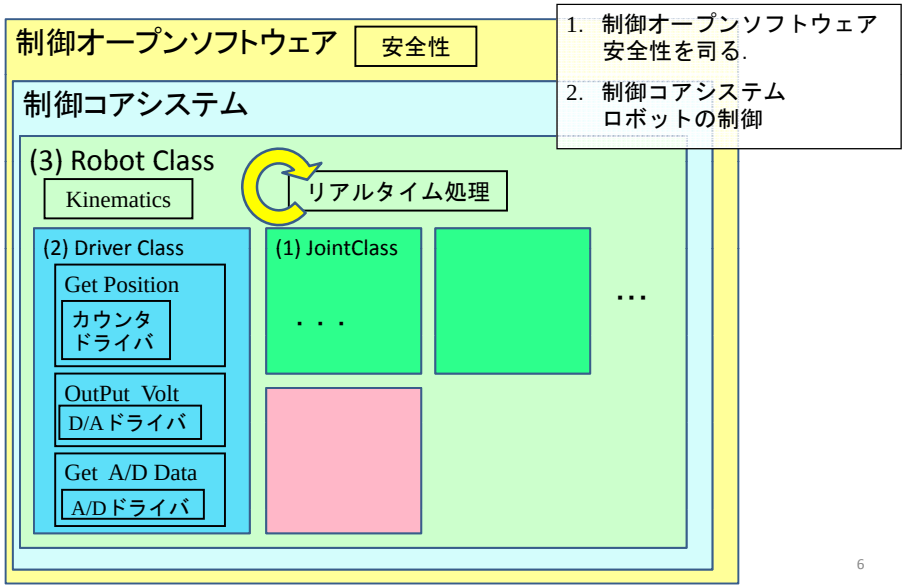
手術ロボット制御オープンソフトウェア

- 関連研究
  - ORiN
    - ネットワークに接続された装置に統合的にアクセスするために必要なインタフェース
    - WindowsPCを通して操作.
  - RTミドルウェア（RT：Robot Technology）
    - 様々なロボット要素（RTコンポーネント）を通信ネットワークを介して自由に組み合わせることで、多様なネットワークロボットシステムの構築を目指す.



- 目的
  - 本研究では，手術ロボットソフトウェアの安全性の基準を示し，オープンでマルチプラットフォームなモジュール化された手術ロボット制御ソフトウェアを開発することを目的とする.

手術ロボット制御オープンソフトウェア

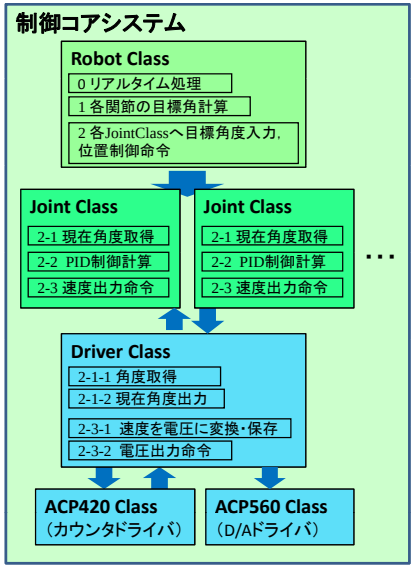


制御コアシステム

- 制御コアシステム
  - ロボットの制御そのものを担う.

以下の三つのクラスに分かれている.

  - Robot Class：
    - ロボットの制御全体を監視.
    - リアルタイム処理，運動学計算
    - JointClassへの制御指示
  - Joint Class：
    - 関節（モータ1軸）の制御（関節数に応じて増設可能）
  - Driver Class：
    - ボードドライバとの仲立ち
    - ハードウェア依存部分の吸収
- 現状
  - 手術用スレーブロボットに実装し，動作確認



開発した制御ソフトによる動作検証実験

クラス定義　—ROBOT Class—

```
class ROBOT{
 private:
    ROBOTDATA* robData;
    void setOrigin();                  //原点をセット（エンコーダデータを0にする）
    void kinCompt(double tipPos);  //機構学計算（各軸の目標角度を計算）
    void jCtrl();                      //ジョイントコントロール（各軸の制御）
    static void timing();              //リアルタイム処理用（内部割り込み）
    JOINT* joint0;                     //ジョイントクラス
    ・・・
    DRIVER driver;          //ドライバクラス
 public:
    void armCtrl(int endtime);       //ロボットクラスのメイン関数
    static SEM_ID timingSem;       //リアルタイム処理用
    ROBOT(int jnum);                   //コンストラクタ（引数：ジョイント数）
    ~ROBOT();
};
```


クラス定義　—JOINT Class—

```
class JOINT{
 private:
    int    jID;
    double Kp;
    double Ti;
    ・・・
    double degPID(double destDeg, double curDeg);          //PID制御
 public:
    DRIVER& dvDriver;                                                  //ドライバクラスを参照
    ~JOINT();
    JOINT(DRIVER& driver, int jID, double Revl_Linr, double pNum,     //定数の初期化
        double maxvolt, double minvolt, double Plus_Minus, double kp,double ti,double td);
    double degCtrl(double destDeg);                //角度（変位）制御
    double    forceCtrl(double destForce);                    //力覚制御
};
```


クラス定義　—DRIVER Class—

```
class DRIVER{
 private:
    ACP420 *acp420;                    //カウンタドライバ
    ACP550 *acp550;                    //A/Dドライバ
    ACP560 *acp560;                    //D/Aドライバ
 public:
    DRIVER(int jnum);                  //初期化（定数の初期化）
    void init(int jid, double rol, double pnum, double maxvolt, double minvolt);
    ~DRIVER();
    void   getEnc();                   //角度取得（ジョイントクラス用）
    void   getVolt();                  //電圧取得
    double outVolt();                  //電圧出力
    void   setEnc(double deg);         //角度セット
    void   zeroVolt();                 //出力電圧→0
    double    reguVolt(int jid, double volt);          //電圧制限
};
```


今後の予定

- 10月
  - プログラムの改良・変更，仮想的にマスタスレーブシステム構築．
  - 異なった機構のロボットにプログラムを実装して移植性を調べる．
- 11月
  - 仮想的マスタスレーブの動作実験
  - 力覚提示クラスの検討
- 12月
  - 力覚提示クラスを作成，マスタへの実装を検討
- 1月
  - マスタ動作実験
- 2月
  - マスタスレーブシステムを構築
- 3月
  - 安全性の議論へ