

玉乗りロボットをつくる

後編：回路と制御ソフトウェア

仙台市地域連携フェロー

熊谷正朗

kumagai@mail.tohoku-gakuin.ac.jp

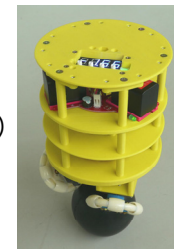
東北学院大学工学部
ロボット開発工学研究室



玉乗りロボットをつくる：構成

○ 前編：全体の構成とメカ設計

- ◇ロボット開発の仕様と構成
- ◇ロボットに用いる原理（発想と式）
- ◇駆動系の設計パラメータの調整
- ◇メカ全体の設計



○ 後編：回路と制御ソフトウェア

- ◇制御回路群（主マイコン、モータ駆動、表示）
- ◇制御の基本部分
- ◇実用性のための上位層

今回の目的 ～メカをロボットにする仕掛け～

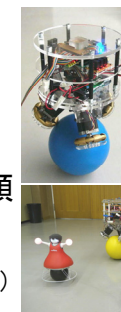
○ 後編：回路と制御ソフトウェア

- ◇ダイジェスト：ロボットの仕様・構成・メカ
 - ・ロボットの目的と基本原理／メカの構造
- ◇ロボットの制御回路
 - ・制御系（マイコン＋センサ）
 - ・電力系（駆動＋電源）
- ◇ロボットの制御ソフトウェア
 - ・制御系／上位操作系

開発の目的

○ 背景：玉乗りロボット

- ◇「球に乗ってバランスするロボットつくりたい」
 - ・という、学生さんの希望・提案(2004, 07)
 - ・ロボットの開発と発表(2008)
 - ◇このロボットの重要性（≠実用性）
 - ・コンテンツ性、教育の導入の話題
 - ・学内外デモンストレーションの筆頭
 - ・たまに学外から問い合わせある
- （※まれな公開企業事例：村田製作所様）



開発の目的

○ 背景: 既存ロボットの課題と要望

◇ 大きくて重い → 小さく軽く



- ・ 運搬の手間 (学内外、計15kg弱)
- ・ 実験時の危険性 (落ちると危険、破損)



◇ 設計データの欠如

- ・ 詳細な設計データが揃っていない
※ファイルの分散、落書き、そもそも無い
→ 問い合わせに答えきれない

開発の目的

○ 目的: 不十分さを解消する新規開発

◇ 小さく軽く、運用性の向上



- ・ 手持ちできるケースに一式入る
※市販のアルミケースを設計目標に
- ・ 準量産性の確保: 複数台運用



◇ 公開しうる設計データ

- ・ メカ: 3D 回路: 基板起こし ソフト: 可読性
- ・ 公開情報だけで、「やればコピーできる」
レベルの精細さを想定

構成の概要

○ 目的を実現するための構成 (メカ系)

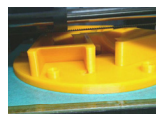
◇ メカの小型化

- ・ 駆動用車輪の小型化設計 (他テーマ兼用)
- ・ 構造見直しによる機構の圧縮



◇ メカの全面3Dプリント化

- ・ 「データがあればつくれる」
- ・ 一般的「切削加工図面→加工依頼」に
比べると試しやすい／改造しやすい



構成の概要

○ 目的を実現するための構成 (非メカ系)

◇ 回路の基板化 (前作もほぼ、再設計)

- ・ データ→実体化しやすい
- ・ 数量を確保しやすい (組み立て、特性均一)

◇ マイコンの変更とプログラムの書き直し

- ・ 世界的に入手性の良いマイコン品種
※海外からの問い合わせが多いため
- ・ 既知のノウハウに基づく書き直し
※試行錯誤・増築し続けてひどかったため



構成の概要

○ 目的を実現するための構成 (運用性)

◇ 単独運用・即起動 (既存仕様を改善)

- ・電源入れてすぐ動くこと 別PCなど不要

◇ 電池の入手性向上

- ・旧: ラジコン用NiCd/MH系充電池
※廃品傾向(Li系置き換え)、充電器の用意

→ 新: ビデオカメラ用Li系充電池



- ・入手性、保護有、充電器も
- ・導入しやすい、増やしやすい



C27 玉乗りロボット (後編: 回路ソフト) Page. 9 基礎からのメカトロニクスセミナー

玉乗りロボットの基本原理

○ 基本構成: バランス制御 + 球を転がす



◇ バランスの制御: 倒立振り子

- ・ほうきを手の上に立てて遊ぶことと類似
- ・立てた棒状のものの下端を移動操作する
※他の形式: 物を回転させる反動を使う

◇ 球を転がす: 3方向

- ・全方向移動用車輪
- ・複数の車輪で球を回転させる
- ・別の車輪の回転を邪魔しない



C27 玉乗りロボット (後編: 回路ソフト) Page. 10 基礎からのメカトロニクスセミナー

基本原理: 倒立振り子制御

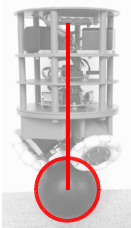
→ C09 制御の基礎

○ 姿勢を維持するフィードバック

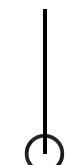
◇ 棒が倒れないように下端を加速的に動かす

- (1) 今傾いている → 直す方に動かす
- (2) 傾く速度がある → 止める方向に

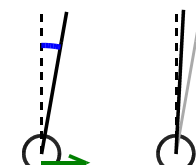
※倒れる動作が加速的 → 対処はそれ以上



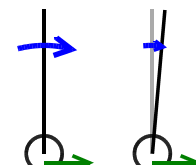
モデル化



目標



傾斜 → 先回り



傾斜中 → 緩和

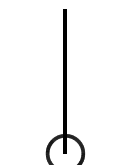
C27 玉乗りロボット (後編: 回路ソフト) Page. 11 基礎からのメカトロニクスセミナー

基本原理: 倒立振り子制御

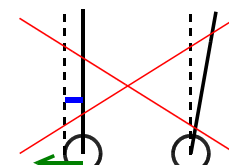
○ 位置を維持するフィードバック

◇ どこまでも走って行かないように位置の制御

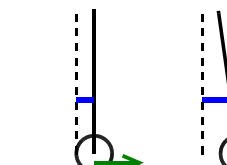
- × 基準位置に戻す方向に動かす
- 基準位置から遠ざかる方向に加速する
※安定判別の出す条件、実験的、考察的に



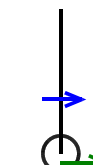
目標



基準方向に戻す



遠ざける → 手前に傾く



速度

C27 玉乗りロボット (後編: 回路ソフト) Page. 12 基礎からのメカトロニクスセミナー

基本原理: 倒立振り子制御

○ 倒立振り子制御の制御式

◇制御式

・移動の加速度 =

$$\begin{aligned} & \text{角度ゲイン} \times \text{姿勢傾斜角} \\ & + \text{角速度ゲイン} \times \text{傾斜角速度} \\ & + \text{位置ゲイン} \times \text{位置} \\ & + \text{速度ゲイン} \times \text{移動速度} \end{aligned}$$

※ゲイン: 反応の程度を調整するための定数

・移動の加速度を操作(指令)する

基本原理: 倒立振り子制御

○ 倒立振り子制御の制御式

◇この制御式の特徴

- ・動作は4個のゲインが決める
 - ※角度と位置に対するPD制御 (→C09)
 - ※ゲインの大小バランスで姿勢重視/位置重視
- ・一般には、
$$\text{トルク(力)} = \text{ゲイン} \times \dots +$$
の式(力操作は制御、ロボット系で一般的)
- ・ステッピングモータ使えるよう加速度操作

玉乗りロボットの基本原理

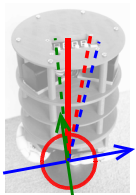
○ 倒立振り子制御を空間で実現する

◇単純なアイデア

- ・左右方向の制御 + 前後方向の制御
 - ※斜め方向に倒れる = 両者の組み合わせ

◇実現するために必要な駆動系

- ・左右 + 前後にきっちり加速度をだせる
 - ※それぞれ任意の大きさでの組み合わせ
- ・左右 + 前後にきっちり速度or位置でも可
 - ※加速度 → 積分 → 速度 → 積分 → 位置

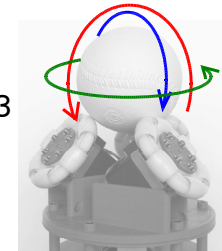


基本原理: 球の駆動

○ 球の回転操作

◇球の任意の回転の自由度は3

- ※自由度 = 回転・直動などの1軸の動きの合計の数



◇球を前後左右に回転できる

→ 倒立振り子制御、移動

◇鉛直軸まわりの回転

→ ロボットのその場での旋回が可能に

基本原理：球の駆動

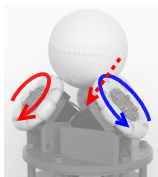
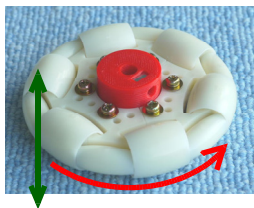
○ 3自由度回転の実現

◇全方向移動ロボット用の車輪＋球

- ・各車輪が、車輪の方向に球を回転させる
- ・他の車輪の回転を、邪魔しない

◇全方向用車輪の特性

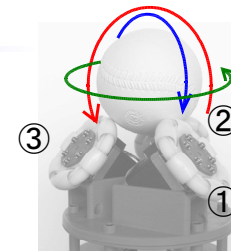
- ・**能動的に駆動**する方向
(回転方向)
- ・**受動的に受け流す**方向
(軸方向)



基本原理：球の駆動

○ 車輪の速度計算式

◇今回の配置に対しての計算
詳細はC26(前編)



$$\text{車輪1} = -0.5A \times \text{前後} - 0.87A \times \text{左右} + B \times \text{旋回}$$

$$\text{車輪2} = -0.5A \times \text{前後} - 0.87A \times \text{左右} + B \times \text{旋回}$$

$$\text{車輪3} = 1.00A \times \text{前後} + 0.00A \times \text{左右} + B \times \text{旋回}$$

※A,Bは別途決まる定数

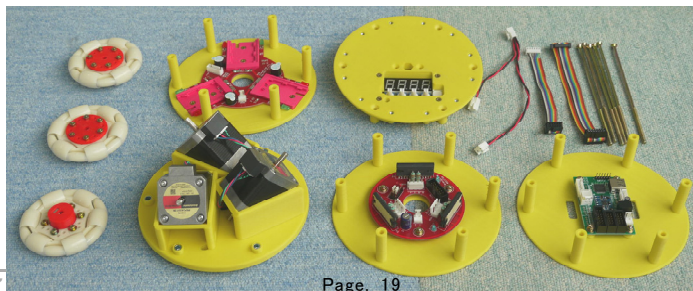
$$\text{※}0.87 = \sqrt{3}/2$$

ロボットのメカ設計：全体構成

○ ロボットの全体構造

◇層構造

- ・ベースとなる板部
- ・支柱(一体成形)
- ・貫通ネジ



回路・ソフトへの要求

○ メカを動作させ玉乗りロボットを実現

◇回路への要求

- ・制御系(マイコン＋センサ)
- ・ステッピングモータの駆動

◇ソフトへの要求

- ・**倒立振り制御**(一定周期)
- ・モータへの速度分配 & モータへの指令
- ・**デモに耐える操作性**などの上位層

玉乗りロボットの回路構成

○ Simple is best

◇回路は最大限シンプルに

- ・コスト、部品点数
- ・組込マイコンがなんとかしてくれることを前提とした回路設計

※周辺機能、ソフト処理性能向上

◇データシート(取説)の記述に忠実に

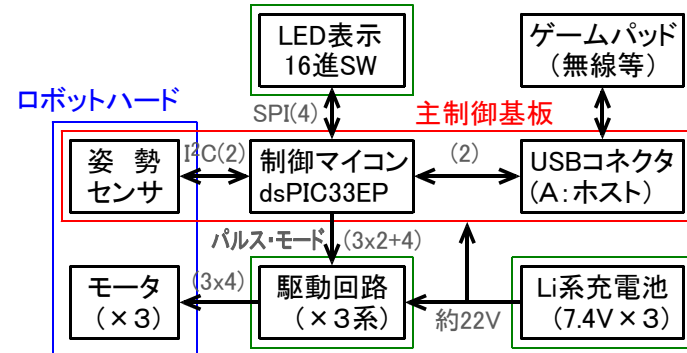
◇汎用性

- ・回路資源の流用(多様な開発の手間削減)

玉乗りロボットの回路構成

○ 回路の構成図

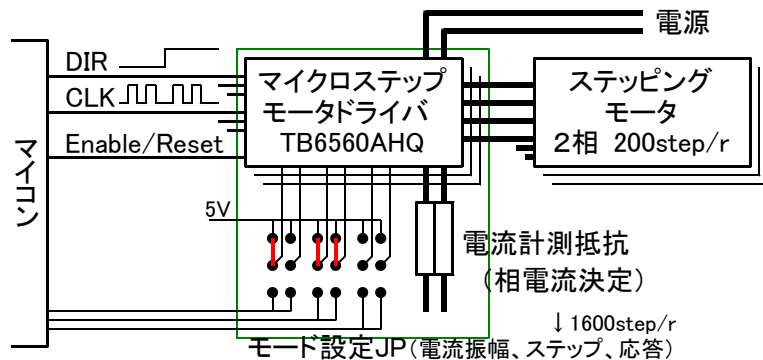
◇機能毎に回路は4グループ ※(線本数)



玉乗りロボットの回路解説

○ ステッピングモータの駆動回路

◇マイクロステップ駆動IC(のみ)

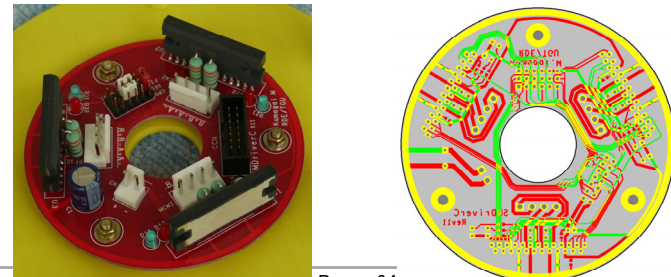


玉乗りロボットの回路解説

○ ステッピングモータの駆動回路

◇基板設計

- ・120度対称(見た目重視&質量バランス)
- ・部品の座標計算→配置インポート

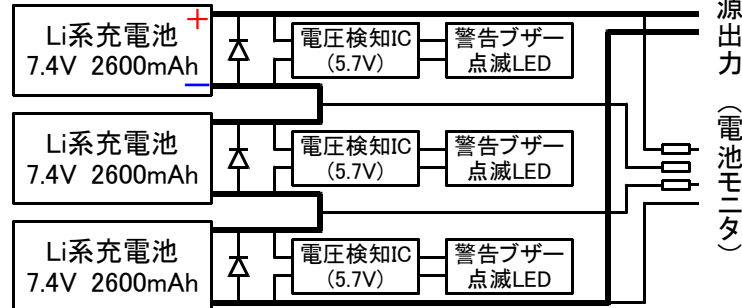


玉乗りロボットの回路解説

○ 電源部

◇電池＋電池バイパス＋低下警報

・ビデオカメラ用電池の内蔵保護前提



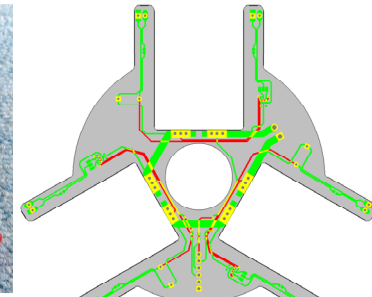
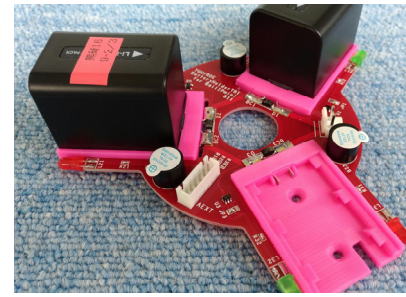
C27 玉乗りロボット (後編: 回路ソフト) Page. 25 基礎からのメカトロニクスセミナー

玉乗りロボットの回路解説

○ 電源部

◇基板化 (←旧: 空中配線)

・形状はCAD設計→DXFインポート



C27 玉乗りロボット (後編: 回路ソフト) Page. 26

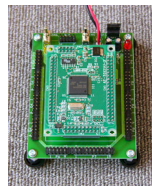
玉乗りロボットの回路解説

○ 主制御基板

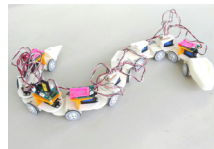
◇制御基板の設計開発方針

△ 玉乗りロボットの制御基板

○ 研究室の今後数年の小型ロボット制御用



- ・旧: 秋月H8＋コネクタ分配簡易母板
- ・マイコン基板はそうそう新造できない
※金額コストよりも手間/暇/精神力
- ・マイコン基板＋ハード対応ソフト流用で
開発負担の大幅低減



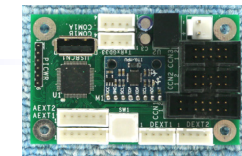
C27 玉乗りロボット (後編: 回路ソフト) Page. 27 基礎からのメカトロニクスセミナー

玉乗りロボットの回路解説

○ 主制御基板

◇制御基板(およびマイコン)の要件

- ・標準的なマイコンコア
- ・姿勢センサを搭載(不要なら実装せず)
- ・研究室標準コネクタをなるべく多く搭載
通信系(0, 3.3, デジタル入出力×2)
汎用系(0, 3.3/5, デジタル入出力×8)
アナログ系(0, 3.3, 5, アナ対応×3)
- ・電源回路を搭載(～40V供給)



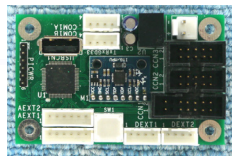
C27 玉乗りロボット (後編: 回路ソフト) Page. 28 基礎からのメカトロニクスセミナー

玉乗りロボットの回路解説

○ 主制御基板

◇制御基板(およびマイコン)の追加仕様

- ・USBホストができる
USB-OnTheGo対応マイコン
→ ゲームコントローラ等が使える
- ・基板上にインジケータと操作
2色LED内蔵スイッチ
- ・3相モータの制御ができる
モータ制御基板の試作用に

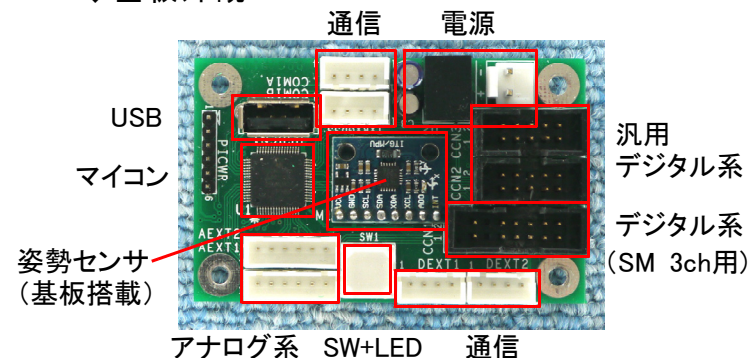


玉乗りロボットの回路解説

○ 主制御基板

基板寸法 72x45mm
タカス IC-301-60

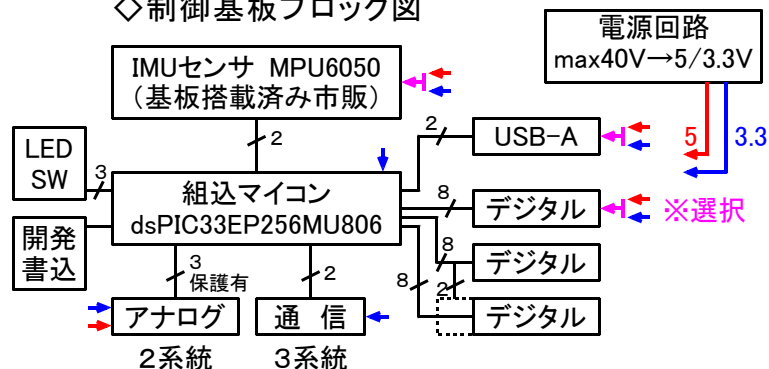
◇基板外観



玉乗りロボットの回路解説

○ 主制御基板

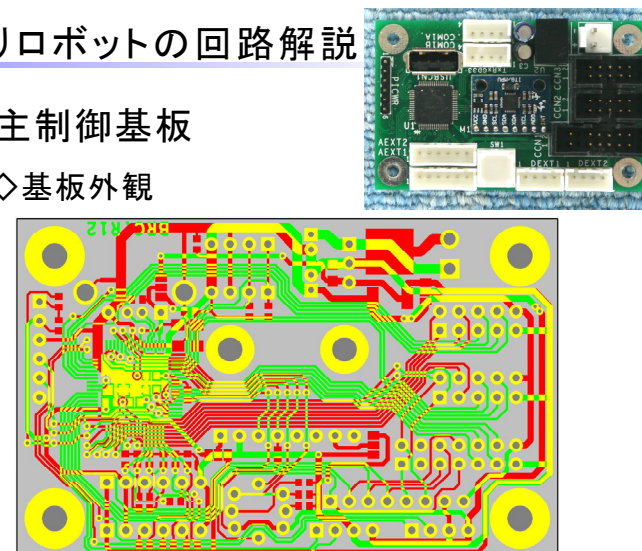
◇制御基板ブロック図



玉乗りロボットの回路解説

○ 主制御基板

◇基板外観

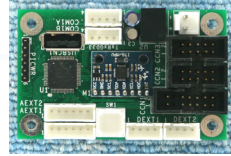


玉乗りロボットの回路解説

○ 主制御基板

◇マイコン基板設計のこつ？

- ・マイコン周り: **データシート通り**
 - ・ **電源供給**、**パスコン**、**クロック源**
 - ・ **リセット**、**開発時書き込み回路**
- ・ **特定機能端子**の割り当て
例) 5V耐性、アナログ入力、I²C
- ・ 配線しやすいように**回路図書き換え**
例) 配線交差→ピン割り当て交換

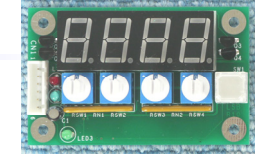


玉乗りロボットの回路解説

○ LED + 設定入力基板

◇4桁の7segLED + 4個の4bit16進SW

- + ステータスLED、LED内蔵スイッチ
- ・ ロボットの状態表示、デバッグ用
- ・ 動作の設定 (パラメータ設定)
- ・ 以前から同様なものを使用し、有用性○
- ・ マイコンとの接続はSPI型、4本のみ

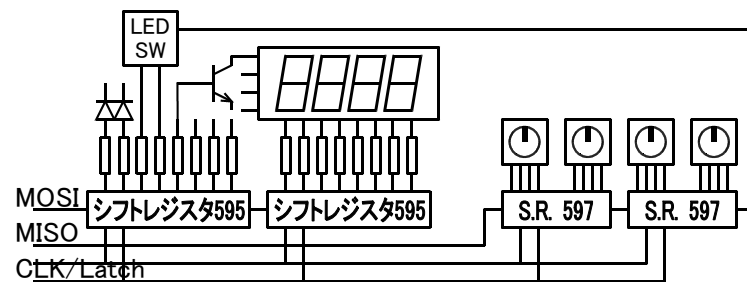


玉乗りロボットの回路解説

○ LED + 設定入力基板

◇ブロック図

- ・ 入出力のシフトレジスタ、ダイナミック点灯



玉乗りロボットの回路解説

○ 回路の製造

◇基板の外注

- ・ 両面シルク付きで**10枚で1000円台**~
※電池基板が70USD (大きさ、2oz仕様)
- ・ PCBGOGO, Elecrow, FusionPCBなど
中国の基板製造業、一部は日本語対応
基板のデータ(ガーバ)を送信 → 基板届く
通常指定で1週間程度
- ・ 部品発注と変わらない速さ?

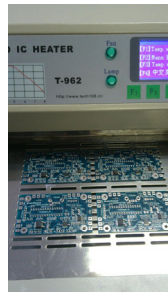


玉乗りロボットの回路解説

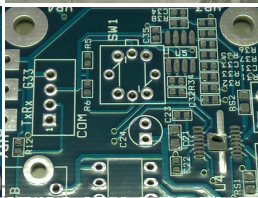
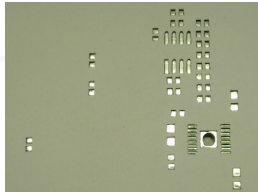
○ 回路の製造

◇基板の組立

- ・がんばって半田付け
- ・卓上簡易リフロー炉



(マスク+クリーム半田
+実装+炉で過熱)



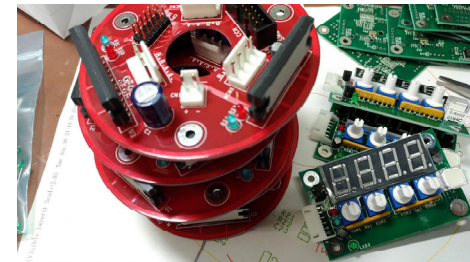
玉乗りロボット (後編: 回路ソフト) Page. 37 基礎からのメカトロニクスセミナー

玉乗りロボットの回路解説

○ 回路の製造

◇基板の組立

- ・そろそろ外注を試してみたい
- ・前記基板製造メーカ等で取り扱い



C27 玉乗りロボット (後編: 回路ソフト) Page. 38 基礎からのメカトロニクスセミナー

玉乗りロボットのソフトウェア

○ 制御ソフトウェアがなすべきこと

◇玉乗りの制御

- ・センサ情報の処理
- ・倒立振り子制御
- ・モータへの動作指令

◇動作シーケンスの制御

- ・初期化、動作の状態遷移

◇使えるロボットとしての操作機能

- ・ゲームコントローラへの対応

C27 玉乗りロボット (後編: 回路ソフト) Page. 39 基礎からのメカトロニクスセミナー

玉乗り制御

○ 制御の中核

◇制御式: 移動の加速度 =

$$\begin{aligned} & \text{角度ゲイン} \times \text{姿勢傾斜角} \\ & + \text{角速度ゲイン} \times \text{傾斜角速度} \\ & + \text{位置ゲイン} \times \text{位置} \\ & + \text{速度ゲイン} \times \text{移動速度} \end{aligned}$$

- ・「×」の右: 必要な情報 左: 要調整
- ・加速度→モータへの指令
- ・一定周期で繰り返し演算

C27 玉乗りロボット (後編: 回路ソフト) Page. 40 基礎からのメカトロニクスセミナー

玉乗り制御

○ 制御の実行に必要なもの

◇ロボットの姿勢傾斜角、角速度

← 姿勢センサ

- ・姿勢センサ(ハード)からの計測値取得
- ・姿勢角・角速度の計算

◇ロボットの位置、速度

比較: 角度センサ

← 操作指令値から積算で求める

◇球の加速回転

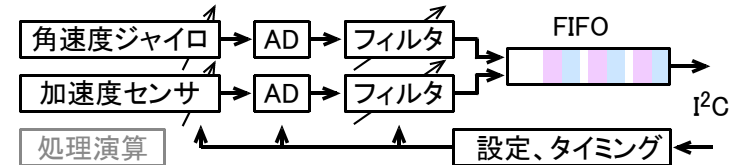
- ・球の駆動指令と各車輪の速度

玉乗り制御

○ 姿勢角の取得

◇デジタル通信型IMUセンサ MPU6050

- ・3軸の加速度 3軸の角速度(ジャイロ)
- ・AD変換、フィルタ処理など内蔵
- ・I²C通信、自前の計測周期、FIFO内蔵
- ・姿勢情報の処理機能内蔵(未使用)



玉乗り制御

○ 姿勢角の取得

◇デジタル通信型IMUセンサ MPU6050

◇ソフト側の処理 (参考→C13 デジタルセンサ)

- ・MPU6050との通信(I²C, 初期化, 平常)
- ・加速度と角速度の合成処理

◇処理方針

- ・センサ側FIFOから1式単位で読み出す
→ 処理 あるだけ繰り返す
- ・センサの周期との同期は不要

玉乗り制御

○ 加速度の操作・モータへの指令

◇ 加速度 → 積分 → 速度 → 積分 → 位置

◇前後・左右方向の倒立振子制御(加速度)

- 前後・左右方向の移動速度
- 3車輪の速度指令値

◇速度に応じたモータへの指令パルス

- ・1パルス=1角度単位の回転 (1/1600回転)
- ・モータの回転速度=対応する周波数

玉乗り制御

○ 速度演算部の実装式

◇ 加速度 a → 積分 → 速度 v → 積分 → 位置 x

・ 周期ごと: $v = v + a$, $x = x + v$

・ 正式には: $v = v + (a \times \text{周期})$

周期一定、 a, v, x は内部の単位系

※制御周期が1単位時間のような

◇ 速度分配式 $\times \sqrt{3/2} = 0.867$

・ 車輪1 = $-0.5 \times \text{前後} - 0.866 \times \text{左右} + \text{旋回}$

・ $ms1 = -(s1 \gg 1) - ((s2 * 222) \gg 8) + s3$

C27 玉乗りロボット (後編: 回路ソフト) Page. 45 基礎からのメカトロニクスセミナー

玉乗り制御

○ パルス出力: DDS型 (Direct Digital Synthesizer)

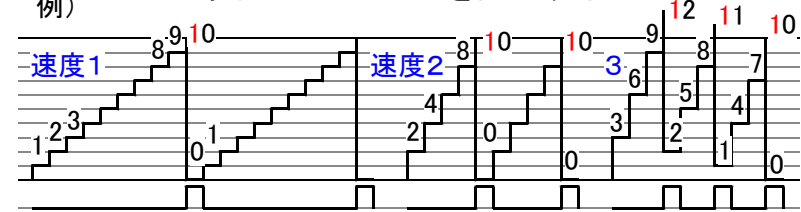
◇ 手法

・ 上限のあるカウンタ変数を用意し、

・ 一定の周期で速度値を加算し、

・ あふれたらパルスを出力する

例)



C27 玉乗りロボット (後編: 回路ソフト) Page. 46 基礎からのメカトロニクスセミナー

玉乗り制御

○ 実装上のその他の主な細工

◇ すべて整数(固定小数)で演算

・ 一般的な浮動小数は計算負荷が高い

※floatよりはlongのほうが分解能高い

・ SI単位系ではない、独自単位系

◇ 割り算を使わない

・ 他の演算に比べてかなり遅い (例 約1/20)

・ $\div (2^n)$ にする → 右シフト演算($\gg n$)

C27 玉乗りロボット (後編: 回路ソフト) Page. 47 基礎からのメカトロニクスセミナー

玉乗り制御

○ 主ループの実装

◇ 制御演算を一定周期で

・ センサの情報処理

※センサ自身の周期とは差、最新値

・ 倒立振り子制御

・ →3個のモータの速度指令

※モータの指令生成DDSは別周期(10k)で

◇ 制御周期

・ 500Hz (2ms)

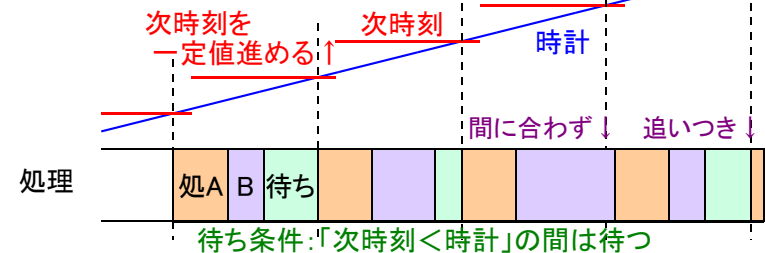
C27 玉乗りロボット (後編: 回路ソフト) Page. 48 基礎からのメカトロニクスセミナー

玉乗り制御

○ 主ループの実装

◇時計待ち型実装

- ・等周期でカウントアップする時計数値
- ・「次の時刻」を待ち、更新する



玉乗り制御

○ 主ループの実装

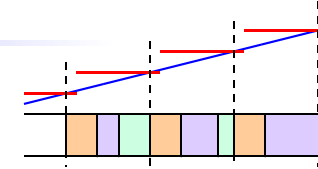
◇プログラム例

・時計変数: **clock** 次時刻: **next** 刻み: **step**

```

next=clock+step; // 初期値
while(1) { // メインループ
    while(clock<next) {
        暇つぶし処理;
    }
    next=next+step; // clock+step
    以降、制御処理等

```

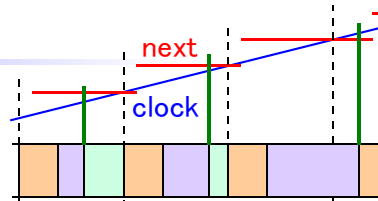


玉乗り制御

○ 主ループの実装

◇この実装の利点

- ・複雑で重たい(?)制御演算を割込にしない
※時刻clockのカウントアップのみを割込
→ 割込にありがちな開発トラブル低減
- ・処理落ちが分かる: **[暇つぶし]の直前に**
正常: $clock < next$ のはず、 $n-c > 0$
落ち: $clock \geq next$ になる、 $n-c \leq 0$
→ $next-clock$ が指標になる

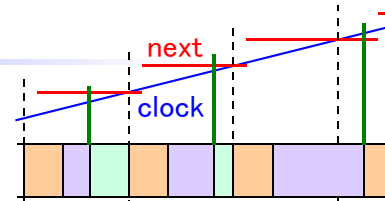


玉乗り制御

○ 主ループの実装

◇補足

- ・ $next-clock$ を周期毎に検証
 - (a) 常に正: 問題なし
 - (b) 負にどんどん増加: 間に合っていない
→ 処理見直し or 周期設定長く
 - (c) 周期的に負が見られる: ほぼOK?
→ 定期的な通信送信などの確認
 - (d) 不規則に負 → 外的イベント等



装置としての必要な追加処理

○ 制御演算だけでは機能しない

◇初期化处理 (ハード全体、姿勢値のみ)

◇起動一停止処理 (状態遷移)

◇通信

- ・PC等とのシリアル通信
- ・オプション: ラジコンサーボとの通信

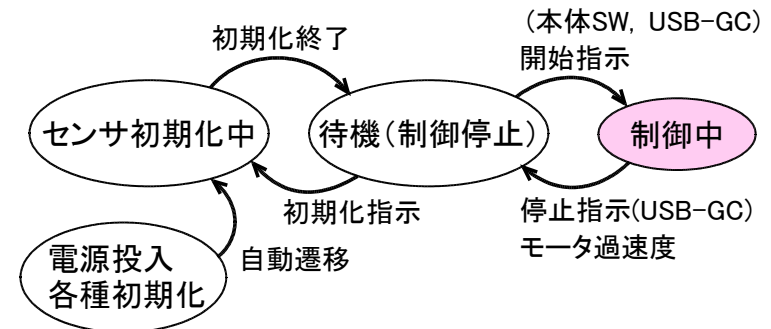
◇人間からの操作の受付

- ・起動一停止等、パラメータ調整
- ・移動などの動作の指示

装置としての必要な追加処理

○ 状態遷移

◇状態: {待機・制御中・姿勢センサ初期化}



装置としての必要な追加処理

○ ゲームコントローラへの対応

◇USBへの対応

- ・USB-OTG対応マイコンを採用
 - USBコネクタへの配線程度の回路
 - ・必要なコード類・事例はメーカーが公開
キーボード、マウス、メモリなど
 - ゲームコントローラへの対応改造
- ※ HIDデバイスとして基本は同じ
デバイスからのデータの解析

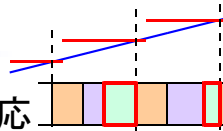
装置としての必要な追加処理

○ ゲームコントローラへの対応

◇ゲームコントローラへの対応改造

- ・デバイスからのデータを16進ダンプ
- ・コントローラのボタンやスティック操作
 - データ列内で変化するところがある
 - 対応関係を解析して、取得コード化
- ・現状では特定の数種のコントローラのみ
対応(自動で対応する手段はあるはず)。

装置としての必要な追加処理



○ ゲームコントローラへの対応

◇制御との干渉回避

- ・サンプルコードは 1kHz の周期割込で USB の処理関数を呼ぶようになっていた
= 制御処理を途中で止める可能性
※割込: 現作業を強制中断して別動作
- ・主制御ループの「周期待ち」で、同関数を呼び出すように ※P50「暇つぶし」
+ 適当なループ毎に送信リクエスト

装置としての必要な追加処理

○ ゲームコントローラへの対応

◇コントローラ操作→ロボットの挙動

- ・起動、センサ初期化等
- ・スティック操作→移動指令
※位置、速度、傾斜(加速)指令モード
※旋回速度指令
- ・パラメータの変更機能
※実験実習用機能
- ・コード量は姿勢制御本体よりも多い

補足: ソフトの最下層

○ ハードと直接対応するソフト

◇各種ハードの初期化

- ・メーカーによるスタートアップコード
- ・動作クロックの設定
- ・ピンの入出力、機能割り当て
- ・割り込みやタイマ類の設定
- ・通信ハード、アナログ入力等の設定
↓
- ・通信機能、センサ処理の初期化等

補足: ソフトの最下層

○ ハードと直接対応するソフト

◇この部分には特殊な知識・情報必要

- ・周辺機能の動作の理解
- ・機能操作についてのお約束的パターン
- ・メーカー毎の癖
- ・マニュアルをちゃんと読む必要

◇書けるようになるには

- ・サンプルやネット参考でとにかく動かす
→ 意味を理解 → マニュアルだけで自力

補足：ソフト開発とExcel

○ プログラムの一部をExcelで書く

◇繰り返しやパターンを表計算で

- ・数表
- ・キーワードをもとにした複数行生成

◇基本テクニック

- ・CSVで別名保存したものを#includeする
※CSVは数式が保存されないのでxls残す
- ・左の方にコードやデータを、次に
コメント化記号を入れて、右は自由に使う

補足：ソフト開発とExcel

○ プログラムの一部をExcelで書く

◇例：定義の自動化 ※ xは行番号

- ・Axセル：
= "#define "&Cx&" "&Dx&" //"
- ・Bx: 15(通し番号等) Cx: ="IO"&Bx
- ・Dx: ="Pin"&VLookup(Bx, ...)

◇例：数表

- ・="int table[128]={"
- ・=cos(Ex), =sin(Ex), "//", 通番号, =Dx...

まとめ

○ 玉乗りロボットの開発(回路+ソフト編)

◇メカを動かすための回路

- ・制御するための組込マイコン
- ・モータの駆動回路
- ・必要なセンサ

◇メカと動かすためのソフト

- ・制御理論のプログラム実装
- ・動作状態の制御
- ・「使える機械」に必要なユーザ対応

まとめ

○ 玉乗りロボットの開発(回路+ソフト編)

◇この玉乗りロボットについては

- ・倒立振子制御
- ・実運用に配慮した上位層
- ・USB接続したゲームコントローラで操作
- ・多用途転用を前提にした開発

◇玉乗りロボットのデータ公開

<http://www.mech.tohoku-gakuin.ac.jp/rde/contents/tech/BallIPMini/indexframe.html>