SCITECS-eyeの紹介

～画像処理部門～

神戸市立科学技術高等学校

登　弘聡

2010年6月27日

出場のきっかけ

- 2008年に溝上さんの話を聞いて、ラインをトレースするための技術が面白そうだと思った

2008年の取り組み

- とりあえず、少しでも長い距離を走れるように工夫した。
  - 画像データの変位に比べ制御量が小さすぎたため、ハンドルのゲインを上げた
  - 周囲の明るさに左右されたのでライトでコースを照らしてその誤差を少なくした

2008年のプログラムで改造したところ

```
const handle2[] = {
    0,
    0,2,2,3,3,
    3,5,7,7,7,
    7,10,10,12,14,
    15,15,15,15,18,
    20,20,20,20,22,
    22,22,24,24,24,
    24,24,25,25,26,
    26,27,27,28,28
};
```

```
Control=DataThinking(7);                    //PhotoGetとの間
の処理を入れる。(Line1->7)
if( Control == 999 ) {                      // Error
        speed(5, 5);
        return -1;
}
//}while( Control == 999 );
printf("Ha=%d ¥n", ha);
if( Control < 0 ){
        ha = -(handle2[abs(Control)]);
} else {
        ha = handle2[Control];
}
```

2008年度結果

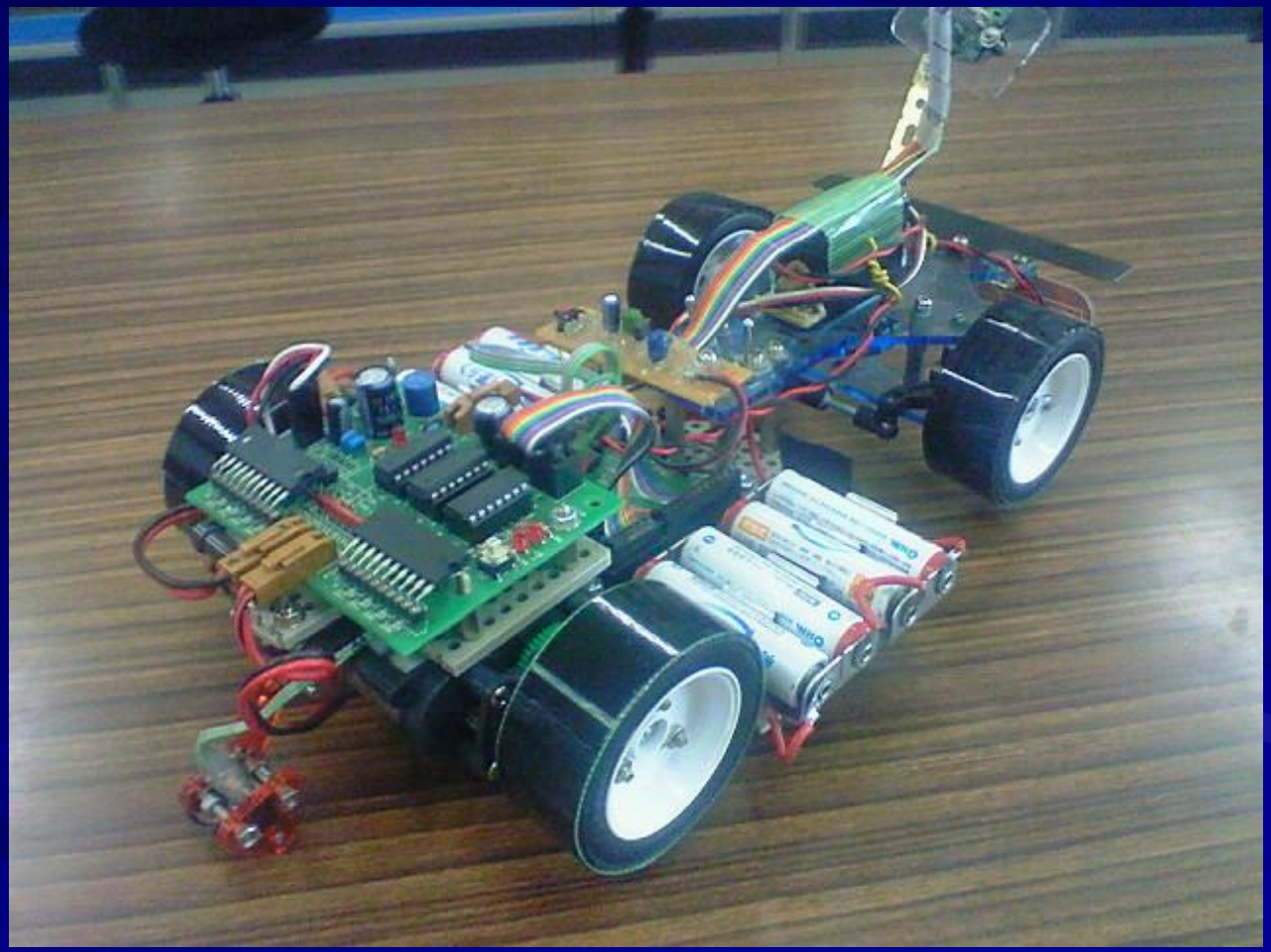
- 結果、2位。
  - かなり長い距離を走ることができたが、電通大の林さんには勝てなかった。
  - 次回は必ず勝とうという気持ちが強くなった。

2009年度結果

- 優勝
  - 昨年度の反省を生かし、全面的にプログラムを見直した。
  - 前回の画像処理大会から1年、もっとすごいマシンが出てくると予想していた。

今のマイコンカー

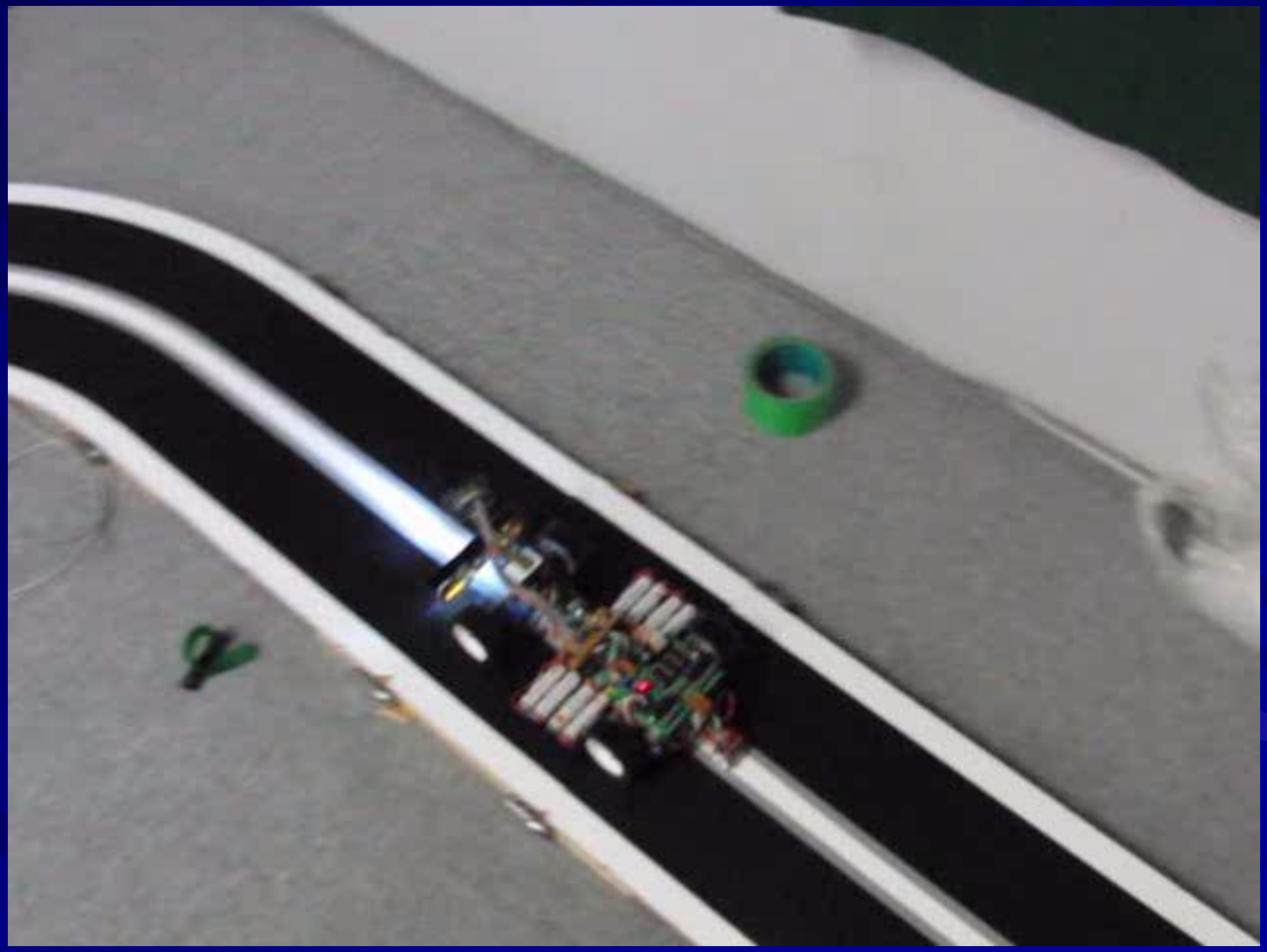
後ろから



SCITECS-eye仕様

項目	内容
カメラ	M64282FP(ゲームボーイ用カメラ)
CPU	H8-3048F ONE TypeH
ドライブ基板	モータドライブ基板 Vol.3
駆動	後輪駆動(タミヤハイスピードギヤ18:1)
エンコーダ	ロータリーエンコーダキットVol.2(両エッジ検出 72パルス/回転)
ライト	10000mCd白色LED、16灯(一列)
電源	単3×8本(CPU;2本を昇圧、駆動;6本)
サーボ	近藤科学 PS-2173FET(8ms周期で駆動)
ステアリング	アッカーマン・ジャント方式(に変更した)

走行の様子



2009年の取り組み

- 当初の目標はクランク、レーンチェンジ以外のコースを完走すること
  - S字、Uターン、タコツボなど
- プログラムの構造をkit07に準じた形に変更
  - 元のプログラムはモジュール化されすぎていて、私にはわかりづらかった。
  - Kit07の形にすることでクランク、車線変更の拡張がしやすいと思った。

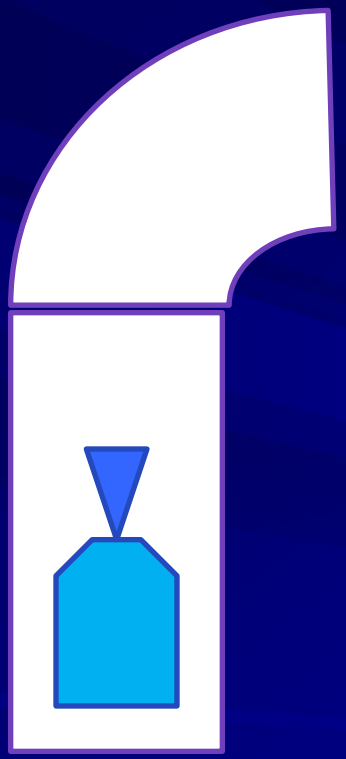
試行錯誤

- とにかく、走らせた
  - PCとマイコンカーをケーブルで接続したまま走らせてログをとった。
  - 多い日は1日に100回近く走らせた。
  - ログを見てカメラでとらえたコース情報がどのように処理されているかを見た。
    - 主に、ハンドル角の調整やマーク検出の方法を考えた

カメラの取り付け

- 2008年の講習会で学んだとき
  - カメラはハンドルとともに動く
    - 切り返しに弱い
    - 横方向の変位が弱まり、うまく曲がれない。
  - かなり前を見ている
    - 車体の状態に関係なく先の情報が入ってしまう。

カメラとハンドルが連動するときの問題点



変位なし

右曲を検出

ハンドルを切ると
変位がなくなる

あまりハンドルを切らずにコースアウトする

カメラとハンドルが連動するときの問題点

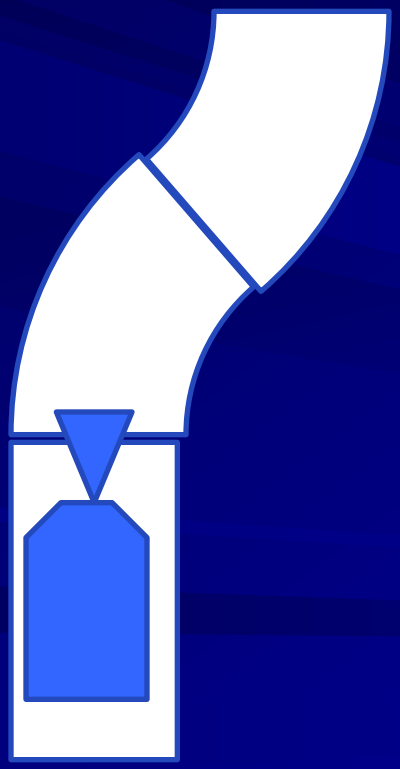
- カメラが先にカーブの方向を見るため、車体が曲がり切るまでに変位がなくなってしまう。
- カメラの見る向きと車体の向きに大きな差ができてしまう。
- S字などのコースに弱くなる。

カメラの取り付け2

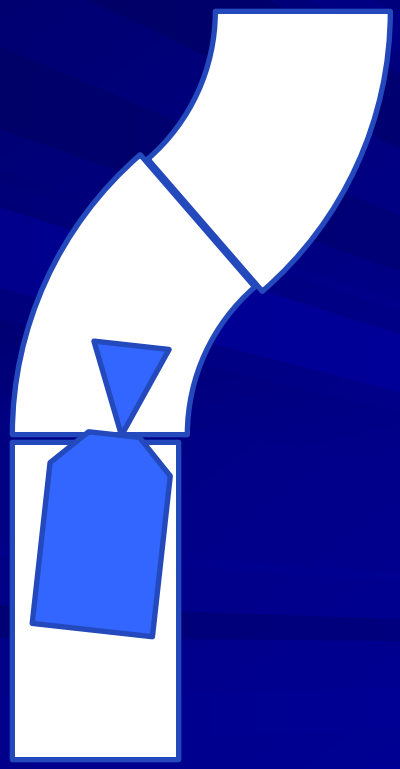
- 2010年の大会では
  - カメラは車体に固定し、まっすぐ前しか見ない
    - わずかな変化が大きくとらえられる
    - カーブの切り返しをうまく読み取れる

カメラを車体に固定した場合

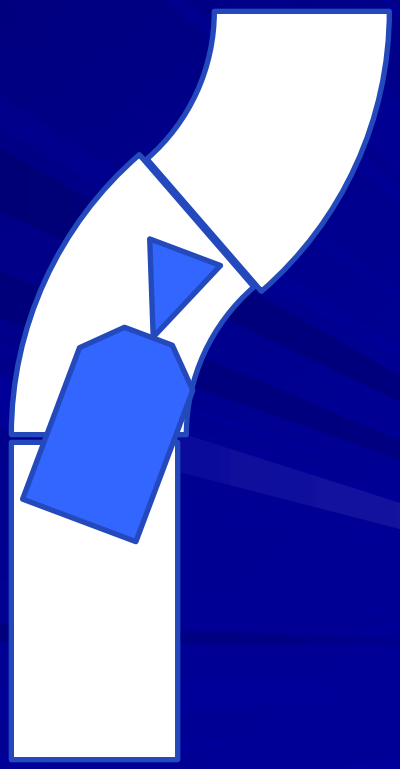
右曲を検出



少し進んでも右曲を検出



この辺りで左曲を検出、ハンドルが戻る



カメラを車体に固定した場合

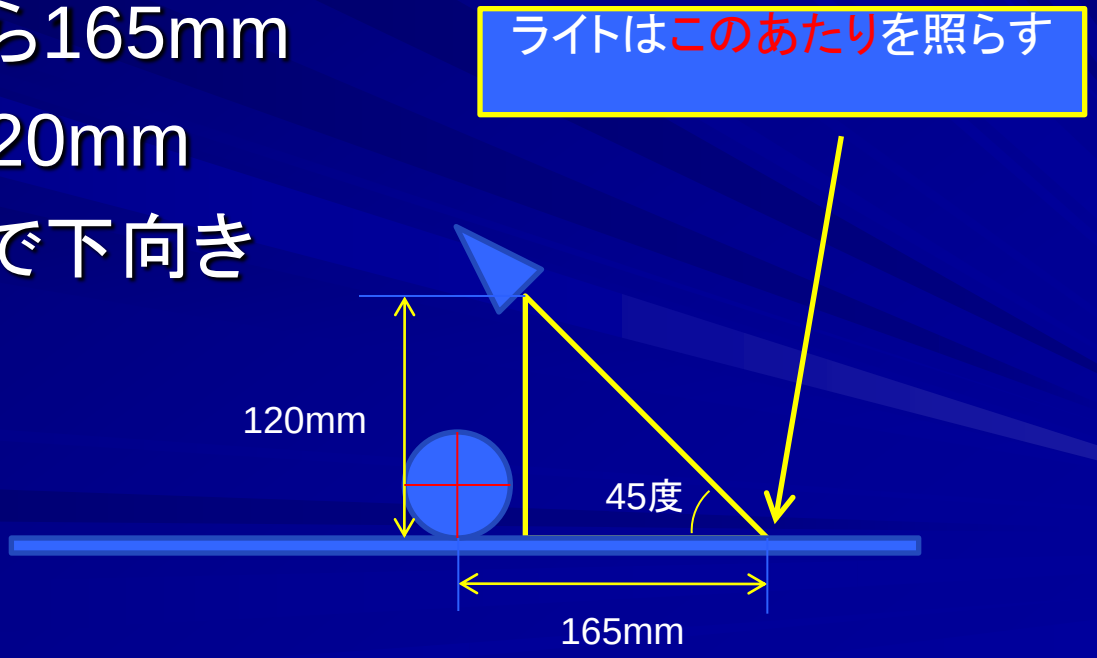
- 車体の向きが変わるまで、カメラで変位を検出できるため、コースに沿って走ることができる。

カメラの取り付けの工夫

- マイコンカーの真ん中でまっすぐ取付けない
  - ソフトの問題かもしれないが、中央から少し右寄りがちょうどよかった。
  - カメラ自体も右に首をかしげるようにつける
    - まっすぐだと、直線を右曲がりだと処理してしまう。
- ソフトウェアで処理しにくい部分はハードウェアでどうにかする。

カメラ取り付けの工夫2

- あまり前を見ない。
  - 余計な情報が入ってしまう。
    - カーブでクロスラインやハーフラインをとってしまう。
  - 前輪の中心から165mm
  - コース面から120mm
  - 水平から45度で下向き



取付け位置の決め方

- 直線コースを走らせてみる
  - センターラインより右寄りを走る時は、右側に寄せる。
  - センターラインより左寄りを走る時は、左側に寄せる。
- 経験的に、少し右寄りを走るぐらいがちょうどよかった。

取付位置を変えた時の問題

- カーブの特性が変わる
  - はじめからどちらかに偏っているので、左右で曲げ角を変える必要がある。
    - 左曲げがきつくなる傾向があった。
- マーク読み取りの特性が変わる
  - カーブと同じように左右どちらかのマークがうまく読めない。
    - 検出条件を変更する

ロータリーエンコーダ

- 元々、溝上さんのマイコンカーにも使用されていた。(プログラムから推測)
- 画像処理をする都合上、できるだけ一定の速度で走りたい。
- カーブでは適切な減速が必要。
- 上り坂は加速、下り坂はブレーキがかかる。
- クランクやレーンチェンジで大活躍

ギヤ比の変更

- 講習会で習ったマイコンカーは11.6:1のギヤ比(オレンジと黄色)
  - トルクが低く、低速での走行に向かなかった。
- ギヤを組み替えて、18:1(赤と緑)に変更。
  - PWMを上げることができた。
  - 加速、ブレーキの性能がよくなり、制御しやすくなった。

ヘッドライトの効果

- コース全体の明るさが均一ではない
  - 照明の位置
  - 窓の位置
  - 人、物による影
- できるだけ外乱の影響が少なくなる方法は無いか？
  - 自分で光源を用意してカメラの前が均一な明るさになるようにすればよい
- 結果として、外乱に強くなり、白黒のコントラストが上がったのでトレースしやすくなった。

偏光フィルム

- 外乱光（蛍光灯、窓）による反射
- ヘッドライトを点灯することにより、コース表面で光が反射してうまく撮影できない。
- コース表面の反射がなくなる角度で取り付ける。

カメラからの信号を処理する

- とりあえず、8行16列の配列にデータを入れる
  - カメラの画素数は128x128
    - 128/16=8…8画素の平均が列データ
    - 行(縦方向)は適当な8ラインを抽出
  - アナログ値
  - 二値化した値
- TeraTermなどのソフトで観測する

信号処理プログラム

```
for( rensya = 1; rensya < 100; rensya++ ) {
  // カメラの初期化
  CameraInit( dExposure, gCameraOffset, gCameraGain );
  // カメラのクロックスタート
  CameraClockStart((16*cLineSkip[0])+cAdd[0]);
  // カメラが準備できるまで待つ
  while( !CameraInterruptFlag );      // カメラ割込みフラグチェック
  for( SkipCnt = 0; mREAD == 0; ) {   // カメラの信号を読み飛ばす？
    mXck = 1;
    SkipCnt++;
    #pragma asm
      nop
      nop
    #pragma endasm
    mXck = 0;
  }
```

つづき

```
for( Line = 0; Line < 8; Line++ ) {
        CameraClockStart((16*cLineSkip[Line])+cAdd[Line]);
        for(; !CameraInterruptFlag; );
        // ソフトウェアクロック
        SoftClock( cByteSkip[Line] );
        // カメラからの情報を1ライン取得
        M6Save( vData, MaxLineDataSu );
        for( i = 0; i < MaxLineDataSu; i++ ) {
            vDataSave16[ Line ][ (i/8) ] += vData[i];
            // 8個ずつまとめてデータとする。
        }
        for( i = 0; i < 16; i++ ) {
            vDataSave16[ Line ][ i ] >>= 3;     // 8で割る
        }
    }
```

カメラからの信号を観察

- ワークスペースを開いてcameraプロジェクトを書き込む
- Tera Termで表示する。
  - 通信速度は、「38400bps」にしてください。
- クロスラインのあるコースを使って、マイコンカーの先端から各行がどのあたりを見ているかを調べる。

マトリクスから傾きを得る

- マトリクスデータで、手前のラインから順にセンターラインを割り出す。
  - 0行目と1行目で見つからなければ計算不能とする。
  - 4行以上見えないと計算不能とする。
  - 前回のセンターの近辺(±3列以内)を調査する。
  - 前回が検出不能なら全部調べる。
- (傾き)＝{(最も奥のセンター位置)－(最も手前のセンター位置)}÷(検出できた行数)×(係数)
  - けっこういい加減なやりかたで計算。

センターライン探索

- 初期状態は、マイコンカーがコースの中心に置かれているものとする。
  - 手前の数ラインは中央付近にくる。

傾き計算プログラム

- deltaX = (Pos[startLine] - Pos[endLine]) * 100 / Enkin[endLine];
- deltaY = endLine - startLine;
- angle0 = deltaX * 10 / deltaY;
- if(angle0 < -30)　angle0 = -30;
- if(angle0 > 30)　angle0 = 30;

サーボコントロール

- もとめた傾きをそのまま角度として使う
  - ±30度を超えないようにリミッタをかける。
- 実際の走行では、探索開始ライン(一番手前)の白線位置から、ハンドル角に補正をかけている。
  - 真ん中付近：そのまま
  - 右寄り：値を加える(右曲げの要素)
  - 左寄り：値を減ずる(左曲げの要素)

ハンドルのテストをする

- まず、サーボのセンターを求める
  - Kit07のsioservoを使って調整
    - カメラ関係のコネクタは外しているほうが無難かもしれません。(ポートI/Oの設定が違うため)
- Servoプロジェクトを書き込む
- Tera Termで表示しながら、マイコンカーをコースに対して傾けてみる

実際の走行プログラム

- できるだけ、kit07のプログラムの構造に近付けた。
  - 普通のマイコンカーに取り組んでいる人でも簡単に画像処理に取り組めると思ったから。
  - エンコーダ、EEP-ROMなどの拡張が容易になる。
  - パターン方式を採用
  - 必要な関数もメインプログラムに記述した。
    - モジュール化という観点から見ると悪いプログラム

移植にあたって苦労したこと

- 定義されているレジスタの表現が少し変わった。
  - ヘッダファイルが変わったため。
    - Kit07で使用しているヘッダファイルでは、バイト単位でレジスタにアクセスする。
    - 標準のヘッダファイルは、ビット単位でレジスタにアクセスできる。
  - カメラを使うにあたって、各レジスタにビット単位でアクセスする必要があるため仕方ない。

レジスタの対応表(一例)

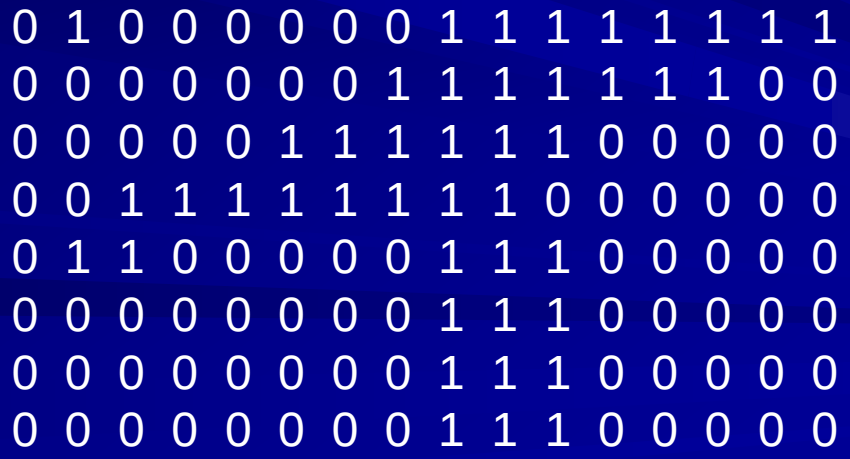
内容	Kit07	Eyechan2010
ポートレジスタ	P?DR	P?DR.BYTE
ITU	ITU3_TCR ITU_FCR ITU3_GRA ITU3_GRB ITU3_BRB ITU4_GRA ITU4_BRA ITU4_GRB ITU4_BRB ITU_TOER ITU_STR	ITU3.TCR.BYTE ITU.TFCR.BYTE ITU3.GRA.WORD ITU3.GRB.WORD ITU3.BRB ITU4.GRA.WORD ITU4.BRA ITU4.GRB.WORD ITU4.BRB ITU.TOER.BYTE ITU.TSTR.BYTE
ポートレジスタ(ビット)	P?DR &= 0x??	P?.DR.BIT.B?

通常走行

- とりあえず、通常走行を試してみましょう。
  - Run test プロジェクトを書き込みます。
  - 可能なら、TeraTermを起動して通信しながら、PCを抱えて犬の散歩みたいにして走らせるとデータが取れます。
- 詳しいプログラムは、別紙の通り。

クランク、レーンチェンジ

- クロスライン、ハーフラインを検出する
  - 走行しながら撮影するため、マークがきれいに映らない。
    - 横方向の走査をしている間にもマシンが進むためにずれが生じる。



検出方法

- 何行かまとめて見ると、クロスラインがあることがわかる。
  - 列ごとに複数行でビットORを取れば検出可能

```
      0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1
      0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0
      0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
      0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0
      0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0
      0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
      0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
OR)   0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
-------------------------------------------------------
      0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

問題点

- きついカーブにさしかかった時、センターラインと外側の白線が映る場合がある。
- これをビットORすると、同じような結果になるので、クロスラインと判断する条件を追加する

```
1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0,99,  23
 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0,99,  23
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0,99,  23
 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1, 0,  23
 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1, 0,  23
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1, 1,  23
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1, 1,  23
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1, 1,  23
```

解決法

- センターラインを検出できた行数で判断する。
  - カーブの場合、外側の白線を検出しても、センターラインとは判断させないので、ビットORで条件を満たしても無視させる。
- 探索開始ラインの位置が中央付近かどうか。
  - 探索開始ラインが端のほうに寄っているときは、カーブを処理しているため、無視させる。

クロスライン、ハーフライン検出

- クロスライン
  - Line5~Line0をビットOR→0xc3c3でマスクをかけた結果が0xc3c3、0xc3c2、0x43c3ならクロスラインと判断
- 右ハーフライン
  - Line6~Line0をビットOR→0x03c3でマスクをかけた結果が0x03c3、0x03c2、0x03c1、0x01c3、、0x01c2、0x01c1、0x00c3なら右ハーフライン
- 左ハーフライン
  - 省略

クランクの角を検出

- ハーフライン検出と同じ方法で検出する
  - ただし、探索範囲をLine3~Line0の範囲にする
    - 角を検出してすぐに曲がると内側に落ちるため。

クランク終了処理

- エンコーダにより、曲がり具合を検出する
  - ハンドルの角度でだいたいの回転半径がわかるので、それを元に80度ぐらい回ったときの距離を使う
- カメラで新しいセンターラインを検出する。
  - Line7~Line0でビットORをとり、右は列6~3のどれか、左は列12~9で白線を検出した場合

レーンチェンジ開始

- カメラからの入力でLine7~Line0をビットORした結果が、すべて0なら、レーンチェンジを開始する。

レーンチェンジ終了

- レーンチェンジ開始後一定距離を進んだ後にカメラで新しいセンターラインを見つける
  - エンコーダの取り付け位置により、左右で距離は異なる。
- カメラで、Line4~Line2をビットORした結果、列9~6のいずれかで白を検出したら終了

クランク、レーンチェンジ終了後

- 車体の状態が不安定であると考えられるため、一定距離を進んでからパターン11へ戻るようにしている。
  - とくにレーンチェンジ終了後は車体の角度がついているので不安定である。

総合的な走行

- Scitecsプロジェクトを書き込んで走らせてみる。
  - DataThinking関数などを調整する必要があるが、おそらく走れるはず。
  - 一部訂正あり(次のスライド)

総合走行の訂正(1)

- 行番号1988、2026、2137、2176
  - if((Pos[startLine] >= 6 || Pos[startLine] <= 9) && (deltaX <= HarfDeltaX && deltaX >= -HarfDeltaX)) {

if((Pos[startLine] >= 6 && Pos[startLine] <= 9) && (deltaX <= HarfDeltaX && deltaX >= -HarfDeltaX) && deltaY >=6) {

総合走行の訂正(2)

- 2値化の関数内
  - 元のプログラム

```
//    iShikii = vWHITE * vDataAve / vDataAve10;
//    if( iShikii > MAX_SHIKII )  iShikii = MAX_SHIKII;
```

  - 変更後のプログラム

```
iShikii = (vDataAve10 * 2 -10 ) * vDataAve / vDataAve10;
if( iShikii > (vDataAve10 * 2 - 10 ) )  iShikii = (vDataAve10 * 2 - 10);
```

今後の課題

- マトリクスの作り方を工夫する
  - 偶数列だと、ど真ん中を表現できない。
  - 奇数列(15列ぐらい)に変更して位置情報で中心がでるようにする。
- 遠方のラインは明るさが不足する
  - 明るさの平均をとるラインを変える
  - ソフトの改良をする
    - 各ラインごとに閾値を変えて正しく判断できるようにする。

今後の課題

- 左回りは得意であるが、右回りは不得意
  - 左はR450の定常旋回可能
  - 右はR450の定常旋回ができない(半周回ったぐらいで落ちる。)
  - おそらくハンドル角を求めるプログラムが悪い
- 同じような条件でもうまく走らないときがある。
  - 画像処理は難しい。
  - だから面白い。
- 画像処理を広めるためには、入手可能でH8やR8で制御できるカメラを探す必要がある。

最後に

- 第4回電通大杯では時間を競えるようになると面白い。
  - 良い制御方法や処理方法があれば、情報共有しましょう。

ご清聴ありがとうございました

- 発表者Blog
  - http://kg5mcr.blog130.fc2.com/