Ver 0.2

スクリプトの仕様

谷川郁太
3bjnm015@mail.tokai-u.jp

スクリプトについて

- 教材では、スクリプトに通信の処理を書くことで、Kobukiを制御する
- スクリプトの文法はC言語に似ている (異なる点は後述)
- 各種コマンドが用意されており、それらを用いることで、直接通信プログラムを書く必要が無くなる(コマンドの一覧は後述)

コマンド一覧

移動コマンド

- void Forward()
  - まっすぐ前進
- void Backward()
  - まっすぐ後進
- void TurnLeft()
  - その場で左に曲がる
- void TurnRight()
  - その場で右に曲がる
- void Stop()
  - その場に停止する

移動コマンド

- void BaseControl(short speed, short radius)
  - Kobukiに元から用意されている移動用のコマンドを直接実行する
  - speed : 速度 [mm/s]
    - 正の値で直進、負の値で後進
  - radius : 半径(曲がる量) [mm]
    - 正の値で正面から見て左方向、負の値で右方向に曲がる
    - 与える値が0に近いほどよく旋回する
    - 0が与えられた場合は直進する

設定コマンド

- void SetSpeed (int speed)
  - 直進速度の設定 (0～100 [10mm/s])
- void SetTurnSpeed (int speed)
  - 旋回速度の設定 (0～100 [10mm/s])
- void SetLEDColor (int num, char color)
  - LEDの色設定
  - num はLEDの番号で、1か2
  - colorは ‘N’(無し), ‘R’(赤), ‘G’(緑), ‘Y’(黄)

サウンドコマンド

- void Sound_DO() : ドの音を鳴らす
- void Sound_RE() : レの音を鳴らす
- void Sound_MI() : ミの音を鳴らす
- void Sound_FA() : ファの音を鳴らす
- void Sound_SO() : ソの音を鳴らす
- void Sound_RA() : ラの音を鳴らす
- void Sound_SI() : シの音を鳴らす
- void Sound_DO_2() : ドの音を鳴らす
  - 末尾に”_Sharp”を付けると、ド#、レ#等が出せる
  - サウンドコマンドは、走行しながら実行できない

システムコマンド

- void Sleep(int ms)
  - 指定された時間(ミリ秒)、次のステップに移るのを待つ
- void Exit()
  - プログラムを終了する
  - このコマンドが実行されずにスクリプトの最終行まで到達すると最初の行に戻り、再度実行される
- long ConnectTime
  - 接続開始からの経過時間を表す変数

受信データ1

- int RightBumper
- int LeftBumper
- int CentralBumper
  - それぞれ右、左、真ん中のバンパーを示す
  - 1:押されている, 0:押されていない
- int Button0
- int Button1
- int Button2
  - それぞれボタン0,1,2を示す
  - 1:押されている, 0:押されていない

受信データ2

- int RightWheelDrop
- int LeftWheelDrop
  - それぞれ右、左のホイールが浮いているかどうか
  - 1:浮いている, 0:浮いてない
- ushort LeftEncoder
- ushort RightEncoder
  - 左右モータのエンコーダ値
  - 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm

関数一覧

出力関数

- void OutputString(string str)
  - 渡された文字列を出力
- void OutputInteger(int i)
  - 渡された整数を出力
- void OutputFloat(double f)
  - 渡された浮動小数を出力

数学関連

- double Sin(double a)
  - 角度a(ラジアン)のsinを返す
  - 使用例 : Sin(30 * 3.141592… / 180); // 0.5
- double Cos(double d)
  - 角度d(ラジアン)のcosを返す
- double Tan(double a)
  - 角度a(ラジアン)のtanを返す

乱数

- int GetRandomValue()
  - 0～2,147,483,647の範囲の乱数を返す
  - 使用例 : GetRandomValue() % 101; // 0～100

C言語との違い

C言語との違い

- Cの標準ライブラリは使えない (printfなど)
- staticなローカル変数が使えない(代わりにグローバル変数を使う)
- #include, #defineが使えない
- 関数を作る際、プロトタイプ宣言が不要
- 型名が異なる (後述)
- enumの使い方が異なる (後述)
- 配列の作り方が異なる (後述)
- 無限ループの書き方が異なる (後述)

スクリプトの型名一覧

型名	C言語の型	概要
sbyte	char	符号付き8バイト
byte	unsigned char	符号なし8バイト
short	short	符号付き16バイト
ushort	unsigned short	符号なし16バイト
int	int	符号付き32バイト
uint	unsigned int	符号なし32バイト
long		符号付き64バイト
ulong		符号なし64バイト
char		Unicode 16 ビット文字

enumの違い

- C言語の場合

```
enum State{
    Stop,
    Run,
};
…
enum State state = Stop;
switch(state){
    case Stop:
        …
    case Run:
        …
}
```

- スクリプト言語の場合

```
enum State{
    Stop,
    Run,
}
…
State state = State.Stop;
switch(state){
    case State.Stop:
        …
    case State.Run:
        …
}
```

enumの要素を用いる際は型名が必要となる

← セミコロン不要

↓ enum不要

↑ 型名が必要

↑ 型名が必要

↑ 型名が必要

配列の作り方

- 要素数5のint型配列arrayを作る際の違い

- C言語の場合
  - int array[5]; // 初期化無し
  - int array[5] = {1, 2, 3, 4, 5}; // 初期化あり
- スクリプト言語の場合
  - int[] array = new int[5]; // 初期化無し
  - int[] array = new int[5] {1, 2, 3, 4, 5}; // 初期化あり

無限ループの書き方

- C言語の場合

```
while(1) {
    // 処理
}
```

- スクリプト言語の場合

```
While(true) {
    // 処理
}
```