第19回 数理科学コンクール
ロボットの部
K-Junior 簡易マニュアル

2016年7月16日, 17日

作成　千葉大学 K-Junior の会
浅田 展好
經田 原弘
楠 真輝
黒岩 亮
丹治 香織
細谷 健登

1　概要

ロボットの部では, 参加者が C 言語によってプログラムを作成する．そして, 作成したプログラムをロボットにアップロードすることによって, ロボットを動作させる．図 1.1 のロボットを用いる．また、ロボットには図 1.2 で示す IR センサー (赤外線センサー) が搭載されている．

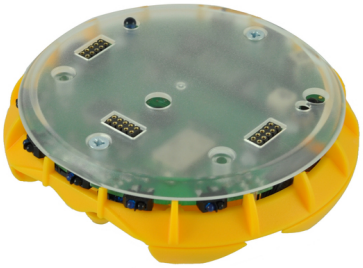

図 1.1: K–junior．作成したプログラムを用いてこのロボットを動作させる．ロボットの側面と底面にはセンサーが付いている．ビープ音を出すことができる．さらに, 上面には, ディスプレイやものをつかむアーム (グリッパー) などの付属品を取り付けることができる．[1] より引用.

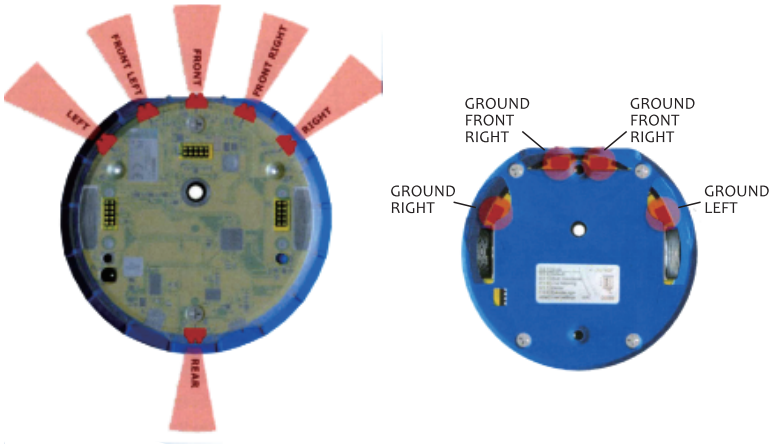

図 1.2: K–Junior の IR センサー．合計 10 個の IR センサーが搭載されている．[2] より引用.

この資料では, ロボットを動作させるために必要な

1. プログラムの基礎
2. ロボットの組み込み関数
3. サンプルプログラム

を説明する.

2　プログラムの基礎

プログラムとは, プログラミング言語[1] において定義された命令の集まりである[2]. 命令の組み合わせによって多様なプログラムを記述できる.

したがって, プログラムを作成するためには, 命令の組み合わせ方が重要となる. 命令を組み合わせる方法をプログラムの構造と呼ぶことにする. プログラムの構造は,

1. 順次
2. 分岐
3. 反復

からなる.

2.1　順次

順次とは, プログラムを 1 行目から終わりまで, 順番に 1 命令ずつ実行することである[3]. 3 つの処理が順次であるとき, 図 2.1 のように上から順番に 1 命令ずつ実行する.

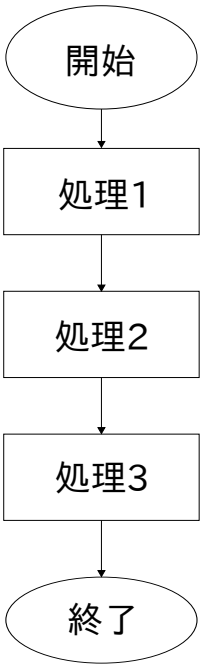

図 2.1: 順次の処理. 処理 1, 2, 3 の順番に処理をする.

[1] 今回は C 言語.

[2] プログラミング言語において, 命令は厳密である. したがって, 命令を入力するときに, 1 文字でも定義と異なると作成したプログラムは動作しない.

[3] C 言語の場合, main 関数 (main(void) { . . .} と書いてある部分) において { . . .} 内に記述された命令を上から順番に実行する.

2.2 分岐

分岐とは, 条件にしたがい, 命令を実行するかどうかを決定することである．図 2.2 は分岐の処理を示す．つまり, 条件をみたすならば, 処理 1 を実行する．一方で, 条件をみたさない場合, 処理 2 を実行する．

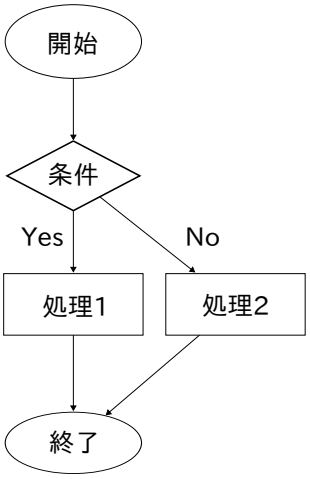

図 2.2: 分岐の処理．条件をみたすならば, 処理 1 を実行する．みたさないならば, 処理 2 を実行する．

C 言語では, **if** 文によって分岐を行なう．if 構文は下記の 3 種類ある．

```
// if 構文 1
if( 条件 ) {
   処理
}

// if 構文 2
if( 条件 ) {
   処理 1
} else {
   処理 2
}

// if 構文 3
if( 条件 1 ) {
    処理 1
} else if( 条件 2 ) {
    処理 2
} else {
    処理 3
}
```

if 構文 2 が図 2.2 に対応する．if 構文 1 は図 2.2 において, 処理が 1 つのみの場合である．if 構文 3 は図 2.2 において, 条件が 2 つ, 処理が 3 つの場合である [4].

2.3 反復

反復とは, 命令を繰り返し実行することである．一般的に, 命令の実行回数を決定するために, 条件を追加する．しかし, ロボットを制御する場合, ロボットを動作させ続けたいため, 無限に反復を繰り返すこともある．図 2.3 が反復の処理の例である．図 2.3 では, もし条件をみたすならば, 処理 1 を実行後, 再び条件の判断を行なう．条件がつねに成り立つ場合, 無限に処理 1 を繰り返す.

C 言語では, **for** 文, **while** 文を利用して反復を行なう．for 文, while 文の構文は, それぞれ

```
// for 構文
   for( 初期値; 条件; 初期値として用いる変数の変化量 ) {
       処理
   }

// while 構文
   while( 条件 ) {
       処理
   }
```

である.

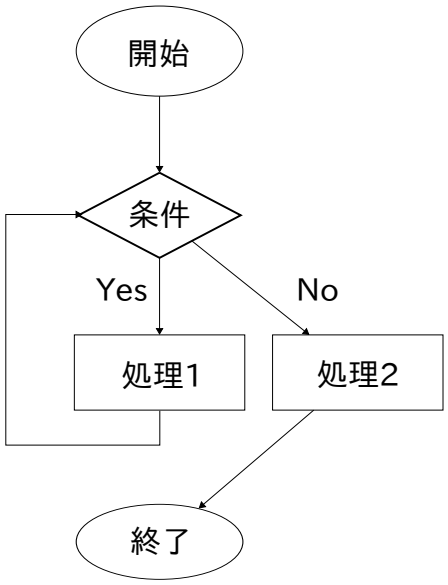

図 2.3: 反復の処理．条件が成り立つ間ずっと, 処理 1 を繰り返す．条件が成り立たないならば, 処理 2 を実行する.

[4]条件は順序が重要となる．例えば, if 構文 3 において, 条件 1 と条件 2 が同時に成り立つ場合, 処理 1 のみが実行される．これは, プログラムは順次に実行されるために生じる.

表 2.1: 算術演算子．計算するために用いる．

演算子	意味
a + b	a と b の加法
a − b	a と b の減法
a ∗ b	a と b の乗法
a / b	a と b の除法
a % b	a / b の剰余

表 2.2: 関係演算子．条件を記述するために用いる．

演算子	意味
a == b	a と b が等しい
a ! = b	a と b が等しくない
a < b	a は b より小さい
a <= b	a は b 以下である
a > b	a は b より大きい
a >= b	a は b 以上である

2.4 演算子

演算子とは, 計算や条件を記述するために用いる記号である．計算に用いる算術演算子と条件の記述に用いる関係演算子に大別される．

2.4.1 算術演算子

算術演算子には, 四則演算 (加法, 減法, 乗法, 除法) と剰余の演算[5] が定義されている．表 2.1 のように記述する．

2.4.2 関係演算子

関係演算子として, 等号 (=), 等号否定 (≠), 不等号 (>, <), 等号つき不等号 (≥, ≤) が定義されている．プログラム上では表 2.2 のように関係演算子を記述する．表中の a, b は変数である．なお, 等号 (=) 1 つでは, 別の変数への値の受け渡し (代入) を意味する[6].

条件として用いた式が 0 でないときに, 条件が成り立つ．一方で, 条件がゼロのときには条件は成り立たない[7]．このため, 表 2.2 の関係演算子は, 成り立つときに 1 を返し, 成り立たないときに 0 を返す.

[5]正式にはモジュロ演算子とよぶ.
[6]自身へ代入することもある.
[7]したがって, 条件が負数である場合, 条件は成り立つ.

条件は同時に複数を考慮できる．複数の条件を考える場合, 条件どうしの関係を決定する必要がある．つまり, 条件が同時に成り立つ, または, どちらかが成り立つことを決定する必要がある．C言語では,

```
// 条件 1 と条件 2 が同時に成り立つならば，処理を実行する．
   if( ( 条件１ ) && ( 条件２ ) ) {
       処理
   }

// 条件 1 と条件 2 のどちらか一方が成り立つならば，処理を実行する．
   if( ( 条件１ ) || ( 条件２ ) ) {
       処理
   }
```

と記述する．

3 基礎関数

K-Junior を動作させる基礎関数を紹介する. 関数は, 指定された入力にしたがい出力を返す, 命令の集まりである. 基礎関数として, K-Junior を動作させる命令が定義されている.

下記にある基礎関数は, C 言語自体には定義されていない. ヘッダファイル “KJunior.h” や “hemigripper.h” に記述されている [8].

3.1 基礎関数のリファレンス

3.1.1 ロボットの設定関数

1. void K-junior_init(void);
   →ロボットの初期化.

2. void K-junior_config_auto_refresh_sensors(int1 Bit);
   → IR センサの読み方の設定. (MANUAL か REFRESH) を指定.

3. void K-junior_config_auto_refresh_tv_remote(int1 Bit);
   → TV リモコン (IR センサー) の初期化設定. (MANUAL か REFRESH) を指定.

4. void K-junior_config_rs232_control(int1 Bit);
   → “シリアルリモートコントロール” モードの許可・非許可. (DISABLE か ENABLE) を指定.

5. void K-junior_config_tv_remote_control(int1 Bit);
   → “TV リモートコントロール” モードの許可・非許可. (DISABLE か ENABLE) を指定.

3.1.2 “フラグ” の読み取り関数

1. int1 KJunior_flag_sensors_refreshed(void);
   → IR センサの値がリフレッシュされたかどうかを調べる.

2. void KJunior_flag_sensors_reset(void);
   → IR センサのリフレッシュレートフラグを 0 にセットする.

3. int1 KJunior_flag_rs232_filtering(void);
   → “シリアルリモートコントロール” モードがオンかどうかを調べる.

4. int1 KJunior_flag_tv_data_refreshed(void);
   → “TV リモートコントロール” の値がリフレッシュされたかどうかを調べる.

5. int1 KJunior_flag_tv_data_emitting(void);
   → TV リモコン (IR センサー) の情報が発信され続けているか調べる.

6. void KJunior_flag_tv_data_reset(void);
   → “TV リモートコントロール” のリフレッシュフラグを 0 にセットする.

[8] 実際には, ヘッダファイルには関数の宣言と別の C プログラムがインクルードされている. 別の C プログラムにおいて, 関数の処理が記述されている.

表 3.1: KJunior_get_switch_state の返り値とスイッチの状態の関係

返り値	ビット表現	スイッチ 1	スイッチ 2	スイッチ 3
0	000	OFF	OFF	OFF
1	001	OFF	OFF	ON
2	010	OFF	ON	OFF
3	011	OFF	ON	ON
4	100	ON	OFF	OFF
5	101	ON	OFF	ON
6	110	ON	ON	OFF
7	111	ON	ON	ON

3.1.3 ロボットの I/O 関数

1. signed int16 KJunior_get_proximity(char Sensor);
   → (FRONT, FRONTLEFT, FRONTRIGHT, LEFT, RIGHT, REAR, GROUNDLEFT, GROUNDRIGHT, GROUNDFRONTLEFT, GROUNDFRONTRIGHT) の 10 センサーがある. 指定位置の名前を Sensor にそのまま入れる.
   該当センサを用いて距離を計測して値を返す.

2. signed int16 KJunior_get_brightness(char Sensor);
   → (FRONT, FRONTLEFT, FRONTRIGHT, LEFT, RIGHT, REAR, GROUNDLEFT, GROUNDRIGHT, GROUNDFRONTLEFT, GROUNDFRONTRIGHT) の 10 センサーがある. 指定位置の名前を Sensor にそのまま入れる.
   該当センサを用いて明るさを計測して値を返す.

3. unsigned char KJunior_get_switch_state( );
   → 3 スイッチ (モードスイッチ) の状態を返す. (0 から 7) の値を返す.
   値を 2 進数表記したときのそれぞれのビットがそれぞれのスイッチの状態を表す. 表 3.1 参照.

4. unsigned char KJunior_get_tv_data(void);
   → TV リモコン (IR センサー) から受信したデータを返す. (0 から 63) の値を返す. 例えば, TV リモコンのボタン 2 を受信した場合, 返り値が 2 となる.

5. unsigned char KJunior_get_tv_addr(void);
   → TV リモコン (IR センサー) で受信中のアドレス (チャンネル) を返す. (0 から 31) の値を返す.

6. void KJunior_send_tv_value(unsigned char addr, unsigned char data);
   → K-Junior の IR エミッターを用いてアドレス (チャンネル) を指定してデータを送る. アドレスは (0 から 31), データは (0 から 63) を指定できる. この値を KJunior_get_tv_data ( ) 等で受け取ることができる.

7. void KJunior_set_speed(signed int8 LeftSpeed, signed int8 RightSpeed);
   → K-Junior のスピードを左右の車輪それぞれで設定する.
   なお, 設定値は −20 から 20 の値を設定する.

8. void KJunior_beep(unsigned char Freq);
   → ビープ音の周波数を設定する. (0 から 48) の値を入力する. 0 は OFF, 1 は 42kHz, 48 は 1.975kHz の周波数のビープ音を表す.

9. void KJunior_led_left(int1 State);
   →側面左側の LED の ON/OFF を設定する. ON か OFF を指定する.

10. void KJunior_led_frontleft(int1 State);
    →前左側の LED の ON/OFF を設定する. ON か OFF を指定する.

11. void KJunior_led_frontright(int1 State);
    →前右側の LED の ON/OFF を設定する. ON か OFF を指定する.

12. void KJunior_led_right(int1 State);
    →右の LED の ON/OFF を設定する. ON か OFF を指定する.

13. void KJunior_led_onoff(int1 State);
    →電源スイッチの LED の ON/OFF を設定する. ON か OFF を指定する.

14. void KJunior_manual_refresh_sensors( );
    →手動で IR センサのリフレッシュを行う.

3.1.4 遅延・時間関数

1. void KJunior_delay_s(unsigned int Delay);
   → (0 から 65535) 秒間遅延させる. 次の命令は (指定した秒) 後に実行される.

2. void KJunior_delay_ms(unsigned int Delay);
   → (0 から 65535) ミリ秒間遅延させる.

3. void KJunior_delay_us(unsigned int Delay);
   → (0 から 65535) マイクロ秒間遅延させる.

4. unsigned int32 KJunior_get_time(void);
   → K–Junior が起動してからの経過時間をミリ秒で返す.

5. void KJunior_set_time(unsigned int32 Time);
   →指定したミリ秒で, K–Junior が起動してからの経過時間を強制的に書き換える.

3.2 LCD ディスプレイモジュールの接続と操作関数

LCD ディスプレイは 2 行で構成され, それぞれ 12 バイトまで表示できる.
次のように定義されている.

- char Line1[12], Line2[12];

また 3 ボタンがあり, 次のように定義されている.

- int1 SW1, SW2, SW3;

LCD ディスプレイ接続時には下記の関数を利用できる.

1. void HemLCD_Init(void);
   →モジュールの初期化. 使用する際は一番先頭にもってくること.

2. void HemLCD_Read_Version(void);
   →ファームウェアのバージョンを読み込む.

3. void HemLCD_Set_Backlight(unsigned char Value);
   →ディスプレイのバックライトの明るさを設定する. (0 から 255) までの値を渡す. 255 が輝度最大で 0 がバックライトなし.

4. void HemLCD_Set_Contrast(unsigned char Value);
   →ディスプレイのコントラスト (明るい部分と暗い部分の輝度の差) を設定する. (0 から 255) までの値を渡す. 255 がコントラスト最大で 0 がコントラストなし.

5. void HemLCD_Clear_Screen(void);
   →スクリーン全体をクリアする.

6. void HemLCD_Clear_Line1(void);
   → 1 行目をクリアする.

7. void HemLCD_Clear_Line2(void);
   → 2 行目をクリアする.

8. void HemLCD_Read_Interruptors(void);
   → 3 つのボタンの状態を読む.

9. HemLCD_Line1_Left(char* Data);
   → Line1[] もしくは Line2[] を渡す.
   1 行目に左詰めで表示する.

10. HemLCD_Line1_Centered(char* Data);
    → Line1[] もしくは Line2[] を渡す.
    1 行目に中央合わせで表示する.

11. HemLCD_Line1_Right(char* Data);
    → Line1[] もしくは Line2[] を渡す.
    1 行目に右詰めで表示する.

12. HemLCD_Line2_Left(char* Data);
    → Line1[] もしくは Line2[] を渡す.
    2 行目に左詰めで表示する.

13. HemLCD_Line2_Centered(char* Data);
    → Line1[] もしくは Line2[] を渡す.
    2 行目に中央合わせで表示する.

14. HemLCD_Line2_Right(char* Data);
    → Line1[] もしくは Line2[] を渡す.
    2 行目に右詰めで表示する.

3.3 “グリッパー” のモジュールの接続と操作関数

1. void HemGripper_Init(void);
   →グリッパーの初期化.

2. Unsigned int16 HemGripper_Read_Arm_Position(void);
   →グリッパーの腕部分の位置を現在読み込む.
   値は (0 から 7000) の間である.
   なお, 0 が下げきった状態で 7000 があげきった状態.

3. Unsigned int16 HemGripper_Read_Gripper_Position(void);
   →グリッパーのつかむ部分の位置を現在読み込む.
   値は (0 から 5100) の間である.
   なお, 0 が閉じきった状態で 5100 が開ききった状態.

4. Unsigned int16 HemGripper_Read_Arm_Consign(void);
   →グリッパーの腕部分を動かす目的地の値を返す.
   値は (0 から 7000) の間である.
   なお, 0 が下げきった状態で 7000 があげきった状態.

5. Unsigned int16 HemGripper_Read_Gripper_Consign(void);
   →グリッパーのつかむ部分を動かす目的地の値を返す.
   値は (0 から 5100) の間である.
   なお, 0 が閉じきった状態で 5100 が開ききった状態.

6. Unsigned char HemGripper_Read_Arm_Speed(void);
   →腕部分の可動速度を返す. この値は (0 から 255) の間である.

7. Unsigned char HemGripper_Read_Gripper_Speed(void);
   →つかむ部分の可動速度を返す. この値は (0 から 255) の間である.

8. Unsigned char HemGripper_Read_Arm_Disable(void);
   →変化のないポジションに到達した際, 腕部分の ON/OFF のフラグを読む.

9. Unsigned char HemGripper_Read_Arm_Disable(void);
   →変化のないポジションに到達した際, 腕部分の ON/OFF のフラグを読む.

4　サンプルプログラム

K-Junior のサンプルプログラムのコードを示す. サンプルプログラムとして,

- 前進してその後後退するプログラム
- 左折してその後右折するプログラム
- 正方形を描くように回るプログラム
- グリッパーで物を上げ下げするプログラム
- 物を拾って所定の位置に置くプログラム

を示す.

4.1　前進してその後後退するプログラム

```
void main(void) {
  int i = 0;
  int speed = 10;

  // ロボットの初期化
  KJunior_init();

  //*******************************Main loop*************************************//
  While (1) {
    if((SerialCommandOK == 1) && (Enable_RS232_Control == 1)) {
      SerialCommandHandler();

      // 前進する.
      KJunior_set_speed(speed, speed);
      KJunior_delay_s(2);

      // 変数 speed を (-1) 倍する. つまり, speed を逆方向にする.
      speed *= -1;
      // 前進と逆方向に進む. つまり, 後進する.
      KJunior_set_speed(speed, speed);

      // 2 秒間待機する
      KJunior_delay_s(2);
      Kjunior_set_speed(0, 0);

      break;
    }
  }
}
```

4.2 左折してその後右折するプログラム

```
void main(void) {
  int i = 0;
  int speed = 10;

  // ロボットの初期化
  KJunior_init();

  //*******************************Main loop*************************************//
  While (1) {
    if((SerialCommandOK == 1) && (Enable_RS232_Control == 1)) {
      SerialCommandHandler();

      // 2 秒間左に曲がる.
      KJunior_set_speed(0, speed);
      KJunior_delay_s(2);

      // 2 秒間右に曲がる.
      KJunior_set_speed(speed, 0);
      KJunior_delay_s(2);
      Kjunior_set_speed(0, 0);
      break;
    }
  }
}
```

4.3 正方形を描くように回るプログラム

```
void main(void) {
  int i = 0;

  // ロボットの初期化
  KJunior_init();

  //*******************************Main loop*************************************//
  While (1) {
    if((SerialCommandOK == 1) && (Enable_RS232_Control == 1)) {
      SerialCommandHandler();
      for(i = 0; i < 4; ++i) {
        // speed の引数は, 数値を直接入力してもよい.
        KJunior_set_speed(10, 10);
        KJunior_delay_s(3);
```

```
        KJunior_set_speed(0, 17);
        KJunior_delay_s(1);
      }
      Kjunior_set_speed(0, 0);
      break;
    }
  }
}
```

4.4 グリッパーで物を上げ下げするプログラム

```
void main(void) {
  // ロボットの初期化
  KJunior_init();

  // グリッパーの初期化
  HemGripper_Init();

  //*****************************Main loop*************************************//
  While (1) {
    if((SerialCommandOK == 1) && (Enable_RS232_Control == 1)) {
      SerialCommandHandler();
      HemGripper_Set_Arm_Speed(50);
      HemGripper_Set_Gripper_Speed(50);
      KJunior_delay_s(2);

      //グリッパーを開く
      HemGripper_Set_Gripper_Consign(4000);
      KJunior_delay_s(2);

      //グリッパーを閉じる
      HemGripper_Set_Gripper_Consign(500);
      KJunior_delay_s(2);

      //アームを上げる
      HemGripper_Set_Arm_Consign(6000);
      KJunior_delay_s(2);

      //アームを下げる
      HemGripper_Set_Arm_Consign(100);
      KJunior_delay_s(2);

      //グリッパーを開く
```

```
      HemGripper_Set_Gripper_Consign(4000);
      KJunior_delay_s(2);
      break;
    }
  }
}
```

4.5 物を拾って所定の位置に置くプログラム

```
void main(void) {
  int i = 0;
  int speed = 10;

  // ロボットの初期化
  KJunior_init();

  // グリッパーの初期化
  HemGripper_Init();

  //*****************************Main loop*************************************//
  While (1) {
    if((SerialCommandOK == 1) && (Enable_RS232_Control == 1)) {
      SerialCommandHandler();

      HemGripper_Set_Arm_Speed(50);
      HemGripper_Set_Gripper_Speed(50);
      HemGripper_Set_Gripper_Consign(500);
      KJunior_delay_s(2);
      HemGripper_Set_Gripper_Consign(4000);
      while(HemGripper_Read_Gripper_Consign() < 3900) {
      }
      KJunior_delay_s(2);

      // 前進する
      KJunior_set_speed(speed, speed);

      // 2 秒間待機する
      for(i = 0; i < 20; ++i) {
        KJunior_delay_ms(100);
      }

      KJunior_set_speed(0, 0);
      HemGripper_Set_Gripper_Consign(500);
```

```
      KJunior_delay_s(2);
      HemGripper_Set_Arm_Consign(6000);
      KJunior_delay_s(2);

      speed *= -1;

      // 後進する
      KJunior_set_speed(speed, speed);

      // 2 秒間待機する
      KJunior_delay_s(2);
      KJunior_set_speed(0, 17);
      KJunior_delay_s(1);

      speed *= -1;

      KJunior_set_speed(speed, speed);

      // 2 秒間待機する
      KJunior_delay_s(2);
      KJunior_set_speed(0, 0);
      HemGripper_Set_Arm_Consign(100);
      KJunior_delay_s(2);
      HemGripper_Set_Gripper_Consign(4000);
      KJunior_delay_s(2);

      speed *= -1;

      KJunior_set_speed(speed, speed);

      KJunior_delay_s(2);

      KJunior_set_speed(0, -17);
      KJunior_delay_s(1);
      KJunior_set_speed(0, 0);
      break;
    }
  }
}
```

参考文献

[1] K-Junior V2, http://www.k-team.com/mobile-robotics-products/k-junior

[2] K-Junior V2 Pack (Autonomous programmable evolutionary robot) TECHNICAL MANUAL,

[3] B.W. カーニハン and D.M. リッチー (石田晴久 訳), プログラミング言語 C 第 2 版 ANSI 規格準拠, 共立出版, 1989