総合実験 1

LEGO 回転型倒立振子

氏名 佐々木佑（1935098T）

提出日 2021 年 6 月 23 日

１．実験の目的

「センサー情報処理－アクチュエータ」系による外界とのインタラクションを伴う知能行動体の設計と 製作を体験する．実験を通し、関連講義科目の重要性，必要性，意義を体感する．そして，自分で 考え工夫すること，設計製作すること，グループで協力することの楽しさを体験する．

２．ロボットの説明

他の生徒のロボットと異なると思われる特徴は５つ。

１．光センサを前方に２つ、後方に１つ備える。

２．距離センサを前方に２つ備える。

３．角度センサ（フィールドに対するロボットの角度を取得）を備える。

４．アームを前方に２つ備える。

以下がロボットの定義である。

```
Robot {
  translation -0.40235 0.0398085 0.448802
  rotation 0.0017674855636001065 0.9999780244747116 -0.006389566686054809
2.8557611165062147
  children [
    InertialUnit {
      translation 0 0.042 0
      children [
        DEF INNER Shape {
          appearance PBRAppearance {
            baseColor 0 1 0.498039
            roughness 1
            metalness 0
          }
          geometry Cylinder {
            height 0.01
            radius 0.01
          }
        }
      ]
```

```
      name "inner"
      boundingObject USE INNER
      physics Physics {
      }
    }
    Solid {
      translation -0.0682849 0.012 0.0975769
      rotation 0.5000000690830143 0.7071080976983041 -0.499998069082738 -
3.1415853071795863
      children [
        Shape {
          appearance PBRAppearance {
          }
          geometry DEF ARM Cylinder {
            height 0.15
            radius 0.005
          }
        }
      ]
      name "solid(5)"
      boundingObject USE ARM
      physics Physics {
      }
    }
    DistanceSensor {
      translation 0.03 -0.015 0.05
      rotation 0 1 0 -1.2708
      children [
        DEF DS_SENSOR Shape {
          appearance PBRAppearance {
            baseColor 0 0 1
            roughness 1
            metalness 0
          }
          geometry Box {
            size 0.01 0.01 0.01
```

```
        }
      }
    ]
    name "ds_left"
    boundingObject USE DS_SENSOR
    physics Physics {
    }
}
DistanceSensor {
    translation -0.03 -0.015 0.05
    rotation 0 1 0 -1.8708
    children [
      DEF DS_SENSOR Shape {
        appearance PBRAppearance {
          baseColor 0 0 1
          roughness 1
          metalness 0
        }
        geometry Box {
          size 0.01 0.01 0.01
        }
      }
    ]
    name "ds_right"
    boundingObject USE DS_SENSOR
    physics Physics {
    }
}
LightSensor {
    translation 0 0.042 -0.042
    rotation 0 1 0 1.59
    name "back"
    lookupTable [
      0 0 0
      4 1024 0
    ]
```

```
}
Gyro {
  translation 0 0.042 0
  lookupTable [
    4 1024 0
  ]
}
Solid {
  translation 0.0824266 0.0120003 0.097574
  rotation 0.28108510281603044 -0.6785992482197749 0.6785972482190434 2.59357
  children [
    Shape {
      appearance PBRAppearance {
      }
      geometry DEF ARM Cylinder {
        height 0.15
        radius 0.005
      }
    }
  ]
  name "solid(4)"
  boundingObject USE ARM
  physics Physics {
  }
}
LightSensor {
  translation 0.02 0.042 0.042
  rotation 0 1 0 -1.07
  name "ls1"
  lookupTable [
    0 0 0
    4 1024 0
  ]
}
LightSensor {
  translation -0.02 0.045 0.042
```

```
      rotation 0 1 0 -2.07
      name "ls0"
      lookupTable [
        0 0 0
        4 1024 0
      ]
    }
    DEF BODY Shape {
      appearance PBRAppearance {
        baseColor 1 0 0
        roughness 1
        metalness 0
      }
      geometry Box {
        size 0.1 0.06 0.1
      }
    }
    HingeJoint {
      jointParameters HingeJointParameters {
        position 2409.756664438725
        anchor -0.606 0 0
      }
      device [
        RotationalMotor {
          name "wheel2"
        }
      ]
      endPoint DEF WHEEL2 Solid {
        translation -0.06013416313823039 0.0005747212824638923
0.00013172678965499735
        rotation -0.12102501339737204 0.11899381816315739 0.9854914598164305
1.586472604814353
        children [
          DEF WHEEL Shape {
            appearance PBRAppearance {
              baseColor 0 0 0
```

```
                roughness 1
                metalness 0
              }
              geometry Cylinder {
                height 0.02
                radius 0.04
              }
            }
          ]
          name "solid(1)"
          boundingObject USE WHEEL
          physics Physics {
          }
        }
      }
      HingeJoint {
        jointParameters HingeJointParameters {
          position 2654.1212897863497
          anchor 0.06 0 0
        }
        device [
          RotationalMotor {
            name "wheel1"
          }
        ]
        endPoint DEF WHEEL1 Solid {
          translation 0.06010467964266675 6.807536285244513e-05
3.6080659947722033e-05
          rotation -0.33704546135062097 0.35755036894334413 0.8709524043548105
1.77240237713873
          children [
            DEF WHEEL Shape {
              appearance PBRAppearance {
                baseColor 0 0 0
                roughness 1
                metalness 0
```

```
            }
            geometry Cylinder {
              height 0.02
              radius 0.04
            }
          }
        ]
        boundingObject USE WHEEL
        physics Physics {
        }
      }
    }
    Solid {
      translation 0 -0.03 -0.04
      children [
        DEF CASTER Shape {
          appearance PBRAppearance {
            baseColor 0 0 0
            metalness 0
          }
          geometry Sphere {
            radius 0.01
          }
        }
      ]
      name "solid(2)"
      boundingObject USE CASTER
      physics Physics {
      }
    }
    Solid {
      translation 0 -0.03 0.04
      children [
        DEF CASTER Shape {
          appearance PBRAppearance {
            baseColor 0 0 0
```

```
            metalness 0
          }
          geometry Sphere {
            radius 0.01
          }
        }
      ]
      name "solid(3)"
      boundingObject USE CASTER
      physics Physics {
      }
    }
  ]
  boundingObject USE BODY
  physics Physics {
  }
  controller "test_go_light"
  controllerArgs [
    ""
  ]
}
```

３．フロチャート

フローは大きく４つに分かれる

１．光センサを使ってボールにある程度近づく

２．距離センサを使ってボールをアームの内側に置く

３．角度センサを使ってロボットをゴールの方向へ回転させる

４．ゴールまでボールを運ぶ。

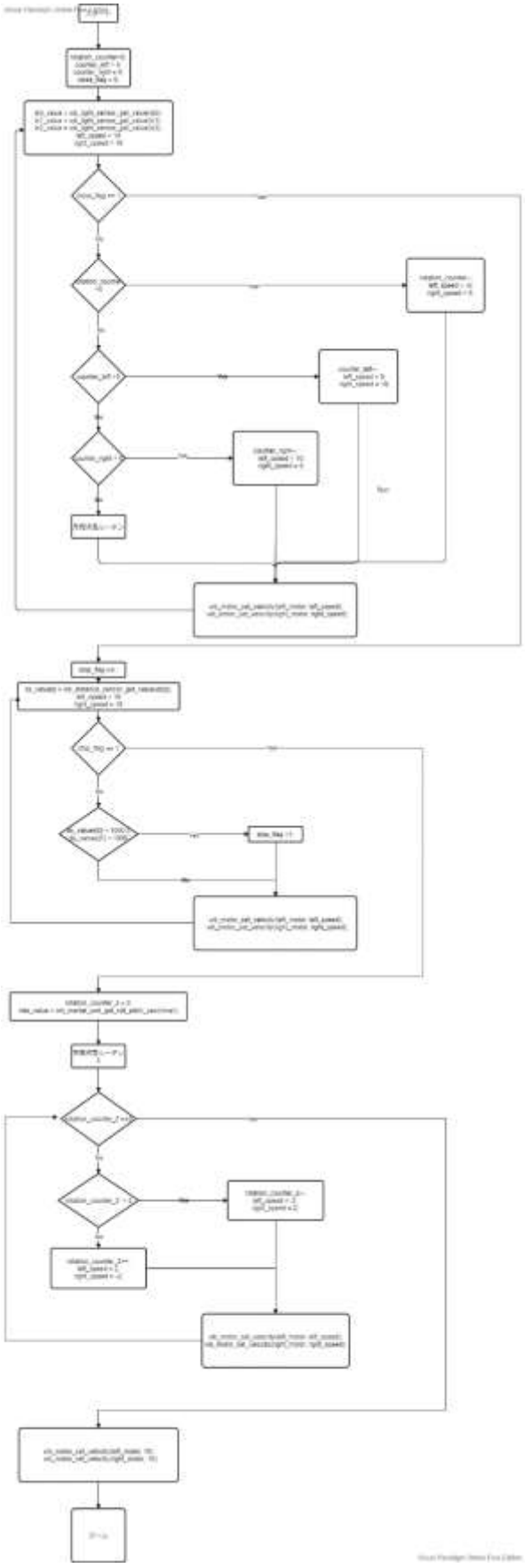

図１　メインルーチン

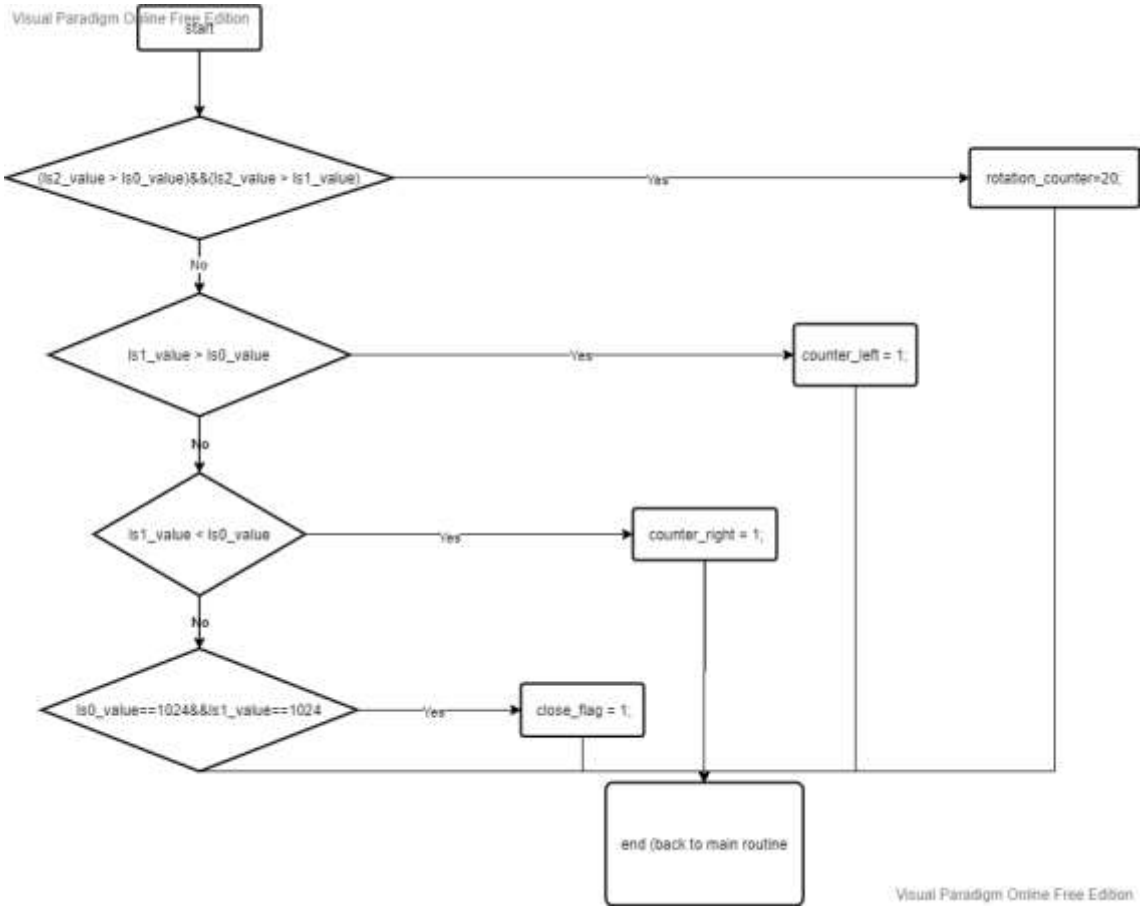
Visual Paradigm Online Free Edition
start
(ls2_value > ls0_value)&&(ls2_value > ls1_value)
Yes
rotation_counter=20;
No
ls1_value > ls0_value
Yes
counter_left = 1;
No
ls1_value < ls0_value
Yes
counter_right = 1;
No
ls0_value==1024&&ls1_value==1024
Yes
close_flag = 1;
end (back to main routine
Visual Paradigm Online Free Edition

図２　方向決定ルーチン

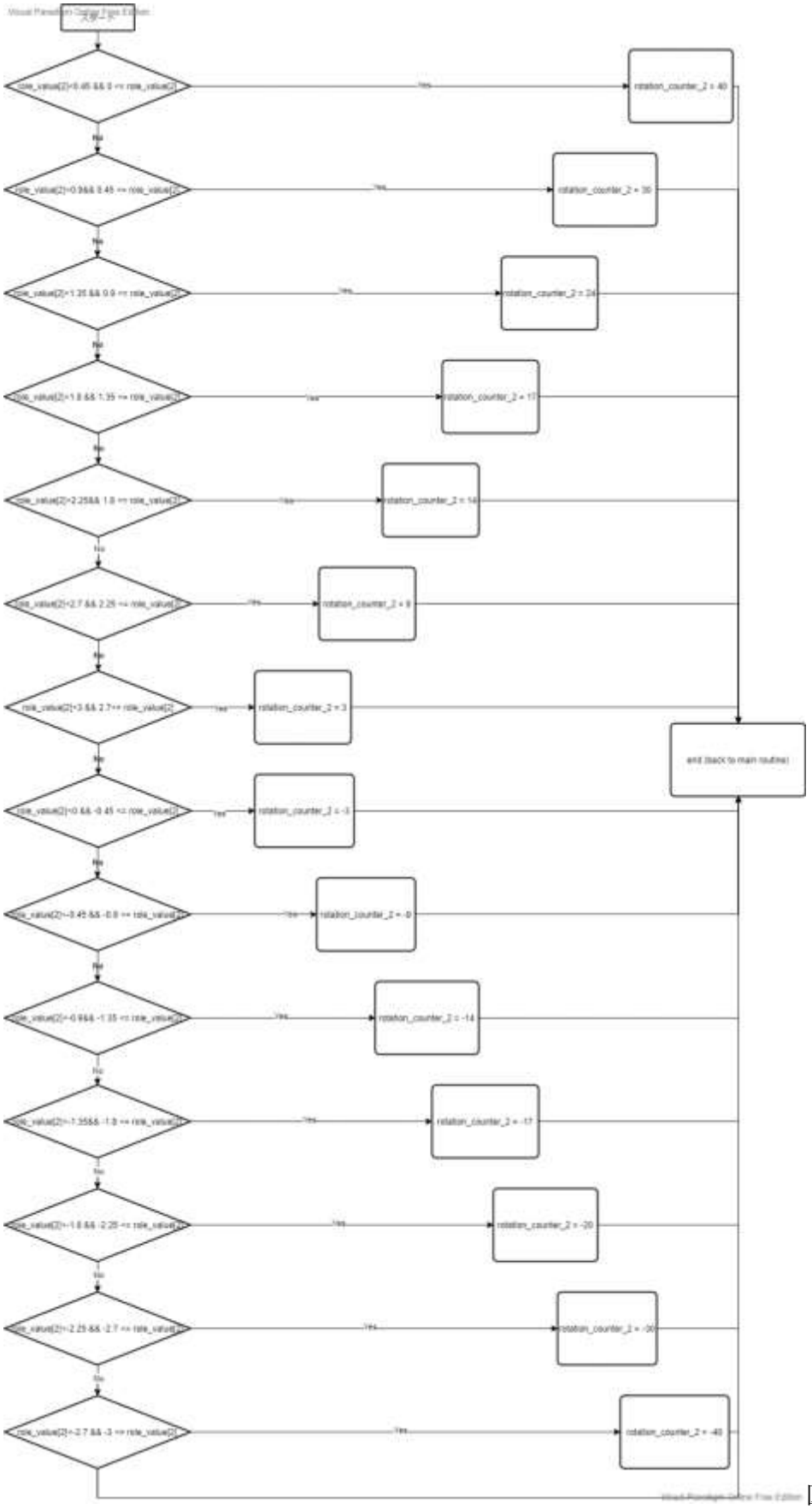
rotation_counter_2 = 40
rotation_counter_2 = 36
rotation_counter_2 = 24
rotation_counter_2 = 17
rotation_counter_2 = 14
rotation_counter_2 = 9
rotation_counter_2 = 3
end (back to main routine)
rotation_counter_2 = -3
rotation_counter_2 = -9
rotation_counter_2 = -14
rotation_counter_2 = -17
rotation_counter_2 = -20
rotation_counter_2 = -30
rotation_counter_2 = -40

図３　方向決定

ルーチン 2

４．プログラムの説明

プログラムは以下の通り。フロチャートを忠実に実装している。

```
#include <stdio.h>
#include<unistd.h>
#include <webots/light_sensor.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/inertial_unit.h>
#define MAX_SPEED 10
#define SPEED 6
#define TIME_STEP 64

int main() {
WbDeviceTag ls0, ls1,ls2, left_motor, right_motor,inner;
int i;
wb_robot_init();
/* get a handler to the distance sensors. */
ls0 = wb_robot_get_device("ls0");
ls1 = wb_robot_get_device("ls1");//left
ls2 =    wb_robot_get_device("back");
wb_light_sensor_enable(ls0, TIME_STEP);
wb_light_sensor_enable(ls1, TIME_STEP);
wb_light_sensor_enable(ls2, TIME_STEP);

WbDeviceTag ds[2];
   char ds_names[2][10] = {"ds_left", "ds_right"};

   for (i = 0; i < 2; i++) {
   ds[i] = wb_robot_get_device(ds_names[i]);
   wb_distance_sensor_enable(ds[i], TIME_STEP);
}
```

```
 inner = wb_robot_get_device("inner");
 wb_inertial_unit_enable(inner,TIME_STEP);


int rotation_counter = 0;
int counter_left = 0;
int counter_right = 0;
int stop_flag = 0;
int close_flag = 0;
//int close_left = 0;
//int close_right = 0;
/* get a handler to the motors and set target position to infinity (speed control). */
left_motor = wb_robot_get_device("wheel1");
right_motor = wb_robot_get_device("wheel2");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);

/*
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);
*/

while (wb_robot_step(TIME_STEP) != 1) {

  /* read sensor values */
  const double ls0_value = wb_light_sensor_get_value(ls0);
  const double ls1_value = wb_light_sensor_get_value(ls1);
  const double ls2_value = wb_light_sensor_get_value(ls2);


  printf("%f %f %f ¥n", ls1_value,ls0_value,ls2_value);

    double left_speed = 10;
    double right_speed = 10;
```

```
if(rotation_counter >0){
   //真後ろにあるので回転中
   rotation_counter--;
   left_speed = -5;
   right_speed = 5;

}else if(close_flag == 1 ){
//近づいて止まる
 break;

}else if(counter_left >0){
     counter_left--;
     left_speed = 0;
     right_speed = 10;
     printf("a¥n");

}else if(counter_right > 0){
     counter_right--;
     left_speed = 10;
     right_speed = 0;
     printf("b¥n");

}else{
     if((ls2_value > ls0_value)&&(ls2_value > ls1_value)){
      //ボールがロボットの真後ろにある時　回転する
         rotation_counter=20;

     }else if(ls1_value > ls0_value){
     //向かって左にボールがある
        counter_left = 1;

     }else if(ls1_value < ls0_value){
      //向かって右にボールがある
        counter_right = 1;
     }
     else if(ls0_value==1024&&ls1_value==1024){
```

```
            //真正面にあるとき
              close_flag = 1;

          }
    }
        /* Set the motor speeds. */
    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);

}


while (wb_robot_step(TIME_STEP) != 1) {
   double ds_values[2];

        for (i = 0; i < 2; i++)
          ds_values[i] = wb_distance_sensor_get_value(ds[i]);

    printf("%f %f ¥n",ds_values[0],ds_values[1]);

    double left_speed = 10;
    double right_speed = 10;

    if(stop_flag == 1){
    break;

      }else {
      if(ds_values[0] < 1000 || ds_values[1] < 1000 ){
      stop_flag =1;
      }

    }

    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);
}
```

```
  wb_motor_set_velocity(left_motor, 0);
  wb_motor_set_velocity(right_motor, 0);

int rotation_counter_2 = 0;
double *role_value;
role_value = wb_inertial_unit_get_roll_pitch_yaw(inner);

printf("現在の角度　%f¥n",role_value[2]);

//黄色のゴールだと　3.14 の角度の方向に
//青色のゴールだと 0 の方向に回転する
//今回は黄色のゴールに向かう

if(role_value[2]<0.45 && 0 <=role_value[2]){
   rotation_counter_2 = 40;

}else if(role_value[2]<0.9 && 0.45 <=role_value[2]){
      rotation_counter_2 = 30;
}
else if(role_value[2]<1.35 && 0.9 <=role_value[2]){
      rotation_counter_2 = 24;

}
else if(role_value[2]<1.8 && 1.35 <=role_value[2]){
      rotation_counter_2 = 17;
}
else if(role_value[2]<2.25 && 1.8 <=role_value[2]){
      rotation_counter_2 = 14;
}
else if(role_value[2]<2.7 && 2.25 <=role_value[2]){
      rotation_counter_2 = 9;

}else if(role_value[2]<3 && 2.7 <=role_value[2]){
      rotation_counter_2 = 3;
}
else if(role_value[2]>= -0.45 && 0 > role_value[2]){
```

```
   rotation_counter_2 = -40;
}
else if(role_value[2]>= -0.9 && -0.45 > role_value[2]){
   rotation_counter_2 = -30;

}else if(role_value[2]>= -1.35 && -0.9 > role_value[2]){
   rotation_counter_2 = -24;

}else if(role_value[2]>= -1.8 && -1.35 > role_value[2]){
   rotation_counter_2 = -17;
}
else if(role_value[2]>= -2.25 && -1.6 > role_value[2]){
   rotation_counter_2 = -14;

}
else if(role_value[2]>= -2.7 && -2.25 > role_value[2]){
   rotation_counter_2 = -9;

}else if(role_value[2]>= -3 && -2.7 > role_value[2]){
   rotation_counter_2 = -3;
}

while(wb_robot_step(TIME_STEP) != 1){
     double left_speed = 10;
     double right_speed = 10;

   if(rotation_counter_2 == 0 ){
   //回転する必要なし
   printf("a¥n");
   break;

   }else if(rotation_counter_2 > 0){
   //左周りに回転してゴールに正対する
   rotation_counter_2--;
   left_speed = -2;
   right_speed = 2;
```

```
    printf("b¥n");

   }else {
   //右周りに回転してゴールに正対する
    rotation_counter_2++;
    left_speed = 2;
    right_speed = -2;
    printf("c¥n");

   }

    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);
}

 //ゴールまで運ぶ
 wb_motor_set_velocity(left_motor, 10);
 wb_motor_set_velocity(right_motor, 10);

wb_robot_cleanup();
return 0;
}
```

５．実行結果

成功例のスナップショットを連続でのせる。

状態１：初期状態

状態２：光センサを使ってボールにある程度近づいた状態

状態３：距離センサを使ってアームの内側にボールを置いた状態

状態４：角度センサを使ってゴール方向に回転した状態

状態５：ゴールした状態

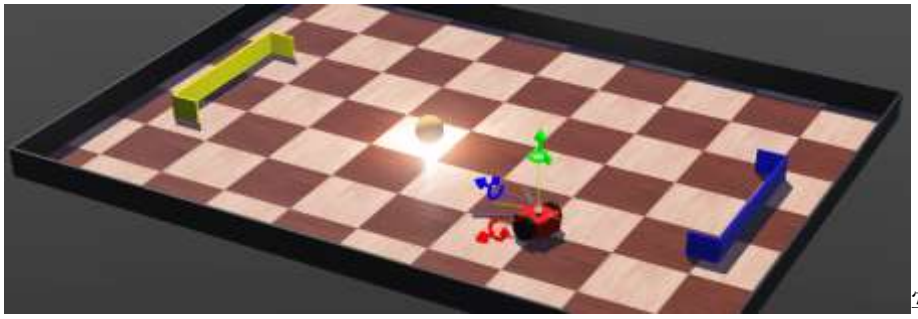
写真１　状態１

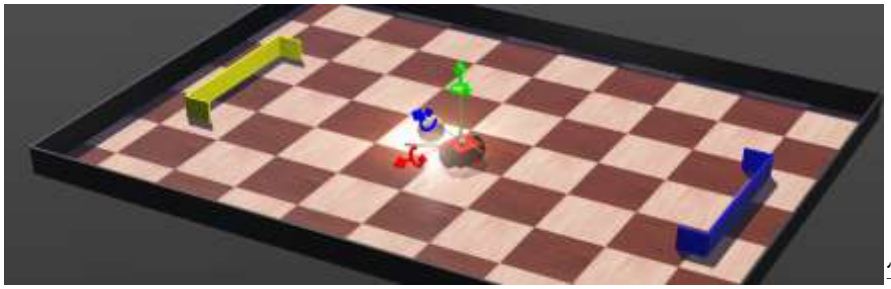
写真２　状態２


写真３　状態３

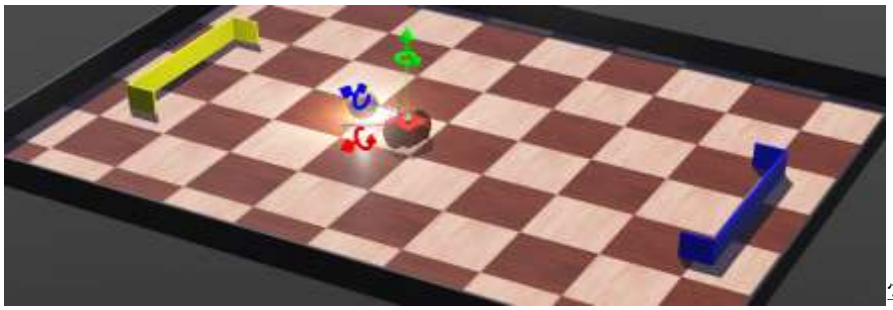
写真４　状態４


写真５　状態５

６．今後の課題

今回、解決しきれなかった点は以下の４つ。

１．ボールへの近づき方の効率化を図る。

現在はライトセンサーの値が異なると、すぐにロボットが進む方向を修正している。方向修正の頻度が多くて、ロボットのスピードがでない。

「ライトセンサーの左右の値の差が小さいときは、方向修正せずにとりあえず直進する。」みたいなアルゴリズムを組むと、ロボットのスピードが損なわれることもなく、もっと早い時間でボールにたどり着けると考える。

２．ロボットが回転したときに、アームの内側にセットしたボールが、飛び出る。

ロボットが回転したときに、アームがボールを外側に押し出してしまうことがある。

アームの形を工夫する。例えばアームの形を楕円にしてその焦点をアームの内側に設定したら、回転したときにボールがアームにあたっても、アームの内側にボールは跳ね返るのではないか。

３．壁際でアームが壁に引っかかり、身動きが取れなくなる。

アームが大きすぎると壁に引っかかりやすい。

４任意の位置のボールをゴールに運ぶ。

現在はボールがフィールド中央にあるときだけゴールに運べる。

角度センサと GPS センサを組みわせるとゴールへの向きを計算できる。