РОБОТОТЕХНИЧЕСКИЙ ИНЖЕНЕРНЫЙ ЦЕНТР ФМЛ №30

Техническая документация робота “ФМЛ30\_кегельринг\_макро V 2.0” для соревнований “кегельринг-макро”

Крылов Георгий

Руководитель – Лузин Д. В.

Санкт-Петербург

2013

Оглавление

[1. Регламент. 3](#_Toc367395508)

[2. Задача. 6](#_Toc367395509)

[3. Реализация. 6](#_Toc367395510)

[3.1. Конструкция. 6](#_Toc367395511)

[3.1.1. Используемые ресурсы. 6](#_Toc367395512)

[3.1.1. Oписание конструкции. 7](#_Toc367395513)

[3.2. Алгоритм. 8](#_Toc367395514)

[3.2.1. Разбор всех шагов по пунктам. 8](#_Toc367395515)

[3.2.2. Схема. 12](#_Toc367395516)

[3.3. Проблемы, возникшие в процессе разработки. 14](#_Toc367395527)

[3.4. Программа. 15](#_Toc367395528)

[3.5. Модель робота в “Lego Digital Designer Virtual Building Software”. 28](#_Toc367395529)

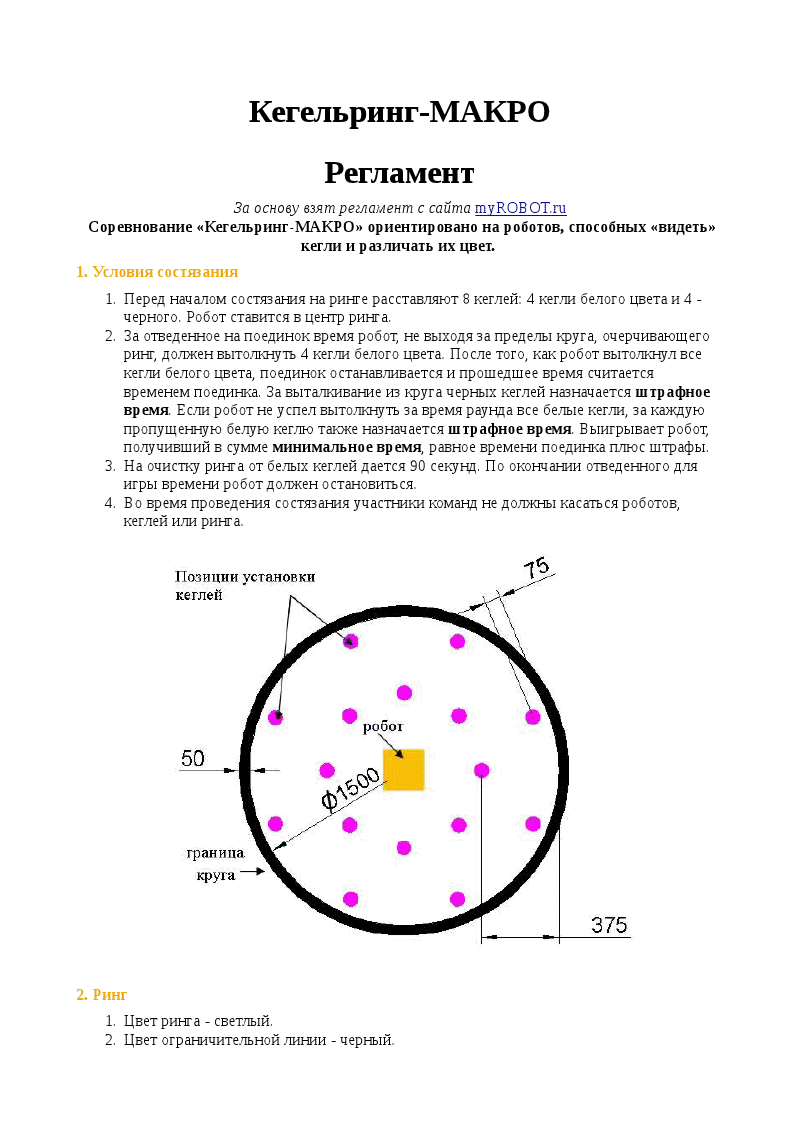
[6. Итоги. 29](#_Toc367395530)

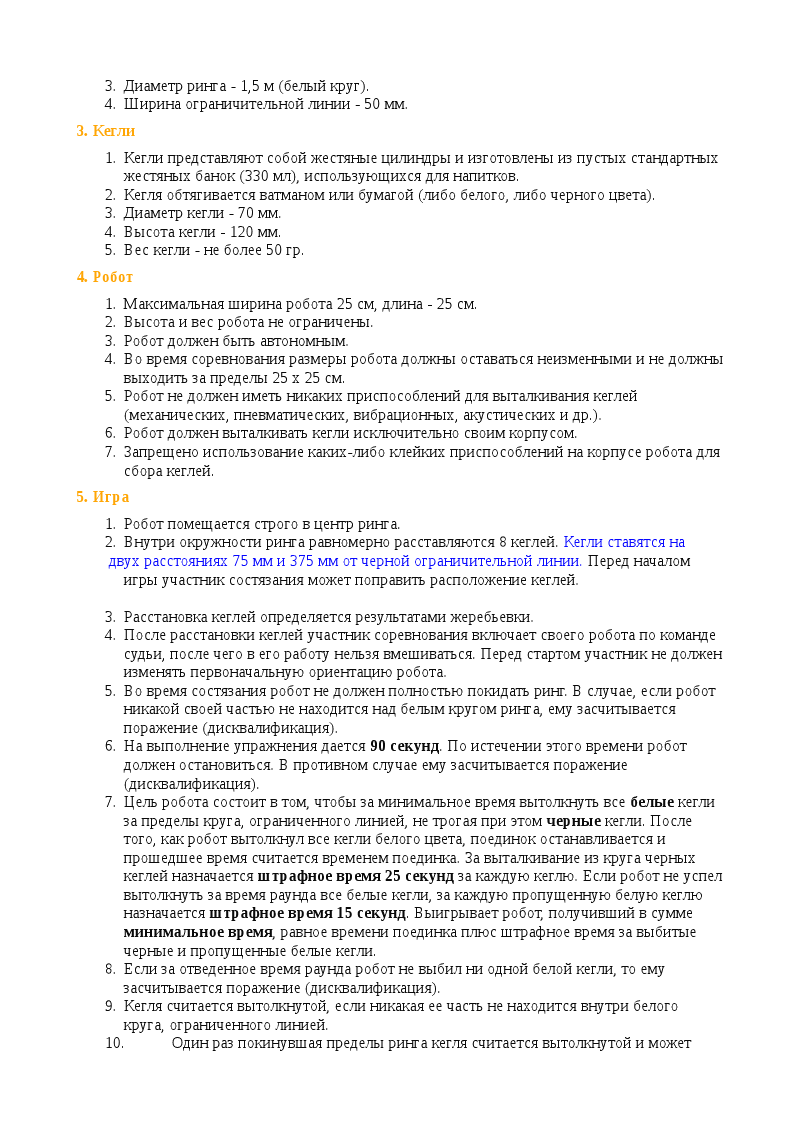
[6.1. Недостатки. 29](#_Toc367395531)

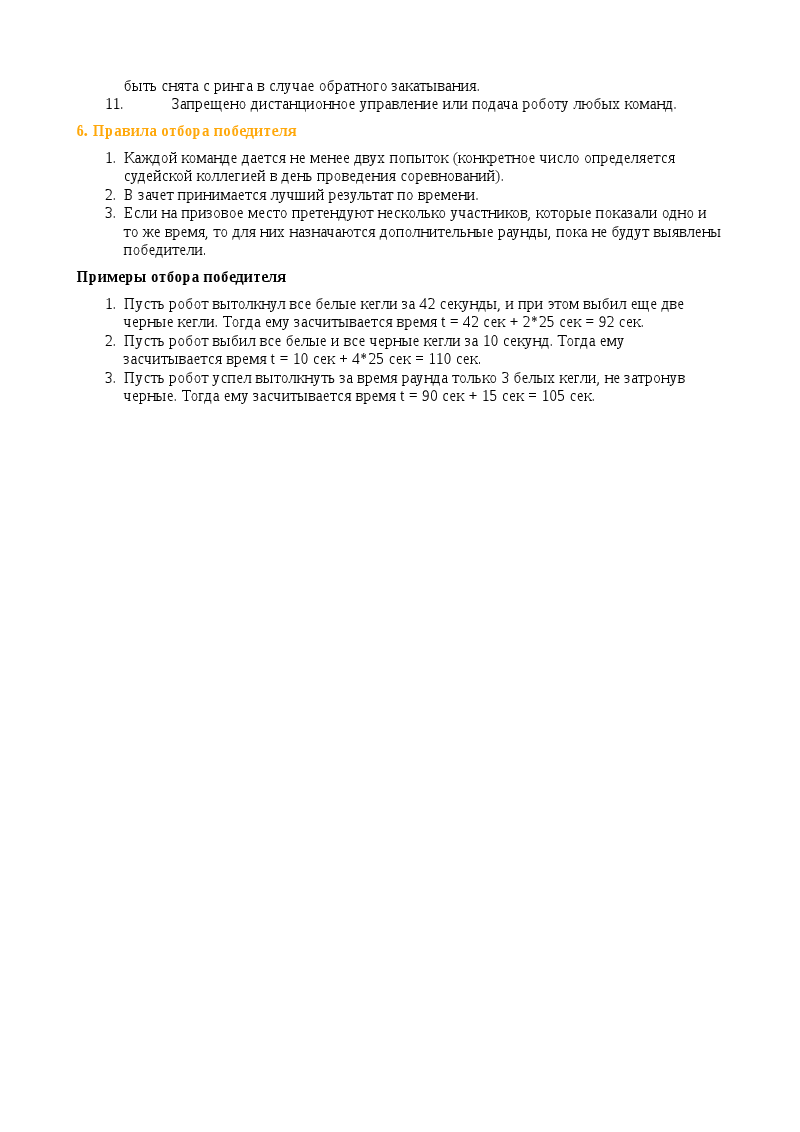
[6.2. Идеи для улучшения. 29](#_Toc367395532)

[6.3. Результат. 29](#_Toc367395533)

# 1. Регламент.







# 2. Задача.

Реализовать робота, который будет максимально надёжно и быстро, независимо от внешних факторов выполнять задание соревнования "Кегельринг-макро"

# 3. Реализация.

## 3.1. Конструкция.

### 3.1.1. Используемые ресурсы.

1. Детали:

//появится список, когда будет реализована модель в “ Lego Digital Designer”

2. Блок NXT.

3. Два (2) интерактивных сервопривода NXT (артикул LEGO № W979842) В дальнейшем просто сервопривод.

4. Датчики:

- “HiTechnic NXT Compass Sensor for LEGO Mindstorms NXT” (артикул NMC1034)

В дальнейшем просто “Compass” (одна (1) штука)

- “LEGO Education NXT Color Sensor” (артикул 9694)

В дальнейшем “Color RGB” (две (2) штуки)

- “LEGO Education Light Sensor” (артикул 9844)

В дальнейшем “Light” (одна (1) штука)

- “LEGO Education Touch Sensor” (артикул 9843)

В дальнейшем “Touch” (две (2) штуки)

5. Белый картон (один лист А4)

6. Пять стандартных проводов фирмы Lego.

Все необходимое кроме датчика Compass, Light, второго Color RGB и белого картона есть в наборе “Lego Mindstorms NXT 2.0.” (артикул 8547).

### 3.1.1. Oписание конструкции.

1. Для того, чтобы датчик “Color RGB” который отвечает за обнаружение черной линии, ограждающей поле, и движение по ней, не зависел от освещения, он был помещен в коробку, полностью закрывающую датчик.

2. Для того, чтобы датчик “Light” не зависел от освещения, был сделан специальный закрытый навес из картона, которым робот наезжал на банки.

3. Спереди, чтобы не занимать места, вместо колес были установлены полусферы.

4. Чтобы наехать навесом на банку, которая стоит немного в стороне от линии движения робота, спереди были установлены специальные стенки под углом.

5. Для того, чтобы выталкивать банки при обороте вокруг колеса, сзади были установлены стенки под углом.

6. Для того, чтобы при движении задняя часть не перевешивала, и робот не запинался ей об пол, спереди, для противовеса, были установлены датчики касания с разных сторон робота.

7. Вид сверху:

- датчик касания

- боковой

датчик “Color RGB”

- банка

- “Light” сенсор

- нижний датчик “Color RGB”

## 3.2. Алгоритм.

### 3.2.1. Разбор всех шагов по пунктам.

Прохождение поля разделено на два этапа: сначала выбиваются банки на внутреннем круге, потом выбиваются банки на внешнем круге.

1. Прохождение внутреннего круга.

С помощью датчика “Compass”, робот, находясь в центре поля, поворачивается на позицию, при движении вперёд из которой, робот окажется на позиции, где может находиться банка внутреннего круга.

Если на позиции возможного положения банки на внутреннем круге, находится черная банка или вообще нет банки, то робот возвращается на исходную позицию в центр поля и поворачивается на следующую позицию, при движении вперёд из которой, робот окажется на позиции, где может находиться банка внутреннего круга.

Если на позиции возможного положения банки на внутреннем круге находится белая банка, то робот выталкивает ее за черную линию, определяя черную линию по датчику “Color RGB”. После робот с помощью “Encoder” возвращается в исходную позицию в центр поля и прибавляет один к количеству белых банок, которое он вытолкнул.

Если робот вытолкнул определенное число белых банок, то после того, как робот вытолкнул последнюю белую банку, он не возвращается в исходную позицию, а приступает к прохождению внешнего круга.

Если робот побывал на всех позициях на внутреннем круге, где может находиться белая банка, но так и не вытолкнул определенное количество белых банок, то после того, как робот оказывается в последней позиции внутреннего круга, где может находиться банка, робот разворачивается на 180\*, чтобы не вытолкнуть черную банку (если она там есть), выезжает на границу поля и приступает к прохождению внешнего круга.

2. Прохождение внешнего круга.

Сначала робот разворачивается так, чтобы нижним датчиком “Color RGB” оказаться на границе черной линии и внутренней зоны поля.

После этого робот двигается по черной линии на пропорциональном регуляторе.

Если боковой датчик “Color RGB” фиксирует белую банку, то робот делает оборот вокруг колеса, ближнего к банке, тем самым выбивая ее с поля, и продолжает движение по линии.

3. Для того, чтобы не приходилось в ручную перенастраивать робота при смене освещения, перед стартом калибруются датчики “Color RGB” и “Light”:

- калибровка датчика “Color RGB”, который отвечает за обнаружение черной линии, ограждающей поле, и движение по ней, на линии и на поле.

- калибровка датчика “Light”, отвечающего за цвет банки находящейся на внутреннем круге, на белой и черной банке (либо на белой банке и пустоте).

- калибровка датчика “Color RGB”, находящегося на внешнем корпусе робота, на белой и черной банке (либо на белой банке и пустоте).

4. Из-за того, что показания компаса зависят от того места, где его используешь, в начале происходит калибровка в четырех направлениях движения робота – на стенки квадрата в центре поля. Порядок калибровки – по часовой стрелке.

5. После калибровки, перед стартом задается направление движения робота на внутреннем круге кнопками на NXT левая кнопка – движение против часовой стрелки, правая – движение по часовой стрелке.

6. Робот запускается нажатием на одну из трех кнопок: левую, правую и центральную (оранжевую). Запуск левой кнопкой означает, что робот будет искать на внутреннем круге одну белую банку, центральной – две банки, правой – три.

7. После того, как робот выехал из центра поля на позицию на внутреннем круге, где может находиться банка, и вернулся обратно в центр поля, робот с помощью датчика “Compass” поворачивается ровно в ту позицию, из которой он выезжал.

8. Робот поворачивается на 4 позиции, которые находятся между 4 позициями откалиброванными датчиком “Compass”, используя encoder ровно на 45\*.

9. Чтобы остановиться до того, как пройдет 90 секунд с начала выполнения задания, заведен таймер, который постоянно проверяет, что времени осталось больше, чем одна миллисекунда, если меньше или равно, то робот останавливается.

### 3.2.2. Схема.

Калибровка

Выбор направления и количества банок

Выезжает на место

да

нет

Это последняя позиция?

# 

да

нет

Банка белая?

# 

Выносим банку

Отъезжаем в центр

# 

Поворачиваем на следующую позицию

Разворачиваемся на 180\* и выезжаем

Выносим банку

да

нет

Банка белая?

Конец!

нет

Делаем оборот

да

Разворачиваемся на линию

Проезжаем по линии

нет

Есть белая банка?

да

Еще есть время?

## 3.3. Проблемы, возникшие в процессе разработки.

1. Показания датчиков “Color RGB” и “Light” сильно зависят от внешнего освещения.

2. Если робот находится в центре поля, то датчик “Sonar” (здесь и далее “Sonar” - ультразвуковой датчик расстояния “LEGO Education Ultrasonic Sensor” (артикул 9846)) не распознаёт банки, находящиеся на внешней окружности.

3. Во время поиска банок на внутренней окружности, когда робот вращается в центре поля, при нахождении их ультразвуковым датчиком, робот, при остановке, несколько откланяется от точного направления на банку.

4. Когда робот выталкивал банку и возвращался на прежнее положение (в центр поля), то робот оказывался немножко смещенным относительно того положения, из которого выезжал, чтобы вытолкнуть банку. Из-за этого, если использовать “Encoder” (здесь и далее “Encoder” - датчик угла поворота для сервопривода) для поворота робота на позиции, при движении из которых вперед робот может столкнуться с банкой, используя “Encoder”, то робот смещается и поворачивает мимо этих позиций.

5. Из-за неодинаковой скорости вращения приводов, иногда, когда робот подъезжал к банке, она оказывалась не по центру робота.

6. Из-за электромагнитных полей, создаваемых электрическими приборами, датчик “Compass” ориентируется по ним, а не по магнитному полю Земли. В результате, показания датчика “Compass” меняются в зависимости от электромагнитных полей.

## 3.4. Программа.

#pragma config(Sensor, S1, color2, sensorCOLORRED)

#pragma config(Sensor, S2, color, sensorCOLORRED)

#pragma config(Sensor, S3, light, sensorLightActive)

#pragma config(Sensor, S4, comp, sensorI2CHiTechnicCompass)

int colorw()

{

int q;

while (nNxtButtonPressed != 3)

{

wait10Msec(1);

q = SensorValue[color];

nxtDisplayString(3, "%d", q);

}

wait10Msec(5);

q = SensorValue[color];

nxtDisplayString(3, "%d", q);

wait10Msec(100);

PlaySound(soundBeepBeep);

return q;

}

int colorw2()

{

int q;

while (nNxtButtonPressed != 3)

{

wait10Msec(1);

q = SensorValue[color2];

nxtDisplayString(3, "%d", q);

}

wait10Msec(5);

q = SensorValue[color2];

nxtDisplayString(3, "%d", q);

wait10Msec(100);

PlaySound(soundBeepBeep);

return q;

}

int position()

{

int q;

while (nNxtButtonPressed != 3)

{

wait10Msec(1);

q = SensorValue[comp];

nxtDisplayString(3, "%d", q);

}

wait10Msec(5);

q = SensorValue[comp];

nxtDisplayString(3, "%d", q);

wait10Msec(100);

PlaySound(soundBeepBeep);

return q;

}

int lightw()

{

int q;

while (nNxtButtonPressed != 3)

{

wait10Msec(1);

q = SensorValue[light];

nxtDisplayString(3, "%d", q);

}

wait10Msec(5);

q = SensorValue[light];

nxtDisplayString(3, "%d", q);

wait10Msec(100);

PlaySound(soundBeepBeep);

return q;

}

void rotate\_on\_button(int pos, int pog)

{

wait10Msec(2);

while (abs(pos - SensorValue(comp)) > pog)

{

motor[motorB] = 10;

motor[motorC] = -10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void X2rotate3(int X2tr)

{

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorC] = 0;

while (nMotorEncoder[motorC] < X2tr)

{

motor[motorB] = -10;

motor[motorC] = 10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void X2rotate\_on\_button(int pos, int pog)

{

wait10Msec(2);

while (abs(pos - SensorValue(comp)) > pog)

{

motor[motorB] = -10;

motor[motorC] = 10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

/\*void backride(int speed2)

{

motor[motorB] = 0;

motor[motorC] = 0;

while (nMotorEncoder[motorB] > -1)

{

motor[motorB] = speed2;

motor[motorC] = speed2;

}

motor[motorB] = 0;

motor[motorC] = 0;

}\*/

void on\_botton\_from\_back(int q1, int pog4)

{

wait10Msec(5);

if (SensorValue[comp] > (q1 - pog4))

{

while (SensorValue[comp] > q1)

{

motor[motorB] = -10;

motor[motorC] = 10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

else

{

while (SensorValue[comp] < q1)

{

motor[motorB] = 10;

motor[motorC] = -10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

}

void X2on\_botton\_from\_back(int q1, int pog4)

{

wait10Msec(5);

if (SensorValue[comp] > (q1 - pog4))

{

while (SensorValue[comp] > q1)

{

motor[motorB] = 10;

motor[motorC] = -10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

else

{

while (SensorValue[comp] < q1)

{

motor[motorB] = -10;

motor[motorC] = 10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

}

void smallride(float smallrad, float p, float wheeldiam, int speed1)

{

float w;

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorB] = 0;

w = smallrad / (p \* wheeldiam) \* 360;

while (nMotorEncoder[motorB] < w)

{

motor[motorB] = speed1;

motor[motorC] = speed1;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

int color\_bank(float whiteBlackL)

{

wait10Msec(5);

if (SensorValue(light) > whiteBlackL)

return 1;

else

return 0;

}

int full\_small\_ride (float smallrad, float p, float wheeldiam, float whiteBlackL, float whiteBlackC, int speed1, int speed3)

{

int o = 0;

smallride(smallrad, p, wheeldiam, speed1);

if (color\_bank(whiteBlackL) == 1)

{

o = 1;

wait10Msec(5);

//PlaySound(soundBeepBeep);

while (SensorValue[color] > whiteBlackC)

{

motor[motorB] = speed3;

motor[motorC] = speed3;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

return o;

}

void backride2(int speed2, float blackwhite, int speed22, int dis)

{

motor[motorB] = 0;

motor[motorC] = 0;

while (nMotorEncoder[motorB] > dis)

{

motor[motorB] = speed2;

motor[motorC] = speed2;

}

while (nMotorEncoder[motorB] > 0)

{

motor[motorB] = speed22;

motor[motorC] = speed22;

}

motor[motorB] = 10;

motor[motorC] = 10;

}

void backride(int speed2, float blackwhite, int speed22, int dis)

{

motor[motorB] = 0;

motor[motorC] = 0;

while (nMotorEncoder[motorB] > 0)

{

motor[motorB] = speed2;

motor[motorC] = speed2;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void rotate3(int tr)

{

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorB] = 0;

while (nMotorEncoder[motorB] < tr)

{

motor[motorB] = 10;

motor[motorC] = -10;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void rotate\_in\_end ()

{

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorB] = 0;

while (nMotorEncoder[motorB] < 404)

{

motor[motorB] = 50;

motor[motorC] = -50;

}

motor[motorB] = -5;

motor[motorC] = 5;

wait10Msec(5);

}

void small\_ride\_on\_line2(float k2, int speed\_line2, float qw, float wheeldiam, int smallrad16)

{

float p = 3.14, w;

nMotorEncoder[motorB] = 0;

motor[motorB] = 0;

motor[motorC] = 0;

w = -smallrad16 / (p \* wheeldiam) \* 360;

wait10Msec(5);

while (nMotorEncoder[motorB] > w)

{

motor[motorB] = speed\_line2 + k2 \* (SensorValue[color] - qw);

motor[motorC] = speed\_line2 - k2 \* (SensorValue[color] - qw);

}

motor[motorB] = 0;

motor[motorC] = 0;

}

int X2ride(float smallrad, float p, float wheeldiam, float whiteblack, float blackwhite, int speed1, int speed2, int speed3, int pog, int pog4, int q1, int q2, int q3, int q4, float X2tr, int n, float k2, int smallrad17,int speed\_line2, int speed22, int dis)

{

int i = 0;

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

X2on\_botton\_from\_back(q1, pog4);

X2rotate3(X2tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1,speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

X2rotate\_on\_button(q4, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q4, pog4);

X2rotate3(X2tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

X2rotate\_on\_button(q3, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q3, pog4);

X2rotate3(X2tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

X2rotate\_on\_button(q2, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q2, pog4);

X2rotate3(X2tr);

if (full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3) == 0)

{

rotate\_in\_end ();

while (SensorValue[color] > blackwhite)

{

motor[motorB] = -speed3;

motor[motorC] = -speed3;

}

motor[motorB] = 10;

motor[motorC] = 10;

wait10Msec(1);

nMotorEncoder[motorC] = 0;

while (nMotorEncoder[motorC] > -350)

{

motor[motorB] = 0;

motor[motorC] = -70;

}

motor[motorB] = 0;

motor[motorC] = 10;

wait10Msec(2);

small\_ride\_on\_line2(k2, speed\_line2, blackwhite, wheeldiam, smallrad17);

return 0;

}

return 1;

}

int ride(float smallrad, float p, float wheeldiam, float whiteblack, float blackwhite, int speed1, int speed2, int speed3, int pog, int pog4, int q1, int q2, int q3, int q4, float tr, int n, float k2, int smallrad17,int speed\_line2, int speed22, int dis)

{

int i = 0;

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q1, pog4);

rotate3(tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1,speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

rotate\_on\_button(q2, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q2, pog4);

rotate3(tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

rotate\_on\_button(q3, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q3, pog4);

rotate3(tr);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

rotate\_on\_button(q4, pog);

i = i + full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3);

if (i >= n)

return 1;

backride(speed2, blackwhite, speed22, dis);

on\_botton\_from\_back(q4, pog4);

rotate3(tr);

if (full\_small\_ride (smallrad, p, wheeldiam, whiteblack, blackwhite, speed1, speed3) == 0)

{

rotate\_in\_end ();

while (SensorValue[color] > blackwhite)

{

motor[motorB] = -speed3;

motor[motorC] = -speed3;

}

motor[motorB] = 10;

motor[motorC] = 10;

wait10Msec(1);

nMotorEncoder[motorC] = 0;

while (nMotorEncoder[motorC] > -350)

{

motor[motorB] = 0;

motor[motorC] = -70;

}

motor[motorB] = 0;

motor[motorC] = 10;

wait10Msec(2);

small\_ride\_on\_line2(k2, speed\_line2, blackwhite, wheeldiam, smallrad17);

return 0;

}

return 1;

}

int ride\_on\_line\_only(float k, int speed\_line, float qw, float wq)

{

if (SensorValue[color2] <= wq)

{

motor[motorB] = speed\_line + k \* (SensorValue[color] - qw);

motor[motorC] = speed\_line - k \* (SensorValue[color] - qw);

return 0;

}

motor[motorB] = 0;

motor[motorC] = 0;

return 1;

}

void small\_ride\_on\_line(float k2, int speed\_line2, float qw, float wheeldiam, int smallrad15)

{

float p = 3.14, w;

nMotorEncoder[motorB] = 0;

motor[motorB] = 0;

motor[motorC] = 0;

w = -smallrad15 / (p \* wheeldiam) \* 360;

wait10Msec(5);

while ((nMotorEncoder[motorB] > w) && (time1[T1] < 59000))

{

motor[motorB] = speed\_line2 + k2 \* (SensorValue[color] - qw);

motor[motorC] = speed\_line2 - k2 \* (SensorValue[color] - qw);

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void ride\_with\_bank (float k, int speed\_line, float qw, float wq, float k2, int speed\_line2, int smallrad15, float wheeldiam)

{

while (time1[T1] < 59000)

{

while ((ride\_on\_line\_only(k, speed\_line, qw, qw) == 0) && (time1[T1] < 59000))

wait1Msec(1);

motor[motorB] = 5;

motor[motorC] = 5;

wait1Msec(5);

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorC] = 0;

while ((nMotorEncoder[motorC] > - 1600) && (time1[T1] < 59000))

{

motor[motorB] = 0;

motor[motorC] = -100;

}

motor[motorB] = 0;

motor[motorC] = 0;

small\_ride\_on\_line(k2, speed\_line2, qw, wheeldiam, smallrad15);

}

}

void smallride2(float smallrad, float p, float wheeldiam, int speed1)

{

float w;

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorB] = 0;

w = smallrad / (p \* wheeldiam) \* 360;

while (nMotorEncoder[motorB] > -(w + 1))

{

motor[motorB] = speed1;

motor[motorC] = speed1;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

void rotate90(int i)

{

motor[motorB] = 0;

motor[motorC] = 0;

nMotorEncoder[motorC] = 0;

while (nMotorEncoder[motorC] < i)

{

motor[motorB] = 0;

motor[motorC] = 30;

}

motor[motorB] = 0;

motor[motorC] = 0;

}

task main()

{

float smallrad = 40, smallrad2 = 7, smallrad15 = 3, smallrad16 = 7, smallrad17 = 15;

float tr = 101, X2tr = 115;

float k = 1.4, k2 = 5;

int n, y;

int i = 320, dis = 1;

int speed01 = -30, speed1 = 30, speed2 = -40, speed3 = 50, speed\_line = -70, speed\_line2 = -40, speed22 = -40;

//speed01 = -30, speed1 = 30, speed2 = -40, speed3 = 50, speed\_line = -70, speed\_line2 = -50;

int pog = 1, pog4 = 2;

float p = 3.14, wheeldiam = 7.6;

int q1, q2, q3, q4;

int white, black, whiteBlackL, whiteBlackC, whiteBlackC2;

//int varSysTime;

q1 = position();

q2 = position();

q3 = position();

q4 = position();

white = lightw();

black = lightw();

whiteBlackL = (white + black) / 2;

white = colorw();

black = colorw();

whiteBlackC = (white + black) / 2;

white = colorw2();

black = colorw2();

whiteBlackC2 = (white + black) / 2;

PlaySound(soundLowBuzz);

while (true)

{

nxtDisplayString(3, "napr");

if (nNxtButtonPressed == 1)

{

y = 0;

break;

}

else if (nNxtButtonPressed == 3)

{

y = 1;

break;

}

}

PlaySound(soundBlip);

wait10Msec(100);

while (true)

{

if (nNxtButtonPressed == 3)

{

n = 2;

break;

}

else if (nNxtButtonPressed == 1)

{

n = 1;

break;

}

else if (nNxtButtonPressed == 2)

{

n = 3;

break;

}

}

//varSysTime = nSysTime;

ClearTimer(T1);

if (y == 0)

{

if (ride(smallrad, p, wheeldiam, whiteBlackL, whiteBlackC, speed1, speed2, speed3, pog, pog4, q1, q2, q3, q4, tr, n, k2, smallrad17, speed\_line2, speed22, dis) == 1)

{

motor[motorB] = 0;

motor[motorC] = 0;

smallride2(smallrad2, p, wheeldiam, speed01);

rotate90(i);

small\_ride\_on\_line2(k2, speed\_line2, whiteBlackC, wheeldiam, smallrad16);

}

}

else

{

if (X2ride(smallrad, p, wheeldiam, whiteBlackL, whiteBlackC, speed1, speed2, speed3, pog, pog4, q1, q2, q3, q4, X2tr, n, k2, smallrad17, speed\_line2, speed22, dis) == 1)

{

motor[motorB] = 0;

motor[motorC] = 0;

smallride2(smallrad2, p, wheeldiam, speed01);

rotate90(i);

small\_ride\_on\_line2(k2, speed\_line2, whiteBlackC, wheeldiam, smallrad16);

}

}

motor[motorB] = 0;

motor[motorC] = 0;

// while (time1[T1] < 59000)

ride\_with\_bank (k, speed\_line, whiteBlackC, whiteBlackC2, k2, speed\_line2, smallrad15, wheeldiam);

}

## 3.5. Модель робота в “Lego Digital Designer Virtual Building Software”.

//Будет добавлена, когда будет реализована

# 6. Итоги.

## 6.1. Недостатки.

1.Датчик Color RGB, находящийся на корпусе робота с внешней стороны, зависит от освещения, из-за этого иногда робот делает оборот там, где нет банки.

2. При изменении магнитного поля (например рядом прошел человек с телефоном) датчик Compass сбивается и не верно работает дальше.

## 6.2. Идеи для улучшения.

Можно использовать один датчик для определения цвета банки на внутреннем круге и на внешнем, закрепить его на третьем моторе, и поворачивать датчик на 90\*, когда мы выезжаем на внешний круг.

А на освободившееся место для датчика установить Sonar или лазерный дальномер и проверять, есть ли хоть какая-то банка на позиции внутреннего круга, это сэкономит время.

## 6.3. Результат.

На весенних соревнованиях робот сбил все белые банки и не одной черной за очень короткое время и занял второе место, опоздав на несколько секунд.

На соревнованиях в ДДТ на 9 линии робот занял второе место, проиграв тому же роботу.

Последнее исправление 19.09.2013  
Версия 2.2