21st OCTOBER, 2015

Contents

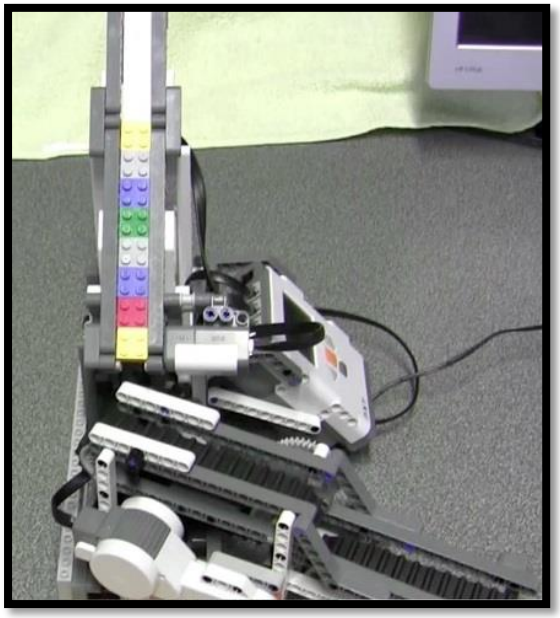
PICK AND PLACE ROBOT

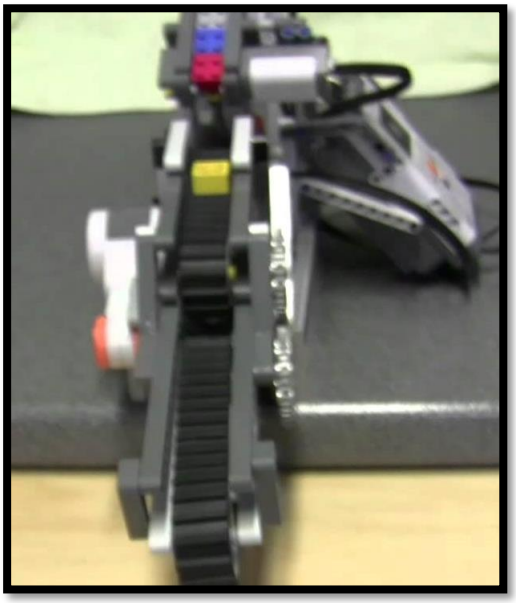
Objective

The assignment of this robot is to sort different colour of the Lego bricks and place them in different zones according to their colour.

Construction

The robot's structural design was as shown in the Figures below. The design was accomplished with the use of components readily in the given Lego Mindstorms toolkits. The robot had an input buffer which forms an integral part of the robot itself. It had two conveyors which have different elevations, located immediately after a gate above which hanged a colour sensor calibrated to sense three different block colours: black, blue and grey.





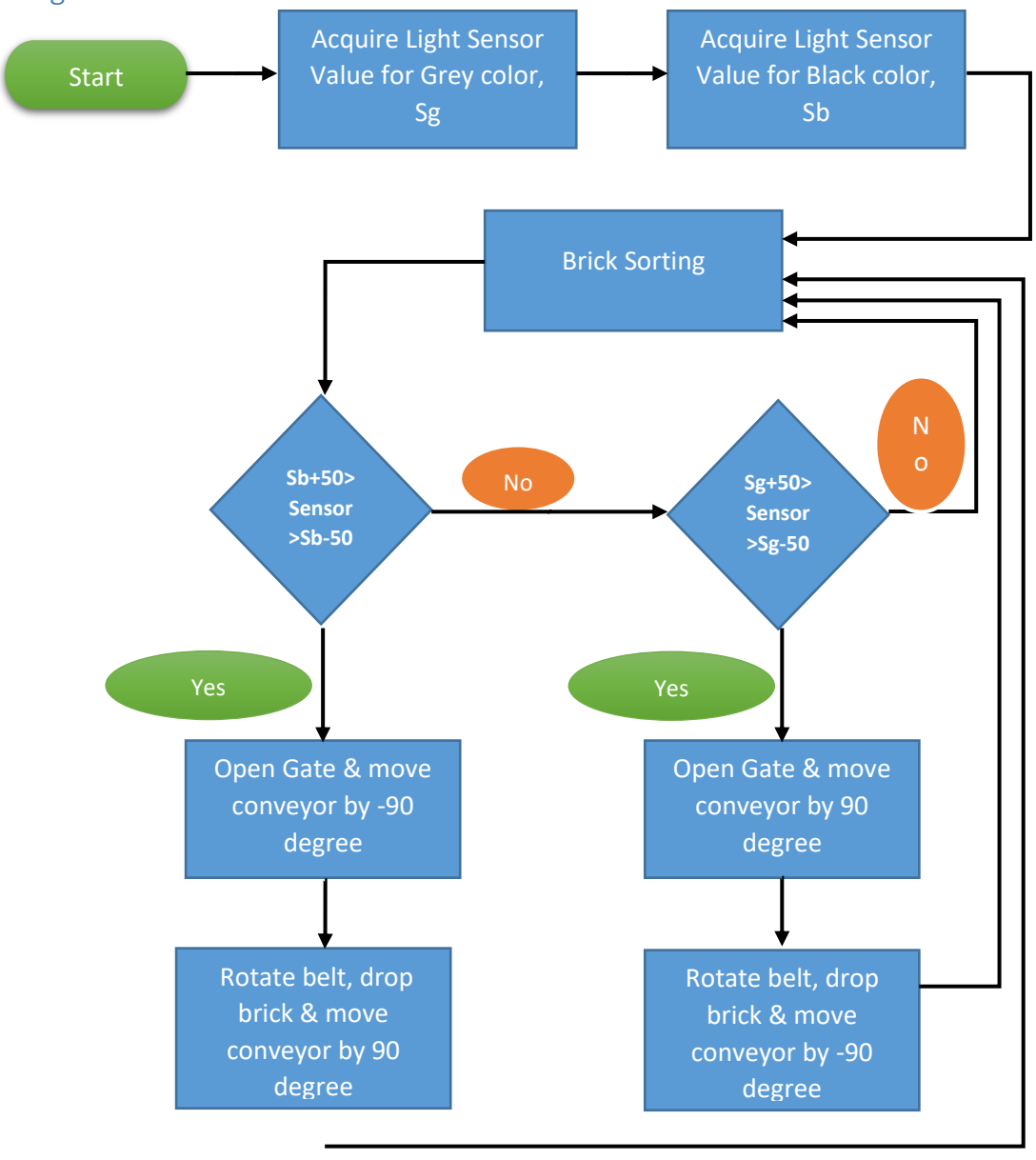
Programming Algorithm describtion

After construction, the robot was programmed using NXC programming language. The program code is named “BrickSorterProgram_Final.nxc”.

The robot was programmed so that upon running the code on the NXT brick (control), the gate actuator (motor) opens after the colour sensor detected and calibrated block(s) at the input buffer. The gate pushed the block onto the first conveyor (higher) which further transferred it to the second conveyor. At the exit of the conveyor, the blocks were dropped into one of left, right, or centre storage locations depending on the colour.

The robot wrote the colour of the detected block onto the brick display screen, and after all the blocks in the buffer must have been sorted. It went to sleep until such a time when new set of block(s) were loaded.

Program Flow Chart



Program Code

```
#define greyAngle 180
#define PlusMinus 50
#define gateangle 360
#define gatespeed 50
#define BeltMoSpeed 90
#define BeltWait 2000
#define SortMoSpeed 80
sub belt(){
    OnRev(OUT_C,BeltMoSpeed);
    Wait(BeltWait);
    Off(OUT_C);
    }
sub sortMo(int angle){
    RotateMotorEx(OUT_B, SortMoSpeed, angle, 0, false, true); // sort motor
    }
sub gate(){
    RotateMotorEx(OUT_A, gatespeed, -gateangle, 0, false, true);
    }

task main(){

  SetSensorType(IN_1,  SENSOR_TYPE_LIGHT_ACTIVE);
  SetSensorMode(IN_1, SENSOR_MODE_RAW);
  SetSensorType(IN_4, SENSOR_TYPE_TOUCH);
  int grey, black;
```

```
  // Calibration
 TextOut(1, LCD_LINE2,"Place & Calibrate Grey block");
 until (SENSOR_4 == 1);
 grey=SENSOR_3;
 NumOut(15, LCD_LINE3,SENSOR_3);
 Wait(2000);
 ClearScreen();
 TextOut(1, LCD_LINE1,"Place & Calibrate BLACK Block");
 until (SENSOR_4 == 1);
 black=SENSOR_3;
 NumOut(15, LCD_LINE3,SENSOR_3);
 Wait(2000);
 ClearScreen();
 TextOut(5, LCD_LINE3,"Calibration Done");
 TextOut(5, LCD_LINE4,"Rock n' Load");
 Wait(3000);



  while(true)
  {
   if(SENSOR_1>(grey-PlusMinus) ){
      gate();
      sortMo(-greyAngle);
      belt();
      sortMo(greyAngle);
      }
```

```
    else if(SENSOR_1>(black-PlusMinus) & SENSOR_1<(black+PlusMinus)){
      gate();
      belt();
      }
    else{
      TextOut(15,LCD_LINE3,"BLOCK NOT IDENTIFIED");
      Wait(5000);
      ClearScreen();
      }
  }
}
```

LINE FOLLOWER ROBOT

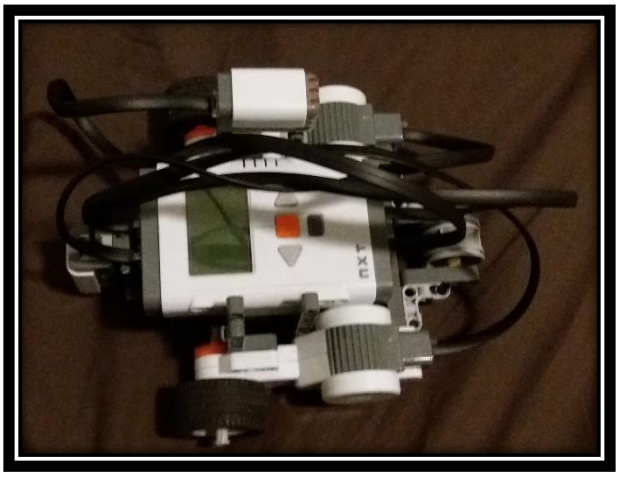
Objective

The main objective of the robot was to follow black line(s) (virtual shape-network) drawn on the floor. It should follow the lines with very accurately, as a robot having a wide deviation from the line was not acceptable.

Construction

The structural design of the robot was as shown in the Figures below. The design was accomplished with the use of components readily in the given Lego Mindstorms toolkits. The robot was a motorised vehicle (mobile robot) which had two tyres and a roller at the back as the traction devices.

The robot had a downward facing colour sensor that calibrates the light intensity of the track under the robot's track and uses that to control (steer) the robot along the desired path (black circle in the case of the demonstration arena).

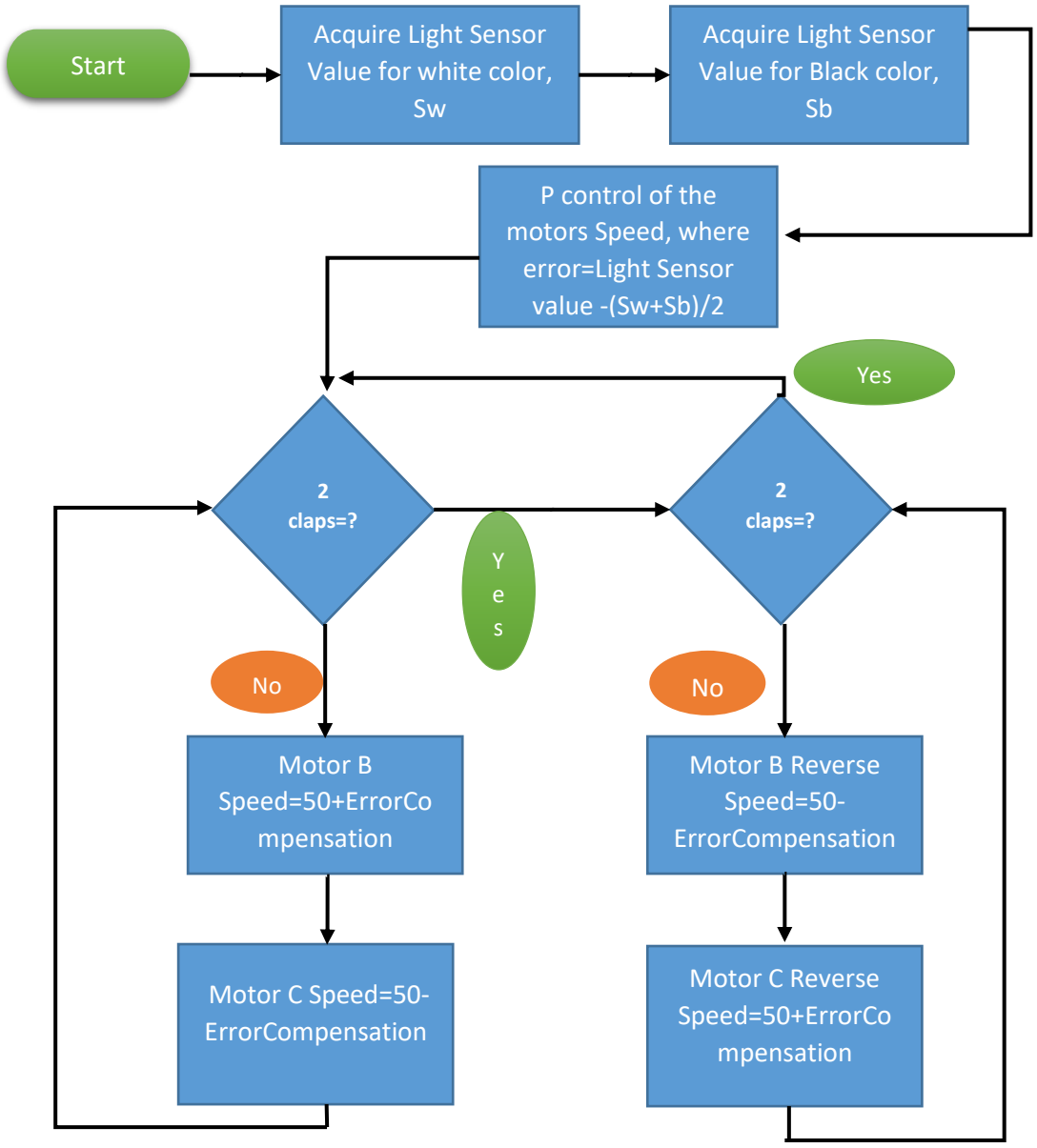


Programming Algorithm describtion

After construction, the robot was programmed using NXC programming language. The program code is named "LineFollower_Final.nxc".

The robot was programmed so that upon running the code on the NXT brick (control), the two motors B and C are powered **on** and moved in the forward direction along the edge of the black loop on the floor. The path of the robot was maintained by the use of a P control with proportional gain (k_p =0.6). The error (deviation of the robot's path from the desired line path) was computed by the calibrated value of the line colour and the colour of its contrasting ambience which enables the computation of the robot's motor power used to control power.

Program Flow Chart



Program Code

```
#define clap 70
#define clapInterval 3000

int dir=1;
int white=0;
int black=0;
int mid=0;
```

```
task moveYo(){
float computedPower,kp,error;
string eStr,eOut,cpStr,cpOut;
   kp=0.6;
    while(true){
      error=SENSOR_3-mid;
      computedPower=error*kp;
      if (computedPower>50){
        computedPower=50;
      }

      if (computedPower<-50){
        computedPower=-50;
      }
      ClearScreen();
      eStr = NumToStr(error);
      eOut = StrCat("Error: ",eStr);
      cpStr = NumToStr(computedPower);
      cpOut = StrCat("Cp: ",cpStr);
      NumOut(15,LCD_LINE5,SENSOR_3);
      TextOut(1,LCD_LINE1,eOut);
      TextOut(1,LCD_LINE3,cpOut);

      if(dir==1){
       OnFwd(OUT_B, 50+computedPower);
       OnFwd(OUT_C, 50-computedPower);
      }
      else if(dir==2){
       OnRev(OUT_C, 50+computedPower);
       OnRev(OUT_B, 50-computedPower);
      }
    }
}

task check_clap(){
  while (true){
    if (SENSOR_2 > clap){
      int t0 = CurrentTick();
      while( (CurrentTick()-t0)  <clapInterval){
        if (SENSOR_2>clap){
          if(dir==1)
          {dir=2;}
          else if (dir==2)
```

```
          {dir=1;}
        }
      }
    }
  }
}

task main(){
  SetSensorType(IN_3,  SENSOR_TYPE_LIGHT_ACTIVE);
  SetSensorMode(IN_3, SENSOR_MODE_RAW);
  SetSensorType(IN_4, SENSOR_TYPE_TOUCH);
  SetSensorSound(IN_2);

  // Calibration
  TextOut(10, LCD_LINE2,"Calibrate WHITE");
  until (SENSOR_4 == 1);
  white=SENSOR_3;
  NumOut(15, LCD_LINE3,SENSOR_3);
  Wait(3000);
  ClearScreen();
  TextOut(10, LCD_LINE1,"Calibrate BLACK");
  until (SENSOR_4 == 1);
  black=SENSOR_3;
  NumOut(15, LCD_LINE3,SENSOR_3);
  TextOut(5, LCD_LINE4,"Calibration Done");
  Wait(3000);
  mid=(black+white)/2;
  ClearScreen(); // Calibration Done

  Precedes(moveYo, check_clap);
}
```