UNIVERSITY
OF GÄVLE

FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT

A MODEL OF LINE FOLLOWING ROBOT USING PID CONTROLLER

An Educational Platform Based on LEGO Mindstorms NXT Kit

Zimu Wang

September 2015

Bachelor's Thesis in Electronics

Study Programme in Electrical Engineering

Examiner: Niclas Björsell

Supervisor: Nauman Masud

Acknowledgement

I would like to express my sincere thanks to my supervisor Mr. Nauman Masud, for his ideas, encouragement and support. I am sincerely grateful to Mr. Niklas Rothpfeffer for his supporting of the equipment and his hints. I would also like to thank my parents for their deep love.

.

Abstract

PID controllers are widely used in industrial control systems since it could significantly improve the performance of the systems. It is a basic and important tool for an engineering student.

An educational PID controller model using LEGO Mindstorms NXT kit as a line following robot was presented in this thesis. This robot could be controlled by a PC via Bluetooth. MATLAB was used for analyzing data and controlling the robot. A MATLAB Toolbox named “RWTH—Mindstorms NXT Toolbox” was utilized for remote controlling and gathering the feedback data from the robot in real-time via Bluetooth.

The algorithm and tuning methods of PID controllers were studied in this thesis. Moreover, the effectiveness of each parameter would be studied by comparing the result of P, PI and PID control type.

Keywords: Line follower, PID controller, LEGO Mindstorms, MATLAB.

Table of contents

Acknowledgement i

Abstract iii

Table of contents v

1. Introduction 1

1.1 Aim of thesis 1

1.2 Thesis outline 2

2 Method 3

2.1 Structure of the educational platform 3

2.1.1 LEGO Mindstorms NXT 3

2.1.2 Layout 5

2.1.3 Principle of tracking 6

2.2 PID controller 7

2.2.1 Algorithm 7

2.2.2 Tuning method 9

2.3 Programming 10

3 Results 12

3.1 P control type 13

3.2 PI control type 14

3.3 PID control type 15

4 Discussion 17

5 Conclusions and future work 18

References 19

Appendix A1

1. Introduction

Learning and teaching with new method could help the students learning and understanding in engineering education [1]. It is reported that robotic educational platform could be utilized as new methods for helping and motivating student learning the knowledge related to many engineering disciplines such as electronics, automatic control, computer science and signal processing [2]. LEGO Mindstorms series are widely used as a modeling or an educational tool since the robots are reusable, low-cost, programmable, modular and multifunction [3]. The kit contains a programmable mini-computer which is called NXT Intelligent Brick and some other components including sensors and motors. In this research, a model of industrial line following robot using LEGO Mindstorms NXT kit will be introduced. This model is supposed to utilize the PID control concept to improve the performance and efficiency.

PID controller is a basic and important concept for control engineering student. A PID controller for controlling the water level in water tanks was introduced as a simple example in common control system courses. PID controller could also be used for automatic steering systems. The earliest application of PID controller was an automatic ship steering designed by Nicolas Minorsky for the US Navy [4]. Automatic steering system is also be utilized for the line following robot. The source of motion error could be the delay, inertia of the robot and instability of the system. PID controller is a feasible solution for minimizing the errors in these systems.

1.1 Aim of thesis

In this research, an educational platform will be built for students to understand how a PID controller works in an automatic steering system. MATLAB is supposed to be used for visualizing and analysis of the errors measured from the sensors. The errors could be estimated by calculating the difference between the measured value and target. The students are supposed to analyze and compare the results of a line follower using P controller, PI

controller, and PID controller. Moreover, the principles of PID controller will be studied by comparing the corresponding results in this paper.

1.2 Thesis outline

The remainder of this paper consists of four sections. In section 2, the structures of this educational platform and programming environment will be shown. The algorithm of PID controller and the tuning method will be studied in this section as well. The result of different control types will be shown in section 3. The advantages and disadvantages of this educational platform will be discussed in section 4. The conclusions will be made in the final section.

2 Method

The operating principles of the line following robot would be introduced in this chapter. Moreover, the principles of PID controller would be studied in this chapter. Due to the difficulty of determining the coefficient of each controller, tuning method would be studied in this chapter. The robot would be remotely controlled via Bluetooth connection and it is operated by MATLAB with RWTH Toolbox. The method of MATLAB programming would be introduced in this chapter also.

2.1 Structure of the educational platform

2.1.1 LEGO Mindstorms NXT

The structure of LEGO NXT Intelligent Brick is shown in Figure 2.1. Four input ports and three output ports could be connected with sensors and motors which are included in the LEGO Mindstorms NXT kit.

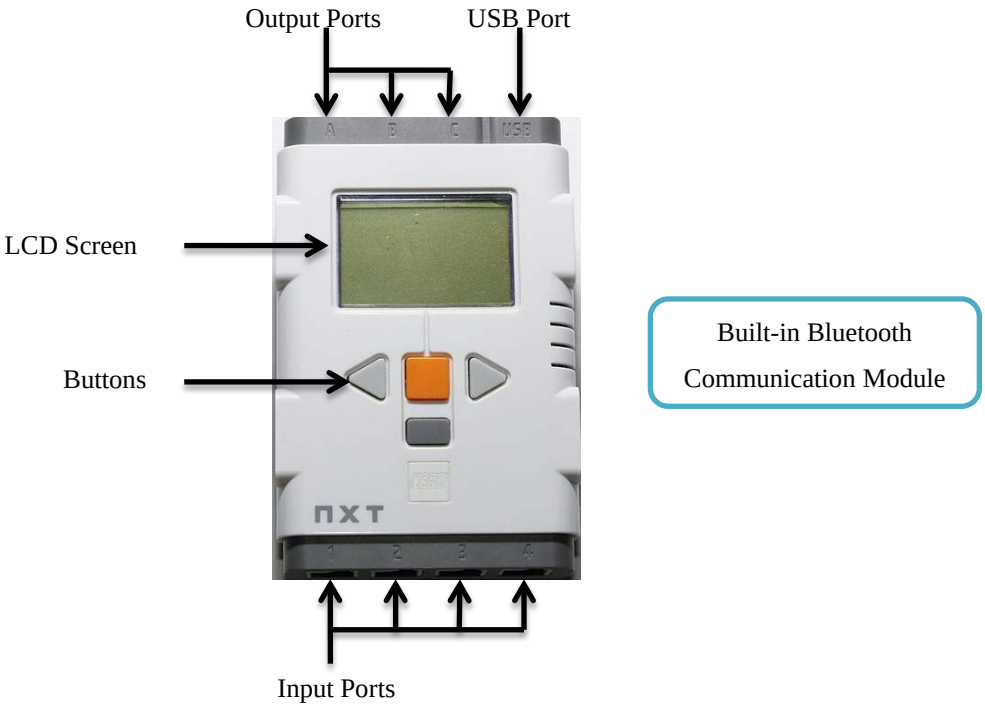

Figure 2.1: Structure of LEGO Mindstorms NXT Intelligent Brick

The Interactive Servo Motor as shown in Figure 2.2 has a built-in rotation sensor which could feedback the actual rotation speed and distance of the motor within one degree accuracy [5]. The power level of the motor could vary from -100 to 100 according to the RWTH Toolbox [6] and it could rotate both clockwise and counterclockwise.

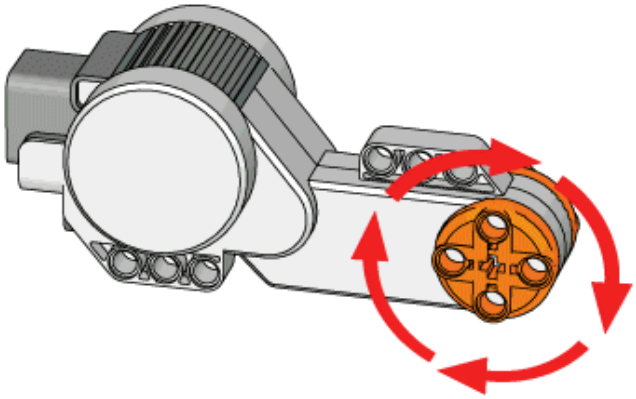

Figure 2.2: LEGO NXT Interactive Servo Motor [7]

A photo resistor and a light emitted diode (LED) are integrated in the NXT Light Sensor as shown in Figure 2.3.

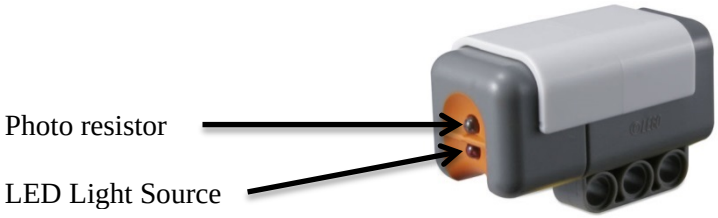

Figure 2.3: LEGO NXT Light Sensor

The photo resistor is able to measure the intensity of the ambient light and the intensity of reflected light which is illuminated from the LED. The measured light value could vary from 0 to 1023 as the default setting of RWTH Toolbox [6].

2.1.2 Layout

The NXT Brick was connected to a light sensor which could measure the strength of the reflected light as shown in Figure 2.4.

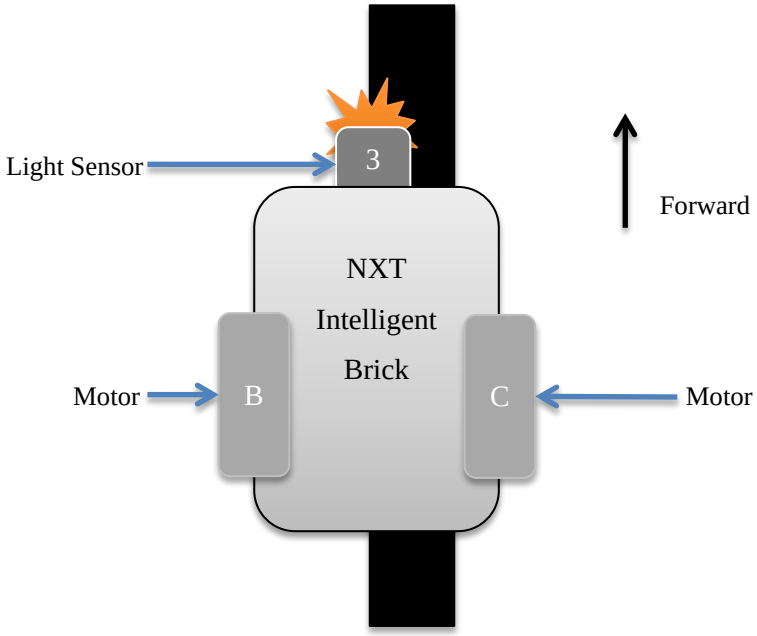

Figure 2.4: Schematic diagram of a line follower robot

The light sensor was connected to port 3. When the robot moves exactly to the boarder of the line, the sensor will get a unique value of the reflected light which is called target. When the robot is operating, the sensor could measure the reflected light in real-time and subtract with the target to get the deviation.

Figure 2.5: Layout of line following robot

"B" and "C" was shown in Figure 2.4 are two interactive servo motors connected with port B and port C. As it shown in Figure 2.5, the robot could move forward when the powers of two motors are equivalent and it could turn when the powers are different. In this thesis, the steering power was determined by the difference value between the target and the measured reflected light strength.

2.1.3 Principle of tracking

The NXT robot is desired to follow the boundary of the black line in this thesis. At the beginning of test, the light sensor of the robot would be placed to focus on the boundary of the line to measure the brightness of the target in ambient light.

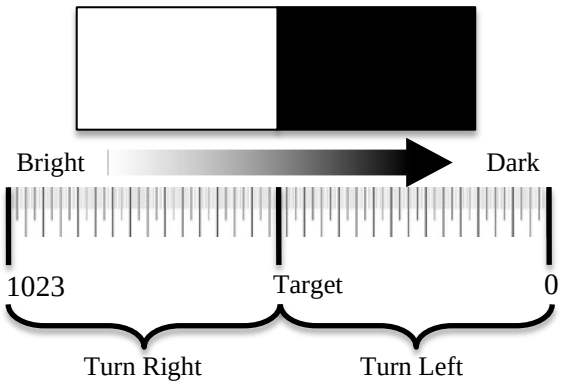

Figure 2.6: Principle of tracking

As it shown in Figure 2.6, the robot would compare the measured value of brightness with the target when it is running. When the measured brightness is brighter than the target, the robot would turn right and it would turn left if the measured result is darker than the target. The robot would continuously swing when the above tracking principle applied. Therefore, a PID controller would be applied for solving this problem in this thesis.

2.2 PID controller

2.2.1 Algorithm

Proportional-integral-derivative (PID) controllers are widely used in control systems. A PID controller could minimize the error by analyzing the statistics of errors. As the block diagram shown in Figure 2.7, the algorithm of a PID controller contains three terms: proportional, integral and derivative values. These values based on the analyzing of present error, integration of past errors and the prediction of future errors.

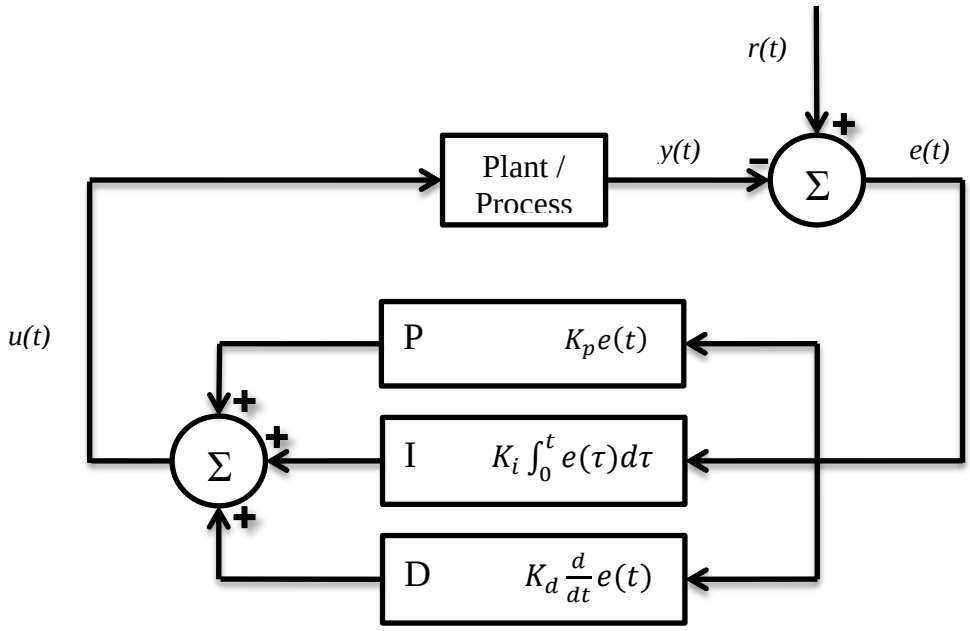

Figure 2.7: Algorithm of a PID controller system

Error could be estimated by:

$$e(t) = r(t) - y(t)$$

where $r =$ target value

$y =$ measured value

The output of an ideal continuous time PID controller is given as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

where K_p = proportional gain

K_i = integral gain

K_d = derivative gain

Hence, the transfer function could be obtained as:

$$u(s) = \left[K_p + \frac{K_i}{s} + K_d \cdot s\right] e(s)$$

where K_p = proportional gain

K_i = integral gain

K_d = derivative gain

In this thesis, the PID controller should operate in a programming loop. Therefore, the continuous time PID controller could be transformed in discrete time domain which could be expressed as:

$$u(z) = \left[K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_d}{T_s}(1 - z^{-1})\right] e(z)$$

where $T_s =$ loop time

2.2.2 Tuning method

The robot might be out of control or overreaction without tuning the PID controller. Although there are only three parameters in a PID controller, it's difficult to tune the PID controller manually since these three parameters could influence or even conflict with each other.

Several feasible PID tuning methods are available for the solution. Due to the limited time and cost budget, software tuning methods and pure manual tuning methods are not realistic in this thesis. Hence, one of the tuning formulas would be considered as a suitable solution in this paper. There are several well-known PID tuning formulas. Compared with Cohen-Coon(C-C) method, internal model control (IMC) method, gain-phase margin (G-P) method and integral-error based methods, Ziegler–Nichols method was reported as the best tuning method when both performance and robustness are considered [8].

The parameters of PID controller would be estimated by Ziegler–Nichols tuning method in this paper. As it shown in table 2.1, K_p, K_i and K_d are proportional gain, integral gain and derivative gain. K_u is the ultimate gain which could be obtained when the output starts to oscillate. T_u is the ultimate period and it can be estimated by measure the period when the ultimate gain is obtained.

Type	K_p	K_i	K_d
P	$0.5\ K_u$	-	-
PI	$0.45\ K_u$	$1.2\ K_p/T_u$	-
PID	$0.6\ K_u$	$2\ K_p/T_u$	$K_pT_u/8$

Table 2.1: Ziegler–Nichols tuning method [9]

In this paper, the parameters K_p , K_i and K_d could be estimated by obtaining the ultimate gain K_u and ultimate period T_u. For estimating the parameters, all of the three parameters set as 0 at the beginning. Then slightly increase the value of K_p again and again, when the robot start oscillate around the boarder of the line, the proportional gain is the ultimate gain and the oscillating period is the ultimate period. Then the parameters K_p , K_i and K_d could be calculated according to Table 2.1.

2.3 Programming

NXT Intelligent Brick could be operated by NXT-G programming language or toolkits of some other programming languages. In this thesis, the desired programming language is MATLAB since its feasibility and convenience of real-time data analysis. An open-source MATLAB toolbox named RWTH was utilized to control the NXT robots in this thesis. The RWTH—Mindstorms NXT Toolbox which is developed by a project group of RWTH Aachen University, Germany, is implemented for controlling the LEGO Mindstorms NXT robots via USB or Bluetooth connection. The functions which based on the LEGO Mindstorms NXT Bluetooth communication protocol [5] are extendable and those functions are pre-implemented to provide a direct way of real-time communication between the Mindstorms NXT Brick and MATLAB [10].

The motors could be enabled using RWTH Toolbox by the following code:

```
left = NXTMotor( MOTOR_B );
right = NXTMotor( MOTOR_C );
```

The base power of each motor was set to 10 to make the robot moving forward. The robot could turn by adding or subtracting a steering power which is determined from the output of the PID controller. Moreover, the steering power was limited from -6 to 6 to make the robot easy to control and it would affect both motors as the following code:

```
floor_turn=max(-turn_power,min(turn_power,floor(turn)));

left.Power = power + floor_turn;
left.SendToNXT();

right.Power = power - floor_turn;
right.SendToNXT();
```

where the `floor_turn` is the limited steering power.

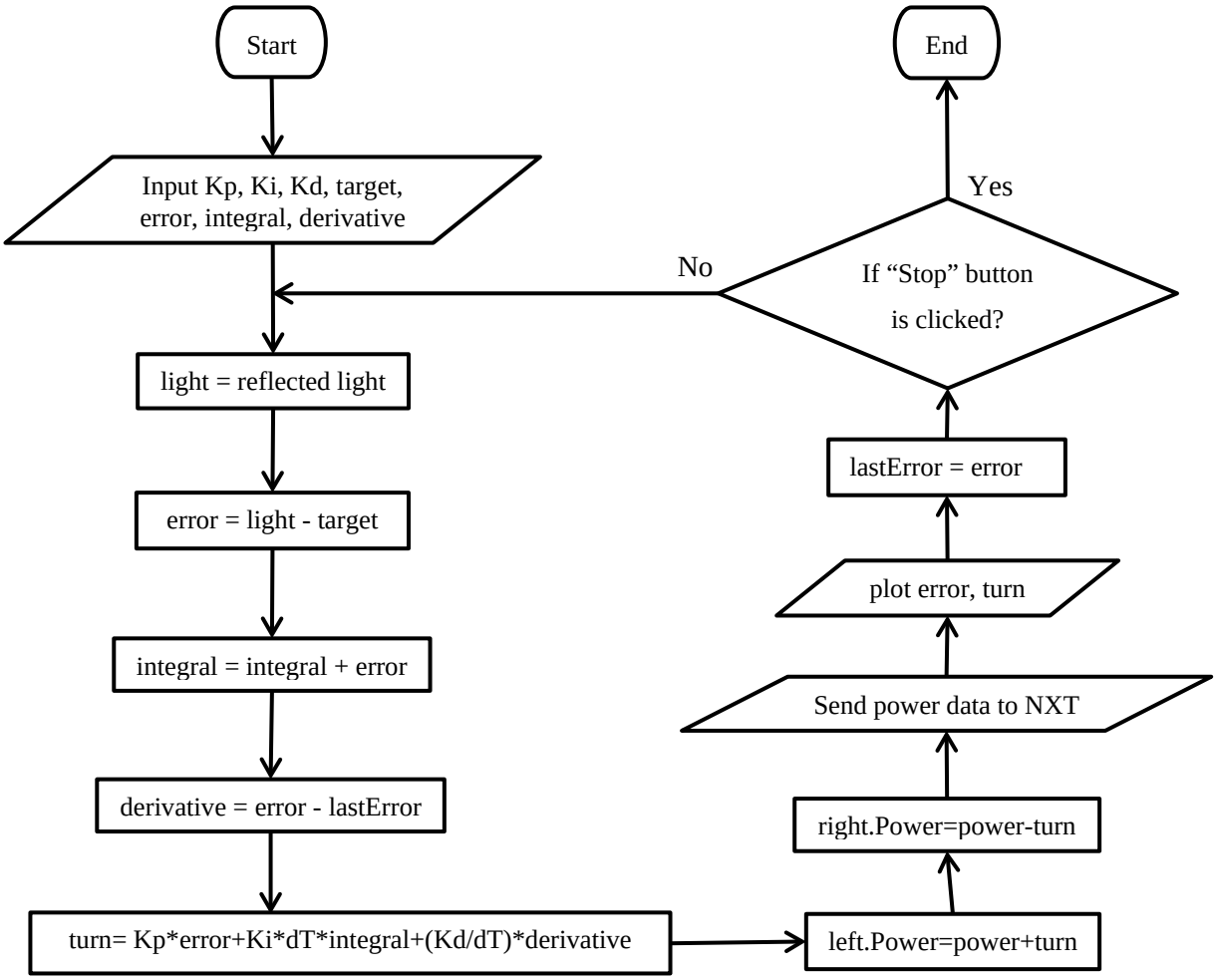

Figure 2.9: Core function flowchart of line following robot using LEGO Mindstorms NXT

The steering power which is the output of the PID controller was obtained by:

```
turn = Kp * error + Ki * dT * integral + (Kd/dT) * derivative;
```

where `error`, `integral` and `derivative` could be obtained as the flowchart shown in Figure 2.9. The error was estimated by subtracting the strength of the reflected light and the target light strength. The integral was estimated by accumulating the errors in each loop. And the derivative was estimated by subtracting the current error with the error estimated in the last loop.

3 Results

The graphical user interface (GUI) as shown in Figure 3.1 was designed with Graphical User Interface Design Environment(GUIDE) of MATLAB. All of the functions in the educational platform are integrated in this GUI.

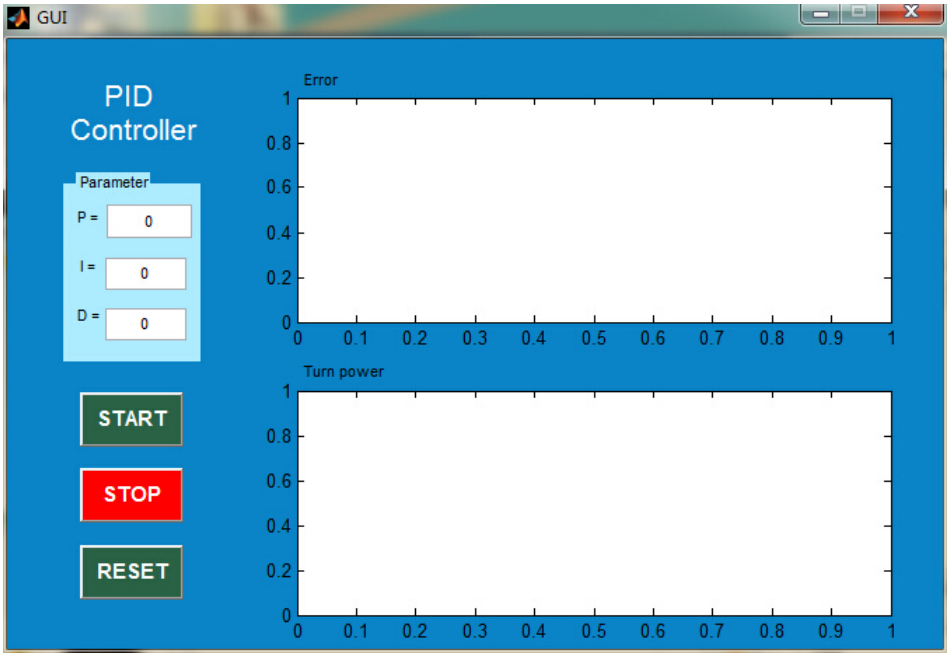

Figure 3.1: Graphic user interface

The three parameters in the PID controller can be adjusted as functions when the robot is not running. When “START” button is clicked, the GUI will callback the core function as shown in Figure 2.9 to make the program start. The loop of the core function is a while loop and it will run when `stopbit` is 1. The callback function of “STOP” button is changing `stopbit` to 0. Therefore, whenever the “STOP” button is clicked, the while loop of the core function will stop running. When the ”RESET” button is clicked, it will callback a function to make all of the data cleared and the robot will be ready to do the next operation. The “Error” coordinate can plot the measured error in real time and the “Turn power” can plot the actual differential powers between two motors.

3.1 P control type

The integral gain and the derivative gain were set to 0 to allow the robot controlled by proportional controller. The loop time was measured as 0.18s by measuring the operation time of 1000 loops. When the proportional gain was set to 16.6 as shown in Figure 3.2, the ultimate period was estimated as 4 second.

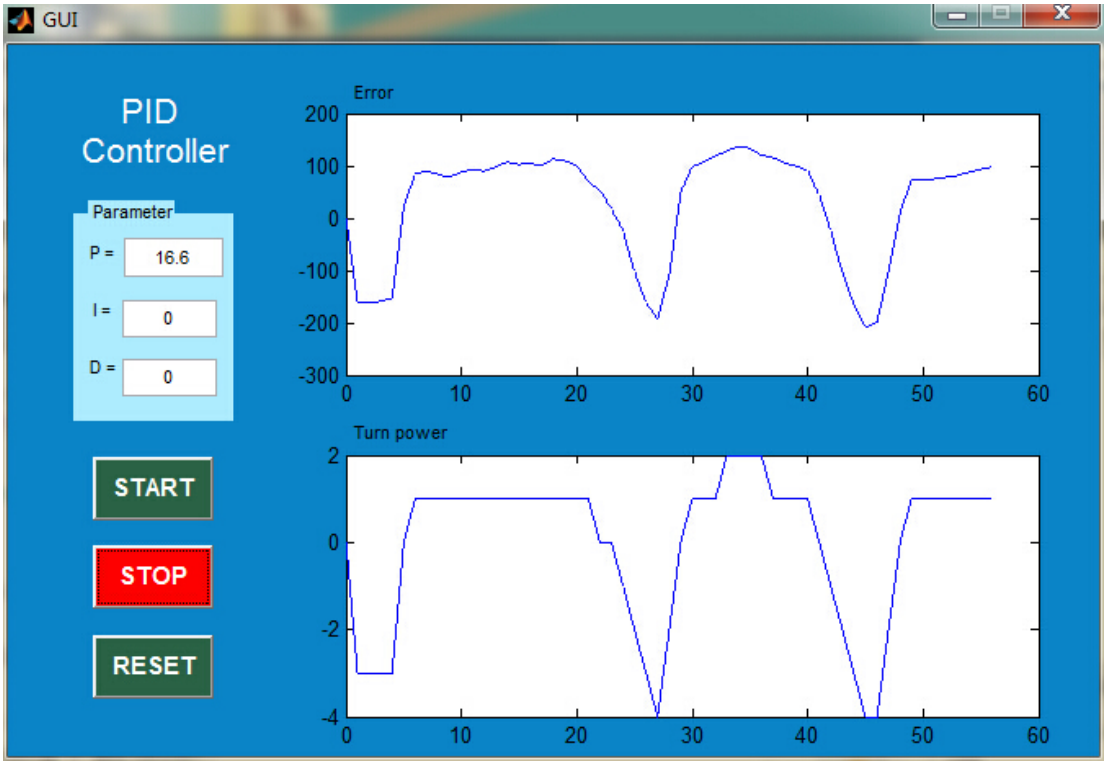

Figure 3.2: Corresponding result using P control type

The robot was continuously oscillating when we set P value larger than 16.6. The magnitude of error was increasing when the P value increased. If the P value is too high, the robot could be out of control. Conversely, the robot would not be able to adjust itself when the proportional gain is too small. Thus, the motion of the robot would be adjusted based on the current error when proportional controller was used with a suitable proportional gain.

3.2 PI control type

The commissioning parameters of PI control type which were obtained according to Table 1 are:

K_p = 7.5

K_i = 2.25

The robot could run properly with the above parameters when it runs along a straight line. However, the robot would not be able to turn when it runs along the curve of the line. Inevitably, the parameters were manually tuned to:

K_p = 15

K_i = 1.9

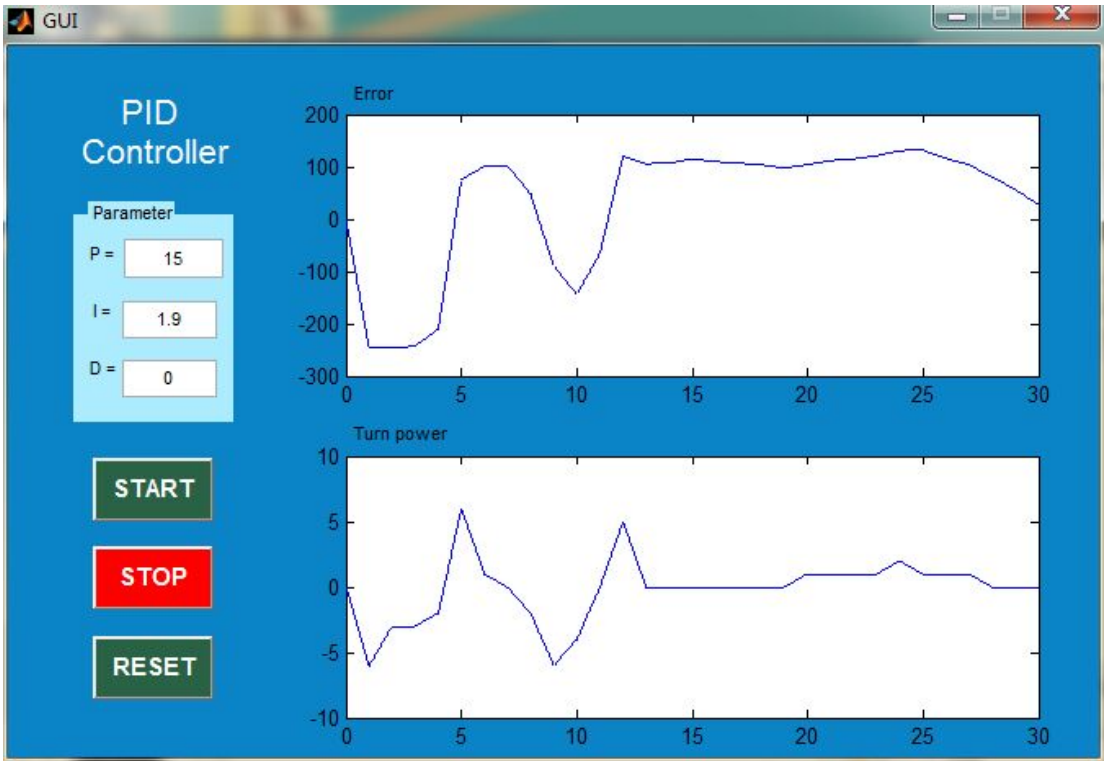

Figure 3.3: Corresponding result using PI control type

The integral term in this PI controller is the sum of the errors in all past loops. It could give the cumulative offset that should have been adjusted before. Hence, the performance of this system could be improved by using integral term. It is observed that the robot oscillated at the beginning and the error magnitude tend to be smaller and smaller. Comparing with the P control type, it can be observed that the integral term of this PI controller is effective.

3.3 PID control type

The commissioning parameters of PID control type which were obtained according to Table 1 are:

$K_p = 10$

$K_i = 5$

$K_d = 5$

For the same reason as it was mentioned in PI control type, the parameters were manually tuned to:

$K_p = 11$

$K_i = 3.6$

$K_d = 26.6$

The corresponding result of the PID control type is shown in Figure 3.4, which could be observed that, the performance of the robot was further improved.

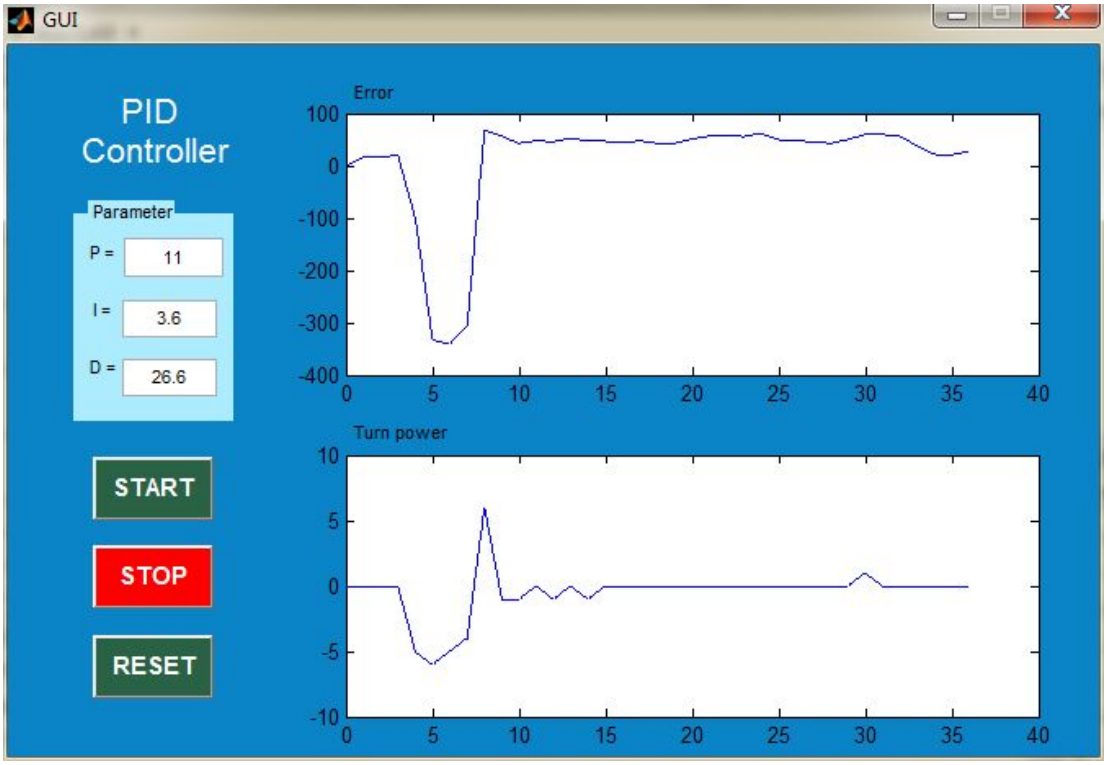

Figure 3.4: Corresponding result using PID control type

The derivative term is calculated by determining the current derivative of the error over time. It could give an offset by estimating the error trend. In other words, the “D” controller could predict the errors and thus reduce the errors in the future to improve the performance of the system.

4 Discussion

Compared with the experiment of PID controlling of water level in water tanks, the educational platform introduced in this paper is more intuitive, comprehensive and practical. The cost could be lower since all of the components are reusable in any other experiments.

However, this platform is more complex and unstable in operation. The stability of the robot could be reduced when the light environment changed since the ambient light could interference the measured result of the light sensor. To solve this problem, two or more light sensors could be applied to obtain a dynamic target by minimizing the differential value between the sensors instead of setting a static light intensity value. The delay of Bluetooth communication also reduced the stability of this system. The loop time of this PID controller was relatively high. It could be improved by using the built-in microcontrollers using NXT-G programming language. However, it's not available to collect the data in real-time when the programming language is NXT-G.

5 Conclusions

In this paper an educational PID controller model using LEGO Mindstorms NXT kit as a line following robot was presented. The algorithm and tuning methods of PID controllers was studied. It was found that Ziegler–Nichols tuning method is a suitable tuning solution in this thesis. However, it was a tricky problem that the robot might not be able to run along the curve if the commissioning parameters applied. Therefore, manual tuning is essential in practical application.

A MATLAB program using RWTH Toolbox was developed to control the robot and collect data in real time via Bluetooth. Discrete PID controller was realized in MATLAB program environment as well. The effectiveness of each parameter in PID controller was observed by adjusting the parameters. The test result shows that the PID controller could significantly improve the performance of the line following robot.

References

[1] Richard M. Felder and Linda K. Silverman, "Learning and teaching styles in engineering education." *Engineering Education 78.7*, pp: 674-681, 1988.

[2] Mondada Francesco, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Klaptocz Adam, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, Alcherio Martinoli, "The E-puck, a Robot Designed for Education in Engineering", *Proceedings of the 9th conference on Autonomous Robot Systems and Competitions*, Vol. 1, pp: 59-65, 2009.

[3] Hung-Shiang Chuang, Ying-Chun Chuang, Chun-Hsiang Yang, "Development of a Low-cost Platform for Control Engineering Education", *Proceedings of the 2014 International Conference on Machine Learning and Cybernetics*, pp: 444-448, July, 2014.

[4] Bennett Stuart, *A history of control engineering, 1930-1955.* pp: 67, Peter Peregrinus Ltd, 1993.

[5] Advanced Users - Developer Kit. Available: http://www.lego.com/en-gb/mindstorms/downloads

[6] RWTH - Mindstorms NXT Toolbox for MATLAB. Available: http://www.mindstorms.rwth-aachen.de/trac/wiki/Download

[7] LEGO NXT Motor. Available: http://trivox.tripod.com/Lego-nxt-Motor_direction.gif

[8] Wen Tan, Jizhen Liu, Tongwen Chen and Horacio J. Marquez, "Comparison of some well-known PID tuning formulas", *Computers and Chemical Engineering*, vol. 30, pp: 1416–1423, 2006.

[9] Ziegler J.G and Nichols N. B., *Optimum Settings for Automatic Controllers*, pp: 759–768, trans. ASME, 1942.

[10] Alexander Behrens, Linus Atorf, Robert Schwann, Bernd Neumann, Rainer Schnitzler, Johannes Ballé, Thomas Herold, Aulis Telle, Tobias G. Noll, Kay Hameyer, and Til Aach, "MATLAB Meets LEGO Mindstorms—A Freshman Introduction Course Into Practical Engineering", *IEEE Transactions on Education*, vol. 2, no. 2, May 2010.

Appendix

```
function varargout = GUI(varargin)
% Last Modified by GUIDE v2.5 16-Aug-2015 19:59:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)

COM_CloseNXT('all');
handle = COM_OpenNXT('bluetooth.ini');
COM_SetDefaultNXT(handle);

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% --- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

```
function P_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function P_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function I_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function I_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function D_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function D_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function P_KeyPressFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function text4_CreateFcn(hObject, eventdata, handles)

% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
global stopbit;
stopbit=0;
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
stop.

function stop_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
global stopbit;
stopbit=1; % When press the "stop" button in the GUI, stopbit will
change to 0 and the robot will stop running.
```

```
% Ziegler-Nichols tuning method, Ku=16.67, Tu=4, Ts=0.188
% Read parameters from the GUI
Kp=str2num(get(handles.P,'string'));

Ki=str2num(get(handles.I,'string'));

Kd=str2num(get(handles.D,'string'));

NXT_PlaySoundFile('! Attention', 0);

left = NXTMotor( MOTOR_B );
right = NXTMotor( MOTOR_C );
left.SmoothStart = false;
right.SmoothStart = false;

port =  SENSOR_3;
OpenLight(port, 'ACTIVE' ); % Active the light sensor

power = 10; % Set the max power of the motors
turn_power=6; % Set the max value of turning power

light = GetLight(port);
% Read the reflected light strength by the light sensor as the target


% Initializing

integral=0;
integ=0;
lastError=0;
derivative=0;
i=0;
k=1;
lastturn=0;
dT=0.188;

target = light;
fprintf('Reflected light is %d \n', light);

while stopbit

i=i+1;
pause( 0.05 )

error = target - light;

integral = integral + error; % Summarize the errors
integ = integ + abs(error);
derivative = error - lastError;



light = GetLight(port);
```

```
turn = Kp * error + Ki * dT * integral + (Kd/dT) * derivative;

turn=turn/1000;

floor_turn = max( -turn_power, min( turn_power,floor( turn ) )  );
if abs(floor_turn)>0.1*turn_power % Reduce the sensitivity
    floor_turn=floor_turn;
else floor_turn=0;
end
    left.Power = power + floor_turn;
    left.SendToNXT();

    right.Power = power - floor_turn;
    right.SendToNXT();

    X=linspace(i-1,i,50);
    Y1=linspace(lastError,error,50);
    Y2=linspace(lastturn,floor_turn,50);

    if i==k*200
       k=k+1;
       hold(handles.Error,'off');
       hold(handles.TurnPower,'off');
    else
       plot(handles.Error,X,Y1);
       hold(handles.Error,'on');
       plot(handles.TurnPower,X,Y2);
       hold(handles.TurnPower,'on');
    end
    lastError = error;
    lastturn=floor_turn;
    stopbit = stopbit;

end

NXT_PlaySoundFile('Woops', 0);

% Turn off the motors
left.Stop('off');
right.Stop('off');

% --- Executes during object creation, after setting all properties.
function Error_CreateFcn(hObject, eventdata, handles)
% set(handles.Error,'ylim',[-1000 1000]);

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
cla(handles.Error);
cla(handles.TurnPower);
```

```
% --- Executes during object creation, after setting all properties.
function TurnPower_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TurnPower (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% set(handles.TurnPower,'ylim',[-10 10]);

% Hint: place code in OpeningFcn to populate TurnPowe0072
```