CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter explains the concept of a line follower, control systems, and Real-Time Operating System (RTOS). The concept of line follower is divided into hardware and software that have been used in the previous project. The control system will clarify the concept of Proportional (P), Integral (I), Differential (D), Proportional Integral Differential (PID), root locus, and implementation PID in line follower robot. The Real Time Operating System RTOS will focus and describe the concept ChibiOS.

2.2 Line Follower Concept

In the concept of making line follower robot is divided into three parts: hardware and algorithms, control system, and RTOS. The contents of the section which will be explained derived from a review of previous studies and also from various sources.

2.2 Hardware

Hardware is physical aspect of devices. Designing and utilizing hardware correctly will establish good devices. In the manufacture the Arrow-bot line follower needs to conduct a study and reviews of the hardware components that will be used such as sensors, motors, etc. And also algorithm used. In this section will explain how previous research and a wide variety of sources in use in the hardware line follower.

2.2.1 Sensor

In the making of a line follower robot requires sensors to detect a line that will follow, the accuracy and capability of sensors are very important. A description of the sensors and algorithms that will used in the previous study are described as follows. In the following study line follower robot used phototransistor sensors and LED and optical sensors reflective as IR sensor. Explanation below will be divided into two, using digital and analog inputs.

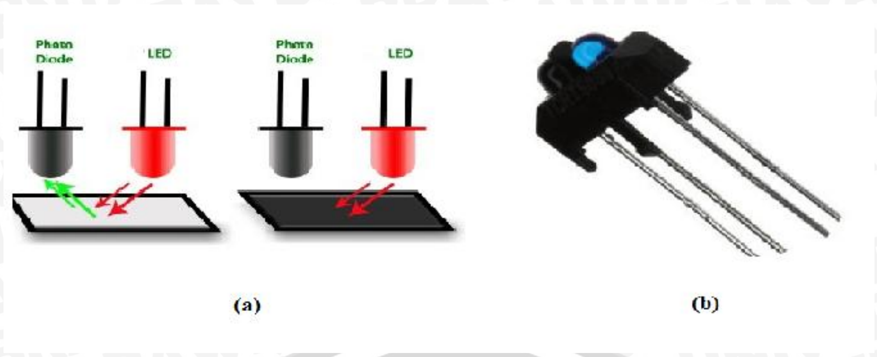

Figure 2.1 Sensor line detection via a) phototransistor with LED, and b) reflective optical sensor as IR sensor.

2.2.1.1 Digital Sensor

a) Banquet Serve Autonomus Robot (Mu`izz, 2011)

The banquet serve autonomous robot consists of four pairs of infrared sensors that are used as inputs. Each combination of sensors will give a different product and will determine the direction movement of the robot. Table 2.1 shows the microcontroller algorithms of banquet serve autonomous robot.

Table 2.1 Microcontroller algorithms for banquet serve autonomous robot to follow white line (Mu`izz, 2011).

Rx1	Rx2	Rx3	Rx4	Actions of robot
1	1	1	1	Stop
1	0	0	1	
1	0	1	1	Forward
1	1	0	1	
0	0	1	1	Turn right
0	1	1	1	
1	1	0	0	Turn left
1	1	1	0	

This robot designed with five modes of operation, turn left/right, forward/reverse, and stop. To move forward, both of the motors are turn on and rotate forward simultaneously. For the turning right, the left motor are turn on and right motor are turn of and vice versa for turning left (Mu`izz, 2011).

The outer sensors are the most important for the line tracking. Figure 2.2 shows the mode of movement banquet serve autonomous robot.

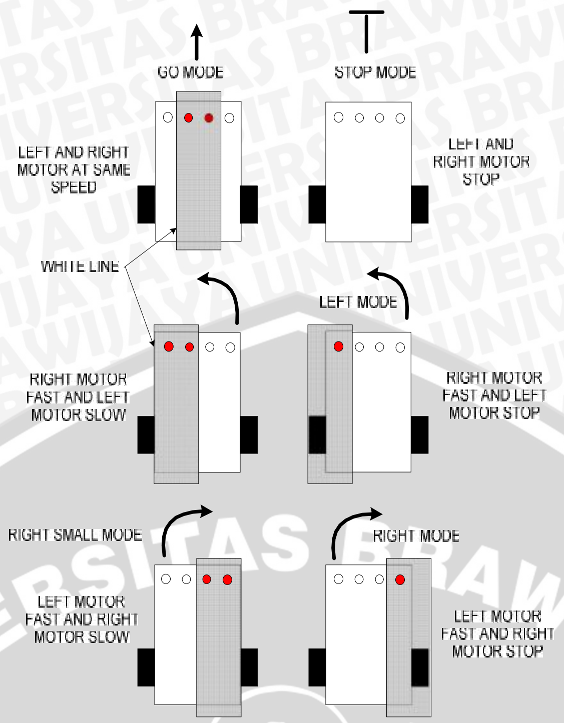
GO MODE
STOP MODE
LEFT AND RIGHT MOTOR AT SAME SPEED
LEFT AND RIGHT MOTOR STOP
WHITE LINE
LEFT MODE
RIGHT MOTOR FAST AND LEFT MOTOR SLOW
RIGHT MOTOR FAST AND LEFT MOTOR STOP
RIGHT SMALL MODE
RIGHT MODE
LEFT MOTOR FAST AND RIGHT MOTOR SLOW
LEFT MOTOR FAST AND RIGHT MOTOR STOP

Figure 2.2 The mode of movement for banquet serve autonomous robot (Mu`izz, 2011).

b) Line Follower MultipleX Robot (Fahmi, 2010)

In this robot used eight sensors IR with phototransistor and LED. With many sensors readings logic mapping sensor line to the line will be more accurate. Thus, it would be a lot of conditions that can be mapped by the ease in designing programming algorithm. The following Table 2.2 describes the sensor algorithm line follower MultipleX robot works.

Table 2.2: Sensors algorithm for line follower MultipleX robot.

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8	Value (PV)
1	1	1	1	1	1	1	0	-7
1	1	1	1	1	0	0	0	-6
1	1	1	1	1	1	0	0	-6
1	1	1	1	1	1	0	1	-5
1	1	1	1	0	0	0	1	-4
1	1	1	1	1	0	0	1	-4
1	1	1	1	1	0	1	1	-3
1	1	1	0	0	0	1	1	-2
1	1	1	1	0	0	1	1	-2
1	1	1	1	0	1	1	1	-1
1	1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1	1
1	1	0	0	1	1	1	1	2
1	1	0	0	0	1	1	1	2
1	1	0	1	1	1	1	1	3

1	0	0	0	1	1	1	1	4
1	0	0	1	1	1	1	1	4
1	0	1	1	1	1	1	1	5
0	0	0	1	1	1	1	1	6
0	0	1	1	1	1	1	1	6
0	1	1	1	1	1	1	1	7
1	1	1	1	1	1	1	1	8/-8

In this project PV as reference to find error. Where error = set point - PV. Set point in this case using the value 0. Figure 2.3 is illustration of how the sensor works.

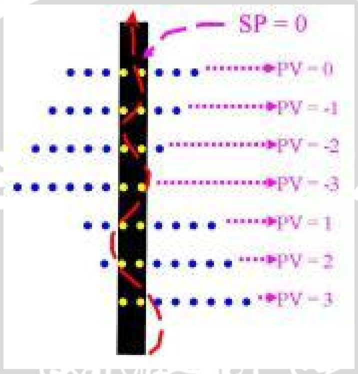

Figure 2.3 Ilustration how sensor works.

2.2.1.1 Analog Sensor

a) An Intelligent Line-Following Robot Project for Introductory Robot Courses (Su J.H, 2010)

On this robot used seven sensors reflective optical sensor. Where each sensor is calibrated to get the right value following is the calculation of the sensor calibration.

The values in Figure 2.4 show that the seven reflective optical sensors actually do not give identical outputs even if the test conditions are the same. The calibration procedure tries to map the output values of each reflective optical sensor with respect to 20% and 80% of grey to the same maximum and minimum values using the following formula (Su J.H, 2010):

$$y_{jo} = y_{min} + \frac{y_{max}-y_{min}}{x_{max}-x_{min}}(x_{ji} - x_{minj}) \quad \text{(Eq 2.1)}$$

Where $x_{max,i}$, $x_{min,i}$, are the maximum and minimum values of the *i*th reflective optical sensor outputs; y_{max},y_{min} are the predefined maximum and minimum values for all reflective optical sensor outputs; *x ji* is the *jth* output value for the *ith* reflective optical sensor, and *y jo* is the normalised output value for *xji*. It can be seen in Figure 2.4 that the output values of different reflective optical sensors are quite similar to one another after the calibration using Equation (2.1). Once the calibration procedure is done, the output values of these seven reflective optical sensors are ready to be used in estimating the line position (Su J.H, 2010).

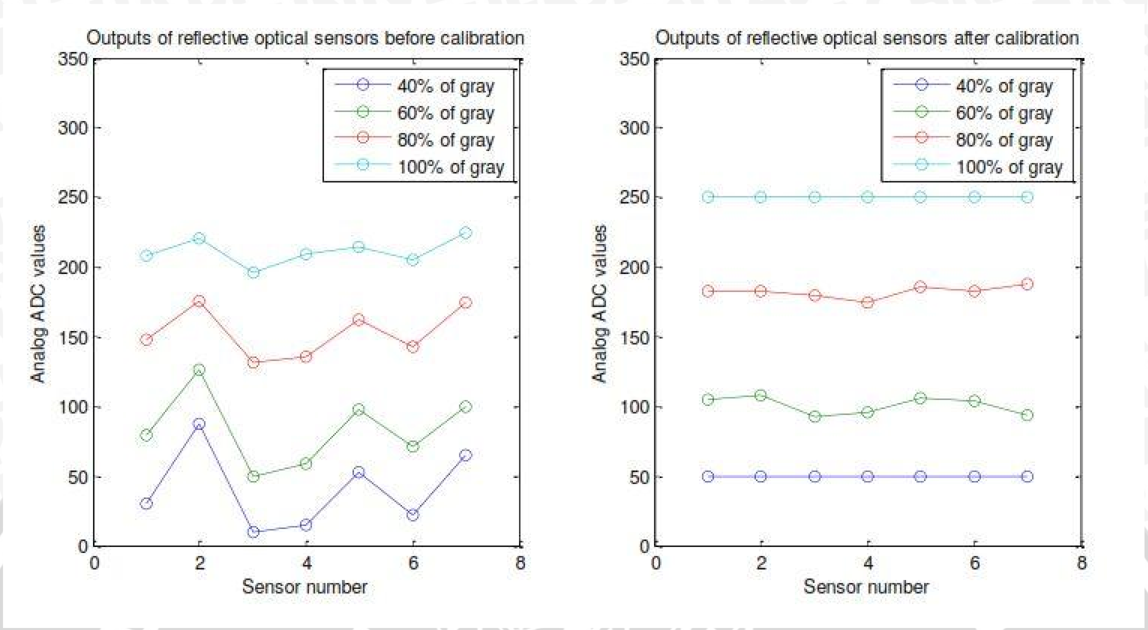

Figure 2.4 The reflective optical sensor outputs for different grey scales before (left) and after calibration (right) (Su J.H, 2010).

Line Detection Algorithm via Weighted Average

This technique is the commonly used in defuzzification procedures of fuzzy systems (Zimmermann, 2001). It can also be viewed as finding the center of mass. Suppose that the coordinate of the 7 reflective optical sensors are *x*, respectively, and the corresponding analog output values are *y*0, *y*1, *y*2, *y*3, *y*4, *y*5, *y*, as shown in Figure 2.5 The estimated line position can then be calculated by the following weighted average formula:60,*x*1, *x*2, *x*3, *x*4, *x*5, *x*6 (Su J.H, 2010).

$$x = \frac{\sum_{i=0}^{7} x_1 y_1}{\sum_{i=0}^{7} y_1} = \frac{3(y_6-y_0)+2(y_5-y_1)+(y_4-y_2)}{\sum_{i=0}^{7} y_1} \qquad \text{(Eq. 2.2)}$$

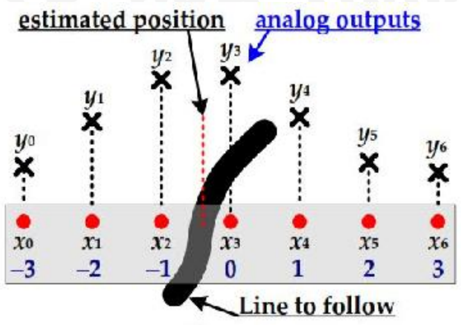

Figure 2.5 The line detection algorithm via weighted average (Su J.H, 2010).

b) A Hands-on Laboratory for Autonomous Mobile Robot Design Courses (Lee C.S, 2008)

On this robot in the process is similar (An Intelligent Line-Following Robot Project for Introductory Courses). In this project also be calibrated with the same method. But used different algorithm, this robot used line detection via quadratic interpolation algorithm.

Line Detection Algorithm via Quadratic Interpolation

Let the coordinate of the left most sensor be - 3, and the distance between two consecutive sensors be 1. The output of the reflective optical sensor circuit in Figure 2.6 is higher when the sensor is closer to the black line. Therefore, one can always find 3 consecutive sensors with higher output readings than the other 4 sensors as shown in Figure 2.6. Assume that the coordinate of these 3 sensors are *x*1, *x*1+1, and *x*+2, and the true shape of the sensor output values in the range of [*x*1, *x*1+2] can be approximated by a quadratic curve (Lee C.S, 2008). One can then find the following relationships between the coordinate of the sensors and the output values:

$$y_1 = a{x_1}^2 + bx_1 + c,\; y_2 = a(x_1+1)^2 + b(x_1+1) + c,$$

$$y_3 = a(x_2+1)^2 + b(x_2+1) + c \quad \text{(Eq. 2.3)}$$

The coordinate value at which the output value of the quadratic curve is the maximum is considered as the true position of the line (Lee C.S, 2008). By using the basic calculus, one would know that the coordinate value is:

$$x = -\frac{b}{2a}, and\ a = \frac{y_1+y_2-2y_2}{2}, b = y_2 - y_1 - 2ax_1 - a \quad \text{(Eq. 2.4)}$$

It is assumed that the coordinate for the center position of the line-following robot is 0. Therefore, the error *e* between the line position and the center position of the robot is e= 0-x =-x.

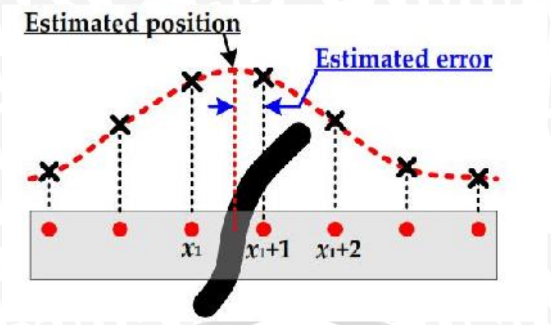

Figure 2.6 The line detection algorithm via quadratic interpoation (Lee C.S, 2008).

2.2.2 Kinematic Model a Differential Drive Robot

Consider a simplified diagram of a mobile robot comprising of two wheels presented in Figure 2.7(a). For this robot,

– L is the distance between two wheels,

– R is the radius of each wheel,

– w_l is the rate at which left wheel is turning.

– w_r is the rate at which right wheel is turning.

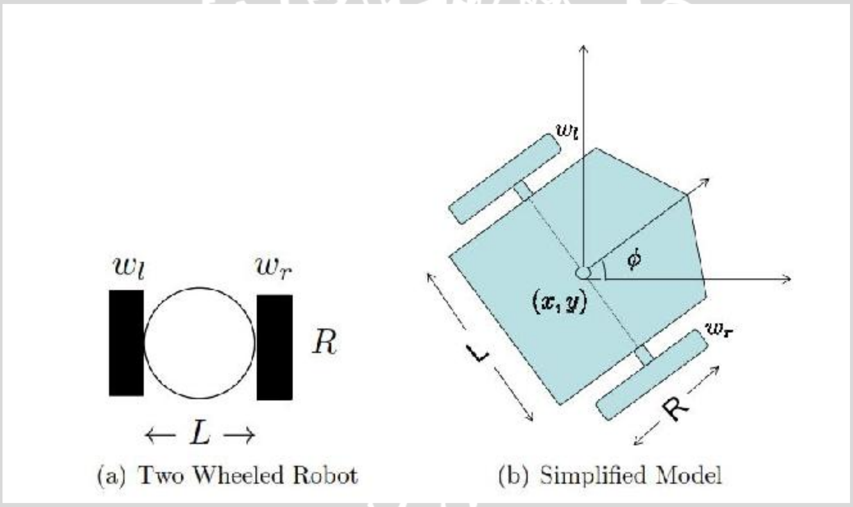

Figure 2.7: Mobile robot (Dudek and Jenkin, 2000).

The objective of this robot is to move in a planar region along some trajectory, so the states of this system are (x, y, ϕ), where (x, y) correspond to the the location of the center point of the robot in a Euclidean plane and ϕ corresponds to the orientation of the robot see Figure 2.7(b). A differential drive robot contains two wheels each of which is equipped with a separate DC motor that controls its rotation. For this system, the inputs are going to be w_r and w_l, i.e., rates at which left and right wheel are turning. This robot is called differential drive because the steering of this system is based on difference in the turning rates of the two wheels. If w_r and w_l, then the robot will travel in a straight line. If w_r / = w_l, then the robot will turn in the direction of the wheel that is rotating at a slower rate. In order to model the trajectory followed by this robot, we have to track the movement of the center point of the robot which is (x, y). Given w_r,w_l, R, and L, the forward kinematic problem is to find the trajectory of the robot (x(t), y(t)).The robot trajectory is given by the following set of equations (Dudek and Jenkin, 2000).

$$\dot{x} = R\left(\frac{w_r+w_l}{2}\right)\cos(\phi) \qquad \text{(Eq. 2.5)}$$

$$\dot{y} = R\left(\frac{w_r+w_l}{2}\right)\sin(\phi) \qquad \text{(Eq. 2.6)}$$

$$\dot{\phi} = \frac{R}{L}(w_r + w_l) \qquad \text{(Eq. 2.7)}$$

Uncycle Model

To control a differential drive mobile robot, the input to the system has to be wr and wl. However, when authors are designing this system, it is inconvenient to think in terms of wheel rotations. A more natural set of control parameters is the speed at which our robot should move and its orientation. Therefore, instead of wr and w, authors design our system in terms of v and w, where v is the linear speed of the robot and w is the angular velocity, and a model that uses this set of parameters for is a unicycle model (Anonymous, 2015). According to this model,

$$\dot{x} = V cos(\phi) \qquad \text{(Eq. 2.8)}$$

$$\dot{y} = V sin(\phi) \qquad \text{(Eq. 2.9)}$$

$$\dot{\phi} = w \qquad \text{(Eq. 2.10)}$$

Now, two sets of equations for $(\dot{x}, \dot{y}, \dot{\phi})$. Equating both of them yields

$$w_r = \frac{2v+wL}{2R} \qquad \text{(Eq. 2.11)}$$

$$w_l = \frac{2v-wL}{2R} \qquad \text{(Eq. 2.12)}$$

Equations (2.11& 2.12) allow us to design our system in terms of more convenient parameter, i.e. (v, w) and then use the values of these parameters to find the actual inputs to our system, i.e., (wr,wl). Once the input parameters (wr, wl) are computed that guarantee to drive the robot to the desire location, the next step is to find the motor parameters that will ensure that each wheel rotates at the desired rate (Anonymous, 2015).

2.2.3 Motor DC

An electrical machine functions as an electric motor in the event of electrical energy conversion process into mechanical energy in it. DC motor is a motor that requires a supply voltage of the coil in the direction of the anchor and field coils to be converted into mechanical energy. Based on the characteristics, direct current motor has a wide area round setting compared with alternating current motors, so that is still widely used in factories production machines require extensive round setting (Aisyah and Ya'umar, 2012)

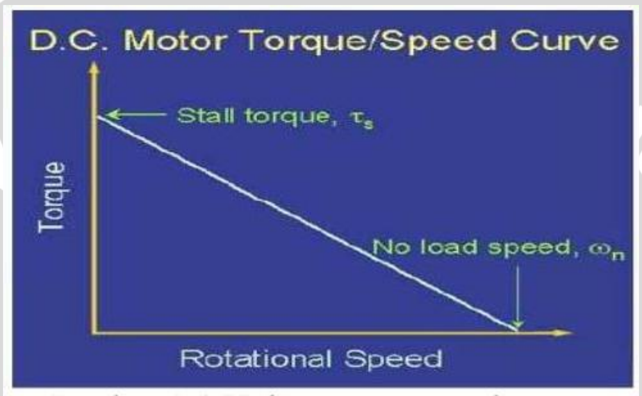

Figure 2.8 The relationship between power with torque / speed (Aisyah and Ya'umar, 2012).

From the graph 2.8 shows that the torque is inversely proportional to the speed of rotation, in other words there is a trade off between a large torque generated by the motor rotation speed of the motor. Two important characteristics seen from the graph, namely:

a. Stall torque, indicates the point on the graph where the maximum torque, but there is no rotation of the motor.

b. No load speed, showing a point on the graph where there is a maximum rotation speed, but there is no load on the motor (Kumara, 2010).

Which is often used to the velocity tuning and the position adjustment. In controlling the armature current motor, if detained field coil current constant, and armature current is controlled by the voltage Va. In this case, the motor torque generated linear to current motor (Meshram and Kanojiya, 2012).

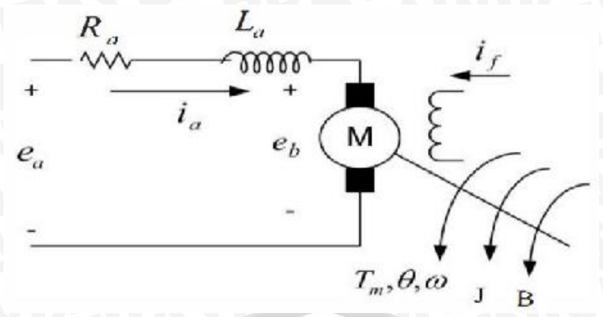

Figure 2.9 The control equivalent circuit of DC motor (Meshram and Kanojiya, 2012).

Where,

R_a	: Armature resistance
L_a	: Armature inductance
i_a	: Armature current
i_f	: Field current
e_a	: Input voltage
e_b	: Back electromotive force (EMF)
T_m	: Motor torque
ω	: An angular velocity of rotor
J	: Rotating inertial measurement of motor bearing
K_b	: EMF constant
K_T	: Torque constant
B	: Friction constant

Because the back EMF eb is proportional to speed ω directly, then

$$e_b(t) = K_b \frac{d\theta}{dt} = K_b \omega(s) \qquad \text{(Eq. 2.13)}$$

Making use of the KCL voltage law can get

$$e_a(t) = R_a\, i_a(t) + L_a \frac{di_a(t)}{dt} + e_b(t) \qquad \text{(Eq. 2.14)}$$

From Newton law, the motor torque can be obtained as

$$T_m(t) = J\frac{d^2\theta(t)}{dt} + B\frac{d\theta(t)}{dt} = K_T i_a(t) \qquad \text{(Eq. 2.15)}$$

Take (Eq. 2.13), (Eq. 2.14), and (Eq. 2.15) into Laplace transform, respectively, the equations can be formulated as

$$E_a(s) = (R_a + L_a S)I_a(s) + E_b(t) \qquad \text{(Eq. 2.16)}$$

$$E_b(s) = K_b \omega(s) \qquad \text{(Eq. 2.17)}$$

$$T_m(t) = B\omega(s) + J\omega(s) = K_T i_a(s) \qquad \text{(Eq. 2.18)}$$

Figure 2.10 describes the DC motor armature control system function block diagram from equations (Eq. 2.13) to (Eq. 2.18).

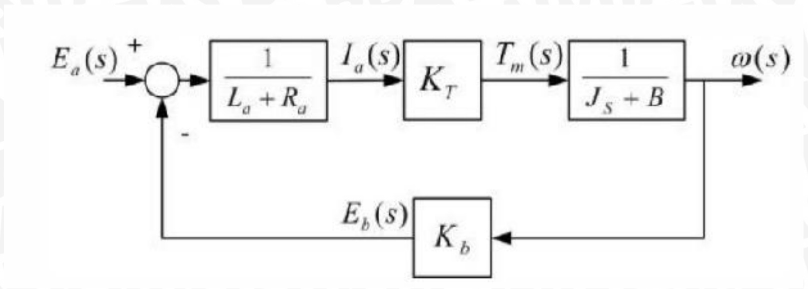

Figure 2.10 System function block diagram (Meshram and Kanojiya, 2012).

The transfer function of DC motor speed with respect to the input voltage can be written as follows,

$$\frac{\omega(s)}{T_m(s)} = \frac{K_T}{(L_a(s)+R_a)(J(s)+B)+K_bK_T} \quad \text{(Eq. 2.19)}$$

DC motor speed depends on the magnitude of duty cycle is given in the DC motor. In the PWM signal, the signal frequency is constant while varying the duty cycle of 0% -100%. By adjusting the duty cycle will obtain the desired output (Ardyani, 2012). Duty cycle is the magnitude of the control signal supplied to the motor. The equation for calculating duty cycle shown in equation 2.20 with Ton is high logic period, and T is whole period.

Duty cycle = Ton/T x 100% (Eq. 2.20)

PWM signal is generally shown in Figure 2.11.

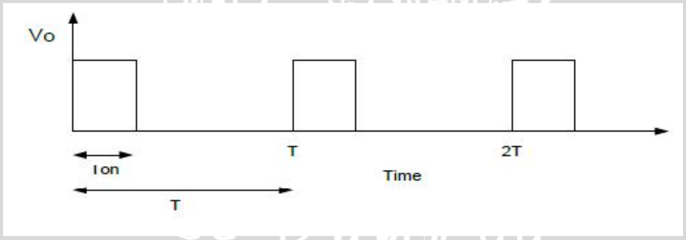

Figure 2.11 PWM signal (Ogata K, 1997).

Description:

Ton = High logic Period

T = Whole Period

Vo = Amplitude

2.3 Control System

Control system was designed to do and accomplish specific tasks. The main requirement is the control system must be stable. In addition to the absolute stability, the system should have a relative stability, which is a benchmark for the quality of the stability of the system by analyzing the extent to which the boundaries of the system stability when subjected to disturbance (Ogata K, 1997). In addition, the analysis was also conducted to determine how the speed of the system in response to the input, and how mitigation against the surge (over shoot).

A system is said to be stable if given the interference, the system will return to a steady state in which the output is in a steady state as there is no interference. The system is said to be stable if its output oscillates continuously when subjected to a disturbance. Because a control system typically involves the energy storage system output when given an input, can not follow the input simultaneously, but shows the transient response in the form of a damped oscillations before reaching a steady state (Lewis, 1997) In the section below will explain proportional integral differential (PID) and root locus method in the design of proportional integral differential (PID) controller.

2.3.1 The Proportional Controller (P)

Controller proportional is a controller that has a characteristic speed up the response. The relationship between the controller output m (t) and a driving error signal e (t) shown in equation 2.21:

$$m(t)=K_p\, e(t) \qquad \text{(Eq. 2.21)}$$

or, in the amount of Laplace transforms shown in equation 2.22

$$\frac{M(s)}{E(s)} = Kp \qquad \text{(Eq. 2.22)}$$

Where Kp is the proportional sensitivity or reinforcement.

Whatever the form of the actual mechanisms and any form of motive power, proportional controller is basically an amplifier with adjustable amplification (Ogata K, 1997). Proportional controller block diagram shown in Figure 2.12.

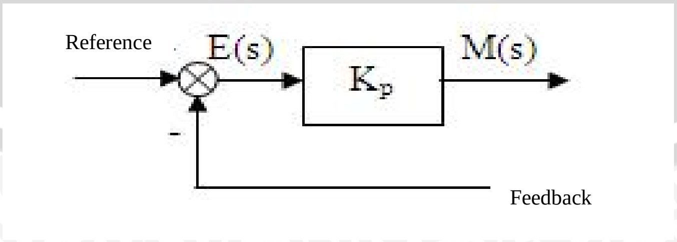

Figure 2.12 Block diagram proportional controller (Ogata K, 1997).

If a controler function can be devised that improves the inherent stability of a system, the conflict between relative stability and loop gain is eased, and the designer has an opportunity to attain improvements in a board range of performance characteristic.

2.3.2 The Integral Controller (I)

Integral controller has the ability to reduce the offset left by a proportional controller. Controller output price m (t) is changed at a rate that is proportional to the error signal driving e (t). Ki controller equation shown in equation 2.23. (Ogata K, 1997).

$$\frac{dm(t)}{dt} = Kie(t) \qquad \text{(Eq. 2.23)}$$

Ki is the integral reinforcement. Figure 2.13 shows a block diagram of an integral controller.

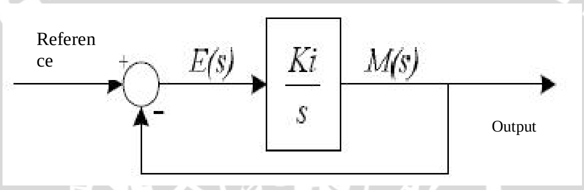

Figure 2.13 Block diagram integral controller (Ogata K, 1997).

2.3.3 The Differential Controller (D)

This controller is used to improve or speed up the transient response of a control system by increasing the phase leads to the strengthening of control and reduce the phase lag reinforcement (Ogata K, 1997).Differential controller output can not be issued if there is no change in input, besides the differential controller can not be used for processes that contain noise. The relationship between the controller output m (t) and a driving error signal e (t) is expressed by equation 2.24.

$$\frac{M\,(s)}{E(s)} = Kd.s \qquad \text{(Eq. 2.24)}$$

Kd is the differential reinforcement. Figure 2.14 shows a block diagram of differential controller.

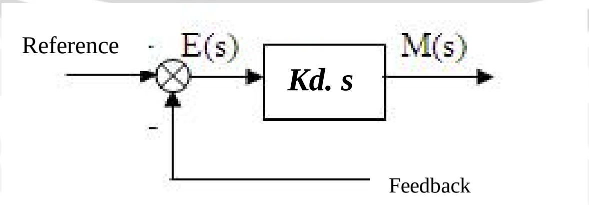

Figure 2.14 Block diagram differential controller (Ogata K, 1997).

2.3.4 Proportional Integral Differential Controller (PID)

The combined action of proportional control, integral, and differential has advantages compared with each of the three control actions. Each - each controller P, I, and D serves to accelerate the reaction system, eliminating the offset, and gain extra energy when the load changes.

The PID controller equation can be expressed in equation 2.25:

$$m(t) = Kp \,.\, e(t) + \frac{Kp}{Ti}.\, e(t)dt + Kp.Td\frac{de(t)}{dt} \qquad \text{(Eq. 2.25)}$$

In the Laplace transformation is expressed in equation 2.26:

$$\frac{M(s)}{E(s)} = Kp\;(1 + \frac{1}{Ti\,.s} + Td.s) \qquad \text{(Eq. 2.26)}$$

Ti is the integral time and Td is the time derivative. Figure 2.15 shows a block diagram of the PID controller.

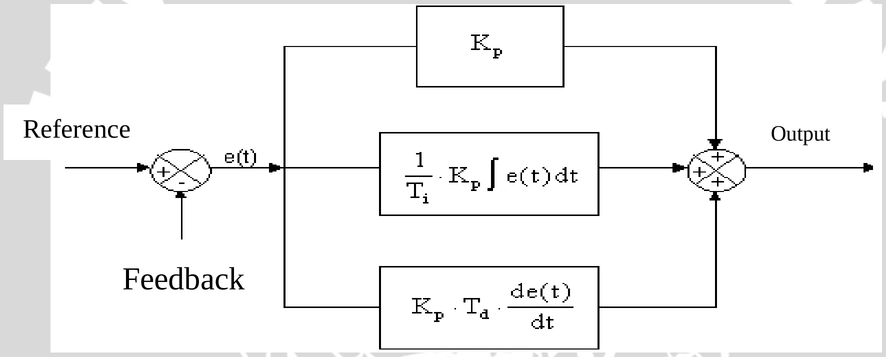
K_p
Reference
e(t)
$\frac{1}{T_i}\cdot K_p \int e(t)dt$
Output
Feedback
$K_p \cdot T_d \cdot \frac{de(t)}{dt}$

Figure 2.15 Block diagram PID controller (Ogata K, 1997).

2.3.5 Controller Design Method Proportional Integral Differential (PID) Using Root Locus.

Closed-loop control system design using root locus allowed to set at least some pole lies the closed-loop system that can adjust the transient response at a certain level and its influence on the steady-state response [15]. Analytical procedures PID controller design using root locus method described in Feedback Control System by Charles L. Phillips and Royce D. Harbour can be seen in Figure 2.16 below:

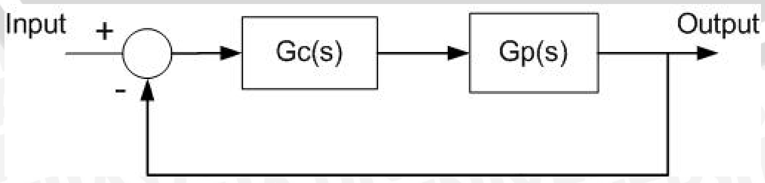

Figure 2.16 Close loop system (Ogata K, 1997).

For these systems, the characteristic equation is given equation 2.27

$$1 + Gc(s)Gp(s) = 0 \qquad \text{(Eq. 2.27)}$$

Suppose the desired root locus through $s = s_1$, then the result of the equation shown in equation 2.28.

$$Gc(s_1)Gp(s_1) = -1$$
$$Gc(s_1)\left|Gp(s_1)\right|e^{j\cdot} \;\; = 1e^{j\Pi} \qquad \text{(Eq. 2.28)}$$

PID controller transfer function after Laplace transform is expressed by equation 2.29.

$$Gc(s) = Kp + \frac{Ki}{s} + Kd\ s \qquad \text{(Eq. 2.29)}$$

Calculation of the equation 2.28 shown in equation 2.30,

$$Gc(s_1) = \frac{1}{\left|Gp(s_1)\right|}e^{j(\Pi - \cdot\;)} \qquad \text{(Eq. 2.30)}$$

Substitution equation 2.30 on equation 2.29, equation 2.31,

$$Kd\ s_1^2 + Kp\,s_1 \;+ Ki = \frac{e^{j(\Pi - \cdot\;)}}{\left|Gp(s_1)\right|} \qquad \text{(Eq. 2.31)}$$

With

$$s_1 = \left|s_1\right|e^{j} \qquad \text{(Eq. 2.32)}$$

The results from the substitution of equation 2.32 to 2.31 equations, obtained in equation 2.33.

$$Kd\left|s_1\right|^2\left(\cos 2\;\; + j\sin 2\;\;\right) + Kp\left|s_1\right|\left(\cos\;\; + j\sin\;\;\right) + Ki$$
$$= \frac{\left|s_1\right|}{\left|Gp(s_1)\right|}\left[\cos(\;\; + \Pi - \cdot\;) + j\sin(\;\; + \Pi - \cdot\;)\right] \qquad \text{(Eq. 2.33)}$$

Equating real and imaginary real with the imaginary, the result in equation 2.34.

$$\begin{bmatrix} \left|s_1\right|^2 & \left|s_1\right|\cos \\ \left|s_1\right|^2 & \left|s_1\right|\sin \end{bmatrix}\begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} \dfrac{\left|s_1\right|}{Gp(s_1)}\cos(\;\; + \Pi + \cdot\;) - Ki \\ \dfrac{\left|s_1\right|}{Gp(s_1)}\sin(\;\; + \Pi + \cdot\;) \end{bmatrix} \qquad \text{(Eq. 2.34)}$$

Or can be shown in equation 2.35.

$$\begin{bmatrix} \left|s_1\right|^2 & \left|s_1\right|\cos \\ \left|s_1\right|^2 & \left|s_1\right|\sin \end{bmatrix}\begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} -\dfrac{\left|s_1\right|}{Gp(s_1)}\cos(\cdot\; - \;\;) - Ki \\ \dfrac{\left|s_1\right|}{Gp(s_1)}\sin(\cdot\; - \;\;) \end{bmatrix} \qquad \text{(Eq. 2.35)}$$

From the equation it can be seen that for the design of PID controllers, one of three reinforcement Kp. Ki, Kd, should be determined in advance. As for the design of PI or PD, corresponding strengthening of the equation is made equal to zero (Philip and Harbor, 1996)

2.3.6 Controller Proportional Integral Differential (PID) in Line Follower Robot

In this section to explain how the implementation of PID controller in line follower robot that has been done by previous study.

a) Implementation of PID Control to Reduce Wobbling in A Line Following Robot (Anirudh, S.N.,et all, 2013)

From this study they initialized the various constants such as Kp, Ki and Kd. They also give a value to the Tp Variable, which is the maximum speed given to the wheels for straight condition. Also they initialize last error, which is the previous error value & other variables such as integral, derivative and proportional. They set the value of the Measured Position Variable for each sensor condition (Anirudh, S.N.,et all, 2013). These are the values we set are as in Table 2.3.

Table 2.3: Value for measured position.

Sensor Measured Position	Given Value
0-0-0	100
0-0-1	2
0-1-1	1
0-1-0	0
1-1-0	-1
1-0-0	-2
1-1-1	0

Then they calculated the error value as

Error = Measured Value – Target Position (Eq. 2.36)

A higher error value tells us that the robot is farther away from the line, than a lower error value. They then set the integral value as sum of current integral value & the error. The derivative value was updated to the sum of current derivative value & the last error (Anirudh, S.N.,et all, 2013).

They then calculated TURN Value as the sum of the product of ERROR Value & the Kp constant, product of Ki constant and integral and the product of Kd constant & the derivative. They then give the output to the left wheel and the right wheel as a

PWM (Pulse Width Modulation) output, which directly controls the speed of the motor (Anirudh, S.N.,et all, 2013).

LeftPWM = Tp + TURN (Eq. 2.37)

RightPWM = Tp – TURN

This results in both the wheels moving at different speed, which depends on the sensor input and is computed above. At the end of the program they set the last error value as current error value (Anirudh, S.N.,et all, 2013).

This algorithm keeps repeating in a loop to give an effective PID algorithm. The above explained computation happens many times in a second to give an efficient line following robot (Anirudh, S.N.,et all, 2013).

b) A Differential Steering Control with Proportional Controller for An Autonomous Mobile Robot (Saidonr,et all, 2011)

This research focused on analyzing closed-loop properties of PID controllers and improving tuning on closed-loop stability to get desired speed. PID compares the measure output, the speed, to the desired value. Proportional control makes duty cycle is increased when the speed too low. Derivative control makes duty cycle decreased when the speed increases quickly. Integrating control make the duty cycle increased when speed has been consistently too low over a period of time. All this effects (PID) are combined into one resulting of the duty cycle (Saidonr,et all, 2011).

PID controller compares the setpoint (SP) to the process variable (PV) to obtain the error *(e)*.

$$e = SP - PV \quad \text{(Eq. 2.38)}$$

Then the PID controller calculates the controller action, *u(t)*, where *Kc* is controller gain.

$$u(t) = K_c\left(e + \frac{1}{T_i}\int_0^t e\,dt + T_d\frac{de}{dt}\right) \quad \text{(Eq. 2.39)}$$

Closed-loop tuning is very accurate, the process have to in steady-state oscillation and observe the PV on a graph. The following step is to perform the closed loop manual tuning (Saidonr,et all, 2011).

- Set both the derivative time and the integral time on PID controller to 0 Carefully increase the proportional gain (*Kc)* is small increments. Make a small change in SP to disturb the loop after each increment. As increased *Kc*, the value of PV should begin to oscillate. Keep making changes until the oscillation is sustained, neither growing nor decaying over time.

- Increase integral gain ($KI=K_c/T_i$) until any offset is correct in sufficient time for the process. However, too much K will cause instability.
- Finally, increase derivative gain ($K_d=K_cT_d$) until the loop is acceptably quick to reach its reference after a load disturbance. However, too much *Kd* will cause excessive response and overshoot.

2.4 Real-Time Operating System (RTOS)

A Real-Time Operating System (RTOS) is a computing environment that reacts to input within a specific time period. A real-time deadline can be so small that system reaction appears instantaneous. The term real-time computing has also been used, however, to describe "slow real-time" output that has a longer, but fixed, time limit. Real time operating system is how responsive the operating system is in servicing internal and external events (Pise, S.J, 2011)

2.4.1 ChibiOS

ChibiOS/RT is designed for deeply embedded real time applications where execution efficiency and compact code are important requirements. This RTOS is characterized by its high portability, compact size and, mainly, by its architecture optimized for extremely efficient context switching (ChibiOS, 2015).

ChibiOS has adventage such as efficient and portable preemptive kernel, Dynamic extensions, dynamic objects are supported by an optional layer built on top of the static core and Support for priority inheritance algorithm on mutexes. HAL component supporting a variety of abstract device drivers: Port, Serial, ADC, CAN, EXT, GPT, I2C, ICU, MAC, MMC, PWM, RTC, SDC, SPI, UART, USB, USB-CDC. Extensive test suite with benchmarks (ChibiOS, 2015).

A certain set of minimum system requirements must be satisfied in order to use ChibiOS/RT on a new architecture:

- 8bits CPU architecture minimum.
- Support for maskable interrupt sources.
- Have a minimum of 2 KB of RAM.
- Have a memory for the program by 16 KB.

If used ChibiOS on Arduino, should be added library ChibiOS first. ChibiOS on robot Arrow-Bot line follower will design using the command thread and memory usage.