Sudan University of Sciences and Technology

College of Engineering

School of Electrical and Nuclear Engineering

Control of Differential Wheel Mobile Robot

التحكم في الروبوت ذو العجلات فرقية الحركة

A Project Submitted In Partial Fulfillment for the Requirements of the Degree of B.Sc. (Honor) In Electrical Engineering

Prepared By:

1. Ahmed Masri Ahmed Gibriel
2. Osama Shibeka Abdin Mohammed
3. Jouher Bashir Abd Elrahman El Shaikh

Supervised By:

Dr. Awadalla Taifour Ali

October 2016

الآية

قال تعالى:

((إِنَّ فِي خَلْقِ السَّمَوَاتِ وَالْأَرْضِ وَاخْتِلَافِ اللَّيْلِ وَالنَّهَارِ لَآيَاتٍ لِأُولِي الْأَلْبَابِ (190) الَّذِينَ يَذْكُرُونَ اللَّهَ قِيَامًا وَقُعُودًا وَعَلَى جُنُوبِهِمْ وَيَتَفَكَّرُونَ فِي خَلْقِ السَّمَوَاتِ وَالْأَرْضِ رَبَّنَا مَا خَلَقْتَ هَذَا بَاطِلًا سُبْحَانَكَ فَقِنَا عَذَابَ النَّارِ(191)))

سورة آل عمران: الأيات 191-192

DEDICATIONS

This study is dedicated with gratitude to our great lovely parents for their endless support, believe, and devotion all along the road.

ACKNOWLEDGEMENT

We wish to thank our supervisor Dr. Awadalla Tayfour Ali for his great support, encouraging and precious time.

ABSTRACT

The differential drive mobile robot is one of the robots which is commonly used in the robotics field due to it is ease of control relatively to the other type of the mobile robots. The system main parts are two wheels with a DC motor for each one, with a third Omni wheel to keep the balance of the robot, infrared sensors, encoders, and the controller. In this system the angular velocity of the system is controlled using a PID controller while the translational velocity is kept constant. A go to goal and avoid obstacles behaviors were used in the controller to navigate the around environment and reach the goal location. The infrared sensors mapped the area around the robot and the encoder used to update the system location, the updated location and the goal location are used to calculate an orientation vector towards the goal location. This vector changes whenever the current location of the robot changes the PID controller minimize the error between the desired angular velocity and the current angular velocity.

In this research, a mathematical model for the system was driven. The system response was captured using a based robot simulator Sim.i.am. A PID algorithm was implemented in the avoid obstacle and go to goal behavior to minimize the error between the desired angular velocity and the current angular velocity and then the PID send the translational velocity that have been calculated in the controller to achieve the desired angular velocity for the robot to each motor of the two wheels.

The MATLAB Sim.i.am simulator was used to simulate the physical world and the response of the robot when it is applying the avoid obstacle and go to goal behavior and when it was tracked by another robot

المستخلص

الروبوت ذو العجلات فرقية الحركة هو أحد أكثر الروبوتات شيوعا واستخداما في مجال الروبوتات حرة الحركة، نسبة لسهولة التحكم في حركته مقارنة مع الروبوتات الأخرى في نفس المجال .يتكون الروبوت من هيكل بلاستيكي مثبت عليه محركي تيار مستمر لنقل الحركة الى عجلتين يتم التحكم فيهما كل على حدى مع وجود عجلة ثالثة حرة الحركة لغرض التوازن، كما يحتوي على خمس محساسات لقياس المسافة وعداد مسافة.في هذا النظام يتم التحكم في السرعة الزاوية عن طريق متحكم تناسبي تفاضلي تكاملي مع الابقاء على السرعة الخطية للروبوت ثابتة. تم استخدام نموذج للاتجاه نحو الهدف وتفادي العوائق للتحكم في الروبوت في اي بيئة فيزيائية والوصول للهدف. تساعد المحساسات وعداد المسافة على رسم البيئة المحيطة بالروبوت وتحديد مكانه و ترسل بيانات كل منهما الى المتحكم الذي يقوم بتحديد متجه في اتجاه احداثيات الهدف هذا المتجه يتغير كلما تغير موضع الروبوت في الاحداثيات ليقوم المتحكم بحساب متجه جديد. تم استخدام المتحكم التناسبي التكاملي التفاضلي لتقليل الخطأ بين السرعة الزاوية المطلوبة واللحظية للروبوت.

تم ايجاد النموذج الرياضي الذي يمثل النظام ومن ثم دراسة المشكلة الاساسية التي تم عرضها في هذا البحث وهي كيفية التحكم في الروبوت لنقله بين نقطتين في بيئة فيزيائية معينة مع دراسة تأثير التغيير الديناميكي للبيئة على اداء النظام. تم استخدام برنامج محاكاة مطور خصيصا لهذا النوع من الروبوتات لرسم استجابة الروبوت الاساسي والروبوت المتعقب للروبوت الاساسي .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

WMR	Wheel Mobile Robot
DC	Direct Current
IR	InfraRed
emf	electromotive force
PID	Proportional Integral Derivative
AO and GTG	Avoid Obstacle and Go To Goal
RAM	Random Access Memory
eMMC	Embedded Multimedia Card
USB	Universal Serial Bus
HDMI	High Definition Multimedia Interface
PRU	Programmable Real-Time Unit

LIST OF SYMBOLS

Kp	Proportional gain
Ki	Integral gain
Kd	Derivative gain
Ti	Integral time
Td	Derivative time
T	Motor torque in N. m
Kt	Torque constant
ke	Back emf constant
θ	Angular displacement
e	Electromotive force
i	Armature Current in ampere
R	Radios of the wheel in meter
l	Distance between two wheels in meter
x	Position in x-axis
y	Position in y-axis
vr	The angular velocity of the right motor in rad/s
vl	The angular velocity of the left motor in rad/s
ø	The orientation of the robot in rad
v	Linear velocity of the robot in m/s
w	Angular velocity of the robot in rad/s
r_B	Magnitude of the behavior vector
$Ø_B$	Orientation of the behavior vector
dj	Distance to the closest obstacle detected by sensor ,m

D	Safety distance at which AO behavior start affecting the system
B_{OA}	Obstacle avoidance behavior
$r_{BOA,j}$	Magnitude of the avoid obstacle behavior
C_{AT}	Constant magnitude
B_{AT}	Approach target behavior
B_S	Sliding mode behavior
dr	The arc moved by the right wheel
dl	The arc moved by the left wheel
dc	The arc moved by the robot
$tick'$	The total number of encoder ticks
$tick$	The previous number of ticks
n	Number of ticks per revolution

CHAPTER ONE

INTRODUCTION

1.1General Concepts

Autonomous mobile robot represent a major example of a reactive, mobile, embedded control system that has received considerable attention during the last decade. The flurry of research activities in this area can be directly traced to the many exciting current and future applications where robotic systems are safer, cheaper, and more effective than their human counterparts. Examples of current applications is the domestic service robots, such as autonomous vacuum cleaners, lawn mowers, and pool cleaners.

Notably absent from this list are the many robots employed in industrial settings. Such industrial robots are not considered here since they typically operate in highly structured environments.

A differential wheeled mobile robot is an example of autonomous mobile robot which's movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion. This robot could be used in several useful applications such as space exploration, firefighting, dismantling of bombs, and every environment that the human cannot deal with it directly [3].

1.2 Problem Statement

The problem is to develop an autonomous mobile robot that is able to drive safety from point A to point B taking in consideration the real world environment.

1.3 Objectives

- Design behaviors to control the robot navigation.
- Find a suitable simulator program to simulate the robot movement on the real environment.
- Design proportional integral derivative (PID) controller to control the robot behavior and improve system response.

1.4 Methodology

- Study of all previous studies.
- Mathematical modeling for robot behaviors.
- Design MATLAB software program to control robot movement.
- Use the sim.i.am simulator for simulation purposes.
- Design of PID controller using manual tuning.

1.5 Thesis Layout

This study consist of five chapters: chapter one represent an introduction shows the problem statement, objectives, and the methodology used in the study. Chapter two discusses the theoretical background and literature review, DC motor, and PID controller. Chapter three presents the system mathematical modeling and the. Chapter four is meant to show the system design and simulation results and hardware structure. Chapter five is for the conclusions and recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 Control Systems Preview

Control engineering is based on the foundations of feedback theory and linear system analysis, and it integrates the concepts of network theory and communication theory. Therefore control engineering is not limited to any engineering discipline but is equally applicable to aeronautical, chemical, mechanical, environmental, civil, and electrical engineering. For example, a control system often includes electrical, mechanical, and chemical components. Furthermore, as the understanding of the dynamics of business, social, and political systems increases, the ability to control these systems will also increase. A control system is an interconnection of components forming a system configuration that will provide a desired system response. The basis for analysis of a system is the foundation provided by linear system theory, which assumes a cause–effect relationship for the components of a system .Therefore a component or process to be controlled can be represented by a block, as shown in Figure 2.1. The input–output relationship represents the cause-and-effect relationship of the process, which in turn represents a processing of the input signal to provide an output signal variable, often with a power amplification. An open-loop control system utilizes a controller or control actuator to obtain the desired response, as shown in Figure 2.2. An open-loop system is a system without feedback. [1]

An open-loop control system utilizes an actuating device to control the process directly without using feedback:

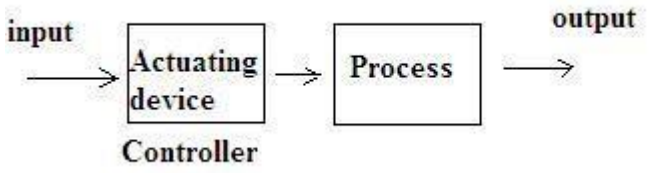

Figure 2.1: Open Loop System

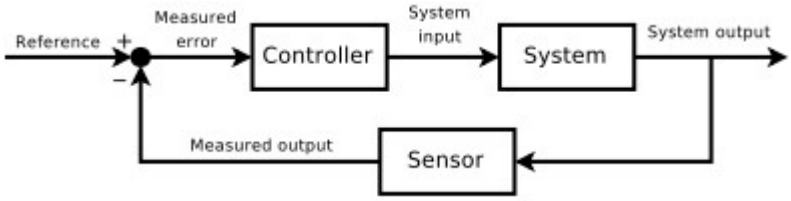

Figure 2.2: Closed loop System

2.2 General Overview on Robotics

Robotics has achieved its greatest success to date in the world of industrial manufacturing. Robot arms, or manipulators, comprise a $ 2 billion industry. Bolted at its shoulder to a specific position in the assembly line, the robot arm can move with great speed and accuracy to perform repetitive tasks such as spot welding and painting .In the electronics industry, manipulators place surface mounted components with superhuman precision, making the portable telephone and laptop computer possible. [2]

2.2.1 Autonomous mobile robots

An example of a reactive, mobile, embedded control system that has received considerable attention during the last decade is the

autonomous mobile robot. The flurry of research activities in this area can be directly traced to the many exciting current and future applications where robotic systems are safer/cheaper/more effective than their human counterparts. Examples of current applications include:

Domestic service robots, such as autonomous vacuum cleaners, lawn mowers, and pool cleaners.

- Planetary exploration robots, such as the NASA Mars rovers Spirit and Opportunity.
- Autonomous robots for military applications, including surveillance and search and-destroy robots.
- And Robots for monitoring, exploring, and securing unsafe environments, such as bomb sniffers, disaster site robots, and mine sweepers.[3]

2.2.2 Differential wheel mobile robots

For control engineers and researchers, there is a wealth of literature dealing with Wheeled Mobile Robots (WMR) control and their applications. However, while the subject of kinematic modeling of WMRs well documented and easily understood by students, the subject of dynamic modeling of WMR has not been addressed adequately in the literature. The dynamics of WMR are highly nonlinear and involve non-holonomic constraints which makes difficult their modeling and analysis especially for new engineering students starting their research in this field. Therefore, a detailed and accurate dynamic model describing the WMR motion need to be developed to offer students a general framework for simulation analysis and model based control system design. [4]

2.3 IR Range Sensors

The orientation of 5 IR sensors (relative to the body of the QuickBot), is 90°, 45°, 0°, -45° and -90° degrees, respectively. IR range sensors are effective in the range 4 cm to 30 cm only. However, the IR sensors return raw values in the range of [0.4, 2.75] V instead of the measured distances. To complicate matters slightly, the Beagle Bone Black digitizes the analog output voltage using a voltage divider and a 12-bit, 1.8V analog-to-digital converter .To use the sensor readings, you will have to convert them to actual distances. For that you need to convert from the ADC output to an analog output voltage, and then from the analog output voltage to a distance in meters .Converting from the analog output voltage to a distance is a little bit more complicated, because a) the relationships between analog output voltage and distance is not linear, and b) the look-up table provides a coarse sample of points. It is possible to use any way you like to convert between sensor readings and distances. For example you can fit the provided points with a high-degree polynomial and use this fit. It is important to note that the IR proximity sensor on the actual Quick Bot will be influenced by ambient lighting and other sources of interference. For example, under different ambient lighting conditions, the same analog output voltage may correspond to different distances of an object from the IR proximity sensor. The effect of ambient lighting (and other sources of noise) are *not* modelled in the simulator, but will be apparent on the actual hardware. [5]

2.4 DC Motor

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables,

can provide translational motion. The electric equivalent circuit of the armature and the free body diagram of the rotor are shown in the following figure.

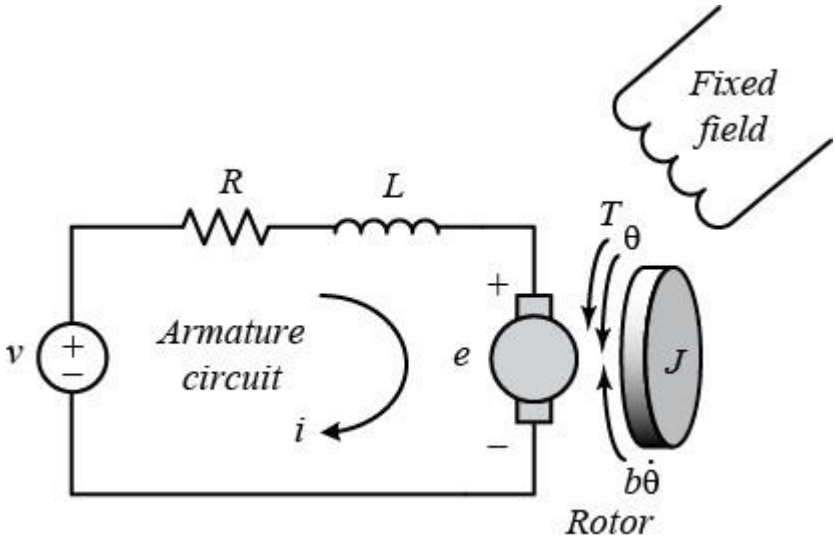

Figure 2.3: DC motor circuit

For this example, we will assume that the input of the system is the voltage source (*V*) applied to the motor's armature, while the output is the rotational speed of the shaft d(*theta*)/dt. The rotor and shaft are assumed to be rigid. We further assume a viscous friction model, that is, the friction torque is proportional to shaft angular velocity.

In general, the torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torque is proportional to only the armature current *i* by a constant factor *Kt* as shown in the equation below. This is referred to as an armature-controlled motor.

$$T = K_t i \quad (2.1)$$

The back emf, *e*, is proportional to the angular velocity of the shaft by a constant factor *Ke*.

$$e = K_e \theta \quad (2.2)$$

In SI units, the motor torque and back emf constants are equal, that is, *Kt = Ke*; therefore, we will use *K* to represent both the motor torque constant and the back emf constant. [6]

2.5 Proportional Integral Derivative Controller

A Proportional Integral Derivative (PID) controller is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable.

The PID controller algorithm involves three separate constant parameters as shown is Figure 2.6 and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. Simply put, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future

errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve, a damper, or the power supplied to a heating element, Table 2.1 shows the effect of increasing a parameter independently.

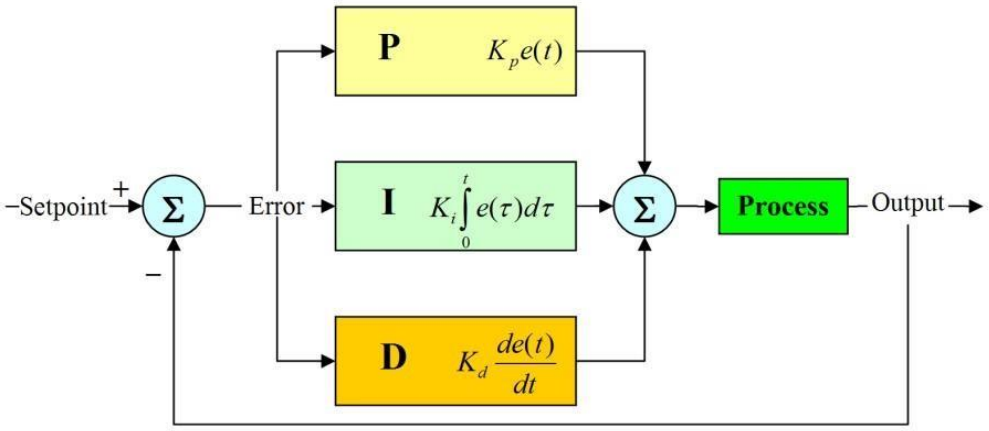

Figure 2.4: proportional integral derivative controller

2.5.1 Effects of proportional, integral and derivative action

Proportional control response for system with transfer function of P(s) $=1/(s+1)^3$ is illustrated in Figure 2.5 with Ti $=\infty$ and Td=0. The figure shows that there is always a steady state error in proportional control. The error will decrease with increasing gain, but the tendency towards oscillation will also increase.

Figure 2.6 illustrates the effects of adding integral to system with transfer function of P(s) $=1/(s+1)^3$. It follows that the strength of integral action increases with decreasing integral time Ti. The

figure shows that the steady state error disappears when integral action is used. The tendency for oscillation also increases with decreasing Ti.

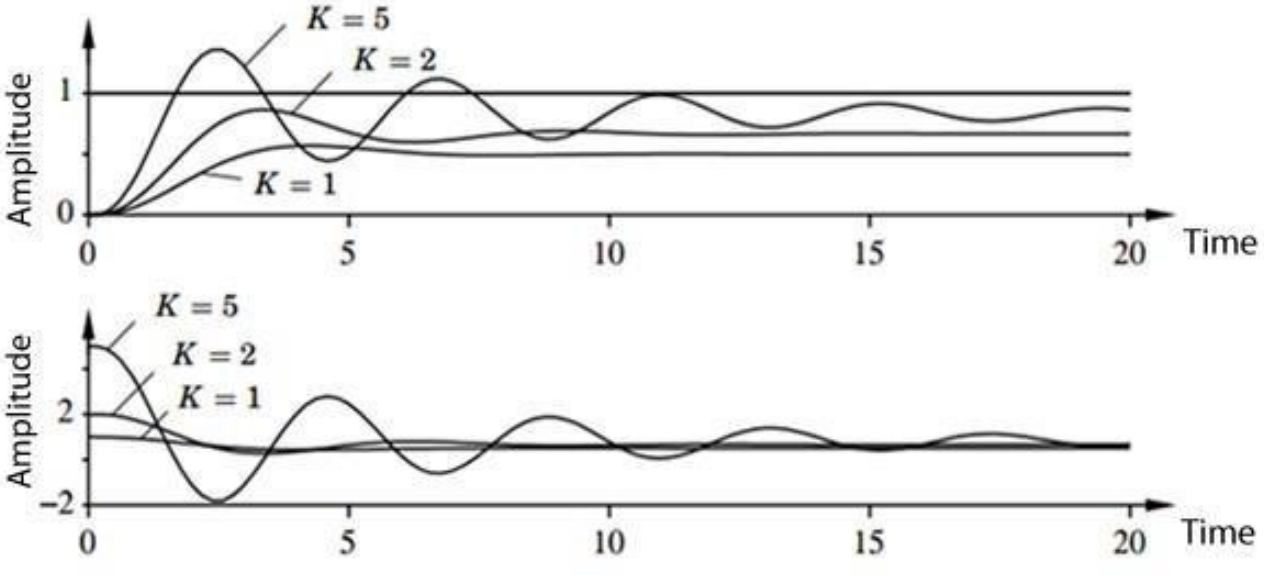

Figure 2.5: Response of the system with proportional control

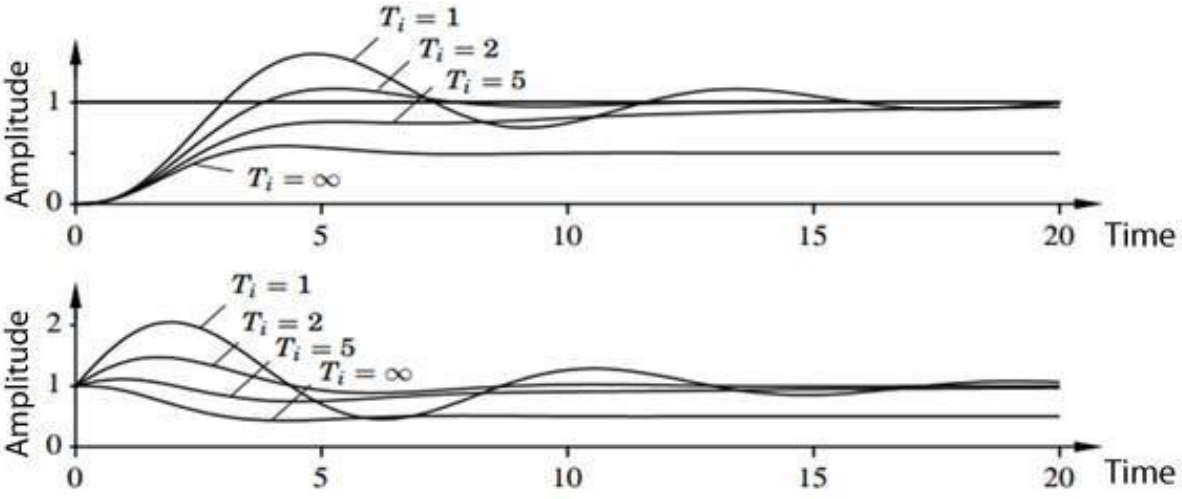

Figure 2.6: Response of the system with proportional and integral control

Table 2.1: Effects of increasing a parameter independently

Parameter	**Rise time**	**Overshoot**	**Settling time**	**Steady-state**
K_P	Decrease	Increase	Small change	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	No change	Decrease	Decrease	Small effect

2.5.2 Proportional controller

The Proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p called the proportional gain constant.

The proportional term is given by:

$$P=K_p(e(t)) \quad (2.3)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.

2.5.3 Proportional integral controller

The main function of the integral action is to make sure that the process output agrees with the set point in steady state. With Proportional control, there is normally a control error in steady state. With integral action, small positive error will always lead to an increasing signal, and a negative error will give a decreasing control signal no matter how small the error is.

$$PI= K_p (e(t)) + K_i \int (t) \quad (2.4)$$

2.5.4 Proportional derivative controller

The purpose of the derivative action is to improve the close-loop stability. Because of the process dynamics, it will take some time before a change in the control variable is noticeable in the progress output. Thus, the control system will be late in correction for an error. The action of a controller with proportional and derivative may be interpreted as if the control is made proportional to the predicted process output, where the prediction is made by extrapolating the error by the tangent to the error curve.

$$PD=K_p(e(t))+k_d\left(\frac{de(t)}{dt}\right) \quad (2.5)$$

2.5.5 Proportional integral derivative controller:

The PID controller has three terms. The proportional term (P) corresponds to proportional control. The integral term (I) give a control action that is proportional to the time integral of the zero. The derivative term (D) is proportional to the time derivative of the control error. This term allows prediction of the future error. There are many variations of the PID algorithm that will substantially improve its performance and operability. Those variations are discussed in the next section [8].

$$\text{PID (t)} = K_p(e(t)) + k_i \int e(t)\, dt + k_d \left(\frac{de(t)}{dt}\right) \quad (2.6)$$

By taking Laplace transform (PID) controller transfer function become:

$$C(S) = k_p + \frac{k_i}{s} + k_d * s \quad (2.7)$$

2.5.6 Tuning of proportional integral derivative controller

The process of selecting the controller parameters to meet given performance specifications is known as controller tuning. There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, and then choosing P, I, and D based on the dynamic model parameters

In particular, when the mathematical model of the plant is unknown and therefore analytical design methods cannot be used, PID controls

prove to be most useful. In the field of process control systems, it is well known that the basic and modified PID control schemes have proved their usefulness in providing satisfactory control, although in many given situations they may not provide optimal control.

If a mathematical model of the plant can be derived, then it is possible to apply various design techniques for determining parameters of the controller that will meet the transient and steady-state specifications of the closed loop system. However, if the plant is so complicated that its mathematical model cannot be easily obtained, then an analytical or computational approach to the design of a PID controller is not possible. Then we must resort to experimental approaches to the tuning of PID controllers [7].

CHAPTER THREE

SYSTEM MODELING

3.1 System Model

For mobile, autonomous robots the ability to function in and interact with a dynamic, changing environment is of key importance. As such, they fall under a class of reactive, mobile systems where environmental changes trigger changes in what objectives the control system must meet. The standard way of structuring the control system in order to deal with this problem is within a multi-modal control framework sometimes referred to in the robotics literature as the behavior-based robotics frame work. The main idea is to identify different controllers, responses to sensory inputs, with desired robot behaviors. This way of structuring the control system into separate behaviors, dedicated to performing certain tasks, has gain significant momentum within the robotics community. This momentum stems from the fact that a modular design both simplifies the design process and also makes it possible to add new behaviors to the system without causing any major increase in complexity.

Once a collection of behaviors has been designed, different options present themselves at the supervisory level. For instance, one can let different behaviors run concurrently in the sense that they all can have an effect on the low-level motor commands according to some coordination rule. This construction with concurrent behaviors makes it relatively straightforward to stress robustness issues explicitly, since,

for example, an “avoidance behavior” can just be given a higher priority or weight than a “reach target behavior.” However, as multiple behaviors are allowed to affect the system simultaneously, a number of theoretical as well as practical issues present themselves [3].

3.2 Dynamical Model

The system model is derived from the unicycle model to differential model since the unicycle model is easy to deal with.

3.2.1 Differential model

$$\dot{x} = \frac{R}{2}(\text{vr+vl})\cos \emptyset \tag{3.1}$$

$$\dot{y} = \frac{R}{2}(\text{vr+vl})\sin \emptyset \tag{3.2}$$

$$\dot{\emptyset} = \frac{R}{l}(\text{vr-vl}) \tag{3.3}$$

Where (x) the position in the x-axis, (y) the position in the y-axis, (R) the radius of the wheel, (l) the distance between the two wheel, (vr) the angular velocity of the right motor, (vl) the angular velocity of the left motor and (ø) the orientation of the robot.

3.2.2 Unicycle model

$$\dot{x} = v\cos(ø) \tag{3.4}$$

$$\dot{y} = v\sin(ø) \tag{3.5}$$

$$\dot{ø} = w \tag{3.6}$$

Where (v) is the linear velocity of the robot and w is the angular velocity of the robot. By substituting the Unicycle model in the Differential Wheel Mobile Robot we get

$$vr = \frac{(2v+wl)}{2R} \tag{3.7}$$

$$vl = \frac{(2v-wl)}{2R} \tag{3.8}$$

Thus the system model in state space for both unicycle and the differential wheel respectively are

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = g_1(q)v + g_2(q)\omega = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega,$$

3.3 Behaviors

If we let the autonomous robot be modeled at the kinematic level as a unicycle as stated in equations (3.4), (3.5) and (3.6). where (x,y) denotes the position of the robot, and φ denotes its orientation, a behavior is characterized by the way sensory data is mapped to the control inputs v and ω, corresponding to translational and angular velocities, respectively. Now, relative to this robot model, a straightforward way of specifying the effect of individual behaviors is to let the behavior define a vector

$$B=r_B \begin{pmatrix} \cos(\text{ø}B) \\ sin(\text{ø}B) \end{pmatrix} \tag{3.9}$$

Where $r_B$is the magnitude of the behavior vector, and $\emptyset B$is its orientation. This vector formalism allows us to map behavior vectors to control values using some appropriate map $(B) = (v,\omega)^T$. For example, one can let

$$\binom{v}{w} = F(B) = \binom{\min\{v0, 1/r_B\}}{c(\emptyset B - \emptyset)} \quad (3.10)$$

Here, the translational velocity achieves its nominal value v0 > 0 when the magnitude of the behavior vector is small, but is reduced as this magnitude grows. Furthermore, the angular velocity is simply given by a proportional error feedback law, with C > 0 being the gain. Note that it is also quite standard to let the gain vary as a function of r_B.

Now, if we are given b1 and b2, i.e., two vectors corresponding to two different behaviors, they can be combined directly using a vector addition operation b1 +b2in order to produce a new behavior, and this semi group property is why the vector notation is particularly appealing. Here the coordination mechanism is thus explicitly given. Moreover, the magnitude of the behavior vector, rb is what determines how much weight that particular behavior is given in the summation. As we will see in the next few paragraphs, avoidance behaviors should increase in magnitude, typically according to an inverse square law, as the robot draws closer to the obstacles.

To make matters more concrete, let us in consider an obstacle-avoidance behavior denoted OA in what follows in some detail. Most mobile robots are equipped with a collection of k range sensors, such as ultrasonic

sonars or infrared sensors, and a standard sonar ring typically consists of 8 or 16 sensors. Each of these sensors measures the distance to the closest obstacle along a particular, fixed relative orientation we let dj denote the distance to the closest obstacle detected by sensor j, and we let $øj$ be the corresponding angle. We can then define the obstacle avoidance behavior,, through the vector summation

$$B_{OA} = B_{OA}1 + B_{OA}, 2 + \ldots + B_{OA}, k \quad (3.11)$$

$$r_{BOA,j} = f(x) = \begin{cases} 0 & Dj > D \\ c_{OA}\left(\frac{(D-dj)}{d_j{}^3}\right) & otherwise \end{cases} \quad (3.12)$$

$$ø_{BOA,} = \pi + øj \quad (3.13)$$

Where $C_{oa} > 0$, and D is the safety distance at which the obstacle-avoidance behavior starts affecting the system .In a similar manner, we can define an

"Approach target behavior" b_{at}, as

$r_{BAT} = c_{at}$

(3.14) $øB_{AT}$=arctan((yg−y)/(xg−x))

(3.15) Where $C_{AT} > 0$ is the constant magnitude, and the goal is located at (xg, yg) [3].

3.4 Regularizations

However, it may not always be desirable to let different behaviors affect the system simultaneously, even though such an approach results both in notational convenience as well as an intuitively appealing mechanism for combining multiple control objectives. Unfortunately, such an approach ruins the modularity that comes with a switched control strategy in the sense that if a new behavior is introduced, its impact on

the system is almost impossible to characterize analytically. This lack of analytical characterization tools is one of the main reasons why emergent behaviors, i.e., unpredictable global behaviors obtained through local rules, have received Considerable attention in the literature. Moreover, if an obstacle-avoidance behavior has been designed so that the robot is guaranteed not to hit static obstacles, by combining this behavior with other behaviors, this guarantee no longer holds.

A remedy to this problem is to let the control system switch between different behaviors. Unfortunately, such an approach may have a negative impact on the performance of the system since it increases the risk of introducing chattering into the system. Chattering is a phenomenon that occurs when two vector fields, corresponding to two different behaviors, both point in toward the switching surface that dictates when the robot should switch between the behaviors. In other words, if we switch from mode 1, where $\dot{x}c = f_1(x)$, to mode 2, where $\dot{x} = f_2(x)$, when x leaves the region $g(x) < 0$, where g is a smooth map from R^n to R, then chattering occurs if

$$\frac{\partial g(x)}{\partial x} f_1(x) > 0 \text{ and} \frac{\partial g(x)}{\partial x} f_2(x) < 0 \tag{3.16}$$

On the boundary $g(x) = 0$.

The standard way out of this problem is to regularize the system so that sliding is allowed to occur. For example, assume that we have access to instantaneous heading control in our control laws. When an obstacle is closer to the robot than D, the obstacle-avoidance behavior is active. Since the repulsive potential field from that behavior will be orthogonal

to the surface on which the behavior becomes active, the sliding solution is given by

$$B_S = \alpha B_{OA} + (1 - \alpha) \tag{3.17}$$

For some $\alpha \in [0,1]$ such that $B_S \perp B_{OA}$.

Some results from applying this regularization approach to the chattering problem are shown in Fig. 2, where Figure 3.1 shows a situation when vector summation is used. Figure 3.2 corresponds to switches between the behaviors, and it is clear that a chattering-like behavior is produced. In Figure 3.3 the regularized solution is shown. Even though we only have one behavior active at a time, the performance is clearly satisfactory in that case.

By incorporating this type of information about the different behaviors, it is possible to generate the sliding modes automatically. It furthermore suggests that this method would scale when more than two behaviors affect the motion of the robot, as long as an automatic procedure for designing the sliding solutions can be identified. Hence we assume throughout the remainder of this chapter that only one behavior affects the robot at each time instant, and that, when appropriate, a sliding mode may be induced from the system dynamics.

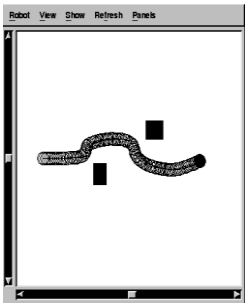

Figure 3.1: Combined behaviors using vector summation

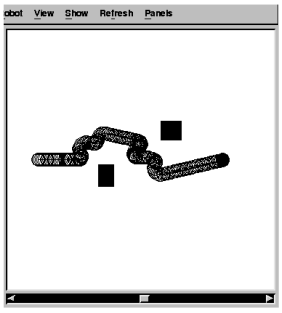

Figure 3.2: Switches between the different behaviors

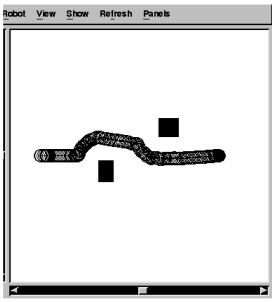

Figure 3.3: A regularize solution

3.5 Wheel Encoders

$$dc = \frac{dr+dl}{2} \tag{3.18}$$

$$x = \acute{x} + dc\cos(\phi) \tag{3.19}$$

$$y = \acute{y} + dc\sin(\phi) \tag{3.20}$$

$$\phi' = \phi + \frac{(dr-dl)}{l} \tag{3.21}$$

$$\Delta tick = tick' - tick \tag{3.22}$$

$$d = 2\pi R\frac{\Delta tick}{n} \tag{3.23}$$

Where (dc) is the arc moved by the robot, (dr) the arc moved by the right wheel, (dl) the arc moved by the left wheel, ($tick'$) the total number of encoder ticks and ($tick$) the previous number of ticks and n is the number of ticks per revolutions.

CHAPTER FOUR

SYSTEM DESIGN AND HARDWARE PARTS

4.1 The Sim.I.am Simulator

This simulator program was developed by **GORGIA INSTITUTE OF TECHNOLOGY** to perform the following tasks

- Understanding the robot (to process the information from the robot.
- Transformation from unicycle to differential drive dynamics.
- Odometry, such that as the robot moves around, its pose is estimated based on how far each of the wheels have turned.
- Implementing the PID controller by implementing the different parts of a PID regulator that steers the robot successfully to some goal location.
  This is known as the go-to-goal behavior.
- Ensuring the right angular velocity by tackling the first of two limitations of the motors on the "QuickBot". The first limitation is that the robot's motors have a maximum angular velocity, and the second limitation is that the motors stall at low speeds.
- To generate Avoid Obstacles by implementing the different parts of a controller that steers the robot successfully away from obstacles to avoid a collision. This is known as the avoid-obstacles behavior.

- Mixing go-to-goal and avoid-obstacle controllers and testing two arbitration mechanisms: blending and hard switches. Arbitration between the two controllers will allow the robot to drive to a goal, while not colliding with any obstacles on the way.
- To realize wall following behavior (whether the obstacle on the left or right is followed).

This simulation software with generated controllers can be efficiently used for controlling the real “QuickBot” to make it full autonomous. This dependence is not known a priori, as it depends on the motors, the wheels and the surface. To be able to control your robot reliably, you have to measure this dependence and put into code. On this way, it is possible by combining the go-to-goal, avoid obstacles, and follow-wall controllers into a full navigation system for the robot. The robot will be able to navigate around a cluttered, complex environment without colliding with any obstacles and reaching the goal location successfully. [5]

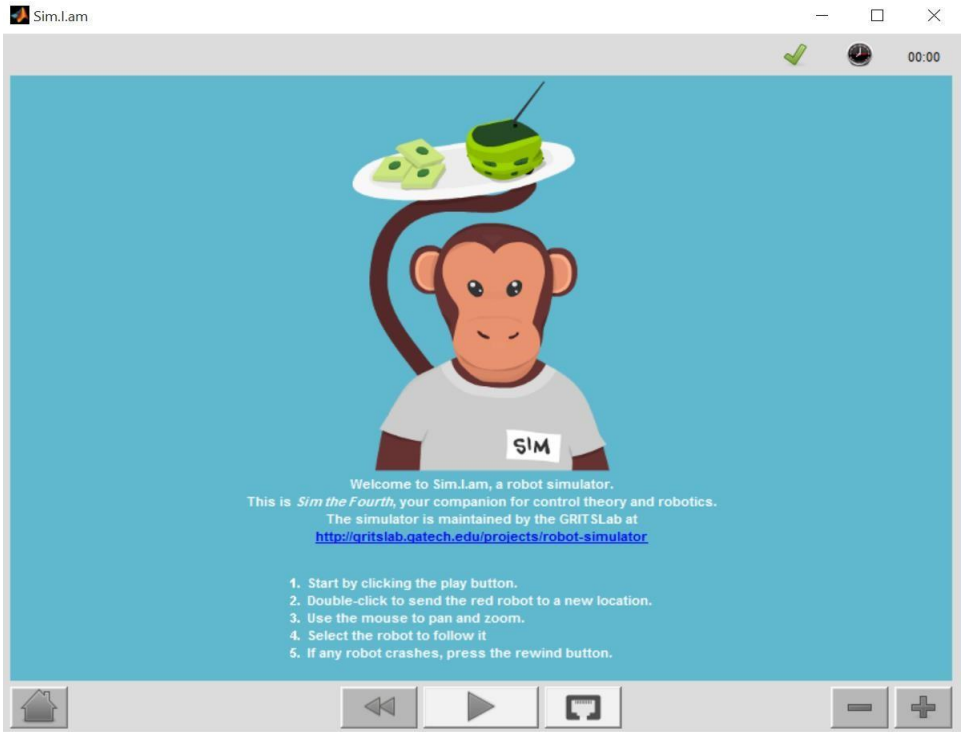

Figure 4.1: Sim.i.am graphical user interface

4.2 System Design

The system model was represented in the state space forum hence, the design will be carry out in the state space forum.

Parameters design and simulation results

The method used for designing the PID controller for the system is the manual tuning at first the proportional part value increased till a first oscillation was observed in the response and then the derivative part is increased from zero till the transient response characteristic were optimized and finally the integral part implemented to reduce the steady state error.

The adjustment of the different parts of the PID controller will be adjusted in the avoid obstacle and go to goal blended in one controller stored at the file AOandGTG.m.

- First attempt

The proportional part is set at 1 the derivative and integral both set at zero. In the following figures the green line indicate the desired response and the blue line indicates the response of the robot based on the simulator.

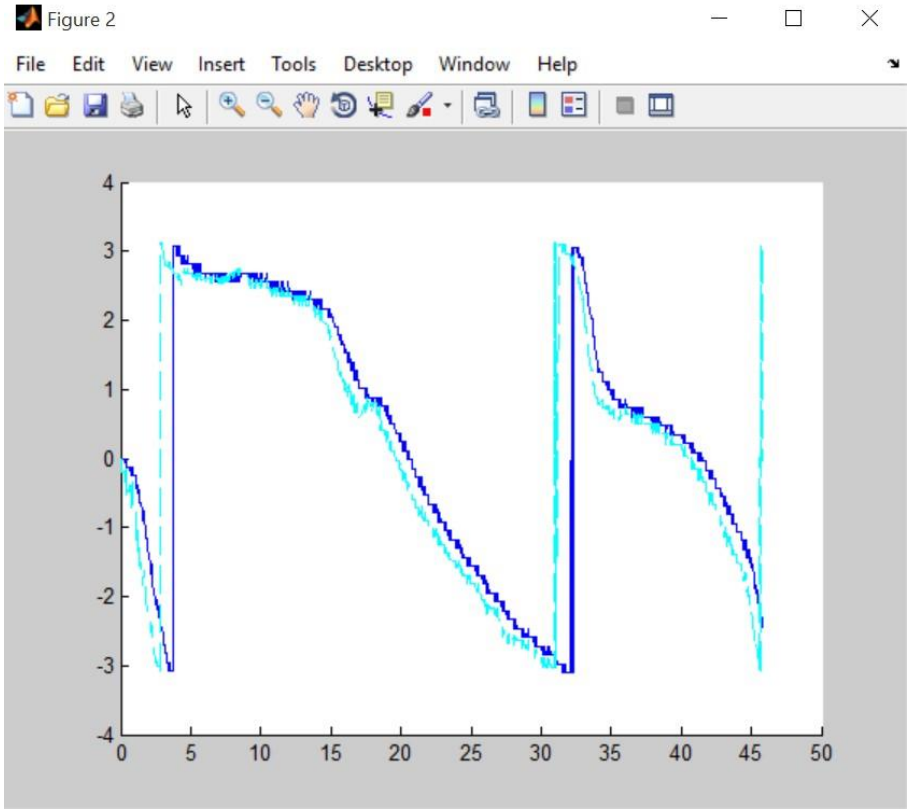

Figure 4.2: Robot response with Kp =1, Ki =0, Kd =0

Where green line indicate to current angular velocity (w) and blue line indicate to estimated angular velocity (w)

The system response is very slow and the desired angular velocity was not achieved accurately.

- Second attempt

By increasing the proportional kp=2 and set both integral derivative controllers at zero

Kp=2, ki=0, kd=0

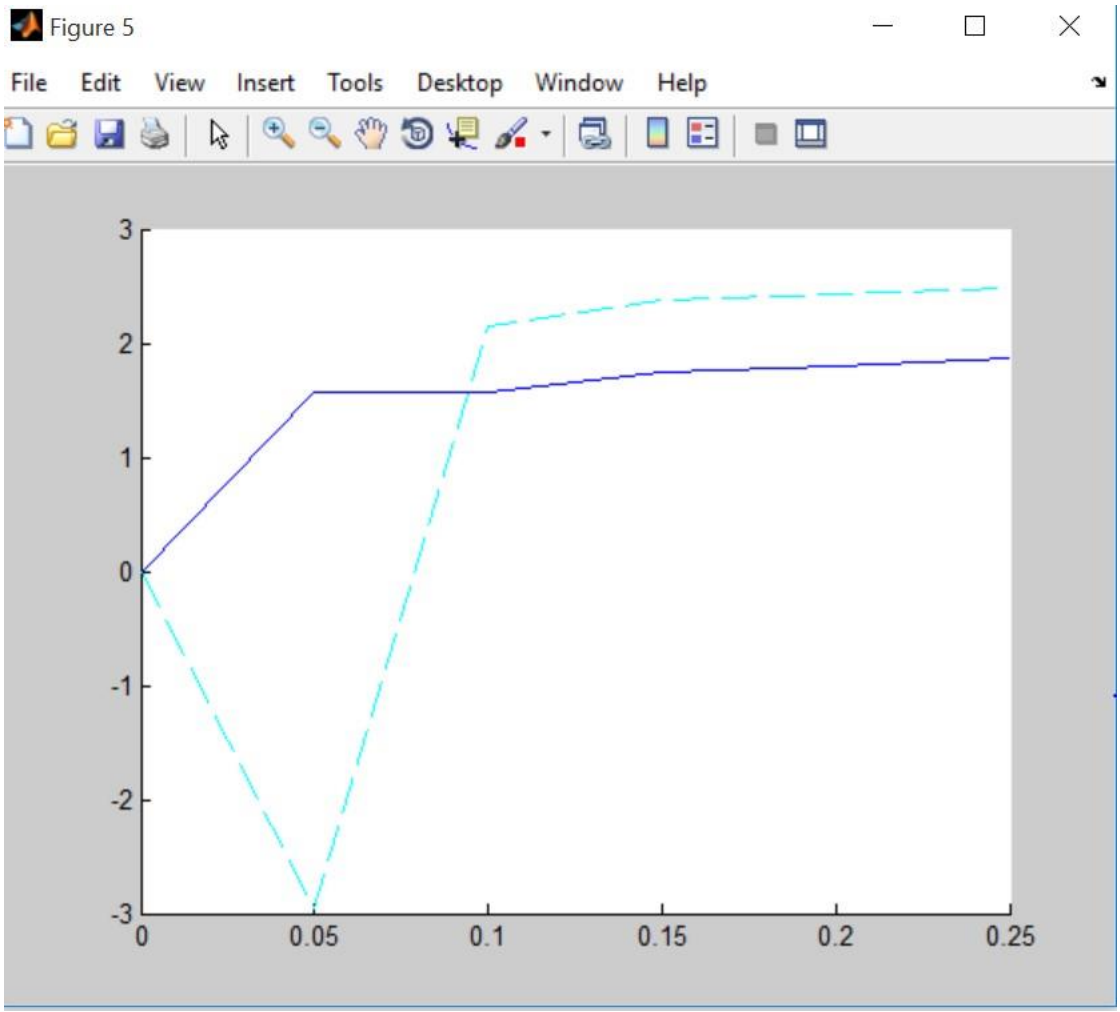

Figure 4.3: Robot response with Kp =2, Ki =0, Kd =0

- Third attempt

By more increasing in the proportional constant and small tunning in both integral and derivative constants we have found that the most optimal values for these parameters is as follows

Kp = 5, Ki = 0.01, Kd = 0.01

The following curves show the best results for the controller which were tested in the simulator.

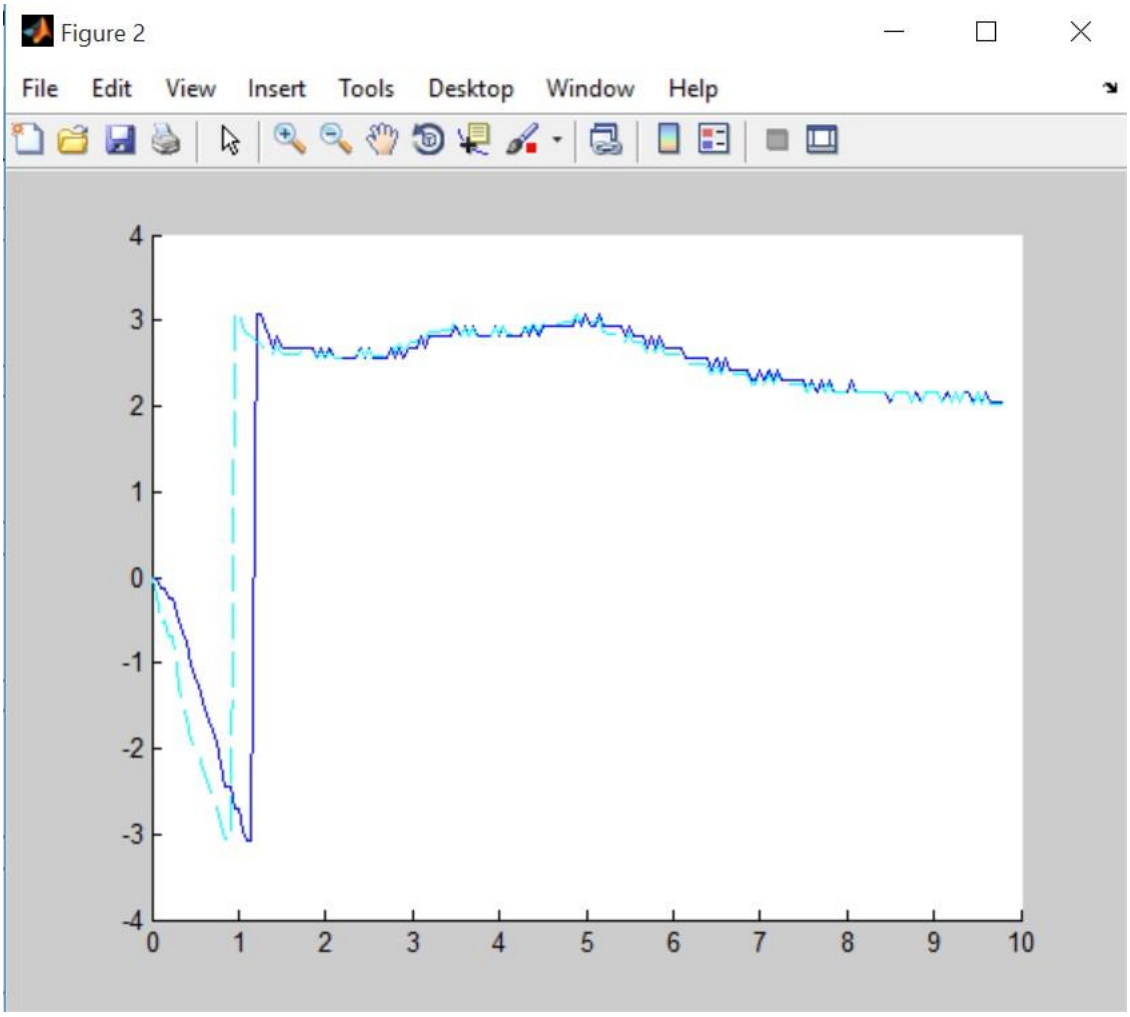

Figure 4.4: Robot response with Kp =5, Ki =0.01, Kd=0.01

4.3 System Hardware Parts

The system hardware structure consist of the following parts:

4.3.1 Beagle Bone Black

The Beaglebone Black is a low power open source hardware single board computer produced by (Texas Instruments) in association with (Digi-Key) and (Newark element14).The Beaglebone Black was also designed with open source software with the following features:

- Processor
- AM335x 1GHz ARM® Cortex-A8.
- 512MB DDR3 RAM.
- 4GB 8-bit eMMC on-board flash storage.
- 3D graphics accelerator.
- NEON floating-point accelerator.
- 2x PRU 32-bit microcontrollers.
- Connectivity
- USB client for power & communications.
- USB host.
- Ethernet.
- HDMI.
- 2x 46 pin headers.
- Software compatibility
- Debian.
- Android.
- Ubuntu.
- Cloud9 IDE on Node.js w/ BoneScript library.
- Power supply
- 5-volt, 2-ampere power supply source.

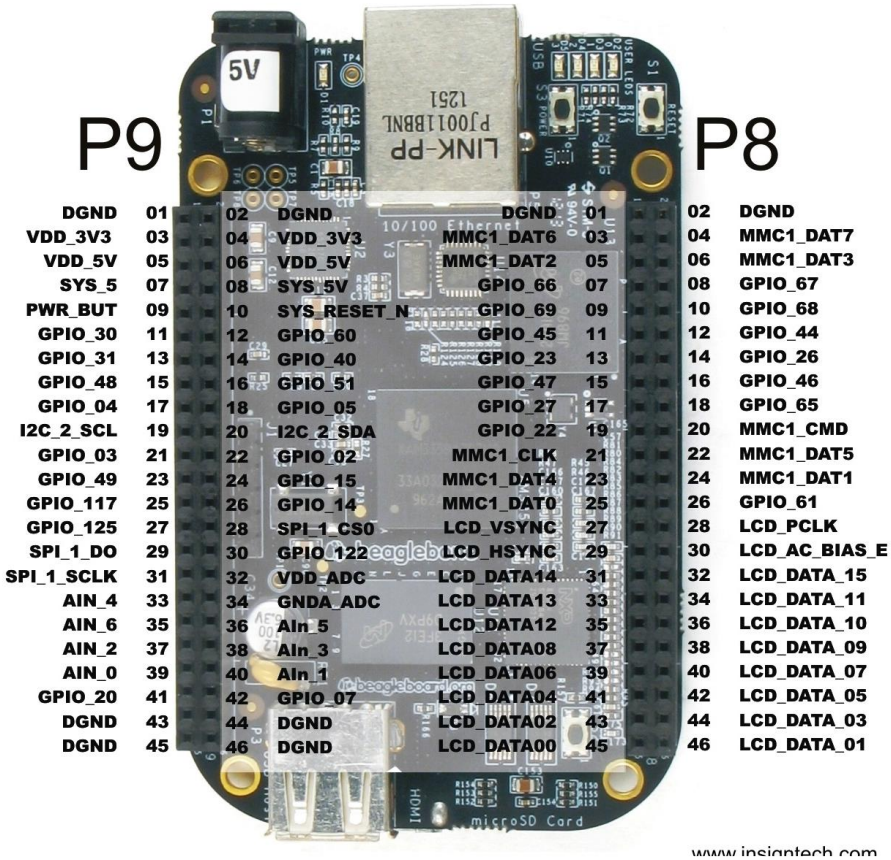

Figure 4.5: Beagle bone black

4.3.2 IR sensor

An infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view and it has measuring distance between zero and 0.2 meter.

Figure 4.6: IR infrared sensor

4.3.3 Wheel encoder

The encoder is a sensor attached to a rotating object (such as a wheel or motor) to measure rotation. These pulses can be used as part of a feedback control system to determine translation distance, rotational velocity, and/or angle of a moving robot or robot part.

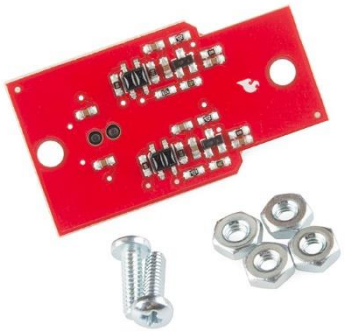

Figure 4.7: Wheel encoder

4.3.4 DC motor

- **Features:**

- Suggested Voltage: 4.5VDC
- No Load Speed: 140RPM
- No Load Current: 190mA
- Max. Load Current: 250mA
- Torque: 800 gf-cm

Figure 4.8: DC motor

4.3.5 Motor driver

The (SN754410) is a quadruple high-current half-H driver designed to provide bidirectional drive currents up to 1 A at voltages from 4.5 V to 36 V.

The device is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

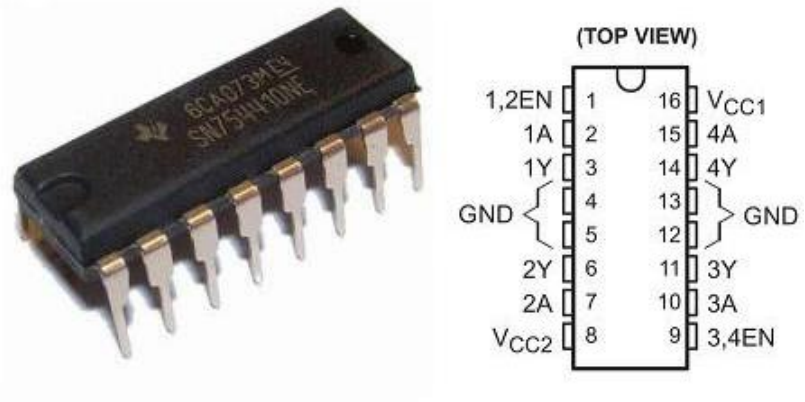

Figure 4.9: Motor driver

4.3.6 Wi-Fi antenna

The Beagle Bone black uses a 4-inch-(2.7 x 1.3 x 1.3 cm) dimension compatible antenna.

Figure 4.10: 4-Inch antenna

4.3.7 Mechanical parts

- Micro Magician Robot Chassis Kit with (4) screws, brackets, and stands.
- Two (32.5) mm Radius wheels with (99.5) mm distance between the two wheels.
- One holder wheel for balancing purposes.

Figure 4.11: Chassis and wheels

4.3.8 Basic requirements

- One mini-size breadboard.
- One half-size breadboard.
- Two battery holders.
- Heat sink.
- On-off switch.
- Jumper wires.
- Resistors.
- Capacitor.
- Diodes.
- (8) Rechargeable batteries.
- Ld1085v50 regulator.

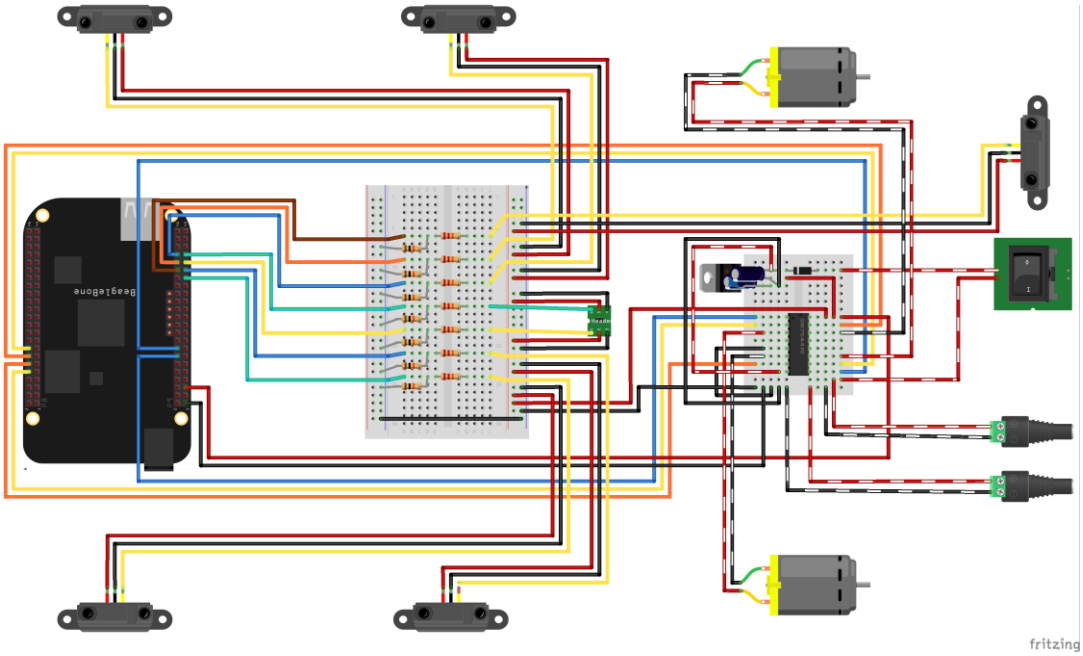

Figure 4.12: Electrical and electronic connections

Figure 4.13: QuickBot

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A mathematical model of the differential wheel mobile robot system was developed by using physical laws. A simplified mathematical model was derived by system parameters. The controller parameters values (Kp, Ki and Kd) were obtained by using manual tuning method from simulation model so as to perform best system response. From experimental results, it is found that the best controller parameters which gave the best response of the system are Kp= 5, Ki=0.01 and Kd= 0.01. The accuracy of the system is tested adjusting the angular velocity of the differential wheel mobile robot.

5.2 Recommendations

1- It is recommended that controlling the system using the machine learning method and the artificial intelligence method.
2- Study the response of the other common used digital controller e.g Beaglebone Black, Raspberry pi and Arduino and compare the response of the system with each one of them.

3-Prove the Beaglebone Black code to get a real time results of the system and compare it to the simulation results.

REFERENCES

[1] Rohan Munasinghe, “Classical control systems”, ISBN 978-81-8487-1944.

[2] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza,

“Introduction to Autonomous Mobile Robots”,ISBN 978-0-262-01535-6

[3] Magnus Egerstedt, “Control of Autonomous Mobile Robots”, School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA30332, U.S.A”

[4] Dhaouadi R, Hatab AA (2013),“Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework”. Adv Robot Autom 2: 107. doi: 10.4172/2168-9695.1000109.

[5] ZelijkoDespotovic, AleksanderCosic, BrankoMiloradovic, “QuickBot as Educational and Research Platform for Multi Mobile Robotic Systems”, 2015.

[6] Bill Messner and Dawn Tilbury,“Control Tutorials for matlab and Simulink”, University of Michigan By,9781491905395.

[7]K.Ogata, “Modern control engineering”. Englewood Cliffs, N.J.:

PrenticeHall, 1970.

APPENDIX A

MATLAB m.file of the AOandGTG controller

```
classdefAOandGTG<simiam.controller.Controller

% Copyright (C) 2013, Georgia Tech Research
Corporation
% see the LICENSE file included with this software

prop
erti
es

% memory banks
E_k
e_k_1

% gains
Kp
 Ki
Kd

% plot support
p
```

```
% sensor
geometry
calibrated
sensor_placem
ent end

properties (Constant)
inputs = struct('x_g', 0, 'y_g', 0,
'v', 0); outputs = struct('v', 0, 'w',
0) end

m
e
t
h
o
d
s

functionobj = AOandGTG() obj =
obj@simiam.controller.Controller('ao_and_gtg');
obj.calibrated = false;

obj.K
p =
5;
obj.K
```

```
i =
0.01;
obj.K
d =
0.01;
 obj.E_k = 0;
obj.e_k_1 = 0;

%              obj.p =
simiam.util.Plotter(); end

function outputs = execute(obj, robot,
state_estimate, inputs, dt)

% Compute the placement of the
sensors if(~obj.calibrated)
obj.set_sensor_geometry(robot);
end

% Unpack state estimate
             [x, y, theta] =
state_estimate.unpack();

% Poll the current IR sensor values 1-9
ir_distances = robot.get_ir_distances();
nSensors = numel(ir_distances);
```

```
% Interpret the IR sensor measurements
geometrically ir_distances_wf =
obj.apply_sensor_geometry(ir_distances,
state_estimate);

% 1. Compute the heading vector for obstacle
avoidance

%              sensor_gains = [1 1 0.5
1 1]; if (nSensors == 5) % QuickBot
sensor_gains = [1 1 0.5 1 1]; elseif
(nSensors == 9)
% Khepera3
sensor_gains =
ones(1,nSensors); end

u_i = (ir_distances_wf-
repmat([x;y],1,nSensors))*diag(sensor_gains
); u_ao = sum(u_i,2);

% 2. Compute the heading vector for go-
to-goal x_g = inputs.x_g; y_g =
inputs.y_g; u_gtg = [x_g-x; y_g-y];

% 3. Blend the two
vectors alpha = 0.25;
u_ao_gtg = alpha*u_gtg+(1-alpha)*u_ao;
```

```
% 4. Compute the heading and error for the PID
controller
theta_ao_gtg = atan2(u_ao_gtg(2),u_ao_gtg(1));

e_k = theta_ao_gtg-theta;
e_k =
atan2(sin(e_k),cos(e_k));
 e_P = e_k; e_I =
obj.E_k + e_k*dt;
e_D = (e_k-
obj.e_k_1)/dt;

% PID control on w
v = inputs.v;
            w = obj.Kp*e_P + obj.Ki*e_I +
obj.Kd*e_D;

% Save errors for next
time step obj.E_k = e_I;
obj.e_k_1 = e_k;

% plot
            obj.p.plot_2d_ref(dt, theta,
theta_ao_gtg, 'c');
```

```
%              fprintf('(v,w) = (%0.4g,%0.4g)\n',
v,w);
              v =
0.25/(log(abs(w)+2)+1);

outputs.
v = v;
outputs.
w = w;
end

% Helper functions

functionir_distances_wf =
apply_sensor_geometry(obj, ir_distances,
state_estimate)

% 1. Apply the transformation to robot frame.
nSensors = numel(ir_distances);

ir_distances_rf =
zeros(3,nSensors);
fori=1:nSensors x_s =
obj.sensor_placement(1,i); y_s
= obj.sensor_placement(2,i);
theta_s =
obj.sensor_placement(3,i);
```

```
                R =
obj.get_transformation_matrix(x_s,y_s,theta_s)
; ir_distances_rf(:,i) = R*[ir_distances(i); 0;
1]; end

% 2. Apply the transformation to world frame.

            [x,y,theta] = state_estimate.unpack();

            R =
obj.get_transformation_matrix(x,y,theta);
ir_distances_wf = R*ir_distances_rf;

ir_distances_wf =
ir_distances_wf(1:2,:); end

functionset_sensor_geometry(obj, robot)
nSensors = numel(robot.ir_array);

obj.sensor_placement = zeros(3,nSensors);
fori=1:nSensors
                [x, y, theta] =
robot.ir_array(i).location.unpack();
obj.sensor_placement(:,i) = [x; y;
theta]; end
obj.calibrated =
true; end
```

```
function R = get_transformation_matrix(obj, x, y,
theta)
            R = [cos(theta) -sin(theta) x;
sin(theta) cos(theta) y; 0 0 1]; end
 function
reset(obj)
% Reset accumulated and previous
error obj.E_k = 0;
obj.e_k_1 = 0; end  end  end
```

APPENDIX B

MATLAB m.file of the supervisor

```
classdef Supervisor < handle
%% SUPERVISOR switches between controllers and
handles their inputs/outputs.
%
% Properties:
%   current_controller      - Currently selected
controller
%   controllers             - List of available
controllers
%   goal_points             - Set of goal
points %   goal_index               - Pointer to
current goal point
%   v                       - Robot velocity
%
% Methods:
%   execute - Selects and executes the current
controller.
 properties
%%
PROPERTIES
current_con
troller%
Currently
```

```
selected
controller
controllers
% List of
available
controllers
robot% The
robot
state_estimate% Current estimate of the robot's
state end
 methods
%%
METHODS
 functionobj =
Supervisor()
        %% SUPERVISOR Constructor

% initialize the controllers
obj.controllers{1} =
simiam.controller.Controller('default');

% set the initial controller
obj.current_controller = obj.controllers{1};

obj.robot = []; obj.state_estimate =
simiam.ui.Pose2D(1,2,1.54); end
```

```
functionattach_robot(obj, robot, pose)
obj.robot = robot;
            [x, y, theta] =
pose.unpack();
obj.state_estimate.set_pose([x, y,
theta]); end
 function
execute(obj, dt)
        %% EXECUTE Selects and executes the
current controller.
%   execute(obj, dt) will select a controller from
the list of
%   available controllers and execute it.
%
%   See also
controller/execute end
end end
```